

# Adatbázisok 1.

## Vizsgainformációk

# Információk az írásbeli vizsgáról

- Vizsgaalkalmak canvas felületen keresztül, de **személyes jelenléttel**:
  - Időpontok: **2022. május 24-től június 28-ig** minden **kedden 8:30-10:30-ig**
  - Utó- és javítóvizsga-alkalom is egyben az utolsó:  
**2021. június 28. (kedd) 8:30-10:30-ig**
  - **A vizsga kezdete minden esetben 8:30!**
  - **Az Adatbázis labor (00-807) és a Nyelvi labor (00-803) helyszíneken**
  - **Kérem, hogy próbáljanak már a vizsga előtt 15-20 perccel bejelentkezni a canvasba (technikai problémák elkerülése miatt)!**
- Időkorlát: **100** perc (személyes jelenléttel)

## Információk az írásbeli vizsgáról

- A canvasban a vizsga a következőképpen épül majd fel (személyes jelenléttel):

1. Beugró kvíz – **50** perc, **50** pont, amelyből min. **30** pontot el kell érni!
  2. Megoldandó gyakorlati feladatok
  3. Elméleti kérdés
- } **50** perc, **35** pont

- A kérdéssor befejezése és beadása után már nincs lehetőség módosításra!
- A vizsgadolgozat megírását **ÖNÁLLÓAN** kell elvégezni!
- Nem szabad másolni más valaki megoldását, nem szabad külön csatornán a megoldásokat megbeszélni stb.
- A feladatokat és a megoldásokat nem szabad közzé tenni semmilyen formában se (email, facebook, fórumok stb.)!
- Függetlenül attól, hogy ki adta le korábban a megoldást, egyértelmű másolás esetén az összes abban résztvevőnek elégtelen lesz a vizsgája!

# Beugró kvíz

- Egyszerűbb kérdések feleletválasztós, többszörös választás, több lenyíló, igaz/hamis, párosítás, numerikus válasz stb. formában
- A lényeg, hogy olyan kérdések lesznek, amelyeket a rendszer **automatikusan** fog javítani
- **37** kérdés, amelyből **24** kérdés **1** pontos, **13** kérdés **2** pontos
- Összesen **50** pontot lehet szerezni
- Ha valaki **30** pontnál kevesebbet ér el ebből, akkor sikertelen a vizsga
- Időkorlát: **50** perc (személyes jelenléttel)

## Folytatás (vizsga második része)

- Akinek a beugrója sikeres volt (min. 30 pontot ért el), annak folytatódik a vizsga, de önmagában 30 pont még nem elegendő
- Várhatóan esszékérdések formában
- Lesznek gyakorlati feladatok, **10-10** pont:
  - előadás során látott feladatokhoz hasonlóak lesznek
  - további részletekről alább
- Lesznek elméleti kérdések, **15** pont:
  - Közepesen hosszú válaszokat várva, amelyek az előadáson elhangzott és diasorokon szereplő tananyag alapján, annak mélyebb megértése által készíthető el
- Összesen **35** pontot lehet szerezni
- Időkorlát összesen **50** perc (személyes jelenléttel)

## Korábbi videók

- A lejátszási lista linkje:

[https://youtube.com/playlist?list=PLcYvuyQskS84zxRtNcWcd\\_1Wr6N\\_csaGB](https://youtube.com/playlist?list=PLcYvuyQskS84zxRtNcWcd_1Wr6N_csaGB)

- Ezekből a legtöbb szükséges, nyilván a bevezető részből a tananyagot érintő részeit kell figyelembe venni
- Tartalom szerint nagyjából hasonló információkat mondtam el ebben a félévben is

# Osztályzás

## Ponthatárok:

Szükséges a beugrón min. **30** pontot elérni, ha ez teljesül, akkor az összes szerezhető **85** pont alapján:

