MPP Adattárház Teradata alapokon

Tanulmány az Adatbázisok haladóknak c. tárgyhoz

Lévai Ákos

PRISE Kft.

2012/2013 tanév I. félév

Tartalomjegyzék

Bevezetés

Az adattárházak (angolul Data Warehouse, DWH) széleskörű elterjedése a kilencvenes évek végére tehető. Népszerűségük töretlen, egyre integránsabb részét képezik a vállalati IT infrastruktúrának.
A rohamosan növekvő adatmennyiségek és az egyre komplexebb feladatok kihívások elé állítják a technológiai szállítókat. A tanulmány célja, hogy betekintést adjon egy hatékony megoldás részleteibe.

Előzmények

Az adattárháznak két elterjedt definícióját szoktuk használni Bill Inmon és Ralph Kimball tollából:

Inmon
"A data warehouse is a subject oriented, integrated, nonvolatile, and time variant collection of data in support of management's decisions."

Nagyjából ez annyit tesz, hogy az adattárház egy témaorientált, integrált, nem változó, idősoros adathalmaz, amit döntéstámogatás céljából tárolunk.

Kimball
"The conglomeration of an organization's data warehouse staging and presentation areas, where operational data is specifically structured for query and analysis performance and ease-of-use."

Ez a definíció az adattárház felépítéséből indul ki: Az adattárház egy szervezet/vállalat azon adattároló és megjelenítő infrastruktúrája, ahol az üzletmenetet biztosító rendszerek adatait lekérdezések és analízis céljából teljesítményre optimalizálva átstrukturálják.

Az első (korábbi) definíció inkább a megoldás célját, míg a második a felépítését/módszertanát írja le.

Az MPP adattárház

Adattárházakról röviden

Mire is jó az adattárház?

Már a definíciókból is sejthető, hogy az adattárházakat elsősorban arra használják, hogy riportokat, elemzéseket futtassanak rajtuk. Az idő haladtával ezen rendszerek sokat fejlődtek, és már a múlt elemzése mellett a jövő megjóslására, sőt az üzletmenet aktív támogatására is használják.
Nézzünk konkrét példákat! Amikor az egyik legnépszerűbb internetes könyváruházban böngészgetünk, akkor az általunk megtekintett tételek mellett különböző ajánlásokat is kapunk. Emögött egy „aktív adattárház" működik, ami folyamatosan követi a vásárlási szokásainkat, és ez alapján próbál olyan termékeket kínálni, amik valószínűsíthetően érdekelnek minket (cross-selling)
Teljesen más példa a társadalombiztosításokat támogató megoldás, ahol a felírt recepteket analizálják valós időben, lehetővé téve a téves diagnózisok kiszűrését, ill. a felesleges, vagy indokolatlanul drága gyógyszerek felírását.

Miben különbözik az adattárház egy operatív/tranzakciós rendszertől?

Az operational (vagy operatív) rendszerek elsődleges felelőssége, hogy biztosítsák az akadálytalan üzletmenetet, és tegyék ezt racionális költségszint mellett. Az ilyen rendszerek felépítésének kialakításakor fő szempontok a belső konzisztencia biztosítása, gyors válaszidők és jól körülírt feladatok optimális végrehajtása. Ennek érdekében a következő tulajdonságok szoktak teljesülni:

- Idegen kulcsok (Foreign Key) aktiválva
- Indexek a keresendő mezőkre
- 3NF adatmodellezés
- Korlátozott idejű tranzakció tárolás (amíg az üzletmenethez kell)
- Nem historikus referenciák (pl. egy kódhoz csak a most érvényes megnevezés)

Az adattárházak ezen szempontok mentén általában pont az ellenkező tulajdonságokkal rendelkeznek, mivel mások a célok. Emellett az adattárházakra általában igaz, hogy:

- Nem szükséges 100% pontosság
- Nem kell minden adatnak azonnal megjelennie benne
- Cég szinten egységes fogalmak és kódolás (pl. ügyfél-azonosító)

