Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Magas szintű jelentés és lekérdezés specifikációs nyelv készítése Essbase adatbázishoz

Önálló labor beszámoló

Készítette
Lakatos Zsolt

Konzulens
Semeráth Oszkár

2016. május 26.

Tartalomjegyzék

Kivonat

Napjainkban a vállalatoknál egyre több adattal dolgoznak. Ezeket az adatokat tipikusan adattárházba mentik, majd erre OLAP (On Line Analitical Processing) adatbázist építenek, annak érdekében az adatok összefüggéseit lehessen vizsgálni. Az adatok vizsgálatánál olyan lekérdezéseket fogalmaznak meg, hogy az adatok összevontan jelenjenek meg (például összegezve vagy átlagolva). Ilyen OLAP adatbázis például, a gyakorlatban igen sokszor használt és elterjedt adatbázis, az Essbase.

Az előzőekben említett lekérdezések megfogalmazásánál nehézséget okoz fejlesztéskor, hogy (i) a fejlesztő környezet nem validálja a megírt kódot, (ii) mivel a nyelv deklaratív, nem lehet benne hibát keresni, (iii) hibáknál gyakran üres megoldást kapunk. Valamint, a fejlesztő környezet is kevés segítséget nyújt, mert a lekérdezés írásakor egy egyszerű szöveges fájlt kell szerkeszteni. Továbbá az eredmény halmaz egy n-dimenziós mátrix, ami átláthatatlan és nehéz feldolgozni. Ezen túlmenően, mivel kétféle különböző funkcionalitással bíró lekérdező nyelv van az Essbase-hez, sokszor előforduló jelenség, hogy ezeket kombinálva kell használni egy adott problémához, ami további nehézségeket okoz. Ilyen probléma például a költségfelosztás, amikor egyik elemben vagy csoportban lévő költséget (az eltárolt számot), két másik elemre vagy egy egész csoportra szeretnénk valahogy felosztani valamilyen, arány szerint.

Ezeket a problémákat szeretném kezelni dolgozatomban, mégpedig úgy hogy saját lekérdező nyelvet és hozzá szerkesztőt készítek, ami egyrészt megoldást kínál a felvetett problémákra, valamint kombinálja a kétféle lekérdező nyelv (Riport, MDX) előnyeit egybe. Célkitűzésem tehát egy magas szintű jelentés és lekérdező nyelv készítése, ami ezen problémákra választ ad. Dolgozatomban bemutatok egy olyan Xtext alapú nyelvet, amely segítségével le lehet kérdezni az Essbase adatbázisból, valamint hogy ezt a gyakorlatba is lehessen használni, készítek hozzá egy Eclipse alapú szöveges szerkesztőt, ahol a megírt lekérdezést lehet futtatni. Végül a lekérdezések eredményét Latex segítségével egy PDF dokumentumba ábrázolom, amely tetszőlegesen szemléltetheti a megoldást.

Mindezek alapján a fejlesztés sokkal gyorsabb és hatékonyabb lesz, mert nem kell kétféle programozási nyelvet használni, hatékonyabban, kevesebb munkával meg lehet ugyanazokat, vagy még bonyolultabb Essbase lekérdezéseket vagy speciális műveleteket írni.

1. fejezet

Előismeretek

1.1. Essbase adatbázis

Alapvetően kétféle adatbázis típus létezik lekérdezés szempontjából. Az egyik az OLTP (On Line Transaction Processing, online tranzakciófeldolgozás), ahol sok tranzakció van és ezek kis módosításokat (UPDATE,INSERT), vagy kisebb, rövidebb ideig tartó lekérdezéseket végeznek el az adatbázisban. A másik az OLAP (On Line Analitical Processing vagy online analitikai feldolgozás), ami kimondottan a lekérdezésekre lett optimalizálva, jelentések készítésére. Ilyen adatbázis például az Essbase.

OLAP Kocka: Az elemezni kívánt dimenziók szintjei és hierarchiája alapján összevont értékeket tároló adatszerkezet. A kockák sokféle dimenziót – időt, földrajzi helyet stb. – képesek összekapcsolni különböző összesített adatokkal – például a gépjárműveket a szervezeti egységekkel. A kockák nem a szigorúan vett matematikai értelemben kockák, mivel nem biztos, hogy egyenlők az oldalaik. Ennek ellenére egy nagyon szemléletes metaforái egy bonyolult koncepciónak. Az Essbase adatbázis OLAP kockák összességéből áll.

