Megszorítások

Idegen kulcsok
Lokális és globális megszorítások
Triggerek

Megszorítások és triggerek

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

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 Felszolgál (
    kocsma   CHAR(20),
    sör      VARCHAR(20),
    ár       REAL,
    PRIMARY KEY (kocsma, sör)
);
```

Idegen kulcsok

- Egy reláció attribútumainak értékei egy másik reláció értékei között is meg kell, hogy jelenjenek együttesen.
- Példa: a Felszolgál(kocsma, sör, ár) táblánál azt várnánk, 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 (
  kocsma 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 Felszolgál (
  kocsma 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 szúrjanak 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* : a rendszer nem hajtja végre a törlést.
2. *Továbbgyűrűzés* : 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 (
  kocsma    CHAR(20),
  sör       CHAR(20),
  ár        REAL,
  FOREIGN KEY(sör)
     REFERENCES Sörök(név)
     ON DELETE SET NULL
     ON UPDATE CASCADE
);
```

Attribútum alapú ellenőrzések

- Egy adott oszlop értékeire vonatkozóan adhatunk meg megszorításokat.
- Adjuk hozzá a CHECK(<condition>) utasítást az attribútum deklarációjához.
- A feltételben csak az adott attribútum neve szerepelhet, más attribútumok (más relációk attribútumai is) csak alkérdésben szerepelhetnek.

Példa: attribútum alapú ellenzőrzés

```
CREATE TABLE Felszolgál (
  kocsma CHAR(20),
  sör    CHAR(20)  CHECK ( sör IN
         (SELECT name 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 törlünk, ezt a feltételt nem ellenőrzi a rendszer.

Oszlop-alapú 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: oszlop-alapú megszorítások

- Csak Joe bárjában lehetnek drágábbak a sörök 5 dollárnál:

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

Globális megszorítás

- Ezek az adatbázissémához tartoznak a relációsémákhoz és nézetekhez hasonlóan.
- CREATE ASSERTION <név>
  CHECK (<feltétel>);
- A feltétel tetszőleges táblára és oszlopra hivatkozhat az adatbázissémából.

Példa: globális megszorítás

```
CREATE ASSERTION CsakOlcsó CHECK (
   NOT EXISTS (
        SELECT kocsma
        FROM Felszolgál
        GROUP BY kocsma
        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 kocsma, mint alkesz.

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

Globális 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 TöbbAlkesz megszorításra.

Miért hasznosak a triggerek?

- A globális megszorításokkal sok mindent le tudunk írni, az ellenőrzésük azonban gondot jelenthet.
- Az attribútum- és oszlop-alapú megszorítások ellenőrzése egyszerűbb (tudjuk mikor történik), ám ezekkel nem tudunk mindent kifejezni.
- A triggerek 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*: általában valamilyen módosítás a adatbázisban, például egy sor beszúrása.
- *Feltétel* : bármilyen SQL igaz-hamis-(ismeretlen) feltétel.
- *Akció* : bármilyen SQL utasítás.

Példa: trigger

- Ahelyett, hogy visszautasítanánk a Felszolgál(kocsma, 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

A feltétel

Az akció