Üzleti elvárások

Az adattárházak felhasználói az alábbi igényeket támasztják a rendszerrel szemben:

- Ad-hoc és rendszeres lekérdezések futtatása. Ezek között az a lényegi különbség, hogy míg a rendszeres lekérdezések igényeit figyelembe lehet venni akár már tervezési időben, addig az ad-hoc lekérdezések előre nem kiszámíthatók, ezekre nem lehet általánosságban felkészülni (pl. indexeléssel)
- Hosszú időszak lefedése. Az üzletmenetben megfigyelhetők különböző időállandójú periodikus folyamatok (pl. távközlésben a nyári roaming), amik elemzéséhez/tervezéséhez legalább az előző periódus, de inkább több megelőző időszak adatai lehetnek szükségesek. Ez a példa esetében akár több évre visszamenő tárolást jelent. A trendek elemzése pedig akár még hosszabb időszakok elérhetőségét is feltételezi.
- Megfelelő válaszidő. A különböző feladatok nem egyforma válaszidőt igényelnek. Az Amazon-os vásárlás során az információra „azonnal" van szükség, azaz <1sec a tolerálható késedelem. Egy elemző, aki azt szeretné megtudni, hogy az általa kitalált roaming árstruktúra mekkora nyereséget hozna a következő nyaralási időszakban, türelmesen vár 1-2 órát a válaszra. Ha mindenkit „gyorsan" akarnánk kiszolgálni, az nagyon megemelné az adattárház költségeit.
- Skálázhatóság. Ez az igény főként pénzügyi szempontból fontos. Egy IT rendszer infrastruktúrájának pénzügyi szempontból optimális méretezése nem egyszerű feladat. Jól meg kell becsülni az adat és feladat (workload) mennyiség alakulását, a feldolgozó és tároló kapacitás árának jövőbeli viszonyait, valamint az upgrade-ek költségeit. Ha egy rendszer biztosítja annak a lehetőségét, hogy nem kell a teljes kapacitást előre megvásárolni, hanem az lényegében modulárisan bővíthető, nagyban megkönnyíti az optimális méretezést.

Válaszok a kihívásokra

- Céloptimalizált alkalmazás. Ez a megközelítés általános célú alapokon (főként Relációs Adatbázis kezelő /RDBMS) építkezik, annak optimalizálásával, a pontos felhasználási célhoz igazított tárolási struktúrák kialakításával.
- Céloptimalizált RDBMS. Ezen adatbázis-kezelők architektúrája az adattárház feladatokra van optimalizálva, a pontos felhasználás ismerete nélkül hatékonyan birkózik meg a feladatokkal.

IT technikák az adattárház megoldásokban

RDBMS-en kívüli megoldás

Az utóbbi években alakult ki az adatbáziskezelőn kívüli párhuzamos feldolgozás új technológiája (Hadoop, MapReduce), ami kifejezetten alkalmas a web alapú rendszerek adatainak (clickstream) analízisére. Előnye, hogy igen olcsón készíthetők megoldások, hiszen a technológia opensource, és a legolcsóbb hardvereken is futtathatók. Hátrányuk, hogy nem az adatkezelésben standard SQL nyelven írhatók le a műveletek.

Adatmodellezés

Az operatív rendszerek esetében leggyakrabbak 3NF adatmodellezést használunk, ami a rendszeren belüli adatkonzisztanciák hatékony fenntartását és az egytáblás alacsony adatmódosítási válaszidőket kiválóan támogatja. Azonban a megoldás hátránya, hogy egy-egy komplexebb lekérdezés esetében számos (műveletigényes) tábla összekapcsolást (JOIN) kell elvégezni. Az adattárházak esetében gyakran élünk a denormalizálás eszközével, ami a lekérdezési hatékonyságot nagyban növeli. Denormalizálást logikai modellezéskor is használhatunk, illetve a mai RDBMS-ek már szinte mindegyike támogatja az ún. „materializált view"-k létrehozását, ami a normalizált adattáblák felett karbantart egy denormalizált réteget.

