

Adatbázisok 1. vizsgakidolgozások

2017.05.23

I.

1. SELECT filmcím, év FROM Filmek WHERE stúdió='Disney' AND műfaj/='Rajzfilm' AND filmcím LIKE %tenger%
2. $\Pi_{filmcím, év}(Filmek - \Pi_{Filmek1}(\sigma_{Filmek1.hossz < Filmek2.hossz}(Filmek1 \times Filmek2))$
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 filmcím FROM Filmek Régi WHERE

év < ANY (SELECT év FROM Filmek WHERE filmcím=Régi.filmcím)

II.

1.

- $R'(D,E): \{(3,2), (5,6), (2,0), (6,8), (7,12)\}$ – Sum(E): 28
- Megfelel a feltételnek: $\{(2,4), (2,5), (3,4), (1,4)\}$ csoportosítás C, Sum(B) szerint: $\{(4,6), (5,2)\}$
- $R \bowtie S$:

A	B	C
1	2	4
1	2	5
2	3	4
0	2	4
0	2	5

Csoportosítás A, Sum(C) szerint: $\{(1,9), (0,9), (2,4)\}$

2. SELECT AVG(fizetés) átlag FROM Dolgozók WHERE belépés>1989 GROUP BY foglalkozás HAVING COUNT(dnév)>2 ORDER BY átlag DESC
 - SELECT záradékában szerepelhet *, tábla attribútum nevei, vagy kifejezések és konstansok is
3. **Eljut Feladat:** „Mely (x, y) várospárokra lehetséges egy vagy több átszállással eljutni x városból y városba?”

Datalog:

$Eljut(x,y) \leftarrow J\acute{a}ratok(l,x,y,i,\acute{e})$

$Eljut(x,y) \leftarrow Eljut(x,z) \text{ AND } Eljut(z,y)$

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 * FROM Eljut

III.

1.

- **Kulcs:** Azt mondjuk, hogy az egy vagy több attribútumból álló $\{A_1, A_2, \dots, A_n\}$ 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 A_1, A_2, \dots, A_n -nen megegyezne.
 - Nincs olyan valódi részhalmaza $\{A_1, A_2, \dots, A_n\}$ halmaznak, amely funkcionálisan meghatározná R összes többi attribútumát, azaz a kulcsnak minimálisnak kell lennie.
 - *Megadás:*
 - \Rightarrow Egyszerű: $\langle \text{attribútumnév} \rangle \langle \text{típus} \rangle \text{PRIMARY KEY}$ vagy $\langle \text{attribútumnév} \rangle \langle \text{típus} \rangle \text{UNIQUE}$
 - \Rightarrow Összetett: A CREATE TABLE utasításban az attribútum deklaráció után a kiegészítő részben meg lehet adni további tábla elemeket: *PRIMARY KEY* ($\text{attrnév}_1, \text{attrnév}_k$)
- **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:*
 - $\Rightarrow \langle \text{attribútumnév} \rangle \langle \text{típus} \rangle \text{REFERENCES relációnév}(\text{attribútum lista})$
 - $\Rightarrow \text{FOREIGN KEY}(\text{attribútum lista}) \text{REFERENCES relációnév}(\text{attribútum lista})$

Védekezés idegen kulcs megszorítás megsértése ellen:

Előfordulhat olyan eset, hogy az adatok frissítése sértene az idegen kulcs megszorítást. Ezek megkerüléséhez, a tábla deklarálásakor meg kell adni, hogy a kulcs ellenőrzése késleltethető (DEFERRABLE) vagy nem (NOT DEFERRABLE). Ez utóbbi az alapértelmezés. Ha egy megszorítást DEFERRABLE-ként deklarálunk, lehetőség van arra, hogy a megszorítás ellenőrzését késleltesse a tranzakció végéig. A DEFERRABLE szót követi INITIALY DEFERRED (tranzakció végéig késleltetve van az ellenőrzés) vagy INITIALY IMMEDIATE (az ellenőrzés minden egyes utasítás után azonnal megtörténik) kiegészítés

- **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 Filmek(cím char(20) év number . . . PRIMARY KEY(cím, év));	CREATE TABLE Szerepel(cím char(20) év number színész char(20) FOREIGN KEY(cím, év) REFERENCES Filmek(cím, év));
--	--

2. SQL DML utasítások:

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

⇒ A_i attribútumhoz v_i 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: *UPDATE Filmek SET műfaj='Vígjáték' WHERE hossz=(SELECT MAX(hossz) FROM Filmek)*

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;*

- **Használat:**

⇒ Megnyitás: *OPEN kurzornév [aktuális paraméterlista];*

⇒ Betöltés változóba: *FETCH {kurzornév | kurzorváltozó név} INTO {rekordnév | változónév lista}*

⇒ Bezárás: *CLOSE {kurzornév | kurzorváltozó név};*

Feladat:

```
CREATE PROCEDURE kiír (s: char) IS
    Sor Filmek%ROWTYPE;
    CURSOR curs(st char) IS SELECT filmcím, műfaj FROM Filmek WHERE stúdió=s
    ORDER BY műfaj;
BEGIN
    OPEN curs(s);
    LOOP
        FETCH curs INTO Sor;
        EXIT WHEN curs%NOTFOUND;
        Out: Sor;
    END LOOP;
    CLOSE curs;
END;
```

IV.

