Fiola de SQL (27)

Procedularitate

– *Marin Fotache*

Luna aceasta vom ataca o problemă care a fost deja „atinsă“ (fără a fi însă „răpusă“) mai pe la începuturile tratamentului cu SQL (fiola nr.5 - septembrie 2000). Pornim de la trei tabele, AUTORI, CĂRȚI și AUTCĂRȚI, cu schema prezentată în listing 1, prin care gestionăm cărțile aflate într-o bibliotecă sau cărțile publicate de o editură sau, mai rău, scrise de profesorii unei catedre, facultăți etc.

Ca să spun, măcar azi, adevărul până la capăt, mi-am reamintit de problemă lucrând la noul site al facultății (aflat pe picior de publicare de luni bune), în care este necesară preluarea lucrărilor - cărți, articole, lucrări/comunicări prezentate la congrese/conferințe - și listarea lor, la nivel de autor, colectiv, catedră, facultate sau domeniu de interes. Dar, mai întâi, să umplem cele trei tabele cu înregistrările necesare verificării corectitudinii soluțiilor - listing 2.

Afișarea a maximum doi autori

O carte poate fi scrisă de mulți autori. Pe copertă, aceștia apar într-o anumită ordine, informație preluată în tabela AUTCĂRȚI de atributul Ordine. În principiu, scopul fiolei nr. 5 era de a obține o situație precum cea din figura 1, în care, pentru fiecare carte, se afișează ambii autori, dacă sunt numai doi, iar dacă numărul autorilor este mai mare sau egal cu trei, se afișează numai numele primului, urmat de nedreptul (pentru ceilalți) *s.a.*

Față de fiola cu pricina, am modificat un pic și modul de prezentare a rezultatului, dar și sintaxa soluției care e croită pe PostgreSQL:

Listing 1. Crearea celor trei tabele

```
CREATE TABLE autori (
  codaut NUMERIC(6) PRIMARY KEY,
  nume VARCHAR(20),
  prenume VARCHAR(20),
  tara  VARCHAR(30) DEFAULT 'Romania',
  url VARCHAR(100)
 ) ;

CREATE TABLE carti (
  isbn VARCHAR(16) NOT NULL PRIMARY KEY,
  titlu VARCHAR(500),
  editura VARCHAR(50),
  an NUMERIC(4)
 ) ;

CREATE TABLE autcarti (
  isbn VARCHAR(16) NOT NULL,
  codaut NUMERIC(6) NOT NULL,
  ordine NUMERIC(3),
  calitate VARCHAR(15) NOT NULL   CHECK (calitate IN
           ( 'Autor', 'Coordonator', 'Editor')),
PRIMARY KEY (isbn, codaut,  calitate),
FOREIGN KEY (codaut)  REFERENCES AUTORI(codaut),
FOREIGN KEY (isbn) REFERENCES CARTI(isbn)
) ;
```

```
SELECT ac1.autor ||  CASE WHEN ac3.ISBN IS NULL
THEN  COALESCE(', ' || ac2.autor,' ') ELSE  ' s.a.'  END AS
   autori,
      titlu, editura, an, ac1.ISBN
FROM
(SELECT autcarti.isbn, nume || ' ' || prenume AS autor,
   titlu, editura, an
FROM autcarti INNER JOIN autori ON
   autcarti.codaut=autori.codaut
      INNER JOIN carti ON autcarti.isbn=carti.isbn
WHERE ordine=1) ac1 LEFT OUTER JOIN
(SELECT isbn, nume || ' ' || prenume AS autor
 FROM autcarti INNER JOIN autori ON
   autcarti.codaut=autori.codaut
 WHERE ordine=2) ac2 ON ac1.ISBN=ac2.ISBN
LEFT OUTER JOIN
(SELECT isbn, nume || ' ' || prenume AS autor
 FROM autcarti INNER JOIN autori ON
   autcarti.codaut=autori.codaut
 WHERE ordine=3) ac3 ON ac1.ISBN=ac3.ISBN
```

