

4.előadás/2.része: Adatbázisok-I.

dr. Hajas Csilla (ELTE IK) (2020)

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

SQL gyakorlatban: SQL DDL és DML

CREATE TABLE ... CONSTRAINTS ...

2.3. Relációsémák definiálása SQL-ben

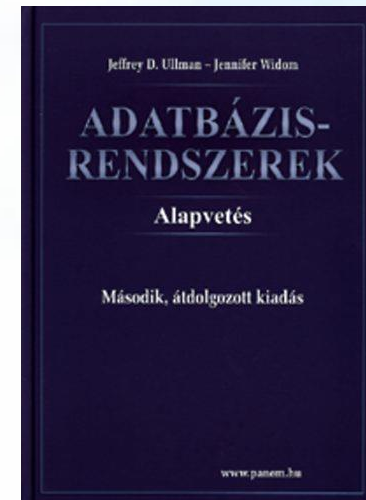
2.3.6. Kulcsok megadása

7.fej. Táblák+megszorítások létrehozása

7.1. Kulcsok és idegen kulcsok

7.2. További megszorítások

(Folyt.később: önálló megszorítások és triggerek)



[ismétlés] 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

[ismétlés] SQL DDL

Adatbázis relációsémák definiálása

[új rész] megszorításokkal (constraints)

- Az SQL tartalmaz **adateleíró 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
 - Nézetáblák CREATE VIEW
 - Átmeneti munkatáblák (WITH utasítás)
- **Alaptáblák** megadása: **CREATE TABLE**

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)

Tankönyv példa: Filmek séma

Filmek(

cím:string,
év:integer,
hossz:integer,
műfaj:string,
stúdióNév:string,
producerAzon:integer)

FilmSzínész(

név:string,
cím:string,
nem:char,
születésiDátum:date)

Stúdió(

név:string,
cím:string,
elnökAzon:integer)

Mit jelentenek az aláhúzások?

Tankönyv példája, hibás fordítás:
title=(film)cím és address=(lak)cím

Tervezéssel később foglalkozunk, ez a példa hibás, az elnevezések, de így jó lesz, hogy a lekérdezéseknél megnézzük hogyan kezeljük.

SzerepelBenne(

filmCím:string,
filmÉv:integer,
színészNév:string)

GyártásIrányító(

név:string,
cím:string,
azonosító:integer,
nettóBevétel:integer)

Példa megszorításokra: Kulcs

- Előző példában: attribútumok aláhúzása mit jelent?
- Kulcs = minimális szuperkulcs (azonosító attribútumok)
- Filmek: elvárjuk, hogy ne legyen a megengedett előfordulásokban két különböző sor, amelyek megegyeznek cím, év attribútumokon.
- Egyszerű kulcs egy attribútumból áll, de egy kulcs nem feltétlenül áll egy attribútumból, ez az összetett kulcs. Például a **Filmek** táblában a cím és év együtt alkotják a kulcsot, nem elég a cím, ugyanis van például (King Kong, 1933), (King Kong, 1976) és (King Kong, 2005).
- A kulcsot aláhúzás jelöli: **Filmek** (cím, év, hossz, ...)

(1.a) Kulcs megadása

- **PRIMARY KEY** vagy **UNIQUE**
- Nincs a relációnak két olyan sora, amely a lista minden attribútumán megegyezne.
- Kulcs esetén nincs értelme a DEFAULT értéknek.
- 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 (attrnév₁, ... attrnév_k)**
- Példa:

```
CREATE TABLE Felszolgal (
    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: $R = \text{Felszolgál}$, $S = \text{Sörök}$.
- 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 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: továbbgyűrés

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

Példa: Set NULL

- A Bud sort töröljük a **Sörök** táblából:
 - a **Felhasználó** 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 Felszolgal (
    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 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 Felszolgal (
  söröző CHAR(20) NOT NULL,
  sör     CHAR(20) REFERENCES Sörök(név)
  ár     REAL CHECK ( ár <= 5.00 ) );
```

--- vagy ugyanez CHECK feltétellel:

```
CREATE TABLE Felszolgal (
  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.

(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 sörözőben lehetnek drágábbak a sörök 5 dollárnál:

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

Megszorítások elnevezése

- 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.\%');

Kérdés / Válasz

- **Köszönöm a figyelmet! Kérdés/Válasz?**
- **Jön:** Több táblára vonatkozó lekérdezésekkel folytatjuk, az SQL SELECT további részletei, halmazműveletek, összekapcsolások és alkérdések használata a lekérdezésekben.