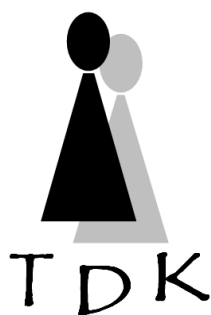




Adattárház fejlesztési kihívások és tapasztalatok a gyakorlatban

Tudományos Diákköri Dolgozat

Konzulens: Dr. Kő Andrea



**Farkas Gábor,
Grünstein István,
Németh Tamás Gábor**

2013.03.25.

Tartalom

1	Bevezetés	5
1.1	A Gemini Issue Tracking rendszer	5
2	Az alaphelyzet.....	7
2.1	A megoldás előnyei:	7
2.2	A megoldás hátrányai:	7
2.3	A megoldási javaslat.....	7
3	BI Business Case.....	8
3.1	Üzleti vezetők	9
3.2	Üzlet elemzési issuek.....	9
3.3	Költség-bevétel elemzés	10
3.4	Kockázat elemzés	11
3.5	Business Case a Consensenél	12
4	Az adatkocka.....	14
4.1	OLAP.....	14
4.2	Kocka terv.....	16
4.2.1	A kocka célja	17
4.2.2	Dimenziók	18
4.2.3	Metaadatok	19
5	ETL	21
5.1.1	Dimenziókezelés	21
5.1.2	Ténytábla	22
5.2	ETL terv készítés	23
5.2.1	Sztenderd ETL készítési folyamat.....	23
5.2.2	ETL terv készítés a Consense-nél	23
5.3	Fejlesztés	28
5.3.1	Project dimenzió.....	29
5.3.2	Idő dimenzió.....	36
5.3.3	Issues ténytábla	38
5.3.4	Kocka generálás	39
5.4	Adatbetöltés időzítése.....	39
6	Tesztelés.....	39

6.1.1	ETL teszt	39
6.1.2	Funkcionális teszt.....	40
6.1.3	Teljesítmény teszt.....	40
6.1.4	Biztonsági teszt	40
6.1.5	UAT.....	40
6.1.6	End-to-end teszt.....	41
6.2	A tesztelés menete a projektünkben	41
6.2.1	ETL adatkinyerési teszt.....	41
6.2.2	ETL adattartalom teszt	42
6.2.3	ETL SCD teszt	43
7	Riportolás.....	44
8	A projekt eredménye.....	46
9	Irodalomjegyzék	48
	Internetes források	48
10	Ábrajegyzék	49
11	Mellékletek.....	50

1 Bevezetés

A Consense Kft. és Egyetemünk Információrendszerek Tanszéke lehetőséget nyújtott az üzleti intelligencia tárgy keretében, hogy gyakorlatban szerezzünk tapasztalatot arról, hogyan készül egy BI megoldás. A projektben hárman vettünk részt, név szerint: Farkas Gábor, Grünstein István, Németh Tamás Gábor. A projekt megvalósításában nagy segítséget nyújtott a Consense vezetője, Köblös István és csapata. A hardver és szoftver háttérrel az Egyetem biztosította.

A projekt a Consense Kft. egy belső igényéből indult. A cég a hibák, kérelmek, igények (a továbbiakban az angol nyelvből átvett issue szót használom a dolgozat során, mivel a fogalom maga kifejezi a hibát, kérelmet, igényt) kezelésére a Countersoft által fejlesztett Gemini Issue Tracking rendszert használja.

1.1 A Gemini Issue Tracking rendszer

A Gemini egy összetett, mégis jól használható rendszer. A rendszer háttérében egy jól normalizált adatbázis található, ahol a különböző adattáblákban minden adatot megtalálunk, amelyek az issuekhoz kapcsolódnak. Tárolja az issue indulásának / zárásnak idejét, aktuális státuszát, milyen a prioritása, ki a felelőse, melyik projekt milyen komponenséhez tartozik stb. A Gemini adatbázis közel 100 adattáblából áll, és minden adattáblának legalább 5-10 attribútuma van. Így belátható, hogy egy elég nagy adathalmazról beszélhetünk.

A vezetés számára hasznos információt nyújthat a cég folyamatainak követése és az optimálisabb döntések érdekében, ha a Geminiben lévő adatokból különböző szűrések és rendezések által kimutatások készülnek. Milyen adatokról és azokból kinyerhető információkról van szó? A teljesség igénye nélkül leírok néhány mutatószámot, és hogy azok milyen információt nyújtanak a vezetés számára.

Az issuek a rendszerben státuszok szerint vannak csoportosítva. Néhány példa a státuszokra: kezdhető, folyamatban van, átadott, lezárt, törölve stb. Ha a döntéshozók számára olyan kimutatást készítünk, hogy a jelenlegi issuek milyen státuszban vannak, akkor levonható néhány következtetés. Például, ha az látható, hogy túl sok a nyitott issue, akkor feltételezhető, hogy valamilyen súlyosabb probléma van az egyik projekttel, folyamattal. A sok lezárt issue pedig mutathatja, hogy a hibajavítási, igénykezelési folyamatok megfelelően működnek a cégen belül.

Az issuek egy másik fontos nézőpontja a prioritás. Nagy átlagban az issue tracking rendszerek négy prioritás-típust különböztetnek meg. Ezek a következők: magas, közepes, alacsony, kritikus. A vezetők számára egy statisztika az issuek prioritás szerinti felosztásáról szintén nagyon szükséges lehet, mivel például a túl sok magas, vagy netán kritikus issue jelenléte a rendszerben valamilyen problémára utalhat, aminek jobb a lehető leghamarabb utána nézni.

Egy issueról sokat árulhat el a kora, tehát az issue felvételének dátuma és a jelenlegi dátum közötti különbség. Egy megfelelő szűrés által és korcsoportokra felosztva megtudható, hogy a rendszerben az issuek milyen idősök. Információkat szolgáltat a döntéshozók számára, ha a mutatók azt jelzik, hogy a rendszerben sok a több hetes, akár egy-két hónapos hibajegy. Az ilyen előfordulás jelezheti azt, hogy probléma van valahol és célszerű kivizsgálni az okát, majd lépéseket tenni a megoldás felé, ha tényleg szükséges.

A Gemini tárolja a projekteket és a hozzájuk kapcsolódó komponenseket. Megfelelő szűréseket használva grafikus eszközök segítségével láthatóvá tehető, hogy a jelenlegi folyamatokban mennyi issue van számon tartva, melyik projekthez mennyi tartozik, milyen státuszban vannak az adott issuek. A vezetők számára azért fontos látni ezeket a számokat, mert így látható, hogy melyik projekt éppen milyen állapotban van. Sőt, hosszú távú következtetéseket is levonhatunk például abból, ha hasonló projekteket vetünk össze. Amely komponensekhez kevés issue tartozik, az mutathatja, hogy ezeken a területeken erősebb a cég, mivel kevesebb hibával dolgozott. A sok hibajegyet tartalmazó projektek mutathatják, milyen területen lehet még továbbfejleszteni a cég tudásbázisát.

Összefoglalva a fent leírtakat, tehát a cég vezetősége számára a Geminiben olyan adatok vannak tárolva, amelyek megfelelő környezetben hasznos információkat nyújthatnak több szempontból is. A Consense számára a legfontosabb adatok az issuekról a következők:

- Státusza
- Kora
- Típusa
- Prioritása
- Melyik projekthez tartozik
- Melyik komponenshez tartozik
- Külső beszállítónál van-e

Továbbá ezeket a tulajdonságokat nem csak egy dimenzió mentén érdemes vizsgálni, hanem kettő vagy több nézőpontból. Például hasznos lehet kimutatni, hogy az A projekt B

komponenséhez milyen prioritású issuek tartoznak, vagy a X státuszú riportoknak milyen a típusa stb.

2 Az alaphelyzet

A fent leírtak alapján a Consense heti jelleggel készít kimutatást, amely a leírtakra ad választ. A statisztikákat Microsoft Excelben készülnek egy Gemini dump alapján. A mutatószámok kiterjednek a fent említett adatokra és grafikus megjelenítéssel, szűrési lehetőséggel mutatja a számokat.

2.1 A megoldás előnyei:

- Viszonylag könnyen és gyorsan elkészíthető.
- A pivot tábláknak köszönhetően több szempont alapján is megvizsgálhatjuk az adatokat.
- Olcsó megoldás.
- Nincs szükség szerver fenntartásra, felügyeletre.

2.2 A megoldás hátrányai:

- Nem automatikus a riport készítése, így erőforrást köt le az elkészítés.
- A változások követése nem megoldott.
- Az időbeliség követése sem teljesen megfelelő.
- Nem alkalmas ad-hoc lekérdezésre.
- Komplexebb statisztikákat nem vagy csak nehezen tudunk kimutatni.
- Nem képes nagy mennyiségű adattal megfelelően dolgozni.

2.3 A megoldási javaslat

Az eddig leírt problémák megoldása lehet egy adattárház készítés hozzá kapcsolódó reporting szerverrel, ami az alábbi részekből áll:

- Üzleti igény felmérés
- BI megoldás(ok) kiválasztása
- OLAP kockaterv készítés
- ETL terv készítés
- OLAP kocka fejlesztés
- ETL fejlesztés
- Tesztelés
- Riport fejlesztés

A projekt megvalósítása során szerepkörök szerint lettek felosztva a feladatok. Az egyik szerepkör a konzulens, aki az igénylő és a fejlesztő csapat közötti hidat képezi. Kettős szerepe azt jelenti, hogy felméri, és megérti, hogy mit szeretne az ügyfél, majd kommunikálja az

igényt a fejlesztés felé. Feladatai közé tartozik továbbá a kocka és ETL terv elkészítése, projekt haladásának nyomon követése, változások és igények kommunikálása minkét oldal felé, tesztelés folyamatának segítése tesztesetek kidolgozásával.

Másik szerepkör a fejlesztő, akinek feladat elkészíteni az OLAP kockát és az ETL package-
ket, a megadott tervek alapján.

A fejlesztések és tesztelések után a riportok elkészítése a következő lépés. A riport fejlesztőnek egyszerre kell érteni a fejlesztéshez és egy bizonyos szinten kreatívnak lennie a riport kinézetét illetően, ha erre nem érkezik külön design terv.

A közös projekt során a szerepköröket nem választottuk el teljes mértékben a folyamatok megfigyelése és a tapasztalatok gyűjtése miatt. Az üzleti igényfelmérést, a kocka- és ETL tervek készítését közösen végeztük. Külön választottuk az OLAP kocka és az ETL elkészítését, valamint a teszteseket és a riportok kidolgozását. Minden folyamathoz szakmai segítséget a Consense Kft. dolgozói biztosítottak.

3 BI Business Case

Az első lépés egy üzleti intelligencia megoldás bevezetésénél, hogy indoklásra kerüljön, miért érdemes megvalósítani, főleg azért, mert egy nagyvállalati környezet estén egy komplex megoldás több millió forintba kerülhet. Az indoklás legfontosabb nézőpontja, hogy mekkora a megvalósítás költsége és cserébe mennyi haszonra teszünk szert a jövőben. Egy BI megoldás esetén a hasznosság kiszámolása nem igazán egyszerű, hiszen hogyan számszerűsíthető, hogy a felsővezetők jobb döntéseket hozhatnak, az ügyfél elégedettség javult, nőtt a szervezet hírneve és hasonló? Mégis mikor egy vezetőnek meg kell hoznia a döntését, hogy befektet-e több millió forintot egy rendszerbe, látnia kell számokban, statisztikákban, hogy mi lesz a következmény.

Egy BI megoldás indoklásához a legjobb kezdő lépés, ha megállapításra kerülnek a szervezet hosszú és rövid távú stratégiai céljai. Ha tisztán láthatóak a jövőbeli elképzelések a vállalatot illetően, akkor az üzleti intelligencia megoldások széles palettája közül kiválasztásra kerülhetnek azok, amelyek a lehető legjobban képesek támogatni a vállalati törekvéseket. Mivel a döntést menedzserek, vállalati vezetők hozzák meg egy ilyen esetben, így fontos megjegyezni, hogy egy ilyen indoklásnak mindig üzlet központúnak kell lennie és nem technikainak.

Az üzleti képviselő (konzulens) felelős elsődlegesen azért, hogy a célok tisztázása után, meghatározza, milyen BI megoldás segíthetnek a jelenlegi problémák megoldására, milyen módon kerülnek támogatásra a stratégiai célok. Nagy segítséget nyújthat számára az IT osztály, akik rámutathatnak a jelenlegi rendszerek problémáira, továbbá rámutathatnak az egyes csoportok igényeire. Például más információk szükségesek egy felső vezető, egy projekt menedzser és egy csoportvezető számára, továbbá ezek az információ igények egyénenként, szakterületenként változóak.

Az IT az alábbi négy területen segítheti az üzleti indoklási folyamatot:

1. **Business drivers (Üzlet vezetők):** Meg kell határozni mik az üzlet legfőbb mozgatórugói, továbbá a BI eszközök kiválasztása a stratégiai célok támogatásához.
2. **Business analysis issues (Üzleti elemzési issuek):** Meg kell határozni mik azok a magas szinten levő információk, amelyek szükségesek az üzlet számára.
3. **Cost-benefit analysis (Költség-bevétel elemzés):** Számszerűsíteni kell az üzleti intelligencia megoldás bevezetésének költségeit és a várható bevételeket.
4. **Risk assessment (Kockázat elemzés):** Értékelni kell a kockázatokat technológiai, projektbeli, integrációbeli, szervezeti és pénzügyi szempontból.

(Moss, 2003)

3.1 Üzleti vezetők

Az indoklási folyamatban ebben a komponensben azonosítjuk a legfőbb vezetőket, akik a szervezet stratégiáját meghatározzák. Fontos belátni, hogy egy stratégiaalkotás során csak célirányok kerülnek kitűzésre, az akciók megtervezése alacsonyabb szinten történik. Így ebben a lépésben fontos megérteni a jövőbeni célokat és olyan megoldásokat javasolni, amelyek a cél elérésében segítenek. Például a felsővezetők azt a célt tűzik ki, hogy növeljük az ügyfelek elégedettségét. Ezt az üzenetet közvetítik a menedzserek és üzleti elemzők felé. Az ő feladatuk, hogy erre akciókat dolgozzanak ki, megoldásokat eszközöljenek. Itt többek között az alábbi kérdések merülhetnek fel: Hogyan tudjuk mérni az ügyfelek elégedettségét? Milyen adatok állnak a rendelkezésünkre? Milyen adatokra van szükség? Hogyan gyűjtjük össze? A BI szakértő feladata, hogy az adott igények kielégítésére, az arra alkalmas üzleti intelligencia megoldást javasolja.

3.2 Üzlet elemzési issuek

Az üzleti elemzők számára nagy kihívás egy teljes üzleti intelligencia megoldás bevezetése esetén a forrásadatok meghatározása. Nagyvállalati környezetben a használt rendszerek és a

hozzájuk kapcsolódó adatbázisok száma meghaladhatja akár a több százat is. Így az adatok a többféle forrás miatt nagyon heterogének. Ezen adatok meghatározása szintén fontos komponense egy üzleti indoklásnak.

Az adatok három legfőbb forrása:

- Belső vállalati adatok – logisztika, pénzügy, értékesítés, számlázás stb.
- Személyes adatok – táblázatok, dokumentumok, egyéb fájlok stb.
- Külső forrásból származó adatok – statisztikák, vásárlói szokások, vásárlói adatok stb.

3.3 Költség-bevétel elemzés

A Business Case legfőbb komponense (főleg a szponzorok számára) a költség-bevétel elemzés. Ahogy már említésre került, nem könnyű meghatározni egy üzleti intelligencia eszköz bevezetésének bevételeit. Nehéz megmondani az ügyfél elégedettség növekedés vagy a mutatóknak köszönhető optimálisabb döntéshozatalnak mennyi a tényleges bevétele. Az ilyen kézzel nem megfogható dolgokra csak becsléseket lehet adni korábbi statisztikákból, saját tapasztalatból.

