

DIPLOMAMUNKA

Thurzó Máté István

**Debrecen
2008**

**Debreceni Egyetem
Informatikai Kar**

ADATTÁRHÁZAK ÉS ÜZLETI INTELLIGENCIA

Témavezető:
Dr. Juhász István
egyetemi adjunktus

Készítette:
Thurzó Máté István
programtervező matematikus

*Debrecen
2009*

Tartalomjegyzék

BEVEZETÉS.....	4
ADATTÁRHÁZAK.....	6
1. Adattárházak története, háttere	6
2. Adattárházakkal kapcsolatos fogalmak, és architektúra.....	7
2.1 Definíció	7
2.2 ETL.....	7
2.3 Adattárház architektúra.....	9
3. Adattárházakat támogató technológiák	12
3.1 Adattárházak és OLTP rendszerek különbségei.....	12
3.2 Adattárházban használt technológiák	13
3.2.1 Bittérkép indexek (Bitmap indexes)	13
3.2.2 Bitmap Join Index.....	20
3.2.3 Parallel futtatás (Parallel execution).....	21
3.2.4 Materializált nézetek.....	25
3.2.5 Partícionálás	31
ÜZLETI INTELLIGENCIA.....	36
1. Az üzleti intelligencia definíciója.....	36
2. Az üzleti intelligencia által megoldandó feladatok, tipikus problémák	37
3. Enterprise Performance Management (EPM).....	38
4. Oracle BI és DW architektúra	39
4.1 Oracle üzleti intelligencia csomagok: BI Suite Enterprise Edition és BI Suite SE One .	40
4.1.1 Nagyvállalati verzió (enterprise edition)	40
4.1.2 Standard verzió (BI Suite SE One).....	45
5. Oracle BI példa.....	45
5.1 Belépés az Interactive Dashboards-ba.....	46
5.2 Oracle Answers	47
5.3 Pivot.....	49

5.4 Grafikon hozzáadása.....	51
5.5 Megjelenítés a vezérlőpulton.....	52
5.6 Konklúzió	54
ÖSSZEFOGLALÁS	55

BEVEZETÉS

Az informatika tudományban a fejlődés egyik fő mozgatórugója mindig az volt, hogy a piac, illetve az ipar igényeire próbált az informatika válaszokat kínálni. Erre tökéletes példa az adattárházak fejlődése: az elmúlt évtizedekben a nagyvállalatok még inkább nagyok lettek; a növekedésükkel, és a piacok szélesedésével szükségessé vált, hogy hatékonyabban tudjanak a szervezet egészét érintő döntéseket hozni. A feladat megoldásához teljesen új adattárolási elgondolások, és eltérő adatkezelési filozófiák születtek meg, mely aztán kihívást teremtett az informatikai piac szereplőinek, hogy alkalmazzák ezeket az elgondolásokat. Az adatbáziskezelő gyártók számára a kihívást az jelentette, hogy ugyanazon motor szolgálja ki a már megszokott rendszereket, illetve, hogy képes legyen kezelni az új elgondolású igényeket is. Így születtek meg az adattárházak, és a vállalatok, szervezetek döntéshozatalához ezek adják az egyik legnagyobb alapot.

Elméleti és gyakorlati jelentősége egyaránt nagy az adattárház témakörnek. Az elméleti részben kiemelkedő fontosságú a már említett a hagyományos tranzakció alapú rendszerektől való eltérés, és az új koncepciók az adatkezelés terén. Gyakorlati jelentőségét pedig nem lehet eléggé hangsúlyozni, ma már nincsen olyan nagyvállalat ahol ne foglalkozna egy részleg a cég adattárházaival, illetve az üzleti döntéstámogatással.

Szakdolgozatom első részének célja betekintést nyújtani az adattárházak koncepciójába, az eltérő adattárház modellekbe, felhívni a figyelmet az eltérésekre egy általános adatbázis rendszer és egy adattárház között, valamint megismertetni az olvasót pár tipikusan adattárházban jellemző technológiával.

A dolgozat második felében az üzleti intelligenciát tárgyalom részletesebben, amely szervesen kapcsolódik az adattárházakhoz és az üzleti döntéshozáshoz. A BI (business intelligence) vagyis üzleti intelligencia szintén ékes példája az említett informatikai fejlődésnek amikor is az üzlet szabja az igényt az IT pedig válaszol. A fogalom régóta ismert ám csak az utóbbi évtizedben terjedt el szélesebb körben és ez idő alatt hatalmas fejlődésen ment át. Gyakorlati jelentősége a

jelen nehéz gazdasági helyzetben sem csökkent, hiszen most még inkább fontos lehet, hogy a rendelkezésre álló információkból helyes döntéseket hozzon egy szervezet vezetése.

A szakdolgozat második felében tehát áttekintjük mit értünk ma üzleti intelligencia alatt, hogyan kapcsolódik az adattárházhoz, valamint szándékomban áll bemutatni néhány üzleti intelligencia jellegű alkalmazást.

Szakdolgozatom, holott az elméleti részben tárgyaltak között általános dolgok is szerepelnek, elsősorban az Oracle adattárházakat kiszolgáló technológiáit valamint az Oracle által nyújtott üzleti intelligencia alkalmazásokat használja a tárgyalt témakörök szemléltetésére. Ennek az oka, hogy az Oracle termékpalettája az elmúlt években jelentősen nőtt az adattárház illetve BI jellegű eszközökkel, sok esetben neves BI eszközök felvásárlásával, amely az mutatja, hogy az adatbáziskezelésben kivívott előkelő pozíciója mellé igyekszik az adattárház illetve BI területen is kiemelkedő pozíciót szerezni. A dolgozat írása közben azonban igyekszem más gyártók, más technológiáit is bemutatni összehasonlítás jelleggel.

ADATTÁRHÁZAK

1. Adattárházak története, háttere

Az informatika fejlődése tette lehetővé a cégek számára, hogy adataikat, mindennapi működésükhöz szükséges információkat adatbázisokba rendezhessék. Eleinte költséges mainframe számítógépeken futottak ezek az adatbázisok, csak a nagyvállalatok engedehették meg maguknak. A reportokat (jelentéseket) több napba vagy hétbe tellett megírni, így nehézkes volt bármilyen kimutatást készíteni ezekből a rendszerekből. A hardver árak esése és a technológia fejlődése együttesen járult hozzá, hogy az adatbázisok terjedése felgyorsuljon, valamint az OLTP (Online transaction processing) rendszerek elterjedjenek. Az OLTP rendszerek tranzakció orientált alkalmazások, amelyek lehetővé teszik az adatok gyors bevitelét, könnyű módosíthatóságát. Tipikus OLTP alkalmazás például az értékesítésben használt megrendelések kezelése, vagy pénzügyterületen a főkönyvi funkciók. Az OLTP rendszerek lehetővé tették a vállalatok számára, hogy ne kelljen saját alkalmazásokat írni adataik gyors kezelésére, hanem nagy és flexibilis OLTP rendszereket szabtak saját cégüknek megfelelő formára.

Az OLTP alkalmazások segítették ugyan az adatok hatékony kezelését, azonban a kihívás egy idő után a felgyülemlett adatok értelmezése lett, miként tudnak az üzlet számára értékelhető információt kinyerni. Miként lehet a hatalmas mennyiségű adatot konszolidáltan kezelni és miként lehet kérdésekre választ kapni ebből az adathalmazból.

Adatnak nevezhetünk egy nem értelmezett szimbólumot, jelentés nélküli szintaktikai fogalmat. Egy adatbázis számtalan adatot tartalmaz, megrendelési számokat, pénzmennyiségeket, életkorokat, címeket, azonban ezek önmagukban általában nem sok információtartalommal bírnak. Az adattárházak, és általában véve az üzleti intelligencia ezekből az adatokból hivatott információt kinyerni, ami az értelmezett adat, szemantikai fogalom. Az információt felhasználva pedig a vállalat vezetése döntéseket hozhat. Az adattárházak tehát döntés támogatást segítenek, segít feltárni a trendeket egy vállalat működése mögött, használni lehet előrejelzéshez egy adott termék tervezése közben, és a lista szinte végtelen. Azokat a rendszereket amelyek segítik ezeket a tevékenységeket nevezzük OLAP (Online analytical processing) rendszereknek.

2. Adattárházakkal kapcsolatos fogalmak, és architektúra

2.1 Definíció

Mit is nevezünk tehát adattárháznak? Egy olyan általános adatbázis, amely már konszolidált, integrált, aggregált, struktúrált, többféle forrásból származó adatokat tartalmaz, amely adatbázis ezáltal támogatja az üzleti folyamatok analizálását, és a döntéshozást.

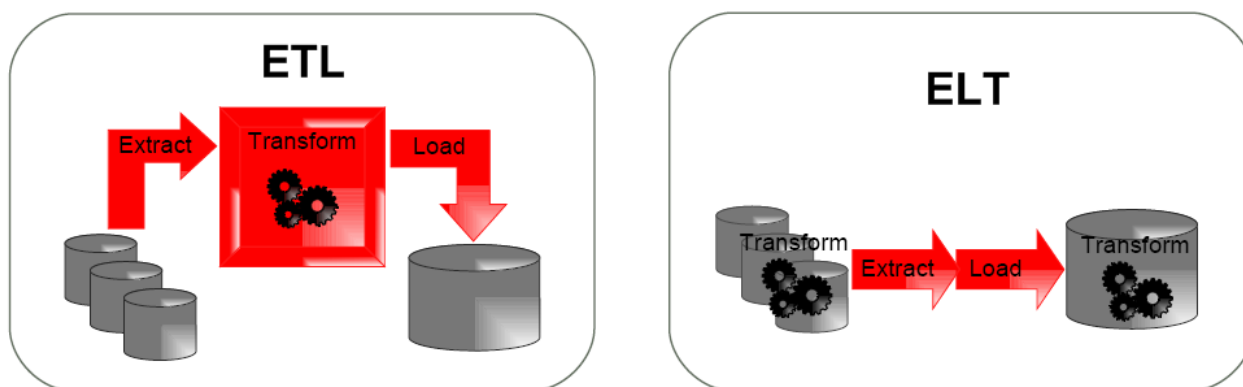
Egy másik meghatározás, a gyakran idézett Bill Inmon féle definíció szerint „Az adattárház tárgyorientált, integrált, tartós és időfüggő adatgyűjtemény a vezetői döntéstámogatás szolgálatában”.

2.2 ETL

A forrásrendszerek homogenitása nem garantált, egy adattárház többféle típusú rendszerből képes adatot fogadni, hiszen a definíció szerint az adatok először átesnek egy transzformációs folyamaton. A forrásrendszerekből az adattárházba vezető út első lépése az adatok kinyerése (Extract). Ez többnyire valamilyen relációs adatbázisokból történik, de forrásrendszer lehet akár egy web-es gyűjtőprogram is. Az adatok kinyerésére általában valamilyen szűk időablak áll rendelkezésre, mivel a folyamat során nem cél sokáig terhelni a forrásrendszereket. Napi betöltés esetén például általában éjszaka lehet egy-két óra alatt kinyerni az adatokat. Különösen szűk az idő, hogyha nemzetközi szervezettel van dolgunk, hiszen az időeltolódás miatt a rendszerek szinte állandó terhelés alatt vannak. Ebből az okból kifolyólag gyakran, egy átmeneti helyre töltik be először az adathalmazt.

A kinyert adatok ezután egy alakításon (Transform) mennek át, közös formára kell hozni őket. Például egy ember nemét jelölheti 'F' és 'N' vagy 'ferfi' és 'no', azonban az adattárházban már csak az egyiket hagyjuk meg, vagy egy új értéket vezetünk be a nemek jelölésére. Ez a lépés egyfajta törzsadat létrehozás, hogy különböző rendszerekből származó fogalmak az adattárházban már ugyanazt jelentsék. A transzformálás egészét vagy egy részét igen gyakran egy a forrásrendszer és az adattárház között elhelyezhető, a már elmített átmeneti adatbázisban (Staging Area) hajtják végre. A transzformálás lépése azonban nem csak a duplikált értékek kiszűrése, vagy a közös nevezőre hozás, hanem ebben a lépésben történhet valamilyen szűrés, rendezés is, vagy egy pontosvesszővel elválasztott értéksorozat több oszlopra való bontása.

Utolsó lépésként a cél adattárházba való betöltés következik (Load). Az utolsó lépésnek is fontos tulajdonsága a heterogenitás, mindeféle célrendszerbe tölthetünk be adatokat. Az imént leírt folyamat a fázisok angol nevükből alkotott betűszóval ETL folyamatnak nevezzük. Létezik olyan felfogás is azonban, és manapság már ez a népszerűbb, ami felcseréli a transzformálás és a betöltés lépéseket, és az adattárházba betöltés után alakítja át az adatokat. Ebben az esetben ELT folyamatról beszélhetünk.



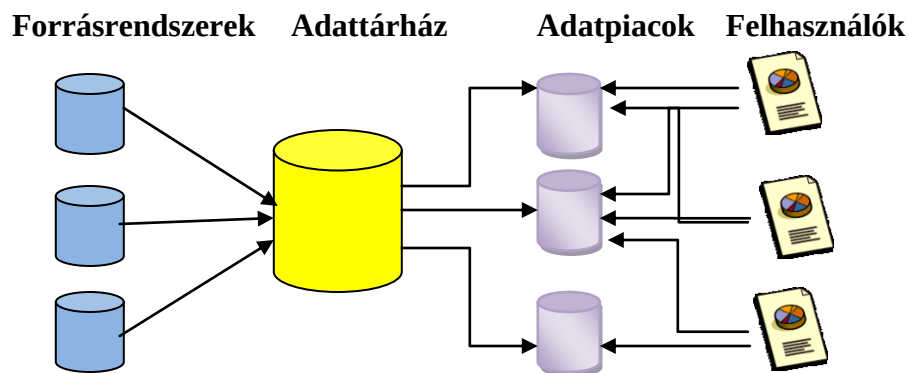
1. ábra ETL és ELT

Egy általános adatbetöltési folyamat során jogos igény lehet, hogyha nem minden adatot akarunk betölteni mindig az adattárházba, hanem csak a megváltozott adatokat. Ehhez természetesen valamilyen módon követnünk kell a változásokat a forrásrendszerben. Ehhez a technológiát a forrásrendszer és az ETL eszköz együttesen nyújtják.

ETL eszközök választásakor számtalan lehetőségünk adódik. Fejleszthetünk akár saját magunk is megfelelő programcsomagot amit a szervezetünk igényeire van szabva, bár itt mindenképpen számolnunk kell a fejlesztési, és karbantartási szükségletekkel valamint más a szoftverfejlesztés során előforduló faktorokkal. Másik lehetőségként valamilyen szoftver szolgáltatásait vesszük igénybe: az Oracle a Data Integrator nevű programját kínálja erre a feladatra vagy az Oracle Warehouse Bulider-t, míg a Microsoft termékek esetén az SQL Serverhez az SQL Server Integration Services szoftverét használhatjuk.

2.3 Adattárház architektúra

Egy szervezeten belül több adattárház lehet. Az architektúra kialakítása során figyelembe kell venni az adatok mennyiségét, a forrásrendszerek jellegét, és egyéb más meghatározó faktorokat. Nincsen jó vagy rossz architektúra, mindig a vállalati rendszer felépítése dönti el az adattárház architektúra jellegét is.

