Kerepes Tamás – Czinkóczki László

Adatbázisok

Az adatbázis logikai tervezése (Database Design) második rész

Database Design: miről is van szó?

- Adatbázisok: Egy informatikai rendszerben olyan az adatbázis, mint az emberi testben a memória (az emlékezet).
- Azt mondtuk, hogy a mai adatbázisok többnyire relációsak. És az információt táblákban tárolják.
- A „logikai adatbázis” arról szól, milyen relációs sémákat választunk. (Más szóval milyen táblákat és oszlopokat választunk az adatok tárolására)
- Amikor a tábláinkat és oszlopainkat választjuk ki, akkor a kiválasztott adatmodell elsősorban attól függ, a valóságnak melyik részét szeretnénk tárolni.
- Emellett a kiválaszott „adatmodell” nagyban függ attól, milyen célokat szolgál majd az adatbázis.

Az adatbázisok „használati körülményei”

- Aszerint, hogy milyen jellegű a felhasználásuk, beszélhetünk:
  - Online tranzakciókezelő rendszerekről (Online transaction Processing, OLTP)
  - Adattárházakról („Datawarehouse”) vagy döntéselőkészítő rendszerekről , üzleti elemzési rendszerekről („Business Intelligence”)
  - A kettő **sikeres** kombinációjára hatalmas üzleti igény mutatkozik, de most még hiánycikk ez. Az adatbáziskezelő rendszerek gyártói nem boldogulnak egyelőre ezzel.
  - Az Oracle tett egy kísérletet e kettő együttes használatára („Oracle Database InMemory Option”) de ez még túl új ahhoz, hogy végleges véleménye legyen róla a piacnak. Még nem terjedt el.

OLTP rendszerek

- Sok egyidejű felhasználó
- Rengeteg egyidejű kicsiny tranzakció
- Rengeteg SQL utasítás (pl. másodpercenként többezer SQL, esetenként még ennél is több)
- Az SQL-ek önmagukban leginkább egyszerűek
- A lekérdezések döntő többsége kevés sort érint
- A rendszer fő problémája: megérteni ezt a rengeteg SQL utasítást. Mert maga a végrehajtás már szinte mindig könnyű
- Az ilyen adatbáziskezelő rendszerekben a felhasználók egymás közötti koordinációja (a tranzakciókezelés és az adatok konzisztenciája) az egyik fő probléma.

OLTP rendszerek tipikus adatmodellje

- Félünk a módosítási, törlési és beszúrási anomáliáktól, és ezért igyekszünk normalizálni, amennyire csak lehet
- Az 1NF betartása magától értetődő
- A 2NF betartása szinte kötelező (ami a kulcs egy részétől függ csupán, azt külön táblába tesszük)
- A 3NF betartása is szinte kötelező (ha egy „nem-kulcs” oszlop tranzitívan függ egy kulcstól, akkor azokat az oszlopokat külön táblába tesszük)
- A Boyce-Codd normál form betartása is ajánlott (két egymást „részlegesen átfedő” kulcs kell ahhoz, hogy egyáltalán fennáljon a veszély, hogy egy 3NF tábla esetleg ne legyen BCNF-ben. Igen ritka az ilyen eset.
- Röviden: az OLTP rendszerek normalizáltak.

OLTP rendszerek és a normalizáltság

- Már a tervezés során – kiindulópontként - eleve normalizált adatszerkezetekben gondolkodunk
- A továbbiak során egyes igen speciális helyzetek miatt esetleg – ritkán - szelektíven denormalizálunk
- Igyekszünk nem tárolni kiszámítható értékeket. Ezt is egyes speciális körülmények felülírhatják
- Az adatmodell kiválasztása során nemcsak az adatmodell „elméleti tisztasága” számít, hanem a majdani használat gyakorisága
- Olyan adatmodellt választunk, amely hűen tükrözi a „világ azon darabkájáról” alkotott fogalmainkat
- Olyan táblákra van szükségünk, amelyek a gyakorlati hatékonysági elvárásoknak is eleget tesznek
- Az adatmodellező nagyon részletes elemzés után választja csak meg az adatmodellt.

Adattárházak

- Olyan adatbázisok, amelyekben a használat más jellegű. Viszonylag kis számú felhasználó (nevezik őket adatelemzőknek, adatbányászoknak is) viszonylag kis számú, de gyakran igen nehéz lekérdezést futtat.
- Nem jellemzők az egyidejű tranzakciók – nem akkora gond a konzisztencia
- Gyakran nem jellemző az adatok online változtatása
- A lekérdezések jellemzően összesített adatokat kérnek
- Jellemzően nem cél az adatok normalizáltsága, hanem az ún. csillag séma (star schema) szerinti az adatmodell

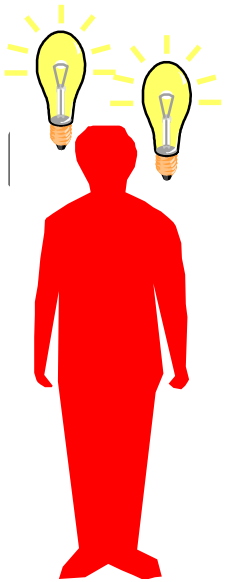
Az OLTP és az adattárházak „összeférhetetlensége”

- Üzletileg az lenne a hasznos (olcsó és praktikus), ha ugyanaz az adatbázis szolgálnák ki az OLTP és az adattárház jellegű ígényeket
- Sajnos ez egyelőre egyik gyártónak sem sikerül.
- A probléma: vagy az OLTP-s sok kicsiny felhasználó lehetetleníti el a nagy lekérdezéseket/ adatelemzéseket, vagy fordítva
- A két különféle használati mód más és más adatmodellt igényel(normalizált versus csillag séma)
- Az indexek is másfélék a kettőben
- A kiszámítható értékekhez is másként viszonyulunk

Adattárházak „életciklusa”

- A legtöbb adattárház napközben kizárólag adatelemzéseket végez (leginkább SELECT)
- Éjszaka jellemzően az ETL folyamat zajlik (Extraction, Transformation, Loading): áttöltik az adatokat az OLTP rendszerekből az adattárházba
- Az ETL egy nehéz folyamat. Gyakran egész éjszaka ez zajlik. Van aki ETL szakértőként dolgozik évekig
- Nemcsak az adatok áttöltését jelenti az ETL, hanem gyakran indexek eldobását, majd újraépítését, materializált nézetek frissítését, statisztikák kiszámítását, stb
- Ezek szerint az adattárház jellemzően nem online adatokat tartalmaz, hanem a „tegnapi adatokat”
- Az online business intelligence, vagy a near-online BI csak most kezd megjelenni

