Adatbázisok elmélete 1. előadás

Katona Gyula Y.
Budapesti Műszaki és Gazdaságtudományi Egyetem
Számítástudományi Tsz.
I. B. 137/b
kiskat@cs.bme.hu
http://www.cs.bme.hu/˜kiskat

2004

Tudnivalók

Katona Gyula kiskat@cs.bme.hu http://www.cs.bme.hu/˜kiskat/adatbazis/

Előadás:

- Szerda 8:15-10:00, I. B. 28.
- Csütörtök 14:15-16:00, I. B. 28.

Tankönyv

- Gajdos Sándor: Adatbázisok, Műegyetemi kiadó, 2001.
- Ullman – Widom: Adatbázisrendszerek, alapvetés, PANEM – Prentice-Hall, 1998
- Garcia-Molina – Ullman – Widom: Adatbázisrendszerek magvalósítása, PANEM – John Wiley & Sons, 2001
- Letölthető fóliák és egyéb anyagok a honlapon.

Aláírás és vizsga

Az aláírásért (10. hét, november 19.) az évközi zárhelyit legalább elégségesre meg kell írni. Ez egy 60 pontos zárthelyi lesz, stílusában hasonló, mint a tavalyiak, de nem feltétlenül 6 darab 10 pontos feladat lesz benne (természetesen az egyes feladatok pontszámai rajta lesznek a feladatsoron). A 60 pontból 20 pont kell a ketteshez, 30 pont a hármas szinthez, 40 pont a négyes szinthez és 50 pont az ötös szinthez. A zh pontszáma beleszámít a vizsgajegybe, amennyiben jobb, mint a vizsga pontszáma (erről lásd a vizsgánál írottakat.)

Akinek van régebbről aláírása, annak nem kötelező ZH-t írni, de írhat, ha akar. A régi aláírás egy esetleges rossz zhval sem vész el.

Pótzárthelyi 1 héttel a ZH után lesz (11. hét, november 26.), ezen már csak az aláírást lehet megszerezni, az eredmény semmiképpen nem számít a vizsgajegybe.

Sem a ZH-n, sem a pótzh-n nem lehet használni semmilyen segédeszközt (a zh és a pótzh closed book).

Aláírás és vizsga

Akinek nem sikerül a pótzh sem, az első három vizsgahét valamelyik vizsgáján próbálkozhat, feltéve, hogy az egyetemi adminisztráció által előírt formai követelményeket (iv vagy különeljárási díj, engedély, stb.) teljesíti. Csak egyetlen alkalommal lehet megpróbálni ilyen módon az aláírás megszerzését, az aláíráshoz a vizsgán legalább kettest kell elérni. Az itt elért pontszám nem számít bele a későbbi vizsgajegybe.

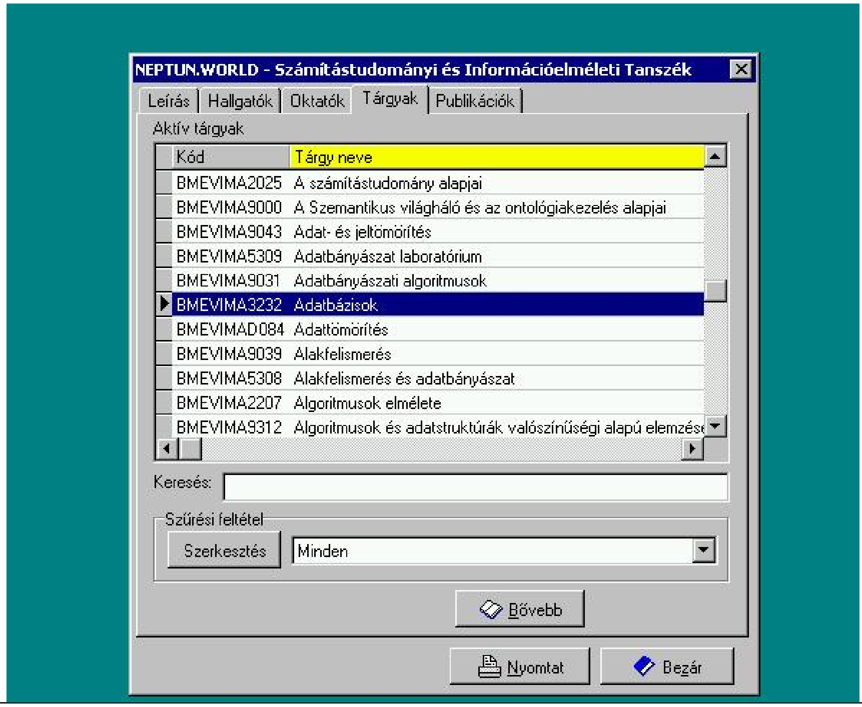
A vizsga: A vizsgaidőszakban három vagy négy vizsgaalkalom lesz. Mindegyik írásbeli, itt sem lehet használni semmilyen segédeszközt. A vizsgán 60 pontot lehet szerezni, ennek egy részét elméleti kérdésekkel, a maradékot feladatokkal. Stílusában hasonló lesz, mint a korábbi vizsgák, de itt is előfordulhat, hogy nem 6 darab 10 pontos feladat lesz.

