NoSQL adatbázisok

BEVEZETÉS

„Klasszikus” adatbázis jellemzői

- Centralizált
  - Tárolás és menedzsment szintjén is
- Relációs szerkezetű
  - Az adatelemek viszonyai határozzák meg a felépítését (→ normalizálás)
- Kevés adat (aktuális viszonyokhoz képest)
  - Adatok jelentős része az üzemeltetőtől származnak

Új alkalmazások

- Google, Facebook, Amazon, stb
  - Egész világot lefedik
  - Sok felhasználó → sok adat
  - felhasználói szokások nyomon követése értékké válik
- Szenzorhálózatok, Internet of Things (IoT)
  - Tipikusan kezelhető mennyiségű forrás
  - DE nagy frekvencia!

Vertikális skálázás (hardver növelése) sem elegendő!

Új igények

- Elosztott működés
  Több szerver működik együtt
  - Tárolás
  - Kiszolgálás
  - Hibatűrés
- Új szerkezet a tárolandó adatoknak
  - Hogy lesz minden BCNF-ben?! - Lehetetlenség ...
- Óriási mennyiségű adat
  - melynek túlnyomó részét a felhasználói aktivitás generálja

Big data – mennyire „big”

- A Facebook-on 60 másodpercenként 510000 poszt, 293000 állapot frissítés, 136000 fotó feltöltés! (~ 2018)

- Havi aktív Twitter felhasználók: 335 millió (2018 második negyedév)

- Átlagosan 40000 keresést dolgoz fel másodpercenként a Google, ami több mint 3,5 milliárd napi keresés, ami 1,2 billió keresés évente világszerte! 1 200 000 000 000

Mit kezdjünk vele?

- Az információnak üzleti értéke van!
  - Hogyan lehetséges kinyerni ekkora adatmennyiségből (több TB) kinyerni a hasznosítható információkat?
- Elvárás a közel real-time feldolgozás
  - „Aki megvásárolta ezt a terméket, ezeket is megnézte ...”
  - Zenei, filmajánló rendszerek
  - Az adatok tárolási sorrendje releváns!

NoSQL adatbázisok

- NoSQL: not SQL / not only SQL
- Gyűjtőnév
  - Több konkrét, hasonló alapelven (nem relációs felépítés) alapuló adatbázis
  - Konkrét modellek nagyon különbözőek
- Figyelembe veszi az új kihívásokat
  - Skálázhatóság
  - Változó adatstruktúrák
  - Objektumközpontú szemlélet
  - Óriási adatmennyiség

NoSQL adatbázisok

Közös tulajdonságok

- Nem relációs szerkezet
  az adatok határozzák meg a felépítését
- Felhasználás-központú adatbázis tervezés
- Jó horizontális skálázhatóság
  - Skálázható:
    erőforrásokat növelve a rendszer teljesítménye a hozzáadott erőforrásokkal arányosan javul
  - Horizontális
    újabb gépek (szerverek) bevonhatósága
- Elosztott működésre tervezve

Az elosztott tárolás típusai

- Két megközelítés – kétféle cél
  - Replikáció
  - Sharding

- Van olyan adatbázis, mely mindkettőt tudja

Replikáció

- Ugyanazok az adatok több példányban tárolódnak
  a szerverek adattartalma azonos
- Cél
  - BIZTONSÁG
    ha egy szerver kiesik, ne legyen adatvesztés
  - Másodlagos cél
    terhelés megosztás bizonyos műveletekre (tipikusan olvasásnál)