Egy jó BI megoldás az alábbi bevételekkel járhat:

Bevétel növekedés

- Újabb piacok, piaci részek azonosítása
- Gyorsabb piacra kerülés
- Hatékonyabb eladás
- A lehetőségek gyorsabb felismerése

Profit növekedés

- Jobban célzott marketing
- Alulteljesítő termékcsoporthoz azonosítása
- Hatékonyabb árukezelés
- Belső elégedetlenségek azonosítása

Költségcsökkentés

- Felesleges termékek azonosítása
- Hibaszámok csökkenése
- Gyorsabb reagálás elcsúszott projektekre

Piaci rész növekedés

- A jelenlegi ügyfelek megtartása
- Új ügyfelek beszerzése
- Költséghatékonyabb működés

(Inmon, 2007)

3.4 Kockázat elemzés

Mivel a Business Case a vezetés számára készül és indokolja, miért lehet szükséges egy BI rendszer bevezetés, így a költség-haszon elemzésen kívül a legfontosabb információ a bevezetési döntés meghozatalához az előre látható kockázatok meghatározása. A kockázatokat legfőképpen az alábbi hat csoportba sorolható egy BI projekt során:

- Technológia kockázatok
- Komplexitásból származó kockázatok
- Az adatok integrációja miatti kockázatok
- Szervezeti kockázatok
- Projekt csapattagok képességeiből és elkötelezettségéből származó kockázatok
- Pénzügyi kockázatok

(Moss, 2003)

„Best practise” alapján érdemes minden területen megállapítani az egyes előre látható kockázatokat és besorolni színek alapján három csoportba. A kockázat bekövetkezésének lehetősége és annak hatása alapján a színek lehetnek: zöld, sárga, piros. A zöld kockázatok bekövetkezése esetén a projekt megvalósítás szinte zavartalanul mehet tovább. A sárga jelzésűek kihathatnak a projekt egyes részeire, így érdemes figyelmet fordítani rájuk. A piros besorolás a legrosszabb esetet mutatja, akár a projekt kudarcához, leállításához is vezethet.

Összefoglalva a fent leírtakat: egy üzleti intelligencia projekt indítása előtt meg kell indokolni miért szükséges, miért lehet hasznos a rendszer létrehozása. Így egy Business Case megírása az első lépés, amely üzleti szempontok alapján készül, mivel a vezetés, szponzorok és menedzserek számára készül, hiszen az ő döntésük alapján indulhat el a projekt.

A Business Case megírásának lépései a következők:

- Meghatározni az üzleti igényt
- Felmérni a jelenlegi megoldásokat
- Meghatározni a forrásokat és folyamatokat
- Felmérni a versenytársak megoldásait
- Meghatározni a BI alkalmazás célkitűzéseit
- BI megoldás(ok) ajánlása
- Költség-bevétel elemzés készítés
- Kockázat elemzés
- Értékelő jelentés írása

3.5 Business Case a Consensenél

A projekt megkezdésének első lépése csakúgy, mint ahogy az elmélet is előírja, az üzleti igény felmérése volt. Mivel az estünkben nem több milliós projektről és nem egy nagyvállalati környezetről beszélünk, így az előbb leírt elméleti lépések nem mindegyikén mentünk végig. A projekt során az alábbi lépéseket követtük:

1. Meghatároztuk az üzleti igényt
2. Felmértük a jelenlegi megoldásokat
3. Meghatároztuk a forrásokat és folyamatokat
4. Meghatároztuk a BI alkalmazás célkitűzését
5. BI megoldást ajánlottunk

A harmadik fejezetben említett Geminiből készült excel táblázat volt a tárgyalás kiindulási pontja. A Consense egyik alkalmazottja volt az igénylő, az ügyfél. Ő készítette el minden héten a táblázatot és a statisztikákat. A megbeszélésen mindhárman ott voltunk, továbbá a Consense vezetője és az üzleti igénylő. A megbeszélés során az alábbi üzleti igények merültek fel (az ügyfél a megoldás igénylőjének ügyfele):

- **Adatok időtávja**
 - A feladatok megoldásának időbelisége a fő információ, a trendeket szeretné követni a felhasználó
 - nyáron indította a projekt menedzsment a kimutatásokat, mert nem tudta követni a feladatok megoldását
 - a megoldás a teljes projekt időt le kell, hogy fedje
- **Szűrési lehetőségek**
 - **komponens**
 - a projekteket bontja le, de nincs benne minden projekt; nem mindegyik komponens jelenik meg a kimutatásban, de a Geminiben nincs olyan attribútum, ami alapján ezt le lehetne szűrni
 - ahol nem volt soha aktív jegy, az nem is jelenik meg a kimutatásokban
 - a komponens bontás nem ugyanazt a logikát követi minden projektnél (pl. egyik helyen ügyfél alapján, máshol más logika alapján)
 - meg kell állapítani milyen szabály alapján legyenek a komponensek kiválogatva
 - **típus**
 - garanciális hiba, támogatási igény, fejlesztési igény kell a kimutatásokba
 - a többi feladat típus nem látszik az ügyfél felé

- megoldásban minden típus legyen benne, de az ügyfél felé menő kimutatások csak a fenti hármat tartalmazzák
- fontos megvizsgálni, hogy projektenként változó-e az issuek típusai
- van-e olyan attribútum, ami alapján automatizálható az ügyfél által is látható típusok leválogatása
- **LS-re vár**
 - az alvállalkozó neve az LS (LandShark), azt mutatja, hogy az alvállalkozóra vár-e a megoldás
 - a Geminiben Partner ID mező tartalmazza a partneri azonosítót, ha van értéke, akkor LS-re vár
 - az LS-re vár flag számítási módját egyeztetni kell az igénylővel
- **prioritás**
 - ezt az ügyfél határozza meg
 - ha az ügyfél mindent magasra ad fel, akkor a projekt menedzsment egyeztet velük, hogy adjanak prioritást
 - a kimutatásokban egy adott időpillanatot vizsgálva az akkori prioritás kell, hogy megjelenjen
- **korcsoport**
 - három csoportba soroljuk 30+; 15-29; 0-14 nap
 - ügyfelenként és projektenként változhat a korcsoport bontása
 - újranyitott feladat az eredeti keletkezési dátummal szerepel a rendszerben
- **státusz**
 - sokféle státusz van, a kimutatásokon kevesebb szerepel
 - a státuszok egy része az ügyfél felé csoportosítva van
 - az alap excelben van a képlet, hogy mi alapján csoportosítjuk főstátusz és alstátuszra
 - megjelenítésben a státusz színe fontos
 - heti státusz riport az ügyfélnek
- **adatok kapcsolat**
 - a Geminiben van egy lista, amit lehet excelbe exportálni, ez a jelenlegi megoldás alapja
 - minden hétfő reggel frissül, de lehetnek alkalmi frissítések
 - riasztási feltételek kidolgozása további egyeztetést igényel
 - ügyfél áttekintő riport tartalma további egyeztetést igényel

- **További igények**

- milyen az erőforrások elosztása, vannak túlterhelt munkatársak?
- az egyes munkatársak tartják-e a feladat prioritizálását?
- előrejelzés, hogy milyen trend szerint fognak jönni a feladatok (adattányaszt)

A fent leírt igényekre a megoldás egy Geminivel összekapcsolt adattárház építése, amelyhez kapcsolódik egy riport server a mutatók elkészítéséhez. A megvalósítás következő lépése az OLAP kocka, majd abból következően az ETL megtervezése.

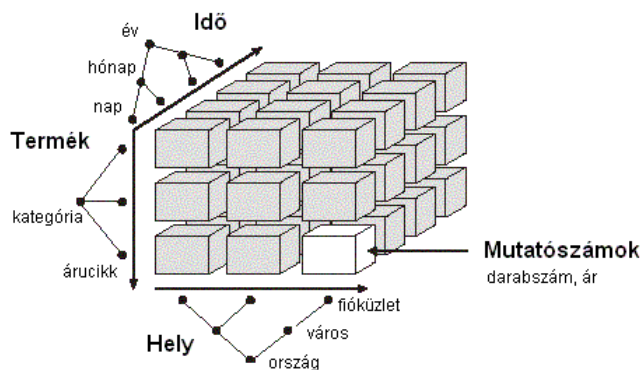
4 Az adatkocka

Miután az ügyféllel történő konzultáció elért egy bizonyos szintet és a business case kezdett összeállni, visszafelé gondolkodva elkezdhattuk megtervezni a kocka adatstruktúráját, a hierarchiákat és a függőségeket.

A következő néhány gondolatban megvizsgáljuk, mit is értünk az említett fogalmakon, valamint definiáljuk a hozzájuk kapcsolódó egyéb meghatározásokat is.

4.1 OLAP

A fő fogalmak megértéséhez először az OLAP (On-Line Analytical Processing) rendszereket mutatjuk be. Az OLAP rendszerek (ahogy a nevéből is olvashatjuk) olyan rendszerek, melyek valós idejű analitikus elemzéseket tesznek lehetővé. Sokszor ezt a megoldást az OLTP rendszerekkel szokták összehasonlítani, hiszen ebből kiderül, hogy mi az elvi különbség a két rendszer között. A két rendszer közötti különbségeket most nem részleteznénk hosszasan, a 1. mellékleten láthatjuk a legfontosabb eltéréseket. Ami minket igazán érint az az adatszerkezet. Az OLAP rendszerek adatszerkezete a szokványos relációs adatmodelltől eltérően multidimenzionális (általában csillagséma – később lesz róla szó), melynek több fontos vonzata is van. Egyrészt az általunk is megvalósított adatmodell csak részben normalizált, nem feltétlenül 3NF alakúak. Egy tranzakció-orientált működésnél ez komoly problémákat okozhat, de a mi célunk elsődlegesen a riportgenerálás hatékonyságának növelése, melyhez nem mindig a redundancia mentes modell a legmegfelelőbb. Másrészt az OLTP rendszerek technológiavezéreltségétől eltérően az OLAP rendszereket az üzleti céloknak megfelelően alakítják ki, tehát arra optimalizáltak, hogy az üzlet által kívánt specifikus igényeket a lehető leghatékonyabban tudják kiszolgálni.



1. ábra: Adatkocka (Sidló, 2004)

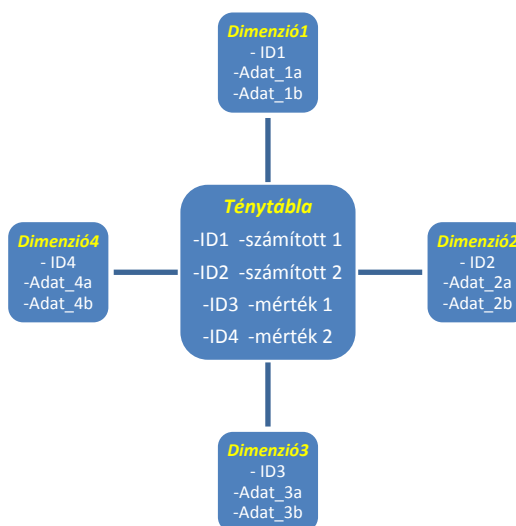
Az OLAP rendszerek építőkövei az adatkockák. A kocka elnevezés nem teljesen helyes, hiszen nem érzékelteti pontosan a multidimenzionális szerkezetet, de mivel négy vagy több dimenzióban egyelőre nehéz ábrázolni,

kénytelenek vagyunk beérni a háromdimenziós megjelenítéssel. Ennek

ellenére az adatkocka azért jó formátum, mivel megfelelően lehet érzékeltetni az elemzések készítésének egyszerűségét. Az alábbi ábrán a kocka élei a dimenziókat jelölik. A dimenziók azok az ismérvek, melyek alapján az elemzéseinket el akarjuk készíteni. A kocka egyes építőelemei a dimenziók által meghatározott egy darab aggregált mutatószám értéket tartalmaznak. Az ábrán a kiemelt építőelem például az egy fióküzletben egy árucikkből valamelyik napon eladott termékek darabszámát és árát jelöli.

Itt szeretném bevezetni a granularitás fogalmát. A granularitás az adatkocka legkisebb kimutatható egysége, melyet még vizsgálni tudunk az elemzések során. Nagyon fontos, hogy ezt még a kocka tervezés elején meghatározzuk, mivel ez lesz az egész kocka alapja.

Immáron tudjuk, hogy mi az adatkocka és mire lehet használni. A kocka felépítéséről azt is tudjuk, hogy több dimenzióból áll. Ám ez még nem határozza meg pontosan, hogy milyen szerkezetben kapcsolódnak egymáshoz ezek a dimenziók. A multidimenzionális szerkezet legelterjedtebb típusai a csillagséma (star join) és a hópéhely séma (snowflake scheme). A hópéhely séma lényegében csak a



2. ábra: Csillagséma (saját szerkesztés)

normalizáltság mértékében tér el a csillagsémától. A mi projektünkben csillagsémát használtunk, mivel nem láttuk különösebb értelmét a normalizálásnak és ezt az üzleti igény

sem kívánta meg. Másrészt a függőségek helyes beállítása a kívánt szintet tudta biztosítani. A csillagséma felépítését az alábbi ábrán láthatjuk.

A ténytábla (fact table) középen helyezkedik el és körülötte szerepelnek a dimenziótáblák (dimension table). Az egyes dimenziótáblák értelemszerűen az egyes dimenziók adatait tartalmazzák és általában egy ID jellegű kulccsal csatlakoznak a ténytáblához. A ténytáblában ezek a kulcsok, valamint a mértékek (measure) találhatóak, illetve a számított értékek, ha vannak.

A következő fogalom a hierarchia. A hierarchiák a dimenziótáblákon belül az egyes dimenzióelemek közötti szülő-gyerek viszonyt határozzák meg. Például az idő dimenzióban a január lehet az első negyedév gyereke, és az első negyedév az első félévé. A hierarchiák alkalmazása specifikusabb elemzéseket tesz lehetővé, segíti a lefűrást. Ugyanakkor nagy adattömegeknél lassíthatja a rendszer működését. A hierarchiáknak szintjei vannak, előbbi példában lehetnek szintek a „hónapok”, a „negyedévek” és a „félévek”.

Az utolsó fogalom, ami a kockatervhez kapcsolódik a függőségek meghatározása. Ez a fogalom a relációs és multidimenziós szerkezetben is megjelenik. Tulajdonképpen a függőség multidimenziós környezetben funkcionális függőséget jelent egy dimenzióon belül. Úgy tudjuk érthetően megfogalmazni, hogy egy „a” attribútum függésben áll egy másik „b” attribútumtól abban az esetben, ha a „b” attribútum értéke egyértelműen meghatározza „a” attribútum értékét.

A fenti meghatározásokat fontosnak tartottuk tisztázni, hiszen e fogalmak nélkül nehéz lenne megérteni a gyakorlatban elkészített kocka struktúráját, működését. A további szakkifejezéseket akkor fogjuk magyarázni, mikor azok logikailag következnek a tanulmányban. A következőkben bemutatom a kocka tervét, elkészítésének, valamint a megvalósításának folyamatát.

4.2 Kocka terv

Az adatkocka tervét egy sablonban vázoltuk fel. Ez két okból is megkönnyítette a dolgunkat: egyrészt tudtuk, hogy mely adatokat kell meghatároznunk ahhoz, hogy a kockát le tudjuk majd fejleszteni. Másrészt a sablon megléte önmagában adott egy kiindulópontot és egy alaphelyzetet a kocka tervezéséhez (pl. mintaadatok révén). A sablon bizonyos részeit nem is használtuk fel, de elméleti tudásunkat gyarapította, hogy felmerült bizonyos „metaadatok” szükségességének kérdése.

A kockatervben igyekeztünk olyan jelöléseket, fogalmakat használni, melyekből a fejlesztő számára egyértelműen kiderül, hogy mit is szeretnénk a gyakorlatban látni. Egy nagyobb vállalati környezetben (ahol a terveket nem feltétlenül az az ember gyártja, aki magát a kockát fejleszti) ez fontos momentum lehet, hiszen a tervezési szakaszt nagyon el tudja nyújtani, ha a fejlesztőnek folyamatosan utána kell járni, hogy mit is akar a megrendelő. Ez alapvetően nem is a fejlesztő feladata lenne.

4.2.1 A kocka célja

Az első fontos elem a kockaterv leírás létrehozásában, hogy mi az üzleti cél, igény melyet el akarunk érni. Az üzleti igény a mi esetünkben: relációs adatbázist alapul véve egy olyan adatmodell létrehozása, melyet hatékonyan fel lehet használni riportok automatikus generálásához, valamint elemzések készítéséhez. A projekt célja, hogy a megrendelő egy automatizált rendszeren keresztül folyamatosan nyomon tudja követni a hibajegyek számát a meghatározott szempontok szerint. Az ügyféllel történt konzultáció során megfogalmazódtak olyan kérdések a megrendelő részéről, hogy „Milyen hatékony a hibajegyek javítása?” , „Mennyire hatékony az erőforrás-kiosztás?”, vagy „Jósolható-e, hogy mikor ugrik meg a hibajegyek száma?”. Ezek a kérdések folyamatosan a szemünk előtt kellett lebegjenek az adatstruktúra kialakítása közben, hiszen ezek lesznek az elemzések fő irányvonalai.

Mindezek alapján meg tudtuk határozni a kocka granularitását. A mi esetünkben ez az egyszerűség kedvéért (és mert semmi nem indokolta, hogy másként legyen) hibajegy*napok (adatátöltés napjai) lett, de meg kell jegyeznünk, hogy sok esetben ez nem ennyire triviális. Ez a granularitás azt jelentette, hogy a legkisebb egység, melyet a kimutatások során tudunk vizsgálni, a „hibajegy betöltési naponként” lesz.

