7.előadás: Adatbázisok-I.

dr. Hajas Csilla (ELTE IK)

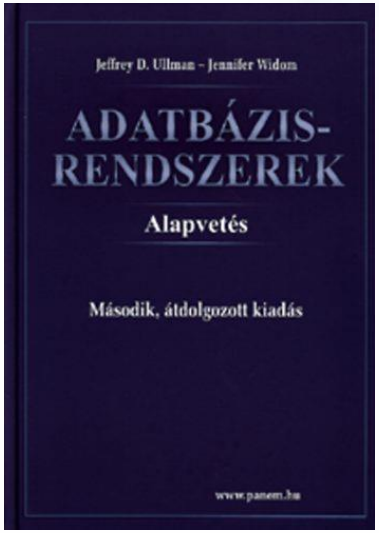
http://sila.hajas.elte.hu/

SQL a gyakorlatban: SQL DML, DDL

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

7.fej. Táblák és megszorítások létrehozása
CREATE TABLE, CONSTRAINTS
Önálló megszorítások, Triggerek

8.fej. Nézettáblák, tárolt nézettáblák



7.előadás: SQL DML, DDL [SQL3.pdf]

Előzmények, az első 6 előadás témakörei:

-- Egy táblára vonatkozó ismeretek

- [01] TERV1.pdf (Relációs modell és az E/K modell bev)
  SQL1.pdf (SQL bev, create table/1.tipusok, kulcsok)
- [02] REL1.pdf (Egytáblás lekérdezések, vetítés, szűrés)
- [03] REL2.pdf (Egytáblás lekérdezések, csoportosítás)

-- Több táblára vonatkozó ismeretek

- [04] TERV2.pdf (E/K haladó, megszorítások, alosztályok)
  SQL2.pdf (create table/2., constraints, hivatk.épség)
- [05] REL3.pdf (Több táblás lekérd. relációs algebrában)
- [06] REL4.pdf (Több táblás lekérdezések az SQL-ben)

-- SQL a gyakorlatban SQL DML, DDL; SQL/PSM

- [07] SQL3.pdf Ma: Tk.6.5.(DML), Tk.7.fej., 8.fej. (DDL)
- [08] SQL4.pdf Következő héten: Tk.9.fej. SQL/PSM
  és alkalmazása a gyakorlatban: Oracle PL/SQL

SQL fő komponensei

- Az SQL elsődlegesen lekérdező nyelv (Query Language)
  SELECT utasítás (az adatbázisból információhoz jussunk)
- Adatkezelő nyelv, DML (Data Manipulation Language)
  INSERT, UPDATE, DELETE, SELECT
- Sémaleíró nyelv, DDL (Data Definition Language)
  CREATE, ALTER, DROP
- Adatvezérlő nyelv, DCL (Data Control Language)
  GRANT, REVOKE
- Tranzakció-kezelés
  COMMIT, ROLLBACK, SAVEPOINT
- Procedurális kiterjesztések
  SQL/PSM és a gyakorlatban Oracle PL/SQL

(Tk.6.5) SQL DML utasítások
Adatbázis tartalmának módosítása

- 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> [(<attr.lista>)]
  VALUES (<konkrét értékek listája>);

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

  INSERT INTO <reláció> [(<attr.lista>)]
  (<alkérdés>);

Beszúrás, attribútumok megadása

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

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

- ahol a reláció séma Szeret(név, sör) attribútumai sorrendjében (lásd create table) adjuk meg az értékeket.
- Megadhatjuk a reláció neve után az attribútumokat, amivel olvashatóbbá válik az utasítás, illetve ez akkor jó, ha
  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');
```

Több sor beszúrása ---1

- Az előbb ismertett legegyszerűbb INSERT utasítás csak egy sort szúr be a relációba. Egy alkérdés segítségével meghatározhatunk több beszúrandó sort is:

  INSERT INTO <reláció> [(<attr.lista>)]

  (<alkérdés>);

- Az SQL szabvány előírja, hogy a lekérdezést teljesen ki kell értékelni, mielőtt a sorokat beszúrnánk a táblába!

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

Több sor beszúrása ---2

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

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

(SELECT) a másik sörivó

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

Több sor beszúrása ---3

- Példa: átírjuk az előző példában szereplő alkérdést, hogy abban is szerepelhet alkérdés. (Ekkor az alkérdést teljesen ki kell értékelni, mielőtt a sorokat beszúrnánk!)

