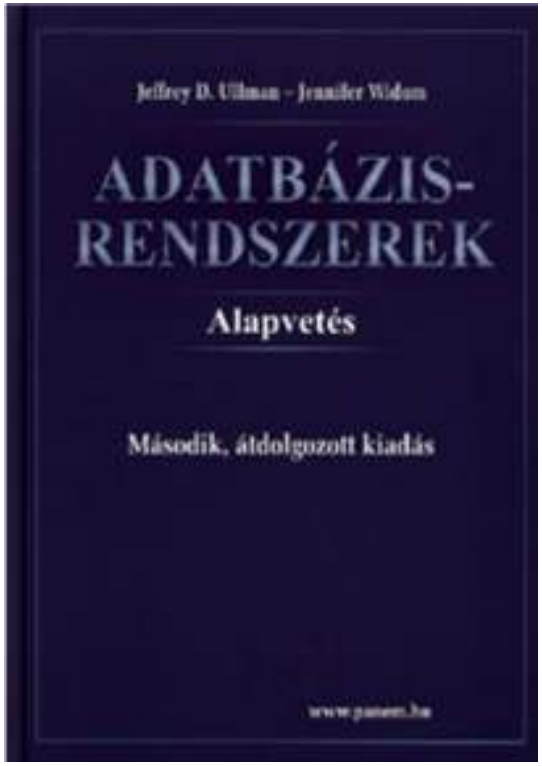
Adatbázis tartalmának módosítása
SQL DML utasítások

Tankönyv: Ullman-Widom:

Adatbázisrendszerek Alapvetés

Második, átdolgozott kiadás,

Panem, 2009



6.3. Alkérdések a záradékokban (folyt) (where, having és group by)

6.5. Változtatások az adatbázisban: SQL DML adatkezelő utasítások: INSERT, DELETE, UPDATE

1.fejezet: Adatbázis-kezelő rendszerek (alapfogalmak)

Alkérdések

- A FROM listán és a WHERE záradékban (valamint a GROUP BY HAVING záradékában) zárójelezett SFW SELECT-FROM-WHERE utasításokat (alkérdéseket) is használhatunk.
- Szintaktikus alakja: zárójelbe kell tenni a lekérdezést
- Hol használható? Ott, ahol relációnevet használunk:

(1) WHERE és HAVING záradékban: kifejezésekben, feltételekben

(2) FROM listában: új listaelem (rel.név változó SQL-ben)

(lekérdezés) [AS] sorváltozó

Ez felel meg annak, ahogyan a relációs algebrában tetsz.helyen használhattuk a lekérdezés eredményét.

Alkérdések használata FROM listán

- FROM záradékban alkérdéssel létrehozott ideiglenes táblát is megadhatunk. Ilyenkor a legtöbb esetben meg kell adnunk a sorváltozó nevét. **Szintaktikus alakja**:

  (lekérdezés) [AS] sorváltozó

- **Szemantikája**: A FROM záradékban kiértékelődik az alkérdés, utána a sorváltozót ugyanúgy használjuk, mint a közönséges adatbázis relációkat.

- **Példa:** Keressük meg a Joe's bár vendégei által kedvelt söröket (a feladatnak sok megoldása van)

Alkérdések használata FROM listán

- FROM záradékban alkérdéssel létrehozott ideiglenes táblát is megadhatunk. Ilyenkor a legtöbb esetben meg kell adnunk a sorváltozó nevét.
- **Példa:** Keressük meg a Joe's bár vendégei által kedvelt söröket.

Sörivók, akik látogatják Joe's bárját.

```
SELECT sör
FROM Szeret, (SELECT név
              FROM Látogat
              WHERE bár = 'Joe''s bar')JD
WHERE Szeret.név = JD.név;
```

Ismétlés: Alkérdések a WHERE záradékban
Fontos! Ugyanezt használjuk a mai anyagban SQL DML utasítások WHERE záradékában!

WHERE és HAVING záradékban:

(i) Az alkérdés eredménye egyetlen skalárérték, vagyis az alkérdés olyan, mint a konstans, ami egy új elemi kifejezésként tetszőleges kifejezésben használható.