A következő fontos tervezési elem a mértékek meghatározása, tehát hogy mit is akarunk látni a kimutatásokon. A projekt ezt úgy határozta meg, hogy alapvetően az összes hibajegyek számát szeretnénk vizsgálni, illetve fontos lenne a csak „aktív” hibajegyek számát is külön mérni. Mivel ez meghatározható az egyik attribútumból, nem kellett volna feltétlenül mértékként létrehozni, de mivel a riportkészítést jelentősen segítette, valamint szinte semmilyen pótlólagos erőforrást nem igényel, ezt második mérőszámként mégis létrehoztuk. Az összes hibajegy száma egy számláló (count) aggregáció, míg az aktív hibajegyek száma egy összegző (sum) típusú aggregáció lett. (Az hibajegyek aktivitásának eldöntésére a tény táblában egy külön bináris attribútumot hoztunk létre, melynek értéke akkor „1”, ha a hibajegy aktív, így az összegzéskor az aktív hibajegyek darabszáma kerül meghatározásra.)

A mértékekből lehetőség lett volna számított mezőket kalkulálni (pl. az aktív hibajegyek hány százalékát teszik ki az összes hibajegynek), de jelen esetben nem volt szükség ilyen értékek számítására.

4.2.2 Dimenziók

A terv következő pontja a dimenziók meghatározása. Ahogy azt korábban már ismertettük, a kiindulási alap egy manuálisan generált grafikon volt MS Excel környezetben. Meg kellett vizsgálnunk, hogy milyen elemzések születtek, hiszen ezeknek mindenképpen szerepelniük kellett a végső riportban is az üzleti igény szerint. Gyorsan kiderült, hogy nagyrészt az ügyfél által szükségesnek vélt kimutatások elemzési szempontjai (ez gyakorlatban a kapott excel tábla lapok neveinek feleltek meg) lesznek a dimenziók. Persze voltak olyan vizsgálati szempontok is (pl. hibajegyek prioritása vagy, hogy alvállalkozói hibajegye-e), melyekre felesleges lett volna új dimenziót létrehozni, hiszen ezeknek csupán egy attribútumuk lett volna (az azonosító mellett). Végül összesen hét dimenziót tartottunk szükségesnek, hogy a kívánt riportokat és elemzéseket létre tudjuk hozni. A dimenziók listája a következő:

Dimenziók	
Dimenzió név	Megjelenési név
Projekt	Projekt
Típus	Típus
Idő	Pillanatkép dátuma
Issue kora	Issue kora
Státusz	Státusz
Felelős	Felelős
Issue	Issue

3. ábra: Dimenziólista

A dimenziók bemutatása előtt szeretnék tisztázni még egy fogalmat, mellyel most találkozunk először a kockatervezés során és nemcsak a kockafejlesztésnél, hanem az ETL folyamatnál is jelentősége lesz. Mivel a kockába egy meglévő relációs adatbázisból töltjük az adatokat, a rekordok azonosítása az új struktúrában nem triviális. Emiatt minden dimenziótáblában szerepeltetünk egy mesterséges kulcs (surrogate key , SK) azonosítót. Ez azért fontos, mert a relációs adatbázisból érkező adatok közt az eltérő modellszerkezet miatt inkonzisztencia léphetne fel a dimenziókban. Éppen ezért az egyes rekordoknak nem az azonosító lesz az elsődleges kulcsa, mint a forrásadatbázisban, hanem az általunk adott SK. Ez azért is hasznos, mivel a fejlesztők így pontosan be tudják azonosítani saját adattáruk azonosítóját, valamint az esetleges karbantartási műveleteknél is átláthatóbbá, áttekinthetőbbé teszi a fejlesztő vagy a karbantartást végző számára az adatbázist. A forrásból jövő azonosítót

természetesen érdemes megtartani, hiszen az (üzleti) felhasználói oldal csak a saját azonosítója alapján tudja azonosítani az adott rekordot. Ezt találóan üzleti kulcsnak (business key, BK) hívjuk, az ETL folyamatoknál lesz róla szó részletesen.

A dimenziótábláknál minden esetben meghatároztuk a dimenzió nevét, leírását, az adatforrás helyét, nevét, valamint az ún. „AttributeAllMemberName” és „UnknownMemberName” értékeket. Előbbi azt mutatja, hogy amennyiben a dimenzió összes elemét ki szeretnénk választani, akkor mi legyen a „Mindent kijelöl” címke. Utóbbi pedig az ismeretlen (null) értékeket jelöli. A megrendelővel egyébként ezt is érdemes előre tisztázni, hiszen elképzelhető, hogy neki nincs szüksége ilyen értékekre vagy nem azt az értéket szeretné látni, amit a fejlesztő logikusnak ítél.

Ami még további általános dimenziótábla-jellemző, hogy minden tábla minden sorának van egy bináris érvényességi mezője (valid), valamint érvényességi időtartama (valid_from, valid_to). Ezek a mezők mutatják meg, hogy az adott sor érvényes-e, illetve mikortól meddig tart érvényessége. Miért is fontos ez? Egyrészt a folyamatos töltések miatt bizonyos sorok érvényessége lejárhat. A másik ok, hogy amennyiben elemzéseket akarunk készíteni (pl.: adatbányászati feladatok), fontos lehet, hogy az adott sor „életét” nyomon tudjuk követni. Ezzel részletesebben szintén az ETL fejlesztés kapcsán fogunk foglalkozni.

4.2.3 Metaadatok

A következőkben röviden áttekintjük, hogy az egyes dimenziókon belül milyen attribútum-adatokat (metaadatok) kell megadnunk.

Az attribútum nevének, amennyiben lehetséges, olyan értéket célszerű megadni, mely beszédes és egyedi az egész adatbázisban. A mi hét dimenziós rendszerünkben végülis olyan nagy galibát nem okozna ennek figyelmen kívül hagyása, de egy nagyobb – 20-30 dimenziós – kockában már komoly problémákat okozhat, ha azonos nevű attribútumokkal találkozunk a fejlesztő. Az informatikusok, fejlesztők nagy része hajlamos (pl. megszokásból) ékezetmentes, általános neveket adni az attribútumoknak, pedig itt adhatunk beszédes, magyar nyelvű megnevezéseket is (pl. főtátusz megnevezése, alstátusz azonosítója).

A felhasználói élményt és a megjelenítés áttekinthetőségét javíthatja, ha a user számára érdektelen adatokat elrejtjük. Jellemzően a hibajegy generált azonosítója, vagy az általunk létrehozott SK nem fontos az ügyfél számára. A dimenziótáblákban tehát érdemes szerepeltetni egy bináris mezőt annak eldöntésére, hogy az adatot láthatóvá akarjuk-e tenni, vagy sem.

További, átláthatóságot javító attribútum tulajdonság a kulcstípusú attribútumok megjelölése. Az általunk fejlesztett kockában csak az SK-k voltak kulcstípusúak, de ez nem feltétlenül van így, hiszen előfordulhat, hogy egy táblában az elsődleges kulcsot (egyértelmű azonosító) két mező adja együttesen.

A kockatervben kell szerepeltetni, hogy a kialakítandó mezőt honnan fogjuk feltölteni értékkel a forrásadatbázisból. Meg kell különböztetnünk a kulcstábla és az értéktábla (valamint a hozzájuk kapcsolódó típus és mező) funkcióját. A legtöbb esetben a két tábla megegyezik, de nem mindig van így. A kulcstábla az a forrástábla, melyben a hivatkozni kívánt adat elsődleges azonosítója (kulcsa) található. Az értéktábla pedig az a forrástábla, melyben a kívánt adat valójában megtalálható. A terv készítése során csupán egy alkalommal tért el egymástól a két tábla. Mivel a relációs adatbázisok tábláiban az elsődleges kulcs által azonosított adat megnevezése (értéke) általában ugyanabban a táblában van, mint az kulcs, ezért a kulcstábla és az értéktábla a legtöbb esetben megegyezik, de érdemes lehet mindig ellenőrizni ezt.

Miután pontosan meghatároztuk, hogy az egyes dimenziótáblákban milyen mezőket akarunk megjeleníteni, meg kell állapítanunk az egyes mezők közötti függőségeket. A függőségeket tulajdonképpen felfoghatjuk egy belső kulcsolásnak a dimenzióon belül, hiszen itt is funkcionális függőségről beszélünk.

A hierarchiák a lefűrást teszik lehetővé, mely a mi projektünket kevésbé érintette, de sok esetben nagyon hasznos lehet (pl. vezetői riportok, dashboardok készítésénél). Természetesen minden esetben meghatároztuk a hierarchiákat.

Egy függőségeket és hierarchiákat is tartalmazó dimenzióterv képe a 2. mellékleten látható.

A dimenziótáblák létrejöttével gyakorlatilag előállt a ténytábla is, hiszen a dimenziók SK kulcsai hivatkoznak a ténytábla attribútumaira, melyet persze külön be kell állítani a fejlesztőeszközben. A ténytáblát még további két mezővel egészítettük ki, melyek kritériumok voltak az ügyfél számára, viszont dimenziótáblát felesleges lett volna készíteni miattuk. A „prioritás” és „partnerre vár?” mezőket ezért közvetlenül a forrásból töltjük a ténytáblába.

A fenti kockaterv alapján magát a kockát microsoftos környezetben SQL Server Business Intelligence Development Studioval (BIDS) fejlesztettük, mely minden szempontból tökéletes eszköznek bizonyult.

5 ETL

Az adattárház adatokkal való feltöltéséért az adatbetöltés folyamata felel, amely három fő lépésből áll: adatkinyerés (extract), transzformációk (transform), adatok betöltése (load). A következőkben az adattárház ETL folyamatának fejlesztését mutatjuk be. A konkrét fejlesztés bemutatása előtt azonban néhány fogalmat tisztáznunk kell, amelyek ismerete elengedhetetlen a folyamat megértéséhez.

5.1.1 Dimenziókezelés

Az adattárház időfüggő, azaz a bekerült adatok idősorosan és hosszú ideig tárolódnak, így a dimenziótáblák rekordjainak idővel történő változását is kezelni kell. Mivel a dimenziók változása lassú ezért ezeket lassan változó dimenzióknak (Slowly Changing Dimensions, SCD) is szokás nevezni. Az SCD kezelésnek három alap típusát különböztetjük meg. (Kimball-Caserta, 2004)

SCD1-es követés

Az első típusú dimenziókezelés során, ha egy rekord megváltozik, azt egyszerűen felülírjuk az új értékkel. Ennek előnye, hogy könnyű implementálni, viszont ezzel elveszítjük az adott rekord historikus változásának követését. (Kimball-Caserta, 2004)

SCD2-es követés

A második típus során, ha a dimenziótábla egy sorában változás történik, egy új sort hozunk létre az új attribútumok rögzítésére. Ilyenkor egy rekord lényegében kétszer szerepel a táblában, amik között különbséget kell tudni tenni. Erre szolgálnak a segéd oszlopok, amelyekben a rekord érvényességét kezelik: mikortól, meddig, aktuális érvényessége. (Kimball-Caserta, 2004)

SCD3-as követés

A harmadik típus célja, hogy ha egy rekord attribútuma megváltozik, akkor annak új és az előző állapotát is tárolni tudjuk a rekordon belül, új rekord létrehozása nélkül. Ez egy új sor bevezetésével érhető el úgy, hogy az új oszlop mindig az attribútum megelőző állapotát mutatja és az attribútumot magát egyszerűen, mint az SCD1-es kezelésnél felülírjuk. (Kimball-Caserta, 2004)

Az alap típusokon kívül léteznek még hibrid követések valamint az utóbbi években megjelent egy 0. típusú követés. A hibrid követések a három alap típus kombinálásából keletkeznek, de a dolgozat szempontjából nem relevánsak ezért tárgyalásukat nem tartottuk fontosnak. A 0. típusú követés lényege, hogy ha egy rekord SCD0-ás attribútuma megváltozik a forrásrendszerben, azt nem változtatjuk meg az adattárházban. Ez a típusú követés nem szerepel a Ralph Kimball és Joe Caserta szerzőpáros The data warehouse ETL toolkit című könyvében ezért nem is soroltam az alap követések közé. Vincent Rainardi az adattárházokról és üzleti intelligenciáról szóló blogjában 2011-ben készített egy bejegyzést az újonnan megjelent típusról, melyben leírja a feltételezhető származását és alkalmazási területeit. (Rainardi, 2011)

5.1.2 Ténytábla

A ténytábla tartalmazza a tényadatokat, melyek a felhasználók által elemezni kívánt mérőszámokból és mutatókból állnak. A ténytáblának három fajtáját különböztetjük meg attól függően, hogy a tábla egy rekordja mit reprezentál. (Kimball-Caserta, 2004)

Tranzakciós ténytábla

A tranzakciós ténytáblában, mint a nevéből is adódik, valamilyen üzleti tranzakciókat tárolunk idősorosan. Az operatív tevékenységek legkisebb felbontása a tranzakciók szintje, a tábla egy sora egy ilyen tranzakciót reprezentál. A tranzakciós ténytábla a három típus közül a legnagyobb és a legrésztetesebb. (Kimball-Caserta, 2004)

Pillanatfelvételek

A második típusú ténytáblába előre meghatározott időközönként készített pillanatfelvételeket tárolunk a forrásrendszerről. A pillanatfelvétel a két periódus közötti intervallumot reprezentálja. (Kimball-Caserta, 2004)

Gyűjtött pillanatfelvételek

A harmadik típusú ténytábla egy jól definiált kezdő és végponttal rendelkező folyamat leírására szolgál, mint például rendelésteljesítés vagy a legtöbb workflow. Nem alkalmas hosszadalmas és folytonos folyamatok követésére, mint például gyártási folyamatok vagy bankszámlakövetés. Az egyik különlegessége a sok dátum típusú mező, amelyekbe a folyamat egyes mérföldkövei bekövetkeztének dátuma kerül. (Kimball-Caserta, 2004)

5.2 ETL terv készítés

5.2.1 Sztenderd ETL készítési folyamat

Üzleti követelmények

Az ETL tervezés alapja, hogy megértésre kerüljenek a korábban felmért és dokumentált üzleti követelmények. Továbbá a követelmények megértésén túl szükséges, hogy megállapításra kerüljenek a szükséges adatok is, ha már elérhetőek.

- Megvalósítandók
 - Logikai leírása, hogyan fog történni az ETL folyamat, tehát a kinyerés, átalakítás és betöltés.
 - Az igénylővel alá kell íratni a tervet
- Kimenet
 - Üzleti ETL igénylő sablon

Fizikai ETL tervezés

Ebben a pontban meg kell határozni a ki- és bemeneti adatokat. Továbbá azonosítani kell az adatok forrását és a betöltés helyét is „tábla/mező” formában, az adatok várható méretét, típusát, SCD kezelést, valamint a transzformáció módját.

- Megvalósítandók
 - ki-és bementő adatok meghatározása
 - forrás és cél adattáblák helye
 - becslés az adatok méretére
 - transzformáció módja
- Kimenet
 - ETL fizikai terv

5.2.2 ETL terv készítés a Consense-nél

ETL terv sablon

Az ETL tervet a csapat közösen készítette el a Consense alkalmazottak szakértői támogatásával. A Consense a rendelkezésünkre bocsájtott egy általuk használt ETL terv sablont. A sablon egy excel táblázat, amelynek minden egyes munkafüzete egy-egy adatbázis a kockában.

Verzió:	Felelős:	Táblák:	GRANULARITÁS:	Szűrés:					
CélMező	Cél mező típusa	Mező leírás	Forrás (Tábla)	Forrás (mező)	Művelet	megjegyzés	ki	mikor	mit

4. ábra: ETL terv sablon (saját szerkesztés)

A táblázat a következő részekből áll:

- Verzió – verziókövetés miatt, meg kell adni egy verziószámot a tervhez, hogy követhetőek legyenek a változások.
- Felelős – Az ETL terv elkészítéséért felelős személy.
- Táblák – Azt mutatja meg, hogy milyen forrástáblákat használunk fel a tervezésben. Az ETL fejlesztés során segíti a fejlesztők munkáját.
- Granularitás – A forrástábla méretét adja meg, úgy hogy megadunk egy olyan attribútumot, ami legnagyobb elemszámot mutatja. Ez általában valamilyen egyedi azonosító.
- Szűrés – Azt mutatja, hogy van-e szükség a forrás adatok szűrésére.
- Célmező – A kockában az adott dimenzióra vonatkozó mező neve, ahova adatot akarunk feltölteni.
- Célmező típusa – Fontos megállapítani a mezőtípusát ahova feltöltjük, hogy az ETL fejlesztés során ne okozzon problémát az eltérő mezőtípusok. Lehet szám (integer, double, float), szöveg (char, varchar), dátum.
- Mező leírás – Egy rövid leírás a mező tartalmáról. Pl típus azonosító, státusz leírás stb.
- Forrás (Tábla) – Az adat forrástáblájának neve.
- Forrás (Mező) – Az adat helye a forrástáblában mezőszinten. Szintén ebben a részben kerül leírásra az adat transzformációjának, keresésének módja pszeudókódban.
- Művelet – Az SCD kezelés leírásának helye.
- Megjegyzés – Bármilyen megjegyzés a tervhez.
- Ki, mikor, mit – Bármilyen változtatás a terven ide kerül dokumentációra. Az ellenőrizhetőség miatt dokumentálni kell, hogy ki végezte a változtatást, mikor tette és hogy mit változtatott meg.

