Adatbáziskezelés, bevezető

Csima Judit

BME, VIK,
Számítástudományi és Információelméleti Tanszék

2018. szeptember 5.

Órák, emberek

- heti két óra: szerda 14.15-16.00 (H601) és péntek 8.15- 10.00 (H406)
- előadás és gyakorlat (akár egy órán belül is) vegyesen, laborok külön alkalmak
- előadás: slide-ok és tábla
- gyakorlat: feladatsorok, órai munka, házi feladat gyakorlásnak
- labor: 6 labor, előre bejelentett időpontban, maximum egyet lehet hiányozni, első labor szeptember 19., szerda
- Csima Judit (előadás és gyakorlat): csima@cs.bme.hu
- Katona Gyula (előadás, gyakorlat): kiskat@cs.bme.hu
- Hadházy László (labor): laszlo.hadhazy@gmail.com
- weboldal: www.cs.bme.hu/adatb

Követelmények

- órákra járni, a laborokból legfeljebb egyszer lehet hiányozni
- félévközi zh az aláírásért, pontszáma beszámítható a vizsga pontszámába
- a zh időpontja: november 21., a szerdai órán, pótzh a 14. héten külön megbeszélt időpontban
- év végén írásbeli és szóbeli részből álló vizsga

A tárgy célja

- ha adatokat akarunk tárolni, módosítani, lekérdezni: hogyan célszerű?
- megoldás: adatbáziskezelő rendszerek (DBMS)
- kérdések:
  - milyen részei vannak egy adatbáziskezelő rendszernek?
  - melyik rész mire való?
  - az egyes részek működésének alapjai (elmélet)
  - mit látunk ebből használóként, mit kell tudni a használathoz? (gyakorlat)

Miért kellenek az adatbáziskezelő rendszerek?

- kezdetben adatok tárolására: file-ok, minden file egy táblázat (pl. diákok jegyei egyes tárgyakból, minden tárgyhoz egy újabb táblázat)
- problémák:
  - ha a diákokról alapadatokat is tárolni akarunk, akkor redundancia vagy még egy táblázat kell, de akkor ezt az újat mi kapcsolja a többihez?
  - adott diák összes eredményét hogyan szedjük össze?
  - bonyolultabb lekérdezéseket hogyan? (pl. minden diák, aki már felvette az egyik, de nem vette fel a másik tárgyat)
  - több felhasználó egyszerre, jogosultságok kezelése

Mire jók az adatbáziskezelő rendszerek?

- formális keret adatbázis séma tervezésére:
  - mik az adattáblák (milyen adatokat tárolunk egyben): pl. diákok alapadatai, illetve, hogy melyik diák milyen tárgyakat vett fel és tárgyanként egy adattábla az eredmények nyilvántartására
  - az egyes adattáblákon belüli megkötések (pl. neptun kód egyedi az alapadatoknál vagy (neptun-kód, félév) pár meghatározza a jegyet a tárgyaknál)
  - adattáblák közti kapcsolat (pl. a tárgyhoz tartozó adattáblában szereplő neptun-kódnak szerepelnie kell az alapadatos táblában)
- tervezés után adatbázis séma létrehozása magas szinten, ehhez adott egy rendszertõl függõ DDL (Data Definition Language)