Afișarea a... mai mulți autori

Ei bine, în cazul site-ului facultății, pentru a diminua (la jumătate) din nemulțumiri, încă de la început s-a pus problema afișării tuturor autorilor unei cărți, oricât de mulți ar fi dumnealor. Lucru necesar și pentru căutarea lucrărilor unui profesor ce a publicat și în echipe mai mari. Din acest moment a devenit clar că SQL-ul se va izbi frontal de problema numărul variabil, de la carte la carte, al autorilor. O propunere a fost ca, întrucât numărul nu poate fi mai mare de zece (cel puțin la facultatea noastră), să se extindă interogarea de mai sus, în loc de trei subconsultări și joncțiuni externe, folosindu-ne de 10. Soluția, însă, e atât de rușinoasă, încât am rugat redacția s-o publice numai pe site-ul Web revistei. E drept, rezultatul e corect, după cum arată figura 2.

Listing 2. Popularea cu înregistrări (listing trunchiat)

```
DELETE FROM autori ;
INSERT INTO autori VALUES (1,'Date','Chris','SUA',NULL);
...
DELETE FROM carti ;
INSERT INTO carti VALUES ('0-672-32260-9', 'PostgreSQL.
   Developer s Handbook', 'SAMS', 2002) ;
...
DELETE FROM autcarti ;
INSERT INTO autcarti VALUES ('0-672-32260-9', 2, 1,
   'Autor') ;
...
COMMIT;
```

Figura 1. O formă simplificată de afișare a autorilor

autori	titlu	editura	an	isbn
Date Chris	An Introduction to Database Systems	Addison-Wesley	2000	0-672-32260-9
Geschwinde Ewald, Schonig Hans Jurgen	PostgreSQL. Developer s Handbook	SAMS	2002	0-201-68419-5
Fotache Marin	SQL. Dialecte DB2, Oracle si Visual FoxPro	Polirom	2001	973-683-709-2
Fotache Marin s.a.	Visual FoxPro. Ghidul dezvoltarii aplicatiilor profesionale	Polirom	2002	973-683-889-7

Figura 2. Afișarea tuturor autorilor

autori	titlu	editura	an	isbn
Date Chris	An Introduction to Database Systems	Addison-Wesley	2000	0-672-32260-9
Geschwinde Ewald, Schonig Hans Jurgen	PostgreSQL. Developer s Handbook	SAMS	2002	0-201-68419-5
Fotache Marin	SQL. Dialecte DB2, Oracle si Visual FoxPro	Polirom	2001	973-683-709-2
Fotache Marin,Cretu Liviu, Brava Ioan, Strimbei Catalin	Visual FoxPro. Ghidul dezvoltarii aplicatiilor profesionale	Polirom	2002	973-683-889-7

Soluție procedurală PL/pgSQL pentru extragerea tuturor autorilor

Chiar dacă reușim să trecem rapid peste jenă, rămâne problema cărților cu număr foarte mare de autori, cum ar fi monografiile, pentru care ne e imposibil să stabilim o limită maximă a numărului celor ce au contribuit la scrierea lucrării. Așa că nu-i rău să apelăm niței la procedularitate. Și cum tot nu v-am răsfățat prea mult cu proceduri și funcții PostgreSQL, vom insista pe o variantă PL/pgSQL, limbajul procedural al PostgreSQL-ului. După cum îi zice și numele, asemănarea sa cu PL/SQL (limbajul procedural Oracle) este tulburătoare, deși e încă departe de finețurile acestuia din urmă.

Începem cu o funcție simplă - `f_numeaut` - care primește codul autorului și returnează, concatenate, numele și prenumele acestuia - vezi listing 3.

