Fiola de SQL (28)

O samă de balanțe contabile

– Marin Fotache

Nici nu se răci bine fiola cu nr. 24, în care aniversam doi ani de la începutul tratamentului cu SQL, și iată că trebuie să ne luăm adio. Din septembrie, *NetReport*-ul va avea un alt format, mai atractiv, în care fiola nu se regăsește, sau cel puțin nu în forma aceasta. Este posibil să se fi descoperit ceva fiole expirate, sau tratamentul să fi generat reacții alergice în rândul cititorilor. N-am să fiu patetic, deși despărțirea e o ocazie minunată pentru puseuri emoționale. Neavând o condiție filosofică prea elevată, realizez, totuși, că lucrurile lumești trebuie să aibă un început și un sfârșit. Așadar, iată fiola care ar fi mers și la rubrica *Ultima*...

Nu e nici un secret, vorba unui coleg, că se scot bani serioși din chestiuni plicticoase, neliniare și nerecursive. Or, ce exemplu este mai la îndemână pentru informaticieni decât contabilitatea ? Stufoasă, dar riguroasă, pe alocuri arogantă și rigidă, contabilitatea a constituit, fără îndoială, una dintre primele ținte ale informaticii de întreprindere, *supra* (sau *sub*) numită *informatică economică*. După terminarea facultății, generații întregi de informaticieni, teoreticieni sau „aplicativi", au pornit la drum cu grafuri orientate și lanțuri Markov și au sfârșit în neagra mizerie a fișelor de cont.

Multe aplicații economice pornesc de la documentele financiar-contabile ce trebuie preluate, prelucrate sau listate: fișe de magazie, fișe de mijloace fixe, fișe ale furnizorului/clientului, registru de casă, extras de cont, foaie de pontaj, mergându-se până la fișe de cont și balanțe de verificare. O bună parte dintre documentele financiar-contabile și operative au o formă balanțieră, în sensul că:

- sunt întocmite pentru fiecare operațiune/tranzacție sau interval calendaristic – zi, lună, an;
- înaintea de operațiune se înscrie stocul sau soldul inițial al operațiunii;
- după soldul inițial și suma operațiunii se afișează și stocul/soldul final.

Un asemenea gen de document constituie obiectul fiolei de luna aceasta. Inițial, aveam de gând să exemplific fișa de magazie, însă un registru de casă e mai interesant, câtă vreme se lucrează cu bani. Practic, pornim de la o situație precum cea din *figura 1*.

Listing 1. Popularea cu înregistrări (sintaxă PostgreSQL)

```
DELETE FROM registru_casa ;
INSERT INTO registru_casa VALUES (timestamp '2002-08-01 00:01',  'I', 2500000,
   NULL, 'Sold Initial') ;
INSERT INTO registru_casa VALUES (timestamp '2002-08-01 08:30',  'I', 500000,
   'Chitanta BG44556', 'Imputatie') ;
INSERT INTO registru_casa VALUES (timestamp '2002-08-01 09:30',  'P', 200000,
   'Ordin deplasare 456/1 aug', 'Deplasare Popescu Vasile') ;
INSERT INTO registru_casa VALUES (timestamp '2002-08-01 09:45',  'P', 1200000,
   'Disp. de plata-casierie nr.445', 'Diferenta de plata Popvici Viorel') ;
INSERT INTO registru_casa VALUES (timestamp '2002-08-01 12:05',  'I', 2000000,
   'Chitanta BG44556', 'Incasare factura 65788') ;
COMMIT ;
```

Figura 1. Un model (neautentic de registru de casă)

dataora	incasari	plata	document	explicatii
2002-08-01 00:01:00+03	2500000	0		Sold initial
2002-08-01 08:30:00+03	500000	0	Chitanta BG44556	Imputatie
2002-08-01 09:30:00+03	0	200000	Ordin deplasare 456/	Deplasare Popescu Vasile
2002-08-01 09:45:00+03	0	1200000	Disp. de plata-casie	Diferenta de plata Popvici Viorel
2002-08-01 12:05:00+03	2000000	0	Chitanta BG44556	Incasare factura 65788

Figura 2. Forma balanțieră ce se dorește a fi obținută

dataora	sold_initial	incasari	plata	sold_final	document	explicatii
2002-08-01 00:01:00+03		2500000	0			Sold initial
2002-08-01 08:30:00+03	2500000	500000	0	3000000	Chitanta BG44556	Imputatie
2002-08-01 09:30:00+03	3000000	0	200000	2800000	Ordin deplasare 456/	Deplasare Popescu Vasile
2002-08-01 09:45:00+03	2800000	0	1200000	1600000	Disp. de plata-casie	Diferenta de plata Popvici Viorel
2002-08-01 12:05:00+03	1600000	2000000	0	3600000	Chitanta BG44556	Incasare factura 65788

