

# Megszorítások (constraints) és triggerek, nézettáblák és indexek

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

---

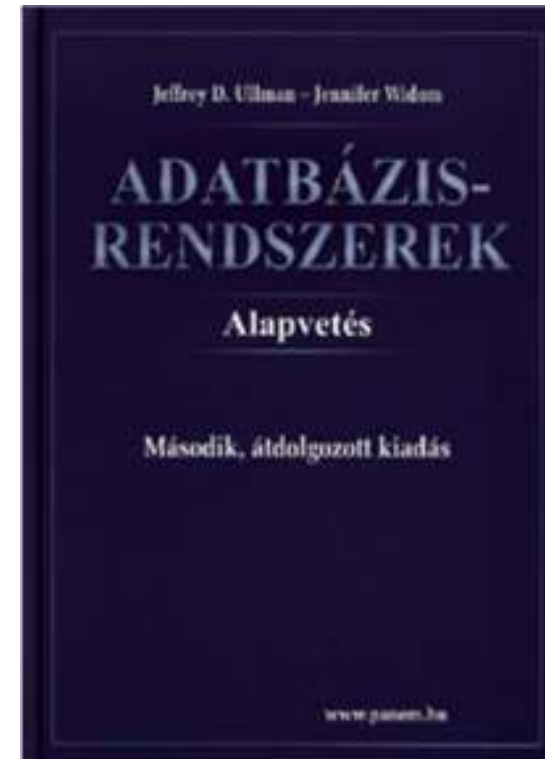
7.1.-7.4. Megszorítások

7.5.-7.6. Triggerek

8.1.-8-2. Nézettáblák

8.5.-8.6. Tárolt nézettáblák

-- Megj.: 8.3.-8.4. Indexek (Adatbázisok-2 kurzuson lesznek)



# Ismétlés: Relációs lekérdező nyelvek

Adatbázisok-1 kurzuson háromféle nyelvet tanulmányozunk:

- **Relációs algebra:** procedurális, algebrai megközelítés, megadjuk a kiértékelési tervet, többféle lehetőség összevetése, hatékonysági vizsgálatok.
- **Datalog:** deklaratív, logika alapú megközelítés, amely az összetett lekérésekénél, például rekurziónál segítség.
- **SQL szabvány relációs lekérdező nyelv:** gyakorlatban, SQL története, szabványok, az SQL fő komponensei: SQL DDL (sémaleíró nyelv) **milyen objektumok lehetnek?** DML (adatkezelő és lekérdező nyelv), tranzakció-kezelés, DCL (vezérlő nyelv) **milyen jogosultságok, privilégiumok?** SQL-2003/PSM, ezt a gyakorlatban: PL/SQL (Oracle)

--- Ma az SQL DDL sémaleíró nyelv utasításait nézzük át!

# Ismétlés: relációsémák definiálása

- Az SQL tartalmaz **adateleíró részt (DDL)**, az adatbázis **objektumainak** a leírására és megváltoztatására. **Objektumok** leíró parancsa a **CREATE** utasítás.
- **CREATE** – létrehozni, az objektumok leíró parancsa
- **DROP** – eldobni, a teljes leírást és mindazt, ami ehhez kapcsolódott hozzáférhetetlenné válik
- **ALTER** – módosítani a leírást
- A relációt az SQL-ben táblának (TABLE) nevezik, az SQL alapvetően háromféle táblát kezel:
  - Alaptáblák (permanens) **CREATE TABLE**
  - Nézetáblák **CREATE VIEW**
  - Átmeneti munkatáblák (WITH utasítás)
- **Alaptáblák** megadása: **CREATE TABLE**

# Megszorítások (áttekintés) => (1)

## (1) Kulcsok és idegen kulcsok megadása

- A hivatkozási épség fenntartása
- Megszorítások ellenőrzésének késleltetése

## (2) **Értékekre vonatkozó feltételek**

- NOT NULL feltételek
- Attribútumra vonatkozó CHECK feltételek

## (3) **Sorokra vonatkozó megszorítások**

- Sorra vonatkozó CHECK feltételek

## (4) **Megszorítások módosítása (constraints)**

## (5) **Önálló megszorítások (assertions)**

## (6) **Triggerek (triggers)**

# Idegen kulcsok megadása

- Az első előadáson a táblák létrehozásánál vettünk kiegészítő lehetőségeket **Kulcs és idegen kulcs (foreign key) hivatkozási épség megadása**
- Az egyik tábla egyik oszlopában szereplő értékeknek szerepelnie kell egy másik tábla bizonyos attribútumának az értékei között.
- **A hivatkozott attribútumoknak** a másik táblában kulcsnak kell lennie! (PRIMARY KEY vagy UNIQUE)
- **Példa: Felszolgál(söröző, sör, ár)** táblára megszorítás, hogy a sör oszlopában szereplő értékek szerepeljenek a **Sörök(név, gyártó)** táblában a név oszlop értékei között.