```
INSERT INTO LehetBarát
(SELECT név
 FROM Látogat
 WHERE név <> 'Zsu'
      AND söröző IN
          (SELECT söröző
           FROM Látogat
           WHERE név = 'Zsu' ));
```

(SELECT) a másik sörivó

aki nem Zsu és jár olyan söröző be, ahová Zsu is, előző példa itt alkérdéssel

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; --- ~SELECT FROM

vagy DELETE Szeret; --- ~ UPDATE relációnév

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

- Módosítsuk a Felszolgál(söröző, sör, ár) táblát úgy, hogy legfeljebb 4 dollárba kerülhessenek a sörök:

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

- UPDATE esetén is használhatóak alkérdések a WHERE záradékban, ahogyan a SELECT-nél tanultuk, sőt a SET-ben szereplő érték helyén használhatunk skalár értéket adó alkérdés is:

Példa: Több sor módosítása alkérdések használatával

- Módosítsuk a Felszolgál(söröző, sör, ár) táblát úgy, hogy minden „Soproni” gyártójú sör árát növeljük meg a legolcsóbb sör 10%-ával:

```
UPDATE Felszolgál
SET ár = ár + (SELECT 0.1 * MIN(ár)
                FROM Felszolgál)
WHERE sör IN (SELECT név FROM Sörök
                WHERE gyártó = ’Soproni’);
```

(Tk.6.6.) Tranzakciók az SQL-ben Miért van szükség tranzakciókra?

- Az adatbázis rendszereket általában több felhasználó és folyamat használja egyidőben.
  - Lekérdezések és módosítások egyaránt történhetnek.
- Példa: Egy időben ketten töltenek fel 100 dollárt ugyanarra a számlára ATM-en keresztül.
  - Az adatbázis rendszernek biztosítania kell, hogy egyik művelet se vesszen el.

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.
- Adatbázisok-1 kurzuson csak a tranzakciók ACID tulajdonságát, COMMIT és ROLLBACK utasításokat nézzük át, a tranzakció-kezelésről részletesen az Adatbázis-2 kurzuson lesz szó.

ACID tranzakciók

ACID tulajdonságok:

- 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
- Opcionálisan: gyengébb feltételek is megadhatóak.

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.

(Tk.7.fejezet) SQL DDL Adatbázis relációsémák definiálása megszorítások és triggerek

- Az SQL tartalmaz adatleíró részt (DDL), az adatbázis objektumainak a leírására és megváltoztatására. Objektumok leíró parancsa a CREATE utasítás.
- A relációt az SQL-ben táblának (TABLE) nevezik, az SQL alapvetően háromféle táblát kezel:
  - Alaptáblák (permanens) CREATE TABLE
  - Nézettáblák CREATE VIEW
  - Átmeneti munkatáblák (WITH utasítás)
- Alaptáblák megadása: CREATE TABLE

Tábla/reláció sémák SQL-ben

- A legegyszerűbb formája:

```
CREATE TABLE relációnév (
        Attribútum deklarációk listája,
        Kiegészítő lehetőségek
);
```

- Az attribútum deklaráció legalapvetőbb elemei:

  Attribútumnév típus [kiegészítő lehetőségek]

-- itt: a **típus** olyan, amit az SQL konkrét megvalósítása támogat (gyakorlaton Oracle környezetben nézzük meg), Típusok, pl: INTEGER, REAL, CHAR, VARCHAR, DATE

-- **Kiegészítő lehetőségek** például [DEFAULT], [UNIQUE], [PRIMARY KEY], [FOREIGN KEY, REFERENCES], stb.

Megszorítások (áttekintés)

(1) Kulcsok és idegen kulcsok

- (1a) Kulcsok (egyszerű, összetett)
- (1b) A hivatkozási épség fenntartása
- Megszorítások ellenőrzésének késleltetése

(2) Értékekre vonatkozó feltételek

- (2a) NOT NULL feltételek
- (2b) Attribútumra vonatkozó CHECK feltételek

(3) Sorokra vonatkozó megszorítások

- Sorra vonatkozó CHECK feltételek

(4) Megszorítások módosítása (constraints)

(1.a) Kulcs megadása

- PRIMARY KEY vagy UNIQUE
- Kulcs = minimális szuperkulcs (azonosító attribútumok) ahol szuperkulcs, egy vagy több attribútum K halmaza, amelyre a reláció megengedett előfordulásokban nincs két különböző sor, amelyek megegyezne K minden attribútumán. Vagyis, ha két sor a K attribútumain megegyezik, akkor minden attribútumon is megegyezik.
- Több lehetséges kulcs is lehet, ilyenkor választunk közülük egy „elsődleges kulcsot” PRIMARY KEY, és a többi pedig UNIQUE lehet (erre szükség lehet például a hivatkozási épség megszorításnál kulcsra hivatkozunk)
- Kulcsok megadásának két változata van:
  - Egyszerű kulcs (egy attribútum) vagy
  - Összetett kulcs (attribútumok listája)

Egyszerű kulcs megadása

- Ha a kulcs egyetlen attribútum, akkor ez az attribútum deklarációban megadható

  <attribútumnév> <típus> PRIMARY KEY

  vagy <attribútumnév> <típus> UNIQUE

- Példa:

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

Összetett kulcs megadása

- Ha a kulcs több attribútumból áll, akkor a CREATE TABLE utasításban az attribútum deklaráció után a kiegészítő részben meg lehet adni további tábla elemeket: PRIMARY KEY (attrnév$_1$, … attrnév$_k$)
- Példa:

```
CREATE TABLE Felszolgál (
        söröző      CHAR(20),
        sör         VARCHAR2(20),
        ár          NUMBER(10,2),
        PRIMARY KEY (söröző, sör)
);
```

PRIMARY KEY vs. UNIQUE

- Csak egyetlen PRIMARY KEY lehet a relációban, viszont UNIQUE több is lehet.
- PRIMARY KEY egyik attribútuma sem lehet NULL egyik sorban sem. Viszont UNIQUE-nak deklarált attribútum lehet NULL, vagyis a táblának lehet olyan sora, ahol a UNIQUE attribútum értéke NULL vagyis hiányzó érték.
- az SQL lekérdezésnél adjuk meg hogyan kell ezzel a speciális értékkel gazdálkodni, hogyan lehet NULL-t kifejezésekben és hogyan lehet feltételekben használni
- Következő héten visszatérünk a megszorítások és a hivatkozási épség megadására.

(1.b) Idegen kulcsok megadása

- Az első előadáson a táblák létrehozásához veszünk 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));
```

Hivatkozási épség, idegen kulcs megszorítások megőrzése

- Példa: *R* = Felszolgál, *S* = 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* = Felszolgál, *S* = Sörök.
- Nem engedjük, hogy Felszolgál táblába a Sörök táblában nem szereplő sört szúrjanak 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 Felszolgál 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 írnunk:

  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 deklarálható DEFERRABLE (késleltethető) vagy NOT DEFERRABLE-ként (vagyis minden adatbázis módosításkor a megszorítás közvetlenül utána ellenőrzésre kerül). DEFERRABLE-ként deklaráljuk, 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)

(1) Kulcsok és idegen kulcsok

- (1a) Kulcsok (egyszerű, összetett)
- (1b) A hivatkozási épség fenntartása
- Megszorítások ellenőrzésének késleltetése

(2) Értékekre vonatkozó feltételek

- (2a) NOT NULL feltételek
- (2b) Attribútumra vonatkozó CHECK feltételek

(3) Sorokra vonatkozó megszorítások

- Sorra vonatkozó CHECK feltételek

(4) Megszorítások módosítása (constraints)

(2.) Értékekre vonatkozó feltételek

- Egy adott oszlop értékeire vonatkozóan adhatunk meg megszorításokat.
- (2a) A CREATE TABLE utasításban az attribútum deklarációban NOT NULL kulcsszóval
- (2b) az attribútum deklarációban CHECK(<feltétel>) , ahol a feltétel olyan, mint egy WHERE feltétel

Példa: értékekre vonatkozó feltétel

```
CREATE TABLE Felszolgál (
  söröző CHAR(20) NOT NULL,
  sör    CHAR(20) REFERENCES Sörök(név)
  ár     REAL CHECK ( ár <= 5.00 ) );
--- vagy ugyanez CHECK feltétellel:
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 törlünk, ezt a feltételt nem ellenőrzi a rendszer.

(3.) 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 söröző̋ben 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)
);
```

(4) Megszorítások módosítása

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

- 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)

(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 (create assertion)

(6) Triggerek (create trigger)

(5.) Önálló megszorítások: Assertions

- SQL aktív elemek közül a leghatékonyabbak nincs hozzárendelve sem sorokhoz, sem azok komponenseihez, hanem táblákhoz 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

- A Felszolgál(söröző, sör, ár) táblában nem lehet olyan söröző, ahol a sörök átlagára 5 dollárnál több

```
CREATE ASSERTION CsakOlcsó CHECK
 (
NOT EXISTS (
    SELECT söröző
    FROM Felszolgál
    GROUP BY söröző
    HAVING 5.00 < AVG(ár)
));
```

(SELECT .. olyan sörözők, ahol a sörök átlagosan drágábbak 5 dollárnál)

Példa: önálló megszorítások

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

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

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.

