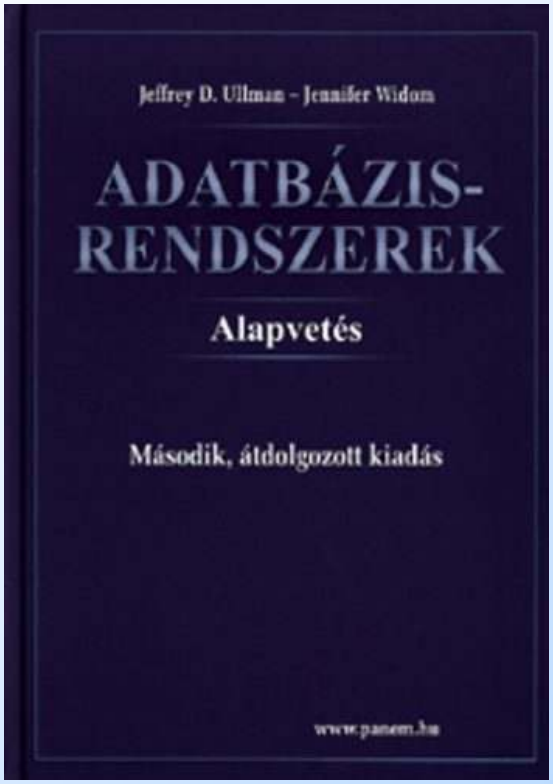
Relációs adatbázisok tervezése 3.rész (normálformák)



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

3.3. Relációs adatbázissémák tervezése
- Boyce-Codd normálforma (BCNF)

3.5. Harmadik normálforma (3NF)

(Jeffrey D. Ullman 2007 előadásdiái alapján, Benczúr András, Kiss Attila és Kósa Balázs előadásainak felhasználásával, Hajas Csilla)

Boyce-Codd normálforma

- *R* reláció *BCNF* normálformában van, ha minden *X -> Y* nemtriviális FF-re *R*-ben *X* szuperkulcs.
  - *Nemtriviális:* *Y* nem része *X*-nek.
  - *Szuperkulcs:* tartalmaz kulcsot (ő maga is lehet kulcs).

Példa

Sörivók(<u>név</u>, cím, <u>kedveltSörök</u>, gyártó, kedvencSör)

FF-ek: név->cím kedvencSör, kedveltSörök->gyártó

- Itt egy kulcs van: {név, kedveltSörök}.
- A baloldalak egyik FF esetén sem szuperkulcsok.
- Emiatt az *Sörivók* reláció nincs BCNF normálformában.

Még egy példa

Sörök(<u>név</u>, gyártó, gyártóCím)

FF-ek: név->gyártó, gyártó->gyártóCím

- Az egyetlen kulcs {név} .
- név->gyártó nem sérti a BCNF feltételét, de a gyártó->gyártóCím függőség igen.

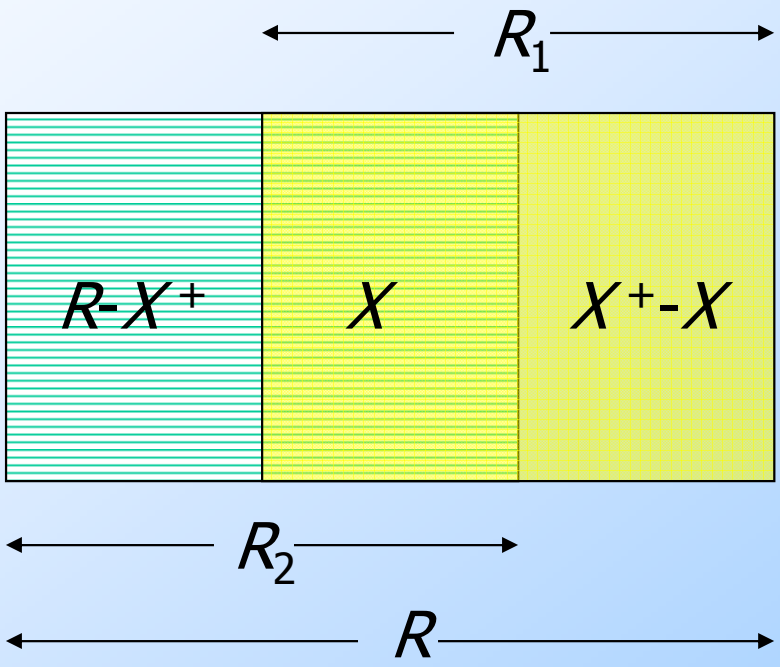
BCNF-re való felbontás

- Adott *R* reláció és *F* funkcionális függőségek.
- Van-e olyan *X* -> *Y* FF, ami sérti a BCNF-t?
  - Ha van olyan következmény FF *F*-ben, ami sérti a BCNF-t, akkor egy *F*-beli FF is sérti.
- Kiszámítjuk X^+-t:
  - Ha itt nem szerepel az összes attribútum, *X* nem szuperkulcs.