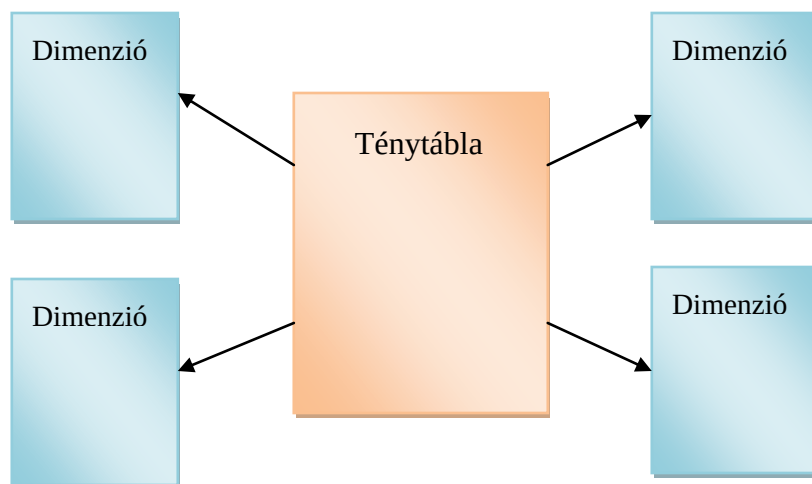


2. ábra Egy általános adattárház architektúra

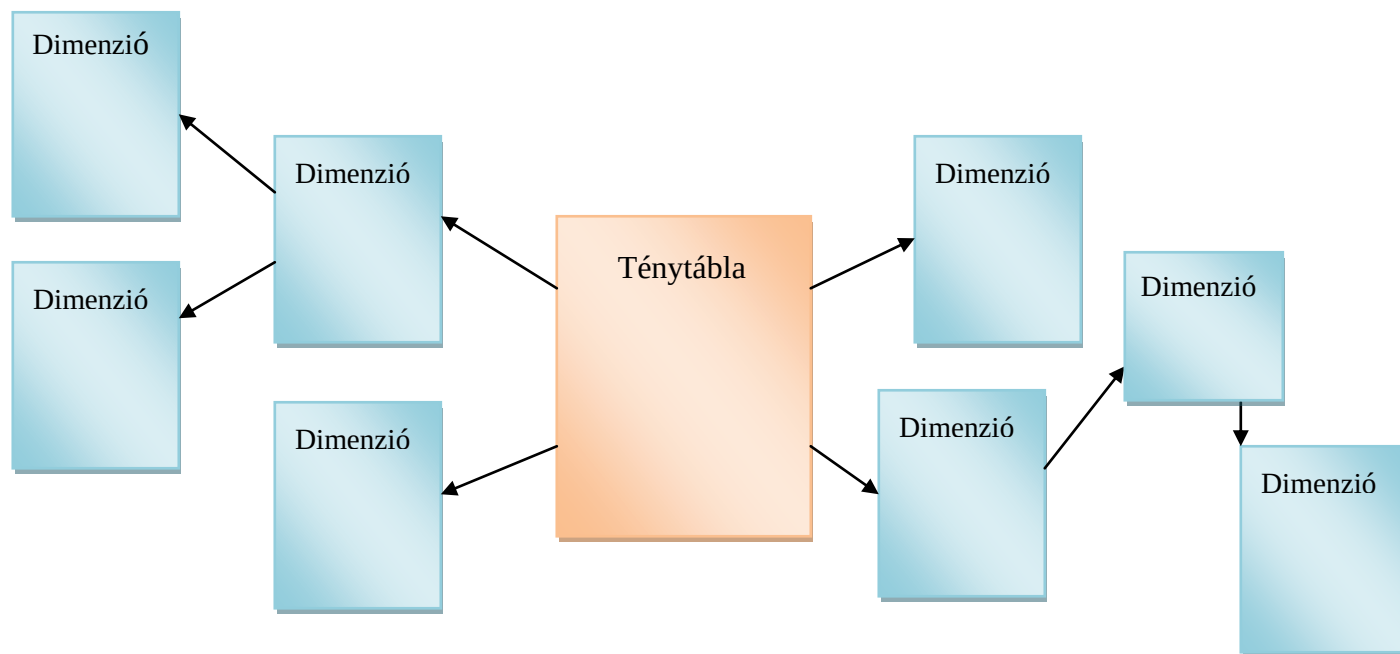
Egy általános adattárház (Data Warehouse, ill röviden DW) architektúra azonban nem csak forrásrendszerekből (Source Systems) áll, beszélhetünk még adatpiacokról (Data Mart) is. Az adatpiacok fogalma szorosan kapcsolódik az adattárházak fogalmához, általában kisebb méretű, specifikus feladatokat ellátó adattárház megoldások, vagy egy adott adattípus, téma köré csoportosul. Az adatpiacok többnyire az adattárházból képzett adatokat tartalmaznak.

Az adattárházak és adatpiacok alapjául szolgáló adatbázisokban többféle modell szerint tárolhatjuk az adatokat. Az egyik lehetőség a hagyományos adatbázisokban megszokott Egyed-Kapcsolat modell (Entity-Relationship vagy ER modell), ahol az adatok harmadik normálformában vannak. Egy másik igen elterjedt modell egy dimenzionális modell az úgynevezett csillag séma. A csillag séma alapját a tény tábla vagy táblák alkotják, ezek tartalmazzák a fő adatokat. A tény táblák további táblákhoz, úgynevezett dimenziótáblákhoz kapcsolódnak, amelyek „kifejtik” az adott ténytábla tartalmát az adott dimenzió mentén. A dimenziótáblák jelentik az adattárház definícióban a tárgyorientáltságot, ezen témakörök mentén tehetem fel a kérdéseimet.

A csillagséma dimenziótáblái denormalizáltak, ennek megszüntetésével átalakítható hópehely sémává. A két séma szorosan kapcsolódik egymáshoz, ahogyan az ábrákon is látszanak.

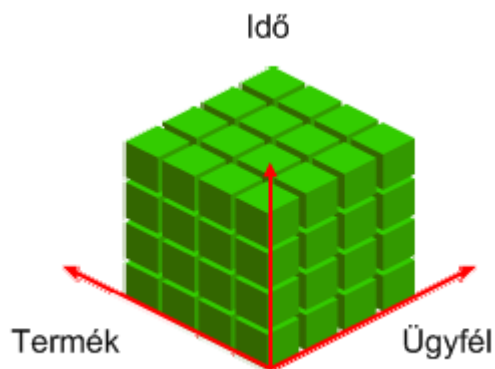


3. ábra Csillag séma



4. ábra Hópehely séma

A csillag illetve a hópehely sémához szorosan kötődik a multidimenzionális kocka, vagy OLAP kocka fogalma. Az OLAP kockát a relációs adatbázisok limitációinak leküzdésére találták ki, és gyorsabban lehet vele analízisokat, lekérdezéseket futtatni. Egy kockát általában a csillag vagy hópehely sémából származtatnak. Egy kocka tartalmaz számszerű tényeket úgynevezett mértékeket (measure) amelyek dimenziók szerint vannak kategorizálva. A mérték a ténytábla rekordjaiból származtatódik, a dimenziók pedig a dimenzió táblákból.



5. ábra Értékesítés OLAP kocka

Az ábrán látható példán 3 dimenzió látható, azonban egy kocka magában foglalhat több dimenziót is, így szokás a hiperkocka elnevezést is használni. Az Oracle a kockákat egy úgynevezett analitikus munkaterületen (analytic workspace) tárolja, ami fizikailag egy relációs tábla egy Binary LOB oszlopát jelenti.

Az eltérő adatszerkezethez eltérő műveletek is társulnak, tekintsük át az OLAP kocka tipikus műveleteit.

- Fúrás fel/le (Drill Down/Up): a kevésbé részletes adatok felől navigálunk a nagyobb részletezettségűek felé
- Felgöngyölítés (Roll-up): Egy adott dimenzió mentén kiszámolunk minden adatkapcsolatot
- Vágás (Slice): Adott értéktől függő részhalmazát adja meg egy multidimenziós tömbnek

3. Adattárházakat támogató technológiák

3.1 Adattárházak és OLTP rendszerek különbségei

Az adattárházakban tárolt adatok jellege, és az OLTP rendszerekhez képest eltérő tárolás miatt, valamint a két rendszerbeli különbségek miatt, ahhoz hogy hatékonyan tudjunk lekérdezéseket vagy riportokat futtatni, egészen más technológiákat kell alkalmazni egyikben és másikban.

Általában véve elmondható hogy az összes adatbáziskezelő (beleértve természetesen az Oracle termékét is) szoftverről, hogy eleinte csak OLTP rendszerek építésére használták őket, az adattárház később kezdett teret nyerni magának, és így kerültek beépítésre az adattárházakat támogató adatbázis technológiák is.

	OLTP	DW
Táblaösszekapcsolások (Join) száma	Sok	Viszonylag kevés
Lekérdezésekkel elért adatmennyiség	Általában kevesebb	Nagy
Adatok normalizáltsága	Normalizált	Nem normalizált
Származtatott adatok és aggregátumok	Ritka	Gyakori

6. ábra OLTP és DW rendszerek összehasonlítása

Adattárházakban viszonylag kevés táblát kell összekapcsolnunk egy átlagos OLTP rendszerhez viszonyítva. Például ha egy adott vevőhöz le akarjuk kérdezni az utolsó rendelést, akkor számba kell venni a vevőket tartalmazó táblákat, a rendeléseket, a rendelés részleteit, raktárakat, az áru egyedítípust tartalmazó táblát, vevő kapcsolattartó táblát, és így tovább. Míg egy adattárház környezetben a tény tábla tartalmazza a lényegi információt, utána csak a dimenziótáblákat kell hozzákapcsolnunk, ha éppen szükség van rá.

Az egyik legfontosabb különbség az adattárház és egy OLTP rendszer között, a lekérdezésekkel megfogott adatok mennyisége. Egy DW rendszerben bár a végeredmény elképzelhető, hogy csak egy összeg, vagy pár mérőszám, ahhoz hogy ezt megkapjuk az OLTP rendszerhez képest jóval nagyobb mennyiségű adatot kell bejárni.

Megszokott dolog, hogy egy OLTP jellegű adatbázisban normalizáltan tároljuk le az adatokat, nem tárolunk kiszámítható értékeket, ne tároljuk ugyanazt az értéket többször. Ezzel szemben az

adattárházakban tapasztalható némi denormalizáció, ez egy szükséges rossz a gyors lekérdezések érdekében.

Érdemes szót ejteni a DML többi utasításáról is (a SELECT utasítás félig meddig kilóg a DML kategóriából). Míg egy OLTP rendszer átlagos napi működése közben a felhasználók, új adatokat rögzítenek be, új megrendeléseket, új ügyfeleket. Módosíthatják a már meglévőket, az újonnan megszerzett adatokkal, például tervez-e az ügyfél mást is vásárolni a közeljövőben. Törölhetik is a meglévőket, bár a tendencia inkább az hogy többnyire nem törölnek egy OLTP rendszerből. Így a SELECT-ek mellett legalább akkora súllyal vannak jelen az INSERT, UPDATE és DELETE utasítások.

Ezzel szemben az adattárházakban egyértelmű a SELECT dominanciája, ezen kívül még a beszúrás művelete gyakori ellenben módosítás vagy törlés szinte soha nem fordul elő.

3.2 Adattárházban használt technológiák

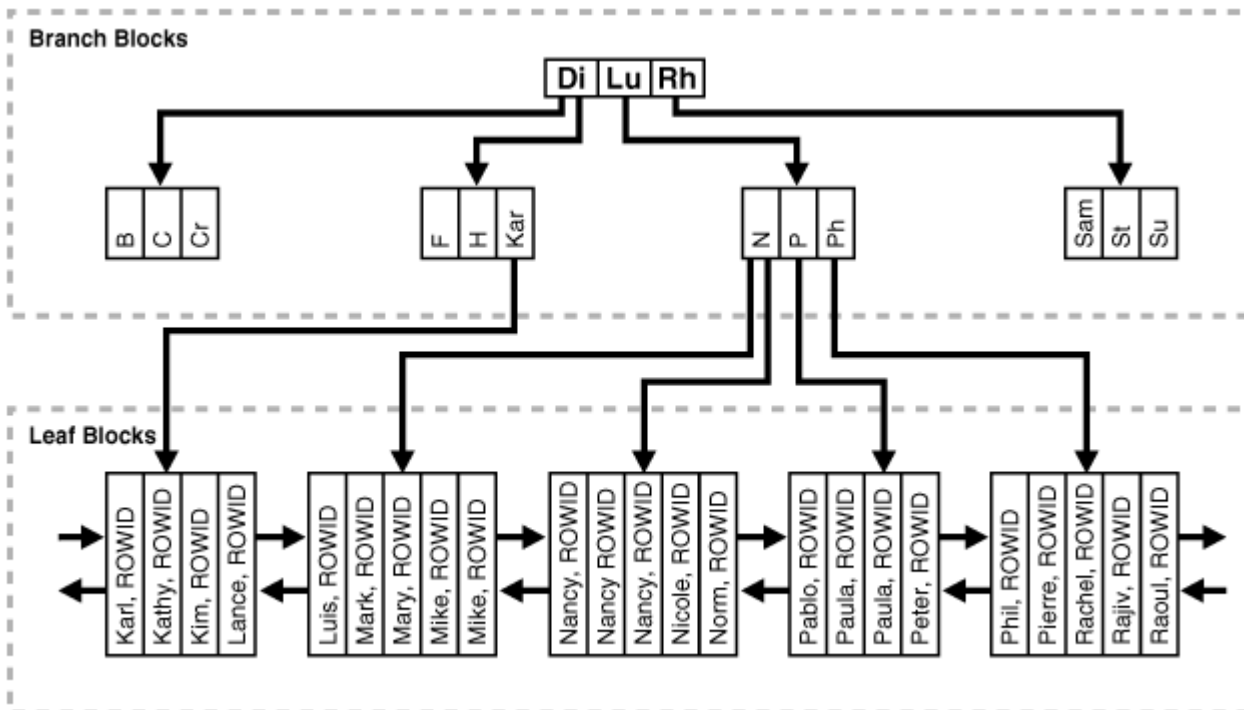
Tekintsük át tehát milyen technológiák hasznosíthatók DW környezetben.

3.2.1 Bittérkép indexek (Bitmap indexes)

A bittérképes index haszna igazán olyan oszlopoknál mutatkozik meg amelyekben kevés a különböző érték, viszont ezekből sok fordul elő, és az értékek egyenletes eloszlást mutatnak. Ilyen lehet például ha színeket tárolunk egy oszlopban, vagy akár személyek neveit. Felmerülhet a kérdés hogy miért nem hatékony ilyen esetekben a hagyományos B-fa index. Ahhoz, hogy ezt a kérdést meg tudjuk válaszolni ismernünk kell az Oracle B-fa index szerkezetét: a fa felső részében vannak az úgynevezett ág-blokkok (branch blocks), a fa alján vannak a levélelemek (leaf blocks). Indexben történő kereséskor először az ág blokkok segítségével tudjuk megfogni a levélelemek egy csoportját, amik kétirányban össze vannak láncolva, így a tényleges keresett értéket a láncolt listában való léptetéssel találhatjuk meg. A levélelemekben nem csak az index kulcsa tárolódik el, hanem az ún. ROWID vagyis egy sorazonosító is. Ha egy egyedi értéket keresünk az indexben azt a végrehajtási lépést idegen szóval index unique scan-nek nevezzük, vagyis index egyedi keresésnek. Ha azonban több értéket szeretnénk visszakapni, és adattárházban ez a gyakoribb, index range scan-ről beszélhetünk, vagy index tartomány

keresésről. A keresett értéket megtalálása után a futtató motor a rowid alapján nyúl vissza a táblába és kéri el az adott sort onnan. Ez azt jelenti, hogy a keresett értékekhez mindig fel kell olvasni a rowid-t.