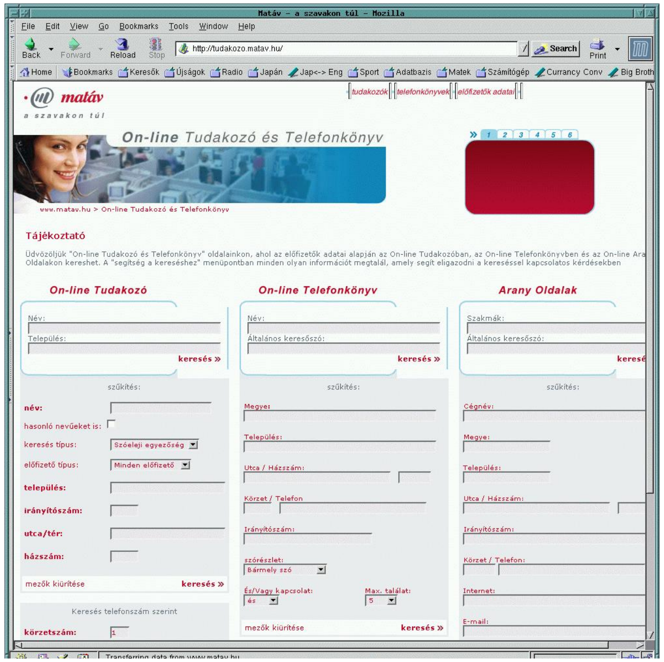
Aláírás és vizsga

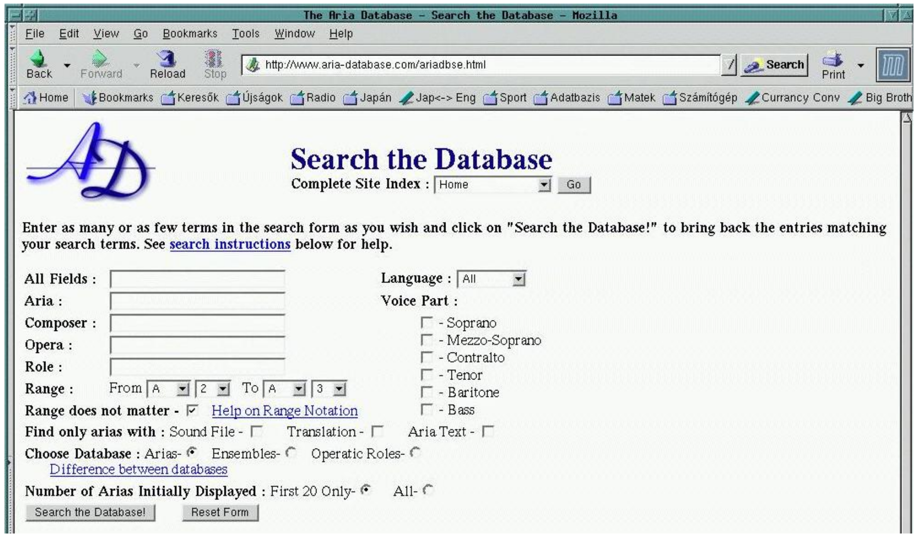
A megajánlott vizsgajegy: 20 ponttól kettes, 30-tól hármas, 40-től négyes és 50-től ötös, illetve ha a vizsgán szerzett pontszám legalább 20 (a vizsga legalább kettes) és a félévközi zh eredménye jobb, mint a vizsgáé, akkor a két pontszám átlagára kapott osztályzat. Ha a megajánlott jegy egyes, az mindenképp bekerül az indexbe. Ha a megajánlott jegy legalább kettes, lehetőség van egy szóbeli kérdésre felelni a kiosztáskor (ezt mindenki maga dönti el, hogy kéri-e), és ezzel a megajánlott jegyen legfeljebb lehet egyet javítani vagy rontani. Az eredményhirdetésre és a szóbeli feleletekre az írásbelik után néhány nappal, egy időben kerül sor.