- **[0%, 40%)** (0-33.99 pont) : elégtelen (1)
- **[40%, 55%)** (34-46.74 pont) : elégséges (2)
- **[55%, 70%)** (46.75-59.49 pont): közepes (3)
- **[70%, 85%)** (59.5-72.24 pont): jó (4)
- **[85%, 100%]** (72.25-85 pont): jeles (5)

# Információk az írásbeli vizsgáról

- Vizsgatematika:
  - A relációs adatmodell
  - A relációs algebrai kifejezések optimalizációja
  - SQL
  - Megszorítások
  - Tranzakciók, nézetek, indexek
  - Jogosultságok
  - Relációs adatbázisok tervezésének elmélete
  - Többértékű függőségek
  - Egyed-kapcsolat modell
  - Objektum-relációs ismeretek
  - XML, DTD, XML séma, XPath

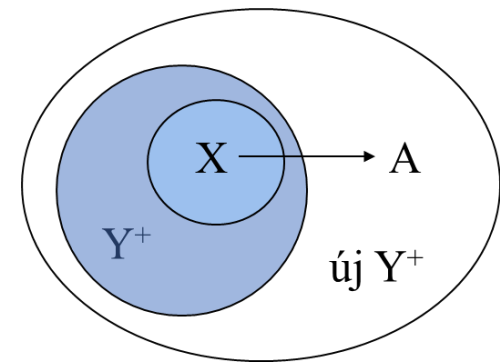


Gyakorlati feladattípusokhoz kapcsolódó fogalmak, példák

# Armstrong-axiómákkal levezetés

- Armstrong-axiómák:
  - (A1) **Reflexivitás**: ha  $Y \subseteq X \subseteq R$ , akkor  $X \rightarrow Y$ . Az ilyen függőségeket **triviális** függőségeknek nevezzük.
  - (A2) **Bővítés**: ha  $X \rightarrow Y$  teljesül, akkor tetszőleges  $Z \subseteq R$ -ra  $XZ \rightarrow YZ$  teljesül.
  - (A3) **Tranzitivitás**: ha  $X \rightarrow Y$  és  $Y \rightarrow Z$ , akkor  $X \rightarrow Z$ .
- Legyen  $R = ABCD$  és  $F = \{ A \rightarrow C, B \rightarrow D \}$ . Bizonyítsuk be levezetéssel, hogy  $AB \rightarrow ABCD$ !
  1.  $A \rightarrow C$  adott.
  2.  $AB \rightarrow ABC$  (A2) alapján.
  3.  $B \rightarrow D$  adott.
  4.  $ABC \rightarrow ABCD$  (A2) alapján.
  5.  $AB \rightarrow ABCD$  (A3) alapján 2-ből és 4-ből.

# Lezárási algoritmus – 1



- Adott  $R$  reláció és  $F$  FF halmaza mellett,  $Y$  *lezártja*: jelölésben  $Y^+$  az összes olyan  $A$  attribútum halmaza, amire  $Y \rightarrow A$  következik  $F$ -ből.
- $Y^+$ -nak kiszámítására egy lezárási algoritmus:
  - **Kiindulás**:  $Y^+ = Y$ .
  - **Indukció**: Olyan FF-ket keresünk, melyeknek a baloldala már benne van  $Y^+$ -ban. Ha  $X \rightarrow A$  ilyen,  $A$ -t hozzáadjuk  $Y^+$ -hoz.
  - Ha  $Y^+$ -hoz már nem lehet további attribútumot adni  $\rightarrow$  vége.
- Legyen a Hallgatók (neptun-kód, név, jegyek, hely)
- $F = \{\text{neptun-kód} \rightarrow \text{név}, \text{neptun-kód} \rightarrow \text{jegyek}, \text{név} \rightarrow \text{jegyek}, \text{név} \rightarrow \text{hely}\}$
- Mi lesz a neptun-kód<sup>+</sup>?