# Idegen kulcs megadása: attribútumként

**REFERENCES** kulcsszó használatának két lehetősége:  
attribútumként vagy sémaelemként lehet megadni.

## 1.) Attribútumonként (egy attribútumból álló kulcsra)

Példa:

```
CREATE TABLE Sörök (  
    név      CHAR(20) PRIMARY KEY,  
    gyártó  CHAR(20) );
```

```
CREATE TABLE Felszolgál (  
    söröző  CHAR(20) ,  
    sör     CHAR(20) REFERENCES Sörök(név) ,  
    ár     REAL );
```

# Idegen kulcs megadása: sémaelemként

2.) Sémaelemként (egy vagy több attr.-ból álló kulcsra)

**FOREIGN KEY** (attribútum lista)

**REFERENCES** relációnév (attribútum lista)

Példa:

```
CREATE TABLE Sörök (  
    név          CHAR(20),  
    gyártó       CHAR(20),  
    PRIMARY KEY (név) );
```

```
CREATE TABLE Felszolgál (  
    söröző       CHAR(20),  
    sör          CHAR(20),  
    ár           REAL,  
    FOREIGN KEY (sör) REFERENCES Sörök(név) );
```

# Idegen kulcs megszorítások megőrzése

- Példa:  $R = \text{Felszolgál}$ ,  $S = \text{Sörök}$ .
- Egy idegen kulcs megszorítás  $R$  relációról  $S$  relációra kétféleképpen sérülhet:
  1. Egy  $R$ -be történő beszúrásnál vagy  $R$ -ben történő módosításnál  $S$ -ben nem szereplő értéket adunk meg.
  2. Egy  $S$ -beli törlés vagy módosítás „lógó” sorokat eredményez  $R$ -ben.



# Hogyan védekezzünk? --- (1)

- Példa:  $R = \text{Felszolgál}$ ,  $S = \text{Sörök}$ .
- Nem engedjük, hogy **Felszolgál** táblába a **Sörök** táblában nem szereplő sört szűrjanak be vagy **Sörök** táblában nem szereplő sörre módosítsák (nincs választási lehetőségünk, a rendszer visszautasítja a megszorítást sértő utasítást)
- A **Sörök** táblából való törlés vagy módosítás, ami a **Felszolgál** tábla sorait is érintheti (mert sérül az idegen kulcs megszorítás) 3-féle módon kezelhető (lásd köv.oldal)

# Hogyan védekezzünk? --- (2)

1. **Alapértelmezés (Default)** : a rendszer nem hajtja végre a törlést.
2. **Továbbgyűrűzés (Cascade)**: a Felszolgál tábla értékeit igazítjuk a változáshoz.
  - **Sör törlése**: töröljük a Felszolgál tábla megfelelő sorait.
  - **Sör módosítása**: a Felszolgál táblában is változik az érték.
3. **Set NULL**: a sör értékét állítsuk NULL-ra az érintett sorokban.

# Példa: továbbgyűrűzés

- Töröljük a Bud sort a **Sörök** táblából:
  - az összes sort töröljük a **Felszolgál** táblából, ahol sör oszlop értéke 'Bud'.
- A 'Bud' nevet 'Budweiser'-re változtatjuk:
  - a **Felszolgál** tábla soraiban is végrehajtjuk ugyanezt a változtatást.

# Példa: Set NULL

- A Bud sort töröljük a **Sörök** táblából:
  - a **Felzolgál** tábla **sör** = 'Bud' soraiban a Budot cseréljük NULL-ra.
  - 'Bud'-ról 'Budweiser'-re módosítunk:
    - ugyanazt kell tennünk, mint törléskor.

