SQL: Poizvedbe, Integritetne omejitve, Prožilci

Iztok Savnik

Primeri relacij

- Instance ta bel *Rezervacije* in *Mornarji* bodo uporabljane v primerih.
- Če bi ključ tabele *Rezervacije* vseboval samo *mid* in *lid*, kakšen pomen bi imela relacija?

R1

mid	lid	dan
22	101	10/10/96
58	103	11/12/96

S1

mid	mime	ocena	star
22	novak	7	45.0
31	kranjc	8	55.5
58	petelin	10	35.0

S2

mid	mime	ocena	star
٢٨	volk	٩	٣٥.٠
٣١	kranjc	٨	٥٥.٥
٤٤	jauk	٥	٣٥.٠
٥٨	petelin	١٠	٣٥.٠

Osnovna SQL poizvedba

```
SELECT    [DISTINCT] seznam-izbire
FROM      seznam-relacij
WHERE     pogoj-izbire
```

- *seznam-relacij* Seznam relacij (lahko z uporabo spremenljivk).
- *seznam-izbire* Seznam atributov, ki so rezultat poizvedbe ali shema tabele -rezulata.
- *pogoj-izbire* Logični pogoj (Atr *op* const, Atr1 *op* Atr2, kjer je *op* eno izmed $<, >, =, \leq, \geq, \neq$) ; primerjave so povezane z logičnimi operacijami AND, OR in NOT.
- DISTINCT je opcijska ključna beseda, ki pove da naj se iz rezulatata odstranijo duplikati. Privzeto duplikati *niso* odstranjeni!

“Konceptualna” evaluacijska strategija

- Pomen SQL poizvedbe se lahko izrazi z naslednjo strategijo evaluacije:
  - Izračunaj Kartezijski produkt *seznama-relacij*.
  - Izloči tiste n-terice, ki ne izpolnjujejo *pogoja-izbire*.
  - Izloči atribute, ki niso v *seznama-izbire*.
  - Če je specificiran DISTINCT potem se izločijo duplikati.
- Opisana strategija je zelo verjetno najmanj učinkovit način evaluacije poizvedbe!
- Optimizator bo poiskal najbolj učinkovito strategijo, ki izračuna *isti odgovor.*

Primer konceptualne evaluacije

```
SELECT  S.mime
FROM    Mornarji M, Rezervacije R
WHERE   M.mid=R.mid AND R.lid=103
```

(mid)	mime	ocena	star	(mid)	lid	dan
22	novak	7	45.0	22	101	10/10/96
22	novak	7	45.0	58	103	11/12/96
31	kranjc	8	55.5	22	101	10/10/96
31	kranjc	8	55.5	58	103	11/12/96
58	petelin	10	35.0	22	101	10/10/96
58	petelin	10	35.0	58	103	11/12/96

Spremenljivke

- Potrebne so samo v primeru, da ista relacija uporablja dvakrat. Prejšnje vprašanje se lahko napiše tudi na sledeč način:

```
SELECT  M.mime
FROM    Mornarji M, Rezervacije R
WHERE   M.mid=R.mid AND lid=103
```

ALI

```
SELECT  mime
FROM    Mornarji, Rezervacije
WHERE
Mornarji.mid=Rezervacije.mid
AND lid=103
```

Dober stil: Vedno uporabljaj spremenljivke !

Poišči mornarje, ki so rezervirali vsaj eno ladjo

```
SELECT  M.mid
FROM  Mornarji M, Rezervacije R
WHERE  M.mid=R.mid
```

- Kaj se zgodi, če dodamo DISTINCT ?
- Kako vpliva zamenjava *M.mid* z *M.mime* na izvajanje SELECT stavka? Kaj če dodamo DISTINCT ?