ETL terv készítés a Consense-nél

A fejezetben bemutatásra kerül, hogyan történt a projektünk ETL tervének elkészítése. A tervet mind a fact, mind pedig a dimenziókra el kell készíteni. A dimenzió táblák ETL tervezését a d_status tábla elkészítésén keresztül kerül bemutatásra.

Első lépésben, a kockatervben már meghatározott célmezőket átimportáljuk a táblázat „Célmező” oszlopába. Majd az SCD kezelés miatt kiegészítjük három mezővel, amelyek minden esetben megegyeznek (valid to; valid from; valid). A „Célmező típusa” oszlopot szintén azonnal fel tudjuk tölteni, mivel ezek az adatok szintén rendelkezésre állnak számunkra a kockatervben. Az SCD kezelés miatt a három sorban a „valid to” és „valid from” mezők dátum típusúak, míg a „valid” egy 1 karakterből álló szöveg típus, aminek Y-N értéket vesz fel.

Verzió: 1.0 (2012.10.17) Felelős: Gábor Táblák: gemini_issuestatus GRANULARITÁS: gemini_issuestatus.s Szűrés: tatusid									
CélMező	Cél mező típusa	Mező leírás	Forrás (Tábla)	Forrás (mező)	Művelet	megjegyzés	ki	mikor	mit
status_sk	EGÉSZ SZÁM	surrogate key							
status_id	EGÉSZ SZÁM	Státusz gemini id-je							
description	SZÖVEG(255)	Státusz megnevezése							
main_status	SZÖVEG(255)	főstátuszok							
sub_status	SZÖVEG(255)	alstátuszok							
final	SZÖVEG(1)	lezárt-e a státusz							
image	SZÖVEG(255)	A státuszhoz tartozó kép elérési útja a Gemini szerveren (relatív url)							
valid_from	DÁTUM	Érvényesség kezdete							
valid_to	DÁTUM	Érvényesség vége							
valid	SZÖVEG(1)	Aktív-e a rekord							

5. ábra: ETL tervezés első lépései (saját szerkesztés)

A következő lépés, hogy a kitöltjük a mező leírás oszlopot. Ahogy az ábrán is látszik ezek a következők:

status_sk – surrogate key: Az OLAP kockában szereplő általunk adott egyedi azonosító. Az SCD kezelés szempontjából fontos, hogy ne a forrás adatforrás egyedi azonosítóit használjuk a kockában, mivel annak változását ilyen módon nem tudnánk kezelni.

- status_id – Státusz gemini id-je: A forrás adatbázisban lévő egyedi azonosítók.
- description – Státusz megnevezés: A Geminiben található státuszok megnevezését mutatja (pl. új, kezdhető, folyamatban, törölve stb.)

- `main_status` – főstátuszok: Az üzleti igény alapján a státuszok főstátuszok szerinti csoportosítása.
- `sub_status` – alstátusz: Az üzleti igény alapján a státuszok alstátuszok szerinti csoportosítása.
- `final` – lezárt-e a státusz: Az adott issue lezárásra került-e?
- `image` - A státuszhoz tartozó kép elérési útja a Gemini szerveren (relatív url)
- `valid from` – érvényesség kezdete
- `valid to` – érvényesség vége
- `valid` – Aktív-e a rekord

A terv már tartalmazza, hogy milyen adatokat szeretnénk látni az OLAP kocka státusz dimenziójában. Az ETL tervezés következő lépése, hogy megállapításra kerüljenek az adatok forrásbeli helyei, és ha szükséges, milyen adat transzformációra van szükség.

A „Forrás (Tábla)” oszlop kitöltése a kockaterve alapján történik, mivel ott már szintén meghatároztuk, hogy a Gemini melyik adattábláit kell felhasználni. A „Forrás (mező)” oszlop kitöltése két féle módon történik. Ha nincs szükség a feltöltés során transzformációra, úgy az eredeti értékkészlet kerül átvételre. Ilyen esetben egyszerűen meg kell adni a Geminiben lévő mezőt. Például a `description` mező feltöltése az eredeti forrásadatból lesz feltöltve, aminek így táblanévként a `gemini_issuestatus`-t kell megadni és mezőértékként a `statusdesc`-t.

Amennyiben transzformálni szeretnénk az adatokat, úgy a „Forrás(mező)” részben kell leírni a transzformálást szabályát pszeudókóddal. Az ilyen megoldás előnye, hogy a tervezőnek nem kell ismernie a pontos programnyelvet, a fejlesztő mégis megérti a leírt szabályt, mert a természetes nyelv és programozási nyelv elemeit tartalmazza a kód.

Egy példa a pszeudókódra:

```
„CASE WHEN statusdesc = 'Új' OR statusdesc = 'Kezdheto' THEN 'Nem történik semmi'
WHEN statusdesc = 'Folyamatban' OR statusdesc = 'Kész' OR statusdesc = 'Újra nyitva' OR
statusdesc = 'Belső akadály' THEN 'Dolgozunk rajta' WHEN statusdesc = 'Külső akadály'
THEN 'Ügyfélre vár' ELSE statusdesc”
```

A fejlesztő számára ez a kód ezt jelenti:

Ha a forrásban lévő `statusdesc` mező értéke „Új” vagy „Kezdheto” akkor az alstátusz a kockában „Nem történik semmi”. Ha az érték „Folyamatban” vagy „Kész” vagy „Újra nyitva”

vagy „Belső akadály” akkor az alstátusz csoport a „Dolgozunk rajta”. Ha „Külső akadály” akkor az alstátusza „Ügyfélre vár”. Minden más esetben a kockában az alstátusz mezőben a forrásban szereplő értékét veszi fel.

A tervezés következő lépése, hogy megállapításra kerüljenek az SCD kezelések, amit a táblázat „Művelet” oszlopa mutat. Az oszlop az alábbi értékeket veheti fel:

- SK – Surrogate Key
- BK – Business Key
- SCD0
- SCD1
- SCD2

Verzió: 1.0 (2012.10.17) Felelős: Gábor Táblák: gemini_issuestatus GRANULARITÁS: gemini_issuestatus.s Szűrés: tatusid									
CélMező	Cél mező típusa	Mező leírás	Forrás (Tábla)	Forrás (mező)	Művelet	megjegyzés	ki	mikor	mit
status_sk	EGÉSZ SZÁM	surrogate key			SK				
status_id	EGÉSZ SZÁM	Státusz gemini id-je	gemini_issuestatus	statusid	BK				
description	SZÖVEG(255)	Státusz megnevezése	gemini_issuestatus	statusdesc	SCD2				
main_status	SZÖVEG(255)	főstátuszok		CASE WHEN statusdesc = 'Lezárt' THEN statusdesc WHEN statusdesc = 'Átadott' THEN statusdesc WHEN statusdesc = 'Elhalasztva' THEN statusdesc WHEN statusdesc = 'Törölve' THEN statusdesc ELSE 'Aktív'	SCD1				
sub_status	SZÖVEG(255)	alstátuszok		CASE WHEN statusdesc = 'Új' OR statusdesc = 'Kezdeti' THEN 'Nem történik semmi' WHEN statusdesc = 'Folyamatban' OR statusdesc = 'Kész' OR statusdesc = 'Újra nyitva' OR statusdesc = 'Belső akadály' THEN 'Dolgozunk rajta' WHEN statusdesc = 'Külső akadály' THEN 'Ügyfélre vár' ELSE statusdesc	SCD1				
final	SZÖVEG(1)	lezárt-e a státusz	gemini_issuestatus	CASE isfinal = "true" THEN "I" ELSE "N"	SCD1				
image	SZÖVEG(255)	A státuszhoz tartozó kép elérési útja a Gemini szerveren (relatív url)	gemini_issuestatus	imagepath	SCD1				
valid_from	DÁTUM	Érvényesség kezdete							
valid_to	DÁTUM	Érvényesség vége							
valid	SZÖVEG(1)	Aktív-e a rekord							

6. ábra: Elkészült ETL terv a status dimenzióra (saját szerkesztés)

Utolsó lépésként az ETL terv fejlécében kitöltjük a „Verzió”, „Felelős” oszlopokat. A fejlesztők és a későbbi átvizsgálás miatt előnyös ha felsoroljuk a „Táblák oszlopban ” az adott dimenzió ETL –je során felhasznált forrástáblákat. A példa esetben egy táblából dolgoztunk, de egy összetettebb dimenzió tervezése esetén ez akár több forrástábla is lehet (lásd. 4. melléklet). Majd meghatározni a méret megbecsülése szempontja miatt az adott adatbázis granularitását. A példa esetben ez a mező a forrásadatban lévő egyedi azonosító, tehát a „gemini_issuestatus.statusid”.

Az ETL tervezés során a fact táblát (lásd. 5. melléklet) és az összes dimenzió táblát a fent leírt módon külön-külön el kell készíteni a sablon excel táblázat alapján. Az elkészült ETL terv a

fejlesztőhöz kerül, aki megkezdheti a terv megvalósítását ETL fejlesztői eszközök és kódok segítségével.

5.3 Fejlesztés

Az ETL fejlesztést a Microsoft Business Intelligence Development Studio (BIDS) fejlesztőeszközzel végeztük. Maga az adatbetöltés SQL Server Integration Services (SSIS) csomagok (package) szekvenciális lefutásából áll, melyek időzítését és sorrendiségét később részletezünk. Az SSIS az a platform, amely lehetővé teszi vállalati szintű adatintegrációs és transzformációs feladatok elvégzését, mely feladatokat SSIS package-ekben írják le. (MSDN, é.n.) Egy package tartalmazhat csupán egyetlen műveletet is (pl: egy adattábla sorainak átmásolása egy másik táblába), de akár összetett lépések sorozatát is (adat kinyerési, tisztítási és betöltési folyamatok együttesét). Az így elkészített package-eket a Microsoft Windows SQL Agent szolgáltatása képes időzítve futtatni.

Az adatbetöltés során a forrás adatbázisból érkező adatokat nem egyszerűen beillesztjük a céladatbázisunkba, hanem a megfelelő dimenziókezelés érdekében össze kell tudni vetni a két rendszer eltéréseit és azokat kezelni. A dimenziók SCD követéséhez egy nem Microsoft fejlesztésű SSIS Dimension Merge nevű komponenst használtunk, mely egy a BIDS-be integrálható open-source kiegészítő. A kiegészítő használatáról a dimenziók fejlesztésénél bővebben kitérünk.

Az adatbetöltésünket úgy tagoltuk, hogy egy package egy tábla töltéséért felelős az adattárházban. Ezzel a tagolással könnyen átlátható a betöltés folyamata, illetve a futás közben felmerülő hibák forrása gyorsabban beazonosítható.

A fejlesztések menetét két dimenzión illetve a ténytáblán keresztül mutatjuk be:

- project dimenzió
- idő dimenzió
- issues ténytábla

Az első jól bemutatja, hogy a dimenzió táblák fejlesztése során milyen lépéseken haladtunk végig, a második esetben szintén dimenzió tábla töltéséről van szó, viszont az idő dimenzióról, melynek töltése eltér a többitől. Végül a ténytábla töltéséhez szükséges műveleteket vesszük sorra, melyek lényegesen eltérnek a dimenziókétól.

5.3.1 Project dimenzió

Az összes package fejlesztését egy BIDS projektbe készítettük el. A projekthez felvehetünk adatforrásokat (datasources), melyek a mi esetünkben az adatbetöltéshez szükséges forrás és cél adatbázisok. Ezeket előre beállítva nem szükséges minden egyes package-nél egyesével újra beállítani. Egy új adatforrás felvételét varázsló segíti melyben a kapcsolódási adatokat megadva gyorsan és könnyen kiválasztható a felvenni kívánt adatbázis. A projektünkben a forrás adatbázist GeminiSource a cél adatbázist pedig ***_Gemini_Target néven vettük fel. Az elnevezésekkel az volt a célunk, hogy bárki számára egyértelműen megkülönböztethető legyen a forrás és a célrendszer.

Minden package két fő részből épül fel:

- Control Flow, a betöltési logikáért felel
- Data Flow, az adatok mozgatásáért felel

Control Flow

A Control Flow tartalmazza a package-ben végrehajtandó feladatokat (task), a mi esetünkben ide kerül egy Data Flow Task nevű vezérlő, mely az adatátöltéshez szükséges eszközöket tartalmazza. A vezérlőt Load to DW-nek neveztük el, amely utal a feladatára, azaz az adattárházba tölt be adatokat.

Data Flow

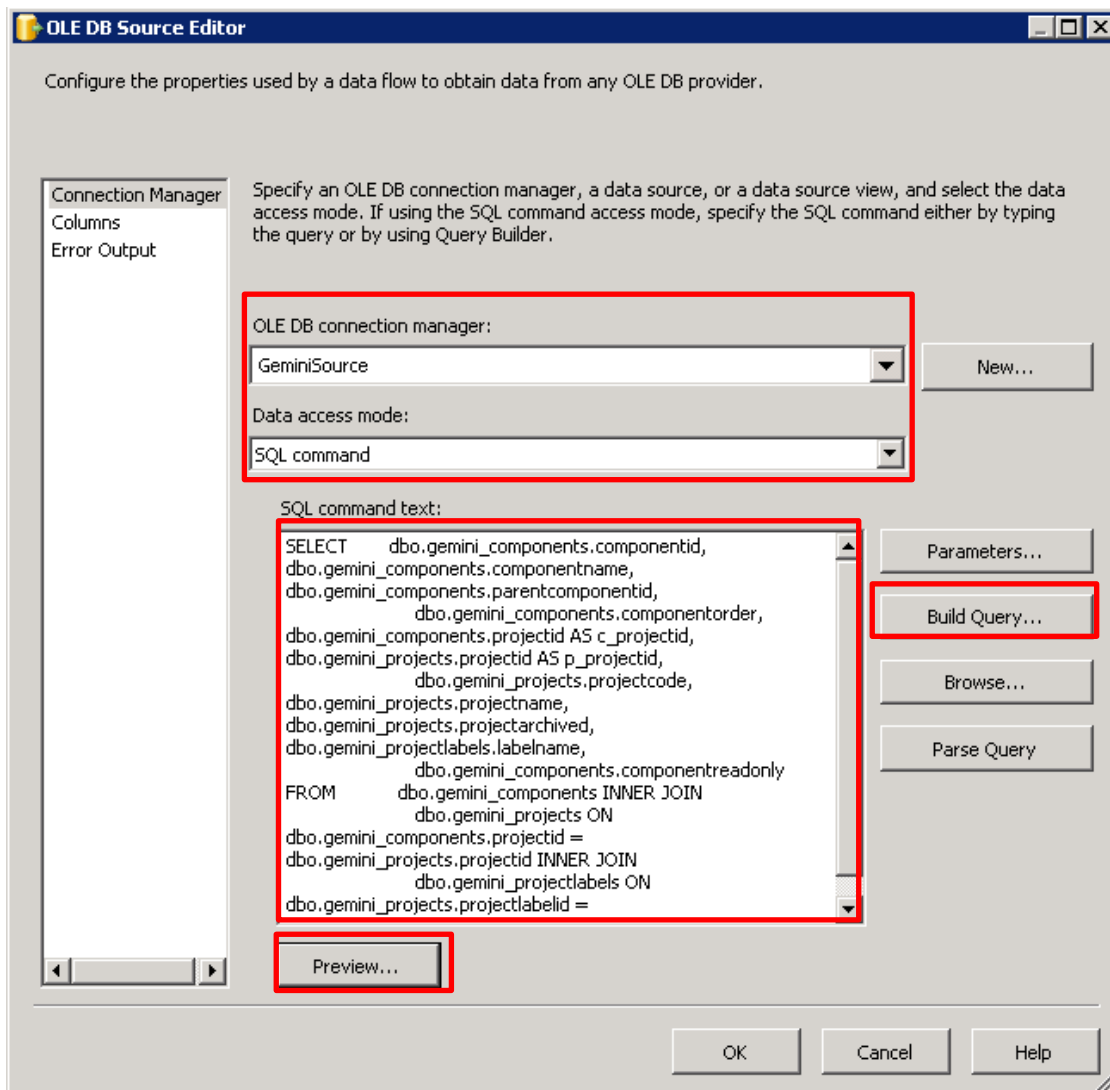
A Data Flow-ban a Control Flow felületén kiválasztott vezérlőhöz tartozó műveletek tartoznak. A Toolbox-ból különböző vezérlők érhetők el, melyek segítségével összeállíthatunk egy adatfolyamot, a mi esetünkben az ETL folyamatát.