# A stratégia kiválasztása

- Ha egy idegen kulcsot deklarálunk megadhatjuk a SET NULL és a CASCADE stratégiát is beszúrásra és törlésre is egyaránt.
- Az idegen kulcs deklarálása után ezt kell írunk:  
ON [UPDATE, DELETE][SET NULL CASCADE]
- Ha ezt nem adjuk meg, a default stratégia működik.

# Példa: stratégia beállítása

```
CREATE TABLE Felszolgál (  
    söröző CHAR(20),  
    sör     CHAR(20),  
    ár     REAL,  
    FOREIGN KEY(sör)  
        REFERENCES Sörök(név)  
        ON DELETE SET NULL  
        ON UPDATE CASCADE  
);
```

# Megszorítások ellenőrzésének késleltetése

- Körkörös megszorítások miatt szükség lehet arra, hogy a megszorításokat ne ellenőrizze, amíg az egész tranzakció be nem fejeződött.
- Bármelyik megszorítás deklarálnak **DEFERRABLE** (késleltethető) vagy **NOT DEFERRABLE**-ként (vagyis minden adatbázis módosításkor a megszorítás közvetlenül utána ellenőrzésre kerül). **DEFERRABLE**-ként deklarálnak, akkor lehetőségünk van arra, hogy a megszorítás ellenőrzésével várjon a rendszer a tranzakció végéig.
- Ha egy megszorítás késleltethető, akkor lehet
  - **INITIALLY DEFERRED** (az ellenőrzés a tranzakció jóváhagyásáig késleltetve lesz) vagy
  - **INITIALLY IMMEDIATE** (minden utasítás után ellenőrzi)

# Megszorítások (áttekintés) => (2)

## (1) Kulcsok és idegen kulcsok

- A hivatkozási épség fenntartása
- Megszorítások ellenőrzésének késleltetése

## (2) Értékekre vonatkozó feltételek

- NOT NULL feltételek
- Attribútumra vonatkozó CHECK feltételek

## (3) Sorokra vonatkozó megszorítások

- Sorra vonatkozó CHECK feltételek

## (4) Megszorítások módosítása (constraints)

## (5) Önálló megszorítások (assertions)

## (6) Triggerek (triggers)



# Értékekre vonatkozó feltételek

- Egy adott oszlop értékeire vonatkozóan adhatunk meg megszorításokat.
- A CREATE TABLE utasításban az attribútum deklarációban **NOT NULL** kulcsszóval
- az attribútum deklarációban **CHECK(<feltétel>)**  
A **feltétel**, mint a WHERE feltétel, alkérdés is használható. A feltételben csak az adott attribútum neve szerepelhet, más attribútumok (más relációk attribútumai is) csak alkérdésben szerepelhetnek.

# Példa: értékekre vonatkozó feltétel

```
CREATE TABLE Felszolgál (  
    söröző CHAR(20) NOT NULL,  
    sör     CHAR(20) CHECK ( sör IN  
        (SELECT név FROM Sörök)),  
    ár     REAL CHECK ( ár <= 5.00 )  
);
```

# Mikor ellenőrzi?

- Érték-alapú ellenőrzést csak **beszúrásnál** és **módosításnál** hajt végre a rendszer.
- **Példa:** CHECK (ár <= 5.00) a beszúrt vagy módosított sor értéke nagyobb 5, a rendszer nem hajtja végre az utasítást.
- **Példa:** CHECK (sör IN (SELECT név FROM Sörök)), ha a Sörök táblából törölünk, ezt a feltételt nem ellenőrzi a rendszer.

# Megszorítások (áttekintés) => (3)

## (1) Kulcsok és idegen kulcsok

- A hivatkozási épség fenntartása
- Megszorítások ellenőrzésének késleltetése

## (2) Értékekre vonatkozó feltételek

- NOT NULL feltételek
- Attribútumra vonatkozó CHECK feltételek

## (3) Sorokra vonatkozó megszorítások

- Sorra vonatkozó CHECK feltételek

## (4) Megszorítások módosítása (constraints)

## (5) Önálló megszorítások (assertions)

## (6) Triggerek (triggers)

# Sorokra vonatkozó megszorítások

- A **CHECK (<feltétel>)** megszorítás a séma elemeként is megadható.
- A feltételben tetszőleges oszlop és reláció szerepelhet.
  - De más relációk attribútumai csak alkérdésben jelenhetnek meg.
- Csak beszúrásnál és módosításnál ellenőrzi a rendszer.