(7.) Megszorítások v.s. triggerek

- Aktív elemek – olyan kifejezés vagy utasítás, amit egyszer eltároltunk az adatbázisban és 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 triggerek?

- 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 triggerek 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, SQL/PSM tárolt eljárás

Példa triggerre

- Ahelyett, hogy visszautasítanánk a Felszolgál(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 triggereket a rendszer csak akkor ellenőrzi, ha bizonyos *események* bekövetkeznek.
  A megengedett események általában egy adott relációra vonatkozó beszúrás, törlés, módosítás, vagy a tranzakció befejeződése.

Triggerek --- 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)
BEGIN
    DELETE FROM GyártásIrányító
    WHERE (név, cím, azonosító) IN ÚjAdat;
    INSERT INTO gyártásIrányító
        (SELECT * FROM RégiAdat);
END;
```

Tankönyv példája (7.7. ábra)

-- A beszúrt sorok NULL értékeinek helyettesítésére, itt csak egyszerűen 1915-tel helyettesíti a trigger a NULL értéket, de ez akár egy bonyolult módon kiszámított érték is lehet:

(A BEFORE triggerek egy fontos alkalmazása, amikor egy beszúrandó sort a beszúrás előtt megfelelő formára hoznak)

```
CREATE TRIGGER ÉvJavítóTrigger
BEFORE INSERT ON Filmek
REFERENCING
    NEW ROW AS ÚjSor,
    NEW TABLE AS ÚjAdat
FOR EACH ROW
WHEN ÚjSor.év IS NULL
UPDATE ÚjAdat SET év=1915;
```

(Tk. 8.fejezet) 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	AHUNO_D	590.423.4567	03-JAN-90	IT_PROG	90
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG	42
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN	58
141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95	ST_CLERK	35
142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-97	ST_CLERK	31
143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	26
144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-96	ST_CLERK	25
149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-JAN-00	SA_MAN	105
174	Ellen	Abel	EABEL	011.44.1644.429267	11-MAY-96	SA_REP	110
176	Jonathon	Taylor	JTAYLOR	011.44.1644.429265	24-MAR-98	SA_REP	86
178	Kimberely	Grant	KGRANT	011.44.1644.429263	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	WGIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	83

EMPLOYEE_ID	LAST_NAME	SALARY
149	Zlotkey	10500
174	Abel	11000
176	Taylor	8600

20 rows selected.

A nézettáblák előnyei

Korlátozható az adatok elérése

Bonyolult lekérdezések egyszerűbben kifejezhetők

Az adatfüggetlenség biztosítása

Ugyanazon adatokat különböző nézőpontokból szemlélhetjük

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

- 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 Filmek
    WHERE stúdióNév = 'Paramount';
```

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 cím FROM ParamountFilm
WHERE év <= 1990;
```

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)

Nézeteken instead-of-triggerek

Példa: Az előző nézettábla módosításánál, hogy az alaptáblába való beszúráskor a stúdióNév attribútum helyes értéke , 'Paramount' legyen, ezt biztosítja az INSTEAD OF (helyette) típusú trigger:

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

(Tk.8.5.) Tárolt nézettáblák

- **CREATE [OR REPLACE]**
  MATERIALIZED VIEW <név>
  AS <lekérdezés>
- Adattárházaknál használják (MSc kurzusok)
- Probléma: minden alkalommal, amikor az alaptáblák valamelyike változik, a materializált nézettábla frissítése is szükségessé válhat.
  - Ez viszont néha túl költséges.
- Megoldás: Periodikus frissítése a materializált nézettábláknak, amelyek egyébként „nem aktuálisak”.

(Tk.10.1) Jogosultság-kezelés

- Egy UNIX-szerű fájlrendszerhez hasonlítva az analógiák: Tipikusan írás, olvasás és végrehajtási jogosultságokról van szó.
- Az adatbázisok lényegesen bonyolultabbak a fájlrendszereknél, ezért az SQL szabványban definiált jogosultágok is összetettebbek.
  - Az SQL kilencféle jogosultságot definiál (SELECT, INSERT, DELETE, UPDATE, REFERENCES, USAGE, TRIGGER, EXECUTE, UNDER)
  - Bizonyos „résztvevőkhöz” sorolja a jogosultságokat, például rendszergazda, korlátozott jogosultságokkal rendelkező felhasználó. Spec. PUBLIC (mindenki)

SQL DCL: GRANT utasítás

- Jogosultságok megadásának szintaktikája:

  GRANT <jogosultságok listája>

  ON <reláció vagy másféle objektum>

  TO <jogosultsági azonosítók listája>;

- Ehhez hozzáadható:

  WITH GRANT OPTION

Példa: GRANT