## Lezárási algoritmus – 2

- Kiindulás:  $\text{neptun-kód}^+ = \{\text{neptun-kód}\}$
- Olyan FF-t keresünk, melyeknek a baloldala már benne van  $\text{neptun-kód}^+$ -ban:
- $\text{neptun-kód} \rightarrow \text{jegyek}$  épp egy ilyen
- A **jegyek-et hozzáadjuk**  $\Rightarrow \text{neptun-kód}^+ = \{\text{neptun-kód}, \text{jegyek}\}$
- Másik ilyen:
- $\text{neptun-kód} \rightarrow \text{név} \Rightarrow \text{neptun-kód}^+ = \{\text{neptun-kód}, \text{jegyek}, \text{név}\}$
- Most már a név is benne van, tehát a  $\text{név} \rightarrow \text{hely}$  FF-et is megtaláljuk
- Végül:  $\text{neptun-kód}^+ = \{\text{neptun-kód}, \text{jegyek}, \text{név}, \text{hely}\}$
- Mj.:  $\text{neptun-kód}^+$  az összes attribútuma Hallgatók-nak  $\Leftrightarrow \text{neptun-kód}$  szuperkulcsa Hallgatók-nak

# Exponenciális algoritmus – 1

- Az algoritmus:

1. Minden  $X$  attribútumhalmazra számítsuk ki  $X^+$ -t.
2. Adjuk hozzá a függőségeinkhez  $X \rightarrow A$ -t minden  $A$ -ra  $X^+ - X$ -ből.
3. Dobjuk ki  $XY \rightarrow A$ -t, ha  $X \rightarrow A$  is teljesül.
  - Mert  $XY \rightarrow A$  az  $X \rightarrow A$ -ból minden esetben következik.
4. Végül csak azokat az FF-eket használjuk, amelyekben csak a projektált attribútumok szerepelnek.

- Néhány trükk:

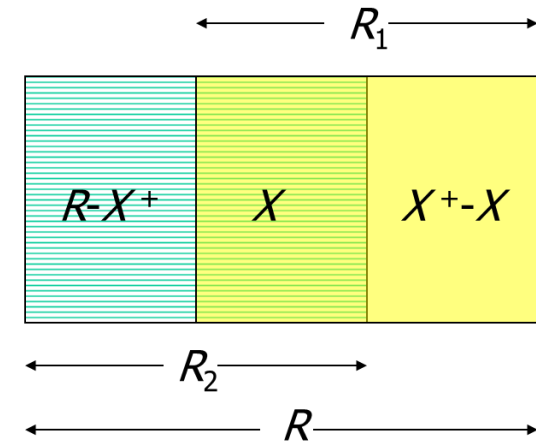
- Az üreshalmaznak és az összes attribútum halmazának nem kell kiszámolni a lezártját.
- Ha  $X^+ =$  az összes attribútum, akkor egyetlen  $X$ -t tartalmazó halmaznak sem kell kiszámítani a lezártját.

## Exponenciális algoritmus – 2

- $ABC$ ,  $A \rightarrow B$  és  $B \rightarrow C$  FF-kel. Projektáljunk  $AC$ -re.
  - $A^+ = ABC$ ; ebből  $A \rightarrow B$ ,  $A \rightarrow C$ .
    - Nem kell kiszámítani  $AB^+$  és  $AC^+$  lezárásokat.
  - $B^+ = BC$ ; ebből  $B \rightarrow C$ .
  - $C^+ = C$ ; semmit nem ad.
  - $BC^+ = BC$ ; semmit nem ad.
- A kapott FF-ek:  $A \rightarrow B$ ,  $A \rightarrow C$  és  $B \rightarrow C$ .
- $AC$ -re projekció:  $A \rightarrow C$ .

