DIPLOMAMUNKA

Szatmári András

Debrecen

2010

Debreceni Egyetem

Informatikai Kar

SQL SERVER ADATTÁRHÁZAK

Témavezető:
Kollár Lajos
Egyetemi tanársegéd

Készítette:
Szatmári András (Sotmari Andrash)
Programtervező matematikus

Debrecen
2010

Tartalom

1 Bevezetés

A mai felgyorsult üzleti életben nagyon fontos az, hogy a vállalkozások, azok vezetői megfelelő időben hozzanak megfelelő döntéseket, alakítsanak ki üzleti stratégiákat, különben piaci versenytársaikkal szemben hátrányba kerülnek. Természetesen ez csak akkor lehetséges, ha a döntés meghozatalához szükséges minden információ rendelkezésre áll. Ezen információszükséglet kielégítése azonban gyakran nem lehetséges a már megszokott és egy-egy területen jól bevált eszközeinkkel és rendszereinkkel.

Erre az információszükségleti problémára adnak megoldást az üzleti intelligencia rendszerek és a ma már szerves részükké vált multidimenzionális adatbázis-kezelők és adattárházak (2-4 fejezet).

Munkám célja elsősorban egy adattárház építési folyamatának és az ehhez felhasználható eszközök fejlesztői szemmel történő bemutatása. Gyakorlati példákon keresztül fogom bemutatni, hogy mik azok a tervezési és kivitelezési szempontok, amelyeket mindenképpen figyelembe kell vennünk egy adattárház építése folyamán, melyek azok a hibák, amelyeket könnyen elkövethetünk, és melyek azok az eszközök, amelyekkel a fejlesztést megkönnyíthetjük (5-7 fejezet).

Munkám folyamán elsősorban a Microsoft SQL Server 2008 programcsomagot fogom használni, egyrészt mivel az SQL Server használatában már több éves tapasztalattal rendelkezem, másrészt mivel az egyetemi könyvtár katalógusát áttekintve meglehetősen kevés SQL Serverrel foglalkozó munkát találtam. Be fogom mutatni az SQL Server Integration Servicest, mint általános célú adatbetöltő eszközt, amely a Microsoft állítása szerint a világon jelenleg a leggyorsabb.

2 Az adattárház koncepció

2.1 Az adattárház fogalma

Az adattárházak olyan speciális adatbázisok, melyeket a hagyományos tranzakciós rendszerektől eltérően bonyolult elemzési feladatok és összetett lekérdezések végrehajtására terveztek. Az adattárházban legfőképpen történeti adatokat tárolunk, melyeket több más rendszerből nyertünk ki.

Lássuk, hogyan definiálta az adattárházat az adattárház elmélet két úttörője: Ralph Kimball és Bill Inmon[1]

2.2 Adattárház definíciók

2.2.1 Kimball definíciója

Ralph Kimball definíciója: ***Data Warehouse****: "The conglomeration of an organization's data warehouse staging and presentation areas, where operational data is specifically structured for query and analysis performance and ease-of-use."[2]* Az adattárház fogalma itt tehát egy adott szervezet azon adatgyűjtő és szolgáltató részeit foglalja magában, ahol a működési adatokat újrastrukturálják riportkészítési, jó teljesítményű és egyszerűen kezelhető elemzésekhez. Kimball ezen definícióját főleg azért szokták kedvelni és idézni, mert sok mindent nem határoz meg, pl. az adattárház nem feltétlenül döntéstámogatási célú [13].

2.2.2 Inmon definíciója

"A data warehouse is a subject oriented, integrated, nonvolatile, and time variant collection of data in support of management's decisions."[1]

„Az adattárház az adatok egy tárgyorientált, integrált, tartós és időfüggő, a vezetői döntések támogatására létrehozott kollekciója."

Nézzük végig az említett jellemzőket!

[1] Eredetileg William Harvey Inmon

2.2.2.1 Tárgyorientált

Tárgyorientált (subject oriented), tematikus, esetleg témaorientáltnak is szokás fordítani. Hagyományosan az alkalmazásainkat annak funkcióit, feladatait szem előtt tartva tervezzük, azok köré építjük. Az adattárház tárgy-orientáltságát, tematikus felépítését ehhez képest olyan értelemben szokás használni, miszerint most adott tárgyterületek köré, a meglévő és kapcsolódó adatokat szem előtt tartva ("data driven") tervezünk.

Példaként nézzünk egy vállalatot, amely kábeltévés szolgáltatást nyújt. Hagyományos rendszerei megvalósítanak sok feladatot, a számlázás folyamatát, a beszerzést, a karbantartást és így tovább. Minden ilyen alkalmazás támogatott valamilyen saját adathalmazzal. Az adattárház építésénél azonban szeretnénk a meglévő adatokat a vevő szerint csoportosítva, összegyűjtve kezelni, vagy más hasonló tárgyterület köré csoportosítva látni (mint például az adás-kimaradások). Az adattárházban minden adatunkat ezek köré a tárgyterületek köré csoportosítjuk, gyűjtjük. Megjegyzem gyakran ezek a forrásrendszerek funkcióinak központi szereplői, mint pl. számlázási rendszerből a számlák, a raktárkészlet-nyilvántartásból a termékek, stb. [13].

2.2.2.2 Integrált

Az előző pontban említett tárgyorientált, adatvezérelt tervezéshez szorosan kapcsolódik az integráltság fogalma a következő értelemben: az adattárház az említett tárgyterületekhez kapcsolódó adatokat az érintett adatforrásokból szabványosított formára alakítva egy helyre gyűjti és egységbe rendezve kezeli [13]. Vagyis az adattárháznak fel kell oldania a különböző rendszerekből érkező adatok között fennálló konfliktusokat, egységes elnevezéseket és egységes szerkezeteket használni az adatok tárolására függetlenül attól, hogy honnan érkeznek.

2.2.2.3 Tartós

Tartós (nonvolatile - nem illékony) jelenti azt, hogy az adattárházban jelen lévő adatok alapvetően változatlanok. Ha a forrásrendszer adatai változnának, az adattárház a változást követi, de úgy, hogy a bent lévő adatot megfelelő időbélyeggel (érvényességi idővel) látja el, majd felveszi az új állapotot is, megfelelő időbélyeggel. A bekerült adatok tehát tartósan meg is maradnak [13].

2.2.2.4 Időfüggő

Időfüggő, vagy időben változó (time variant): Adataink általában adott időpillanatra, a jelen állapotra vonatkoznak. Annak érdekében, hogy az elemzők megfelelő információkhoz jussanak történeti adatok tárolására van szükség. Az adattárház ennek megfelelően az adatokat időfüggően, az adatokat időpontok és időintervallumok szerint tárolják és kezelik, a forrásrendszerek változását nyomon követve [13].

2.3 A "Data Warehousing" fogalma

Data Warehousing alatt értjük adott szervezet adatainak adattárház eszközzel való kezelésének folyamatát, az adatok keletkezésének helyétől indulva egészen az elemzési célú megjelenítésig [13].

"Data Warehousing is the process, whereby organizations extract value from their informational assets through the use of special stores called data warehouses."[3] Ennek megfelelően az adattárház működése három fontos kulcsmozzanat köré szerveződik, ezek:

- Adatkinyerés a tranzakciós (vagy más vállalat-működtetési) rendszerekből
- A kinyert adatok átformálása riport (beszámoló) készítés számára
- A riportok, beszámolók elérhetővé tétele a döntéshozók számára.

3 Adattárház technológián alapuló rendszerek

Mielőtt belekezdenénk az adattárházak szerkezeti és tervezési részleteinek tárgyalásába, nézzünk egy pár példát arra, hogy pontosan mire is használjuk az adattárházakat, melyek azok a rendszerek, melyeknek szerves részét képezik. Talán ez a megszokottól eltérő megközelítés azonban így jobban megérthetjük az adattárházak létjogosultságát és szükségességét.[2]

3.1 Az üzleti intelligencia

Az üzleti intelligencia (Business Intelligence, BI) első meghatározását sokáig Howard Dresnernek tulajdonították, aki 1989-ben úgy definiálta az üzleti intelligenciát, mint olyan módszerek, fogalmak összessége, melyek a döntéshozás folyamatát javítják az úgynevezett tényalapú rendszerek segítségével.

Később aztán (2009-ben) valószínűleg Claudia Imhoff blogbejegyzése [9] nyomán visszadatálták az üzleti intelligencia első definícióját egy bizonyos H. P. Luhn nevű úrra, aki az IBM Journal 1958 októberi számában írt egy cikket „A Business Intelligence System" címmel. Luhn 1958-ban egy olyan világról vizionált, ahol egyre több információt fognak a különböző szervezetek és egyének létrehozni, és ezeket automatikus, intelligens rendszerek fogják kinyerni, feldolgozni, tárolni és szétosztani, hogy támogassák az emberek, szervezetek tevékenységét. Bár neki még mást jelentett az információ (mikrofilm, papír alapú dokumentumok) mint amit ma nekünk jelent, de az üzleti intelligencia célját, az adattárházak működését nagyon jól előrevetítette.

Ennek ellenére az üzleti intelligencia csak a 90-es évek végén épült be az informatikai szállítók és a szervezetetek szókincsbe. Elterjedése előtt hazánkban főleg döntéstámogató rendszereknek (Decision Support System, DSS) neveztük a ma üzleti intelligencia rendszereknek hívott rendszereket, vagy használtuk a vezetői információs rendszer, az OLAP, az adattárház és az adatbányászat szakszavakat az átfogó üzleti intelligencia kifejezés helyett.[3]

---

[2] A 3.1, 3.2, 3.3 fejezetek rendre a [4], [5], [6] források alapján készültek.

[3] Az itt felsorolt fogalmakat később részletesebben tárgyaljuk.

Nagy általánosságban azt mondhatjuk, hogy üzleti intelligencia rendszereket azért vezetnek be a vállatok, hogy javítsanak a meglévő adataik elérhetőségén, azaz hogy könnyebben, gyorsabban, szélesebb körben hozzáférhessenek adataikhoz, úgy és olyan formában, ahogy a munkájukhoz szükségeltetik.

Egy üzleti intelligencia rendszer kiépítéséhez általában több szoftverre is szükségünk lesz. Jellemzően kell egy adatbázis-kezelő, ahol az elemzéshez szükséges adatokat tárolni fogjuk, kell egy adatbetöltő, amellyel fel tudjuk tölteni üzleti intelligencia rendszert, és kell egy megjelenítő felület is, amin keresztül lekérdezhetjük a BI rendszer adatait, módosíthatjuk annak modelljeit.

Ezek a szoftverek külön-külön független gyártóktól is beszerezhetőek, de ma már a nagy gyártók mindegyike (Microsoft, ORACLE, IBM, SAP, SAS) rendelkezik a vállalat üzleti intelligencia igényét teljes egészében kielégíteni képes, integrált szoftvercsomaggal.

Ahogy üzleti intelligencia szoftver nincs, úgy üzleti intelligencia technológia sem létezik. Egy üzleti intelligencia rendszer összeállításához jellemzően technológiák egész hadát kell mozgósítanunk. Használni fogunk adatbázis-kezelő, adatbetöltő, riportkészítő, stb. technológiákat.

A The Data Warehousing Institute ezen üzleti intelligencia technológiákat aszerint csoportosította, hogy mekkora a komplexitásuk és mekkora a potenciális üzleti értékteremtő képességük [11]:

Eszerint megkülönböztetünk

- Riportkészítő technológiákat, amelyek arra adnak választ, hogy mi történt. Ide tartoznak a lekérdező, riportkészítő és kereső üzleti intelligencia technológiák
- Elemző technológiákat, melyek a miért kérdésre adnak választ. Ide tartoznak az OLAP és az adat-vizualizációs technológiák
- Monitorozó technológiákat, amelyek a Mi történik most? kérdésre adnak választ. ide tartoznak a teljesítmény menedzsment eszközök, az irányítópult (dashboard) és a mutatószámrendszer (Scorecard) technológiák
- És az előrejelző technológiákat, melyek a Mi történhetne? kérdésekre adják meg a választ. Ebbe a kategóriába az előrejelző és adatbányász technológiák tartoznak.

A lista elejétől a vége felé haladva egyre komplexebbek a technológiák és egyre nagyobb hatásuk lehet a vállalat életére, teljesítményére, profitjára.

Összefoglalva:

Az üzleti intelligencia egy gyűjtőfogalom. Howard Dresner pont azért alkotta meg az üzleti intelligencia kifejezést, hogy egyszerűen és egységesen lehessen kezelni és hivatkozni mindazon rendszerekre, eszközökre és technológiákra, amelyek célja a döntési folyamat támogatása, javítása.

Ezen üzleti intelligencia definícióba tartozó technológiák pedig a következők:

- Adattárházak
- OLAP (Többdimenziós (multidimensional) adatbázis-kezelők)
- Üzleti tervező (Planning), előrejelző (Forecasting) és konszolidáló alkalmazások
- Riportkészítő (Reporting) alkalmazások
- Irányítópultok (Dashboard), mutatószám (Scorecard) rendszerek
- Teljesítmény monitorozó (Performance Monitoring) eszközök
- Adat, szöveg és hangbányászat (Data Mining, Text Mining és Voice Mining)
- Adatvizualizáció (Data visualization)

3.2 Döntéstámogató rendszerek

A döntéstámogató rendszereket (Decision Support System, DSS) kezdetben úgy definiálták, mint olyan számítógép alapú rendszerek, amelyek segítik a döntéshozás folyamatát. Ez a definíció azonban túl általános, mivel ez alapján döntéstámogató rendszer lehet akár egy Excel tábla, amely tartalmaz valamilyen szimulációs modellt vagy egy Pivot tábla, de egy fejvadásznak ilyen lehet például az iwiw is, vagy akár a Google is.

Később a definíció kiegészült: a döntéstámogató rendszer egy interaktív, számítógép alapú rendszer mely adatbázisok és modellek felhasználásával segíti a döntéshozókat a nem jól strukturált problémák megoldásában.[4]

[4] Nem jól strukturáltnak nevezünk egy problémát, ha nem ismerjük annak összes megoldási alternatíváit és az egyes alternatívák értékét, egymáshoz viszonyított preferenciáit

3.2.1 Döntéstámogató rendszerek szoftvertechnológiája

Amikor döntéstámogató rendszerek építéséről beszélünk, 5 fő szoftvertechnológiát kell megemlítenünk. Ezek:

- Adattárház technológiák
- OLAP (többdimenziós adatbázis-kezelők)
- Oszlopalapú adatbázis-kezelők
- Adatbányász technológiák
- Riportkészítő vagy más néven kiaknázó eszközök

Döntéstámogató rendszereket ezen technológiai komponensek felhasználásával építhetünk. Mindennek az alapja az adattárház illetve az adattárház-technológiák alkalmazása. A döntéstámogató rendszerek nem követelik meg az adattárházat, de használják annak technológiáit egy saját adatbázis létrehozásához és napi töltésének biztosításához.

3.2.2 A BI és a DSS közti különbség

Hétköznapi értelemben az üzleti intelligencia rendszer és a döntéstámogató rendszer között nincs különbség, mindkettőt használhatjuk döntéstámogatásra. Ebből az okból kifolyólag gyakran keverik is a kettőt. A leglényegesebb különbség az, hogy a döntéstámogató rendszerek elsősorban a nem jól strukturált döntési problémák megoldására lettek kitalálva, az üzleti intelligencia rendszereket pedig használhatjuk jól strukturált döntési problémák megoldására is.

3.3 Vezetői információs rendszer

A Dictionary of Accounting Terms szerint a vezetői információs rendszer (Management Information System) számítógép alapú vagy manuális rendszer, amely a döntéshozók számára használható információvá alakítja az adatot. A Vezetői információs rendszernek 3 fő feladata van:

- Eseti (Ad hoc) és ismétlődő riportkészítés. Például pénzügy beszámolók, készletértékek vagy a vállalat teljesítményét bemutató riportok

- A menedzsment „Mi lenne ha" kérdéseinek megválaszolása. Például milyen hatása lenne a vállalati cash-flow-ra, ha megváltoztatnánk a fizetési feltételeket?
- A döntéshozás támogatása

A wikipédia szerint a vezetői információs rendszer (Management information system) egy részhalmaza a vállalat teljes üzleti folyamatát lefedő, a menedzserek által üzleti problémák megoldására használt dokumentumoknak, technológiáknak, eljárásoknak. (mint például egy termék vagy szolgáltatás árazása vagy a stratégia meghatározása)

A hazai köznyelv szerint a vezetői információs rendszer egy olyan rendszer, amelyen keresztül az összegzett adatoktól *lefúrhatunk* egészen az elemi adatokig (tranzakciókig).

Sajnos egyik definíció sem teljesen helytálló. Vezetői információs rendszert elsősorban azért építünk, hogy elérhetővé tegyük az információkat a döntéshozóknak, döntés-előkészítőknek, akkor és olyan formában, ahogy szükségük van rá.

Ahhoz, hogy az összetettebb kérdésekre is válaszolni tudjunk, szakítanunk kell a meglévő adatmodelljeinkkel és a hagyományos adatbázis-kezelő technológiákkal. Egy új rendszert kell építenünk, amely már kifejezetten arra lesz optimalizálva, sőt csak arra lesz optimalizálva, hogy belőle az adatok könnyen és gyorsan kinyerhetők legyenek. Ezt a rendszert nevezik vezetői információs rendszernek

Ahhoz tehát, hogy az összetett lekérdezéseket, a gyors válaszidőket és az egyszerű kezelhetőséget megvalósítsuk, el kell szakadnunk meglévő technológiáinktól és egy új bázisra kell felépíteni a vezetői információs rendszert. Ez az új bázis pedig az OLAP.

4 OLTP és OLAP rendszerek

Az eddigiekben többször említésre kerültek az OLAP rendszerek. De mi is az az OLTP és az OLAP, és miben térnek el egymástól?

Amikor OLTP (On-Line Transactional Processing – On-line tranzakció feldolgozó) rendszerekről beszélünk, akkor olyan rendszerekről van szó, amelyek tranzakció orientáltak. Ez alatt általában a hagyományos adatbáziskezelő-rendszereket értjük, de ide tartozhatnak pl. a szöveges adatbázis-kezelők is. Az OLTP rendszerek jellemzője az egyidejűleg futó akár több ezer tranzakció, több száz felhasználó, akik a CRUD[5] műveletek akár mindegyikét végrehajthatják. Ezek a műveletek általában csak nagyon kevés rekordot érintenek, viszont nagyon sok van belőlük. A valós idejű üzleti műveletek végrehajtására tervezték őket.

Jellemző a feladat-orientált megközelítés, a normalizáltság, az adatbázisok sok táblából és sok kapcsolatból állnak, tapasztalatlan szemmel nézve már-már átláthatatlan, pókháló-szerű rendszert alkotnak.

Egy szervezeten belül tipikusan több OLTP rendszer fut, melyek általában nem átjárhatóak, így adataink szétszórva helyezkednek el, nehezen rendszerezhetők, összesíthetők, a bonyolult üzleti kérdések megválaszolására ilyen formában alkalmatlanok.

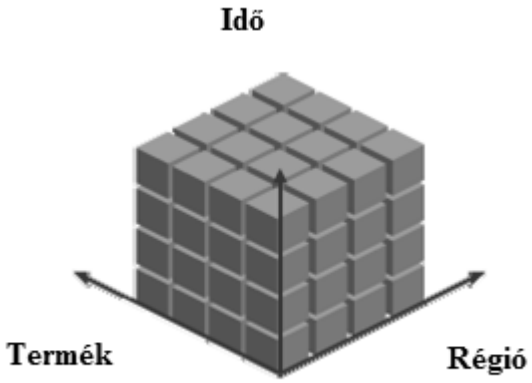
Az OLTP rendszerekben a nagytömegű adatok lekérdezése, analitikai elemzések végrehajtása, kimutatások készítése igencsak bonyolult, hosszadalmas és erőforrás-igényes.

Ezekre a problémákra nyújtanak megoldást az OLAP (On-Line Analytical Processing - On-line analitikus feldolgozó) rendszerek. Az OLAP nem más, mint egy speciális adatbázis-kezelő, amely úgy lett megtervezve, hogy nagyon gyorsan tudjunk lekérdezni számunkra fontos információkat. Az OLAP rendszerekben lényegesen kevesebb a párhuzamosan futó műveletek, műveletsorok száma, azonban a műveletek általában jóval nagyobb adatmennyiséget érintenek, mint az OLTP rendszerek esetében. Az OLAP rendszerekbe az

[5] CRUD – Create, Read, Update, Delete – Létrehozás, Olvasás, Módosítás, Törlés

adatok egy ETL [6] folyamat eredményeként más rendszerekből kinyerve, transzformálva, tisztítva és konszolidálva kerülnek.

Az OLAP rendszerekben az adatokat dimenzionális modellben tároljuk. Ezt úgy kell elképzelnünk, mint egy több dimenziós kockát, vagy több dimenziós mátrixot, melynek élei (a dimenziók) azok a szempontok, amelyek alapján adatainkat összegezni, vagy elemezni szeretnénk, és mezői a számunkra releváns számszerű adatok. Erre később még részletesebben kitérünk.



1. ábra. Példa OLAP kockára

[6] ETL – Extraction, Transformation, Load – Kinyerés, Transzformáció és Betöltés

Végül egy táblázatban összefoglalva az OLAP és OLTP rendszer tulajdonságait [13]:

Tulajdonságok	OLTP	OLAP
Orientáció	tranzakciók	adatanalízis
Felhasználó	vállalat adminisztrációt végző alkalmazottai	döntéshozók és őket információval támogató alkalmazottak
Feladat	napi folyamatok követése	döntéstámogatás, hosszú távú információgyűjtés és szolgáltatás
Adatbázis tervezése	Egyed-Kapcsolat modell, alkalmazás orientált	tárgy-orientált, csillagséma
Adatok	aktuális, up-to-date	történeti adatok, időben archiválva
Aggregált adatok	nem jellemző; részletes felbontás	felösszegzett, egyesített adatok
Adatok nézete	részletezett, relációs	felösszegzett, multidimenzionális
Felhasználók hozzáférése	olvasás/írás	legtöbbször olvasás, adattárház adatait nem módosítják
Hangsúly	adatbevitelen	információkinyerésen
Feldolgozandó rekordszám	tízes nagyságrendű rekord alkalmanként	akár milliós rekordszám
Felhasználók száma	viszonylag sok	kevés, közép és felsővezetők ált.
Prioritás	állandó rendelkezésre állás és megbízhatóság	rugalmasság, felhasználói önállóság

2. ábra Az OLTP és OLAP rendszerek összehasonlítása

A táblázatból is láthattuk, hogy mekkora különbség van az OLTP és OLAP rendszerek között. Feltehetjük az is, hogy különbözőségeik miatt az OLAP rendszerek másféle adatbázist igényelnek, mint az OLTP rendszerek. Ez így is van, az OLAP rendszerek adatbázisa az adattárház.

5 Adattárház-tervezési kérdések

5.1 Bevezetés

Az adattárházak a riportokhoz és analízishez történő adatgyűjtéssel, adatrendszerezéssel, az olyan eszközökkel, mint az OLAP vagy az adatbányászat az üzleti döntések támogatói. Bár az adattárházak legtöbbször relációs adatbázis technológiára épülnek, egy adattárház adatbázis szerkezete alapjaiban különbözik egy OLTP rendszerétől.

5.1.1 Adattárház, OLTP, OLAP, Data Mining

A relációs adatbázisokat egy meghatározott célra, hozták létre. Mivel az adattárházak rendeltetése különbözik az OLTP rendszerek rendeltetésétől, így az adattárházak tervezési jellemzői is különböznek az OLTP rendszerek tervezési jellemzőitől.

Adattárház adatbázis	**OLTP adatbázis**
Üzleti intézkedések, üzleti jellemzők analízisére tervezett.	Valós-idejű üzleti műveletek végrehajtására tervezett
Nagymennyiségű adat felvitelére, és összetett, kiszámíthatatlan, táblánként nagyon sok sort érintő lekérdezésekre optimalizált	Átlagos tranzakciókra optimalizált. Általában táblánként csak egy vagy kevés sor hozzáadása vagy elérése.
Érvényes (valid) adatokkal van feltöltve, nincs szükség valós idejű érvényesítésre (validálásra).	A tranzakciók során beérkező adatok validálására optimalizált;
Az OLTP-hez képest csak kevés konkurens felhasználót támogat.	Több ezer konkurens felhasználó

Az adattárházak már puszta létükkel is támogatják, segítik az OLTP rendszerek működését, mivel helyet biztosítanak az OLTP adatbázisokban felgyülemlett adatoknak (data offload), és olyan szolgáltatásokat biztosítanak, amelyeket ha hagyományos OLTP adatbázisban hajtanánk végre, akkor azok igencsak rontanák, csökkentenék az adatbázis teljesítményét.

A történeti (hitorical) adatokat tároló adattárházak nélkül az adatok statikus adathordozókra (pl. mágnesszalag) kerülnek archiválásra, vagy továbbra is az OLTP adatbázisban gyülemlenek, rontva annak teljesítményét.

Ha az adatokat csak megőrzésre archiváljuk, azok nem elérhetők az elemzők és döntéshozók számára. Ha adatainkat elemzés céljából továbbra is OLTP adatbázisban gyűjtjük, az adatbázis méretének növekedése folytatódik és egyre több indexre van szükségünk az elemző és riportozó lekérdezésekhez is. Ezek a lekérdezések folyamatosan növő mennyiségű adatot érintenek és dolgoznak fel, jelentőst terhelést okozva ez által az adatbázisnak. Már csak ezeknek az indexeknek a karbantartása is igen jelentős többletmunkát jelent az adatbázis számára. Ezen lekérdezések megírása, fejlesztése sem egyszerű feladat az OLTP adatbázisok tipikusan komplex felépítése miatt.

Az adattárház tehermentesíti az OLTP adatbázist, mely lehetővé teszi az OLTP számára a tranzakciók maximális hatékonysággal történő végrehajtását. A nagy elemzési és riportozó lekérdezéseket is az adattárház kezeli, amelynek nincs szüksége további indexekre ezek támogatásához. Adataink az adattárházba rendszerezve kerülnek, állandósulnak, hatékonyabbá téve ezzel az analitikai jellegű lekérdezések futását.