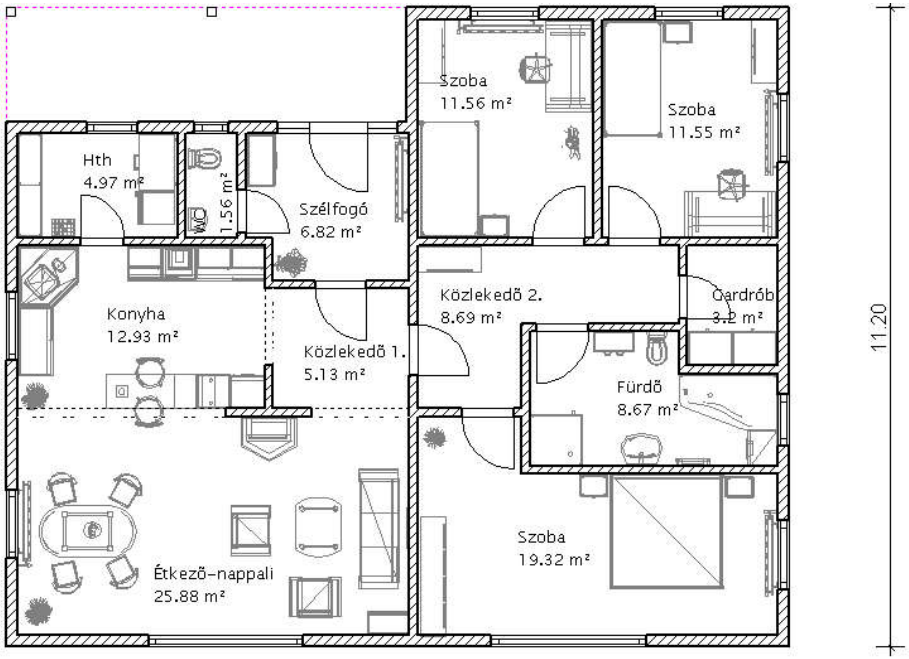
Kis kirándulás az „építészet” területére



Kis kirándulás az „építészet” területére

Családi ház (megrendelésre)

Kis kirándulás az „építészet” területére



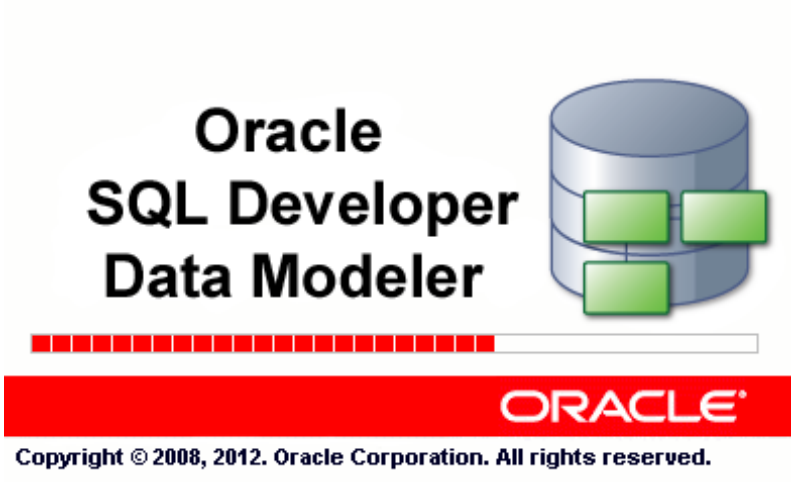
Tervdokumentációk

Az adatmodellezés (data modeling)

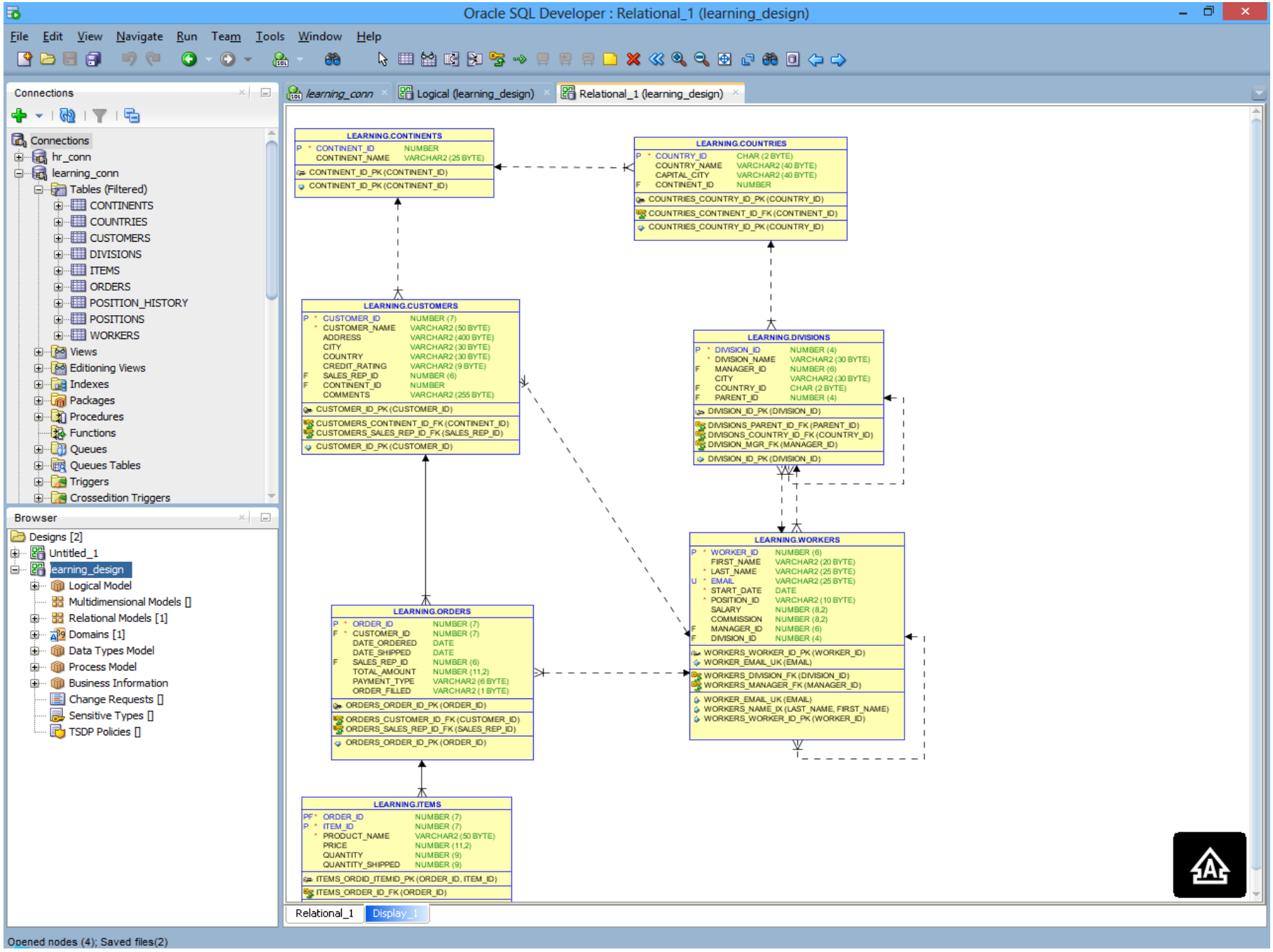
- A szoftvermérnökök világában az adatmodellezés egy olyan folyamat, amely során a készülőfélben lévő információs rendszer számára egy formális módon elkészül az adatmodell (lásd még: „Wikipedia: Data Modeling”
- Relációs adatbázisok esetén ez a modell entitásokat és relációkat, végül relációs táblákat eredményez
- Elég sok alternatív jelöléstechnika létezik
- Igen sok modellező szoftver termék is létezik (lásd: „Wikipedia: Comparison of Data Modeling Tools”)
- Ezek a termékek grafikus segítséget/támogatást nyújtanak az adatmodellező munkája során

Példa: az ORACLE adatmodellező eszközei – csupán „mutatóba” egy a sok gyártó sok terméke közül

SQL Developer Data Modeller.

