Fiola de SQL (16)

Un spor mic și niște rețineri mari

– Marin Fotache

Este de așteptat ca această fiolă să fie una dintre cele mai antipatice, atâta vreme cât se referă la calcularea, în SQL, a impozitului pe venit. Eu, unul, îmi cer scuze pentru lipsa de inspirație în găsirea unui exemplu mai tonic/pozitiv. Poate o să mă revanșez cu un material mult mai generos dedicat evaziunii fiscale… Până una-alta, pentru a mai atenua din nemulțumire, o să discutăm, mai întâi, despre sporul de vechime care e, ce-i drept, prea mic pentru niște rețineri așa de mari.

Pornim de la aceeași tabelă – PERSONAL2 – în care adăugăm un atribut (data_sv) ce indică data de la care se calculează sporul de vechime pentru fiecare angajat. Deși am folosit această tabelă de numărate ori, îi prezentăm structura prin comanda de creare specifică Postgres-ului.

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

Comezile de inserare în PostgresSQL sunt identice formatului Oracle, fiind prezentate în versiunea on-line a articolului.

```
INSERT INTO personal2 VALUES (1, 'ANGAJAT 1',
   TO_DATE('01-07-1962','DD-MM-YYYY'),
 'DIRECTIUNE',NULL, 6000000,
   TO_DATE('11-07-1987','DD-MM-YYYY')) ;

INSERT INTO personal2 VALUES (2, 'ANGAJAT 2',
   TO_DATE('11-10-1977','DD-MM-YYYY'),
 'FINANCIAR',  1, 4500000,
   TO_DATE('15-06-2001','DD-MM-YYYY') );

INSERT INTO personal2 VALUES (3, 'ANGAJAT 3',
   TO_DATE('22-08-1962','DD-MM-YYYY'),
 'MARKETING', 1, 4500000,
   TO_DATE('01-03-1979','DD-MM-YYYY')) ;

INSERT INTO personal2 VALUES (4, 'ANGAJAT 4',          NULL,
   'FINANCIAR',    2, 3800000,
   TO_DATE('01-10-1972','DD-MM-YYYY') );

INSERT INTO personal2 VALUES (5, 'ANGAJAT 5',
   TO_DATE('30-04-1965','DD-MM-YYYY'),
 'FINANCIAR',    2, 4200000,
   TO_DATE('31-08-1993','DD-MM-YYYY')) ;

INSERT INTO personal2 VALUES (6, 'ANGAJAT 6',
   TO_DATE('09-11-1965','DD-MM-YYYY'),
'FINANCIAR',    5, 3500000,
   TO_DATE('15-09-1990','DD-MM-YYYY') ) ;
INSERT INTO personal2 VALUES (7, 'ANGAJAT 7',
   NULL,
'FINANCIAR',    5, 2800000,
   TO_DATE('17-05-1996','DD-MM-YYYY') ) ;

INSERT INTO personal2 VALUES (8, 'ANGAJAT 8',
   TO_DATE('31-12-1960','DD-MM-YYYY'),
'MARKETING',    3, 2900000,
   TO_DATE('24-09-1986','DD-MM-YYYY') ) ;

INSERT INTO personal2 VALUES (9, 'ANGAJAT 9',
   TO_DATE('28-02-1976','DD-MM-YYYY'),
'MARKETING',    3, 4100000,
   TO_DATE('01-12-2000','DD-MM-YYYY') ) ;

INSERT INTO personal2 VALUES (10,'ANGAJAT 10',
   TO_DATE('29-01-1972','DD-MM-YYYY'),
'RESURSE UMANE',1, 3700000,
   TO_DATE('01-07-1994','DD-MM-YYYY')) ;
```