## BCNF-re való felbontás – 1

- $R$  reláció **BCNF** normálformában van, ha minden  $X \rightarrow Y$  nemtriviális FF-re  $R$ -ben  $X$  superkulcs.
  - **Nemtriviális**:  $Y$  nem része  $X$ -nek.
  - **Szuperkulcs**: tartalmaz kulcsot (ő maga is lehet kulcs).
- Adott  $R$  reláció és  $F$  funkcionális függőségek.
- Van-e olyan  $X \rightarrow Y$  FF, ami sérti a BCNF-t?
- Kiszámítjuk  $X^+$ -t:
  - Ha itt nem szerepel az összes attribútum,  $X$  nem superkulcs.
- Ha ilyet találtunk, akkor:
  - $R$ -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.



## BCNF-re való felbontás – 2

- Példa

Főnökök(név, cím, kedveltSörök, gyártó, kedvencSör)

$F = \text{név} \rightarrow \text{cím}, \text{név} \rightarrow \text{kedvencSör}, \text{kedveltSörök} \rightarrow \text{gyártó}$

- Vegyük  $\text{név} \rightarrow \text{cím}$  FF-t:
- $\{\text{név}\}^+ = \{\text{név}, \text{cím}, \text{kedvencSör}\}$ .
- A dekomponált relációsémák:
  1. Főnökök1(név, cím, kedvencSör)
  2. Főnökök2(név, kedveltSörök, gyártó)



## BCNF-re való felbontás – 3

- Meg kell néznünk, hogy az Főnökök1 és Főnökök2 táblák BCNF-ben vannak-e.
- Az FF-ek projektálása könnyű.
- A Főnökök1(név, cím, kedvencSör), az FF-ek név->cím és név->kedvencSör.
  - Tehát az egyetlen kulcs: {név}, azaz az Főnökök1 BCNF-ben van.
- A Főnökök2(név, kedveltSörök, gyártó) esetén az egyetlen FF: kedveltSörök->gyártó, az egyetlen kulcs: {név, kedveltSörök}.
  - Sérül a BCNF.
- $\text{kedveltSörök}^+ = \{\text{kedveltSörök}, \text{gyártó}\}$ , az Főnökök2 felbontása:
  1. Főnökök3(kedveltSörök, gyártó)
  2. Főnökök4(név, kedveltSörök)

## BCNF-re való felbontás – 4

- Az *Főnökök* dekompozíciója tehát:
  1. *Főnökök1*(név, cím, kedvencSör)
  2. *Főnökök3*(kedveltSörök, gyártó)
  3. *Főnökök4*(név, kedveltSörök)
- Az *Főnökök1* az főnökökről, az *Főnökök3* a sörökről, az *Főnökök4* az főnökökről és kedvelt söreikről tartalmaz információt.

# Chase-teszt veszteségmentesség ellenőrzéséhez – 1

- Ha  $r = \Pi_{R_1}(r) \mid X \mid \dots \mid X \mid \Pi_{R_k}(r)$  teljesül, akkor az előbbi összekapcsolásra azt mondjuk, hogy **veszteségmentes**. Itt  $r$  egy  $R$  sémájú relációt jelöl.
- $\Pi_{R_i}(r)$  jelentése:  $r$  sorai az  $R_i$  attribútumaira projektálva.
- Igaz, hogy  $r \subseteq \Pi_{R_1}(r) \mid X \mid \dots \mid X \mid \Pi_{R_k}(r)$  mindig teljesül.
- Chase-teszt: a fordított irány teljesül-e?
- Példa: adott  $R(A, B, C, D)$ ,  $F = \{ A \rightarrow B, B \rightarrow C, CD \rightarrow A \}$  és az  $R_1(A, D)$ ,  $R_2(A, C)$ ,  $R_3(B, C, D)$  felbontás. Kérdés veszteségmentes-e a felbontás?
- Vegyük  $R_1 \mid X \mid R_2 \mid X \mid R_3$  egy  $t = (a, b, c, d)$  sorát. Bizonyítani kell, hogy  $t$   $R$  egy sora. A következő tablót készítjük el:

A	B	C	D
a	$b_1$	$c_1$	d
a	$b_2$	c	$d_2$
$a_3$	b	c	d

## Chase-teszt veszteségmentesség ellenőrzéséhez – 2

- Az F-beli függőségeket használva egyenlővé tesszük azokat a szimbólumokat, amelyeknek ugyanazoknak kell lennie, hogy valamelyik függőség ne sérüljön.
  - Ha a két egyenlővé teendő szimbólum közül az egyik index nélküli, akkor a másik is ezt az értéket kapja.
  - Két indexes szimbólum esetén a kisebbik indexű értéket kapja meg a másik.
  - A szimbólumok minden előfordulását helyettesíteni kell az új értékkel.
- Az algoritmus véget ér, ha valamelyik sor  $t$ -vel lesz egyenlő, vagy több szimbólumot már nem tudunk egyenlővé tenni.

# Chase-teszt veszteségmentesség ellenőrzéséhez – 3

A	B	C	D
a	b <sub>1</sub>	c <sub>1</sub>	d
a	b <sub>2</sub>	c	d <sub>2</sub>
a <sub>3</sub>	b	c	d

$A \rightarrow B$



A	B	C	D
a	b <sub>1</sub>	c <sub>1</sub>	d
a	b <sub>1</sub>	c	d <sub>2</sub>
a <sub>3</sub>	b	c	d

$B \rightarrow C$



A	B	C	D
a	b <sub>1</sub>	c	d
a	b <sub>1</sub>	c	d <sub>2</sub>
a <sub>3</sub>	b	c	d

$CD \rightarrow A$




A	B	C	D
a	b <sub>1</sub>	c	d
a	b <sub>1</sub>	c	d <sub>2</sub>
a	b	c	d

## Chase-teszt veszteségmentesség ellenőrzéséhez – 4

- Ha nem kapjuk meg  $t$ -t, akkor viszont a felbontás nem veszteségmentes.
- Példa:  $R(A, B, C, D)$ ,  $F = \{ B \rightarrow AD \}$ , a felbontás:  $R_1(A, B)$ ,  $R_2(B, C)$ ,  $R_3(C, D)$ .

A	B	C	D
a	b	c <sub>1</sub>	d <sub>1</sub>
a <sub>2</sub>	b	c	d <sub>2</sub>
a <sub>3</sub>	b <sub>3</sub>	c	d

$B \rightarrow AD$



A	B	C	D
a	b	c <sub>1</sub>	d <sub>1</sub>
a	b	c	d <sub>1</sub>
a <sub>3</sub>	b <sub>3</sub>	c	d

Itt az eredmény jó ellenpélda, hiszen az összekapcsolásban szerepel  $t = (a, b, c, d)$ , míg az eredeti relációban nem.

# Egyed-kapcsolat modell – tervezési technikák

1. Redundancia elkerülése.
  2. A gyenge egyedhalmazok óvatos használata.
  3. Ne használjunk egyedhalmazt, ha egy attribútum éppúgy megfelelne a célnak.
- Egy egyedhalmaznak legalább egy feltételnek eleget kell tennie az alábbiak közül:
    - Többnek kell lennie, mint egy egyszerű név, azaz legalább egy nem kulcs attribútumának lennie kell.
- Vagy..
- a „sok” végén szerepel egy sok-egy kapcsolatnak.