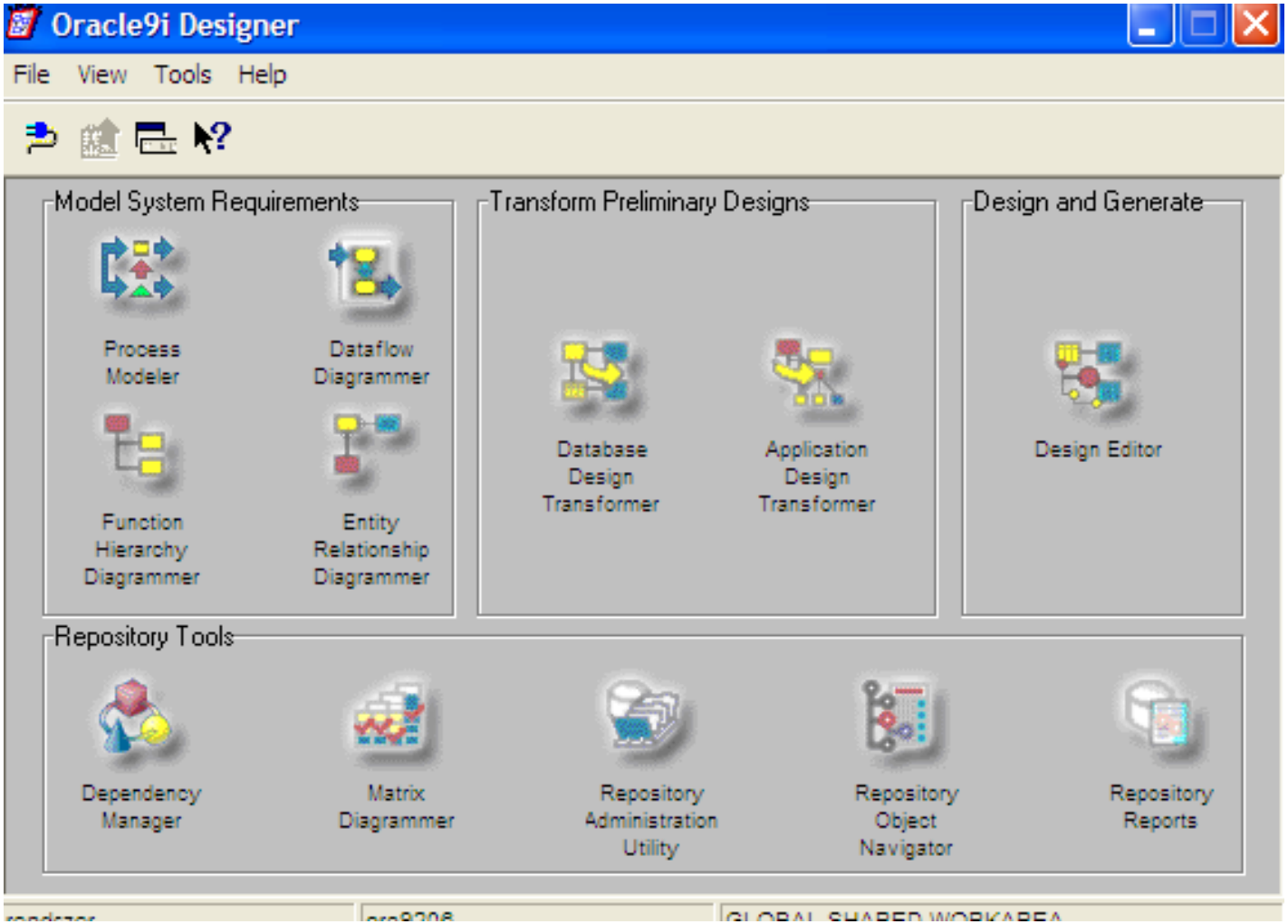


A LEARNING séma

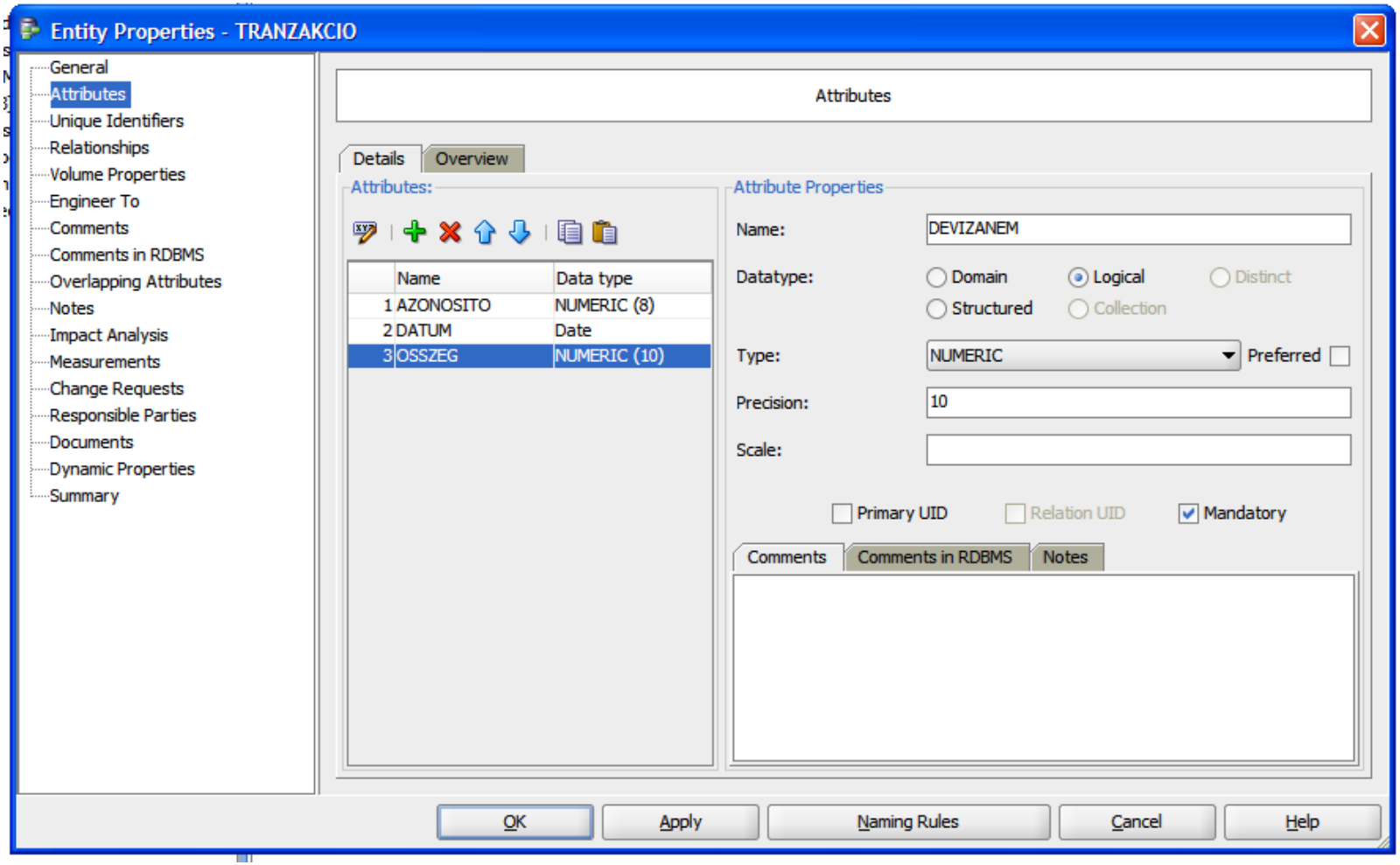


Példa: az ORACLE adatmodellező eszközei – csupán „mutatóba” egy a sok gyártó sok terméke közül

Oracle9*i* Designer



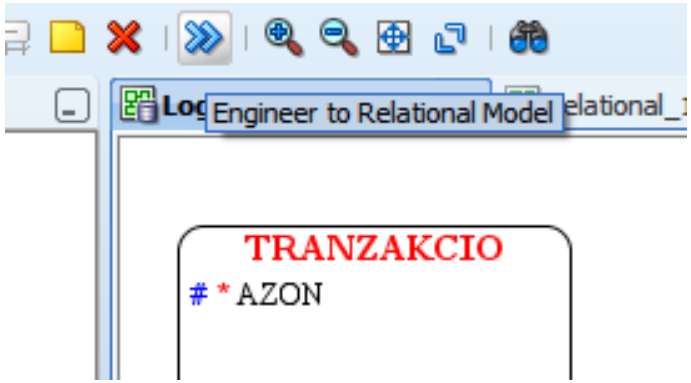
Példa: az ORACLE adatmodellező eszközei – csupán „mutatóba” egy a sok gyártó sok terméke közül (II)