(ii) Skalár értékekből álló multihalmaz logikai kifejezésekben használható: [NOT] EXISTS (lekérdezés)
kifejezés [NOT] IN (lekérdezés)
kifejezés Θ [ANY | ALL] (lekérdezés)

(iii) Teljes, többdimenziós tábla a visszatérő érték:
[NOT] EXISTS (lekérdezés)
(kif_1, … kif_n) [NOT] IN (lekérdezés)

Emlékeztető: Példa csoportosításra

```
SELECT onev, AVG(fizetes) + 100 emelt
FROM dolgozo d, osztaly o
WHERE d.oazon=o.oazon AND telephely='Bp'
GROUP BY o.oazon, onev
HAVING COUNT(dkod) > 3
ORDER BY onev;
```

$\tau_{onev}(\pi_{onev,átlagfiz+100 \rightarrow emelt}$
$(\sigma_{COUNT(dkod)>3}\ (\gamma_{o.oazon,onev,\ AVG(fizetes),\ COUNT(dkod)}$
$(\sigma_{telephely='Bp'}\ (d \bowtie o))))))$

--- HF: Az operátorok egymás utáni alkalmazását kifejezésfa formájában is rajzolhatjuk fel!

Példa alkérdésre a HAVING-ben --1

- Felszolgál(söröző, sör, ár) és Sörök(név, gyártó) táblák felhasználásával adjuk meg az átlagos árát azon söröknek, melyeket
  - legalább három sörözőben felszolgálnak,
  - vagy Pete a gyártójuk.

Példa alkérdésre a HAVING-ben --2

```
SELECT sör, AVG(ár)
FROM Felszolgál
GROUP BY sör
HAVING COUNT(söröző) >= 3  OR
     sör IN (SELECT név
             FROM Sörök
             WHERE gyártó = 'Pete');
```

(HAVING…) Sör csoportok, Melyeket legalább három nem-NULL bárban árulnak, Vagy Pete a gyártójuk.

(SELECT…) Sörök, melyeket Pete gyárt (ez az Ullman mo., de

H.F.: más mo-okkal

Adatbázis tartalmának módosítása

Tankönyv 6.5. Változtatások az adatbázisban

- A módosító utasítások nem adnak vissza eredményt, mint a lekérdezések, hanem az adatbázis tartalmát változtatják meg.
- 3-féle módosító utasítás létezik:

  INSERT - sorok beszúrása

  DELETE – sorok törlése

  UPDATE – sorok komponensei értékeinek módosítása

Beszúrás (insert into)

- Két alakja van:
- 1.) ha egyetlen sort szúrunk be:

  INSERT INTO <reláció>
  VALUES (<konkrét értékek listája>);

- 2.) ha több sort, egy lekérdezés eredményét visszük fel alkérdés segítségével:

  INSERT INTO <reláció>
  (<alkérdés>);

Beszúrás, attribútumok megadása

- Példa: A Szeret táblába beírjuk, Zsu szereti a Bud sört.

```
INSERT INTO Szeret
VALUES('Zsu', 'Bud');
```

- A reláció neve után megadhatjuk az attribútumait.
- Ennek alapvetően két oka lehet:
  1. elfelejtettük, hogy a reláció definíciójában, milyen sorrendben szerepeltek az attribútumok.
  2. Nincs minden attribútumnak értéke, és azt szeretnénk, ha a hiányzó értékeket NULL vagy default értékkel helyettesítenék.

Példa:

```
INSERT INTO Szeret(sör, név)
VALUES('Bud', 'Zsu');
```

Default értékek megadása

- A CREATE TABLE utasításban az oszlopnevet DEFAULT kulcsszó követheti és egy érték.
- Ha egy beszúrt sorban hiányzik az adott attribútum értéke, akkor a default értéket kapja.

```
CREATE TABLE Sörivók(
    név CHAR(30) PRIMARY KEY,
    cím CHAR(50) DEFAULT 'Sesame St'
    telefon CHAR(16)  );
INSERT INTO Sörivók(név)
    VALUES('Zsu');
```

Az eredmény sor:

név	cím	telefon
Zsu	Sesame St	NULL

Több sor beszúrása

- Egy lekérdezés eredményét is beszúrhatjuk:

  INSERT INTO <reláció>

  (<alkérdés>);

- A Látogat(név, söröző) tábla felhasználásával adjuk hozzá a LehetBarát(név) táblához Zsu „lehetséges barátait”, vagyis azokat a sörivókat, akik legalább egy olyan söröző látogatnak, ahova Zsu is szokott járni.

Megoldás: Több sor beszúrása

(SELECT) a másik sörivó

(FROM) névpárok:
az első Zsu,
a második nem Zsu,
de van olyan bár,
amit mindketten
látogatnak.

```
INSERT INTO LehetBarát
(SELECT l2.név
 FROM Látogat l1, Látogat l2
 WHERE l1.név = 'Zsu' AND
              l2.név <> 'Zsu' AND
             l1.söröző = l2.söröző
 );
```

Tk.Példa INSERT INTO utasításra

- A lekérdezést teljesen ki kell értékelni, mielőtt a sorokat beszúrnánk.
- Tankönyv 6.36 példa: új stúdiók beszúrása

```
INSERT INTO Stúdió (név)
        (SELECT DISTINCT stúdióNév
         FROM Filmek
         WHERE stúdióNév NOT IN
                (SELECT név  FROM Stúdió));
```

Törlés (delete)

- A törlendő sorokat egy WHERE feltétel segítségével adjuk meg:

  DELETE FROM <reláció>

  WHERE <feltétel>;

- Példa:

  DELETE FROM Szeret

  WHERE nev = 'Zsu' AND

  sör = 'Bud';

- Az összes sor törlése:

  DELETE FROM Szeret;

Példa: Több sor törlése

- A Sörök(név, gyártó) táblából töröljük azokat a söröket, amelyekhez létezik olyan sör, amit ugyanaz a cég gyártott.

