Adatbázisok mintazh megoldás

1. A főállományban 3×10^6 rekord van. Mivel rekordok nem lóghatnak át laphatáron, ezért a főállományhoz kell legalább 10^6 lap. Az első ritka indexben lesz ennyi bejegyzés, azaz 10^6 és mivel egy lapra pontosan 20 bejegyzés fér: ez legalább 5×10^4 lap. Ez azt is jelenti, hogy a második szintű ritka indexben lesz 5×10^4 bejegyzés, ehhez kell még 25×10^2 lap. Ez összesen 1052500 lap.

2. (a) SQL-ben:

```
SELECT Látogat.személy
FROM Látogat, Kedvel, Kapható
WHERE Látogat.személy = Kedvel.személy AND
          Látogat.söröző = Kapható.söröző AND
          Kedvel.sör = Kapható.sör
```

   (b) oszlopkalkulussal:

   $$\{e \mid \exists k \ \ \exists s \ [\ \mathit{Látogat}(e,k) \wedge \mathit{Kapható}(k,s) \wedge \mathit{Kedvel}(e,s)]\}$$

3. (a) Ez igaz. Bizonyítás: Azt kell megmutatni, hogy $(r \cap r') \bowtie s \subseteq (r \bowtie s) \cap (r' \bowtie s)$ és $(r \bowtie s) \cap (r' \bowtie s) \subseteq (r \cap r') \bowtie s$ is fenáll.

   Az első tartalmazás igazolására megmutatjuk, hogy ha egy t sorra $t \in (r \cap r') \bowtie s$ fennáll, akkor $t \in (r \bowtie s) \cap (r' \bowtie s)$ is igaz.
   Mivel $t \in (r \cap r') \bowtie s$ igaz, ezért léteznie kell olyan $u \in r \cap r'$ és $v \in s$ soroknak, melyeknek természetes illesztéséből t előáll. De ekkor $u \in r$ és $u \in r'$ miatt $t \in r \bowtie s$ és $t \in r' \bowtie s$ is igaz, így $t \in (r \bowtie s) \cap (r' \bowtie s)$ fennáll.

   A másik irány: ha $t \in (r \bowtie s) \cap (r' \bowtie s)$ igaz, akkor léteznek olyan $r_1 \in r$, $r_2 \in r'$, $s_1 \in s$ és $s_2 \in s$ sorok, hogy t előáll r_1 és s_1, illetve r_2 és s_2 illesztésével is. Mivel t r-re és s-re eső vetülete is egyértelmű, ezért $r_1 = r_2$ és $s_1 = s_2$, így $r_1 = r_2 \in r \cap r'$ és $s_1 = s_2 \in s$ miatt $t \in (r \cap r') \bowtie s$ is igaz.

   (b) Ez is igaz, bizonyítása hasonlóan, mint (a)-nál.

   Ha $t \in (r \cup r') \bowtie s$ igaz, akkor léteznek olyan $u \in r \cup r'$ és $v \in s$ sorok, melyeknek természetes illesztéséből t előáll. De ekkor vagy $u \in r$ vagy $u \in r'$ teljesül, ezért vagy $t \in r \bowtie s$ vagy $t \in r' \bowtie s$ igaz lesz„ így $t \in (r \bowtie s) \cup (r' \bowtie s)$ fennáll.

   A másik irány: ha $t \in (r \bowtie s) \cup (r' \bowtie s)$ igaz, akkor vagy léteznek olyan $r_1 \in r$ és $s_1 \in s$ sorok, amiknek illesztésével t előáll, vagy pedig léteznek olyan $r_2 \in r'$ és $s_2 \in s$ sorok, amiknek illesztésével t előáll. Mivel $r_1 \in r \cup r'$ és $r_2 \in r \cup r'$ is fennáll az így kapott r_1 illetve r_2 sorokra, ezért $t \in (r \cup r') \bowtie s$ igaz lesz.

   (c) Ez nem lesz igaz, egy ellenpélda:

   Ha r és s az alábbi két reláció:

A	B
a	b
a’	b’

B	C
b	c

akkor $\{a, b\} = \Pi_{AB}(r \bowtie s) \neq r = \{\{a, b\}, \{a', b'\}\}$.

4. (néhány megoldás a sok lehetséges közül, természetesen egy zh megoldásban nem kell ezeket mindet leírni):

```
interface Csapat (key név){
    attribute string név;
    attribute string szín;                          \\ (1)
    relationship Set<Játékos> játékosai;
        inverse Játékos: csapata;
    relationship Játékos kapitánya;                 \\ (2)
        inverse Játékos: kapitányItt;               \\
    relationship Set<Múlt> játékosaVolt;
        inverse Múlt: csapatban;
    relationship Set<Szurkoló> szurkolói;
        inverse Szurkoló: kedvencCsapat;
};


interface Játékos (key név){                        \\ nincs két azonos nevű játékos
    attribute string név;                           \\ (*)
    relationship Csapat csapata;
        inverse Csapat:játékosai;
    relationship Csapat kapitányItt;                \\ (8) nullérték megengedett
        inverse Csapat:kapitánya;                   \\
    relationship Set <Szurkoló> nekikKedvence;
        inverse Szurkoló : kedvencJátékosa;
    relationship Set < Múlt > játszottItt;
    inverse Múlt: játékos;
};


interface Szurkoló (key név){                       \\ nincs két azonos nevű szurkoló
    attribute string név;                           \\ (*)
    relationship Csapat kedvencCsapat;              \\ (3) csak egy kedvenc csapat van
        inverse Csapat:szurkolói;
    relationship Játékos kedvencJátékosa;           \\ (4) csak egy kedvenc játékos van
        inverse Játékos:nekikKedvence;              \\
};


interface Múlt(key (kezdete, név)){                 \\ (7)
    attribute int kezdete;                          \\ (5)
    attribute int vége;                             \\ (6)
    relationship Játékos játékos;
        inverse Játékos: játszottItt;
    relationship Csapat csapatban ;
        inverse Csapat: játékosaVolt;
};
```