Tabela care poate gestiona aceste date primare poate fi creată prin comanda `CREATE TABLE` astfel (sintaxa PostgreSQL 7.2):

```
CREATE TABLE registru_casa  (
dataora TIMESTAMP PRIMARY KEY DEFAULT
    CURRENT_TIMESTAMP,
    tip CHAR(1) NOT NULL DEFAULT 'P'
    CHECK (tip IN ('P', 'I')),
    suma NUMERIC(14),
    document VARCHAR(20),
    explicatii VARCHAR(40)
    )
```

Atributul `DataOra` desemnează momentul operațiunii, `Tip` indică dacă operațiunea este o încasare ('I') sau o plată ('P'), `Suma` reprezintă valoarea încasării/plății, `Document` - tipul seria și numărul documentului justificativ a încasării/plății, iar `Explicații` furnizează câteva detalii cu privire la încasarea/plata curentă. În PostgreSQL, pentru consemnarea cu exactitate a momentului producerii unei operațiuni, tipul de dată indicat este `TIMESTAMP`, în Oracle `DATE` (acesta conține și ora, minutul, secunda și fracțiunile de secundă), iar în Visual FoxPro `DATETIME`. Atenție, însă, VFP face nazuri: chiar dacă în comandă `CREATE` atributul `DataOra` e declarat `DATETIME`, în baza de date va fi de tip `DATE` !!! Singura soluție de remediere e cea „manuală" – de fapt *mouse*-ală.

Tot pe sintaxa PostgreSQL 7.2, *listing-ul 1* conține comenzile de populare a tabelei cu câteva înregistrări necesare verificării interogărilor.

Structura tabelei e un pic schimbată, însă obținerea raportului din *figura 1* e simplă (sintaxa Oracle/PostgreSQL):

```
SELECT dataora, CASE WHEN tip='I' THEN
   suma ELSE 0 END AS Incasari ,
CASE WHEN tip='P' THEN suma ELSE 0 END
   AS Plata, document, explicatii
FROM registru_casa ORDER BY dataora
```

Problema noastră ține de faptul că trebuie să (re)calculăm soldul după fiecare operațiune, iar soldul inițial al unei operațiuni este soldul final al operațiunii precedente, după cum se observă în *figura 2*. Or, calculul soldului inițial al unei operațiuni ca sold final al operațiunii precedente este un clenci interesant.

Soluție procedurală Visual FoxPro

Primele variante de rezolvare pe care le vom discuta sunt și cele mai neelegante, cel puțin prin lungimea lor și prin proceduralitare (un SQL-ist cu pretenții va strâmba din nas aproape întotdeauna la variantele procedurale – așa că vă rog să vă potriviți grimasa pentru cele ce urmează).

Programul VFP din *listing 2* se bazează pe o frază `SELECT` ce apelează două funcții, `f_soldinitial` și `f_soldfinal`, a căror utilitate reiese din chiar denumirea lor. Funcțiile sunt mai complicate decât ar trebui, deoarece VFP nu se comportă întotdeauna cum ne-am aștepta la execuția frazelor `SELECT`.

Cele două funcții jonglează cu două variabile publice, `sold_in` și `sold_fin`. Pentru a asigura corectitudinea rezultatului, funcției `f_soldinițial` i se pasează explicația operațiunii, Dacă aceasta descrie prima operațiune, înseamnă că avem de-a face cu valoarea soldului inițial, iar `sold_in` se reinițializează. Pentru toate celelalte linii, funcția returnează valoarea variabilei, care este actualizată în `f_soldfinal`. `F_soldfinal` prezintă doi parametri, `tip_` și `suma_`, în funcție de care se calculează `sold_fin` - soldul final al operații curente și `sold_in` - soldul inițial al operațiunii următoare.

Soluție procedurală Oracle

În PL/SQL variabilele publice pot fi gestionate cu ajutorul pachetelor. Orice variabilă ce apare în partea declarativă a unui pachet are regim public – vezi *listing 3* -, referirea sa realizându-se prin prefixarea cu numele pachetului, ca în funcțiile `f_sold_initial` și `f_sold_final`, prezentate în listing-urile 4 și 5.

Spre deosebire de logica soluției VFP, prezenta variantă este mai elegantă. Prima variabilă publică, `sold_in`, are aceeași funcționalitate, însă a doua, denumită `prima_operațiune`, conține momentul primei operațiuni din tabela `REGISTRU_CASĂ`. Funcția `f_sold_inițial` nu se mai bazează pe explicații, ci verifică dacă momentul producerii operațiunii curente coincide cu momentul primei operațiuni, caz în care variabila `sold_in` se reinițializează.

Cât despre funcția `f_sold_final`, nu sunt prea multe de adăugat, logica sa fiind identică funcției corespondente din VFP.