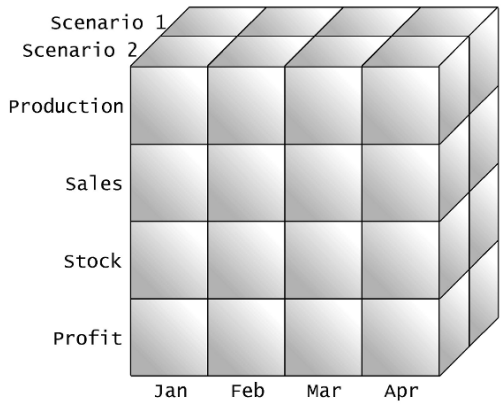
Érték: A kockákban lévő értékek, amelyek a kocka ténytáblájának valamely oszlopán alapulnak, és általában numerikus értékek. A mértékek a kocka központi értékei – előfeldolgozhatók, aggregálhatók és elemezhetők. A mértékekre gyakori példa a forgalom, a profit, a bevétel és a költség. Egy mérték a kocka összes lekötött dimenziójából adódik. Például meg kell adni milyen év, melyik szervezeti egység, melyik konkrét autó stb.

Tag: Azok az attribútumok, amikhez értékek kötődnek, például egy konkrét gépjármű.

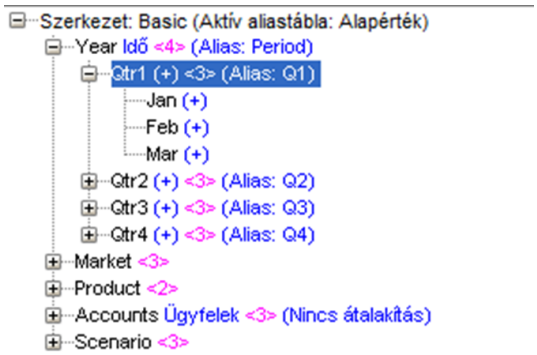
Csoport: Néhány tag összessége.

Dimenzió: A szintek szervezett hierarchiáinak együttese egy kockában, amit a felhasználó megért, és az adatelemzés alapjaként használ. A földrajzi hely dimenziója például tartalmazhatja az ország, a megye és a város szinteket. Az idő dimenziója pedig tartalmazhatja az év, a negyedév, a hónap és a nap szintet. A kimutatásokban és kimutatásdiagramokban minden hierarchia mezőegyüttessé válik, amelyek kibonthatók és becsukhatók az alacsonyabb és magasabb szintek felfedése érdekében.

Hierarchia: A kocka megjelenítése két dimenzióban.



1.1. ábra. Példa kocka



1.2. ábra. Példa Hierarchia

1.1.1. Essbase lekérdezések

Essbase-ből alapvetően kétféleképpen lehet lekérdezni, riportban és mdx-ben. A riport lekérdező nyelv tipikusan kimutatásokhoz használható jól, ha valamit összevontan szeretnénk lekérdezni a kockából. Ezzel szemben az MDX egy standard lekérdező nyelv, amibe egyrészt lehet sql-szerű lekérdezéseket írni, másrészt adatmódosító műveleteket is lehet használni. Ilyen adatmódosító művelet például az Allocation parancs, ahol egy csoportról vagy tagról áttesszük az értékeket egy másik csoportra vagy tagra.

