Adatbázisok elmélete 11. előadás

Csima Judit
Budapesti Műszaki és Gazdaságtudományi Egyetem
Számítástudományi Tsz.
I. B. 136/b
csima@cs.bme.hu

2003. Március 19.

DML utasítások — SELECT

Ezzel valósítható meg a kiválasztás, vetítés és a szorzat.

Szintaxis: SELECT <reláció$_i$>.<attrib$_1$>, ..., <reláció$_j$>.<attrib$_n$>
FROM <reláció$_1$>, ..., <reláció$_m$>
WHERE <kifejezés>

Relációs algebrabeli megfelelője (de nem pontosan, mert SQL-ben SELECT nem küszöböli ki a többszörös sorokat):

$$\pi_{<\text{attrib}_1>,\ldots,<\text{attrib}_n>}\sigma_{<\text{kifejezés}>}(<\text{reláció}_1> \times \cdots \times <\text{reláció}_m>)$$

Példa 1: A budapesti mozik azonosítói és nevei
SELECT mozi.moziID, mozi.nev FROM mozi WHERE mozi.varos="Budapest"

Példa 2: A pénteken hétkor kezdődő filmek azonosítói
SELECT vetit.filmID FROM vetit WHERE vetit.nap="péntek" AND vetit.ido="19:00"

Megjegyzés:

- kiértékelés: minden egyes FROM utáni relációnak megfelel egy-egy sorváltozó, ami az egyes relációk sorain megy végig (egymásba ágyazott ciklusokkal például). Ha találat van, azaz a WHERE feltétel igaz az aktuális értékekre, akkor a SELECT utáni mezők kiíródnak
- úgy gondolhatunk a kiértékelésre, mintha először vennénk a FROM utáni relációk direkt szorzatát és aztan arra csinálnánk a kiválasztást és a vetítést.
- ha többszörös sorokat nem akarunk: SELECT DISTINCT (ennek ára van!!!)
- WHERE el is hagyható
- WHERE-ben mi állhat: erről később
- az eredmény az ORDER BY kulcsszó segítségével rendezhető, megadható hogy mely oszlopok szerint és hogy növőleg vagy csökkenőleg
- A fenti két példa mutatja, hogy a kiválasztás és a vetítés megy, a szorzatra a sorváltozók bevezetése után nézünk példát

SQL Sor- és oszlopváltozók

A FROM után felsorolt relációkhoz **sorváltozókat** rendelhetünk. Szintaxis (FROM után <reláció$_i$> helyén): <reláció$_i$> AS <sorváltozó>

A SELECT után elhelyezett attribútum-hivatkozásokhoz **oszlopváltozókat** rendelhetünk. Szintaxis (SELECT után <reláció$_i$>.<attrib$_j$> helyén):
<reláció$_i$>.<attrib$_j$> AS <oszlopváltozó> Így átnevezés lehetséges az eredmény megjelenítésekor:

Például:
SELECT nev AS Filmszínház,
varos AS Hely FROM mozi

$\Longrightarrow$

	Filmszínház	Hely
	⋮	

Az oszlopváltozók valójában csak az eredményreláció attribútumainak elnevezésére használhatók, a SELECT utasításon belül nem hivatkozhatunk rájuk.

A <reláció$_i$>. előtag elhagyható, ha egyértelmű, hogy melyik relációról van szó, továbbá a <reláció$_i$>. előtag helyett <sorváltozó>. előtag is szerepeltethető.

