Fiola de SQL (24)

Numerotarea liniilor

– Marin Fotache

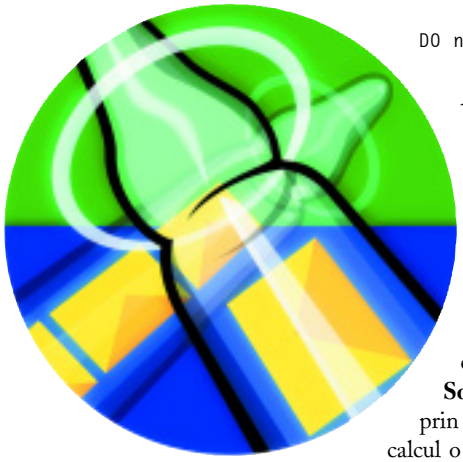
Iată, dragi cetitori, că am ajuns la fiola 24, ceea ce, fără prea multe rotunjiri, înseamnă doi ani bătuți pe muchie de *terapie extensivă* cu fiole de SQL. Cu această ocazie, îmi urez, mie, dar și fiolei, ca să nu mai spun de dvs., cei câțiva prea-răbdători amici în ale bazelor de date, cum spuneam, îmi urez și vă urez..., mă rog, n-are importanță ce, dar măcar s-ajungem la fiola de argint (nr. 50), că până la cea de aur o să vă plictisiți, vorba lui Topîrceanu, iremediabil...

Și dacă tot m-am apucat de numărat fiolele, mi-am zis că n-ar strica să cheltuim două-trei pagini pe numerotarea liniilor în rezultatul unei consultări. Este o problemă aparent simplă, și n-ar fi vorba de o aparență autentică dacă în spatele ei nu s-ar ascunde câteva dificultăți. Scoatem de la naftalină tabela `PERSONAL2` folosită în câteva episoade anterioare, dar, pentru a vă scuti navigarea prin arhiva NetReport-ului, vă prezint comanda de creare (în sintaxa PostgreSQL, apropiată de Oracle):

```
CREATE TABLE personal2 (
    marca INTEGER PRIMARY KEY,
    numepren VARCHAR(40),
    datanast DATE,
    compart CHAR(20) NULL,
    marcasef INTEGER NULL REFERENCES
                    personal2 (marca),
    saltarifar INTEGER
    )
```

Și tot pentru a vă scuti de un efort, vă reproduc, în aceeași sintaxă PostgreSQL, comenzile de populare a tabelei - vezi listing 1.

Iată și problema: să se afișeze angajații în ordine alfabetică, în rezultat atributele `Marca`, `Numepren` și `Compart` fiind precedate de numărul curent.



Două soluții Visual FoxPro

În VFP, numărul pe care îl primește automat fiecare înregistrare la inserarea sa în tabelă, extras prin funcția `RECNO()`, nu este de prea mare folos atunci când `SELECT`-ul conține clauza `ORDER BY`. Spre exemplu, interogarea următoare va conduce la rezultatul din figura 1.

```
SELECT RECNO() AS Nr, marca, numepren, compart  ;
FROM personal2 ;
ORDER BY numepren
```

O primă de variantă de rezolvare, conținută în script-ul din listing 2, salvează liniile ordonate într-un cursor - `cPersonal2` - care este apoi afișat folosindu-se `RECNO()`-ul „numărător“:

Nu e o varianta rea, ținând seama că nu avem prea multe la dispoziție. Ceea ce găsesc deranjant ține de copierea integrală tabelei în cursor. Altminteri, problema e rezolvată.

A doua soluție v-o prezint de dragul înfloriturilor pe care le conține. În plus, este și o avanpremieră la o iminentă fiolă dedicată SQL-ului dinamic. Ideea ar fi ca numărul curent să fie determinat pe baza unei funcții căreia, pentru generalitate, i se transmite și criteriul (criteriile) de ordonare). La prima execuție a funcției (prima linie din rezultat), se crează un vector în care sunt stocate, în ordinea criteriiilor de ordonare, valorile cheii primare a relației (`Marca`). La fiecare apel, funcția va retura poziția în vector a mărcii corespunzătoare angajatului „procesat“, poziție care este chiar numărul curent - vezi listing 3.

