Üzleti Intelligencia záróvizsga kérdések

A záróvizsgán az alábbiakhoz hasonló kérdésekre kell készülni. A vizsgán 1-2 kérdés várható.

Tartalomjegyzék

1. ***Ismertesse az üzleti intelligencia rendszerekben használt alapfogalmakat (adat, információ, jellemző KPI, esemény). Milyen típusú adatforrásokat tudunk megkülönböztetni? Mik a fő eltérések OLTP és BI rendszerek között?***

Üzleti intelligencia

- „*Techniques and tools for the transformation of raw data into meaningful and useful information for business analysis purposes.*"

- „*Applications, infrastructure, tools and best practices that enable access to and analysis of information to improve and optimize decisions and performance.*"

Üzleti intelligencia

- Mást jelenthet
  - > Üzleti nézőpont: döntés támogatáshoz eszköz
    - – Milyen színű autók fogynak jobban melyik országokban? Hogyan érdemes készletet tartani?
  - > Technológiai nézőpont: szoftver és/vagy hardver megoldás
    - – SQL Server Analytics, Oracle BI, Hadoop, stb.
  - > Fejlesztői szempont: újfajta eszközök, algoritmusok
    - – Relációs modelltől eltérő adattárolás
    - – Hatékony adattárolás és adatfeldolgozás
      - Hatékony adatreprezentáció
      - Masszívan párhuzamos feldolgozás
    - – Adatbányászati algoritmusok

Adat és információ

- Adat ≠ információ
- Adat: nyers tény
  - > Tegnap eladtunk 5 terméket.
- Információ: újdonság
  - > Ez a legnépszerűbb termékünk.
  - > Adatokból derül ki.
- Nemtriviális
  - > Triviális: Hány terméket adtunk el. -> Ez még nem üzleti intelligencia.
  - > Nemtriviális: Mely termék árát érdemes csökkenteni annak érdekében, hogy többet vásároljanak és összességében többet költsenek?

Adat forrás típusok

- Strukturált (structured)
  - > Meghatározott struktúra és tartalom.
  - > Excel táblázat, adatbázis
- Félig struktúrált (semi-structured)
  - > Valamilyen szintű struktúra adott, de azon belül a tartalom kötetlen.
  - > Szöveges fájl, XML, logfájl
- Struktúrálatlan (unstructured)
  - > Képek, videók, szabad szövegek
  - > Weboldalak, tweetek, Facebook post-ok, Word dokumentumok

Jellemző

- Mérhető
- Szokták KPI-nek (Key performance indicator) nevezni
  - > Üzleti terminológia is (pl. KPI az eladott termékek darabszáma)
- Önmagában nem elég
  - > Összefüggések, következtetések alapja
- Pl.
  - > Hányszor indítják el a felhasználók az új funkciót?
  - > Hány kérést kap a backend?
  - > Milyen hosszan telefonál egy előfizető?
  - > Mennyi a bevétel az adott negyedévben?
- Üzleti folyamatból adódik
  - > Nem mindig triviális
  - > Lásd: adatbányászat

Esemény

- Az adat egysége egy esemény
  - > X termékből Z darabot Y áron eladtunk T dátummal
  - > Webshopunkban az X termék oldalát megnézték
  - > X funkciót T időpillananatban elindította Z felhasználó
- Esemény
  - > Mérhető: darab, hossz, időtartam, kg, $, stb.
  - > Általában additív (összeg értelmezhető)
  - > Általában értelmezhető valamilyen aggregáció (átlag, min, max, stb.)
  - > Egyéb attribútumok, pl. mikor, ki, hol

KPI kiválasztás

- KPI = Key Performance Indicator
- Olyan értékek adása a menedzsment kezébe, amelyen keresztül át tudják látni a felelősségük alá tartozó üzleti területeket, látják azok minden mozdulatát és egyből észreveszik, ha elkezdenek letérni az egyensúlyi pályáról.
- KPI jellemzője:
  - Aktuális érték, azaz hol állunk most
  - Terv érték, azaz mit terveztünk, hol fogunk állni most
  - Eltérés érték, azaz mennyivel és milyen irányban tértünk el a tervtől
  - Végül egy trend érték, ami megmutatja a tény értékek trendjét
- Komoly kihívás a KPI kiválasztása és megjelenítése

Üzleti rendszerek

- Vállalatirányítási rendszer (ERP)
  - Tranzakció alapú (OLTP)
    - Általában egyszerre egy tranzakciót futtat, egy tranzakció rövid
    - Jól definiálható műveletekre (tranzakció típusokra) épül
  - Minél több információ felvitele, módosítása
  - Sok adatot érintő lekérdezés lassabb
  - Aktuális információt tartalmaz
    - Tipikusan kevés historikus adatot tárol
- Üzleti intelligencia rendszer
  - Adat feltöltése lassú
  - Sok adatot érintő lekérdezés gyors
  - Sok adat szükséges egy lekérdezés kiértékeléséhez
  - Historikus információval (is) dolgozik

2. Ismertesse az üzleti intelligencia rendszerek általános felépítését, valamint a Kimball és Immon féle architektúrákat. Ismertesse az építőkövek funkcióit, és mutasson be konkrét szoftvereket vagy technológiákat amelyek az adott funkciót megvalósítják.

Üzleti intelligencia rendszer

- Üzleti intelligencia rendszer
- Nem egy szoftver
- Több szoftver, ezek integrációja
  - Adattárház
  - Adat betöltő
  - Feldolgozó
  - Megjelenítő

Üzleti intelligencia rendszer funkciói

- Riport
  - *Mi történt?*
  - Múltbeli adatok alapján.
- Elemző
  - *Miért?*
  - Összefüggések feltérképezése.
- Monitorozó
  - *Mi történik most?*
  - Teljesítmény monitorozók, irányítópultok (dashboard), mutatószámok (scorecard).
- Predikció, adatbányászat
  - *Mi fog történni?*
  - Modellek felállítása, előrejelzés.
  - Nem ismert összefüggések, kapcsolatok feltérképezése.

Üzleti intelligencia rendszer beüzemelése

- Beruházás
  - Hardver, szoftver
  - Fejlesztés: adattárolás, áttöltés, folyamatok
- Többszereplős feladat
  - Projekt vezető
  - Architekt
  - DBA
  - Üzleti elemző
  - Fejlesztő
  - Front-end tervező
- Bevezetés után
  - Oktatás, támogatás
  - Továbbfejlesztés
    - Új adat, új lekérdezések, riportok, stb.

Rendszer követelmények

- Könnyű hozzáférés az adatokhoz
  - A felhasználók (itt: üzleti döntéshozók) számára hozzáférhető, érthető, intuitív
- Konzisztens adat
  - Sok helyről származó információk
  - Meg kell tisztítani, közös nevezőre hozni
    - Eltérő rendszerek eltérő terminológiát használnak („lakcím”, „cím”, „székhely”)
  - Ellenőrizni kell az adatot betöltés előtt
    - Ne kerüljön hibás adat a rendszerbe
- Megbízható információforrás
  - Elegendő és megfelelő adat a kérdések megválaszolásához

Rendszer követelmények

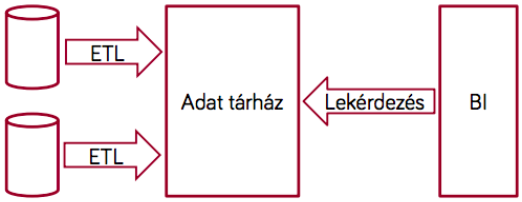
- Igények, folyamatok, adatok változásának kezelése
  - Folyamatosan változó környezet, a rendszernek alkalmazkodnia kell
  - Új kérdések megválaszolásához ne kelljen fejlesztés
  - Új adat bevezetéséhez ne kelljen fejlesztés
- Megfelelő teljesítmény
  - A lekérdezések nem tarthatnak órákig
  - Úgy kell feldolgozni, előkészíteni, szervezni az adatot, hogy lehetővé tegye a gyors lekérdezéseket
- Biztonság
  - Adatokhoz történő hozzáférés szabályozása

Tipikus architektúra elemek:

- Forrás rendszerek
- Forrás-cél integráció
- Állomásoztató terület
- Nyers adatok
- Adattárház séma
- Metadata séma
- Operational Data Store

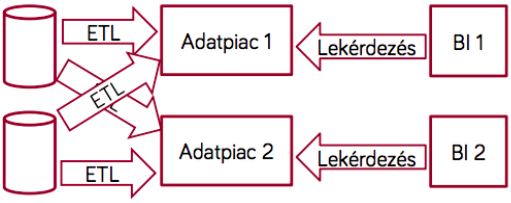
Kimball modell

- Központi adat tárház



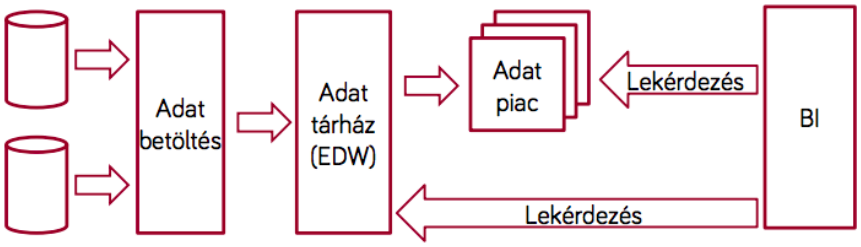
Független adatpiacok

- Egy közös adattárház helyett adott felhasználási területre szabott adatpiacok (data mart)



Immon modell

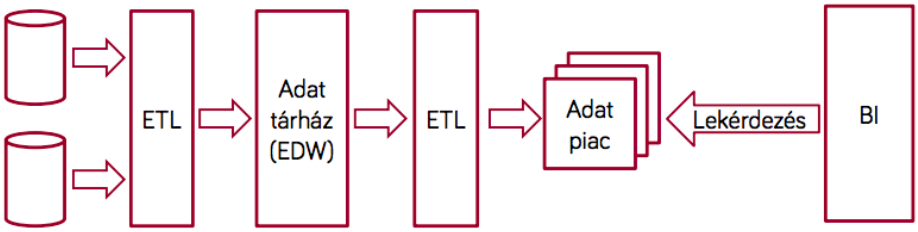
- Köztes adattárház (Enterprise Data Warehouse)
  - Normalizált (3NF) formában
- Felhasználási területre szabott adatpiacok
  - Általában nem a teljes adattartalom
  - Felhasználási területre szabott adatok, esetleg részben előre feldolgozva
- Lekérdezéseket az EDW vagy az egyes adatpiacok szolgálják ki



Hibrid Kimball-Immon modell

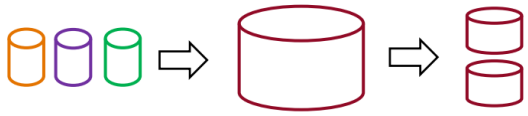
- Köztes adattárház (Enterprise Data Warehouse)
  - Normalizált (3NF) formában
- Felhasználási területre szabott adatpiacok
  - Általában nem a teljes adattartalom
  - Felhasználási területre szabott adatok, esetleg részben előre feldolgozva
- Lekérdezéseket az egyes adatpiacok szolgálják ki

- Könnyű hozzáférés az adatokhoz
  - BI szoftver feladata
  - Nem kell SQL-eket írni
- Konzisztens adat
  - ETL folyamat feladata biztosítani
- Megfelelő teljesítmény
  - Megfelelő adatreprezentáció az adatpiacokban
- Igények, folyamatok, adatok változásának kezelése
  - Lokalizált változás
  - Ha forrásadat változik: ELT változás
  - Ha igény változik: más lekérdezés
- Biztonság
  - Forrás adat érintetlen
  - BI szoftver biztosítja



Adattárház, adat piac

- Vállalat sok, különböző jellegű adatbázissal rendelkezik
  - Pl. termék katalógus, raktárkészlet, bérszámfejtés, CRM, stb.
- Adattárház (data warehouse) egy központi hely ezen adatoknak
  - *Nem* egy központi adatbázis szerver
  - *Nem* helyettesíti a különböző rendszerek saját adatbázisát
  - Kombinálja az adatokat, tisztítva, normalizálva, egy központi helyen teszi elérhetővé
  - Tipikusan 3NF jellegű modell -> könnyű szinkronizálás végett
- Adatpiac (data mart): adattárház adatainak részhalmazát tárolja
  - Egy adott feladatkör, szakterület számára készül, csak az ezen célra szükséges adatokat tartalmazza
  - Tipikusan felhasználási célnak legmegfelelőbb tárolási modell, akár előre aggregálva

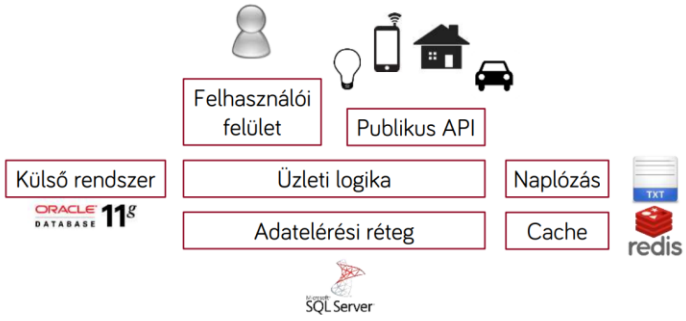


Alkalmazások:

- Microsoft SQL
- Oracle BI
- RapidMiner
- Pentaho (Java, Open source)
- Hadoop

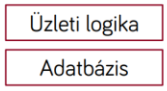
3. Ismertesse, milyen kihívásokkal szembesülnek a modern alkalmazások az adattárolás és feldolgozás terén, amire a tradicionális relációs adatbázisok nem minden esetben nyújtanak megoldást. Mutassa be, milyen módszerekkel kezelhetőek ezen kihívások. Térjen ki a „near data processing”-re és a dimenzionális adatmodellre is.

Mai alkalmazások

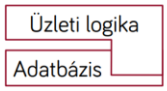


Üzleti logika

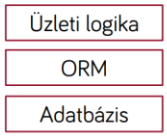
- Tradicionálisan



- Szerver oldali programozás
  - Tárolt eljárások, triggerek
  - Jobb teljesítmény, zárt, biztonságos futtató környezet
  - Platformfüggő, nem jól skálázódó



- ORM keretrendszerek (pl. EF, JPA)
  - Plusz réteg
  - Absztrakció növelése
  - Tárolt eljárásoknál, SQL-nél kényelmesebb

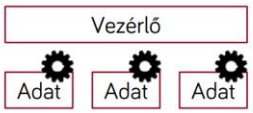


Kihívások

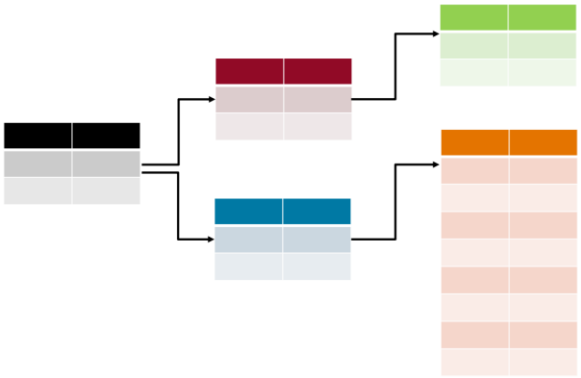
- Heterogén tárolás
  - > Több rendszerben tárolt adat
- Változatos formátumok
  - > Relációs, szöveges, gráf, stb.
- Nagy adatmennyiség
  - > BigData
- Folyamatosan generálódó új adat
  - > Pl. logok minden nap keletkeznek

Kihívások

- Új adattárolási megoldások
  - > Memória adatbázisok, NoSQL adatbázisok
- Új módszerek
  - > „Near data processing”
    - Feldolgozás idejét az adat hozzáférés ideje határozza meg
    - Nagy adathalmazok utaztatása költséges
    - Adat helyett vigyük a feldolgozó logikát az adathoz
  - > Masszívan párhuzamos feldolgozás



Relációs adatmodell

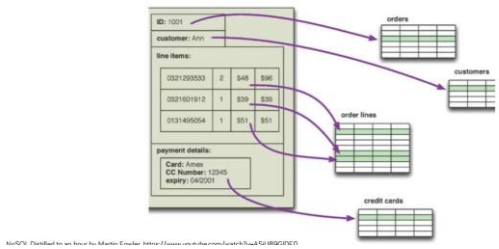


Relációs adatmodell

- Entitások és köztük levő relációk (kapcsolatok)
- Relációs adatbázisok implementálják
- Normalizált séma (általában 3NF)
- Beszúrás, módosítás kevés rekordot érint
  - > Normalizálás miatt
  - > Erre optimalizált működés
- Adatmodell jól ismert célokat szolgál
  - > Előre ismert lekérdezések
  - > Erre optimalizált séma

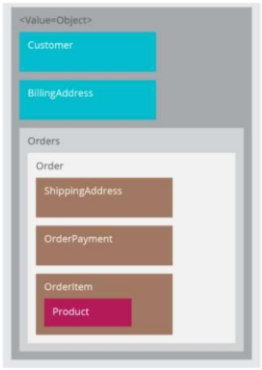
Relációs adatmodell

- Adatmodell eltérő logikával épül fel, mint ahogy az adatokat memóriában tároljuk, vagy a felhasználói felületen megjelenítjük
  - > Adatbázis: több tábla és több sor
  - > Memória: hierarchikus struktúrák (pl. listák listája)
  - > Megjelenítés: több helyről származó információ egy helyen



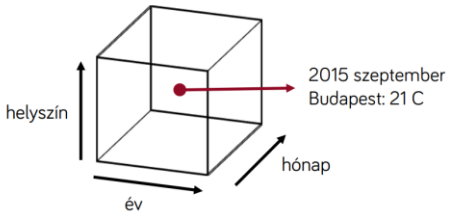
Objektum (orientált) adatbázisok – kitérő

- Relációs adatbázisoktól eltérő modell
  - > Objektum-orientált paradigmához illeszkedik
  - > Objektumok tárolása
- Nem terjedtek el
- Reinkarnáció: lásd NoSQL dokumentum alapú adatbázisok