Attribútumhivatkozások

Amikor egy attribútumra akarunk hivatkozni, három lehetőségünk van:

- <attribútum> (ha ez egyértelmű)
- <reláció>.<attribútum> (ha ez egyértelmű – N.B.: egy reláció többször is szerepelhet a FROM után, lesz példa)
- <sorváltozó>.<attribútum> (mindig használható)

Példa 3: A pénteken vetített filmek címei és rendezői (természetes illesztés)
SELECT cim, rendezo FROM film, vetit WHERE vetit.filmID = film.filmID AND nap="péntek"

Példa 4: Azok a várospárok, ahol vannak azonos nevű mozik (FONTOS)
SELECT m1.varos, m2.varos FROM mozi AS m1, mozi AS m2 WHERE m1.nev = m2.nev AND m1.varos <> m2.varos

Megjegyzés: a várospárok mindkét sorrendben megjelennek, és több azonos nevű mozi esetén többször is megjelennek. Az elsőre megoldás: $<>$ helyett legyen $<$, amúgy meg DISTINCT

A WHERE kifejezés

Kifejezés felépítése:

- logikai műveletek: AND, OR, NOT
- összehasonlítás: =, <>, >=, <=, LIKE, BETWEEN
- aritmetikai műveletek: +, -, *, DIV, MOD
- változóhivatkozások: <sorváltozó>.<attribútum>, <reláció>.<attribútum>, <attribútum>
- konstans (szám,karakterlánc): 137, 42e-3, "füzér"
- NULL érték vizsgálata IS NULL, IS NOT NULL (később lesz)
- alkérdés is lehet itt (majd erről később)

LIKE és BETWEEN használata

LIKE használata:

- "_" egy tetszőleges karakterre illeszkedik
- "%" tetszőleges karakterláncra illeszkedik

BETWEEN használata: BETWEEN a AND b jelentése $a \leq . \leq b$

Példa 5: A 150 és 200 közötti azonosítójú mozik közül azok, amelyek B-vel kezdődő nevű városban vannak, és a nevük hárombetűs.
SELECT nev FROM mozi WHERE moziID BETWEEN 150 AND 200 AND varos LIKE "B%" AND nev LIKE "___"

Műveletek relációkkal

A részeredményül kapott relációkkal (**ha azok sémája azonos!**) halmazműveleteket (unió, metszet, különbség) végezhetünk.

Unió (valamely eredményrelációban szereplő sorok):

- Szintaxis: <eredményreláció1> UNION <eredményreláció2>
- Példa 6: A pénteken vagy szombaton játszott filmek (nem hatékony!):
  (SELECT cim FROM film, vetit WHERE vetit.nap = "péntek" AND film.filmID = vetit.filmID)
  UNION
  (SELECT cim FROM film, vetit WHERE vetit.nap = "szombat" AND film.filmID = vetit.filmID)

Metszet (mindkét eredményrelációban szereplő sorok):

- Szintaxis: <eredményreláció1> INTERSECT <eredményreláció2>
- Példa 7: A pénteken és szombaton is játszott filmek:
  (SELECT cim FROM film, vetit WHERE vetit.nap = "péntek" AND film.filmID = vetit.filmID)
  INTERSECT
  (SELECT cim FROM film, vetit WHERE vetit.nap = "szombat" AND film.filmID = vetit.filmID)

Különbség (az első reláció azon sorai, melyek a másodikban nem szerepelnek):

- Szintaxis: <eredményreláció1> MINUS <eredményreláció2>
- Példa 8: A pénteken igen, de szombaton nem játszott filmek:
  (SELECT cim FROM film, vetit WHERE vetit.nap = "péntek" AND film.filmID = vetit.filmID)
  MINUS
  (SELECT cim FROM film, vetit WHERE vetit.nap = "szombat" AND film.filmID = vetit.filmID)

A szabványban MINUS helyett EXCEPT szerepel, de a gyakorlatban a MINUS használatos.

Állítás. *Az SQL relációsan teljes.*

Bizonyítás: Most láttuk az uniót és különbséget, a többi pedig már volt, de újra:

vetítés: $\pi_{A_{i_1},A_{i_2},\ldots,A_{i_k}}(R)$-nek megfelelő lekérdezés: SELECT $A_{i_1},A_{i_2},\ldots,A_{i_k}$ FROM R

kiválasztás: $\sigma_F(R)$-nek megfelel a
SELECT * FROM R WHERE F'
ahol F'az, ami F-ből jön átírással ($\wedge,\vee,\neg$ helyett AND, OR, NOT