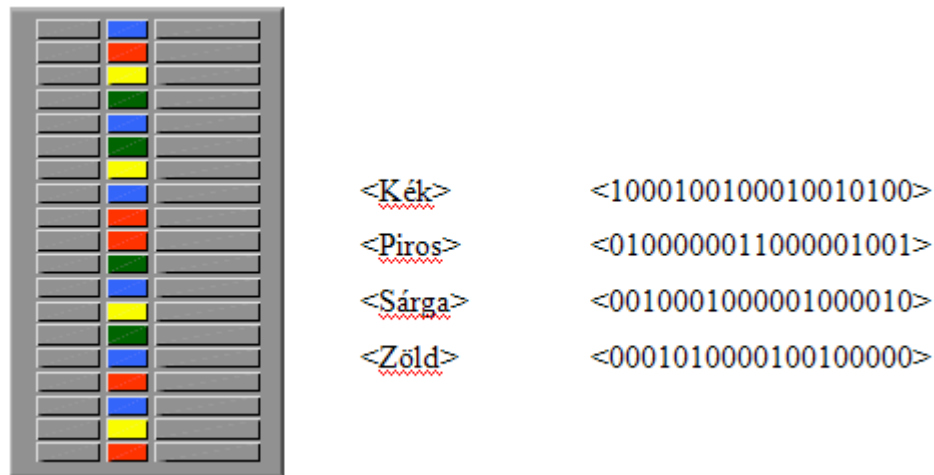
Ha azonban egy indexelt oszlopban egyenletes eloszlású kevés egyedi érték van, akkor a levélelemek csoportjai nagyra duzzadnak, sok sorazonosítót olvas fel a rendszer egy utasítás futtatása közben. E mellé társul az is hogy előállhat olyan indexszerkezet, blokkmérettől függően, valamint attól függően hogy hány sor fér bele egy blokkba, hogy egy blokkot többször is meg kell fogni ahhoz hogy elérjünk minden általunk keresett sorazonosítót. Ez értelemszerűen több lemezműveletet igényel.



7. ábra B-fa index szerkezete

Tekintsük át, hogy segíthet a fentebb leírt szituáción a bitmap index. Egy bittérképes indexben bitvektorok helyezkednek el, például ha az oszlop színeket tartalmaz akkor annyi bitvektor ahány szín van. Minden egyes bitvektorban annyi bit ahány sora van a táblának. Ha a tábla első sorában piros érték szerepel a szín oszlopban akkor a piros bitvektorban az első bit értéke 1-es lesz míg a többi szín bitvektorának első bitje 0-ás. Ha a második sorban kék szín szerepel a szín oszlopban

akkor a kék bitvektor második bitje 1-es lesz míg a többi vektor második bitja nulla, és ezt így lehet folytatni az egész táblára. Az ilyen indexek rendkívül jól tömöríthetők, egyértelműen kisebb a mérete egy bitmap-nek mint egy egész faszerkezetnek. Indexben való kereséskor nincs szükség teljes rowid-k felolvasására, egy-egy bit vizsgálatával könnyedén el lehet dönteni, hogy az éppen vizsgált sor beleesik-e a keresett tartományba avagy nem.



8. ábra Bittérképes index szerkezete

Több bitmap index egyidejű vizsgálatát segítheti, hogy köztük bitenkénti logika műveleteket tudunk végrehajtani.

Ami miatt a bittérképes index nem használható hatékonyan OLTP környezetben, az a zárolási tulajdonsága. Valamilyen DML művelet során nem zárolhatunk csak egy bitmap pozíciót az indexben, a legkisebb zárolási egysége a bittérkép szegmens, ami egy adatblokk fele is lehet akár a méretét tekintve. Egy OLTP rendszerben ahol állandó jelleggel vannak jelen DML utasítások, ez nagy hátrányt jelent.

Többször említésre került a fejezet tárgyalása során, hogy egyenletes eloszlásúnak kell lennie az oszlopban előforduló értékeknek, ahhoz hogy a bittérképes indexet megérje használni. Ennek szemléltetésére, tekintsük a következő példát: adott egy státusz oszlop amiben 0-tól 7-ig állhatnak egész számok, amik státusz kódok. A 0-ás az éppen beérkezett kérést jelenti míg a 7-es már a feldolgozottat. Egy rendszer normális működése során feltételezhetjük, hogy 7-esből rengeteg

van, míg nullás kódból éppen csak pár darab. Ilyen esetben ha a nullás kódú kérésekre szeretnénk lekérdezéseket futtatni, bittérképes index helyett még jó szolgálatot tehet a B-fa is, hiszen kevés olyan blokkot kell olvasni az indexből amelyek a nullás kódhoz tartoznak.

Az Oracle adatbáziskezelőjében bitmap index létrehozásakor külön jelezni kell a BITMAP kulcsszóval a létrehozni kívánt index jellegét.

```
CREATE BITMAP INDEX  
dw.cars_colors_bidx  
ON dw.cars(car_color);
```

Tekintsünk át a következőkben egy példát ami szemléletti Oracle adatbáziskezelőben a b-fa és a bitmap indexek közötti különbséget!

Hozzunk létre egy táblát amibe szűrjünk be adatokat, majd a test_flag oszlopára hozzunk létre egy b-fa indexet!

```
CREATE TABLE bitmap_idx_test (  
  id NUMBER,  
  test_flag VARCHAR2(1)  
) STORAGE(initial 256k next 256k);  
CREATE INDEX b_tree_idx ON bitmap_idx_test(test_flag) storage(initial 256k next  
256k maxextents 200);  
INSERT INTO bitmap_idx_test  
SELECT round(dbms_random.value(1, 5000)),  
case  
  when mod(level, 2) = 0 then 'U'  
  when mod(level, 3) = 0 then 'F'  
  else 'M'  
end case  
FROM dual  
CONNECT BY level <= 686038;
```

A jobb teszt érdekében a sorok beszúrásakor érdemes lehet nem egyszerre beszúrni a majd 700 ezer oszlopot, hanem a test_flag értékeket befolyásoló case szerkezetben változtatni az U, F, és M karaktereket százezres viszonylatban.

Ha ezzel megvagyunk, kérdezzük le az index méretét majd az autotrace opció bekapcsolásával futtasunk le egy az oszlopra szűrő lekérdezést.

```
set echo on
```

```
SELECT segment_name, bytes/1024 kilobytes, blocks
FROM dba_segments
WHERE segment_name = 'B_TREE_IDX';
```

SEGMENT_NAME	KILOBYTES	BLOCKS
-----	-----	-----
B_TREE_IDX	28672	3584

```
set autotrace on
```

```
SELECT /*+index(bit b_tree_idx) */ *
FROM bitmap_idx_test bit
WHERE test_flag = 'F';
686038 rows selected.
```

```
Execution Plan
```

```
-----
 0   SELECT STATEMENT Optimizer Mode=CHOOSE (Cost=531 Card=1 K
Bytes=19 K)
 1  0   TABLE ACCESS BY INDEX ROWID APPS.BITMAP_IDX_TEST (Cost=531
Card=1 K Bytes=19 K)
 2  1   INDEX RANGE SCAN APPS.B_TREE_IDX (Cost=1207 Card=522)
```

Statistics

```
-----
0 recursive calls
0 db block gets
6917 consistent gets
3890 physical reads
0 redo size
0 Parallel operations downgraded 50 to 75 pct
0 Parallel operations downgraded 25 to 50 pct
0 Parallel operations downgraded 1 to 25 pct
0 PX remote messages rcv'd
1366394 buffer is pinned count
686038 rows processed
```

Most dobjuk el az indexet és hozzunk létre egy bitmap indexet, majd kérdezzük le ismét az index méretét.

```
DROP INDEX b_tree_idx;
```

```
CREATE BITMAP INDEX bitmap_idx ON bitmap_idx_test(test_flag) storage(initial 256k
next 256k maxextents 200);
```

```
SELECT segment_name, bytes/1024 kilobytes, blocks
FROM dba_segments
WHERE segment_name = 'B_TREE_IDX';
```

SEGMENT_NAME	KILOBYTES	BLOCKS
-----	-----	-----
B_TREE_IDX	1024	128

Az első különbség a két indextípus között máris észrevehető: a bitmap index jóval, majd 27 megabyte-tal kisebb, és 3584 blokk helyett csak 128 blokkot foglal.

Futtassuk le ismét a teszt lekérdezésünket, és nézzük meg milyen eredményt kapunk.

```
SELECT /*+index(bit b_tree_idx) */ *
```

```
FROM bitmap_idx_test bit
```

```
WHERE test_flag = 'F';
```

```
686038 rows selected.
```

Execution Plan

```
-----  
0    SELECT STATEMENT Optimizer Mode=CHOOSE (Cost=426 Card=2 K  
Bytes=35 K)
```

```
1 0    TABLE ACCESS BY INDEX ROWID APPS.BITMAP_IDX_TEST (Cost=426  
Card=2 K Bytes=35 K)
```

```
2 1    BITMAP CONVERSION TO ROWIDS
```

```
3 2    BITMAP INDEX SINGLE VALUE APPS.B_TREE_IDX
```

Statistics

```
-----  
34 recursive calls
```

```
0 db block gets
```

```
4409 consistent gets
```

```
3016 physical reads
```

```
0 redo size
```

```
0 Parallel operations downgraded 50 to 75 pct
```

```
0 Parallel operations downgraded 25 to 50 pct
```

```
0 Parallel operations downgraded 1 to 25 pct
```

```
0 PX remote messages recv'd
```

```
1366394 buffer is pinned count
```

```
686038 rows processed
```

Látható, hogy a consistent get-ek száma, valamint a physical read-ek száma is kevesebb lett. A consistent get a memóriából törénő blokkolvasásokat jelenti, míg a physical read a lemezzől történő tényleges olvasást. A bitmap index esetében mind a kettő érték kisebb, így ebben az esetben valóban hasznos volt a használata.

A Microsoft SQL Server termékében nem található hasonló szerkezetű index, mint az Oracle-féle bittérképes index. Az SQL Server a lekérdezés optimalizálóra bízta az alacsony kardinalitású oszlopok hatékony kezelését. Az optimalizáló által előállított végrehajtási tervben bittérkép szűrőként (bitmap filter) ismerhetjük fel az ilyen futtatási lépéseket.

3.2.2 Bitmap Join Index

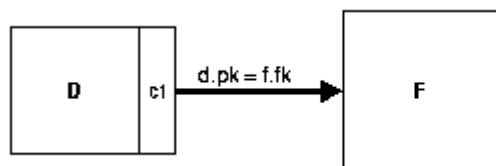
A join index (összekapcsoló index) olyan index, amely az alapjául szolgáló táblán kívül, egy vagy több más táblában lévő oszlopot is magába foglal egy összekapcsoláson keresztül.

A bitmap join index legegyszerűbb formájában egy normális bitmap index az F táblán, és magába foglal oszlopokat a $D_1..D_n$ táblákból, ahol a D_i tábla csillag- vagy hópehely sémában kapcsolódik az F táblához. Egy adattárház környezetben az F tábla tipikusan a tény tábla, a D_i táblák pedig a dimenzió táblák, az összekapcsolás pedig a dimenziótáblák elsődleges kulcsai és a ténytábla külső kulcsai között létezik.

Az adat mennyisége, amiket muszáj egy lekérdezés érdekében összekapcsolni, csökkenthető bitmap join indexek használatával, mert ezek már előre ki vannak számolva. Szintén hasznos lehet egy olyan join index amely több dimenzió táblát foglal magába, mivel ezzel elkerülhetjük a bitenkénti műveletek futását.

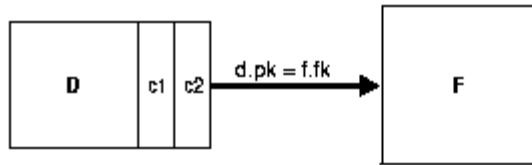
Bitmap join indexeknél négy különböző összekapcsolási modellt különböztetünk meg.

1. Egy dimenzió tábla oszlop kapcsolódik a ténytáblához



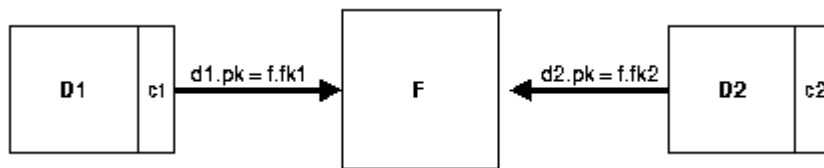
```
CREATE BITMAP INDEX bji ON f (d.c1) FROM f, d WHERE
d.pk = f.fk;
```

2. Kettő vagy több dimenzió tábla oszlop kapcsolódik a ténytáblához



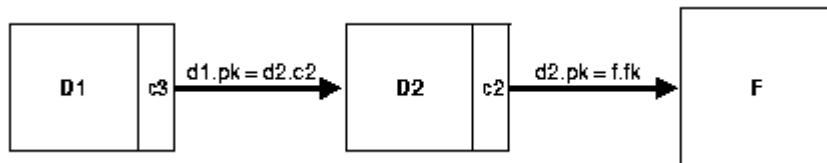
```
CREATE BITMAP INDEX bji ON f (d.c1, d.c2) FROM F, d
WHERE d.pk = f.fk;
```

3. Több dimenzió tábla oszlopai kapcsolódnak a ténytáblához



```
CREATE BITMAP INDEX bji ON f (d1.c1, d2.c2) FROM f, d1,
d2
WHERE 1.pk = f.fk1 AND d2.pk = f.fk2;
```

4. Hópehely séma



```
CREATE BITMAP INDEX bji ON f (d1.c3) FROM f, d1, d2
WHERE d1.pk = d2.c2 AND d2.pk = f.fk;
```

3.2.3 Parallel futtatás (Parallel execution)

A parallel futtatás egyszerűen kifejezve azon az ötleten alapszik, hogy szétszedjük a folyamatot több részre, és ahelyett hogy egyetlen rendszerfolyamat kezelne minden munkát egy lekérdezés kapcsán, több szál is ugyanazon a lekérdezés egy egy részét dolgozza fel. Példa lehet az ilyen futtatásra ha négy folyamat dolgoz fel különböző negyedéveket, egyetlen folyamat helyett. Ez az

eljárás jelentősen tudja javítani az adatmennyiség szempontból intenzív lekérdezéseket, amelyek tipikusan előfordulnak egy adattárház környezetben.

A parallel futtatás azonban nem minden esetben hoz drámai változást, teljesülnie kell bizonyos peremfeltételeknek ahhoz, hogy érdemi eredményt kapjunk. A technológiát a következő olyan rendszerekben érdemes használni amit a következő jellemzőkkel bírnak:

- A rendszer fürtözött, vagy SMPs (szimmetrikus multiprocesszoros)
- Elegendő a be és kimeneti sávszélesség
- Kihasztnálatlan processzorok vannak a rendszerben. Ha a kihasználtság mértéke 30% alatti akkor beszélhetünk általában kihasztnálatlan processzorról.
- Elegendő memória van a rendszerben a megnövekedett memóriaszükséglet kiszolgálásához. Az extra memóriaigényt a rendezések, hash műveletek, stb. adják.