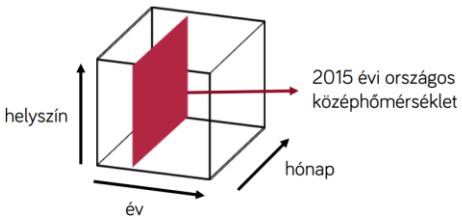
Dimenzionális adatmodell

- Többdimenziós kocka amelynek dimenziói a mért tartományok
- A kocka egy pontja a dimenziók adott pontjában mért érték
- Pl. havi középhőmérséklet



Dimenzionális adatmodell

- Többdimenziós kocka amelynek dimenziói a mért tartományok
- A kocka egy pontja a dimenziók adott pontjában mért érték
- Pl. havi középhőmérséklet



Dimenzionális adatmodell

- Dimensional modeling
- Adattárházakban használt technika
- Célja
  - Gyors lekérdezés támogatás
  - Aggregáció támogatása
  - Könnyen értelmezhető adat
- Miért *nem* jó erre a 3NF
  - Kapcsolatok megkötések -> lassabb lekérdezések sok adat esetén
  - Normalizált adatreprezentáció -> értelmezéséhez ismerni kell a teljes adatmodellt

Dimenzionális adatmodell

- Dimenzionális modellt lehet relációs adatbázisban is tárolni
  - Nem feltétlenül hatékony
- OLAP (Online Analytical Processing)
  - Többdimenziós adatbázis
  - Nem relációs
  - Hatékonyabb, mint a relációs adatbázis
  - Hatékony végrehajtáshoz a rendszer előre kiszámol összesített értékeket

Dimenzionális adatmodell

- OLAP kockán végrehajtható műveletek
  - Lefúrás
    - Aggregált érték felbontása a komponensekre
    - Pl. éves középhőmérséklet -> havi bontású középhőmérséklet
  - Aggregáció
    - Aggregált értékek egy választott dimenzió mentén
    - Pl. Budapesti napi középhőmérséklet -> Budapesti havi középhőmérséklet
  - Csoportosítás / pivot
    - Más tengely szerinti nézet
    - Pl. éves középhőmérséklet évenként -> éves középhőmérséklet városonként
  - Szűrés
    - Adott dimenzió mentén egy pont kiválasztása
    - Pl. adott naphoz tartozó hőmérséklet adatok
  - Szeletelés
    - Egy-két dimenzió mentén szűrés egy tartományra
    - Pl. nyári középhőmérséklet a Dunántúlon

4. Mutassa be a memória adatbázisok működését, ismertesse, miben különböznek a tradicionális adatbázisoktól. Ismertessen egy választott memória adatbázist. Mutassa be az oszlop alapú- és a sor alapú tárolási módot, ismertesse előnyeiket és hátrányaikat.

A konkrét feladat és követelményei ismeretében IMDB alkalmazása mellett vagy éppen ellen dönteni: ez a rendszert tervező mérnök feladata. Ma már rendelkezésre állnak hibrid rendszerek is, amelyek az adatbázis egy részét IMDB, másik részét DRDB elven kezelik, és teszik ezt a programozó számára teljesen transzparens módon. Ezen rendszerek bizonyos korlátok között képesek az egyes relációk hozzáférési statisztikái alapján migrálni az adatok megfelelő részét az IMDB és a DRDB alrendszer között. Az adatbázis tervezője azonban nagyobb rálátással rendelkezik az elkészítendő rendszerre, így – a rendszer indulásakor legalábbis – jobb konfigurációt definiálhat, mint az automatizmus.

- Index hash alapú - segíti a cache-elést, de tól-ig keresésre nem alkalmas. Naplózás - csak redo.
- ACID tartósság:
  - Tranzakciós naplózás fájlba (parancsok, műveletek sor szintű naplózása)
  - Snapshot - fájlba mentés
  - Replikáció
  - NVRAM

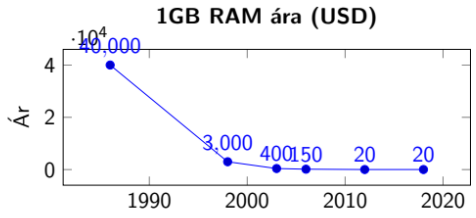
Memóriaadatbázis: alapelv

Memóriaadatbázis-rendszer (IMDB) (In-Memory DataBase)

- az adatok elsődleges példánya a fizikai memóriában
- DRDB: az elsődleges példány diszken van (Disk-Resident DataBase)

  mondjuk inkább úgy: blokkos tárolón (vö. SSD)

Az alapelv létjogosultsága



- Sok és olcsó RAM

  OLTP, OLAP rendszerek számára
- Opció: adatbázis particionálása IMDB és DRDB részre
  - automatikusan vagy kézzel konfigurált módon
  - az adatok migrációja a két rendszer között

IMDB használatának előnyei

- gyorsabb adatelérés, tranzakció-feldolgozás

  akár hard real-time alkalmazások
- egyes esetekben egyszerűbb alkalmazás-logika

  diszk-IO alrendszer eliminálása
- saját adatszerkezetek helyett kész komponens

  olcsóbb fejlesztés, olcsóbb termék

IMDB: hátrányok, nehézségek

- bizonyos esetekben bonyolultabb alkalmazás-logika
  - tartósság biztosítása
  - biztonsági mentések szervezése
- a rendszer indulásakor fel kell tölteni az adatbázist
  - mentésből, fejállomásról letöltve vagy on-line adatgyűjtésből
  - segédstruktúrák online felépíthetők
- Megéri-e: előnyök vs. hátrányok: a konkrét feladat határozza meg

IMDB: megvalósítási kihívások

- optimalizált adatszerkezetek
  - a fizikai memória véletlen elérését kihasználják
  - zárkezelés, tranzakciók ütemezése
  - megfelelő granularitású zárak
- tartósság
  - HW támogatás: elemes RAM, hálózaton szinkronizáló RAM-kártyák
  - memrisztor (RRAM: Resistive random-access memory)
  - szinkron vagy aszinkron naplózás
  - nem követelmény

Tranzakciós tulajdonságok I.

Szemben a DRDB-nél megismertekkel, az IMDB rendszerekben:

- A – atomicitás
  - nincs markáns különbség
  - gyorsabb tranzakciók, magasabb zár-granularitás
- C – konzisztencia
  - nincs markáns különbség
- I – izoláció
  - sorosítható helyett valódi soros ütemezés
  - környezetváltások: a processzor-cache, mint környezet
- D – tartósság

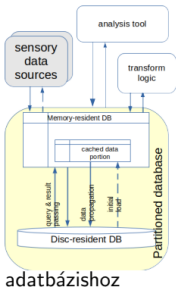
Tranzakciós tulajdonságok II. A tartósság

Az adatbázisba „írt” adatok „megmaradnak”

- diszk: passzív
- memória: aktív
  - tápfeszültség kimaradásakor törlődik
  - elemmel támogatott RAM
- bithiba – modul meghibásodások
  - ECC (vö: diszkek: RAID)
  - modultöbbszörözés
- naplózás (és persze mentések)

Architektúra: partícionálás

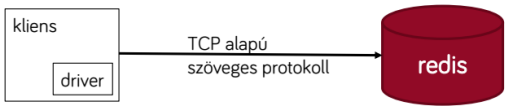
- ha a teljes adatbázis IMDB-ben nem
  - megoldható v.
  - racionális
- Hibrid megoldás
  - tiszta IMDB
  - in-memory cache
  - DRDB



Az ábrán: az elemzőeszközök a partícionált adatbázishoz csatlakoznak, amíg az adatforrások friss adatai közvetlenül az IMDB részbe érkeznek, hogy minél hatékonyabb lehessen az előfeldolgozásuk.

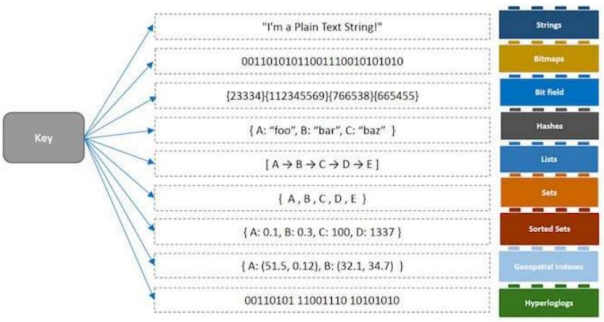
Redis

- Adatbázis
  - > „Data structure server”
  - > Kulcs-érték tár
- Perzisztens
  - > Memóriában tárolt → GYORS!
  - > Snapshotok lemezre *konfiguráció szerint*
- Rendszer architektúra



Redis - adattípusok

Kép forrása: https://www.infoworld.com/article/3230455/how-to-use-redis-for-real-time-metering-applications.html



+ TTL + atomi adatmódosítás

Redis – replikáció & perzisztencia

- Replikáció támogatás
  - > Aszinkron a háttérben
  - > Master-slave
  - > Nincs konzisztencia garancia
    - Bizonyos esetekben elveszhet módosítás
- Memóriában tárolt konfiguráció függően diszkre mentett
  - > Maximum annyi adatot tárolhat, amennyi memóriában elfér
  - > Snapshot („RDB”)
  - > Redo log („AOF”)

Redis - tulajdonságok

- Tranzakció támogatott
  - > Serializable jellegű
  - > DE! Nem atomi
    - Sikertelen művelet esetén folytatja a következővel
- Biztonság
  - > „Trusted environment” modell
  - > Nincsenek userek, jogok, authorizáció
  - > Authentikáció: 1 clear text jelszó
- Lua script támogatás
  - > Kizárólagos adat hozzáférés
  - > Végrehajtási idő limit

Sor és oszlop alapú tárolás

1	A	Q
2	B	W
3	C	Z

- Sor

1	A	Q	2	B	W	3	C	Z

  - Sor szintű hozzáféréshez praktikus
  - Egy sor adatai egymás után helyezkednek el
  - OLTP rendszerekben

- Oszlop

1	2	3	A	B	C	Q	W	Z

  - Oszloponkénti hozzáféréshez praktikus, pl. aggregálás oszlop mentén
  - Egy oszlop adatai egymás után helyezkednek el
  - OLAP feldolgozáshoz

5. Ismertesse a NoSQL adatbázisok típusait. Mutassa be általánosságában az egyes fajtákat és ismertessen egy-egy konkrét szoftvert. Ismertesse a CAP tételt.

NoSQL

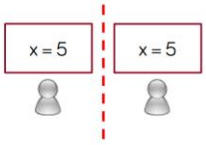
- „Not only SQL”
- Nem relációs modellre épülő adatbázis
- 1960-as évek óta léteznek de csak ca. 2009 óta „NoSQL”
- Alapelv: erős konzisztencia helyett rendelkezésre állás és skálázhatóság

NoSQL

- Nincs igazi definíciója
- NoSQL
  - Nem relációs
  - Open-source
  - Klaszterezhető
  - Séma nélküli
  - ~~Nem SQL~~ -> semmi köze az SQL nyelv támogatásához

CAP tétel

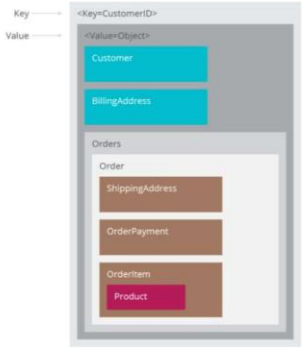
- CAP tétel: elosztott rendszer a három alapvető képesség közül legfeljebb kettőt tud megvalósítani
  - Consistency (konzisztencia): minden csomópont azonos az állapotot lát
  - Availibity (rendelkezésre állás): minden kérésre érkezik válasz
  - Partition-tolerance (particionálás-tűrés): hálózati és egyéb hibák esetén is működőképes marad a rendszer



- Skálázáshoz P szükséges C és A közötti tradeoff
  - Inkább: szabályozható testre szabható

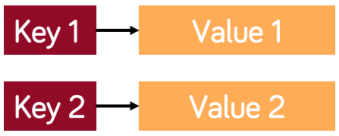
NoSQL

- Típusai
  - Dokumentum
    - Tárolt adat egy objektum mezőkkel és azok értékeivel
    - Bejárható („select *”)
    - A mezőkre is lehet keresni indexelni



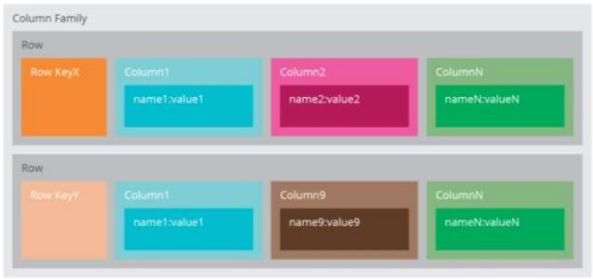
NoSQL

- Típusai
  - > Kulcs-érték
    - – Érték a rendszer által nem értelmezett
    - – Csak a kulcs birtokában kapható meg az adat



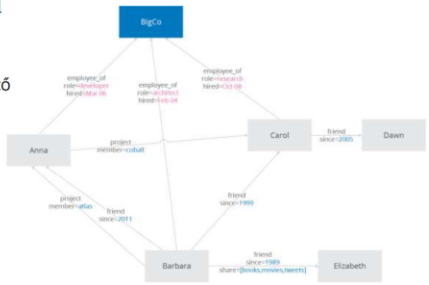
NoSQL

- Típusai
  - > Oszlop alapú
    - – Oszlop családok (Column family)
    - – Nem minden sor tartalmazza az oszlopot



NoSQL

- Típusai
  - > Gráf
    - – Entitások és közül levő kapcsolatok
    - – Speciális célokra
    - – Speciális lekérdező nyelvek

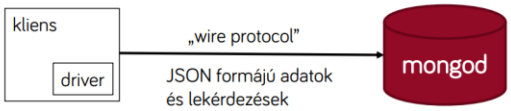


Relációs / NoSQL

- Relációs
  - > Séma
    - – Konzisztencia
  - > Join
  - > Nehezebben skálázható
  - > SQL
    - – Kvázi standard
    - – Nagy kifejezőerő
  - > Erős konzisztencia
    - – Zárolás
  - > Tranzakciók
- NoSQL
  - > Nincs séma
    - – Rugalmas
  - > Nincs join
    - – Nem is szükséges
  - > Horizontálisan skálázható
    - – Sharding
  - > Nincs egységes API
  - > Konzisztencia szabályozható
  - > Tranzakció nincs
    - – Csak atomi módosítás dokumentum szinten

MongoDB

- Dokumentum alapú elosztott, adatbázis
- Rendszer architektúra



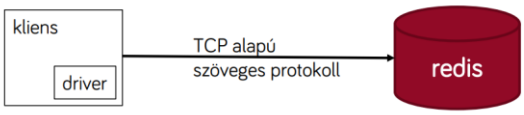
- Logikai felépítés
  - > Klaszter
    - – Szerver
      - Adatbázis
        - Gyűjtemény
          - Dokumentum

Cassandra

- Elosztott adatbázis
  - > Minden csomópont egyenrangú
- Lineárisan skálázható
  - > Kérések száma a csomópontok számának függvényében
- Replikáció támogatás
- ACID támogatás
  - > Konzisztencia szabályozható: hány csomópontnak kell végrehajtania az írást
  - > Eventually consistent: írás után régi értékek még léteznek a replikált csomópontokon
- Gyűjtemény oszlopok
  - > Halmaz, lista, map
  - > 1-több kapcsolatok leképzése külső kulcs helyett

Redis

- Adatbázis
  - > „Data structure server”
  - > Kulcs-érték tár
- Perzisztens
  - > Memóriában tárolt → GYORS!
  - > Snapshotok lemezre *konfiguráció szerint*
- Rendszer architektúra



6. Mutassa be, miről szól az adatbányászat. Milyen adatbányászati problémákat ismer? Milyen tipikus feladatokat tudunk megoldani adatbányászati eszközökkel üzleti intelligencia rendszerekben?

Adatbányászat

The nontrivial extraction of implicit, previously unknown, and potentially useful information from data.

- > Nem triviális, rejtett
- > információk és összefüggések feltárása
- > nagy adathalmazokban.
- > Ezek felhasználása ismeretlen adatok becslésére.

Mi számít adatbányászatnak?

Példa: élelmiszerüzlet

- > Vásárlók száma
  - > triviális információ
- > Előző havi bevétel
  - > triviális információ
- > Leggyakrabban együtt vásárolt termékek
  - > gyakori elemhalmazok keresése
  - > frequent itemsets
- > Hány kasszát kell kinyitni pénteken 16 órakor?
  - > vevőszám és vevő kiszolgálási idejének modellezése
  - > idősorok elemzése

Multidiszciplináris terület

matematika	lineáris algebra, mátrixalgebra, optimalizáció, statisztika
informatika	adatgyűjtés, adattisztítás, gépi tanuló algoritmusok használata
egyéb	bioinformatika, számítógépes nyelvészet, társadalomtudományok

Adatok reprezentációja I.

- > Vektor-tér modell
  - > egy megfigyeléshez egy vektor tartozik
  - > a megfigyelt tér tulajdonságainak(feature-ök) száma adja a vektorok dimenzióját
  - > pl. spamdetekció esetén egy emailhez egy vektor tartozik
  - > A vektor dimenziója választható

Adatok reprezentációja II.

- Idősor
  - természetes rendezést adó dimenzió szerinti adatok
  - pl. napi csapadékmennyiség 2014-ben a 11. kerületben
- Gráf
  - egymással valamilyen kapcsolatban álló adatok (pl. oksági kapcsolat)
  - pl. betegségek kialakulásának modellezése
  - pl. Über utazások

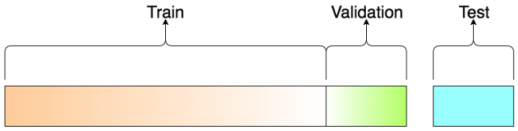
A helyes reprezentáció választása kulcsfontosságú.

