# ADATBÁZISOK 1.

## 1. A RELÁCIÓS ADATMODELL[1] ALAPJAI: ATTRIBÚTUMOK, SOROK, RELÁCIÓSÉMÁK, ELŐFORDULÁSOK, KULCSOK (22-29. OLDAL)

A relációs modellben az adatok egyszerűen reprezentálhatók: kétdimenziós táblázatban, ún. **relációkban**.
A reláció oszlopait az **attribútumok** látják el névvel, ezek az oszlopok fejrészében találhatók (oszlopcímek), általában megadják az oszlopban szereplő adatok jelentését. [2] Minden attribútumhoz tartozik egy értéktartomány, azaz egy elemi típus. (string, integer…)

A reláció szerkezetét a **relációséma** adja meg, ami a reláció nevének és a reláció attribútumok halmazának együttese (megadása: Név(a,b,c)). Ugyan a reláció attribútumok halmazt alkotnak, mégis fontos a megadás sorrendje! A relációs modellben az adatbázis egy vagy több relációsémát tartalmaz. Az ezekből álló halmazt **(relációs) adatbázissémának** nevezzük.

**Soroknak (tuple)** nevezzük a reláció azon sorait, amelyek különböznek az attribútumokból álló fejléctől. A reláció minden egyes attribútumához tartozik a sorban egy *komponens.*[3] Követelmény, hogy minden sor minden komponense atomi, azaz elemi típusú legyen (pl.: egész vagy karaktersorozat). Nem megengedett pl.: lista, halmaz, rekord, stb.. Valamint minden egyes komponensnek, az attribútumához tartozó értéktartományból kell származnia. Maga a reláció, **sorokból álló halmaz**, ebből kifolyólag a sorok sorrendje lényegtelen; az attribútumok sorrendjét felcserélhetjük, a relációt ez nem változtatja meg (de az oszlopok sorrendjét igen!).

Egy reláció nem állandó, léte során az gyakran változik, új sorok kerülnek bele, sorokat veszünk ki belőle, meglévő sorokat változtatunk meg, stb.. A séma változása kevésbé általános, de előfordulhat. Adott reláció sorainak előfordulását **reláció-előfordulásnak** nevezzük. Azokat a sorokat, amelyek „most" vannak a relációban **aktuális előfordulásnak** nevezzük[4].

Az egyik legalapvetőbb megszorítás, a **kulcs megszorítás.** Attribútumok egy halmaza egy kulcsot alkot egy relációra nézve, ha a reláció előfordulásaiban nincs két olyan sor, amelyek a kulcs összes attribútumának értékein megegyeznének (séma megadásánál ezeket az attribútumokat aláhúzzuk). Gyakran használnak mesterséges kulcsokat, amelyeket direkt úgy állítanak elő, hogy kulcsként működhessenek.

---

[1] Mi az adatmodell?
Információ vagy adatok leírására szolgáló jelölés. Általában három részből áll:
- Az adat struktúrája (fizikai adatmodell)
- Az adatokon végezhető műveletek
- Az adatokra tett megszorítások

[2] **Konvenció:** a relációk nevét nagybetűvel kezdjük, míg az attribútumokét kicsivel.

[3] Amikor külön írjuk le a sorokat, nem pedig egy reláció részeként, akkor vesszőkkel választjuk el őket egymástól és zárójelek közé tesszük. Ilyenkor az attribútumokat nem láthatjuk, így meg kell adnunk valamilyen hivatkozást, hogy melyik relációhoz tartozik az adott sor, és mindig ugyanabban a sorrendben írjuk fel a komponenseket, ahogyan az attribútumokat felsoroltuk a relációsémában.
[4] Azokat az adatbázisokat, amelyek az adat időben korábbi verzióit is nyilvántartják, *temporális adatbázisoknak* nevezzük.

Készítette: Plézer Viktor István

# 2. Relációsémák definiálása SQL[5]-ben (30-35. oldal)

SQL relációk:
1. Táblák – tárolt relációk. Benne vannak az adatbázisban, soraik változtatásával megváltoztathatók, soraik lekérdezhetők.
2. Nézetek – számításokból kapott relációk. Nem tároljuk őket, de részben vagy teljes egészében szerkeszthetők, ha szükséges.
3. Ideiglenes táblák – az SQL nyelvi feldolgozója készíti őket, amikor valamilyen lekérdezéseket, adatmódosításokat végez el. Eldobja, nem tárolja!

Relációk megadásához először is ismernünk kell az adattípusokat:
1. Karaktersorok[6]:
   a. CHAR (n) – rögzített n hosszúságú karaktersor.
   b. VARCHAR (n) – legfeljebb n hosszúságú karaktersor.
2. Bitsorok:
   a. BIT (n) – n hosszúságú bitsor
   b. BIT VARYING (n) – legfeljebb n bitből álló bitsort jelképez.
3. BOOLEAN (TRUE, FALSE, UNKNOWN)
4. INT (EGER), SHORTINT – egész számok
5. Lebegőpontos értékek:
   a. FLOAT, REAL (megegyeznek)
   b. DOUBLE PRECISION (nagyobb pontossággal tárolhatók az értékek)
6. Fixpontos valós számok:
   a. DECIMAL (n, d) – n számjegyből áll, a tizedesponttól jobbra d számú tizedes jegy áll.
7. DATE, TIME

Egy tábla létrehozása: **CREATE TABLE Filmek (filmcím CHAR (100), hossz INT…);**
Egy tábla törlése: **DROP TABLE Filmek;**
Séma módosítása: **ALTER TABLE Filmek ADD műfaj CHAR (20);**
**ALTER TABLE Filmek DROP műfaj;**

Alapértelmezésben a komponensek értéke minden sorban NULL lesz, ha egy új attribútumot vezetünk be, de ezt megváltoztathatjuk:
**CREATE TABLE Emberek (név VARCHAR (100), magasság INT, nem CHAR (1) DEFAULT '?', születésiDátum DATE DEFAULT DATE '0000-00-00');**

**ALTER TABLE Filmek ADD műfaj CHAR (20) DEFAULT 'nem ismert';**

Kulcsjelleg megadása:
- PRIMARY KEY - az attribútumok nem vehetnek fel NULL értéket
- UNIQUE – az attribútumok felvehetnek NULL értéket

Kulcs megadása, kétféleképpen történhet (CREATE TABLE utasításon belül):
1. attribútumlista megadásakor kulcsként adható meg az attribútum
   **CREATE TABLE Színész (név CHAR (30) PRIMARY KEY, cím VARCHAR (255)…);**
2. olyan deklarációban, amely azt fejezi ki, hogy a szóban forgó attribútumhalmaz kulcsot alkot a relációra nézve (ha a kulcs egynél több attribútumból áll, akkor csak ezt a módszert használhatjuk.)
   **CREATE TABLE Filmek (cím CHAR (100), év INT, hossz INT, PRIMARY KEY (cím, év));**

[5] Structured Query Language – Struktúrált lekérdező nyelv, két szempontnak megfelelő része van:
- adatdefiníciós résznyelv (adatbázissémák megadásához)
- adatmanipulációs résznyelv (lekérdezéshez, módosításhoz, stb.)

[6] Különbségük implementációfüggő, általában a CHAR-t kiegészítjük n hosszúságúra, VARCHAR esetén használunk valamiféle lezáró jelet. Ésszerű típuskényszerítés megengedett

# 3. RELÁCIÓS ALGEBRA[7] (39-52. OLDAL)

Adott relációkból új relációkat hoz létre.

Atomi operandusai a következők lehetnek:
- A relációkhoz tartozó változók
- Konstansok, amelyek véges relációt fejeznek ki.