Sharding

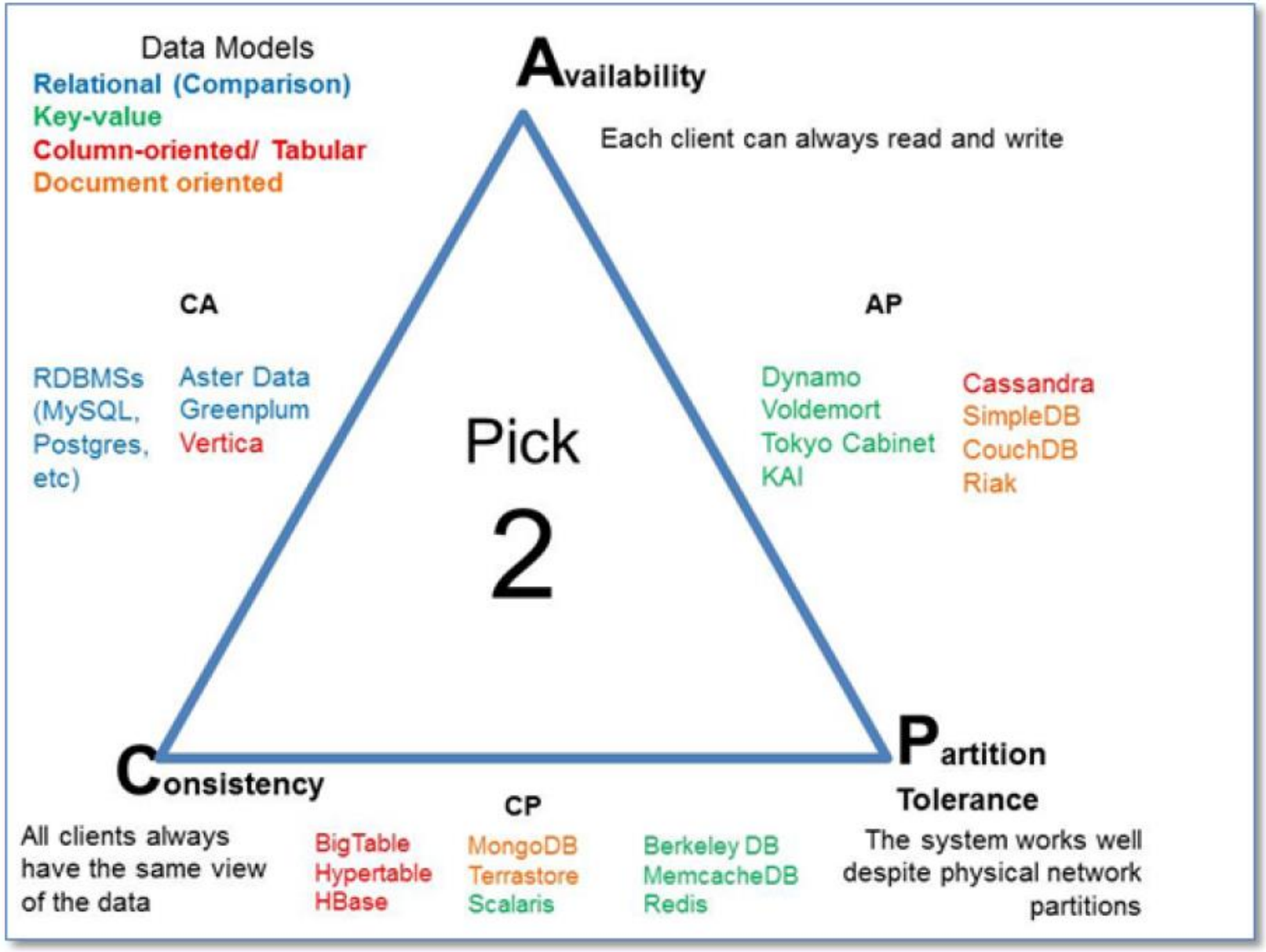
- Az adatok egy példányban tárolódnak, de több szerveren
  a szerverek adattartalma különböző (hacsak nem kombinálják replikációval)

- Cél
  terhelés megosztása
  minden szerver kiszolgál olvasás-írás műveleteket is