# Egyed-kapcsolat modell – diagramok átírása relációsémává

- Egyedhalmaz  $\rightarrow$  reláció.
  - Attribútumok  $\rightarrow$  attribútumok.
- Kapcsolat  $\rightarrow$  relációk, melyeknek az attribútumai csak:
  - az összekapcsolt egyedhalmazok kulcs-attribútumait,
  - és a kapcsolat attribútumait tartalmazzák.
- Egy relációba összevonhatók:
  1. Az  $E$  egyedhalmazból kapott reláció,
  2. valamint azon sok-egy kapcsolatok relációi, melyeknél az  $E$  a „sok” oldalon szerepel.
- Egy gyenge egyedhalmazokból kapott relációnak a teljes kulcsot tartalmaznia kell (a más egyedhalmazokhoz tartozó kulcs-attribútumokat is), valamint a saját, további attribútumokat.
- A támogató kapcsolatot nem írjuk át, redundanciához vezetne.



# Egyed-kapcsolat modell – példák

- Egy részletes példa is szerepelt:
- Orvosi adatbázis (11\_EgyedKapcsolat\_E.pdf)
- Még egyet megnézünk a most következőben

# Egyed/kapcsolat modell példa – könyvesbolt

<https://stackoverflow.com/questions/22284150/how-to-relate-weak-with-weak-entity> alapján

- Könyvesbolt adatbázist készítünk. Minden könyvesboltról számon tartjuk a könyvesbolt egyértelmű azonosítóját és címét.
- A könyvesbolt szekciókat tartalmaz. Ezeknek van egy szekciószáma és egy külső megjelenése. A szám önmagában nem azonosítja a szekciót, mert több könyvesbolthoz is tartozhat ugyanolyan számú szekció.
- Egy szekciót polcokra osztották fel. A polcoknak csak egy száma van, amely alapján még nem tudhatjuk, hogy melyik polcra van szó, mert több szekcióhoz is tartozhat ugyanolyan számú polc.

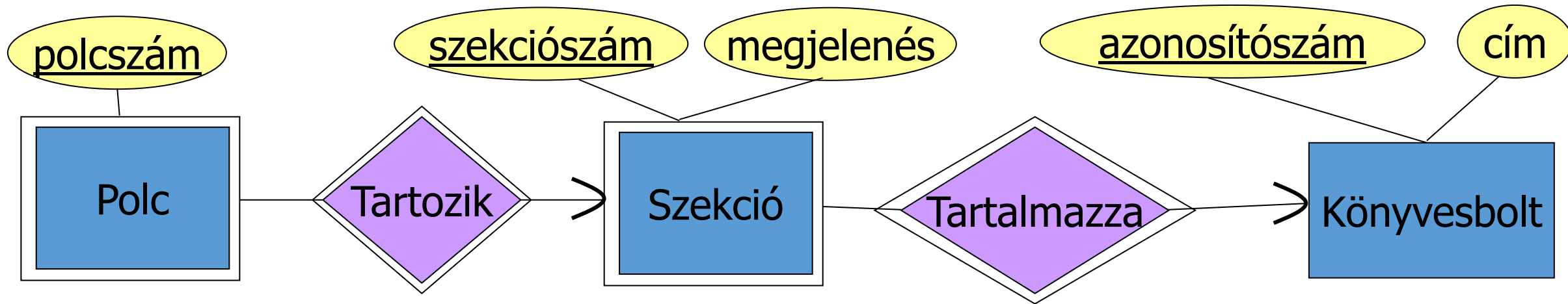
# Egyed/kapcsolat modell példa – könyvesbolt

<https://stackoverflow.com/questions/22284150/how-to-relate-weak-with-weak-entity> alapján

- Könyvesbolt adatbázist készítünk. Minden **könyvesboltról** számon tartjuk a könyvesbolt egyértelmű **azonosítószámát** és **címét**.
- A könyvesbolt **szekciókat** tartalmaz. Ezeknek van egy **szekciószáma** és egy **külső megjelenése**. A szám önmagában nem azonosítja a szekciót, mert több könyvesbolthoz is tartozhat ugyanolyan számú szekció.
- Egy szekciót **polcokra** osztották fel. A polcoknak csak egy **polcszáma** van, amely alapján még nem tudhatjuk, hogy melyik polcról van szó, mert több szekcióhoz is tartozhat ugyanolyan számú polc.