Izrazi in nizi

```
SELECT  M.star, star1=M.star-5, 2*M.star AS star2
FROM  Mornarji M
WHERE  M.mime LIKE 'B_%N'
```

- Ilustracija uporabe aritmetičnih izrazov in ujemanja vzorcev pri nizih: *poišči trojice (starost mornarjev in dva polja opisana z izrazi) za vse mornarje katerih ime se začne z B, konča z N in vsebuje vsaj tri znake.*
- AS in = sta dva načina poimenovanja polj v rezultatu.
- LIKE se uporablja za primerjanje nizov. `_' pomeni katerikoli znak in `%' pomeni 0 ali več poljubnih znakov.

Poišči id-je mornarjev, ki so rezervirali rdečo ali zeleno ladjo

- UNION: Uporablja se za izračun unije dve *unija-kompatibilnih* množic n-teric (ki so rezultat SQL poizvedbe).
- Kaj dobimo, če zamenjamo OR z AND v prvi verziji poizvedbe?
- Uporaba EXCEPT (Kaj dobimo, če UNION zamenjamo z EXCEPT?)

```
SELECT  M.mid
FROM  Mornarji M, Ladje L, Rezervacije R
WHERE  M.mid=R.mid AND R.lid=L.lid
AND (L.barva='rdeca' OR L.barva='zelena
```

```
SELECT  M.mid
FROM  Mornarji M, Ladje L,
Rezervacije R
WHERE  M.mid=R.mid AND  L.lid=R.lid
            AND L.barva='rdeca'
UNION
SELECT  M.mid
FROM  Mornarji M, Ladje L,
Rezervacije R
WHERE  M.mid=R.mid AND L.lid=R.lid
            AND L.barva='zelena'
```

Poišči id-je mornarjev, ki so rezervirali rdečo in zeleno ladjo

- INTERSECT: lahko uporabljamo nad *unija-kompatibilnimi* množicami n-teric.
- Vključena je v SQL/92 standard; nekateri sistemi operacije ne podpirajo.

```
SELECT  M.mid
FROM  Mornarji M, Ladje L1, Rezervacije R1,
          Ladje L2, Rezervacije R2
WHERE  M.mid=R1.mid AND R1.lid=L1.lid
  AND  M.mid=R2.mid AND R2.lid=L2.lid
  AND (L1.barva='rdeca'
  AND L2.barva='zelena')
```

Ključ!

```
SELECT  M.mid
FROM  Mornarji M, Ladje L, Rezervacije R
WHERE  M.mid=R.mid AND L.lid=R.bid
            AND L.barva='rdeca'
INTERSECT
SELECT  M.mid
FROM  Mornarji M, Ladje L, Rezervacije R
WHERE  M.mid=R.mid AND L.lid=R.lid
            AND L.barva='zelena'
```

Vgnezdena vprašanja

Poišči imena mornarjev, ki so rezervirali ladjo #103:

```
SELECT M.mime
FROM Mornarji M
WHERE M.mid IN (SELECT R.mid
                FROM Rezervacije R
                WHERE R.lid=103)
```

- Zelo izrazna lastnost SQL: stavek WHERE lahko vsebuje SQL poizvedbo !
  - Kot tudi stavka FROM an HAVING.
- Mornarji, ki niso rezervirali ladje #103: NOT IN.
- Semantika vgnezdenih poizvedb:
  - Vgnezdene zanke: *Za vsakega mornarja, preveri pogoj poizvedbe, ki vsebuje vgnezdeno poizvedbo.*

Vgnezdena vprašanja (2)

Poišči imena mornarjev, so rezervirali ladjo #103:

```
SELECT M.mime
FROM Mornarji M
WHERE EXISTS (SELECT *
              FROM Rezervacije R
              WHERE R.lid=103 AND M.mid=R.mid)
```

- EXISTS: primerjava množice s prazno množico.
- Vgnezdeno vprašanje se izvede za vsakega mornarja.
- Če je uporabljen UNIQUE in je * zamenjana z *R.lid*, potem iščemo mornarje, ki imajo največ eno rezervacijo ladje #103. (UNIQUE preveri obstoj duplikatov; * pomeni vse atribute. Zakaj moramo zamenjati * z *R.lid*?)