Példa: Több sor törlése

- A Sörök(név, gyártó) táblából töröljük azokat a söröket, amelyekhez létezik olyan sör, amit ugyanaz a cég gyártott.

```
DELETE FROM Sörök s
WHERE EXISTS (
   SELECT név FROM Sörök
   WHERE gyártó = s.gyártó
      AND név <> s.név);
```

(WHERE) azok a sörök, amelyeknek ugyanaz a gyártója, mint az s éppen aktuális sorának, a nevük viszont különböző.

A törlés szemantikája

- Tegyük fel, hogy az Anheuser-Busch csak Bud és Bud Lite söröket gyárt.
- Tegyük fel még, hogy *s* sorai közt a Bud fordul elő először.
- Az alkérdés nem üres, a későbbi Bud Lite sor miatt, így a Bud törlődik.
- Kérdés: a Bud Lite sor törlődik-e?

A törlés szemantikája

- Válasz: igen, a Bud Lite sora is törlődik.
- A törlés ugyanis két lépésben hajtódik végre.
  1. Kijelöljük azokat a sorokat, amelyekre a WHERE feltétele teljesül.
  2. Majd töröljük a kijelölt sorokat.

Módosítás (update)

- Bizonyos sorok bizonyos attribútumainak módosítása.

  UPDATE <reláció>

  SET <attribútum értékadások listája>

  WHERE <sorokra vonatkozó feltétel>;

- Fecó telefonszámát 555-1212-re változtatjuk (Fecó itt egy sörivó neve):

  UPDATE Sörivók

  SET telefon = ’555-1212’

  WHERE név = ’Fecó’;

Példa: Több sor módosítása

- Legfeljebb 4 dollárba kerülhessenek a sörök:

```
UPDATE Felszolgál
SET ár = 4.00
WHERE ár > 4.00;
```

- Olcsó sörök árát duplázzuk

```
UPDATE Felszolgál
SET ár = 2 * ár
WHERE ár < 1.00;
```

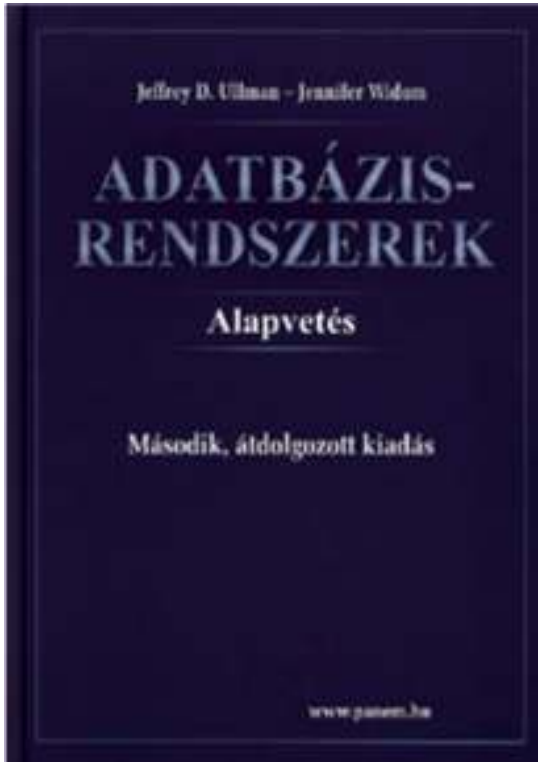
Tk.Példa UPDATE utasításra

- Tankönyv 6.39 példa:

```
UPDATE GyártásIrányító
SET név = 'Ig.' || név
WHERE azonosító IN
        (SELECT elnökAzon FROM Stúdió)
```

Adatbázis-kezelő rendszerek felépítése

Tankönyv: Ullman-Widom:
Adatbázisrendszerek Alapvetés
Második, átdolgozott kiadás,
Panem, 2009



1.fejezet: Az adatbázis-kezelő rendszerek (DBMS) felépítése, alapfogalmak, ACID tranzakciók

