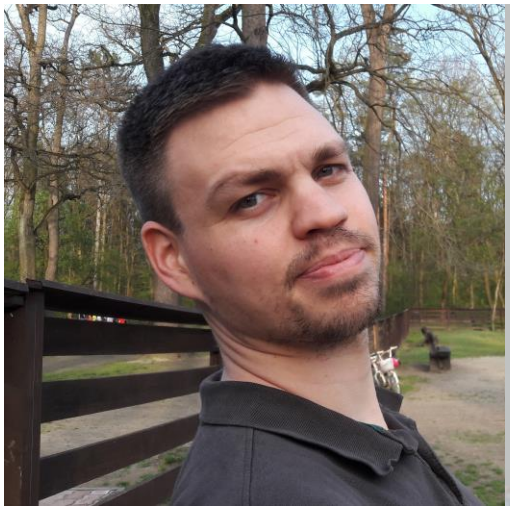



Dr. Hegedűs Péter, Dr. Ferenc Rudolf

Nagyméretű adatbázisok

Jelen tananyag a Szegedi Tudományegyetemen készült az Európai Unió támogatásával.

Projekt azonosító: EFOP-3.4.3-16-2016-00014

Szegedi Tudományegyetem
Cím: 6720 Szeged, Dugonics tér 13.
www.u-szeged.hu
www.szechenyi2020.hu

NoSQL Hadoop felett

Összefoglalás

Ez az olvasólecke a Hadoop ökoszisztéma azon eszközeit veszi górcső alá, amelyek NoSQL alapú adattárolást és lekérdezést tesznek lehetővé HDFS klaszteren. Az olvasó áttekintést kap a következő eszközökről: HBase, MongoDB, Cassandra. Ezek alapvető célja és funkcionalitása mellett az olvasólecke kitér arra is, hogy miben hasonlítanak illetve különböznek ezek az eszközök, és hogy milyen esetben melyiket célszerű használni.
A lecke fejezetei:

- 1. fejezet: **A NoSQL általános koncepciójának áttekintése, NoSQL adatbázisok típusai (olvasó)**
- 2. fejezet: **Az Apache HBase bemutatása (olvasó)**
- 3. fejezet: **Egyéb NoSQL adatbázisok (MongoDB, Cassandra) bemutatása, Hadoop integrációjuk lehetőségei (olvasó)**

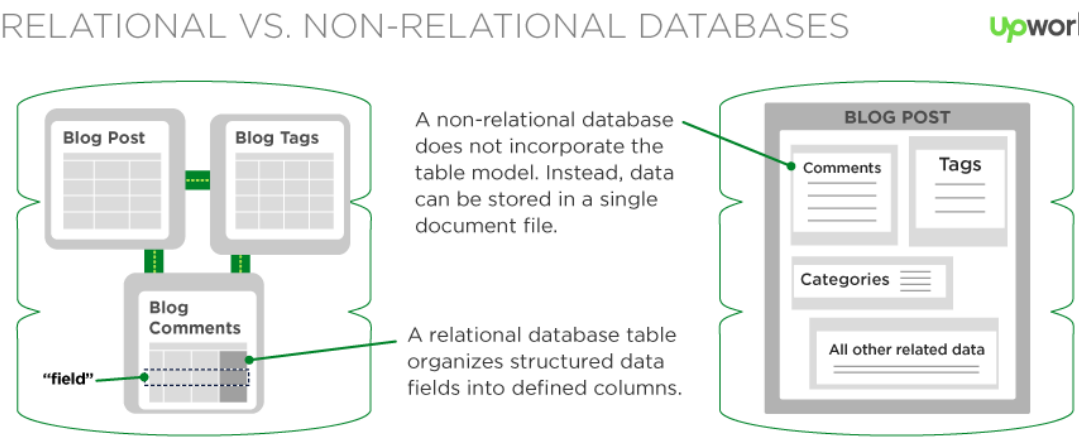
Téma típusa: **elméleti**
Olvasási idő: **50 perc**

📖 1. fejezet
Bevezetés a NoSQL adatbázisokba

A NoSQL egy adatbázis-kezelő rendszer (DBMS) osztály, amely nem követi a relációs DBMS (RDBMS) összes szabályát, és nem tudja a hagyományos SQL-t felhasználni adatok lekérdezésére. A NoSQL kifejezés kissé félrevezető, ha azt "nem SQL"-ként értelmezzük, inkább a "Nem csak SQL" áll közelebb a valósághoz, mivel az ilyen típusú adatbázisok általában nem helyettesítik, hanem kiegészítik az RDBMS-ket és az SQL-t.