# Példa: sor-alapú megszorítások

- Csak Joe bárja nevű sörözőben lehetnek drágábbak a sörök 5 dollárnál:

```
CREATE TABLE Felszolgál (  
    söröző CHAR(20),  
    sör     CHAR(20),  
    ár      REAL,  
    CHECK (söröző= 'Joe bárja' OR ár <= 5.00)  
);
```

# Tankönyv példája

Attribútumokra és sorokra vonatkozó megszorítások

Példa: Ha egy színész neme férfi, akkor  
a neve nem kezdődhet 'Ms.'-el

```
CREATE TABLE FilmSzínész (  
    név CHAR(30) PRIMARY KEY,  
    cím VARCHAR(255) NOT NULL,  
    nem CHAR(1),  
    születésiDátum DATE,  
    CHECK (nem = 'N' OR név NOT LIKE 'Ms.%')  
);
```

# Megszorítások (áttekintés) => (4)

## (1) Kulcsok és idegen kulcsok

- A hivatkozási épség fenntartása
- Megszorítások ellenőrzésének késleltetése

## (2) Értékekre vonatkozó feltételek

- NOT NULL feltételek
- Attribútumra vonatkozó CHECK feltételek

## (3) Sorokra vonatkozó megszorítások

- Sorra vonatkozó CHECK feltételek

## (4) Megszorítások módosítása (constraints)

## (5) Önálló megszorítások (assertions)

## (6) Triggerek (triggers)



# Megszorítások elnevezése

- Nevet tudunk adni a megszorításoknak, amire később tudunk hivatkozni (könnyebben lehet törölni, módosítani)

Tankönyv példái:

- név CHAR(30) **CONSTRAINT** NévKulcs PRIMARY KEY,
- nem CHAR(1) CONSTRAINT FérfiVagyNő  
CHECK (nem IN ('F', 'N')),
- CONSTRAINT Titulus CHECK (nem = 'N' OR  
név NOT LIKE 'Ms.\%')

# Megszorítások módosítása

Tankönyv példái:

- **ALTER TABLE** FilmSzínész **ADD CONSTRAINT** NévKulcs PRIMARY KEY (név);
- ALTER TABLE FilmSzínész ADD CONSTRAINT FérfiVagyNő CHECK (nem IN ('F', 'N'));
- ALTER TABLE FilmSzínész ADD CONSTRAINT Titulus CHECK (nem = 'N' OR név NOT LIKE 'Ms.\%');

# Megszorítások (áttekintés) => (5)

## (1) Kulcsok és idegen kulcsok

- A hivatkozási épség fenntartása
- Megszorítások ellenőrzésének késleltetése

## (2) Értékekre vonatkozó feltételek

- NOT NULL feltételek
- Attribútumra vonatkozó CHECK feltételek

## (3) Sorokra vonatkozó megszorítások

- Sorra vonatkozó CHECK feltételek

## (4) Megszorítások módosítása (constraints)

## (5) Önálló megszorítások (assertions)

## (6) Triggerek (triggers)

# Önálló megszorítások: Assertions

- SQL aktív elemek közül a leghatékonyabbak nincs hozzárendelve sem sorokhoz, sem azok komponenseihez, hanem **táblához kötődnek**.
- Ezek is **az adatbázissémához tartoznak** a relációsémákhoz és nézetekhez hasonlóan.
- **CREATE ASSERTION** <név>  
**CHECK** (<feltétel>);
- A feltétel tetszőleges táblára és oszlopra hivatkozhat az adatbázissémából.

# Példa: önálló megszorítások

- Az Sörvívó(név, cím, telefon) és Söröző(név, cím, engedély) táblákban nem lehet több bár, mint amennyi sörívó van.

```
CREATE ASSERTION KevésBár CHECK (  
    (SELECT COUNT(*) FROM Söröző) <=  
    (SELECT COUNT(*) FROM Sörívó)  
);
```

# Önálló megszorítások ellenőrzése

- Alapvetően az adatbázis bármely módosítása előtt ellenőrizni kell.
- Egy okos rendszer felismeri, hogy mely változtatások, mely megszorításokat érinthetnek.
  - **Példa:** a **Sörök** tábla változásai nincsenek hatással az iménti KevésBár megszorításra. Ugyanez igaz a **Sörivók** táblába történő beszúrásokra is.