Pentru ca lucrurile să fie în regulă și la următorul apel al procedurii, după execuția frazei `SELECT` se șterge din memorie vectorul. Avantajul soluție ține de generalitatea ei. Dacă vrem să obținem rezultatul la problema propusă, nu avem decât a lansa în execuție:

```
DO numar WITH 'numepren'
```

Putem formula însă și un criteriu de ordonare mai sofisticat, de genul:

```
DO numar WITH 'compart ASC, numepren DESC'
```

Atenție, însă, numărul înregistrărilor din tabelă nu poate depăși dimensiunea unui vector (implicit 65000) !

Trei soluții PostgreSQL 7.1

PostgreSQL-ul nu prezintă funcții sau pseudo-coloane de genul `RECNO()` din VFP sau `ROWNUM`, `ROW_NUMBER()` din Oracle (după cum vom vedea cât de curând). Aceasta nu înseamnă că problema nu are rezolvare.

Soluția 1. Prima dintre cele trei variante se distinge prin faptul că este destul de complicată și că nu ia în calcul o facilitate pe care o valorifică soluția a doua. Ne folosim de aceasta și pentru a exemplifica o primă funcție (banală) PostgreSQL, ținând seama de sărăcia bilbliografiei românești în acest sens. Mai întâi, creăm o tabelă în care vom stoca numărul curent:

```
CREATE TABLE secventa ( nr INTEGER ) ;
```

apoi îi inserăm unica linie:

```
INSERT INTO secventa VALUES (0) ;
```

Atributul `Nr` al tabelei SECVENTA va fi incrementat și extras printr-o funcție simplă scrisă în limbajul „nativ“ (alături de C) al PostgreSQL-ului care este SQL:

Listing 1. Popularea tabelei PERSONAL2 (sintaxa PostgreSQL 7.1)

```
INSERT INTO personal2 VALUES (1,'ANGAJAT 1', CAST
   ('01-07-1962' AS DATE), NULL, NULL, 20000000)  ;
INSERT INTO personal2 VALUES (2,'ANGAJAT 2', CAST
   ('01-07-1977' AS DATE),'FINANCIAR',NULL, 15000000) ;
INSERT INTO personal2 VALUES (3,'ANGAJAT 3', CAST
   ('01-07-1962' AS DATE),'FINANCIAR',NULL, 2000000) ;
INSERT INTO personal2 VALUES (4,'ANGAJAT 4', CAST
   ('01-07-1977' AS DATE),'MARKETING',NULL, 2000000) ;
INSERT INTO personal2 VALUES (5,'ANGAJAT 5', CAST
   ('01-07-1965' AS DATE),'MARKETING',NULL, 2000000) ;
INSERT INTO personal2 VALUES (6,'ANGAJAT 6', CAST
   ('01-07-1965' AS DATE),'MARKETING',NULL, 1500000) ;
INSERT INTO personal2 VALUES (7,'ANGAJAT 7', CAST
   ('01-07-1966' AS DATE),'FINANCIAR ',NULL, 1500000) ;
INSERT INTO personal2 VALUES (8,'ANGAJAT 8', CAST
   ('01-07-1966' AS DATE),'CONTABILITATE',NULL, 150000) ;
INSERT INTO personal2 VALUES (9,'ANGAJAT 9', CAST
   ('01-07-1966' AS DATE),'CONTABILITATE',NULL, 2000000);
INSERT INTO personal2 VALUES (10,'ANGAJAT 10', CAST
   ('01-07-1962' AS DATE),'CONTABILITATE',NULL, 2500000);
INSERT INTO personal2 VALUES (11,'ANGAJAT 11', CAST
   ('01-07-1962' AS DATE),'FINANCIAR',NULL, 1500000) ;
COMMIT ;
```

```
CREATE FUNCTION increment2 () RETURNS int4 AS '
   UPDATE secventa SET nr = nr + 1 ;
   SELECT nr FROM secventa ;
' LANGUAGE 'sql' ;
```

Înainte de lansarea frazei SELECT, e bine să ne asigurăm că numărul curent e re-inițializat:

```
UPDATE secventa SET nr = 0
```

Iată și fraza SQL finală ce apelează funcția de mai sus:

```
SELECT increment2() AS NrCrt, temp1.*
FROM
(SELECT marca, numepren, compart
     FROM personal2 ORDER BY numepren) temp1
```

Soluția beneficiază de o posibilitatea declarării unei fraze SELECT în clauza FROM a alteia, așa încât ordonarea e deja rezolvată, funcția INCREMENT2() fiind cea care adaugă 1 la valoarea curentă a atributului SECVENTA.Nr.