Műveletei:
- 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.
  - **metszet** (R ∩ S): R és S metszete azon elemek halmaza, amelyek az R-ben és az S-ben is benne vannak.
  - **különbség** (R-S): R és S különbsége azon elemek halmaza, amelyek benne vannak R-ben, de nincsenek S-ben. (R-S ≠S-R)
  - **Halmazműveletek alkalmazásakor, figyelni kell a következőkre:**
    1. **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.**
    2. **R és S oszlopait rendezni kell úgy, hogy az attribútumok sorrendje egyforma legyen a két relációban.**
- Műveletek, amelyek a reláció egyes részeit eltávolítják:
  - **kiválasztás**: olyan új relációt hoz létre, amely R sorainak egy részhalmazát tartalmazza. Az eredménybe csak azok a sorok kerülnek, amelyek teljesítenek egy adott, R attribútumaira megfogalmazott C feltételt.
    - $\sigma_C(R)$**.** Az eredmény sémája megegyezik R sémájával.
  - **vetítés**: a régi R relációból egy olyan új reláció hozható létre, amelyik csak R bizonyos oszlopait tartalmazza.
    - $\pi_{A1, A2,\ldots,An}(R)$ kifejezés értéke az a reláció (sémája: {A1,A2,…An} ), amelyik az R relációnak csak A1,A2,…An attribútumokhoz tartozó oszlopait tartalmazza.
- Műveletek a relációk sorainak kombinálására:
  - **Descartes-szorzat (R x S)**: azon párok halmaza, melynek első eleme R tetszőleges eleme, a második pedig S egy eleme. Ha R és S relációk voltak, akkor a szorzat maga is egy reláció. R egy sorának párosítása S egy sorával olyan hosszú sort eredményez, amelyben az alkotó sorok mindegyik komponense megjelenik. R attribútumai megelőzik sorrendben S attribútumait. Sémája R és S sémájának egyesítése. Ha R és S relációknak vannak azonos nevű attribútumai, legalább az egyiknek új nevet kell adni. (A közös attribútum, akkor R.A és S.A lehetségesek.)
  - Különböző összekapcsolási műveletek:
    - **Természetes összekapcsolás:** R és S azon sorait párosítjuk össze, amelyek értékei megegyeznek R és S sémájának összes közös attribútumán. Az összekapcsolt sor megegyezik az r sorral R minden attribútumán és megegyezik s sorral is S minden attribútumán, valamint megegyezik azon attribútumokon, amelyek mindkét sémában szerepelnek. Jelölése **R ⋈ S**
    - **Théta-összekapcsolás: R $\bowtie_C$ S.** Sémája itt is R és S sémájának összesítése.
      - Kiszámításának módszere:

[7] Egy algebra általában műveleteket és atomi operandusokat tartalmaz. Lehetővé teszi kifejezések megfogalmazását, gyakran zárójelek használatával.

1. Kiszámoljuk R és S szorzatát.
2. Kiválasztjuk a szorzatból azokat a sorokat, amelyek eleget tesznek a C feltételnek.

- Átnevezés, ami nem befolyásolja a reláció sorait, de megváltoztatja a reláció sémáját, azaz az attribútumok neveit és/vagy a reláció nevét.
  - $\rho_{S(A1,A2,...An)}(R)$ :
    - eredményreláció neve S
    - sorai megegyeznek R soraival
    - attribútumai: A1, A2,…,An
  - Ha az attribútumok nevét meg akarjuk őrizni, csak a reláció nevét akarjuk megváltoztatni, akkor: $\rho_S(R)$

Néhány összefüggés:

- $R \cap S = R - (R-S)$
- $R \bowtie_C S = \sigma_C(R \times S)$
- $R \bowtie S = \pi_L(\sigma_C(R \times S))$, ahol:
  - C feltétel: R.A1 = S.A1 AND … AND R.An = S.An
  - L attribútumlista, amelyben szerepel R összes attribútuma és emellett S-ből mindazok az attribútumok, amelyek nincsenek benne R sémájában.

## 4. MEGSZORÍTÁSOK: HIVATKOZÁSI ÉPSÉG, KULCSMEGSZORÍTÁS (61-65. OLDAL)

Megszorítások megadása relációs algebra segítségével:

1. Ha R egy relációs algebrai kifejezés, akkor R = Ø egy olyan megszorítás, amelynek jelentése: „R-nek üresnek kell lennie" vagyis „R eredményében egyetlen sor sincs"
2. Ha R és S relációs algebrai kifejezések, akkor $R \subseteq S$ egy olyan megszorítás, melynek jelentése: „R eredményének minden sora benne kell legyen S eredményében". S tartalmazhat R sorain kívül más sorokat is.

**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: $\pi_A(R) - \pi_B(S) = \emptyset$) .

**Kulcsmegszorítás:** ezt a megszorítást relációs algebrában egy implikációval fejezhetjük ki, ha két sor megegyezik a kulcs attribútumon, akkor meg kell egyeznie a többi komponensnek is.

$$\sigma_{R.KULCS = S.KULCS \text{ AND } R.NEMKULCS \neq S.NEMKULCS}(R \times S) = \emptyset$$

**További példák megszorításokra: 64. oldal.**

# 5. SQL 1. – EGYSZERŰ LEKÉRDEZÉSEK, SQL SELECT EGY RELÁCIÓRA, SPECIÁLIS ÉRTÉKEK, HIÁNYZÓ ÉRTÉKEK (258-269.OLDAL)

Az SQL legegyszerűbb lekérdezései azon sorokra vonatkoznak, melyek egy bizonyos relációban eleget tesznek egy adott feltételnek. Egy ilyen lekérdezés a relációs algebra kiválasztási műveletének felel meg, az SQL három alapvető kulcsszavát (SELECT, FROM, WHERE) használja:

- **SELECT**: megadja a lekérdezésre adott válaszban megjelenítendő attribútumokat, speciális eset a *, amikor minden egyes attribútumot (teljes sort) meg kell jeleníteni.
- **FROM:** azon relációt, vagy relációkat adja meg, amely(ek)re a lekérdezés vonatkozik.
- **WHERE:** itt adhatjuk meg a feltételt, a lekérdezés válaszába azon sorok kerülnek be, amelyek kielégítik ezt a feltételt. (igazából ez felel meg a relációs algebra kiválasztás műveletének)

**Vetítés az SQL-ben**: a SELECT záradékban * helyett felsoroljuk a FROM záradékban megadott relációk bármely attribútumát.

Ha az adott attribútumot át akarjuk nevezni, a következőt tegyük (a lekérdezésre kapott válaszban az eredeti oszlopnév helyett az AS után következő fog szerepelni):

**SELECT name AS név, height AS magasság FROM EMBEREK;**

A SELECT záradékba **nem csak attribútumnevek, hanem kifejezések is kerülhetnek:**

**SELECT NAME AS név, height * 100 AS magasságCentiméterben;**[8]

Valamint **konstansok is:**

**SELECT name AS név, height*100 AS magasság, 'cm' AS centiméterben;**

**Kiválasztás SQL-ben:** a WHERE záradék kiterjeszti a relációs algebra kiválasztás operátorát, feltételek követhetik. Használható műveletek:

- = (egyenlő), <> (nem egyenlő), <,>,<=,>=, valamint tetszőleges zárójelezés
- +,-,*,/ például: (év-1930) * (év-1930) <100
- || konkatenáció
- AND, OR, NOT

**Karakterláncok összehasonlítása:** két karakterlánc egyenlő, ha a karaktereknek ugyanabból az egymás után következő sorozatából állnak. Ha különböző típusú (CHAR és VARCHAR) adatokat hasonlítunk össze, mindig csak az aktuális karakterláncok hasonlítódnak össze, a kitöltő karaktereket („**pad**") figyelmen kívül hagyjuk. Aritmetikai összehasonlító operátorokkal a lexikografikus viszonyukat vizsgáljuk. Lehetőség van **mintával való összehasonlításra** is:

**s LIKE p:** ahol s egy karakterlánc, p pedig egy minta. Használható Joker karakterek:

- % - 0 vagy nagyobb hosszúságú sorozatot helyettesít: filmcím LIKE '%''s%' ('s –t tartalmazó filmcímek)
- _ - egy karaktert helyettesít: filmcím LIKE 'Halálos _____'
- néha szükséges lehet, hogy olyan mintára illesszünk, amely tartalmazza %,_ jeleket. Ehhez az ESCAPE kulcsszó használható, amellyel megjelöljük, mi lesz az a karakter, ami semlegesíti ezeket a joker karaktereket.
  - s LIKE 'x%%x%' ESCAPE 'x' (olyan karaktersorozatok keresése, amelyek %-lel kezdődnek és végződnek)

Dátum, idő (szintén összehasonlítható):

- DATE '1988-11-08'
- TIME '15:00:02.5'
- TIMESTAMP '1988-11-08 12:00:00'

[8] Az SQL nem tesz különbséget kis és nagybetű között, csak akkor ha a kifejezés idézőjelek között van: 'FROM' ≠ 'from'

**NULL:** az SQL lehetővé teszi, hogy az attribútum értéke egy speciális nullérték legyen. Gyakran találkozhatunk vele, keletkezhet új sor beszúrásakor, relációk összekapcsolásakor. A következőképpen vizsgálható: „x IS NULL" vagy „x IS NOT NULL".
Értelmezésére több lehetőségünk is van:

- Ismeretlen érték: „Tudom, hogy valamilyen értéknek ott kell lenni, de nem tudom, melyik ez az érték." Például egy nem ismert születésnap.
- Alkalmazhatatlan érték: „Nincs olyan érték, aminek itt értelme lenne."
- Visszatartott érték: „Nem vagyunk feljogosítva rá, hogy ismerjük a megfelelő értéket."