CAP tétel

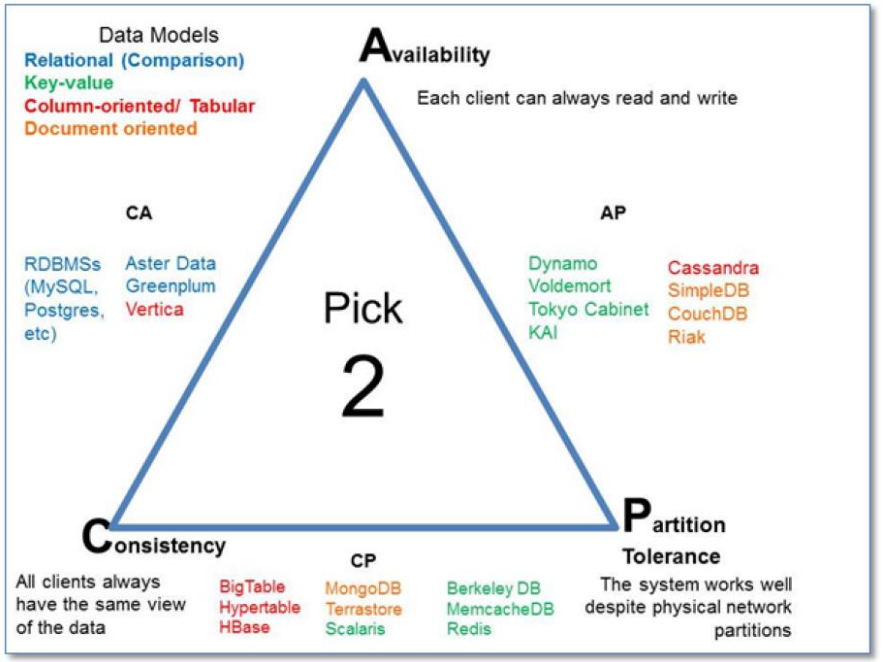
Brewer tétele (Eric Brewer, 1999)

Egy elosztott rendszer a három alapvető képesség közül (konzisztencia, rendelkezésre állás, particionálás tűrés) legfeljebb kettőt tud megvalósítani.



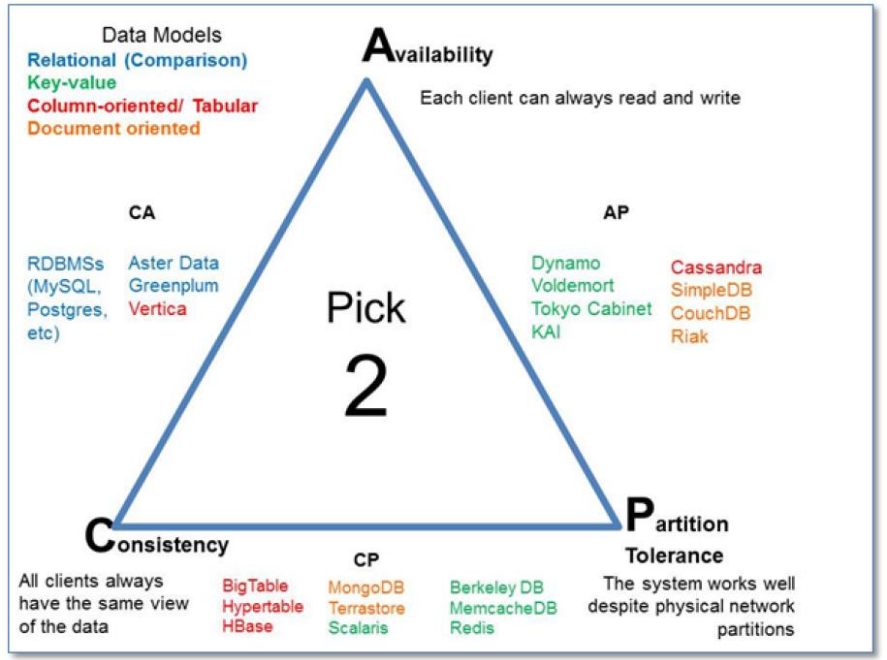
CAP tétel - konzisztencia

- Nem ugyanaz, mint a relációs adatbázisok konzisztenciája!!
- Az adatok több példányban tárolódnak (replikáció), különböző csomópontokon
- Konzisztencia
  adott időpillanatban bármely csomópontról való olvasás ugyanazt a (legfrissebb) értéket adja, vagy hibát.
  - Szó szerint értelmezve nem valósítható meg, hiszen a szinkronizálás is időbe telik ...