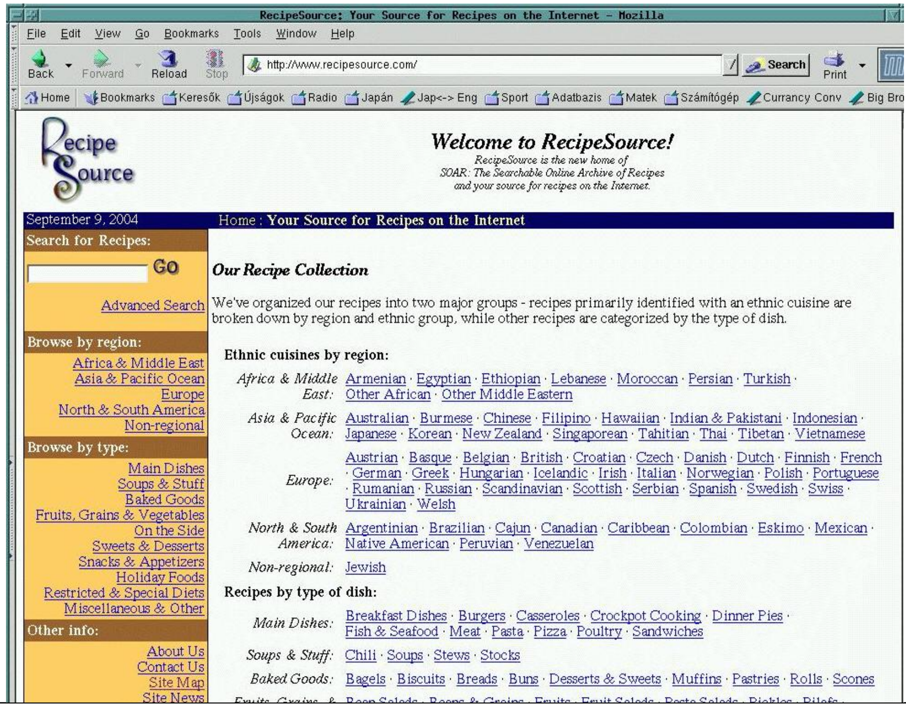
Adatbázis

telefonkönyv, könyvtár, notesz, internet









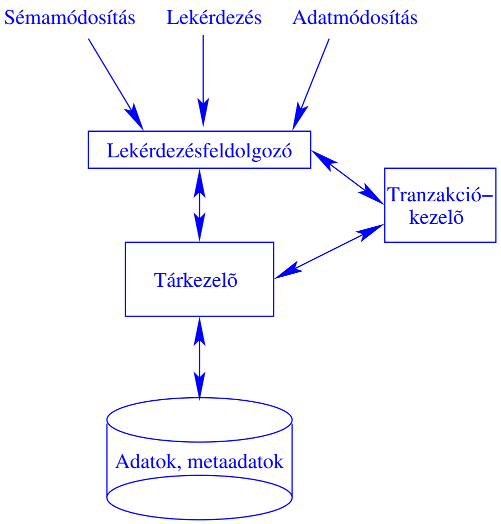
Adatbáziskezelő rendszerek (DBMS) jellemzői

