

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. $\sigma_{T1.név=T2.név \wedge T1.város=T2.város \wedge T1.tulaj \neq T2.tulaj}(T_1 \times T_2) = \emptyset$

Megszorítások típusai

- Kulcsok.
- Idegen kulcsok (*foreign keys*), vagy hivatkozási épség megszorítás.

$$\Pi_{\text{tea}}(\text{Felszolgál}) \subseteq \Pi_{\text{név}}(\text{Teák})$$

Megszorítások típusai

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

$$\sigma_{(\text{város} \neq \text{'Budapest'}) \wedge (\text{város} \neq \text{'Madrid'})}(T) = \emptyset$$

Megszorítások típusai

- Kulcsok.
- Idegen kulcsok, vagy hivatkozási épség megszorítás.
- Attribútum alapú (érték-alapú) megszorítás.
 - Egy adott attribútum lehetséges értékeiről mond valamit.
- Sor-alapú megszorítás (*tuple-based constraint*).
 - Mezők közötti kapcsolatok leírása.

Megszorítások típusai

- Kulcsok.
- Idegen kulcsok, vagy hivatkozási épség megszorítás.
- Attribútum alapú (érték-alapú) megszorítás.
 - Egy adott attribútum lehetséges értékeiről mond valamit.
- Sor-alapú megszorítás.
 - Mezők közötti kapcsolatok leírása.
- Globális megszorítás (*assertion (!)*): bármilyen SQL kifejezés.

Emlékeztető: egy attribútumos kulcsok

- PRIMARY KEY vagy UNIQUE.

- Példa:

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

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felhasználó(teázó, tea, ár)

Látogat(vendég, teázó)

Emlékeztető: kulcsok több attribútummal

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felhasználó(teázó, tea, ár)

Látogat(vendég, teázó)

```
CREATE TABLE Felszolgal (
    teázó      CHAR(20),
    tea        VARCHAR(20),
    ár         REAL,
    PRIMARY KEY (teázó, tea)
);
```

Idegen kulcsok

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

- Egy reláció attribútumainak értékei egy másik reláció értékei között is meg kell, hogy jelenjenek együttesen.
- **Példa:** a **Felszolgál(teázó, tea, ár)** táblánál azt várnánk, hogy az itteni teák szerepelnek a **Teák** 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

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felhasznál(teázó, tea, ár)

Látogat(vendég, teázó)

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

CREATE TABLE Felhasznál (
    teázó        CHAR(20),
    tea          CHAR(20) REFERENCES Teák(név),
    ár           REAL );
```

Példa: a séma elemeként

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felhasználó(teázó, tea, ár)

Látogat(vendég, teázó)

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

CREATE TABLE Felhasználó (
    teázó        CHAR(20),
    tea          CHAR(20),
    ár           REAL,
    FOREIGN KEY (tea) REFERENCES
        Teák(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)

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felhasználó(teázó, tea, ár)

Látogat(vendég, teázó)

- **Példa:** $R = \text{Felhasználó}$, $S = \text{Teák}$.
- Nem engedjük, hogy **Felhasználó** táblába a **Teák** táblában nem szereplő teát szűrjanak be.
- A **Teák** táblából való törlés, ami a **Felhasználó** 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. *Továbbgyűűzés*

Hogyan védekezzünk? --- (2)

1. *Default*
2. *Továbbgyűűzés (cascade)*
 - Tea törlése

Hogyan védekezzünk? --- (2)

1. *Default*
2. *Továbbgyűűzés (cascade)*
 - Tea törlése
 - Tea módosítása

Hogyan védekezzünk? --- (2)

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

Példa: továbbgyűrés

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

- Töröljük a Brisk sort a **Teák** táblából:
 - az összes sort töröljük a **Felszolgál** táblából, ahol tea oszlop értéke 'Brisk'.

Példa: továbbgyűrés

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

- Töröljük a Brisk sort a **Teák** táblából:
 - az összes sort töröljük a **Felszolgál** táblából, ahol tea oszlop értéke 'Brisk'.
- A 'Brisk' nevet 'Brisk Iced Tea'-re változtatjuk:
 - a **Felszolgál** tábla soraiban is végrehajtjuk ugyanezt a változtatást.

Példa: Set NULL

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felhasználó(teázó, tea, ár)

Látogat(vendég, teázó)

- A Brisk sort töröljük a **Teák** táblából:
 - a **Felhasználó** tábla **tea** = 'Brisk' soraiban a Brisket cseréljük NULL-ra.

Példa: Set NULL

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felhasználó(teázó, tea, ár)

Látogat(vendég, teázó)

- A Brisk sort töröljük a **Teák** táblából:
 - a **Felhasználó** tábla **tea** = 'Brisk' soraiban a Brisket cseréljük NULL-ra.
- 'Brisk'-ről 'Brisk Iced Tea'-re módosítunk:
 - ugyanazt kell tennünk, mint törléskor.

Egy stratégia kiválasztása (*choosing a policy*)

- 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 (
    teázó          CHAR(20) ,
    tea            CHAR(20) ,
    ár             REAL,
    FOREIGN KEY (tea)
        REFERENCES Teák(név)
        ON DELETE SET NULL
        ON UPDATE CASCADE
);
```