Riport példa

A következő riport példában kilistázom a gépjármű csoport összes költségeit, költségnemenként.

```
{SUPEMPTYROWS}
{TabDelim}
<Column ("KN_Koltsegnem")
<Row ("Objektum","GEPJ")
<Ichild "Objektum"
<Ichild "GEPJ" !
```

MDX példa

A következőkben pedig, egy allocation példát írtam, ahol a költséghely F1015-ről átteszem a költségeket a gépkocsi csoportra, egyenlő részben.

```
execute allocation process on database DEMO.DEMOXXX with
pov '
 NonEmptySubset(
CrossJoin({([KH_F1015])},
         {( [Partner_KN_Nincs], [2015], [Terv]
         )}
 ), ([C_Input]))...
amount  1 * [C_Input]
amountcontext [Partner_Obj_Nincs]
target  [C_V2_FTECH01_In]
offset [C_V2_FTECH01_Out],
range Descendants([GEPK], [Partner_.ZGEPK].DIMENSION.LEVELS(0))
basis  [C_V2_Driver] ...
```

1.1.2. Essbase esettanulmány

Tegyük fel, hogy van egy céges autónk és szeretnénk eltárolni az autó forgalmi adatait, hová ment, közbe mennyi volt a fogyasztása. Ezt tipikusan OLTP adatbázisba érdemes tárolni, mert sok tranzakció keletkezik, sok beszúrás művelettel. Ellenben tegyük fel, hogy több céges autónk van és az autókra keletkeznek kölségek, például benzin, amortizáció. Valamint tételezzük föl hogy ezekből a költségekből van olyan, ami az egész szervezeti egységhez kötődik, például a benzint egyszerre vesszük meg és mindenki tankol ebből annyit amennyire szüksége van. Ezek a költségek az OLTP adatbázisunkba vannak eltárolva egy táblába autókra és szervezeti egységekre külön. Ha megvannak ezek az adataink, akkor gyakran felmerülő igény, hogy le szeretnénk kérdezni valamilyen szempont szerint aggregálva ezeket a költségeket. Például összeadni vagy átlagolni hogy egy autóra és szervezeti egységre mennyi költség keletkezett. Egy ilyen lekérdezés, nagy tábla esetén sokáig tart, mert a tábla összes során végig kell menni. Ebben az esetben már érdemes felépíteni egy OLAP kockát, erre az OLTP adatbázisra, ami automatikusan képes az ilyen jellegű aggregációra, összeadásra, átlagolásra stb. Ezen kívűl képes arra is, hogyha keletkezik olyan költség ami egy szervezeti egységhez köthető (az előzőekben említett benzin amit egyszerre veszünk), azt le tudja osztani minden egyes autóra lebontva, például megtett út alapján.

1.2. Xtext keretrendszer

Az Xtext egy olyan keretrendszer, amiben saját szakterület specifikus programozási nyelvet lehet definiálni. Ha ez kész, akkor ebben a programozás nyelvben lehet utána fejleszteni, majd ezt a kódot az Xtext tudja validálni és feldolgozni, ezeket az adatokat objekumokba megkapjuk és java-ban ezt utána fel lehet dolgozni.

1.3. Eclipse plugin

Az Eclipse fejlesztő környezet egy modulokkal (úgynevezett pluginekkel) bővíthető keretrendszer, melyek segítségével saját funkciókkal szabhatjuk testre a környezetet. Önálló laboratórium során, én is készítek egy saját plugint, szerkesztőt, amibe a létrehozott programozási nyelv kódját lehet szerkeszteni.

1.4. Latex és riportolási technikák

Megjelenítésre, riport kimenetnek a dolgozatomban én latexet választottam, mert a forráskódja java kódból generálható és vannak olyan grafikus megjelenítő csomagjai, amivel az Essbase-ből kijövő nyers adatokat meg lehet jeleníteni. Valamint közbe lehet iktatni statisztikai, és diagram kezelő szoftvereket is, az adatok jobb megjelenítésére, ilyen például az R.

1.5. Xtend

Sok új nyelvi elemmel rendelkező java nyelv továbbfejlesztése, például sablon nyelv, funkcionális nyelv, lambda kifejezések, extensuion method. Ezekből én a sablon nyelvet használom, amivel lehet olyan szöveg sablont készíteni, amiből az Xtend java osztályokat generál és egyszerű paraméterezéssel meg lehet adni a sablonnak a változó értékeket.