Și a doua tabelă este una celebră – SPORURI – și are următoarea structură:

```
CREATE TABLE sporuri (
   an INTEGER,
   luna INTEGER,
   marca INTEGER REFERENCES personal2 (marca),
   sporvechime INTEGER,
   spornoapte INTEGER,
   sporcd INTEGER,
   altespor INTEGER,
   PRIMARY KEY (an,luna,marca)
   ) ;
INSERT INTO sporuri VALUES (2001, 4, 1, 0, 0, 0, 320000) ;
INSERT INTO sporuri VALUES (2001, 4, 2, 0, 450000, 0, 170000);
INSERT INTO sporuri VALUES (2001, 4, 3, 0, 560000,1200000,
   570000);
INSERT INTO sporuri VALUES (2001, 5, 1, 0, 0, 0, 0) ;
INSERT INTO sporuri VALUES (2001, 5, 2, 0, 450000, 0, 170000);
INSERT INTO sporuri VALUES (2001, 5, 3, 0, 0, 0, 0);
INSERT INTO sporuri VALUES (2001, 5,10, 0, 0, 0, 860000) ;
INSERT INTO sporuri VALUES (2001, 6, 1, 0, 0, 0, 0) ;
INSERT INTO sporuri VALUES (2001, 6, 2, 0, 0, 0, 150000) ;
INSERT INTO sporuri VALUES (2001, 6, 4, 0, 150000, 880000,
   150000) ;
INSERT INTO sporuri VALUES (2001, 6, 5, 0, 150000, 0, 50000);
INSERT INTO sporuri VALUES (2001, 6,10, 0, 80000, 0, 60000);
INSERT INTO sporuri VALUES (2001, 7, 1, 0, 0, NULL, NULL) ;
INSERT INTO sporuri VALUES (2001, 7, 2, 0, 0, 0, 170000) ;
INSERT INTO sporuri VALUES (2001, 7, 3, 0, 0, 0, 0) ;
```

```
INSERT INTO sporuri VALUES (2001, 7, 4, 0, 150000, NULL,
   150000) ;
INSERT INTO sporuri VALUES (2001, 7, 5, 0, 0, 0, 140000) ;
INSERT INTO sporuri VALUES (2001, 7, 6, 0, 575000, 0, 0) ;
INSERT INTO sporuri VALUES (2001, 7, 7, 0, 875000, 0, 0) ;
INSERT INTO sporuri VALUES (2001, 7, 8, 0, 0,1500000, 0) ;
INSERT INTO sporuri VALUES (2001, 7, 9, 0, 560000, 775000, 0);
INSERT INTO sporuri VALUES (2001, 7,10, 0, 0, 0, 860000) ;
```

Pornind de la aceste două tabele, și adăugând altele pe parcurs, ne vom concentra asupra a două probleme: calculul sporului de vechime și al impozitului pe venit pentru, să zicem, luna iulie din 2001.

Sporul de vechime

Sporul de vechime se acordă diferențiat, pe tranșe, în funcție de anii de muncă pe care îi numără fiecare angajat. Pentru simplificarea discuției și eludarea realității, considerăm că:

- până la trei ani de vechime, nu se acordă acest spor;
- peste trei ani, dar sub șase ani, procentul sporului de vechime este 3% aplicat la salariul tarifar;
- între 6 și 10 ani precentul este de 5%;
- între 10 și 20 de ani – 10 %;
- peste 20 de ani – 15%.

Calcularea anilor de vechime se realizează prin diferență între 1 ale lunii curente (în cazul nostru, 1 iulie 2001) și data de la care se determină sporul de vechime (care este, de multe ori, data primei angajări). Dacă în fiola nr. 12 (aprilie 2001) am prezentat câteva funcții pentru lucrul cu date calendartistice în DB2, Oracle și Visual FoxPro, acum vom zăbovi câteva rânduri asupra acestor funcții în PostgreSQL.

Date calendaristice în PostgreSQL

În materie de gestionare a informațiilor de tip dată calendaristică și timp, PostgreSQL este generos, tipurile acceptate fiind:

- `timestamp` – data/ora (inclusiv minute, secunde, microsecunde);
- `timestamp with time zone` – data/ora plus zona temporală (orară);
- `interval` – intervale de timp (ani, luni, zile….);
- `date` – numai dată calendaristică;
- `time` – numai ora (inclusiv minute, secunde, microsecunde);
- `time with time zone` – ora (inclusiv minute, secunde, microsecunde) zona orară.

