INTELLIGENS RENDSZEREK II. (ÜZLETI INTELLIGENCIA) LABORATÓRIUM

Adattárházak elemzése és megjelenítése

4. laborgyakorlat feladatai

Készítette:
Kovács Dániel László
(dkovacs@mit.bme.hu)

Méréstechnika és Információs Rendszerek Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem

2008, november.

1. Áttekintés

A gyakorlat célja: Megismertetni a hallgatókat az adattárházak analízisének (OLAP) és vizualizációjának (riporting) alapjaival.

A gyakorlat tartalma: A gyakorlat során a hallgatók egy adattárházból kiindulva adatkockát építenek OLAP elemzés céljából. Az elemzést optimalizálják, majd pedig web-alkalmazásként hozzáférhető, automatikusan frissülő, interaktív on-line riportokat, grafikonokat, és statisztikákat generálnak. A gyakorlat a következő főbb elemekből fog állni:

- OLAP adatprojekt létrehozása
- Adatmodell áttekintése (csillag/hópihe séma)
- OLAP meta-adatok meghatározása (dimenziók, attribútumok, mértékek, szintek, hierarchiák)
- Adatkocka létrehozása, és validálása
- Mértékek tulajdonságainak finomhangolása
- OLAP meta-adatok importálása XML-ből
- Materializált nézetek (MQT-k) létrehozása, közzététele, és alkalmazása
- Adatvizualizáció adatforrásainak megadása
- Alphablox web-alkalmazások létrehozása, közzététele, és módosítása
- Adatkockák vizualizációja
- Alphablox web-alkalmazás JSP kódjának megismerése, és ilyen módon a megjelenítés módosítása
- MDX nyelvű Alphablox-os adatkocka-lekérdezések kialakítása és módosítása

2. Kritériumok

A gyakorlat során a hallgatóknak 3,5 órányi tiszta munkaidő áll a rendelkezésére. Ezalatt az idő alatt kell teljesíteniük a következő szakaszban felsorolt feladatokat. A gyakorlat során végzett munka a foglalkozás végén kerül megítélésre (teljesítmény, és minőség tekintetében). A jobb megítélés előfeltétele, hogy a hallgató koncepcionálisan is minél jobban átlássa/értse, hogy a gyakorlat során konkrétan mivel foglalkozott.

3. Feladatok

1. Indítsuk el a „`C:\VMWare-images\oktatas\VIMIA430_-_Intelligens_Rendszerek_2_labor_-_DW350ERC4_-_4th_exercise`” könyvtárban található „`4th exercise.vmx`” virtuális gép „`MeresElott`” nevű snapshot-ját (**dupla kattintás a VMX fájlra**, **VM → Snapshot → MeresElott → Yes, Start this virtual machine, View → Full Screen**).

2. A virtuális gépen lépjünk be a Windows-ba „`db2admin`” felhasználóként, „`ibm2blue`” jelszóval (és várjuk meg, amíg minden rendesen betöltődik).

A következőkben a **Cube Views** nevezetű eszközzel fogunk megismerkedni, amely az *IBM DB2 Data Warehouse Edition* egy olyan hasznos kiegészítése, amely lehetővé teszi megfelelő relációs adatbázisban tárolt adatok **OLAP** (**O**n-**L**ine **A**nalytical **P**rocessing) alapon történő kezelését, és elemzését. Először is egy klasszikus **csillag (hópihe) séma**hoz hozunk létre megfelelő OLAP meta-adatokat, amiket tovább bővítünk úgynevezett **mértékek (measures)** bevezetésével, majd definiáljuk az OLAP meta-adatokhoz tartozó **adatkocka dimenzióinak hierarchiáját**, ami meghatározott **szintekre (levels)** fog épülni. Lényegében tehát egy OLAP adatkockák fogunk létrehozni, amelynek meta-leírását végül kiexportáljuk egy Cube Views-ra felkészített, szokványos relációs adatbázisba.

3. Indítsuk el a „Design Studio”-t az asztalon, és miután elindult, hozzunk létre egy új Data Design Project-et: **File → New → Data Design Project (OLAP)**

4. A projekt neve legyen: **DW350 CV Data Project**. Kattintsunk a **Finish** gombra. A megfelelő adattervezési projekt ennek hatására máris megjelenik a Data Project Explorer-ben (baloldalt fent).

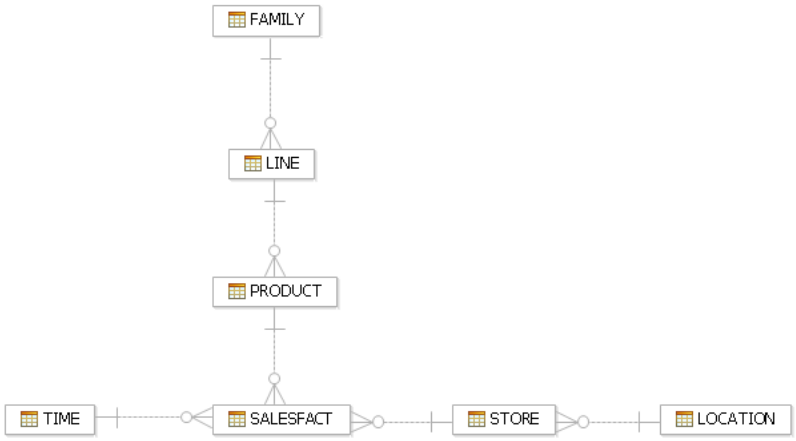
5. Most csatlakozzunk a CVSAMPLE nevű adatbázishoz, ahol lesz egy csillagséma, amit adattervezési projektünk során modellezni fogunk. A **Database Explorer**-ben (baloldalt lent) bontsuk ki a **Connections** fület, és kattintsunk jobb egérgombbal a CVSAMPLE adatbázisra, majd válasszuk a **Reconnect** opciót (`db2admin/ibm2blue`).

6. Miután létrejött a kapcsolat, bontsuk ki, és tekintsük meg a DB2ADMIN séma adatbázis tábláit a Database Explorer-ben (**CVSAMPLE → CVSAMPLE → Schemas → DB2ADMIN → Tables**), majd jelöljük ki mind a 7, azaz hét táblát (pl. rákattintva a legfelsőre a bal egérgombbal, majd a legalsóra is úgy, hogy közben még a Shift gombot is nyomjuk). Ezek után – a bal egérgombot nyomva tartva – mozgassuk át a kijelölt táblákat a Data Project Explorer-ben látható „DW350 CV Data Project” projektünk alatti **Data Models** részbe.

7. Ennek hatására a Data Models részben létrejön egy „visszafejtett” **CVSAMPLE.dbm** modell. Fúrjunk le ezen belül a DB2ADMIN sémáig (**CVSAMPLE.dbm → CVSAMPLE → DB2ADMIN**). A DB2ADMIN séma alatt pedig kattintsunk jobb egérgombbal a **Diagrams**-ra, és válasszuk a **New Overview Diagram** opciót.

8. Kattintsuk ki az **Infer implicit relationships** opciót, kattintsuk viszont be a **DB2ADMIN** jelölődobozt, és **OK**.

Ennek hatására létrejön a diagram, azonban első ránézésre talán nem a legtökéletesebb elrendezésben. Így, ha kedvünk tartja, esetleg **rendezzük át** a táblák és a köztük lévő kapcsolatok pozícióját úgy, hogy jobban át tudjuk tekinteni.