2. fejezet

A lekérdező program

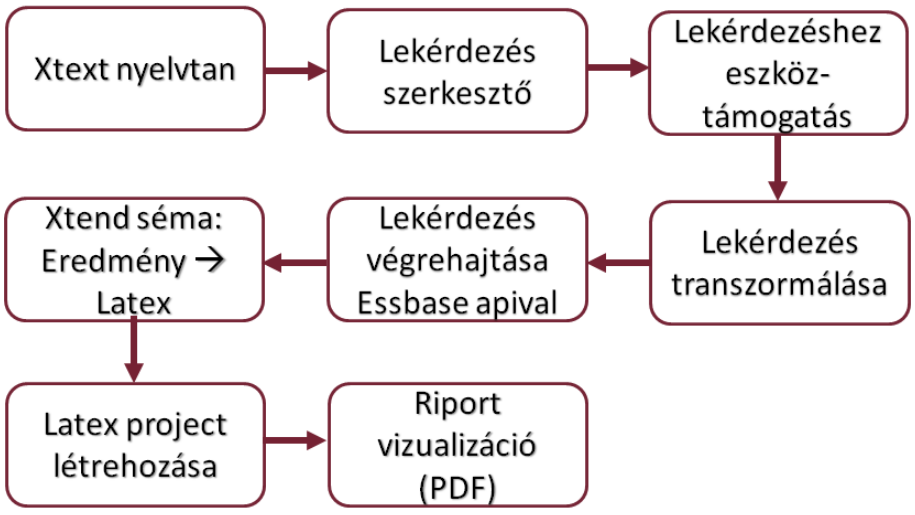
2.1. Fejlesztés menetének áttekintése

Első lépésként a bevezetőben megfogalmazott problémák megoldása érdekében, készítettem egy magas szintű lekérdező nyelvet az Essbase-hez, hogy a riport valamint az mdx nyelv összetevői alapján megfogalmaztam Xtextben egy metamodellt. Ezt a metamodellt az Xtend konkrét lekérdezések írásánál, fel tudja használni arra hogy validálja azt.

Ezek után az általam újonnan létrehozott nyelvbe, létrehoztam saját lekérdezést, amit futtatva, meg lehet nézni a havi kimutatást, költségeket.

Ezután készítettem egy szerkesztőt, amibe a fejlesztők tudnak dolgozni, ehhez egy Eclipse pugint csináltam, beépítve az Xtext nyelvtan elemző programomat.

A program felhasználó kérésre transzformálja a megírt lekérdezést olyan Essbase lekérdezéssé amit az Essbase értelmezni tud, majd hálózaton elküldi a szervernek. A visszakapott objektumot ezután, a megfelelő parancsal meg lehet jeleníteni, riportolni, egy Latex pdf-be, amit a program generál.



2.1. ábra. A folyamat áttekintése

- Xtext nyelvtan: Xtext metamodell elkészítése

- Lekérdezés szerkesztő: Eclipse fejlesztő környezetbe beépülő modul segítségével szerkesztő készítése
- Lekérdezéshez eszköz támogatás: Eclipse környezetből futtatjuk a lekérdezésünket
- Lekérdezés transzformálása: a program transzformálja a lekérdezést Essbase lekérdezéssé, és elkéri az adatokat a szervertől
- Xtend séma->Latex pdf: Xtend sémába beleteszi a program az Essbase szerver visszaadott adatait
- Latex projekt létrehozása: a program létrehozza a riport megjelenítésre szánt latex projektet
- Riport vizualizáció (PDF): végül kigenerálja a program a megfelelő diagrammokal a Latex pdf-et

2.2. Lekérdező nyelv bemutatása

Először is Xtextbe létrehoztam egy metamodelt, amibe definiáltam a saját riport lekérdezőmbe a valid parancsokat, amik kombinálják az MDX és a Riport parancsok lehetőségeit, így mindkét nyelv lehetsőségeit lehet használni egy nyelvbe.

Jelenleg elkészült parancsok, amiket a metamodellezőbe már definiáltam.