# Megszorítások (áttekintés) => (6)

## (1) Kulcsok és idegen kulcsok

- A hivatkozási épség fenntartása
- Megszorítások ellenőrzésének késleltetése

## (2) Értékekre vonatkozó feltételek

- NOT NULL feltételek
- Attribútumra vonatkozó CHECK feltételek

## (3) Sorokra vonatkozó megszorítások

- Sorra vonatkozó CHECK feltételek

## (4) Megszorítások módosítása (constraints)

## (5) Önálló megszorítások (assertions)

## (6) Triggerek (triggers)

# Megszorítások v.s. triggerek

- **Aktív elemek** – olyan kifejezés vagy utasítás, amit egyszer eltároltunk az adatbázisban és az azt várjuk tőle, hogy a megfelelő pillanatban lefusson (pl. adatok helyességének ellenőrzése)
- **A megszorítás** adatelemek közötti kapcsolat, amelyet az adatbázis-kezelő rendszernek fent kell tartania.
- **Triggerek** olyankor hajtódnak végre, amikor valamilyen megadott esemény történik, mint például sorok beszúrása egy táblába.



# Miért hasznosak a triggererek?

- **Az önálló megszorításokkal** (assertions) sok mindent le tudunk írni, az ellenőrzésük azonban gondot jelenthet.
- **Az attribútumokra és sorokra vonatkozó megszorítások** ellenőrzése egyszerűbb (tudjuk mikor történik), ám ezekkel nem tudunk minden kifejezni.
- **A triggererek** esetén a felhasználó mondja meg, hogy egy megszorítás mikor kerüljön ellenőrzésre.

# Esemény-Feltétel-Tevékenység szabályok

- A triggereket esetenként *ECA szabályoknak* (*event-condition-action*) **esemény-feltétel-tevékenység** szabályoknak is nevezik.
- **Esemény**: általában valamilyen módosítás a adatbázisban, INSERT, DELETE, UPDATE.
- **Mikor?**: BEFORE, AFTER, INSTEAD
- **Mit?**: OLD ROW, NEW ROW FOR EACH ROW  
OLD/NEW TABLE FOR EACH STATEMENT
- **Feltétel** : SQL igaz-hamis-ismeretlen feltétel.
- **Tevékenység** : SQL utasítás, BEGIN..END,  
PSM tárolt eljárás

# Példa triggerre

- Ahelyett, hogy visszautasítanánk a **Felhasználó(söröző, sör, ár)** táblába történő beszúrást az ismeretlen sörök esetén, a **Sörök(név, gyártó)** táblába is beszúrjuk a megfelelő sort a gyártónak NULL értéket adva.

# Példa: trigger definíció

```
CREATE TRIGGER SörTrig
  AFTER INSERT ON Felszolgál
  REFERENCING NEW ROW AS ÚjSor
  FOR EACH ROW
  WHEN (ÚjSor.sör NOT IN
        (SELECT név FROM Sörök))
  INSERT INTO Sörök(név)
    VALUES(ÚjSor.sör);
```

Esemény

Feltétel

Tevékenység

# Triggerek --- 1

- A *triggerek*, amelyeket szokás *esemény-feltétel-tevékenység* szabályoknak is nevezni, az eddigi megszorításoktól három dologban térnek el:
- A triggeret a rendszer csak akkor ellenőrzi, ha bizonyos *események* bekövetkeznek.  
A megengedett események általában egy adott relációra vonatkozó beszúrás, törlés, módosítás, vagy a tranzakció befejeződése.

# Triggererek --- 2

- A kiváltó esemény azonnali megakadályozása helyett a trigger először egy *feltételt* vizsgál meg
- Ha a trigger feltétele teljesül, akkor a rendszer végrehajtja a triggerhez tartozó *tevékenységet*. Ez a művelet ezután megakadályozhatja a kiváltó esemény megtörténtét, vagy meg nem történtté teheti azt.