Látható, hogy a diagramon az elsődleges (primary) és külső (foreign) kulcsok is megjelennek. Valójában egy viszonylag egyszerű hópihe sémával van dolgunk, amelynek középpontja a SALESFACT tábla, avagy a **ténytábla**. Ebben vannak lényegében az adatok. A SALESFACT tábla külső kulcsokkal hivatkozik más, ún. **dimenziótáblák** elsődleges kulcsokkal azonosított adataira, így téve teljessé az általa tényként adott adatokat. A SALESFACT közvetlenül a TIME, a PRODUCT, és a STORE dimenziótáblákra hivatkozik (a megfelelő azonosítókkal). Viszont a PRODUCT, mint egyfajta al-ténytábla, ugyancsak tovább hivatkozik egy al-dimenziótáblára, a LINE-ra, amely pedig hasonlóképp a FAMILY-re. Sőt, a STORE is tovább hivatkozik a LOCATION táblára. Így alakul ki tehát esetünkben a hópihe-séma. Lényegében tehát a TIME az *egyik* dimenzió, a PRODUCT, a LINE, és a FAMILY együttesen a *másik* (pl. Product nevű) dimenzió, és a STORE, és a LOCATION együttesen a *harmadik* (pl. Market nevű) dimenzió. Az adott dimenzióhoz tartozó táblákban szereplő adatok együttesen nyilván az adott **dimenzió elemei** (esetünkben a lehetséges időpontok, termékek, és üzletek).

9. Most pedig hozzuk létre az előbbi hópihe-sémának megfelelő adatkocka modelljét a Cube Views segítségével! A Data Project Explorer-ben, a DB2ADMIN séma alatt, a Diagrams mellett bontsuk ki az **OLAP objects**-et. Ezen belül kattintsunk jobb gombbal a **Cube Models**-re, és válasszuk az **Add Cube Model – Quick Start** opciót.

10. Kattintsunk a **Next**-re. Most válasszuk ki a ténytáblánkat! Bontsuk ki a **DB2ADMIN**-t, jelöljük ki a **SALESFACT**-et, és **Next**.

11. Itt láthatjuk, hogy lenne egy SALESFACT nevezetű adatkocka modellünk. Bontsuk is ki! Ezen belül láthatjuk a modellhez tartozó táblákat, és egyéb adatokat. Bontsuk ki a SALESFACT táblát, és azon belül vessünk egy pillantást a **Measures** és **Attributes** részre. Mindkettőt automatikusan határozta meg számunkra a rendszer.

A SALESFACT tábla ugyebár 6, azaz hat oszlopból áll. Ebből 3, azaz három külső kulcs (hivatkozik a TIME, PRODUCT, és STORE táblák elsődleges kulcsaira) – ezekből lettek a **tények attribútumai (attributes)**, míg a maradék három oszlop, ami se nem elsődleges, se nem külső kulcs (esetünkben szerencsére decimális szám) lett **tényekre vonatkozó mérték**

(measure).[1]. Az előbbiek segítségével (konkrét behelyettesítésével) azonosítható tehát 1-1 **tény (fact)**, míg az utóbbiak 1-1 ilyen konkrét tény „mértékeit" adják meg (pl. hogy hány darab lett eladva az adott termékből az adott időben az adott üzletben).

12. Az adatkocka-modell létrehozásának befejezéséhez kattintsunk a **Finish**-re. Ennek hatására a Data Project Explorer-ben felépül az adatkocka modellje. Bontsuk ki ezen belül (ha magától még nem bomlott ki) a SALESFACT ténytábla mértékeit (**…→Cube Models → SALESFACT → SALESFACT → Measures**).

13. Itt újra láthatjuk az előbbi 3 oszlopot, amely a tényekről ad további (többnyire numerikus) jellemzést. Tegyük fel, hogy ez számunkra momentán kevés, és szeretnénk további mértékeket is definiálni. Ehhez kattintsunk a jobb egérgombbal a **Measures**-re, és válasszuk az **Add Calculated Measure** opciót.

Ezzel tehát nem módosítjuk a ténytábla struktúráját, csak újabb, származtatott meta-adatokat helyezünk el mellé (az OLAP meta-adatokon belül).

14. Legyen első származtatott mértékünk az üzlet szempontjából egyik legfontosabb adat, a profit! A **Data → CVSAMPLE → Columns → SALESFACT** részben, és az **Operators** részben kattintgatva hozzuk létre a következő SQL kifejezést (**Expression**):

```
@Column(DB2ADMIN.SALESFACT.SALES) - @Column(DB2ADMIN.SALESFACT.COGS) - @Column(DB2ADMIN.SALESFACT.ADVERTISING)
```

15. Most **Validate**, **OK**, és **OK**. Ennek hatására létre is jön a Data Project Explorer-ben a megfelelő helyen az új, származtatott mértékünk. Ráadásul ki is van jelölve, így hát mielőtt bármi egyebet tennénk, adjunk neki gyorsan nevet: **Profit**, majd **ENTER**.

16. Ezek után hozzunk létre hasonlóképp még egy származtatott mértéket **ProfitMargin** néven. Ennek SQL kifejezése a következő legyen:

```
DECIMAL(FLOAT(@Measure(DB2ADMIN.Profit))/FLOAT(@Measure(DB2ADMIN.SALES(SALESFACT)))*100,31,2)
```

Ez egy fokkal komplikáltabb, hiszen itt immár függvényeket is használunk, amik ráadásul egymásba ágyazottan jelennek meg (*ügyeljünk tehát a zárójelekre!!!*). Nagy segítséget jelenthet, ha észrevesszük, hogy ott jön létre dupla-kattintás után az Expression-ben a kívánt elem, ahol éppen a kurzor volt. A fenti kifejezés lényegében a Profit, és a SALES mértékek lebegőpontossá konvertált változatainak valós hányadosa decimális formára konvertálva.

17. Miután a Data Project Explorer-ben létrejött az új **ProfitMargin** mérték, válasszuk ki, és – mivel ez a mérték százalékos részaránytjelöl, amit nincs értelme aggregálni – a **Properties** részben (lent), azon belül az **Aggregations** fülben a **Function for all of the dimensions**-t állítsuk **None**-ra.

18. Nézzük meg, hogy a Profit-nál ez a beállítás mi volt, majd mentsük eddigi munkánkat (**File → Save All**).

---

[1] Gondoljunk csak a következő logikai analógiára: a SALESFACT tábla felel meg a logikai predikátum nevének, aminek argumentumai a TIME, PRODUCT, és STORE – most attribútumok. A konkrét tények a predikátum argumentumainak behelyettesítése révén állnak elő: adott idő, termék, és üzlet azonosító (atom) mellett. Egy-egy ilyen konkrét tényhez (literálhoz) pedig bizonyos ismérvek is tartozhatnak. Ezeket nevezzük most mértékeknek.

19. A következőkben dimenzió-hierarchiákat fogunk kialakítani, azaz **szinteket (levels)** a dimenziókon belül – kezdjük a STORE dimenzióval! Ehhez először is bontsuk ki a STORE dimenzió attribútumait a Data Project Explorer-ben: **…→Cube Models → SALESFACT → STORE → STORE → Attributes**

Itt megfigyelhetjük, hogy a rendszer a „STORE dimenziót" (aminek az a 3-tengelyes kis koordinátarendszer az ikonja) lényegében a STORE és a LOCATION dimenziótáblákból állította elő. A dimenzió bizonyos attribútumai az egyik, mások pedig a másik táblából valók.
(Megjegyzés: ahogyan előbb származtatott mértékeket hoztunk létre, úgy itt most akár származtatott attribútumokat is létrehozhatnánk...)

20. Kattintsunk most jobb gombbal a STORE dimenzió attribútumaival egy szinten lévő **Levels** fiókra, és válasszuk az **Add Level** opciót. A szint neve legyen **RegionLev**.

21. A RegionLev szint **Properties**-ében (lent) válasszuk ki a **Level Key** fület, itt kattintsunk a **kis zöld plusz jelre**, és válasszuk a **REGION_NAME** attribútumot, **OK**.

22. A **Related** fülön belül hasonlóképp válasszuk ki a **REGION_NAME**, és **REGION_DIRECTOR** attribútumokat, majd **OK**. Bizonyosodjunk meg arról, hogy a REGION_NAME a **Default** kapcsolódó attribútum.

23. Az **Optimize** fülön belül kattintsuk be a **Functional Dependency** jelölődobozt (aminek hatására a Level Key és a kapcsolódó attribútumok között függvény kapcsolat definiálódik). Ezt azt jelenti, hogy a Level Key attribútumot egyértelműen meghatározza a Related-ben adott attribútumok értéke.