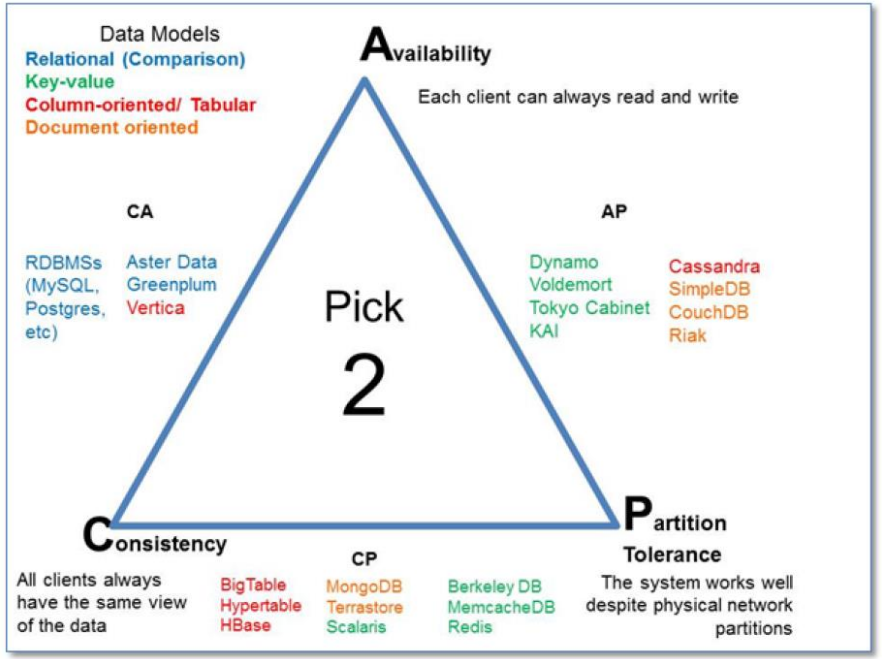
CAP tétel – rendelkezésre állás

- Az adatok (minimális késleltetéssel) folyamatosan elérhetőek, azaz minden kérésre érkezik válasz arról, hogy a kérés végrehajtása sikeres vagy sikertelen volt-e
  - Konzisztenciát ez nem garantál!
  - Nincs olyan kritikus elem a rendszerben, amelynek kiesése működésképtelenséget eredményez
  - Több csomópont – redundancia
    persze ha az összes csomópont egyszerre esik ki, az ellen nem véd...
  - Master szerver (ír- olvasási kérés) – slave server (írást nem szolgál ki).
    Ha kiesik a master, slavek megszavazzák ki lesz a master



CAP tétel – partíció tolerancia

- A rendszer akkor is működőképes marad (a kliensek kéréseire választ ad), ha a csomópontok között tetszőleges számú üzenet elvész
  - Pl. a hálózat két csomópont között megszakad, vagy valamelyik csomópontban valamilyen hardver vagy szoftver hiba történik
  - A hiba megszűnése után tudni kell szinkronizálni!
  - Minden elosztott rendszernél előfordulhat hálózatkiesés, tehát ez szükséges tulajdonság!



Lehetőségek

- Elhagyni a particionálás tűrést
  rakjunk mindent egy gépre, vagy egyetlen, atomi módon elbukó egységre (pl. egy rack-be), skálázhatósági problémát jelent
- Elhagyni a rendelkezésre állást
  egy esemeny bekövetkezésekor az érintett szolgáltatások megvárják, amíg az adatok konzisztenciája helyreáll, ezalatt elérhetetlenné válnak (particionálás tűrés másik oldala)
- Elhagyni a konzisztenciat:
  ez a valosagban is sokszor így van
  - egy árucikk-adatbázis konzisztens-e, ha a raktáros épp most tör össze valamit
  - ha egyszerre két rendelés érkezik egyetlen árucikkre, akkor csak az egyiket tudjuk azonnal kiszolgálni, de a másik is beszerzés alá kerül, ha erről értesítjük az ügyfelet (néha nem probléma, de helyfoglalásnál már igen)
  - a rendelkezésre állás sokszor fontosabb a konzisztenciánál

NoSQL adatbázis típusok (népszerűségi sorrendben)

- Kulcs-érték tárolók (key-value stores)
  - Redis, Riak, Oracle NoSQL, ...
- Oszloptárolók, oszlopcsaládok (wide-column stores, column families)
  - Hbase, Cassandra, Google Cloud Bigtable, ...
- Gráf adatbázisok (graph databases)
  - Neo4J, Virtuoso, InfiniteGraph, ...
- Dokumentumtárolók (document stores)
  - MongoDB, CouchDB, Couchbase, ...