ORACLE adatmodellező eszközök fő funkcionalitása

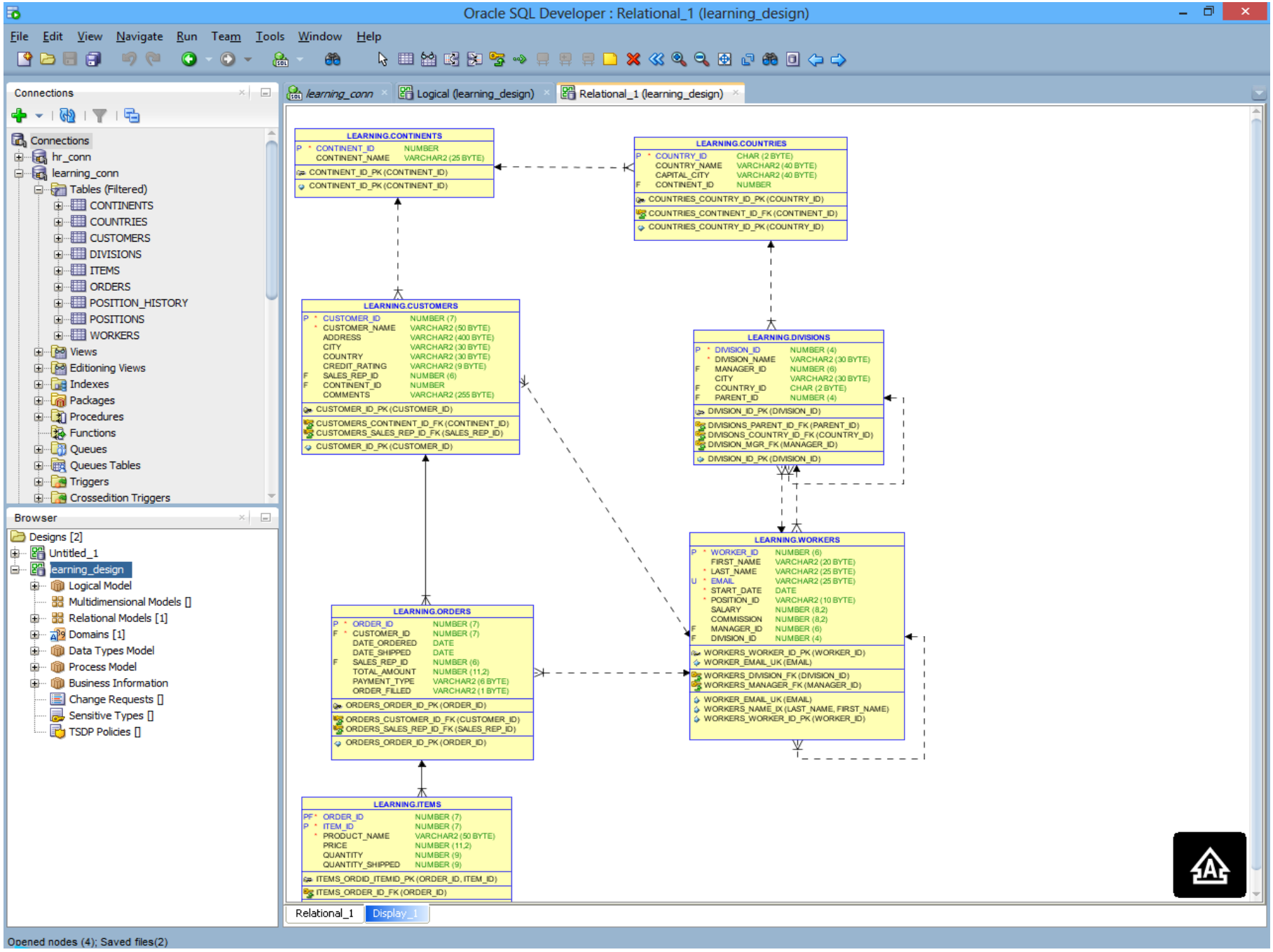
- Adatmodellek grafikus ábráinak elkészítése, karbantartása
- Nem grafikus információk rögzítése
- Adatmodellből fizikai adatbázisterv generálása
- Szabványok alkalmazása fejlesztés során
- Dokumentáció