Funcțiile standard SQL referitoare la dată, oră și dată/oră curente sunt, de asemenea, prezente: `CURRENT_DATE`, `CURRENT_TIME`, `CURRENT_TIMESTAMP`. Câteva exemple sunt nimerite:

- extragerea datei curente: `SELECT CURRENT_DATE`
- extragerea datei/orei curente: `SELECT CURRENT_TIMESTAMP`
- care va fi ora exactă, la 41 de ore și 34 de minute după ora 11 fără un sfert dimineață ?

```
SELECT TIME '10:45:00' + INTERVAL '41 HOURS 34 MINUTES'
```

- ce dată va fi peste trei ani și 37 de săptămâni (începând cu data curentă)?

```
SELECT CURRENT_DATE + INTERVAL '3 YEARS 37 WEEKS'
```

Interesant este că PostgreSQL acceptă și fraze select incomplete, fără clauza `FROM` (în Oracle am fi avut nevoie de tabela-sistem `DUAL`).

Pentru extragerea anului, lunii, zilei etc. din atribute, variabile sau constante de tip dată, timp sau dată/timp, se poate folosi funcția `EXTRACT`. Ca elemente pregătitoare ale determinării sporului de vechime, de care ne vom ocupa cât de curând, încercăm să calculăm numărul de zile și numărul (întreg) de ani care s-au scurs de la data-etalon a vechimii în muncă (data primei angajări, decalată cu perioada/perioadele în care salariatul nu a lucrat, sau a lucrat la negru (de fapt, mai rar la negru și mai des la neamț, italian etc.)) până la 1 iulie 2001. Întrucât în PostgresSQL nu am găsit nici o funcție specială, gen `MONTHS_BETWEEN` din Oracle, pentru calculul numărului (întreg, fără rotunjire) de ani am recurs la o structură `CASE` nu prea complicată:

```
SELECT data_sv,
TO_DATE('01/07/2001', 'DD/MM/YYYY') - data_sv AS
   Zile_Diferenta,
CASE WHEN
  7 > EXTRACT (MONTH FROM data_sv) OR
 (7 = EXTRACT (MONTH FROM data_sv) AND
EXTRACT (DAY FROM data_sv)=1 )
THEN 2001 - EXTRACT (YEAR FROM data_sv)
     ELSE 2001 - EXTRACT (YEAR FROM data_sv) - 1
END AS Ani_Diferenta
FROM PERSONAL2
```

Varianta 1 de calcul a sporului de vechime

Sistemul de tranșe poate fi reprezentat în mai multe moduri. Cea mai simplă, pentru interogările următoare, dar și cea mai redundantă modalitate, ține de gruparea într-o tabelă denumită nesugestiv `TSV` (de la Tranșe Spor Vechime) a limitelor inferioară, superioară și a procentului pentru fiecare tranșă:

```
CREATE TABLE tsv (
 LimInf INTEGER,
 LimSup INTEGER,
 Procent NUMERIC (2,2)
) ;

INSERT INTO tsv VALUES (0, 2, 0) ;
INSERT INTO tsv VALUES (3, 5, .03) ;
INSERT INTO tsv VALUES (6, 9, .05) ;
INSERT INTO tsv VALUES (10, 19, .10) ;
INSERT INTO tsv VALUES (20, 999, .15) ;
```

Pentru aflarea procentului de calcul sporului de vechime, în funcție de anii de muncă, se poate folosi o variantă bazată pe theta-joncțiune în care condiția este compusă: încadrarea anilor de vechime în tranșele reprezentate de liniile tabelei `TSV`.

Începem cu o soluție Postgres (nu cea mai elegantă) care crează o tabelă derivată – `VECHIME` – tabelă derivată ce conține, pentru fiecare angajat, salariul tarifar și numărul anilor de vechime la data de 1 iulie 2001:

```
CREATE VIEW vechime AS
SELECT marca, saltarifar,
CASE WHEN
 7 > EXTRACT (MONTH FROM data_sv) OR
 (7 = EXTRACT (MONTH FROM data_sv) AND
EXTRACT (DAY FROM data_sv)=1 )
  THEN 2001 - EXTRACT (YEAR FROM data_sv)
  ELSE 2001 - EXTRACT (YEAR FROM data_sv) - 1
END AS ani_vechime
FROM PERSONAL2
```