A NoSQL-alapú rendszereket általában nagyon nagy adatbázisok esetén használják, amelyek különösen hajlamosak az SQL megszorításai és az adatbázisok relációs modellje által okozott teljesítmény problémákra. Sokan úgy gondolják, hogy a NoSQL a megfelelő modern választás adatbázisok terén, amely megfelel a kor webes alkalmazásaival szemben támasztott követelményeinek. A NoSQL néhány figyelemre méltó megvalósítása a Apache Cassandra [1] (eredetileg a Facebook fejlesztette), a Google BigTable [2] vagy az Amazon SimpleDB [3] és a Dynamo [4].

A NoSQL adatbázisok nem feltétlenül követik azokat a szigorú szabályokat, amelyek a relációs adatbázisok tranzakcióira vonatkoznak. Ezek a szabályok ACID (atomicity, consistency, isolation, durability) néven ismertek [5]. Például a NoSQL adatbázisok nem használnak rögzített sémastruktúrákat és SQL tábla összekapcsolásokat.

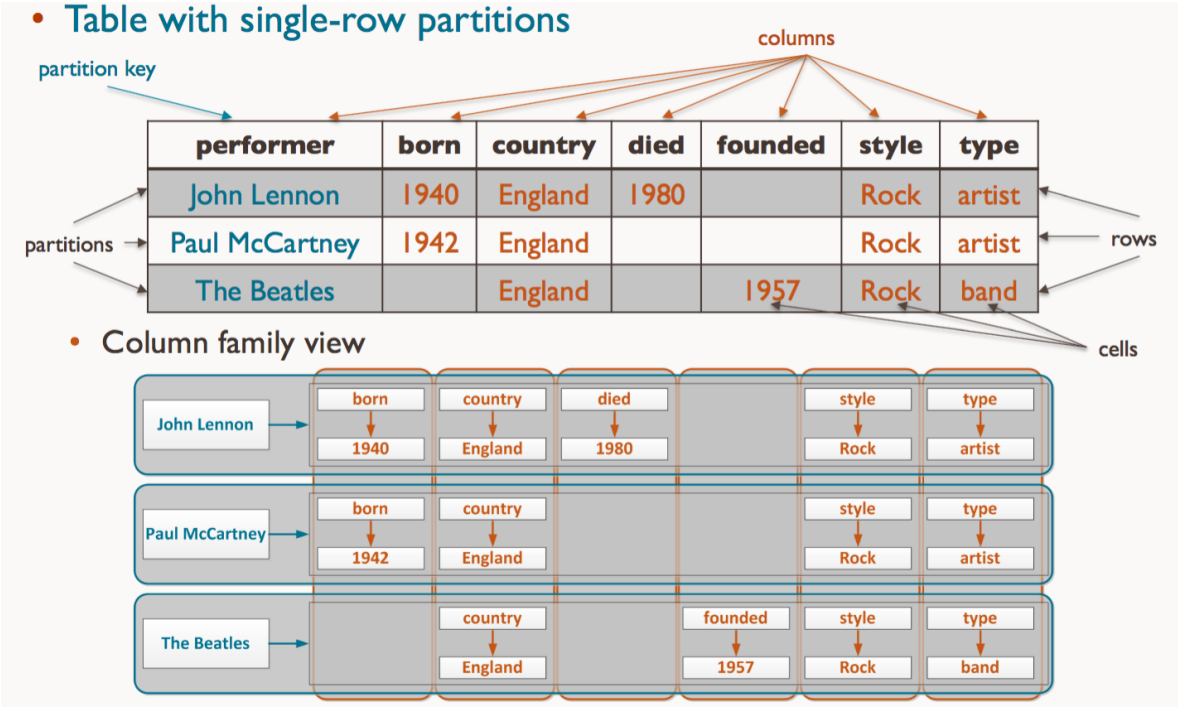


> https://assets-global.website-files.com/5ec7dad2e6f6295a9e2a23dd/5ee016f44336f8b9a40aa050_relational-vs-nonrelational-databases.png

A NoSQL adatbázisok könnyen skálázhatók, hiszen az alapoktól elosztott, több csomóponton futó szolgáltatásként tervezték őket. Ezáltal sokszor nagyobb teljesítményt, és az adatokra vonatkozó szigorú strukturális megkötések hiányában egyszerűbb felépítést és használatot biztosítanak az RDMBS-ekhez képest. Ugyanakkor sok esetben hátrányt jelenthet az, hogy nem tudják biztosítani az RDBMS által nyújtott sokrétű garanciát az adatokra vonatkozólag, illetve a komplex elemzési funkciókat.

