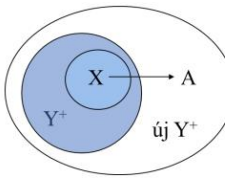
Gyakorlati feladattípusokhoz kapcsolódó fogalmak, példák

Armstrong-axiómákkal levezetés

- Armstrong-axiómák:
  - (A1) Reflexitivitás: 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 → C, B → D }. Bizonyítsuk be levezetéssel, hogy AB → ABCD!
  1. A → C adott.
  2. AB → ABC (A2) alapján.
  3. B → D adott.
  4. ABC → ABCD (A2) alapján.
  5. AB → 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->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 -> A* ilyen, *A*-t hozzáadjuk Y^+-hoz.
  - Ha Y^+-hoz már nem lehet további attribútumot adni → vége.
- Legyen a `Hallgatók(neptun-kód, név, jegyek, hely)`
- F = {`neptun-kód` → `név`, `neptun-kód` → `jegyek`, `név` → `jegyek`, `név` → `hely`}
- Mi lesz a neptun-kód$^+$?

Lezárási algoritmus – 2

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

Exponenciális algoritmus – 1

- Az algoritmus:

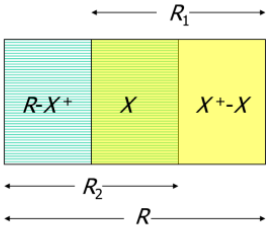
1. Minden X attribútumhalmazra számítsuk ki X^+-t.
2. Adjuk hozzá a függőségeinkhez $X \to A$-t minden A-ra $X^+ - X$-ből.
3. Dobjuk ki $XY \to A$ -t, ha $X \to A$ is teljesül.
   - Mert $XY \to A$ az $X \to A$ -ból minden esetben következik.
4. Végül csak azokat az FF-ket használjuk, amelyekben csak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

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

BCNF-re való felbontás – 1

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



- **Példa**

Főnökök(<u>név</u>, cím, <u>kedveltTeák</u>, gyártó, kedvencTea)

F = név->cím, név->kedvencTea, kedveltTeák->gyártó

- Vegyük név->cím FF-t:
- {név}$^+$ = {név, cím, kedvencTea}.
- A dekomponált relációsémák:
  1. Főnökök1(<u>név</u>, cím, kedvencTea)
  2. Főnökök2(<u>név</u>, <u>kedveltTeák</u>, gyártó)

- Meg kell néznünk, hogy az Főnökök1 és Főnökök2 táblák BCNF-ben vannak-e.
- Az FF-ek projektálása könnyű.
- A Főnökök1(<u>név</u>, cím, kedvencTea), az FF-ek név->cím és név->kedvencTea.
  - Tehát az egyetlen kulcs: {név}, azaz az Főnökök1 BCNF-ben van.
- A Főnökök2(<u>név</u>, <u>kedveltTeák</u>, gyártó) esetén az egyetlen FF:

kedveltTeák->gyártó, az egyetlen kulcs: {név, kedveltTeák}.

  - Sérül a BCNF.
- kedveltTeák$^+$ = {kedveltTeák, gyártó}, az *Főnökök2* felbontása:
  1. Főnökök3(<u>kedveltTeák</u>, gyártó)
  2. Főnökök4(<u>név</u>, <u>kedveltTeák</u>)

- Az *Főnökök* dekompozíciója tehát:
  1. Főnökök1(<u>név</u>, cím, kedvencTea)
  2. Főnökök3(<u>kedveltTeák</u>, gyártó)
  3. Főnökök4(<u>név</u>, <u>kedveltTeák</u>)
- Az *Főnökök1* az főnökökről, az *Főnökök3* a teákról, az *Főnökök4* az főnökökről és kedvelt teáikról tartalmaz információt.

