Adatbázisok 1. vizsgakidolgozások

2017.05.30

I.

1.

- SELECT név FROM SZERET WHERE sör='Dreher' INTERSECT SELECT név FROM SZERET WHERE sör='Borsodi'
- $\Pi_{név(}\ \sigma_{sör='Dreher'}\ (Szeret)) \cap \Pi_{név(}\ \sigma_{sör='Borsodi'}\ (Szeret))$

2. $\Pi_{tulaj}((Felszolgál - \Pi_{Felszolgál1}\ (\sigma_{Felszolgál1.ár<Felszolgál2.ár}(Fe1szolgál1 \times Felszolgál2))) \bowtie$ Söröző)

3. **Korrelált alkérdés**: Olyan, beépített alkérdés, mely többször értékelődik ki és minden egyes kiértékelés megfelel egy olyan értékadásnak, amely az alkérdésen kívüli sorváltozóból származik.

SELECT név FROM Sörök Régi WHERE

Gyártó NOT IN (SELECT gyártó FROM Sörök WHERE név/=Régi.név)

II.

1.

- GROUP BY: Csoportosítás

⇒ A sorok csoportosítását a WHERE záradékot követő GROUP BY záradékban tudjuk megadni. A GROUP BY kulcsszót a *csoportosító* attribútumok listája követi. A legegyszerűbb esetben a FROM záradék csak egy sorváltozót tartalmaz, és a reláció sorait csoportosítjuk a csoportosító attribútumoknak megfelelően. A SELECT záradékban szereplő összesítési operátorokat a csoportokra kell alkalmazni.

- ORDER BY: Rendezés

⇒ Lekérdezés eredményének bizonyos sorrendbe rendezése. A sorrend valamely attribútum értékén alapulhat, egyenlőség esetén egy második attribútum értékén, további egyenlőség esetén egy harmadik attribútum értékén és így tovább. Az eredmény rendezése érdekében a SELECT-FROM-WHERE utasításhoz a következő záradékot adjuk hozzá: *ORDER BY<attribútumlista>.*

a. DESC- Csökkenő
b. ASC- Növekvő

- HAVING

⇒ A HAVING záradék (GROUP BY után) hatására, csak bizonyos feltételnek megfelelő csoportok kerülnek kialakításra.

Feladat: SELECT AVG(ár) átlag, város FROM Sörözők NATURAL JOIN Felszolgál GROUP BY város HAVING COUNT(sör)>9 ORDER BY átlag

$\Pi_{város,avg(ár)}(\ \tau_{avg(ár)}(\ \delta_{count(sör)>9}(\gamma_{város}(Söröző \bowtie$ Felszolgál))))

2. **Hagyományos Halmazműveletek:**

- **Unió (R U S):** R és S egyesítése azon elemek halmaza, amelyek vagy az R-ben, vagy az S-ben vannak. Egy elem, csak egyszer szerepel az egyesítésben, még akkor is, ha mindkét relációban benne is van.
  ⇒ **SQL:** UNION
- **Metszet (R ∩ S):** R és S metszete azon elemek halmaza, amelyek az R-ben és az S-ben is

benne vannak.

⇒ **SQL:** INTERSECT

- **Különbség (R-S):** R és S különbsége azon elemek halmaza, amelyek benne vannak R-ben, de nincsenek S-ben. (R-S ≠S-R)

  ⇒ **SQL:** EXCEPT

- **Halmazműveletek alkalmazásakor, figyelni kell a következőkre:**
  - ⇒ R és S relációk sémájának ugyanazt az attribútumhalmazt kell tartalmaznia, illetve a típusoknak (értéktartományoknak) az összes megfelelő attribútumpárra meg kell egyezniük R-ben és S-ben.
  - ⇒ R és S oszlopait rendezni kell úgy, hogy az attribútumok sorrendje egyforma legyen a két relációban.

Halmazműveletek multihalmazok esetén: Legyenek R és S multihalmazok, és legyen t az R n- szer, illetve az S m-szer előforduló sora. Megengedjük, hogy m és/vagy n 0 legyen.

⇒ R U S – t sor n+m-szer fog előfordulni

⇒ R ∩ S – t sor min(n,m)-szer fog szerepelni

⇒ R - S – t sor max(0,n-m)-szer fog előfordulni

3. **Eljut Feladat:** „Mely (x, y) várospárokra lehetséges egy vagy több átszállással eljutni x városból y városba?"

Datalog:

Eljut(x,y) <— Járatok(l,x,y,i,é)

Eljut(x,y) <— Eljut(x,z) AND Eljut(z,y)

SQL:

```
WITH RECURSIVE Eljut AS
(SELECT honnan, hova FROM Jaratok
UNION
SELECT Eljut.honnan, Jaratok.hova FROM Eljut, Jaratok
WHERE Eljut.hova = Jaratok.honnan)
SELECT hova FROM Eljut WHERE honnan='Budapest' AND költség<5000
```

III.

