

Adatbázisok 1.

Megszorítások

Idegen kulcsok

Lokális és globális megszorítások

Triggerek

Megszorítások és triggererek

- A *megszorítás* adatelemek közötti kapcsolat, amelyet az AB rendszernek fent kell tartania.
 - Példa: kulcs megszorítások.

Megszorítások és triggererek

- A *megszorítás* adatelemek közötti kapcsolat, amelyet az AB rendszernek fent kell tartania.
 - Példa: kulcs megszorítások.
- *Triggererek* olyankor hajtódnak végre, amikor valamilyen megadott esemény történik, mint pl. sorok beszúrása egy táblába.

Megszorítások típusai

- Kulcsok. $\sigma_{B1.név=B2.név \wedge B1.város=B2.város \wedge B1.tulaj \neq B2.tulaj}(B_1 \times B_2) = \emptyset$

Megszorítások típusai

- Kulcsok.
- Idegen kulcsok, vagy hivatkozási épség megszorítás.

$$\Pi_{\text{sör}}(\text{Felszolgál}) \subseteq \Pi_{\text{név}}(\text{Sör})$$

Megszorítások típusai

- Kulcsok.
- Idegen kulcsok, vagy hivatkozási épség megszorítás.
- Attribútum alapú (érték-alapú) megszorítás.
 - Egy adott attribútum lehetséges értékeiről mond valamit.

$$\sigma_{(\text{város} \neq \text{'Budapest'}) \wedge (\text{város} \neq \text{'Madrid'})} (B) = \emptyset$$

Megszorítások típusai

- Kulcsok.
- Idegen kulcsok, vagy hivatkozási épség megszorítás.
- Érték-alapú megszorítás.
 - Egy adott attribútum lehetséges értékeiről mond valamit.
- Sor-alapú megszorítás.
 - Mezők közötti kapcsolatok leírása.

Megszorítások típusai

- Kulcsok.
- Idegen kulcsok, vagy hivatkozási épség megszorítás.
- Érték-alapú megszorítás.
 - Egy adott attribútum lehetséges értékeiről mond valamit.
- Sor-alapú megszorítás.
 - Mezők közötti kapcsolatok leírása.
- Globális megszorítás: bármilyen SQL kifejezés.

Emlékeztető: egy attribútumos kulcsok

- PRIMARY KEY vagy UNIQUE.

- Példa:

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

Emlékeztető: kulcsok több attribútummal

```
CREATE TABLE Felszolgal (
    kocsmas CHAR(20),
    sor      VARCHAR(20),
    ar       REAL,
    PRIMARY KEY (kocsmas, sor)
);
```

Idegen kulcsok

- Egy reláció attribútumainak értékei egy másik reláció értékei között is meg kell, hogy jelenjenek együttesen.
- **Példa:** a **Felhasználó(kocsi, sör, ár)** táblánál azt várjuk, hogy az itteni sörök szerepelnek a **Sörök** tábla **név** oszlopában is.

Idegen kulcsok megadása

- A REFERENCES kulcsszót kell használni:
 1. egy attribútum után (egy-attribútumos kulcs)
 2. A séma elemeként:
FOREIGN KEY (<attribútumok listája>
REFERENCES <reláció> (<attribútumok>)
- A hivatkozott attribútum(ok)nak kulcsnak kell lennie / lenniük (PRIMARY KEY vagy UNIQUE).

Példa: egy attribútum

```
CREATE TABLE Sörök (  
    név      CHAR(20) PRIMARY KEY,  
    gyártó   CHAR(20) );  
  
CREATE TABLE Felszolgál (  
    kocsmá   CHAR(20) ,  
    sör       CHAR(20) REFERENCES Sörök(név) ,  
    ár        REAL );
```

Példa: a séma elemeként

```
CREATE TABLE Sörök (  
    név          CHAR(20) PRIMARY KEY,  
    gyártó       CHAR(20) );  
  
CREATE TABLE Felszolgal (  
    kocsmá       CHAR(20),  
    sör          CHAR(20),  
    ár           REAL,  
    FOREIGN KEY (sör) REFERENCES  
        Sörök(név) );
```

Idegen kulcs megszorítások megőrzése

- Egy idegen kulcs megszorítás R relációról S relációra kétféleképpen sérülhet:
 1. Egy R -be történő beszúrásnál S -ben nem szereplő értéket adunk meg.
 2. Egy S -beli törlés „lógó” sorokat eredményez R -ben.

Hogyan védekezzünk? --- (1)

- **Példa:** R = Felszolgál, S = Sörök.
- Nem engedjük, hogy **Felszolgál** táblába a **Sörök** táblában nem szereplő sört szűrjanak be.
- A **Sörök** táblából való törlés, ami a **Felszolgál** tábla sorait is érintheti (mert sérül az idegen kulcs megszorítás) 3-féle módon kezelhető.