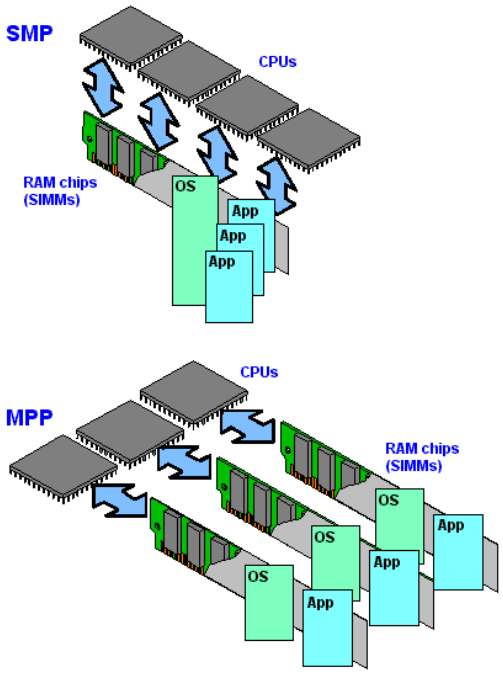
„Multi-temperature” adattárolás
Az adatok tárolásának költsége nagyságrendben változik attól függően, hogy az adott tárolás milyen sebességű/késleltetésű hozzáférést biztosít. Ezzel párhuzamosan az adathozzáférésekben is megfigyelhető a „Pareto-elv”, miszerint az esetek 80%-ában az adatok 20%-ára van csak szükség. Ezen tények ismeretében érdemes olyan megoldásokat építeni, ahol a ritkán elért adatokat lassabb, de jóval olcsóbb médián tároljuk.

Fejlett erőforrás-kezelés
Az adattárházakban a különböző válaszidő igények egyetlen rendszeren belül léteznek, így megoldandó, hogy bizonyos csatornák (pl. dedikált userek) akkor is gyors választ kapjanak, ha a rendszer egyéb csatornákon nagyon le van terhelve. Léteznek olyan megoldások, amikor a rendszer erőforrásainak egy részét dedikálják egy csatornához (pl. időosztás), azonban sokkal hatékonyabb, ha a határok dinamikusan változnak az igényeknek megfelelően.

Párhuzamosítás
Az adatkezelési feladatok legtöbbjének megoldására létezik olyan algoritmus, ami hatékonyan párhuzamosítható. A kommunikációs hálózatok kapacitása és fajlagos költsége olyan szintet ért el, hogy a párhuzamosítást érdemes „dobozon kívülre”, azaz több/sok számítógépre kiterjeszteni.

MPP RDBMS architektúra

Az MPP (Massive Parallel Processing) rendszerek sajátja, hogy sok, egyforma unit működik egymással párhuzamosan, miközben az egyes unitok számára dedikált erőforrások állnak rendelkezésére (RAM,CPU,DISK,OS). A megoldás előnye, hogy amennyiben a feladatok egyenlően osztódnak el az egyes unit-ok között, akkor kiküszöbölhető az erőforrásokért való versenyzés, ami a terhelés növekedtével egyre nagyobb overheadet jelent, a tényleges munkavégzés rovására.



Ábra 1: SMP és MPP architektúra összehasolítása

RDBMS-t több gyártó készít MPP architektúrában, pl. GreenPlum (EMC), Netezza (IBM), ill. Teradata. Ez utóbbival foglalkozunk a továbbiakban részletesen.