Extract

OLE DB Source

Az adatok kinyerését OLE DB Source vezérlővel végeztük, melynek Connection Manager menüpontja alatt az adatbázist kiválasztva (OLE DB connection manager alatt megjelennek a korábban felvett adatforrásaink) választhatunk, hogy hogyan szeretnénk elérni az adatokat: adott tábla vagy nézet (Table or view), ugyanez paraméterek segítségével, SQL parancs megadásával (SQL command), illetve SQL parancs paraméterek segítségével. Az elsőt választva tállózhatunk az adatbázis táblái között, a harmadik esetben lehetőség van a Microsoft Office Access-hez hasonló úgynevezett Query Builder („Build Query...” gomb) használatára, ahol grafikusán megjelenített felületen állíthatunk össze lekérdezéseket vagy a

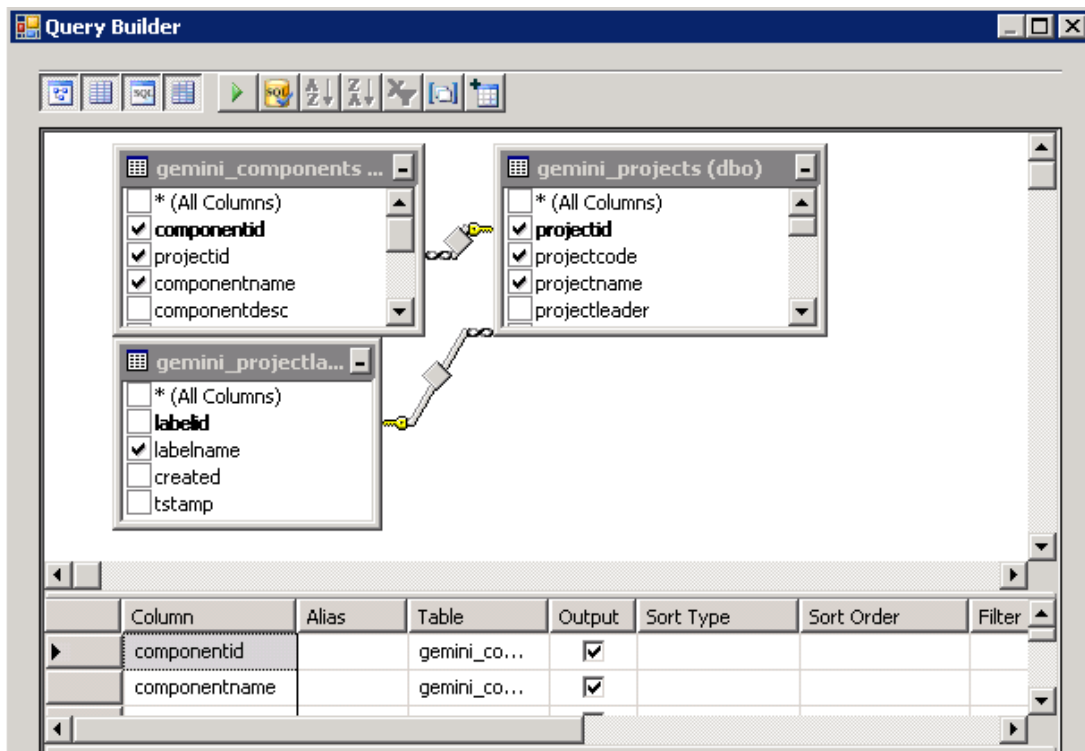
végrehajtandó SQL parancsok begépelésével kódból is összeállíthatjuk a lekérdezésünket. A második és negyedik során paraméterek segítségével állíthatók össze a lekérdezések.



7. ábra: OLE DB Connection vezérlő (saját szerkesztés, BIDS)

A package-ek fejlesztése során az első vagy a harmadik lehetőséget alkalmaztuk. Ha olyan dimenzióhoz kellett adatokat kinyernünk, amelyik csak egyetlen táblára hivatkozik akkor az elsőt, ha összetettebb lekérdezésre volt szükség, akkor a harmadik lehetőséget választottuk. Bizonyos esetekben már az adatok kinyerésekor transzformációt végeztünk velük, ilyenkor SQL kódból állítottuk össze a lekérdezést. A project dimenzió esetén három táblából kellett adatokat kinyernünk, viszont az adatokon nem végeztünk a kinyeréskor transzformációkat ezért Query Builder segítségével állítottuk össze a lekérdezéshez szükséges tábla kapcsolatokat (JOIN). Az ETL tervből egyértelműen kiderül, hogy mely táblák mely

kulcsokon keresztül kapcsolódnak össze, ezért itt már nem volt szükség a táblák kapcsolatainak feltérképezésére.



8. ábra: Query Builder grafikus felület lekérdezések készítéséhez (saját szerkesztés, BIDS)

A másik két menüpontban (Columns, Error Output) beállíthatjuk, hogy mely oszlopokra van szükségünk illetve átnevezhetjük az oszlopokat, valamint, hogy hogyan viselkedjen hibás futás esetén.

A forrás adatokon kívül, a dimenziókövetés miatt a céladatbázisból is ki kell nyernünk adatokat, hogy ezeket összevethessük a forrásrendszerből érkezőkkel. A célrendszerből is a fent már említett OLE DB Connection vezérlő segítségével nyerünk ki adatokat. Mivel egy package-ünk csak egy dimenziótáblát tölt, itt mindig csak az adott tábla rekordjait kérdeztük le.

5.3.1.1 Transform

Az előző bekezdésben már említettük, hogy a forrás rendszerből érkező adatokat összevetjük a célrendszerünkben találhatóakkal. Ahhoz, hogy ezt meg tudjuk tenni a forrás adatokat a célrendszerben található formára kell transzformálnunk. Ebben a részben az általunk az adatok transzformálására használt vezérlőket és műveleteket mutatjuk.

OLE DB Command

Bizonyos esetekben az adatok transzformációját egyszerűbb az adatok kinyerésére használt lekérdezésekbe belefoglalni, vagy legalább előkészíteni. Ilyen a felhasználók dimenziója, ahol a monogramokat a felhasználók nevéből állítottuk össze oly módon, hogy figyeltünk a magyarban használt kettős és hármas betűkre.

```
CASE
  WHEN LEFT(surname, 3) LIKE 'Dzs' THEN LEFT(surname, 3)
  WHEN (LEFT(surname, 2) LIKE 'Cs' OR
        LEFT(surname, 2) LIKE 'Dz' OR
        LEFT(surname, 2) LIKE 'Gy' OR
        LEFT(surname, 2) LIKE 'Ly' OR
        LEFT(surname, 2) LIKE 'Ny' OR
        LEFT(surname, 2) LIKE 'Sz' OR
        LEFT(surname, 2) LIKE 'Ty' OR
        LEFT(surname, 2) LIKE 'Zs') THEN LEFT(surname, 2)
  ELSE LEFT(surname, 1) END AS monogram_s
```

9. ábra: A kettős és hármas betűk szűrésére használt SQL kód (saját szerkesztés)

Derived Column

Ebben a vezérlőben az adatokkal a típus konverzióktól kezdve matematikai és szöveges műveleteken át sokféle módosítás elvégezhető. Itt végeztük az adatok különböző újra kódolását (pl.: Y(es),N(o) -> I(gen),N(em)), szöveg összefüzéseket (pl.: az előkészített monogramok összefűzése), csoportosításokat (pl.: az issue korcsoportok), valamint a forrás adatok típuskonverzióit.

Lookup

A project dimenzióknak projekt komponenseket tartalmaznak, amiknek lehet szülő komponensük. A szülő komponens mesterséges kulcsának meghatározásához egy Lookup vezérlőt használtunk, amiben megkerestük a szülő komponens üzleti kulcsához általunk hozzárendelt mesterséges kulcsot.

Dimension Merge Slowly Changing Dimension

Mint azt korábban említettük a dimenziók SCD követéséhez egy, az SSIS beépített vezérlőjét kiváltó open source vezérlőt alkalmaztunk, amit a <http://dimensionmergescd.codeplex.com/> honlapról töltöttünk le. A telepítés után a vezérlő megjelenik a többi vezérlő között a toolboxban.

A vezérlő bemenetnek két adatfolyamot vár: a forrás rendszerből érkező adatokat és az adattárházban lévő adatokat. Az adatforrások bekötése után a vezérlőben minden egyes

oszlopba be kell állítanunk annak SCD típusát: nincs használatban (Not Used), üzleti kulcs (Business Key), mesterséges kulcs (Surrogate Key), SCD1-es követésű (SCD1), SCD2-es követésű (SCD2), SCD2-es érvényesség kezdete (SCD2 Effective Date), SCD-2 érvényesség vége (SCD2 Expiry Date), SCD2-es érvényesség (SCD Current Record). Ezeken felül további típusokat is be lehet állítani, amelyeket a jelen projektünkben nem alkalmaztunk pl.: audit oszlopok.

A következő felületen a forrásrendszerből érkező sorokat a célrendszerben található sorokkal kell összepárosítanunk, hogy a vezérlő tudja, mely oszlopokat kell összevetnie. Amennyiben a két rendszerből érkező oszlopnevek megegyeznek az Auto-Map funkció megtalálja az összetartozó oszlopokat, ellenkező esetben kézzel kell a párosítást elvégezni.

A következő lapon az eltérések detektálását állíthatjuk illetve, hogy elfogadhatók-e null értékek az egyes oszlopokban. Az eltérés detektálás során beállítható, hogy a kis és nagybetűk megegyezzenek (Ignore Case) illetve, hogy a program figyelmen kívül hagyja a tartalmat megelőző illetve követő üres karaktereket (Ignore Leading/ Trailing Spaces).

Az SCD date handling fülön az SCD dátumkezeléssel kapcsolatos beállítások találhatók:

- az újonnan bekerült SCD2-es követésű adat érvényessége mikortól kezdődjön,
- az SCD követés granularitása (a legkisebb egység a milliszekundum a legnagyobb a nap, amit mi is alkalmaztunk),
- az érvényességek lezárásának menete (pl.: a mi esetünkben az újadat mától érvényes a régi tegnapig),
- az éppen érvényes rekord érvényességének vége (a mi esetünkben az MS SQL dátum függvényének a maximális értéke, ami 9999-12-31),
- az érvényességét mutató oszlopban mi szerepeljen (pl.: Y, N; 1,0).

A következő lapon a mestersége kulcs kezelését állíthatjuk be. Választhatunk, hogy a vezérlő kezelje a mesterséges kulcsokat illetve, hogy ha a vezérlő kezeli, akkor új sor felvételekor az a kulcsot mekkora léptékben növelje.

Az utolsó általunk használt menüpont a kimenetek kezelésével foglalkozó Output Column Selection. A projektünk során öt kimenetet kezeltünk le, a többi kimenettel (pl.: statisztikák és audit kimenetek) nem foglalkoztunk. Az öt kimenet az új rekordok, az új SCD2-es rekordok, a törölt rekordok, a lejárt SCD2-es rekordok SCD1-es változtatásokkal valamint a megváltozott SCD1-es rekordok. A kimenetekből kettőt-kettőt átirányítottuk egymásba, így a vezérlőnek

három kimenete keletkezik, amelyeken a mi projektünkben adatok folyhatnak. Az első ilyen kimenet az új rekordok, ahova az új SCD2-es rekordokat is irányítottuk. A második a lejárt SCD2-es rekordok SCD1-es változtatásokkal, ahova a törölt elemeket is irányítottuk. A harmadik kimenetünk a megváltozott SCD1-es rekordok.

Load

A betöltés folyamatában a transzformáció során képzett három kimenet végére bekötjük a célrendszerünket (új rekordok esetén) vagy egy SQL parancsot (megváltozott rekordok esetén).

OLE DB Destination

Abban az esetben, ha új rekordot kell a célrendszerünkben létrehoznunk OLE DB Destination vezérlőt használunk. Ezek az esetek, ha teljesen új rekord érkezik a forrásrendszerből vagy egy SCD2-es követésű rekord megváltozik. A vezérlőben az OLE DB Source-nál bemutatott módon be lehet állítani, hogy melyik táblába szeretnénk a rekordokat felvinni. A megfelelő tábla kiválasztása után a Mappings menüpont alatt beállíthatjuk, hogy az adatfolyamon érkező rekordok mely oszlopai felelnek meg a forrásrendszerben található oszlopoknak.

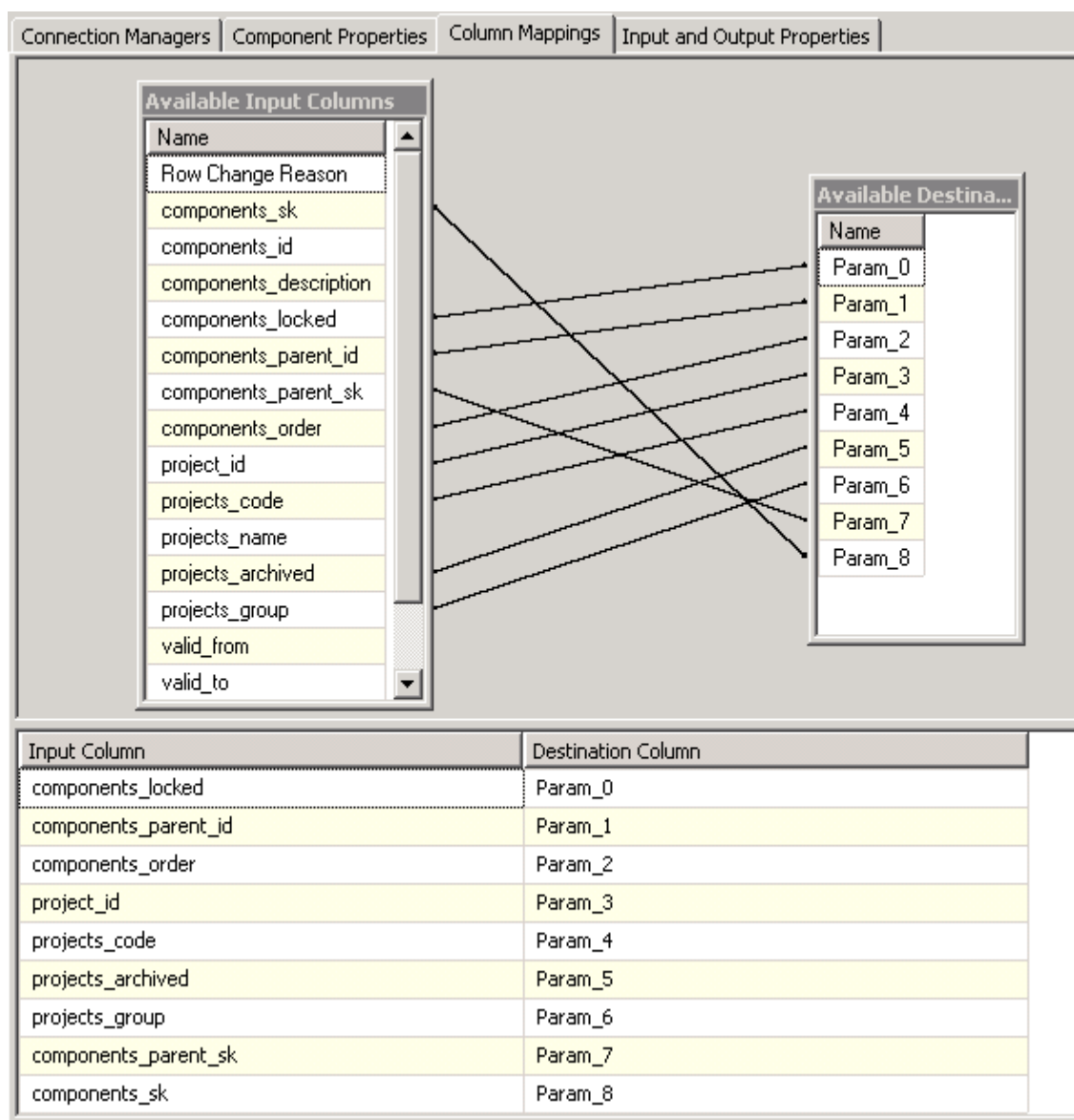
OLE DB Command

A másik két adatfolyam esetén egy UPDATE SQL utasítás segítségével vezetjük fel a rendszerbe a változtatásokat, amit az OLE DB Command vezérlő segítségével tesszük meg. Mindkét esetben paraméterekkel választjuk ki, hogy pontosan mely rekordok mely attribútumait kell megváltoztatni. A vezérlőben az adatkapcsolat beállítása után a Component Properties fül alatt az SQLcommand sorban megadható a végrehajtandó SQL parancs. Ide írtuk be a rekordok frissítését végző kódsorokat paraméterezve.

```
UPDATE d_project
SET
    components_locked= ?,
    components_parent_id= ?,
    components_order= ?,
    project_id= ?,
    projects_code= ?,
    projects_archived= ?,
    projects_group= ?,
    components_parent_sk=?
WHERE components_sk = ?
```