Megjegyzések:

1. Hogy ne kelljen a (*)-gal jelölt sorokat mindig leírni: felvehetünk egy Személy objektumot, aminek alosztálya lesz a Játékos és a Szurkoló osztály is.

```
interface Személy (key név){                    \\ nincs két azonos nevű személy
    attribute string név;
};
```

A Játékos és a Szurkoló osztály leírását módosítjuk: kimarad a (*)-os sor és:
interface Játékos::Személy{....
illetve
interface Szurkoló::Személy{...
lesz a leírás eleje.

2. A fenti leírás egy csapatnak egy tetszőleges színt enged meg. ((1) sor a Csapat leírásában.) Más lehetőségek (1) helyett:

attribute enum Színek{kék, zöld, piros, ...} színe;
ekkor előre megadjuk a lehetséges színeket

attribute List <enum Színek{kék, zöld, piros, ...}> színei;
ekkor is előre megadjuk a lehetséges színeket, de lehet több szín is

3. Ha nem akarunk null-értéket megengedni (2)-nál:
Lesz egy új osztály, a Kapitány, ami alosztálya lesz a Játékosnak:

```
interface Kapitány::Játékos {
    relationship Csapat kapitányItt;
          inverse Csapat:kapitánya;
};
```

A Csapat laírásánál (2) helyett

```
relationship Kapitány kapitánya;
    inverse Kapitány: kapitányItt;
```

áll és a Játékos leírásából a csapatkapitányságra vonatkozó két sort ((8)) kitöröljük.

4. Ha (3)-ban és (4)-ban Csapat és Játékos helyett Set < Csapat>-ot és Set < Játékos>-t írunk, akkor egy embernek több kedvenc csapatot és játekost is megengedünk.

5. A Múlt-beli szerződések kezdetének és végének pontosabb megadása:
(5) és (6) helyett írhatnánk az

attribute Struct Dátum{int, év, int hónap, int nap} kezdete;

illetve az

attribute Struct Dátum{int, év, int hónap, int nap} vége;
sorokat.

6. Ha meg akarjuk engedni, hogy azonos nevű játékosok vagy azonos nevű szurkolók is legyenek, akkor újabb attribútumot vehetünk fel, pl. lakcím. Ekkor a kulcs a (név, cím) pár lesz.

7. (7)-nél feltettük, hogy a szerződés kezdete és a név azonosít, ez természetes feltevés: egy játékos nem lehet egyszerre két csapattal szerződésben.

5.

(1)	AB	$\rightarrow$	E	ez az 1. szabály
(2)	E	$\rightarrow$	G	ez a 4. szabály
(3)	AB	$\rightarrow$	G	(1) és (2)-ből tranzitivitással
(4)	AB	$\rightarrow$	EB	(1)-ből kiegészítéssel
(5)	EB	$\rightarrow$	I	ez a 3. szabály
(6)	AB	$\rightarrow$	I	(4) és (5)-ből tranzitivitással
(7)	AB	$\rightarrow$	ABI	(6)-ból kiegészítéssel
(8)	ABI	$\rightarrow$	GI	(3)-ból kiegészítéssel
(9)	AB	$\rightarrow$	GI	(7) és (8)-ból tranzitivitással
(10)	GI	$\rightarrow$	H	ez az 5. szabály
(11)	AB	$\rightarrow$	H	(9) és (10)-ből tranzitivitással
(12)	AB	$\rightarrow$	ABH	(11)-ből kiegészítéssel
(13)	ABH	$\rightarrow$	GH	(3)-ból kiegészítéssel
(14)	AB	$\rightarrow$	GH	(12) és (13)-ból tranzitivitással

6. (a) Ez nem igaz, ellenpélda: $R(A, B, C)$ sémában, ha csak $AB \rightarrow C$ függés áll fenn, akkor A zárt, de AB nem.

(b) Ez igaz, bizonyítása:

Tegyük fel indirekt, hogy X és Y zárt, de $X \cap Y$ nem az.

Ekkor létezik olyan $T \rightarrow U$ függés F-ben, hogy $T \subseteq X \cap Y$, de $U \notin X \cap Y$, mivel különben az $(X \cap Y)^+(F)$-et számoló lezárási algoritmus el sem tudna indulni és így $(X \cap Y)^+(F) = X \cap Y$ lenne, ami ellentmond annak, hogy $X \cap Y$ nem zárt.

Mivel $U \notin X \cap Y$, ezért vagy $U \notin X$, vagy $U \notin Y$. Tegyük fel, hogy $U \notin X$ (a másik esetben hasonlóan jutunk ellentmondásra). Mivel $T \subseteq X \cap Y \subseteq X$, ezért $X \rightarrow T$ fennáll, ezt kombinálva a $T \rightarrow U$ függéssel $X \rightarrow U$-t kapjuk, ami (mivel $U \notin X$) ellentmond X zártságának.