A NoSQL adatbázisoknak különböző típusai léteznek, amelyek egy-egy konkrét probléma csoport megoldására koncentrálva implementálják a NoSQL koncepciókat. A leggyakrabban használt NoSQL adatbázisok a következők:

- *Kulcs-érték tárolók* - az adatokat kulcs-érték párokban tárolják, az értékeket a kulcs alapján gyorsan el lehet érni. Kulcsok rendezése, tartományok kezelése hatékony, de ezen kívül kevés egyéb művelet támogatott. Példák kulcs-érték tárolókra [6]: Riak, Cassandra, Redis.
- *Dokumentum tárolók* - tárolt elemek egysége a *dokumentum*. Ezek tetszőleges XML, YAML, JSON, stb. adatok lehetnek, séma nélküliek és a felépítésük sem kell, hogy megegyezzen. Minden dokumentum egyedi kulccsal rendelkezik, de a lekérdezésükhöz bonyolultabb lekérdező nyelvek (pl. MapReduce) illetve API-k is rendelkezésre állnak. Példák dokumentum tárolókra [7]: CouchDB, Elasticsearch, MongoDB.
- *Objektum tárol*ók - az objektum-orientált programozásban megszokott objektumokat tárolnak (OODBMS). A legtöbb objektumtároló valamilyen nyelvet is implementál az adatok lekérdezéséhez. Példák objektum tárolókra: JADE, ObjectDB.
- *Oszlop tárolók* - a relációs adatbázisokhoz hasonlóan sorokba és oszlopokba rendezve tárolják az adatokat, és az oszlopokat osztályokba (family) csoportosítják (a struktúrát így előre kell definiálni). Azonban az adatokat az RDBMS-ektől eltérő módon tárolják, és míg az előbbi a gyors sor műveletekre optimalizált, utóbbi az oszlopokon való gyors műveletekre van kitalálva. Első sorban komplex, változó tartalmú rekordok relációs tárolására ideális (lásd lenti ábra). Ilyen tárolók például a Google BigTable vagy az Apache HBase.
- *Gráf adatbázisok* - gráf típusú adatok (csomópontok és köztük lévő élek) hatékony tárolása. Ideális hálózati topológiák, közösségi háló, Wiki oldalak, stb. tárolásához. Példák gráf adatbázisokra [8]: Neo4j, OrientDB, MarkLogic.



📖 2. fejezet
Apache HBase

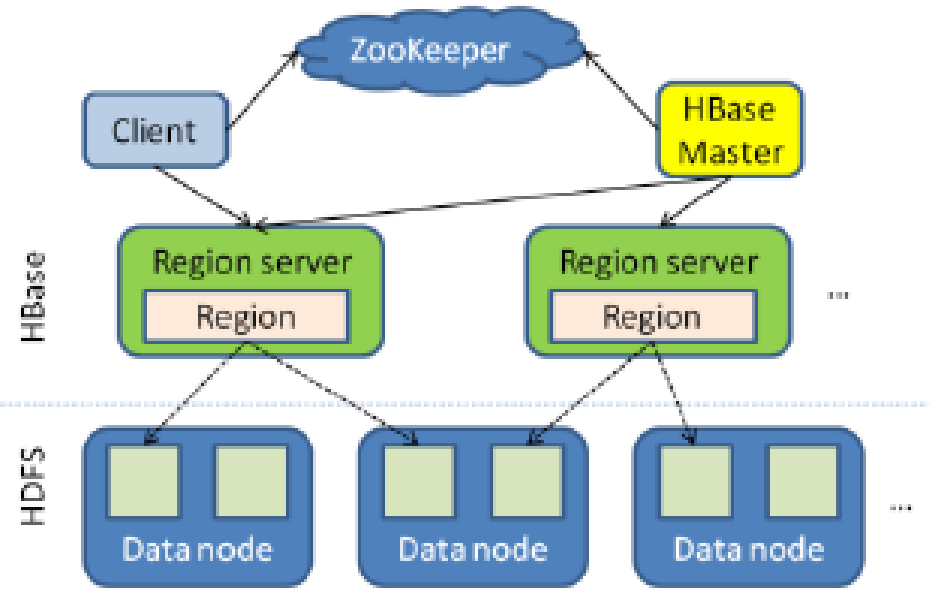
Az Apache HBase [9] egy elosztott, skálázható BigData adattároló Hadoop fölött. Egy oszlop tároló NoSQL adatbázis, amit a Google BigTable adatbázisa inspirált. Ahogyan a BigTable hatékonyan kihasználta a Google File System-en [10] elosztottan tárolt adatokat, a HBase ugyanúgy kihasználja a HDFS nyújtotta elosztott tárolást a hatékony adatkezeléshez.