# Tankönyv példája (7.5. ábra)

-- Nem engedi csökkenteni a gyártásirányítók nettó bevételét:

```
CREATE TRIGGER NetBevétTrigger
AFTER UPDATE OF nettóBevétel ON GyártásIrányító
REFERENCING
    OLD ROW AS RégiSor,
    NEW ROW AS ÚjSor
FOR EACH ROW
WHEN(RégiSor.nettóBevétel > ÚjSor.nettóBevétel)
UPDATE GyártásIrányító
SET nettóBevétel = RégiSor.nettóBevétel
WHERE azonosító = ÚjSor.azonosító;
```

# Tankönyv példája (7.6. ábra)

-- Az átlagos nettó bevétel megszorítása:

```
CREATE TRIGGER ÁtlagNetBevétTrigger
AFTER UPDATE OF nettóBevétel ON GyártásIrányító
REFERENCING
    OLD TABLE AS RégiAdat,
    NEW TABLE AS ÚjAdat
FOR EACH STATEMENT
WHEN(500000 > (SELECT AVG(nettóBevétel)
              FROM GyártásIrányító)
DELETE FROM GyártásIrányító
WHERE (név, cím, azonosító) IN ÚjAdat;
INSERT INTO gyártásIrányító (SELECT * FROM ...)
```



# Nézettáblák

- Ez volt a Tankönyv 7.fejezete az integritási megszorításokról és a triggererekről.
- Ezután következik a Tankönyv 8.fejezete a nézettáblákról, és az adatok módosításáról a nézettáblákon keresztül.

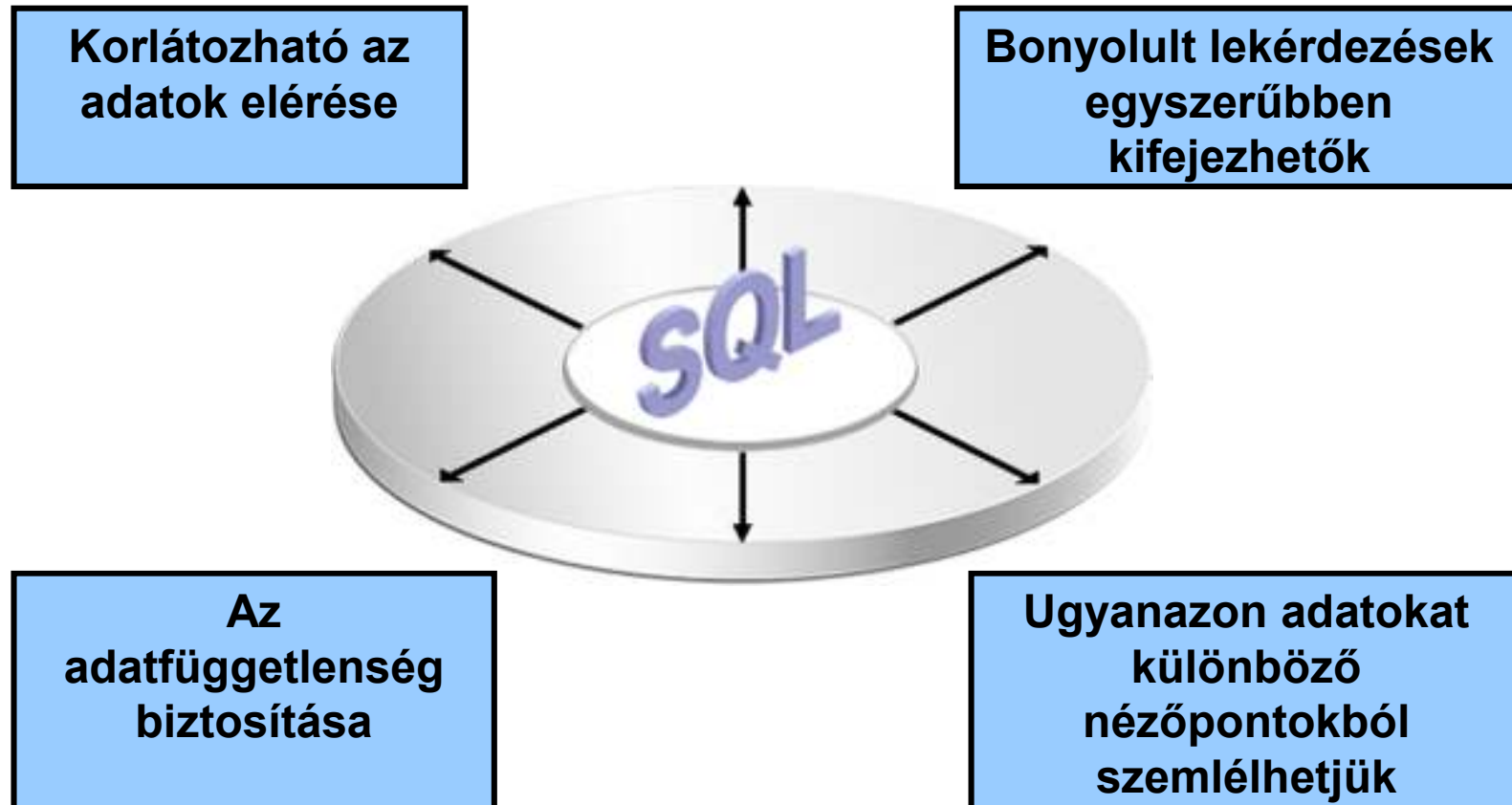
# Mik a nézettáblák?

