Adatbázisok elmélete 8. előadás

Katona Gyula Y.
Budapesti Műszaki és Gazdaságtudományi Egyetem
Számítástudományi Tsz.
I. B. 137/b
kiskat@cs.bme.hu
http://www.cs.bme.hu/~kiskat

2005

QUEL

Sorkalkulus alapú.

Pl. Budapesti mozik nevei:

$\{t[2] \mid \text{MOZI}(t) \wedge t[3] = \text{'Budapest'}\}$

Meg kell mondani, hogy a sorváltozó melyik reláció sorain fut:

$R(t) \longleftrightarrow$ **range of** t **is** R

Hivatkozni kell a sorváltozó komponenseire:

$t[i] \longleftrightarrow$ **t.<*i*-edik attributum neve>**

Pl. $t[3] \longleftrightarrow t$.város

Lekérdezés:

retrieve (<lekérdezendő attribútumok> **) where** <feltétel>

QUEL példák

range of m **is MOZI**
range of f **is FILM**
range of v **is VETÍT**
range of u **is VETÍT**

- Budapesti mozik nevei
  retrieve (m.név **) where** m.város ='Budapest'
- Pénteken 16 órakor kezdődő filmek címe, rendezője
  retrieve unique (f.cím, f.rendező **)**
  where f.filmID=v.filmID **and** v.nap = 'péntek' **and** v.idő = '16:00'
- Pénteken nem vetített filmek címe:
  $\left\{t^{(3)}[2] \mid \text{FILM}(t) \wedge \nexists s \left(\text{VETÍT}(s) \wedge s[2] = t[1] \wedge s[3] = \text{'péntek'}\right)\right\}$
  QUEL-ben nincsenek igazi kvantorok, csak egy **any** nevű aggregátum!
  any()=1, ha van olyan sor, ami kielégíti a feltételt és =0, ha nincs

  retrieve (f.cím **)**
  where any (f.filmID **where** f.filmID=v.filmID **and** v.nap = 'péntek'**)**=0

- Pénteken és szombaton is vetített filmek címe
  retrieve unique (f.cím **)**
  where f.filmID=v.filmID **and** $v1$.filmID=v.filmID **and**
  v.nap = 'péntek' **and** $v1$.nap = 'szombat'

Lehet beszúrni és törölni:

retrieve into FILM1 **unique** f.cím
retrieve into FILM2 **unique** f.cím **where** f.filmID=v.filmID **and** v.nap='péntek'
range of $f1$ **is** FILM1
range of $f2$ **is** FILM2
delete $f1$ **where** $f1$.filmID=$f2$.filmID
retrieve FILM1

QBE

- Oszlopkalkulus alapú lekérdezések, kétdimenziós
  - A lekérdezés elemei változókkal és konstansokkal kitöltött sablon(ok)
    - jelölések

változó	aláhúzott változónév
konstans	nem aláhúzott érték
egyszer említett változó	üres cella
kimenetre kerülő attribútum	**P.** prefix

    - Példa: Budapesti mozik nevei:

MOZI	moziID	név	város
	1	**P.**mozinév	Budapest

vagy

MOZI	moziID	név	város
		P.	Budapest

  - Összetett lekérdezések is lehetségesek (használatukkor az azonos nevű változók illesztése történik meg).
    - használható több soros sablon (ekkor a kiértékeléskor mindegyik sornak egy-egy futó oszlopváltozó fog megfelelni, és ha illeszkedés van, akkor megtörténik a kiírás)
    - használható több sablon (kiértékelés hasonlóan, mint a többsoros kérdésnél, csak az oszlopváltozók nem ugyanazon reláció sorait futják be)