Kulcs-érték tárolók

- *(K,V)* párok együttese, ahol *K* kulcs, *V* pedig egy érték
- Legegyszerűbb típus
- Kulcs-érték párokat tárol
  - Lényegében egy asszociatív tömb
- Adattípusok rugalmas kezelése
- Jellemzően kulcs szerinti lekérdezések
- A lastVisit kulcshoz tartozó érték az utolsó látogatás időbélyege, a user kulcshoz tartozó érték pedig egy táblázat, mely az ügyfél-azonosítót, az ügyfél nevét és országkódját és az időzónát tárolja.

```
{
  "lastVisit":1324669989288,
  "user":{
    "customerId":"91cfdf5bcb7c",
    "name":"Márton Ispány",
    "countryCode":"HUN",
    "tzOffset":"CET"
  }
}
```

Kulcs-érték tárolók

Mikor használjuk

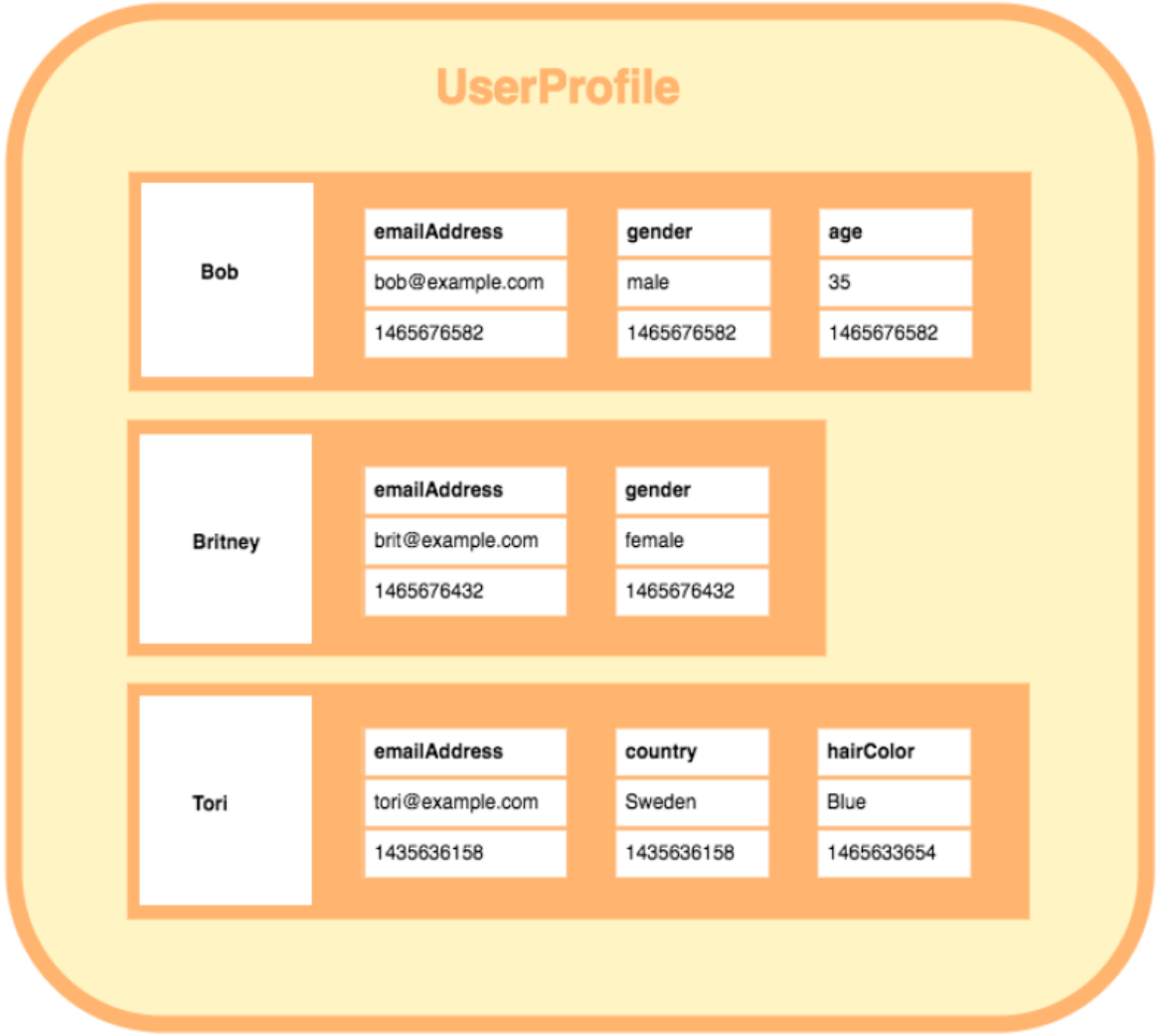
- Munkamenet adatok (lehetséges kulcs: sessionid)
- Bevásárlókosarak adatai (lehetséges kulcs: userid)
- Felhasználói profilok, beállítások