Az adatbázisrendszerek világa

Tk.1.fejezete Az adatbázis-kezelő rendszerek áttekintése

- Adatbázisok-1 kurzuson mit láttunk eddig és mit fogunk venni az adatbázisrendszerek világából?
  - Adatbázist, adatok gyűjteményét kezeli, relációs modell típus: sortípus, gyűjtemény: reláció
  - Hogyan tervezzük meg milyen gyűjteményünk legyen? Lesz majd tervezés: E/K modell, UML diagramok, Relációs adatbázis sématervezés (FF, TÉF, NF)
  - Metaadatok kezelése: DDL sémaleíró nyelv
  - Táblák tartalmának lekérdezése (select) és módosítása: insert-delete-update: DML adatkezelő nyelv
  - Lekérdezések feldolgozása: alap és kiterjesztett relációs algebra, SQL: SELECT, program (SQL/PSM, PL/SQL)

(1) Adatbázis-kezelés

Adatbázis-kezelés:

- (1) Háttértárolón tárolt, nagy adatmennyiség hatékony kezelése (lekérdezése, módosítása)
- (2) Adatmodell támogatása
- (3) Adatbázis-kezelő nyelvek támogatása
- (4) Több felhasználó támogatása
- (5) Adatvédelem, adatbiztonság
- (6) Tranzakció-kezelés
- (7) Konkurencia-kezelés
- (8) Naplózás és helyreállíthatóság
- (9) Lekérdezések végrehajtásának optimalizálása

(2) Adatmodell támogatása

- Az adatmodell a valóság fogalmainak, kapcsolatainak, tevékenységeinek magasabb szintű ábrázolása
  - File-kezelés indexekkel együtt, ezt váltotta fel a
  - CODASYL szabvány, hálós adatmodell (hatékony keresés)
  - Hierarchikus adatmodell (apa-fiú kapcsolatok gráfja)
  - Ted Codd - Relációs adatmodell (táblák rendszere, könnyen megfogalmazható műveletek)
  - Objektum-orientált adatmodell (az adatbázis-kezelés funkcionalitásainak biztosítása érdekében gyakran relációs adatmodellre épül), + Objektum-relációs adatmodell
  - Logikai adatmodell (szakértői rendszerek, tények és következtetési szabályok rendszere)
  - Dokumentumok - Félig strukturált adatmodell, az XML (szabvány adatcsereformaként jelent meg)

(3) Adatbázis-kezelő nyelvek támogatása

- SQL – relációs (és objektum-relációs) adatbázis-kezelő szabvány nyelv, fontosabb szabványok:

  SQL86, SQL89, SQL92 (SQL2), SQL:1999 (SQL3),

  SQL: 2003, SQL:2006, SQL:2008

- DDL (Data Definition Language) adatdefiniáló (sémaleíró) nyelv: sémák, adatstruktúrák megadása, objektumok létrehozása, módosítása, törlése: CREATE, ALTER, DROP
- DML (Data Manipulation Lang.) adatkezelő és lekérdező nyelv: INSERT, DELETE, UPDATE és SELECT
- DCL (Data Control Lang.) adatvezérlő nyelv, jogosultságok kiosztása és visszavonása: GRANT, REVOKE
- Tranzakció-kezelés: COMMIT, ROLLBACK

(4) Több felhasználó támogatása

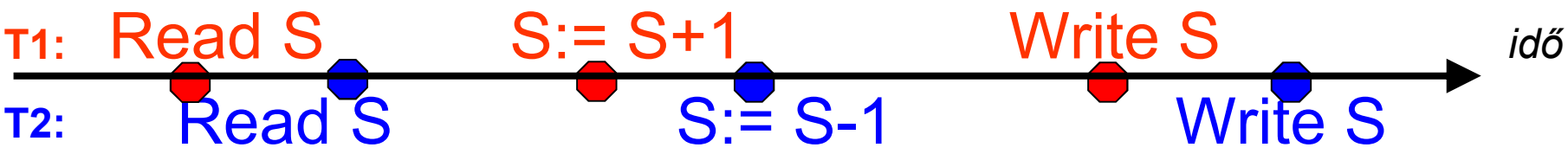
- Felhasználói csoportok. Kulcsemberek:
  - DBA adatbázis-rendszergazda
    - felügyeli az adatbázis-példányokat és adatbázis-szervereket
    - felépíti a rendszert, implementálja és optimális adatbázis-megoldást biztosít
  - Adatbázis-tervező (sématervezés)
  - Alkalmazás-fejlesztő, programozó (kódolás)
  - Felhasználók (akik használják a rendszert)