szorzat: SELECT $R.A_1,R.A_2,\ldots,R.A_k,S.B_1,\ldots,S.B_l$ FROM R,S $\surd$

Multihalmazok-halmazok

- Az SQL alapértelmezésben nem tünteti el a többszörös sorokat, kivétel: UNION, INTERSECT, EXCEPT, ennél a háromnál eltűnnek az ismétlődések
- Ha el akarjuk tüntetni az ismétlődéseket: SELECT DISTINCT
- Ha a halmazműveleteknél mégsem akarom eltüntetni az ismétlődéseket: UNION ALL, EXCEPT ALL, INTERSECT ALL
- Nem (mindig) éri meg közben is törekedni arra, hogy ne legyen ismétlődés, elég a végén, mert:
- Az ismétlődés kiküszöbölése sok munka, mert rendezni kell az egész relációt hozzá

Aggregátumok

- Aggregátumok számolása: SUM, MIN, MAX, AVG, COUNT
- Az, hogy COUNT hogyan kezeli a többszörös sorokat, az rendszerfüggő. Ha biztosra akarunk menni: DISTINCT, ALL
- Lehetőségünk van bizonyos attribútumok értéke szerint csoportosítani az eredményt, és így aggregált sorokat képezni.

Erre az utóbbira példa a következő reláció:

MOZI	moziID	nev	varos	szekszam
	1	Corvin	Budapest	2500
	2	Elit	Sopron	300
	3	Sopron Plaza Megaflex	Sopron	2000
	4	Szindbád	Budapest	600
	5	Tabán	Budapest	200
	6	Uránia	Pécs	500

Aggregátumok

Csoportosítsunk a varos attribútum szerint:

MOZI	moziID	nev	varos	szekszam
	1	Corvin	Budapest	2500
	4	Szindbád	Budapest	600
	5	Tabán	Budapest	200
	6	Uránia	Pécs	500
	2	Elit	Sopron	300
	3	Sopron Plaza Megaflex	Sopron	2000

Képezzük minden városra a székszámok összegét:

MOZI	varos	ossz_szekszam
	Budapest	3300
	Pécs	500
	Sopron	2300

Példa 9: Mindez SQL-ben
SELECT varos, SUM(szekszam) AS ossz_szekszam FROM mozi GROUP BY varos

Példa 10: Az egyes városok legkisebb és legnagyobb mozijának mérete
SELECT varos, MIN(szekszam), MAX(szekszam) FROM mozi GROUP BY varos

Példák, ahol nincs csoportosítás:

Példa 11: A létező legnagyobb és a legkisebb székszám
SELECT MIN(szekszam), MAX(szekszam) FROM mozi

Példa 12: Az összes székszám
SELECT SUM(szekszam) FROM mozi

Aggregátumok

- Kiértékelés: Vesszük a FROM utáni relációk direkt szorzatát (egy reláció szerepelhet többször is a szorzatban, ha sorváltozókat adtunk meg hozzá), a WHERE feltételt teljesítő eseteket a GROUP BY szerint csoportosítjuk, majd kiszámoljuk minden csoportra az aggregátumot és kiírjuk.
- Amennyiben aggregátumokat képzünk a GROUP BY segítségével, akkor csak azokra az attribútumokra hivatkozhatunk közvetlenül a SELECT-ben, ami szerint csoportosítottunk. Ezen attribútumok értékei ugyanis egy aggregátumon belül jól meghatározottak. A többi attribútum az aggregátumon belül többféle értéket is felvehet. Ezért rájuk csak oszlopfüggvényeken keresztül hivatkozhatunk
- Lehet több oszlop szerint is GROUP BY, ekkor azok a sorok lesznek egy csoportban, ahol mindegyik GROUP BY után felsorolt oszlop értéke megegyezik.

- Lehet GROUP BY aggregátum nélkül is

  Példa 13:
  SELECT varos FROM mozi GROUP BY varos

  Kiírja az összes várost (pontosan egyszer), ahol van mozi. Ugyanaz, mint a SELECT DISTINCT varos FROM mozi

Feltétel a csoportokra — HAVING

A csoportosítással együtt tehetünk feltételt a csoportokra. Ebben az esetben csak azokra a csoportokra számolódik ki az aggregátum, amik a feltételnek eleget tesznek.