Adatbázisterv generálása adatmodellből

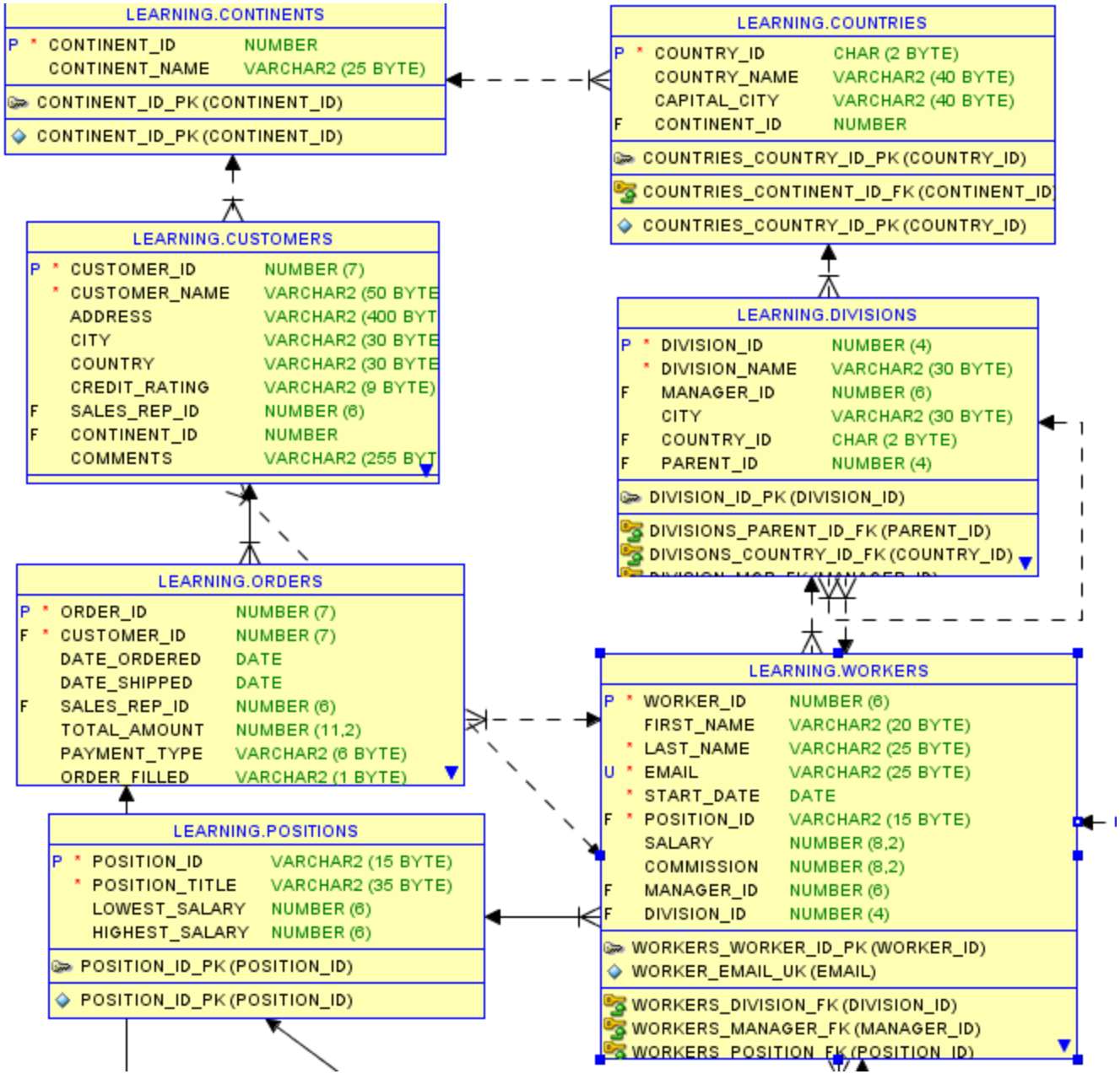


Kiinduló fizikai adatbázisterv automatikusan készül: táblák, oszlopok, megszorítások

A LEARNING séma



A LEARNING séma egy részlete



„Tervezzük meg az adatmodellt” a LEARNING séma esetében

- Még egy ilyen egyszerű rendszer esetén is többboldalas „követelményspecifikáció” szükséges
- Ez alapján többnapos feladat a táblák tervezése
- Mi itt egy végletekig leegyszerűsített esetet vizsgálunk: „Tervezzük meg a relációs sémákat és azok attribútumait (vagy: a táblákat és azok oszlopait) egy olyan rendszerhez, amelyben cégünk ügyfelei vannak különböző kontinenseken és országokban.”
- Emellett a cégünknek részlegei vannak ezekben az országokban, a részlegekben pedig dolgozók. A dolgozóknak munkakörük van, és minden munkakörhöz létezik az arra vonatkozó minimum és maximum fizetés
- Az ügyfeleink rendeléseket adnak fel. Ezekben olyan dolgozóinktól vásárolnak, akik kereskedők.

„Tervezzük meg az adatmodellt” a LEARNING séma esetében (II)

- Majd menet közben merül fel megannyi kérdés
- Az előző leírás alapján egy olyan entitás mutatkozik logikusnak, hogy „CONTINENTS”, majd egy másik: „COUNTRIES”.
- Ezek egymással hierarchikus viszonyban vannak. A „szülő” tábla a CONTINENTS, a gyerek pedig a COUNTRIES.
- A CONTINENTS attribútumai: kell-e az azonosító (mint mesterséges kulcs), vagy maga a név lesz a kulcs? Döntsünk pl. úgy, hogy lesz CONTINENT_ID és CONTINENT_NAME. Egyéb attribútum nem kell! ☺
- Az elsődleges kulcs a CONTINENT_ID lesz
- Döntsünk pl. úgy, hogy a CONTINENT_NAME-et nem tartjuk kulcsnak
- Döntsünk úgy, hogy a kontinensek neve max. 25 byte

„Tervezzük meg az adatmodellt” a LEARNING séma esetében (III)

- Eddigiek alapján megvan az első táblánk:
- CREATE TABLE continents (
  continent_id NUMBER PRIMARY KEY,
  continent_name VARCHAR(25 BYTES));

- Fenti tábladefinició egyelőre még a fizikai részleteket nem tartalmazza, csupán a logikaiakat
- Viszont nemcsak neve van és oszlopai (meg az oszlopoknak neve és adattípusá), hanem elsődleges kulcsa is
- A kulcsokról és egyéb deklaratív adatintegritási szabályról nem ezen az előadáson beszélünk részletesen.

„Tervezzük meg az adatmodellt” a LEARNING séma esetében (IV)

- Most a COUNTRIES entitás következik
- Felmerül a kérdés: lehet-e egy ország több kontinensen is? Tegyük fel, hogy az a döntés születik, hogy a mi rendszerünkben egy ország csakis 1 kontinenshez tartozhat. Mert pl. egy cég szereti így strukturálni magát.
- Felmerül a kérdés, hogy lehet-e 1-1 ország „még nem létező kontinensen”. Vagyis olyanon, amelyről még nem tudunk. Itt az a döntés, hogy nem. Tehát 1 ország 1 és csakis 1 kontinenshez tartozhat.
- Következő dilemma: lehet-e, hogy egy országról még nem tudjuk, melyik kontinenshez tartozik? Ezúttal a döntés: nem lehet az országnak „ismeretlen” a kontinense. Nem lehet NULL érték abban az oszlopban.