A HBase tökéletes választás, amennyiben valós idejű véletlen írás/olvasási hozzáférésre van szükségünk a BigData-nkhoz. A HBase képes ilyen hatalmas adatmennyiséget kiszolgálni (akár milliárdos nagyságrendű sort milliós nagyságrendű oszloppal) hagyományos gépekből összeállított klaszteren. A HBase sokkal inkább "adattár" mint "adatbázis", mert számos jól megszokott RDBMS funkcióval nem rendelkezik, mint például a típusos oszlopok, másodlagos indexek, komplex lekérdező nyelv, stb.

Cserébe a HBase számos olyan funkcióval rendelkezik, ami támogatja a lineáris és moduláris skálázhatóságot. A HBase klaszterek egyszerűen növelhetők ún. `RegionServer`-ek hozzáadásával. Amennyiben például 10-ről 20-ra emeljük a `RegionServer`-ek számát, az azt jelenti, hogy megduplázzuk a klaszter adattárolási és számítási kapacitását is. Ezt a fajta skálázást RDBMS-ek esetén sokkal nehezebb és költségesebb elérni. Néhány figyelemre méltó HBase tulajdonság a következő:

- Erős adat konzisztencia mind íráshoz mind olvasáshoz (nem a legtöbb NoSQL által használt "eventual consistency" modellt használja)
- Automatikus adatelosztás a klaszteren belül
- Automatikus RegionServer helyreállítás hiba után
- Hadoop/HDFS támogatás az elosztott adattároláshoz
- MapReduce támogatás: HBase adatbázis job-ok adatforrásaként vagy eredmények tárolásához
- Java kliens API HBase programból történő eléréséhez
- REST API támogatás nem Java alkalmazások számára

Fontos leszögezni, hogy a HDFS és a HBase két különböző dolog (lást áttekintő ábra lent). A HDFS nem támogatja az adatok tetszőleges rekordjainak gyors keresését. Ugyan a HBase is HDFS-en tárolja az adatokat, azok eléréséhez oszloptároló alapú gyors tetszőleges rekord alapú keresést támogat. A HBase használatának több száz millió adatrekord esetén van értelme, az alatt célszerű RDBMS-ben tárolni az adatainkat.



> https://www.researchgate.net/profile/Xiaoming_Gao2/publication/228440170/figure/fig2/AS:301903809400834@1448990885592/HBase-architecture.png

HBase adat modell

A HBase-en belül az adatok táblában tárolódnak, amelynek sorai és oszlopai vannak. Ez a terminológia ugyan egybeesik a RDBMS definíciókkal, de nem érdemes így gondolnunk rá. A HBase táblákat sokkal inkább egy több dimenziós *map* adatszerkezetként érdemes elképzelni. A HBase által használt terminológia a következő.

Table

Egy HBase táblázat több sorból (row) áll.

Row

Egy sor egy kulcsból és egy vagy több oszlopból áll, amelyekhez érték van rendelve. A sorok kulcs szerint sorba rendezésre kerülnek tároláskor.

Pontosan emiatt a kulcs tervezése komoly jelentőséggel bír. Célszerű olyan kulcs értékeket választani, hogy az egymással kapcsolatban álló sorok közel

kerüljenek egymáshoz rendezéskor. Egy gyakori megközelítés kulcs értékekre a web domain-ek használata fordított sorrendben (hasonlóan pl. a Java

package-k elnevezéséhez). Így az egyes különböző al-domain-ek közel kerülnek egymáshoz a közös fő domain miatt.

Column

Egy oszlop a HBase-ben egy oszlop család (*column family*) és egy oszlop megnevezés (*column qualifier*) kombinációjából áll, amelyek `:`-al vannak

elválasztva.

Column Familiy

Az oszlop családok fizikailag összefogják az oszlopok egy halmazát és azok értékeit (főleg performancia megfontolásból). Minden oszlop család

rendelkezik bizonyos tárolási tulajdonságokkal, mint például az értékeiket gyorsítótárazni kell-e memóriában, hogyan kerülnek az adatai tömörítésre, a

sor kulcsa milyen kódolást használ, stb. Egy táblában szereplő összes sorhoz ugyanazok az oszlop családok tartoznak, ám lehetnek olyan sorok,

amelyek nem tárolnak semmit egy adott osztály családban.

Column Qualifier

Az oszlop megnevezés egy bizonyos adat jelölése érdekében adódik hozzá egy oszlop családhoz. Legyen pl. a `content` egy oszlop család, egy oszlop

megnevezés lehet a `content:html` vagy a `content:pdf`. Noha az oszlop családok tábla létrehozáskor definiálásra kerülnek, az oszlop megnevezések

módosíthatók és az egyes sorok nagyon különböző megnevezéseket tartalmazhatnak egy-egy osztályon belül.