Az OLAP olyan technológia, amely kifejezetten arra készült, hogy kiváló teljesítményt nyújtson a BI lekérdezések számára. Úgy tervezték, hogy hatékonyan működjön az adattárházakban általánosan használt dimenzionális modellbe rendezett adatokon.

Az OLAP az adattárház dimenzionális modelljében tárolt adatokat többdimenziós kockákba rendezi, ezután ezeket a kockákat úgy dolgozza fel, hogy az adatokat különböző módon összesítő, összegző lekérdezéseket a maximális teljesítménnyel szolgálja ki. Például egy olyan lekérdezés, amelyben egy cég termékeinek eladásából származó bevételre és az eladott mennyiségre vagyunk kíváncsiak, adott termékekre, termékcsoportokra, földrajzi területre és időintervallumra szűkítve, pár másodperc alatt végrehajtható még több százmillió tárolt adatsor mellett is.

Az OLAP-ot nem nagymennyiségű szöveges vagy bináris adat tárolására, sem nagymennyiségű update tranzakció végrehajtására találták ki. Az adatok stabilitása és

konzisztenciája az adattárházban biztosítja az alapot az OLAP kiváló teljesítményéhez az analitikai lekérdezésekhez szüksége információ összesítésében.

Az adatbányászat olyan technológia, amely kifinomult és komplex algoritmusokat használ az adatok elemzésére és az elemzők és döntéshozók számára érdekes információk feltárására. Míg az OLAP az elemzők számára használható modellbe rendezi az adatokat, addig az adatbányászat végrehajtja az adatok elemzését és az eredményeket közvetlen a döntéshozóknak szolgáltatja. Vagyis az OLAP a modell vezérelt analízist (model-driven analysis), az adatbányászat pedig az adat-vezérelt analízist (data-driven analysis) támogatja.

Hagyományosan az adatbányászat az adattárház nyers adatain operált, sőt többnyire olyan szöveges adatállományokon, amelyeket az adattárházból nyertek ki. A modern rendszerek olyan eszközöket nyújtanak melyek segítségével elemezhetőek az OLAP kockák adatai és a relációs adatbázisokban tárolt adatok is. Sőt az adatbányászat eredményei beépíthetők az OLAP kockába egy új dimenzió felvételével, ezzel további támogatást nyújtva a modell vezérelt analízisnek.

5.2 Előfeltételek

Mielőtt belevágnánk egy adattárház kivitelezésébe, szükséges, hogy tisztázzuk és megértsük az adattárház tervezési céljait (architectural goals). Mivel az adattárházunk célja a felhasználók kiszolgálása, először is meg kell vizsgálnunk a felhasználók típusait, szükségleteiket, és az interakcióik jellemzőit.

5.2.1 Adattárház-tervezési célok

Az adattárház a felhasználók kiszolgálására jött létre, ezek a felhasználók elemzők és döntéshozók. Az adattárházat úgy kell kialakítanunk, hogy megfeleljen a következő követelményeknek:

- Felhasználói élmény biztosítása – a siker mércéje a felhasználói elégedettség.
- OLTP rendszerek akadályozása nélküli működés
- Konzisztens adatok egy központi gyűjtőhelyének biztosítása
- Összetett lekérdezések gyors megválaszolása
- Hatékony analitikai eszközök biztosítása (mint az OLAP és az adatbányászat)

A legsikeresebb adattárházaknak, amelyek teljesítik ezeket a követelményeket a következő jellemzőik vannak:

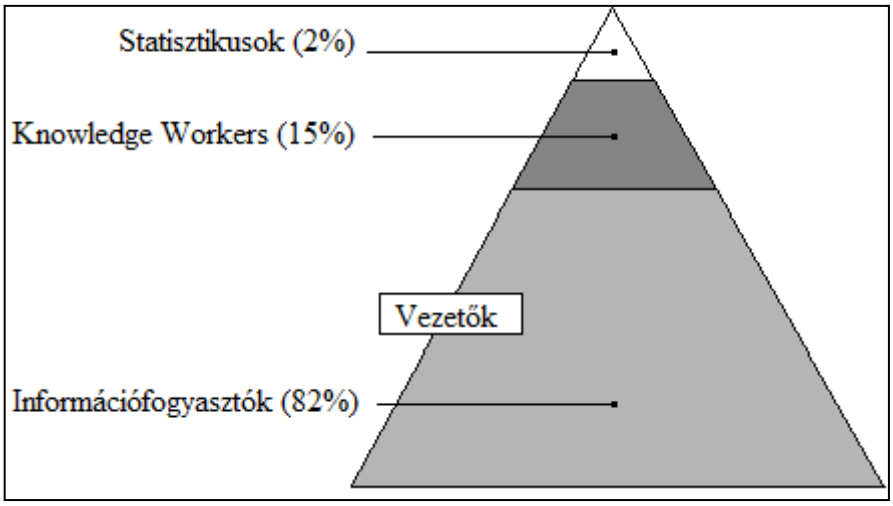
- Dimenzionális modellen alapulnak
- Történeti adatokat tartalmaznak
- Részletezett (detailed) és összesített (summarized) adatokat is tartalmaznak
- Több forrásból származó különböző adatok konszolidálása konzisztencia megtartásával
- Egyetlen célra összpontosítanak pl.: eladások, raktárkészlet vagy pénzügyek

5.2.2 Az adattárház felhasználói

Az adattárház sikere kizárólag felhasználóinak elégedettségétől függ. Felhasználók nélkül történeti adatainkat akár a picében is tárolhatnánk mágnesszalagra archiválva. A sikeres adattárház-építés a felhasználók és szükségleteik megértésével kezdődik.

Adattárházunk felhasználóit négy fő kategóriára oszthatjuk: Statisztikusok (Statisticians), Knowledge Workers[7], Információfogyasztók (Information Consumers),Vezetők (Executives), ahogy a következő ábrán látható. Természetesen az ábra a nagy általánosságot tükrözi, gyakorlatban a cég profiljától, gazdasági szerepétől, nagyságától és még egy sor más tényezőtől függően ezek az arányok a legváltozatosabb módon alakulhatnak.

[7] Sajnos a kifejezésre nincs igazán megfelelő magyar fordítás. Talán a legjobb fordítás a tudásmunkás, vagy szellemi munkás. Ebbe a csoportba általában a magasan képzett, különleges, kreatív munkát végző embereket soroljuk.



3. ábra. Az adattárház felhasználónak megoszlása egy vállalaton belül [17].

Statisztikusok: minden vállalatnál tipikusan csak kevés kifinomult elemző van (statisztikusok és operations research). Kicsiny számuk ellenére ők adattárházunk legjobb felhasználói. Ők azok, akinek a munkája a leginkább hozzájárul azon zárt belső rendszerek és folyamatok kialakításához, amelyek leginkább befolyásolják a vállalat műveleteit és nyereségességét. Létfontosságú, hogy ezek a felhasználók szeressék az adattárházunkat. Ezek az emberek legtöbbször nagyon önállóak, „önellátóak", elég számukra egy néhány egyszerű utasítást, instrukciót adnunk, hogy hogyan férhetnek hozzá az adatokhoz, és hogy a napnak melyik az a szaka, amikor saját fejlett és nagy mennyiségű adatot érintő lekérdezéseiket futtathatják. Innentől egyedül is elboldogulnak.

Knowledge Workers: Az üzleti elemzők csak egy nagyon kis csoportja végez új lekérdezéseket és elemzéseket az adattárház segítségével. Ezek azok a felhasználók, akik a „Designer" vagy „Analyst" verziójú felhasználói hozzáférési eszközöket kapják. Ők maguk fogják kitalálni, hogy hogyan lehet számszerűsíteni egy adott tárgyterületet. Egy pár iterációs lépés után lekérdezéseik és riportjaik közzétételre kerülnek az információfogyasztók számára. Knowlede Workerek azok, akik a legközelebbi kapcsolatban állnak az adattárházzal, a tervezés folyamán mindenképpen figyelembe kell venni őket. Részletes támogatást és egyéb információkat, néha tréningeket igényelnek.

Információfogyasztók: Adattárházunk legtöbb felhasználója információfogyasztó. Ők azok, akik valószínűleg soha nem fognak saját ad hoc jellegű lekérdezést írni. Statikus, vagy egyszerűbb interaktív riportokat használnak, amelyeket mások fejlesztettek. Könnyű

megfeledkezni ezekről a felhasználókról, hiszen csak a mások által előállított eszközökön keresztül érintkeznek az adattárházzal, de nem szabad elhanyagolnunk őket. Ez egy igen nagyszámú csoport, riportjaik meghatározóak a vállalat működése szempontjából. A tervezés folyamán velük jó kommunikációt kell kialakítanunk és figyelembe kell vennünk visszajelzéseiket.

Vezetők: A vezetők az információfogyasztók speciális fajtája. Néhány vezető ténylegesen a saját lekérdezéseit futtatja, azonban általában nem ez a jellemző. Egy vezető legcsekélyebb eltűnődése is feladatok egész sorát eredményezheti a többi felhasználó körében. Az ügyes adattárház-építő/implementáló/tulajdonos egy fejlett irányítópultot ad a vezető kezébe, feltéve, ha van értelme ezt tenni. Általában ez a többi adattárházzal kapcsolatos munkát követi, de sosem árt lenyűgözni a főnököket.

5.2.3 A felhasználók hozzáférése az adattárházhoz

A felhasználók számára szükséges információk érkezhetnek közvetlenül az adattárház alapjául szolgáló adatbázisból, vagy az olyan szolgáltatásokon keresztül, mint az OLAP vagy az adatbányászat. Az adatbázishoz irányuló közvetlen lekérdezéseket korlátoznunk kell. Csak olyan lekérdezéseket hajthatunk így végre, amelyek nem végrehajthatók már meglévő más eszközeinkkel, mivel ezek az eszközök általában jóval hatékonyabbak és sokkal kisebb terhelést jelentenek az adatbázisunk számára.

Riport(jelentés)készítő eszközök és egyéb egyedi alkalmazások gyakran fordulnak közvetlen az adatbázishoz. A statisztikusok rendszeresen nyernek ki adatokat speciális analitikai eszközökkel való felhasználásra. Elemzők gyakran írhatnak olyan adatok kinyerésére és feldolgozására irányuló lekérdezéseket, amelyek nem feltétlen elérhetők már meglévő eszközeinken keresztül. Az információfogyasztók nem lépnek közvetlen kapcsolatba a relációs adatbázissal, de kaphatnak e-mailben olyan riportokat, és hozzáférhetnek olyan weboldalakhoz, amelyek ezt teszik. A vezetők szabványos riportokat használnak, vagy másokat kérnek arra, hogy riportokat készítsenek számukra.

5.3 Adattárház-építés

Az adattárház projekt lépései nagyjából megegyeznek a legtöbb adatbázis projektével:

- Követelmények feltárása és összegyűjtése
- Dimenzionális modell kialakítása
- Az Operational Data Store (ODS)-t is magába foglaló architektúra kialakítása
- Adatbázis és OLAP kockák tervezése
- Adatkarbantartó alkalmazások fejlesztése
- Elemző (analysis) alkalmazások fejlesztése
- Tesztelés és üzembe helyezés

5.3.1 Követelmények feltárása és összegyűjtése

Mielőtt tárgyalásokba kezdünk a felhasználókkal, meg kell értenünk az üzlet, a vállalat működését. Ezután tárgyalunk és együttműködünk a felhasználókkal, és szükségleteiket projekt követelményekké alakítjuk. Ki kell derítenünk milyen információkra van szükségük munkájuk hatékonyabb végzéséhez, de nem azt, hogy mik azok az adatok, amikről azt gondolják, hogy az adattárházban kéne lenniük. Az adattárház-építő feladata az, hogy eldöntse milyen adatok szükségesek ezen információk szolgáltatásához. A megbeszéléseknek a felhasználók célkitűzéseiről és feladatairól kell szólniuk, és arról, hogy hogyan és mi alapján hoznak üzleti döntéseket. Mielőtt továbbhaladnánk, és egyből belecsapnánk a lecsóba tisztáznunk kell, hogy egyáltalán képesek vagyunk-e eleget tenni a felhasználói elvárásoknak, majd tisztázzuk azt is, hogy mit nem kell tudnia a rendszernek, mivel ezzel sok későbbi problémától kímélhetjük meg magunkat.

Az üzleti felhasználók szoros kapcsolatban kell, hogy álljanak a tervező csapattal a logikai tervezés időszaka alatt, mivel ők azok az emberek, akik ismerik és értik a létező adatok jelentését. Sikeres adattárház projektekben ezek az emberek adatszakértőkként jelen vannak a tervező csapatban is. Mindegy milyen a csapat összetétele, fontos, hogy a felhasználók sajátjuknak érezzék a létrejövő rendszert.

Az „adatszakértőkkel” csak a többi felhasználó után tárgyaljunk. Tőlük kell megtudnunk, hogy milyen adatok léteznek és hol, de ezt csak azután tehetjük miután megértettük a végfelhasználók alapvető üzleti szükségleteit. Már a tervezés korai fázisában ismereteink kell legyenek az elérhető adatokról, még mielőtt bejeznénk az üzleti szükségletek felmérését, de a meglévő adatok és azok milyensége nem befolyásolhatják a tárgyalások, az igényfelmérés menetét.

Kommunikáljunk gyakran és átfogóan a felhasználókkal. A követelmények szilárdulásával folytassuk a tárgyalásokat, úgy, hogy mindenki részt vegyen követelmény definíció összeállításában. Általános tapasztalatom az, hogy az informatikában járatlan felhasználók – bár saját szakterületükön kiváló szakemberek – nem tudják azt, hogy mely információk, mely részletek lehetnek fontosak, mindent „úgy kell belőlük kihúzni”.

5.3.2 A dimenzionális modell tervezése

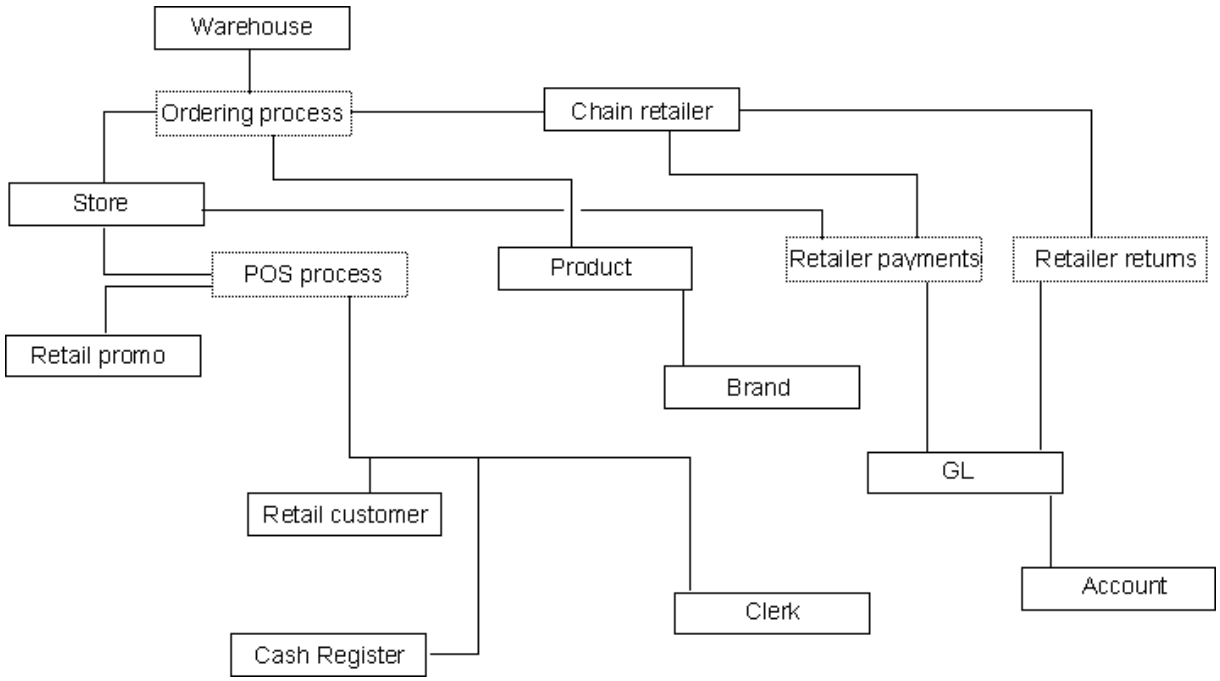
A felhasználói követelmények és a meglévő adatok alakítják a dimenzionális modell tervezését, a tervezés során mindenképpen üzleti célokat kell megcéloznunk, foglalkoznunk kell a részletesség szintjével, és azzal, hogy a dimenzionális modell milyen tényeket és dimenziókat tartalmazzon.

A dimenzionális modellnek meg kell felelnie a felhasználói követelményeknek és megkönnyíteni a közvetlen hozzáférést is. Úgy kell a modellt megterveznünk, hogy könnyű legyen karbantartani és könnyen alkalmassá tudjuk tenni a jövőbeni változásokhoz. A modell kialakítása végén egy olyan relációs adatbázist kapunk, amely támogatja az OLAP kockákat, hogy azonnali eredményeket szolgáltathassunk az elemzők számára.

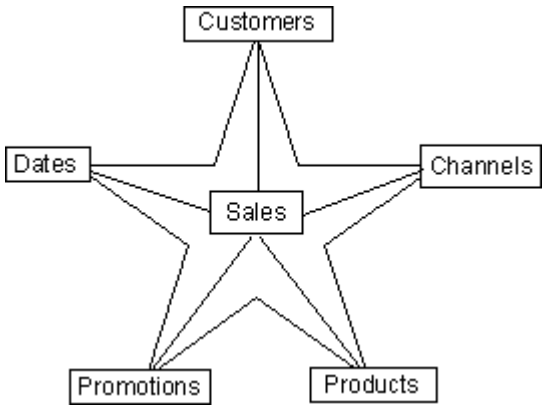
Egy OLTP rendszernek normalizált struktúrára van szüksége a redundancia minimalizálása, a bemeneti adatok validálása és a nagymennyiségű gyors tranzakció végrehajtása érdekében. Egy tranzakció általában csak egy üzleti eseményt foglal magába, mint például egy rendelés leadása, vagy egy számla fizetési könyvelése. Egy OLTP modell úgy néz ki, mint egy több száz, vagy több ezer egymással kapcsolatban lévő táblából álló nagy pókháló.

Ezzel szemben a tipikus dimenzionális modell csillag-, vagy hópehely modellt használ, amit könnyű megérteni, összekapcsolni az üzleti szükségletekkel, támogatja az egyszerűsített üzleti lekérdezéseket, és nagymértékű teljesítménynövekedést nyújt a táblák közötti kapcsolatok minimalizálásával.

Példának okáért hasonlítsunk össze egy nagyon leegyszerűsített OLTP adat modell az adattárházak dimenzionális modelljével:



4. ábra. Példa OLTP adatmodellre [17].



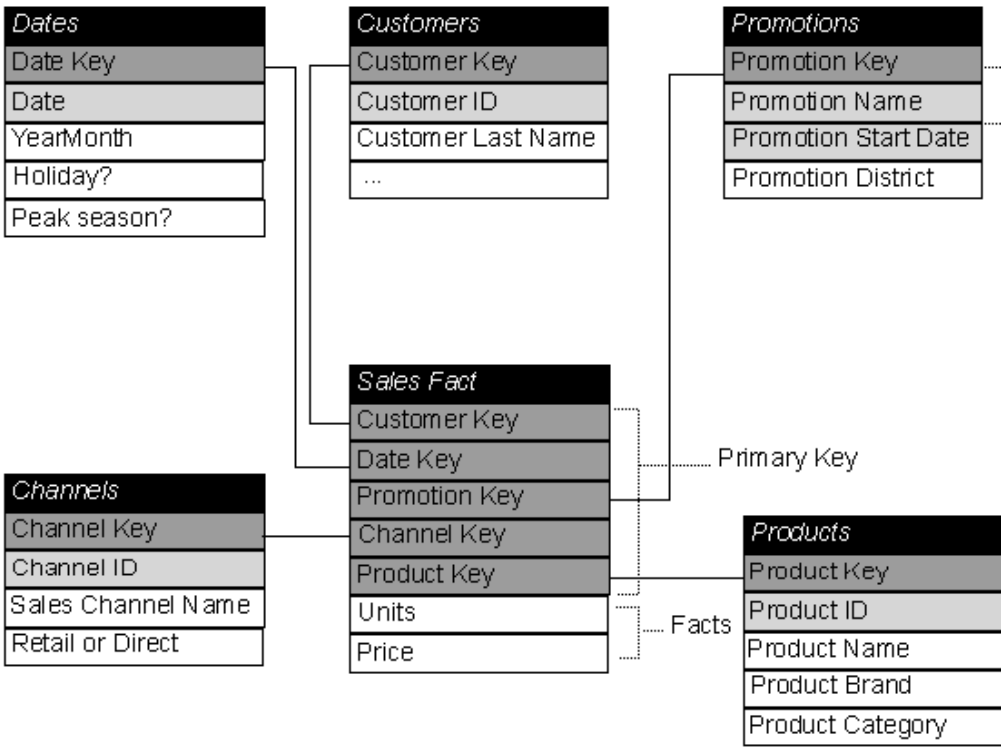
5. ábra. Példa dimenzionális modellre: a csillag modell [17].

5.3.2.1 Dimenzionális modell sémák

A dimenzionális modell meghatározó karakterisztikája a részletes üzleti tények egy olyan halmaza, amelyet több dimenzió vesz körül, amelyek leírást adnak ezekről a tényekről. Magyarul a dimenzionális modell központjában tények állnak, melyet a tényekről bővebb leírást adó dimenziók vesznek körül. A sémát adatbázisban létrehozva az tartalmazni fog egy központi ténytáblát és több dimenziótáblát. A dimenzionális modell eredményeként létrejöhet egy csillag- vagy hópehely séma.

5.3.2.1.1 Csillag sémák

Egy sémát csillag sémának hívunk, ha az összes dimenziótábla közvetlen a ténytáblához kapcsolható. A következő diagram egy klasszikus csillag sémát ábrázol:

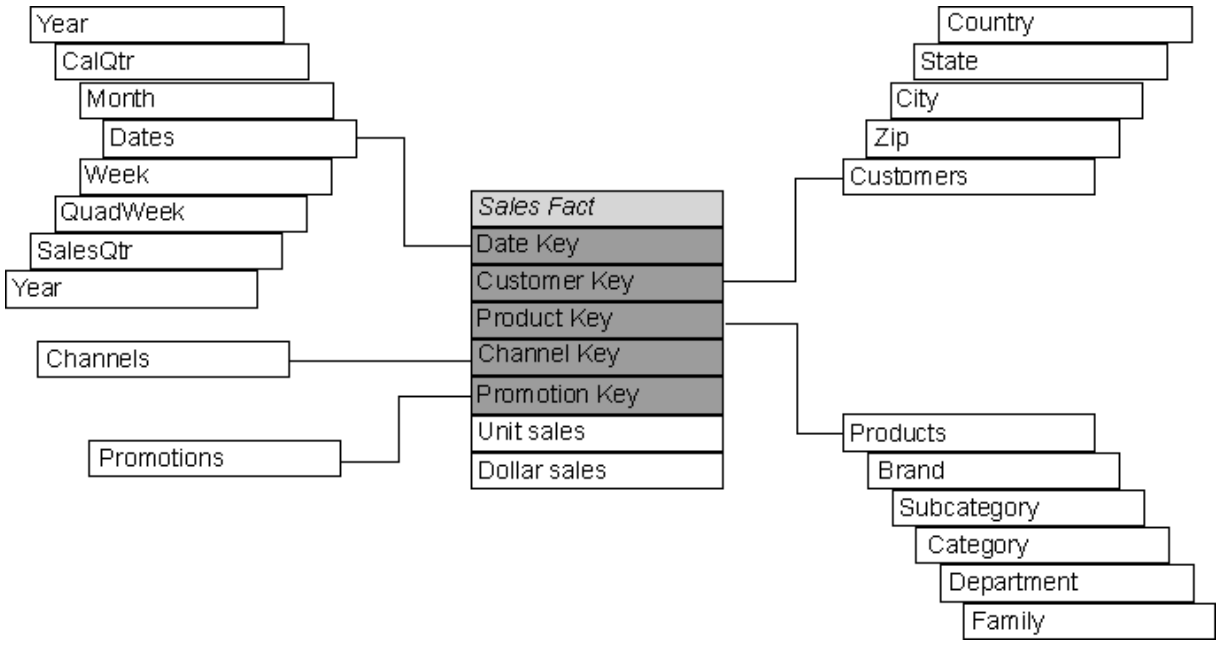


6. ábra. Klasszikus csillagséma (Sales - Eladások) [17].

5.3.2.1.2 Hópehely sémák

Egy sémát hópehely sémának nevezünk, ha a dimenziótáblák nem közvetlenül, hanem más dimenziótáblákon keresztül kapcsolódnak a ténytáblához.

A következő diagramon egy hópehely séma látható több hópehely dimenzióval:



7. ábra. Példa hópehely sémára [17].

5.3.2.2 Csillag vagy hópehely?

Mind a csillag-, mind a hópehely sémák dimenzionális modellek. A különbség fizikai implementációjukban van. A hópehely sémák esetén egyszerűbb a dimenziók karbantartása, mivel jóval normalizáltabbak. A csillag séma megfelelőbb közvetlen felhasználói hozzáférés esetén és leginkább az egyszerűbb és hatékonyabb lekérdezéseket támogatja. Az, hogy egy dimenziót csillagként, vagy hópehelyként modellezünk, függ magától a dimenzió természetétől. Például: milyen gyakran változik, mely elemei változnak. Gyakran kompromisszumokat kell kötnünk a könnyű felhasználhatóság és a könnyű karbantarthatóság között. Ugyanakkor fontos egy olyan egyszerű felhasználói interfészt fejlesztenünk, aminek segítségével felhasználóink ad hoc jellegű lekérdezéseiket futtathatják a dimenzionális adatbázison. Ilyen szempontból a csillag séma előnyösebb. Kompromisszumok helyett ezt gyakran úgy érik el, hogy létrehoznak egy úgynevezett virtuális csillagot. Ez általában egy hópehely dimenzió egy indexelt nézete.

5.3.2.3 Dimenziótáblák

A dimenziótáblák magukba foglalják a tényekhez kacsolódó attribútumokat és logikailag különálló csoportokba sorolják őket, pl.: idő, földrajzi elhelyezkedés, termékek, vásárlók és így tovább.