# Egyed/kapcsolat modell példa – könyvesbolt

<https://stackoverflow.com/questions/22284150/how-to-relate-weak-with-weak-entity> alapján



# Egyed/kapcsolat modell példa – könyvesbolt

<https://stackoverflow.com/questions/22284150/how-to-relate-weak-with-weak-entity> alapján

- Adatbázis-séma:
- Könyvesbolt(azonosítószám, cím)
- Szekció(szekció szám, könyvesbolt azonosítószám, megjelenés)
- Polc(polc szám, szekció szám, könyvesbolt azonosítószám)

# Objektum-relációs adatbázisok – 1

- UDT és használati módjai
  - CREATE TYPE, CREATE TABLE <táblanév> OF <típusnév>, CREATE TABLE <táblanév> (... <attribútumnév> <típusnév>, ...)
- A REF, típuskonstruktor, navigáció (mezőelérés), „->”, generátor, mutátor, Deref
- Példa típusok, táblák:

```
CREATE TYPE BarType AS OBJECT (  
    name    CHAR(20),  
    addr    CHAR(20)  
);  
CREATE TYPE BeerType AS OBJECT (  
    name    CHAR(20),  
    manf    CHAR(20)  
);
```

```
CREATE TYPE MenuType AS OBJECT (  
    bar      REF BarType,  
    beer     REF BeerType,  
    price    FLOAT  
);  
CREATE TABLE Bars OF BarType;  
CREATE TABLE Beers OF BeerType;  
CREATE TABLE Sells OF MenuType;
```

# Objektum-relációs adatbázisok – 2

- **Beszúrás az előző táblákba:**

```
INSERT INTO Bars VALUES (BarType('Joe''s bar', 'Maple str 1'));
INSERT INTO Beers VALUES (BeerType('Soproni IPA', 'Soproni sgy.'));
INSERT INTO Sells
    SELECT MenuType(REF(ba), REF(be), 3.5)
    FROM Bars ba, Beers be
    WHERE ba.name = 'Joe''s bar' AND be.name = 'Soproni IPA';
```

- **Lekérdezések:**

```
SELECT Deref(se.bar) AS BAR, Deref(se.beer) AS BEER, price FROM Sells se;
```

BAR	BEER	PRICE
BarType('Joe''s bar', 'Maple str 1')	BeerType('Soproni IPA', 'Soproni sgy.')	3.5

```
SELECT se.bar.name AS BAR_NAME, se.beer.name AS BEER_NAME, price FROM Sells se;
```

BAR_NAME	BEER_NAME	PRICE
Joe's bar	Soproni IPA	3.5

## Objektum-relációs adatbázisok – 3

- Oracle beágyazott táblák:
- Megengedi, hogy a sorok egyes komponensei teljes relációk legyenek.
- Ha  $T$  egy UDT, létrehozhatunk egy  $S$  típust, amelynek az értékei relációk, amelyeknek a sortípusa viszont  $T$ :

```
CREATE TYPE S AS TABLE OF T;
```

- Oracle valójában nem tárolja el a beágyazott relációkat külön relációkként
- Ehelyett, egy  $R$  reláció van, amelyben egy  $A$  attribútumra az összes beágyazott táblázatot és azok összes sorát eltárolja:

```
NESTED TABLE A STORE AS R
```



# Objektum-relációs adatbázisok – 4

- Példa:

```
CREATE TYPE BeerType AS OBJECT (  
    name      CHAR(20),  
    kind      CHAR(10),  
    color     CHAR(10)  
);
```

```
CREATE TYPE BeerTableType AS  
    TABLE OF BeerType;
```

```
CREATE TABLE Manfs (  
    name      CHAR(30),  
    addr      CHAR(50),  
    beers     BeerTableType  
)  
NESTED TABLE beers STORE AS BeerTable;
```

- Beágyazott táblázat ugyanúgy jeleníthető meg, nyomtatható ki mint bármilyen más érték.