Cell

A cella egy sor, oszlop család és oszlop megnevezés kombinációja, és egy értéket valamint egy időbélyegzőt tartalmaz, ami az adat verzióját jelöli.

Timestamp

Az időbélyegző minden egyes érték írásakor szintén eltárolódik, és egy adat verzióját azonosítja. Alapértelmezett esetben az időbélyegző a

RegionServer idejét jelenti az adat írásának pillanatában. Mindazonáltal lehetőség van az időbélyegző értékét máshogy definiálni az adat írásakor.

Adatok elvi nézete

Lássuk egy példán keresztül az adatok tárolásának elvi módját a fenti definíciók ismeretében. Adott egy tábla, `webtable` néven (lásd lent), ami két sort (`com.cnn.www` és `com.example.www`) és három oszlop családot (`contents`, `anchor` és `people`) tartalmaz. Az alábbi példában az első sorban az `anchor` családhoz két oszlop tartozik (`anchor:cssnsi.com`, `anchor:my.look.ca`), míg a `contents`-hez egy (`contents:html`). A `com.cnn.www` kulcsú sor a példában 5 verziót tartalmaz, míg a `com.example.www` kulcsú egyet. A `contents:html` oszlop tartalmazza egy web oldal teljes HTML tartalmát. Az `anchor` család megnevezései azokat a külső web oldalakat tartalmazzák, amelyek a sor által jelölt web oldalra mutatnak, a link-nél szereplő szöveggel együtt. A `people` család az oldalhoz köthető embereket írja le.

Oszlop nevek
Konvenció szerint az oszlopok megnevezése az oszlop család *prefix*-ből és a *megnevezésből* áll össze. A fenti példában a *contents:html* egy oszlop, amely a *contents* oszlop családból és a *html* megnevezésből tevődik össze, a kettőt egy kettőspont (:) választja el egymástól.

Row Key	Time Stamp	ColumnFamily `contents`	ColumnFamily `anchor`	ColumnFamily `people`
"com.cnn.www"	t9		anchor:cnnsi.com = "CNN"	
"com.cnn.www"	t8		anchor:my.look.ca = "CNN.com"	
"com.cnn.www"	t6	contents:html = "..."		
"com.cnn.www"	t5	contents:html = "..."		
"com.cnn.www"	t3	contents:html = "..."		
"com.example.www"	t5	contents:html = "..."		people:author = "John Doe"

A fenti táblázatban az üresnek tűnő cellák a valóságban nem foglalnak helyet, sőt, HBase-ben nem is léteznek. Ezáltal a HBase hatékonyan tudja tárolni az ilyen "ritka" adatokat. Az ilyen tabulált nézet nem az egyetlen módja a HBase adatok megjelenítésének, sőt, nem is a legpontosabb. A következő többdimenziós map ábrázolás sokkal közelebb áll a valósághoz:

```
{
  "com.cnn.www": {
    contents: {
      t6: contents:html: "<html>..."
      t5: contents:html: "<html>..."
      t3: contents:html: "<html>..."
    }
    anchor: {
      t9: anchor:cnnsi.com = "CNN"
      t8: anchor:my.look.ca = "CNN.com"
    }
    people: {}
  }
  "com.example.www": {
    contents: {
      t5: contents:html: "<html>..."
    }
    anchor: {}
    people: {
      t5: people:author: "John Doe"
    }
  }
}
```

Adatok fizikai nézete

Ellentétben azzal, hogy elvi szinten a táblák sorok ritka adatainak összességeként jelennek meg, a gyakorlatban teljesen máshogy, oszlop családok alapján vannak tárolva fizikailag. Egy új oszlop megnevezés bármikor hozzáadható egy oszlop családhoz.

Az `anchor` oszlop család tárolása:

Row Key	Time Stamp	Column Family `anchor`
"com.cnn.www"	t9	`anchor:cnnsi.com = "CNN"`
"com.cnn.www"	t8	`anchor:my.look.ca = "CNN.com"`

A `contents` oszlop család tárolása:

Row Key	Time Stamp	ColumnFamily `contents:`
"com.cnn.www"	t6	contents:html = "..."
"com.cnn.www"	t5	contents:html = "..."
"com.cnn.www"	t3	contents:html = "..."