A dimenziótáblákat több helyen is felhasználhatjuk, ha az adattárház több ténytáblát is tartalmaz, vagy az adatok eloszlanak adatpiacok (data martok) között. Például a Termék (Product) dimenzió használható az eladásoknál és a raktárkészletnél is, vagy egy másik szervezeti egységhez tartozó data martnál is. Az ilyen dimenziókat, amiket egynél több sémában használunk fel „conforming dimensionnak”[8] hívják, ha a dimenzió minden egyes másolata megegyezik. A confroming dimensionok használata kritikus a sikeres adattárház-dizájn szempontjából.

A dimenziótáblák kialakításában, ahogy már említettük a felhasználók és a meglévő riportok segíthetnek. A felhasználó, aki azt mondja, hogy ő az adatokat az eladás helyére, vagy az eladott termékre csoportosítva szeretné látni, már azonosított is számunkra két fontos dimenziót. Az üzleti jelentések, amelyek az eladásokat az eladó vagy vásárló szerint csoportosítják, újabb dimenziókat azonosítottak. Majdnem minden adattárház rendelkezik idő dimenzióval is.

A ténytáblákkal ellentétben a dimenziótáblák általában kicsik, és lassan változnak, ritkán állnak kapcsolatban dátummal.

A dimenziótáblák rekordjai egy-a-többhöz kapcsolatban állnak a ténytáblával. Például, nagyon sokszor adhatunk el ugyanannak a vásárlónak, vagy egy adott termékből nagyon sokszor adhatunk el. A dimenziótábla a dimenziós bejegyzéssel kapcsolatos attribútumokat tartalmaz. Ezek az attribútumok gazdag, és felhasználóbarát leírások. Pl.: a vásárló neve, címe stb. Ezeket látjuk a lekérdezésekben és ezek szolgálhatnak megszorításokként is. Az OLTP adatbázisban lévő attribútumainkat át kell alakítanunk. Pl.: egy termék kategória az OLTP adatbázisban egy egyszerű integer kód, azonban a dimenziótáblába a kategória nevének kell bekerülnie. A kódot is betehetjük attribútumként a dimenziótáblánkba, hogy ez könnyítse a későbbi karbantartást. Ez a denormalizáció egyszerűsíti a lekérdezéseket, és javítja hatékonyságukat, valamint leegyszerűsíti a felhasználói lekérdező eszközöket is.

Gyakran hasznos létrehozni egy „nincs ilyen elem” vagy „ismeretlen elem” (unknown member) rekordot minden dimenziótáblában, amihez az árván maradt tény rekordok köthetők

[8] ~összehangoló dimenzió

a feltöltés alatt. A későbbiekben eldönthetjük, hogy továbbra is szükségünk van-e ezekre a rekordokra [14].

5.3.2.4 Hierarchiák

A dimenziótáblák adatai természetüknél fogva általában hierarchikusak. Ezeket a hierarchiákat meghatározza az, hogy hogyan kell az adatokat összesítenünk és csoportosítanunk. Például az idő dimenzió gyakran tartalmazza a következő elemeket: (összes), Év, Negyedév, Hónap, Nap. Egy dimenzió több hierarchiát is tartalmazhat. Az idő dimenzió tartalmazhatja a hagyományos naptári hierarchiát és a könyvelői felosztást is. Egy gyakran használt földrajzi hierarchia a következő: (összes), Ország vagy Régió, Értékesítési Régió, Állam (nálunk inkább Megye), Város, Áruház.

Mindegyik példában van egy (összes) bejegyzés. Ez a felső-szintű bejegyzés egy mesterséges kategória, amit a dimenzió első-szintű kategóriáinak csoportosítására használunk, és lehetővé teszi a dimenzióhoz tartozó összes tény adat összegzését egyetlen értékké.

A hierarchia lehet kiegyensúlyozott, kiegyensúlyozatlan, egyenetlen vagy szülő-gyermek kapcsolat, mint például egy szervezeti struktúra [14].

5.3.2.5 Helyettesítő kulcsok[9]

Az adattárház-tervezés kritikus része a dimenziók helyettesítő kulcsainak létrehozása. A helyettesítő kulcs a dimenziótáblák elsődleges kulcsa és független minden az eredeti adatforrásból érkező kulcstól. A helyettesítő kulcsokat az adattárház hozza létre és tartja karban, lehetőleg úgy, hogy ne tartalmazzanak információt magáról a rekord tartalmáról. Automatikusan növekvő egészek tökéletesek erre a célra. A rekordok eredeti kulcsait is elhelyezhetjük a dimenziótáblában, de nem használhatjuk elsődleges kulcsként. A helyettesítő kulcsok biztosítanak eszközt az adatok karbantartására, amikor a dimenziók változnak. A dátum és idő dimenziótáblákban speciális kulcsokat használunk, amelyek különböznek ezektől.

[9] Surrogate Keys, más fordításban: mesterséges kulcsok

5.3.2.6 A Dátum és Idő dimenziók

Az adattárházban minden művelet egy maghatározott időpontban megy végbe és az adatokat is gyakran összesítjük, vagy csoportosítjuk adott idő intervallum(ok) alapján. Bár az üzleti tények dátuma és ideje rögzítve van az adatforrásokban, a speciális dátum és idő dimenziók segítségével sokkal hatékonyabb mechanizmusokat kapunk az idő-orientált elemzések végrehajtására, mint egy egyszerű időbélyeg.

A dátum dimenzió általában kétféle hierarchiát tartalmaz: egy naptári évet és egy jogi (vagy könyvelői) évet.

5.3.2.6.1 Idő szemcsézettség

Egy dátum dimenzió naponként egy rekorddal elegendő, ha felhasználóinknak nincs szüksége egy napnál finomabb szemcsézettségre. Egy naponkénti dátum dimenziótábla így évente 365 rekordot fog tartalmazni.

Olyan finomabb szemcsézettséghez, mint a perc, vagy másodperc egy külön idő dimenziótáblát kell alkotnunk. Egy ilyen tábla naponként percek esetében 1440 sort, másodpercek esetén 86400 sort fog tartalmazni. Ennél tovább nem szabad mennünk, különben az idő dimenziótáblánk kezelhetetlenül nagy lesz. Ha egy esemény pontos időpontjára van szükségünk, azt a tény táblában kell tárolnunk.

Amikor külön idő dimenziótáblát alkotunk, akkor a tény tábla tartalmaz egy külső kulcsot a dátum dimenzióra és egy másikat az idő dimenzióra. A külön dátum –és idő dimenziók megkönnyítik a szűrő műveleteket, mivel elegendő lehet egyszerre csak az egyikkel összekapcsolnunk (join) a ténytáblát.

Az óránkénti szemcsézettséghez az óránkénti bontást a dátum dimenzióba ágyazhatjuk, vagy külön dimenzióban helyezhetjük el. Az üzleti szükségletek befolyásolják ezt a tervezési döntést is. Ha lekérdezéseinkben főleg napok határán átnyúló időszakok szerepelnek (pl: 2010-01-22 8:00-tól 2010-04-08 16:00-ig), akkor érdemesebb az órát és a napot ugyanabban a dimenzióban tárolni. Ezzel ellentétben a periodikus, ismétlődő napi események elemzése könnyebb különálló dimenziók esetén. Ha csak nincs erős okunk arra, hogy a dátum és idő dimenziókat kombináljuk, általában jobb, ha külön dimenziókban helyezzük el őket [14].

5.3.2.6.2 A Dátum és Idő dimenziók attribútumai

Gyakran hasznos a dátum dimenziókban is felvennünk néhány attribútum oszlopot, például a kiemelt időszakok jelzésére egy Boolean értéket, jelzőbitet, de a felhasználónak érdemes ezt majd valahogy dekódolnunk, mivel számára könnyebb azt látni és értelmezni, hogy kiemelt/nem kiemelt, mint azt, hogy Igaz/Hamis. Ilyen kiemelt időszak lehet például egy étteremláncnál az ebédidő, vagy egy internetszolgáltatónál a legnagyobb terheltség órái.

5.3.2.7 Lassan változó dimenziók (Slowly Changing Dimensions)

A dimenziók adatainak jellemző tulajdonsága, hogy azok lassan változnak. Új adatok kerülhetnek hozzáadásra, amikor új termékekkel bővül a kínálat, vagy új felvásárlóval szerződünk, de az olyan adatok, mint a meglévő termékek vagy felvásárlók nevei, csak nagyon ritkán változnak. Ugyanakkor előfordulhatnak olyan események, amelyek ezen adatok megváltozásához vezetnek, és természetesen ezeket a változásokat az adattárházban is kezelnünk kell. Különös figyelmet kell fordítanunk arra, hogy ez hogyan befolyásolja a történeti adatok követését és összesítését. A lassan változó dimenziók kifejezést szokás használni a dimenzió attribútumok megváltozásából eredő problémák tárgyalásánál. Az ilyen problémák leküzdésére általában a következő három tervezési megközelítést alkalmazzák [14]:

- A dimenzió rekord felülírása (1-es típus)
- Új dimenzió rekord felvétele (2-es típus)
- Új mezők létrehozása a dimenzió rekordban (3-as típus)

5.3.2.7.1 A dimenzió rekord felülírása (1-es típus)

1-es típus esetén az előzményeket (history) is felülírjuk, ami befolyásolhatja az analízis eredményeket, ha olyan attribútumot változtatunk meg, amely valamilyen csoportosítás alapját képezte. Az olyan dimenzió attribútumok megváltoztatása, amelyeket nem használunk analízisben, könnyen véghezvihető, egyszerűen csak felül kell írnunk az attribútum értékét. Például, ha a felvásárló címét a felvásárló dimenziótáblában tároljuk, és megváltozik a vásárló házszáma, az nagy valószínűséggel nem fogja befolyásolni az összesített adatokat, de ha a vásárló másik városba költözik vagy megyébe, annak hatása van földrajzi helyzet alapján összegzett adatokra.

A relációs adattárházban az 1-es típus a legegyszerűbb módja a lassan változó dimenziók kezelésének. A karbantartás egyszerűen az érintett attribútum érték megváltoztatása. Ez azonban összetett kezelési problémákhoz vezethet az aggregát táblák és az OLAP kockák esetében. Különösen igaz ez akkor, ha módosított attribútum tagja valamely hierarchiának, amely alapján az aggregátumok előre számítottak.

Az üzleti felhasználók elől az ilyen változás fontos információt rejthet el. Az attribútum módosítása után az attribútum előző értékei elvesznek.

5.3.2.7.2 Új dimenzió rekord felvétele (2-es típus)

2-es típusú változás esetén a történeti adatokat a változás pillanatában két részre osztjuk. Az esemény bekövetkezte előtti adatokat továbbra is ugyanúgy összegezzük, mint a változás előtt, az új adatokat viszont az új attribútum értékkel összhangban összegezzük és elemezzük.

Tekintsük a következő példát: egy értékesítéssel foglalkozó vállalkozásban az eladók minden eladás után jutalékot kapnak. Ezek a jutalékok befolyásolják a menedzserek és az ügyvezetők jutalékát és értékesítési csoportonként vannak összesítve. Amikor egy eladó csoportot vált, akkor az ez előtti eladásaiból származó jutalékoknak a régi csoportnál kell maradniuk, míg az új jutalékokat az új csoportnál kell elszámolni. Emellett az eladó teljes jutalék történetének hozzáférhetőnek kell maradnia, még akkor is, ha időközben az eladó többször is csoportot váltott. Az 1-es típusú megoldás ez esetben nem megfelelő, hiszen azzal az eladó összes jutalékát az új csoportnál számolnánk el.

A 2-es megoldás az, hogy megtartjuk az eladó már létező dimenzió rekordját és egy új rekordot veszünk fel, amely tartalmazza az új információt. Az eredeti rekord a jutalékok történeti adatait továbbra is az eredeti csoporthoz társítja és az új rekord az új jutalék adatokat az új csoporthoz társítja. A dimenziótábla létrehozásakor szokás extra oszlopokat felvenni az ilyen jellegű változások dokumentálására. Lásunk pár példát az ilyen oszlopokra:

- „Row Current” (Jelenlegi sor, aktuális sor), egy Boolean mező, mely jelzi, hogy a rekord az aktuális állapotot reprezentálja.
- „Row Start”, egy dátum mező, mely azonosítja, hogy a rekord mikor volt hozzáadva
- „Row Stop”, egy dátum mező, mely azonosítja, hogy a rekord mikor szűnt meg aktuálisnak lenni

Ahogy már többször, ez esetben is szükség van a helyettesítő kulcsok használatára. Az OLTP rendszerben nagy valószínűséggel az eladó alkalmazott azonosítóját (employee number) használták kulcsként. Még ha más kulcsot is használtak, nem valószínű, hogy az OLTP rendszerben több rekordot hoztak létre ehhez az egy személyhez. A dimenziótáblában sem tudunk létrehozni egy újabb rekordot ehhez a személyhez, hacsak nem helyettesítő kulcsokat használunk az eredeti elsődleges kulcs helyett. Emellett ha a dimenziótábla attribútumaként eltároljuk az eladó alkalmazott azonosítóját, az továbbra is segítségünkre lesz az adatok összesítésében és a teljes jutalék történet követésében, függetlenül attól, hogy az eladó hány csoportnál dolgozott.

Néhány lekérdezés még a 2-es típus esetén is módosulhat. Az eladó példában azok a riportok, amelyek az eladó dimenzió alapján összegeznek, most már két bejegyzést fognak mutatni egy eladóhoz. Valószínűleg nem ez a kívánt eredmény, és módosítanunk kell a lekérdezést, hogy a helyettesítő kulcs helyett az alkalmazott azonosító alapján összesítsen.

5.3.2.7.3 Új mezők létrehozása a dimenzió rekordban (3-as típus)

A harmadik megoldás új mezőket vesz fel a régi adatok tárolására kísérletet téve a változások horizontális követésére a dimenziótáblában. Gyakran csak az eredeti és az aktuális értékek visszanyerhetők, és a közbenső értékek figyelmen kívül hagyásra kerülnek. Ennek a megoldásnak az előnye az egy entitáshoz tartozó több dimenzió rekord elkerülése, azonban hátránya a történeti adatok zavara, és az ezeket az új mezőket használó lekérdezések nagyfokú összetettsége. Ha adattárházunkat az 1-es és 2-es típusú változások kezelésére alkalmas, nincs szükségünk a 3-as megoldásban rejlő karbantartási problémákra és összetettségre.

5.3.2.8 Gyorsan változó dimenziók

Egy dimenziót gyorsan változó dimenziónak (Rapidly Changing Dimension, Large Slowly Changing Dimension) tekintünk, ha egy vagy több attribútuma rendszeresen változik akár több sorban is. Gyorsan változó dimenziók esetén a dimenziótábla a 2-es megoldást használva igen gyorsan igen nagyra nőhet. A „gyors” és „nagy” kifejezések természetesen relatívak. Például egy 50.000 rekordos vásárló tábla vásárlónkénti évi 10 változással 10 éven belül 5 millió sorosra nő, feltéve, ha közben a vásárlók száma nem nő. Ez talán még egy elfogadható

növekedési ráta, de ha kiindulásként egy 10 millió rekordos táblát veszünk vásárlónként évi átlag két változással, akkor 10 éven belül táblánk több százmillió sorosra nő.

A folyamatosan változó értékekre, mint az életkor, a méret, a súly vagy a bevétel, sávokat vezethetünk be, melyek csökkentik a változások számát. Például a bevétel a következő sávokra osztható: [0 – 1 millió], [1 millió – 5 millió], [5 millió – 10 millió] és így tovább. Bár a 2-es típusú változást nem kellene a kor, életkor nyomon követésére használnunk, az életkor sávokat gyakran más célokra használják, mint pl. analitikai csoportosítás. A születési dátum könnyen használható a kor kiszámítására, amikor szükség van rá. Az üzleti szükségletek határozzák meg, hogy milyen attribútumok alkalmasak a sávokká konvertálásra.

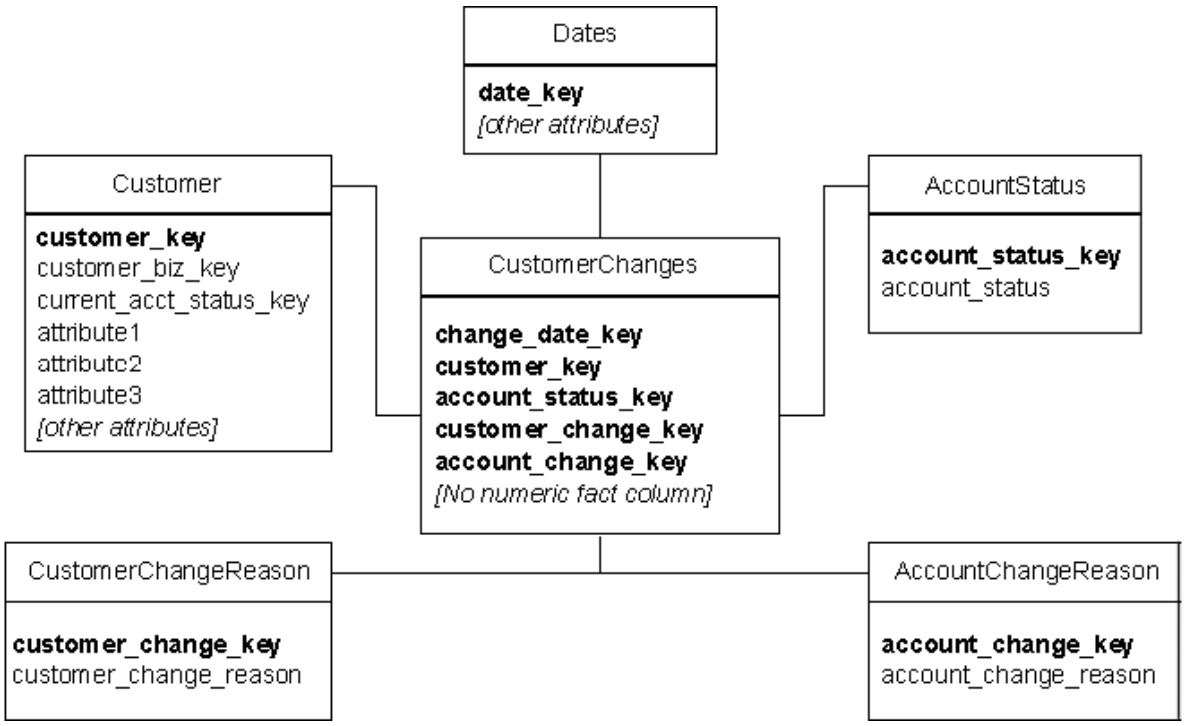
Gyakran a helyes megoldás a gyorsan változó attribútumokkal rendelkező dimenziók kezelésére az, hogy a problémás attribútumokat kiemeljük, és egy vagy több új dimenziót alkotunk. Figyeljük meg a következő példát:

A vásárlók fontos tulajdonsága a számla állapotuk (jó, késésben, tartós késésben, hátralékban, felfüggesztve), és a számla állapotának története. Az idő múlásával sok vásárló fog egyik állapotból a másikba kerülni. Ha az attribútumot a dimenziótáblában hagyjuk, és 2-es típusú változtatást végzünk, minden egyes alkalommal, ennek az egy attribútumnak a megváltozása esetén egy új sor kerül a táblánkba. A megoldás ennek az attribútumnak egy külön számla állapot (account status) dimenzióba kiemelése, melynek összesen 5 tagja van, ez az 5 lehetséges állapot.

A vásárló táblában egy külső kulcs mutat a számla_állapot táblára, amely a számla jelenlegi állapotát jelzi. A vásárló tábla rekordján 1-es típusú változást hajtunk végre, ha a számla állapota változik. A ténytábla is tartalmaz a számla-állapotra mutató külső kulcsot. Amikor új rekordot veszünk fel a tény táblába, a vásárló azonosító segítségével a vásárló táblából kikereshetjük a vásárló jelenlegi számla állapotát, és elhelyezhetjük a tény rekordban. Így a vásárló számlatörténetét is elhelyeztük a ténytáblában. Ráadásul a gyorsan változó mező eltávolításából származó hasznok mellett, a külön számla állapot dimenzió az OLAP kockákban egyszerű pivot analízist tesz lehetővé a számla állapot alapján. Ugyanakkor a teljes számla történet megtekintéséhez szükség van a ténytábla, a vásárló tábla és a számla állapot tábla összekapcsolására, és a vásárló azonosító alapján történő szűrésre, ami nem túl hatékony a vásárlók számlatörténetének gyakori lekérdezése mellett.

Ez az eljárás kifejezetten jól működik egyetlen gyorsan változó attribútum esetén, de mi van, ha tíz vagy annál is több attribútum változik? Csináljunk mindegyiknek külön dimenziót? Talán, de így túlságosan elszaporodnának a dimenziótáblák, és a ténytábla külső kulcsainak száma igen gyorsan megnövekedne. Az egyik megközelítés az, hogy kombináljunk több ilyen kis dimenziót egy nagyobb fizikai dimenzióvá. Még így is elég nehézkes ezen attribútumok lekérdezése, mert ahhoz, hogy az attribútumokat vásárlókhoz rendeljük, a tény táblát is be kell vonnunk a lekérdezésbe. Sajnos az üzleti felhasználókat igen gyakran érdekli ez a fajta történeti információ, pl.: egy adott vásárló mozgása a különböző számla állapotok között.

Ha a felhasználóinknak gyakran van szükségük olyan dimenziók lekérdezésére, amelyek így széttöredeztek, a legjobb megoldás egy „ténymentes" séma létrehozása, amely az attribútum változásokra fókuszál. Például képzeljünk el egy olyan elsődleges adattárház sémát, amely a vásárlók vásárlásait követi. A vásárló dimenziót 2-es típusú lassan változó dimenzióként implementáltuk, és a számla állapot külön dimenzióba került. Hozzunk létre egy új ténytáblát VásárlóVáltozások (CustomerChanges) néven, amely nyomon követi a vásárlói adatok és a számla állapot változásokat. Erre egy példa látható a következő ábrán:



8. ábra. CustomerChanges - Példa egy ténymentes sémára [17].

A ténytáblába (CustomerChanges) csak akkor kerül új sor, ha a Vásárló (Customer) táblában olyan változás történt, amely információt tartalmaz a vásárló jelenlegi számla állapotáról. A ténytáblának nincs semmilyen numerikus mérőszáma vagy ténye. A tábla egy bejegyzése azt jelenti, hogy vásárló számunkra érdekes módon változott. Opcionálisan megvalósítható a változás okának követése is: CustomerChangeReason és AccountChangeReason dimenziótáblák.

Esetenként az adattárház szerkezete egyszerűsíthető azzal, hogy több kisebb, egymással különösebb kapcsolatban nem álló dimenziót egyetlen fizikai dimenzióvá egyesítünk, ezt „hulladék" („junk") dimenziónak nevezzük. Ez nagymértékben csökkentheti a ténytábla méretét, mivel nem kell annyi külső kulcsot tárolnunk egy rekordban. Ezt a dimenziót gyakran előre előállíthatjuk a több kisebb dimenzió Descartes szorzatát képezve. Ha ez az eljárás túl nagy dimenziótáblát eredményezne, a táblát feltölthetjük az adattárház feltöltése vagy frissítése során is a ténylegesen előforduló értékkombinációkkal [8].

5.3.2.9 Ténytáblák

Egy ténytáblának egy üzleti problémát, egy üzleti folyamatot és a felhasználók igényeit kell megcéloznia. Ezen információk nélkül a ténytábla kialakítása közben könnyű átsiklani fontos adatok felett, vagy beépíteni olyan nem használt adatokat, amelyek csak feleslegesen növelik az összetettséget, a foglalt tárhelyet és a számítási követelményeket.

A ténytáblák tartalmazzák az üzleti események részleteit. Gyakran igen nagyok, több százmillió sort tartalmazhatnak és akár több terrabájt helyet is foglalhatnak. Mivel a dimenziótáblák tartalmazzák a tényeket leíró rekordokat, így a ténytábla egyszerűen szűkíthető a dimenziótáblák külső kulcsait és numerikus tényadatokat tartalmazó oszlopokra. Szöveget, nagyméretű objektumot, nem-normalizált adatokat általában nem tárolunk ténytáblában.

Több tény táblát olyan adattárházakban használunk, amelyek több üzleti funkciót is megcéloznak, például eladások, raktárkészlet, pénzügyek. Mindegyik üzleti funkcióhoz tartoznia kell egy ténytáblának és valószínűleg több különböző dimenziótáblának is. Azok a dimenziótáblák, amelyek jelen vannak több üzleti funkciónál is, ugyanúgy kell hogy reprezentálják a dimenziót, ahogy ezt már korábban tárgyaltuk a Dimenziótáblák alatt.

Minden üzleti funkciónak saját sémája van, amely magában foglal egy ténytáblát, több hasonló dimenziótáblát és olyan dimenziótáblákat, melyek üzleti funkciónként egyediek. Az ilyen üzletág specifikus sémák lehetnek a központi adattárház részei, vagy külön adatpiacként (data mart) implementálva.

A nagyméretű ténytáblák méretüknél fogva partícionálásra szorulnak. A felosztás általában egy dimenzió mentén történik. A legtöbbször ez a dimenzió az idő, mivel ez általánosan jelen van az adattárházban leggyakrabban előforduló történeti adatoknál. Régebbi rendszerekben ez fizikailag több táblára osztást jelentett, melyből az SQL UNION művelet segítségével kaphattuk meg az eredeti halmazt. A modern rendszerek ennél jóval fejlettebben kezelik a partíciókat, és a fizikailag akár külön háttértáron elhelyezkedő „táblarészeket" is tudják egy egészként kezelni.

5.3.2.9.1 Additív és nem-additív mértékek

Az értékek, melyek számszerűsítik a tényeket általában numerikusak és gyakran mértékként (measure – mérce, mérték, mérőszám) hivatkozunk rájuk. A mérőszámok additívak egy dimenzió mentén, mint például az eladott mennyiség (Quantity) az eladások ténytáblában. A vásárlónkénti, termékenkénti, időszakonkénti eladott mennyiség mindig jelentőséggel bíró érték.