R dekomponálása X -> Y felhasználásával

- R-t helyettesítsük az alábbiakkal:
  1. $R_1 = X^+$.
  2. $R_2 = R - (X^+ - X)$.
- *Projektáljuk* a meglévő F-beli FF-eket a két új relációsémára.

Dekomponálási kép



Példa: BCNF dekompozíció

Sörivók(<u>név</u>, cím, <u>kedveltSörök</u>, gyártó, kedvencSör)

F = név->cím, név->kedvenSör,
kedveltSörök->gyártó

- Vegyük név->cím FF-t:
- {név}$^+$ = {név, cím, kedvencSör}.
- A dekomponált relációsémák:
  1. Sörivók1(<u>név</u>, cím, kedvencSör)
  2. Sörivók2(<u>név</u>, <u>kedveltSörök</u>, gyártó)

Példa -- folytatás

- Meg kell néznünk, hogy az Sörivók1 és Sörivók2 táblák BCNF-ben vannak-e.
- Az FF-ek projektálása könnyű.
- A Sörivók1(<u>név</u>, cím, kedvenSör), az FF-ek név->cím és név->kedvencSör.
  - Tehát az egyetlen kulcs: {név}, azaz az Alkeszek1 BCNF-ben van.

Példa -- folytatás

- Az Sörivók2(<u>név</u>, <u>kedveltSörök</u>, gyártó) esetén az egyetlen FF

  kedveltSörök->gyártó, az egyetlen kulcs: {név, kedveltSörök}.
  - Sérül a BCNF.
- kedveltSörök$^+$ = {kedveltSörök, gyártó}, a *Sörivók2* felbontása:
  1. Sörivók3(<u>kedveltSörök</u>, gyártó)
  2. Sörivók4(<u>név</u>, <u>kedveltSörök</u>)

Példa -- befejezés

- Az *Sörivók* dekompozíciója tehát:
  1. Sörivók1(<u>név</u>, cím, kedvencSör)
  2. Sörivók 3(<u>kedveltSörök</u>, gyártó)
  3. Sörivók 4(<u>név</u>, <u>kedveltSörök</u>)
- A *Sörivók1* a sörivókról, a *Sörivók3* a sörökről, az *Sörivók4* a sörivók és kedvelt söreikről tartalmaz információt.

Miért működik a BCNF?

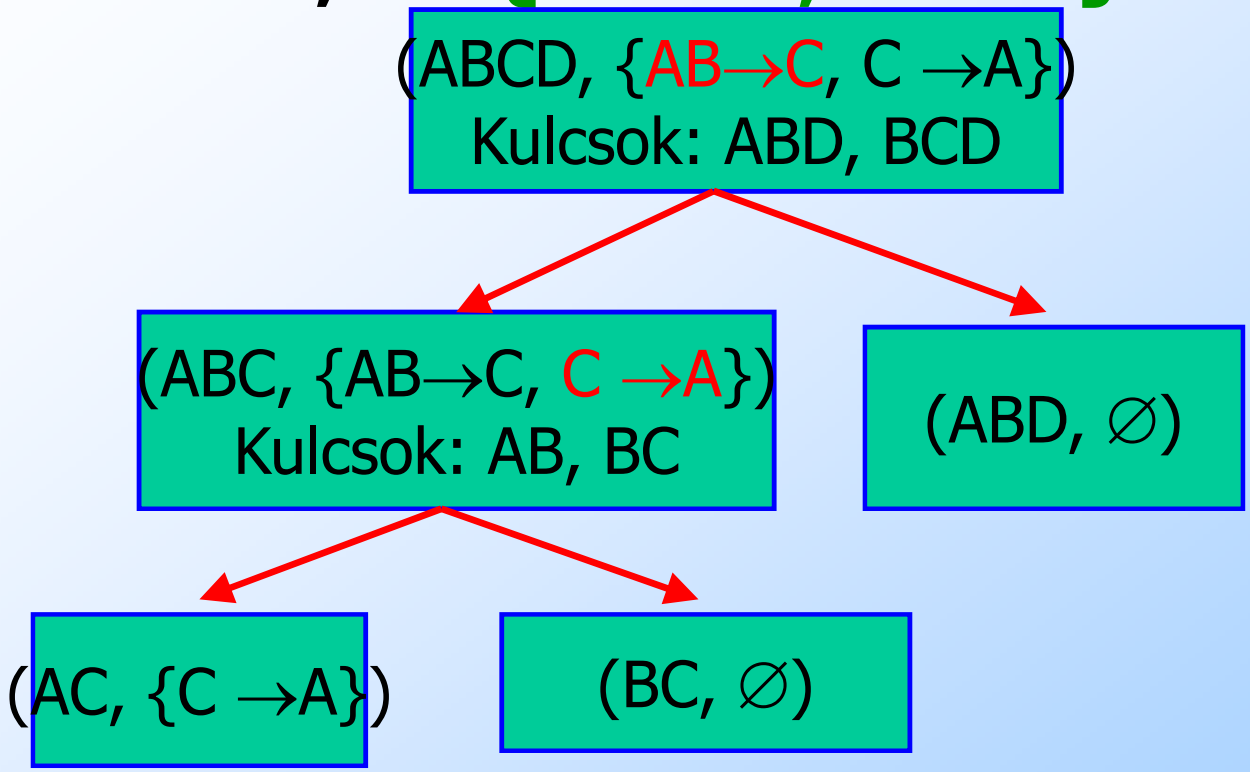
- (R, F) esetén ha $R_1,\ldots, R_k$ egy veszteségmentes felbontás, S_1, S_2 pedig R_1 veszteségmentes felbontása, akkor S_1, S_2, $R_2,\ldots, R_k$ is veszteségmentes felbontás.
- Könnyen ellenőrizhető, hogy a fenti R_1, R_2 veszteségmentes. (Miért?)
- Minden két attribútumú séma BCNF normálformában van.