Fraza `SELECT` ce furnizează rezultatul dorit prin invocarea celor două funcții este:

```
SELECT dataora, f_sold_initial(dataora)
       AS sold_initial,
   CASE WHEN tip='I' THEN suma ELSE 0
        END AS Incasari,
   CASE WHEN tip='P' THEN suma ELSE 0
        END AS Plati,
    f_sold_final(tip, suma) AS
      Sold_Final, document, explicatii
FROM registru_casa
ORDER BY dataora
```

Soluție procedurală PostgreSQL

Neavând pachete și nici alt mecanism de lucru cu variabile publice, în PostgreSQL vom încerca o altă variantă de rezolvare. Funcția

Listing 2. Program pentru afișarea în VFP6 a soldului de casă după fiecare operațiune

```
PUBLIC sold_in, sold_fin
sold_in = 0
sold_fin = 0

SELECT dataora, f_soldinitial(explicatii) AS sold_initial, ;
   IIF(tip='I', suma, 000000000000000) AS Incasari, ;
   IIF(tip='P', suma, 000000000000000) AS Plati, ;
   f_soldfinal(tip, suma) AS Sold_Final,  ;
   document, explicatii ;
FROM registru_casa ;
ORDER BY dataora

FUNCTION f_soldinitial
PARAMETER expl_
IF expl_ = 'Sold Initial'
   sold_in = 0
   RETURN 00000000000000
ELSE
   RETURN sold_in
ENDFUNC

FUNCTION f_soldfinal
PARAMETER tip_, suma_
IF tip_ = 'I'
   sold_fin = sold_in + suma
ELSE
   sold_fin = sold_in - suma
ENDIF
sold_in = sold_fin
RETURN sold_fin
ENDFUNC
```

Listing 3. Pachetul PL/SQL ce conține variabilele globale

```
CREATE OR REPLACE PACKAGE variabile_globale  IS
sold_in NUMERIC (16) := 0 ;
prima_operatiune registru_casa.dataora%TYPE := NULL ;
END;
```

Listing 4. Funcția F_SOLD_INITIAL (PL/SQL)

```
CREATE OR REPLACE FUNCTION f_sold_initial (
dataora_ registru_casa.dataora%TYPE )
    RETURN  registru_casa.suma%TYPE IS
BEGIN
    -- se verifica daca este initializata variabila
    -- ce contine operatiunea initiala (sold initial)
    IF variabile_globale.prima_operatiune IS NULL THEN
        SELECT MIN(dataora)
        INTO variabile_globale.prima_operatiune
        FROM registru_casa ;
    END IF ;

    -- se reseteaza variabila publica SOLD_IN daca
    -- linia curenta corespunde primeo operatiuni
    IF dataora_ = variabile_globale.prima_operatiune THEN
          variabile_globale.sold_in := 0 ;
    END IF ;
    RETURN variabile_globale.sold_in ;
END;
```

f_sold_inițial va însuma, pentru o operațiune dată, toate operațiunile precedente (cu semnul + pentru încasări și – pentru plăți). $1 desemnează primul parametru de intrare care e, în cazul nostru, și ultimul.

Fraza SELECT va avea forma:

```
SELECT dataora, f_sold_initial
   (dataora) AS Sold_Initial,
CASE WHEN tip='I' THEN suma ELSE 0 END
   AS Incasari ,
CASE WHEN tip='P' THEN suma ELSE 0 END
   AS Plata,
f_sold_initial (dataora) + CASE WHEN
   tip='I' THEN suma ELSE 0 END -
CASE WHEN tip='P' THEN suma ELSE 0 END
   AS Sold_Final,
document, explicatii
FROM registru_casa
ORDER BY dataora
```

Soluție SQL în PostgreSQL 7.2 și Oracle 8i bazată pe subconsultări în clauza FROM

De acum trecem la variante curat SQL, adică neprocedurale. Prima soluție speculează o facilitate a multor servere de baze de date, și anume prezența subconsultărilor în clauza FROM a frazei SELECT principale. Sintaxa următoare este Oracle 8i:

```
SELECT rc.dataora, Sold_Initial,
   CASE WHEN rc.tip='I' THEN rc.suma
        ELSE 0 END AS Incasari,
   CASE WHEN rc.tip='P' THEN rc.suma
        ELSE 0 END AS Plati,
Sold_Initial + CASE WHEN rc.tip='I'
   THEN rc.suma ELSE 0 END -
   CASE WHEN rc.tip='P' THEN rc.suma
        ELSE 0 END AS SoldFinal,
   document, explicatii
FROM registru_casa rc,
   (SELECT rc1.dataora, SUM (
CASE WHEN rc2.tip = 'I' THEN rc2.suma
     ELSE -rc2.suma END
) AS Sold_Initial
FROM registru_casa rc1, registru_casa
     rc2
WHERE rc1.dataora > rc2.dataora (+)
GROUP BY rc1.dataora ) s_i
WHERE rc.dataora = s_i.dataora
```

Subconsultarea, al cărei rezultat este referit prin s_i, calculează, prin însumare, soldul inițial al fiecărei operațiuni, joncțiunea externă după condiția rc1.dataora > rc2.dataora (+) asigurând includerea în rezultat și a operațiunii inițiale. Prin joncționarea naturală REGISTRU_CASĂ - S_I, se vor putea calcula toate valorile din *figura 2*.