Vannak olyan mérőszámok, melyek nem additívak sem egy sem több dimenzió mentén, pl. a raktározott mennyiség (Quantity-on-Hand) egy raktárkészlet rendszerben, vagy az ár (Price) egy eladási (sales) rendszerben. Vannak olyan mérőszámok, melyek az idő dimenzió mentén nem additívak, viszont más dimenziók mentén igen. Az ilyen mérőszámokat szemi-additívnak nevezzük. Például a raktározott mennyiség (Quantity-on-Hand) lehet additív egy raktár (Warehouse) dimenzió mentén, hogy képet kapjunk az adott raktárban raktározott teljes mennyiségről egy adott időpillanatban. Viszont az idő dimenzió mentén nincs értelme összegeznünk az adatokat, ilyenkor egy aggregát függvény, például az átlag nyújthat jelentőséggel bíró adatokat a raktározott készletről. Az olyan mérőszámokat melyek egyik dimenzió mentén sem additívak nem-additívnak (nonadditívnak) nevezzük.

A nem-additív mérőszámok gyakran kombinálhatók additív mérőszámokkal, úgy hogy egy új additív mérőszámot kapjunk.

5.3.2.9.2 Számított mértékek

A számított mérőszámok olyan mérőszámok, amelyeket úgy kapunk, hogy már meglévő mérőszám(ok)ra alkalmazunk függvényeket. Ilyen lehet például a profit, vagy az ÁFA.

A számított mérőszámok lehetnek előreszámítottak, melyeket még a feltöltés folyamata közben kiszámítunk és a ténytáblában tárolunk, vagy lehetnek olyanok, amelyeket csak a felhasználáskor (on the fly) számítunk ki. Tervezési megfontolás az, hogy mely mérőszámok legyenek előre számítottak és melyek nem.

5.3.2.9.3 A ténytáblák kulcsai

A ténytábla logikai modellje tartalmaz egy külső kulcsot minden egyes dimenziótáblához. Ezen külső kulcsok kombinációja megadja a ténytábla elsődleges kulcsát. Tehát logikailag nincs szükségünk külön azonosító oszlopra, azonban az olyan fizikai tervezési megfontolások miatt, mint például a tábla partícionálás, lekérdezési teljesítmény, nem biztos, hogy megfelelő nekünk ez az összetett kulcs. Ezeket később részletesebben tárgyaljuk.

A ténytábla kulcsainak segítségével így a dimenziótáblák között több a többhöz kapcsolatot alakíthatunk ki.

5.3.2.9.4 Szemcsézettség

A ténytábla egy szemcséjének mérete a tény oszlopok azonosítása után derül ki. A granularitás az egy ténytábla bejegyzés által meghatározott részletességi szint nagysága. Lehet, hogy érdemes már a ténytáblában is valamilyen szinten összegzett adatokat (nagyobb szemcséket) tárolni, azonban a részletesség szintjének ilyenkor is elegendőnek kell lennie az üzleti szükségletek kielégítésére.

Inkább az üzleti szükségleteknek mintsem az fizikai implementációs megfontolásoknak kell meghatározniuk a ténytábla szemcséjének minimális méretét. Ugyanakkor jobb, hogy ha az adatokat olyannyira szemcsézetten tároljuk, ahogy csak lehet, mivel – bár a jelenlegi üzleti szükségletek nem indokolják – a jövőbeli üzleti elemzésekhez nagyobb részletességre lehet szükségünk. Az Analysis Services úgy van megalkotva, hogy gyorsan és hatékonyan összesítse a részletezett adatokat is az OLAP kockákban, így a nagyon finoman szemcsézett

ténytáblák nem jelentenek teljesítménycsökkenést a válaszidő szempontjából. A finom szemcsézettség az adatbányászat szempontjából is előnyös lehet a jelentékenyebb információrögök felfedésében [8].

Ne adjunk a ténytáblához olyan rekordokat, melyek összesítik a már amúgy is meglévő adatokat, ezeket tároljuk külön táblákban. A szemcsézettség minden szintjéhez külön táblára van szükség. Szerencsére ezeket nem feltétlen nekünk kell kézzel létrehoznunk és kezelnünk, az Analysis Services ezeket automatikusan létre tudja hozni.

Fontos figyelnünk a forrásrendszerekből származó rekordokra, hiszen ezek is tartalmazhatnak összesített adatokat. Például a megrendelések. A megrendelések tételeket tartalmaznak, és a tételek összesített adatait: szállítási költségek, adók, engedmények. A tételek a tények. A többi adat a tények összesítéséből származik. Ezeket nem kell elhelyeznünk a ténytáblában, azonban a megrendelés azonosítóját igen, hogy egy adott megrendelésre tudjunk majd összegezni. A külön összeg rekord a ténytáblában nem csak szükségtelen, de egyúttal használhatatlanná is tenné azt.

Az ilyen összegzett adatok kezelésének a legegyszerűbb módja, ha lebontjuk őket tényszintre. Az adók esetében ezt könnyű implementálni, szállítási költségek esetén a tömegből, és a termék értékéből kell számítanunk, esetleg valamilyen más adatot is figyelembe kell venni, de gyakran vannak még ennél bonyolultabb esetek is. Felhasználóink gyakran nem értik, hogy miért van szükségünk egy ilyen sémára, de erőltetnünk kell ennek kialakítását, mivel az így létrejövő séma sokkal hasznosabb és használhatóbb lesz.

5.3.3 Az architektúra fejlesztése

Az adattárház-architektúra az üzleti igényeknek megfelelően kialakított dimenzionális modellt tükrözi. A dimenziók kialakítása nagyban befolyásolja a dimenziótáblák kialakítását és a tények definíciója meghatározza a ténytáblák kialakítását.

Az hogy csillag vagy hópehely séma mellett döntünk inkább implementációs és karbantartási megfontolásoktól függ, mintsem az üzleti szükségletektől. Az információt mindkét esetben ugyanúgy tudjuk a felhasználók felé prezentálni, függetlenül attól, hogy melyik sémát használtuk. Az adattárház sémák elég egyszerűek és egyértelműek, szemben a többszáz táblával és kapcsolattal rendelkező OLTP adatbázisokkal. Ugyanakkor az adattárházban

tártolt nagymennyiségű adat miatt különös figyelmet kell fordítanunk a hatékonyságra és a teljesítményre.

Az adattárház-architektúrát úgy kell megterveznünk, hogy befogadja a jövőbeni adatmódosításokat, és hogy a későbbi bővülés minimális hatással legyen a meglévő szerkezetre. Szerencsére a dimenzionális modell eléggé egyértelmű sémáival egyszerűsíti ezeket a tevékenységeket. A ténytáblák adatai periodikus időközönként kerülnek hozzáadásra, legtöbbször csak kis hatással a dimenziókra. Például egy meglévő termék egy meglévő vásárlónak való eladásának nem lesz semmilyen hatása a termék, vásárló vagy áruház dimenziókra, csak a ténytáblába kerül egy új rekord. Ha a vásárló új, akkor az új tény rekorddal együtt egy új rekordot adunk a vásárló dimenziótáblához. Az adattárházban tárolt adatok történeti jellegéből adódóan törölni csak hibák javítása esetén kell. A forrás adatok hibái általában az ETL folyamat alatt kiderülnek.

Az adattárházban a dátum és idő dimenziók karbantartása független a többi dimenzió vagy ténytáblától. Elég évente frissíteni őket előre hozzáadva a következő év rekordjait.

A dimenzionális modell könnyű bővíthetőséget kölcsönöz önmagának. Új dimenzió attribútumok, vagy egy új dimenzió könnyen hozzáadható, általában hatás nélkül a meglévő sémákra. A meglévő történeti adatoknak változatlanoknak kell maradniuk. Ilyenkor az adattárházat karbantartó alkalmazásokat ki kell bővítenünk, de felhasználói alkalmazásainknak módosítás nélkül kell továbbműködniük, bár valamelyiket majd biztosan frissítenünk kell, hogy hasznát is vegyük az új információnak.

Az adattárházhoz teljesen új sémát is adhatunk, anélkül, hogy ennek bármilyen hatása lenne a meglévő funkcionalitásra. Hozzáadhatunk egy teljesen új üzleti tárgyterületet, ugyanúgy megtervezve a ténytáblát és a dimenziókat, melyek specifikusak az adott tárgyterületre nézve. A meglévő dimenziókat módosítás nélkül újra felhasználhatjuk, hogy megtartsuk az összhangot az adattárházban. Ha kevésbé finom szemcséjű dimenzióra van szükségünk, akkor a meglévő dimenziótábla alapján új, kevésbé szemcsézett táblát hozhatunk létre, de ennek is összhangban kell lennie az eredetivel, és azzal együtt kell karbantartanunk is.

Ezek a változtatások az OLAP kockákon is könnyen végrehajthatóak.

5.3.4 A relációs adatbázis és az OLAP kockák kialakítása

Ebben a szakaszban létrehozzuk a csillag- vagy hópehely sémát a relációs adatbázisban, definiáljuk a helyettesítő kulcsokat, és az elsődleges- és külső kulcsok alapján a létrehozzuk a kapcsolatokat. Létrehozunk indexeket, nézeteket, partíciókat. Az OLAP kockákat úgy alakíjuk ki, hogy felhasználóink szükségleteit szolgálják.

5.3.4.1 Kulcsok és kapcsolatok

A relációs adatbázis tábláit a dimenziótáblák helyettesítő kulcsainak definiálása és az elsődleges kulcs – külső kulcs kapcsolatok azonosítása után implementáljuk. Az elsődleges/külső kulcs kapcsolatokat létre kell hoznunk az adatbázis sémában, úgy ahogy azt az előzőekben már egy ábrán láthattuk.

A ténytábla összetett elsődleges kulcsát költséges karbantartani:

- Csak az index mérete majdnem akkora, mint a ténytábláé.
- Az elsődleges kulcs indexe legtöbbször klaszterezett indexként (clustered index) jön létre. Sok esetben a klaszterezett index kiváló lekérdezési teljesítményt nyújt. Ugyanakkor a ténytábla összes többi indexe a nagy klaszterezett index kulcsot használja, így az összes index igen nagy lesz, további tárterületre van szükség és a lekérdezési teljesítmény csökken.

Ennek eredményeképp sok csillag sémát elsődleges kulcs helyett integer típusú helyettesítő kulccsal definiálunk, vagy akár elsődleges kulcs nélkül. Az ajánlás az, hogy a ténytáblán definiáljuk az összetett elsődleges kulcsot, de mellette hozzunk létre egy IDENTITY oszlopot, amit egyedi klaszterezett indexként (unique clustererd index) használunk. Az adatbázis adminisztrátor dönti el, hogy ez a konstrukció jobb teljesítményt eredményez-e.

5.3.4.2 Indexek

A dimenziótáblákat elsődleges kulcsaikon – amelyek a már sokat emlegetett helyettesítő kulcsok – indexelnünk kell. A ténytábla elsődleges kulcsán egyedi klaszterezett indexnek kell lennie. Vannak olyan esetek, amikor az elsődleges kulcs indexeinek klaszterezettnek kell lenniük, más esetekben nem (lásd fentebb). Minél nagyobb a sémában a dimenziók száma,

annál kevesebb haszna van a ténytábla klaszterezett indexének. Nagyszámú dimenzió esetén előnyösebb az IDENTITY oszlop bevezetése.

5.3.4.3 Nézetek

Nézeteket azoknak a felhasználóknak kell létrehoznunk, akiknek az adatokhoz közvetlen hozzáférésre van szükségük. A nézetekhez való hozzáférésre engedélyt adhatunk a felhasználóknak, anélkül, hogy magukhoz az adatokhoz hozzáférésük lenne. A nézeteken keresztüli lekérdezések teljesítményének javítására használhatunk indexelt nézeteket.

A nézeteket és azok oszlopait úgy kell elneveznünk, hogy azt a felhasználóink is értsék. Ha az Analysis Services az adattárház elsődleges lekérdező motorja, akkor is egyszerűbb a kockákat megalkotni, ha a nézeteknek értelmes, olvasható neveik vannak.

5.3.4.4 Operational Data Store-ok fejlesztése

Néhány üzleti probléma a legjobban úgy oldható meg, hogy egy külön adatbázist hozunk létre a taktikai döntéstámogatáshoz. Az Operational Data Store (ODS) egy olyan üzemeltetési konstrukció, amely az adattárházak, és a hagyományos tranzakciós rendszerek egyes elemeit is tartalmazza. Mint az adattárház: több rendszerből konszolidált és tárgyterületenként csoportosított adatokat tartalmaz. Mint a tranzakciós rendszer: az ODS-ben módosíthatnak az üzleti felhasználók, és relatív kevés történeti adatot tartalmaz.

Az ODS klasszikus üzleti példája a telefonos ügyfélszolgálat. Az ügyfélszolgálat operátorainak nem igazán van szüksége az ügyfelek viselkedésében megfigyelhető trendeket felfedő analitikus lekérdezések futtatására. Az ő szükségleteik általában közvetlenek: a legfrissebb információra van szükségük a betelefonáló ügyfélt érintő összes tranzakcióról. Ezek az adatok több forrásrendszerből származhatnak, de egy egyszerűsített konszolidált módon kell, hogy az operátor előtt megjelenjenek [8].

Az ODS-ek implementációi eléggé változatosak az üzleti szükségletektől függően. Nincsenek szigorú szabályok az ODS implementálására nézve. Egyik üzleti probléma megoldásánál lehet csupán az OLTP rendszer tükrözésével replikált adatbázis, más probléma esetén pedig a csillag séma lehet a leghatékonyabb. A legjobb ODS megoldások e közé a két véglet közé esnek és magukban foglalnak valamilyen szintű adatintegrációt és –transzformációt.

Lehetséges úgy kialakítani az ODS-t, hogy mindemellett az ETL folyamat során az adattárház „staging area”-jaként szolgáljon.

Az ODS-ek további tárgyalása túlmutat ezen munka keretein.

5.3.5 Adatkarbantartó alkalmazások fejlesztése

Ebben a fázisban végre eljutottunk oda, hogy adattárházunkba valóban adatok kerüljenek. Ez az a fázis, ami a már többször emlegetett ETL – Extraction, Transformation, Load (Kinyerés, Transzformáció, Betöltés) folyamatáról szól. Ez az a folyamat, amely során összegyűjtjük a heterogén forrásrendszerekből (adatbázisok, szöveges állományok, táblázatok, stb.) származó adatokat, azokat a számunkra kívánatos formába konvertáljuk és elhelyezzük az adattárházban.

Az ETL folyamatok implementálása kulcsfontosságú egy adattárház projektben, akár a teljes projektre szánt idő 70%-át is erre kell fordítanunk. Így erről egy kicsit részletesebben beszélünk.

Alapvetően ezt a fázist is további két fázisra oszthatjuk:

- Az adattárház kezdeti feltöltése (warehouse population)
- A keletkező új adatok ütemezett betöltése

Ezek azonban csak logikailag különülnek el. A legtöbb esetben ugyanolyan, vagy nagyon hasonló alkalmazásokat használunk mindkét célra. Természetesen ezeket úgy kell elkészítenünk, hogy automatizáltak legyenek, ne kössenek le feleslegesen erőforrásokat. Az SQL Server Integration Services (SSIS) és az SQL Server Agent erre kiváló eszközöket biztosít. (Lásd lentebb)

5.3.5.1 Adatprofilozás

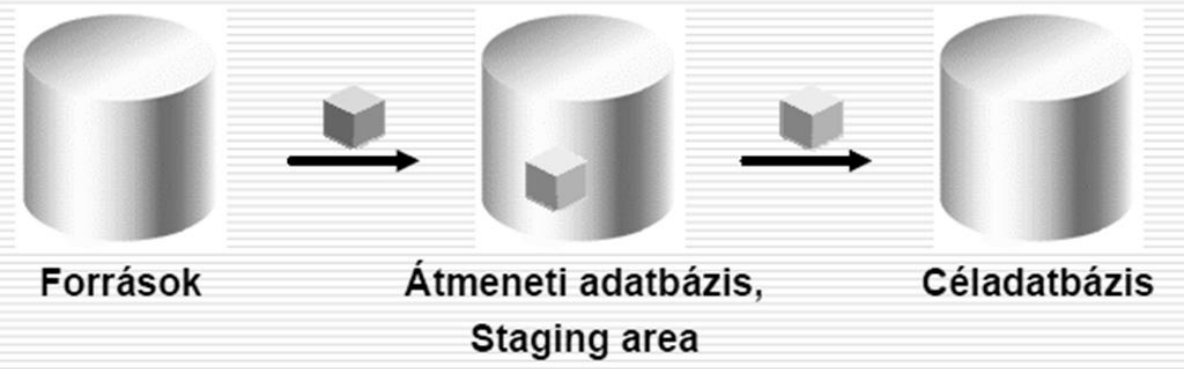
Az adatprofilozás az a folyamat, amelynek során megvizsgáljuk a forrásrendszerek adatait, azokról statisztikákat készítünk (például hány NULL értéket tartalmaznak az oszlopok), és információt gyűjtünk az adatok minőségéről (mennyire tiszták az adatok). Elsősorban nem ebben a fázisban használjuk, hanem a fizikai modell megtervezésekor, azért itt említem, mivel igen nagy hasznát vesszük a napi szinten érkező adatok vizsgálatakor is. Ha az érkező adatok

az adattárházban lévőkhöz képest nagy eltérést mutatnak, a betöltést megszakítjuk, és/vagy elhalasztjuk.

Természetesen a profilozást is gépi úton végezzük, erről később az SSIS-t bemutató fejezetben még olvashatunk.

5.3.5.2 Extraction

Extraction, vagy Capture, azaz kivonatolás. Ez az a folyamat, amikor forrásrendszereinkből kiemeljük[10] az adattárházunkhoz szükséges adatokat. A kivonatolás során szokás egy átmeneti adatbázist úgynevezett „staging area”-t használni, ahová az adatok a további feldolgozás előtt kerülnek. A staging area egy speciálisan erre a célra kialakított adatbázis, vagy adatbázistáblák. Szokták az Operational Data Store-okat is staging areaként használni.



9. ábra. Adatkinyerés, átmeneti adatbázis használata

A szakirodalom a kivonatolás következő formáit különbözteti meg [8]:

- Alkalmazás-támogatott kivonatolás – maga az adatot módosító alkalmazás továbbítja a változást az adattárház felé
- Triggerelt kivonatolás – az adatok változását triggerekkel követjük, ezek továbbítják a megváltozott adatokat az adattárház, pontosabban a staging area felé
- Naplózott kivonatolás – az adatbázis-kezelő naplóállományát megvizsgálva állapítjuk meg, hogy milyen változások történtek, és ezeket kivonatoljuk.

[10] A kiemelés szó nem félreértendő, természetesen az adatok az eredeti helyükről nem törlődnek, ott is megmaradnak.

- Időbélyeg alapú kivonatolás – az adatbázisbeli rekordoknak rendelkezniük kell valamilyen időbélyeg oszloppal, amiben a legutóbbi módosítás időpontját tároljuk, ez alapján döntjük el, hogy az adott rekordot kivonatoljuk-e
- Állomány-összehasonlításon alapuló kivonatolás – minden kivonatoláskor másolatot készítünk a teljes állományról, a következő kivonatoláskor ehhez hasonlítjuk az új állományt

Mindegyik módszernek megvan a maga előnye és hátránya. A valós életben arra törekszünk, hogy a forrásrendszereket a lehető legrövidebb ideig, a lehető legkevésbé terheljük. A kivonatolást általában a hajnali órákban szoktuk végrehajtani, amikor nagyon kevés, vagy egyetlen felhasználó sem használja őket, és a forrásrendszerek ütemezett karbantartó feladatai is már lefutottak (ezeket általában éjfél környékére időzítik). Ha nagy nemzetközi vállalatról van szó, aminek munkatársai világszerte rögzítenek adatokat, akkor az időzónák eltérése miatt nagyon meg kell fontolnunk, hogy egy-egy forrásrendszert mikor terhelünk.

5.3.5.2.1 Audit

Az adattárházba rekordot csak úgy töltünk be, hogy kiegészítjük azt a származására vonatkozó információkkal. egy adattárházba bekerülő rekordról mindenképpen tudnunk kell, hogy melyik forrásrendszerből érkezett, melyik betöltéssel (mikor) érkezett, és az, hogy mely adatkarbantartó alkalmazás (esetünkben SSIS csomag) töltötte be.

Az ilyen származásra vonatkozó információkat nevezzük audit információnak.

5.3.5.3 Transformation

Transzformálás alatt azt a folyamatot értjük mely során a forrásrendszerek formátumában meglévő adatokat az adattárház formátumára konvertáljuk (Pl. szöveges dátum mezőket alakítunk át datetime típusra, erre később látunk példát).

5.3.5.3.1 Adattisztítás

A transzformálás előtt az adatoknak tisztításon (data cleansing) kell átesniük, ugyanis a forrásrendszerekből származó adatokban gyakran vannak hibák (elgépelés, csonkolt adat,

stb.), ha figyelembe vesszük azt is, hogy adataink nem is egy forrásrendszerből érkeznek, akkor ez a probléma hatványozottan jelentkezik.

Vegyünk kiindulásként egy címet. Rossz esetben ezt a címet a forrás rendszerünk egyetlen oszlopában tárolják. Ezzel valamit kezdenünk kellene, hogy használhatóbbá tegyük. Mit tehetünk?

- Elemekre bontás: címünket elemi részekre bontjuk, pl. irányítószám, város, utca, házszám, és így tovább
- Egységesítés, sztandardizálás: bizonyos elemekre egységes jelöléseket vezetünk be. Pl.: sugárút = sgt
- Verifikálás: elemek konzisztenciájának ellenőrzése: az irányítószámhoz valóban ez a város tartozik?
- Illesztés: az egyik legérdekesebb feladat. Az illesztés segítségével megnézzük, hogy a rekordunk vagy egy hozzá bizonyos kritériumok alapján nagyon hasonló szerepel-e a cél adatbázisban, valamilyen referencia táblában, vagy esetünkben az adattárházban. Ma már lehetőség van arra, hogy meghívjunk webszolgáltatásokat, amelyek egyszerűen visszaküldik a megtisztított rekordot. Nem nálunk van a referencia adatbázis, nem mi tartjuk karban, felesleges munka nélkül elvégeztük ugyanazt a feladatot. Ha két rekord a kritériumoknak bizonyos mértékben eleget tesz, egyezőnek tekintjük őket (általában 98% az elfogadási küszöb). A rekordok ilyen módon történő megkeresését nevezzük fuzzy lookupnak.
- Dokumentálás: legvégül a meta adatok karbantartásával dokumentáljuk a folyamatot.

5.3.5.3.2 Transzformációk típusai

A transzformációk megvalósítása legtöbbször igen összetett feladat. Itt hat olyan alapeset van, amelyből építkezhetünk [8]:

- Szelekció: a rendelkezésre álló adatoknak csak egy részét használjuk fel, gyakran már a kivonatolás alatt eleve csak a felhasználandó adatokat kérdezzük le.
- Szeparálás és konkatenáció: az ugyanazon üzleti elemhez tartozó információkat választjuk szét, vagy egyesítjük
- Normalizálás és denormalizálás

- Aggregáció: a részletezett adatokból felösszegzett adatokat állítunk elő
- Konverzió
- Dúsítás: mezőszintű transzformáció, egy vagy több mező kombinálásából hozunk létre új mezőt

Az SSIS, ha nem is pontosan ebben a felosztásban, de mindehhez kiváló támogatást nyújt.

Most ott tartunk, hogy vannak adataink, melyeket megtisztítottunk, és a megfelelő formára transzformáltunk, már „csak” be kell töltenünk őket az adattárházba.

5.3.5.4 Load

A betöltés folyamán első és legfontosabb szempont, hogy nem törlünk semmit és nem írunk semmit felül, nem úgy, mint az OLTP rendszereknél. Ahol mégis szükség van változtatásra ott a lassan változó dimenzióknál már ismertetett eljárásokat használjuk. Itt nyernek igazán jelentőséget a már sokat emlegetett helyettesítő kulcsok, vagy mesterséges kulcsok az egyes rekordok azonosítására.

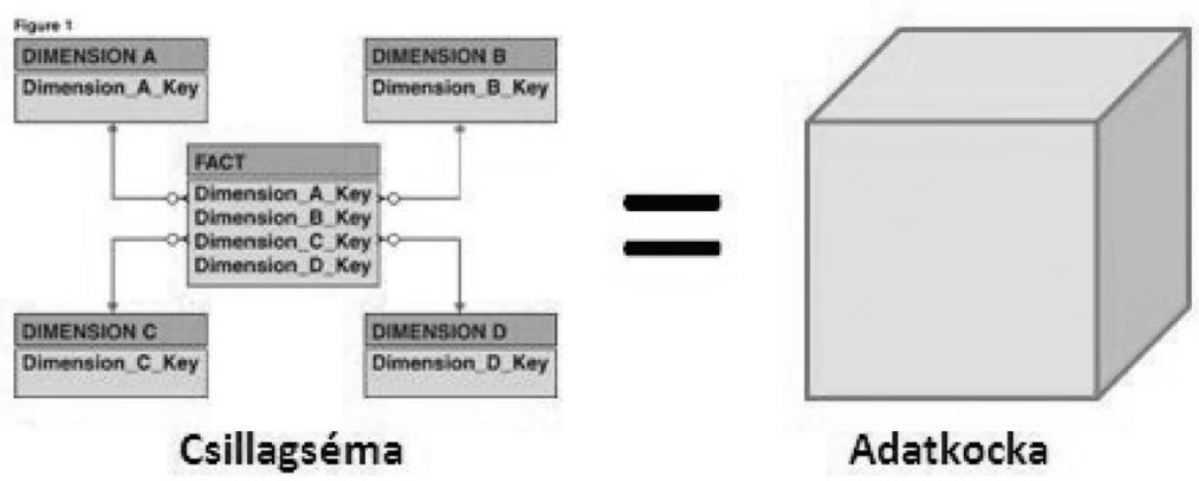
Mivel adattárház sémánk még mindig egy relációs séma, bizony figyelembe kell vennünk a kulcsmegszorításokat, tehát előbb azokat a táblákat töltjük fel, melyeknek kulcsaitól függ más tábla is. Például a ténytáblában külső kulcsként használjuk a dimenziótábla elsődleges kulcsát, tehát addig nem tudunk a ténytáblába az adott dimenzió bejegyzéshez kapcsolódó rekordot szúrni, amíg a megfelelő dimenzió bejegyzés nem létezik.