A WHERE záradékban fel kell készülnünk rá, hogy valamely sor éppen felhasznált komponensének értéke NULL is lehet. Két fontos szabályt kell figyelembe vennünk:

- Ha egy aritmetikai műveletben az egyik operandus NULL, a kifejezés eredménye is NULL lesz.
- Ha egy összehasonlító utasításban van NULL, az eredmény ISMERETLEN (UNKNOWN) lesz.

## 6. SQL 2. – TÖBBRELÁCIÓS LEKÉRDEZÉSEK, DIREKT SZORZAT, ÖSSZEKAPCSOLÁS, SORVÁLTOZÓK, UNION, INTERSECT, EXCEPT (273-281.OLDAL)

Több reláció összekapcsolásához fel kell sorolni az összes relációs a FROM záradékban, ezután a SELECT és WHERE záradékok a relációk minden attribútumát felhasználhatják. Például:
**SELECT név FROM Filmek, GyártásIrányító WHERE Filmek.Filmcím = 'Csillagok háborúja' and producerAzon = azonosító;**

Előfordulhatnak olyan lekérdezések, amelyekben két vagy több reláció szerepel és ezekben a relációkban két vagy több attribútumnak ugyanaz a neve. Megkülönböztetésükre használjuk a reláció nevét, utána egy pontot és az attribútumot. Például: Filmszínész.név. Ezt a módszert használhatjuk bármikor, nem kell feltétlenül konfliktus legyen az attribútumnevek között.

Előfordulhat, hogy ugyanarra a relációra többször van szükségünk egy lekérdezésben. A FROM záradékban annyiszor sorolhatjuk fel, ahányszor szükségünk van rá, megkülönböztetésükre használjuk a **sorváltozókat** (AS kulcsszó):
**SELECT Színész1.név, Színész2.név FROM Filmszínész AS Színész1, Filmszínész AS Színész2 WHERE Színész1.cím = Színész2.cim;**

**Halmazműveletek alkalmazásához a lekérdezéseknek ugyanolyan attribútumhalmazúnak – megegyező nevűnek és típusúnak kell lenni!**

**Metszet:**

(SELECT név,cím FROM filmszínész WHERE nem='N')
**INTERSECT**
(SELECT név,cím FROM gyártásirányító WHERE nettóbevétel > 10000)
**(olyan filmszínésznők, akik gyártásirányítok is, keresetük nagyobb mint 10000)**

**Különbség:**

(SELECT név, cím FROM filmszínész)
**EXCEPT**
(SELECT név,cím FROM gyártásirányító)
**(olyan színészek, akik nem gyártásirányítók)**

**Unió:**

(SELECT filmcím,év FROM Filmek)
**UNION**
(SELECT filmCím AS filmcím, filmÉv AS év FROM szerepelBenne)

Készítette: Plézer Viktor István

# 7. SQL 3. – ALKÉRDÉSEK, KORRELÁLT ALKÉRDÉSEK (284-290.OLDAL)

Az olyan lekérdezést, amely egy másik lekérdezés része, **alkérdésnek** nevezzük. Alkérdésnek is lehet alkérdése, és így tovább. Alkérdés kerülhet a WHERE záradékba (ha konstans, vagy reláció az eredménye), vagy a FROM záradékba (ha reláció az eredménye).

Az olyan atomi értékeket, amelyek egy sor egyik komponenseként előfordulhatnak, **skalárnak** nevezzük.

```
                                SELECT név
                                FROM gyártásirányító
                                WHERE azonosító =
/*Innentől alkérdés:*/         (SELECT producerAzon
                                FROM Filmek
                            WHERE filmcím = 'Csillagok háborúja');
```

**Logikai értékeket visszaadó SQL operátorok:**

- **EXIST R** – akkor és csak akkor igaz, ha R reláció nem üres.
- **s IN R** – akkor és csak akkor igaz, ha s egyenlő valamelyik R-beli értékkel. (R **egyoszlopos** reláció)
- **s > ALL R** – akkor és csak akkor, igaz ha s nagyobb mint az R **egyoszlopos** reláció minden értéke. Bármelyik aritmetikai összehasonlító operátor használható.
- **s > ANY R** – akkor és csak akkor igaz, ha s nagyobb az R **egyoszlopos** reláció legalább egy értékénél. Szintén használható bármelyik aritmetikai összehasonlító operátor.

A **NOT** kulcsszót használva az egész kifejezés előtt, tagadhatjuk a kifejezést.

Az SQL-ben egy sor skalárértékek zárójelek közötti felsorolása, komponensei lehetnek konstansok, attribútumok, keverve is. A fent ismertetett logikai értékeket visszaadó SQL operátorok használhatók sorokra is, ilyenkor figyelni kell, hogy a sor attribútumainak sorrendje megegyezzen a reláció attribútumainak sorrendjével. Például:

```
SELECT név
FROM GyártásIrányító
WHERE azonosító IN
(SELECT producerAzon
FROM filmek
WHERE (filmcím,év) IN
(SELECT filmcím, filmév
FROM szerepelBenne
WHERE színész = 'Harrison Ford'));
```

**Harrison Ford filmjeinek gyártásirányítói**

**Korrelált alkérdések:** olyan, beépített alkérdés, mely többször értékelődik ki és minden egyes kiértékelés megfelel egy olyan értékadásnak, amely az alkérdésen kívüli sorváltozóból származatik. Használatuk közben figyelembe kell vennünk a nevek *érvényességi körére* vonatkozó szabályokat (mindig a legbelső lekérdezéshez tartozik, amely megnevezi azt). Például:

```
SELECT filmcím
FROM filmek AS Régi
WHERE év < ANY
(SELECT év
FROM filmek
WHERE filmcím = Régi.filmcím);
```

# 8. A KITERJESZTETT RELÁCIÓS ALGEBRA MŰVELETEI: ISMÉTLŐDÉSEK MEGSZŰNTETÉSE, ÖSSZESÍTÉSEK, CSOPORTOSÍTÁS, RENDEZÉS, MULTIHALMAZ MŰVELETEK (217-236.OLDAL)

A relációs algebrát halmazokon értelmezzük, a kiterjesztett relációs algebrát multihalmazokon fogjuk, azaz ugyanaz a sor többször is megjelenhet egy adott relációban, de azok sorrendje itt sem számít. A relációk multihalmazként való kezelése több módon is gyorsíthatja a relációs műveleteket:

- Ha két multihalmazként értelmezett reláció **unióját vesszük**, egyszerűen lemásoljuk az első reláció sorait, majd mögéjük másoljuk a második relációét, **nem kell törődnünk az ismétlések megszűntetésével.**
- **Vetítéskor** ugyanez, nem kell törődnünk az ismétlődő sorok kiszűrésével.

A multihalmazként kezelt eredmény lehet, hogy nagyobb méretű lesz, de gyorsabb és könnyebb kiszámolni, mert kevesebb műveletet végzünk.

**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 U S – t sor n+m-szer fog előfordulni
- R ∩ S – t sor min(n,m)-szer fog szerepelni
- R - S – t sor max(0,n-m)-szer fog előfordulni

A **kiválasztás** művelete ugyanúgy működik, de itt sem szűrjük ki az ismétlődő sorokat.

Multihalmazok **Descartes-szorzata:** az első reláció minden egyes sorát össze kell párosítanunk a második reláció mindegyik sorával függetlenül attól, hogy az adott sor hányszor szerepel az adott relációban. Következésképpen ha t sor n-szer szerepel R-ben és m-szer S-ben, akkor R x S –ben n*m-szer fog szerepelni.

Multihalmazok **természetes és théta-összekapcsolása** nem különbözik az eddigiektől, csak nem szűrjük az ismétlődő sorokat.