Adatok vektoros reprezentációja

- minta (sample): egy elem reprezentációja - vektor
- jegy (feature): egyetlen jellemző - a vektor egy eleme. Bármi lehet egy feature, pl. email esetén:
  - email hossza
  - email feladója
  - tartalmazza-e a „Rolex" szót?
  - tartalmazza-e a „Trust fund" kifejezést?
- címke (label): „válasz"

Tanitó, Validációs és Teszthalmaz

- tanítóhalmaz (training dataset): tanításra használt minták - mátrix
- validációs halmaz: paraméterek beállítására, illetve a tanítás korai leállítására használt halmaz. Tanitás közben mérhető vele az algoritmus teljesítménye
- teszthalmaz: tanítás utáni tesztelésre használt halmaz. Csak egyszer futtatható rajta az algoritmus



Adatok típusai

1. címkézett vs. címkézetlen
   - címkézett
     - tudjuk a jó választ
     - pl. filmek értékelései
     - sokszor kevés adatunk van, előállítása drága
     - felügyelt tanulás
   - címkézetlen
     - nem tudjuk a jó választ
     - pl. internetes kommentek
     - ebből általában jóval több van
     - felügyeletlen tanulás
2. folytonos vs. diszkrét

Adatbányászat folyamata

1. Adattisztítás
   - Zajszűrés, outlier szűrés
   - Hiányzó adatok kezelése
2. Mintavételezés (szükség esetén)
   - Ha túl sok az adat
3. Adattranszformáció
   - Dimenziócsökkentés (egyre kevésbé)
   - Normalizálás, standardizálás
4. Adatbányászati algoritmus futtatása
5. Kiértékelés

Felügyelt és felügyeletlen tanulás

- A felügyelt tanulás
  - Olyan probléma, ahol a bemeneti adathoz(x) tartozik egy kimeneti változó(y)
  - A kimeneti adatok elkészitése általában emberi erőforrások bevonásával történik - Cimkézett adat
- A felügyeletlen tanulás
  - A probléma során csak a bemeneti változó(x) áll rendelkezésünkre
  - Hasznos, mivel a Cimkézett adat drága

Feladattípusok

- Osztályozás (O) – felügyelt, diszkrét osztályok
  - Elemek besorolása előre meghatározott halmazokba
  - pl. spam vagy ham egy levél
- Regresszió (R) – felügyelt, folytonos célváltozó
  - Elemek attribútumai közti kapcsolatok modellezése
  - pl. lakásárak előrejelzése méret, elhelyezkedést stb. alapján
- Klaszterezés (K) – felügyeleletlen, diszkrét klaszterek
  - Elemek csoportosítása valamilyen hasonlósági metrika szerint
  - Cél: azonos klaszterben lévő elemek „hasonlítsanak" egymásra, eltérő klaszterben lévők minél kevébé.
  - pl. marketszegmentálás

Tipikus algoritmusok

- Support vector machine (Osztályozás, Regresszió)
- Decision Tree (Osztályozás, Regresszió)
- Random Forest (Osztályozás, Regresszió)
- Logistic Regression (Osztályozás)
- Linear Regression (Regresszió)
- Neural Networks (Osztályozás, Regresszió)
- K-means (Klaszterezés)
- LDA (Klaszterezés)

Tovább olvasható: https://scikit-learn.org/stable/

Kiértékelés

- Osztályozás
  - confusion matrix alapján:
  - pontosság (precision), fedés (recall), F-mérték (F-score)
- Regresszió
  - átlagos négyzetes hiba (RMSE), abszolút hiba
- Klaszterezés
  - nagyon nehéz kiértékelni
  - klaszterek minősége (mennyire tömör, milyen távol vannak egymástól)

7. Ismertesse, milyen célokra használunk statisztikai szoftvereket. Milyen funkciókat biztosítanak az ilyen programok? Milyen feladatokat lát el egy "data scientist"?

Statisztikai szoftverek

- Statisztikai és numerikus számításokhoz készült szoftverek
- Saját script nyelvvel rendelkeznek
  - Speciális adattípusok és adatreprezentáció a matematikai műveletekhez
    - Pl. vektor, mátrix (≠tömb)
  - Alapvető nyelvi elemek a matematikai műveletek
    - Pl. mátrix szorzás
  - Adat vizualizációt is támogatnak

Statisztikai szoftverek

- Miért van rá szükség?
  - Általános programozási nyelv (Java, C++, C#) nem alkalmas matematikai műveletekre
    - Kevés matematikai művelet
    - Adatkezelése nem illeszkedik a feladatokhoz
    - Nincs szükség objektum orientáltságra
    - Numerikus pontosság nem megfelelő
  - Gyakori funkciókra beépített megoldások
    - Pl. CSV fájl beolvasása
  - Prototípus implementációra is használható
    - Gyorsabb kipróbálni egy ötletet

The R Project for Statistical Computing

- Ingyenes szoftver
- Windows, Linux, MacOS
- https://www.r-project.org
- R nyelv
  - > Script nyelv
  - > Procedurális
  - > S nyelvre épül, továbbfejlesztve
- Írhatók függvények, osztály szerűségek
  - > Nem OO
- Alap eszközkészlet mellett sok *package*
  - > Funkciók kiterjesztése

The R Project for Statistical Computing

- R egy környezet
  - > R kódot képes futtatni
  - > Kezeli a változókat
    - – Session szerű
    - – Változó addig él, amíg a session él, vagy a változó explicit törlésre nem kerül
  - > Alap eszközkészlet funkcióit adja
    - – Függvények, adattípusok
  - > Programozás
    - – Bármilyen szövegszerkesztő
    - – Konzol

Data scientist / data engineer

- Feladatai
  - > Adat feldolgozás, vizualizáció
  - > A/B tesztelés
  - > Trendek, összefüggések felismerése és verifikálása
- Szoftver fejlesztőként
  - > Analitikus gondolkodás
  - > Szoftver eszközök ismerete
    - – SQL, R, Python, Hadoop, ...
  - > Nem feltétel a mély matematikai ismeret
    - – Ha most kezdenek ezzel foglalkozni, egyszerű eredmények is eredmények
- Kutatás-fejlesztés
  - > Adat feldolgozás, elemzés mint termék
  - > Matematika, statisztika, adatbányászat, szöveg- és kép feldolgozás

8. Ismertesse a „big data 3V”-jét. Mutasson példákat, ahol nagy adat keletkezik, és ahol annak feldolgozása kihívás. Ismertesse az átalános big data stacket.

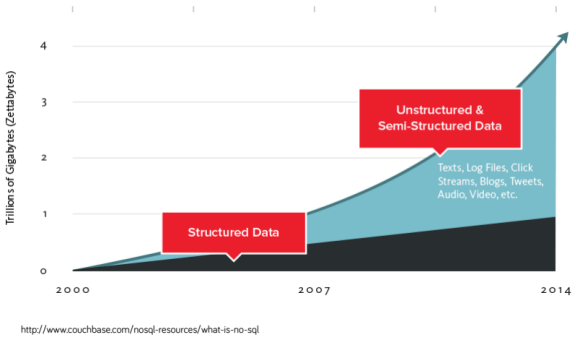
3Vs of big data

Volume

Velocity

Variety

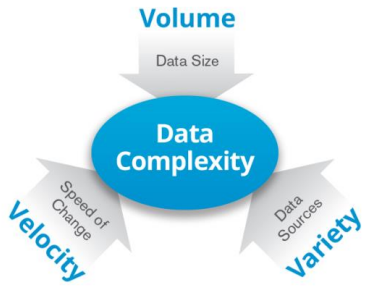
3Vs of big data



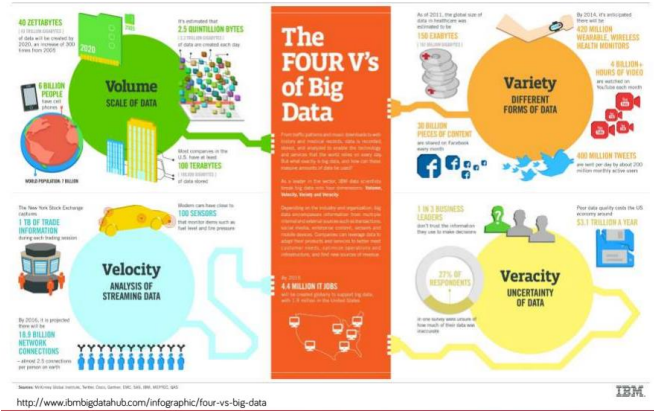
http://www.couchbase.com/nosql-resources/what-is-no-sql

A Big Data fogalma

Olyan méretű komplex adathalmaz, amit hagyományos technológiákkal nem tudunk kezelni.



3(+1)Vs of big data

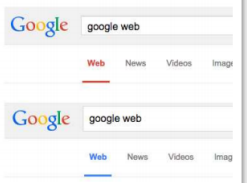


Honnan származik az adat?

- „Digitális lábnyom”
  - > Aktivitáshoz kötött
    - Megvásárolt termék, megnézett weboldal
  - > Beszélgetések
    - Facebook, Twitter, Skype, stb.
  - > Fotó, videó
  - > Szenzor
    - GPS, RFID
  - > IoT
    - Minden, ami eddig nem volt az internetre kötve
    - Pulzus mérő, tévé, hűtő, ébresztőóra, lámpa izzó, …

Mire használható az adat?

- Adat -> információ -> üzleti előny
- Data-driven decision
- Testreszabott felhasználói élmény és szolgáltatások
- Csalás detektálás
- Üzleti folyamatok optimalizálása
- Egészségügy
- Sport
- Smart city



Big data mögötti mozgatórugók

- Olcsó tárolás
  - > Hardver költségek csökkenése
  - > Szalagos tárolás helyett HDD
- Költség csökkentés
  - > Nagy, drága, komplex rendszerek helyett olcsóbb hardver és szoftver
- Felhő
  - > „… as a service”
  - > Adat feldolgozás is szolgáltatás

Mit jelent számunkra?

- Adat --?--> információ
  - > Hogyan
    - Rögzítsük?
    - Továbbítsuk?
    - Tároljuk?
    - Dolgozzuk fel?
    - Használjuk fel?
  - > Új fajta eszközök, technikák, megoldások

Web2

- Amazon
  - > Személyes felhasználói élmény
    - – Termék ajánlás
  - > Felhasználói szokások
    - – Raktárkészlet tervezés
- Facebook
  - > Pl. statisztikák, spam elemzés
  - > Hive

Big data alkalmazási területek

Orvosi diagnosztika

- Viselhető eszközök
  - > Pulzus mérő, vérnyomás mérő
- Új fajta diagnosztika
  - > Nem csak egy-egy vizsgálat során mért adat
  - > Hosszú távú mérések
- Fertőzések terjedésének vizsgálata

Biztonság

- PRISM
- Biztonsági kockázatok elemzése
- Betörésre utaló minták
  - > Bankkártya használat
  - > Rendszer használat

Internet of Things (IoT)

- Fizikai világ összekötése, digitalizálása
  - > Hozzáférés információhoz
  - > Monitorozás
  - > Beavatkozás

Szenzor

„Smart”

Ubiquitous computing

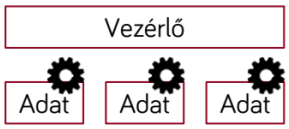
Big data eszközök

Data lake

- Nagy adat tároló és feldolgozó
- Költséghatékony
  - > Adatmennyiséggel skálázódik
- NEM adattárház, nem strukturált
  - > Inkább pl. HDFS
- Sok adatforrás, változatos adattípusok
  - > Batch export, streaming
- Pl. a Hadoop ökoszisztéma

Big data eszközök alapja

- Adat tárolás elosztottan
- Masszívan párhuzamos feldolgozás
  - > Az adathoz közel
- Központi vezérlő
  - > Egységes rendszerkép



Big data stack

- Megjelenítés
- Feldolgozó infrastruktúra
  - > Elosztott, párhuzamos
- Adat
  - > Nem igazi funkcionális réteg, de elkülönül a többitől
  - > Feldolgozás és a platform között mozog
- Platform infrastruktúra
  - > Elosztott, redundáns, meghibásodásra nem érzékeny
- Adat tárolás
- Fizikai infrastruktúra

Big data stack - Infrastruktúra

Megjelenítés
Feldolgozó
Adat
Platform
Tárolás
Infrastruktúra

- Szempontok
  - > Teljesítmény
  - > Rendelkezésre állás
  - > Skálázhatóság
  - > Költség
- Relatív olcsó és megbízható hardver
  - > „Commodity" hardver
    - – Piacon elérhető termékek
    - – Több gyártótól
    - – Jó ár-érték arány
  - > Klaszterek

Big data stack - Tárolás

Megjelenítés
Feldolgozó
Adat
Platform
Tárolás
Infrastruktúra

- Nagy mennyiségű adat
- Műveletek
  - > Írás: új adat hozzáadása
    - – Ritka, de akkor nagy mennyiség
  - > Adat olvasás
    - – Szekvenciális, elejétől a végéig
- Diszken tárolt
  - > Elosztottan, redundánsan
  - > Flat file, NoSQL, relációs adatbázis, ...

Big data stack - Platform

Megjelenítés
Feldolgozó
Adat
Platform
Tárolás
Infrastruktúra

- Számítógép klaszter egy egységként
- Kezeli a klaszterbe tartozó csomópontokat
- Redundáns, megbízható, meghibásodásra nem érzékeny
- Ha egy kiesik, átadja a feladatot másnak

Big data stack - Adat

Megjelenítés
Feldolgozó
Adat
Platform
Tárolás
Infrastruktúra

- Nem funkcionális réteg
- Mégis elkülönül a többi rétegről
  - > Pl. RDBMS: adatbázis = tárolás + platform + adat + feldolgozás egyben
- A platform réteg szolgáltatja az adatot a feldolgozó rétegnek
  - > Nem feltétlenül az adat mozgatásával
  - > Inkább: a feldolgozás megy az adathoz
- A rendszerben nem egy féle adat él
  - > Ugyanazon rendszer sok független adat, feladat feldolgozására képes

Big data stack - Feldolgozás

Megjelenítés
Feldolgozó
Adat
Platform
Tárolás
Infrastruktúra

- Elosztott rendszer feletti algoritmusok
- A platform szolgáltatásaira épülve
- Beadott feladat végrehajtásáért felelős
  - > Időzíti, futtatja
  - > Elosztja a platform gépei között
  - > Szervezi a feladat szétbontását, a részeredmények összefésülését

Big data stack - Megjelenítés

Megjelenítés
Feldolgozó
Adat
Platform
Tárolás
Infrastruktúra

- Rendszer egésze feletti menedzsment
- Feladatok beadása, státusz ellenőrzés
- Eredményekhez hozzáférés, vizualizáció

Eszközök

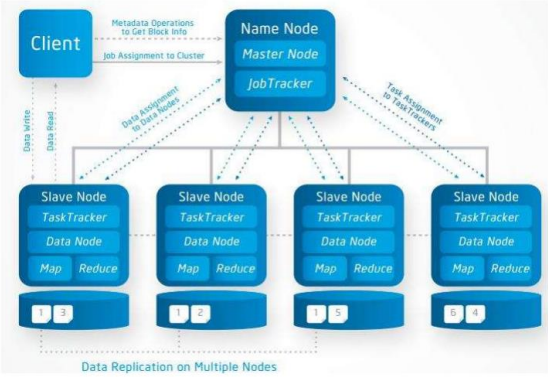
- Hadoop
  - > Hive, Pig, Impala, ...
- Spark

9. Mutassa be a Hadoop-ot. Térjen ki az architektúrájára, skálázhatóságára, az adattárolási megoldására, a feldolgozási modelljére. Ismertessen olyan Hadoop komponenseket, amelyek az alap rendszerre épülnek kiegészítve azok működését.

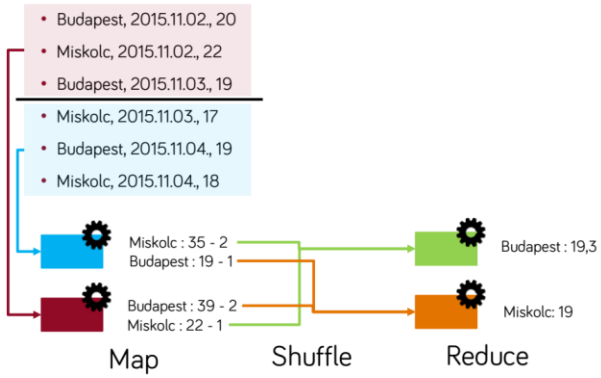
Hadoop

- Apache projekt
- Open source
- Master / slave modell
- Komponensei
  - > HDFS
    - – Elosztott fájlrendszer
  - > MapReduce
    - – Elosztott adatfeldolgozási és programozási modell
  - > YARN
    - – Erőforrás kezelő -> klaszter csomópontok kezelése

Hadoop



Hadoop - MapReduce



Hadoop - MapReduce

- Programozási paradigma
- Masszívan párhuzamos feldolgozás

Hadoop – Pig, Hive, Impala

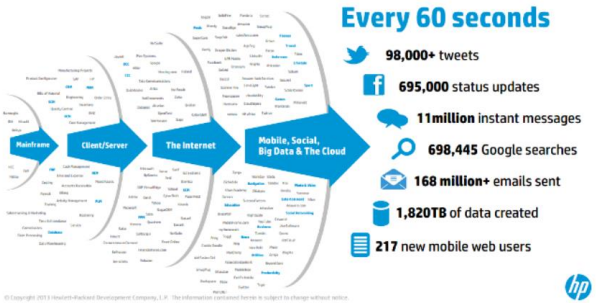
- Hadoop-ra épülnek
  - > Script nyelvek
  - > Java kód helyett magasabb absztrakciós szint
  - > Adat áll a központban
  - > Nem a végrehajtás (imperatív), hanem az eredmény (deklaratív) leírása
- Absztrakció
  - > Pl. tábla, join
- UDF támogatás
- Pig, Hive
  - > Pig: ETL
  - > Hive: SQL szerű lekérdezések
  - > Map-reduce jobokká fordul
- Impala
  - > SQL szerű, kis késleltetésű
  - > Saját futató engine (nem MR)

Spark

- MapReduce feldolgozási séma kiváltása
  - > Több lépéses feldolgozás
  - > Memóriában tárolt adatok
  - > Nagyságrendekkel jobb teljesítmény
- Hadoop infrastruktúra felett képes futni
- Java, Python, Scala, R integrációs lehetőség
- Spark Streaming, Spark SQL, Spark MLlib

A Big Data fogalma

„Az összes adat 90%-át az elmúlt két évben generáltuk."