24. Végül a **Type** fülön belül bizonyosodjunk meg arról, hogy a szint típusa **Regular**.

Ezzel tehát létrehoztunk a STORE dimenzión belül egy szintet, amely a **régiókat (region)** azonosítja. A régiókon belül azonban (az USA-ra gondolva, pl.) több **állam (state)** is lehet, azon belül több **város (city)**, azon belül pedig több **üzlet (store)**. Jó volna tehát még tovább tagolni a STORE dimenziót (ha majd részletekbe menő riportokat szeretnénk a későbbiekben kapni). A következőkben ennek megfelelően még 3 szintet hozunk létre (**StateLev**, **CityLev**, és **StoreLev**), és a meglévő szinteket nyomban (dimenzió)hierarchiába szervezzük.

25. Az előbbiekhez hasonlóan tehát hozzunk most létre egy **StateLev** nevű szintet. A **Level Key** legyen **REGION_NAME**, és **STATE_NAME** (mivel a „gép" szerint elviekben ugyanaz a State előfordulhatna más Region-ben is (ahogyan például ugyanaz a hónap előfordul minden évben), ezért kell még a Region-t is megadni itt). A kapcsolódó attribútumok legyenek a **STATE_NAME**, **STATE_DIRECTOR**, és a **STATE_POPULATION**, amik közül a STATE_NAME legyen az alapértelmezett (**Default**). Az **Optimize** fülön belül most ne kattintsuk be a funkcionális függést (hiszen egy állam lakosságának számossága nem határozza meg egyértelműen azt, hogy melyik államról van szó).

26. Hozzunk létre még egy szintet **CityLev** néven. A **Level Key** legyen **POSTALCODEID (STORE)**. Kapcsolódó attribútum legyen a **CITY_NAME**, és a **CITY_POPULATION**, amiből az előbbi a **Default**.

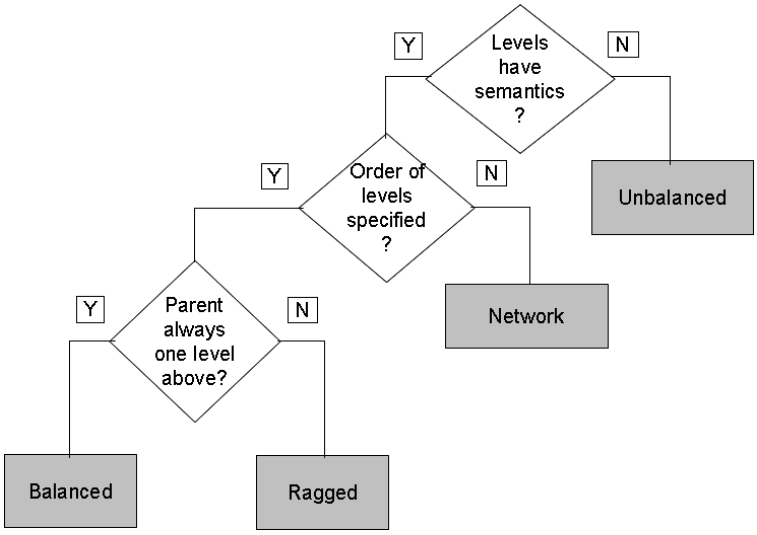
27. Az utolsó, negyedik szint a STORE dimenzión belül a **StoreLev** nevet viselje. A **Level Key** legyen a **REGION_NAME**, **STATE_NAME**, **POSTALCODEID (STORE)**, és **STOREID** (a sorrend nem számít). A kapcsolódó attribútum a **STORE_NAME** legyen, ami egyben alapértelmezett is.

28. Most pedig szervezzük az előbb létrehozott szinteket hierarchiába! Kattintsunk jobb egérgombbal a Data Project Explorer-ben a STORE dimenzión belül az Attributes-szal, és Levels-el egy szinten lévő **Hierarchies** fiókra, és válasszuk az **Add Hierarchy** opciót. A hierarchia neve legyen **Geography**.

29. Az újonnan létrehozott hierarchiánk **Properties**-ében (lent) a **Levels** fül alatt kattintsuk be az **All Level** jelölődobozt, és adjuk neki az **All Regions** nevet. Ezzel tehát létrehoztunk egy szintet, amely minden szintet magába foglal (az összes halmaz halmaza – a „Nagy Omega”).

30. Ezek után kattintsunk a **kis zöld plusz jelre**, majd a **Select All** gombra, és **OK**. *(Megjegyzés: mivel a szinteket megfelelő, absztrakció szerint csökkenő sorrendben hoztuk létre, azaz a legáltalánosabbtól a legspeciálisabb felé haladva (régió → állam → város → üzlet), ezért ez a sorrend megfelelő lesz most. Amennyiben esetleg valamiért nem ez lenne a sorrend, rendezzük át a szinteket ennek megfelelően a listában. A legfelső a legáltalánosabb, a legalsó pedig a legspeciálisabb szint.)*

31. A **Type/Deployment** fülben a **Balanced/Standard** opciót válasszuk. A *miért*-re a választ lényegében az alábbi ábrán láthatjuk. Azért kiegyensúlyozott jelen dimenzió-hierarchiánk, mivel vele kapcsolatban mindhárom alábbi kérdésre igenlő válasz adható. A hierarchiában szereplő egyes szinteknek ugyanis valódi jelentése van, a szintek sorrendje fix, és minden egyes szinten belül az **elemeknek (members)** mindig van egy szülő-eleme a hierarchia rákövetkező szintjén (pl. esetünkben minden üzletnek van városa, minden városnak van állama, és minden államnak van régiója).



32. Az **Optimize** fülön belül kattintsuk még be a **Functional dependency**-t (hiszen a szintek elemei meghatározzák egymást, mivel a szintek egymásba vannak ágyazva).

33. A következőkben az alábbi táblázatnak megfelelően a PRODUCT dimenzióhoz is hozzunk létre szinteket, és hierarchiát.

Szint neve	Level Key	Default attribútum	Related attribútumok
FamilyLev	FAMILYID (FAMILY)	FAMILY_NAME	
LineLev	LINEID (LINE)	LINE_NAME	
ProductLev	PRODUCTID	PRODUCT_NAME	PRODUCT_DESCRIPTION

A szinteket a táblázat szerint, fentről lefelé haladva hozzuk létre, egymás után, épp úgy, ahogyan majd a hierarchiában is – aminek a neve legyen **Product** – szerepelniük kell. Minden szint esetében definiáljuk a **funkcionális függőség**et (a Level Key és a kapcsolódó attribútumok között). Az **alapértelmezett attribútumok egyben kapcsolódóak is** legyenek. – A hierarchián belül legyen egy **All Level** szint, aminek a neve **All Products**. A hierarchia típusa legyen **Balanced**, és érvényesüljön benne a **Functional dependency**.

34. Térjünk most rá a SALESFACT utolsó, harmadik dimenziójára, a TIME-ra. Ha kibontjuk a TIME dimenzió attribútumait (**…→Cube Models → SALESFACT → TIME → TIME → Attributes**), láthatjuk, hogy a negyedév (quarter) kapcsán csak egy szám szerepel, a QUARTER_NUMBER. Hozzunk tehát létre egy ennél beszédesebb (származtatott) attribútumot. Kattintsunk a jobb egérgombbal az **Attributes**-ra, és válasszuk az **Add Calculated Attribute** opciót.

35. Itt írjuk be a következő SQL kifejezést, **validáljuk**, és ha rendben van, akkor kattintsunk az **OK**-ra. Az attribútum neve **QuarterName** legyen.

```
CONCAT('Q', CHAR(@Column(DB2ADMIN.TIME.QUARTER_NUMBER)))
```

36. A következőkben, az alábbi táblázatra támaszkodva, az eddigiekhez hasonlóan hozzuk létre a TIME dimenzió hierarchiájához szükséges szinteket.