A Teradata RDBMS architektúrája

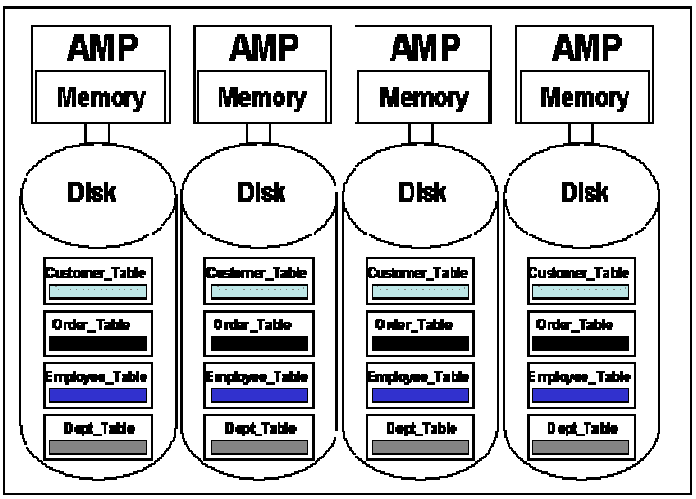


Figure 2: Teradata architektúra

Az AMP (Access Module Processor, virtuális processzor egység) dedikált memória területtel gazdálkodik, valamint dedikált fizikai diszkek sokaságát kezeli kizárólagosan. Ezt a felépítést nevezzük Share Nothing architektúrának.
Minden egyes adatbázis táblából egy szelet megtalálható minden AMP „alatt". Ezeket a felhasználók közvetlenül nem érhetik el, kizárólag az AMP szoftvere számára biztosított a hozzáférés. Minden egyes rekord esetében egyértelműen eldöntött, hogy melyik AMP „alá tartozik", azaz fizikailag melyiken keresztül érhető el, melyik képes műveleteket végezni rajta. Az egyértelmű leosztást (disztribúció) egy *hash függvény* biztosítja. Minden táblán ki van jelölve a tábla egy, vagy több mezője (pl. customer_id), mint a hash függvény bemenete. A kimenet pedig meghatározza, hogy melyik AMP-on helyezendő el/keresendő az adott rekord.

Az SQL nyelven megadott feladatok végrehajtása a következő séma szerint történik:

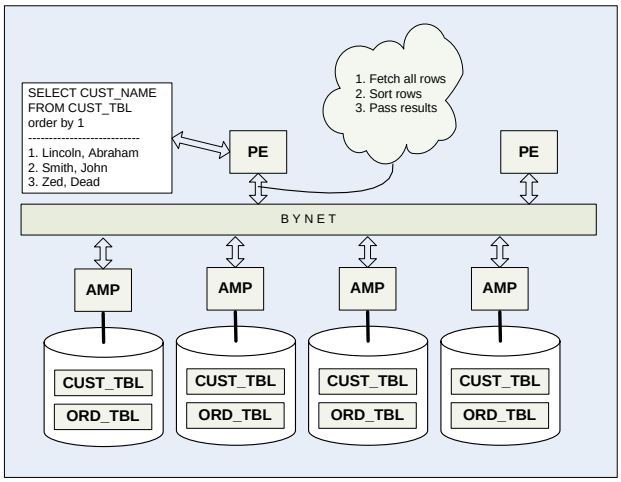


Figure 2: SQL végrehajtás Teradata-n

A felhasználói kapcsolatok (session) a Parsing Engine-ekhez (PE) kapcsolódnak. Egy adott rendszerben az AMP-ok száma fix, a PE-k száma a párhozamosan kapcsolódó session-ök számától függ, mivel egy PE korlátozott számú kapcsolatot képes kezelni.
A PE értelmezi az SQL utasítás szövegét, és a data dictionary segítségével (objektumok, statisztikák) egy optimális költségű végrehajtási tervet készít. Ezen tervet utána alacsony szintű lépésekre bontja, amely lépéseket, mint AMP parancsokat a BYNET közvetítésével eljuttatja az AMP-okhoz. Az AMP-

ok egymás közötti kommunikációja (pl. rekordok átmozgatása, redisztribúció) is a BYNET-en történik. Ha az összes lépés elvégeztetett, akkor az eredményt a BYNET-en a PE közvetítésével a rendszer visszaadja a felhasználónak.