Big Data – elosztott rendszer

- Big Data problémákra megoldás az elosztott rendszerek használata. Miért?

- Hagyományos „Scale-up" nézet, 1 erős gép:
  - > 12-24 TB tárhely
  - > 32-64 GB RAM

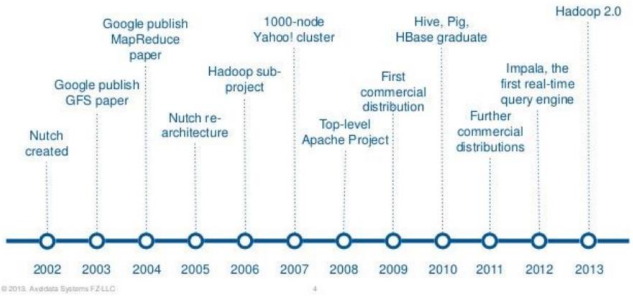
- De! 2 TB-os merevlemez tartalmának leolvasása ~6 óra

Big Data – elosztott rendszer

- Sok gép dolgozik egy közös feladaton – „Scale out"
  - > Előző dia merevlemezes példa:
    - Elosztott adattárolás -> sokkal gyorsabb írás és kiolvasás
  - > Hasonló gyorsulás igaz számításokra is

- Hibatűrés, 1-2 gép kiesése nem okoz gondot:
  - > Replikáció az adattárolásnál
  - > Redundáns szerepek a gépek közt

Hadoop - történet



Hadoop - történet

- Doug Cutting és Mike Cafarella
- Nutch nevű keresőmotorhoz fejlesztették a Yahoo! -nál
- Google által kiadott cikkek alapján
  - > GFS – Google File System
  - > MapReduce

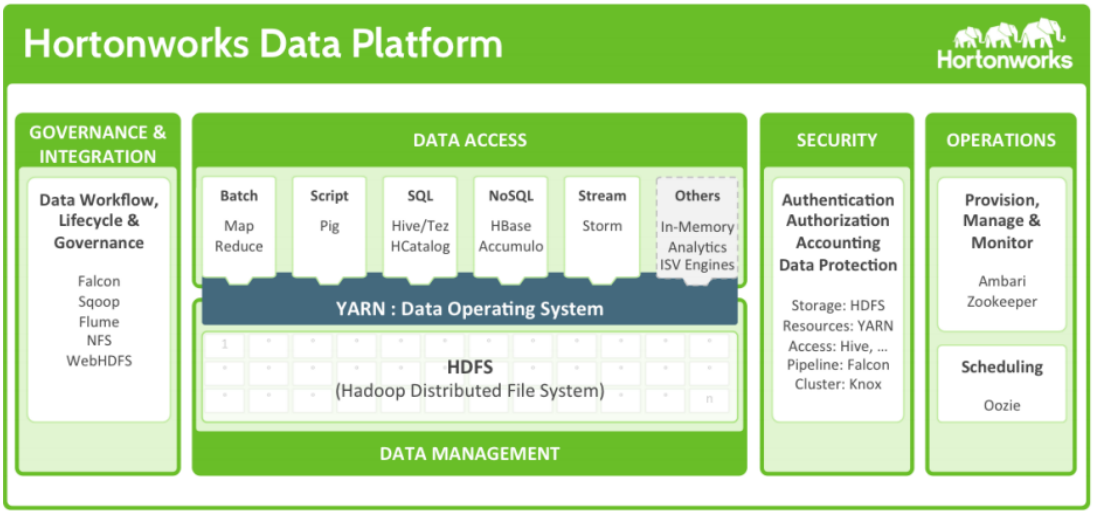
- Doug Cutting, fiának játékelefántjáról nevezte el

Hadoop - disztribúciók

- Szolgáltatások sokaságát kell telepíteni és összehangolni
- Disztribúciók

cloudera MAPR Hortonworks Pivotal

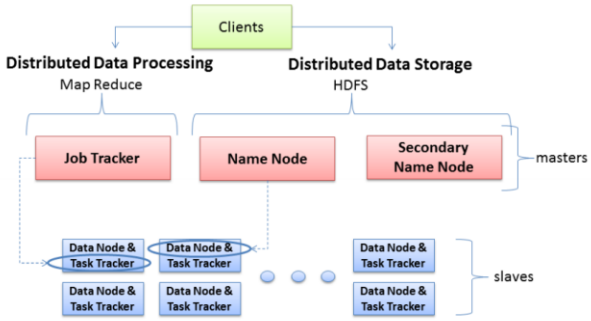
Hadoop - komponensek



Hadoop - HDFS

- HDFS (Hadoop Distributed File System)
- Hibatűrő, blokk alapú, elosztott fájlrendszer
  - > Blokkok replikálása – többszörös tárolás
    - Default replikációs faktor: 3
  - > Nagyméretű fájlok tárolása
    - 64-256 MB méretű blokkok (merevlemez elérési idő)
  - > Átlagos PC-ken futhat
    - Költséghatékony

Hadoop - HDFS



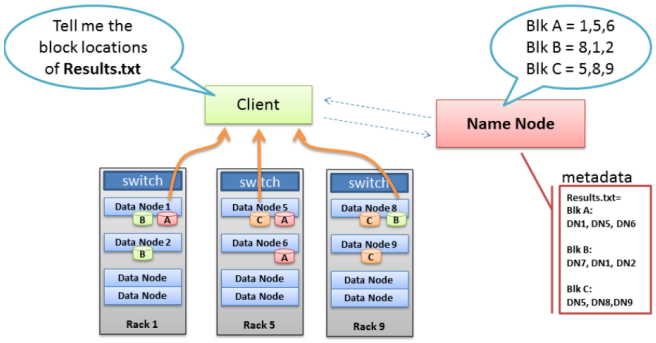
Hadoop - HDFS

- Name Node (NN)
  - > Általában 1 db / klaszter
  - > Metaadatok tárolása
    - Namespace image (NSI) – lenyomat a fájlokról
    - Edit log – legutóbbi NSI-hez képest történt változások, időnként hozzáfűzés
  - > Data Node-ok heartbeat-jeinek figyelése
  - > SPOF – ha NN kiesik nincs HDFS, ha nincs biztonsági másolat minden adatunk olvashatatlanná válik, noha fizikailag megvannak
    - RAID
    - Name Node High Availability – több NN, extra teher

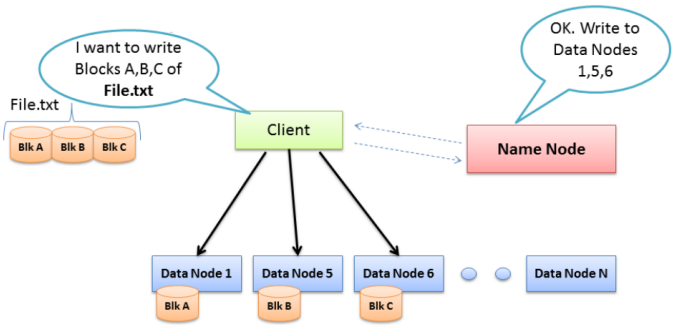
Hadoop - HDFS

- Secondary Name Node (SNN)
  - > NEM tartalék NN
  - > Feladata
    - NN segítése a NSI karbantartásában
    - NSI backup
    - Edit log belefűzése a NSI-be
- Data Node (DN)
  - > Fájlok blokkjainak tárolása
  - > Report a NN felé
    - Élek-e még
    - Milyen blokkok vannak rajtam

Hadoop – HDFS olvasás



Hadoop – HDFS írás



Hadoop – YARN / MRv2

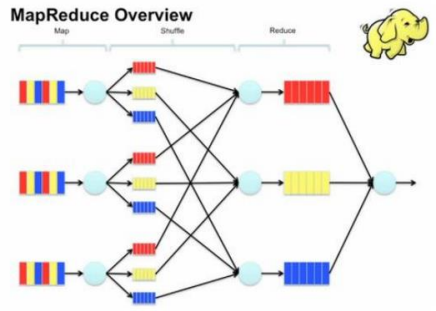
- Yet Another Resource Manager
  - Hadoop 2.0
  - Data Operating System
  - Erőforráskezelés
- Konténerek foglalása az alkalmazások számára
  - Ezek méretezése
  - Életciklus kezelés

Hadoop – YARN / MRv2

- Map Reduce
  - Programozási paradigma, nyelv független
  - Google tette népszerűvé, a kereső index generálására fejlesztették ki
  - Kulcs – érték párokkal dolgozik
  - 2 fő, 3 lépés összesen
    - **Map**: a worker node-ok végrehajtják a saját adathalmazukon a map() függvényt
    - **Shuffle**: adatok csoportosítása kulcs szerint és küldés a reduce-er ekhez
    - **Reduce**: reduce függvény végrehajtása kulcsonként, párhuzamosan
  - Map fázis: (k1, v1) -> list(k2, v2)
  - Reduce fázis: (k2, list(v2)) -> list(k3, v3)

Hadoop – YARN / MRv2

- Példa – wordcount
  - Map fázis: (sorszám, sor) -> list(szó, 1)
  - Reduce fázis: (szó, list(1)) -> (szó, szószám)



Hadoop – Hive

- Hadoop-hoz készült adattárház megoldás
- Nagy adathalmazok kezelése
  - Táblák létrehozása
    - Séma kényszerítése fájlokra olvasási időben
    - Alapvetően text fájlok, de más is támogatott
  - Lekérdezések
  - Majdnem SQL kompatibilis (HiveQL)
  - HiveQL scriptek MapReduce / Spark jobokká fordulnak
- Lassú (Hadoop alapvetően batch jellegű)
  - Gyorsabb lekérdezésekhez Impala

Hadoop – Impala

- Hive-hoz hasonló adatbázis motor Hadoop-ra
- Saját feldolgozó motor, nem MR job alapú -> Hive-nál gyorsabb
- Hive-al közös metastore
  - Hive-al definiált táblákon tud dolgozni és fordítva

Hadoop – Spark

- Általános végrehajtó motor elosztott adatfeldolgozáshoz
- MR utódja?
  - Egyes esetekben akár 100-szor gyorsabb
- Java, Scala, Python, R támogatás
- Könyvtárak
  - Spark Streaming – microbatch alapú realtime feldolgozás
  - MLib – gépi tanulási könyvtár
- Sokféleképp futhat
  - YARN fölött
  - Standalone módban

Hadoop – Spark

- RDD – Resilient Distributed Dataset
  - Elosztott párhuzamosan feldolgozható adathalmaz
  - Létrehozása
    - sparkContext.parallelize(collection)
    - sparkContext.textFile("/hdfs/path/to/my/file")
  - Kétféle művelet
    - Transformation – új RDD-t hoz létre, pl map
    - Action – egy értéket kiszámít az RDD elemein és azzal tér vissza, pl count
- Lazy végrehajtás
  - Transzformációkat csak egy akció hívásakor hajtja végre

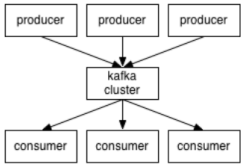
Hadoop – Oozie

- Workflow készítő és ütemező eszköz, ideális ETL-ek összeállításához
- DAG ként definiálhatunk folyamatokat
- Időközönként vagy új adat érkezésekor futhatnak
- Lehetséges task-ok
  - Spark job
  - Hive script
  - HDFS fájlművelet
  - Java kód
  - Sqoop művelet
  - MapReduce job
  - ...

Hadoop – Kafka

- Elosztott publish – subscribe üzenetküldő rendszer
- Gyors
  - Egy Kafka bróker több ezer klienst tud kiszolgálni, másodpercenkénti több száz megabájtos forgalommal
- Megbízható
  - Az üzeneteket lemezre menti és replikálja
- Skálázható
  - Az adatfolyamokat szükség esetén elosztja node-ok közt

Hadoop – Kafka



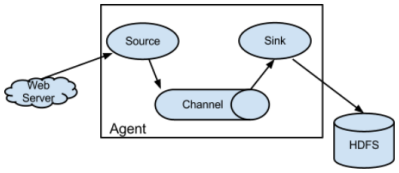
- Producer
  - adatokat küld a Kafka-nak, általa megjelelölt topic-ba
- Topic
  - az üzenetek topic-okra vannak osztva, topic-ba lehet üzenetet küldeni és fel lehet rájuk iratkozni
- Consumer
  - feliratkozik topic(ok)ra és kiolvassa az üzeneteket
- Broker
  - Kafka instance, elosztott módban több is van

Hadoop – Kafka

- Use case-ek:
  - Üzenetsor
    - Kafka-nak nagyobb a troughput-ja
    - Elosztott, megbízható -> ideális nagyméretű üzenet feldolgozó alkalmazásokhoz
  - Web aktivitás gyűjtés
    - Erre készült a LinkedIn-nél
    - UI-on történő kattintások események rögzítése
    - Topic minden aktivitás típusra
    - Nagyon nagy, gyors adattömeg -> képes kezelni
  - Log gyűjtés
    - Flume helyett használható (vagy kombinálva, lásd később)
    - Nem fájl hanem esemény szinten megy a feldolgozás
    - Flume-hoz képest kisebb késleltetés, nagyobb megbízhatóság

Hadoop – Flume

- Elosztott szolgáltatás nagymennyiségű adat gyűjtésére
  - Eredetileg szerver logok gyűjtésére és HDFS-re mentésére
- Megbízható - nem veszik el adat Flume-on belül



- Testreszabható források, csatorna és nyelők
  - Külső forrásból érkezik adat a Source-ba
  - Channel tárolja, amíg a Sink-en át ki nem kerül a rendszerből

Hadoop – Flume

- Források
  - HTTP (pl.: JSON)
  - JMS
  - Kafka
  - Avro
  - ...
- Csatornák
  - Memory
  - JDBC
  - File
  - ...
- Nyelők
  - HDFS
  - Hive
  - Kafka
  - Cassandra
  - ...

- A rendszerben lévő adat alapegysége az Event
  - Header + némi adat, kis méretű
- Agent
  - A Flume-beli adatfolyam kezeléséért felel
  - Egy agent több source-ot, channel-t és sink-et is kezelhet

Hadoop – Flume

- Interceptorok
  - Ráülnek az adatfolyamra és minden Event-re lefutnak
    - Transzformáció
    - Filterezés
    - Adminisztráció

Hadoop – Flume példa

- Példa konfiguráció
  - HTTP forrás
  - JSON adat fog érkezni
  - HDFS nyelő
  - JSON key és dátum szerint rendezzük mappákba

Hadoop – Sqoop

- Eszköz adatok mozgatására különböző források között
  - Struktúrált források
    - RDBMS
  - Nem struktúrált
    - HDFS
    - Cassandra
    - Hbase
- MapReduce alapú
- Párhuzamosan dolgozik több Data Node-on
  - Felosztja az adatokat

Hadoop – Sqoop

- Leggyakoribb use-case:
  - SQL adatbázisbeli adattömeg betöltése HDFS-re
- Példa
  - sqoop import --connect
    jdbc:mysql://impala.aut.bme.hu:3306/demo_db
    --username <uname>
    --password <pass>
    --table demo
    --target-dir /user/tomosvari/sqoopimport

Hadoop – Solr

- Enterprise search szerver (json, xml, csv, bináris)
- Apache Lucene engine-re épít
- REST API
- Szöveges adatok indexelése és kereshetősége
- Elosztott architektúra
- Spark alapú indexelés

Hadoop – Avro

- Szerializálási framework, formátum
  - Bináris formátum
  - Sémákon alapszik
    - A sémát az adattal együtt tárolja a fájlban (bármikor, bármi olvashatja)
    - JSON formátumú sémadefiníciók
  - Kódgenerálás nem szükséges
- Vetélytársak
  - Thrift (Apache)
  - Protocol Buffers (Google)

Hadoop – Zookeeper

- Service repository
- Nyílvántartja az elosztott rendszerek és szolgáltatások elérhetőségeit neveit

10. Ismertesse a tény-dimenzió modellezés alapelveit. Adjon példát tény és dimenzió táblára. Milyen előnyei vannak egy csillag sémának? Mit nevezünk degenerált dimenziónak? Mit nevezünk aggregációnak, adjon példát.

A ténytáblák és a dimenziótáblák olyan oszlopokat tartalmaznak, amelyek a modell adatait tárolják:

- A ténytáblák mérőszámokat tartalmaznak, amelyek olyan oszlopok, melyek definícióiba összesítések vannak beépítve. Például az Árbevétel és az Egységek mérőszámoszlopok.
- A dimenziótáblák olyan attribútumokat tartalmaznak, amelyek leírják az üzleti egységeket. Például a Vevő neve, Régió és Cím attribútumoszlopok.

A ténytáblák és a dimenziótáblák az üzletmenete olyan szempontjait képviselik, amelyeket szeretne jobban megismerni.

Tény táblák

A tény táblák (fact tables) a dimenzionális modellezés központi elemei: ezek azok a táblák, ahol az adott üzleti folyamat számszerű mértékei szerepelnek. Például egy üzletlánc napi eladásait reprezentáló táblában ez a mérték lehet az eladott mennyiség, vagy az értük kapott pénzösszeg. Minden nap, bármelyik boltban bármelyik termék értékesítésre kerül, készül egy bejegyzés is. A dimenziók ezen listája határozza meg a tény tábla finomságát, felbontását.

