Fiola de SQL

Trei dintre diferenţele SQL între Visual FoxPro şi Oracle

Pentru a trage un pic de timp, încep cu câteva explicaţii. (Pseudo) rubrica de faţă nu se vrea cu nici în chip (nu se citeşte *cip*) un concurent al SQL for Smarties ţinută glorios de Joe Celko în revista Intelligent Enterprise (ex. DBMS Magazine şi Data Base Programming & Design). De fapt, nu că nu ar vrea, dar nu cred că poate.

Obiectivul este de a prezenta câteva exemple simple (cum este cel de faţă) sau ceva mai complexe (cum va fi cel de peste 17 numere) în care SQL îşi arată, când virtuţile, când colţii. Pentru a avea o doză minimă de pragmatism, cam toate exemplele şi soluţiile vor fi prezentate în Visual FoxPro şi Oracle. De ce numai aceste două ? Pentru numai pe acestea le cunosc (din vedere, mai mult, dar ne-am plăcut şi…).

De ce Fiola de SQL ? Pastila de SQL ar fi sunat prea a brigăzi artistice de amatori pe la Cântarea României. Marcat de faptul că m-am născut şi trăiesc în triunghiul bahic Odobeşti-Cotnari-Huşi, m-am gândit, la un moment dat, la titulatura Canistra (sau bidonul) de SQL, dar am renunţat ruşinat. Aşa că…

Subiectul de astăzi este ca, pornind de la două exemple simple, să identificăm trei dintre diferenţele majore în redactarea frazei SQL în VFP şi Oracle 8.

Tabela - cobai

Tabela utilizată în exemple se numeşte PERSONAL. Conţine date despre angajaţii unei firme: marca; numele şi prenumele; data naşterii; compartiment; marca şefului (direct); salariu tarifar.

În Oracle comanda de creare a tabelei este:

```
CREATE TABLE personal (marca INTEGER CONSTRAINT pk_personal PRIMARY KEY, numepren VARCHAR2(40), datanast DATE,  compart VARCHAR2(20), marcasef INTEGER CONSTRAINT fk_personal REFERENCES personal(marca), saltarifar INTEGER) ;
```

În VFP, pentru a defini restricţiile (de cheie primară, referenţiale, la nivel de câmp şi înregistrare) şi valorile implicite este musai ca tabela să aparţină unei baze de date:

```
CREATE DATABASE personal
CREATE TABLE personal(marca INTEGER, numepren CHAR(40), datanast DATE, compart CHAR(20) NULL, marcasef INTEGER NULL ,  saltarifar INTEGER, PRIMARY KEY marca TAG primaru, FOREIGN KEY marcasef TAG marcasef REFERENCES personal TAG primaru)
```

Problema 1. Care sunt angajaţii care au acelaşi salariu tarifar ca al angajatului Munteanu Ghiocel ?

Soluţia 1 (VFP şi Oracle)

Este un gen de probleme simplu de rezolvat în SQL. Probabil cea mai lejeră soluţie este:

```
SELECT *
FROM personal
WHERE saltarifar IN (SELECT saltarifar FROM personal
                     WHERE numepren='MUNTEANU GHIOCEL')
```

Şi VFP şi Oracle permit redactarea (şi execuţia acestei soluţii).

Soluţia 2 – mai interesantă (VFP şi Oracle)

Această variantă de rezolvare e un pic mai ingenuoasă.

```
SELECT p2.*
FROM personal p1, personal p2
```

```
WHERE p1.saltarifar=p2.saltarifar and p1.numepren='MUNTEANU GHIOCEL'
```

Se face joncţiunea tabelei personal cu ea-însăşi, după câmpul salariu tarifar. Pe fiecare linie a tabelei-rezultat vor fi doi angajaţi cu acelaşi salariu. Neoperând o selecţie suplimentară, pe o linie va apărea Munteanu Ghiocel (în p1) în corepondenţă cu el-însuşi (în p2). Pentru a rezolva problema aşa cum am formulat-o, se inserează predicatul de selecţie: p1.numepren='MUNTEANU GHIOCEL'.