„Tervezzük meg az adatmodellt” a LEARNING séma esetében (V)

- Máris újabb kérdés merül fel: az országok nevei egyediek-e, vagy lehet két országnak azonos a neve. Döntsünk úgy, hogy a nevek egyediek és max. 40 betűből állnak
- Ha egyediek is a nevek, kérdés, hogy a neveket akarjuk-e kulcsként szerepeltetni, vagy inkább országkódokat vezetünk be? Döntsünk úgy, hogy országkódokat használunk majd, ráadásul szöveges kódokat, és legyenek ezek fixen 2 byte hosszúságúak.
- A következő dilemma: mi mindent akarunk még megjegyezni az egyes országokról a COUNTRIES táblában, hiszen 1-1 országnak rengeteg jellemzője van. Döntsünk úgy, hogy tudni akarjuk az ország fővárosát az ország neve, kódja és a kontinens mellett. Legyen ez maximum 40 Byte.

„Tervezzük meg az adatmodellt” a LEARNING séma esetében (VI)

- Eddigiek alapján megvan a második táblánk:
- CREATE TABLE countries (
  country_id CHAR (2 BYTES) PRIMARY KEY,
  country_name VARCHAR2 (40 BYTES),
  capital_city VARCHAR2 (40 BYTES)
  continent_id NUMBER REFERENCES continents)

- Megint a fenti tábladefinició egyelőre még a fizikai részleteket nem tartalmazza, csupán a logikaiakat.
- Bármilyen hosszadalmas volt is ez: amit itt megtettünk, az egy sokszorosan leegyszerűsített példa.
- Fenti 2 tábla a sok-sok elmondott megkötés mellett valóban normalizált, tehát OLTP-re alkalmas.

„Tervezzük meg az adatmodellt” a LEARNING séma esetében (VII)

- Foglalkozzunk még (de ne teljeskörűen) a cégünk részlegeivel. Logikusnak tűnik egy újabb entitásként bevezetni a „részleget”. A tábla neve DIVISIONS.
- Minden részlegnek legyen egy egyedi azonosítója (kulcsa): a DIVISION_ID. Legyen ez négy számjegyű.
- Tegyük fel, hogy a részlegek egymáshoz képest hierarchikus viszonyban vannak: minden részlegnek megvan az ő „felettes részlege”, vagy szülő részlege. Nevezzük ezt PARENT_ID-nek.
- Ha egy tábla sorai közötti hierarchikus viszonyt szeretnénk ábrázolni, azt a relációs adatmodellben így tesszük: egyik oszlop az elsődleges kulcs, a másik a külső kulcs, amely az elsődleges kulcsra mutat
- Van egy olyan részleg, amelynek nincs „felettese”. Ott a külső kulcs értéke NULL.

„Tervezzük meg az adatmodellt” a LEARNING séma esetében (VIII)

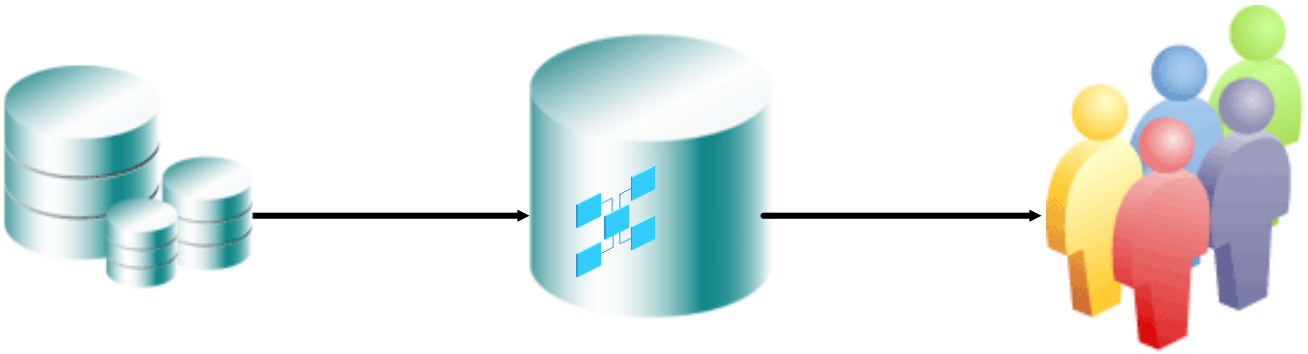
- A CONTINENTS és a COUNTRIES tábla ugyan teljes mértékben normalizáltak lettek, de a további tábláinkba szándékosan elhelyeztünk néhány olyan részletet, amely nem normalizált, illetve amely valószínűleg adatmodell tervezési hiba lehet.
- Ettől persze még működhet később az alkalmazás, csak több lehet vele a gond, mint amennyi gond normalizált adatok esetén felmerülne
- Amennyiben a slide láthatósága (mérete) ezt megengedi, kérem találjanak majd benne hibákat, vitatható döntéseket, és azokat kritizálják ☺

Az adattárházak adatmodellje

- Nem válik be a normalizált adatszerkezet – legfőképpen az adatok mennyisége és a lekérdezések hatékonysága miatt
- Alternatív adatmodelleket keresünk: csillag séma („star schema”), hópehely séma („snowflake schema”)
- Ténytáblák és dimenziós táblák fogalma
- Dimenzionális adat modellezés