1.

- **Kulcs:** Azt mondjuk, hogy az egy vagy több attribútumból álló {A1,A2,...,An} halmaz az R kulcsa, ha:
  - Ezek az attribútumok funkcionálisan meghatározzák a reláció minden más attribútumát, nem lehet R-ben két olyan különböző sor, amely mindegyik A1,A2,...,An–nen megegyezne.
  - Nincs olyan valódi részhalmaza {A1,A2,...,An} halmaznak, amely funkcionálisan meghatározná R összes többi attribútumát, azaz a kulcsnak minimálisnak kell lennie.
  - *Megadás:*
    - ⇒ Egyszerű*: <attribútumnév> <típus> PRIMARY KEY vagy <attribútumnév> <típus> UNIQUE*
    - ⇒ Összetett: A CREATE TABLE utasításban az attribútum deklaráció után a kiegészítő részben meg lehet adni további tábla elemeket*: PRIMARY KEY (attrnév$_1$, attrnév$_k$)*

- **Külső/idegen kulcs:** Az SQL-ben egy reláció azon attribútumát vagy attribútumait idegen kulcsnak deklarálhatjuk, amelyek egy másik reláció (ez lehet akár ugyanaz a reláció is) bizonyos attribútumaira hivatkoznak. Ez két dolgot jelent egyszerre:
    - A másik reláció azon attribútumait, amelyekre hivatkozunk elsődleges kulcsként vagy UNIQUE-ként kell deklarálni abban a relációban.
    - Az idegen kulcs értékeinek, amelyek előfordulnak az első relációban, elő kell fordulniuk a hivatkozott attribútumokban is a másik reláció valamelyik sorában.
    - *Megadás:*

        ⇒ *<attribútumnév> <típus> REFERENCES relációnév(attribútum lista)*

        ⇒ *FOREIGN KEY (attribútum lista) REFERENCES relációnév (attribútum lista)*
- **Hivatkozási épség:** Ha egy érték megjelenik valahol egy környezetben, akkor ugyanez az érték egy másik, az előzővel összefüggő környezetben is megjelenik. Általánosságban, ha az R reláció egy sorának A attribútumában szerepel egy v érték, akkor tervezési szándékaink miatt elvárhatjuk, hogy ez a v érték egy másik S reláció valamely sorának egy bizonyos komponensében (például B-ben) is megjelenjen. Relációs algebrai kifejezéssel: $\pi A(R) \subseteq \pi B(S)$ (ezzel ekvivalens kifejezés: $\pi A(R) - \pi B(S) = \emptyset$) .

Feladat:

```
CREATE TABLE Sörök(
        név char(20) PRIMARY KEY
        gyártó   number
);
```

```
CREATE TABLE Sörözők(
        név    char(20) PRIMARY KEY
        város number
        tulaj   char(20)
        tel     number

);
```

```
CREATE TABLE Felszolgál(
        söröző char(20) FOREIGN KEY REFERENCES Söröző(név)
                ON DELETE SET NULL
                ON UPDATE CASCADE
        sör     char(20) FOREIGN KEY REFERENCES Sörök(név)
                ON DELETE SET NULL
                ON UPDATE CASCADE
        ár      number
);
```

2. **SQL DML utasítások**:
    - **Beszúrás**: *INSERT INTO R(A1,A2,...An) VALUES (v1,v2,...,vn)*

    ⇒ Ai attribútumhoz vi értéket rendeljük. Ha az attribútumlista nem tartalmazza R összes attribútumát, akkor a hiányzó attribútumok az alapértelmezés szerinti értéket kapják. Ha az összes attribútumra szeretnénk értéket beszúrni, az attribútumlista elhagyható. Azonban, ha nem tudjuk pontosan, az attribútumok sorrendjét, soroljuk fel azokat, ahogy tudjuk és adjuk meg abban a sorrendben az értékeket. Beszúrhatunk alkérdésből nyert adatokat is
    - **Törlés:** *DELETE FROM R WHERE <feltétel>;*

    ⇒ R-ből kitörlődik minden olyan sor, amely eleget tesz a feltételnek.
    - **Módosítás:** *UPDATE R SET <új értékadások> WHERE <feltétel>;*

    ⇒ Egy vagy több létező sor komponenseinek értékét megváltoztatjuk.