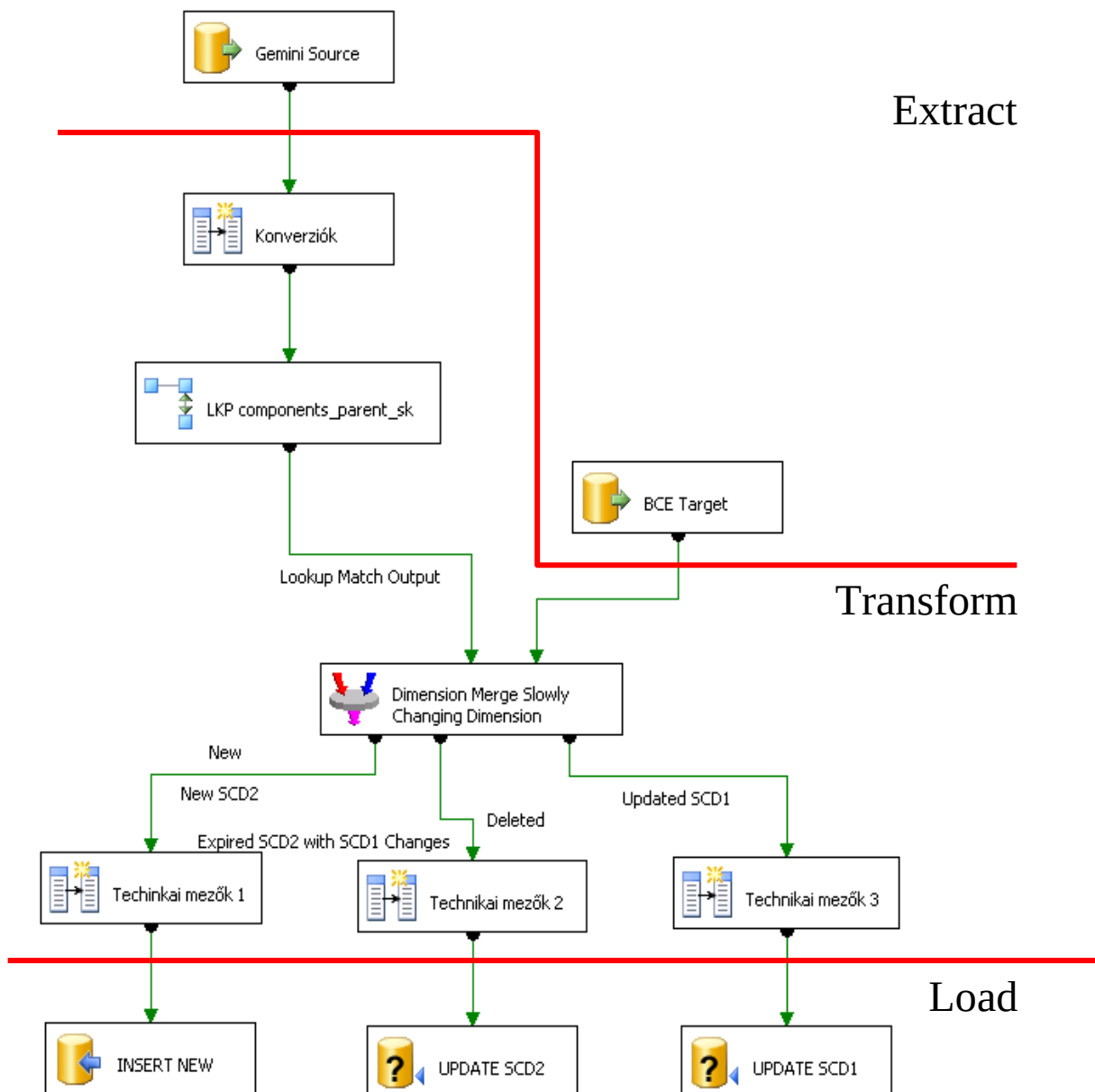
10. ábra: Az SCD1-es változtatások frissítését végző paraméteres SQL kód (saját szerkesztés)

A 10. ábrán látható, hogy a parancs a módosítandó rekordot a mesterséges kulcs alapján azonosítja be (components_sk). A kérdőjelek a paramétereket jelentik, amelyeket a vezérlő Column Mappings felületén kell összerendelnünk a frissítendő tábla oszlopaival. A paramétereket a kódban használt sorrendben nullától számozva kell értelmezni és a megfelelő oszloppal összerendelni, ebből adódóan a grafikus megjelenítés során az oszlopok közötti összerendelést mutató vonalak átláthatatlanná válhatnak. A megértést és az átláthatóságot javítja, hogy az összerendelt oszlopok táblázatos formában is megjelenjenek.



11. ábra: Az oszlopok és paraméterek összerendelése (saját szerkesztés, BIDS)

A lejárt SCD2-es rekordok SCD1-es változással, illetve törölt rekordok esetén az SQL parancsunk leegyszerűsödik, hiszen csak az SCD2-es követés miatti érvényességek lezárását kell felvezetnünk a rendszerbe. Így itt az érvényesség vége (valid_to) illetve az érvényesség (valid) oszlopokat kell az előzőekben leírt módon paraméterekkel frissíteni.



12. ábra: A d_project dimenzió Data Flow nézete (saját szerkesztés, BIDS)

5.3.2 Idő dimenzió

Az idő dimenzió töltése eltér a többiétől, mert nem hajtódik végre minden egyes adatbetöltés során. A dimenziót egy SQL utasítással előre feltöltöttük 2010. 01.01- től kezdődve

2050.12.31-ig tartó dátumokkal. A ténytáblában használt dátumok ennek a táblának az idegen kulcsaira hivatkozik. Ennek előnye, hogy az idő dimenzió táblájában előre felvehetünk olyan attribútumokat az egyes dátumokhoz, amelyeket később csak függvények meghívásával lennének képesek előállítani (pl.: az adott dátum a hét melyik napjára esik?, melyik év?, melyik hónap?, a hónap szövegesen, a hét napja szövegesen, stb.).

A script egy WHILE ciklusban végig iterál a kezdő és vég dátumok között az összes napon.

Mivel az idő dimenziónk tartalmaz szöveges dátum értékeket (pl.: hónapok nevei), amiket magyar nyelven szeretnénk megjeleníteni ezért a nyelvet is beállítottuk a scriptben. Ennek előnye, hogy később bármikor könnyen módosítható illetve más adatbázisban újra felhasználva is tudjuk garantálni, hogy magyar (vagy bármilyen más általunk preferált)

```
--Nyelv beállítás
SET LANGUAGE Hungarian

-- Változók beállítása a ciklushoz
Declare
@StartDate datetime,
@EndDate datetime,
@Date datetime

Set @StartDate = '2010/01/01'
Set @EndDate = '2050/12/31'
Set @Date = @StartDate
```

nyelven jelenjenek meg a szöveges mezők. Három változót vettünk fel a ciklushoz: kezdő

13. ábra: Az idő dimenzió készítésénél használt szkript nyelvet és a változókat beállító részlete (saját szerkesztés)

dátumot a ciklus kezdőpontjának a meghatározásához; vég dátumot a ciklus végpontjának a meghatározásához; valamint egy dátumot, amin az egyes iterációkat végezzük.

Az iterációk során az SQL dátum függvényeinek a segítségével előállítottuk minden olyan értéket, amit a dátumról tárolni szeretnénk. A függvényeket a következő kódrészlet tartalmazza (14. ábra).

```
@Date,
CONVERT(varchar(12), @Date),
CASE WHEN DATEPART(dw, @Date) < 6 THEN 1 ELSE 0 END,
Year(@Date),
DATENAME(qq, @Date),
MONTH(@Date),
DATENAME(mm, @Date),
DATEPART(wk, @Date),
DATEPART(dy, @Date),
DATEDIFF(d, DATEADD(qq, DATEDIFF(qq, 0, @DATE), 0), @DATE) + 1,
DAY(@Date),
DATEPART(dw, @Date),
DATENAME(dw, @Date),
DATEPART(wk, @Date),
datepart(wk, @Date) - datepart(wk, dateadd(m, DATEDIFF(M, 0, @Date), 0)) + 1,
DATEPART(mm, @Date)
--[Date]
--[Date_Name]
--[Weekday]
--[Year]
--[Quarter]
--[Month]
--[Month_Name]
--[Week]
--[Day_Of_Year]
--[Day_Of_Quarter]
--[Day_Of_Month]
--[Day_Of_Week]
--[Day_Of_Week_Name]
--[Week_Of_Year]
--[Week_Of_Month]
--[Month_Of_Year]
```

14. ábra: Az idő dimenzió elkészítésére használt szkript dátum függvényei (saját szerkesztés)

5.3.3 Issues ténytábla

Az issues ténytábla egy pillanatfelvételekből álló ténytábla, aminek a megvalósítása más logikával történt, mint a dimenziótábláké. A betöltési logika (Control Flow) három lépésből áll: a ténytábla megfelelő időszakának törlése (DELETE from DW), adatok betöltése (Load to DW), valamint az utólagosan tölthető adatok betöltése (DW UPDATE).

DELETE from DW

A ténytáblában szereplő pillanatfelvételek egy hetet reprezentálnak, de naponta töltjük őket, hogy a hét folyamán bármelyik napon friss adatokat szolgáltatthasson a felhasználók számára. A naponta történő töltésekhez az

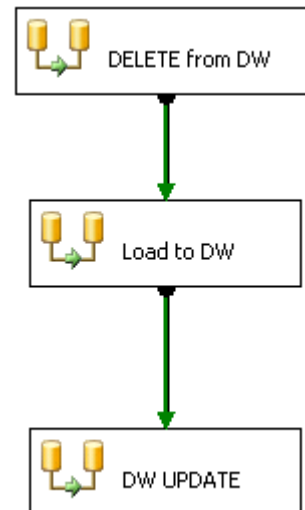
azonos időszakra (az évnek ugyanarra a hetére) vonatkozó pillanatfelvételeknek felül kell írniuk egymást. Ennek a megvalósításához meghatározzuk egy Lookup vezérlővel, hogy a betöltés dátuma az év melyik hetéhez tartozik és megkeressük az ahhoz a héthez tartozó napok mesterséges kulcsait. Ezek után töröljük azokat a rekordokat a ténytáblából, amelyek d_snapshot_date_sk oszlopa tartalmazza ezeket a rekordokat.

Load to DW

Miután az előző lépésben megtörtént az aktuális időszak tényadatainak törlése, ebben a lépésben kerülnek be az újabb tényadatok a forrásrendszerből. A betöltés első részében a hibajegyekhez (issues) tartozó üzleti kulcsok kerülnek meghatározásra, amiket az „o_” előtaggal különböztettünk meg a dimenziókban található mesterséges kulcsoktól. A mesterséges kulcsokat „d_” előtaggal láttuk el és az üzleti kulcsok alapján töltöttük be a ténytáblába. Ezek a műveletek Lookup vezérlők sorozatából épül fel (jobb átláthatóság érdekében minden Lookup vezérlőt, az általa kinyert attribútum nevével láttunk el). Miután az összes adatot kinyertük egyszerűen egy OLE DB Destination vezérlővel betöltöttük a ténytáblába.

UPDATE DW

A ténytáblánk jelenlegi státusz (current_status) attribútuma megmutatja, hogy a korábban betöltött rekordnak (issue) a legfrissebb betöltés idejében mi a státusza. Ez sérti a pillanatfelvételekből álló ténytábla logikáját, miszerint egy időszak lezárása után az adatok



15. ábra: Ténytábla betöltési logikája a Control Flow-ban (saját szerkesztés, BIDS)

már nem módosulnak. Ez egy üzleti igényként merült fel és egy UPDATE segítségével visszamenőlegesen módosítja a rekordok érintett attribútumait.

5.3.4 Kocka generálás

A korábban említett OLAP kockát az adatok frissítése után újra le kell generálni. Ennek automatizálhatóságáért és az adatbetöltéshez igazítható betöltéséért készítettünk egy package-et, ami ezért felel. A package egy Analysis Services Processing Task – ot tartalmaz, amiben a kocka szerver elérési útját beállítva a package elkészíti a kockát.

5.4 Adatbetöltés időzítése

A fejlesztés során előállított package-eket sorba kellett rendezni aszerint, hogy melyek függenek egymástól. A projektünkben két függőséget állapítottunk meg: a ténytábla betöltése csak a dimenziók feltöltése után kezdhető, valamint az OLAP kocka újra generálása csak a ténytábla betöltése után indulhat. Miután a sorrendet megállapítottuk az SQL Agent számára felvettünk egy új job-ot, ami során az általunk meghatározott időpontban (hajnali 2 óra) megkezdzi a package-ek szekvenciális végrehajtását.

6 Tesztelés

Ahogy minden elkészült IT rendszert, úgy az elkészült adattárház is tesztelni kell. Külön szoftver tesztelési módszertanokat nem mutatok be a fejezet során, helyette az adattárház és a hozzá kapcsolódó ETL tesztelésének módszere kerül bemutatásra. Egy adattárház tesztelése során az alábbi hat tesz elvégzése szükséges:

- ETL teszt
- Funkcionális teszt
- Teljesítmény teszt
- Biztonsági teszt
- UAT (User Acceptance Test)
- End-to-end teszt

(Rainardi, 2008)

6.1.1 ETL teszt

A teszt során az ETL működésének megfelelőségét kell tesztelni. Az egyik legfontosabb teszt az adattárház során, hiszen ha rossz adatok szerepelnek a kockában, lehet bármilyen jó adatmodellünk az adattárház használhatatlan, a riportok rossz adatokat fognak tükrözni. Az ETL tesztnek az alábbi pontokra kell kiterjednie:

1. Ellenőrizni kell, hogy minden szükséges adat átkerült a forrásrendszerből az adattárházba a tervezettnak megfelelően.
2. Meg kell vizsgálni, hogy az adat a megfelelő helyre és a megfelelő formátumban töltődött be a tárházba.
3. Szükséges megvizsgálni, hogy az SCD kezelés a tervezettek szerint működik.

(Rainardi, 2008)

6.1.2 Funkcionális teszt

A teszt során azt kell megvizsgálni, hogy az elkészült rendszer teljesíti-e a kért üzleti igényeket. Az üzleti igények már a projekt kezdetén megfogalmazódtak és dokumentálva lettek az Business Case -ben és a projektalapító dokumentumban egyaránt. A tesztfeladat során újra számba kell venni az igényeket és egyesével megvizsgálni, hogy a kész rendszer teljesíti-e az elvárt funkciókat.

6.1.3 Teljesítmény teszt

Ahogy a teszt neve is elárulja, a folyamat során a rendszer terhelhetőségét vizsgáljuk. Több szempontból is meg kell vizsgálni a rendszer terhelhetőségét. Először a betöltési terhelhetőség kerül vizsgálat alá úgy, hogy tömegesen akár több millió adatot is feltöltünk az adattárházba. Így megvizsgálható az ETL optimalizáltsága, mivel ha a rendszer nagyon lelassul vagy legrosszabb esetben a nagy terheléstől le is áll a folyamat, az arra ad következtetést, hogy az ETL folyamat során valahol probléma van és optimalizálásra van szükség.

A teljesítmény teszt másik aspektusa, hogy megvizsgálható a végfelhasználó szempontjából egy lekérdezés, egy riport lefutási idejét. Ha egy riport lekérdezése az adatmennyiségtől függően láthatóan túl sok időt vesz igénybe, akkor érdemes felülvizsgálni a riporting folyamatot.

6.1.4 Biztonsági teszt

Olyan rendszer esetén, amelyet több felhasználó használ egyszerre fontos, hogy a jogosultságok kiosztása megfelelő legyen. A jogosultság igényeket szintén előre definiáltatni kell az igénylővel és legtöbb esetben olyan területet kell biztosítani az ügyfél számára, ahol beállíthatja, hogy ki és mihez férhet hozzá az adattárházban. A teszt során különböző profilokkal kell bejelentkezni a rendszerbe és ellenőrizni, hogy csak olyan jogokkal rendelkezik e az adott felhasználó, mint ami a dokumentációban szerepel.

6.1.5 UAT

A UAT a felhasználói teszt, amikor az igénylő megvizsgálja az elkészült rendszert. Az UAT a fent említett tesztek után kell, hogy következzen. A teszt során néhány kulcs felhasználó

feltérképezi a rendszert és megvizsgálják, hogy kielégítik e a kért igényeket. További fontos szempont, hogy az UAT során szembesülnek a rendszer használhatóságával. Például, hogy a felhasználói felület felhasználó barát-e vagy a színezése, formája megfelelő-e a riportoknak.