Feladat: bizonyítsuk be, hogy ha a fenti R(A, B, C) reláció esetén $B \rightarrow C$ teljesül, akkor az $R_1(A, B)$, $R_2(B, C)$ felbontás mindig veszteségmentes.

- Az algoritmus tehát valóban veszteségmentes felbontást ad, ám sajnos exponenciális lépésszámú is lehet a függőségek vetítése miatt.

Példa: BCNF-ra való felbontás

R=ABCD, F={AB→C, C →A}



Tehát d=(AC,BC,ABD) veszteségmentes BCNF dekompozíció. (∅ azt jelenti, hogy csak a triviális függőségek teljesülnek a sémában.)

Tankönyv 3.5.2. feladata (111.o.)

- Órarend adatbázis: Kurzus(K), Oktató(O), Időpont(I), Terem(T), Diák(D), Jegy(J)
- Feltételek:

  Egy kurzust csak egy oktató tarthat: $K \rightarrow O$.

  Egy helyen egy időben egy kurzus lehet: $IT \rightarrow K$.

  Egy időben egy tanár csak egy helyen lehet: $IO \rightarrow T$.

  Egy időben egy diák csak egy helyen lehet: $ID \rightarrow T$.

  Egy diák egy kurzust egy végső jeggyel zár: $KD \rightarrow J$.
- R=KOITDJ F= $\{K \rightarrow O, IT \rightarrow K, IO \rightarrow T, ID \rightarrow T, KD \rightarrow J\}$
- Feladat: Adjuk meg az algoritmussal egy BCNF dekompozícióját!

A 3normálforma -- motiváció

- Bizonyos FF halmazok esetén a felbontáskor elveszíthetünk függőségeket.
- *AB -> C* és *C -> B*.
  - Példa1: *A* = utca, *B* = város, *C* = irányítószám.
  - Példa2: *A* = oktató, *B* = időpont, *C* = kurzus.
- Két kulcs van: {*A,B*} és {*A,C*}.
- *C -> B* megsérti a BCNF-t, tehát *AC*, *BC*-re dekomponálunk.
- A probléma az, hogy *AC* és *BC* sémákkal nem tudjuk kikényszeríteni *AB -> C* függőséget.

A proléma megoldása: 3NF

- 3. normálformában (3NF) úgy módosul a BCNF feltétel, hogy az előbbi esetben nem kell dekomponálnunk.
- Egy attribútum *prím*, ha legalább egy kulcsnak eleme.
- *X* ->*A* megsérti 3NF-t akkor és csak akkor, ha *X* nem szuperkulcs és *A* nem prím.

Példa: 3NF

- A problematikus esetben az *AB -> C* és *C -> B* FF-ek esetén a kulcsok *AB* és *AC*.
- Ezért *A*, *B* és *C* mindegyike prím.
- Habár *C -> B* megsérti a BCNF feltételét, 3NF feltételét már nem sérti meg.