Egy adattárház szempontjából a leghasznosabb mértékek számszerűek és összeadhatóak, mivel igen ritka az az eset, amikor egyetlen sorra kíváncsi a felhasználó a tény táblából. Éppen ellenkezőleg, általában a sorok százezreinek az aggregált értékére kíváncsi (az elmúlt hónapban eladott termékek mennyisége, bevétel, stb.). A fenti példában az eladott mennyiség és a pénzösszeg is összeadható bármelyik dimenzió mentén.

Nem minden tényadat összeadható, léteznek részlegesen összeadható (semiadditive) mértékek, amelyeket csak bizonyos dimenziók mentén lehet összeadni, és nem összeadható mértékek is. Például egy raktárkészlet aktuális állapotát vagy számlák aktuális egyenlegét reprezentáló tény táblák tipikusan ilyen részlegesen összeadható adatokat tartalmaznak, ugyanis értelmes összeadni a számlaegyenlegeket például ügyfelek szerint, de értelmetlen az idő szerint. Ilyen esetekben a legcélszerűbb megközelítés az átlagolás: az adott periódusra szóló átlag-egyenleg, vagy átlagos raktárkészlet.

Dimenziók

A dimenziók a tény táblák kísérői. Ezek a táblák tartalmazzák a szöveges leírásait az adott üzleti folyamatnak. Egy jól megtervezett dimenzionális modellben egy dimenzió táblának lehető legmagasabb számú oszlopa vagy másnéven attribútuma van, ugyanis ezek az attribútumok játszanak a lekérdezéseknél, elemzéseknél csoportosító, megszorító, vagy magyarázó szerepeket. Emiatt létfontosságú, hogy minél több jól definiált, értelmes dimenzió attribútum legyen, mert ezek határozzák meg az adattárház használhatóságát. A dimenziók jelentik az interfészt az adattárház és a felhasználó között.

Míg a tény adatok főleg számszerűek és folytonos értékkészletűek, addig a dimenzió attribútumok általában szövegesek, és diszkrétek.

A dimenziók sokszor hierarchikus kapcsolatot reprezentálnak. Például egy termék egy adott márkához tartozik, amiket kategóriákba sorolunk, és így tovább. A termék dimenzió táblában minden sorban (minden termékre) eltároljuk az adott termék márkáját és a kategória szöveges jellemzését is. Ez épp ellentétes egy normalizált adatbázissal, ugyanis rengeteg redundáns információt tartalmaz. A dimenzió táblák tipikusan denormalizáltak (kivéve snowflake séma esetében), a performancia és az egyszerűség, könnyen érthetőség érdekében feláldozzák a szükséges tárhely mennyiségét.

Csillag séma

- Tény és dimenzió táblák
- A lekérdezések legtöbb esetben *JOIN* műveletek a tény és dimenzió táblák között
- Dimenziós táblákat tipikusan nem kell *JOIN*-olni
- Automatikus kapcsolat felderítés
- Egyszerű kulcsok a tény és dimenzió táblákban
- A dimenzió táblák tipikusan kisebbek a tény táblákhoz képet
- A dimenzióknak tipikusan van *surrogate* kulcsuk

Surrogate key

- Előny:
  - > Tény táblák méretének csökkentése
  - > Független a forrás rendszer kulcs változásától
  - > Leírhatja az entitások időbeli változását
- Hátrány:
  - > Új kulcs oszlop a tény és dimeziós adatok betöltésekor (lassítja a betöltési folyamatokat)

Dimenzió modellezés

- Egyszerűbb lekérdezés optimalizálás
- Egyszerű a modell bővítése, nincs szükség az adatbázis átrendezésére ha új dimenzió kerül a rendszerbe
- Külsősök számára is egyszerű a lekérdezések megfogalmazása
- Nem szükséges a meglévő lekérdezések módosítása új dimenzió esetén
- A dimenzió modellezés 4 lépése:
  - > Üzleti folyamat azonosítása
  - > Tény adatok azonosítása
  - > Tény adatok granularitásának megfelelő kiválasztása
  - > Dimenzió adatok azonosítása
- Példa tény adatok: rendelések, számlák, szolgáltatás használati adatok, telco adatok stb.

Tény adatok granularitásának helyes megválasztása

- Milyen részletes adatokat tervezünk tárolni
- *Túl részletes*:
  - > Túl sok adat
  - > Túl nagy tárhely követelmény
  - > Lassú lekérdezések, CPU
- *Nem eléggé részletes*:
  - > Nehéz pontos analízist készíteni
- **Fontos a tény adatok pontos jelentésének megfogalmazása és megértése**

Dimenziók és tények

- Dimenziók azonosítása:
  - > Hogyan tervezzük csoportosítani és lekérdezni a tény adatokat?
  - > A dimenziók növelik a report-ok részletességét.
  - > Tipikusan szöveges értékek, számok, idő, hely, stb. adatok.
  - > „Degenerated dimension”: A dimenziós adat valójában csak egy oszlop ezért a tény táblában található (például számla száma).
- Tények azonosítása:
  - > Az események, melyek a „működést” leírják
  - > A tény adatok különböző nézőpontokból vizsgálhatók
  - > Esemény-tény típus: csak egy időbélyeget tartalmaz
  - > Állapot-tény típus: általában két időbélyeget tartalmaz, az új tény létrehozásához egy előzőt be kell fejezni
    - Két időpont
    - Adatbetöltéskor többlet munkát jelenthet

Aggregációk

- Definíció: Előre kiszámított speciális lekérdezés, ahol a tény tábla adatai különböző feltételek mentén aggregálásra kerülnek.
- Tömöríti a hierarchiát a dimenziókban és összesíti a tény adatokat ez alapján (ezért fontos a tény adatok additivátása)
- Performancia szempontjából különösen fontos az aggregációk megfelelő kialakítása
- Gyakran az aggregáció dátum és idő alapján történik

Dimenziós modellezés

- **Dimenziós modellezés előnyei:**
  - lekérdezése könnyen optimalizálható
  - a modell bővítése egyszerű, nem kell átstrukturálni az adatbázist, ha bővül a modell
  - laikusok által is könnyen lekérdezhető

Négylépéses dimenziós modellezés

1. Üzleti folyamat azonosítása
2. Tényadat granularitásának megválasztása (üzleti szinten)
3. Dimenziók (és attribútumaik) azonosítása
4. Tény attribútumok azonosítása

1. Üzleti folyamat izolálása

Példák:

- szolgáltatás használata,
- hitelek igénylése és felvétele,
- bevételek alakulása,
- kinnlevőségek,
- rendelések
- személyzeti ügyek
- számlázás
- javítások és reklamációk, stb.

2. Tényadat granularitásának megválasztása

- milyen részletes adatok tárolását támogatjuk
- túl részletes: sok adat, nagy diszkigény, nagy CPU igény
- nem elég részletes: elemzéseket akadályozhat meg
- LE KELL ÍRNI A TÉNYREKORD PONTOS JELENTÉSÉT

3. Dimenziók azonosítása

- Mi alapján akarjuk rendezni, lekérdezni, csoportosítani a tényadatokat?
- Sok és részletes dimenzió változatosabb analízisek
- Dimenziók azonosítása szigorúan az adatok használata (ld. üzleti igények) alapján
- Dimenzió lesz minden, ami...
- Inkább szöveges attribútumok, de lehet numerikus is

4. Tények azonosítása

- A használandó mennyiségek konkrét meghatározása (pl. eladási ár Ft-ban, darabszám, átlagos kisker. ár, …)
- Általában folytonos értékkészletűek és numerikusak.

Dimenziós tervezési elvek

- A pontosan ismerni és érteni az adatokat
- Dimenziós táblák: leíró attribútumuk, akár 50 is, a rekordok hossza kevéssé kritikus.
- Ténytáblák: a rekordok legyenek rövidek
- Konform dimenziókban gondolkodunk
- Minden dimenziónak legyen **surrogate** (anonym, kiegészítő, jelentés nélküli, mesterséges) kulcsa.

Surrogate kulcs

Előnyei:

- méretcsökkentés a ténytáblában
- forrásrendszeri kulcs változásaitól függetlenek leszünk
- az entitások időbeli változásait is le tudjuk így írni

Hátránya:

- újra kell kulcsolni a tény és dimenziós rekordokat (jelentős betöltési többletteher)

Dimenziós tábla tervezés

- A felesleges dimenziók teljesítményveszteséget eredményeznek.
- A dimenziós adatok nem feltétlenül nyerhetők ki valamely forrásrendszerből.
- Az idő, termék, hely, ügyfél a leggyakoribb dimenziók

Ténytábla tervezés

Tényadatok a lehető legkisebb granularitásban (vö.: hiányzó "vásárlói kosár" analízis).

- Additív tényadatok
  - Hacsak lehetséges, összegezhetőnek kell választani.
- Nem additív tényadatok
  - Egyáltalán nem összegezhetők, egyetlen dimenzió mentén sem.
- Szemi-additív tényadatok
  - minden dimenzió szerint összegezhető, kivéve az időt. (általánosabban: bizonyos dimenziók szerint összegezhetők, mások szerint nem)

Dimenziós tervezési minták I.

Ténynélküli ténytáblák

- pl. diákok óralátogatási szokásai (idő, tárgy, terem, diák, tanár függvényében)
- (kampány) lefedettségi táblák
  Pl. az eladás ténye termék, bolt, idő, kampányjellemzők függvényében. Nem ad választ arra, hogy mit NEM adtak el abból, amiről a kampány szólt!
  Megoldás: egy másik ténytábla rekordja jelentse a kampányban való részvételt

tényrekord jelentése: van olyan...

Valójában klasszikus több-több kapcsolatok

Dimenziós tervezési minták II.

Állapot- és esemény-tények

- Esemény-tény: egyetlen időpont
- Állapot-tény: két időpont
  - Új tényrekord beszúrása egy másik lezárásával jár → alacsonyabb hatékonyság
  - valószínűbb információvesztés (ld. később)
- Általában egymásba átalakíthatók
  - Kik, mikor, hol, mit, stb. vásároltak
  - Kik azok a vásárlók, akiknek van …
  - Melyek azok a termékek, amelyeket eladtak…
  - …
- A lekérdezések hatékonysága erősen különböző!

Dimenziós tervezési minták III.

Role-playing dimenziók

- pl. idő, cím,... többféle jelentést is hordozhat a tényadathoz kapcsolódóan
- egyetlen fizikai dimenzió, amely több idegen kulccsal kapcsolódik a tényrekordhoz

Degenerált dimenziók

Számla, tételekkel. A tételek lesznek a tényadatok.

Mi legyen a számlaszámmal?

- Vannak olyan leíró (rövid, dimenziós jellegű) adatok, amelyeket a ténytáblában helyezünk el kapcsolódó dimenzió nélkül.
- Pl.: dokumentum egyedi azonosító száma
- A forrásrendszerben lehet könnyen azonosítani velük valamit
- Egyedi megfontolás. Normálisak, várhatók, hasznosak

Junk dimenziók

- Flag-ek és szöveges leírók nem mindig szervezhetők értelmes dimenziókba
- Ténytáblában nem célszerű elhelyezni
- Egy vagy néhány jelentés nélküli dimenziót alkothatnak.

Ha a dimenzió is változik idővel… (“slowly changing dimensions”, SCD)

Pl. az ügyfél elköltözik, címe megváltozik

1. régi rekord felülírása
2. “old” mező képzése a dim. táblában
3. új rekord a dim. táblában a surrogate kulcs új értékével

1. felülírás

Pl.: az ügyfelek címei változhatnak, ha elköltözik.

Ügyfél ID	Ügyfél neve	Ügyfél címe
123	Gipsz Jakab	Budapest, Tó u. 15.

1. felülírás

Ügyfél ID	Ügyfél neve	Ügyfél címe
123	Gipsz Jakab	Debrecen, Fő u. 3.

Egyszerű, de nincs history.

2. “old” mező létrehozása

Ügyfél ID	Ügyfél neve	Ügyfél címe
123	Gipsz Jakab	Budapest, Tó u. 15.

2. A jelenlegi és az előző állapot jellemzésével

Ügyfél ID	Ügyfél neve	Ügyfél előző címe	Ügyfél jelenlegi címe
123	Gipsz Jakab	Budapest, Tó u. 15.	Debrecen, Fő u. 3.

egyszerű, de korlátozottak a lehetőségei.

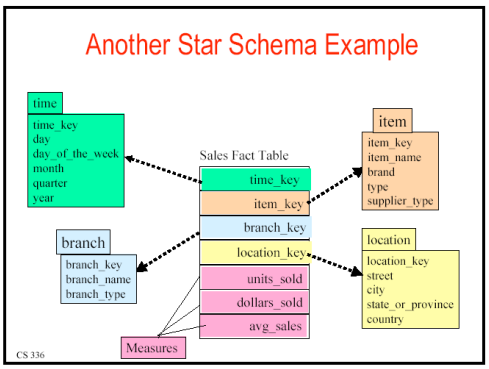
3. Új dim. rekord készítése

Ügyfél ID	Ügyfél neve	Ügyfél címe
123	Gipsz Jakab	Budapest, Tó u. 15.

3. új dimenziós rekord minden változáshoz

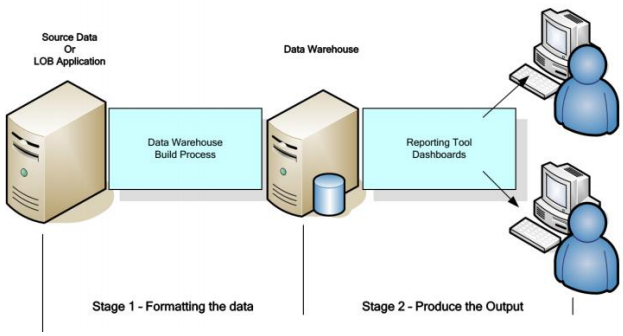
Ügyfél ID	Ügyfél neve	Ügyfél címe	Tól	Ig
123	Gipsz Jakab	Budapest, Tó u. 15.	1989. júl. 15.	2005. szept. 6.
123	Gipsz Jakab	Debrecen, Fő u. 3.	2005. szept. 7.	???????

particionálja a history-t, nehézkesebb a lekérdezés



11. Mit nevezünk ETL-nek? Mi a különbség az ETL és ELT között? Sorolja fel a tipikus ETL transzformációs feladatokat! Milyen ETL eszközöket ismer? Hasonlítson össze két választott eszközt előnyök és hátrányok összevetésével. Mit nevezünk diszkretizálásnak, binarizálásnak? Milyen adat tisztítási lépéseket ismer?

BI rendszerek maga szintű architektúrája



Adattárház (DWH) és ETL/ELT

- A DWH az ahova az adat különböző forrás rendszerekből bekerül feldolgozást követően.
- Az adatok legtöbb esetben döntés támogatás és további vizsgálatok céljául szolgálnak.
- Az adatok különböző típusú és interfészű forrásokból érkezhetnek, különböző formátumokban.
- Az adatok fogadását, formázását és megfelelő helyre való eljuttatását az ETL/ELT folyamatok végzik.
- ETL/ELT: Extract Transform Load / Extract Load Transform

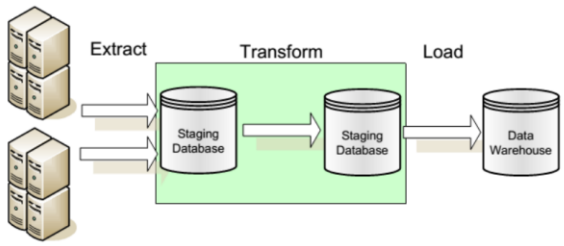
ETL

- Extract: Az adat a forrás rendszerből az állomásoztató területre kerül.
- Transform: Az adat átalakításra kerül a DWH számára megfelelő formátumra.
- Load: Az adat az állomásoztató területről a DWH-ba kerül.

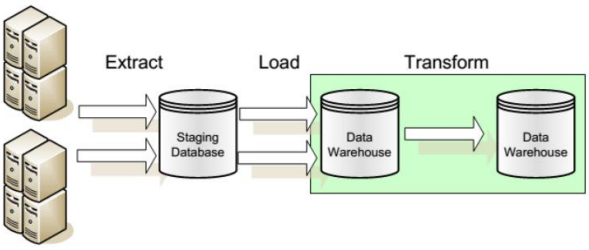
ETL vs. ELT

- ETL: adat kinyerés a külső forrás rendszerekből, feldolgozás és átalakítás (sokszor staging táblák segítségével), majd adat áttöltés a cél rendszerbe
- ELT: adat kinyerés a forrás rendszerekből, betöltés a staging táblákba, majd átalakítás és tárolás véglegesen a DWH-ba
- ELT használatának tipikus eseti:
  - Nagy mennyiségű adat esetén
  - A forrás és a cél adatbázis azonos
  - Az adatbázis motor jól támogatja a transzformációt és a nagy adat mozgatást/átalakítást
- ETL esetén a transzformációt az ETL engine végzi míg ELT esetén tipikusan a cél adatbázis (performancia kérdések)
- Az ELT esetén a forrás rendszer erőforrás igénye minimális, hátránya, hogy az adatbázis oldalra nagyobb feladat hárul, később kerül az adat a DWH végleges helyére és a staging tábláknak nagyobb lehet a helyigényük.
- További variációk:
  - ETT (Extract Transform Transport)
  - ETM (Extract Transform Move)

ETL



ELT



ETL eszközök

- Korábban az ETL feladatok manuálisan készültek SQL és egyéb scriptek/alkalmazások megírásával
  - Sok erőforrást igényel a fejlesztés
  - Nehéz a kód karbantartása
  - Nehéz az ütemezett végrehajtás kialakítása és felügyelete
- ETL eszközök: beépített támogatás a teljes ETL folyamatokra
  - Grafikus felület
  - Ütemező rendszer
  - Adat kinyerés támogatás
  - Adat tisztítás
  - Adat profilozás
  - Adat transzformáció
  - Debug támogatás
  - Adat betöltés támogatás

Népszerű ETL eszközök

Tool Name	Company Name
Informatica	Informatica Corporation
DT/Studio	Embarcadero Technologies
DataStage	IBM
Ab Initio	Ab Initio Software Corporation
Data Junction	Pervasive Software
Oracle Warehouse Builder	Oracle Corporation
Microsoft SQL Server Integration	Microsoft
TransformOnDemand	Solonde
Transformation Manager	ETL Solutions

ETL folyamat feladatai

- ETL folyamat (JOB): egy ETL feladatért felelős folyamat
- ETL rendszer telepítése és üzemeltetése
- Hogyan tervezzük a fejlesztést, workflow-t és ütemezést?
- Hogyan kapcsolódjunk a forrás rendszer(ek)hez és hogyan importáljuk az adatokat?
- Hogyan kapcsolódjunk a cél rendszerhez és hogyan töltsük be a cél rendszerbe az adatokat?
- Mappelés megvalósítása a forrás és a cél között.
- Milyen transzformációkra van szükség?
- Dimenziók betöltése.
- Hogyan felügyeljük az ETL folyamatot?
- Milyen metaadatok létrehozása szükséges az ETL folyamat során?
  - Milyen metaadatok jöhetnek szóba? (idő, sikeresség, mozgatott adat mennyisége, szűrt adat mennyisége, stb.)