6.1.6 End-to-end teszt

Az end-to-end tesztet szokták integrációs tesztnek is hívni. Amikor a rendszer megfelel minden funkcionális és nem funkcionális követelménynek, akkor következik a rendszer élesítése és integrációja. Az élesítés során szintén léphetnek fel olyan probléma, ami az eddigi tesztek során nem jöttek a felszínre. A teszt során néhány napig futtatjuk a rendszert a tervezett működésnek megfelelően és közben folyamatosan figyelemmel kísérjük. Bármilyen a normál működéstől eltérő viselkedést fel kell jegyezni, majd megvizsgálni, hogy befolyásolja-e a munkamenetet és hogy szükséges-e a javítás.

6.2 A tesztelés menete a projektünkben

A projekt nem öltött akkora méreteket, hogy a fent említett összes tesztelési szempont szerint teszteljük az adattárházunkat. Az általunk végzett tesztek az ETL ellenőrzésére szolgáltak. Az ETL teszt során megvizsgáltuk, hogy a folyamat során minden szükséges adatot sikeresen kinyertünk a forrás adatbázisból. Továbbá ellenőriztük, hogy az ETL során a transzformációk a tervezettnak megfelelően lettek végrehajtva. Végül ellenőriztük, hogy az SCD kezelés szintén a tervezettek szerint történik.

6.2.1 ETL adatkinyerési teszt

A teszt során megvizsgáltuk, hogy a kockában lévő adatok száma megegyezik-e a Geminiben lévő számokkal. A tesztet szűrőpróba szerűen végeztük, ugyanis ha minden egyes szűrési kombinációt figyelembe veszünk, akkor a tesztesetek száma túl nagy volna, hogy a teszt elvégezhető legyen.

A teszt úgy történik, hogy a kockanézetből megvizsgálunk egy bizonyos adatot. Például „aktív issue kora 30+ napos” a kockában 640 db ilyen issue –t találtunk. Majd sql lekérdezéssel megvizsgáljuk, hogy a forrásban „ha rászűrünk az aktív 30 napnál idősebb issue –kra, akkor ugyanaz az érték jön ki. Amennyiben eltérés van a két érték között, ki kell vizsgálni, hogy tényleg hibáról van szó vagy olyan körülmény, ami nem számít hibának.

Teszt eset	Kocka érték	Forrás érték
issuek száma összesen dimenziók nélkül	9252	3084
aktív issuek száma összesen dimenziók nélkül	4868	650
issuek száma ahol az aktuális státusz kezdhető	27	27
issuek száma ahol az aktuális státusz folyamatban	218	218
issuek száma ahol az aktuális státusz lezárt	154	154
aktív issuek száma ahol az aktuális státusz kezdhető	27	27
aktív issuek száma ahol az aktuális státusz folyamatban	218	218
aktív issuek száma ahol az aktuális státusz lezárt	0	0
issuek száma ahol a pillanatképi státusz kezdhető	27	27
issuek száma ahol a pillanatképi státusz folyamatban	218	218
issuek száma ahol a pillanatképi státusz lezárt	154	154
aktív issuek száma ahol a pillanatképi státusz kezdhető	27	27
aktív issuek száma ahol a pillanatképi státusz folyamatban	218	218
aktív issuek száma ahol a pillanatképi státusz lezárt	0	0
issuek száma ahol a felelős joker	142	142
issuek száma ahol a felelős pisu	333	336
aktív issuek száma ahol a felelős joker	7	7
aktív issuek száma ahol a felelős pisu	41	44
issue partnerre vár	246	246
pillanatkép dátuma 2012-12-05 összes	3084	3084
pillanatkép dátuma 2012-12-05 aktív	650	650
issuek száma ahol a prioritás kritikus	57	57
issuek száma ahol a prioritás közepes	2236	2236
aktív issuek száma ahol a prioritás kritikus	2	2
aktív issuek száma ahol a prioritás közepes	618	618
issuek száma ahol a típus fejlesztési igény	114	114
issuek száma ahol a típus garanciális hiba	219	219
aktív issuek száma ahol a típus fejlesztési igény	89	89
aktív issuek száma ahol a típus garanciális hiba	30	30
issuek száma ahol a projekt kód 411-VIR	1452	1452
issuek száma ahol a projekt kód KARTAM	273	273
aktív issuek száma ahol a projekt kód 411-VIR	181	181
aktív issuek száma ahol a projekt kód KARTAM	77	77
issuek száma ahol a komponens leírás Adatmodell	6	6
issuek száma ahol a komponens leírás Gemini	55	55
aktív issuek száma ahol a komponens leírás Adatmodell	0	0
aktív issuek száma ahol a komponens leírás Gemini	17	17
issue kora 15-29 napos	10	10
issue kora 30+ napos	3074	3074
aktív issue kora 0-14 napos	10	10
aktív issue kora 30+ napos	640	640

16. ábra: Adatkinyerési teszt (saját szerkesztés)

6.2.2 ETL adattartalom teszt

A teszt azt vizsgálja, hogy az ETL tervben leírt transzformációk a tervezettek szerint mennek végbe. A teszt két részből áll. Először teszteseteket kell kidolgozni az adott dimenzióhoz és a tesztadatokat fel kell tölteni a forrás adatbázisba. A következő lépés, hogy az ETL csomagokat le kell futtatni. A teszttervben előre definiálni kell, hogy az adott bementi adatokból milyen kimeneti adatokat várunk. Az ETL csomagok lefuttatása után az kell megvizsgálni, hogy valóban a várt kimeneteket kaptuk. Az eltéréseket rögzíteni kell és kivizsgálni az okát, ha szükséges javítani. A tesztesetek kidolgozása során érdemes a szélsőséges adatokra gondolni, például null érték, üres mező, nem várt formátum, intervallum határ stb. A tesztet az összes dimenziótáblára és a ténytáblára egyaránt el kell végezni.

gemini_issuestatus						
status_id,	statusdesc,	imagepath,	isfinal,			
használt	használt	használt	használt			
9000	Lezárt	imagepath.image1	true			
9001	Átadott	imagepath.image2	false			
9002	Kezdető	imagepath.image3	false			
9003	Új	imagepath.image4	false			
9004	Törölve	imagepath.image5	true			
9005	Folyamatban	imagepath.image6	false			
9006	NULL	NULL	false			
9007	Elhalasztva	imagepath.image7	false			
elvárt eredmény						
status_sk,	status_id,	description,	main_status,	sub_status,	final,	image,
használt	használt	használt	használt	használt	használt	használt
	9000	Lezárt	Lezárt	Lezárt	I	imagepath.image1
	9001	Átadott	Átadott	Átadott	N	imagepath.image2
	9002	Kezdető	Aktív	Nem történik semmi	N	imagepath.image3
	9003	Új	Aktív	Nem történik semmi	N	imagepath.image4
	9004	Törölve	Törölve	Törölve	I	imagepath.image5
	9005	Folyamatban	Aktív	Dolgozunk rajta	N	imagepath.image6
	9006	NULL	NULL	NULL	N	NULL
	9007	Elhalasztva	Elhalasztva	Elhalasztva	N	imagepath.image7

17. ábra: Adattartalom teszt sablon (saját szerkesztés)

6.2.3 ETL SCD teszt

Az SCD kezelés vizsgálatára szolgáló teszt. A folyamat öt lépésből áll. Először a forrás adatbázist feltöltjük tesztadatokkal. Második lépésben lefuttatjuk az ETL csomagokat. Így az adattárházat feltöltöttük az általunk generált adatokkal. A következő lépésben megváltoztatjuk a tesztadatok minden rekordjában az egyik mező értékét. Újra lefuttatjuk az ETL csomagokat. Végül megvizsgáljuk, hogy az SCD kezelés a tervnek megfelelően lezajlott-e. Az eredmény során az alábbiakat kell figyelembe venni:

- SCD 0 – A kockában lévő adat egyáltalán nem változott meg.
- SCD 1 – A kockában a forrás megváltozásának hatására az adat felülíródik.
- SCD 2 – A kockában a forrás megváltozása miatt a jelenleg érvényes érték lezárásra került és létrejött egy új sor a megváltozott értékkel, valamint egy új egyedi azonosítóval.

dbo.d_status					
status_id,	description,	main_status,	sub_status,	final,	image,
9001	BK/status_id	345ASD	12345ASD	I	123456A
9002	SCD2/description	345ASD	12345ASD	I	123456A
9003	SCD1/main_status	345ASD	12345ASD	I	123456A
9004	SCD1/sub_status	345ASD	12345ASD	I	123456A
9005	SCD1/final	345ASD	12345ASD	I	123456A
9006	SCD1/image	345ASD	12345ASD	I	123456A

18. ábra: Tesztadatok feltöltése (saját szerkesztés)

dbo.d_status						
status_id,	description,	main_status,	sub_status,	final,	image,	
9999	BK/status_id	345ASD	12345ASD	I	123456A	
9002	SCD2/description_V2	345ASD	12345ASD	I	123456A	
9003	SCD1/main_status	123	12345ASD	I	123456A	
9004	SCD1/sub_status	345ASD	345	I	123456A	
9005	SCD1/final	345ASD	12345ASD	N	123456A	
9006	SCD1/image	345ASD	12345ASD	I	567	

19. ábra: Tesztadatok megváltoztatása (saját szerkesztés)

7 Riportolás

Ebben a fejezetben a végső fázisról lesz szó. Mindamellett szeretnénk előrevetíteni, hogy a fejezet nem a saját projektünk riportjait fogja elemezni diagramról diagramra, hanem elsősorban a megoldás során használt eszközöket és beállításokat állítjuk a középpontba. Ezt egyrészt azért is tartjuk fontosabbnak a végül megalkotott diagramoknál, mert ezeknek a beállításoknak csak töredékét használtuk, ugyanakkor, mint lehetőséget, szem előtt kellett tartanunk. Másrészt az egészen sokrétű fejlesztői környezet lehetőségeit mindenképpen érdemesnek tartottuk leírni.

Tehát a projekt eredménye az ügyfél által igényelt automatikus, periodikus riport meghatározott paraméterezéssel. Ezeket a riportokat szintén BIDS-ben alakítottuk ki, SQL Server Reporting Services segítségével.

A BIDS-ben a riportok szerkesztését egy egyszerű felületen keresztül tudjuk megvalósítani. A cél azoknak a diagramoknak a megvalósítása volt, melyeket a megrendelőtől a projekt elején excel formátumban megkaptunk. Ezeket a diagramokat egyszerűen fel tudjuk pakolni a riportfelületre, melyet a szokásos microsoftos attribútumok segítségével tudunk formázni. A mi esetünkben ez hat diagramot jelentett. A fejlesztőeszközben rendelkezésre állt egy „előnézet” funkció, mellyel folyamatosan nyomon követhető volt a kimutatások állapota.

Az első lépés a forrás megadása, vagyis az adatbázis importálása, melyből majd a riport táplálkozni fog. Ez a mi esetünkben a kocka volt. Ezután a kockából ún. „dataset”-ket kellett létrehozunk, melyet a beágyazott diagramok fognak használni. A datasetekben a kockában szereplő mezőket, valamint a mérőszámokat azonosítottuk. Ezenkívül létrejöhetnek még rejtett adatállományok is, melyek a paraméterek miatt lehetnek fontosak.

Ezen a ponton tennénk egy kisebb kitérőt. A datasetek tartalmát (ahogy egyébként minden mást is) MDX¹ kódban is tudjuk módosítani. A Microsoft fejlesztőeszközének egyik előnye, hogy olyan feladatokhoz, lekérdezések írásához, amelyhez egyébként MDX fejlesztői tudásra lenne szükség, grafikus felületen is meg tudjuk oldani. Mivel egy lekérdező nyelvről van szó, ezt egészen hatékonyan meg lehet tenni így. Működésük hasonlít a MS Access-ben tapasztaltakhoz.

Miután minden mező rendelkezésre állt a kockából, ezeket csak be kell húznunk a diagramokba. Ez a megoldás teljesen úgy működik, mint az Office programcsomag grafikon, vagy Pivot-tábla varázslójában. A mi projektünkben a kategóriatengelyen mindig a pillanatfelvétel dátuma szerepelt, valamint az egyes diagramokon az egyes elemzési szempontok megfelelő változója. Az értékek közé pedig a két meghatározott mérőszám közül az éppen relevánszt választottuk (legtöbbször az „aktív issuek száma”-t).

A riport egy másik fontos eleme a szűrők (paraméterek) létrehozása. A BIDS-ben erre is van egy megfelelő funkció, mellyel az egyes datasetekben megjelenő mezők szerint paramétereket tudunk létrehozni, így kiválaszthatjuk, hogy a teljes adatállományból melyek azok a csoportok, melyeket meg szeretnénk jeleníteni a diagramokon. A paraméterek adatait a program szintén datasetekben kezeli. A paraméterezés beállítást azonban nemcsak itt, hanem a SQL Server Reporting Services (SSRS) egyik webes kezelőfelületén (Report Manager) is beállíthatjuk.

Mielőtt rátérnénk az SSRS rövid bemutatására, tennénk egy fontos megjegyzést. Mivel a riport az egész megoldás kifutása, fontos, hogy a megrendelő azt lássa, amit előzőleg megálmodott. A riportok elkészítésénél különösen fontos a folyamatos egyeztetés, kommunikáció az ügyféllel. Soha ne akarjuk kitalálni, hogy az ügyfél mit szeretne, vagy mi lenne a leglogikusabb, mivel az a megoldás szinte biztosan nem fog tetszeni neki. Ez persze nem a megrendelő, illetve a fejlesztő tudatlansága vagy alulképzettsége miatt van így. Egész egyszerűen sokkal hatékonyabb, ha mindig a vevő mondja meg, milyen legyen a termék. A fejlesztő csak javaslatokat tegyen legjobb tudása szerint, így elkerülhető a felesleges többletmunka.

Az SSRS konfigurációs beállításait nem mutatnánk be részletesen, hiszen nem dokumentációt szeretnénk írni. Ennek ellenére azért felsorolásszerűen érdemesnek tartjuk bemutatni, hogy

¹ MultiDimensinal eXpression. Az OLAP lekérdező nyelve, mely nagyon hasonlít az SQL-re struktúrájában.

milyen módosításokat, beállításokat végezhetünk el itt. Ezen a felületen tudjuk az SQL Server riportgeneráló alkalmazásának megmondani, hogy kik azok a felhasználók, akik futtathatják a szolgáltatást. Beállíthatjuk, hogy a webszolgáltatássá varázsolt riportok hova kerüljenek, valamint a fentebb már említett „Report Manager” nevű, rendkívül hasznos kezelőfelület URL-jét is megadhatjuk. Ezenkívül még például adatbázis beállításokat, titkosítási tulajdonságokat, SMTP címet és az elektronikus üzeneteink formális feladóját is itt tudjuk kezelni.

A következőkben a Report Manager (RM) hasznos elemiről lesz szó. Ezen a webes felületen tudjuk az elkészített, telepített (deployed) riport beállításait megadni. Itt jelennek meg (az SSRS-ben megadott mappában lévő) kész riportok, melyeket további speciális tulajdonságok mentén tudunk menedzselni. Számos olyan beállítás elvégezhető ezen a felületen is, melyet már a BIDS-ben is megadhattunk.

A RM egyik legfontosabb funkciója az előfizetések (subscriptions) kezelése. Itt adhatjuk meg, hogy a kész riportot milyen formában (pl: e-mail, cloud), milyen formátumban, kiknek, hogyan és mikor küldjük el. Akár fejléct is csatolhatunk a riport tetejére, vagy megfelelő paraméterezéssel generált üzeneteket csatolhatunk személyre szabottan. Az előfizetések beállításai között minden előfizető (megrendelő) számára testreszabott formában tehetjük elérhetővé a riportot egyéni szűrési (paraméterezési) beállítások mellett.

A paraméterek beállításait az RM felületén keresztül is tudjuk szerkeszteni, persze az MDX kódba már nem tudunk belenyúlni, de alapszintű tulajdonságokat be tudunk állítani (pl: paraméter felvétele, törlése, alapérték). További lehetőség még a riport alapadatainak (név, cím) és adatforrásának módosítási lehetősége, pillanatfelvétel és cache beállítások, illetve a biztonsági szabályok beállítása.

A 3. mellékleten láthatóak az elkészült riportok, valamint a riportokhoz tartozó egyéb ábrák és képernyőképek.

8 A projekt eredménye

Az általunk elkészített üzleti intelligencia megoldás sikeresnek mondható. Az adattárház és az ETL jelenleg is üzemben van és a tervezettnél megfelelően működik. A kockából készült riportok napi jelleggel futnak és grafikus módon mutatják az issuek állapotát. Az üzleti

igények nagyrészt teljesültek bár a riport szűrési feltételein és kinézetén lehetne tökéletesíteni, ha a későbbiekben üzleti döntésekhez szeretnénk felhasználni őket.

