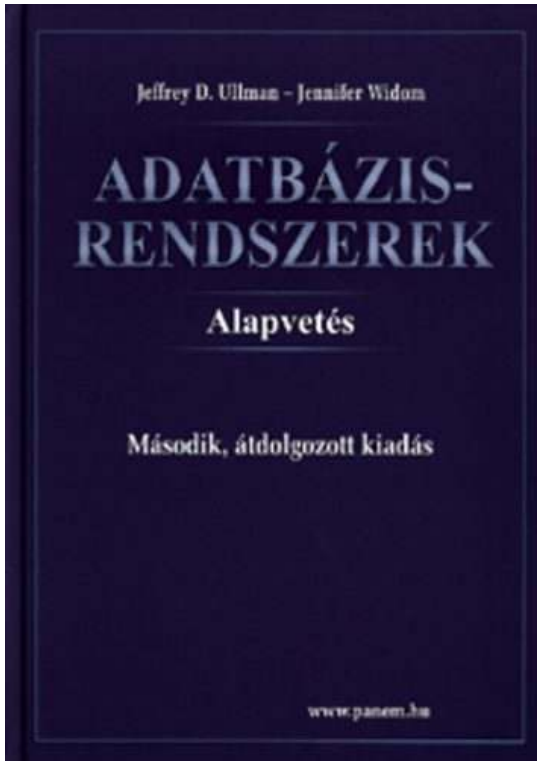
SQL DDL: triggerek és nézetek

Tankönyv: Ullman-Widom:
Adatbázisrendszerek Alapvetés
Második, átdolgozott kiadás,
Panem, 2009



7.5. Triggerek
8.1. Nézettáblák (views)
8.2. Adatok módosítása nézettáblákon keresztül
8.5. Tárolt nézettáblák

7.fejezet: Megszorítások és triggerek

- Aktív elemek – olyan kifejezés vagy utasítás, amit egyszer eltároltunk az adatbázisban és az azt várjuk tőle, hogy a megfelelő pillanatban lefusson (pl. adatok helyességének ellenőrzése)
- 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.

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*) 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
- *WHEN feltétel* : bármilyen SQL igaz-hamis-(ismeretlen) feltétel.
- *Tevékenység* : SQL utasítás, BEGIN..END, PSM tárolt eljárás

Előzetes példa egy triggerre

- Ahelyett, hogy visszautasítanánk a Felszolgál(bár, 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)

```
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;
-- Nem engedi csökkenteni a gyártásirányítók nettó bevételét.
```

Tankönyv példája (7.6. ábra)

```
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)
        DELETE FROM GyártásIrányító
        WHERE (név, cím, azonosító) IN ÚjAdat;
        INSERT INTO gyártásIrányító (SELECT * FROM …)
-- Az átlagos nettó bevétel megszorítása
```

8. fejezet: Nézettáblák

- Adatmodellek három szintje: fizikai, fogalmi és logikai szint. Nézettáblák használatának előnyei
- A *nézettábla* olyan reláció, amit tárolt táblák (*alaptáblák*) és más nézettáblák felhasználásával definiálunk.
- Kétféle létezik:
  1. *virtuális* = nem tárolódik az adatbázisban; csak a relációt megadó lekérdezés.
  2. *Materializált* = kiszámítódik, majd tárolásra kerül.

Nézettáblák létrehozása

- Deklaráció:

  CREATE [MATERIALIZED] VIEW <név>
  AS <lekérdezés>;
- Alapesetben virtualizált nézettábla jön létre.

Példa: nézettábla létrehozása

- Mit_ihat(név, sör) nézettáblában a sörivók mellett azon söröket tároljuk, amelyeket legalább egy bárban felszolgálnak az általa látogatottak közül:

```
CREATE VIEW Mit_ihat AS
    SELECT név, sör
    FROM Látogat, Felszolgál
    WHERE L.bár = F.bár;
```

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 sör FROM Mitihat
WHERE név = 'Sally';
```

Tankönyv példája: nézettáblára

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 Film
    WHERE stúdióNév = 'Paramount';
```

Triggers on Views

- Generally, it is impossible to modify a virtual view, because it doesn't exist.
- But an INSTEAD OF trigger lets us interpret view modifications in a way that makes sense.
- Example: View Synergy has (drinker, beer, bar) triples such that the bar serves the beer, the drinker frequents the bar and likes the beer.

Example: The View

Pick one copy of each attribute

```
CREATE VIEW Synergy AS
    SELECT Likes.drinker, Likes.beer, Sells.bar
    FROM Likes, Sells, Frequents
    WHERE Likes.drinker = Frequents.drinker
        AND Likes.beer = Sells.beer
        AND Sells.bar = Frequents.bar;
```

Natural join of Likes, Sells, and Frequents

Interpreting a View Insertion

- We cannot insert into Synergy --- it is a virtual view.
- But we can use an INSTEAD OF trigger to turn a (drinker, beer, bar) triple into three insertions of projected pairs, one for each of Likes, Sells, and Frequents.
  - Sells.price will have to be NULL.

The Trigger

```
CREATE TRIGGER ViewTrig
   INSTEAD OF INSERT ON Synergy
   REFERENCING NEW ROW AS n
   FOR EACH ROW
   BEGIN
        INSERT INTO LIKES VALUES(n.drinker, n.beer);
        INSERT INTO SELLS(bar, beer) VALUES(n.bar, n.beer);
        INSERT INTO FREQUENTS VALUES(n.drinker, n.bar);
   END;
```

Tankönyv példája

Példa: INSTEAD OF (helyette) típusú triggerre

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

Materializált nézettáblák

- 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”.

Példa: levelezési lista

- A következő levelezési lista cs145-aut0708 valójában egy materializált nézettábla, ami a kurzusra beiratkozott hallgatókat tartalmazza.
- Ezt négyszer frissítik egy nap.
  - A feliratkozás után közvetlen még nem feltétlen kapja meg az ember az akkor küldött emaileket.

Példa: adattárház

- Wal-Mart minden áruházának minden eladását egy adatbázisban tárolja.
- Éjszaka az új adatokkal frissítik az áruház lánc *adattárházát*, ami itt az eladások materializált nézeteiből áll.
- Az adattárházat aztán elemzők használják, hogy trendeket figyeljenek meg és odamozgassák az árukat, ahol azok a legjobb áron értékesíthetők.