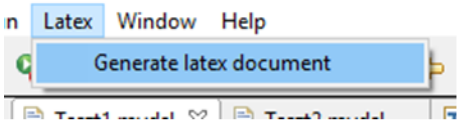
- $query(param1, param2, param_n)$: lekérdez az adatbázisból, a paraméterezésének megfelelően, gyakorlatilag egy referenciaként lehet használni, amit ezután ki lehet riportálni
- *dim* : segítségével létrehozhatunk dimenzió referenciát, amivel ezután tudunk hivatkozni más parancsokban
- *group* : segítségével létrehozhatunk group referenciát, amivel ezután tudunk hivatkozni más parancsokba
- *row* : definiálhatjuk vele a sorokat, ami majd a kimeneti riportban meg fog jelenni, dim vagy group
- *link* : meg lehet adni hogy a kimeneti riportban milyen szintig menjen le riport, meg lehet adni dimenziót, csoportot, vagy membert is
- *reportParameter* : speciális paraméter megadására használható, amit a fejlesztő környezet felhasznál, és megjelenítést befolyásolja
- *report* : bemenete egy query és futtatáskor a fejlesztő környezet generál egy pdf riportot

2.3. Validációs módszerek

Az általam létrehozott új programozási nyelv tehát tudja validálni a megírt kódot már írás közben, egyszerre lehet riportokat és lekérdezéseket fejleszteni, elírások ellen véd a változó hivatkozás.

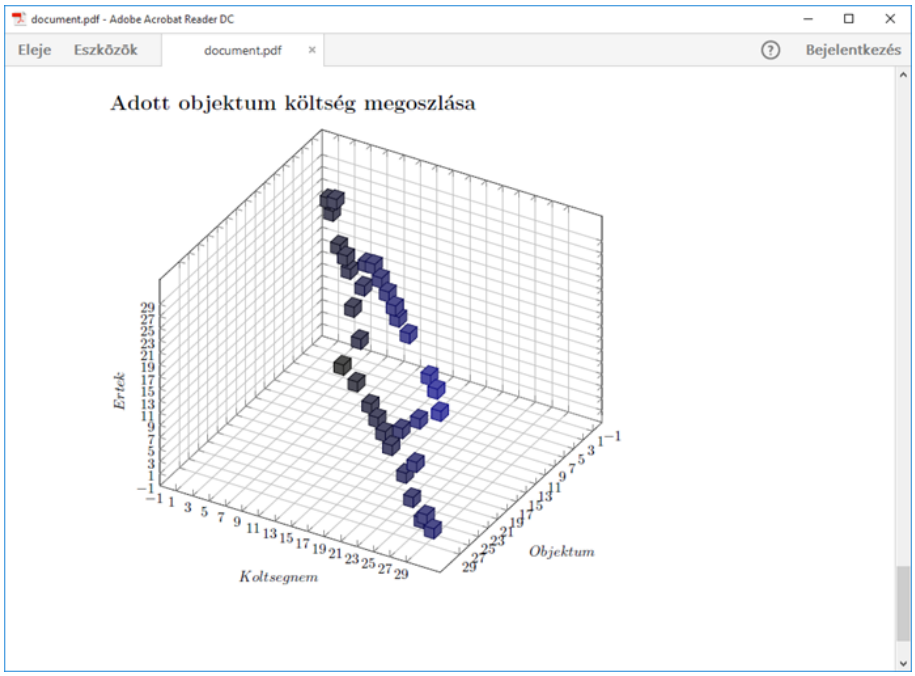
2.4. Futtatás és a riport kimenete

A megírt lekérdezést Eclipse-be lehet futtatni egy plugin segítségével a képernyőn látható módon:



2.2. ábra. Riport futtatás Eclipse-ből

Majd amikor futtatjuk a lekérdezésünket a program generál egy latex projektet, egy Tex fájlal, amibe generálja a program az általunk megírt riport kimenetét:



2.3. ábra. Riport diagram példa

3. fejezet

Implementáció

3.1. Xtext metamodell

Az új lekérdező nyelv fejlesztése során először is létrehoztam egy metamodellt Xtext-ben, ahol megadtam hogy hogyan épül föl Essbase lekérdezés, milyen parancsokból áll, közöttük milyen logika van. Mindezt azért hogy az Xtext a már kész lekérdezést majd tudja validálni.