(5) Adatvédelem, adatbiztonság

- Jogosultságok (objektumok olvasása, írása, módosítása, készítése, törlése, jogok továbbadása, jogok visszavonása)
- GRANT és REVOKE utasítás
- Jogosultságok tárolása rendszertáblákban történik
- Jogosultságok kezelése, felhasználók, jelszavak, hozzáférési jogok
- Adatbázissémák korlátozása (virtuális) nézettáblák segítségével
- Tárolt adatok, hálózati adatforgalmak titkosítása (nagy prímszámok, RSA, DES)

(6) Tranzakció-kezelés

- Tranzakció: adatkezelő műveletekből (adategység írása, olvasása) álló sorozat
- Cél: tranzakciók párhuzamos végrehajtása



- Tranzakció = olyan folyamat, ami adatbázis lekérdezéseket, módosításokat tartalmaz.
- Az utasítások egy „értelmes egészt” alkotnak.
- Egyetlen utasítást tartalmaznak, vagy az SQL-ben explicit módon megadhatóak.

(6) Miért van szükség tranzakciókra?

- Az adatbázis rendszereket általában több felhasználó és folyamat használja egyidőben.
  - Lekérdezések és módosítások egyaránt történhetnek.
- Az operációs rendszerektől eltérően, amelyek támogatják folyamatok interakcióját, az adatbázis rendszereknek el kell különíteniük a folyamatokat.

(6) Példa: rossz interakció

- Egy időben ketten töltenek fel 100 dollárt ugyanarra a számlára ATM-en keresztül.
  - Az adatbázis rendszernek biztosítania kell, hogy egyik művelet se vesszen el.
- Ezzel szemben az operációs rendszerek megengedik, hogy egy dokumentumot ketten szerkesszenek egyidőben. Ha mind a ketten írnak, akkor az egyik változtatás elvész (elveszhet).

(6) Tranzakciók

- Tranzakció = olyan folyamat, ami adatbázis lekérdezéseket, módosításokat tartalmaz.
- Az utasítások egy „értelmes egészt” alkotnak.
- Egyetlen utasítást tartalmaznak, vagy az SQL-ben explicit módon megadhatóak.

(6) A tranzakciók ACID tulajdonságai

- Atomiság (atomicity): a tranzakció egységesen lefut vagy nem, vagy az összes vagy egy utasítás sem hajtódik végre.
- Konzisztencia (consistency): a tranzakció futása után konzisztens legyen az adatbázis, megszorításokkal, triggerekkel biztosítjuk.
- Elkülönítés (isolation): párhuzamos végrehajtás eredménye egymás utáni végrehajtással egyezzen meg
- Tartósság (durability): a befejezett tranzakció eredménye rendszerhiba esetén sem veszhet el

(6) COMMIT és ROLLBACK

- A COMMIT utasítás a tranzakció sikeres befejeződését eredményezi. Egy sikeresen befejeződött tranzakció a kezdete óta végrehajtott utasításainak módosításait tartósan rögzíti az adatbázisban
  - vagyis a módosítások *véglegesítődnek*.
- A ROLLBACK utasítás megszakítja a tranzakció végrehajtását, és annak sikertelen befejeződését eredményezi. Az így befejezett tranzakció SQL utasításai által végrehajtott módosításokat a rendszer meg nem történtekké teszi
  - Vagyis az összes utasítás *visszagörgetésre kerül*, a módosítások nem jelennek meg az adatbázisban.

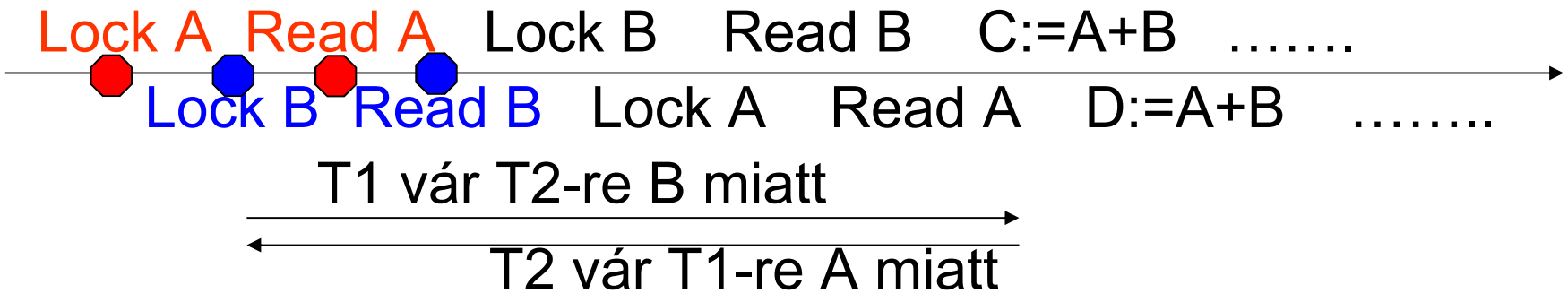
(7) Konkurencia-kezelés

- Zárolások (Lock, Unlock)