# Objektum-relációs adatbázisok – 5

- Egy beágyazott táblát hagyományos relációvá lehet konvertálni a TABLE() alkalmazásával
- Ezt a relációt, ugyanúgy mint bármely másikat, a FROM záradékban lehet alkalmazni.
- Lekérdezés példa:

```
SELECT bb.name
FROM TABLE (
    SELECT beers
    FROM Manfs
    WHERE name = 'Anheuser-Busch'
) bb
WHERE bb.kind = 'ale';
```

# Objektum-relációs adatbázisok – 6

- Bármely reláció megfelelő számú attribútummal és azok illeszkedő adattípusaival egy beágyazott tábla értékei lehetnek.
- Használjuk a `CAST(MULTISET(...)) AS <type>` ) utasítást a reláción azért, hogy a helyes adattípussal rendelkező értékeivel egy beágyazott táblázattá alakítsuk.
- Beszúrás példa:

```
INSERT INTO Manfs VALUES (  
    'Pete''s', 'Palo Alto',  
    CAST(  
        MULTISET(  
            SELECT bb.beer  
            FROM Beers bb  
            WHERE bb.manf = 'Pete''s'  
        ) AS BeerTableType  
    )  
);
```

*Ezekon kívül érdemes megnézni az összes hasonló példát,  
amit előadáson vettünk!*

# DTD, XML séma – 1

- XML dokumentumok, tagek, „jól formáltság” / „validság”
- DTD, ELEMENT, ATTLIST, ID, IDREF, IDREFS
- XML séma, névtér (`xmlns:név="URI"`), `xs:element`, `xs:attribute`, összetett típus: `xs:complexType`, egyszerű típus: `xs:simpleType`, `xs:restriction`

## DTD, XML séma – 2

- **Legyen az alábbi egyszerű DTD példa, adjuk meg ugyanezt XML-sémaként!**

```
<!DOCTYPE kocsmák [  
  <!ELEMENT kocsmák (kocsma*)>  
  <!ELEMENT kocsma (sör+)>  
    <!ATTLIST kocsma név CDATA #REQUIRED>  
  <!ELEMENT sör EMPTY>  
    <!ATTLIST sör név CDATA #REQUIRED>  

```

# DTD, XML séma – 3

- **Ugyanez XML-sémaként:**

```
<? xml version = "1.0" encoding = "utf-8" ?>  
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">  
  <xs:complexType name = "sörTípus">  
    <xs:attribute name = "név"  
      type = "xs:string"  
      use = "required" />  
  </xs:complexType>
```

...

# DTD, XML séma – 4

- **Ugyanez XML-sémaként:**

...

```
<xs:complexType name = "kocsmatípus">
  <xs:sequence>
    <xs:element name = "sör"
      type = "sörTípus"
      minOccurs = "1" maxOccurs = "unbounded" />
  </xs:sequence>
  <xs:attribute name = "név"
    type = "xs:string"
    use = "required" />
</xs:complexType>
```

...

# DTD, XML séma – 5

- **Ugyanez XML-sémaként:**

...

```
<xs:complexType name = "kocsmákTípus" >  
  <xs:sequence>  
    <xs:element name = "kocsmá"  
      type = "kocsmáTípus"  
      minOccurs = "0" maxOccurs = "unbounded" />  
  </xs:sequence>  
</xs:complexType>  
<xs:element name = "kocsmák" type = "kocsmákTípus" />  
</xs:schema>
```



# DTD, XML séma – 6

- **Ugyanez XML-sémaként – alternatív megoldás:**

...

```
<xs:element name = "kocsmák">
  <xs:complexType>
    <xs:sequence>
      <xs:element name = "kocsma"
        type = "kocsmaTípus"
        minOccurs = "0" maxOccurs = "unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```