Klasszikus adattárházak és BIG DATA: Pro és Kontra

Szabó Rozalinda

Adattárház szakértő, Architekt, oktató

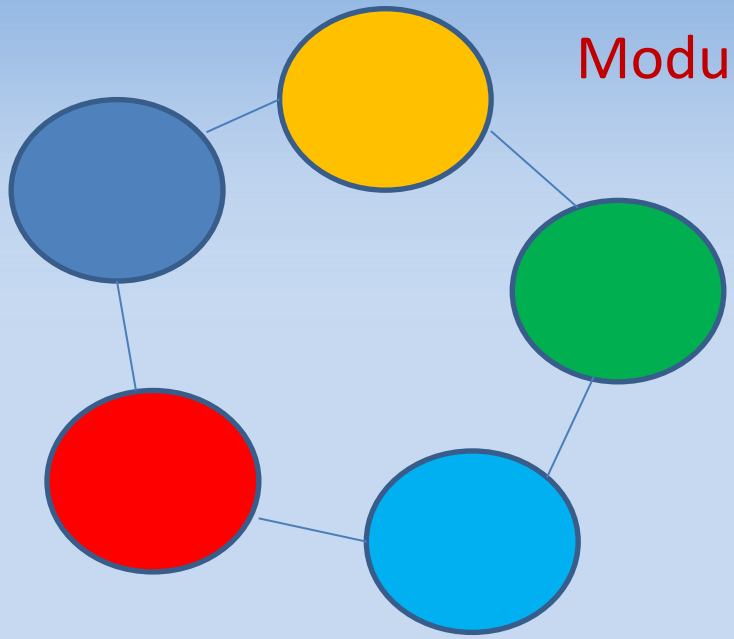
szabo.rozalinda@gmail.com

Adattárházak fejlődése

- Magyarországon már kb. 20 évvel ezelőtt indultak az első adattárházak, akkor még relációs környezetben
- Kb. 10 évvel ezelőtt már virágzott az adattárház piac, de még mindig főként relációs alapokon
- Azóta az adattárházak is sokat fejlődtek az új technológiáknak köszönhetően:
  - Big data (már akár tranzakció kezeléssel is)
  - Felhő
  - Elosztott rendszerek (már relációsok is tudják kezelni)
  - Streamelés

Relációs adatbázisok és Hadoop eltérései

Hadoop



Modulok együttműködése:

- Adat betöltő
- Adat lekérdező
- Jogosultság kezelő
- Folyamat vezérlő
- Streaming
- stb.

Relációs adatbázis-kezelő

- Összeintegrált
- Zárt rendszer

- Nyílt forráskódú
- Választható modul készlet

Relációs adatbázisok és Hadoop eltérései

Az alábbi eltérések miatt mindenképpen más szemlélet szükséges az adattárház építésekor:

- SQL nyelvek és mögöttes technológiai eltérések
- Adat struktúra ellenőrzés, hibás adat kezelése
- Integritási megszorítások
- Mesterséges azonosító
- Tranzakciókezelés, Lockolási mechanizmus
- Indexek használata
- PARQUET file formátum használata
- UPDATE, DELETE, MERGE műveletek
- Tábla adatok helyre állítása
- Sor szintű jogosultság kezelése

SQL nyelv használata – Relációs adatbázis kezelő

Relációs adatbázis kezelők (pl. Oracle, MSSQL) egy féle SQL nyelvet használnak egy adatbázison belül.

Például:

Oracle	MSSQL
Oracle SQL	T - SQL

SQL nyelv használata – HADOOP

Hadoop világban előfordul, hogy akár kétféle SQL motort is érdemes használni egy adatbázis felett.

Például:

HIVE	IMPALA

Ezek nem csak szintaxisban, hanem működésben is nagyon eltérnek.

Ez bonyolultabbá teszi a használatot.

Adat struktúrák ellenőrzése

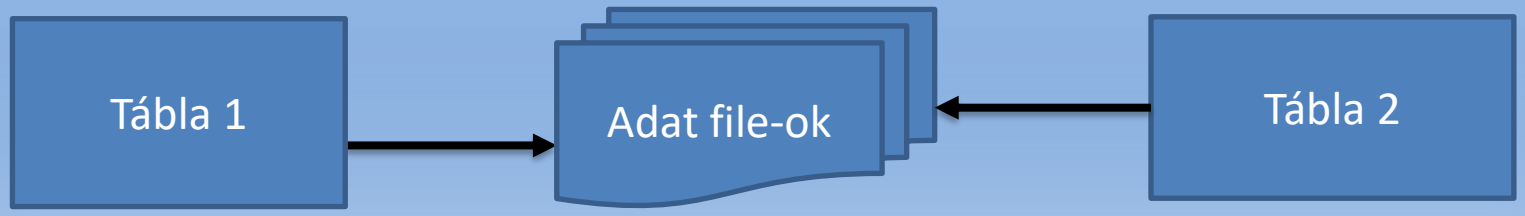
Az is eltérő, hogy a relációs adatbázis-kezelők és a Hadoop esetén mikor és hogyan történik az adatstruktúra ellenőrzése.