Operacije za primerjanje množic

- Spoznali smo že IN, EXISTS in UNIQUE. Obstajajo še NOT IN, NOT EXISTS in NOT UNIQUE.
- Na voljo so še: *op* ANY, *op* ALL, *op* IN
- Poišči mornarje kateri imajo oceno večjo od vseh mornarjev z imenom “miha”:

$>,<,=,\geq,\leq,\neq$

```
SELECT *
FROM Mornarji M
WHERE M.ocena > ANY (SELECT M2.ocena
                     FROM Mornarji M2
                     WHERE M2.mime='miha')
```

Poizvedbe z INTERSECT : uporaba IN

Poišči id-je mornarjev, ki so rezervirali rdečo in zeleno ladjo:

```
SELECT  M.mid
FROM  Mornarji M, Ladje L, Rezervacije R
WHERE  M.mid=R.mid AND R.lid=L.lid AND L.barva='rdeca'
    AND M.mid IN  (SELECT  M2.mid
                   FROM  Mornarji M2, Ladje L2, Rezervacije R2
                   WHERE  M2.mid=R2.mid AND R2.lid=B2.lid
                      AND  L2.barva='zelena')
```

- Podobno, EXCEPT poizvedbe lahko prepišemo z uporabo NOT IN.
- Iskanje *imen* (ne *mid*) Mornarjev, ki so rezervirali rdečo in zeleno ladjo samo zamenjaj *M.mid* z *M.mime* v stavku SELECT. (Kaj je z INTERSECT poizvedbo?)

Deljenje v SQL

Poišči mornarje, ki so rezervirali vse ladje.

- Težja pot; brez EXCEPT:

(1)

```
SELECT  M.mime
FROM  Mornarji M
WHERE  NOT EXISTS
          ((SELECT  L.lid
            FROM  Ladje L)
           EXCEPT
            (SELECT  L.lid
             FROM  Rezervacije R
             WHERE R.mid=M.mid
                AND R.lid=L.lid))
```

(2)

```
SELECT  M.mime
FROM  Mornarji M
WHERE  NOT EXISTS  (SELECT  L.lid
                    FROM  Ladje L
                    WHERE  NOT EXISTS  (SELECT  R.lid
                                        FROM  Rezervacije R
                                        WHERE  R.lid=L.lid
                                           AND R.mid=M.mid))
```

Mornarji tako, da ...

ne obstaja ladja brez da nebi ...

obstajala rezervacija M ladje L

Agregacijske operacije

- Pomembna razširitev relacijske algebre.

COUNT (*)
COUNT ([DISTINCT] A)
SUM ([DISTINCT] A)
AVG ([DISTINCT] A)
MAX (A)
MIN (A)

single column

```
SELECT  COUNT (*)
FROM  Mornarji M
```

```
SELECT  AVG (M.star)
FROM  Mornarji M
WHERE  M.ocena=10
```

```
SELECT  M.mime
FROM  Mornarji M
WHERE  M.ocena= (SELECT  MAX(M2.ocena)
                 FROM  Mornarji M2)
```

```
SELECT  COUNT (DISTINCT M.ocena)
FROM  Mornarji M
WHERE M.mime='miha'
```

```
SELECT  AVG ( DISTINCT M.star)
FROM  Mornarji M
WHERE  M.ocena=10
```

Poišči imena in starost najstarejšega mornarja (-ev)

- Prva poizvedba ni legalna! (Razlog malce kasneje, ko si bomo ogledali GROUP BY.)
- Tretja poizvedba je ekvivalentna drugi in je legalna v okviru SQL/92 standarda; ni podprta v nekaterih sistemih.

```
SELECT  M.mime, MAX (M.star)
FROM  Mornarji M
```

```
SELECT  M.mime, M.star
FROM  Mornarji M
WHERE  M.star =
          (SELECT  MAX (M2.star)
           FROM  Mornarji M2)
```

```
SELECT  M.mime, M.star
FROM  Mornarji M
WHERE  (SELECT  MAX (M2.star)
          FROM  Mornarji M2)
          = M.star
```