Această tabelă derivată va fi folosită apoi într-o interogare în care sporul de vechime se calculează prin produsul `saltarifar * procent`, pe baza theta-joncțiunii dintre `VECHIME` și `TSV` prin condiția: `ani_vechime >= tsv.LimInf AND ani_vechime <= tsv.LimSup` (sau `ani_vechime BETWEEN tsv.LimInf AND tsv.LimSup`):

```
SELECT marca, saltarifar, ani_vechime, procent,
saltarifar * procent as spor_vech
FROM vechime INNER JOIN tsv
ON ani_vechime >= tsv.LimInf AND ani_vechime <= tsv.LimSup
```

Revenind la tabela SPORURI, pentru calculul sporului de vechime corespunzător lunii iulie 2001 se folosește o comandă UPDATE al cărei ingredient principal este o interogare corelată, astfel:

```
UPDATE sporuri
SET sporvechime =
(SELECT saltarifar * procent
FROM vechime INNER JOIN tsv
ON ani_vechime >= tsv.LimInf AND ani_vechime <= tsv.LimSup
WHERE vechime.marca=sporuri.marca)
WHERE an=2001 and luna=7
```

Soluția Oracle 8i pe care o prezentăm mai jos este ceva mai elegantă, deoarece folosește subconsultări în clauza FROM:

```
UPDATE sporuri
SET sporvechime =
(SELECT saltarifar * procent
            FROM
            (SELECT marca, saltarifar,
                 CASE WHEN 7  >
   TO_NUMBER(TO_CHAR(data_sv,'MM'))
OR (7  = TO_NUMBER(TO_CHAR(data_sv,'MM'))
AND TO_NUMBER(TO_CHAR(data_sv,'DD')) = 1)
                THEN 2001 -
   TO_NUMBER(TO_CHAR(data_sv,'YYYY'))
                ELSE 2001 -
   TO_NUMBER(TO_CHAR(data_sv,'YYYY')) - 1
                END AS ani_vechime
            FROM PERSONAL2 ) vechime, tsv
 WHERE ani_vechime >= tsv.LimInf AND ani_vechime <=
   tsv.LimSup
AND  vechime.marca=sporuri.marca)
WHERE an=2001 and luna=7
```

Varianta 2 pentru calculul sporului de vechime

În tabela TSV, prezența pe fiecare linie a limitelor inferioară și superioară de ani din fiecare tranșă ar putea genera stâmbături nazale din parte unor concetățeni, dat fiind că avem de-a face cu o redundanță: limita superioară a unei tranșe este limita inferioară a tranșei următoare. Prin urmare structura cea mai economicoasă ar fi:

```
CREATE TABLE tsv2 (
 LimSup INTEGER,
 Procent NUMERIC (2,2)
) ;

INSERT INTO tsv2 VALUES (3, 0) ;
INSERT INTO tsv2 VALUES (6, .03) ;
INSERT INTO tsv2 VALUES (10, .05) ;
INSERT INTO tsv2 VALUES (20, .10) ;
INSERT INTO tsv2 VALUES (999, .15) ;
```

Eu, unul, sunt de părere că ne complicăm inutil, dar cum sunt mulți cârcotași în patria noastră frumoasă & bogată, haideți să vedem cum se rezolvă problema în noile condiții. Pentru a nu irosi prea mult spațiu, vă prezint o singură soluție, și aceea în Oracle 8i:

```
UPDATE sporuri
SET sporvechime =
(SELECT MIN(saltarifar * procent)
 FROM
(SELECT marca, saltarifar,
CASE WHEN 7  > TO_NUMBER(TO_CHAR(data_sv,'MM'))
OR (7  = TO_NUMBER(TO_CHAR(data_sv,'MM'))
AND TO_NUMBER(TO_CHAR(data_sv,'DD')) = 1)
                      THEN 2001 -
   TO_NUMBER(TO_CHAR(data_sv,'YYYY'))
                      ELSE 2001 -
   TO_NUMBER(TO_CHAR(data_sv,'YYYY')) - 1
END AS ani_vechime
             FROM PERSONAL2 ) vechime, tsv2
         WHERE ani_vechime < tsv2.LimSup AND
   vechime.marca=sporuri.marca )
    WHERE an=2001 and luna=7
```

Condiția de joncționare este acum ani_vechime < tsv2.LimSup, iar, pentru ca procentul să fie calculat pe baza primei tranșe în care anii de vechime sunt sub limita superioară a tranșei, se folosește funcția MIN.