Feladat: *DELETE Sörözők WHERE név NOT IN (SELECT söröző FROM Felszolgál) AND név NOT IN (SELECT söröző FROM Látogat)*

3. **Kurzorok:**

- **DEF:** Iterátorok ahhoz, hogy adatbázisok sorait tudjuk kezelni PL/SQL-ben

⇒ Implicit:

FOR ciklusváltozó_név IN (SELECT
utasítás)
LOOP
utasítások;
END LOOP;

⇒ Explicit: *CURSOR név [paraméterlista] [RETURN sortípus] IS select utasítás;*

- **Tábla megváltoztatása:**

⇒ Kurzor definicíója után*: FOR UPDATE / DELETE*

⇒ Legutóbbi FETCH által betöltött sor módosíthatóvá tétele: *UPDATE / DELETE <utasítás> WHERE CURRENT OF <kurzornév>*

Feladat:

```
CREATE PROCEDURE kiír (gy: char) IS
    CURSOR curs(gya char) is SELECT * FROM Felszolgál NATURAL JOIN Sörök
    WHERE gyártó=gya;
    Sor curs%ROWTYPE;
BEGIN
   OPEN curs(gy);
   LOOP
      FETCH curs INTO Sor;
          EXIT WHEN curs%NOTFOUND;
      UPDATE Felszolgál Régi SET ár=Régi.ár*1,10 WHERE CURRENT OF curs;
 END LOOP;
 CLOSE curs;
END;
```

IV.

1. **Átírás:**

Tantargy(tkod,megnev,kredit,elofelt)

Felelos(tantargy,tanszek)

Tanszek(tazon,tsznev,weblap)

Dolgozik(tanszek,oktato)

Oktato(eazon,nev,beosztas,fizetes)

2.

- **Funkcionális függőség:** Ha R két sora megegyezik az A1,A2,...,An attribútumokon (azaz ezen attribútumok mindegyikéhez megfeleltetett komponensnek ugyanaz az értéke a két sorban) , akkor meg kell egyezniük más attribútumok egy B1,B2,...,Bm sorozatán.
  Formálisan A1A2...An -> B1B2...Bm –mel jelöljük és azt mondjuk hogy:„A1A2...An funkcionálisan meghatározza B1B2...Bm-et. Ha biztosak lehetünk abban, hogy az R reláció minden előfordulása olyan, amelyen az adott FD igaz, akkor azt mondhatjuk, hogy R kielégíti FD-t.

- **Armstrong Axiómák:**
  - ⇒ *Reflexivitás:* Ha $\{B_1,B_2,..,B_m\} \subseteq \{A_1,A_2,...,A_n\}$, akkor $A_1A_2...A_n \rightarrow B_1B2 ...B_m$. Ezeket neveztük triviális függőségeknek.
  - ⇒ *Bővítés:* Ha $A_1A_2...A_n \rightarrow B_1B_2...B_m$,akkor $A_1A_2...A_n\ C_1C_2...C_k \rightarrow B_1B_2...B_mC_1C_2...C_k$ bármely $\{C_1, C2\ ,\ ...,\ C_k\}$ attribútumhalmazra. Mivel tudjuk, hogy a C-k lehetnek *A*-k és B-k is, a bal oldalról ki kell szűrnünk a duplikált attribútumokat, majd ugyanezt kell tennünk a jobb oldalakkal is.
  - ⇒ *Tranzitivitás:* Ha $A_1A_2...A_n \rightarrow B_1B_2...B_m$ *és* $B_1B_2...B_m \rightarrow C_1C_2...C_k$ teljesül , akkor $A_1A_2...A_n \rightarrow C_1C_2...C_k$ szintén teljesül.

Feladat:

a. **Tranzitivitás + Összevonhatósági szabály:** Az $A_1A_2...A_n$—>Bi (i=1,...,m) függőséget helyettesíthetjük az $A_1A_2...A_n \rightarrow B_1B_2...B_m$ funkcionális függőségekből álló halmazzal.
b. ???

3. **Boyce-Codd normálforma:** Az R reláció BCNF-ben van akkor és csak akkor, ha minden olyan esetben, ha R-ben érvényes egy A1A2...An->B1B2...Bm nem triviális függőség, akkor az {A1,A2,...,An} halmaz R szuperkulcsa.

   - **Triviális függőség:** Ezek azok az A1A2...An—>B1B2...Bm függőségek, ameylekre: $\{B1B2...Bm\} \subseteq \{A1A2...An\}$.
   - **Szuperkulcs:** Azokat az attribútumhalmazokat, amelyek tartalmaznak kulcsot, szuperkulcsoknak nevezzük (kulcsnál bővebb halmaz). Minden kulcs egyben szuperkulcs is.

   Feladat:

   AB->C $(AB)^+=\{A,B,C,D,E\}$=>Kulcs

   C->D $(C)^+=\{C,D,B,E\}$ $(AC)^+=\{A,C,D,B,E\}$=>Kulcs

   D->B $(B)^+=\{D,B,E\}$ $(AD)^+=\{A,D,B,C,E\}$=>Kulcs

   D->E

   Sérti: C->D, D->B, D->E;