Motivacija za grupiranje

- V prejšnjih primerih so se agregacijske operacije izvajale nad celotnimi tabelami. Včasih potrebujemo kaj izračunati nad *skupinami* n-teric.
- Primer: P*oišči starost najmlajših mornarjev za vsako oceno.*
  - V splošnem ne vemo koliko ocen obstaja in kakšne so vrednosti ocene.
  - Recimo, da vemo da so vrednosti ocene od 1 do 10; lahko napišemo 10 vprašanj kot je naslednje: ☺

For $i = 1, 2, \ldots, 10$:

```
SELECT  MIN (M.star)
FROM  Mornarji M
WHERE  M.ocena = i
```

Poizvedbe z GROUP BY in HAVING

```
SELECT      [DISTINCT]  seznam-izbire
FROM        seznam-relacij
WHERE       pogoj-izbire
GROUP BY    seznam-skupine
HAVING      pogoj-skupine
```

- *seznam-izbire* vsebuje: (i) imena atributov (ii) izraze z agregacijskimi operacijami (npr., MIN (*M.star*)).
  - seznam-izbire (i): mora biti podmnožica *seznama-skupine*. Intuitivno, predstavlja vsaka n-terica rezultata skupino n-teric.
  - *Skupina* je množica n-teric, ki ima isto vrednost vseh atributov iz *seznama-skupine*.

“Konceptualna” evaluacija

- Najprej izračunamo kartezijski produkt relacij iz *seznam-relacij*.
- Izločimo n-terice, ki ne izpolnjijejo pogoja *pogoj-izbire*.
- Izločimo nepotrebne atribute.
- Preostale n-terice se razvrstijo v skupine glede na vrednost atributov iz *seznama-skupine*.
- Iz množice n-teric, ki predstavljajo skupine se izločijo tiste, ki ne zadoščajo *pogoju-skupine.* Izrazi v *pogoju-skupine* morajo imeti *eno vrednost za celotno skupino!*
- Atributi v *pogoju-skupine* so bodisi argumenti v agregacijskih funkcijah ali pa se pojavijo v *seznamu-skupine*.
- Ena n-terica se generira za eno od izbranih skupin.

Poišči starost najmlajšega mornarja, ki je star več kot 18, v skupinah, ki pripadajo ocenenam in vsebujejo vsaj dva takšna mornarja.

```
SELECT  M.ocena,  MIN (M.star)
            AS minstar
FROM  Mornarji M
WHERE  M.star >= 18
GROUP BY  M.ocena
HAVING  COUNT (*) > 1
```

Odgovor:

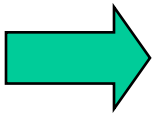
ocena	minstar
3	25.5
7	35.0
8	25.5

Sailors instance:

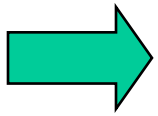
mid	mime	ocena	star
22	novak	7	45.0
29	pevec	1	33.0
31	kranjc	8	55.5
32	andrej	8	25.5
58	petelin	10	35.0
64	tom	7	35.0
71	kos	10	16.0
74	tom	9	35.0
85	miha	3	25.5
95	janez	3	63.5
96	egon	3	25.5

Poišči starost najmlajšega mornarja, ki je star več kot 18, v skupinah, ki pripadajo ocenam in vsebujejo vsaj dva takšna mornarja.

ocena	star
7	45.0
1	33.0
8	55.5
8	25.5
10	35.0
7	35.0
10	16.0
9	35.0
3	25.5
3	63.5
3	25.5



ocena	star
1	33.0
3	25.5
3	63.5
3	25.5
7	45.0
7	35.0
8	55.5
8	25.5
9	35.0
10	35.0

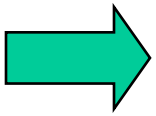


ocena	minstar
3	25.5
7	35.0
8	25.5

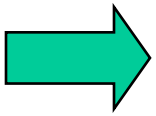
Poišči starost najmlajšega mornarja, ki je star več kot 18, v skupinah, ki pripadajo ocenenam, vsebujejo vsaj dva takšna mornarja in so vsi pod 60.