**A kiterjesztett relációs algebra plusz műveletei:**

- δ – az ismétlődések megszűntetésének művelete, a multihalmazt halmazzá alakítja.
  - δ(R) – R relációnak csak a különböző sorait tartalmazza (halmaz)
- Összesítő műveletek:
  - SUM – az oszlop értékeinek összegét határozza meg
  - AVG – az oszlop értékeinek átlagát határozza meg
  - MIN, MAX – az oszlop értékeinek minimumát illetve maximumát határozza meg. Karaktersorozatok esetén a lexikografikailag legkisebbet, legnagyobbat választja.
  - COUNT – az oszlopban található (nem feltétlenül különböző) elemek számát határozza meg
- Csoportosítás - a reláció sorainak „csoportokba" történő beosztása a reláció egy vagy több attribútumának értékétől függően. γ csoportosítási művelet kombinálja a csoportosítást és az összesítést.
  - Az γ alsó indexeként szereplő L elemeknek egy listája, ahol egy elem az alábbiak közül bármelyik lehet:
    - R reláció egy attribútuma, amelyre csoportosítást végzünk. Ezt nevezzük csoportosítási attribútumnak.
    - A reláció valamely attribútumára vonatkozó összesítési művelet. Ha erre az eredményre névvel szeretnénk nyilatkozni, akkor egy nyilat és egy új nevet kell írnunk. A kiindulásul szolgáló attribútumot összesítési attribútumnak nevezzük.
  - Az $\gamma_L(R)$ kifejezés eredményrelációjának felépítése a következő:

        - Osszuk R sorait csoportokba. Egy csoport azokat a sorokat tartalmazza, amelyeknek az L listán szereplő csoportosítási attribútumokhoz tartozó értékei megegyeznek. Ha nincs csoportosítási attribútum, akkor az egész R reláció egy csoportot képez.
        - Minden csoporthoz hozzunk létre olyan sort, amelyik tartalmazza:
            - A szóban forgó csoport csoportosítási attribútumait.
            - Az L lista összesítési attribútumaira vonatkozó összesítéseket.
- Kiterjesztett vetítési művelet: kiterjeszti π hatását azzal, hogy a néhány oszlopra történő vetítés mellett azt is lehetővé teszi, hogy az érintett oszlopok valamilyen összesítési relációja alapján új oszlopok kiszámítását is elvégezhessük.
- τ – rendezési művelet, egy relációt a sorainak egy vagy több attribútumtól függő rendezett listájává alakít. $\tau_L(R)$ (L = A1,A2,…,An) – R-t először A1, majd A2, majd … An szerint rendezi.
- Külső összekapcsolás - az összekapcsolás egyik fajtája, amely a lógó sorokat (nem kapcsolhatók össze más sorokkal, nincs egyezés a közös attribútumokon) is megőrzi. Jele: ⋈ (fölötte karika). Először elvégzi a természetes összekapcsolást, majd hozzáveszi a két reláció lógó sorait, azokat kiegészítve NULL értékekkel minden olyan attribútumon, amelyekkel eddig nem rendelkezett. Bal és jobb oldali külső összekapcsolás, csak a bal vagy jobb oldalon lévő reláció lógó sorait adja hozzá a természetes összekapcsolás eredményéhez (kiegészítve null szimbólumokkal). Van théta-összekapcsolás, sőt, annak is van bal és jobb oldali változata is.

## 9. SQL 4. – ISMÉTLŐDÉSEK KEZELÉSE, CSOPORTOSÍTÓ MŰVELETEK, GROUP BY, HAVING, ORDER BY ZÁRADÉKOK (297-304. OLDAL)

Az SQL lekérdezések eredményeképp létrejött relációk nem halmazok, ismétlődő sorokat tartalmazhatnak. Ezek megszűntetéséhez a SELECT kulcsszó után a **DISTINCT** kulcsszót használjuk.

A halmazműveletek alapesetben megszűntetik ezeket az ismétlődéseket, azaz a multihalmazok halmazokká konvertálódnak. Az ismétlődések megmaradásához az UNION, INTERSECT, EXCEPT kulcsszavak után az **ALL** kulcsszót használjuk, például: R UNION ALL S.

Összesítő függvények:
- SUM, AVG, MIN, MAX, COUNT
    - SELECT AVG(fizetés) FROM dolgozók
- COUNT(*)
    - SELECT COUNT(*) FROM szerepelBenne (összes sorok száma)
- COUNT(DISTINCT x) – x oszlopban található különböző értékek száma.

Csoportosítás:

```
SELECT stúdiónév, SUM(hossz)
FROM Filmek
GROUP BY stúdiónév;
```

Egy összesítéseket tartalmazó SELECT záradékban csak a GROUP BY záradékban is megtalálható attribútumok jelenhetnek meg összesítési operátor nélkül.

A NULL értékeket bármilyen összesítés során figyelmen kívül hagyjuk.
Csoportosításkor a NULL-ok egy csoportba kerülnek.
Az összesítő függvények az üres csoportokra NULL értéket adnak eredményül, kivéve a COUNT, mely üres csoportra 0 eredményt ad.

A **HAVING** záradék (GROUP BY után) hatására, csak bizonyos feltételnek megfelelő csoportok kerülnek kialakításra.

## 10. VÁLTOZTATÁSOK AZ ADATBÁZISBAN, INSERT, DELETE, UDPATE (307-312.OLDAL)

**Beszúrás:**

**INSERT INTO R(A1,A2,…An) VALUES (v1,v2,…,vn)**

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, például:

```
INSERT INTO Stúdió(név)
SELECT DISTINCT stúdióNév
FROM filmek
WHERE stúdióNév NOT IN
(SELECT név
FROM stúdió);
```

**Új stúdiók beszúrása**

**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. Például:

```
UPDATE gyártásIrányító
SET név = 'Ig.' || név
WHERE azonosító IN
(SELECT elnökAzon FROM stúdió);
```

## 11. MEGSZORÍTÁSOK AZ SQL-BEN: KULCSOK, IDEGEN KULCSOK, HIVATKOZÁSI ÉPSÉG FENNTARTÁSA, AZONNALI-KÉSLELTETETT ELLENŐRZÉS (329-336. OLDAL)

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 hovatkozott attribútumokban is a másik reláció valamelyik sorában.

Deklarálása kétféleképpen lehetséges (példák: 331. oldal):

- ha az idegen kulcs egyetlen attribútum, akkor a neve és típusa után adhatjuk meg, hogy az egy másik tábla egy attribútumára hivatkozik.
    - REFERENCES <tábla> (<attribútum>)
- CREATE TABLE utasításban, az attribútumok listája után egy külön utasításban adjuk meg, hogy bizonyos attribútumok idegen kulcsot alkotnak
    - FOREIGN KEY (<attribútumok>) REFERENCES <tábla> (<attribútumok>)

Hivatkozási épség fenntartása:
Megszorítást megsértő utasításokat a rendszer alapértelmezésben elutasítja, de van két másik lehetőség, hogy az ilyen módosítások mégis végbemehessenek:

- továbbgyűrűző eljárás: ezzel az eljárással az idegen kulcsok hivatkozott attribútuma(i) is megváltoznak (CASCADE).
- NULL értékre állítás módszere. A hivatkozott reláció egy idegen kulcsának értékét érintő módosításnál a hivatkozott reláció megfelelő értékeit NULL értékre változtatjuk (SET NULL).

CREATE TABLE Stúdió (
név CHAR(30) PRIMARY KEY,
cím VARCHAR(255),
elnökAzon INT REFERENCES gyártásIrányító(azonosító)
ON DELETE SET NULL
ON UPDATE CASCADE );

**Ha beszúrásnál keletkezne hiba, azt az SQL azonnal elutasítja.**

Előfordulhat olyan eset, hogy az adatok frissítése sértené 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 követhezi INITIALLY DEFERRED (tranzakció végéig késleltetve van az ellenőrzés) vagy INTIALLY IMMEDIATE (az ellenőrzés minden egyes utasítás után azonnal megtörténik) kiegészítés. Példa:

CREATE TABLE Stúdió (
név CHAR(30) PRIMARY KEY,
cím VARCHAR(255),
elnökAzon INT UNIQUE
REFERENCES gyártásIrányító (azonosító)
DEFERRABLE INITIALLY DEFERRED);

## 12. ATTRIBÚTUMOKRA ÉS SOROKRA VONATKOZÓ MEGSZORÍTÁSOK (338-342.OLDAL)

CREATE TABLE utasításon belül fogalmazhatók meg.

Attribútumra vonatkozó feltételek:

- **NOT NULL** – nem engedi meg olyan sor létrehozását, amely az adott attribútumon NULL értéket venne fel. Erre az attribútumra (ha az idegen kulcs), nem használhatjuk a NULL értékre állítás módszerét sem.
  **CREATE TABLE tábla (… elnökazon INT REFERENCES tábla2(név) NOT NULL…);**
- **CHECK –** bonyolultabb megszorítások rendelhetők hozzá egy attribútumhoz, ha a deklarációjában a CHECK kulcsszót használjuk, mögötte zárójelek között egy feltétellel. A feltétel hivatkozhat az éppen használt attribútumra, de használható alkérdés is. Az attribútum alapú CHECK csak akkor ellenőrzi a feltételt, ha a módosítás érinti az adott attribútumot.
  **elnökazon INT REFERENCES gyártásirányító(azonosító) CHECK (elnökazon >=10000)**
  **elnökazon INT CHECK (elnökazon IN (SELECT azonosító FROM gyártásirányító))**

Sorokra vonatkozó megszorítások:
Megadhatók úgy, hogy a CREATE TABLE utasításban az attribútumok, a kulcsok és az idegen kulcsok deklarációja után megadjuk a CHECK kulcsszót, majd zárójelek között egy feltételt. A rendszer a reláció sorára vonatkozó feltételként értékeli ki, amelyben nevükkel hivatkozhatunk a reláció attribútumaira. A rendszer minden új, vagy módosított sorra kiértékeli a feltételt, ha az hamis, elutasítja a műveletet.