Adattárház, mint extraktum

- Kontrollált
- Megbízható
- Minőségi
- Egy adat csak egyszer

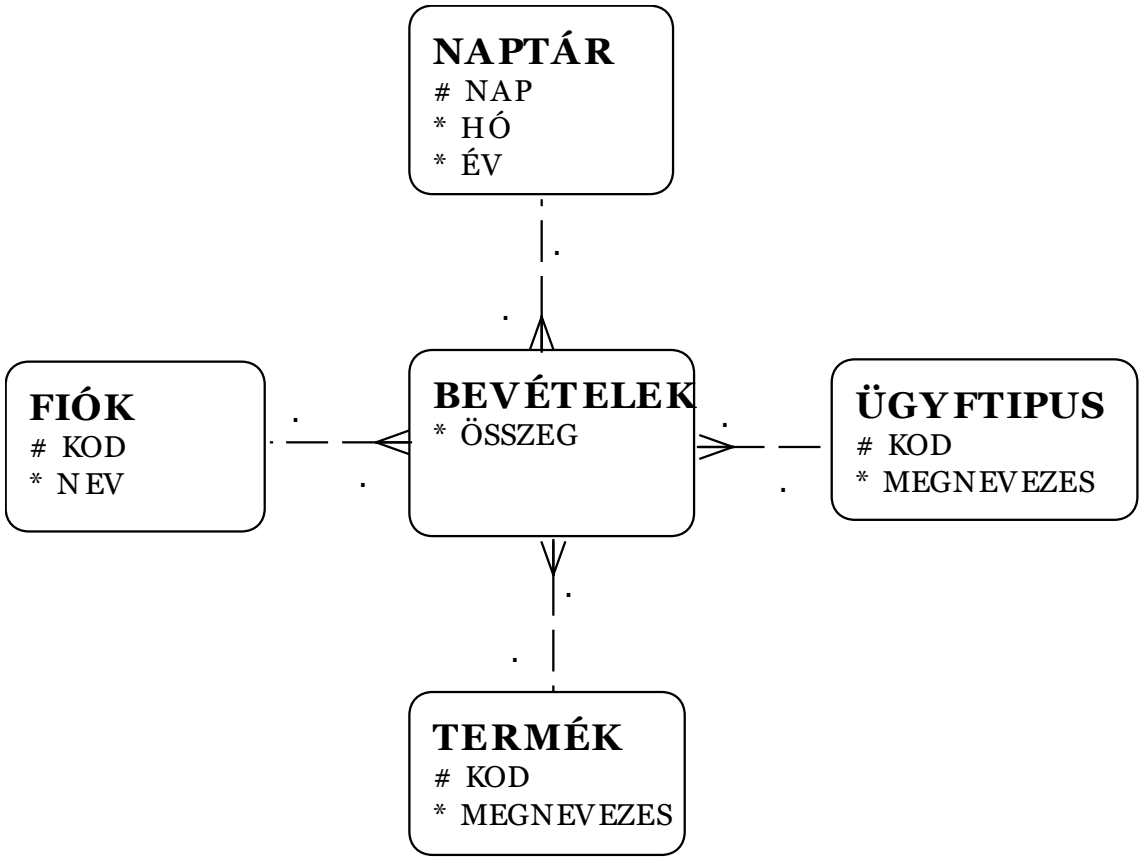


Külső és belső adatforrások

Adat-tárház

Döntéshozók

Dimenzionális modell



Dimenzionális modell alapelemei

- **Tényadatok**

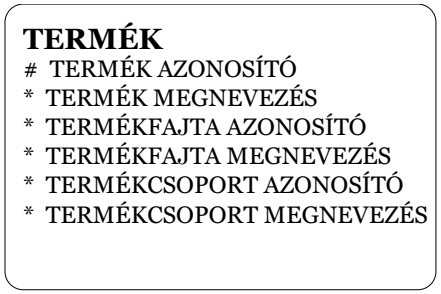
  A folyamatok teljesítményét, eredményességét leíró számok

- **Dimenziók**

  A folyamatok olyan elemzési paraméterei, amelyek kategorizálják azokat elemzési szempontból. Általában konstans vagy diszkrét értékekkel rendelkeznek a dimenziók.