T1: (Lock S, Read S, S:=S+1, Write S, Unlock S)

T2: (Lock S, Read S, S:=S-1, Write S, Unlock S)

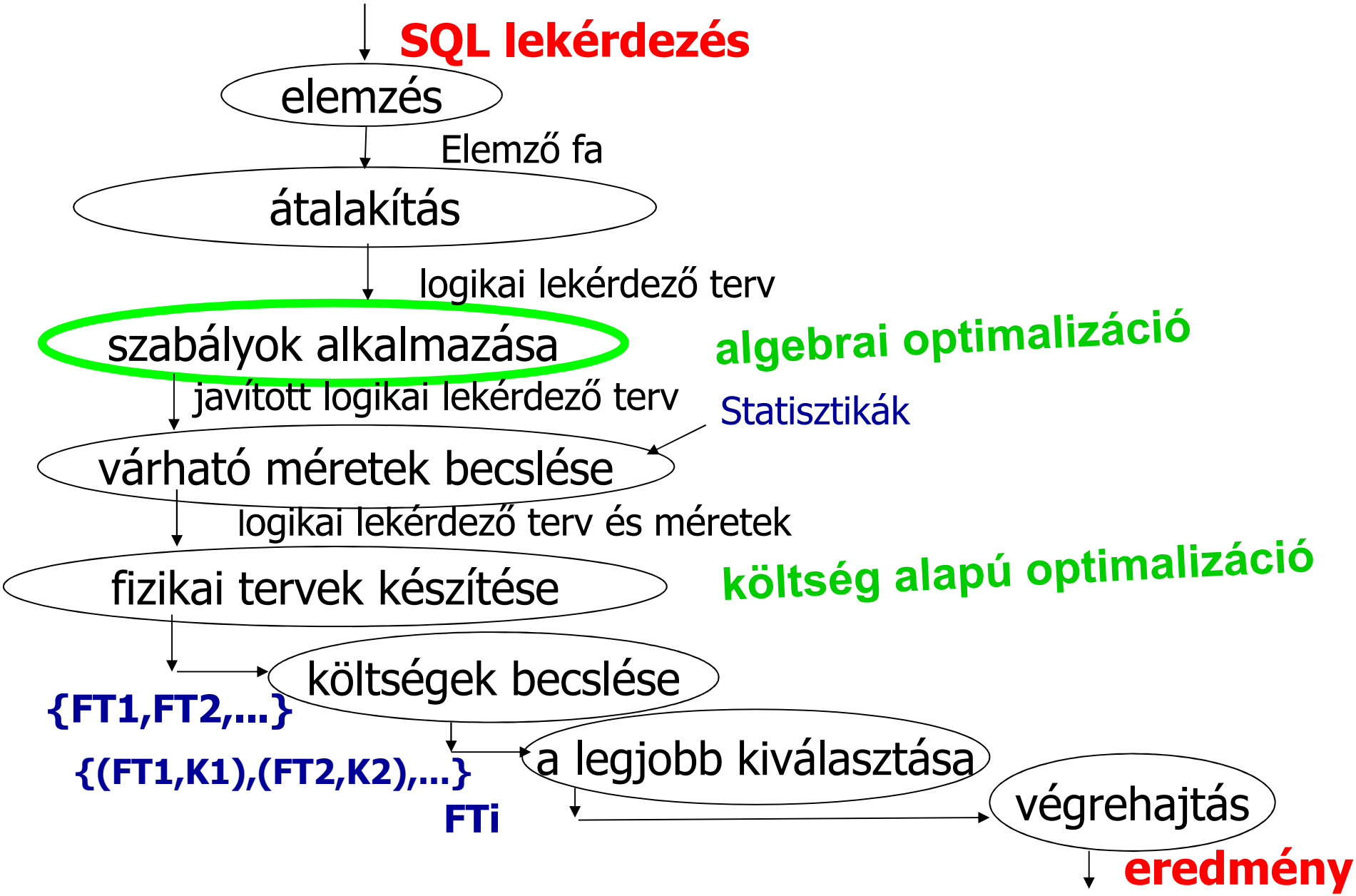
- A zár kiadásához meg kell várni a zár feloldását.
- Csökken a párhuzamosíthatóság
- Zárak finomsága (zárolt adategység nagysága, zárolás típusa) növeli a párhuzamosíthatóságot
- **Holtpont probléma:**