Szint neve	Level Key	Default attribútum	Típus - Time or Date
YearLev	YEAR	YEAR	Years
QuarterLev	YEAR, QUARTER_NUMBER	QuarterName	Quarters
MonthLev	YEAR, MONTH_NUMBER	MONTH_NAME	Months
DayLev	YEAR, DAY_OF_YEAR	DAY_NAME	Days

A szintek definiálásának sorrendje újfent a táblázatban való sorrendjük legyen. Vegyük figyelembe, hogy bizonyos szinteknél a Level Key-ben több attribútum is szerepel (a dimenzió-elemek egyediségének biztosítása érdekében). A Default attribútumok egyben kapcsolódóak is, nyilván. Arra is ügyeljünk, hogy a szintek **Properties**-részében, a **Type** fülnél a szint típusát Regular-ról **Time or Date**-re állítsuk, és azon belül kiválasszuk a megfelelő értéket. A **Functional dependency**-t csak a **MonthLev** és **DayLev** szintek esetében kapcsoljuk be. A **MonthLev** esetén továbbá az **Order** fülben még azt is állítsuk be, hogy a **MONTH_NUMBER** oszlop szerint rendezze az értékeket (mivel bizonyos alkalmazások e nélkül – akár alapértelmezésben – ABC sorrend szerint rendeznék őket).

37. Most pedig rendezzük hierarchiába a létrehozott szinteket! A hierarchia neve **Time** legyen. Legyen egy **All Level**, aminek **All Years** a neve. A hierarchia típusa **Balanced** legyen, és a szintek közt álljon fent **funkcionális függőség**.

Most pedig, miután rendelkezésünkre áll a hópihe-séma, hozzá az adatkocka modell, benne a mértékekkel, dimenziókkal, és azon belül a szintekkel, és azok hierarchiájával, létrehozhatjuk – a modell alapján – az első adatkockánkat.

38. A Data Project Explorer részben, a SALESFACT adatkocka modell alatt kattintsunk jobb gombbal a **Cubes**-ra (**…→Cube Models → SALESFACT → Cubes**), válasszuk az **Add Cube** opciót, és adjuk az adatkockának a **SalesCube** nevet.

39. A kocka **Properties** részében kattintsunk a **Dimensions** fülre, és ott adjuk hozzá a **STORE**, **PRODUCT**, és **TIME** dimenziókat.

40. A Data Project Explorer-ben, a SalesCube adatkocka alatt válasszuk ki a **Cube Facts (SalesCube)** fiókot, és a **Properties** részben, a **Measures** fülben adjuk hozzá az összes mértékünket (3+2=5 darab). Bizonyosodjunk meg arról, hogy ezek közül a **SALES** lett a **Default** mérték.

41. Most pedig a Data Project Explorer-ben válasszuk ki a SalesCube adatkocka **STORE (SalesCube)** kocka-dimenzióját (**…→Cube Models → SALESFACT → Cubes → SalesCube → STORE (SalesCube)**), és a **Properties** részben kattintsunk a **Cube Hierarchy** fülre. Itt láthatjuk az adatkocka adott dimenziójához kiválasztott dimenzió-hierarchiát. Ne feledjük: egy adott dimenzióhoz több hierarchia is tartozhat, viszont egy-egy adatkocka esetében ezek közül csak az egyik lehet érvényben.

42. Ha több hierarchiát is definiáltunk volna a STORE dimenzióhoz, akkor most a „…” gombra kattintva választhatnánk ki a megfelelőt. Hasonlóképp vessük egy pillantást SalesCube adatkockánk **PRODUCT (SalesCube)**, és **TIME (SalesCube)** dimenziójának hierarchiájára is.

43. Mentsük eddigi munkánkat (**File → Save All**).

Most pedig, miután végre létrehoztunk egy adatkockát az adatkocka modellje alapján, ellenőrizzük le, hogy megfelelően jártunk-e el – **validáljuk a SalesCube**-ot!

44. Kattintsunk jobb egérgombbal a **SALESFACT** adatkocka modellre (közvetlenül a Cube Models alatt, ahol az ikon egy kis kék adatkocka), és válasszuk az **Analyze Model** opciót. Minden **szabályt (rules)** hagyjunk kiválasztva. Esetleg fussuk át őket, hogy lássuk, nagyjából miknek kell eleget tennie egy-egy ilyen adatkocka modellnek, majd bárminemű változtatás nélkül kattintsunk a **Finish** gombra. Amennyiben a Console-on megjelenő üzenet semmiféle hibát nem jelez, folytassuk a következő lépésnél, egyébként javítsuk a hibát, és próbálkozzunk újra.

45. Ha adatkocka modellünk megfelelőnek bizonyult, váltsuk valóra! Magyarán tegyük közzé (deploy-oljuk) a SALESFACT adatkocka modellt egy megfelelő, erre előre felkészített adatbázisba. Emlékezzünk vissza, hogy az előző gyakorlaton a CVSAMPLE adatbázist engedélyeztük a Cube Views számára. Kattintsunk most tehát jobb gombbal a **SALESFACT** adatkocka modellre, válasszuk a **Deploy to Database** opciót, válasszuk ki a **CVSAMPLE** adatbázist, és kattintsunk a **Finish**-re, majd **OK**.

46. Most csiszoljuk kicsit tovább a modellünket! Válasszuk ki a SALESFACT adatkocka modell alatt a **SALESFACT** ténytáblát (**…→Cube Models → SALESFACT →**

SALESFACT), és a **Properties**-ben, a **Measures** fülön belül vegyük szemügyre a mértékeinket.

47. Válasszuk ki az **ADVERTISING**-et (konkrétan a mérték nevére kattintva a bal egérgombbal), és kattintsunk felette a nagy szigma (Σ) ikonra, azaz az **Edit aggregations**-re.

48. Itt láthatjuk (a jobb oldali, Aggregation script nevű panelben), hogy az aktuális aggregáló script az ADVERTISING-et a STORE, PRODUCT, és TIME felett összegzi (SUM). Összegezzük csak a TIME felett! Ehhez kattintsunk a jobb oldali panelben a **SUM**-ra, és a **kis sárga lefelé nyíllal** vigyük közvetlenül a **TIME** fölé.

49. Bővítsük még tovább ezt a script-et! A bal oldali panelben (Column functions) kattintsunk duplán az **AVG**-re, és miután megjelent a jobboldali panelben, vigyük közvetlenül a **STORE** fölé, majd **Validate**, és **OK**.

Lépjünk most még egyel tovább, és – az eddigi SUM, és AVG helyett – hozzunk létre a származtatott mértékek felett komplex aggregáló függvényeket! Próbáljuk meg nyakon csípni az eladások (sales) és a reklám (advertising) közötti korrelációt!

50. Továbbra is a SALESFACT ténytábla (**…→Cube Models → SALESFACT → SALESFACT**) Properties részének Measures fülében maradva, kattintsunk a lista felett a nagy szigmától kettővel balra lévő, **Create calculated measure** ikonra. A Data panelben kattintsunk duplán a **SALES** oszlopra (**CVSAMPLE → SALESFACT → SALES**), hogy bekerüljön az Expression részbe, majd nyomjuk meg az **OK**-t.

51. Látható, hogy létrejött egy újabb mérték **Measure** néven. Válasszuk ki (magát a nevet), és írjuk át **Correlation**-re, majd kattintsunk újra az **Edit aggregations** ikonra (Σ). A feljövő ablakban, a Column functions panel feletti legördülő listából válasszuk ki a **Multiparamer functions**-t, majd alább a **CORRELATION** függvényt (pusztán egy kattintással, csak válasszuk ki), és kattintsunk alatta az **Add to script** gombra.

52. A Second parameter részben a **Use an existing measure** legyen kiválasztva. A hozzá tartozó legördülő listából válasszuk az **ADVERTISING (SALESFACT)**-et, és kattintsunk az **OK**-ra. A jobb oldali panelből (az aggregáló script-ből) töröljük ki a SUM-ot: válasszuk ki, és kattintsunk a **kis piros X**-re, majd **validáljuk** a script-et, és **OK**.

Most pedig, miután létrehoztuk azt a mértéket, amely a korrelációt jelzi, hozzunk létre egy újabb adatkockát (a SALESFACT adatkocka modell alapján), amelyben immár a SALES és ADVERTISING közti korrelációt is nyomon tudjuk követni!