HAVING COUNT (*) > 1 AND EVERY (M.star <=60)

ocena	star
7	45.0
1	33.0
8	55.5
8	25.5
10	35.0
7	35.0
10	16.0
9	35.0
3	25.5
3	63.5
3	25.5



ocena	star
1	33.0
3	25.5
3	63.5
3	25.5
7	45.0
7	35.0
8	55.5
8	25.5
9	35.0
10	35.0



ocena	minstar
7	35.0
8	25.5

Kaj je rezultat spremembe EVERY v ANY?

Poišči starost najmlajšega mornarja, ki je star več kot 18, v skupinah, ki pripadajo ocenenam in vsebujejo vsaj dva mornarja stara od 18 do 60.

```
SELECT  M.ocena,  MIN (M.star)
            AS minstar
FROM  Mornarji M
WHERE  M.star >= 18
   AND M.star <= 60
GROUP BY  M.ocena
HAVING  COUNT (*) > 1
```

Odgovor

ocena	minstar
3	25.5
7	35.0
8	25.5

Mornarji

mid	mime	ocena	star
22	novak	7	45.0
29	pevec	1	33.0
31	kranjc	8	55.5
32	andrej	8	25.5
58	petelin	10	35.0
64	tom	7	35.0
71	kos	10	16.0
74	tom	9	35.0
85	miha	3	25.5
95	janez	3	63.5
96	egon	3	25.5

Za vsako rdečo ladjo poišči število rezervacij.

```
SELECT  L.lid,  COUNT (*) AS mcnt
FROM  Mornarji M, Ladja L, Rezervacije R
WHERE  M.mid=R.mid AND R.lid=L.lid AND L.barva='rdeca'
GROUP BY  L.lid
```

- Grupiranje po stiku treh relacij.
- Kaj dobimo, če umaknemo *L.barva='rdeca'* ?
- Kaj dobimo, če damo ta pogoj v stavek HAVING ?
- Kaj se zgodi, če umaknemo relacijo *Mornarji* in stik z *mid* ?

Poišči vse ocene za katere je povprečna starost minimalna po vseh ocenah

- Agregacijske operacije se ne morejo gnezditi!

```
SELECT  M.ocena
FROM  Mornarji M
WHERE  M.star =
          (SELECT  MIN (AVG (M2.star))  FROM Mornarji M2)
```

- Korektna rešitev z SQL/92):

```
SELECT  Temp.ocena, Temp.povprecje
FROM  (SELECT  M.ocena, AVG (M.star) AS povprecje
          FROM  Mornarji M
          GROUP BY  M.ocena) AS Temp
WHERE  Temp.povprecje = (SELECT  MIN (Temp.povprecje)
                                   FROM  Temp)
```

Null vrednosti

- Vrednosti polj so včasih *neznane* (npr., ocena ni bila vnešena).
  - SQL nudi posebno vrednost *null* za takšne situacije.
- Prisotnost vrednosti *null* vnaša v jezik več posledic:
  - Posebne operacije so potrebne za preverjanje če je dana vrednost enaka *null*.
  - Je *ocena>8* true ali false, ko je ocena enaka *NULL*?

    Kaj se zgodi z AND, OR in NOT operacijami?
  - Logika s tremi vrednostmi (true, false in *unknown*).
  - Pomen gradnikov mora biti definiran zelo pazljivo. (npr., WHERE ne izloči n-terice za katere je pogoj true.)
  - Nove operacije (*zunanji stiki*) mogoče/potrebne.