5.3.6 Analízis alkalmazások fejlesztése

Készen vagyunk? Igen és nem. Az adattárház adatbázisunkat felépítettük, de hol van az bizonyos multidimenzionális OLAP kocka, amiből oly egyszerűen és gyorsan tudunk lekérdezni?

Az OLAP adatbázisok multidimenzionális modellben, többdimenziós kockákban tárolják az adatot, mi pedig meglehetősen sok munka árán eddig mindössze egy olyan relációs sémát (csillagsémát) építettünk, ami valahogyan eltér az eddig megszokottól. Azért tettük ezt, mivel az OLAP kockák szerkezete nagyon hasonló a relációs csillagsémás adatmodellekhez. Az OLAP adatbázis dimenziói a csillagséma dimenziótábláinak, az adatkocka pedig a

csillagséma ténytáblájának feleltethető meg, így az OLAP adatbázis-kezelő képes a csillag sémát is lekérdezni, és a csillagséma alapján OLAP kockát is állíthatunk elő.



10. ábra. Csillagséma - Adatkocka megfeleltetés [15].

Miért érdemes tehát megduplázni adatainkat és letárolni mindkét-féle adatbázis-kezelőben?

Az OLAP-adatbázisokban kockákban tároljuk az adatokat, és egy adatkockába adatokat beszúrni meglehetősen nehéz. Az MDX-nek – az Analysis Services és még egynéhány OLAP-adatbázis-kezelő lekérdezőnyelvének – például nincs INSERT utasítása. Csak SELECT és UPDATE. Épp ezért az OLAP kockákat ritkán, naponta jellemzően egyszer töltjük, és ekkor vagy teljesen újraépítjük az egészet, vagy az új adatokat hozzáillesztjük a már meglévő kockához. Az adatok duplikálásával a beszúrás feladatát áttettük a relációs oldalra, ugyanakkor az OLAP-adatbázis-kezelőt teljes körűen használhatjuk, amely rendelkezik néhány olyan előnnyel, amellyel relációs társa nem [15].

- Lényegesen jobb a lekérdezési sebessége, mint relációs társáé. Nem csoda, hiszen erre optimalizálták (és nehéz úgy megfektetni egy lekérdezéssel, mint a relációst).
- Sokkal kifinomultabb és hatékonyabb jogosultságkezeléssel rendelkezik, mint a relációs adatbázisok. (Az OLAP-adatbázisban akár cellaszinten kezelhetünk jogosultságokat.)
- Lényegesen fejlettebb elemzést támogató funkcióval rendelkezik, mint a relációs társa. Például az előző év azonos időszakának kimutatására az OLAP már több mint 10 éve tartalmaz beépített függvényeket, a relációs adatbázis-kezelők továbbfejlesztési terveiben pedig csak most kezd feltűnni.

- És ami a legfontosabb: klasszisokkal jobb riportkészítő és lekérdező eszközök léteznek hozzá, mint a relációshoz, és ez az, amivel lehetőséget teremtünk a felhasználóknak, hogy saját maguk legyenek képesek riportokat készíteni.

5.3.7 Tesztelés és üzembe helyezés

Fontos, hogy a felhasználókat is bevonnunk a tesztelésbe. A kezdeti fejlesztők és tesztelő csapat által végzett tesztek után engedjük meg felhasználóinknak, hogy futtassák saját lekérdezéseiket, olyan lekérdezéseket, mint amilyeneket a rendszer üzembe helyezése után futtatnának. Így ellenőrzött körülmények között figyelhetjük meg, hogy hogyan viselkedik rendszerünk valós terhelés alatt. A felhasználók bevonása a tesztelésbe több előnnyel is jár:

- Kiküszöbölhetjük az esetleges hibákat, félreértelmezéseket, eltéréseket.
- A felhasználók szokják a rendszert.
- Az indexeket továbbhangolhatjuk, optimalizálhatjuk a teljesítményt

6 SSIS – SQL Server Integration Services

Az SSIS egy SQL Server komponens, a DTS (Data Transformation Services) utódja. Legegyszerűbben megfogalmazva arra használható, hogy adatokat töltsünk be egyik adatbázisból a másikba. Azonban az SSIS ennél jóval több.

Az SSIS egy általános célú adatbetöltő (ETL) eszköz, amely korántsem csak adatbázisok közt képes az adatok mozgatására.

Bár az SSIS az SQL Server programcsomag része és leginkább az SQL Serverbe történő adatbetöltésre van kihegyezve, le lehet tölteni hozzá úgynevezett „konnektorokat” többek között Oracle-höz, SAP-hez és a Teradatához. Így valóban általános célú eszközzé válik.

6.1 SSIS csomagok

Az SSIS fejlesztő eszköze a Visual Studioba beépülő Business Intelligence Development Studio. Ezzel – több más mellett – egy SSIS csomag készíthető. Az SSIS csomag az, amit tulajdonképpen futtatunk.

Egy csomag futtatható parancssorból (pl: dtexec /f c:\Eleresiut\Test-SSIS-Csomag.dtsx), vagy ütemezető SQL Server Agent segítségével rendszeres, periodikus futtatásra.

Egy csomag nem más, mint egy XML állomány. Egy SSIS projekt egy vagy több csomagot tartalmazhat. A csomagok könyvtárstruktúrába szervezhetők. A fájlrendszeren belül akárhol tárolhatók, vagy az SQL Server Package Store-ban, vagy SQL Server msdb adatbázisban.

Egy SSIS csomagnak két fő része van:

- Control Flow
- Data Flow

6.1.1 Control Flow

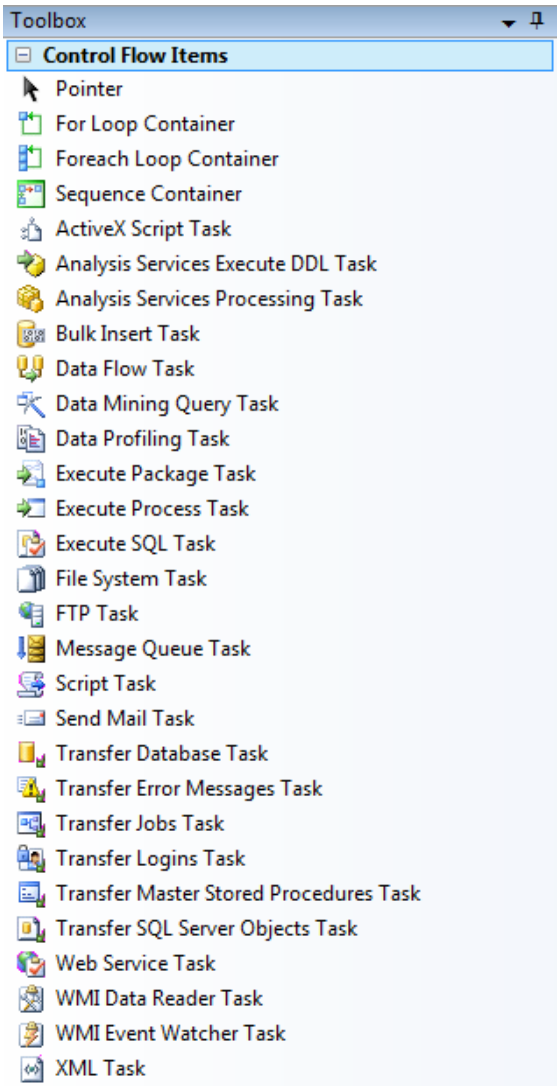
A Control Flow gyakorlatilag a betöltő csomag folyamatábrája, a Data Flow pedig maga az adat-transzformáció. Mindkettő elemei Taszkok. A taszk az SSIS csomag legkisebb önállóan

futtatható része. A taszknak van sikeres és sikertelen kimenete, ezeket zöld és piros nyilak jelölik.

A Data Flow tulajdonképpen egy speciális Control Flow Task, amely további adatmozgatással, transzformációval kapcsolatos taszkokat foglal magába.

A Control Flow további elemei a

- Containerek, melyek csoportosításra adnak lehetőséget. Több taszkot és további konténereket tartalmazhatnak.
- Precednence Constraintek: a feladatok végrehajtási sorrendjét szabályozzák az előző feladat kimenetelének, és/vagy feltételeknek megfelelően.



11. ábra. SSIS Control Flow Taskok

Amint láthatjuk a Control Flow elemek között korántsem csak adatmanipulációval kapcsolatos taszkok szerepelnek. Többek között e-mailben jelentéseket küldhetünk az egyes feladatok kimeneteléről felhasználóinknak vagy a rendszergazdának, FTP kapcsolaton küldhetünk, vagy fogadhatunk adatokat.

Külön kiemelném az Adat profilozó (Data Profiling), Csomagfuttató (Execute Package) és a Processz-futtató (Execute Process) taszkokat.

A Data Profiling Task a következő profilozó eljárásokat támogatja:

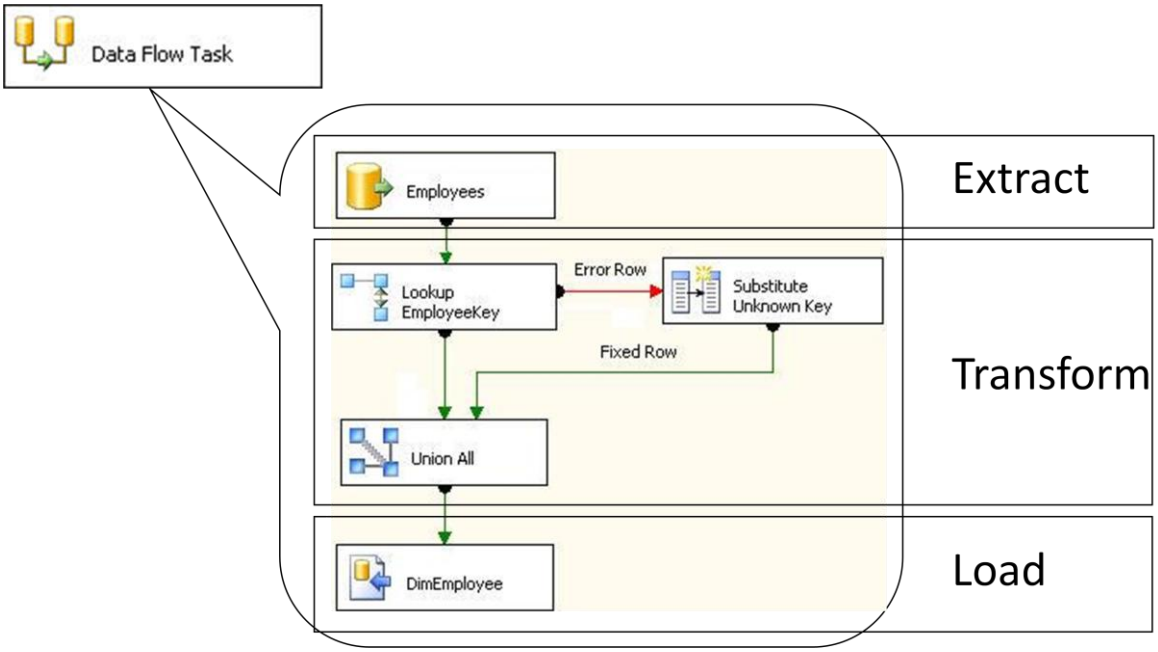
- kitöltöttség-analízis
- kulcsképesség-elemzés
- minták keresése
- oszlopstatisztikák
- értékeloszlás-analízis
- összefüggés vizsgálat (hierarchiákat kereshetünk vele a táblákon belül)
- olyan részhalmazok keresése, amelyekkel adatkapcsolatokat deríthetünk fel két tábla között

Az Execute Package segítségével csomagunkból, akár egy másik csomag futását is elindíthatjuk. Az Execute Process segítségével pedig tetszőleges processzt elindíthatunk rendszerünkön.

6.1.2 Data Flow

Egy Data Flow tasknak a következő elemei lehetnek:

- Források (Sources), ahonnan az adatokat kinyerjük (Extract)
- Transzformációk (Transformations)
- Célok (Destinations), ahová a transzformált adatokat betültjük (Load)



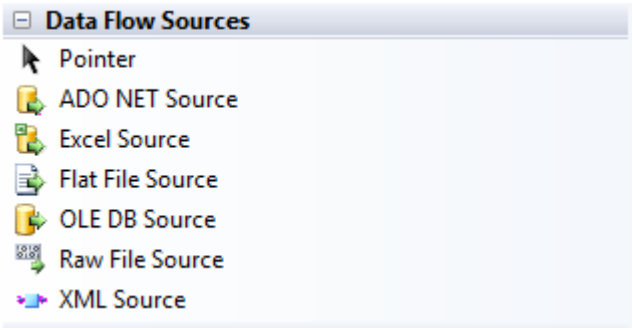
12. ábra. SSIS Data Flow Task - példa

Nézzük ezeket részletesen.

6.1.2.1 Data Flow Sources

Nincs bemenetük, csak kimenetük.

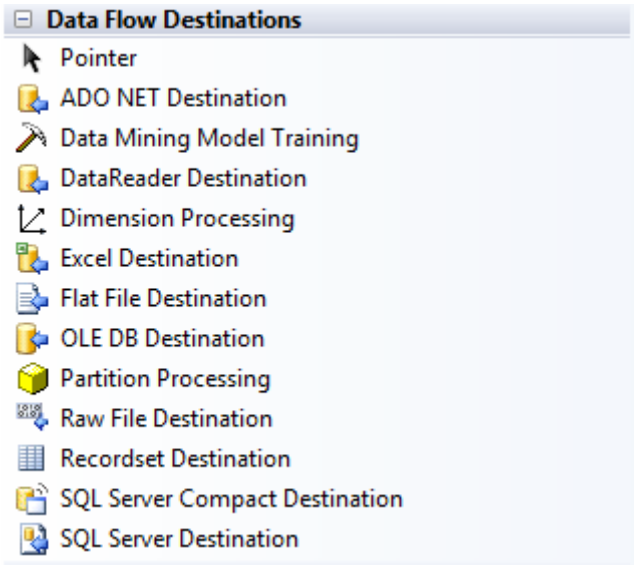
Forrás adatok kinyerésére használjuk őket: fájlokból, relációs adatbázis táblákból és nézetekből, Analysis Services adatbázisokból



13. ábra. SSIS Data Flow Sources

6.1.2.2 Data Flow Destinations

Nincs kimenetük, csak bemenetük. Adatok betöltése: relációs táblákba, Analysis Services adatbázisokba, fájlokba, DataReaderekbe vagy RecordSetekbe.



14. ábra. SSIS Data Flow Destinations

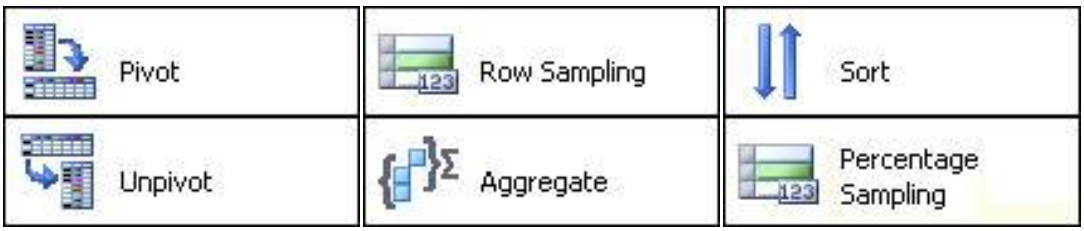
6.1.2.3 Data Flow Transformations

Az adatok átalakítására alkalmazzuk őket. A forrásokat kötik össze a célokkal. Általában egy bementtel és egy kimenettel rendelkeznek, a nyilak ezúttal az adatáramlás irányát mutatják.

- Egysoros transzformációk

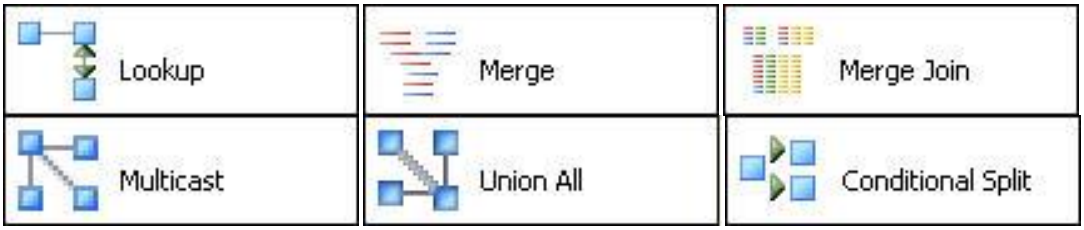
15. ábra. Az SSIS egysoros adat-transzformációi

- Többsoros transzformációk



16. ábra. Az SSIS többsoros adat-transzformációi

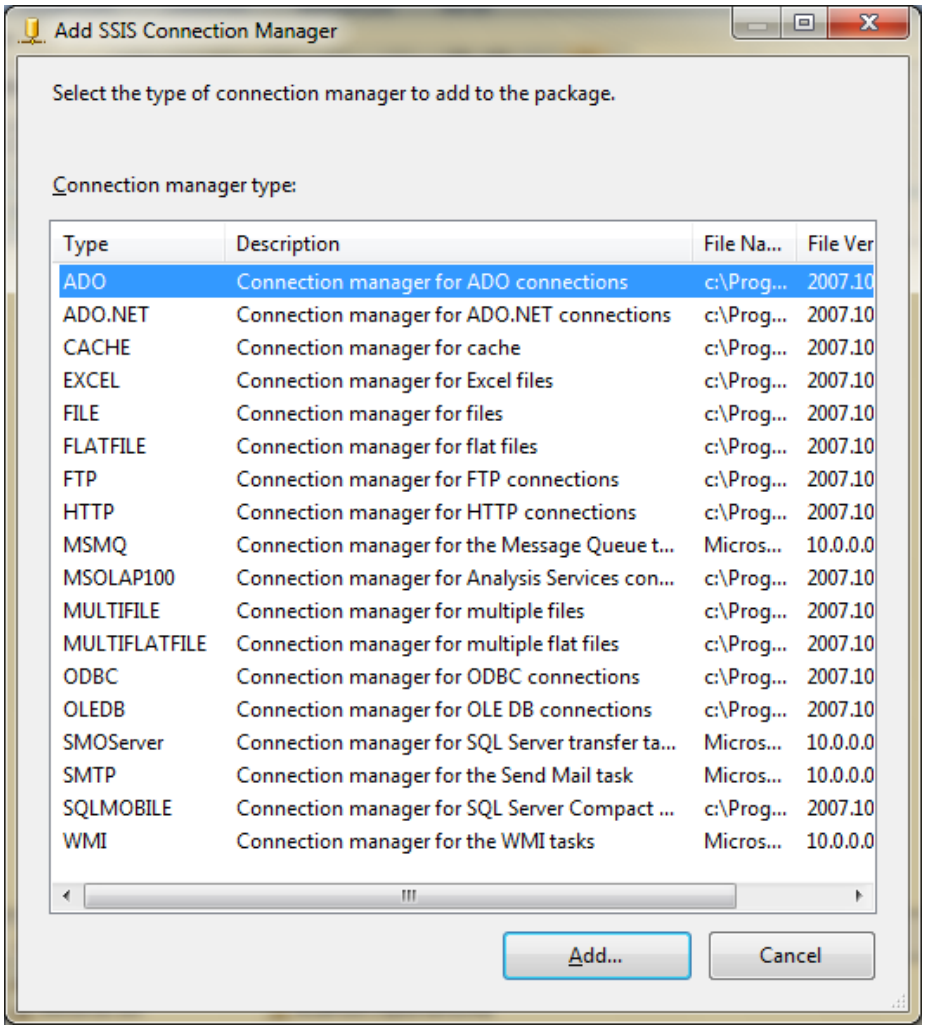
- Több bemenet vagy több kimenet



17. ábra. Az SSIS több bemenettel vagy több kimenettel rendelkező adat-transzformációi

6.1.2.4 Connection Managers

A forrásoknak és a céloknak az erőforrások használatához Connection Managerekre van szükségük. Szinte bármilyen adatforrásra mutathatunk velük. Természetesen a Business Intelligence Development Studioval együtt települő Connection Managerek mellé szükség esetén továbbiak telepíthetőek.



18. ábra. SSIS Connection Managerek

6.2 Példa SSIS használatára

Most, hogy valamennyire megismertük az SSIS-t, vegyünk egy valós életből származó példát. Következő feladattal kb. egy éve futottam össze, bár nem adattárház fejlesztés során, de akár ott is előfordulhatott volna, és jól szemlélteti az SSIS hasznosságát.

Adottak a megrendelő forrásrendszeréből rendszeresen érkező szöveges állományok. Ezek pontosvessző szeparált, meglehetősen nagyméretű CSV fájlok (köztük több is meghaladja az 1 gigabyte méretet és a négymilliós rekordszámot). Az állományok meghatározott időközönként érkeznek. Ezeknek a fájloknak a tartalmát szeretnénk saját adatbázistábláinkban elhelyezni.

Szemléltessük a problémát a Cikktörzs állományon, mely a többihez képest viszonylag kicsi (~35 MB, ~75 000 rekord), azonban több ilyenkor előkerülő problémára is jó példát ad. A fájl tartalmát a Cikktorzs táblában szeretnénk elhelyezni, úgy hogy amikor friss fájl érkezik, akkor a meglévő cikkek adatai frissüljenek, és az esetleges új cikkek bekerüljenek az adatbázisba. A fájlban nemcsak a legutóbbi importálás óta végbement változások érkeznek és olyan oszlopok is, melyek számunkra irrelevánsak, tehát akár el is hagyhatók (a példa leegyszerűsítése végett ezek mellett további oszlopokat hagytam el).

6.2.1 Megoldás SSIS nélkül

Első problémánk, hogy a fájl és az adatbázis között valamilyen összeköttetést kell teremtenünk, ahhoz, hogy egyáltalán adatokat tudjunk a kettő között mozgatni.

Megoldás: használjuk a fájlt ODBC adatforrásként. Ilyenkor a fájl tartalma ugyanúgy lekérdezhető mintha adatbázistábla lenne. Ehhez azonban a fájlt tartalmazó könyvtárban a schema.ini állományban definiálnunk kell az egyes oszlopok típusát. Ez nagyjából így nézhet ki:

```
[cikktorzs.csv]
ColNameHeader=False
Format=Delimited(;)
Col1=F1 Integer
Col2=F2 Char Width 255
Col3=F3 Char Width 50
```

```
Col4=F4 Date
Col5=F5 Integer
Col6=F6 Integer
Col7=F7 Char Width 255
Col8=F8 Char Width 255
…
```

Természetesen az oszlopneveket is megadhatjuk, de ettől most eltekintünk.

Ezután hajthatjuk végre a kívánt műveletet, nézzünk erre is egy példát:

```
TRUNCATE TABLE [dbo].[Cikktorzs]
GO

INSERT INTO [dbo].[Cikktorzs]
           ([cikkszam]
           ,[cikknev]
           ,[cikk_rovidnev]
           ,[utolso_mozgas_datum]
           ,[raktar_keszlet]
           ,[cikkosztaly_kod]
           ,[aktiv])

SELECT F1, F2, F3, F15, F16, F34, F35 FROM
OPENDATASOURCE
('Microsoft.ACE.OLEDB.12.0',
'Data Source=c:\Eleresiut\;Extended
Properties="Text;HDR=No;FMT=Delimited;"'
)...[cikktorzs#CSV]
GO
```

Ez a megoldás sajnos nem túl kellemes, mivel szöveges fájl elérése meglehetősen lassú és még a példában szereplő egyszerű INSERT utasítás is több percig fut nagy rekordszám esetén, és mi ennél jóval bonyolultabb műveleteket szeretnénk végezni.

Másik megoldás: BULK INSERT művelet használata. A BULK INSERT művelet tipikusan arra van kitalálva, hogy nagymennyiségű adatot mozgassunk gyorsan. Gyorsasága több tényezőből is adódik. Ezek közül néhány: kikerüli az adatbázis tranzakció logját, az elvégzett BULK műveletről nem jelenik meg a logban információ; az adatok validálása minimális és ekkor is csak meglehetősen nagyszámú hiba után tekinti a rendszer sikertelennek a műveletet.

Van azonban hátránya is. Az adatbázistábla szerkezetének tükröznie kell a fájlunk szerkezetét, magyarul a tábla és a fájl oszlopainak száma meg kell hogy egyezzen és típusaiknak legalább konvertálhatónak kell lenniük.

```
BULK INSERT Cikktorzs
FROM 'c:\Eleresiut\cikktorzs.csv'
WITH
(
FIELDTERMINATOR = ';',
ROWTERMINATOR = '\n',
KEEPNULLS
)
GO
```

Tipikus hiba volt, hogy ahol üres dátumnak kellet volna szerepelnie, ott a fájlban a ’0000.00.00’ érték volt megadva, amit az SQL Server 2005 nem tudott datetime típusra konvertálni[11]. Az [aktiv] oszlop értékei az eredeti fájlban: ’Igen’ és ’Nem’, ehelyett mi inkább bit típust használnánk 1/0 (True/False) értékekkel. A raktárkészlet a következő formában volt megadva: 1,234.56, amit szintén meglehetősen nehéz konvertálni. Ezeket a hibákat valahogy kezelni kell.

Mivel eddig még csak az INSERT műveletet oldottuk meg, azt is meglehetősen sok hibával és felesleges adattal, úgy döntöttem, hogy létrehozok egy extra táblát, amelybe a közvetlen importálás történik az eredeti szöveges formában (amolyan „staging area”), és a saját táblánkba csak a szükséges adatok kerülnek végleges formájukban. Ez a legtöbb esetben az adattárházak feltöltésekor is hasonlóan történik.