A következő fontos kérdés a párhuzamosítást illetően az, hogy milyen műveleteket lehet parallelizálni:

- Hozzáférési műveleteket, például table scan, index full scan, partícionált index range scan
- Összekapcsolási műveleteket, úgy mint nested loop, sort merge, stb.
- DDL utasítások: CREATE TABLE AS SELECT utasítást, amikor egy lekérdezés alapján hozunk létre egy táblát
- DML utasítások: INSERT AS SELECT esetén, mikor egy lekérdezés alapján szúrunk be a táblába, valamint UPDATE, DELETE, MERGE műveleteket
- Egyéb SQL műveleteket, mint például a GROUP BY utasításrészt
- Lekérdezéseket: lehetőség van parallelizálni lekérdezéseket, és allekérdezéseket a SELECT utasításokban, valamint a lekérdező részeket DDL, vagy DML utasításokban

Most, hogy áttekintettük mi szükséges ahhoz hogy a parallel futtatás hatékony legyen, nézzük meg hogyan is működik.

Korábban már volt szó róla, hogy a munkát ami egy SQL utasítás végrehajtását jelenti több részre bontja a rendszer, és mindegy egyes részfeladatot más-más processz futtat. A beolvasott adat

(táblák, indexek, partíciók) is több kisebb részekre bonthatóak úgynevezett szemcsékre, vagy idegen szóval granule-okra. A granule-ok az elemi egységei a párhuzamos feldolgozásnak.

A felhasználói árnyékprocessz ami az utasítást futtatni kívánja, egyúttal a parallel futtatás koordinátora is lesz (query coordinator vagy parallel execution coordinator). A koordinátor feladata elemezni az utasítást, és meghatározni a parallelizáltság fokát, ezután lefoglalja a szolgálakat vagy processzeket. (A parallelizáltság fokának az egy utasításhoz társított az adott adatbázis példány futtató szervereinek számát nevezzük.) Ezen kívül az utasítás végrehajtásának vezérlése is a koordinátor feladatkörébe tartozik, amit a szolgálknak küldött utasításokkal tesz meg, valamint eldönti, hogy melyik bemenő adatot melyik szolga dolgozza fel. Végül de nem utolsó sorban a koordinátor fogja az eredményt szolgáltatni a felhasználó felé.

Amikor egy adatbázis példány elindul, az Oracle létrehoz egy tárolóhelyet (pool) azokból a futtató szerverekből amelyek, képesek párhuzamos futást kezelni. A PARALLEL_MIN_SERVERS inicializációs paraméter mondja meg a példány számára, hogy hány ilyen szervert kell létrehozni. Parallel utasítás futtatásakor a futtató koordinátor ehhez a pool-hoz fog fordulni, ha futtató szerverre van szüksége. Ha elfogytak a szerverek azonban szükség lenne többre akkor a rendszer újakat hozhat létre. A szerverek maximum száma a PARALLEL_MAX_SERVERS inicializációs paraméter segítségével állítható, és a rendszer sosem fog ettől a paramétertől több szervert felhasználni. Fontos megemlíteni még a PARALLEL_SERVER paramétert is ami igaz állás esetén parallel módban indítja a szervert.

Az elméleti betekintés után nézzük meg a párhuzamos futtatás gyakorlati oldalát.

Kérdezzük le az adatbázistól a parallelizáltságra vonatkozó információkat:

```
SELECT name, value  
FROM v$parameter  
WHERE name IN ('parallel_max_servers', 'parallel_min_servers');
```

A fenti lekérdezést lefuttatva a következő eredmény kaphatjuk:

NAME	VALUE
parallel_min_servers	0
parallel_max_servers	10
2 rows selected	

Egy lekérdezést többféle képpen is párhuzamosíthatunk. Az egyik legegyszerűbb módja ha úgynevezett hint-et használunk rá. A hint-ek a optimalizálónak szóló ajánlások, amik a végrehajtás módját szabályozhatják. Oracle esetében az optimalizáló dönthet úgy hogy nem veszi figyelembe a hint-et.

```
SELECT /*+PARALLEL(opps 3)*/ *
FROM opportunities opps;
```

A hint egyik paramétere a tábla neve, a másik a párhuzamosítás foka, ami a fenti esetben három lesz majd.

A második módja egy párhuzamos lekérdezés futtatásának, ha egy táblára megadjuk, hogy a táblát érintő utasítások parallel módban fussanak.

```
ALTER TABLE opportunities PARALLEL 3;
```

Ha nem adunk meg számot a PARALLEL szó után, akkor egy alapértelmezett érték lesz a párhuzamosítás foka, amely egy formula alapján számolódik ki: tipikusan ez az érték a gépben található processzormagok kétszerese, ha fürtözött környezetben dolgozunk akkor kétszer a processzormagok száma szorozva a csomópontok számával. Kikapcsolni a NOPARALLEL utasításrésszel tudjuk.

A harmadik lehetőség a párhuzamosítóra, hogyha a munkamenet jellemzőit állítjuk át.

```
ALTER SESSION FORCE PARALLEL QUERY;
```

Nem ilyen egyszerű a helyzet viszont a DML utasításoknál, ugyanis DML utasításokat csak akkor lehet parallel módon futtatni, ha előtte explicit módon beállítjuk ezt az adott munkamenetre:

```
ALTER SESSION ENABLE PARALLEL DML;
```

Ugyanígy hogyha szeretnénk kikapcsolni akkor ENABLE helyett DISABLE-t kell használni.

3.2.4 Materializált nézetek

A materializált nézet, vagy idegen szóval materialized view egy adatbázis objektum, ami lekérdezés eredményt tárol, így a gyakran használt hosszú és költséges lekérdezéseket nem kell újra és újra kiszámolni, és futtatni.

Fizikai megközelítésben a materializált nézetek olyanok akár a táblák, hiszen saját adat szegmenssel rendelkeznek, azonban mutatnak hasonlóságot a nézetekkel is, hiszen valamilyen előre megadott lekérdezés eredményét tárolja, és mutatja a felhasználó felé.

Az adattárházban relatíven ritkán bár, de változnak az adatok, így valamilyen módon a materializált nézet előre kalkulált adatait is szükséges frissíteni. Többféle mód kínálkozik Oracle környezetben ennek a frissítési módnak a megadására, ezt a nézet létrehozásakor kell megadni:

- COMPLETE azaz teljes frissítés, újra lefut a materializált nézetet meghatározó lekérdezés
- FAST vagyis gyors frissítés, csak az alaptáblákon bekövetkezett változásokat érvényesíti a materializált nézeten. Ehhez a művelethez szükség van egy naplóra ami nyilván tartja a változásokat. Ezt nevezzük idegen szóval materialized view log-nak.
- FORCE, ha lehet akkor a FAST frissítést alkalmazza, ha ez nem lehetséges akkor COMPLETE frissítést hajt végre.
- És végül NEVER, ami azt jelenti hogy a nézet tartalma nem fog frissülni sosem.

A materializált nézetekről információt, beleértve a frissítés módját is, az adatszótár nézetekben találhatunk, az ALL_MVIEWS, vagy DBA_MVIEWS, vagy USER_MVIEWS nézetekben.

A frissítés kapcsán a másik jellemző amiről szót kell ejteni az a frissítés gyakorisága illetve módja:

- ON DEMAND, ami kézzel való frissítést jelent, a frissítéshez nyújt segítséget a DBMS_MVIEW csomag
- ON COMMIT, ami akkor frissíti a materializált nézetet hogyha valamelyik alaptáblát módosítottuk. Ez a mód ellenben csak akkor lehetséges, hogyha a materializált nézet nem túl komplex, vagyis frissíthetjük FAST módban
- Szintén automatikus frissítést szolgáltat a módszer, hogyha időintervallumot adunk meg hogy mikor frissüljön a nézet. Ezt a START WITH és NEXT utasításrészekkel tehetjük meg.

Az eddig áttekintett csoportosítások mellett nézzünk most egy újabbat, ami ezúttal nem a materializált nézet definíciójára, hanem a benne foglalt lekérdezésre vonatkozik.

- Materializált nézetek aggregátumokkal a lekérdezésben
Az ilyen nézetek többnyire bizonyos aggregátumokat tartalmaznak. Ezek az aggregátumok a SUM, COUNT, AVG, VARIANCE, MIN, MAX, STDDEV függvények segítségével képződik. Ahhoz, hogy egy ilyen nézetet FAST módban frissíthessünk, az összes a SELECT kifejezésben lévő táblára kell naplónak léteznie, valamint a kiválogatott oszlopok között szerepelnie kell minden oszlopnak amely a GROUP BY részben is szerepel, és lennie kell egy COUNT(*) valamint egy COUNT(oszlopnév) résznek minden aggregált oszlopra.

Példa:

```

CREATE MATERIALIZED VIEW LOG ON products WITH
SEQUENCE, ROWID

(prod_id, prod_name, prod_desc, prod_subcategory,
prod_subcategory_desc,

prod_category, prod_category_desc, prod_weight_class,
prod_unit_of_measure,

prod_pack_size, supplier_id, prod_status, prod_list_price,
prod_min_price)
INCLUDING NEW VALUES;


CREATE MATERIALIZED VIEW LOG ON sales
WITH SEQUENCE, ROWID

(prod_id, cust_id, time_id, channel_id, promo_id, quantity_sold,
amount_sold)
INCLUDING NEW VALUES;


CREATE MATERIALIZED VIEW product_sales_mv
PCTFREE 0 TABLESPACE demo
STORAGE (INITIAL 8k NEXT 8k PCTINCREASE 0)
BUILD IMMEDIATE
REFRESH FAST
ENABLE QUERY REWRITE
AS SELECT p.prod_name, SUM(s.amount_sold) AS dollar_sales,
COUNT(*) AS cnt, COUNT(s.amount_sold) AS cnt_amt
FROM sales s, products p
WHERE s.prod_id = p.prod_id GROUP BY p.prod_name;

```

Az alábbi táblázat szemlélteti az aggregátumok viszonyát egy ilyen típusú materializált nézetben. Hogyha az X aggregátum jelen van a lekérdezésben, akkor Y is szükséges, valamint ekkor Z viszont opcionális.

X	Y	Z
COUNT(kif)	-	-
MIN(kif)		
MAX(kif)		
SUM(kif)	COUNT(kif)	-
SUM(oszlop), ahol oszlop NOT NULL	-	
AVG(kif)	COUNT(kif)	SUM(kif)
STDDEV(kif)	COUNT(kif) SUM(kif)	SUM(kif * kif)
VARIANCE(kif)	COUNT(kif) SUM(kif)	SUM(kif * kif)

- A lekérdezés csak összekapcsolásokat tartalmaz

Ezek a típusú materializált nézetek, csak összekapcsolásokat tartalmaznak, aggregátumokat nem. Így a költséges összekapcsolásokat előre ki tudjuk számolni, hogy egy másik lekérdezésben már ne kelljen.

Példa:

```
CREATE MATERIALIZED VIEW detail_sales_mv
PARALLEL
BUILD IMMEDIATE
REFRESH FORCE AS
SELECT s.rowid "sales_rid", c.cust_id, c.cust_last_name,
s.amount_sold,
s.quantity_sold, s.time_id
FROM sales s, times t, customers c
WHERE s.cust_id = c.cust_id(+) AND s.time_id = t.time_id(+);
```

- Beágyazott materializált nézetek

Találóbbr név inkább az egymásba ágyazott jelző (nested materialized views). Az ilyen típusú nézetet egy már meglévő materializált nézet alapján hozzuk létre. Például, sok aggregátumot tartalmazó nézetünk van már, mondjuk olyanok amelyek különböző dimenziók mentén „göngyölítik fel” az adatokat. Ezekre az aggregátumokat tartalmazó

nézetekre szeretnénk ráhúzni egy összekapcsolásokat tartalmazó nézetet. Ebben az esetben minden olyan materializált nézetnek vagy táblának amit ebben a fő nézetben használunk rendelkeznie kell naplóval. Olyan nézetet ami nem tartalmaz összekapcsolásokat, vagy aggregálásokat, nem ágyazhatunk be. Minden a fő nézet alatt lévő objektum táblaként lesz kezelve.

Példa:

```
CREATE MATERIALIZED VIEW LOG ON sales WITH
ROWID;

CREATE MATERIALIZED VIEW LOG ON customers WITH
ROWID;

CREATE MATERIALIZED VIEW LOG ON times WITH
ROWID;

CREATE MATERIALIZED VIEW join_sales_cust_time
REFRESH FAST ON COMMIT AS
SELECT c.cust_id, c.cust_last_name, s.amount_sold, t.time_id,
       t.day_number_in_week, s.rowid srid, t.rowid trid, c.rowid crid
FROM sales s, customers c, times t
WHERE s.time_id = t.time_id AND s.cust_id = c.cust_id;

CREATE MATERIALIZED VIEW LOG ON join_sales_cust_time
WITH ROWID (cust_last_name, day_number_in_week,
amount_sold)
INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW sum_sales_cust_time
REFRESH FAST ON COMMIT AS
SELECT COUNT(*) cnt_all, SUM(amount_sold) sum_sales,
COUNT(amount_sold)
       cnt_sales, cust_last_name, day_number_in_week
FROM join_sales_cust_time
GROUP BY cust_last_name, day_number_in_week;
```

Láthattuk eddig, hogy a frissítési módok közül a FAST mód a legproblémásabb. Bizonyos peremfeltételeknek teljesülnie kell ahhoz, hogy ezt a módot használni tudjuk.

- A materializált nézet nem tartalmazhat referenciát RAW vagy LONG RAW típusokra
- Nem tartalmazhat analitikus függvényeket a SELECT utasításrészben
- Nem tartalmazhat HAVING utasításrészt allekérdezéssel
- A START WITH, valamint CONNECT BY használata nem megengedett
- A materializált nézet nem tartalmazhat referenciát nem ismétlődő kifejezésekre mint például a SYSDATE vagy ROWNUM
- Nem tartalmazhat olyan allekérdezést amelyek ANY, ALL, NOT EXISTS utasításrészekhez kötődnek

Ezen általános érvényű szabályok mellett még számos egyéb megkötés létezik a FAST frissítési módhoz, attól függően hogy milyen típusú materializált nézetről beszélünk. Az alábbiakban egy-két fontos szabályt szeretnék kiemelni a teljesség igénye nélkül.

Plusz megszorítások csak összekapcsolást tartalmazó lekérdezésre vonatkozóan:

- Nem tartalmazhat GROUP BY utasításrészt vagy aggregátumokat
- Minden a FROM rész után szereplő táblának a ROWID-ja szerepelnie kell a SELECT részben

Plusz megszorítások az aggregátumokat tartalmazó lekérdezésekre vonatkozóan:

- Csak a SUM, COUNT, AVG, STDDEV, VARIANCE, MIN és MAX támogatott FAST frissítés esetén
- COUNT(*)-nak minden esetben lennie kell
- A SELECT-ben nem szerepelhet több alaptáblából származó oszlopot érintő komplex kifejezés. Erre megoldást jelenthetnek az egymásba ágyazott materializált nézetek.

A materializált nézeteknek még van egy fontos tulajdonsága amiről mindenképp meg kell emlékeznünk, és ez az úgy nevezett query rewrite, vagyis a lekérdezés átírás. Ez azt jelenti, hogy

az Oracle képes felismerni, illetve átírni egy lekérdezést, hogy a meglévő materializált nézeteket használja, ezáltal jelentősen felgyorsítva a válaszdíót. Legtöbbször így használják a végfelhasználók a nézeteket: írnak egy lekérdezést bizonyos alaptáblákra ami választ ad az ő kérdéseikre, majd amikor le akarják futtatni a rendszer felismeri, hogy azok az alaptáblák szerepelnek már egy materializált nézetben, így inkább az előre letárolt eredményt olvassa fel ahelyett, hogy újra lefuttatná a lekérdezést teljes egészében. Ilyen módon a végfelhasználó, vagy akár egy kliensprogram írója nem is értesül arról a tényről, hogy ő a materializált nézetet érte el végső soron. Természetesen ez nem jelenti azt hogy közvetlenül a nézetet nem kérdezhetnénk le.