Și acum, impozitul

Modalitățile de calcul a impozitului pe venit au suferit modificări de mare amploare în ultimul deceniu, astfel încât dezvoltatorii de aplicații pentru gestiunea datelor privind salarizarea au avut pâinea asigurată. Dacă avem în vedere tot mai marea deschidere a economiei românești către exterior, dar și impresionanta capacitatea de improvizație a tuturor guvernelor care ne-au răsfățat până acum, nu avem de ce să ne facem probleme în acest sens.

Mai întâi, să analizăm noile reglementări cu privire la scutirea de impozit a categoriilor de specialiști din zona IT: programatori, analiști, administratori de rețele, administratori de baze de date, mai puțin profesori (ca mine, ca să dau un exemplu la întâmplare). Înțeleg strategia guvernului: scopul scutirii este de a opri exodul de specialiști IT în lumea civilizată. Ori, să fii profesor de IT în România înseamnă că ai un defect, odată ce refuzi un trai decent pe o leafă jenantă de bugetar. Prin urmare, nu-i nici un pericol, dacă tot n-au de gând să plece, ba chiar nu putem scăpa de ei, la ce bun să le mai acordăm scutiri profesorilor ?

Modificarea schemei baze de date se poate face în cel puțin două moduri:

- adăugarea unui atribut nou în tabela PERSONAL (nomenclatorul angajaților) care să indice dacă salariatul este sau nu scutit de impozit – scutit_imp;
- inserarea unui atribut de genul ocupație (job, profesie etc.) în nomenclatorul de personal și crearea unui nomenclator al ocupațiilor în care să apară un atribut care să indice dacă respectiva categorie de personal este scutită de impozit.

Dacă ar fi să alegem prima variantă, noul atribut, scutit_imp, ar trebuie să fie de tip BOOLEAN, lucru posibil în unele SGBD-uri, precum Postgres (sau LOGICAL în Visual FoxPro):

```
ALTER TABLE personal2 ADD scutit_imp BOOLEAN DEFAULT FALSE
```

Alte produse, precum Oracle și DB2, nu suportă tipul de date BOOLEAN, caz în care se poate folosi un șir (de un caracter):

```
ALTER TABLE personal2 ADD scutit_imp CHAR(1) DEFAULT 'N'
```

Elementul de maximă noutate în sistemul de impozitare actual îl reprezintă sistemul de deduceri care se aplică pentru copiii și celelalte persoanele aflate în întreținerea salariatului. Dar să o luăm metodic. Baza de calcul a impozitului lunar pentru fiecare salariat se detrmină astfel:

```
BC := Vb   -    (cfș   +   cas   +   cass)   -
   ded_pers_bază   -   ch_profes   --   ded_pers_suplim
```

unde:
BC – baza de calcul

Calculul impozitului

Între	și	Suma fixă		plus procentul * ceea ce depășește limita inferioară
0	1.458.000	0	+	18%
1.458.001	3.578.000	262.440	+	23% pt. suma peste 1.458.000
3.578.001	5.699.000	750.040	+	28% pt. suma peste 3.578.000
5.699.001	7.952.000	1.343.920	+	34% pt. suma peste 5.699.000
7.952.001	Bill Gates	2.109.940	+	40% pt. suma peste 7.952.000

Vb – venitul brut
cfș – contribuția (individuală) la fondul de șomaj (1%)
cas - contribuția (individuală) la asigurările sociale (11,67%)
cass - contribuția (individuală) la asigurările sociale de sănătate (7%)
ded_pers_bază – deducerea personală de bază – se stabilește semestrial și este, în prezent, de 1 273 000 lei /lună
ch_profes – cheltuieli profesionale (reprezintă 15% din deducerea personală de bază)
ded_pers_suplim – deducerea personală suplimentară – se acordă pentru persoanele aflate în întreținerea angajatului, inclusiv pentru situațiile speciale ale acestora.

Lucrurile devin cu adevărat interesante la calculul deducerii personale suplimentare care poate încorpora una sau mai multe dintre următoarele componente:
- 0,6 * *ded_pers_bază* - dacă soțul sau soția este în întreținerea angajatului
- 0,35 * *ded_pers_bază* – pentru fiecare dintre primii doi copii minori aflați în întreținere
- 0,20 * *ded_pers_bază* - pentru fiecare dintre următorii copii minori (al treilea, al patrulea…)
- 1,00 * *ded_pers_bază* - pentru fiecare copil și persoană cu handicap – prima categorie
- 0,50 * *ded_pers_bază* - pentru fiecare copil și persoană cu handicap – a doua categorie.