Miért hasznos 3NF és BCNF?

- A dekompozícióknak két fontos tulajdonsága lehet:
  1. *Veszteségmentes összekapcsolás* : ha a projektált relációkat összekapcsoljuk az eredetit kapjuk vissza.
  2. *Függőségek megőrzése* : a projektált relációk segítségével is kikényszeríthetőek az előre megadott függőségek.

3NF és BCNF -- folytatás

- Az (1) tulajdonság teljesül a BCNF esetében.
- A 3NF (1) és (2)-t is teljesíti.
- A BCNF esetén (2) sérülhet.
  - Az *utca-város-iSzám* erre volt egy példa.

A veszteségmentes összekapcsolás ellenőrzése

- Ha R-t R_1, R_2,..., R_k relációkra bontjuk, visszakapjuk-e R-t összekapcsolással?
- Az összekapcsoltban R minden sora benne lesz majd.
- A kérdés tehát: nem kerül-e be az eredménybe olyan sor, ami eredetileg nem volt ott?

Minimális bázis létrehozása

1. Jobboldalak szétvágása.
2. Próbáljuk törölni az FF-eket egymás után. Ha a megmaradó FF-halmaz nem ekvivalens az eredetivel, akkor nem törölhető az épp aktuális FF.
3. Egymás után próbáljuk csökkenteni a baloldalakat, és megnézzük, hogy az eredetivel ekvivalens FF-halmazt kapunk-e.

3NF-re bontás – (2)

- A minimális bázis minden FF-e megad egy sémát a felbontásban.
  - A séma a jobb- és baloldalak uniója lesz.
- Ha a minimális bázis FF-jei között nincs kulcs, akkor hozzáadunk a felbontáshoz egy olyan sémát, amelyet egy kulcs ad meg.

Példa: 3NF felbontás

- A reláció: R = ABCD.
- FF-ek: *A->B* és *A->C*.
- Felbontás: AB és AC az FF-ekből és AD-t is hozzá kell venni, mert AB, AC egyike sem kulcs.

Miért működik?

- Megőrzi a függőségeket: minden FF megmarad a minimális bázisból.
- Veszteségmentes összekapcsolás: a CHASE algoritmussal ellenőrizhető (a kulcsból létrehozott séma itt lesz fontos).
- 3NF: a minimális bázis tulajdonságaiból következik.

Minimális bázist kiszámító algoritmus

1. Kezdetben G az üreshalmaz.
2. Minden $X \rightarrow Y \in F$ helyett vegyük az $X \rightarrow A$ függőségeket, ahol $A \in Y - X$).
   Megj.: Ekkor minden G-beli függőség $X \rightarrow A$ alakú.
3. Minden $X \rightarrow A \in G$-re, amíg van olyan $B \in X$-re $A \in (X - B)^+$ a G-szerint, vagyis $(X - B) \rightarrow A$ teljesül, akkor $X := X - B$.
   Megjegyzés: E lépés után minden baloldal minimális lesz.
4. Minden $X \rightarrow A \in G$-re, ha $X \rightarrow A \in (G - \{ X \rightarrow A \})^+$, vagyis ha elhagyjuk az $X \rightarrow A$ függőséget G-ből, az még mindig következik a maradékból, akkor $G:=G - \{X \rightarrow A\}$.
   Megjegyzés: Végül nem marad több elhagyható függőség.

- Példa: $F = \{ H \rightarrow T, U \rightarrow HS, HD \rightarrow KU \}$.
  Mi a minimális bázis?

Normálformák (3NF)

- **Definíció: $F^*:=\{X\rightarrow Y \mid F \vdash X\rightarrow Y\}$ F-ből levezethető összes függőség (F levezethetőség szerinti lezártja).**
- **Nézzük meg, hogy lehet minimálisra csökkenteni egy F függőségi halmazt.**
- **G az F minimális bázisa (másképpen minimális fedése), ha**
  - **$F^*=G^*$ (bázis, vagy másképpen fedés),**
  - **G minden függőségében a jobb oldalak egyeleműek, (jobb oldalak minimálisak)**
  - **G-ből nem hagyható el függőség, hogy F bázisa maradjon, (minimális halmaz)**
  - **G függőségeinek bal oldala nem csökkenthető, hogy F bázisa maradjon (bal oldalak minimálisak).**
- **Kérdések:**
  - **Minden F-nek létezik minimális bázisa?**
  - **Egyértelmű-e a minimális bázis?**
  - **Hogyan lehet adott F esetén meghatározni egy minimális bázist, lehetőleg hatékonyan?**

Normálformák (3NF)

Mohó algoritmus minimális bázis előállítására:

1. **Jobb oldalak minimalizálása:**

   $X \to A1,\ldots,Ak$ függőséget cseréljük le az

   $X \to A1, \ldots, X \to Ak$ k darab függőségre.