3.2.5 Particionálás

Egy adatbáziskezelő működése során nagyon nagy mennyiségű adat halmozódhat fel egy-egy táblában. Adattárház környezetben óriási mennyiségű adatokkal dolgozunk, azonban egy klasszikus tranzakciókezelő rendszerben is előfordulhat, hogy egy tábla nagyon nagyra duzzad. A méret növekedésével egyre hosszabb ideig tartanak a karbantartási műveletek ezeken a táblákon, és egy idő után elérnek egy olyan méretet amikor már elfogadhatatlanul nagy válaszdíót produkál a művelet. Ilyen karbantartási művelet lehet egy index létrehozása a táblán, több óráig vagy esetleg napig tartana egy index létrehozás, vagy ha le akarnánk menteni a táblát, vagy akár át akarjuk helyezni másik táblatérbe. Az ilyen nagyméretű táblák miatt csökken a rendszer rendelkezésre állása, karbantarthatósága. Ennek kivédésére találták ki a particionálást, aminek az az alapötlete, hogy a problémás táblát osszuk fel több kisebb darabra, úgynevezett partíciókra, majd ezeken a kisebb darabokon már el tudjuk végezni külön-külön a karbantartást.

Jó példa a karbantarthatóságra, az úgynevezett Rolling Window (gördülő ablak) módszer ami a partíciókra támaszkodik. Ennek lényege az, hogy egy táblának az adatait valamilyen időintervallumokra osztjuk, például hónapokra, és ahogyan telik az idő mindig új partíciót hozunk létre az újonnan érkezett adatoknak, a régieket, mondjuk az egy évnél öregebb partíciókat, viszont mindig eldobjuk. Így a táblánk mindig egy év adatait fogja tartalmazni, valamint a karbantarthatósága ennek a táblának sokkal jobb mintha mindig valamilyen programmal kellene törölni az egy évnél öregebb adatokat, hiszen elég csak a partíciót eldobni.

A partícionálásnak azonban nem csak karbantarthatósági szempontok szerint van haszna, hanem a lekérdezések oldaláról is.

- A lekérdezés optimalizáló jelezni fogja, hogy melyik partíciót kell vizsgálni, és amelyik partícióról nincsen adat azt nem fogja végigolvasni. Például, hogyha hónapok szerint van partícionálva egy tábla, és mi egy olyan kérdést teszünk fel a rendszernek, ahol megszorítjuk az időintervallumot, akkor az optimalizáló felismeri, és kihagyja azokat a partíciókat amelyek kívül esnek az általunk megadott intervallumon. Ezt hívják idegen szóval Partition Pruning-nak
- OLTP rendszerben is hasznos lehet az a tulajdonsága a partícióknak, hogy párhuzamosan lehet őket módosítani, párhuzamosan lehet rajtuk INSERT, DELETE, vagy UPDATE műveletet futtatni, hiszen a zárolásokat külön külön oldják meg.
- Fogasíthatunk némi terheléselosztást is a háttértárolók között is, a különböző partíciók jó megszervezésével.

Partícionálni nem csak táblákat, hanem indexeket is lehet, valamint ezeket kombinálni is lehet, elképzelhető olyan szituáció, hogy partícionált táblához nem partícionált index tartozik, vagy mindkettő partícionálva van. Természetesen mindegyik rendszer más, így nem lehet általános érvényű szabályt felállítani erre, de gyakran partícionált tábla mellé partícionált indexet raknak.

Partícionálni nem csak egyféle szisztéma szerint lehet:

- Range, vagyis valamilyen értékhatárok, intervallumok szerinti partícionálás. Ezek az intervallumok különböző partíciók lesznek. Az adatok attól függően hogy melyik értékhatárok közé esnek, abba az adott partícióba fognak kerülni. Az eddig említett példa, amikor hónapok szerint volt felosztva a tábla, ebbe a kategóriába esik.

Példa:

```
CREATE TABLE sales_range
(salesman_id NUMBER(5),
salesman_name VARCHAR2(30),
sales_amount NUMBER(10),
sales_date DATE)
PARTITION BY RANGE(sales_date)
(
PARTITION sales_jan2000 VALUES LESS
THAN(TO_DATE('02/01/2000','DD/MM/YYYY')),
PARTITION sales_feb2000 VALUES LESS
THAN(TO_DATE('03/01/2000','DD/MM/YYYY')),
PARTITION sales_mar2000 VALUES LESS
THAN(TO_DATE('04/01/2000','DD/MM/YYYY')),
PARTITION sales_apr2000 VALUES LESS
THAN(TO_DATE('05/01/2000','DD/MM/YYYY'))
);
```

- List, vagyis valamilyen felsorolással definiálja, hogy melyik partícióba milyen értékek kerüljenek. Ebben az esetben meg kell adnunk egy statikus listát, amely eldönti az adatok további sorsát. Ilyen lehet például valamilyen régiók szerinti felosztás.

Példa:

```
CREATE TABLE sales_list
(salesman_id NUMBER(5),
salesman_name VARCHAR2(30),
sales_state VARCHAR2(20),
sales_amount NUMBER(10),
sales_date DATE)
PARTITION BY LIST(sales_state)
(
PARTITION sales_west VALUES('California', 'Hawaii'),
PARTITION sales_east VALUES ('New York', 'Virginia',
'Florida'),
PARTITION sales_central VALUES('Texas', 'Illinois')
PARTITION sales_other VALUES(DEFAULT)
);
```

- Hash partícionálás, egy hash függvény segítségével véletlenszerűen szórjuk szét az adatokat a különböző partíciókban. A hash függvény dönti el, hogy az adat hova fog kerülni. Ilyen módon megközelítőleg egyenlő méretű partíciók jönnek létre, és így az erre a táblára ráengedett lekérdezés valószínűleg nem fog nagyon lassan eredményt produkálni. Szintén érdemes ezt a módszert választani, hogyha a párhuzamos DML-ek, vagy a partition pruning fontosak az adott táblánál.

Példa:

```
CREATE TABLE sales_hash
(salesman_id NUMBER(5),
salesman_name VARCHAR2(30),
sales_amount NUMBER(10),
week_no NUMBER(2))
PARTITION BY HASH(salesman_id)
PARTITIONS 4
STORE IN (data1, data2, data3, data4);
```

- A partícionálási módszerek lehet kombinálni is.
 - Range-Hash, vagyis feloszthatom a táblát valamilyen intervallumok szerint, aztán tovább hash függvény szerint
 - Range-List, hasonlóan mint az előzőnél, felosztom a táblát, mondjuk hónapokra, és utána még egy ország régiói szerint tovább bontom a már meglévő partíciókat.

Példa:

```
CREATE TABLE sales_composite
(salesman_id NUMBER(5),
salesman_name VARCHAR2(30),
sales_amount NUMBER(10),
sales_date DATE)
PARTITION BY RANGE(sales_date)
SUBPARTITION BY HASH(salesman_id)
SUBPARTITION TEMPLATE(
SUBPARTITION sp1 TABLESPACE data1,
SUBPARTITION sp2 TABLESPACE data2,
SUBPARTITION sp3 TABLESPACE data3,
SUBPARTITION sp4 TABLESPACE data4)
(PARTITION sales_jan2000 VALUES LESS
THAN(TO_DATE('02/01/2000','DD/MM/YYYY'))
PARTITION sales_feb2000 VALUES LESS
THAN(TO_DATE('03/01/2000','DD/MM/YYYY'))
PARTITION sales_mar2000 VALUES LESS
THAN(TO_DATE('04/01/2000','DD/MM/YYYY'))
PARTITION sales_apr2000 VALUES LESS
THAN(TO_DATE('05/01/2000','DD/MM/YYYY'))
PARTITION sales_may2000 VALUES LESS
THAN(TO_DATE('06/01/2000','DD/MM/YYYY')));
```

ÜZLETI INTELLIGENCIA

1. Az üzleti intelligencia definíciója

Az előző fejezetben egy technikaibb jellegű megközelítéssel megismerkedhettünk a nagyvállalatok döntéshozást támogató rendszereinek az alapjával, az adattárházakkal és a hozzájuk tartozó, vagy legalábbis az ott gyakori technológiákkal. Míg az adattárházak a döntéstámogató rendszerek (Decision Support Systems – DSS) az alját, az alapját képezik, az üzleti intelligencia a tetején van ennek az architektúrának, és ez az a réteg ami elsősorban a felhasználóval érintkezik.

Az üzleti intelligencia fogalmának meghatározásakor talán még nehezebb dolgunk van mint az adattárház fogalom kialakításakor, ezért a legcélszerűbb az lehet, hogyha a történelmet hívjuk segítségül. Hans Peter Luhn IBM fejlesztő 1958-ban egy cikkében már használta az üzleti intelligencia kifejezést: Az intelligencia „a képesség, hogy megértsük a tények közötti kapcsolatot, úgy hogy az a kitűzött cél felé vezessen minket”.

Később Howard Dresner (későbbi Gartner Csoport elemző) 1989-ben javasolta, hogy a BI vagyis az üzleti intelligencia egy összefogó fogalom legyen ami leírja, „az elképzeléseket, és metódusokat az üzleti döntéshozás fejlesztésére tény alapú támogató rendszereket használva”.

Később egyre nagyobb hangsúlyt nyert magának az üzleti életben az üzleti intelligencia, és az informatika egyik nagy területévé nőtte ki magát a döntéstámogatás, ahogyan azt Dr. Drótos György is megfogalmazta 2003 évi cikkében: "Az informatika üzleti jelentősége ma már messze túlnő az automatizálási szerepkörön, az Integrált Vállalatirányítási Rendszerek és az azokra épülő Üzleti intelligencia megoldások szorosan összefonódnak a cég alapvető folyamataival, képességeivel. Ezért nélkülözhetetlen, hogy a szervezet stratégiájával összhangban kidolgozásra és megvalósításra kerüljön az informatikai stratégia is."

A döntéstámogató rendszerek komplexitásuknak köszönhetően nagyfokú tervezést igényelnek, még a megvalósítási szakasz előtt. A leggyakoribb kérdések egy ilyen rendszer bevezetésénél például: Mennyire sikeresek a bevezetési projektek?; Melyek a jellemző problémák?; Milyen

tényezők befolyásolják leginkább a projektek sikerességét?; Melyek a leggyakoribb buktatók és hogyan kerüljük el őket?" (Kristóf Péter Krisztián – 2005)

Látható, hogy akárcsak az adattárházak esetében, itt sem egyértelmű a definíció. Összefoglalva az üzleti intelligencia olyan képességeket, technológiákat, alkalmazásokat vagy akár gyakorlatokat jelent amiket az üzlet érdekében használnak, hogy annak kereskedelmi összefüggéseit segítsen feltárni. Az üzleti intelligencia alkalmazások szolgáltatnak jelen idejű, historikus, és prediktív adatokat is az üzleti műveleteket illetően.

Az üzleti intelligencia alkalmazások rendkívül sokrétűek lehetnek ennek megfelelően, hiszen az egyik részfeladata ennek a területnek a megfelelő adatok gyors kinyerése, és annak a felhasználó felé prezentálása.

2. Az üzleti intelligencia által megoldandó feladatok, tipikus problémák

A BI definíciójának megalkotása után tekintsük át az üzleti intelligencia feladatait, problémáit, milyen kérdésekre kell választ adnia egy általános rendszernek.

Egy nagyvállalat működése során számtalan beszállítótól származó szoftvert használhat. Egyet a humán erőforrások kezelésére, egyet a vámügyek lebonyolítására, használhat külön levelezőt, vagy akár munkaidő nyilvántartó programot. Ezekben az alkalmazásokban a normális napi használat során felhalmozódik az adat, és minden alkalmazás saját magának megoldja az adatainak prezentálását a felhasználók felé. Az előbb leírt helyzetnek több gyenge pontja is van, egyfelől a heterogenitás: az adatok szét vannak szórva különböző rendszerekben, másfelől az hogy nincsenek csoportosítva, aggregálva a felhasználók számára hasznos formába rendezve. Nagyon sok adat van jelen a különböző alkalmazásokban, viszont az üzlet ezekbe az adatokba úgy mond nem lát bele, nincs átfogó képe róla ami segíthetne az üzlet hatékonyságának növelésében. Erre jelenthet megoldást az adattárház amelybe előre összegyűjtjük az adatokat rendezett formában, és előre definiálhatunk aggregált lekérdezéseket a felhasználók számára materializált nézetekkel például. Ez az adattárház fogja az alapot jelenteni egy üzleti intelligencia rendszerhez.

Szintén problémát okoz egy üzleti döntéshozás során, hogyha a jelentések többsége kézzel készül. Egyik oldalról nagyon sok időbe is telhet egy-egy kimutatás elkészítése manuálisan elkészítve, amíg természetesen az emberünket is lefoglalja a jelentés elkészítése. Az üzleti intelligencia rendszerek, gyorsan képesek jóval kevesebb emberi beavatkozással ugyanarra az eredményre. A kézi riportoknak azonban nem csak ez a hátránya, nem megoldott a biztonság kérdése is. Egy üzleti intelligencia rendszernek, valamint egy adattárháznak erre is megfelelő választ kell tudnia adni. Ki milyen adatot láthat, hogyan és milyen lekérdezésekhez férhet hozzá, és így tovább. Egy üzleti intelligencia rendszer előkészíti az adatokat, és egy intuitív felületen teszi lehetővé, hogy a lekérdezéseket megalkossuk.

A következő problémakör az adatok időbelisége. Leginkább múltbéli adatokat tudunk elemezni, azonban jó, hogyha ezeket az adatokat egy adott esemény bekövetkezte után minél hamarabb meg tudjuk vizsgálni. Kézzel készített jelentések esetén például ez akár több nap, vagy hét is lehet. Ha egy negyedéves zárást tekintünk példának, az üzleti döntéshozóknak minél hamarabb célszerű megvizsgálni az elmúlt negyedéves adatokat, hogy a következő negyedévre döntéseket hozhassanak. Ezzel el is érkeztünk az időbeliség ellenpontjához, azaz a múltbéli adatok alapján sokszor jövőbe mutató konklúziókat próbálunk levonni, vagy valamiféle jóslást áll szándékunkban adni.

3. Enterprise Performance Management (EPM)

Az EPM, vagyis a vállalati teljesítmény menedzsment fogalmát a Gartner elemző cég 2001-ben vezette be, gyakran a BPM (Business Performance Management), vagy a CPM (Corporate Performance Management) betűszót használják helyette.

Az EPM magába foglalja a folyamatokat, módszereket, mérést, valamint a rendszereket, amelyek segítenek a szervezeteknek, hogy optimalizálják az üzleti teljesítményüket. Keretet ad az üzleti folyamatok mérésére, szervezésére, automatizálására, amelyek segítségével hatékonyabbá tudjuk őket tenni. Másképp megfogalmazva segít az üzletnek a különböző erőforrások, pénzügyi, humán, stb. hatékony kihasználását. Némely források szerint az EPM az üzleti intelligencia új generációja.