Hogyan védekezzünk? --- (2)

1. Default

Hogyan védekezzünk? --- (2)

1. *Default*
2. *Továbbgyűrűzés*

Hogyan védekezzünk? --- (2)

1. *Default*
2. *Továbbgyűrűzés*
 - Sör törlése

Hogyan védekezzünk? --- (2)

1. *Default*

2. *Továbbgyűrűzés*

- Sör törlése
- Sör módosítása

Hogyan védekezzünk? --- (2)

1. *Default*
2. *Továbbgyűrűzés*
 - Sör törlése
 - Sör módosítása
3. *Set NULL*

Példa: továbbgyűrűzés

- Töröljük a Bud sort a **Sörök** táblából:
 - az összes sort töröljük a **Felhasználók** táblából, ahol sör oszlop értéke 'Bud'.

Példa: továbbgyűrés

- 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.

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 módosításra 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 (
    kocsmasma      CHAR(20) ,
    sör            CHAR(20) ,
    ár             REAL,
    FOREIGN KEY (sör)
        REFERENCES Sörök(név)
        ON DELETE SET NULL
        ON UPDATE CASCADE
);
```

Attribútum alapú (érték alapú) ellenőrzések

- Adott oszlop értékeire vonatkozóan
- A CHECK(<feltétel>) hozzáadása az attribútum deklarációjához
- Feltételben csak az adott attribútum neve, **más attribútumok (más relációk attribútumai is) csak alkérdésben**

Példa: attribútum alapú ellenőrzés

```
CREATE TABLE Felszolgal (
    kocsmas CHAR(20),
    sor CHAR(20) CHECK ( sor IN
        (SELECT név FROM Sörök) ),
    ár REAL CHECK ( ár <= 5.00 )
);
```

Mikor ellenőriz?

- Attribútum-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.

Sor-alapú megszorítások

- CHECK (<feltétel>) megszorítás a séma elemeként
- Feltételben tetsz. oszlop és reláció
 - De más relációk attribútumai csak alkérdésben jelenhetnek meg.
- Csak beszúrásnál és módosításnál ellenőrzi a rendszer.

Példa: sor-alapú megszorítások

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


Globális megszorítás

- Adatbázissémához tartoznak
- `CREATE ASSERTION <név>
CHECK (<feltétel>);`
- A feltétel tetszőleges táblára és oszlopra hivatkozhat az adatbázissémából.

Példa: globális megszorítás

```
CREATE ASSERTION CsakOlcsó CHECK (  
    NOT EXISTS (  

```

```
        SELECT kocsm  
        FROM Felszolgál  
        GROUP BY kocsm  
        HAVING 5.00 < AVG(ár)
```

```
    ));
```

← Kocsmák, ahol
a sörök
átlagosan
drágábbak 5
dollárnál.

Példa: globális megszorítás

- Az **Alkesz(név, cím, telefon)** és **Kocsma(név, cím, engedélySzám)**, táblákban nem lehet több kocsmá, mint alkesz.

```
CREATE ASSERTION TöbbAlkesz CHECK (  
    (SELECT COUNT(*) FROM Kocsma) <=  
    (SELECT COUNT(*) FROM Alkesz)  
);
```

Globális megszorítások ellenőrzése

- Alapvetően az adatbázis bármely módosítása előtti ellenőrzés
- Egy okos rendszer felismeri, hogy mely változtatások, mely megszorításokat érinthetnek.
 - Példa: a Sörök tábla változásai nincsenek hatással az iménti TöbbAlkesz megszorításra.

Miért hasznosak a triggererek?

- A globális megszorításokkal sok mindent le lehet írni, de az ellenőrzésük gondot jelenthet.
- Az attribútum- és sor-alapú megszorítások ellenőrzése egyszerűbb, de ezekkel nem tudunk mindent kifejezni.
- A triggererek esetén a felhasználó mondja meg, hogy egy megszorítás mikor kerüljön ellenőrzésre.

Esemény-Feltétel-Akció szabályok

- A triggereket esetenként *ECA szabályoknak* (*event-condition-action*) is nevezik.
- *Esemény*
- *Feltétel*
- *Akció*

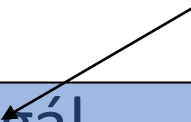
Példa: trigger

- Ahelyett, hogy visszautasítanánk a **Felhasználó(kocsi, 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  
BEFORE 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);
```

Az esemény



Példa: trigger definíció

```
CREATE TRIGGER SörTrig  
BEFORE 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);
```

Az esemény

A feltétel

Példa: trigger definíció

```
CREATE TRIGGER SörTrig
```

```
BEFORE INSERT ON Felszolgál
```

Az esemény

```
REFERENCING NEW ROW AS ÚjSor
```

```
FOR EACH ROW
```

```
WHEN (ÚjSor.sör NOT IN  
      (SELECT név FROM Sörök))
```

A feltétel

```
INSERT INTO Sörök(név)  
VALUES(ÚjSor.sör);
```

Az akció