  - A kiválasztás feltételeinek megadása
    - Az egyenlőség konstanshoz való illesztéssel vizsgálható (mint az előbb),
    - egyéb egyszerű relációkhoz a ¬ =, >, <, >=, <, <= operátorok használhatók,
    - összetett feltételeket (pl. két változó közt a < relációt) külön feltételsablon megadásával lehet vizsgálni.
- További nyelvi elemek
  - mintaillesztés
  - aritmetika
  - kimenet rendezése
  - csoportosítás
  - aggregátumok kezelése
  - reláció tranzitív lezártjának kezelése
  - adatmódosító műveletek
  - típusdefiníció, sémalétrehozás

Megjegyzés: dupla példányt kiirtja, azaz többszörös sorok nincsenek

QBE példák még

- Nem budapesti mozik nevei

MOZI	moziID	név	város
		P.	¬ = Budapest

- Pénteki és szombati kezdési időpontok

VETÍT	moziID	filmID	nap	idő
			P.péntek	**P.**
			P.szombat	**P.**

- Időpontok, amikor pénteken és szombaton is kezdődik film

VETÍT	moziID	filmID	nap	idő
			péntek	**P.**kezdes
			szombat	kezdes

- Pénteken vetített filmek adatai

FILM	filmID	cím	rendező
	1	**P.**	**P.**

VETÍT	moziID	filmID	nap	idő
		1	péntek	

- Azok a városok, ahol van legalább két mozi:

MOZI	moziID	név	város
	1		**P.**városnév
	2		városnév

CONDITIONS
1≠2

Az SQL nyelv

- Relációs nyelv, mint az eddigiek
- oszlopkalkulus jellegű, de némi sorkalkulusos beütéssel

Termékek, (amik szükségszerűen relációs nyelvet is tartalmaztak):

- IBM: System/R
- Relational Software: Oracle
- Relational Systems: Ingres
- Microsoft: SQL server 2000

Szabványok

- SQL89 (SQL1)
- SQL92 (SQL2, mi nagyrészt ezt nézzük most)
- SQL99 (SQL3, ebből is pár dolog, pl. triggerek, rekurzió)

Működő rendszerekben ezek verziói vannak (főleg SQL2).

Fontosabb utasítások

Data Definition Language:

- CREATE - séma létrehozása
- ALTER - séma módosítása
- DROP - séma törlése

Data Modification Language:

- INSERT - adatok beszúrása
- UPDATE - adatok módosítása
- DELETE - adatok törlése
- SELECT - adatok lekérdezése

Természetesen előbb mindig a sémát kell létrehozni, és utána dolgozhatunk vele, de most fordítva tárgyaljuk mert eddig a lekérdező nyelvekről volt szó.

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):

$$\pi_{<attrib_1>,\ldots,<attrib_n>}\sigma_{<kifejezés>}(<reláció_1> \times \cdots \times <reláció_m>)$$

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

Példa 2: A pénteken hétkor kezdődő filmek azonosítói
SELECT vetít.filmID FROM vetít WHERE vetít.nap="péntek" AND vetít.idő="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 WHEREfeltétel 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

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):

$$\pi_{<attrib_1>,\ldots,<attrib_n>}\sigma_{<kifejezés>}(<reláció_1> \times \cdots \times <reláció_m>)$$

Példa 3: A budapesti mozik azonosítói és nevei
SELECT mozi.moziID, mozi.név FROM mozi WHERE mozi.város='Budapest'

Példa 4: A pénteken hétkor kezdődő filmek azonosítói
SELECT vetít.filmID FROM vetít WHERE vetít.nap='péntek' AND vetít.idő='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 név AS Filmszínház,
város 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 5: A pénteken vetített filmek címei és rendezői (természetes illesztés)
SELECT cím, rendező FROM film, vetít WHERE vetít.filmID = film.filmID AND nap='péntek'

Példa 6: Azok a várospárok, ahol vannak azonos nevű mozik
SELECT m1.város, m2.város FROM mozi AS m1, mozi AS m2 WHERE m1.név = m2.név AND m1.város <> m2.város

