Baze de Date

Cap. 4. SQL. Subinterogări. Operații pe mulțimi

Textbook: Ramakrishnan, Gehrke, “Database Management Systems”, McGraw Hill, C. 15

2023 UPT Conf.Dr. Dan Pescaru

SQL Subinterogari

1. Subquery este o interogare în interiorul altei interogări. Alternativ: nested query
2. Subinterogările permit scrierea de interogări care selectează înregistrări bazat pe criterii dezvoltate la execuție (în timpul rulării interogării)

Ex.

SELECT listă_proiecție FROM tabelă

WHERE cond_cu_subinterogare

Categorii (I)

1. Există trei tipuri de subinterogări funcție de rezultatul calculat:
   A. Subinterogări care întorc un set (mulțime) de valori. Fiecare element din mulțime trebuie să fie o singură valoare scalară. Se pot folosi împreună cu operatorul IN sau în comparații utilizând modificatorii ANY sau ALL

Categorii (II)

B. Subinterogări care întorc o singură valoare scalară. Utilizabile în comparații simple, fără modificatori

C. Subinterogări care întorc un set (mulțime) de înregistrări. Utilizabilă cu ajutorul operatorului EXISTS pentru testare dacă mulțime este vidă

Subinterogări corelate și necorelate

1. Subinterogări necorelate – sunt independente de interogarea principală. Poate fi rezolvată separat, înainte de evaluarea interogării principale
2. Subinterogări corelate – depind de valori transmise din interogarea principală și ca atare nu pot fi evaluate independent

Subinterogări – reguli generale

1. O subinterogare SELECT este similară cu o interogare singulară
2. Subinterogările nu pot manipula intern rezultatul. Ca atare o subinterogare nu va include clauza ORDER BY pentru ordonare rezultat
3. Rezultatul exprimate prin lista de proiecție a subinterogării trebuie să corespundă ca și fel/tip scopului utilizării (conform celor prezentate la expunerea categoriilor după rezultat)

BD folosită în exemple (Port)

Tabelă Marinar

mid	nume	rang	varsta
22	Ion	7	45
31	Horatiu	1	33
58	Oana	8	54
71	Constantin	9	55

Tabelă Barca

bid	nume	culoare
101	Cleo	Albastra
102	Triton	Rosie
103	Poseidon	Verde

Tabelă Rezervare

rid	Mid	Bid	dată
22	31	101	10/03/2022
231	58	103	18/10/2022
71	31	101	22/10/2022

Subinterogări – operatorul IN

1. Subinterogările folosite împreună cu operatorul IN au forma generală:

   WHERE expresie [NOT] IN (subinterogare)

2. Ex.: toți marinarii care nu au rezervat barca 103

```
SELECT  m.mid, m.nume  FROM Marinar m
    WHERE  m.mid NOT IN (SELECT r.mid
                         FROM Rezervare r
                         WHERE r.bid = 103)
```

3. Evaluarea neoptimizată este echivalentă cu două cicluri suprapuse: pentru fiecare înregistrare din Marinar se va testa condiția cu subinterogare

Despre operatorul IN (I)

1. Pentru o mai bună înțelegere urmărim evaluarea optimizată pas cu pas
2. Nefiind corelată, subinterogarea se poate evalua independent de interogarea principală

```
SELECT r.mid
        FROM Rezervare r
        WHERE r.bid=103
```

3. Rezultatul este mulțimea { 58 }

Despre operatorul IN (II)

1. În continuare se va substitui rezultatul subinterogării ca și operand pentru operatorul IN și se va evalua apoi interogare principală

   SELECT m.mid, m.nume FROM Marinar m
   WHERE m.mid NOT IN { 58 }

2. Rezultatul va fi { Ion, Horatiu, Constantin }

Implementare INTERSECT folosind IN

1. Să se listeze toți marinarii care au rezervat și bărci albastre și bărci verzi.

```
SELECT DISTINCT m.mid, m.nume
    FROM Marinar m, Barca b, Rezervare r
    WHERE m.mid=r.mid AND r.bid=b.bid AND
          b.culoare= 'albastra' AND
          m.mid IN
              (SELECT m1.mid
                  FROM Marinar m1, Barca b1, Rezervare r1
                  WHERE m1.mid=r1.mid AND r1.bid=b1.bid
                                      AND b1.culoare= 'verde')
```

Subinterogări – operatori de comparare

1. Subinterogarea trebuie să returneze o singură valoare scalară. De exemplu o agregare fără grupare (discutată în următorul capitol)
2. Acesta poartă denumirea de subinterogare scalară deoarece returnează un singur rând și o singură coloană conținând doar o valoare scalară
3. Dacă va returna mai mult de o valoare se va genera o eroare.