2. **A halmaz minimalizálása:**

   Hagyjuk el az olyan $X \to A$ függőségeket, amelyek a bázist nem befolyásolják, azaz

   while F változik

   if $(F-\{X \to A\})^* = F^*$ then $F := F-\{X \to A\}$;

3. **Bal oldalak minimalizálása:**

   Hagyjuk el a bal oldalakból azokat az attribútumokat, amelyek a bázist nem befolyásolják, azaz

   while F változik

   for all $X \to A \in F$

   for all $B \in X$

   if $((F-\{X \to A\}) \cup \{(X-\{B\}) \to A\})^* = F^*$ then $F := (F-\{X \to A\}) \cup \{X-\{B\}) \to A\}$

- Belátható, hogy a 3. lépés nem rontja el a halmaz minimalizálást, így minimális bázist kapunk.

Normálformák (3NF)

- **Az algoritmusban különböző sorrendben választva a függőségeket, illetve attribútumokat, különböző minimális bázist kaphatunk.**

- **F={A→B, B→A, B→C, A→C, C→A}**
  (F-{B→A})*=F*, mivel F-{B→A} |— B→A
  F:=F-{B→A}
  (F-{A→C})*=F*, mivel F-{A→C} |— A→C
  F:=F-{A→C}= {A→B,B→C,C→A} minimális bázis, mert több függőség és attribútum már nem hagyható el.

- **F={A→B, B→A, B→C, A→C, C→A}**
  (F-{B→C})*=F*, mivel F-{B→C} |— B→C
  F:=F-{B→C}={A→B,B→A,A→C,C→A} is minimális bázis, mert több függőség és attribútum már nem hagyható el.

Normálformák (3NF)

- **Az algoritmusban különböző sorrendben választva a függőségeket, illetve attribútumokat, különböző minimális bázist kaphatunk.**

- **F={AB→C, A→B, B→A}**

  (F-{AB→C}∪{A→C})*=F*, mivel

  (F-{AB→C})∪{A→C} |— AB→C és F |— A→C.

  F:=(F-{AB→C}∪{A→C})= {A→C,A→B,B→A}
  minimális bázis, mert több függőség és attribútum már nem hagyható el.

- **F={AB→C, A→B, B→A}**

  (F-{AB→C}∪{B→C})*=F*, mivel

  (F-{AB→C})∪{B→C} |— AB→C és F |— B→C.

  F:=(F-{AB→C}∪{B→C})= {B→C,A→B,B→A} is minimális bázis, mert több függőség és attribútum már nem hagyható el.

Normálformák (3NF)

- **Algoritmus függőségőrző 3NF dekompozíció előállítására:**
- **Input: (R,F)**
  - **Legyen G:={X→A,X→B,...,Y→C,Y→D,....} az F minimális bázisa.**
  - **Legyen S az R sémának G-ben nem szereplő attribútumai.**
  - **Ha van olyan függőség G-ben, amely R összes attribútumát tartalmazza, akkor legyen d:={R}, különben legyen**

    d:={S,XA, XB,...,YC,YD,...}.

Normálformák (3NF)

- **Algoritmus függőségőrző és veszteségmentes 3NF dekompozíció előállítására:**
- **Input: (R,F)**
  - **Legyen G:={X→A,X→B,...,Y→C,Y→D,....} az F minimális bázisa.**
  - **Legyen S az R sémának G-ben nem szereplő attribútumai.**
  - **Ha van olyan függőség G-ben, amely R összes attribútumát tartalmazza, akkor legyen d:={R}, különben legyen K az R egy kulcsa, és legyen**

d:={K,S,XA, XB,...,YC,YD,...}.

Normálformák (3NF)

- **Algoritmus függőségőrző és veszteségmentes 3NF redukált (kevesebb tagból álló) dekompozíció előállítására:**
- **Input: (R,F)**
  - **Legyen G:={X→A,X→B,...,Y→C,Y→D,....} az F minimális bázisa.**
  - **Legyen S az R sémának G-ben nem szereplő attribútumai.**
  - **Ha van olyan függőség G-ben, amely R összes attribútumát tartalmazza, akkor legyen d:={R}, különben legyen K az R egy kulcsa, és legyen**

    d:={K,S,XAB...,...,YCD...,...}.
  - **Ha K része valamelyik sémának, akkor K-t elhagyhatjuk.**