- **A nézettábla** olyan reláció, amit tárolt táblák (vagyis alaptáblák) és más nézettáblák felhasználásával definiálunk.
- **EMPLOYEES table**

| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | EMAIL    | PHONE_NUMBER | HIRE_DATE | JOB_ID     | SALA |
|-------------|------------|-----------|----------|--------------|-----------|------------|------|
| 100         | Steven     | King      | SKING    | 515.123.4567 | 17-JUN-87 | AD_FRES    | 240  |
| 101         | Neena      | Kochhar   | NKOCHHAR | 515.123.4568 | 21-SEP-89 | AD_VP      | 170  |
| 102         | Lex        | De Haan   | LDEHAAN  | 515.123.4569 | 13-JAN-93 | AD_VP      | 170  |
| 103         | Alexander  | Hunold    | AHUNOLD  | 590.423.4567 | 03-JAN-90 | IT_PROG    | 90   |
| 104         | Bruce      | Burns     | BBURNS   | 590.423.4568 | 01-MAY-91 | IT_PROG    | 60   |
| 107         | Diana      | Lorentz   | DLorentz | 590.423.4567 | 07-FEB-98 | IT_PROG    | 42   |
| 124         | Irene      | Mouery    | IMOUERY  | 650.123.5234 | 18-NOV-99 | ST_MAN     | 58   |
| 141         | Trenta     | Fay       | TFAY     | 650.121.8009 | 17-OCT-95 | ST_CLERK   | 35   |
| 142         | Curtis     | Davies    | CDAVIES  | 950.121.2094 | 29-JAN-97 | ST_CLERK   | 31   |
| 143         | Randall    | Matos     | RMATOS   | 620.121.2074 | 15-MAR-90 | ST_CLERK   | 20   |
|             |            |           |          |              | JUL-96    | ST_CLERK   | 25   |
|             | 149        | Zlotkey   |          | 10500        | JAN-00    | SA_MAN     | 105  |
|             | 174        | Abel      |          | 11000        | MAY-96    | SA_REP     | 110  |
|             | 170        | Taylor    |          | 06000        | MAR-96    | SA_REP     | 86   |
| 170         | Ramanujam  | Grant     | RGRANT   | 515.124.1094 | 24-MAY-99 | SA_REP     | 70   |
| 200         | Jennifer   | Whalen    | JWHALEN  | 515.123.4444 | 17-SEP-87 | AD_ASST    | 44   |
| 201         | Michael    | Hartstein | MHARTSTE | 515.123.5555 | 17-FEB-96 | MK_MAN     | 130  |
| 202         | Pat        | Fay       | PFAY     | 603.123.6666 | 17-AUG-97 | MK_REP     | 60   |
| 205         | Shelley    | Higgins   | SHIGGINS | 515.123.8080 | 07-JUN-94 | AC_MGR     | 120  |
| 206         | William    | Gietz     | WGIEZT   | 515.123.8181 | 07-JUN-94 | AC_ACCOUNT | 83   |

20 rows selected.

# A nézettáblák előnyei



# Virtuális vagy materializált?

- Kétféle nézettábla létezik:
  - **Virtuális** = nem tárolódik az adatbázisban, csak a relációt megadó lekérdezés.
  - **Materializált** = kiszámítódik, majd tárolásra kerül.