Például:

```
CREATE TABLE FilmSzínész (
név CHAR(30) PRIMARY KEY,
cím VARCHAR(255),
nem CHAR(1),
születésiDátum DATE,
CHECK (nem='N' OR név NOT LIKE 'Ms.%'));
```

## 13. NÉZETTÁBLÁK (361-369.OLDAL)

Az SQL relációk egyik típusa, nem léteznek fizikailag az adatbázisban. A lekérdezésekhez hasonló kifejezéssel definiáljuk, ugyanúgy lekérdezhetők mint a fizikailag létező táblákat, egyes esetekben még módosíthatjuk is őket.

**CREATE VIEW <név> AS <definíció>;**

*Ahol a definíció egy SQL-lekérdezés.*

```
CREATE VIEW FilmProducer AS
SELECT Filmek.filmcím, név
FROM Filmek, GyártásIrányító
WHERE producerAzon = azonosító;
```

Ugyanúgy lekérdezhetők, mint fizikai társaik, megadjuk a nézettábla nevét, és a nézettábla definícióját használva az adatbázisrendszerre bízzuk, hogy előállítsa a lekérdezéshez szükséges sorokat. A lekérdezésekben vegyíthetők a táblák és nézettáblák (FROM FilmProducer, GyártásIrányító). Nézettábla létrehozásakor megadhatjuk, mire legyenek átnevezve az attribútumok, zárójelezett felsorolásban:

```
CREATE VIEW FilmProducer(filmCím, ProducerNév) AS
SELECT Filmek.filmcím, név
FROM Filmek, GyártásIrányító
WHERE producerAzon = azonosító;
```

Nézettábla megszűntetése: **DROP VIEW FilmProducer.** Nincs kihatással az eredeti táblára, de ha az eredeti táblát töröljük, az használhatatlanná teszi a nézettáblát.

R reláción létrehozott nézettábla módosítható, ha:

- definíciójában nincs DISTINCT
- WHERE záradékban nem szerepel R
- a FROM záradékban csak R szerepelhet, és az is csak egyszer
- a SELECT-nek minden olyan attribútumot tartalmaznia kell, ami nem lehet NULL és nincs alapértelmezett értéke.
-

A beszúrás és törlés az alaptáblában történik. Beszúráskor csak azon attribútumoknak tudunk értéket adni, amelyek a nézettáblát definiáló SELECT utasításban benne voltak.

```
INSERT INTO ParamountFilmek VALUES('Paramount','Csillagok hábúroja',1979);
```

ez a Filmek táblára úgy hat, mint a következő:

```
INSERT INO Filmek(stúdióNév,filmcím,év) VALUES('Paramount','Csillagok hábúroja',1979);
```

Törléskor, hogy biztosítsuk azt, hogy csak azok a sorok törlődjenek, amelyek a nézettáblában láthatóak, a törlő utasítás WHERE záradékához hozzá kell és-elni a nézettáblát definiáló utasítás WHERE záradékában lévő feltételt.

```
DELETE FROM Filmek WHERE filmcím LIKE '%háború%' AND stúdióNév = 'Paramount';
```

# 14. SQL: SÉMÁBAN TÁROLT ELJÁRÁSOK (PSM) VAGY PL/SQL (414-418.OLDAL)

Persistent, Stored Modules – Tartós, tárolt modulok. Segítségével eljárásokat fogalmazhatunk meg, sémaelemként letárolhatjuk ezeket. Tárolás után használhatjuk őket a lekérdezésekben, utasításokban.

Eljárás:

**CREATE PROCEDURE <név> (<paraméterek>) <lokális deklarációk> <eljárás törzse>;**

Paraméterek: mód (IN (alapértelmezett, akár el is hagyható), OUT, INOUT), név, típus hármasok
Függvény:

**CREATE FUNCTION <név> (<paraméterek>) RETURNS <típus>**
**<lokális deklarációk> <eljárás törzse>;**

Paraméterei csak IN módúak lehetnek.

```
CREATE PROCEDURE Költözés (
IN régiCím VARCHAR(255),
IN újCím VARCHAR(255))
UPDATE filmSzínész
SET cím = újCím
WHERE cím = régiCím;
```

PSM utasítások:

- CALL – **Eljárás**hívás. **CALL <eljárás neve> (<attribútumlista>);** Érkezhet:
  - Egy befogadó nyelvi programból: EXEC SQL CALL foo(2,3);
  - másik PSM függvény vagy eljárás utasításaként
  - Általános SQL parancsként: CALL foo(1,4);
- RETURN – visszatérési utasítás. Csak függvényben szerepelhet, kiértékeli az utána álló kifejezést, majd az eredményt értékül adja a visszatérési értéknek. **Nem adja vissza azonnal a vezérlést, az a következő utasítással folytatódik.**
- DECLARE – lokális változó deklarálása. **DECLARE <név> <típus>**
- SET – értékadó utasítások. **SET <változó> = <kifejezés>;** Hiányzó kifejezés esetén NULL értéket kap. A kifejezés akár egy SQL lekérdezés is lehet, ha az egy értéket ad vissza.
- **BEGIN**,**END** – pontosvesszővel lezárt utasítások csoportja.
- **Címkék** – utasítás címkézéshez az utasítás elé egy nevet írunk, amit egy kettőspont követ.
- **Elágazások**:

**IF <feltétel> THEN <utasításlista>**
**ELSEIF <feltétel> THEN <utasításlista>**
**…**
**ELSE <utasításlista>**
**END IF;**

- Lekérdezések (előfordulhatnak):
  - feltételekben (minden olyan helyen, ahol SQL-ben lehetne, szerepelhetnek alkérdések)
  - olyan alkérdések, amelyek csak egy értéket adnak vissza, használhatók értékadás jobb oldalán
    - SET elnökFizetése = (SELECT nettóbevétel FROM stúdió, gyártásirányító WHERE elnökazon = azonosító AND stúdió.név = stúdióNév);

  - egy sort visszaadó SELECT utasítás, lehet lokális változó, eljárás paramétere
    - CREATE PROCEDURE vmiEljaras(IN stúdióNév CHAR(15))
      DECLARE igazgatóFizetése INTEGER;
      SELECT nettóbevétel
      INTO igazgatóFizetése
      FROM Stúdió, GyártásIrányító
      WHERE elnökazon = azonosító AND Stúdió.név = Stúdiónév;
      …
  - sormutatók (??? OPEN, FETCH (sorkinyerés), CLOSE)
- Ciklusok: **cikluscímke: LOOP <utasításlista> END LOOP;** A cikluscímke használata opcionális, de hasznos, például a következő esetben: **LEAVE <cikluscímke> .** Ezen utasítás hatására kilépünk a ciklusból. Feltételt deklarálhatunk egy értékhez, ami elősegíti a feltételvizsgálatot:

  DECLARE Nincs_Több CONDITION FOR SQLSTATE '02000';
  IF Nincs_Több THEN LEACE ciklus END IF;
  - WHILE <feltétel> DO <utasításlista> END WHILE;
  - REPEAT <utasításlista> UNTIL <feltétel> END REPEAT;
  - FOR ciklus csak egy sormutatóban történő léptetése használható:
    - FOR <ciklus neve> AS <sormutató neve> CURSOR FOR
      <lekérdezés>
      DO
      <utasításlista>
      END FOR;
- Kivételek: az SQL-rendszer a hibákat egy ötjegyű, nem csak nulla számjegyekből álló SQLSTATE nevű karakterlánc beállításával jelzi. Például: '02000' nem talál sort; '21000' egysoros lekérdezés több mint egy sort eredményezett. Kivételkezelő kódrészlet deklarálható, amely minden esetben kiváltódik, ha ezen hibakódok egy listájának valamely tagja előfordul az utasítás(sorozat) végrehajtása során.