A fenti valós tárolás jól mutatja, hogy az elviekben üres cellák egyáltalán nem is kerülnek tárolásra. Így például a `contens:html` oszlop értékének lekérése a `t8` időbélyegzővel nem adna vissza értéket. Amennyiben nem adunk meg időbélyegzőt, akkor az oszlop legfrissebb verziójának értéke adódik vissza. Amennyiben egy értéknek több verziója is létezik, a legfrissebb időbélyegzővel rendelkezőt találjuk meg legelőször, hiszen az értékek az időbélyegzők szerint csökkenő sorrendbe vannak rendezve. Ha például az összes oszlop értékét kérnénk le időbélyegző nélkül a `com.cnn.www` sorhoz, a következő értékek adódnának vissza: `contents:html` érték a `t6` időbélyegzővel, az `anchor:cnnsi.com` értéke a `t9` időbélyegzővel és az `anchor:my.look.ca` érétke a `t8` időbélyegzővel.

Az adatok lekérdezéséhez a HBase Shell nyújt interaktív felületet, illetve a Java API-n keresztül programból lehet az adatokat manipulálni. A HBase működésének bővebb leírásért lásd a [11, 12]-es hivatkozásokat.

📖 3. fejezet
Egyéb NoSQL megoldások, amelyek integrálhatók a Hadoop ökoszistémába

MongoDB

A MongoDB [13] egy dokumentum tároló NoSQL adatbázis, amely JSON szerű dokumentumokban (BSON) tárolja az adatokat. A lekérdezések támogatására egy gazdag és kifejező nyelvet ad, amivel könnyedén lehet az adatokat rendezni és szűrni tetszőleges mező alapján. A lekérdezéseket szintén JSON formában lehet megfogalmazni, és támogatják a különböző aggregációs műveleteket is. A legtöbb NoSQL-el szemben támogatja az ACID tranzakciókat és a lekérdezések összekapcsolását - join (RDBMS-ekkel azonos követelmények), ráadásul két fajta kapcsolatot is támogat, nem csak egyet, mint a relációs adatbázisok: referencia és beágyazás.

A MongoDB főbb tulajdonságai a következők:

- **Ad hoc lekérdezések**

  A MongoDB 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**

  A MongoDB lehetővé teszi, hogy a dokumentum bármelyik mezője alapján indexet készítsünk.

- **Replikáció**

  A MongoDB támogatja a ~~master-slave~~ primary-worker replikációt. Ebben az esetben a 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.

- **Terheléselosztás**

  A MongoDB horizontálisan skálázható sharding [14] használatával. A fejlesztőnek kell shard kulcsot választania, amely meghatározza, hogyan lesz elosztva gyűjtemény adathalmaza.

- **Fájl tároló**

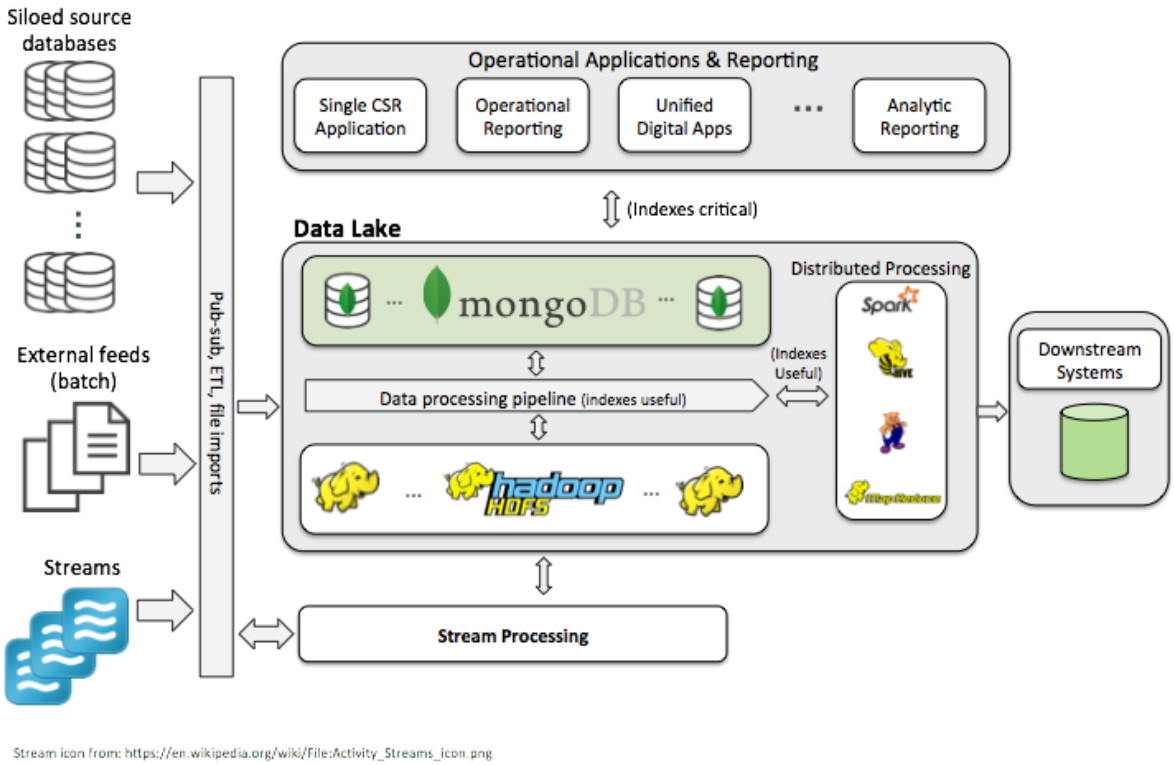
  A MongoDB-t lehet elosztott fájlrendszerként is használni, ezt GridFS-nek hívják.