53. Hozzunk létre tehát a SALESFACT adatkocka modellhez tartozóan egy **TestCorr** nevű adatkockát (a SalesCube adatkocka mellé), **adjuk hozzá** mindhárom dimenziót, majd minden dimenzióinál **rendre ellenőrizzük**, hogy a megfelelő hierarchia van-e kiválasztva (alapértelmezésben), illetve a kocka által vizsgált mértékeknek (**Cube Facts**) a **SALES**-t, az **ADVERTISING**-et, és a **Correlation**-t válasszuk (amiből a SALES legyen **Default**). Tehát ezek a mértékek fognak megjelenni az adatkockán belül, míg a „peremezés” szokás szerint a dimenziók/tény-attribútumok szerint alakul.

54. Mentsük eddigi munkánkat (**Files → Save All**), majd még egyszer **validáljuk** a SALESFACT adatkocka modellt (**Analyze Model**). Ha minden rendben, folytassuk.

Az előzőekben manuálisan hoztuk létre azokat az OLAP meta-adatokat (dimenziókat, szinteket, hierarchiákat, stb), amelyeket adatbázisunkba importáltunk, ám ez nem feltétlen van mindig így. Elképzelhető, hogy valaki más előzőleg már létrehozta a megfelelő meta-adatokat, szabványos formában kiexportálta őket, nekünk pedig már csak be kell olvasni őket. A Cube Views szerencsére erre is lehetőséget ad. Lehetőség van OLAP meta-adatok szabványos **XML** (e**X**tensible **M**arkup **L**anguage) leírásának a beolvasására. A következőkben ezzel foglalkozunk.

55. Kattintsunk a **File → Import** menüpontra, bontsuk ki a **Data Warehousing**-ot, válasszuk ki az **OLAP metadata**-t, és **Next**. Böngésszük ki a **File Name**-nél a `C:\DW35\OLAP\DW350FACT.XML` fájlt, majd válasszuk ki a **CVSAMPLE** adatbázist (*ne a modellt!!!*), és **Next**. Végezetül válasszuk ki a **DB2ADMIN** sémát, és **Finish**.

Ezzel lényegében egy, az előzőekben létrehozottal ekvivalens adatkocka modellt importáltunk be az adatbázisunkba (tekintsünk bele pl. **Notepad**-del az előbbi XML fájlba, és bizonyosodjunk meg erről a saját szemünkkel). A modellhez egyetlen konkrét adatkocka tartozik, a **DW350Cube**. Célunk most az, hogy a beimportált modellt tovább optimalizáljuk. Ez azt jelenti, hogy szeretnénk gyorsabbá/hatékonyabbá tenni a modellhez tartozó adatkockák lekérdezését. Ennek egy bevált módja az **összegző táblák (Summary Tables)** használata, amiket **MQT** (**M**aterialized **Q**uery **T**able) **materializált nézetek**nek is szokás nevezni. Ezek gyakorlatilag olyanok, mint egyfajta „cache”, azaz bizonyos lekérdezések/lekérdezés-részek eredményét tartalmazzák. Amennyiben tehát az adatkocka adott lekérdezése során képesek vagyunk ezeket az MQT táblákat hatékonyan felhasználni, rengeteg időt/erőforrást spórolhatunk meg.

56. A Database Explorer-ben kattintsunk jobb egérgombbal a CVSAMPLE adatbázis DB2ADMIN sémájába újonnan beimportált **DW350FACT** nevű adatkocka modellünkre (**Connections → CVSAMPLE → CVSAMPLE → Schemas → DB2ADMIN → OLAP Objects → Cube Models → DW350FACT**), és válasszuk az **Optimization Advisor** opciót.

57. A **DW350Cube** adatkocka mellett kattintsunk a „**…**” gombra, és határozzuk meg, hogy milyen módon kívánunk majd hozzáférni a kockához. Amennyiben egy relációs adatbázisra épülő OLAP terméket (vagy egy DB2 Alphablox-hoz hasonló, vizualizációs terméket) kívánunk hatékonyabbá tenni, úgy a **Drill-down** opció a megfelelő. Amennyiben riportolni szeretnénk, úgy a **Report**-ot lenne érdemes választani. Minden esetre mi most az **Advanced settings**-et válasszuk, és kattintsunk mellette a **Specify** gombra.

58. A feljövő ablakban hozhatunk létre **optimalizációs szeleteket (optimization slices)**, amik gyakorlatilag a választott adat(hiper)kocka adott módon és adott szinten való „keresztülhasításával” állnak elő. Kattintsunk a **kis zöld plusz jelre**. A különböző dimenziókhoz különböző szinteket adhatunk meg (ezt mutatja a grafikon), az optimalizáció módját pedig adott szín jelöli. Jelenleg (alapértelmezésben) a **Drill-down** van kiválasztva az újonnan létrehozott optimalizációs szeletünkhöz (**Query type**). Ez egyelőre megfelel számunkra.

59. Láthatjuk, hogy az egyes dimenzióknál egyelőre **Any** van kiválasztva. Ebben az esetben az Optimization Advisor saját maga próbálja meg kiválasztani, hogy melyik dimenzióban melyik szinten lenne célszerű az optimalizációs szeletet létrehozni (nyilván a helytakarékosság is egy szempont ilyenkor). Segítsünk egy picit az Advisor-nak! A **DSTORE** dimenziónál válasszuk a **DCityLev** szintet, a **DPRODUCT**-nál pedig a **DProductLev**-et. A **DTIME**-nál maradjon **Any**.

60. Hozzunk létre még egy opt. szeletet! A **Query type** továbbra is **Drill-down** legyen, a **DSTORE** dimenziónál a **DRegionLev** szintet válasszuk, a **DPRODUCT**-nál a **DFamilyLev**-et, és a **DTIME**-nál a **DMonthLev**-et.

61. **OK**, **OK**, **Next**, **Next**. Arra a képernyőre értünk, ahol beállíthatjuk, hogy milyen **korlátozások (limitations)** vonatkozzanak az Advisor által létrehozandó MQT-ékre. Beállíthatjuk, hogy legfeljebb mennyi helyet foglaljanak, mennyi időbe kerüljön az analízis, és hogy ennek során mintavételezze-e az Advisor a kocka adatait. Esetünkben az alapértelmezett beállítások (nincs se tár, se idő korlát, de van mintavételezés) megteszik. Kattintsunk a **Start Advisor** opcióra.

62. Miután lezajlott az MQT (összegző) táblák meghatározása, kattintsunk a **Next**-re. Ezen a képernyőn láthatjuk azokat az SQL script-eket, amelyeket az Advisor generált, amelyek egyrészt létrehozzák a javasolt MQT-ket, és karban is tartják (update-elik) azok tartalmát. A két megfelelő fájl lényegében a következő:

    ```
    C:\IBM\dwe\Client\CreateMQT-DW350FACT-....sql
    C:\IBM\dwe\Client\RefreshMQT-DW350FACT-....sql
    ```

63. Kattintsunk a **Finish** gombra, és – miután megbizonyosodtunk arról, hogy minden eddigi munkánk le van mentve – zárjuk be a Design Studio-t.

A következőkben a kigenerált (MQT-ket létrehozó és frissítő) SQL script-ekre alapozva valóban létrehozzuk (deploy-oljuk) a megfelelő MQT táblákat az adatbázisba. Ennek nyilván rengeteg módja lehet, amik közül mi most a DWE Administration Console-t választjuk.

64. Indítsuk el a **Websphere Application Server**-t (**Start menü → All Programs → IBM WebSphere → Application Server 6 → Profiles → default → Start the Server**), és miután elindult, kattintsunk duplán az asztalon a **DWE Administration Console** ikonra, majd lépjünk be (`db2admin/ibm2blue`).

65. A baloldali menüben bontsuk ki a **DWE OLAP** fiókot, és kattintsunk az **OLAP Optimization**-re. Itt láthatjuk az előző gyakorlat során létrehozott **CVProf** adatbázis profilt, és ezen belül (alább) az adatbázisban aktuálisan meglévő adatkocka modelleket. Válasszuk ki ezek közül a **DW350FACT**-et, és kattintsunk az **Advise** gombra.