ETL fogalmak 1/3

- Forrás rendszer:
  - Adatbázis
  - Alkalmazás
  - File
  - Egyéb adatforrás
- Mappelés:
  - A forrás és a cél adatok közti viszony és adatfolyam.
- Metaadat:
  - Azon adatok melyek leírják az adatstruktúrákat, objektumokat, üzleti szabályokat és folyamatokat. Például egy adatbázis séma leírása egy meta adat tárban kerül tárolásra amit scriptek készítésekor is felhasználhatnak.
  - A folyamatok során készített működési adatok.

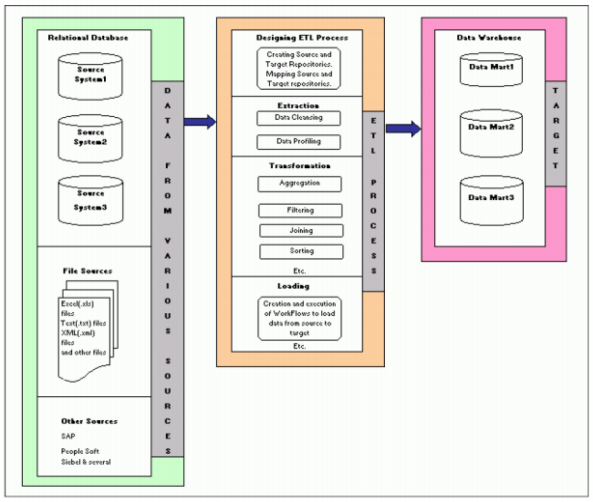
ETL fogalmak 2/3

- Állomásoztató terület (staging):
  - > Az a terület ahol az adat fogadásra és feldolgozásra kerül mielőtt bekerül az adattárházba.
- Adat tisztítás:
  - > Az a folyamat ami feloldja a forrás rendszerektől érkező adatokban levő inkonzisztenciát és esetleges anomáliákat.
- Transzformáció:
  - > Az adat manipuláció folyamata. A másoláson felül minden adat manipuláció ide számít, például adat tisztítás, transzformáció, több adatforrásból való integráció, stb.

ETL fogalmak 3/3

- Transport:
  - > A transzformált adat forrás és cél közti mozgatásának folyamata.
- Cél rendszer:
  - > Adatbázis, alkalmazás, file vagy más egyéb cél, ahova a transzformált adat betöltésre kerül.

Tipikus ETL folyamat



Tipikus ETL transzformációs feladatok

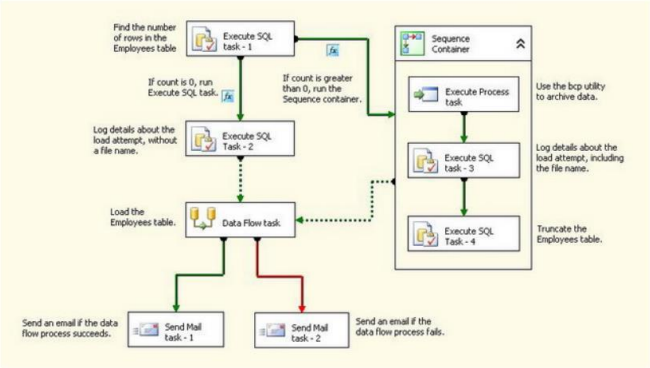
- Aggregálás
- Szűrés
- Join
- Rendezés
- Stb.

SQL Server Integration Services - Microsoft

- Enterprise adat integrációs és transzformációs megoldás
- Tipikus feladatok: másolás, letöltés, e-mail küldés események hatására, betöltés adattárházba, adat tisztítás stb.
- Több forrás és cél rendszert is támogat: flat file, XML, adatbázis stb.
- Grafikus szerkesztő felület, grafikusan szerkeszthető transzformációk

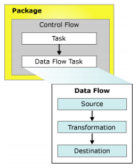
ETL eszközök

SSIS példa



SSIS package

- Az SSIS package egy egység amely magában foglal kapcsolatokat, vezérlő elemeket, adat folyam elemeket és esemény kezelőket
- *Vezérlő eleme*: tipikusan egyedi feladatok amik a folyamat során elő/utó feldolgozást végeznek vagy meta adatokat gyűjtenek
- *Adat folyam elem*: egy ETL feladat
- SQL Server szükséges a fejlesztéshez/üzemeltetéshez



Package funkció kiegészítők

- Esemény kezelő:
  - Független elemként adhatók a package-hez, futtatás előtti eseményre (pl. szabad hely ellenőrzés), hibákra (pl. e-mail a hibáról) vagy egyéb eseményekre lehet reagálni segítségükkel.
- Konfiguráció:
  - A package paraméterei (kulcs-érték párok), melyek a package telepítése után is változtathatók.
- Log:
  - A package futása során keletkező logok egyedileg kezelhetők, elmenthetők és felhasználhatók.
- Változók:
  - A package futása során változók hozhatók létre, melyek értéke dinamikusan változhat és befolyásolhatja a package futását. Rendszer és felhasználói szintű változók is használhatók. A változók használhatók kifejezésekben, scriptekben és konfigurációkban egyaránt.
- Paraméterek:
  - Projekt és package szintű paraméterek használhatók, melyek kívülről is érkezhetnek és befolyásolják a projektben levő packagek, vagy az adott package futását.

Data Integration (Kettle/Spoon) - Pentaho

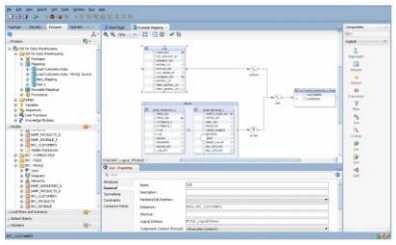
- Pentaho BI rendszer ETL megoldása
- Tipikus felhasználás: adattár feltöltés
- További felhasználási célok:
  - Adat migráció alkalmazások és adatbázisok között
  - Adat export adatbázisokból flat file-ba
  - Nagy mennyiségű adat betöltés adatbázisokba
  - Adat tisztítás
  - Alkalmazások integrálása

Data Integration jellemzői

- Grafikus felület (kód írás nélkül készíthető processek) – Spoon eszköz
- SDK támogatás, testreszabható
  - http://wiki.pentaho.com/display/EAI/Pentaho+Data+Integration+-+Java+API+Examples
- Önállóan vagy a Pentaho BI Suite részeként is futtatható
- Nyílt forráskódú
- Többféle adatforrás és adatbázis támogatás

Oracle ETL

- Két eszköz:
  - Data Warehouse Builder
  - Oracle Data Integrator



Adat tisztítás

- Adatban szereplő hibák felmérése
- Adatfile szerkezeti épségének ellenőrzése
- Hiányzó értékek feltárása
- Értékek eloszlásának vizsgálata
- Eloszlások szélein található extrém értékek vizsgálata és esetleges szűrése
- Nem várt sűrűsödések, ritkulások vizsgálata
- Hibás adatok felülvizsgálata, javítása, szűrése
- Gyakorlati példa: aninimizálás ellenőrzése

Aggregációk készítése

- Néha a „kevesebb több”
- Több objektum egy objektummá egyesítése, adat redukció
- Cél a lényeges adat tömör tárolása
- Gyorsítja a lekérdezést és az elemzéseket
- Aggregáció lehet dátum, idő és hasonló elemek mentén

- Például:
  - Hívásokat tartalmazó rekordok
  - Aggregált adat: melyik számról hány hívás volt

Normalizáció

- Adat normalizációs szint kialakítása
- Megfelelő normalizációs szinttel jobban kezelhető adatbázis hozható létre
- Egyszerűbb és hatékonyabb az adatok áttekintése
- Könnyebb lekérdezéseket megfogalmazni

Tanulónév	Általános	Középiskola
Tóth Aladár	Napsugár Általános Iskola Gyöngyvirág u. 4.	Zengő Gimnázium, Zerge u. 13. Dobogókő Szakközépiskola, Sziklás u. 44.
Kis Virág	Csillagvár Általános Iskola Kocka u. 14.	Zengő Gimnázium, Zerge u. 13. Baradla Gimnázium, Köves tér 3. Kékes Gimnázium, Havas út 51.
Alföldi Noémi	Csillagvár Általános Iskola Kocka u. 14.	Zengő Gimnázium, Zerge u. 13. Dobogókő Szakközépiskola, Sziklás u. 44.
Végső Albert	Tóparti Általános Iskola Strand út 23.	Baradla Gimnázium, Köves tér 3. Kékes Gimnázium, Havas út 51.
Tóth Aladár	Napsugár Általános Iskola Gyöngyvirág u. 4.	Zengő Gimnázium, Zerge u. 13. Dobogókő Szakközépiskola, Sziklás u. 44.

1NF

- A sorok (rekordok) sorrendje tetszőleges
- Minden oszlopnak (mezőnek) egyedi neve van
- Az egyes oszlopokban (mezőkben) azonos típusú és tulajdonságot leíró értékek vannak
- Egy cellában (mezőben) csak egy elemi tulajdonságérték szerepelhet

Tanulónév	Általános	ÁltCím	Középiskola	KözépCím
Alföldi Noémi	Csillagvár Általános Iskola	Kocka u. 14.	Zengő Gimnázium	Zerge u. 13.
Alföldi Noémi	Csillagvár Általános Iskola	Kocka u. 14.	Dobogókő Szakközépiskola	Sziklás u. 44.
Kis Virág	Csillagvár Általános Iskola	Kocka u. 14.	Zengő Gimnázium	Zerge u. 13.
Kis Virág	Csillagvár Általános Iskola	Kocka u. 14.	Baradla Gimnázium	Köves tér 3.
Kis Virág	Csillagvár Általános Iskola	Kocka u. 14.	Kékes Gimnázium	Havas út 51.
Tóth Aladár	Napsugár Általános Iskola	Gyöngyvirág u. 4.	Zengő Gimnázium	Zerge u. 13.
Tóth Aladár	Napsugár Általános Iskola	Gyöngyvirág u. 4.	Dobogókő Szakközépiskola	Sziklás u. 44.
Végső Albert	Tóparti Általános Iskola	Strand út 23.	Baradla Gimnázium	Köves tér 3.
Végső Albert	Tóparti Általános Iskola	Strand út 23.	Kékes Gimnázium	Havas út 51.

2NF

- Legyen az adatbázis első normálformulában (1NF)
- Minden másodlagos attribútum (nemkulcs mező) teljes függőségben álljon minden kulcstól

TOVÁBBTANULÁS	
TanulóAZ	KözépAZ
1	1
1	2
2	3
2	4
2	1
3	2
3	1
4	3
4	4

TANULÓK			
TanulóAZ	Tanulónév	Általános	ÁltCím
1	Alföldi Noémi	Csillagvár Általános Iskola	Kocka u. 14.
2	Kis Virág	Csillagvár Általános Iskola	Kocka u. 14.
3	Tóth Aladár	Napsugár Általános Iskola	Gyöngyvirág u. 4.
4	Végső Albert	Tóparti Általános Iskola	Strand út 23.

KÖZÉPISKOLÁK		
KözépAZ	Középiskola	KözépCím
1	Zengő Gimnázium	Zerge u. 13.
2	Dobogókő Szakközépiskola	Sziklás u. 44.
3	Kékes Gimnázium	Havas út 51.
4	Baradla Gimnázium	Köves tér 3.

3NF

- A reláció második normálformátumban van (2NF)
- Egyetlen másodlagos attribútum sem függ tranzitívan egyetlen kulcstól sem.
- ***Tranzitív függőség*** Adott egy R séma, a séman értelmezett funkcionális függőségek F halmaza, $X \subseteq R$, $A \in R$. A tranzitívan függ X-től, ha $\exists Y \subset R$, hogy $X \longrightarrow Y$, $Y \not\longrightarrow X$, $Y \longrightarrow A$ és $A \notin Y$

TOVÁBBTANULÁS	
TanulóAZ	KözépAZ
1	1
1	2
2	3
2	4
2	1
3	2
3	1
4	3
4	4

TANULÓK		
TanulóAZ	Tanulónév	ÁRAZ
1	Alföldi Noémi	1
2	Kis Virág	1
3	Tóth Aladár	2
4	Végső Albert	3

KÖZÉPISKOLÁK		
KözépAZ	Középiskola	KözépCím
1	Zengő Gimnázium	Zerge u. 13.
2	Dobogókő Szakközépiskola	Sziklás u. 44.
3	Kékes Gimnázium	Havas út 51.
4	Baradla Gimnázium	Köves tér 3.

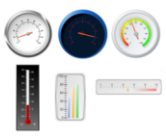
ÁLTALÁNOS ISKOLÁK		
ÁRAZ	Általános	ÁltCím
1	Csillagvár Általános Iskola	Kocka u. 14.
2	Napsugár Általános Iskola	Gyöngyvirág u. 4.
3	Tóparti Általános Iskola	Strand út 23.

Diszkretizálás és binarizálás

- Szükséges lehet, hogy az adatok kategorikus attribútumok formájában legyenek
- Gyakran szükség van a folytonos attribútumok kategorikus attribútumokká alakítására (**diszkretizálás**)
- Továbbá a folytonos és a diszkrét attribútumok átalakítása egyaránt szükséges lehet egy vagy több bináris attribútummá (**binarizálás**)

KPI kiválasztás

- KPI = Key Performance Indicator
- Olyan értékek adása a menedzsment kezébe, amelyen keresztül át tudják látni a felelősségük alá tartozó üzleti területeket, látják azok minden mozdulatát és egyből észreveszik, ha elkezdenek letérni az egyensúlyi pályáról.
- KPI jellemzője:
  - Aktuális érték, azaz hol állunk most
  - Terv érték, azaz mit terveztünk, hol fogunk állni most
  - Eltérés érték, azaz mennyivel és milyen irányban tértünk el a tervtől
  - Végül egy trend érték, ami megmutatja a tény értékek trendjét
- Komoly kihívás a KPI kiválasztása és megjelenítése



12. Milyen ETL folyamat tesztelési módszereket ismer? Ismertess az egyes tesztek elő feltételit és teljesülési kritériumait.

ETL eszközök

- Korábban az ETL feladatok manuálisan készültek SQL és egyéb scriptek/alkalmazások megírásával
  - > Sok erőforrást igényel a fejlesztés
  - > Nehéz a kód karbantartása
  - > Nehéz az ütemezett végrehajtás kialakítása és felügyelete
- ETL eszközök: beépített támogatás a teljes ETL folyamatokra
  - > Grafikus felület
  - > Ütemező rendszer
  - > Adat kinyerés támogatás
  - > Adat tisztítás
  - > Adat profilozás
  - > Adat transzformáció
  - > Debug támogatás
  - > Adat betöltés támogatás

ETL folyamatok tesztelése

- A tesztelés célja az ETL folyamatok helyes működésének ellenőrzése
- ETL teszt fázisok:
  - > Integrációs teszt
  - > Rendszer test
  - > Regressziós teszt
  - > Működés biztosítása
- Teszt stratégia: a teszt céljának egyeztetése, tesztelendő modulok, módszerek, erőforrások és környezet azonosítása.

ETL Integrációs teszt

- A teszt célja a mappeléshez szükséges előkövetelmények ellenőrzése.
- A tesztelésnek igazolnia kell, hogy megfelelő számú sor került át a forrás rendszerből a cél rendszerbe.
- Integrációs teszt során a hibakezelés, mapping változók és további üzleti igények validálhatók.
- Előkövetelmények:
  - > Hozzáférés a megfelelő hálózati könyvtárakhoz
  - > Implementáció megléte és sikeres Unit tesztek
  - > Adat elérhető a teszt környezetben

ETL rendszer teszt

- Integrálja a rendszer komponenseit és egy egységben teszteli a működést
- End-to-end rendszer teszt
- Inicializálás és inkrementális terhelési statisztikák.
- Egész rendszer performancia vizsgálata.
- Hibakezelés verifikálása
- Előkövetelemények:
  - > Fejlesztés lezárása
  - > Minden integrációs teszt sikeres lefuttatása
  - > Sikeres migrálás teszt környezetből a validációs környezetbe
  - > Termék konfigurációk és adatok elérhetősége.

ETL regressziós teszt

- A teszt az után fut le, miután egy jelentett issue javításra került.
- A teszt megvizsgálja, hogy a javítás helyes volt-e és hogy nem hozott-e más hibát a rendszerbe.
- A tesztet akkor is végre kell hajtani, ha nem egy hiba hanem egy CR (Change Request) került megvalósításra.
- Előkövetelmények:
  - > Fejlesztés lezárása.
  - > Integrációs tesztek sikeres lefuttatása

ETL performancia teszt

- Rendszer teljesítményének vizsgálata meghatározott terhelési szint mellett
- Biztosítja a rendszer elvárt sebességét
- Feltárja a performancia szempontjából szűk keretsztmetszeteket
- Típusai: Load, Stress, Volume stb.