1. 3 féle átirás:

- ⇒ **E/R stílusban:** Egy reláció minden alosztályra, de az általános osztályból csak a kulcsokat vesszük hozzá a saját attribútumokhoz.
- ⇒ **Objektumorientált stílusban:** Egy reláció minden alosztályra, felsorolva az összes tulajdonságot, beleértve az öröklötteket is.
- ⇒ **Nullértékek használatával:** Egyetlen reláció az öröklődésben résztvevő összes osztályra. Ha egy egyed nem rendelkezik egy alosztály speciális tulajdonságával, akkor ezt az attribútumot NULL értékkel töltjük majd ki.

Feladat: E/R stílusban

Filmek(cím,év,hossz,műfaj)

Rajzfilm(cím,év,rajzolta)

Bűnügyi(cím,év,fegyver)

Színészek(név,lakcím,cím,év)

2. **Dekompozíció:** R felbontása azt jelenti, hogy R attribútumait szétszítjuk úgy, hogy két új reláció sémáját alakítjuk ki belőlük. Legyen adott R reláció $\{A_1, A_2, \dots, A_n\}$ sémával. R-t felbonthatjuk $S(B_1, B_2, \dots, B_m)$ és $T(C_1, C_2, \dots, C_k)$ relációkra úgy, hogy:

- $\{A_1, A_2, \dots, A_n\} = \{B_1, B_2, \dots, B_m\} \cup \{C_1, C_2, \dots, C_k\}$
- $S = \pi_{B_1, B_2, \dots, B_m}(R)$
- $T = \pi_{C_1, C_2, \dots, C_k}(R)$

A felbontás célja, hogy egy relációt többel helyettesítsünk úgy, hogy ezzel megszüntessük az anomáliákat.

Chase-teszt: Felbontható-e veszteségmentesen?

Tabló:

A	B	C	D
a	b	c ₁	d ₁
a ₂	b	c	d ₂
a ₃	b ₃	c	d

Az első két sor megegyezik B-n, ezért B->C miatt c₁=c

A	B	C	D
a	b	c	c ₁
a ₂	b	c	d ₂
a ₃	b ₃	c	d

Most a sorok megegyeznek C-n, de nincs olyan szabály, ami C-ből következne.

Nincs olyan sor, ahol ne lennének indexelt tagok, ezért a felbontás nem veszteségmentes.

3. **Boyce-Codd normálforma:** Az R reláció BCNF-ben van akkor és csak akkor, ha minden olyan esetben, ha R-ben érvényes egy $A_1A_2...A_n \rightarrow B_1B_2...B_m$ nem triviális függőség, akkor az $\{A_1, A_2, ..., A_n\}$ halmaz R szuperkulcsa.

- **Triviális függőség:** Ezek azok az $A_1A_2...A_nB_1 \rightarrow B_2...B_m$ függőségek, amelyekre: $\{B_1B_2...B_m\} \subseteq \{A_1A_2...A_n\}$.
- **Szuperkulcs:** Azokat az attribútumhalmazokat, amelyek tartalmazzak kulcsot, szuperkulcsoknak nevezzük (kulcsnál bővebb halmaz). Minden kulcs egyben szuperkulcs is.

Feladat:

$A \rightarrow D$ $(A)^+ = \{A, D\} = (AD)^+$ $(AB)^+ = \{A, B, D\}$ Sértik: $A \rightarrow D$, $DE \rightarrow B$

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

$DE \rightarrow B$ $(AE)^+ = \{A, E, D, B\}$

2017.05.30

I.

1.

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

2. $\prod_{tulaj}((Felszolgál - \prod_{Felszolgál1}(\sigma_{Felszolgál1.ár < Felszolgál2.ár}(Felszolgál1 \times Felszolgál2))) \bowtie Sörözők)$

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

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

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. ($R-S \Rightarrow S-R$)

⇒ **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.

⇒ $R \cup S - t$ sor $n+m$ -szer fog előfordulni

⇒ $R \cap S - t$ 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:

$Eljut(x,y) \leftarrow J\acute{a}ratok(l,x,y,i,\acute{e})$

$Eljut(x,y) \leftarrow Eljut(x,z) \text{ AND } Eljut(z,y)$

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 $\{A_1, A_2, \dots, A_n\}$ 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 A_1, A_2, \dots, A_n -nen megegyezne.