Végül a feladat megoldására alkalmas script:

```
--Meglévõ elemek frissítése. Az elemeket cikkszám alapján
azonosítjuk
UPDATE Cikktorzs
      SET     cikkszam = cikkszam
      ,cikknev = forras.cikknev
```

[11] A datetime típus 1753. előtti dátumot nem tud kezelni. Ez abból adódik, hogy az angolszász területeken 1752 őszén tértek át Julianusz naptárról a javított Gergely-naptárra. Az SQL Server 2008 datetime2 típusa már képes az ez előtti dátumokat kezelni (0001.01.01 - 9999.12.31), azonban a nullás dátumra ez is hibát dob

```
      , cikk_rovidnev = forras.rovidnev
      , cikkosztaly_kod = forras.cikko_kod
      , raktar_keszlet = forras.raktar_keszlet
      , utolso_mozgas_datum = forras.utolso_mozgas_datuma
      , aktiv = forras.aktv
FROM Cikktorzs AS cel,
(SELECT
      [cikkszam]
      ,[cikknev]
      ,[rovidnev]
      ,CAST(cikkosztaly_kod AS INT) AS cikko_kod
      ,CAST(CONVERT(MONEY, [raktari_osszkeeszlet], 1) AS FLOAT) AS
raktar_keszlet
      ,CASE WHEN [utolso_mozgas_datuma] = '0000.00.00' THEN NULL ELSE
CONVERT(DATETIME, [utolso_mozgas_datuma], 102) END AS
utolso_mozgas_datuma
      ,CASE WHEN [aktiv]='Igen' THEN 1 ELSE 0 END AS aktv
FROM IMPORT_Cikktorzs) AS forras
WHERE cel.cikkszam = forras.cikkszam

GO
--Új cikkek felvétele (a cikkszám még nem szerepel a Cikktorzs
táblában)
INSERT INTO Cikktorzs
            ([cikkszam]
           ,[cikknev]
           ,[cikk_rovidnev]
           ,[utolso_mozgas_datum]
           ,[raktar_keszlet]
           ,[cikkosztaly_kod]
           ,[aktiv])
SELECT
          [cikkszam]
           ,[cikknev]
           ,[rovidnev]
           ,CAST(cikkosztaly_kod AS INT) AS cikko_kod
           ,CAST(CONVERT(MONEY, [raktari_osszkeeszlet], 1) AS FLOAT)
AS raktar_keszlet
           ,CASE WHEN [utolso_mozgas_datuma] = '0000.00.00' THEN
NULL ELSE CONVERT(DATETIME, [utolso_mozgas_datuma], 102) END AS
utolso_mozgas_datuma
           ,CASE WHEN [aktiv]='Igen' THEN 1 ELSE 0 END AS aktv
FROM IMPORT_Cikktorzs
WHERE cikkszam not in (SELECT cikkszam FROM Cikktorzs )

GO
```

Amint láthatjuk még egy ilyen egyszerű feladat végrehajtása is igen sok vesződséggel jár. Egy adattárház projekt alatt ennél jóval több forrásból érkeznek adataink, és ezeknél jóval összetettebb műveleteket kell végrehajtanunk. Az ilyen feladatokat könnyíti meg az SSIS.

6.2.2 Megoldás SSIS segítségével

Most lássuk az előző feladat megoldását SSIS segítségével. Egészítsük ki még azzal, hogy miután a fájlt betöltöttük, archiváljuk az egy távoli FTP Szerverre. Ezt a funkcionalitást biztosan nem tudjuk elérni Transact-SQL scripttel.

Az egyes lépések menetét itt részletezem, a hozzájuk tartozó képeket az I. melléklet tartalmazza.

Első lépésként meg kell nyitnunk az SLQ Server Business Intelligence Development Studiot (BIDS) és létrehozni egy új Integration Services Projectet

A Control Flow oldalra navigálva két dologra lesz szükségünk egy Data Flow Taskra és egy FTP Taskra. A Data Flow Task kimenetét (zöld nyíl) egyszerű Drag and Droppal összeköthetjük az FTP taskkal.

A Data Flow Taskon kettőt kattintva, vagy a Data Flow fülre kattintva szerkeszthetjük azt.

A Data Flow nagyon egyszerűen fog kinézni:

- Flat File Source – az adatok kinyerésére a szöveges fájlból
- Derived Column Transformation az utolso_mozgas és az aktiv mezők átírására
- Lookup Task: ha a Cikkszám megtalálható Frissítés, ha nem, Beszúrás
- OLE DB Destination az adatbázistáblába beszúrásoz
- OLE DB Command a meglévő cikktörzs rekordok frissítésére
- Esetlegesen beiktathatunk további Data Conversion Taskokat[12]

Ezeken túl szükség van még egy

[12] a szöveges dátum, aktív és raktárkészlet oszlopok megfelelő típusra konvertálására, de egy pár szélsőséges esettől eltekintve ez automatikusan is megtörténik

- Flat File Source Connection Managerre
- OLE DB Connection Mangerre
- FTP Connection Mangerre (a fájl FTP szerverre archiválásához)

amelyek a megfelelő kapcsolatokat biztosítják.

Ezek részletes leírása meghaladja ezen munka kereteit, a részleteket az I. melléklet képei és a CD-n mellékelt Solution tartalmazzák. Azt azonban mindenképpen meg kell említenünk, hogy használatuk igencsak egyszerű, könnyen tanulható, könnyű ráérezni. Egy informatikában valamilyen szinten jártas ember mindenféle oktatóanyag nélkül pár óra próbálgatás után képes az alapvető dolgok használatára. A fenti példa kis gyakorlattal kb. 5 perc alatt elkészíthető, szemben az első megoldással, ahol rengeteg sort kellett kézzel írnunk, felesleges táblákat létrehoznunk és valamilyen egyéb megoldást találni az adatok távoli archiválására. A példa egyébként az én meglehetősen túlterhelt és elavult laptopomon is kevesebb, mint 4 másodperc alatt lefutott.

7 SSAS – SQL Server Analysis Services

Az SLQ Server Analysis Services az SQL Server programcsomag szerves része, bár önállóan is használható, külön mégsem vásárolható meg. Fejlesztőeszköze a BIDS (Business Intelligence Development Studio).

Ha röviden akarunk az SSAS-ről beszélni, akkor azt mondjuk, hogy az SSAS a Microsoft OLAP adatbázis-kezelője, azonban az Analysis Services technológiák egész hadát foglalja magába. Sajnos ezeknek részletes ismertetésére ezen munka keretein belül nincs lehetőség, de nézzünk egy nagyon rövid áttekintést:

Az Analysis Services alapvetően két nagy részre osztható:

- Adatbányászat (Analysis Services – Data Mining)
- Többdimenziós Adatkezelés (Analysis Services – Multidimensional Data)

7.1 Analysis Services Adatbányászat

Az Analysis Services olyan eszközöket és szolgáltatásokat is tartalmaz, amelyekkel komplex adatbányászati megoldásokat hozhatunk létre. Nézzük mik ezek:

- Egy sor ipari szabvány adatbányász algoritmus.
- A Data Mining Designer, aminek segítségével gyorsan hozhatunk létre, kezelhetünk és tárhatunk fel adatbányászati modelleket, amelyeket ezután felhasználva előrejelzéseket készíthetünk.
- A DMX (Data Mining Extensions) nyelvet, melyet a bányászati modellek kezelésére és komplex előrejelző lekérdezések létrehozására használhatunk.

Ezen eszközök, szolgáltatások, és ezek kombinációnak segítségével felfedezhetjük az adatainkban létező trendeket és mintákat, amelyek aztán nagy segítséget nyújtanak a bonyolult üzleti döntések meghozatalában [17].

7.2 Analysis Services Többdimenziós Adatkezelés

Mivel az Analysis Serveces egy teljes értékű OLAP adatbázis-kezelő, ide tartozik nagyjából minden, amiről eddig az adattárházakkal és az OLAP-pal kapcsolatban beszéltünk.

Röviden foglaljuk össze mit tehetünk az Analysis Services és a BIDS segítségével:

- Dimenziókat definiálhatunk és hozhatunk létre
- A dimenziókon hierarchiákat definiálhatunk és hozhatunk létre
- Az attribútumok között függőségeket adhatunk meg
- A dimenziókat tallózhatjuk az egyes attribútumok és a hierarchiák alapján
- Többdimenziós kockákat definiálhatunk és hozhatunk létre
- A kockákon értelmezett számításokat adhatunk meg
- KPI[13]-ket definiálhatunk
- Partícionálhatunk
- Aggregációkat tervezhetünk, ezzel optimalizálva a lekérdezéseseink teljesítményét

[13] Key Performance Indicator – Fő Teljesítménymutató

- Perspektívákat készíthetünk az egyes kockákon
- Fordításokat adhatunk meg
- Mindezt egy fejlett megjelenítő eszközzel tallózhatjuk, tekinthetjük meg a kockák adatait.

Az Analysis Services lekérdező nyelve az MDX (Multidimensional Expression).

Az Analysis Services 2005 nagy újítása volt az UDM (Unified Data Model – Egységes adat-modell) bevezetése, melynek segítségével ugyanúgy tudjuk kezelni és lekérdezni a relációs csillag-sémában és az OLAP kockákban tárolt adatainkat. Sőt, ahogy említettük partícionálhatunk is, azaz meghatározhatjuk, hogy adataink milyen formában tárolódjanak:

- ROLAP – Relációs (Realtional) OLAP: minden adat és az aggregátumok is relációs formában tárolódnak
- MOLAP – Multidimenzionális (Multidimensional) OLAP: minden adat és az aggregátumok is multidimenzionális formátumban tárolódnak
- HOLAP – Hibrid (Hybrid) OLAP: az előző kettő keveréke, általában adatainkat, mértékeinket relációs, az aggregátumokat pedig multidimenzionális formában tároljuk
- Egyéb köztes lehetőségek

Az Analysis Services segítségével lehetőségünk van arra is, hogy előbb az OLAP kockát tervezzük meg, aztán ehhez hozzuk létre a relációs csillagsémát, azonban nem ez a bevett szokás.

A munka elkészítése során az Analysis Services használatával elkészítettem egy OLAP kockát, melynek segítségével egy kitalált cég internetes eladásait elemezhetjük. Az ehhez tartozó Visual Studio Solutiont, amely egy Analysis Services Projectet tartalmaz elhelyeztem a munkához tartozó CD mellékleten. A készítés során készült pár szemléletesebb képernyőképet a II. melléklet tartalmazza.

8 Összefoglalás

A vállalatok működésük során igen nagy mennyiségű adatot állítanak elő. Ez az adatmennyiség az idő előrehaladtával és a technikai eszközök fejlődésével csak tovább nő. Az adattárházak segítségével összegyűjthetjük, rendszerezhetjük, összesíthetjük ezeket az adatokat, és felhasználhatjuk őket az üzleti döntések támogatására.

Mára az adattárházak magas költségeik ellenére meglehetősen elterjedté váltak a vállalatok egyre növekvő információigénye miatt. Ezzel párhuzamosan kialakultak az adattárház-építési technikák és módszertanok is és megjelentek az adattárház-építést segítő szoftver eszközök is. Példákon keresztül láthattuk, hogy milyen eszközöket biztosít mindehhez a Microsoft SQL Server, azonban mára már minden konkurens gyártó rendelkezik ilyen, vagy hasonló eszközökkel. Ugyanakkor az adattárház-építés még mindig nem teljesen automatizálható feladat, komoly megfontolásokat, hosszú tervezést és kivitelezést igényel.

Adattárházakat elsősorban a felhasználók igényeire szabva, az üzleti szükségletek kielégítésére építünk. Láthattuk azt, hogy ezeknek az igényeknek a kielégítése jellemzően más megközelítést igényel, így az adatainkat az adattárházban a megszokottól eltérő dimenzionális modellbe szervezzük, csillag sémában tároljuk, megkönnyítve ezzel az összetettebb lekérdezések végrehajtását és az OLAP kockák kialakítását.

Az adattárház-építés idejének jelentős részét teszi ki az ETL folyamatok megtervezése és implementálása, mivel az adatok általában nem olyan formában állnak rendelkezésre, ahogyan azokra szükségünk lenne. Az SQL Server Integration Services bemutatásán keresztül láthattuk, hogy rendelkezésre állnak olyan eszközök, melyek segítségével ezt a folyamatot is meglehetősen lerövidíthetjük.

8.1 Jövőkép

Bár az üzleti intelligencia, az OLAP, a multidimenzionális adattárolás az adattárház fogalmak közül egyik sem új keletű, meghatározó jelenséget mégis csak az elmúlt 10-15 évben nyertek. Ugyanakkor nem szabad elfelejtenünk azt sem, hogy ezek alatt az évek alatt a terület igencsak dinamikus fejlődésen ment keresztül, amely jelenleg folytatódni látszik. Nemrég jelent meg a DW 2.0 (Data Warehousing 2.0) fogalom, amely a következő generációs adattárházakat

definiálja, egységesíti az adattárház fogalmat és architektúrát. Már az SQL Server 2008 R2-ben is megjelentek a DW 2.0-át támogató funkciók [16].

A Cloud Computing megjelenésével az adattárházak akár a felhőbe költöztethetők, ami újabb lehetőségeket kínál a tervezés során.

Izgalmas új terület, amivel várhatóan egyre többet foglalkozunk majd, a szociális hálókból (Facebook, YouTube, stb.) kinyerhető információk feldolgozása és a rájuk épülő média igények kiszolgálása, valamint a hordozható eszközök egyre nagyobb elterjedésével a mobil üzleti intelligencia kérdése.

9 Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezető tanáromnak, Kollár Lajosnak a diplomamunka elkészítésében nyújtott segítségért.

10 Irodalomjegyzék:

[1] W. H. Inmon: Building the Data Warehouse – Second Edition

[2] Ralph Kimball, Margy Ross: The Data Warehouse Toolkit – Second Edition. John Wiley & Sons, Inc., 2002

[3] Naeem Hashmi: Businness Information Warehouse for SAP

[4] Üzleti intelligencia (Business Intelligence, BI) fogalma - http://biprojekt.hu/Uzleti-intelligencia-Business-Intelligence-BI.htm

[5] Döntéstámogató rendszer (Decision Support System, DSS) http://biprojekt.hu/Dontestamogato-rendszer.htm

[6] Vezetői információs rendszer (Management information system) http://biprojekt.hu/Vezetoi-Informacios-Rendszer.htm

[7] Microsoft SQL Server 2000 Resource Kit, Microsoft Press, 2001

[8] B. Devlin: Data Warehouse from Architecture to Implementation, Addison-Wesley, 1997.

[9] Claudia Imhoff: Where Did The Term BI Come From? http://www.b-eye-network.com/blogs/imhoff/archives/2009/03/where_did_the_t.php

[10] Steve Williams, Nancy Williams: The Profit Impact of Business Intelligence, Morgan Kaufmann, 2007.

[11] BI Topic Portal – TDWI http://tdwi.org/portals/business-intelligence-bi.aspx

[12] Dictionary of Accounting Terms: Management Information System (MIS) http://www.answers.com/topic/management-information-system

[13] Sidló Csaba: Adattárház összefoglaló: http://people.inf.elte.hu/zorro/Work/DW/adattarhazak.htm

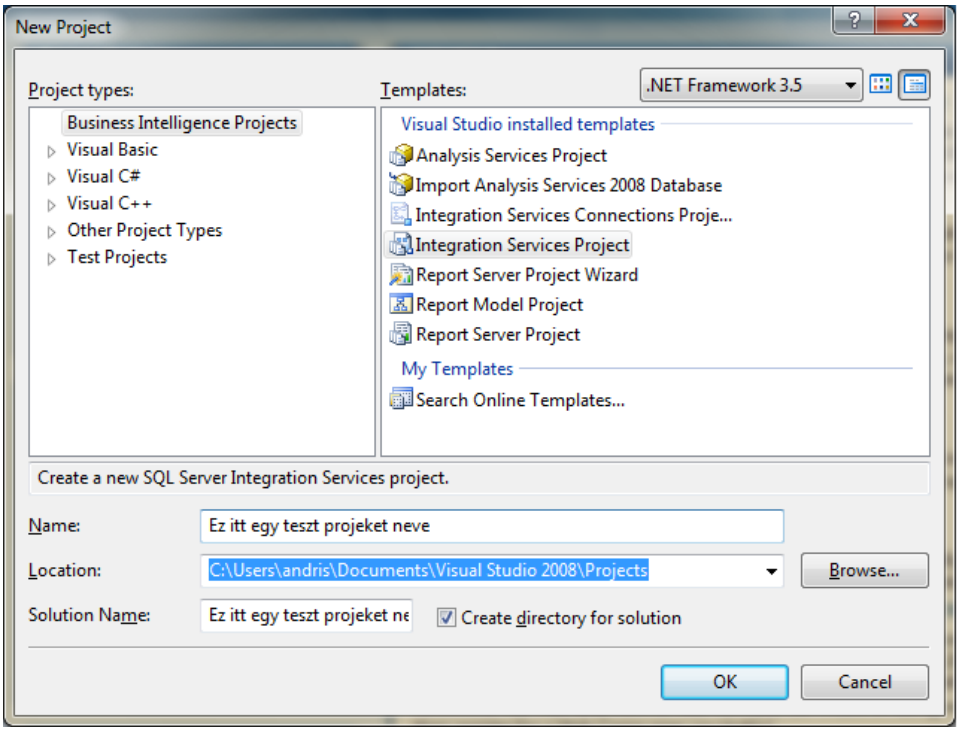
[14] Vincent Rainardi: Building a Data Warehouse with Examples in SQL Server, Apress, 2008.

[15] Adattárház-építés lépésről lépésre, Technet Magazin, 2008. szeptember – októberi szám

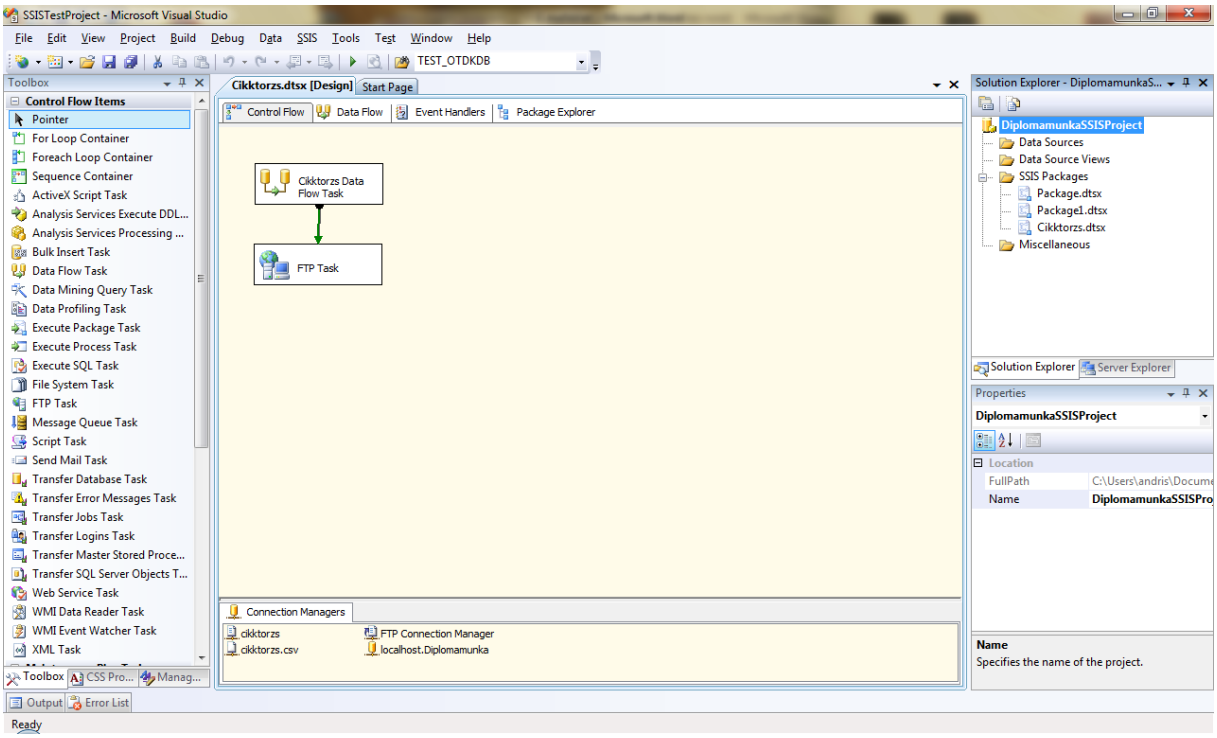
[16] W.H. Inmon: Data Warehousing 2.0 and SQL Server: Architecture and Vision, SQL Server Technical Article, 2009 október

[17] Microsoft Developer Network, http://msdn.microsoft.com/

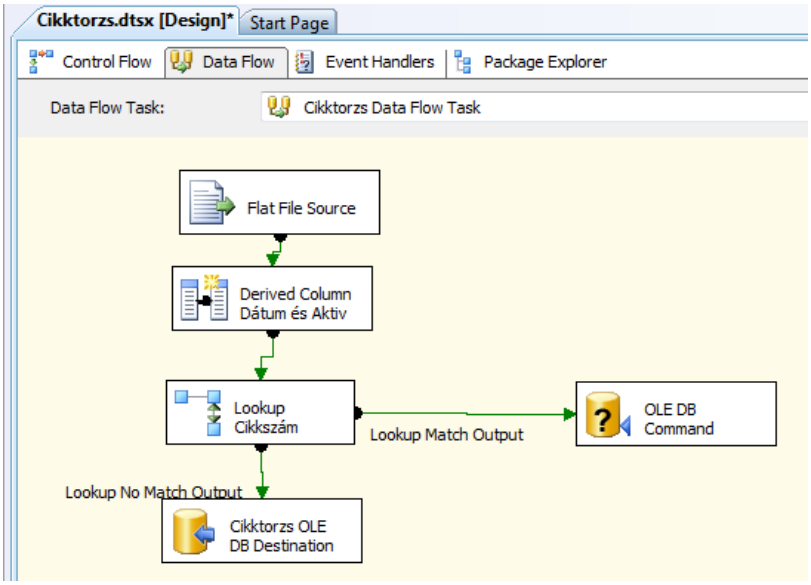
I. melléklet: Egyszerű adatbetöltés Integration Services segítségével



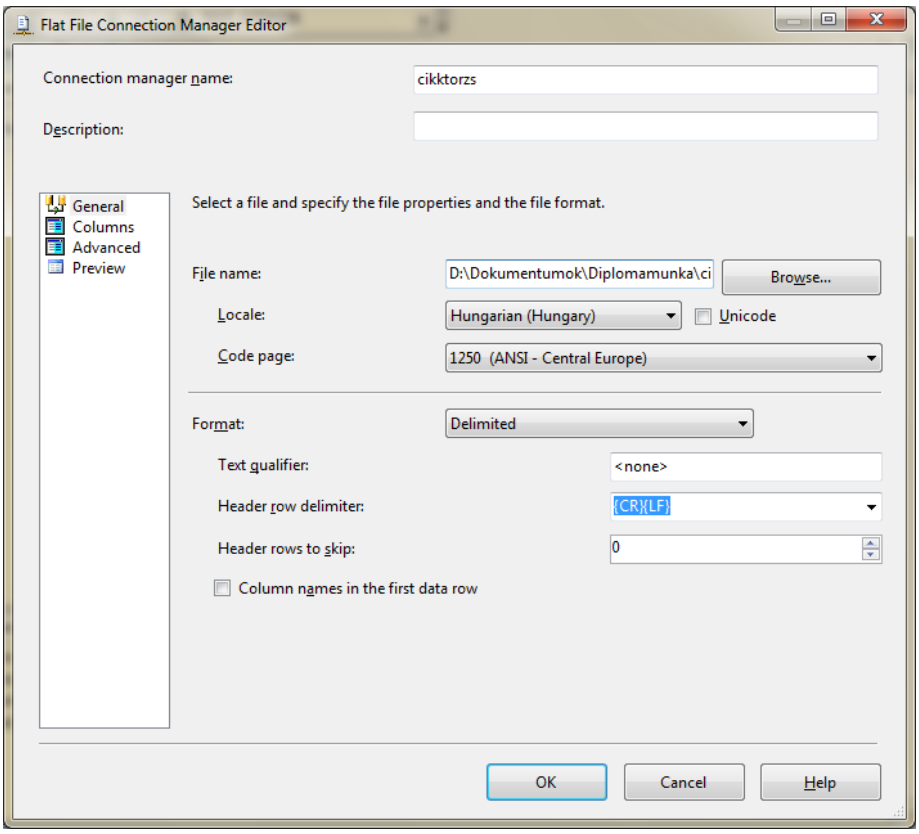
1. ábra. Új Integration Services Project létrehozása



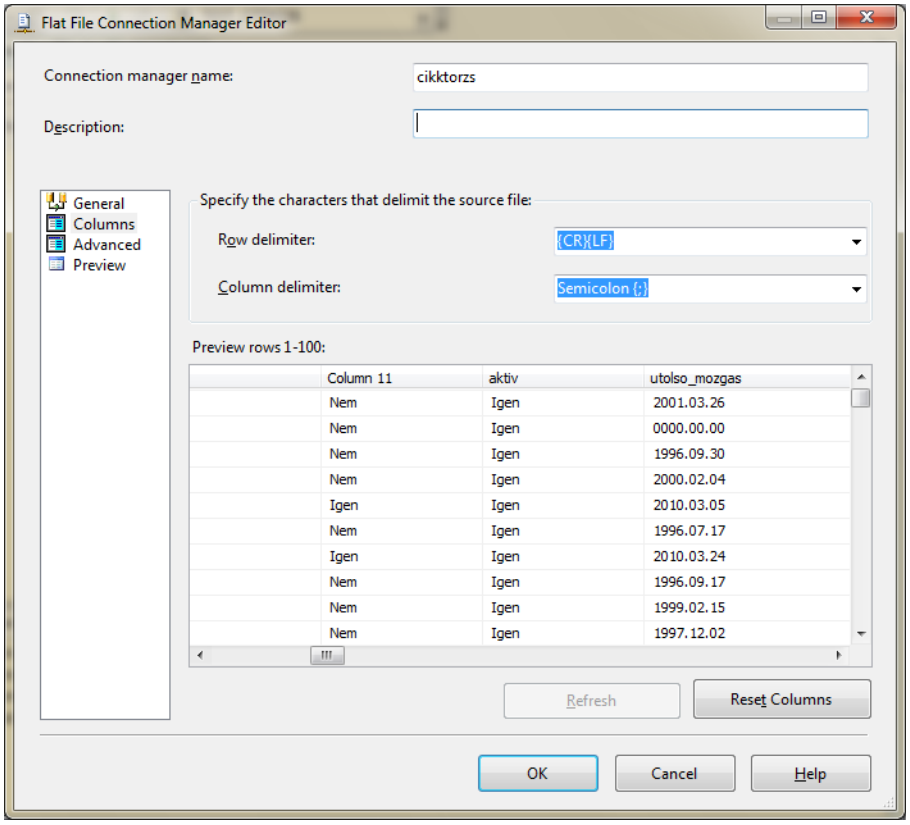
2. ábra. Control Flow a Cikktorzs csomagban