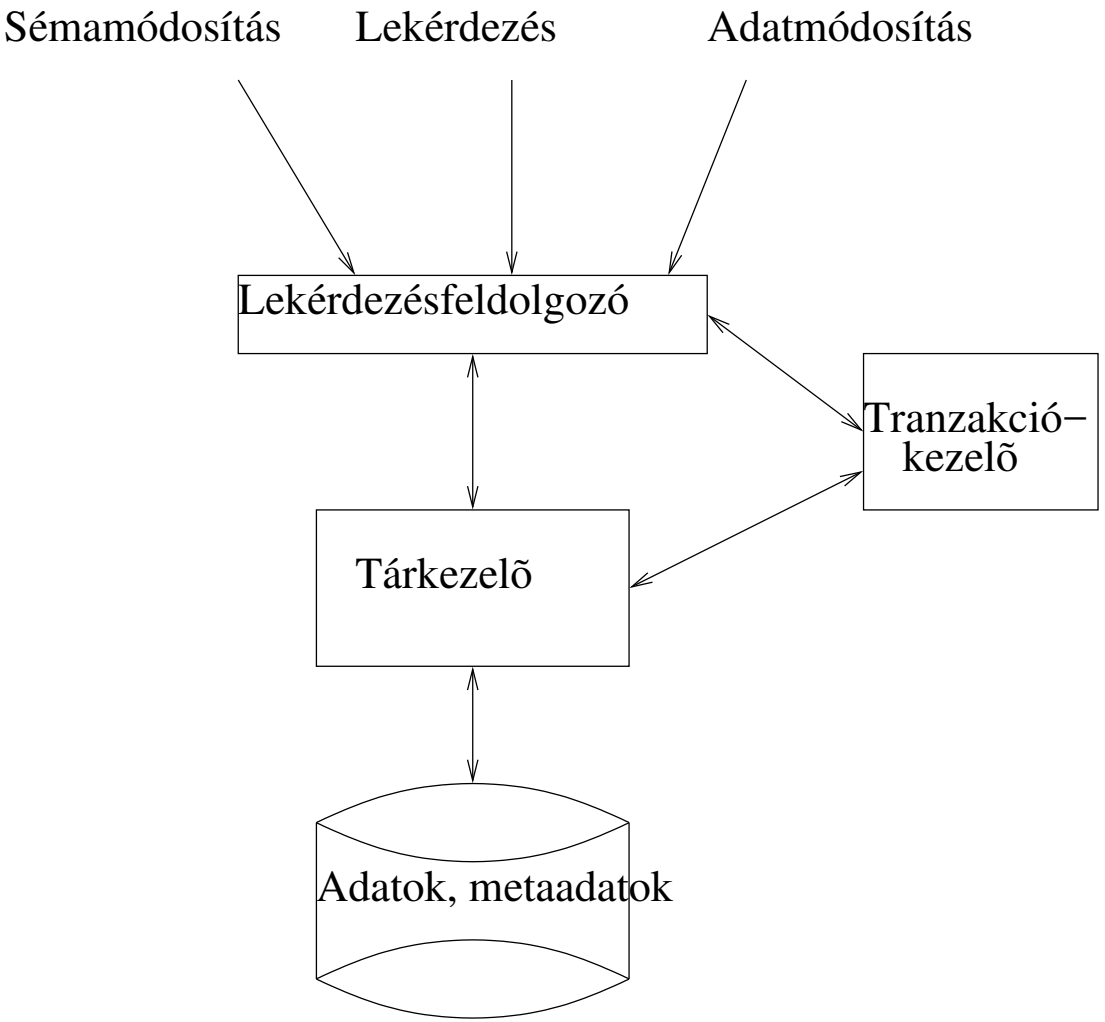
Mire jók az adatbáziskezelő rendszerek?

- kész adatbázisba adatok beillesztése, módosítása magas szinten (az adatbázis fogalmi keretének feltöltése) pl.: új diák felvétele, tárgyfelvétel, jegybeírás, ehhez adott egy rendszertõl függõ DML (Data Manipulation Language)
- magas szintű lekérdezést tesz lehetővé (a tárolás konkrét ismerete nélkül): elég leírni, hogy mit akarunk, a DBMS megoldja a többit, ehhez Query Language (pl.: SQL)
  cél: gyakori kérdéseket könnyû legyen kérdezni (ehhez jól végiggondolt fogalmi keret kell, hatékony tárolás, ehhez fontos a jó tervezés)

Mire jók az adatbáziskezelő rendszerek?

- elválik logikai séma és a fizikai megvalósítás (nem kell az utóbbival bajlódnunk)
- nagy mennyiségû adat tárolása biztonságosan (jogosultságok, illetéktelen hozzáférés megakadályozása illetve rendszerhibák elleni védelem)
- többfelhasználós mûködés támogatása (egyidejû hozzáférés, de ne legyen hibás, inkonzisztens állapot emiatt)

A DBMS felépítése, részei



Adatok, metaadatok

Fizikailag valahol tárolódnak az

- adatok: diákok adatai, ki milyen tárgyat vett fel, ...
- metaadatok: mik a relációk (adattáblák) nevei, attribútumai (miket tárolunk bennük), attribútumok típusai (string, szám), illetve pl. milyen indexek vannak a kereséshez

Tárkezelõ

A kért információ beolvasása, módosítások végrehajtása. Kb. mint az operációs rendszerek file-kezelõje, de itt néha (többfelhasználós mûködésnél pl.) mi mondjuk meg, hova történjen az írás (háttártárra, pufferbe), nem kezelheti teljesen szabadon a puffert.
Részei:

- File kezelõ: nyilvántartja az adatbázis állományát; fizikai I/O-t végez, ha a pufferkezelõ kéri; indexstruktúrába rendezi az adatokat (pl. B-fa)
- Pufferkezelõ: kezeli a memóriát, tárolja a filekezelõ által beolvasott blokkokat

Lekérdezésfeldolgozó

Alapfeladata: sémadefiníciós, adatmódosítós és lekérdezõs kérések fogadása, kezelése, ezt a felületet látjuk mi igazából

- sémamûveletek: az adatbázis logikai struktúrájának kialakítása, módosítása
  eredménye: maga az adatbázisséma, plusz kiegészítõ metaadatok (pl. mit hogyan tároljunk, indexek)
  nagyon fontos, meghatározza a további mûködést
- adatmódosítás: az adatbázis tartalmának módosítása, pl. új diák beillesztése, tárgyfelvétel törlése
  két lehetõség erre: vagy egy külön felületen át vagy alkalmazói programból (a második esetben a programozási nyelvnek van utasításkészlete az adatbázishoz fordulásra)
- lekérdezések: keresõkérdések az adatbázisra vonatkozóan
  itt is lehet külön felület vagy program

Lekérdezésfeldolgozó tennivalói: végrehajtási terv készítése

A magas szintû kérdéseket átalakítjuk elemi utasítások sorozatává.
A (legtöbbször) deklaratív kérdésbõl (ahol nem mondjuk meg, hogy milyen úton akarom megkapni az eredményt, csak azt, hogy mit akarok, ilyen pl. SQL) procedurális kérdést csinálunk (ahol már egy konkrét végrehajtási terv látszik).