(8) Naplózás és helyreállítás

- Szoftver- vagy hardverhiba esetén az utolsó konzisztens állapot visszaállítása
- Rendszeres mentések
  - Statikus adatbázis (módosítás nem gyakori)
  - Dinamikus adatbázis (módosítás gyakori)
- Naplóállományok
- Összefügg a tranzakció-kezeléssel

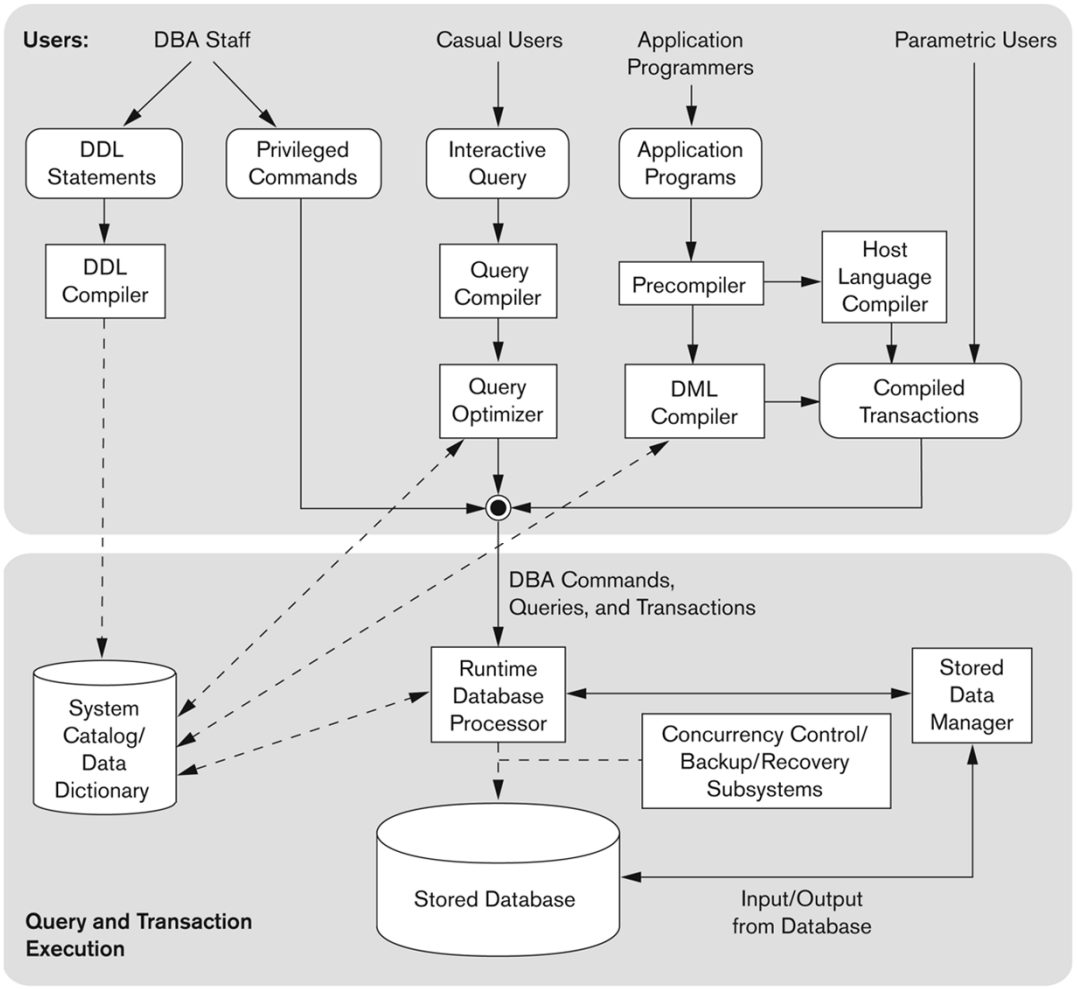
(9) Lekérdezések végrehajtása optimalizálás



Adatbázis-kezelők részei

- Lekérdezés-feldolgozó
  - Lekérdezés szintaktikai ellenőrzése
  - Adatbázis-objektumok létezésének, és a hozzáférési jogoknak az ellenőrzése (metaadatbázis, rendszertáblák)
  - Lekérdezés optimális átfogalmazása
  - Végrehajtási tervek készítése
  - Az adatstruktúrák, méretek statisztikái alapján várhatóan minimális költségű végrehajtási terv kiválasztása
  - Az optimális végrehajtási terv lefuttatása
- Tranzakció-kezelő:
  - Tranzakciók párhuzamos végrehajtásának biztosítása (**a**tomosság, **k**övetkezetesség, **e**lkülönítés, **t**artósság)
- Tárkezelő és pufferkezelő
  - fizikai adatstruktúrák, táblák, indexek, pufferek kezelése