Adattárház feladatelemek

Az adattárházak esetében tipikusak az olyan feladatok, SQL utasítások, aminek végrehajtási tervében azok az elemek szerepelnek, amiket a hagyományos RDBMS-ek esetében igyekszünk messziről kerülni. Ezen adattárház-speciális megoldások úgy épülnek fel, hogy ezen feladatelemek hatékonyan hajtódjanak végre, és ne kelljen kerülni őket lépten-nyomon.

Full table scan
Ez a művelet a tábla teljes tartalmát végigolvassa (indexek mellőzése). Akkor használatos, ha olyan összegzést, kalkulációt kell elvégezni, ahol sok/összes rekordra szükség van, vagy a végigolvasás költsége alacsonyabb, mint a több indexelt elérés eredményének összefésülése. MPP architektúrában a rekordok (elvileg) egyenletesen oszlanak el AMP (Teradata) alatt, így K AMP esetén 1/K-ad részére csökken a művelet időigénye a soros végrehajtáshoz képest.

Rendezés (Sort)
A rendezés N*Log(N) műveletigényű feladat, ha N a rendezendő halmaz mérete. A Merge-Sort algoritmus kifejezetten jól alkalmazható az MPP architektúrákon, mert rendelkezésre áll fixen K darab „rendező motor”. Nagy méretű adathalmazok esetén N >> K, a rendező motorok számának növelésével arányosan csökken a teljes rendezés időszükséglete.

Összekapcsolás (Join)
A rekordok tömeges join-ja sokféle módszerrel elvégezhető, ez általában a join-ban megadott feltételektől, a join-olandó táblák rekordszámától, ill. azok fizikai elhelyezésétől (MPP) függ. Egy A és B tábla közötti elemi join művelet (két rekord összekapcsolása) mindig úgy hajtódik végre, hogy az adott A rekord és B rekord megvizsgáltatik a megadott join feltétellel, és amennyiben ez teljesül, akkor az A-B rekordpár részesévé válik az eredményhalmaznak.
Értelemszerű, hogy egy MPP rendszeren akkor lehet hatékony egy elemi join, ha mind A, mind B rekord ugyanazon unit (AMP) alatt tartózkodik. Ezt többféleképpen el lehet érni. Például ha az egyik tábla nagyon kicsi, a másik pedig óriási (ez egy nagyon gyakori konstelláció), akkor a kicsi tába rekordjait minden AMP alá felmásolva máris teljesül, hogy tetszőleges A-B páros megtalálható lesz minden AMP alatt. Másik tipikus példa, amikor egyik tábla sem kicsi, így a rekord-multiplikálás nem hatékony. Megoldást jelent, ha a két tábla AMP-eloszlását úgy határozzuk meg, hogy azon A-B párosok, akik esélyesek arra, hogy a join feltételt kielégítsék mindig egy AMP alá kerüljenek. Például:

```
select A.TRX_ID,B.USER_NAME
from A
join B on A.USER_ID = B.USER_ID
```

Teradata-n a következő történhet: A tábla és B tábla hash-elése is a USER_ID mentén történik. Így egy tetszőleges USER_ID érték hash-e is ugyanazt az AMP-ot fogja meghatározni mind A, mind B táblára, azaz az összeillő rekordok garantáltan egy AMP alatt fognak tartózkodni.

Teradata aspektus

Transzparencia
A felhasználók felé a rendszer párhuzamossága gyakorlatilag rejtve marad. Az egyetlen MPP specifikus kiterjesztés, hogy minden tábla esetén meg kell adni az AMP-elosztás szabályát. Erre a „PRIMARY INDEX” fogalma került bevezetésre, ami némileg félrevezető,mert NEM INDEX! Nem is egy fizikai struktúra/objektum, hanem mindössze egy szabály, hogy mely mezők tartalma alapján kalkulálódik ki a rekord hash értéke, ami többek között meghatározza azt is, hogy a rekord melyik AMP alatt fog elhelyezkedni. Példa:

```
CREATE TABLE DEMO_TBL
```

```
(
  ID integer,
  A_PROP CHAR(1),
  B_PROP INTEGER
) PRIMARY INDEX (B_PROP)
;
```