13. Mikre kell ügyelni több adatforrás összekapcsolása esetén? Milyen esetben alkalmazna több adatbázis típust egy rendszerben? Milyen előnyei lehetnek memória, NoSQL és relációs megoldások együttes használatának?

Akkor van értelme együtt használni őket, ha mindkettő előnyeit szeretnéd élvezni. Pl. a komplex kapcsolatokkal rendelkező adatstruktúrát lehet relációs adatbázisban tárolni, az eseményeket (pl. logokat) pedig NoSQL adatbázisban. Olyanra is láttam példát, hogy a kettő között szinkronizációt alkalmaztak, hogy mindkét helyen ugyan azok az adatok legyenek. A reláció adatbázis felelt azért, hogy az üzleti követelmények teljesüljenek (pl. az objektumok kapcsolataira), a NoSQL pedig a gyors kereshetőséget biztosította pl.: több mezőben egyszerre kereséssel, vagy szinonima kereséssel.

Az egyik fő különbség, amit számíthatsz arra, hogy szinte bármi, ami támogatja az SQL-t, támogatja az olyan adatokat, mint a triggerek az adatbázisban, azaz szabályokat tervezhetsz az adatbázisba, amelyek célja, hogy biztosítsák az adatok belső következetességét . Például beállíthatja a dolgokat, így az adatbázis azt állítja, hogy egy személynek kell címmel rendelkeznie. Ha ezt teszi, bármikor hozzáad egy személyt, alapvetően arra kényszeríti Önt, hogy társítsa a személyt valamilyen címmel. Lehet, hogy új címet adhat meg, vagy esetleg meglévő címmel társíthatja őket, de így vagy úgy, a személynek rendelkeznie kell egy címmel. Hasonlóképpen, ha töröl egy címet, arra kényszeríti Önt, hogy távolítsa el az összes olyan személyt, aki jelenleg az adott címen van, vagy hozzárendel minden más címet. Ugyanezt tehetjük más kapcsolatok esetében is, például azt, hogy minden embernek anyja van, minden irodának telefonszámot kell tartalmaznia stb.

Ne feledje, hogy ez a fajta dolog garantáltan atomi módon történik, így ha valaki más az adatbázist nézi, amikor hozzáadja a személyt, akkor egyáltalán nem látják a személyt, vagy pedig látják a személy címmel (vagy az anyával, stb.)

A NoSQL adatbázisok többsége nem kísérletet tesz az ilyen típusú végrehajtás biztosítására a megfelelő adatbázisban. Az Ön adataihoz szükséges kapcsolatok érvényesítése az adatbázis használatát végző kódban van rajtad múlik. A legtöbb esetben csak részben helyes adatokat láthatunk, így még akkor is, ha van olyan családfa, ahol minden embernek szülőkkel kell társulnia, lehet, hogy az Ön által megadott korlátozások valójában nem lesznek érvényesíteni. Vannak, akik azt akarják, hogy akarják. Mások garantálják, hogy csak átmenetileg történik, bár pontosan mennyi ideig lehet / lehet tartani a kérdést.

Egyéb szempontok SQL ellen

- Framework-öt használnak (RoR, Django, stb.), amik általában eleve egy ORM rétegen keresztül mennek, eleve “mindegy”
- Always-on: Adatbázis feltétel miatt ne legyen hiba, majd utólag kijavítjuk!
- Drága? Know-how?
- Egyenlőre azért ne temessük az SQL-t, sok webes cégnél a fontos adatokat továbbra is SQL-ben tárolják, de aktívan próbálnak áttérni NoSQL megoldásokra

Mit veszítünk?

- Nincs SQL ☺
- Pl. ha észreveszed, hogy hibás vagy piszkos adat van, akkor ennek helyrehozása már nem olyan egyszerű, mint “UPDATE … WHERE …”
- Márpedig az eddigiek miatt sok ilyen lesz ☺
- Egyszerű analitikát sem lehet csinálni
- Tranzakciók ☹
  - A fejlesztési ciklus vége fele kezd el fájni, de akkor nagyon
- Nincs séma, “Mi van az adatbázisban?”
  - Pl. jön egy új alkalmazott
- Hozzáférési jogosultságok
- Stb.

Saas kihívásai: rapid development

- Agile, Scrum, Sprints, stb.
- Pl. sikeres magyar startupoknál 2 hetes **iterációkban** dolgoznak
- Minél hamarabb rakd ki produkcióba, aztán meglátjuk és iterálunk
- Gyorsan változó kód
- **Gyorsan változó adatséma**
- Nem akarnak SQL sémákat létrehozni és azoknak az életciklusát követni:
  - Fejlesztőgépeken
  - Tesztkörnyezetben
  - Produkciós környezetben
  - Continuous Integration (CI), pl. Jenkins
- **Legyen dinamikus JSON,** esetleg Protocol Buffers

- Bárhonnan elérhető
- Cross platform
- Rapid development: állandó bugfixek, featurek
- Havi bevételek vs. verziónkénti bevételek
- A/B tesztelés
- Rengeteg analitika
  - Funkcionális: mit nyomogat a user
  - Adat: mit hoz létre a user
  - Web: honnan jön a user, stb.

14. Mit nevezünk csalás detektálásnak? Mi a csalás fogalma? Milyen csalás felderítési eszközöket ismer? Adjon példát egy csalás detektálásra.

Mi a csalás detektálás?

- Csalás (szoftver): valamilyen szoftveres rendszer kiskapuinak kihasználása kártékony feladat céljából
- Milliárd dolláros ág, minden évben nő
- 2009-ben egy felmérés szerint a világ cégeinek 30%-a jelentette hogy valamilyen csalás áldozata
- Mi lehet csalás?
  - > Banki tranzakciók
  - > Szolgáltatások ingyenes használata
  - > Kiskapuk kihasználása
  - > Stb.

Csalás detektálás

- Adat elemzés – tradicionális
  - > Előny: csalásra utaló összefüggések feltárása
  - > Hátrány: komplex domain megismerést és összetett fejlesztést igényelhet
- Gyakran ismétlődő események így elemzéssel feltárhatók
- Felelhetők hasonló mintájú de más tartalmú előfordulásban is
- Elsőként telefon társaságok, bankok és biztosítók használtak adat elemzést csalás detektálásra (neurális hálózat)

Csalás fogalma

- Egy vagy több szereplő szándékosan titokban megkárosít valakit vagy felhasznál valamit saját célja érdekében
- A technika fejlődésével egyre több formája jelent meg a csalásnak
- Gyors szoftver fejlődés kedvez a csalásoknak
- Egyre nagyobb hangsúlyt kapnak a biztonsági kérdések
- Biztonság vs. gyors fejlődés

Csalás jellemzők

- Interneten keresztüli csalások száma 12-szer több a hagyományos csalásokhoz képest
- Tipikus csalási környezetek:
  - > Mobil hálózatok
  - > Biztosítási követelések (pl.: automata bejelentések feltűnő összeghatár alatt)
  - > Adó visszakövetelések
  - > Bankkártya műveletek
- Adaptív és gyorsan fejlődő csalási technikák, speciális eszközöket igényel a feltárásuk

Csalás felderítési eszközök

- Tudás feltárás adatbázisokban
- Adatbányászat
- Gépi tanulás
- Statisztikák
- Kiugró értékek keresése
- Automatikus algoritmus futtatás
- A csalás „kezelés” általában egy tudás intenzív tevékenységet igényel

Statisztikai módszerek

- Adat elő feldolgozás, validáció, hiba javítás, hiányzó/hibás adatok javítása
- Statisztikák számítása/karbantartása: átlagok, mennyiségek, teljesítmény metrikák, valószínűségi eloszlások
  - > Példa: átlag karbantartás a hívások hosszáról, számáról havonta; átlagos számla kifizetési idők
- Modellek és valószínűségi eloszlások különböző eseményekről
- Felhasználói profilok elemzése
- Idő függő adatok idősoros elemzése
- Klaszterezés és osztályozás minták felderítése és asszociálás céljából
- Összehasonlítás korábbi adatokkal
- Hibás riasztások felderítése, visszajelzések
- Jövő megjósolása (hipotézis)

Mesterséges intelligencia módszerek

- Adat bányászat osztályozásra, klaszterezésre, szegmentálásra; asszociációk, szabályok és gyanús minták automatikus felderítése
- Minta felismerés közelítő osztályok és klaszterek feltárására
- Gépi tanulási módszerek csalás típusok azonosítására
- Neurális hálózatok amik alkalmasak gyanús minták megismerésére minták alapján, majd később azokat képesek megtalálni valós adatokon
- Gépi tanulási módszerek kategóriái:
  - > Felügyelt tanulás
  - > Nem felügyelt tanulás

További csalás detektálási technikák

- Link analízis
- Bayes-i hálózatok
- Döntés elmélet
- Land Sequence Matching

Csalás detektálás példa

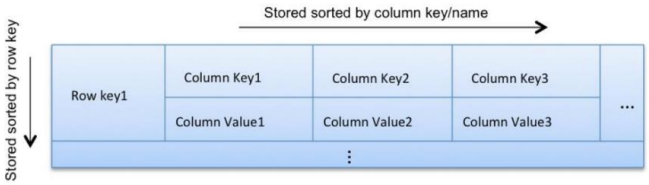
- SPAM detektálás közösségi portálon

Cassandra Fraud Detection

- Nagy adathalmazok hatékony kezelése
- Közel valós idejű adat elemzés
- Adat minták tárolása
- Gyakran jól jön az oszlop orientált szervezés
- További infok:
  - > http://www.planetcassandra.org/blog/functional_use_cases/fraud_detection/

Cassandra adat tárolás - emlékeztető

- NoSQL megoldás, de gyakran hasznos definiálni adatsémát
- Oszlop orientált tárolás
- Csalás detektálás: szeretnénk megtalálni azokat a spam üzeneteket, amiket többször „post”-oltak (ezek gyanúsan spam-ek)



Tárolás struktúra

- Row key: a Social Media tartalom hash-e
- Column Key: Kommentelő egyedi azonosítója
- Column Value: Post azonosítója és időbélyege
- Könnyű megállapítani mennyi alkalommal lett ugyanaz a tartalom „post”-olva, hogyan?
  - > **Meg kell számolni az oszlopokat!**
- **Az időbélyeg miatt idősoros elemzés is végrehajtható és feltárhatók „post”-burstök vagy intervallumok!**

15. Milyen adat vizualizációs lehetőségeket ismer? Vesse össze a vastag, vékony kliens technikákat a reporting eszközökkel? Milyen reporting eszközöket ismer, ezeknek mi az előnye, hátránya?

Három rétegű architektúra

Megjelenítési réteg

Üzleti logikai réteg

Adat réteg

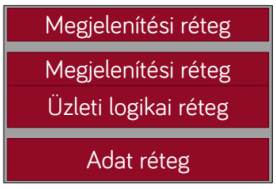
Három rétegű architektúra - Desktop

- Szoros kapcsolat a rétegek között
  - > Gyakran nem is egyértelműen elválasztható rétegek
  - > Közös kódbázis és technológiák, az egész alkalmazás egy egységet alkot

- Előnyei
  - > Teljesítmény
  - > „Capability” – szinte mindent meg lehet csinálni
- Hátrányai
  - > Rugalmatlan
  - > Vagy platform specifikus, vagy túl általános

3 r. arch. – Web (szerver oldali renderelés)

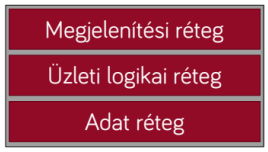
- Jobban elkülönülő rétegek
  - > Adat réteg szinte mindig leválasztva, külön adatbázisban
  - > A megjelenítés webes technológiákkal történik, a kliens böngészőjében



- Megjelenítési réteg ketté válik
  - > Szerver oldali renderelés: a HTML-t a szerver állítja elő, kitöltve a dinamikus részeket
  - > Kliens oldalon megjelenítés, illetve valamennyi logika kezelése, általában JavaScript-tel (pl. auto-complete)
- Sok kész keretrendszer és technológia
  - > Ezek általában „comple solution”-t adnak

3 r. arch. – Web (kliens oldali renderelés)

- Teljesen elkülönülő rétegek
  - > Kommunikáció megállapodásos interfészen (API-n)
  - > Akár technológiailag teljesen eltérő implementációk



- Megjelenítési rétegnek sokkal több feladat, logika kerül ide is
  - > A HTML a sablon alapján a kliens oldalon kerül kitöltésre
  - > Kliens oldali validáció
  - > Pl.: pagelés, térkép kezelés, grafikon kirajzolás

Adat réteg

- Adat tárolás és hozzáférés a rendszer felől
  - > A felhasználók általában nem írnak SQL-t
  - > A rendszer elvásásokat támaszt az adatbázis felé
    - – Konzisztencia garanciák, rendelkezésre állás stb.
- Sok fajta adatforrás
  - > Adatbázisok, fájlok, más rendszerek...
- Strukturált, vagy nem
  - > Relációs adatbázisok
  - > NoSQL adatbázisok
  - > Fájlok

Adat réteg – BI rendszerekben

- ”Vastag” réteg
  - > Sok adatforrás és transzformáció (emlékezzünk vissza az ETL órára)
  - > Nagy mennyiségű adat, de nem minden fontos belőle
- A megjelenítéshez használt részeken fontos a performancia
  - > Ütemezett jobok futhatnak sokáig (például az éjfélkor készülő report ráér reggelre elkészülni)
  - > Online felületek kiszolgálásához viszont sebesség kell

Üzleti logikai réteg

- Az üzleti logika itt van implementálva
  - > A legtöbb keretrendszer arra fókuszál hogy nekünk csak ezt a kódot kelljen megírni
- Lehet egy komponensben vagy elosztottan is
  - > Általában a konkrét felhasználás szabja meg
  - > Tipikus kliens-szerver minta
- Technológia és nyelv függő

Üzleti logikai réteg – BI rendszerekben

- Általában van egy a klienstől elkülönülő *backend*-ünk, amely
  - > Biztosítja az *API*-t (interfészt a kliens felé)
  - > Az adathozzáférést szabályozza
  - > Az adatokat strukturálja
    - – *Data Transfer Object* (ez már lehet a tisztán megjelenítendő adat)
  - > A felhasználó ehhez már hozzáférhet
    - – Pl. *REST API*-t megfelelő paraméterekkel meg tudna már hívni, de kényelmetlen használni, ráadásul nem fejlesztő
- Vastagkliens esetén közvetlen DB csatlakozással is rendelkezhet a kliens
  - > Azonban ez biztonsági rés (a DB-t hozzáférhetővé kell tenni „kifelé”)
  - > Frissítések kiadása fájdalmas
    - – Backend frissítésnél csak kompatibilitást kell figyelni

Megjelenítési réteg

- A felhasználó számára érthető formában jeleníti meg az adatokat
  - > Adat megjelenítés
  - > Adat manipuláció
- Kommunikáció az üzleti logikai réteggel
  - > Vagy „programon belül”
    - – Függvény hívások, callbackek
  - > Vagy API-n keresztül (főleg weben)
    - – HTTP/REST, WebServices, WCF

Megjelenítési réteg – BI rendszerekben

- Elfedi a kommunikációt *backend API*-val
- Grafikus megjelenítést biztosít
  - > Reszponzivitás és akadásmentes megjelenítés alapvető elvárás
- Két nagyon fontos komponens
  - > Kommunikáció
  - > Megjelenítés
  - > A kettő között adatkötés
    - – Hatékonyan
      - Közvetlenül az adatot lehetőleg
      - Ha a *backend* nem így biztosítja, kérjük az *API* változtatást
    - – Talán a legfontosabb UX kérdés

Megjelenítés webes felületen (Frontend)

- Üzleti logika a backend-en található
  - > De a frontend is kell, hogy tartalmazzon logikát
    - Pl. *HTTP 403* (*Forbidden*) a *backend response status code*-ja
      - A frontend tudja, mit jelenít meg ekkor a felhasználó felé
    - Pl. Sokat várunk egy aszinkron kérésre
      - A frontend tudja, mit jelenít meg a válasz megérkezéséig, illetve *timeout*-ra is fel van készítve
    - Pl. 10000 objektum egyoldali listás megjelenítése egyértelműen rossz megoldás, mégis szükséges ennyi adat megjelenítése

Adatvizualizációs elvek

- Sok szempontot kell szem előtt tartani
  - > Hatékonyság
    - Megjelenített adat felhasználhatósága
      - Egy jó forma (grafikon) választása
    - (Adatkötés módja arányos a várakozási idővel)
  - > Átláthatóság
    - Egyszerűség, letisztultság
      - Azt mutatja/emeli ki, ami fontos
  - > A vizuális kommunikáció modern formája
    - A JSON, XML, CSV nem vizuális ☺
- Művészet és tudomány ötvözete
  - > Már nem elég, hogy szép legyen
  - > Megjelent a Big Data It is not owned by any one field

Big Data következményei

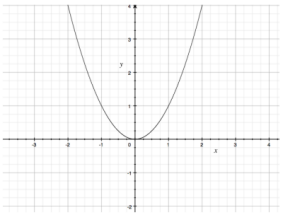
- Az iteráció költséges
  - > Az adattranszformáció minimalizálandó
    - Pl. 10millió objektumban a név elé szeretnénk fűzni egy *prefix*-et (helyette biztosítsa eleve így a *backend*)
- Az adatszinkronizációt össze kell hangolni a kliensoldali eseményekkel, életciklusokkal
  - > Cél a lehető legkevesebb várakozási idő
  - > Az adatok nem egyszerre történő megjelenítésénél használjunk *pagination*-t
    - Egy adott részhalmazt szükséges csak a hálózaton át letölteni és a memóriában tárolni
    - Érdemes cache-lni a szomszéd lapokat a jobb UX céljából

Diagramok szerepe

- Nyers számok és származtatott értékek megjelenítésére
  - > Pl. grafikon (lásd következő dia)
- Folyamatok, kapcsolatok leírására
  - > Ennek ellenére léteznek formális megoldások, szabványok (pl. UML), melyeknek köszönhetően az információ ábrázolása egyértelmű lehet
- Grafikus modellek is lehetnek diagramok