Figura 1. Afișarea eronată în VFP a numărului curent (al înregistrării)

NR	MARCA	NUMEPREN	COMPART
1	1	ANGAJAT 1	DIRECTIUNE
10	10	ANGAJAT 10	RESURSE UMANE
2	2	ANGAJAT 2	FINANCIAR
3	3	ANGAJAT 3	MARKETING
4	4	ANGAJAT 4	FINANCIAR
5	5	ANGAJAT 5	FINANCIAR
6	6	ANGAJAT 6	FINANCIAR
7	7	ANGAJAT 7	FINANCIAR
8	8	ANGAJAT 8	MARKETING
9	9	ANGAJAT 9	MARKETING

Listing 2. Prima soluție VFP de numerotare

```
SELECT marca, numepren, compart  ;
FROM personal2 ;
INTO CURSOR cpersonal2 ;
ORDER BY numepren

SELECT RECNO() AS Nr, * ;
FROM cpersonal2 ;
```

Soluția 2. Asemănător Oracle, în PostgreSQL se pot declara, în schema bazei de date, secvențe gestionabile automat, secvențe din care se pot extrage valorile imediat următoare (NEXTVAL), curente (CURVAL), sau se pot seta valorile curente (SETVAL). Astfel, comanda:

```
CREATE SEQUENCE seq_num
```

va crea o secvență cu numele SEQ_NUM, secvență prin care rezolvăm problema propusă astfel:

```
SELECT NEXTVAL('seq_num') AS NrCrt, marca, numepren, compart
FROM
   (SELECT * FROM personal2 ORDER BY numepren) temp1
```

Necazul care apare în utilizarea secvențelei ține de faptul că, la următoarea execuție de frazei SELECT, secvența își continuă netulburată incrementarea, așa că, înaintea execuției interogării, re-setăm secvența. Din păcate,

```
SELECT SETVAL ('seq_num', 0)
```

întoarce un mesaj de eroare, în sensul că inițializare presupune o valoare mai mare ca zero. Ceea nu e chiar o tragedie, deoarece se poate folosi varianta:

```
SELECT SETVAL ('seq_num', 1)
```

iar apoi:

```
SELECT NEXTVAL('seq_num') - 1 AS NrCrt, *
FROM
(SELECT marca, numepren, compart
    FROM personal2 ORDER BY numepren) temp1
```

Listing 3. Procedura VFP - NUMAR.PRG

```
PARAMETER ordine_
SELECT numar(marca, ordine_) AS nr, ;
marca, numepren, compart ;
FROM personal2 ;
ORDER BY &ordine_

RELEASE vOrdine

FUNCTION numar
PARAMETER marca_, ord_
IF TYPE("vOrdine") = "U"
   SELECT marca FROM personal2 ;
INTO ARRAY vOrdine ORDER BY &ord_
ENDIF
RETURN ASCAN(vOrdine, marca_)
ENDFUNC
```

Soluția 3. Cea de-a treia soluție PostgreSQL este una insolită, deoarece folosește opțiunea de moștenire ce poate fi declarată la crearea unei tabele, opțiune prin care tabela creată preia atributele tabelei „străbun".

```
CREATE TABLE temp_personal2 (nr SERIAL) INHERITS (personal2)
```

Trucul rezolvării ține de includerea în tabela „descendent" - TEMP_PERSONAL2 - a tuturor atributelor din PERSONAL2 și adăugarea unui nou atribut - Nr - care este declarat de tip SERIAL tocmai pentru autoincrementare. Preluarea atributelor nu presupune și preluarea liniilor, așa că este nevoie de o comandă INSERT:

```
INSERT INTO temp_personal2  SELECT * FROM personal2 ORDER
   BY numepren
```

Odată populată, nu ne mai rămâne decât să afișăm conținutul tabelei TEMP_PERSONAL2:

```
SELECT *
FROM temp_personal2
```

În figura 2 se observă că, la moștenire, noul atribut este plasat în coada celorlalte, iar clauza SELECT are avea nevoie de enumerarea atributelor în ordinea dorită.