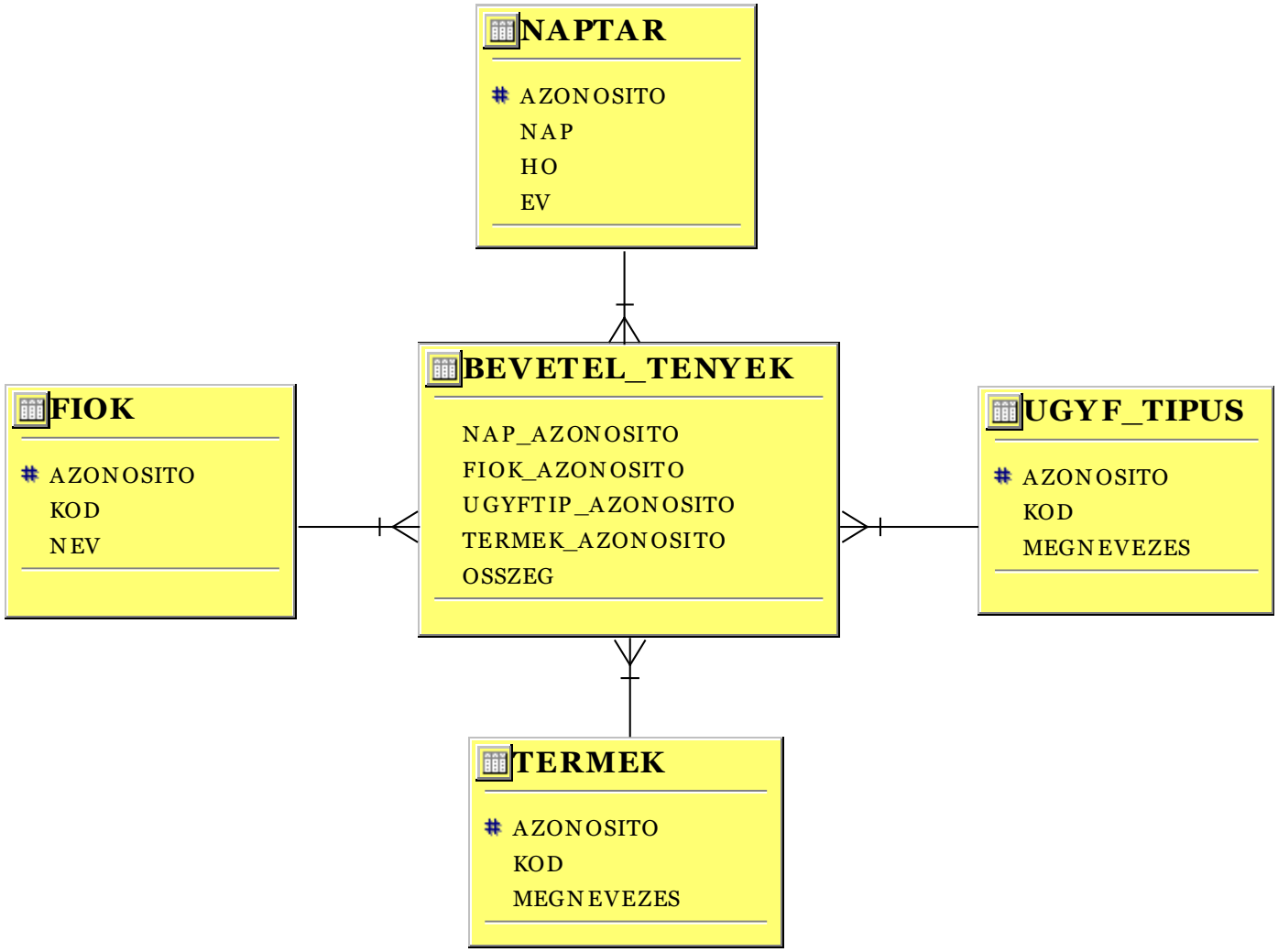
Dimenziók modellezése

- Denormalizált módon



- Normalizált módon

Csillag modell

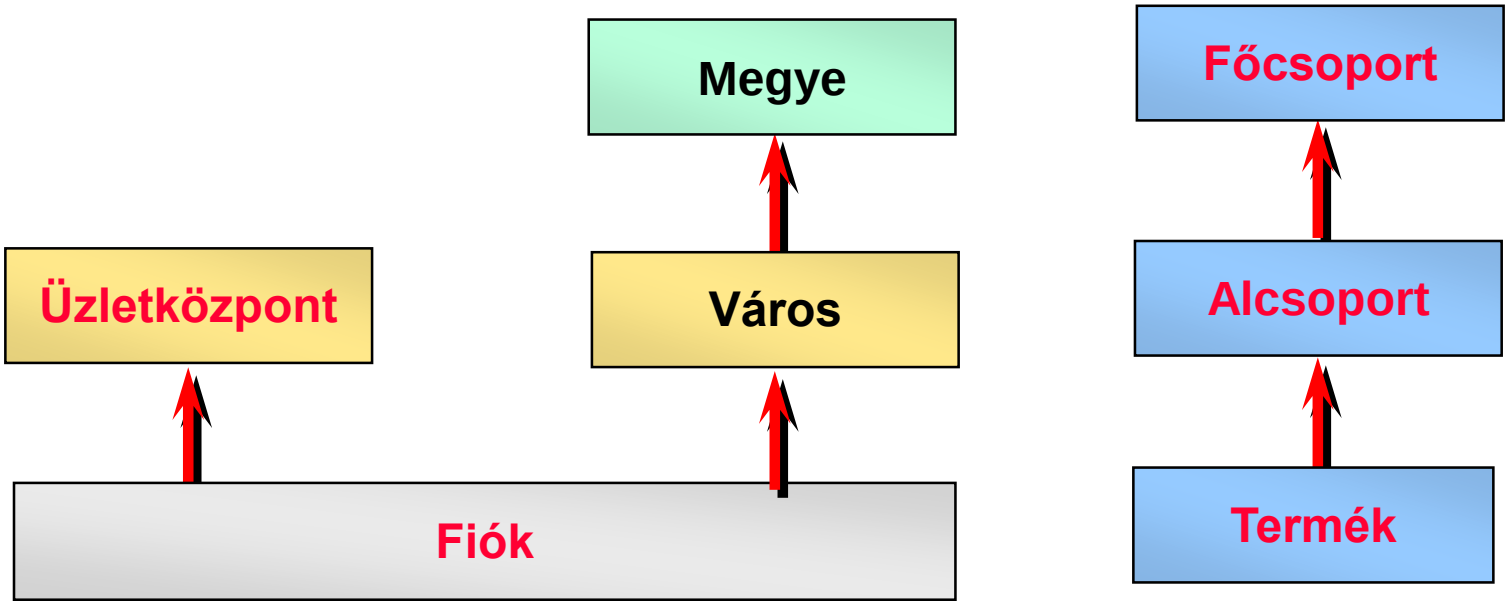


Csillag modell előnyei

- Nagyon hasonlít a felhasználó gondolkodásához
- Hatékony adatbázistervet szolgáltat
- Támogatja a multidimenzionális elemzéseket
- Könnyen kiterjeszthető, bővíthető
- Számos végfelhasználói eszközt nagyon egyszerűen „rá lehet ültetni”
- A lekérdezéseket könnyű optimalizálni

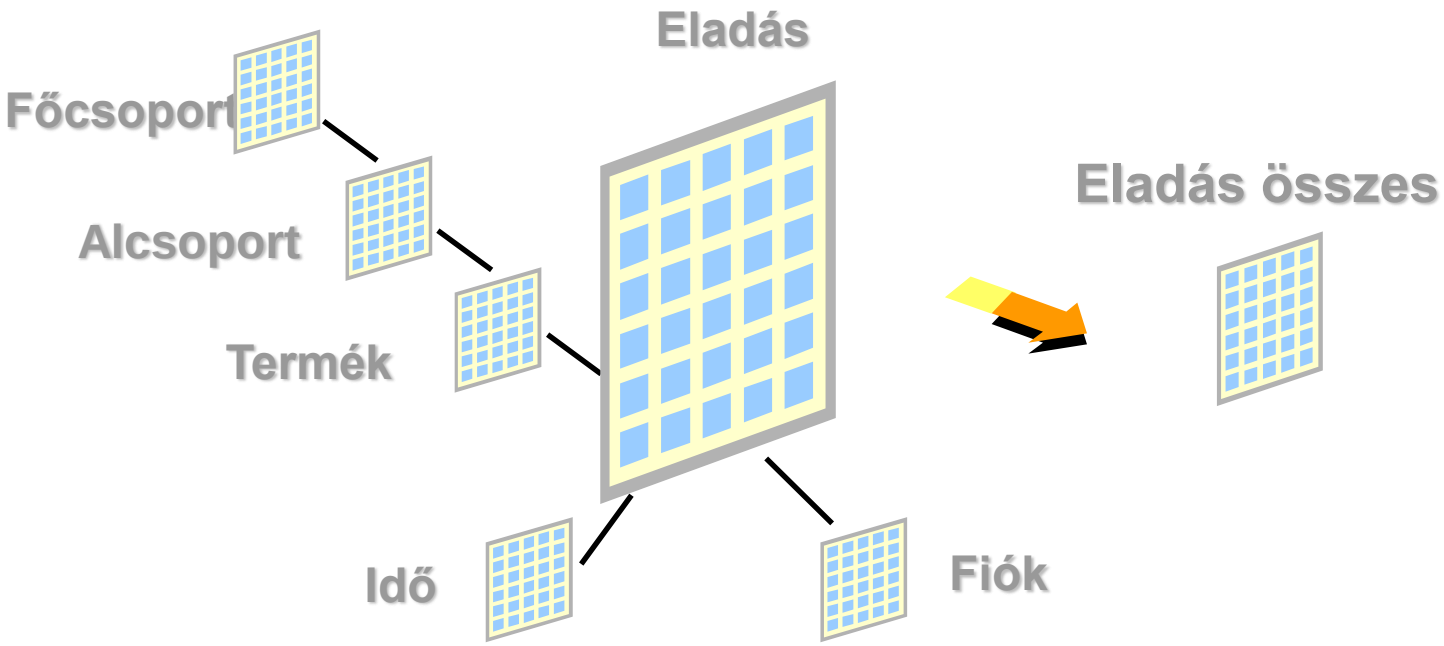
Egyéb megfontolások

- Mesterséges kulcsok használata
  - Történetiség kezelése dimenziókban
- Hierarchiák kiemelt tervezése



„Aggregátum” képzés

- **Felgyűjtött adatok tárolása felhasználói igény alapján**



Merre lehetne az adattárházak tanulmányozását folytatni?

- Adattárházak és adatpiacok („Datamart”)
- Az „Operational data store” fogalma
- Inmon javasolt adatmodellje (Az inmoni architektúra)
- Kimball adatmodellje (A kimballi architektúra)
- A Database Vault adatmodell
- …