Mikor ne használjuk

- Ha kapcsolatok vannak az adatok között
- Többműveletes tranzakciók esetén: pl. Ha több kulcs mentésekor valamelyik sikertelensége esetén a többi hatását is vissza szeretnénk vonni
- Ha a lekérdezést az adatok (és nem pedig a kulcsok) alapján kell végezni
- Ha kulcsok halmazán kell műveletet végezni

Oszlopcsaládok

- Az értékeket mint map-of-maps-of-maps modellezi, oszlopcsaládok, oszlopok és időbélyeggel ellátott verziók segítségével
- Leginkább hasonlít a hagyományos (tábla) szerkezetre
  - Meg a kulcs-érték tárolóra is
- Oszlopcsalád ~ tábla a relációsban
  - Sorok felépítése:
    - Kulcs + oszlop (oszlopnév – érték - időbélyeg)
    - oszlopnév – érték – időbélyeg : tárolás alapeleme
    - Rugalmas szerkezet
- Egy adat beazonosításához szükséges: Táblanév, sor azonosítója, oszlopcsalád és benne az oszlop neve és az időbélyeg (verzió adat)



Oszlopcsaládok

Mikor használjuk

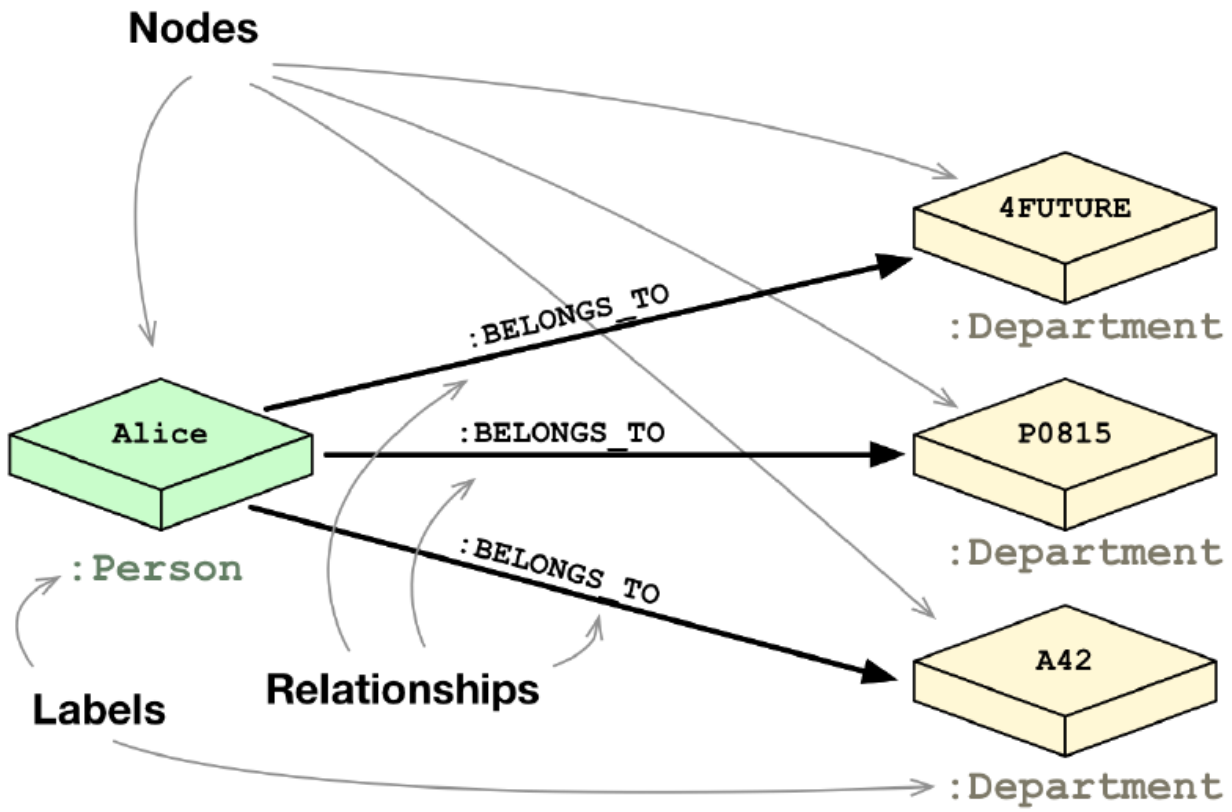
- Eseménynaplózás
- Tartalomkezelő rendszerek, blog platformok
- Számlálók: pl. egy webalkalmazásban meg kell számolni és kategorizálni kell az oldalak látogatóit
- Lejáró oszlopok: pl. demó hozzáférés vagy megadott ideig mutatott hirdetése, a nem kellő oszlopok egy megadott idő (time to live, TTL) után automatikusan törlődnek