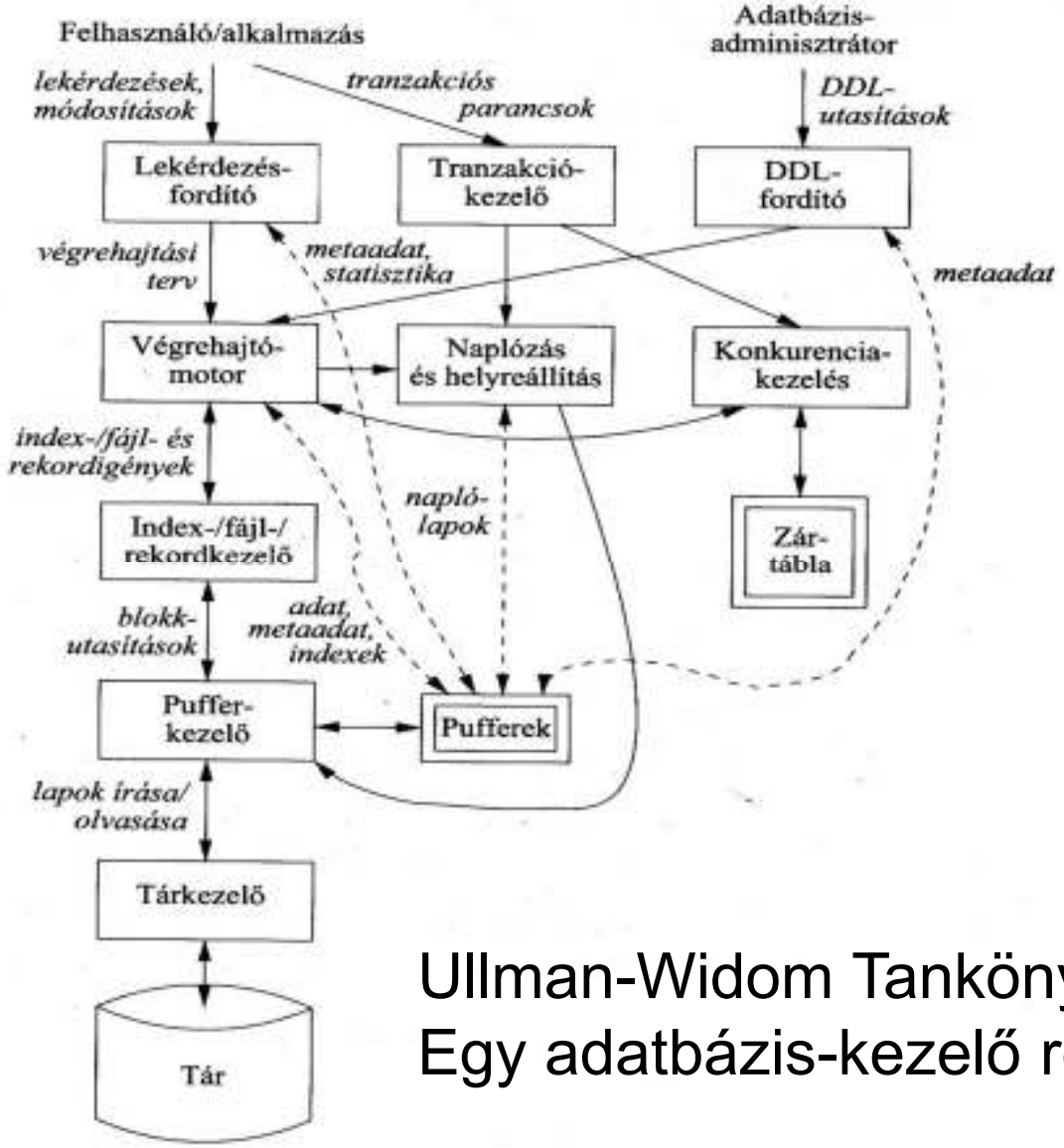
Adatbázis-kezelő rendszer felépítése



Forrás:
Elmasri-Navathe: Fundamentals of Database Systems

Figure 2.3
Component modules of a DBMS and their interactions.

Adatbázis-kezelő rendszer felépítése



Ullman-Widom Tankönyv 1.1. ábra
Egy adatbázis-kezelő rendszer részei

Kérdés/Válasz

- Köszönöm a figyelmet! Kérdés/Válasz?
- Gyakorlás a 5EA-hoz: Több táblára (DEPT és EMP tábla) vonatkozó lekérdezésekre példák, összekapcsolások.
- Házi feladat: Oracle Példatár 3.fejezet feladatai, összekapcsolások és alkérdések használata, de a hierarchikus és rekurzív lekérdezések még nem:

  http://people.inf.elte.hu/sila/eduAB/Feladatok.pdf