```
GRANT SELECT, UPDATE(ár)
    ON Felszolgál
    TO Sally;
```

- Ez után Sally kérdéseket adhat meg a Felszolgál táblára vonatkozóan és módosíthatja az ár attribútumot.

Jogosultságok

- A relációkra vonatkozó jogosultságok:

  SELECT = a reláció lekérdezésének joga.

  INSERT = sorok beszúrásának joga.

  (egyetlen attribútumra is vonatkozhat)

  DELETE = sorok törlésének joga.

  UPDATE = sorok módosításának a joga.

  (szintén egy attribútumra is vonatkozhat)

Példa: jogosultságok

- Az alábbi utasítás esetében:

```
INSERT INTO felh.Sörök(név)
SELECT sör FROM felh.Felszolgál f
   WHERE NOT EXISTS
      (SELECT * FROM felh.Sörök
       WHERE név = f.sör);
```

azok a sörök, amelyek még nincsenek benne a sörök táblában. A beszúrás után a gyártó értéke NULL.

- Ehhez az INSERT utasítás végrehajtásához szükséges: SELECT jogosultság a felh (user) felszolgál és sörök tábláira és INSERT jog a Sörök tábla név attribútumára vonatkozóan.

Adatbázis objektumok

- Jogosultságokat nézetekre és materializált nézetekre vonatkozóan is megadhatunk.
- Egy másik fajta jogosultság lehet pl. adatbázis objektumok létrehozásának a joga: pl. táblák, nézetek, triggerek.
- A nézettáblák segítségével tovább finomíthatjuk az adatokhoz való hozzáférést.

Példa: nézettáblák és jogosultságok

- Tegyük fel, hogy nem szeretnénk SELECT jogosultságot adni az Dolgozók(név, cím, fizetés) táblában.
- Viszont a BiztDolg nézettáblán már igen:

```
CREATE VIEW BiztDolg AS
    SELECT név, cím FROM Dolgozók;
```

- A BiztDolg nézettáblára vonatkozó kérdésekhez nem kell SELECT jog a Dolgozók táblán, csak a BiztDog nézettáblán.

Jogosultságok megadása

- A magunk készítette objektumok esetében az összes jogosultsággal rendelkezünk.
- A felhasználókat egy jogosultsági azonosító (authorization ID) alapján azonosítjuk, általában ez a bejelentkezési név, ennek felhasználásával másoknak is megadhatunk jogosultságokat.
- vagy a PUBLIC jogosultsági azonosítót is használhatjuk, a PUBLIC jogosultság minden felhasználó számára biztosítja az adott jogot.
- A WITH GRANT OPTION utasításrész lehetővé teszi, hogy aki megkapta a jogosultságot, tovább is adhassa azt.

Példa: Grant Option

```
GRANT UPDATE ON Felszolgál TO Sally
  WITH GRANT OPTION;
```

- Ez után Sally módosíthatja a Felszolgál táblát és tovább is adhatja ezt a jogosultságot.
  - Az UPDATE jogosultságot korlátozottan is továbbadhatja: `UPDATE(ár)ON Felszolgál`.

Jogosultságok visszavonása

REVOKE <jogosultságok listája>
ON <reláció vagy más objektum>
FROM <jogosultsági azonosítók listája>;

- Az általunk kiadott jogosultságok ez által visszavonódnak.
  - De ha máshonnan is megkapták ugyanazt a jogosultságot, akkor az még megmarad.

Kérdés/Válasz

- Köszönöm a figyelmet! Kérdés/Válasz?

Gyakorlás: Oracle példatár

- SELECT utasítás, lekérdezések SQL-ben (Példatár 3.fej.)
- DML-utasítások, tranzakciók
  - DML-utasítások: insert, update, delete (Példatár 5.fej.)
  - Adatbázis-tranzakciók: commit, rollback, savepoint
- DDL-utasítások, create table, create view
  - DDL-utasítások: adattáblák létrehozása, módosítása, integritási megszorítások (Példatár 5.fejezet folyt.) és
  - Nézettábla létrehozása és törlése, táblák tartalmának módosítása nézettáblákon keresztül (Példatár 6.fej.)