- Íráskor: az adatok adatbázisba írása előtt
- Olvasáskor: az adatok felolvasásakor

Például:

Relációs adatbázis	HIVE	Impala
Íráskor	Íráskor (ez sem a megszokott módon) **Bizonyos esetekben csak Olvasáskor !!**	Íráskor

Adatok ellenőrzése olvasáskor - HIVE



- Egy tábla mögött lehet több adat file
- Egy file-ra mutathat több tábla is

LOAD műveletnél csak annyi történik, hogy bemásolja az adat file-t a tábla alá, de nem ellenőrzi le ekkor, hogy a file-ban lévő struktúra megfelel-e a táblának.
Csak akkor ellenőriz, amikor azt a táblát SELECT-tel felolvassuk.
Ha a file-ban van olyan adat, amely nem felel meg az előírt struktúrának, akkor a SELECT hibát üzen vagy null-t ad vissza

Adatok ellenőrzése íráskor - HIVE

Hive-ban INSERT [OVERWRITE] használatakor szükség esetén automatikus konverzió történik, de ha az adat így sem felel meg az előírt oszloptípusnak, akkor az adatbázisba NULL értéket rak be, és HIBÁT NEM ÜZEN !!!

Mit tehetünk?

Ha semmi esetre sem szeretnénk adatot veszíteni, akkor azt a mezőt tegyük STRING típusúvá (mondjuk egy STAGE rétegbe töltésnél), és a DWH fentebbi rétegeiben már használhatunk típusos mezőket

Integritási megszorítások – Relációs adatbázis

Bevett szokás, hogy az alábbi megszorításokat használjuk relációs adattárházakban:

- Elsődleges kulcs (egyedi, de nem üres érték)
- Egyedi kulcs (egyedi, de lehet üres érték)
- Kötelező mező (nem lehet üres érték)

Integritási megszorítások - HADOOP

Mit tehetünk, ha Hadoop-ban nem áll rendelkezésre, de szükség lenne rá?

- Üzleti igényektől függően, rendszeresen futó, automatizált ellenőrző lekérdezések futtatása
- Hibrid környezet esetén, ellenőrizendő oszlopok / táblák áttöltése olyan adatbázisba, ahol a megszorítás rendelkezésre áll

Mesterséges azonosító

Relációs adatbázis	Hadoop
- Szekvencia	- Hiányozhat
- IDENTITY oszlop	- Hiányozhat
- Univerzális egyedi azonosító (UUID)	
- HASH kulcs (pl. üzleti kulcsra)	

UUID és HASH hátránya:

- Hosszú
- nem 100%-osan egyedi
  (bár az ismétlődésre nagyon minimális esély van)

Hash kulcs előnye:

Bármikor újra tudjuk reprodukálni ugyanazokat a kulcsokat ugyanazon rekordokhoz

Tranzakció-kezelés

Relációs adatbázisok	HADOOP
Konfigurálható, Default beállítás eltérő ORACLE és MSSQL-ben !	**Hiányozhat!**

Lockolás mechanizmusa

Relációs adatbázisok	HADOOP
Konfigurálható, Default beállítás eltérő ORACLE és MSSQL-ben!	**Impala:** Nem lockol egyáltalán, **még az írás sem!!** (adatkonzisztencia megőrzése miatt itt figyelni kell)
	HIVE: Nem csak a módosítás, de az olvasás is lockolhat bizonyos verziókban (ez viszont lassító tényező a működésben)

Indexek használata – Relációs adatbázisban

Az indexek:

- bizonyos helyzetekben gyorsítani tudják a végrehajtandó műveleteket
- de tudnak lassítani is, ha rosszkor vagy rosszul használjuk!

Relációs adatbázisokban számos különféle index létezik.

Indexek használata – HIVE, IMPALA

A Hive 3.0-tól nincsenek indexek.

IMPALA-ban sincsenek indexek (kivéve LZO tömörített text file).

Mit tehetünk?