# Nézettáblák létrehozása és törlése

- Létrehozása:

```
CREATE [OR REPLACE] [FORCE | NOFORCE]  
[MATERIALIZED] VIEW <név>  
AS <lekérdezés>
```

```
[WITH CHECK OPTION [CONSTRAINT constraint]]  
[WITH READ ONLY [CONSTRAINT constraint]] ;
```

- Alapesetben virtuális nézettábla jön létre.
- Nézettábla megszüntetése:

```
DROP VIEW <név>;
```

# Példa: nézettábla létrehozása

- **Mit\_ihat(név, sör)** nézettáblában a sörivők mellett azon söröket tároljuk, amelyeket legalább egy olyan sörözőben felszolgálnak, amelyet látogat:

```
CREATE VIEW Mit_ihat AS
  SELECT név, sör
  FROM Látogat, Felszolgál
  WHERE L.söröző = F.söröző;
```

# Példa: nézettáblákhoz való hozzáférés

- A nézettáblák ugyanúgy kérdezhetők le, mint az alaptáblák.
- A nézettáblákon keresztül az alaptáblák néhány esetben módosíthatóak is, ha a rendszer a módosításokat át tudja vezetni (lásd módosítások, SQL DML)
- Példa lekérdezés:

```
SELECT sör FROM Mit_ihat  
WHERE név = 'Sally' ;
```

# Módosítható nézettáblák

- Az SQL szabvány formálisan leírja, hogy mikor lehet egy nézettáblát módosítani és mikor nem, ezek a szabályok meglehetősen bonyolultak.
- Ha a nézettábla definíciójában a SELECT után nem szerepel DISTINCT, további kikötések:
- A WHERE záradékban R nem szerepelhez egy alkérdésben sem
- A FROM záradékban csak R szerepelhet, az is csak egyszer és más reláció nem
- A SELECT záradék listája olyan attribútumokat kell, hogy tartalmazzon, hogy az alaptáblát fel lehessen tölteni (vagyis kötelező a kulcsként vagy not null-nak deklarált oszlopok megadása)



# Tankönyv példája: nézettáblára

Tk.8.1. Példa: Egy olyan nézettáblát szeretnénk, mely a Film(cím, év, hossz, színes, stúdióNév, producerAzon) reláció egy részét jelképezi, pontosabban a Paramount stúdió által gyártott filmek címét és gyártási évét

```
CREATE VIEW ParamountFilm AS  
SELECT cím, év  
FROM Film  
WHERE stúdióNév = 'Paramount';
```

# Nézeteken instead-of-triggererek

Tk. 8.8. Példa: Az előző nézettábla módosítható, és hogy az alaptáblába való beszúrásakor a stúdióNév attribútum helyes értéke , 'Paramount' legyen, ezt biztosítja ez az **INSTEAD OF (helyette) típusú trigger:**

```
CREATE TRIGGER ParamountBeszúrás  
  INSTEAD OF INSERT ON ParamountFilm  
  REFERENCING NEW ROW AS ÚjSor  
  FOR EACH ROW  
  INSERT INTO Film(cím, év, stúdióNév)  
  VALUES(Újsor.cím, ÚjSor.év, 'Paramount');
```

# Materializált (tárolt) nézetablák

- Adattárházaknál használják (MSc kurzusok)
- **Probléma:** minden alkalommal, amikor az alaptáblák valamelyike változik, a materializált nézetábla frissítése is szükségessé válhat.
  - Ez viszont néha túl költséges.
- **Megoldás:** Periodikus frissítése a materializált nézetábláknak, amelyek egyébként „nem aktuálisak”.

# Kérdés/Válasz

- Köszönöm a figyelmet! Kérdés/Válasz?
- **Házi feladat: Gyakorlás** az 8. és 9. előadások két hetére:
- Ezen a héten: **DML-utasítások, tranzakciók** (lásd 5EA)
  - Változóhasználat (Példatár 4.fej., + PL/SQL: 8.fej. is)
  - DML-utasítások: insert, update, delete (Példatár 5.fej.)
  - Adatbázis-tranzakciók: commit, rollback, savepoint
- Következő héten: **DDL-utasítások** (lásd a köv. 9EA)
  - DDL-utasítások: adattáblák létrehozása, módosítása, integritási megszorítások (Példatár 5.fejezet folyt.) és
  - Nézetábla létrehozása és törlése, táblák tartalmának módosítása nézetáblákon keresztül (Példatár 6.fej.)

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