- Nincs olyan valódi részhalmaza $\{A_1, A_2, \dots, A_n\}$ halmaznak, amely funkcionálisan meghatározná R összes többi attribútumát, azaz a kulcsnak minimálisnak kell lennie.

- *Megadás:*

⇒ Egyszerű: $\langle \text{attribútumnév} \rangle \langle \text{típus} \rangle \text{ PRIMARY KEY}$ vagy $\langle \text{attribútumnév} \rangle \langle \text{típus} \rangle \text{ UNIQUE}$

⇒ Összetett: A CREATE TABLE utasításban az attribútum deklaráció után a kiegészítő részben meg lehet adni további tábla elemeket: $\text{PRIMARY KEY} (\text{attr}_1, \text{attr}_k)$

- **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:*
 - $\Rightarrow <attribútumnév> <típus> REFERENCES relációnév(attribútum lista)$
 - $\Rightarrow FOREIGN KEY (attribútum lista) REFERENCES relációnév (attribútum lista)$
- **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)*
 - \Rightarrow 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>;*
 - \Rightarrow R-ből kitörlődik minden olyan sor, amely eleget tesz a feltételnek.
- **Módosítás:** *UPDATE R SET <új értékadások> WHERE <feltétel>;*
 - \Rightarrow 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 definíciója 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 A_1, A_2, \dots, A_n attribútumokon (azaz ezen attribútumok mindegyikéhez megfeleltetett komponensnek ugyanaz az értéke a két sorban), akkor meg kell egyezniük más attribútumok egy B_1, B_2, \dots, B_m sorozatán.

Formálisan $A_1A_2\dots A_n \rightarrow B_1B_2\dots B_m$ –mel jelöljük és azt mondjuk hogy: „ $A_1A_2\dots A_n$ funkcionálisan meghatározza $B_1B_2\dots B_m$ -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:**

- \Rightarrow *Reflexivitás:* Ha $\{B_1, B_2, \dots, B_m\} \subseteq \{A_1, A_2, \dots, A_n\}$, akkor $A_1A_2\dots A_n \rightarrow B_1B_2\dots B_m$. Ezeket neveztük triviális függőségeknek.
- \Rightarrow *Bővítés:* Ha $A_1A_2\dots A_n \rightarrow B_1B_2\dots B_m$, akkor $A_1A_2\dots A_n C_1C_2\dots C_k \rightarrow B_1B_2\dots B_m C_1C_2\dots C_k$ bármely $\{C_1, C_2, \dots, C_k\}$ attribútumhalmazra. Mivel tudjuk, hogy a C-k lehetnek A-k és B-k is, a bal oldalról ki kell szűrniük a duplikált attribútumokat, majd ugyanezt kell tennünk a jobb oldalakkal is.
- \Rightarrow *Tranzitivitás:* Ha $A_1A_2\dots A_n \rightarrow B_1B_2\dots B_m$ és $B_1B_2\dots B_m \rightarrow C_1C_2\dots C_k$ teljesül, akkor $A_1A_2\dots A_n \rightarrow C_1C_2\dots C_k$ szintén teljesül.

Feladat:

- Tranzitivitás + Összevonhatósági szabály:** Az $A_1A_2\dots A_n \rightarrow B_i$ ($i=1, \dots, m$) függőséget helyettesíthetjük az $A_1A_2\dots A_n \rightarrow B_1B_2\dots B_m$ funkcionális függőségekből álló halmazzal.
- ???

3. **Boyce-Codd normálforma:** Az R reláció BCNF-ben van akkor és csak akkor, ha minden olyan esetben, ha R-ben érvényes egy $A_1A_2\dots A_n \rightarrow B_1B_2\dots B_m$ nem triviális függőség, akkor az $\{A_1, A_2, \dots, A_n\}$ halmaz R superkulcsa.

- **Triviális függőség:** Ezek azok az $A_1A_2\dots A_n \rightarrow B_1B_2\dots B_m$ függőségek, amelyekre: $\{B_1B_2\dots B_m\} \subseteq \{A_1A_2\dots A_n\}$.
- **Szuperkulcs:** Azokat az attribútumhalmazokat, amelyek tartalmazznak kulcsot, superkulcsoknak nevezzük (kulcsnál bővebb halmaz). Minden kulcs egyben superkulcs is.

Feladat:

AB \rightarrow C	$(AB)^+ = \{A, B, C, D, E\} \Rightarrow$ Kulcs
C \rightarrow D $(C)^+ = \{C, D, B, E\}$	$(AC)^+ = \{A, C, D, B, E\} \Rightarrow$ Kulcs
D \rightarrow B $(B)^+ = \{D, B, E\}$	$(AD)^+ = \{A, D, B, C, E\} \Rightarrow$ Kulcs
D \rightarrow E	

Sérti: C \rightarrow D, D \rightarrow B, D \rightarrow E;