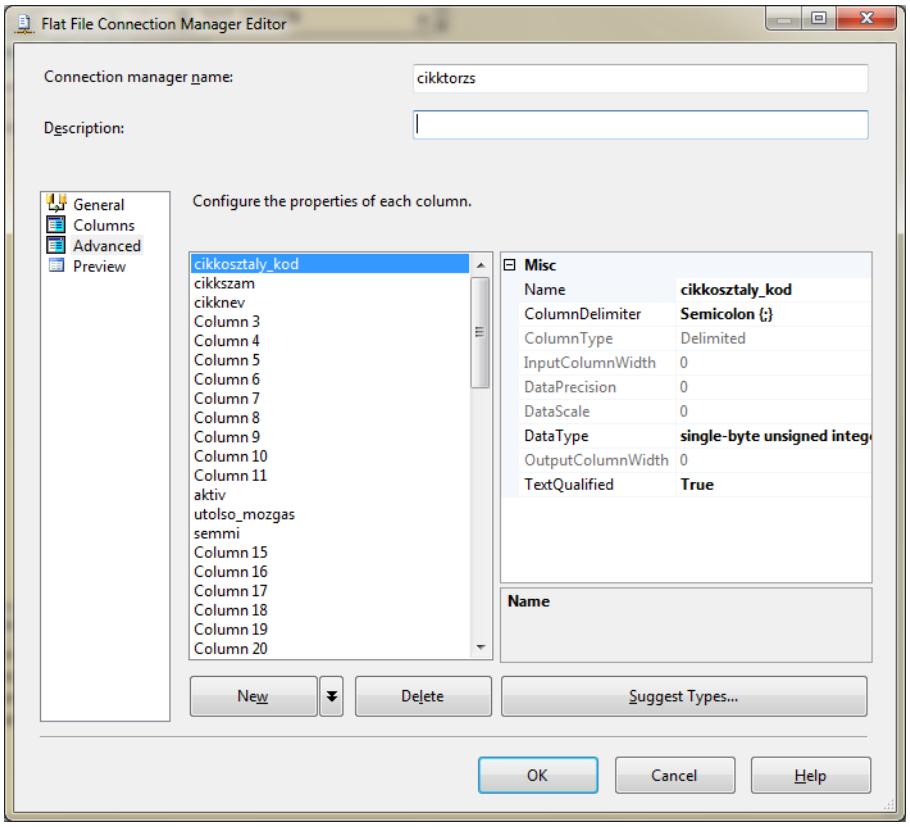
3. ábra. Cikktorzs Data Flow Task részletei



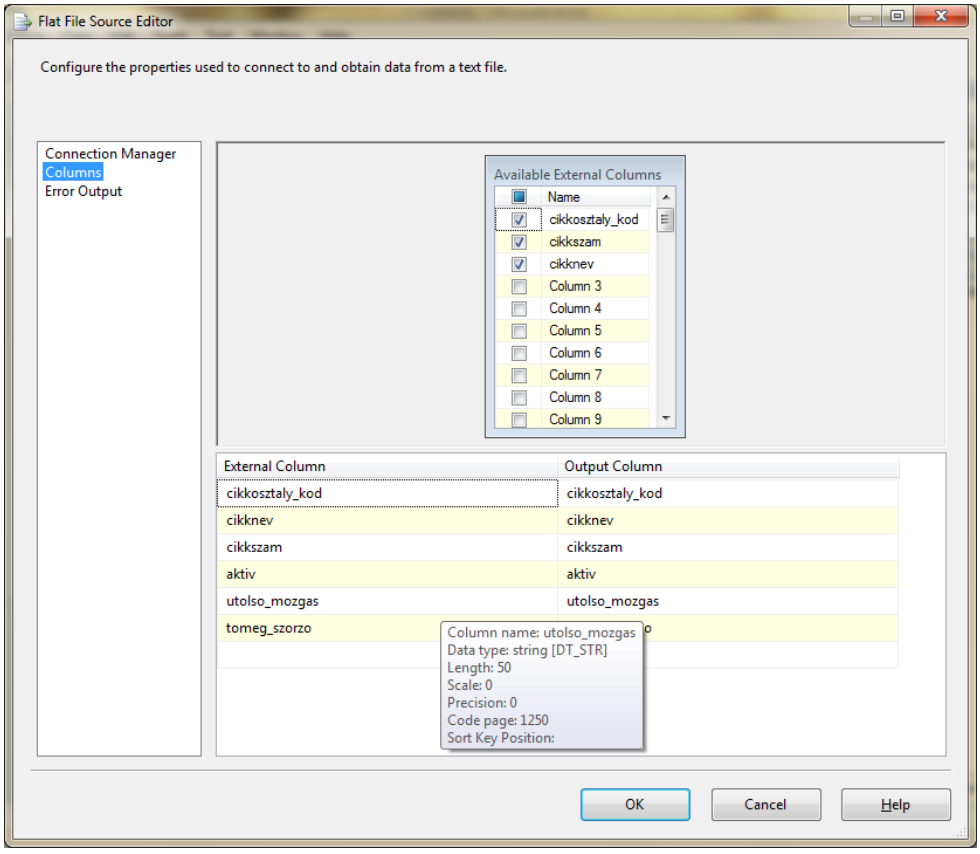
4. ábra. A Flat File Connection általános beállításai: elérési út, kódlap, oszlop fejlécek



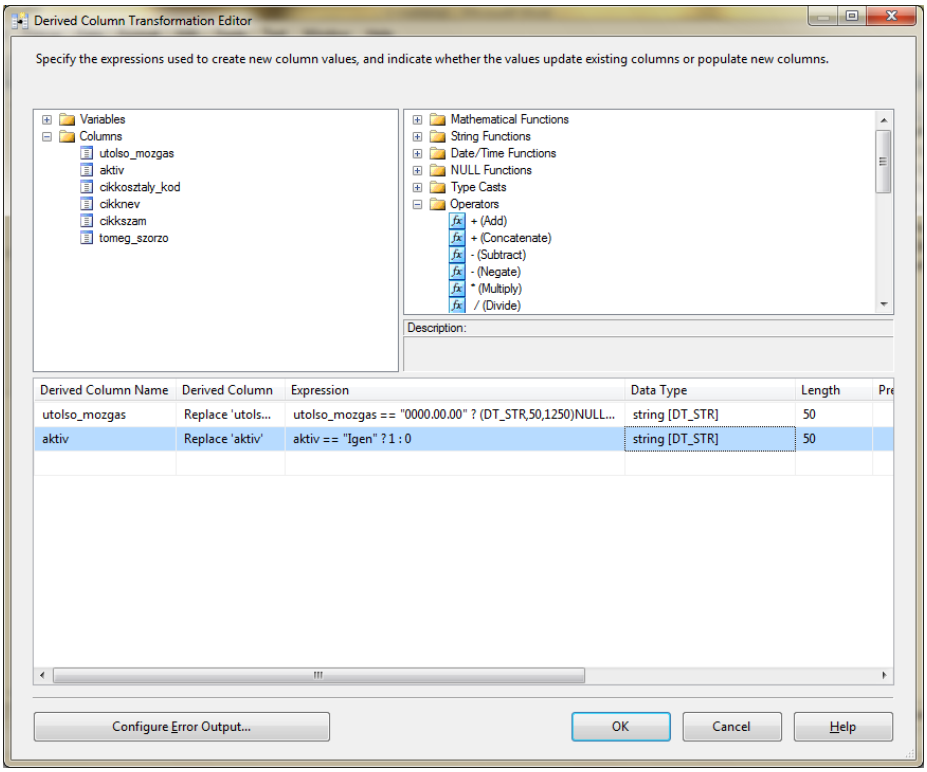
5. ábra. A Flat File Connection belállításai: elválasztó karakterek és előnézet



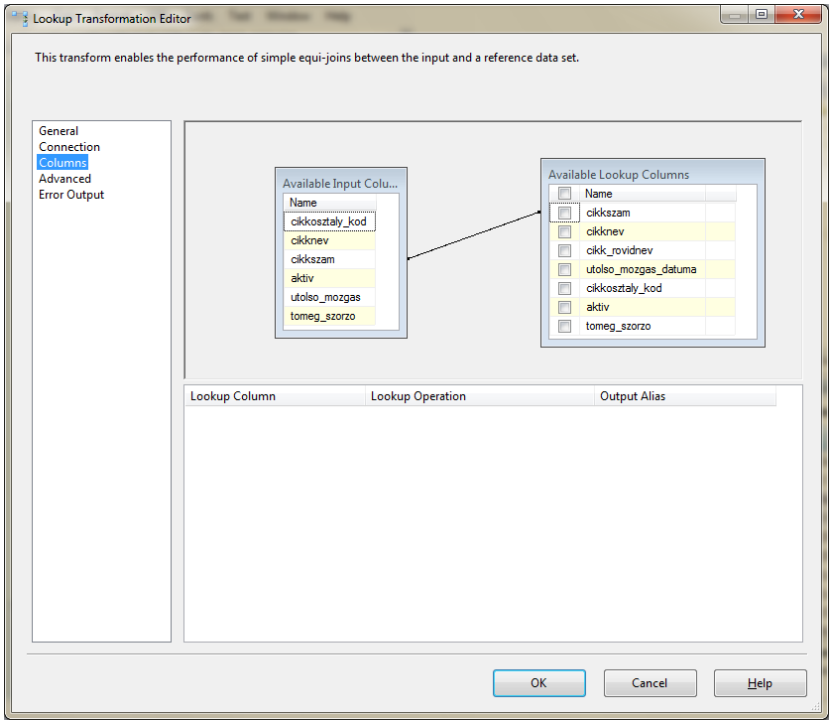
6. ábra. A Flat File Connection oszlop-tulajdonságainak részletes testreszabása



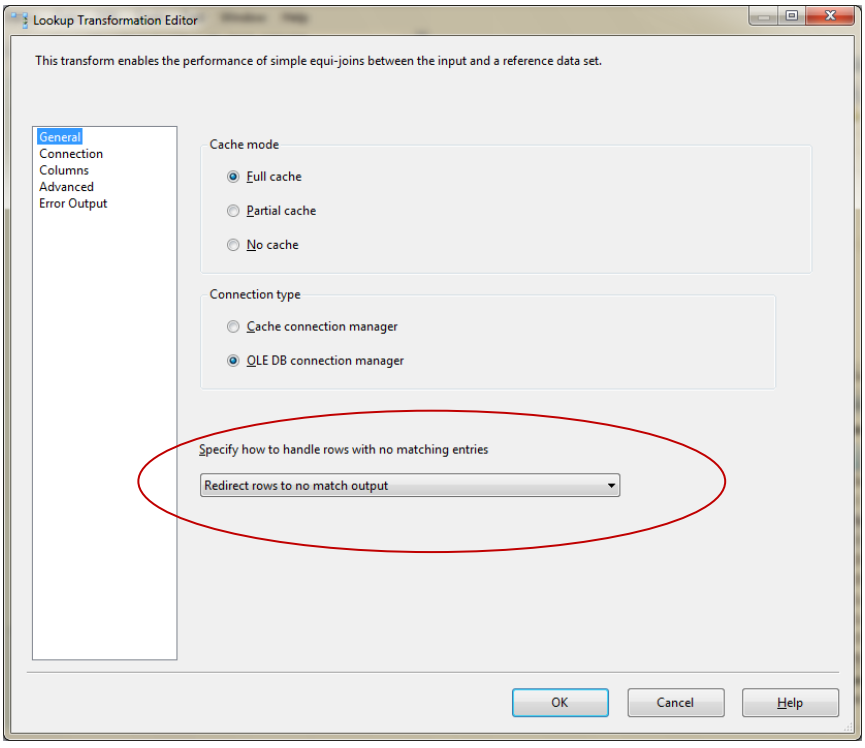
7. ábra. A Flat File Source oszlopainak kiválasztása



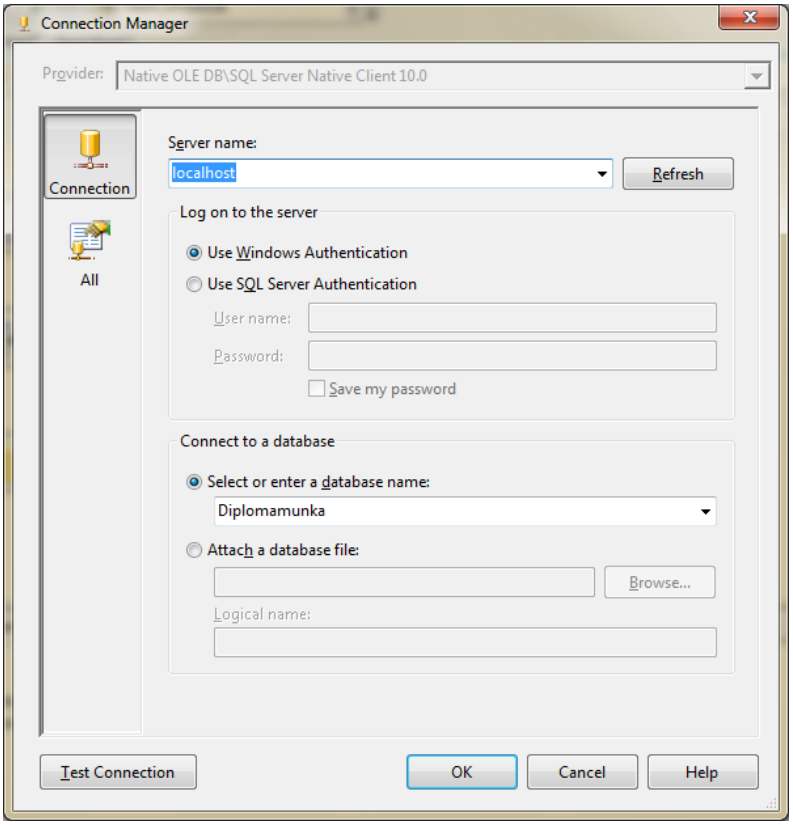
8. ábra. Derived Column Transformation egyszerű kifejezésekkel és operátorokkal



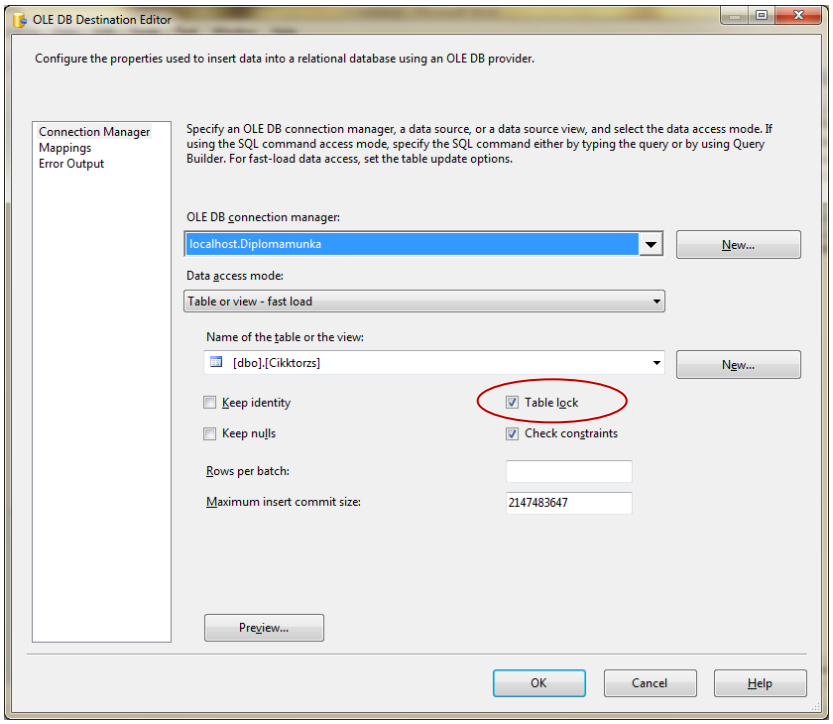
9. ábra. A Lookup Transformation egyeztetni kívánt oszlopainak kiválasztása



10. ábra. Lookup Transformation: A nem egyező sorokat másik kimenetre irányítjuk át.

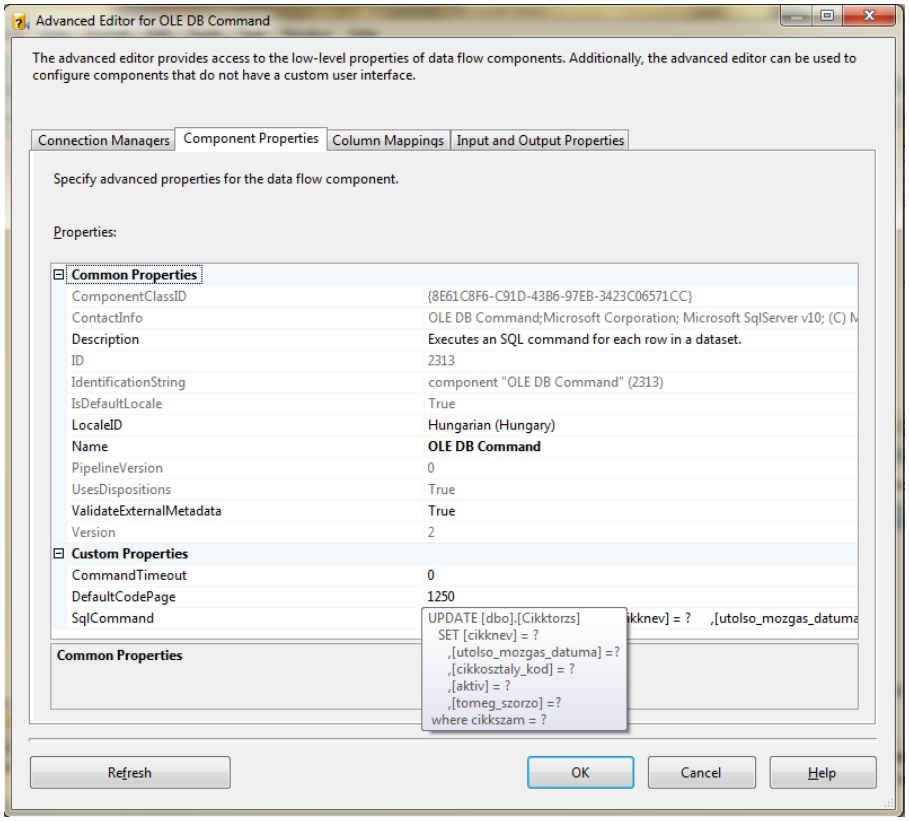


11. ábra. Ahhoz, hogy adatbázis műveleteket hajthassunk végre szükség van egy OLE DB Connection Managerre

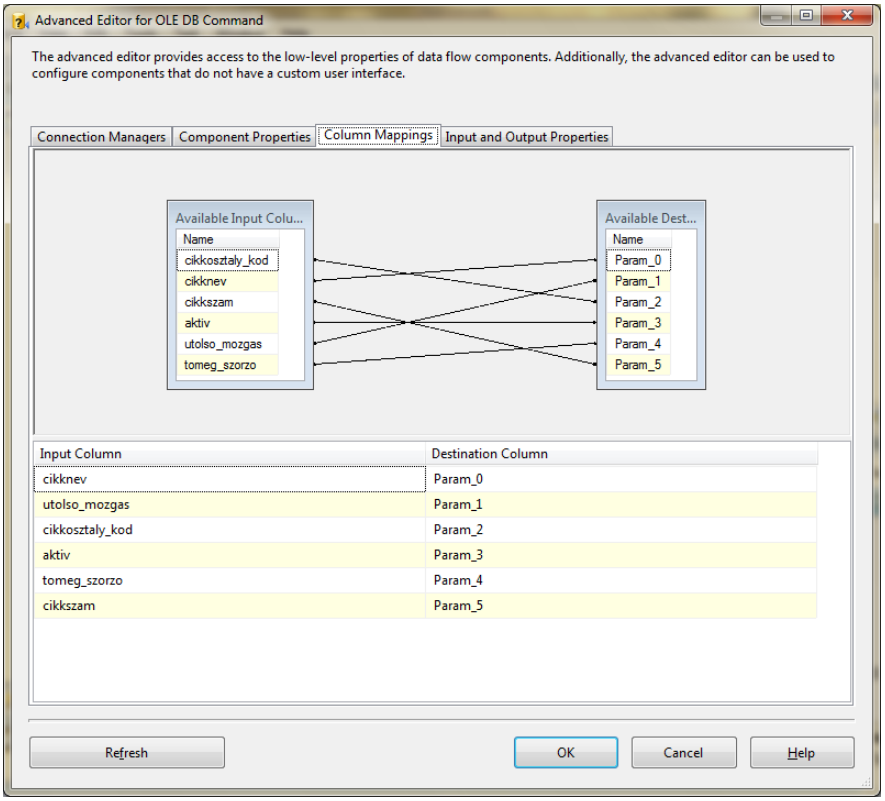


12. ábra. OLE DB Destination, melynek segítségével beszúrjuk a Lookup során nem megtalált oszlopokat.[1]

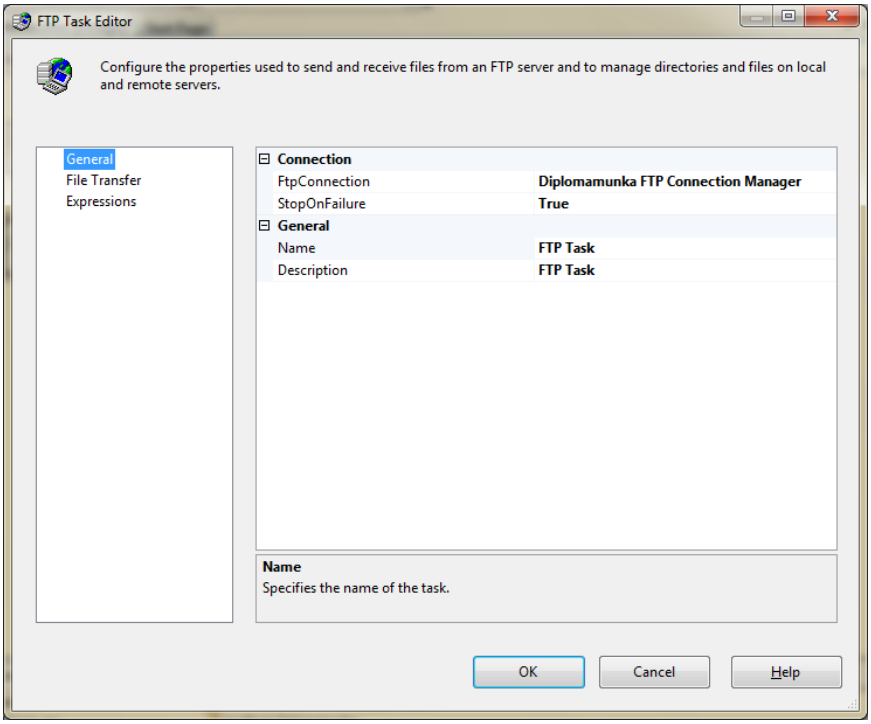
[1] A Table Lock mellől el kell távolítanunk a pipát, mivel a Lookup másik ágán az adatok frissítése párhuzamosan fog folyni



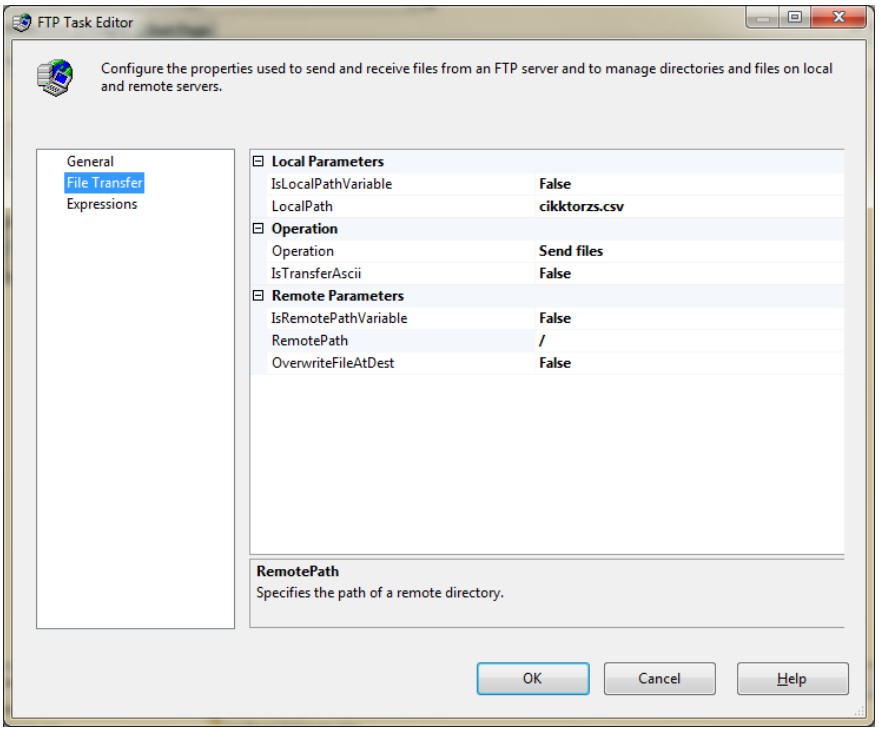
13. ábra. Az OLE DB Command Task beállításai, paraméterezett utasítás megadása



14. ábra. OLE DB Command Task paramétereinek összerendelése az oszlopokkal

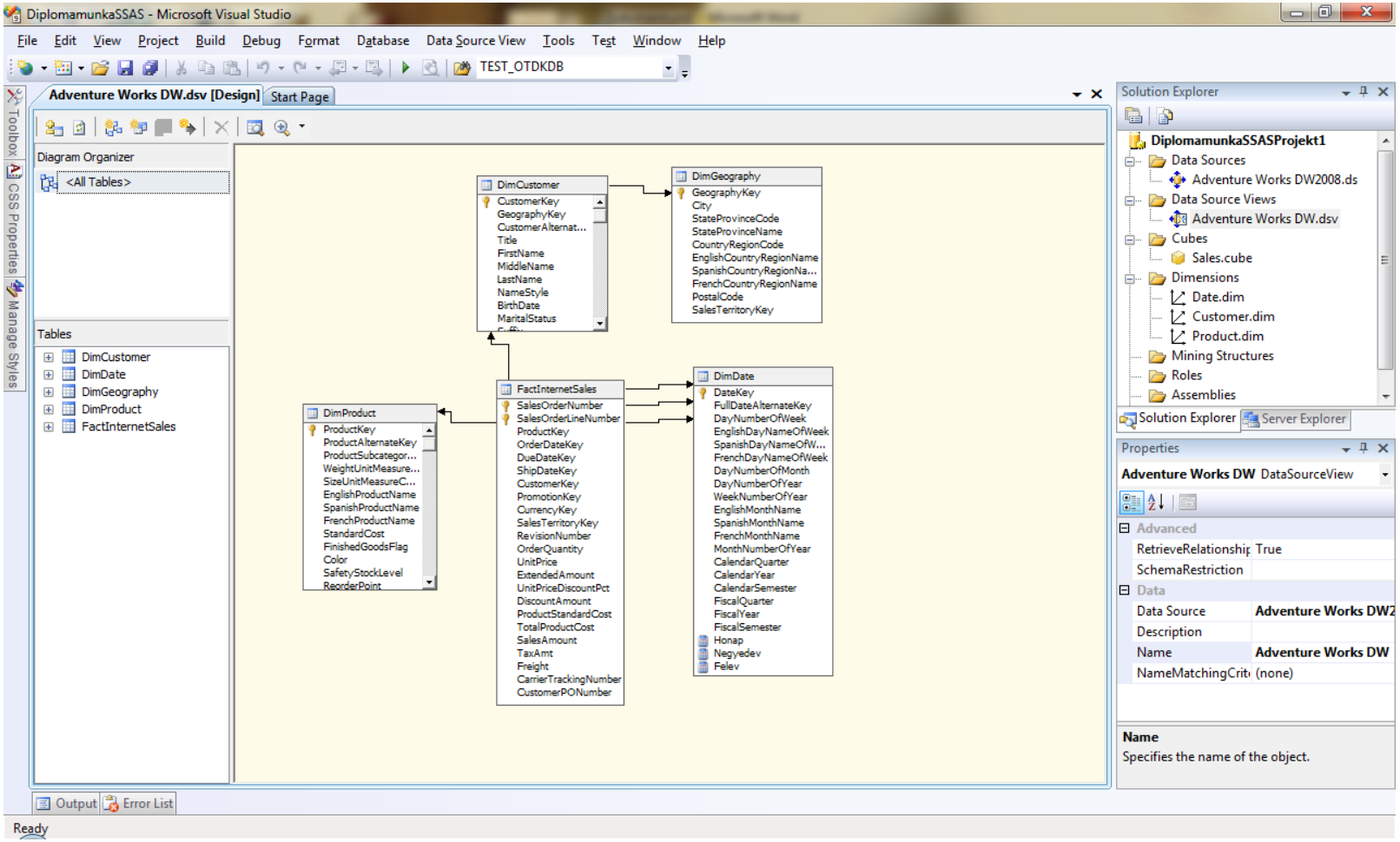


15. ábra. Az FTP task általános beállításai.