  DECLARE <hova menjen> HANDLER FOR <feltétellista> <utasítás>;
  - Hova menjen:
    - CONTINUE: folytatjuk a hibát kiváltó utasítást követő utasítással, a kezelő utasítás(ok) lefutása után.
    - EXIT: a kezelő utasításainak lefutása után kilép az adott BEGIN END blokkból, az azt követő utasítással folytatja.
    - UNDO: ugyanolyan mint az EXIT, de a blokk végrehajtásának eredményeként történt változtatásokat visszavonja.

```
CREATE FUNCTION KiadásiÉv(fc VARCHAR(255)) RETURNS INTEGER
DECLARE Nincs_Találat CONDITION FOR SQLSTATE '02000';
DECLARE Túl_Sok CONDITION FOR SQLSTATE '21000';
BEGIN
DECLARE EXIT HANDLER FOR Nincs_Találat, Túl_Sok RETURN NULL;
RETURN (SELECT év FROM Filmek WHERE filmcím = fc);
END;
```

Függvény meghívása például:

```
INSERT INTO SzerepelBenne(filmcím,filmév,színésznév)
VALUES('Emlékezz a titánokra!',KiadásiÉv('Emlékezz a titánokra!'),'Denzel Washington');
```

# 15. DATALOG 1. – BIZTONSÁGOS DATALOG-SZABÁLY, NEGÁLT PREDIKTÁTUM, JELENTÉS, EXTENZIONÁLIS ÉS INTENZIONÁLIS PREDIKTÁTUMOK (236-242.OLDAL)

Database logic, if-then szabályokat tartalmaz, alternatíva az algebrán alapuló nyelvek mellett. Minden szabály azt az elvet fejezi ki, hogy az egyes relációkban a sorok bizonyos kombinációjából következtethetünk arra, hogy valamilyen másik sornak szerepelnie kell valamilyen másik relációban vagy egy lekérdezés eredményében.

Datalogban a relációkat **prediktátumokkal** reprezentáljuk, mindegyik rögzített számú argumentummal rendelkezik, logikai értékkel térnek vissza, az argumentumaikon keresztül kapnak értéket. Egy prediktátumot, amelyet argumentumai követnek **(relációs) atomnak** nevezünk, például: P(x1,x2,…,xn). P(x1,x2,…,xn) atom értéke TRUE, ha (x1,x2,…,xn) a P egy sora, illetve FALSE, ha nem az. Az argumentumok lehetnek konstansok, illetve változók is. Egy másik fajta atom, az **aritmetikai atom**, ami tulajdonképpen két aritmetikai kifejezés összehasonlítása.

Műveletek leírásának szabályai a következőkből épülnek fel:

- egy **fej**nek nevezett relációs atom
- ezt követi a ← szimbólum, amelyet gyakran „if"-nek olvasunk ki
- ezután következik a törzs, amely egy vagy több **részcél**nak nevezett atomból áll, amelyek lehetnek relációs vagy aritmetikai atomok. A részcélokat AND-del kötjük összes, szükség esetén használható a NOT is.

**HosszúFilm(fc,é) ← Filmek(fc,é,h,m,s,p) AND h> 100**

Akkor igaz a HosszúFilm(fc,é), ha létezik a Filmek relációnak egy olyan sora, amelyre teljesülnek a következők:

- a filmcím és év attribútumoknak megfelelő két első komponens értéke fc és é.
- a hossz attribútumnak megfelelő harmadik komponens (h) értéke legalább 100
- a negyedik, ötödik, hatodik komponens értéke tetszőleges.

HA egy változó mindösszesen egyszer szerepel a szabályban (tehát nem használjuk, csak argumentumként szerepel), azt kipótolhatjuk _ jelekkel:

HosszúFilm(fc,é) ← Filmek(fc,é,h,_,_,_) AND h> 100

Datalogban egy lekérdezés egy vagy több szabály együttese. Ha egynél több relációt tartalmaz a szabályfej, akkor ezen relációk egyike lesz a lekérdezés eredménye, míg a maradék az eredmény megfogalmazásában segít. Meg kell jelölnünk, melyik reláció adja meg a lekérdezésre a választ.

A szabályban szereplő változók felveszik az összes lehetséges értéket, amikor a változók értékei igazzá teszik az összes részcélt, akkor megkapjuk, hogy mi a fej értéke az aktuális változókra és hozzáadjuk a kapott sort a fejben szereplő prediktátumnak megfelelő relációhoz.

**Biztonságossági feltétel:** a szabályban szereplő valamennyi változónak szerepelnie kell valamely nem negált, relációs részcélban is.

**Extenzionális** prediktátumok, amelyekhez rendelt relációk megtalálhatók az adatbázisban

**Intenzionális** prediktátumok, amelyekhez rendelt relációkat egy vagy több Datalog-szabály kiértékelése alapján kapjuk meg.

# 16. DATALOG 2. – RELÁCIÓS ALGEBRAI KIFEJEZÉSEK ÁTÍRÁSA DATALOGBA, VETÍTÉS-KIVÁLASZTÁS-SZORZAT KIFEJEZÉSEK, KÜLÖNBSÉG ÉS UNIÓ DATALOGBAN, REKURZÓMENTES DATALOG PROGRAMOK, REKURZIÓ DATALOGBAN (244-253.OLDAL)

R és S relációkat ugyanannyi attribútummal rendelkezőnek tekintjük, ez a szám legyen n.

R U S: két szabály, n változó: R(a1,a2,…,an) és S(a1,a2,…,an). Mindkét szabálynak fejrésze Válasz(a1,a2,…an). Eredmény: R és S összes sora benne lesz a válaszrelációban.

R ∩ S: Egy szabály, amelynek törzse az alábbi: R(a1,a2,…,an) AND S(a1,a2,…,an). Minden sor pontosan akkor lesz benne a válaszrelációban, ha mindkét relációban benne van. (fej: Válasz)

R – S: Egy szabály, törzse a következő: R(a1,a2,…,an) AND NOT S(a1,a2,…,an). Pontosan akkor lesz benne egy sor a válaszrelációban, ha benne van R-ben, de nincs benne S-ben (fej: Válasz)

Vetítés: Egyetlen szabály, a fej argumentumában azoknak az attribútumoknak megfelelő változók szerepelnek, amelyekre a vetítés történik, és olyan sorrendben, ahogyan azt a vetítésben megkívánjuk: **P(fc,é,h) ←Filmek(fc,é,h,m,s,p)**

Kiválasztás: Ebben az esetben is egy szabályt kell készítenünk, annak két része lesz:
- egy relációs részcél, amely megfelel annak a relációnak, amelyben a kiválasztás történik. Ez annyi különböző változót tartalmaz, ahány attribútummal a reláció rendelkezik.
- megadunk egy aritmetikai részcélt, amely megegyezik az összehasonlítással. Attribútumnevek helyett itt a változókat kell használnunk, amelyekkel az előző lépésben a relációs részcélt megadtuk.

$\sigma_{\text{hossz>100 AND stúdiónév='FOX'}}$(Filmek) = S(fc,é,h,m,s,p) ←Filmek(fc,é,h,m,s,p) AND h>100 AND s='FOX'

Szorzat: egyetlen szabály megadásával előállítható, a fejben az összes olyan argumentum szerepel, amely vagy R-ben vagy S-ben szerepel, R argumentumai megelőzik S-ét.

**P(a,b,c,d,e,f) ← R(a,b,c) AND S(d,e,f)**

Természetes összekapcsolás: nagyon hasonló mint a szorzat, csak a közös argumentumoknak ugyanazt a nevet adjuk.

**TÖ(a,b,c,d) ← R(a,b) AND S(b,c,d)**

Théta-összekapcsolás: mint az előbb, kiegészítve feltételekkel:

**Ö(a,ub,uc,vb,vc,d)←U(a,ub,uc) AND V(vb,vc,d) AND a<d AND ub ≠ vb**

Tetszőlegesen bonyolult kifejezések építhetők:

W(fc,é,h,m,s,p) ← Filmek(fc,é,h,m,s,p) AND h> 100
X(fc,é,h,m,s,p)← Filmek(fc,é,h,m,s,p) AND s='FOX'
Y(fc,é,h,m,s,p) ← W(fc,é,h,m,s,p) AND X(fc,é,h,m,s,p)
Válasz(fc,é)←Y(fc,é,h,m,s,p)

Rekurzió Datalogban (kis gráfos példa):

Út(X,Y)←Él(X,Y)
Út(X,Y)←Él(X,Z) AND Út(Z,Y)

**Magyarázat a 253. oldalon**

Készítette: Plézer Viktor István

# 17. Rekurzió SQL-ben, WITH, az Eljut feladat, PSM ciklus (467-474.oldal)

A WITH utasítás segítségével SQL-ben megfogalmazhatjuk ideiglenes relációnak a rekurzív vagy nem rekurzív definícióját. A WITH utasítás egyszerű alakja:

**WITH R AS <R definíciója> <R-et használó lekérdezés>**

Tehát definiálunk egy R nevű ideiglenes relációt, majd használjuk egy lekérdezésben. Az ideiglenes relációk csak a WITH utasításhoz tartozó lekérdezéseken belül használhatók. Több reláció is definiálható a WITH után, a definíciókat vesszővel elválasztjuk, közülük bármelyik lehet rekurzív. Minden olyan relációt, amely rekurzív definícióban részt vesz, a **RECURSIVE** kulcsszó kell, hogy megelőzzön. Példa:

Járatok(légitársaság, honnan, hova, indulás, érkezés)

„Mely (x,y) várospárokra lehetséges egy vagy több átszállással eljutni x városból y városba?"