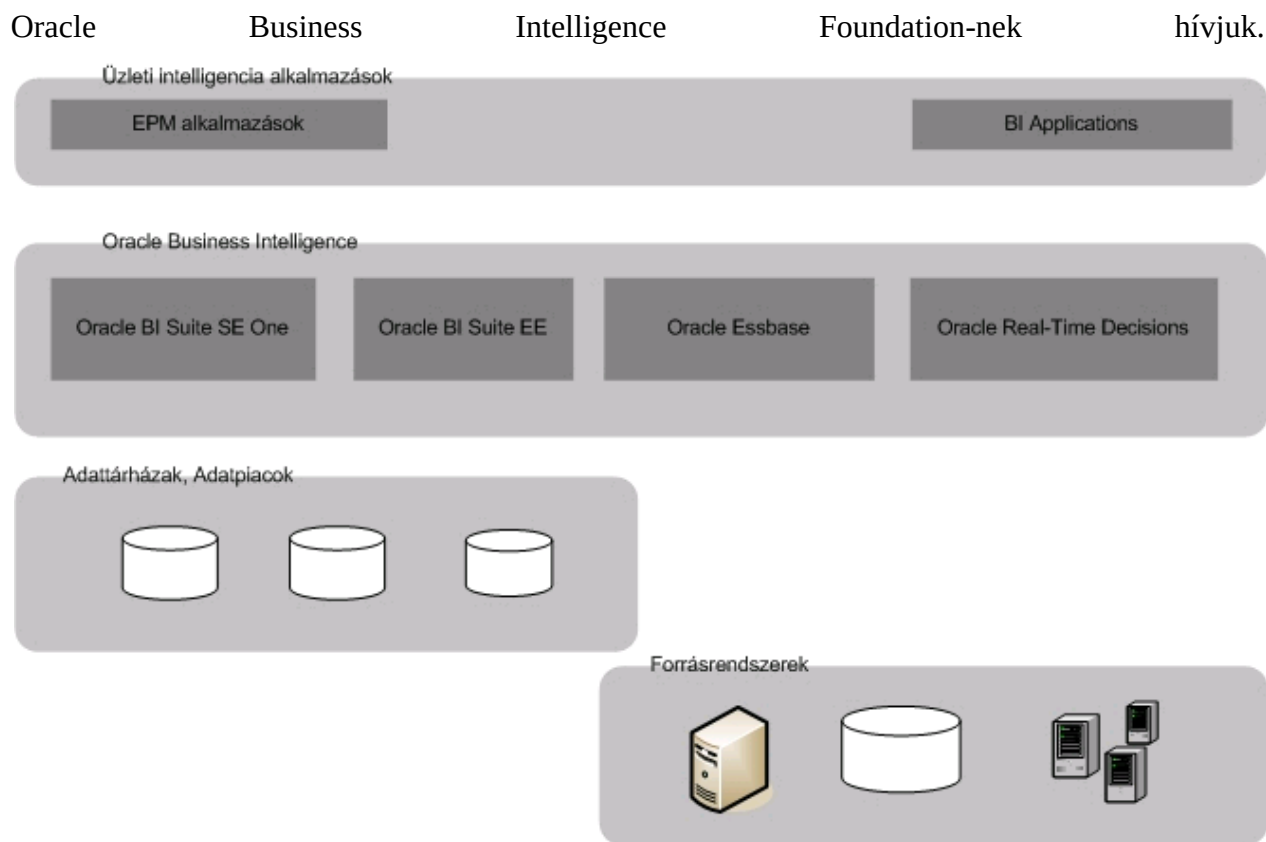
Informatikai szemszögből az EPM alkalmazások, mint például a pénzügyi konszolidáció segítése, vagy a kontrolling támogatása, jövedelmezőség mérés, mind egy-egy üzleti intelligencia alapra épülő célalkalmazások.

Az üzleti intelligencia témakörének alapfogalmai megismerése után tekintsük át, hogy az Oracle milyen megoldást nyújt a felvetett problémákra, és feladatokra.

4. Oracle BI és DW architektúra

Bár az adattárház architektúráról már volt szó, említsük meg azonban még egyszer a teljes kép kedvéért. Emlékezzünk az adattárház forrásrendszerekből táplálkozik, és nincs ez így másként az üzleti intelligencia rendszereknél sem. A forrásrendszerekre építhetünk egy adattárházat, illetve tárházakat, valamint adatpiacokat, azonban az üzleti intelligencia alkalmazások többnyire képesek a forrásrendszerekből közvetlenül adathoz jutni. Jogos lehet a kérdés, hogy ez miért van így. Az adattárházak előre összegzett, aggregált adatokat tartalmaznak, valamilyen múltbéli idővel bezárólag. Ez az adott DW tulajdonságaitól függ, lehet hogy havi betöltést alkalmazunk, és akkor az adataink ennek megfelelően lehetnek akár több hetesek is. Elképzelhető azonban olyan helyzet, olyan adat amelyet ad-hoc módon szeretnénk elemezni, és a kívánt adat még nem került be az adattárházba, mert még nem történt betöltés amióta bekerült a forrásrendszerbe, vagy pedig olyan a részletezettsége, hogy amiatt nem került be a DW-be. Ez indokolja a BI rendszerek közvetlen olvasási képességét a forrásrendszerekből.

E két réteg fölé jön az üzleti intelligencia réteg amely több elemből tevődik össze. Az Oracle többféle üzleti intelligencia csomagot kínál. Ezek az üzleti intelligencia csomagok hasonló célokat szolgálnak, különbségük abból adódik amennyi és amilyen alkalmazásokat foglalnak magukban. Ezen kívül ezt a réteget ki lehet egészíteni más elemekkel is, például az Oracle Essbase-zel, vagy az Oracle Real-Time Decisions-zal. Előbbi egy OLAP szerver amely gyors fejlesztési környezetet biztosít testreszabott analitikus és EPM alkalmazások fejlesztésére. Az Essbase bár ide van sorolva, mint gyors EPM alkalmazás fejlesztő környezet, átmenetet képez a BI réteg és felette lévő EPM réteg között. A Real-Time Decisions (RTD) segítségével valós időben döntéstámogatást valósíthatunk meg rendszerünkben. Ez az üzleti intelligencia réteget



8. ábra Oracle üzleti intelligencia architektúra

A struktúra tetején találhatóak végül a már tárgyalt EPM célalkalmazások, valamint az Oracle BI Applications. Ez utóbbi bizonyos üzleti területekre előre elkészített adatmodelleket tartalmaz, előre elkészített eljárások viszik be az adatokat a különböző forrásrendszerekből, az adott területnek megfelelő adatmodellre, ahol majd szintén előre elkészített elemzőlapokon, riportokon keresztül tudjuk értékelni az adatokat. Az említett üzleti területek lehetnek humánerőforrás, pénzügyi, értékesítés, marketing, és még sok más.

4.1 Oracle üzleti intelligencia csomagok: BI Suite Enterprise Edition és BI Suite SE One

4.1.1 Nagyvállalati verzió (enterprise edition)

A nagyvállalati verzió a kettő közül a nagyobbik csomag.

Ahogy az elvárható egy üzleti intelligencia rendszertől, a BI Suite EE is képes heterogén környezetben működni, bármilyen adatforrásból képes táplálkozni, legyen szó akár milyen

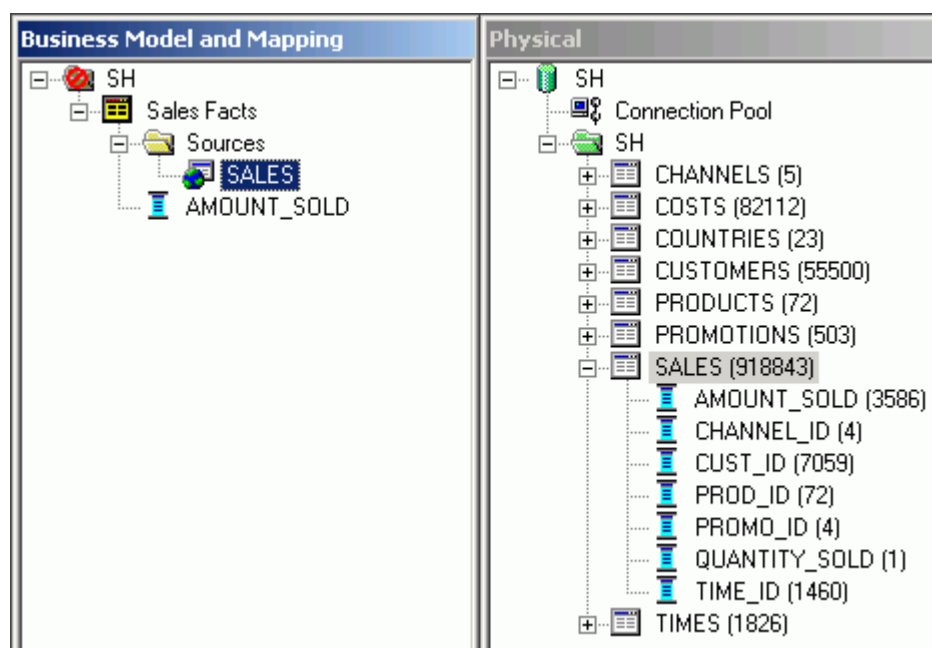
adatbáziskezelőről, natív vagy ODBC elérésről, vagy akár szolgáltatás-orientált architektúráról (SOA). A rendszernek van egy szerver része (BI szerver), amely összekapcsolja a különböző forrásból származó adatokat, és igyekszik ezt optimálisan megtenni. Emellett ehhez a szerverhez szervesen kapcsolódik egy a rendszer egészére kiterjedő metaadat kezelő réteg. A gyakorlatban a metaadat kezelés azt jelenti, hogyha a felhasználó bármilyen eszközzel, legyen szó akár beépített lekérdezésről, akár ad-hoc elemzésről, nyúl a rendszerhez, számára az üzleti fogalmak mindig ugyanazt jelentsék. Az ügyfél fogalom itt is, ott is ugyanazt fogja jelenteni. Ez rendkívül fontos egy BI rendszerben, hiszen a különböző helyről érkező adatok másként jellemezhetik az ügyfél fogalmat, hogy a példánál maradjunk.

Az Oracle 3 rétegű metaadat kezelést valósított meg a rendszereiben:

- Fizikai réteg, ami arról gondoskodik, hogy az adatforrásokhoz tudjunk kapcsolódni. Emlékezzünk ebben az esetben a BI rendszer adatforrása nem csak adattárház lehet.
- Logikai, vagy üzleti modell réteg. Ez a réteg egy logikai csillagsémát készít a különböző adatforrásokból származó adatokból, amik bármilyen formában lehetnek. Ezt a logikai csillagsémát használja aztán majd a különböző elemzési szempontok szerinti vizsgálódásokhoz. Erre a rétegre azért is szükség van, mivel a forrás nem csak adattárház lehet ahol feltételezhetjük a csillag séma meglétét, így az adatok szerkezete jelentősen eltérhet egymástól a különböző forrásrendszerekben. A végső lekérdezésnél, hogyha például a bevétel fogalomra hivatkozunk, akkor a rendszer kezeli le, hogy a különböző rendszerekből a bevétel jelentéssel bíró adatot szedje össze.
- Prezentációs réteg, amely témaköröket tartalmaz, melyek üzleti fogalmakat csoportosítanak és tesznek elérhetővé a felhasználó számára. Például egy ID oszlopot nem akarunk megmutatni a felhasználónak, így az nem lesz benne egy témakörben sem, de benne lesznek számítások és más üzleti jelentéssel bíró dolgok.

A metaadat réteg elkészítésénél, illetve beállításánál, először a fizikai réteggel kezdjük. Megadjuk a forrásrendszerek elérési útját, helyét, például adatbázis esetében a hálózati nevet, felhasználónevet, jelszót, stb. Ezekből a forrásrendszerekből konkrétan nem olvasunk adatokat, csak a réteg elnevezésének megfelelően metaadatokat, például azt hogy a táblák hogyan néznek

ki, milyen oszlopaik vannak, és milyen szerkezetűek. A rendszer ezek kapcsolatát is képes felolvasni. Ezekből az adatforrásokból aztán elkészíthetjük a csillag sémát, a séma tulajdonságai szerint, van egy vagy több tény tábla, amihez vagy amikhez dimenziók kapcsolódnak. Ezen dimenziókhoz még kapcsolódhatnak összegzések, aggregációk. Az utolsó lépés a felhasználóknak szánt megjelenítés, az oszlopok üzleti fogalmakkal való összerendelése, illetve annak meghatározása, hogy mit lásson a felhasználó. A metaadat réteg tervezésére és elkészítésére az Oracle BI Administration Tool használható.



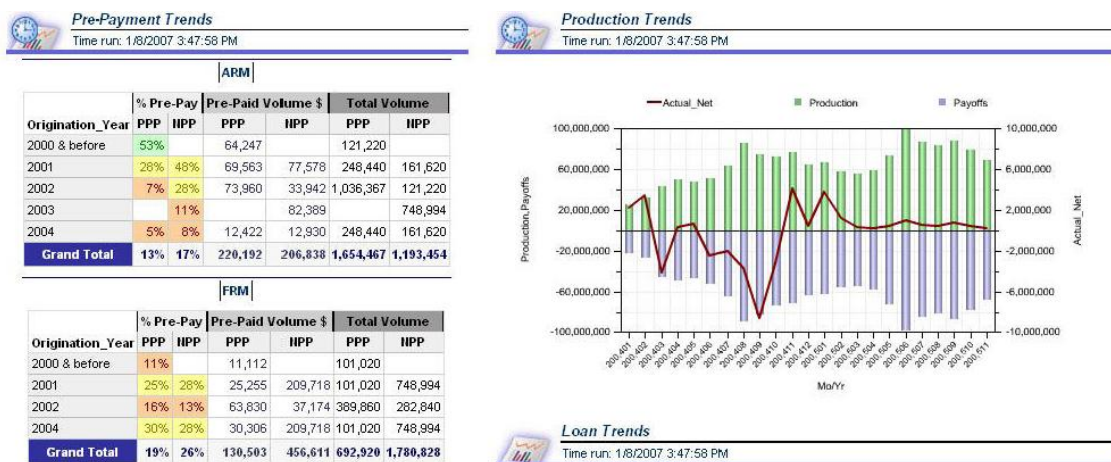
9. ábra Az Oracle BI Administration Tool fizikai réteg és üzleti modell nézetei

Az Administration Tool képes a futó rendszer metaadat rétegét is megnyitni (on-line), vagy szerkeszthetünk metaadatokat offline módon is.