66. Itt láthatjuk, hogy igazából nem sok választásunk van (pl. mindig Drill-down módban optimalizál, stb). Tehát kattintsunk inkább a **Back** gombra, majd a **Run SQL Script**-re (hiszen már van MQT-ket létrehozó, és karbantartó SQL script-ünk).

67. Itt böngésszük ki az előbb kigenerált, MQT-k létrehozásáért felelős SQL script-fájlt, és kattintsunk a **Run Script** gombra. A lefutást követően tekintsük meg, és próbáljuk meg értelmezni a **naplót**.

68. Zárjuk le a naplót, és kattintsunk a baloldali menüben az **OLAP Management**-re. Láthatjuk, hogy lényegében itt tudnánk importálni/exportálni az OLAP meta-adatokat (adatkocka modelleket). Mindazonáltal mi ezt már a Design Studio-ban megtettük.

69. Kattintsunk a **View OLAP Contents**-re (baloldalt), **válasszuk ki** valamelyik adatkocka modellt, és kattintsunk a **Facts** gombra. Ennek hatására megtekinthetjük, hogy az adott adatkocka modellhez tartozóan melyik a ténytábla, és azon belül például a tényekhez milyen attribútumok, és mértékek tartoznak.

A következőkben vegyük használatba a létrehozott MQT táblákat!

70. Indítsuk el a DB2 parancssori konzolt (**Start menü → Run →** **`db2cmd`**), **maximalizáljuk** (duplán kattintva a felugró konzolablak fejlécére), és csatlakozzunk a CVSAMPLE adatbázishoz (**`db2 connect to cvsample`**).

71. Ezt követően hozzunk létre úgynevezett **magyarázó táblákat (explain tables)** a következő, előre definiált **DDL** (**D**ata **D**efinition **L**anguage) script segítségével.

    ```
    db2 -tvf c:\ibm\dwe\sqllib\misc\explain.ddl
    ```

72. A magyarázó táblák a **magyarázó lekérdezések (explain queries)** kapcsán kerülnek előtérbe. A magyarázó lekérdezések lényegében az adatkocka fölött végzett SQL lekérdezések optimalizált változatai. Fussuk át (**Notepad**-ben) az említett DDL script-et, majd váltsunk alkönyvtárat (**`cd \dw35\mqt`**), és hajtsuk végre az egyik ott található SQL-es adatkocka-lekérdezést (**`db2 -tvf query1.sql`**). A végrehajtott SQL lekérdezés a következő:

    ```
    select      region_name, sum(sales) as Totalsales
    from        salesfact f, location l, store s
    where       region_name='East' and
                l.postalcodeid = s.postalcodeid and
                s.storeid = f.storeid
    group by    region_name;
    ```

73. Most pedig – a magyarázó táblákra alapozva – nézzük meg, hogy miképpen lett optimalizálva (azaz átírva) az előbbi lekérdezés. Nyomjuk meg a felfelé gombot a billentyűzeten, és írjuk át a parancsot a következőre.

    ```
    db2 -tvf query1explain.sql
    ```

74. **Vessük össze** a kapott magyarázó lekérdezést az eredetivel. Milyen különbségeket tapasztalunk? Van egyáltalán számottevő különbség?

75. Hajtsunk most végre még egy másik, igen hasonló lekérdezést (**`db2 -tvf query2.sql`**). A végrehajtott SQL lekérdezés:

    ```
    select      region_name, avg(sales) as Avgsales
    from        salesfact f, location l, store s
    where       region_name='East' and
                l.postalcodeid = s.postalcodeid and
                s.storeid = f.storeid
    group by    region_name;
    ```

76. Ezek után vessük újra össze az eredeti lekérdezést, és a hozzá tartozó magyarázó lekérdezést (`db2 -tvf query2explain.sql`). Most mit tapasztalunk? Beépíti az optimalizált lekérdezés az MQT-ket? Miért?

77. Miután megértettük, hogy miről van szó, zárjuk be a parancssori konzolt (`exit`).

Az előbbiekben felépítettünk, és políroztunk egy adatkocka modellt, majd adatkockát hoztunk hozzá létre, és ezt optimalizáltunk, és optimalizált módon lekérdeztük. Felmerülhet azonban a kérdés, hogy: *e sok hűhónak végső soron mi is volt az értelme? Miért hoztunk létre adatkocka modellt, és adatkockát? Mi ennek az egésznek a haszna a gyakorlatban?* – A válasz viszonylag összetett, ámde egyértelmű: az adatkocka lényegében egy rugalmasan kezelhető adatstruktúra, amelyet intuitív módon, igen sokféleképp aggregálva/variálva lehet lekérdezni, és információt kinyerni belőle. Lényegében tehát az adatkocka önmagában véve nem hordoz túl sok értéket, viszont egy megfelelő alkalmazás (pl. egy **VIR** – **V**ezetői **I**nformációs **R**endszer) szemszögéből nézve kulcsfontosságú lehet. Adatkockákra épülő módon kiváló <u>döntéstámogató rendszerek</u> hozhatók létre, amelyek az adatokban lévő összefüggések, mintázatok hatékony, célzott kinyerésére alkalmasak. Tegyük fel, hogy például olyan rendszert szeretnénk készíteni, amely egy üzleti stratéga (aki feltehetőleg nem SQL guru) számára képes vizualizálni az üzleti adatokat (pl. statisztikák, riportok formájában). Amennyiben ez a rendszer egy (vagy több) adatkockára épít, úgy ez hatalmas szabadságot, és rugalmasságot ad a lekérdezések automatikus kialakításakor. Ennek következtében a ráépülő vizualizációs réteg is kellően általános, és rugalmas lehet. Üzleti stratégánk tehát néhány kattintással egészen összetett információkat nyerhet ki az adatokból (pl. bizonyos aggregátumokat, korrelációkat). Amiről tehát szó van, az lényegében az **OLAP** (**O**n-**L**ine **A**nalytical **P**rocessing): *szerteágazó adatok on-line analízise és feldolgozása*. Erre használjuk az adatkockákat. Ez az egész elgondolásnak az értelme. A következőkben tehát ennek megfelelően tovább lépünk, és az adatkockákra, mint adatforrásokra, alapozva olyan web-alkalmazást hozunk létre, amely képes az adatokat magas szinten, rugalmasan vizualizálni, és manipulálni. Az IBM DB2 Data Warehouse csomagon belül erre a **DB2 Alphablox** való.

78. Az előbbiekben láttuk, hogy a **DWE Administration Console** is lehetőséget ad arra, hogy OLAP meta-adatokat importáljuk az adatbázisba. A következőkben ilyen módon fogunk meta-adatokat importálni a QCC nevezetű adatbázisba. Mivel elviekben még mindig fut a **WAS** szerver, és az Administration Console is nyitva van, kattintsunk azon belül a baloldali menüben a **DWE Common** alatt a **Database Profile**-ra, és hozzunk létre egy újabb adatbázis profilt a QCC adatbázis számára.

79. Kattintsunk a **Create** gombra, a profilnak adjuk a **QCCProf** nevet, válasszuk a **Create Profile with a New Data Source** opciót, és **Next**.

80. A következőkben definiáljuk a profilhoz tartozó adatforrást! A **JDBC** (**J**ava **D**ata**B**ase **C**onnectivity) Provider legyen **DB2 Universal JDBC Driver**, az adatforrás neve **QCC** legyen, a **JNDI** (**J**ava **N**aming and **D**irectory **I**nterface) neve legyen **jdbc/dwe/qcc**, az adatbázis neve pedig ugyancsak **QCC**. **Next**. Itt írjuk be a jelszavunkat, és **Create**.

81. Látható, hogy a CVProf mellett immár az új, QCCProf adatbázis profil is létrejött. Válasszuk ki, és kattintsunk a **Test**-re.