PostgreSQL

Probabil singura diferență importantă a interogării PostgreSQL, prin comparație cu cea de mai sus, ține de modul de notație al joncțiunii externe:

```
SELECT rc.dataora, Sold_Initial,
   CASE WHEN rc.tip='I' THEN rc.suma
        ELSE 0 END AS Incasari,
   CASE WHEN rc.tip='P' THEN rc.suma
        ELSE 0 END AS Plati,
   Sold_Initial +  CASE WHEN
       rc.tip='I' THEN rc.suma ELSE 0
       END -
   CASE WHEN rc.tip='P' THEN rc.suma
        ELSE 0 END AS SoldFinal,
   document, explicatii
FROM registru_casa rc,
   (SELECT rc1.dataora, SUM (CASE WHEN
      rc2.tip = 'I' THEN rc2.suma ELSE
      -rc2.suma END ) AS Sold_Initial
   FROM registru_casa rc1 LEFT OUTER
      JOIN registru_casa rc2
ON rc1.dataora > rc2.dataora
   GROUP BY rc1.dataora ) s_i
WHERE rc.dataora = s_i.dataora
```

Listing 5. Funcția F_SOLD_FINAL (PL/SQL)

```
CREATE OR REPLACE FUNCTION f_sold_final  (
    tip_ IN registru_casa.tip%TYPE,
    suma_ IN registru_casa.suma%TYPE
        )   RETURN  registru_casa.suma%TYPE
IS
    sold_fin registru_casa.suma%TYPE    ;
BEGIN
    IF tip_ = 'I' THEN
      sold_fin := variabile_globale.sold_in + suma_ ;
    ELSE
      sold_fin := variabile_globale.sold_in - suma_ ;
    END IF ;
     variabile_globale.sold_in := sold_fin ;
     RETURN sold_fin ;
END;
```

Listing 6. Funcția (PL/pgSQL) F_SOLD_INIȚIAL

```
CREATE FUNCTION f_sold_initial (TIMESTAMP)  RETURNS NUMERIC AS '
DECLARE
   sold registru_casa.suma%TYPE := 0 ;
BEGIN
   SELECT SUM(CASE WHEN tip = ''I'' THEN suma ELSE -suma END) INTO sold
   FROM registru_casa
   WHERE dataora < $1 ;

   RETURN sold ;
END;
' LANGUAGE 'plpgsql' ;
```

Soluție SQL în PostgreSQL 7.2 și Oracle 9i bazată pe interogări scalare

Acolo unde interogările scalare funcționează (DB2, Oracle9i, PostgreSQL etc.) iată o variantă destul de simplă, în care corelarea apare între două instanțe ale tabelei REGISTRU_CASĂ, cea principală – rc1 - și cea din subcosultarea scalară – rc2. Sintaxa următoare este cea a PostgreSQL 7.2.:

```
SELECT dataora,
   (SELECT SUM (CASE WHEN rc2.tip =
      'I' THEN suma ELSE -suma END )
FROM registru_casa rc2 WHERE
   rc2.dataora < rc1.dataora) AS
   Sold_Initial,
CASE WHEN tip='I' THEN suma ELSE 0 END
   AS Incasari ,
CASE WHEN tip='P' THEN suma ELSE 0 END
   AS Plata,
(SELECT SUM (CASE WHEN rc2.tip = 'I'
   THEN suma ELSE -suma END )
FROM registru_casa rc2 WHERE
   rc2.dataora < rc1.dataora) + CASE
WHEN tip='I' THEN suma ELSE 0 END -
   CASE WHEN tip='P'
THEN suma ELSE 0 END AS Sold_Final,
document, explicatii
FROM registru_casa rc1
ORDER BY dataora
```

Interogarea funcționează fără de nici o modificare și în Oracle 9i.

Ca bun moldovean cu serioase simpatii ardelene, ar trebui să închei cu *No, aiasta-i, măi fișiori !* Omul recent din mine nu-mi dă pace, însă, așa că voi adopta tonul unei vedete rock, care în finalul unui concert, cu un aer de suficiență, ridică mâinile a salut și strigă către sală: *God bless you* !

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