A BI Suite EE felhasználói rétegét adják a különböző elemző, lekérdező, és egyéb felhasználó felé információt közlő eszközök:

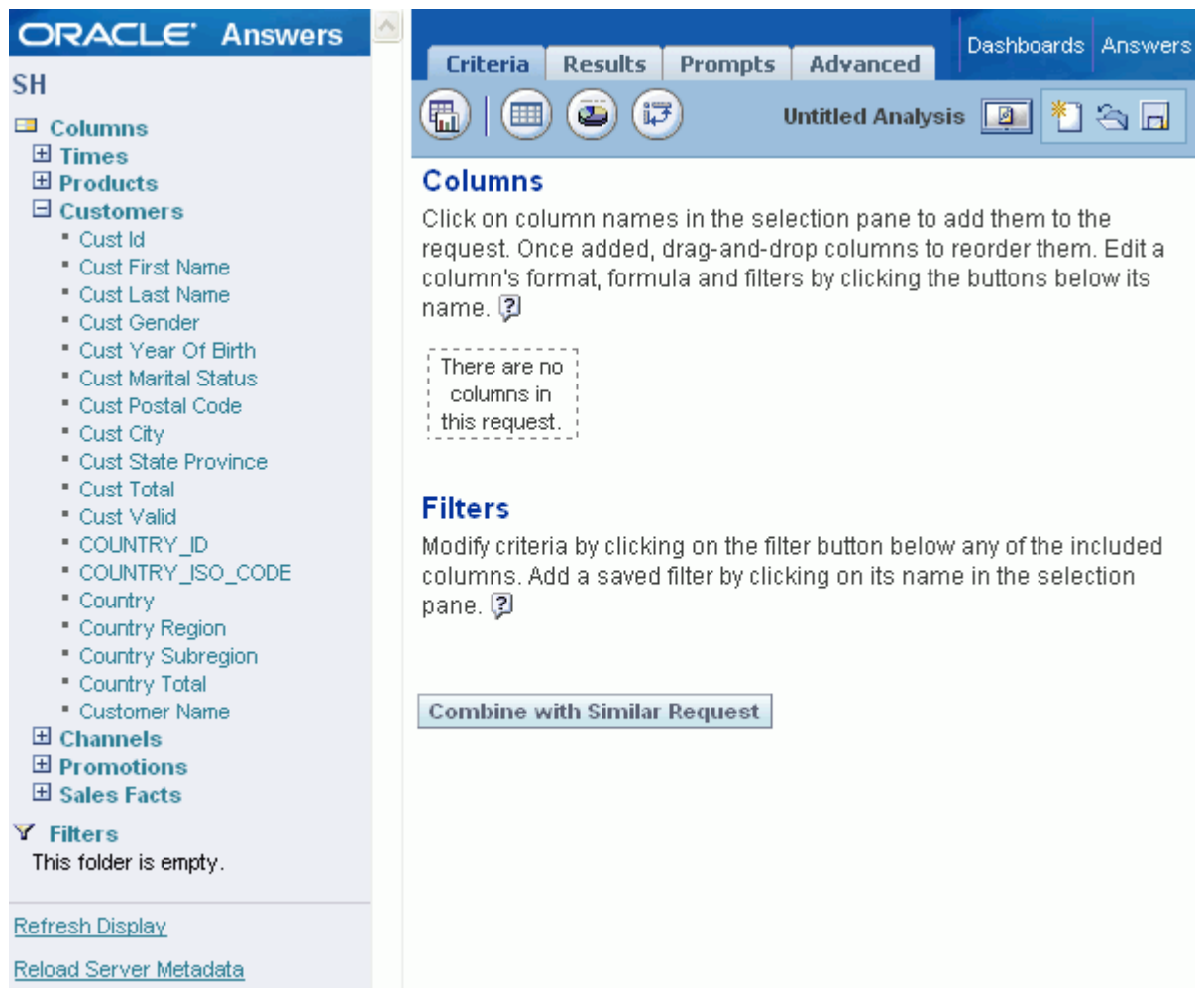
- Dashboard-ok vagy magyarul irányítópultok, ezek csoportosítják a más-más helyről érkező információkat, és megjelenítik azt a böngészőben, ugyanis az Interactive dashboards teljesen böngészőalapú alkalmazás.

Az irányítópultokon megjelenhet az általunk már korábban elmentett lekérdezés, vagy a mások által publikált lekérdezések. A Dashboards lehetőséget nyújt arra, hogyha valaki rendelkezik a megfelelő jogkörrel, akkor mások számára is elérhetővé tehet irányítópult elemeket.



10. ábra Interactive Dashboards részlet

- Lehetőség van ad-hoc elemzések elvégzésére is az Oracle Answers segítségével amikor a felhasználó saját maga tudja összeállítani a saját lekérdezéseit, kimutatásait. Az Answers természetesen támaszkodik a metaadat rétegre amit definiáltunk, a prezentációs réteg itt jelenik majd meg a felhasználó számára. Az alkalmazás tiszta webes felületet nyújt az imént leírt funkcionalitások elvégzéséhez, valamint az elkészített kimutatásokat elmenthetjük, és kirakhatjuk a Dashboard-ok egyikére is.



11. ábra Oracle Answers bal oldalon a metaadatok prezentációs rétegével

Az eszköz használatának első lépéseként el kell döntenünk, hogy milyen témakörben szeretnénk a vizsgálódást folytatni. Ezt a témakört a prezentációs rétegből választjuk ki. Ennek a választásnak nem csak üzleti, hanem jogosultsági oldala is van, megadhatjuk, hogy ki milyen témakörökhöz, milyen adatokhoz fér hozzá. A témakör választás után továbbléphetünk valamilyen dimenzió mentén, ami már ad is számunkra valamilyen kimutatást.

- Szintén fontos elemét képezi a BI Suite-nak a BI Publisher-nek nevezett a megjelenítésért felelős komponens. Ennek feladata az akár nyomdai minőségű riport kimenetek előállítása. Ha a szervezetünkön kívülre készítünk kimutatásoknál különösen, de akár

belső használatra szánt riportoknál is hangsúlyos lehet az elemzés kimenete. A BI Publisher sablonokat használ a kimenetek generálására. Ha a felhasználó elkészít egy sablont, össze lehet rendelni a sablonban meghatározott mezőket a riport mezőivel, és utána a BI Publisher motor elkészíti a kimenetet, ami többféle fájltypus lehet, akár PDF, vagy Word formátum is.

- A BI Suite nagyvállalati verziójában helyett kapott a funkcionalitás, hogy kivételeket lehet definiálni különböző eseményekre, valamint ezen kivételekhez riasztásokat rendelhetünk. Ilyen kivétel lehet például hogyha a valamilyen figyelt érték eltér 10%-tól jobban az előző hónaphoz képest, akkor a rendszer egy riasztást generál számunkra, amely telefonon, emailben, vagy akár RSS Feed formájában érkezhethet hozzánk. Ez a komponens az Oracle BI Delivers.

4.1.2 Standard verzió (BI Suite SE One)

A standard verzió az a kisebb vállalatoknak szánt üzleti intelligencia csomag. Ennek kínálata némileg szegényesebb mint a nagyobb nagyvállalati verzióé, azonban ez a kiserelés is tartalmaz minden üzleti intelligencia szempontból lényeges komponenst tartalmaz. Ezek között van természetesen a BI szerver és a metaadat kezelés, valamint a 3 fő alkalmazás:

- Interactive Dashboards
- Oracle Answers
- BI Publisher

Ezen kívül a csomagban még helyet kapott egy Oracle adatbáziskezelő az adattárház megoldásoknak, valamint az Oracle Warehouse Builder adattárházt támogató eszköz. Az ebben a csomagban foglaltakra ugyanazok érvényesek ami a nagyvállalati verzióban lévő ugyanezen komponensekre.

5. Oracle BI példa

A következőkben tekintsünk át egy példát, amelynek a témaköre az Opportunity-k alakulása egy szervezet életében. Az opportunity, szó szerinti fordításban lehetőség. Az üzleti életben is lehetőséget jelent, lehetőséget az eladásra. Akkor beszélhetünk egy lehetőségről, hogyha egy

vevő intenzíven érdeklődik a termékünk iránt, és elképzelhető hogy meg is veszi. Ha sikerült eladást produkálni az opportunity-ból akkor a lehetőséget megnyertük, ellenben az a szituáció is elképzelhető, hogy a vevő végül eláll a rendelestől, és esetleg egy konkurens cég kínálatából választ magának megfelelőt.

A példában az Oracle BI Enterprise vagyis a nagyvállalati verzióját használtam.

5.1 Belépés az Interactive Dashboards-ba

Az eszközbe való bejelentkezés folyamán, a felhasználónév és jelszó mellett, még választhatunk számtalan nyelv közül amelyen használhatjuk az Oracle BI alkalmazásokat.



Oracle Business Intelligence

Please enter your User ID and Password below, and then press the Log In button.

User ID

Password

Select a Language

Oracle Business Intelligence 10.1.3.4

Copyright © 1997, 2007, Oracle. All rights reserved. The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

12. ábra Oracle BI bejelentkező képernyő

Első bejelentkezéskor az saját vezérlőpultunk még üres, ellenben elérhetünk más osztott vezérlőpultokat is.

13. ábra Saját vezérlőpult, Overview, Ranking & Toppers valamint History & Benching vezérlőpultok

Az Interactive Dashboards-ból közvetlenül elérhető az Answers ad-hoc lekérdező is (Answers link), valamint a saját felhasználói fiókunk beállítási képernyőjét is elérhetjük innen (My Account link), ahol az időzóna, és nyelvi beállítások mellett elérhető az Oracle Delivers beállításai is, itt tudunk megadni mobilszámot, vagy emailcímet a generált értesítések, illetve riasztások céljaként.

14. ábra Navigációs sáv az Interactive Dashboards-ban

5.2 Oracle Answers

Az Answers-be átlépve átnézhetjük az általunk elérhető elemzéseket: az általunk készítéseket vagy a megosztottakat, vagy készíthetünk egy új elemzést is egy tárgyterületet választva, ami ezúttal az Opportunity-k köré épül. Kiválasztása után a már ismert képernyő fogad minket, bal oldalt a metaadatok prezentációs rétegével, középen a szerkesztőmezővel. A bal oldalon lévő oszlopokra kattintva adhatjuk őket a szerkesztőmezőhöz, ahol majd tovább alakíthatjuk az általunk kívánt formára.

A vizsgálódás alapjait képező oszlopok között természetesen helyet kaptak az opportunity-k, szám szerinti bontásban, hány darab van belőlük, valamint egy USA dollár azaz érték szerinti bontásban. Ezeket a szempontokat további részekre bonthatjuk, hogyha a többi oszlopból válogatunk, például létrehozás ideje szerint, vagy földrajzi elhelyezkedés szerint, az opportunity státusza szerint, vagy akár a megnyerés esélye szerint.

A következőkben tekintsünk át egy olyan elemzést, amely megmutatja nekünk, hogy hány darab opportunity-t hoztak létre, és ezt az információt vizsgáljuk meg földrajzi elhelyezkedés és létrehozási idő függvényében! Láthatjuk a bal oldalon azonban, hogy az oszlopok között a

létrehozási idő is, valamint a földrajzi elhelyezkedés is tovább van bontva eltérő szemcsézettség szerint: a létrehozási idő év, üzleti negyedév, hónap, nap, valamint a földrajzi elhelyezkedés kontinens, régi, alrégió, ország szerint. Mi most a legnagyobbat választjuk mint a kettőből, azonban később láthatjuk majd, hogy bele tudunk fűzni a lekérdezésünkbe, hogyha kisebb egységekre is kíváncsiak vagyunk.

Első lépésként a megfelelő oszlopokra való kattintással hozzáadjuk őket a szerkesztő részhez.

ORACLE Answers

Criteria Results Prompts

Opportunities

Columns

- Dimensions ---
- Actual Close Date Dim
- Creation Date Dim
 - Creation Year
 - Creation Quarter
 - Creation Month
 - Creation Date
- Estimated Close Date Dim
- Follow Up Date Dim
- Geographic Location Dim
 - Geo Continent
 - Geo Region
 - Geo Subregion
 - Geo Country
- Last Update Date Dim
- Opportunity Details Dim
- Sales Territory Dim
- Source System Dim
- Stage Dim
- Status Dim
- Win Probability Dim
- Facts ---
- Opportunity Facts
 - Opportunity Amt USD
 - # of Opportunities

Columns

Click on column names in the selection pane to add them to the request. Once added below its name.

Creation Date Dim	Geographic Location Dim	Opportunity Facts
Creation Year	Geo Continent	# of Opportunities

Display Results Remove All

Filters Display and edit views of results

Add filters to the request criteria by holding down the CTRL key and clicking on column names in the selection pane.

15. ábra Oracle Answers bal oldalon a választható oszlopokkal, középen a már hozzáadott oszlopok a szerkesztő részben

Az egyes oszlopok átrendezhetőek „fogd-és-vidd” módszerrel. A lekérdezésünk gyakorlatilag elkészült, a Display Results gombra kattintva megtekinthetjük az eredményt. Az eredmények megtekintése mellett azonban további tevékenységeket tudunk a Results (eredmények) fülön elvégezni, rendelhetünk grafikont a visszakapott eredményhalmazhoz, vagy szűrőket is

helyezhetünk rá, hogyha szűkíteni akarjuk az adathalmazunkat. Továbbá akár le is tölthetjük Excel, Powerpoint vagy más egyéb formátumokban a visszakapott adatokat:

Creation Year	Geo Continent	# of Opportunities
2002	Asia/RoW	5,088
	Americas	13,992
	Asia/RoW	13,846
	Europe	7
2003	Americas	20,08
	Asia/RoW	14,895
	Europe	7
2004	Americas	6,578
	Asia/RoW	9,879
	Europe	12,096
2005	Americas	5,342
	Asia/RoW	9,535
	Europe	12,218
2006	Americas	6,701
	Asia/RoW	9,17
	Europe	12,149
2007	Americas	4,992
	Asia/RoW	5,472
	Europe	7,465
2008	Americas	68
	Asia/RoW	53
	Europe	152

16. ábra A lekérdezésből származó Excel kimenet

5.3 Pivot

Alakítsuk most át a lekérdezésünket úgy, hogy a kontinensek legyenek az oszlopok, az évek legyenek a sorok, és a kettő metszetében legyen az odaillő számérték. Ezt az átalakítást megtehetjük a Pivot Table gomb segítségével (). A gombra kattintva az újonnan megnyílt szerkesztőfelületen áthúzgálhatjuk a különböző értékcsoportokat a megfelelő helyre, legyen ez akár oszlop vagy sor.

Az OLAP kockáknál tárgyalt részből ismerős lehet a Pivot table szerkesztőmező, ugyanazokkal a fogalmakkal dolgozunk itt is: középen vannak a számszerű tények, a measure-ök, amelyek jelen esetben az opportunity mennyiségadatokat fogják jelenteni, amelyek bizonyos dimenziók itt idő,

és földrajzi helyzet szerint vannak kategorizálva. Bár az Oracle BI elrejti előlünk a tárolás mikéntjét, az OLAP kocka és a jelen lekérdezés közötti analógiát felfedezhetjük.

Természetesen itt még további oszlopokat is behozhatunk a lekérdezésünkbe. Szintén lehetőségünk van megváltoztatni az egyes oszlopok formátumát is, valamint azt is megadhatjuk akár, hogy a cellákban mutatott értékeket oszlassa el százalékosan oszlop vagy sor szerint.

A hagyományos érték szerinti megjelenítésen hagyva a beállításokat, tekintsük meg a pivot table kimenetét!

Creation Year	# of Opportunities		
	Asia/RoW	Americas	Europe
2002	5,088		
2003	13,846	13,992	7
2004	14,895	20,080	7
2005	9,879	6,578	12,096
2006	9,535	5,342	12,218
2007	9,170	6,701	12,149
2008	5,472	4,992	7,465
2009	53	68	152

17. ábra A pivot eredménye

A pivot tábla szerkesztése után ha visszatérünk az eredmények fölé, ott láthatjuk, hogy az eredeti lekérdezésünk is megtalálható a lapon, valamint az új pivot eredménye is.