82. Amennyiben az adatkapcsolattal minden rendben, kattintsunk a baloldali menüben a **DWE OLAP** alatti **OLAP Management**-re, és a legördülő listából válasszuk ki a **QCCProf** profilt, majd kattintsunk az **Import** gombra ahhoz, hogy az adatbázisba beimportáljunk egy adatkocka modellt. Böngésszük ki a `C:\DW35\OLAP\QCC_2003_Metadata.xml` fájlt, **Open**, és az importálás megkezdéséhez kattintsunk a **Start Import**-ra. *(Megjegyzés: itt, ha minden rendben, kapunk egy figyelmeztető üzenetet, miszerint az importálás során két hierarchiaszint között nem sikerült létrehozni a funkcionális függőséget. Ezzel most ne foglalkozzunk, nem jelent problémát...)*

83. Kattintsunk inkább a baloldali menüben, a **DWE OLAP** alatt, a **View OLAP Contents**-re. Az adatbázis profil a **QCCProf** legyen. A megfelelő kocka modell legyen kiválasztva, és nézzük meg a **Facts**, illetve a **Dimension Information** részt. Ezek után jelentkezzünk ki a DB2 Administration Console-ból (**Logout** (jobboldalt fent)), és zárjuk be a böngészőt.

84. A következőkben fókuszáljunk a DB2 Alphablox-ra, és próbáljuk meg vizualizálni az iménti adatkocka tartalmát. Kattintsunk duplán az asztalon a **DB2 Alphablox Administrative Pages** ikonra, a böngészőben engedélyezzük az aktív tartalmat (**Allow blocked content** (a címsor alatti vajszínű sorra kattintva)), és lépjünk be (`db2admin/ibm2blue`).

85. Válasszuk ki az **ADMINISTRATION** fület, és definiáljunk pár adatforrást (hasonlóan ahhoz, ahogy előbb a DWE Administration Console-ban is tettük). Kattintsunk a **Data Sources**-ra, aztán a **Create**-re. Az adatforrás neve **QCC** legyen, az adapter típusa **IBM DB2 JDBC Type 4 Driver**, az adatbázis neve **QCC**, írjuk be a belépési nevünket és jelszavunkat, majd kattintsunk a **Save**-re.

86. Ezek után válasszuk a **QCC** adatforrást, és teszteljük le (**Test Selected Data Source**), és ha minden rendben ment, akkor **Close**.

87. Most hozzunk létre még egy adatforrást! A neve legyen **CVSAMPLE**, az adapter típusa ugyanaz legyen, mint előbb, az adatbázis neve is **CVSAMPLE** legyen, írjuk be a belépési nevünket és jelszavunkat, majd **Save**. A létrehozott adatforrást nyomban **teszteljük is le**!

88. Hozzunk létre még egy, harmadik, és egyben utolsó adatforrást is: egy Alphablox adatkocka szervert (Alphablox Cube Server)! Az adatforrás neve tehát **ACS** legyen, az adapter pedig **Alphablox Cube Server Adapter**. Kattintsunk is gyorsan a **Save**-re, és **teszteljük** is nyomban le az adatforrást.

Az elkövetkezőkben, miután felkonfiguráltuk majdani DB2 Alphablox vizualizációs web-alkalmazásunk kontextusát, hozzuk létre magát az alkalmazást, és vessük be (deploy-oljuk) a web-szerverre. Egészen pontosan az **EAR** (**E**nterprise **AR**chive) állományt fogunk létrehozni, és ezt fogjuk a WAS szerverre deploy-olni.

89. Az Alphablox Administrative Pages-en belül, továbbra is az **ADMINISTRATION** fülben maradva kattintsunk az **Applications** menüpontra, majd a **Create** gombra, és adjuk meg a létrehozni kívánt alkalmazás adatait. Az alkalmazás neve legyen **DW350ABApp**, a Display Name legyen **DW350 Alphablox Application**, a Home

URL legyen **`/home/index.jsp`**, az Image URL legyen vagy **`/images/ibm.gif`**, vagy **`/images/cube.gif`**, és végezetül kattintsunk a **Save**-re.

Fentebbi ténykedésünk hatására létrejött a `C:\IBM\dwe\Alphablox\installableApps` könyvtárban egy `DW350ABApp.ear` állomány, amely lényegében a kiindulási Alphablox web-alkalmazásunk összecsomagolt példánya. Ezt kellene a következőkben a WAS szerverre telepíteni, minekután az Alphablox segítségével létrehozhatjuk majd azokat a **JSP** (**J**ava **S**erver **P**ages) és **DHTML** (**D**ynamic **HTML**) kódokat, amiket aztán web-alkalmazásunkba integrálhatunk. – Egyébként a fentebbi létrehozás során lényegében azt mondtuk meg a WAS szervernek (amin ugyebár az Alphablox Administrative Pages web-alkalmazás is fut, amit momentán használunk), hogy a létrehozott Alphablox web-alkalmazásunk lényegében a szerveren a `/DW350ABApp.war/home/index.jsp` lesz, a hozzá tartozó képek pedig a `/DW350ABApp.war/images` könyvtárban találhatók.

90. Indítsuk most el a WAS szerver **Administrative Console**-ját (az asztalon a **lila kör** ikon), de az Alphablox Administrative Pages-t ne zárjuk be. Lépjünk be (`db2admin/ibm2blue`), majd bontsuk ki a baloldali menüben az **Applications** fiókot, és kattintsunk az **Install New Application**-re. Böngésszük ki az előbb létrehozott **EAR** állományt, **Open**, aztán kattintgassunk a **Next**-re egészen addig, amíg a **Step 4**-hez nem érünk (**Map security roles to users/groups**). *(Megjegyzés: ha adott esetben azt írja a böngésző, hogy nem tudja megjeleníteni az adott oldalt, akkor nyomjunk rá a frissítésre, és kezdjük újra. Addig próbálkozzunk, amíg nem sikerül eljutni a 4-dik lépésig.)*

91. Ebben a lépésben az alapértelmezésben létrejött két szerepkörhöz (Alphablox web-alkalmazás adminisztrátor és/vagy felhasználó) társítunk hozzá konkrét felhasználókat. Ha például bekattintanánk az **All Authenticated** jelölődobozt, akkor azzal az adott szerepkört minden belépésre jogosult felhasználó számára megadnánk. Mi azonban most konkrét felhasználókat szeretnénk feljogosítani az egyes szerepkörökre. Válasszuk ki tehát mindkét szerepkört (**Select**), és kattintsunk a **Lookup users** gombra, majd a **Search**-re.

92. A feljövő listából válasszuk ki a **`db2admin`** felhasználót, kattintsunk a **>>** gombra, majd **OK**, **Next**, és **Finish**. Végül **Save to Master Configuration**, és **Save**.

93. Most pedig indítsuk el a sikeresen feltelepített web-alkalmazást! Kattintsunk a baloldali menüben az Applications részben az **Enterprise Applications**-re. Itt egy piros X jelzi, hogy web-alkalmazásunkat a web-szerver éppen nem futtatja. Válasszuk tehát ki **DW350ABApp** web-alkalmazást, és kattintsunk a **Start**-ra. Ha minden rendben ment, **kijelentkezhetünk**, majd pedig **kiléphetünk** a WAS Administrative Console-jából.

94. Most pedig nyissuk meg a **Windows Explorer**-t, és a következő könyvtárban hozzunk létre egy **`home`**, és egy **`images`** alkönyvtárat:

    `c:\IBM\dwe\appServer\profiles\default\installedApps\DW350Node01Cell\DW350ABApp.ear\DW350ABApp.war`

95. Most másoljuk bele a **`c:\DW35\Alphablox\jsps\index.jsp`** fájlt a **`home`** könyvtárba, az **`images`** könyvtárba pedig a **`c:\DW35\Alphablox\Images`** könyvtár teljes tartalmát tegyük át.

Web-alkalmazásunk meghívásakor lényegében először a fentebbi JSP kerül majd meghívásra.