- **Aggregáció**

  MapReduce algoritmusok használhatók kötegelt feldolgozáshoz és aggregációra. Ezeket a programokat JavaScript nyelven kell megírni.

Integráció Hadoop-pal

A MongoDB nem integrálható olyan szorosan a Hadoop-pal, mint a HBase. A HBase-t eleve úgy tervezték, hogy az adatok elosztott tárolásához a HDFS-t használja, míg a MongoDB rendelkezik saját elosztott tárolási megvalósítással és teljesen önállóan tud működni. Azonban lehetőség van arra, hogy a MongoDB Hadoop connector [15] segítségével integráljuk a MongoDB-t a Hadoop ökoszisztémába (lásd lenti ábra).



Stream icon from: https://en.wikipedia.org/wiki/File:Activity_Streams_icon.png

> https://webassets.mongodb.com/_com_assets/cms/big-data-architecture-mongodb-k0znyrbtlz.png

A Hadoop connector lehetővé teszi, hogy a MapReduce job-ok a bemenő adatokat egy MongoDB adatbázisból, vagy annak HDFS exportjából (BSON fájlok) vegyék, illetve az eredményüket szintén BSON formátumban HDFS-re írják (ami aztán importálható a MongoDB-be), vagy közvetlenül egy MongoDB adatbázisba tárolják.