PI: Ha a séma két relációja
diák(név, neptun) és *adatbáziskezelés*(neptun, félév, jegy)

SELECT jegy FROM diák, adatbáziskezelés
WHERE diák.neptun = adatbáziskezelés.neptun AND diák.név="Vombat Sándor"

Vombat Sándor összes eddigi jegye Adatbáziskezelés tárgyból

Lekérdezésfeldolgozó tennivalói: végrehajtási terv készítése

SELECT jegy FROM diák, adatbáziskezelés
WHERE diák.neptun = adatbáziskezelés.neptun AND diák.név=”Vombat Sándor”

Ez csak azt mondja meg, hogy mely relációkból, mit akarok megkapni, de azt nem, hogy hogyan kell ezt megszerezni.

Erre egy (elnagyolt) végrehajtási terv lehet pl. az, hogy ha van index a névre a *diák*-ban, akkor az alapján keressük meg V.S. neptun kódját, aztán ez alapján a másik relációban keressük meg a megfelelõ sorokat és olvassuk ki a jegyeket.

Lekérdezésfeldolgozó tennivalói: optimalizálás

Több lehetséges végrehajtási terv közül kiválasztani egy “legjobbat”. Nem valódi optimalizálás, nem a legjobbat keressük, csak egy elég jót.

Ezt az “elég jó” végrehajtási tervet a renszer maga keresi meg, de (rendszertől függően) bele tudunk nyúlni ebbe.

Fontos:

- a kapott (SQL) kérdés nem ad támpontot, hogy hogyan kell megkapni az eredményt, de nem is kötelez semmire
- érdemes szöszölni egy jobb végrehajtási terv keresésével, mert nagyok lehetnek a különbségek nagy adathalmaz esetén

Tranzakciókezelõ

Két nagyobb problémakör megoldására kell:

- több felhasználó egyszerre használja az adatbázist, egyidejû hozzáférések kezelése
- rendszerhibák, ABORT-ok hatásainak kivédése: ezek bekövetkeztekor sem veszhetnek el adatok, nem maradhat az adatbázis inkonzisztens állapotban

Alapfogalom a
tranzakció: egy felhasználóhoz tartozó, összetartozó utasítások olyan sorozata, melyek vagy mind végrehajtódnak vagy semelyik sem (ez az atomiság)
Pl. banki átutalásnál nem lehet, hogy csak a pénz levonása történik meg az egyik számlán, de nem íródik jóvá a másikon

Kliens-szerver rendszer

- Ez az eddig felsorolt sok alkotórész eloszlik a kliens és a szerver között. A szerver tartja a kapcsolatot az fizikai adatbázissal, a kliens pedig a felhasználóval, a többi funkció eloszlása nagyon változhat rendszertõl függõen.
- Erről majd laboron lesz még szó

Az adatbáziskezelõ használati szintjei

- felhasználó: adatmódosítást csinál, alkalmazói programot futtat, (esetleg egyszerûbb SQL kérdést fogalmaz meg)
- adatbázis programozó: összetett lekérdezések, alkalmazói program írása; jól ismeri az adatbázis felépítését, az adatbáziskezelőkben használt technikákat
- adatbázis tervezõ: sémát hoz létre, fizikai szervezésbe beleszólhat
- adatbázisrendszer megvalósító: az adatbáziskezelõt magát tervezi és írja

A félév anyaga

Amit felhasználóként, programozóként, esetleg tervezőként használunk:

- adatmodellezés: E/K (egyed-kapcsolat) diagram (grafikus jelölésrendszer)
  Az adatbázis fogalmi keretének megadására, tervet lehet vele készíteni, amit aztán majd át kell alakítani az adatbáziskezelõ által használt formális megadási módra.
- relációs adatmodell: nagyon fontos elméleti modell, tipikusan ilyenek az adatbáziskezelők mostanában
  az E/K modellt ilyenre írjuk át
- SQL: séma létrehozása, adatmódosítás és főleg lekérdezés

A félév anyaga

Az adatbáziskezelő rendszer mélyén, a háttárben zajló folyamatok megértése:

- fizikai szervezés: adatszerkezetek, pl. indexek, keresőfák (csak említve)
- tranzakciókezelés
- lekérdezésfeldolgozás: lekérdezési tervek készítése, értékelése, összehasonlítása

Programozás, labor: DBMS telepítése, SQL, SQL elérése Pythonból, stb.