- PARQUET file formátum használata (oszlop alapú tárolás). Az adatblokkban lévő statisztika mutatja egy adott oszlopra vonatkozóan,hogy az adatblokkban milyen értékek vannak. Ezért csak azt a blokkot olvassa fel, ami szükséges!
- Az adatok rendezése is segíthet

Min nem tud ez a file formátum segíteni?

- Rendezetlen Egyedi azonosítók
- Rendezetlen nagy értékkészlettel bíró oszlopok

Ebben az esetben a teljes partíciót / táblát fel kell olvasni!

Parquet tábla ALTER-je HIVE-ban

Teljesen más, mint egy relációs táblán!

Parquet.column.index.access paraméter:

False: az adat file-ban az oszlopot név szerint találja meg

Probléma: **oszlop átnevezése**, mert már többet nem fogja megtalálni azt az oszlopot az új nevén

True: az adat file-ban az oszlopot pozíció alapján éri el

Probléma: **nem utolsó oszlop törlése**, a struktúra definíciója nem lesz összhangban a file-lal, tehát elcsúsznak az oszlopok, és rossz oszlopot olvas ki a file-ból.

Mindkét paraméter beállításnál érhet minket meglepetés!

Parquet tábla olvasása - IMPALA

Impala-ban is állítható, hogy milyen módon olvassa a PARQUET file-t.

PARQUET_FALLBACK_SCHEMA_RESOLUTION paraméter:

1: az adat file-ban az oszlopot **név** szerint találja meg

0: az adat file-ban az oszlopot **pozíció** alapján éri el

Mit tehetünk, hogy ne lehessen az ALTER-ből probléma?

A tábla definíciója és file struktúrája legyen mindig egyforma.

Változáskor szervezzük újra az adatokat.

Így a HIVE és Impala is jól fogja olvasni a táblát a DEFAULT beállításokkal.

UPDATE, DELETE, MERGE művelet – Relációs adatbázis

Tranzakciós rendszerekben sokkal gyakoribb, de azért adattárházakban is használunk DELETE, UPDATE, MERGE műveleteket.

Bár adattárházakban csak nagyon szükséges esetekben használjuk, mivel időigényesebb tud lenni, mint az INSERT művelet.

UPDATE, DELETE, MERGE – HADOOP

Attól függően, hogy milyen STORAGE-ot használunk, és milyen HADOOP környezetet, előfordul, hogy nem áll rendelkezésre UPDATE, DELETE, MERGE művelet.
Ez alapjaiban megrengeti egy adattárház töltési folyamatának (ETL) tervezését.

Teljesen más szemléletet kíván a relációs adatbázis környezethez képest.

Mi ennek a következménye?

Csak úgy tudunk adatokat módosítani vagy törölni SQL-el, hogy ha az érintett teljes partíciót, vagy particionálás hiányában az egész táblát **truncate**-eljük, majd beszúrjuk a táblába:

- a módosulatlan adatokat
- a módosult adatok új állapotát
- Kihagyjuk a törlendő rekordokat

Mit tehetünk?

- Érdemes az adatok érvényességének dátumát elhagyni (abszurdnak tűnik elsőre)
- Megszokotthoz képest másképp kell felépíteni az ETL folyamatot

Tábla adatainak helyre állítása

Relációs adatbázisban nem mindig olyan egyszerű visszaállítani csak egy tábla tartalmát, anélkül, hogy az adatbázis többi része változatlan maradjon.

HADOOP-on mivel tudjuk egészen pontosan, hogy egy táblához mely adat file-ok tartoznak, így egyszerűen ki lehet a tábla alatt cserélni a file-okat, és könnyen vissza lehet állni egy korábbi állapotra.

Sor szintű jogosultságkezelés

Oracle	MSSQL	HADOOP
Van	Van	**Hiányozhat**

Mit tehetünk, ha hiányzik, és szükség van rá?

- A jogosultságkezelést leíró táblákat összekapcsoljuk a sor szintű jogosultságkezelésben érintett adattáblákkal
- Lehet alkalmazni nézeteket is, mely támogatja a fenti szűrést
- Áttehetjük az érintett adatokat egy olyan adatbázis kezelőbe, ahol van sor szintű jogosultságkezelés
- BI eszközre bízzuk rá a sorszintű jogosultságkezelést

Sikeres Adattárház építés

- Sokféleképpen és többféle platformon történhet
- Alaposan meg kell ismerni az aktuális platform nyújtotta lehetőségeket,előnyöket és hátrányokat
- De legalább olyan fontos tudni:
  - A pontos üzleti igényeket és a környezet adottságait
  - Az adatok tulajdonságait és összefüggéseit

Köszönöm a figyelmet!