- Komplex hardver-szoftver rendszer adatok kezelésére (tárolás, módosítás, lekérdezés)
- Nagy adatmennyiség
- Gazdag logikai struktúra (adatmodell) $\Longrightarrow$ magas szintű lekérdezés, módosítás (például egy légitársaságos adatbázisnál kategóriák lehetnek: utas, járat, pilóta)
- hosszú életciklus (jól kell tűrnie a hardver módosításait, a fizikai tárolás változásait)

Elvárások a DBMS-mel szemben

- Új séma létrehozása (fogalmak, metaadatok megadása) pl.: legyen egy *utas* kategória, ennek attribútumai legyenek: név, lakcím, vegetáriánus-e,...
  Ehhez adott egy rendszertől függő DDL (Data Definition Language)
- Adatok beillesztése, módosítása (az adatbázis fogalmi keretének feltöltése) pl.: új utas felvétele, elromlott gép törlése ...
  Ehhez adott egy rendszertől függő DML (Data Manipulation Language)
- Lekérdezés (infót kiszedni az adatbázisból)
  Ehhez Query Language (pl.: SQL)
  Cél: gyakori kérdéseket könnyű legyen kérdezni (ehhez jól végiggondolt fogalmi keret kell, hatékony tárolás $\Longrightarrow$ fontos a jó tervezés)
- Nagy mennyiségű adat tárolása biztonságosan (jogosultságok, illetéktelen hozzáférés megakadályozása illetve rendszerhibák elleni védelem)
- Többfelhasználós működés támogatása (egyidejű hozzáférés, de ne legyen hibás, inkonzisztens állapot emiatt)

A DBMS felépítése, részei



Adatok, metaadatok

Fizikailag valahol tárolódnak az adatok (milyen nevű utas, melyik gépre foglalt helyet) és a metaadatok (mik a relációk nevei és attribútumai és ezek típusai, illetve pl. milyen indexek vannak a kereséshez)

Tárkezelő

A kért információ beolvasása, módosítása. Kb. mint az OR-ek file kezelője, de itt néha (többfelhasználós működésnél pl.) mi mondjuk meg, hova történjen az írás (háttártárra, pufferbe), nem kezelheti teljesen szabadon a puffert.

Részei:

- File kezelő: nyilvántartja a DB állományát; fizikai I/O-t végez, ha a pufferkezelő kéri; indexstruktúrába rendezni az adatokat (pl. B-fa) (I/O mindig lemezblokkonként, hatékonyság mértéke az ilyen műveletek száma)
- Pufferkezelő: kezeli a memóriát, tárolja a filekezelő által beolvasott blokkokat

Lekérdezésfeldolgozó

Alapfeladata: sémadefiníciós, adatmódosítás és lekérdezős kérések fogadása, kezelése

- Sémaműveletek: a DB logikai struktúrájának kialakítása, módosítása
  eredménye: maga az adatbázisséma, plusz kiegészítő metaadatok (pl. mit hogyan tároljunk, indexek)
  Nagyon fontos, meghatározza a további működést
- Lekérdezések: keresőkérdések a DB-re vonatkozóan
  két lehetőség erre: vagy egy külön lekérdezőfelületen át vagy alkalmazói programból (a második esetben a host language-nek van utasításkészlete a DB-hez fordulásra)
- Adatmódosítás: a DB tartamának módosítása, beszúrás, törlés
  Itt is lehet külön felület vagy program

Lekérdezésfeldolgozó tennivalói

Hogyan kezeli ezeket a kéréseket?