Mikor ne használjuk

- Ha ACID tranzakciókra van szükségünk
- Ha a lekérdezés mintái változnak: az oszlopcsaládok áttervezésére lehet szükség

Gráf adatbázisok

- Adatok tárolása csomópontok és irányított élek formájában
- Adatelemek közötti összefüggések egyszerű modellezése
- Komplex, hierarchikus szerkezetű adatok tárolása
- Egyedek, kapcsolatok, tulajdonságok, címkék
- Címke (pl. :Person): sztring azonosító, sokszor típusnak használjuk, de nem hagyományos típus
- Csomópont, kapcsolat tulajdonságai kulcs-érték párok
- Egy lekérdezés tulajdonképpen egy bejárás
- Kiválóan vizualizálható
- Könnyű mintázatokat keresni



Gráf adatbázisok

Mikor használjuk

- Összekapcsolódó adatok: szociális, céges, stb. hálók
- Útvonalválasztás, hely alapú szolgáltatások
- Ajánlói rendszerek

Mikor ne használjuk

- Ha az összes (vagy elegendően sok) csomópontot módosítani kell
- Globális (a teljes gráfot érintő) gráfműveletek

Dokumentumtárolók

- A dokumentumok önleíróak, hierarchikus szerkezetűek, kollekciókat és skalár értékeket tartalmazhatnak
- Kulcs + komplex struktúra
  - Tartalmazhat
    - kulcs-érték párokat,
    - tömböket,
    - beágyazott dokumentumokat
- A sémák tetszőleges bonyolultságúak lehetnek, adatbázis által kezelt indexek is megjelennek
- Általában JSON, BSON vagy XML formátumban tárolt adatok
- Gyűjtemény ~ tábla, dokumentum ~ sor

```
{   "_id" : ObjectId("54c955492b7c8eb21818bd09"),
    "address" : {
        "street" : "2 Avenue",
        "zipcode" : "10075",
        "building" : "1480",
        "coord" : [ -73.9557413, 40.7720266 ]
    },
    "borough" : "Manhattan",
    "cuisine" : "Italian",
    "grades" : [
        {   "date" : ISODate("2014-10-01T00:00:00Z"),
            "grade" : "A",
            "score" : 11
        },
        {   "date" : ISODate("2014-01-16T00:00:00Z"),
            "grade" : "B",
            "score" : 17
        }
    ],
    "name" : "Vella",
    "restaurant_id" : "41704620"
}
```

MongoDB dokumentum részlet (JSON)

Dokumentumtárolók

Mire használjuk

- Eseménynaplózás
- Tartalomkezelő rendszerek, blog platformok
- Webes ill. valósidejű analitika
- E-kereskedelmi alkalmazások

Mikor ne használjuk

- Különböző műveleteken átívelő összetett tranzakciók esetén
- Változó aggregátumokra vonatkozó lekérdezések esetén