**Datalog:**

Eljut(x,y) ← Járatok(l,x,y,i,é)

Eljut(x,y)← Eljut(x,z) AND Eljut(z,y)

**SQL (WITH):**

```
WITH RECURSIVE Eljut(honnan,hova) AS
(SELECT honnan, hova FROM Járatok)
UNION
(SELECT R1.honnan, R2.hova
FROM Eljut R1, Eljut R2
WHERE R1.hova = R2.Honnan)
SELECT * FROM Eljut
//(SELECT hova FROM Eljut WHERE honnan ='DEN'
```

Az SQL csak a lineáris rekurziót támogatja. Egy SQL-rekurzió akkor legális, ha az R rekurzív relációdefiníciója csak olyan kölcsönösen rekurzív S relációt használ (R-et is beleértve), amelynek a használata S-ben „monoton".

# 18. Relációs adatbázisok tervezési elmélete 1. – Funkcionális függőség, kulcs, szuperkulcs, függőségi rendszerek, implikáció és levezetés, szétvághatósági és összevonási szabály (69-78.oldal)

A **funkcionális függőség** egy R reláción a következő formájú állítás:
**„Ha R két sora megegyezik az A1,A2,…,An attribútumokon (azaz ezen attribútumok mindegyikéhez megfeleltetett komponensnek ugyanaz az értéke a két sorban) , akkor meg kell egyezniük más attribútumok egy B1,B2,…,Bm sorozatán."**
Formálisan A1A2…An → B1B2…Bm –mel jelöljük és azt mondjuk hogy:

„A1A2…An funkcionálisan meghatározza B1B2…Bm-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.

Két funkcionális függőségi halmaz, S és T **ekvivalensek**, ha S-et pontosan ugyanazok a reláció- előfordulások elégítik ki, mint amelyek T-t.
S **következik** T-ből, ha minden olyan reláció-előfordulás, amely eleget tesz az összes T-beli függőségnek, eleget tesz minden S-beli függőségnek is.

A1A2…An → B1B2…Bm szétválasztható A1A2…An → B1, …, A1A2…An → Bm –re.

**Szétválaszthatósági szabály:** Az A1A2…An → B1B2…Bm függőséget helyettesíthetjük az A1A2An → Bi (i=1,…,m) funkcionális függőségekből álló halmazzal.

**Összevonhatósági szabály:** Az A1A2…An→Bi (i=1,…,m) funkcionális függőségekből álló halmazt helyettesíthetjük az A1A2…An→B1B2…Bm egyetlen függőséggel.

**Triviális megszorítás:** fennáll a reláció minden előfordulására, attól függetlenül, hogy milyen más megszorításokat fogalmazunk meg. Ha a megszorítások funkcionális függőségek, könnyen meghatározhatjuk hogy egy FD triviális. Ezek azok az A1A2…An→B1B2…Bm függőségek, ameylekre: {B1B2…Bm}⊆{A1A2…An}.

**Triviális függőségi szabály:** A1A2…An→B1B2…Bm ekvivalens az A1A2…An→C1C2…Ck függőséggel, ahol C-kkel jelöltük azokat a B-ket, amelyek nem fordulnak elő az A-k között.

Azt mondjuk, hogy az egy vagy több attribútumból álló {A1,A2,…,An} 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 A1,A2,…,An –nen megegyezne.
- Nincs olyan valódi részhalmaza {A1,A2,…,An} halmaznak, amely funkcionálisan meghatározná R összes többi attribútumát, azaz a kulcsnak minimálisnak kell lennie.

Azokat az attribútumhalmazokat, amelyek tartalmaznak kulcsot, szuperkulcsoknak nevezzük (kulcsnál bővebb halmaz). Minden kulcs egyben szuperkulcs is.

## 19. Relációs adatbázisok tervezési elmélete 2. – Attribútumhalmaz lezárása és algoritmusa, levezetési szabályok teljes halmaza, függőségi halmazok lezárása, minimális bázis, függőségek vetítése (79-87.oldal)

**Attribútumhalmaz lezárása:** Legyen {A1,A2,…,An} egy attribútumhalmaz, S pedig funkcionális függőségeknek egy halmaza. Az {A1,A2,…,An} halmaz S-beli függőségek szerint vett lezártja, azoknak a B attribútumoknak a halmaza, amelyekre minden olyan reláció, amely eleget tesz az összes S-beli függőségnek, eleget tesz A1A2…An→B-nek is. Azaz A1A2…An→B az S-beli függőségekből következik. Jelölése: $\{A1,A2,\ldots,An\}^+$. A1,A2,…,An mindig benne van a lezártban. Algoritmus:
1. Ha szükséges, vágjuk szét a függőségeket úgy, hogy a jobb oldalukon egyetlen attribútum maradjon.
2. Legyen X az az attribútumhalmaz, amely végül a lezárt lesz. Inicializáláskor legyen {A1,A2,…,An}.
3. Ismételten keresünk olyan B1B2…Bm→C funkcionális függőséget S-ből, amelyre minden B1,B2,…,Bm benne van az X attribútumhalmazban, de a C nincs. Ekkor C-t hozzávesszük az X halmazhoz. Mivel X minden lépésben csak nőhet, és minden reláció attribútumainak száma véges, tehát lesz olyan lépés, hogy X-hez már nem lehet elemet hozzávenni. Ekkor az eljárás befejeződik.
4. Az az X halmaz lesz a lezárt helyes értéke, amelyet már nem tudunk tovább bővíteni.

**Tranzitivitási szabály:** Ha A1A2…An→B1B2…Bm és B1B2…Bm →C1C2…Ck teljesül az R relációban, akkor A1A2…An→C1C2…Ck szintén teljesül R-ben.

Ha megadunk egy S függőséghalmazt (olyan függőségekből, amelyek fennállnak a relációban), akkor bármely S-sel ekvivalens függőséghalmazt S **bázisának** nevezünk.

A minimális bázis egy relációhoz az a B bázis, amely az alábbi három feltételt kielégíti:
- B összes függőségének jobb oldalán egy attribútum van
- Ha bármelyik B-beli függőséget elhagyjuk, a fennmaradó halmaz már nem bázis.

- Ha bármelyik B-beli funkcionális függőség bal oldaláról elhagyunk egy vagy több attribútumot, akkor az eredmény már nem marad bázis.

Egyetlen triviális függőség sem lehet a minimális bázisban, mert azt a 2. szabály szerint eltávolíthatjuk.

Funkcionális függőséghalmaz vetületének kiszámítása:

Bemenet: Egy R reláció, és egy másik R1 reláció, amelyet az $R1 = \pi_L(R)$ vetítéssel számítottunk ki. Továbbá adott egy S függőséghalmaz, mely fennáll R-ben.

Kimenet: Az R1-ben fennálló funkcionális függőségek halmaza.

Algoritmus:
- Legyen T a végül előálló funkcionális függőségek halmaza. Kezdetben T üres.
- Minden olyan X attribútumhalmazra, amely R1-nek része, számítsuk ki $X^+$-t. Ezt a kiszámítást az S-beli funkcionális függőségeket figyelembe véve végezzük, és előfordulhatnak olyan attribútumok is, amelyek R-ben szerepelnek, de R1-ben nem. Adjuk hozzá T-hez az összes nem triviális függőségeket, amelyek X→A formátumúak, ahol A eleme az $X^+$ és az R1 attribútumhalmaznak is.
- Ezután a kapott T bázisa az R1-beli funkcionális függőségeknek, de nem feltétlen minimális bázis. A minimális bázist előállíthatjuk T-ből az alábbi módosítások végrehajtásával:
    - Ha szerepel egy F funkcionális függőség T-ben, amely más T-beli függőségekből következik, akkor töröljük F-et a T halmazból.
    - Legyen Y→B egy T-beli funkcionális függőség, ahol Y legalább két attribútumot tartalmaz, és legyen Z az a halmaz, amelyet úgy kapunk, hogy Y-ból egy attribútumot elhagyunk. Ha Z→B függőség következik a T-beli funkcionális függőségekből (beleértve Y→B-t is), akkor cseréljük ki Y→B-t Z→B –re.
    - Ismételjük az előző lépéseket addig, amíg már nem lehet semmilyen módosítást végrehajtani.

## 20. Relációs adatbázisok tervezési elmélete 3. – Anomáliák, relációk felbontása, veszteségmentes összekapcsolás, Boyce-Codd normálforma, BCNF dekompozíció algoritmusa (90-97.oldal)

Azokat a problémákat, amelyek akkor fordulnak elő, amikor túl sok információt próbálunk egyetlenegy relációba belegyömöszölni, **anomáliának** nevezzük. Fajtái:
- **Redundancia**: Az információk fölöslegesen ismétlődhetnek több sorban.
- **Módosítási anomáliák**: Megváltoztatjuk az egyik sorban tárolt információt, miközben ugyanaz az információ változatlanul marad egy másik sorban.
- **Törlési anomáliák**: Ha az értékek halmaza üres halmazzá válik, akkor ennek mellékhatásaként más információt is elveszíthetünk.