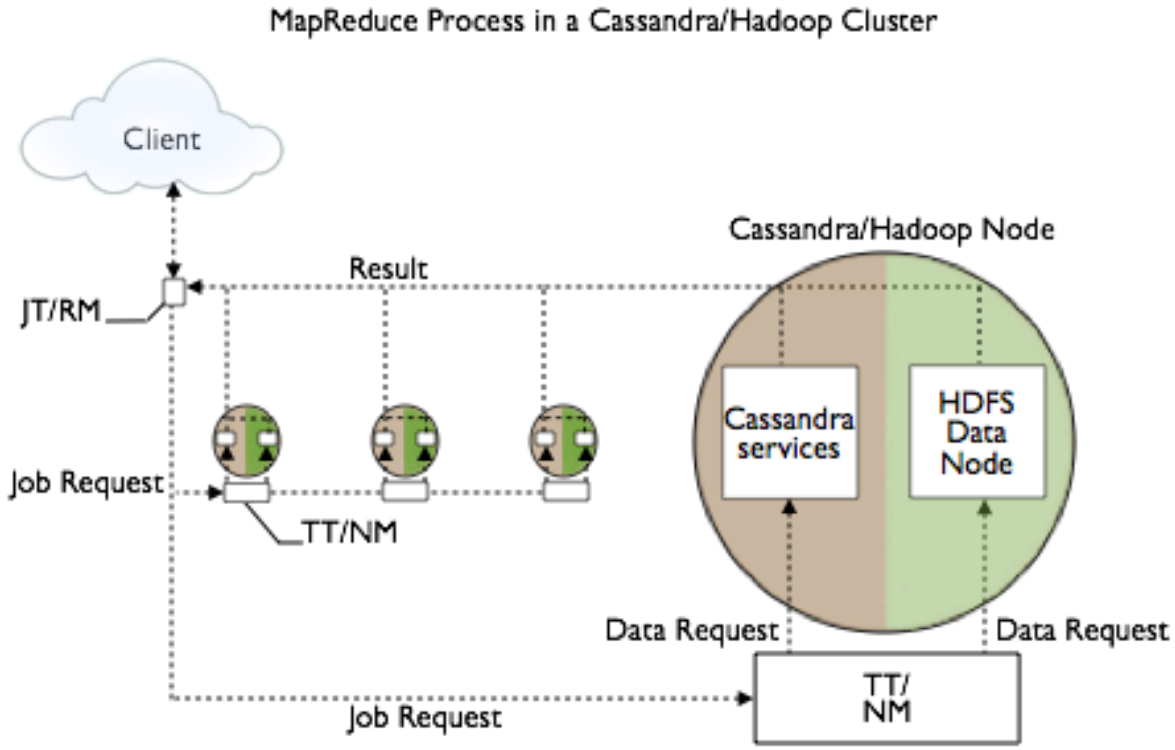
Apache Cassandra

Az Apache Cassandra [16] egy nyílt forráskódú elosztott adatbázis szerver szoftver, amelyet arra terveztek, hogy nagy mennyiségű adatot tároljon alacsony költségű szervereken, magas rendelkezésre állást mellett. A Cassandra fejlesztését az Amazon DynamoDB egyik fejlesztőjeként is ismert Avinash Lakshman és Prashant Malik kezdte a Facebook-nál. A forráskódot 2008-ban adták ki nyílt forráskódú projektként, majd 2009 márciusában lett az Apache inkubátor tagja. 2010 február 17-én lett felső szintű Apache projekt.

A Cassandra adatmodellje a kulcs-érték és az oszlop tárolás hibrid megközelítése. A Cassandra a klaszteren belüli kommunikációhoz egy pletyka protokollt (gossip protocol) használ. Könnyen skálázható (lineáris skálázás hagyományos számítógépeken) és magas rendelkezésre állás jellemzi. A lekérdezéshez a Cassandra egy CQL (Cassandra Query Language) [17] lekérdező nyelvet ad, ami SQL-szerű lekérdezést tesz lehetővé az adatbázison.

Integráció Hadoop-pal

A Cassandra hasonló mélységig integrálható a Hadoop klaszterekkel, mint a MongoDB, azaz az adattárolást és Cassandra elosztott környezetet a Hadoop-tól függetlenül kell üzemeltetnünk. Sőt, szükséges az, hogy bizonyos node-okon csak Cassandra fusson (online tranzakciók feldolgozásához, míg másokon Hadoop DataNode és Cassandra is. Ezután a MapReduce job-ok betölthetik az adataikat a Cassandra adatbázisból (CQL táblákból és indexekből), illetve a kimenetet is írhatják közvetlenül Cassandra-ba.



Ellenőrző kérdések

1. Mit nevezünk NoSQL adatbázisnak?
2. Mik a főbb különbségek a relációs adatbázisok (SQL) és NoSQL adatbázisok között?
3. Milyen NoSQL adatbázis típusokat ismersz?
4. Mik az Apache HBase főbb jellemzői, hogyan kapcsolódik a Hadoop ökoszisztémához?
5. Milyen adatmodellt használ az Apache HBase?
6. Mi a különbség az oszlop az oszlop család és az oszlop megnevezés (column, column family, column qualifier) között?

7. Hogyan küszöböli ki a HBase a ritka adattáblák tárolási problémáját (a sok üres cella tárolását)?
8. Mire szolgál a sor kulcs és az időbélyegző a HBase táblákban?
9. Milyen adatmodellt használ a MongoDB?
10. Hogyan integrálható a MongoDB Hadoop-pal?
11. Milyen adatmodellt használ az Apache Cassandra?
12. Hogyan integrálható a Cassandra Hadoop-pal?
13. Mik az alapvető különbségek a HBase, MongoDB és Cassandra NoSQL adatbázisok között?

Referenciák

[1] https://en.wikipedia.org/wiki/Apache_Cassandra

[2] https://cloud.google.com/bigtable

[3] https://aws.amazon.com/simpledb/

[4] https://aws.amazon.com/dynamodb/

[5] https://www.educative.io/edpresso/what-are-acid-properties-in-a-database

[6] https://en.wikipedia.org/wiki/Key-value_database

[7] https://en.wikipedia.org/wiki/Document-oriented_database

[8] https://en.wikipedia.org/wiki/Graph_database

[9] https://hbase.apache.org/

[10] https://research.google.com/archive/bigtable.html

[11] https://dzone.com/articles/understanding-hbase-and-bigtab

[12] http://0b4af6cdc2f0c5998459-c0245c5c937c5dedcca3f1764ecc9b2f.r43.cf2.rackcdn.com/9353-login1210_khurana.pdf

[13] https://www.mongodb.com/

[14] https://en.wikipedia.org/wiki/Shard_(database_architecture)

[15] https://github.com/mongodb/mongo-hadoop

[16] https://cassandra.apache.org/

[17] https://cassandra.apache.org/doc/latest/cql/index.html