A metamodell alapján egy Essbase lekérdező nyelv alapvetően Query-kből, valamint a riport kimenetének megadásából (Reports-ból áll):

```
Model:
        (Queries+=Query)*
        (Reports+=Report)*
        ;
```

Riport kimenet megadásakor a 'report' kulcsszó után megadunk egy olyan Query referenciát, amit már előzőleg definiáltunk.

```
Report:
        'report' {Report} repout=[Query]
;
```

Definiálom hogy hogyan épül fel egy Query, adok neki egy azonosítót (ID) és előírom hozzá a paramétereket, amiket egy Queryba meg lehet adni:

```
Query:
        name=ID '=' 'query' '{'
        InitialStatement
        (Query+=ReportQueryParameters)* '}'
        ;

ReportQueryParameters:
        Column | Row | Descendants | Declaration | Reference | Child | Link | ReportParameter;
```

Létrehozok egy öröklési hierarchiát, hogy a dimenzió, csoport és tag deklarációkat hasonlóan kezelje a program.

```

Descendants:
        'descendants' group=ID ',' dimension=Reference;

Declaration:
        DimensionDeclaration | GroupDeclaration | MemberDeclaration;
```

A következőkben definiálok objektum referenciákat, amik az Essbase kockában is vannak, ezen itt megadott objektmokra később lehet hivatkozni, elkerülve az esetleges elírásokat. Ilyen objektumok a

- dimenziók

```
DimensionDeclaration:
        'dim' name=ID value=STRING;
```

- csoportok

```
GroupDeclaration:
        'group' name=ID value=STRING;
```

- tagok

```
MemberDeclaration:
        'member' name=ID value=STRING;
```

Végül definiálok olyan parancsokat, amibe meg lehet adni, hogy a visszakapott mátrixba milyen sorok(row) és oszlopok(column) szerepeljenek, valamint a hierarchia milyen szintjéig jelenjenek meg a tagok(link):

```
Row:
        'row' {Row} '{' dimensions+=Reference (',' dimensions+=Reference)* '}';

Column:
        'column' {Column} '{' dimensions+=Reference (',' dimensions+=Reference)* '}'
;
Link:
        'link' {Link} '{' desc=Reference ',' lev=Reference '}'
;
```

3.2. Xtext példa lekérdezés

Az Xtext metamodell megírás után készítettem saját példa lekérdezést is, amit az Xtext már tud validálni. A példámban egy általános lekérdezést fogalmazok meg a kocka 2 dimenziójáról (Költségnem és Objektum), valamint 2 csoportjáról (Objektumok, és Költségnemek), ezeket az adatokat lekérdezzük a kockából, majd latex pdf-be aggregálva egy diagramon megjelenítjük őket.

Példa lekérdezés:

```
SajatQuery1=query {
        {SUPEMPTYROWS}
{DECIMAL 10}
{TABDELIMIT}
{ROWREPEAT}
{SUPBRACKETS}
{SUPCOMMAS}
{NOINDENTGEN}

dim ktg "Koltsegnem"
dim obj "Objektum"
group objects "Objektumok"
group ktgs "Koltsegnemek"

row {obj,ktg}
link {objects,obj}
link {ktgs,ktg}
reportParameter {"UGYF"}   }
report SajatQuery1
```

4. fejezet

Összefoglalás és jövőbeli tervek

Összefoglalva jelenleg a fejlesztés a következő állapotban tart:

- elkészült a lekérdező nyelv metamodellje
- elkészült a lekérdező nyelv szerkesztője Eclipse környezetben
- validálom a lekérdezést már íráskor, az Xtext metamodell segítségével
- az Essbase-től kapott adatokból, Xtend sablonokkal Latex dokumentumokat és diagramokat generálok

A jövőben ezeket szeretném bővíteni extra validációs szabályokkal, valamint szeretném az MDX és a Riport előnyeit összetenni az új lekérdező nyelbve, ebből kifolyólag olyan domén specifikus problémákat is meg lehet majd oldani, amelyeket idáig nem lehetett csak mindkét nyelv együttes felhasználásával. Ilyen probléma például a költségfelosztás, ami azt jelenti hogy egy dimenzióról vagy egy csoportról szeretnénk átteni, a költségeket egy másik dimenzióra, vagy csoportra valamilyen arány szerint, ezt nyelvi elemként szeretném majd a jövőbe támogatni.