Fontos, hogy a PRIMARY INDEX nem jelent Primary Key-t, és egyedinek sem kötelező lennie, mindössze az AMP-eloszlást fogja meghatározni (és a belső tárolással kapcsolatosan néhány egyéb következményt)
A többi SQL műveletben már az MPP-ség eltűnik, és pontosan úgy néznek ki az SQL utasítások, mint egy tetszőleges RDBMS esetében.

Tervezési irányelvek
Az adatbázis tervezés feladata több szempontból is eltér a nem MPP adatbázis kezelőknél alkalmazandókhoz képest. Már logikai adatmodell szintjén is szabadabban gondolkodhat a tervező, akár 3NF adatmodellt is alkalmazhat (szemben a csillag-hópehely sémákkal), mivel a join műveletek nagyon hatékonyan és gyorsan végezhetők. Fizikai adatmodell szintjén a fő MPP specifikus feladat a PRIMARY INDEX-ek meghatározása. Ezek meghatározásához a fogadandó adatokat és a várható workload-ot is figyelembe kell venni, de sokkal inkább fontosak az adatok, mint a worlkload, szemben a nem MPP RDBMS-ekkel, ahol szinte kizárólag a workload alapján tervezik a fizikai adatmodellt (indexek). A fő elkerülendő szituáció a „skewness", mikor vagy a tárolt adatok, vagy a processzálási feladat számottevően aszimmetrikusan osztódik el az AMP-ok között, ami az erőforrások nagymértékű pazarlását, és rossz válaszidőket fog okozni.
Teradata esetében relatíve ritkán használunk indexeket, akkor sem a klasszikus B-fa indexek az elterjedtek, hanem egyéb (indexnek nevezett), pl. alternatív disztribúciót, vagy horizontális/vertikális szűkítést megvalósító materializált nézeteket. Fontosak továbbá a (belső) statisztikák, amik a Teradata cost-based optimizer-ének szolgáltatják az alap információkat (rekordszám, egyedi értékek száma, hisztogram) az optimális végrehajtási tervek elkészítéséhez. Fontos ezen statisztikák megfelelő időközönkénti frissítése. A megfelelő sűrűség meghatározása is feladat, mert az elavult (stale) statisztika nagyon félrevezetheti az optimizert, a brute-force frissítés pedig rengeteg feldolgozó kapacitást pazarol.

Összefoglalás

Dióhéjban az MPP RDBMS-ek előnyei adattárház felhasználásra

- Olcsó SMP építőelemek használata. Nem szükséges fizikai számítógép minden processzáló unit számára, megtörhető a share-nothing szabály komolyabb hátrányok nélkül a processzálás szintjén
- SQL interface – a legleterjedtebb adatkezelési nyelv használható, kompatibilitás
- Optimális esetben 1/K (unitok száma) futásidő. Pl. full table scan
- Ténylegesen lineáris skálázhatóság. Teradata esetében >1000 AMP esetén bizonyítottan
- Dinamikus bővíthetőség. A kapacitás a növekvő igényeknek megfelelően folyamatosan bővíthető

Források

[1] http://www.coffingdw.com/Teradata_Basics/chapter_9___advanced_topics_that_you_will_be_tested_on/teradata_is_a_shared_nothing_architecture.htm
[2] http://encyclopedia2.thefreedictionary.com/MPP
[3] MPP RDBMS, Teradata, Greenplum, Big Data, Hadoop, MapReduce, Teradata Express
[4] http://scs.web.elte.hu/Work/DW/papers/sidlo_dipl.pdf
[5] http://www.cubrid.org/blog/dev-platform/platforms-for-big-data/

Teradata SandBox download:
http://www.teradata.com/teradata-express/
http://www.cubrid.org/blog/dev-platform/platforms-for-big-data/