Și acum, punctul culminant: cumulate, *ded_pers_bază* + *ded_pers_suplim* nu pot depăși 2.5 din *ded_pers_bază*.

Asupra bazei de calcul aflate prin formula de mai sus se aplică impozitul pe venit care, nici el, nu este mai prejos, întrucât trebuie luate în calcul cinci tranșe în care se poate încadra *BC*, vezi tabelul alăturat.

Dacă ceea ce v-am spus nu este adevărat, vă rog să-l mușt ruiți pe unul dintre amicii de la catedra de Contabilitate a facultății (se numește *istrate@uaic.ro*).

Determinarea deducerilor și calculul impozitului devin astfel destul de interesante în aceste condiții. Există mai multe moduri de a organiza baza de date. O variantă relativ simplă este de apela la o tabelă căreia să-i zicem `PERS_INTRETINERE`, a cărei comadă de creare este prezentată după canoanele Oracle 8i:

```
CREATE TABLE pers_intretinere (
    marca INTEGER CONSTRAINT fk_pers_intretinere_personal2
        REFERENCES personal2(marca),
    cnp_pers CHAR(14) NOT NULL,
    numepren_pers VARCHAR2(25) NOT NULL,
    datanast_pers DATE,
    categorie CHAR(1) DEFAULT 'C'
        CONSTRAINT ck_pers_intretinere_relatie
        CHECK (categorie IN ('S', 'C', 'A', 'H')) NOT NULL,
    coeficient_ded NUMBER (4,2),
    CONSTRAINT pk_pers_intretinere PRIMARY KEY (marca,
   cnp_pers, categorie)
    ) ;
```

Pe o linie a acestei tabele se află o persoană aflată în întreținere și un tip (o categorie) de deducere. Categoriile de deducere sunt: *S* (soț/soție), *C* (copil), *H* (persoană cu handicap), *A* (altă persoană aflată în întreținere (bunică, mătușă etc.). Un copil minor cu, fereasc Domnul, handicap apare pe două linii, o dată la categoria *C*, și a doua oară cu *H*. Firește, nu este cea mai bună structură, dar funcționează.

```
sINSERT INTO pers_intretinere VALUES
(1, 'CNP1-1','EMIL',
TO_DATE('01-07-1987','DD-MM-YYYY'), 'C',
.35) ;
INSERT INTO pers_intretinere VALUES (1, 'CNP1-2','EMILIA',
TO_DATE('15-12-1989','DD-MM-YYYY'), 'C', .35) ;
INSERT INTO pers_intretinere VALUES (1, 'CNP1-3','EMILIANA',
TO_DATE('16-04-1993','DD-MM-YYYY'), 'C', .35) ;

INSERT INTO pers_intretinere VALUES (2, 'CNP2-1','IOANA',
TO_DATE('12-10-1978','DD-MM-YYYY'), 'S', .6) ;
INSERT INTO pers_intretinere VALUES (2, 'CNP2-2','IOAN',
TO_DATE('12-10-1997','DD-MM-YYYY'), 'C', 0.35) ;
INSERT INTO pers_intretinere VALUES (2, 'CNP2-2','IOAN',
TO_DATE('12-10-1997','DD-MM-YYYY'), 'H', 1.00) ;
INSERT INTO pers_intretinere VALUES (2, 'CNP2-3','VASILE',
TO_DATE('14-11-1998','DD-MM-YYYY'), 'C', 0.35) ;
INSERT INTO pers_intretinere VALUES (2, 'CNP2-4','ADRIAN',
TO_DATE('15-12-1999','DD-MM-YYYY'), 'C', 0.35) ;
INSERT INTO pers_intretinere VALUES (2, 'CNP2-5','GINA',
TO_DATE('10-01-2000','DD-MM-YYYY'), 'C', 0.35) ;

INSERT INTO pers_intretinere VALUES (3, 'CNP3-1','LOREDANA',
TO_DATE('10-01-1987','DD-MM-YYYY'), 'C', 0.35) ;

COMMIT ;
```