Korábban már említésre került, hogy lehetőség van arra az Answers-ben, hogy belefúrjunk az adatokba egy adott dimenzió mentén. Az előzőleg összerakott két eredményhalmaz valamelyikén kattintsunk az egyik évszámmra, és láthatjuk hogy a lekérdező tovább bontja számunkra az évet negyedévekre, ezáltal automatikusan felveszi a negyedév oszlopot is a lekérdezésekbe, valamint egy szűrőt tesz rájuk miszerint mi az adott évet kívánjuk csak látni az egész eredményhalmazból. Ugyanez elvégezhető a földrajzi helyekre is, ott a következő szemcsézettség a régió lesz. Valamint ez a tevékenység tovább folytatható, a régiók egyikére kattintva alrégiókat fogunk kapni. Ismét emlékezzünk vissza az OLAP kockáknál olvasottakra, ez a fajta tevékenység felel meg az OLAP kockában történő lefűrésnek.

		# of Opportunities		
Creation Year	Creation Quarter	Americas	Asia/RoW	Europe
2008	Q1-2008	1,717	2,115	2,442
	Q2-2008	1,425	1,645	2,026
	Q3-2008	1,177	763	1,548
	Q4-2008	673	949	1,449

18. ábra. A 2008-as évre történő szűrés valamint kibontás után

Ahhoz hogy visszatérjünk az eredeti lekérdezéseinkhez a Criteria (Kritérium) fülön el kell távolítani az automatikusan hozzáadódott szűrőfeltételt, valamint a negyedév oszlopot.

5.4 Grafikon hozzáadása

Végző lépésként adjunk még az analízálásunkhoz egy diagramot az eredmények fül Chart gomjára () kattintva! Ezzel eljutunk a grafikon vagy diagram szerkesztő nézetbe, ahol számtalan diagramtípus közül válogathatunk, valamint lehetőségünk van arra is, hogy megmondjuk melyik adatcsoport melyik részén szerepeljen a diagramnak.

Válasszuk ki a típusok közül a Step-et majd annak a 3D vagyis három dimenziós altípusát. Ezt követően, hogy az évek az X tengely mentén, az opportunity-k száma az Y tengely mentén jelenjenek meg, a kontinenseket pedig külön külön vonal reprezentálja.



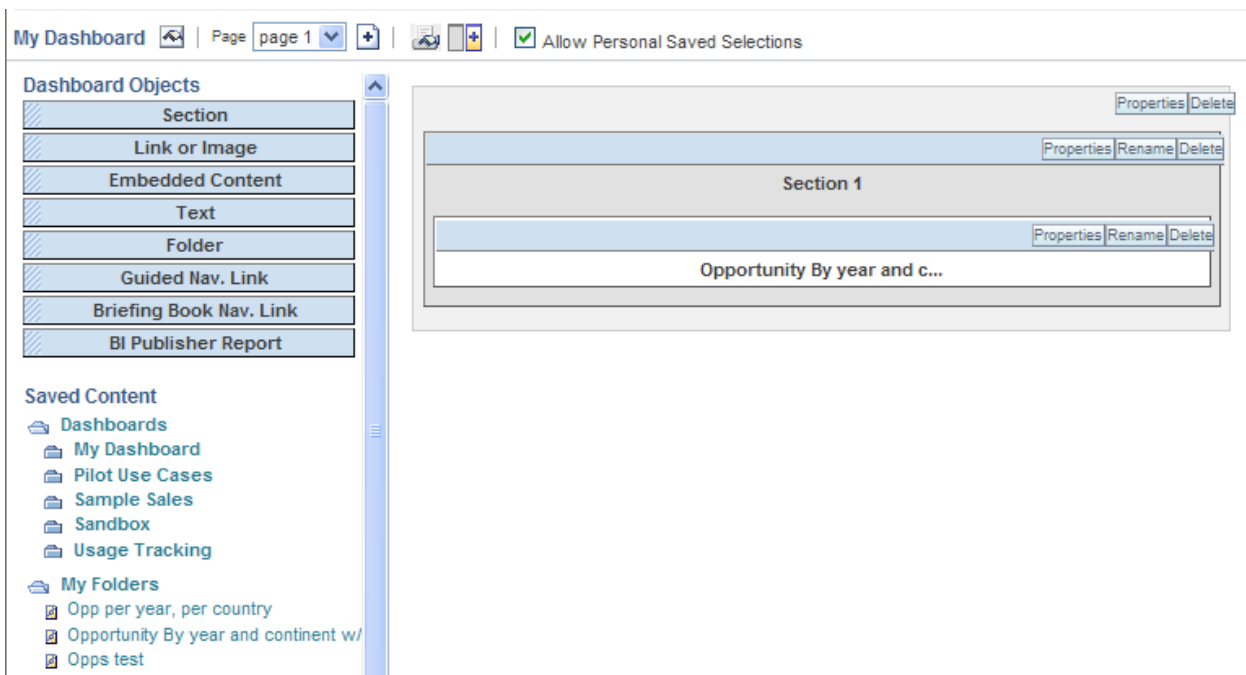
19. ábra Az Answers diagramszerkesztője

Ha ezzel megvagyunk akkor az Ok gombra kattintva véglegesíteni tudjuk a módosításainkat, és ezzel visszatérünk az eredmények fölére.

5.5 Megjelenítés a vezérlőpulton

Ezt követően elmenthetjük az analízis eredményeit, hogy később akár kitegyük egy vezérlőpultra. Mentsük tehát el az egész eredményhalmazt a menüsorban található lemez ikonra kattintva (📁)! A mentés folyamán meg kell adnunk egy úgynevezett könyvtárat ahová a lekérdezést mentjük valamint egy nevet is a későbbi azonosítás céljából, illetve rendelhetünk hozzá egy rövid leírást is.

A vezérlőpultunkra visszatérvén, A Page Options/Edit dashboard menüre kattintva az adott vezérlőpult szerkesztési nézetébe jutunk, ahol kirakhatjuk az imént elmentett lekérdezéseket, valamint más egyéb elemeket is.



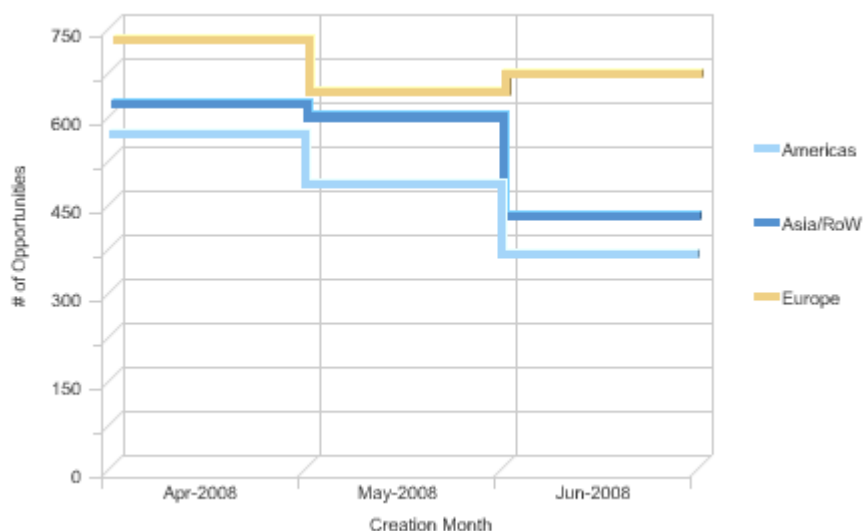
20. ábra Vezérlőpult szerkesztő

A különböző elemeket egyszerű „fogd-és-vidd” módszerrel a szerkesztő zónába visszük, ahol majd további beállításokat eszközölhetünk a Properties gombra kattintva.

Egy vezérlőpult egy vagy több oszlopból állhat, az oszlopokon pedig több szekció (section) is szerepelhet. Ezekebe a szekciókba lehet a különböző elemeket pakolni tetszés szerint. A fenti ábrán egy oszlop szerepel, egy szekcióval benne a saját készítésű analízisünkkel.

A vezérlőpult szerkesztésének végeztével a Save gombra kattintva menthetjük a változásokat.

Az adathalmazba fűrés a vezérlőpulton is működik, akár a grafikonon egyes értékekre kattintva, akár a táblázatok egyikében az évre vagy kontinensre. Itt már nem kell bajlódnunk a szűrőfeltételekkel, egyszerűen csak kattintsunk a Return linkre a lap alján hogyha vissza szeretnénk térni az eredeti képernyőre.



21. ábra Grafikon az adatokba fűrés után

5.6 Konklúzió

Láthattuk, hogy milyen egyszerű az Oracle BI eszközök segítségével lekérdezéseket összerakni, grafikonokkal megtámogatva. Érdekes azonban belegondolni, ahhoz hogy ilyen egyszerűen menjen egy lekérdezés megfogalmazása, illetve megjelenítése a vezérlőpulton milyen háttért igényel, mind adattárházak, mind üzleti intelligencia oldalon.

A példában bemutatottaktól jóval több beállítási, testreszabási lehetőségünk van, akár az Answers-ben készítünk grafikon, akár a vezérlőpultunkat szabjuk testre, amire a hely hiányában nem tudtam kitérni.

ÖSSZEFOGLALÁS

Az adattárházak és üzleti intelligencia alkalmazások kialakulásához az üzlet, az ipar határozta meg az igényt. Ezekre az igényekre az informatika tudomány az addig megszokott adatmodellezési módszerek helyett új koncepciókkal állt elő, mely elgondolások szerint az adatokat integráltan egy a normál napi használatú rendszertől eltérő helyen kell tárolni és onnan kell elemezni. Az eltérő adattárolás és modellezés kapcsán megismerkedhettünk a csillag és hópehely séma ötletével, valamint az OLAP kocka fogalmával ami egy teljesen új koncepciót hoz az adattárház világba, és szakít az eddigi relációs tárolás hagyományaival.

Az adatbetöltés terén ETL és ELT folyamatokkal találkozhattunk, megvizsgáltuk a köztük lévő különbségeket, valamint a hozzájuk kapcsolódó eszközökről volt szó. A két fogalom bár mindkettő ugyanarra jó, az ELT egyre elterjedtebb, hiszen egyre kevesebb idő marad az adatbetöltésre. Láthattuk továbbá, hogy általánosan vett adattárház architektúra nincsen, mindig az üzleti igények, és az adott szervezet felépítése, elképzelése szabja meg az adattárház architektúra jellegét is, de felvázoltunk egy lehetséges adattárház struktúrát, valamint szó esett az adatpiacok létéről és arról, hogy többnyire szerepük az, hogy egy-egy részterületet fedjenek le az adattárház teljes halmazából.

Az eltérő adatmodellekből kifolyólag az addig általános lekérdezést gyorsító technológiák, és technikák adattárház környezetben nem hoznak nagy előrelépést, így újakra volt szükség, amelyek közül néhány fontosat áttekintettünk a szakdolgozatban:

- Particionálás, amely segíti a karbantarthatóságot, valamint némely mértékben a lekérdezések optimalizálását is, valamint szót kerítettünk a particionálás típusaira, úgy mint hash, list, és range, valamint a kombinált partíciók
- Bitérképes indexek, valamint bitmap join indexek, amelyek a lekérdezéseket gyorsíthatják jelentős módon. Láthattuk hogyan működik a hagyományos B-fa alapú index, majd megnéztük a bittérképes index működési elvét is. Majd egy példán keresztül igazoltuk a bittérképes indexek előnyét a hagyományos indexekkel szemben egy adott szituációban.
- Materializált nézetek, amik az adattárházból való elemzés sarokkövét adják, és áttekintettük a főbb materializált nézet csoportokat

- Parallel futtatás, amely segítheti nemcsak a lekérdezések, hanem más adatmanipulációs műveletek lefutását is

Az adattárházak adják az egyik forrását az üzleti intelligencia architektúrának, amit az Oracle kínál számunkra. Az adattárházakon kívül még támaszkodhat közvetlenül más forrásrendszerekre is.

Erre a rétegre épül rá az Oracle BI szerver amely tartalmaz egy metaadat kezelő részt is. Ez a metaadat kezelés kiterjed az egész rendszerre, célja, hogy az üzleti fogalmakat a felhasználó a forrásrendszerektől függetlenül tudja látni. A metaadat kezelés 3 rétegű a BI szerverben, egy fizikai réteg létesít kapcsolatot a BI rendszer forrásrendszereivel, erre épül a logikai csillagsémát előállító logikai réteg, majd a legtetején található a prezentációs réteg amely a felhasználónak szánt üzleti fogalmakat tartalmazza magában.

Az Oracle többféle üzleti intelligencia csomagot kínál, megnéztük ebből a nagyvállalati verziót, valamint a kisebb cégeknek szánt Standard Edition-t. Ezen kívül még többféle variációban lehet az üzleti intelligencia termékekhez hozzájutni.

A BI Suite része az Interactive Dashboards, az Oracle Answers, és a BI Publisher, valamint a nagyvállalati verzió még kiegészül további alkalmazásokkal.

Az Interactive Dashboards egy vezérlőpult alkalmazás, és az Answers-hez hasonlóan teljesen webes felületű. Az Answers ad-hoc lekérdező eszköz által elkészített kimutatásokat, jelentéseket, elemzéseket kitehetjük a vezérlőpultokra, vagy akár publikálhatjuk mások számára is. Így természetesnek vehetjük, hogy Dashboards tartalmaz jogosultságkezelést, illetve csoportkezelést is.

A Dashboards és az Answers hatékonyságát, és gyorsaságát egy példán keresztül is igazoltuk, példa lekérdezésünkben egy szervezet opportunity-ainak (vásárlási esély) idő és térbeli eloszlását figyeltük meg különböző táblázatokon, valamint grafikonon. A példában egy alapvető elemzést tekintettünk meg, ellenben azt is megállapítottuk, hogy számtalan más beállítás, vagy további lehetőségek rejlenek még az Answers-ben valamint a Dashboards-ban.

A szakdolgozat szűk keretei miatt nem jutott sok hely az Oracle Essbase tárgyalására, ami egy OLAP szerver, és könnyedén lehet vele EPM alkalmazásokat fejleszteni.

A szakdolgozatomban igyekeztem egy áttekintést adni abból a hatalmas témakörből amit az adattárházak és az üzleti intelligencia tesz ki, valamint igyekeztem technikai oldalról, aztán üzleti oldalról is megközelíteni. Remélem a munkám elnyerte tetszését, és segítségére vált az üzleti döntéstámogatás alapvető megismerésében.

IRODALOMJEGYZÉK

- Dr. Bognár Katalin: Tudásalapú rendszerek, szeminárium jegyzet
- Lilian Hobbs, Susan Hillson, Shilpa Lawande, Pete Smith: Oracle Database 10g Data Warehousing, Elsevier Digital Press, ISBN: 1-55558-322-9
- Wikipédia: Extract, transform, load szócikk
- Fekete Zoltán: Oracle Data Integrator – ELT, beyond ETL szeminárium, Oracle Junior képzés, 2008
- Czinkóczki László: Using the Oracle Warehouse Builder szeminárium, Oracle Junior képzés, 2008
- Wikipédia: OLAP szócikk
- Kerepes Tamás: Adattárházakat támogató adatbázis technológiák szeminárium az Oracle Junior képzés keretében, 2008
- Oracle9i Database Concepts, hivatalos Oracle dokumentáció, A96524-01
- Oracle Database Data Warehousing Guide, hivatalos Oracle dokumentáció, B28313-02
- H. P. Luhn: A Business Intelligence System (IBM Journal)
- D. J. Power: A Brief History of Decision Support Systems
- Dr. Drótos György: Informatikai stratégia, 2003
- Fekete Zoltán: Oracle üzleti intelligencia és adattárház szeminárium az Oracle Junior képzés keretében, 2008
- David Wade, Ronald Recardo: Corporate Performance Management. Butterworth-Heinemann, 2001 ISBN 0-87719-386-X
- Dale Skeen, Boris Evelson: Next-Generation BI web seminar