A rendszer nem került be az éles környezetbe és magából a Geminiből kapja a forrásadatokat, hanem egy korábban elkészített pillanatfelvétel az adattárház forrása. Az élesítéshez még tökéletesíteni kellene az ETL folyamatokat és elvégezni egy teljesítmény tesztet is, hogy látható legyen a rendszer terhelhetősége nagy adattömeg esetén.

A megoldás tökéletesítésével más BI funkciókra is lehetne használni a rendszert. Jelenleg a végső terméke a megoldásnak az elkészült riportok, amelyek egyfajta mutatóként szolgálnak. Amennyiben megfelelően sok adat kerülne az idők során az adattárházba, úgy akár adatbányászati célokra is lehetne használni. Ilyen alapon ki lehetne mutatni az egyes típusú projektek során, mikor merülnek fel nagyobb számban az issuek, mennyi issue-ra lehet számolni átlagosan, milyen típusú issuek a projekt mely szakaszában jelentkeznek. Így a szervezet előre készülhetne adott projekt során a várható terhelésre és megelőző lépéseket tehet például erőforrás felszabadítással vagy plusz erőforrás bevonásával.

9 Irodalomjegyzék

Kimball, Ralph – Caserta, Joe (2004): The data warehouse ETL toolkit, Wiley Publishing, Inc., Kanada

Moss, Larissa T. - Atre, Shaku (2003): Business Intelligence Roadmap: The Complete Project Lifecycle for Decision- Support Applications. Crawfordsville: Addison Wesley.

Rainardi, V. (2008): Building a Data Warehouse With Examples in SQL Server. USA: Apress.

Internetes források

Sidló, Csaba (2004): Összefoglaló az adattárházak témaköréről.

Forrás: <http://scs.web.elte.hu/Work/DW/adattarhazak.htm>

Letöltve: 2013.03.10.

Giblett, Peter B. (2009): The Advantage of Good Business Intelligence.

Forrás: <http://cio-perspectives.com/2009/02/the-advantage-of-good-business-intelligence/>

Letöltve: 2013.03.11.

Rainardi, V. (2011) : Slowly Changing Dimension (SCD) Type 0

Forrás: <http://dwbi1.wordpress.com/2011/07/12/slowly-changing-dimension-scd-type-0/>

Letöltve: 2013.03.09.

MSDN (é.n.): SQL Server Integration Services

Forrás: <http://msdn.microsoft.com/hu-hu/library/ms141026.aspx>

Letöltve: 2013.03.09.

Inmon H. W. (2007): Cost Justification In The Data Warehouse

Forrás: <http://www.inmoncif.com/view/31>

Letöltve: 2013.03.22

10 Ábrajegyzék

1. ÁBRA: ADATKOCKA (SIDLÓ, 2004).....	15
2. ÁBRA: CSILLAGSÉMA (SAJÁT SZERKESZTÉS).....	15
3. ÁBRA: DIMENZIÓLISTA	18
4. ÁBRA: ETL TERV SABLON (SAJÁT SZERKESZTÉS).....	24
5. ÁBRA: ETL TERVEZÉS ELSŐ LÉPÉSEI (SAJÁT SZERKESZTÉS).....	25
6. ÁBRA: ELKÉSZÜLT ETL TERV A STATUS DIMENZIÓRA (SAJÁT SZERKESZTÉS)	27
7. ÁBRA: OLE DB CONNECTION VEZÉRLŐ (SAJÁT SZERKESZTÉS, BIDS)	30
8. ÁBRA: QUERRY BUILDER GRAFIKUS FELÜLET LEKÉRDEZÉSEK KÉSZÍTÉSÉHEZ (SAJÁT SZERKESZTÉS, BIDS)	31
9. ÁBRA: A KETTŐS ÉS HÁRMAS BETŰK SZŰRÉSÉRE HASZNÁLT SQL KÓD (SAJÁT SZERKESZTÉS)	32
10. ÁBRA: AZ SCD1-ES VÁLTOZTATÁSOK FRISSÍTÉSÉT VÉGZŐ PARAMÉTERES SQL KÓD (SAJÁT SZERKESZTÉS)	34
11. ÁBRA: AZ OSZLOPOK ÉS PARAMÉTEREK ÖSSZERENDELÉSE (SAJÁT SZERKESZTÉS, BIDS)	35
12. ÁBRA: A D_PROJECT DIMENZIÓ DATA FLOW NÉZETE (SAJÁT SZERKESZTÉS, BIDS).....	36
13. ÁBRA: AZ IDŐ DIMENZIÓ KÉSZÍTÉSÉNél HASZNÁLT SZKRIPT NYELVET ÉS A VÁLTOZÓKAT BEÁLLÍTÓ RÉSZLETE (SAJÁT SZERKESZTÉS).....	37
14. ÁBRA: AZ IDŐ DIMENZIÓ ELKÉSZÍTÉSÉRE HASZNÁLT SZKRIPT DÁTUM FÜGGVÉNYEI (SAJÁT SZERKESZTÉS)	37
15. ÁBRA: TÉNYTÁBLA BETÖLTÉSI LOGIKÁJA A CONTROL FLOW-BAN (SAJÁT SZERKESZTÉS, BIDS)	38
16. ÁBRA: ADATKINYERÉSI TESZT (SAJÁT SZERKESZTÉS)	42
17. ÁBRA: ADATTARTALOM TESZT SABLON (SAJÁT SZERKESZTÉS).....	43
18. ÁBRA: TESZTADATOK FELTÖLTÉSE (SAJÁT SZERKESZTÉS).....	43
19. ÁBRA: TESZTADATOK MEGVÁLTOZTATÁSA (SAJÁT SZERKESZTÉS)	44

11 Mellékletek

1. Melléklet – OLTP vs. OLAP

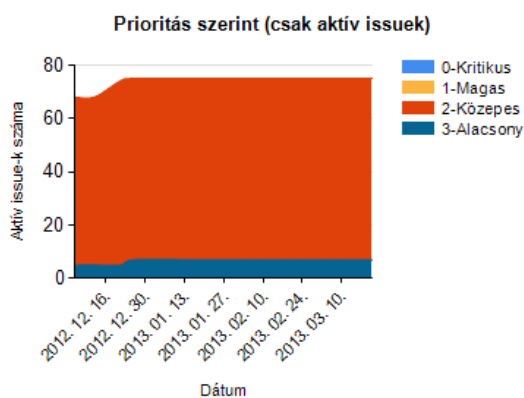
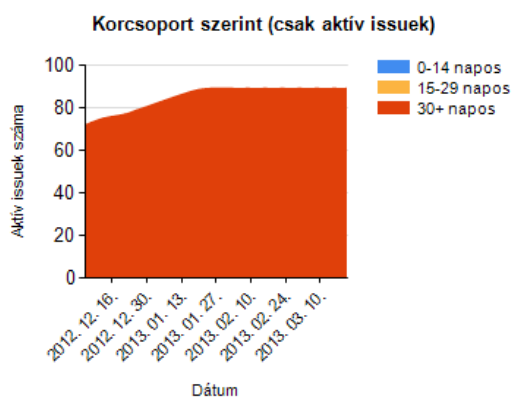
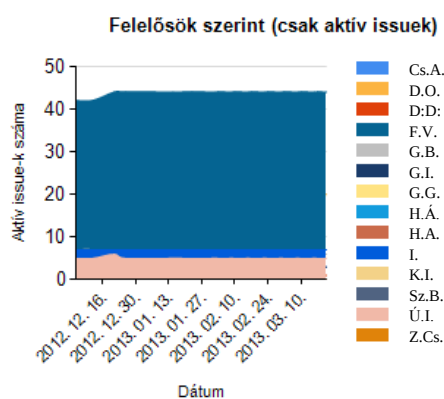
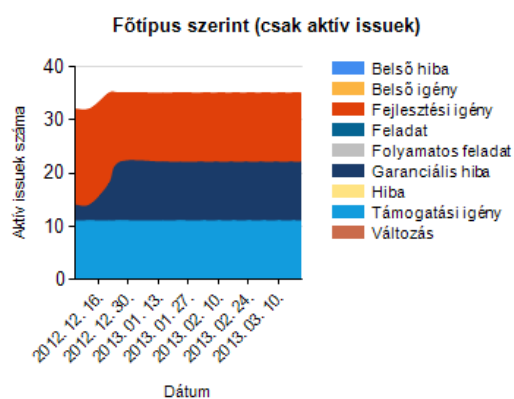
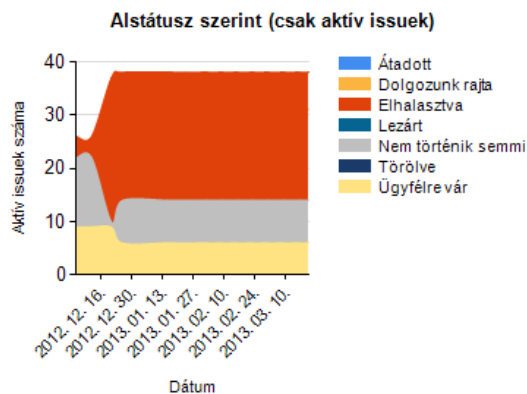
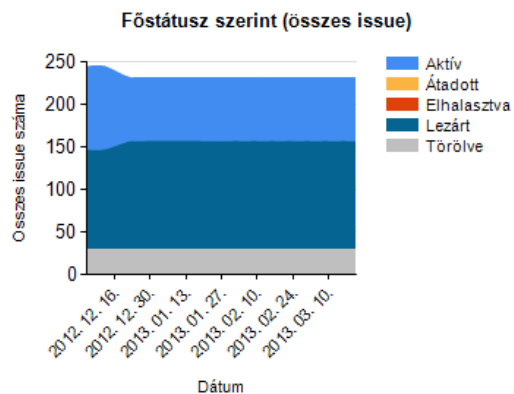
	On-Line Transaction Processing	Data Warehouse
Purpose	To run day-to-day operational activities of an organisation. E.g. to enter or modify purchase orders for a customer.	A copy of transaction data used for management and statutory reporting, information retrieval and analysis.
Structure	May be held on relational and non-relational databases.	A corporate relational data repository used for reporting. Normally based on a relational database.
Data Model	Normally defined as entity relationship models. Mainly normalised, although elements will be de-normalised when performance demands.	Defined as a star-join schema, a denormalised structure to support separate definition of dimensions and factual data.
Access	Via OLTP applications having SQL or middleware or proprietary data interface.	Via analytical applications that enable access to the Data Warehouse or subject-area Data Mart.
Type of data	Data that runs day-to-day activities within the business	Information used to analyse the business and make informed decisions.
Condition of data	Relates to current activities only, subject to change. Old data to be archived and removed.	Non-volatile, historical, and time variant. Historical data may be summarised.

1. melléklet: OLTP vs. OLAP (Giblett, 2009)

2. Melléklet – Dimenzióterv

[illegible]

3. Melléklet – Riport



4. Melléklet - Projekt dimenzió ETL terve

Verzió: 1.0 (2012.10.17)	Felelős: Gábor	Táblák: gemin_i_components, gemin_i_projects, gemin_i_projectlabels	GRANULARITÁS: gemin_i_components.c omponentid	Szűrés:					
CélMező	Cél mező típusa	Mező leírás	Forrás (Tábla)	Forrás (mező)	Művelet negjegyzé	ki	mikor	mit	
components_sk	EGÉSZ SZÁM	surrogate key			SK				
components_id	EGÉSZ SZÁM	gemin_i komponens id-je	gemin_i_components	componentid	BK				
components_description	SZÖVEG(255)	komponens megnevezés	gemin_i_components	componentname	SCD2				
components_locked	SZÖVEG(1)	Zárolt-e a komponens	gemin_i_components	CASE componentreadonly = "Y" THEN "I" CASE componentreadonly = "N" THEN "N" ELSE NULL	SCD1				
components_parent_id	EGÉSZ SZÁM	gemin_i parentcomponentid	gemin_i_components	parentcomponentid	SCD1				
components_parent_sk	EGÉSZ SZÁM	szülőkomponens SK azonosítója		LKP d_project.SK WHERE components.componentid = #components_parent_id	SCD1				
components_order	EGÉSZ SZÁM	komponens rangsorolás	gemin_i_components	componentorder	SCD1				
project_id	EGÉSZ SZÁM	projekt geminibeli ID-ja	gemin_i_components	projectid	SCD1				
projects_code	SZÖVEG(30)	a projekt kódszáma	gemin_i_projects	LKP gemini_projects.projectcode (#project_id = gemin_i_projects.projectid)	SCD1				
projects_name	SZÖVEG(255)	a projekt megnevezése	gemin_i_projects	LKP gemini_projects.projectname (#project_id = gemin_i_projects.projectid)	SCD2				
projects_archived	SZÖVEG(1)	archivált-e?	gemin_i_projects	CASE LKP gemini_projects.projectarchived (#project_id = gemini_projects.projectid) = "Y" THEN "I" CASE LKP gemini_projects.projectarchived (#project_id = gemini_projects.projectid) = "N" THEN "N" ELSE NULL	SCD1				
projects_group	SZÖVEG(255)	projekt csoport	gemin_i_projectlabels	LKP gemini_projectlabels.labelname (gemin_i_projectlabels.labelid = gemin_i_projects.projectlabelid AND #project_id = gemin_i_projects.projectid)	SCD1				
valid_from	DÁTUM	Érvényesség kezdete							
valid_to	DÁTUM	Érvényesség vége							
valid	SZÖVEG(1)	Aktív-e a rekord							

5. Melléklet - Ténytábla ETL terve

Verzió: 1.0 (2012.10.17)		Felelős: Gábor	Táblák: gemini_issues	GRANULARITÁS: gemini_issues.issueid x snapshot_date (heti)	Szűrés:				
CélMező	Cél mező típusa	Mező leírás	Forrás (Tábla)	Forrás (mező)	Művelet/megjegyzés	ki	mikor	mit	
issues_sk	EGÉSZ SZÁM	surrogate key			SK				
d_project_sk	EGÉSZ SZÁM	Projekt		LKP d_project.component_sk (d_project.components_id = #o_project_id)					
o_project_id				LKP gemini_issuecomponents.compone ntid (gemini_issuecomponents.issueid = #o_issue_id)					
d_type_sk	EGÉSZ SZÁM	Típus		LKP d_type.type_sk (d_type.type_id = #o_type_id)					
o_type_id			gemini_issues	issuetypeid					
d_snapshot_date_sk	EGÉSZ SZÁM	Pillanatkép dátuma		LKP d_time.time_sk (d_time.time_id = #o_snapshot_date)	BK				
o_snapshot_date	DÁTUM			betöltés által generált					
d_age_sk	EGÉSZ SZÁM	Issue kora		LKP d_age.issue_age_sk (d_age.issue_age = #o_age_id)					
o_age_id				DATEDIFF (currentdate,gemini_issues.created)					
d_snapshot_status_sk	EGÉSZ SZÁM	Státusz		LKP d_status.status_sk (d_status.o_status_id = #o_snapshot_status_id)					
o_snapshot_status_id			gemini_issues	issuestatusid					
d_current_status_sk	EGÉSZ SZÁM	Jelenlegi Státusz		LKP d_status.status_sk (d_status.o_status_id = #o_current_status_id)		minden betöltéske r frissül			
o_current_status_id			gemini_issues	issuestatusid		minden betöltéske r frissül			
d_assignee_user_sk	EGÉSZ SZÁM	Felelős	d_user	LKP d_user_sk (d_user.user_id = #o_assignee_user_id)					
o_assignee_user_id	EGÉSZ SZÁM	felelős eredeti azonosítója		LKP gemini_issueresources.userid (gemini_issueresources.issueid = #o_issue_id)					
d_issue_sk	EGÉSZ SZÁM	Issue		LKP d_issue.issue_sk (d_issue.o_issue_id = #o_issue_id)					
o_issue_id	EGÉSZ SZÁM	issue eredeti kulcsa	gemini_issues	issueid	BK				
priority	SZÖVEG(30)	prioritás megnevezése	gemini_issuepriorities	LKP gemini_issuepriorities.prioritydesc (gemini_issuepriorities.priorityid = gemini_issues.issuepriorityid)					
partner_issue	SZÖVEG(1)	partneri feladat		case LKP gemini_customfielddata.fielddata (gemini_customfielddata.customfie ldid = 25 AND gemini_issues.issueid = #o_issue_id) IS NULL THEN "N" ELSE "I"					
wait_for_partner	SZÖVEG(1)	partnerre vár		CASE #.partner_issue = "I" AND LKP d_status.description (#o_current_status_id = d_status.status_id) IN ("folyamatban", "belső akadály", "külső akadály", "elhalasztva") THEN "I" ELSE "N"					
issues	EGÉSZ SZÁM	Összes issue száma		ennek a mezőnek az értéke minden esetben 1					
active_issues	EGÉSZ SZÁM	Aktív issue száma		CASE LKP d_status.final (#o_current_status_id = d_status.status_id) = "I" THEN 1 ELSE 0					

6. Melléklet – A kocka csillagsémája