Soluţia 3 – ceva mai dificil de explicat (VFP şi Oracle)

Întotdeauna am avut emoţii în faţa operatorului EXISTS. Considerat mai puternic decât IN (vezi soluţia 1) logica lui EXISTS este ceva mai complicată, încât merită o "fiolă" separată.

```
SELECT *  FROM personal p1 WHERE EXISTS
            (SELECT *  FROM personal p2
             WHERE p1.saltarifar=p2.saltarifar AND
                   p2.numepren='MUNTEANU GHIOCEL')
```

Soluţia 4 – specifică Oracle

În materie de SQL, Oracle are câteva clase deasupra VFP (lucrul acesta se vede şi la preţ). Unul din avantajele implementării SQL în Oracle este că în clauza FROM o tabelă poate fi definită printr-o subconsultare (o frază SELECT inclusă) SQL.

```
SELECT p.*
FROM personal p, (SELECT saltarifar FROM personal WHERE numepren='MUNTEANU GHIOCEL') st
WHERE p.saltarifar=st.saltarifar
```

Tabela personal nu mai este joncţionată cu ea-însăşi, ci cu o variantă optimizată a acesteia, st (de la Salariu Tarifar) ce conţine o singură linie şi o singură coloană. Rezultă un plus de productivitate faţă de soluţia 2.

Problema 2. Care sunt compartimentele cu acelaşi număr de angajaţi ca şi compartimentul în care lucrează Munteanu Ghiocel ?

În Visual FoxPro un asemenea gen se problemă are o rezolvare mixtă, procedură-SQL. Explicaţia este simplă: clauza HAVING nu poate conţine subconsultări, iar, în plus, subconsultările pot fi derulate pe un singur nivel.

Soluţia 1 - VFP

```
* citi angajati numara fiecare compartiment
SELECT compart, count(*) as citi ;
FROM personal  ;
INTO CURSOR compart_nr ;
GROUP BY compart

* citi angajati are compartimentul lui...
SELECT citi ;
FROM compart_nr ;
INTO CURSOR nr_angaj ;
WHERE compart IN (SELECT compart ;
                  FROM personal ;
                  WHERE numepren='MUNTEANU GHIOCEL')

* in fine, raspunsul
SELECT compart ;
FROM compart_nr  ;
WHERE citi IN (SELECT citi ;
               FROM nr_angaj)
```

Soluţia2 – Oracle

În Oracle clauza HAVING poate conţine subconsultări. În plus, spre deosebire de VFP, putem avea oricâte niveluri de imbricare a subconsultărilor.

```
SELECT compart
FROM personal
GROUP BY compart
HAVING COUNT(*) = (SELECT COUNT(*)
                   FROM personal
                   WHERE compart IN  (SELECT compart
                                      FROM personal
                                      WHERE numepren='MUNTEANU GHIOCEL')) ;
```

Soluţia 3 – Oracle

Cu ocazia soluţiei 4 de la problema 1 am aflat că în clauza FROM pot fi definite, ad-hoc, prin subconsultări (fraze SELECT incluse), alte tabele decât cele din bază. Lucrul acesta ne este de mare folos şi pentru rezolarea prezentei probleme.

```
SELECT compart
FROM (SELECT compart, COUNT(*) AS nr FROM personal GROUP BY compart) t1,
        (SELECT COUNT(*) AS nr FROM personal WHERE compart IN
                (SELECT compart FROM personal WHERE numepren='MUNTEANU GHIOCEL'))  t2
WHERE t1.nr = t2.nr     ;
```

Concluzii

Dintre diferenţele în materie de SQL între Oracle şi VFP, două au fost prezentate explicit: subconsultări incluse în clauza FROM şi subconsultări incluse în clauza HAVING. O a treia, numărul nivelelor pe care pot fi incluse (imbricate) subconsultările a fost amintită în treacăt.

Ar mai fi şi alte soluţii de discutat la problemele propuse, dar din fiolă am ajunge la borcanul de SQL. Pe curând !

Marin Fotache