96. Térjünk vissza az **Alphablox Administrative Pages**-re, és kattintsunk az **APPLICATIONS** fülre. A megjelenő lapon immár a mi, **DW350 Alphablox Application** nevű web-alkalmazásunk is látszik. Kattintsunk rá! Ennek hatására feljön egy alapértelmezett, gyakorlatilag üres oldal. Egyelőre ennyi az egész web-alkalmazás. Kattintsunk tehát az **Internet Explorer**-ben a **Back**-re, hogy visszalépjünk az Alphablox Administrative Pages-re.

97. Most hozzunk létre az Alphablox web-alkalmazásunk számára egy adatkockát, amit vizualizálni tud! Válasszuk ki az **ADMINISTRATION** fület, majd kattintsunk fent a **Cubes** menüpontra, és a **Create**-re. A kocka neve **QCCCube** legyen, mellette kattintsuk be az **Enable**-et, válasszuk a **QCC** adatforrást, majd kattintsuk be az **Enable DB2 Cube Views Settings** jelölődobozt, a legördülő listából válasszuk az **ABADMIN.Cube Model qcc_2003** adatkocka modellt, kattintsuk be az **Import cube definition on start, rebuild and edit** jelölődobozt, majd **OK**.

98. Hasonlóképp tegyük elérhetővé az Alphablox számára a gyakorlat kezdetén létrehozott SALESFACT adatkocka modellt is! Az Alphablox kocka neve **CVCube** legyen, **Enabled**, az adatforrás a **CVSAMPLE**, **Enable DB2 Cube Views Settings**, válasszuk a **DB2ADMIN.SALESFACT** adatkocka modellt, **Import cube definition on start, rebuild and edit**, és **OK**.

99. Most pedig indítsuk el mindkét Alphablox kockát (amik a Cube Views-os adatkockákból jöttek létre)! Kattintsunk a **General** menüpontra, majd ott baloldalt, a Runtime Management rész alatt a **DB2 Alphablox Cubes**-ra, majd válasszuk ki az adatkockákat, és **Start**. Mindkettőt indítsuk el.

A DB2 Alphablox nem az SQL-t, hanem egy saját, **MDX** nevezetű lekérdező nyelvet használ arra, hogy az Alphablox-os adatkockákat lekérdezze. A következőkben ezekkel a lekérdezésekkel fogunk foglalkozni (lévén ezek képezik a vizualizációs alkalmazás alapját).

100. Kattintsunk az **APPLICATIONS** fülre, majd indítsuk el a **DB2 Alphablox Query Builder** nevezetű Alphablox-os web-alkalmazást. Kattintsunk fent a **Connection Settings** gombra, és állítsuk be az Alphablox-os adatkocka szerver elérését. Az adatforrás az **ACS** legyen, és **Connect**.

101. Vessünk egy pillantást a **Query** részben az MDX lekérdezésre. Amennyiben nem a **QCCCube** adatkocka szerepel benne, úgy írjuk át megfelelőképp, és kattintsunk az **Execute Query** gombra.

102. Az eredmény máris megjelenik az oldal alsóbb részén látható **PresentBlox**-on belül. Ezen belül, a baloldali részen – az egér bal gombját lenyomva tartva – mozgassuk át a **Time**-ot a **Page** alá. A **Measures**-t vigyük a **Column** alá. A **Scenario**-t a **Measures** és a **Column** közé. Végül a **Markets**-et vigyük a **Row** alá, a **Products**-ot pedig közvetlenül a **Markets** alá.

103. Most a PresentBlox középső részében, a táblázatnál kattintsunk duplán az **All Markets**-re, az **All Products**-ra, majd az **All Scenarios**-ra. Ezzel Drill down-t (lefúrást) hajtottunk végre az adatok adott nézetén belül – kibomlik a táblázat. Vegyük

észre, hogy ezzel együtt a jobboldali chart is szépen lassan részletesebbé vált, hogy tükrözze a táblázat állapotát.

104. Kattintsunk jobb egérgombbal a táblázat legjobboldalibb oszlopának címére, a **Revised**-ra, és válasszuk a **Remove Only** opciót. Ennek hatására eltűnik az adott oszlop.

105. Most vessünk egy pillantást fentebb, a **Query** részre, azon belül pedig az **MDX** lekérdezésre. Ez a lekérdezés éppen az alábbi táblázatot állítja elő. Hát nem nagyszerű, hogy nem nekünk kellett manuálisan megírnunk ezt a lekérdezést?

106. Kattintsunk most a PresentBlox kék fejlécének jobb oldalán látható **Generate Blox Tag** gombra, és a megjelenő, és egyben kiválasztott JSP kódot másoljuk ki a vágólapra (**Ctrl+C**). Ezek után a **Windows Explorer**-rel keressük meg a fentebb létrehozott **.../home/index.jsp** fájlt, és nyissuk meg szerkesztésre a **Notepad**-del.

107. Közvetlenül a **`<html>`** nyitó-tag alá írjuk be a következőt:

```
<%@ taglib uri="bloxtld" prefix="blox" %>
```

108. A **`<header>`** és **`</header>`** között van egy üres sor. Ebbe tegyük a következőt:

```
<blox:header />
```

109. Most pedig másoljuk be az imént vágólapra kitett JSP kódot az **`index.jsp`** **`<h1>`** és **`</body>`** tag-je közötti üres sorba (**Ctrl+V**), majd mentsük a fájlt (**Ctrl+S**), de még ne lépjünk ki a Notepad-ből.

110. Mivel a módosított JSP fájlt újra kell fordítani (és cache-elni), ezért zárjuk le az összes böngésző ablakot, majd lépjünk vissza az Alphablox Administrative Pages-be (újra fogja kérni a felhasználói nevünket és jelszavunkat), és indítsuk el a módosított web-alkalmazásunkat (**DW350 Alphablox Application**). Ha mindent jól csináltunk, akkor immár látni fogjuk benne a PresentBlox-ot, és teljesen interaktív módon lekérdezgethetjük a mögötte álló adatforrásokat (adatkockákat). **Kísérletezzünk** ezzel a felülettel!

Ezzel tehát egy egyszerűbb példa szintjén megvalósítottuk az adatkockákra épülő vizualizációt. Létrehoztunk egy web-alkalmazást, amely alkalmas módon képes megjeleníteni adott adatkockák adattartalmát. Ezeket az elgondolásokat tovább skálázva akár komolyabb VIR rendszereket is létrehozhatnánk. Mi azonban egyelőre elégedjünk meg azzal, hogy ezt a jelenlegi megjelenítést egy kicsit tovább csiszoljuk.

111. Módosítsuk tovább az **`index.jsp`**-t! A **`<blox:present... </blox:present>`** részen belül van egy **`<blox:grid/>`** tag. Cseréljük ezt le a következőre:

```
<blox:grid>
<blox:cellAlert index="1"
scope="{Measures:sales}{Scenario:Actual}"
condition="LT"
value="2000000"
background="red">
</blox:cellAlert>
<blox:cellAlert index="2"
scope="{Measures:sales}{Scenario:Actual}"
condition="between"
value="2000000"
value2="5000000"
background="yellow">
</blox:cellAlert>
<blox:cellAlert index="3"
scope="{Measures:sales}{Scenario:Actual}"
condition="GT"
value="5000000"
background="green"
foreground="white">
</blox:cellAlert>
</blox:grid>
```

112. Mentsük le a módosított **`index.jsp`** fájlt, zárjunk le minden böngésző ablakot, és utána az Alphablox Administrative Pages-be visszalépve indítsuk újra el a **DW350 Alphablox Application** web-alkalmazást. *Milyen különbséget tapasztalunk az előzőekhez képest? Melyik mit jelent?*

113. Emlékezzünk vissza, hogy egy másik, **CVCube** nevezetű Alphablox-os adatkockát is létrehoztunk. Próbáljuk meg ezzel a kockával is nagyjából ugyanazt eljátszani, mint előbb a QCCCube-bal, hogy learassuk a gyakorlat első részében végzett munkánk eredményét.

114. Végül – miután mindent befejeztünk, és a gyakorlatvezetőnek is bemutattuk eddigi munkánkat – zárjunk le minden ablakot, és állítsuk le a WAS szervert (**Start menü → All Programs → IBM WebSphere → Application Server 6 → Profiles → default → Stop the Server**).