Integritetne omejitve (Pregled)

- IC opisuje pogoje, ki jih mora izpolniti vsaka legalna instanca relacije.
  - insert/delete/update, ki kršijo IC niso dovoljeni.
  - Lahko se uporabljajo zato, da zagotavljajo pravilen pomen podatov (npr., *mid* je ključ), ali onemogočijo nekonsistentnost (npr., *mime* mora biti niz in mora biti < 200)
- *Tipi IC*: Omejitve domen, primarni ključ, tuj ključ, splošne omejitve.
  - *Omejitve domen*: Vrednosti polj morajo biti pravilnega tipa.

Splošne omejitve

- Uporabne, ko so potrebne bolj splošne omejitve kot ključi.
- Poizvedbe uporabimo za izražanje omejitev.
- Omejitve lahko imenujemo.

```
CREATE TABLE   Mornarji
    ( mid  INTEGER,
    mime  CHAR(10),
    ocena  INTEGER,
    star  REAL,
    PRIMARY KEY  (mid),
    CHECK  ( ocena >= 1
        AND ocena <= 10 )
```

```
CREATE TABLE  Rezervacije
    ( mid  INTEGER,
    lid  INTEGER,
    day  DATE,
    PRIMARY KEY  (mid,bid,day),
    CONSTRAINT  BrezJezer
    CHECK  (`Jezero' <>
            ( SELECT  L.lime
            FROM  Ladje L
            WHERE  L.lid=lid)))
```

Omejitve preko večih relacij

- Če je relacija *Mornarji* prazna potem je lahko v relaciji *Ladje* karkoli.
- ASSERTION je prava rešitev; ni povezana z nobeno tabelo.

```
CREATE TABLE  Mornarji M
      ( mid  INTEGER,
      mime  CHAR(10),
      ocena  INTEGER,
      star  REAL,
      PRIMARY KEY  (mid),
      CHECK
      ( (SELECT COUNT (M.mid) FROM Mornarji M)
      + (SELECT COUNT (L.lid) FROM Ladje L) < 100 )
```

Število ladij skupaj s številom mornarjev je < 100

```
CREATE ASSERTION  MaliKlub
CHECK
( (SELECT COUNT (M.mid) FROM Mornarji M)
+ (SELECT COUNT (L.lid) FROM Ladje L) < 100 )
```

Prožilci

- Prožilec (trigger): procedura, ki se štarta v primeru, da se zgodi specifična sprememba v podatkovni bazi.
- Trije deli:
  - *Dogodek* - ki aktivira prožilec.
  - *Pogoj* - pove ali naj se procedura sproži.
  - *Akcija* - pove kaj naj naredi prožilec.

Prožilec: Primer (SQL:1999)

```
CREATE TRIGGER ShraniMladeMornarje
   AFTER INSERT ON MORNARJI
REFERENCING NEW TABLE NoviMornarji
FOR EACH STATEMENT
   INSERT
    INTO MladiMornarji(mid, mime, star, ocena)
    SELECT mid, mime, star, ocena
    FROM NoviMornarji N
    WHERE N.star <= 18
```

Pregled

- SQL je bil pomemben pri sprejetju relacijskega podatkovnega modela kot osnovo večine realnih sistemov; jezik je bolj naraven kot starejši proceduralni jeziki.
- Veliko načinov za zapis istega vprašanja; optimizator bi moral poiskati najbolj učinkovit plan evaluacije.
  - V realnosti morajo uporabniki zadosti dobro poznati sistem in podatke, da v robnih primerih zagotovijo optimalno evaluacijo.

Pregled (2)

- SQL je relacijsko kompleten; precej močnejši jezik kot so relacijska algebra.
- NULL se uporablja za neznane vrednosti polj; komplikacije z NULL vrednostmi.
- SQL omogoča specifikacijo bogate množice integritetnih omejitev.
- Prožilci se odzivajo na spremembe v bazi.