Chase-teszt veszteségmentesség ellenőrzéséhez – 1

- Ha $r = \Pi_{R1}(r) \mid X \mid \ldots \mid X \mid \Pi_{Rk}(r)$ teljesül, akkor az előbbi összekapcsolásra azt mondjuk, hogy veszteségmentes. Itt r egy R sémájú relációt jelöl.
- $\Pi_{Ri}(r)$ jelentése: r sorai az Ri attribútumaira projektálva.
- Igaz, hogy $r \subseteq \Pi_{R1}(r) \mid X \mid \ldots \mid X \mid \Pi_{Rk}(r)$ mindig teljesül.
- Chase-teszt: a fordított irány teljesül-e?
- Példa: adott R(A, B, C, D), F = { A → B, B → C, CD → 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

- 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.

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

A → B

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

B → C

A	B	C	D
a	b_1	c	d
a	b_1	c	d_2
a_3	b	c	d

CD → A

A	B	C	D
a	b_1	c	d
a	b_1	c	d_2
a	b	c	d

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

A	B	C	D
a	b	c_1	d_1
a_2	b	c	d_2
a_3	b_3	c	d

B → AD

A	B	C	D
a	b	c_1	d_1
a	b	c	d_1
a_3	b_3	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

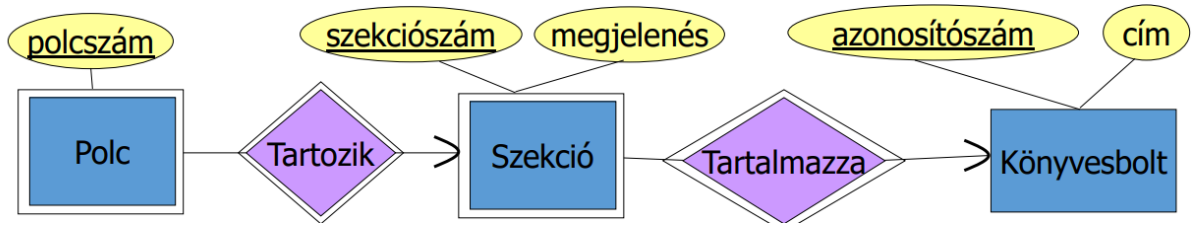
- Egyedhalmaz -> reláció.
  - Attribútumok -> attribútumok.
- Kapcsolat -> relációk, melyeknek az attribútumai csak:
  - az összekapcsolt egyedhalmazok kulcs-attribútumait,
  - és a kapcsolat attribútumait tartalmazzák.
- Egy relációba összevonhatók:
  1. Az *E* egyedhalmazból kapott reláció,
  2. valamint 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 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 száma van, amely alapján még nem tudhatjuk, hogy melyik polcról van szó, mert több szekcióhoz is tartozhat ugyanolyan számú polc.

- 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.



- 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 TeázóTípus AS OBJECT (
       név    CHAR(20),
       cím    CHAR(20)
);
CREATE TYPE TeaTípus AS OBJECT (
       név    CHAR(20),
       gyártó CHAR(20)
);
```

```
CREATE TYPE FelszolgálTípus AS OBJECT (
       teázó  REF TeázóTípus,
       tea    REF TeaTípus,
       ár     FLOAT
);
CREATE TABLE Teázók OF TeázóTípus;
CREATE TABLE Teák OF TeaTípus;
CREATE TABLE Felszolgál OF FelszolgálTípus;
```

- Beszúrás az előző táblákba:

```
INSERT INTO Teázók VALUES (TeázóTípus('Joe''s Teahouse', 'Maple str 1'));
INSERT INTO Teák VALUES (TeaTípus('Pyramid', 'Lipton'));
INSERT INTO Felszolgál
        SELECT FelszolgálTípus(REF(ba), REF(be), 3.5)
        FROM Teázók ba, Teák be
        WHERE ba.név = 'Joe''s Teahouse' AND be.név = 'Pyramid';
```

- Lekérdezések:

```
SELECT DEREF(se.teázó) AS TEÁZÓ, DEREF(se.tea) AS TEA, ár FROM Felszolgál se;
```

TEÁZÓ	TEA	ÁR
TeázókTípus('Joe''s Teahouse', 'Maple str 1')	TeaTípus('Pyramid', 'Lipton')	3.5

```
SELECT se.teázó.név AS TEÁZÓ_NÉV, se.tea.név AS TEA_NÉV, ár FROM Felszolgál se;
```

TEÁZÓ_NÉV	TEA_NÉV	ÁR
Joe's Teahouse	Pyramid	3.5

- 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
```

- Példa:

```
CREATE TYPE TeaTípus AS OBJECT (
  név      CHAR(20),
  fajta    CHAR(10),
  szín     CHAR(10)
);
CREATE TYPE TeaTáblaTípus AS
  TABLE OF TeaTípus;
```

```
CREATE TABLE Gyártók (
              név    CHAR(30),
              cím    CHAR(50),
              teák   TeaTáblaTípus
)
NESTED TABLE teák STORE AS TeaTábla;
```

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

- 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.név
FROM TABLE(
 SELECT teák
 FROM Gyártók
 WHERE név = 'Lipton'
) bb
WHERE bb.fajta = 'herbal';
```

- 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 Gyártók VALUES (
 'Pete''s', 'Palo Alto',
 CAST(
     MULTISET(
          SELECT bb.tea
          FROM Teák bb
          WHERE bb.gyártó = 'Pete''s'

      ) AS TeaTáblaTípus
  )
 );
```

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

- **Legyen az alábbi egyszerű DTD példa:**

```
<!DOCTYPE teázók [
  <!ELEMENT teázók (teázó*)>
  <!ELEMENT teázó (tea+)>
        <!ATTLIST teázó név CDATA #REQUIRED>
  <!ELEMENT tea EMPTY>
        <!ATTLIST tea név CDATA #REQUIRED>
]>
```

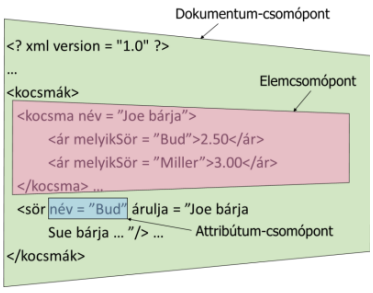
- **Ugyanez XML-sémaként:**

```
<? xml version = "1.0" encoding = "utf-8" ?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">
        <xs:complexType name = "teaTípus">
                <xs:attribute name = "név"
                        type = "xs:string"
                        use = "required" />
        </xs:complexType>
…
…
<xs:complexType name = "teázóTípus">
          <xs:sequence>
                  <xs:element name = "tea"
                   type = "teaTípus"
                   minOccurs = "1" maxOccurs = "unbounded" />
          </xs:sequence>
          <xs:attribute name = "név"
                  type = "xs:string"
                  use = "required" />
</xs:complexType>
…
```

```
...
        <xs:complexType name = "teázókTípus" >
                <xs:sequence>
                        <xs:element name = "teázó"
                                type = "teázóTípus"
                                minOccurs = "0" maxOccurs = "unbounded" />
                </xs:sequence>
        </xs:complexType>
        <xs:element name = "teázók" type = "teázókTípus" />
</xs:schema>
```

```
        <xs:element name = "teázók">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name = "teázó"
                                type = "teázóTípus"
                                minOccurs = "0" maxOccurs = "unbounded" />
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
    </xs:schema>
```

XPath



- XPath
  - általános forma: tételek listája
  - tétel: egyszerű érték, csomópont
  - csomópont: dokumentum-, elem-, attribútum-csomópont
- Utak az XML dokumentumban, útkifejezés, kiértékelés, szűrési feltételek, tengelyek

XQuery

- XQuery
- doc(URL) vagy document(URL), FLWR kifejezések (for, let, where, order by, return), contains, összehasonlítások, „szigorú” összehasonlítások, data, összesítések, összekapcsolások