Ar mai fi de adăugat că, după ștergerea tabelei tampon, TEMP_PERSONAL2 (prin comanda DROP TABLE temp_personal2), în schema bazei rămâne definiția secvenței denumită automat de SGBD drept temp_personal2_nr_seq. Ștergerea acesteia se realizează prin comanda:

```
DROP SEQUENCE temp_personal2_nr_seq
```

Trei variante Oracle 8i/9i

Oracle are, ca orice SGBD profesional, câteva opțiuni care fac din problema formulată o simplă bagatelă. Înainte de aceasta, însă, pentru comparație cu soluția a doua din PostgreSQL, iată varianta bazată pe utilizarea unei secvențe. Mai întâi, se crează secvența :

```
CREATE  SEQUENCE seq_nr INCREMENT BY 1 START WITH 1
```

apoi consultarea care o folosește:

```
SELECT seq_nr.NEXTVAL AS NrCrt, temp1.*
FROM
(SELECT marca, numepren, compart
    FROM personal2 ORDER BY numepren) temp1
```

Figura 2. Afișarea conținutul tabelei „ moștenitor"

NR	MARCA	NUMEPREN	COMPART
1	1	ANGAJAT 1	RESURSE UMANE
2	10	ANGAJAT 10	CONTABILITATE
3	11	ANGAJAT 11	FINANCIAR
4	2	ANGAJAT 2	FINANCIAR
5	3	ANGAJAT 3	FINANCIAR
6	4	ANGAJAT 4	MARKETING
7	5	ANGAJAT 5	MARKETING
8	6	ANGAJAT 6	MARKETING
9	7	ANGAJAT 7	FINANCIAR
10	8	ANGAJAT 8	CONTABILITATE
11	9	ANGAJAT 9	CONTABILITATE

Soluția 1. Oracle furnizează pseudo-coloana ROWNUM, pe care am mai folosit-o în câteva fiole (ultima oară, în fiola 22). Ca și în cazul funcție RECNO() din VFP, nu putem scrie:

```
SELECT ROWNUM AS Nr, marca, numepren, compart
FROM personal2
ORDER BY numepren
```

deoarece rezultatul ar semăna cu cel din figura 1. Spre deosebire de VFP, acum putem rezolva problema ordonării printr-o subconsultare în clauza FROM:

```
SELECT ROWNUM AS Nr, TEMP1.*
FROM
    (SELECT marca, numepren, compart
    FROM personal2
    ORDER BY numepren) TEMP1
```

Soluția 2. Lucrurile n-au fost întotdeauna atât de ferice în Oracle. Până la 8*i*, în subconsultările declarate în clauza FROM a SELECT-ului principal nu putea să apară ORDER BY-ul. Așa ca era necesar un truc pe care eu unul l-am preluat de la Anunaya Shrivastava:

```
SELECT ROWNUM AS nr, marca, numepren, compart
FROM personal2, dual
WHERE numepren = DUMMY (+)
ORDER BY numepren
```

Trucul constă în joncționarea externă a PERSONAL2 cu tabela sistem DUAL (ce conține o sigură coloană - Dummy și o singură linie (valoare) - 'X'). Joncționarea externă va forța ordonarea implicită a ROWNUM. Atributul din clauza WHERE este cel după care se face ordonarea. Spre exemplu, dacă dorim ordonarea după valorile compartimentului, se folosește varianta:

```
SELECT ROWNUM AS nr, marca, numepren, compart
FROM personal2, dual
WHERE compart = DUMMY (+)
ORDER BY compart
```

Atunci când atributul de joncțiune nu este de tip caractere (așa cum e DUAL.dummy), este necesară folosirea funcției de conversie TO_CHAR:

```
SELECT ROWNUM AS nr, marca, numepren, compart
FROM personal2, dual
WHERE TO_CHAR(marca,'99999') = DUMMY (+)
ORDER BY marca
```

Soluția 3. Ultima variantă pe care v-o propun, care este valabilă și pentru DB2, folosește funcția așa-zis „analitică" ROW_NUMBER, pe care nu o vom mai discuta aici (vezi capitolul 7 din cartea *SQL. Dialecte DB2, Oracle și Visual FoxPro* scrisă de un autor):

```
SELECT ROW_NUMBER() OVER (ORDER BY numepren) AS nr,
    marca, numepren, compart
FROM personal2
```

Asta-i tot pentru luna aceasta. Cred că și prezenta fiolă poate fi un exemplu de modalitate în care, folosind o problemă simplă, ne putem complica viața suficient. Toate bune !

Marin Fotache este conferențiar dr. la Catedra de Informatică Economică, UAIC Iași, Facultatea de Economie și Administrarea Afacerilor. Poate fi contactat pe email la: fotache@uaic.ro. ■ 98