Anomáliák megszűntetésének az elfogadott útja a relációk felbontása (dekompozíciója). R felbontása azt jelenti, hogy R attribútumait szétosztjuk úgy, hogy két új reláció sémáját alakítjuk ki belőlük. Legyen adott R reláció {A1,A2,…,An} sémával. R-t felbonthatjuk S(B1,B2,…,Bm) és T(C1,C2,…,Ck) relációkra úgy, hogy:
- {A1,A2,…,An} = {B1,B2,…,Bm} U {C1,C2,…,Ck}
- $S = \pi_{B1,B2,\ldots,Bm}(R)$
- $T = \pi_{C1,C2,\ldots,Ck}(R)$

A felbontás célja, hogy egy relációt többel helyettesítsünk úgy, hogy ezzel megszüntessük az anomáliákat. A Boyce-Codd normálforma kiküszöböli az anomáliákat:

Az R reláció **BCNF-ben** van akkor és csak akkor, ha minden olyan esetben, ha R-ben érvényes egy A1A2…An→B1B2…Bm nem triviális függőség, akkor az {A1,A2,…,An} halmaz R szuperkulcsa.

Azt a felbontási stratégiát követjük, hogy kiválasztunk egy A1A2…An→B1B2…Bm nem triviális funkcionális függőséget, amely megsérti a BCNF-et, azaz {A1,A2,…,An} nem szuperkulcs. A jobb oldalhoz hozzáadjuk az összes {A1,A2,…,An} által funkcionálisan meghatározott attribútumot. Algoritmus (BCNF dekompozíció):

Bemenet: Az R0 reláció az S0 függőséghalmazzal

Kimenet: Az R0 dekompozíciója relációk csoportjára, melyek mindegyike BCNF-ben van.

Algoritmus: A következő lépések bármely R relációra és S függőséghalmazra alkalmazhatók rekurzívan. Kezdetben legyen R = R0 és S = S0.

- Ellenőrízzük, R BCNF-ben van e. Ha igen, akkor készen vagyunk és {R} a válasz.
- Ha vannak BCNF-et megsértő függőségek, akkor X→Y legyen egy ilyen. Állítsuk elő $X^+$-t. Legyen R1 = $X^+$ az egyik relációséma, tovább R2-ben legyenek benne X attribútumai és azok az R-beli attribútumok, amelyek nincsenek $X^+$-ban.
- Határozzuk meg R1 és R2 függőségeit, a kapottak legyenek rendre S1,S2.
- Rekurzívan bontsuk fel R1-et és R2-t ennek az algoritmusnak használatával. A végeredmény a dekompozíciók eredményeinek uniója lesz.

## 21. Relációs adatbázisok tervezési elmélete 4. – 3. normálforma, 3NF szintetizáló algoritmus (108-111.oldal)

Az R reláció harmadik normálformában van akkor és csak akkor, ha:

- Valahányszor létezik R-ben egy A1A2…An→B1B2…Bm nem triviális függőség, akkor vagy az {A1,A2,…,An} halmaz az R szuperkulcsa, vagy azokra az attribútumokra B1,B2,…,Bm közül, amelyek nincsenek az A-k között, teljesül, hogy egy kulcsnak az elemei (nem feltétlenül ugyannak a kulcsnak).

Kifejtjük és indokoljuk, hogyan bontható fel egy R reláció olyan relációk halmazává, amelyekre teljesülnek az alábbiak:

- A felbontásban szereplő relációk mindegyike 3NF-ben van.
- A felbontásnak veszteségmentes összekapcsolása van.
- A felbontásnak függőségmegőrző tulajdonsága van.

Bemenet: Egy R reláció és az R-re vonatkozó funkcionális függőségek F halmaza

Kimenet: Az R felbontása relációk gyűjteményére, melyek mindegyike 3NF-ben van. A felbontás rendelkezik veszteségmentes összekapcsolással és függőségmegőrző tulajdonsággal.

Algoritmus:

- Keressük meg F egy minimális bázisát, amit hívjunk G-nek.
- A G-ben szereplő összes X→A funkcionális függőségre használjuk X A-t sémaként a felbontás egy relációjához.
- Amennyiben a 2. lépésben kapott relációk attribútumhalmazának egyike sem szuperkulcs R-ben, adjunk még egy relációt az eredményhez, melynek sémája kulcs lesz R-hez.

# 22. RELÁCIÓS ADATBÁZISOK TERVEZÉSI ELMÉLETE 5. – TÖBBÉRTÉKŰ FÜGGŐSÉGEK, 4. NORMÁLFORMA (112-119.OLDAL)

A többértékű függőség valamely R relációra vonatkozó állítás, miszerint amikor rögzítjük egy attribútumhalmaz értékeit, akkor bizonyos más attribútumok értékei függetlenek lesznek a relációban szereplő összes többi attribútum értékeitől. Pontosabban kifejezve azt mondjuk, hogy az A1A2…An→→B1B2…Bm többértékű függőség fennáll R relációban, ha tekintve az R sorai közül azokat, amelyek minden egyes A-beli attribútumon ugyanazt a bizonyos értéket veszik fel, akkor ezekre a sorokra azt találjuk, hogy a B-ken felvett értékek halmaz független R-nek az A-któl és B-ktől eltérő összes attribútumán felvett értékek halmazától. Még pontosabban azt mondjuk, hogy ez a többértékű függőség érvényes, ha:
Az R reláció minden olyan t és u sorpárjára, amelyek minden A-n megegyeznek, találhatunk R-ben olyan v sort, amely megegyezik:

- t-vel és u-val az A-kon
- t-vek a B-ken és
- u-val R minden olyan attribútumán, amely nincs az A-kban vagy a B-kben

**Triviális többértékű függőség:** Az A1A2…An→→B1B2…Bm többértékű triviális függőség fennáll bármely relációban, ha {B1,B2,…,Bm} ⊆ {A1,A2,…,An} teljesül.

**Tranzitivitási szabály:** ha érvényesek A1A2…An→→B1B2…Bm és B1B2…Bm→→C1C2…Ck többértékű függőségek egy bizonyos relációra, akkor A1A2…An→→C1C2…Ck is érvényes. Mindazon C-ket, amelyek egyben A-k is, törölhetjük a jobb oldalról.

A többértékű függőség **nem tesz eleget a szétválaszthatósági/összevonhatósági szabály szétválaszthatóságra vonatkozó részének.**

**Funkcionális függőség előléptetése:** Minden funkcionális függőség egyben többértékű függőség is. Azaz amennyiben A1A2…An→B1B2…Bm akkor A1A2…An→→B1B2…Bm.

**Komplementer-szabály:** Ha A1A2…An→→B1B2…Bm többértékű függőség az R relációban, akkor R kielégíti az A1A2…An→→C1C2…Ck többértékű függőséget is, ahol a C-k az R összes attribútumai közül éppen azok, amelyen nincsenek sem az A-k sem a B-k között.

Egy R reláció **negyedik normálformában** van, ha valahányszor az A1A2…An→→B1B2…Bm nem triviális többértékű függőség fennáll, akkor {A1,A2,…,An} szuperkulcs.

Negyedik normálformára bontás algoritmusa:
Bemenet: Az R0 reláció és az S0 funkcionális és többértékű függőségeket tartalmazó halmaz.
Kimenet: Az R0 dekompozíciója olyan relációkra, amelyek mindegyik 4NF-ben van. A dekompozíció veszteségmentes összekapcsolási tulajdonsággal rendelkezik.
Algoritmus:

- Keressük a 4NF feltétel egy megsértését R-ben. Legyen ez A1A2…An→→B1B2…Bm, ahol {A1,A2,…,An} nem szuperkulcs. Megjegyezzük, hogy ez a többértékű függőség lehet egy valódi többértékű függőség, vagy levezethető egy megfelelő A1A2…An→B1B2…Bm S-beli funkcionális függőségből, mivel tudjuk, hogy minden funkcionális függőség egyben többértékű függőség is. Ha nincs ilyen, akkor készen vagyunk, R önmagában egy megfelelő dekompozíció.
- Ha találunk 4NF sértést, akkor osszuk fel a 4NF feltételt megsértő R relációt két sémára:
    - R1, melynek a sémája tartalmazza az A-kat és a B-ket
    - R2, melynek a sémája az A-kból és a sem A-ban sem B—ben nem szereplő attribútumokból áll.

- Keressük meg az R1-ben és R2-ben fennálló funkcionális és többértékű függőségeket. Rekurzívan dekomponáljuk R1-et és R2-t a vetített függőségeikkel.