Subinterogări – operatori de comparare

1. Ex. Găsiți toți marinarii care sunt mai in vârstă decât marinarul care a rezervat barca 103 în data de {10-18-2022}

```
SELECT * FROM Marinar WHERE
  varsta > (SELECT m.varsta
            FROM Marinar m INNER JOIN
                 Rezervare r ON m.mid=r.mid
            WHERE r.bid=103  AND
                       r.data='10-18-2022')
```

Subinterogări – modificatorii ALL și ANY

1. Modificatorii ALL și ANY pot schimba semantica operatorilor de comparare pentru a permite tratarea valorilor multiple din subinterogare
2. O formă generală de WHERE pentru aceștia este

   WHERE <expresie> <operator comparație>
   [ALL | ANY] (subinterogare)

Utilizare ALL

1. Modificatorul ALL schimbă operatorul mai mare (sau mai mic) pentru a însemna mai mare decât (sau mai mic decât) toate valorile returnate

Ex. Găsiți toți marinarii cu rangul cel mai mare

```
SELECT * FROM Marinar
    WHERE
        rang >= ALL ( SELECT rang
                      FROM Marinar)
```

Utilizare ANY

1. Mai puțin restrictiv decât ALL, ANY modifică >= (sau <=) pentru a însemna mai mare decât (sau mai mic decât) cel puțin una dintre valori

Ex. Găsiți toți marinarii cu rang mai mare decât un marinar numit Horatiu

```
SELECT m.* FROM Marinar m
   WHERE  m.rang > ANY
         (SELECT m1.rang FROM Marinar m1
               WHERE m1.nume='Horatiu')
```

IN, ANY, ALL

1. Echivalențe

- IN este echivalent cu =ANY
- NOT IN nu este echivalent cu <>ANY !!!
- NOT IN este echivalent cu <>ALL

Operatorul EXISTS

1. Clauza WHERE a interogării principale testează existența înregistrărilor returnate de subinterogare
2. Subinterogarea nu produce efectiv nici un rezultat; operatorul returnează TRUE sau FALSE

Ex. Găsiți toți marinarii care au rezervat cel puțin o barcă (=> interogări *corelate* !)

```
SELECT m.* FROM Marinar m WHERE
    EXISTS ( SELECT r.* FROM Rezervare r
                WHERE r.mid=m.mid)
```

DIVISION implementată cu EXISTS

1. Toți marinarii care au rezervat toate bărcile (subinterogări corelate multiple)

```
SELECT m.mid, m.nume FROM Marinar m
    WHERE NOT EXISTS ( SELECT b.bid
            FROM Barca b
            WHERE NOT EXISTS (
                    SELECT r.bid
                            FROM Rezervare r
                        WHERE r.bid = b.bid
                                AND r.mid = m.mid ))
```

Subinterogări în FROM

1. Subinterogările pot fi plasate și în clauza FROM. Este utilă la precalcularea agregărilor
2. Ex. Să se formeze perechi dintre bărcile nerezervate încă și marinari

```
SELECT m.*, b.*
    FROM Marinari m,
            (SELECT b1.bid, b1.nume
                    FROM Barca b1
                    WHERE NOT EXISTS (
                        SELECT r.*
                            FROM Rezervare r
                            WHERE r.bid=b1.bid)) b;
```

UNION

1. Combină rezultatele a două interogări și elimină duplicatele
2. Relațiile trebuie să fie Union compatibile
3. UNION ALL permite păstrarea duplicatelor

```
SELECT m1.*
    FROM Port1.Marinar m1
UNION
SELECT m2.*
    FROM Port2.Marinar m2;
```

UNION pentru FULL JOIN

1. UNION poate fi folosit pentru implementarea operatorului de full join in sistemele care nu îl oferă (ex. MySQL)

```
SELECT * FROM T1
    LEFT JOIN T2 ON T1.fk = T2.pk
UNION
SELECT * FROM T1
    RIGHT JOIN T2 ON T1.fk = T2.pk;
```

SQL DIFFERENCE (I)

1. Implementat de operatorul MINUS sau EXCEPT, dependent de sistem

2. Relațiile trebuie să fie union compatibile

3. Nu este implementat de MySQL (poate fi obținut cu subinterogări și operatorul NOT IN)

SQL DIFFERENCE (II)

1. Ex. Marinarii care au rezervat o barcă roșie dar nu și una verde:

```
SELECT DISTINCT m.mid, m.nume
    FROM Marinar m, Rezervare r, Barca b
    WHERE m.mid=r.mid AND r.bid=b.bid AND
                                b.culoare='rosie'
MINUS
SELECT DISTINCT m.mid, m.nume
    FROM Marinar m1, Rezervare r1, Barca b1
    WHERE m.mid=r.mid AND r.bid=b.bid AND
                                b.culoare= 'verde'
```