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

SELECT DISTINCT m1.város, m2.város FROM mozi AS m1, mozi AS m2
WHERE m1.név = m2.név AND m1.város < m2.város

A WHERE kifejezés

Kifejezés felépítése:

- logikai műveletek: AND, OR, NOT
- összehasonlítás: =, <>, >=, <=, LIKE, BETWEEN
- aritmetikai műveletek: +, -, *, /, MOD, POWER, LN, SIN, COS, . . .
- karakterlánc műveletek, összehasonlítás: CONCAT (||), LENGTH, LOWER, SUBSRT, SOUNDEX,. . .
- halmazba tartozás: IN (halmaz), . . .
- 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 7: A 150 és 200 közötti azonosítójú mozik közül azok, amelyek B-vel kezdődő nevű városban vannak, és a nevük hárombetűs.

SELECT név FROM mozi WHERE moziID BETWEEN 150 AND 200 AND város LIKE 'B%' AND név LIKE '_ _ _'

Műveletek relációkkal

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

Unió (valamely eredményrelációban szereplő sorok):

- Szintaxis: <eredményreláció1> UNION <eredményreláció2>
- Példa 8: A pénteken vagy szombaton játszott filmek :
  (SELECT cím FROM film, vetít WHERE vetít.nap = 'péntek' AND film.filmID = vetít.filmID)
  UNION
  (SELECT cím FROM film, vetít WHERE vetít.nap = 'szombat' AND film.filmID = vetít.filmID)

  (nem hatékony!)

Metszet (mindkét eredményrelációban szereplő sorok):

- Szintaxis: <eredményreláció1> INTERSECT <eredményreláció2>
- Példa 9: A pénteken és szombaton is játszott filmek:
  (SELECT cím FROM film, vetít WHERE vetít.nap = 'péntek' AND film.filmID = vetít.filmID)
  INTERSECT
  (SELECT cím FROM film, vetít WHERE vetít.nap = 'szombat' AND film.filmID = vetít.filmID)

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

- Szintaxis: <eredményreláció1> MINUS <eredményreláció2>
- Példa 10: A pénteken igen, de szombaton nem játszott filmek:
  (SELECT cím FROM film, vetít WHERE vetít.nap = 'péntek' AND film.filmID = vetít.filmID)
  MINUS
  (SELECT cím FROM film, vetít WHERE vetít.nap = 'szombat' AND film.filmID = vetít.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: COUNT (DISTINCT <attribútum>), COUNT (ALL <attribútum>)
- Lehetőségünk van bizonyos attribútumok értéke szerint csoportosítani az eredményt, és így aggregált sorokat képezni.

Erre az utóbbira példa a következő reláció:

MOZI

moziID	név	város	székszám
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 város attribútum szerint:

MOZI

moziID	név	város	székszám
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

város	össz_székszám
Budapest	3300
Pécs	500
Sopron	2300

Példa 11: Mindez SQL-ben
SELECT város, SUM(székszám) AS össz_székszám FROM mozi GROUP BY város

Példa 12: Az egyes városok legkisebb és legnagyobb mozijának mérete
SELECT város, MIN(székszám), MAX(székszám) FROM mozi GROUP BY város

Példák, ahol nincs csoportosítás:

Példa 13: A létező legnagyobb és a legkisebb székszám
SELECT MIN(székszám), MAX(székszám) FROM mozi

Példa 14: Az összes székszám
SELECT SUM(székszám) 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 (aggregátumokon) 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 15:
SELECT város FROM mozi GROUP BY város

Kiírja az összes várost (pontosan egyszer), ahol van mozi.
Ugyanaz, mint a
SELECT DISTINCT város 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 16: Azokra a városokra számolunk csak legkisebb és legnagyobb mozit, ahol van legalább 2 mozi
SELECT város, MIN(székszám), MAX(székszám) FROM mozi GROUP BY város HAVING COUNT(név)>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)