Grafikonok szerepe

- A diagram speciális formája
- A különböző értékeket, arányokat kifejező vizuális elemek
- Matematikai értelemben egy 2 dimenziós függvényábrázolás
  - > Egyik tengely
    - Bemeneti érték
  - > Másik tengely
    - Kimeneti érték



Táblázatok szerepe

- Jellemzően kevés objektum megjelenítése egyszerre
  - > *Pagingation*
- Az objektumok legtöbb tulajdonságát leíró elem
  - > Pl. admin felület mindent megjelenít

Rank in the Fifty States, 2012	Rank in all states & territories, 2010	*State or territory*	Population estimate for July 1, 2012	Census population, April 1, 2010	Census population, April 1, 2000	Seats in U.S. House, 2013–2023	Presidential Electors 2012–2020	2010 Census Pop. per House seat[4]	2000 Census Pop. per House seat
1000001	1000001	California	38041430	37253956	33871648	1000053	1000055	702905	639088
1000002	1000002	Texas	26059203	25145561	20851820	1000036	1000038	785799	651619
1000003	1000003	New York	19570261	19378102	18976457	1000027	1000029	668210	654361
1000004	1000004	Florida	19317568	18801310	15982378	1000027	1000029	752052	639295
1000005	1000005	Illinois	12875255	12830632	12419293	1000018	1000020	675296	653647
1000006	1000006	Pennsylvania	12763536	12702379	12281054	1000018	1000020	668646	646371
1000007	1000007	Ohio	11544225	11536504	11353140	1000016	1000018	640917	630730
1000008	1000008	Georgia	9919945	9687653	8186453	1000014	1000016	745204	629727
1000009	1000009	Michigan	9883360	9883640	9938444	1000014	1000016	658909	662563
1000010	1000010	North Carolina	9752073	9535483	8049313	1000013	1000015	733499	619178
1000011	1000011	New Jersey	8864590	8791894	8414350	1000012	1000014	676300	647258
1000012	1000012	Virginia	8185867	8001024	7078515	1000011	1000013	727366	643501
1000013	1000013	Washington	6897012	6724540	5894121	1000010	1000012	747171	654902
1000014	1000014	Massachusetts	6646144	6547629	6349097	1000009	1000011	654763	634910
1000015	1000015	Arizona	6553255	6392017	5130632	1000009	1000011	799002	641329
1000016	1000016	Indiana	6537334	6483802	6080485	1000009	1000011	720422	675609

A hatékony vizuális elemek jellemzői

- Az adatot mutatja
- Rábírja a megtekintőt, hogy a tartalomra fókuszáljon
- Elkerüli az adatok torzítását
- Sok szám kis helyen
- Nagy adathalmazok láthatóak rajta
- Különböző adatok összehasonlítására ösztönzi a megtekintőt

Szaknyelv ismerete

- Költsége
  - > A felhasználók egy speciális csoportja tudja csak értelmezni
- Értéke
  - > Megnöveli az eszköztárat, kifejezési erőt az egyes vizuális elemeknél
- Legjobb, ha a szaknyelvi kiegészítés többlet jelentést hordoz, azonban anélkül is információt hordoz a vizuális elem

Üzleti intelligencia

- Megoldások, eszközök összessége
  - > Melyek a nyers adatot értelmes és hasznos információkká alakítják
- Képes kezelni nagy mennyiségű strukturálatlan adatot
  - > Segít azonosítani, fejleszteni, egyáltalán létrehozni üzleti lehetőségeket
- Célja a nagy adatkötetek könnyű értelmezésének lehetővé tétele
  - > Tények segítségével üzleti döntéshozatal segítése

Célja

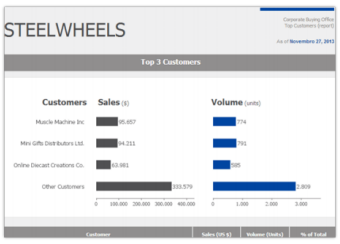
- Meglévő adatok elérhetőségének javítása
  - > Hozzáférhetőség szempontból
    - Könnyebben
    - Gyorsabban
    - Szélesebb körben
  - > Formai szempontból
    - Olyan alakra hozni, ahogy szükség van rá
- A 3 fő érv a BI bevezetésre
  - > Üzleti adatokhoz való hozzáférés sebességének növelése
  - > Különböző rendszerek adatainak integrálása
  - > Az elemzési képességek eljuttatása több felhasználóhoz

BI felhasználhatósága

- Riport-, jelentés- és beszámoló készítése
- Kimutatások, kiegyensúlyozott mutatórendszerek készítése
- Üzleti, statisztikai elemzés
- Tervezés, előrejelzés, üzleti modellezés
  - > Pl. „Mi lenne ha…” futtatása
- Konszolidáció, aggregáció
- Idősoros elemzés
- Adatok földrajzi elemzése (hol mi népszerű)
- Adatvizualizáció, grafikonok, kijelzők
- Adat-, szöveg- és hangbányászat

Pentaho Reporting

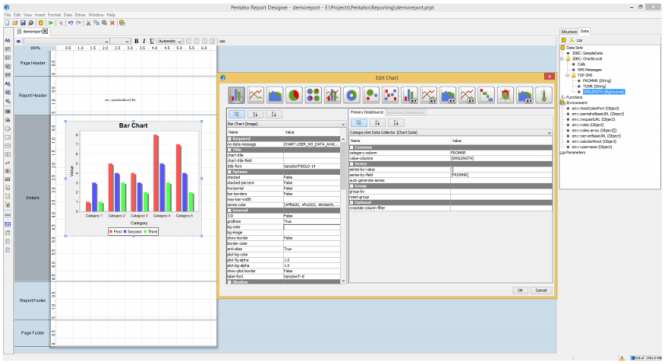
- Reporting: adat transzformálása információvá
- Támogatott kimeneti formátumok.
  - > PDF
  - > Excel
  - > HTML
  - > Text
  - > Rich-Text-File
  - > XML
  - > CSV



Pentaho Reporting

- JFreeReport engine
- Felügyeleti és végfelhasználói felhasználás
- Böngésző alapú megjelenítés
- Testre szabható felület
- Beágyazhatóság más alkalmazásokba (SDK)
- Önállóan vagy BI Suite-ba ágyazva is használható
- Rengeteg adatforrás támogatása

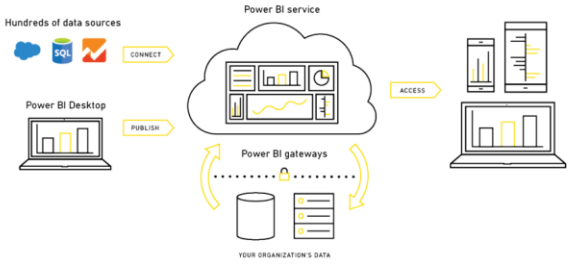
Pentaho Reporting példa



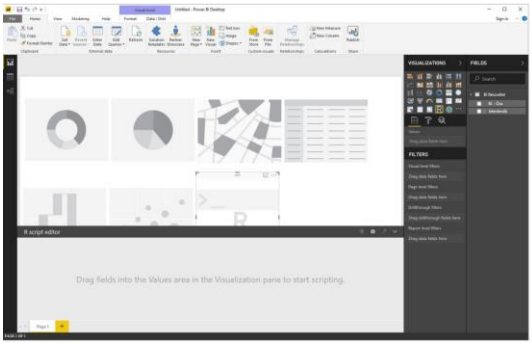
Power BI

- A Business Intelligence után a „másik” Microsoft BI eszköz
- Felhő alapú kiszolgálás (SaaS)
  - > Microsoft Azure
  - > Rest API saját rendszer illesztéséhez
- Desktop, mobil és webes megjelenítés
  - > Hagyományos reportok
  - > Live dashboard
  - > Interaktív reportok
- Machine learning eszközök

Power BI Architektúra



Power BI



16. Mik a JavaFX technológia előnyei a korábbi Swing/AWT-hez képest? Mire szolgál az FXML? Vesse össze a JavaFX és a web alapú alkalmazásokat előnyök és hátrányok szempontjából.

JavaFX - Bevezetés

- A Java technológia sok, különböző környezetben elérhető (desktop, web, mobil, blu-ray, stb.), ez a technológia egyik legfőbb jellemzője
- Oracle felvásárolta a Sun Microsystems-t
- Java elavult felhasználói felület szempontjából (Swing, Awt)
- Megváltozott a felhasználói igény, szebb, látványosabb elemekre van szükség
- JavaFX: a Java válasza a megváltozott igényekre

A JavaFX főbb jellemzői

- Java kliens platform új generációja
- Gazdag felhasználói felület és RIA (Rich Internet Application) támogatás
- Konzisztens működés több platformon
- Gazdag Java alapú API grafikai és média támogatásra
- Hardware támogatás
- Széleskörű média támogatás

JavaFX 2.0 újdonságai

FXML:

- XML alapú deklartív leírónyelv felhasználói felületek tervezésére
- Ideális statikus felületek, mint például űrlapok, táblázatok megjelenítésére
- Támogatja dinamikus felületek létrehozását, script-ek felhasználásával
- Java kóddal együtt tud működni

Előnyei:

- XML alapú, ezért minden fejlesztő számára ismert nyelv
- Főként a web fejlesztők kedvelik, de sok helyen előfordul (pl. Android)
- Nem kell lefordítani a kódot, ha változik az FXML
- Átláthatóbbá teszi a Scene Graph struktúráját
- Könnyebben tud több fejlesztő együtt dolgozni a UI-on

Reporting:

- Adat transzformálása információvá
- Pentaho reporting
- JavaFX

JavaFX grafikonok

- Üzleti alkalmazásoknál tipikusan fontos
- javafx.scene.chart csomag
  - > Pie Chart
  - > Line Chart
  - > Area Chart
  - > Bubble Chart
  - > Scatter Chart
  - > Bar Chart

Electron - Bevezetés

- Cél: web és desktop előnyeinek ötvözése
  - > Web – több platformon egységes megjelenítése, közös nyelven
  - > Desktop – natív élmény, teljesítmény
- Alap ötlet: webes technológiákkal desktop app
  - > HTML + CSS + JavaScript + becsomagolt böngésző
  - > Node.js runtime, Chromium rendering
  - > Rendszer funckiók API-n keresztül

Electron

- Előnyei
  - > Egyszerű cross platform alkalmazást csinálni vele
  - > Minden platformon hasonló felhasználói élmény
    - – Nincs például Java-s rendszer specifikus look and feel
  - > Webes technológiákkal fejleszthető
- Hátrányai
  - > Lassú, egy egész böngészőnek el kell indulnia
  - > Nagy, egy egész böngészőnek bele kell férnie
  - > Instabil, nagyon gyorsan változó környezet (web), desktopon általában hosszabb távra terveznek.

Miért fontos a reszponzivitás?

- Képernyőfüggetlen megjelenítés
  - > Képátló méretbeli eltérések
  - > Képarány eltérések
  - > Felbontásbeli eltérések
- Webes környezetben különösen fontos
  - > Hiszen minden készüléken létezik böngésző okostelefontól okos tv-ig
    - – Minden készülék típusra a lehető legjobb megoldást szeretnénk használni

Reszponzív vs adaptív

- Reszponzív
  - > Egy felület, ami a kliens képernyőméretéhez igazodik
    - – Nincs külön desktop és mobil site
  - > Képernyő dimenziók szerinti layout
  - > Sokféle klienst támogat általánosan
- Adaptív
  - > A kliens típusától függő felület implementációk
    - – Desktop, tablet, mobil
  - > Az adott kliensre optimalizált, esetleg kinézetben is
    - – Pl. iOS natív alkalamzást imitáló téma

AngularJS (Angular 1)

- Egyik legelterjedtebb JavaScript alapú webes keretrendszer
- *Single Page Application*-ök készítésére
- Kétirányú data binding a HTML elemekre rakott attribútumok segítségével
  - > Scope: HTML-ből és JS-ből is elérhető
- Reszponzív támogatás
- Open source és ingyenes

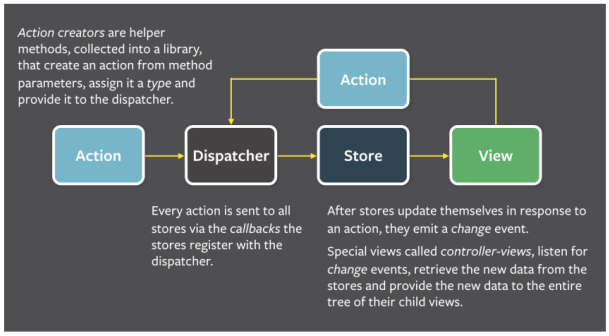
Angular (Angular 2, Angular 4)

- TypeScript és (elméletben) JavaScript
  - > A TypeScript preferált, toolok, doksik ehhez
  - > Típusos, JS-re forduló nyelv
- Újragondolt architektúra
  - > Új DependencyInjection -> Service
  - > Module kezelés változott
- Új eszközök
  - > Angualr-CLI, kiinduló projekt generálás, fejlesztés támogatás
  - > TypeScript tooling (Transpiler, Debugger, TSLint)

React

- JS Library felhasználói felületek készítéséhez
  - > Elsősorban dinamikusan változó felületeknél érdemes használni
  - > Nem keretrendszer, használható más megoldásokkal, pl. Angular
- VDOM – Virtuális DOM, lehet módosítgatni, csak a végeredmény fog valóban renderelődni
- JSX – JS szintaxis kiterjesztése, pl. írhatunk HTML-t egyből a kódba (nem stringként)
- Flux – Architektúrális minta
  - > Egyirányú adatáram, nézetek nem tárolnak adatot csak megjelenítenek
  - > Ha valami interakció „action" történik az végigfut az „elejétől"

React - Flux



Vue.js

- Webes JavaScript keretrendszer
- Fő jellemzői
  - > Moduláris, nem muszáj az egészet használni
  - > Egyszerű, hamar el lehet kezdeni, nincs sok boilerpalte kód
  - > Performancia kiemelt fókuszban
- Fejlett feature set
  - > VDOM
  - > Reaktív komponensek, one-way dataflow

Polymer

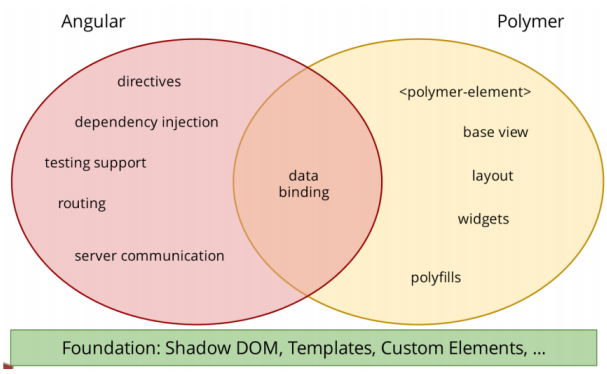
- Egyedi webes komponens készítő könyvtár
  - > Kompatibilitás szem előtt
    - – Régebbi, egyedi böngészőknél
  - > W3C standard-ek és új böngésző API-k
- Nem egy teljes keretrendszer
  - > Habár alkalmas webalkalmazások készítésére
- Hogyan néznek ki ezek az egyedi komponensek?

```
<google-map lat="37.790" long="-122.390"></google-map>
```

Miért érdemes használni?

- Saját komponensek készítése
  - > Egyszerű
  - > Gyors
  - > Reszponzív
  - > Látványos: Material design
- Számos elem előre elkészítve része
  - – Floating Action Button
  - – Ikonok
  - – Menü
  - – Toolbar
  - – Stb.
- Dinamikusan fejlődő könyvtár
  - > Ma még számos hiányossága van sajnos ☹

Angular vs Polymer



Angular és Polymer

- A két rendszer használható közösen is
- Adatkötések használata
  - > Az *Angular* képes
    - – Figyelni a *Polymer*-es elemek DOM eseményeit
    - – Állítani az attribútumait
    - – Gyerek elemeket beszúrni
    - – Úgy kezelni a *Polymer*-es elemeket, mint bármely más *DOM* elemet
  - > A *Polymer* a ***Node.bind()*** új metódust használja fel adatkötésre
    - – Az *AngularJS* nem, így a *Polymer* –ből kezdeményezett változásokra nem tud figyelni

Diagrammkönyvtárak

- Miért kellenek?
  - > Látványosak
    - Emelik a felhasználói élményt
  - > Tömörek
    - Egy speciális nézetet biztosítanak, melyről minden leolvasható, letisztult, kifejező
  - > Könnyű felhasználni
    - Szinte minden projekthez szükségesek
      - Nem szeretnénk mindig új komponenseket készíteni
    - Bő funkcionalitású
      - Egy szakértő csapat készíti, sok böngésző támogatásával, gazdag animációs eszköztárral

Hogyan használjuk?

- 1. lépés: függőség behúzása (JS, ritkábban CSS is)
- 2. lépés: A könyvtárat felhasználva példányosítunk komponenst
  - > Fontos, hogy a kód a függőség behúzása után legyen, hogy tudja értelmezni -> Interpretált nyelv hátránya ☹
- 3. lépés: Adat transzformáció a könyvtár által elvárt formára
  - > Ha nagyon sok adat van /erőforrásigényes a transzformáció, érdemes lehet az API-t úgy alakítani, hogy a megfelelő formában biztosítsa a választ a kliensnek
- (4. lépés): Ellenőrizzük le az életciklusokat, hangoljuk össze az oldal betöltésével, az üres adathalmazra is teszteljünk
  - > Felhasználói élmény megtartása

D3JS könyvtár

- Data-Driven Documents JavaScript
- „*bring data to life*”
- Lépések
  - > Adatkötés DOM-hoz
  - > Ábrázoláshoz szükséges transzformációk (adat vizuális elemmé)
    - Pl. egy lista táblázatba helyezése
- Nem egy mindent megoldó keretrendszer, ami az összes funkciót szolgáltatja
  - > Hatékony adatkötés és megjelenítés sok adathoz
  - > Dinamikus adatváltozás és animációk támogatása