Példa 14: Azokra a városokra számolunk csak legkisebb és legnagyobb mozit, ahol van legalább 2 mozi
SELECT varos, MIN(szekszam), MAX(szekszam) FROM mozi GROUP BY varos HAVING COUNT(nev)>1

- a csoportra vonatkozó feltételt a HAVING kulcsszó vezeti be
- olyan feltételt írunk ide, ami csoportra vonatkozik (különben WHERE-be írnánk)
- csak GROUP BY-jal együtt használható
- a kiértékelés során a csoportosítás után minden egyes csoportra megnézzük a feltételt és eldobjuk azokat a csoportokat, amikre a feltétel nem áll és a maradékkal dolgozunk tovább

- HAVING megkerülhető, mindent, amit lehet HAVING-gel, lehet máshogy is (majd lesz erről szó az alkérdéseknél)

A hat alapkulcsszó

- SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY
- Ebben a sorrendben jönnek
- SELECT és FROM kell, a többi opcionális
- HAVING csak GROUP BY-jal

Alkérdések

- Az alkérdés eredménye mindig egy reláció, szintaxisa pedig a lekérdezés szintaxisával azonos.
- Tipikusan WHERE feltételében áll, ezáltal sokkal összetettebb kiválasztási feltételek jönnek létre, mint a relációs algebrában

Alkérdés FROM záradékban

A kiválasztáshoz használt relációk lehetnek alkérdés által származtatott relációk is.
Példa 15: A filmek címe, rendezője és a rendező filmjeinek száma
SELECT f1.cim, f1.rendezo,f2.filmszam FROM
film AS f1,
(SELECT rendezo, COUNT(*) AS filmszam FROM film GROUP BY rendezo) AS f2
WHERE f1.rendezo = f2.rendezo

Vigyázat! Itt nem jött létre f2 nevű reláció, csak annyi történik, hogy az f2 nevű sorváltozó befutja az alkérdés eredményéül kapott reláció sorait. Egyszer kiszámolódik az alkérdés és ennek eredményét használjuk a továbbiakban.

Alkérdés WHERE záradékban

Az alkérdés eredményét valamely attribútumok értékeivel hasonlítjuk össze a kiválasztáshoz.
Ezeknek az attribútumok számában meg kell egyezniük az alkérdés eredményének oszlopszámával.

- **Egyenlőség vizsgálata**
  Csak akkor lehetséges, ha az alkérdés egysoros relációt ír le (azaz az eredménye egyetlen érték vagy érték-vektor).
  Az egyenlőség fennáll, ha az adott attribútumok értékei megegyeznek az alkérdés által adott reláció megfelelő attribútumainak értékével.
  Szintaxis: SELECT ... WHERE (<attrib$_{11}$>, ..., <attrib$_{1n}$>) = (SELECT <attrib$_{21}$>, ..., <attrib$_{2n}$> FROM ...)

  A nem egyenlőség vizsgálatára a <> használandó.
  Példa 16: A legnagyobb mozik nevei
  SELECT nev FROM mozi WHERE mozi.szekszam = (SELECT MAX(szekszam) FROM mozi)
- **Tartalmazás vizsgálata**
  Több sort adó alkérdésre is értelmezett.
  A tartalmazás fennáll, ha a vizsgált attribútumok értéke megegyezik az alkérdés eredményének valamely sorával.
  Szintaxis: SELECT ... WHERE (<attrib$_{11}$>, ..., <attrib$_{1n}$>) IN (SELECT <attrib$_{21}$>, ..., <attrib$_{2n}$> FROM ...)
  A nem tartalmazás vizsgálatára a **NOT IN** használandó.
  Példa 17: A nem vetített filmek címe és rendezője
  SELECT cim, rendezo FROM film AS f1 WHERE f1.filmID NOT IN (SELECT v1.filmID FROM vetit AS v1)