1. Végrehajtási terv készítése

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

Pl: Ha egy banki séma két relációja
ügyfél(név, cím, szemszám) és
számla(szemszám, számlaszám, egyenleg)
és mi Bill Gates számlájának egyelegét szeretnénk megkapni, akkor erre egy SQL kérdés:

SELECT egyenleg FROM ügyfél, számla
WHERE ügyfél.szemszám=számla.szemszám AND ügyfél.név="Bill Gates"

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 az *ügyfél*-ben, akkor az alapján keressük meg Bill Gates személyi számát, aztán ez alapján a másik relációban keressük meg a megfelelő sort és olvassuk ki az egyenleget.

2. "Optimalizálás"

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

Tanulság:
1. A kapott kérdés nem ad támpontot, hogy hogyan kell megkapni az eredményt, de nem is kötelez semmire.
2. Érdemes szöszölni egy jobb végrehajtási terv keresésével, mert nagyok lehetnek a különbségek.

Tranzakciókezelő

Két nagyobb problémakör megoldására jó:

- Több felhasználó egyszerre használja a DB-t, egyidejű hozzáférések kezelése
- Rendszerhibák, ABORT-ok hatásainak kivédése: ezek bekövetkeztekor sem veszhetnek el adatok, nem maradhat a DB inkonzisztens állapotban

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

Elvárások a tranzakciókezelésben

- A (atomicity, atomiság): egy tranzakció vagy teljesen végrehajtódik vagy semmi se hajtódik végre belőle
- C (consistency, konzisztencia): vannak a rendszernek helyes, konzisztens állapotai, ezek között mozog. Inkonzisztens állapot csak ideiglenesen lehet.
- I (isolation, elkülönítés): több tranzakció egyidejű lefutása után úgy nézzen ki a rendszer, mintha a tranzakciók egymás után, elkülönülten futottak volna le. Tehát pl. az alábbi párhuzamos lefutás:

  hatása legyen ugyanaz, mint vagy a tr1 tr2 sorrend, vagy a tr2 tr1 sorrend hatása.
- D (durability, tartósság): ha egy tranzakció sikeresen befejeződött, akkor annak eredménye nem veszhet el később.

Eszközök a tranzakciókezelésben — ízelítő

- **Zárolás**: ha egy tranzakció csinálni akar valamit egy adattal, akkor zárat rak rá, ekkor más nem, vagy csak korlátozottan fér hozzá (lehet több féle zárat is használni). Zárolni lehet sort, blokkot, egész táblát is: minél nagyobbat zárolunk, annál könnyebb az adminisztráció, de annál többet kell a tranzakcióknak várniuk egymásra.
- **Naplózás**: van egy biztos, hibáktól védett helyen tartott napló, ahol minden történést feljegyzünk. Hiba esetén ez megmarad, innen vissza lehet állítani egy helyes állapotot.

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 rendszertől függően.

Az adatbáziskezelő használati szintjei

- Felhasználó: egyszerűbb SQL kérdést fogalmaz meg, adatmódosítást csinál, alkalmazói programot futtat
- Adatbázis programozó: összetett kérdések írása, alkalmazói program írása. Jól ismeri a DB felépítését, a DBMS-ben használt technikákat.
- Adatbázis tervező: sémát hoz létre, fizikai szervezésbe beleszólhat.
- Adatbázisrendszer megvalósító: az adatbáziskezelőt magát tervezi és írja.

A félév anyaga

Főleg a tervezés és megvalósítás kérdései:

- **Tervezés**:
  - ⋆ Adatmodellezés: adatbázis-tervező technikák (ODL, objektumos tervezés és E/K (egyed-kapcsolat) diagram, grafikus jelölésrendszer). Az adatbázis fogalmi keretének megadására jók, tervet lehet velük készíteni, amit aztán majd át kell alakítani az adatbáziskezelő által használt formális megadási módra.
  - ⋆ Relációs adatmodell: nagyon fontos, tipikusan ilyenek a DBMS-ek mostanában.
- **Tervezés/megvalósítás**:
  - ⋆ fizikai szervezés
  - ⋆ tranzakciókezelés
  - ⋆ lekérdezésfeldolgozás
- **Programozás**: nem nagyon lesz, majd laboron a következő félévben, de azért: SQL, adatmódosítás, adatdefiniálás, triggerek, megszorítások kezeléséről lesz szó.