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgal(teázó, tea, ár)

Látogat(vendég, teázó)

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

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felhasznál(teázó, tea, ár)

Látogat(vendég, teázó)

```
CREATE TABLE Felszolgal (
    teázó    CHAR(20),
    tea      CHAR(20) CHECK ( tea IN
                          (SELECT név FROM Teák) ),
    ár       REAL CHECK ( ár <= 5.00 )
);
```

Mikor ellenőriz?

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felhasználó(teázó, tea, ár)

Látogat(vendég, teázó)

- 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 (tea IN (SELECT név FROM Teák) , ha a Teák 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

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felhasználó(teázó, tea, ár)

Látogat(vendég, teázó)

```
CREATE TABLE Felszolgal (
    teázó      CHAR(20),
    tea        CHAR(20),
    ár         REAL,
    CHECK (teázó = 'Joe teázója' 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

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

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

Teázók, ahol
a teák
átlagosan
drágábbak 5
dollárnál.

Példa: globális megszorítás

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

- Az **Vendégek(név, cím, telefon)** és **Teázók(név, cím, engedélySzám)**, táblákban nem lehet több teázó, mint vendég.

```
CREATE ASSERTION TöbbVendég CHECK (  
    (SELECT COUNT (*) FROM Teázók) <=  
    (SELECT COUNT (*) FROM Vendégek)  
);
```

Globális megszorítások ellenőrzése

- Alapvetően az adatbázis bármely módosítása előtti ellenőrzés
- Egy okos rendszer felismeri, hogy mely változtatások, mely megszorításokat érinthetnek.
 - Példa: a Teák tábla változásai nincsenek hatással az iménti TöbbVendég 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* (*event*)
- *Feltétel* (*condition*)
- *Akció* (*action*)

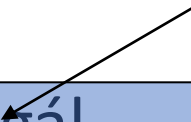
Példa: trigger

- Ahelyett, hogy visszautasítanánk a **Felhasználó(teázó, tea, ár)** táblába történő beszúrást az ismeretlen teák esetén, a **Teák(név, gyártó)** táblába is beszúrjuk a megfelelő sort a gyártónak NULL értéket adva.

Példa: trigger definíció

```
CREATE TRIGGER TeaTrig
BEFORE INSERT ON Felszolgál
REFERENCING NEW ROW AS ÚjSor
FOR EACH ROW
WHEN (ÚjSor.tea NOT IN
      (SELECT név FROM Teák))
INSERT INTO Teák(név)
VALUES(ÚjSor.tea);
```

Az esemény



Példa: trigger definíció

```
CREATE TRIGGER TeaTrig
```

```
BEFORE INSERT ON Felszolgál
```

Az esemény



```
REFERENCING NEW ROW AS ÚjSor
```

```
FOR EACH ROW
```

```
WHEN (ÚjSor.tea NOT IN  
      (SELECT név FROM Teák))
```

A feltétel



```
INSERT INTO Teák(név)  
VALUES(ÚjSor.tea);
```

Példa: trigger definíció

```
CREATE TRIGGER TeaTrig
```

```
BEFORE INSERT ON Felszolgál
```

Az esemény

```
REFERENCING NEW ROW AS ÚjSor
```

```
FOR EACH ROW
```

```
WHEN (ÚjSor.tea NOT IN  
      (SELECT név FROM Teák))
```

A feltétel

```
INSERT INTO Teák(név)  
VALUES(ÚjSor.tea);
```

Az akció