Una din deosebirile față de PL/SQL este că, pentru înlocuirea corpului unei funcții în schema bazei, nu este posibilă folosirea clauzei `CREATE OR REPLACE FUNCTION`, ci ștergerea prealabilă a acesteia. PL/pgSQL permite supraîncărcarea funcțiilor, așa că, la ștergere, trebuie specificat și tipul fiecărui argument al funcției:

```
DROP FUNCTION f_numeaut(NUMERIC)
```

Pentru parametrii de intrare se specifică numai tipul, nu și numele, problemă rezolvabilă prin declararea unui alias (în cazul nostru, `codaut_`) pentru fiecare parametru (`$1`). În corpul funcției, verificarea existenței măcar a unei linii care să satisfacă clauza `WHERE` a `SELECT`-ului se face simplu, prin testul `IF NOT FOUND...`

Odată creată, funcția poate fi invocată din orice frază SELECT:

```
SELECT codaut, f_numeaut(codaut) AS nume
FROM autori
```

Listing 3. Funcție ce primește codul autorului și furnizează numele și prenumele acestuia

```
CREATE FUNCTION f_numeaut (NUMERIC)  RETURNS  VARCHAR AS '
DECLARE
   codaut_ ALIAS FOR $1 ;
   numeaut_ VARCHAR(41) ;
BEGIN
   SELECT nume || '' '' || prenume
   INTO numeaut_
   FROM autori
   WHERE codaut = codaut_ ;

   IF NOT FOUND THEN
         RETURN '' '' ;
   ELSE
      RETURN numeaut_ ;
   END IF ;
END;
' LANGUAGE 'plpgsql' ;
```

Pe noi, însă, ne interesează ca aceasta funcție să fie apelată dintr-alta, mai sofisticată, căreia i se pasează ISBN-ul cărții, și care, pe baza tabelei `AUTCĂRȚI` și apelul funcției de mai sus, furnizează un șir de caractere ce reprezintă numele și prenumele tuturor autorilor cărții cu pricina. Funcția se numește f_autori și este prezentată în listing 4.

Ca și în PL/SQL, PL/pgSQL permite declarea tipului unei variabile (`isbn_`) prin specificarea atributului al cărui tip îl preia (`autcarti.isbn`). Însă `rec_autori` trebuie declarată explicit ca fiind de tip `RECORD` (echivalentul lui `ROWTYPE`). Cum PL/pgSQL nu se simte prea în largul lui cu cursoare, este necesară folosirea clauzei `FOR rec_autori IN SELECT...`, ceea ce rezolvă onorabil problema. Altminteri, prea de multe de adăugat nu sunt. Rezultatul din figura 2 se obține acum printr-o interogare de genul:

```
SELECT f_autori (isbn) AS autori, titlu, editura, an, isbn
FROM carti
```

Procedurile stocate în Oracle și MySQL

Just for the record, cum am auzit într-un film american, în listing-urile 5 și 6, disponibile pe situl Web al revistei, este prezentată versiunea PL/SQL a celor două funcții.

Și tot *just for the record*, cum am auzit în același film american, discuția nu este valabilă pentru MySQL. Deoarece nu există proceduri stocate. Așa că ne vedem la o versiune viitoare...

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

Listing 4. Funcție ce furnizează numele tuturor autorilor unei cărți date

```
CREATE FUNCTION f_autori (VARCHAR)  RETURNS  VARCHAR AS '
DECLARE
   isbn_ autcarti.isbn%TYPE ;
   sir_ VARCHAR(1000) :='''' ;
   rec_autori RECORD ;
BEGIN
   isbn_ := $1 ;
   FOR rec_autori IN
SELECT ordine, codaut FROM autcarti
WHERE isbn = isbn_ ORDER BY ordine
LOOP
            IF rec_autori.ordine > 1 THEN
              sir_ := sir_ || '', '' ;
      END IF ;
      sir_ := sir_ || f_numeaut(rec_autori.codaut)  ;
   END LOOP ;
   RETURN sir_ ;
END;
' LANGUAGE 'plpgsql' ;
```