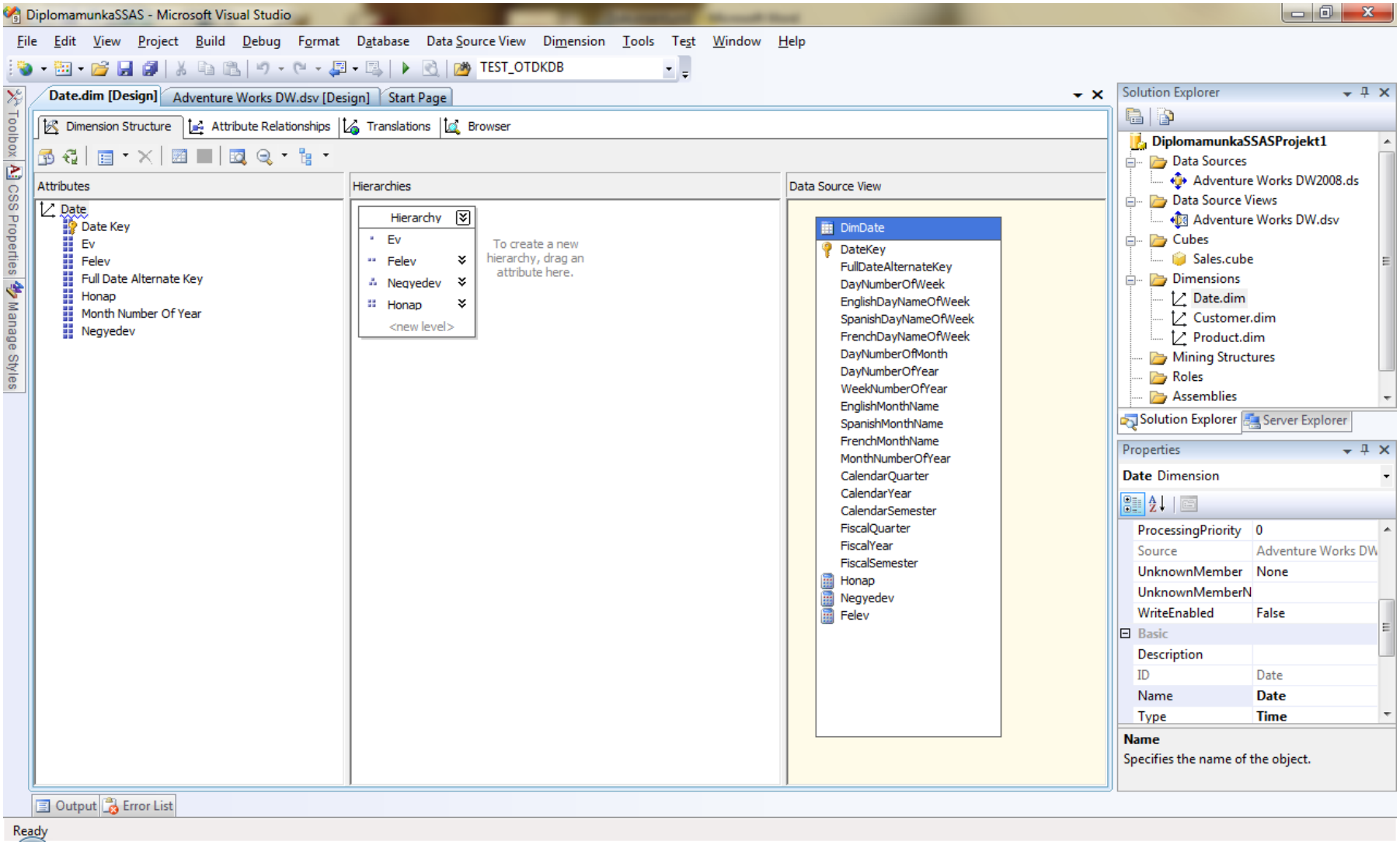


16. ábra. Az FTP Task átviteli beállításai.

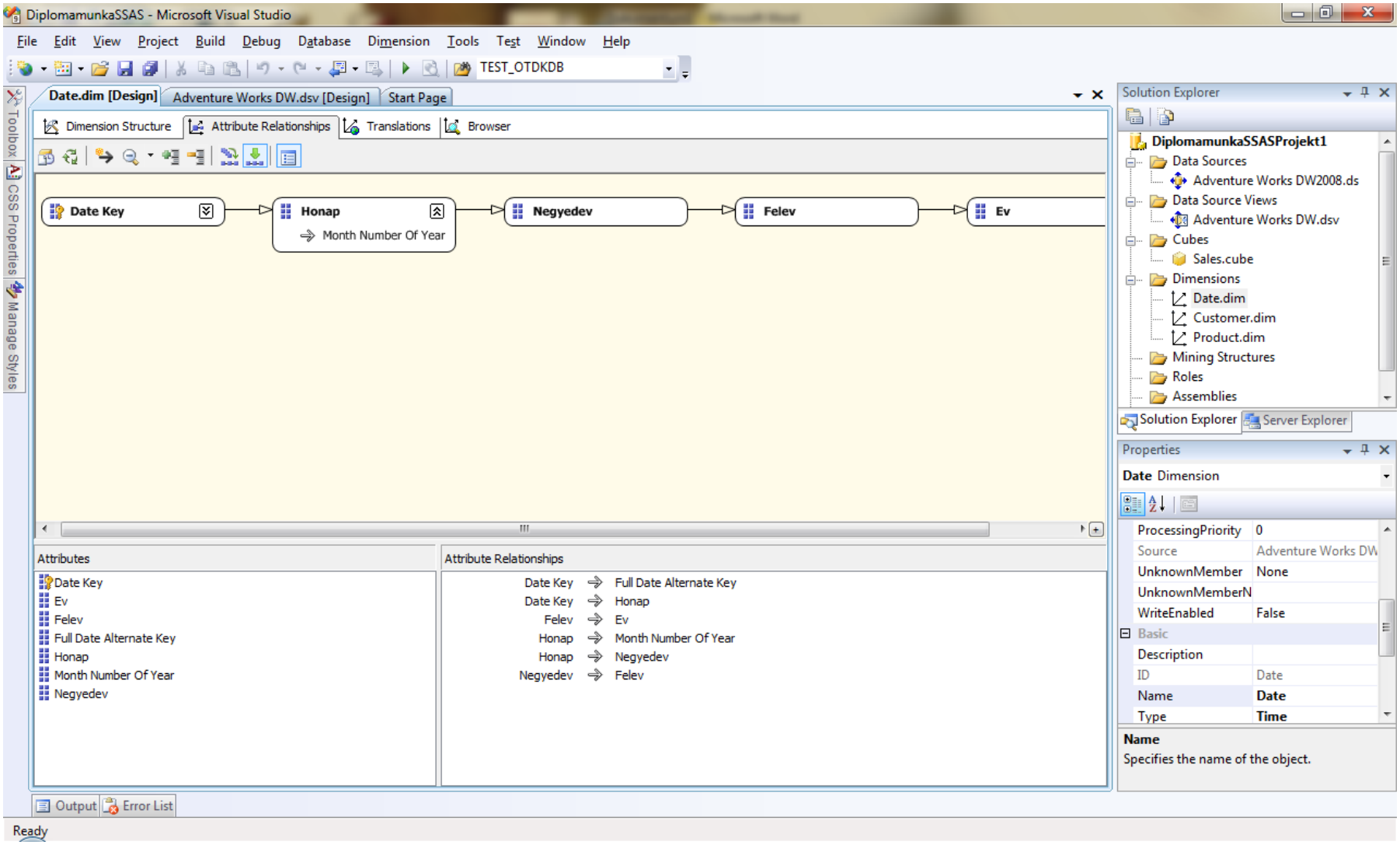
II. melléklet: Dimenziók és kocka létrehozása Analysis Services segítségével



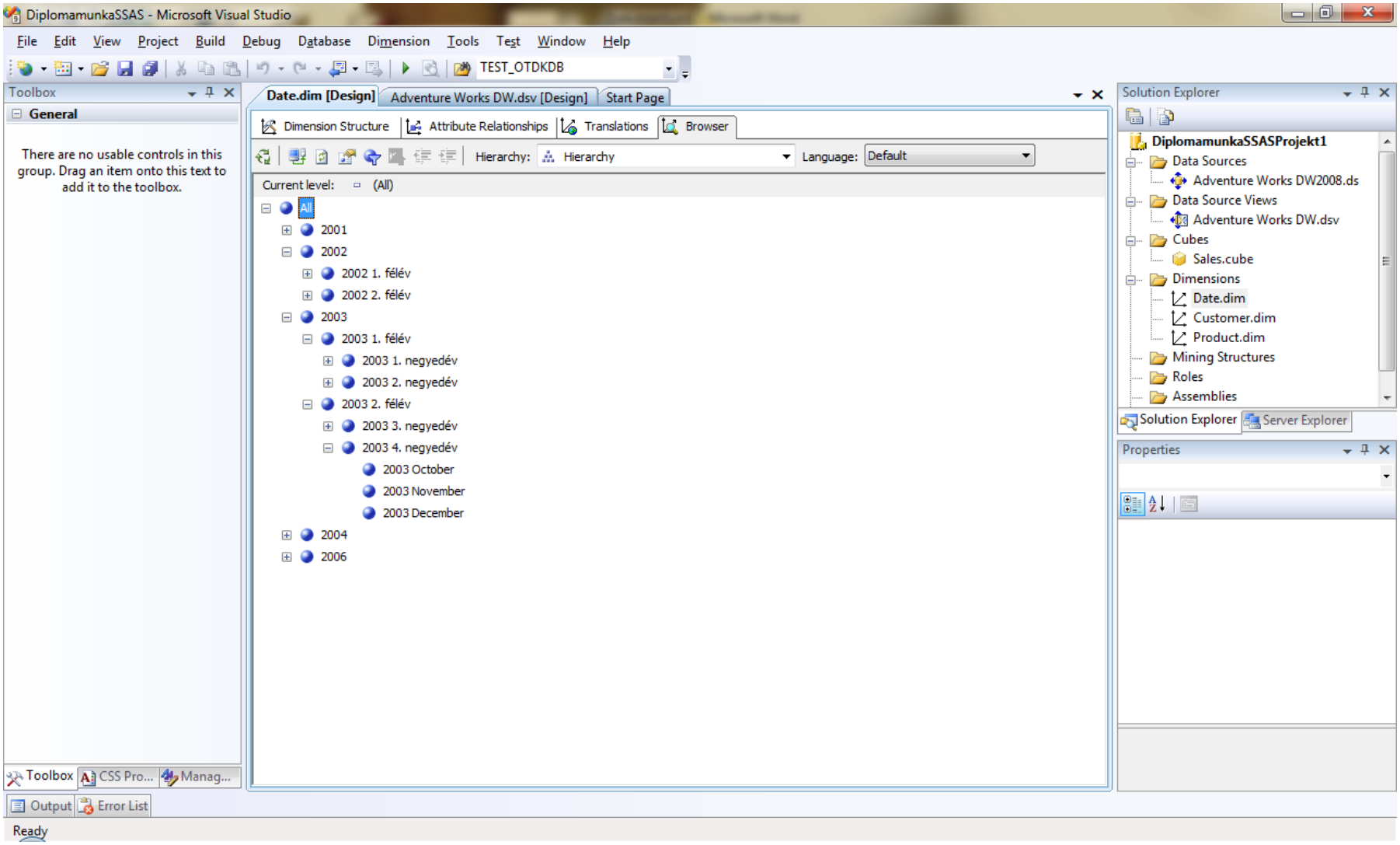
1. ábra. Egy adatforrás-nézeten keresztül megadjuk, hogy mely adatbázis táblákat használjuk fel a kocka készítésénél



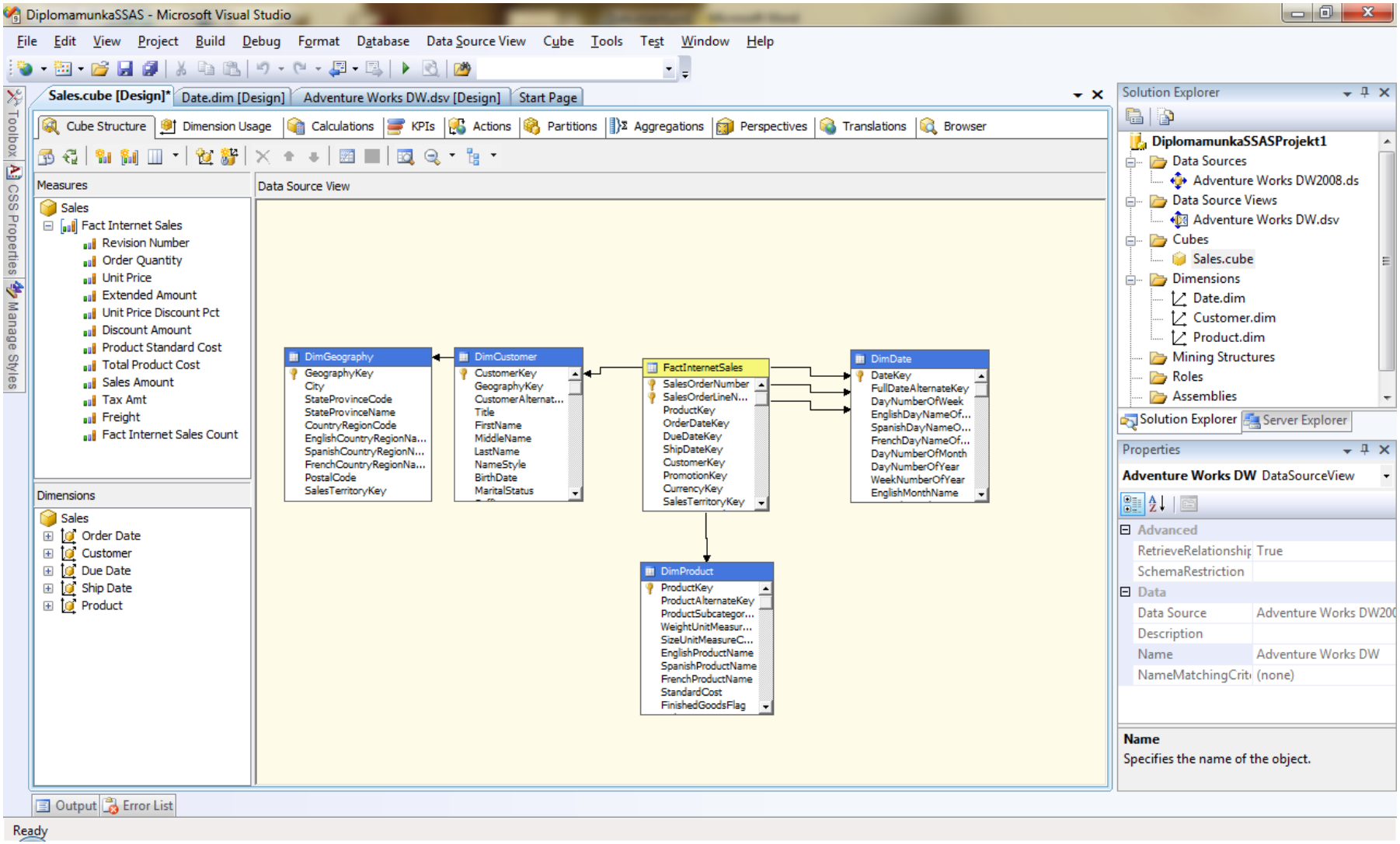
2. ábra. Dátum dimenzió létrehozása, a szükséges attribútumok kiválsztása, hierarcgiák megadása.



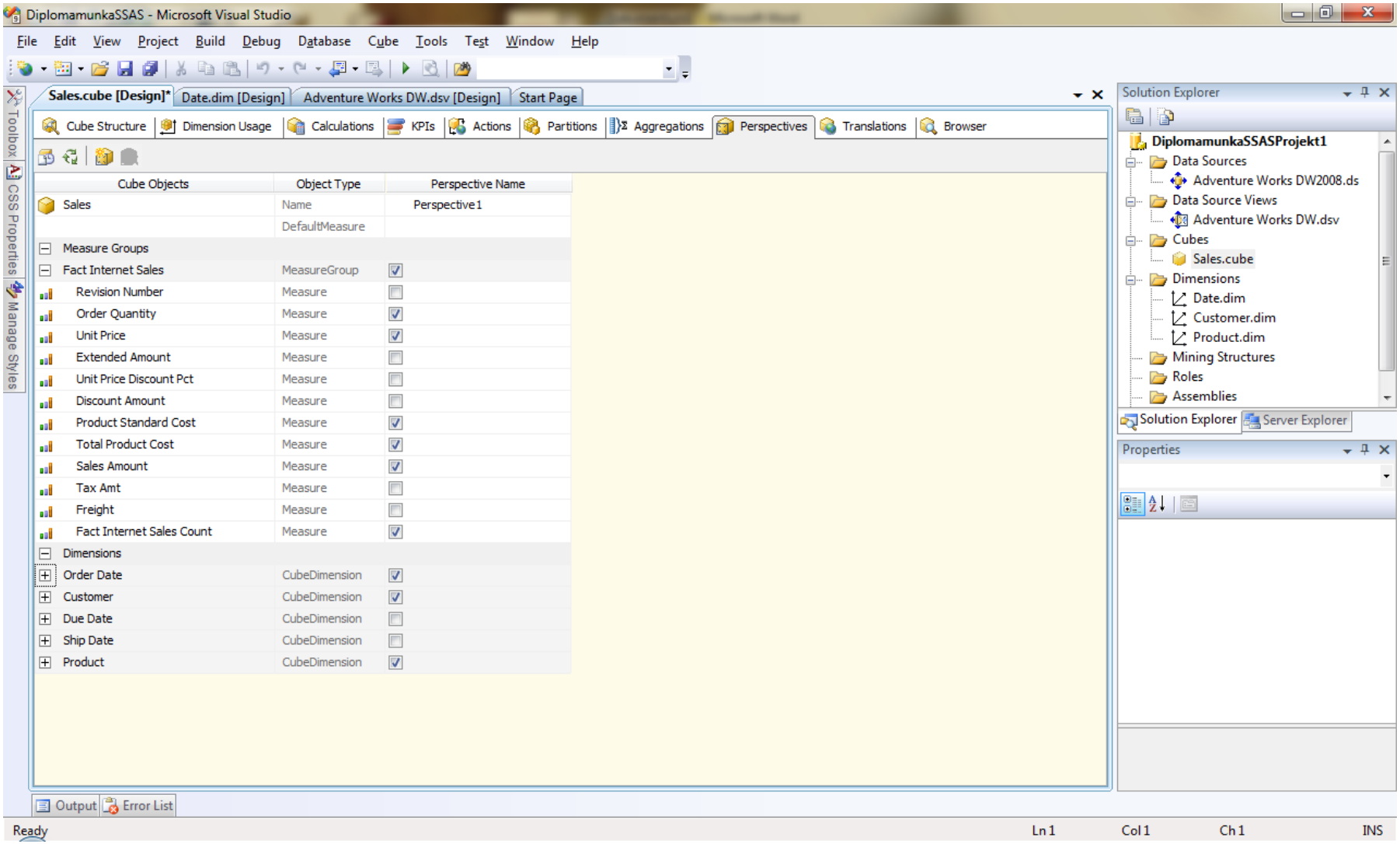
3. ábra. Attribútum-függőségek megadása.



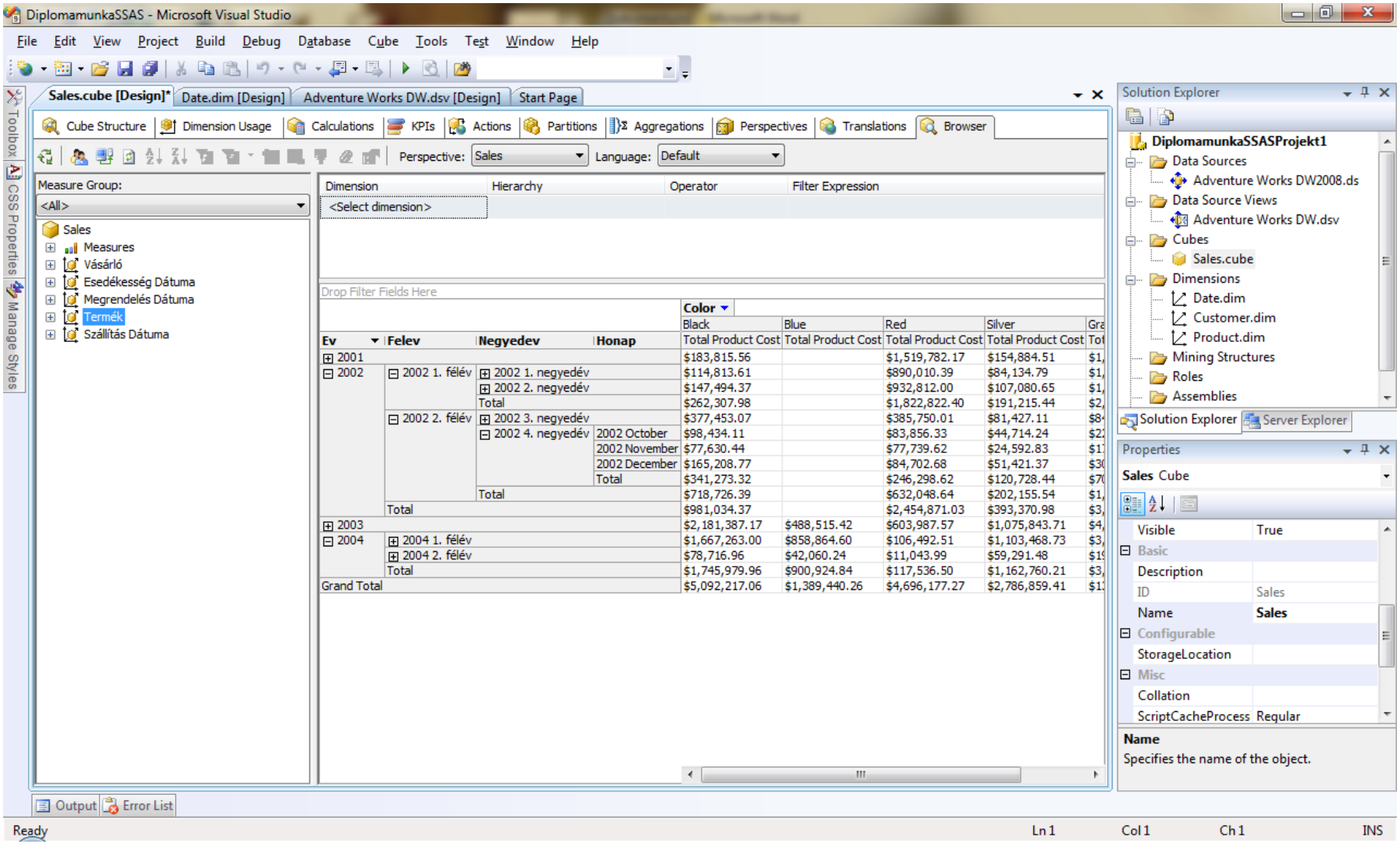
4. ábra. Az elkészült dimenzió tallózása hierarchia alapján.



5. ábra. A létrehozott kocka szerkesztőfelülete.



6. ábra. Perspektíva létrehozása a kockán.



7. ábra. A kocka adatainak tallózása. A feliratok egy része magyarul, mivel fordításokat is adhatunk meg.