Interogarea SQL prin care se calculează, pentru fiecare angajat, coeficientul deducerilor suplimentare are următoarea formă în Oracle:

```
SELECT marca,
    LEAST( 2.5,
    SUM (CASE WHEN coeficient_ded = .35 AND nr <=2 OR
    coeficient_ded <> .35
                    THEN coeficient_ded * nr
                    ELSE 2 * coeficient_ded + (nr - 2) * .20
         END)
         ) AS Coeficient
FROM
    (SELECT marca, categorie, coeficient_ded, COUNT(*) AS nr
     FROM pers_intretinere
     GROUP BY marca, categorie, coeficient_ded)
GROUP BY marca
```

Funcția `LEAST` ne asigură că, pentru o persoană, coeficientul nu poate depăși 2,5. La însumare, se are în vedere ca, începând cu al treilea copil aflat în întreținere, coeficientul să fie 0,2 și nu 0,35.

Dacă presupunem că venitul brut lunar este suma dintre salariul tarifar și totalul sporurilor, atunci baza de impozitare ar putea fi calculată cu următoarea interogare (Oracle):

```
SELECT venituri.marca, venituri.saltarifar, venituri.venituri,
    venituri * .01 AS cfs, venituri * .1167 AS cas,
    venituri * .07 AS casa,
    1273000 AS ded_pers_baza, .15 * 1273000 AS ch_profes,
     NVL(coeficient,0) * 1273000 AS ded_pers_suplim,
    ROUND(GREATEST (venituri - venituri * .01 - venituri *
   .1167 - venituri * .07 -
```

```
        1273000 - .15 * 1273000 - NVL(coeficient,0) *
   1273000, 0),-3) AS baza_calcul
FROM
    (SELECT p2.marca, p2.saltarifar, p2.saltarifar +
    NVL(sporvechime,0) + NVL(spornoapte,0) + NVL(sporcd,0)
   + NVL(altespor,0) AS sporuri,
    p2.saltarifar +     NVL(sporvechime,0) +
    NVL(spornoapte,0) +
        NVL(sporcd,0) + NVL(altespor,0) AS venituri
    FROM personal2 p2, sporuri
    WHERE p2.marca = sporuri.marca (+) AND sporuri.an (+)
   =2001  AND
sporuri.luna (+) = 7
    ) venituri,
    (
    SELECT marca,
        LEAST( 2.5,
        SUM (CASE WHEN coeficient_ded = .35 AND nr <=2 OR
   coeficient_ded <> .35
                        THEN coeficient_ded * nr
                        ELSE 2 * coeficient_ded + (nr -
   2) * .20
            END)
            ) AS Coeficient
    FROM
        (SELECT marca, categorie, coeficient_ded, COUNT(*)
   AS nr
         FROM pers_intretinere
         GROUP BY marca, categorie, coeficient_ded)
    GROUP BY marca ) ded_suplim
WHERE  venituri.marca = ded_suplim.marca    (+)
```

Calculul efectiv al impozitului se face analog sporului de vechime, cu ajutorul unei tabele, BAREM_IMPOZIT:

```
CREATE TABLE barem_impozit (
 LimInf INTEGER,
 LimSup INTEGER,
 suma_fixa INTEGER,
 Procent NUMERIC (2,2)
) ;

INSERT INTO barem_impozit VALUES ( 0, 1458000, 0, 0.18) ;
INSERT INTO barem_impozit VALUES (1458001, 3578000, 262440,
   0.23) ;
INSERT INTO barem_impozit VALUES (3578001,   5699000,
   750040, 0.28) ;
INSERT INTO barem_impozit VALUES (5699001,   7952000,
   1343920, 0.34) ;
INSERT INTO barem_impozit VALUES (7952001, 999999999,
   2109940, 0.40) ;

COMMIT ;
```

Având drept model interogarea pentru determinarea sporului de vechime, impozitul pe venit corespunzător fiecărui angajat pe luna iulie 2001 se determină prin consultarea următoare:

```
SELECT baza.*,
    CASE WHEN scutit_imp = 'D' THEN 0
       ELSE ROUND(GREATEST(suma_fixa + procent *
       (baza_calcul - (liminf + 1)),0),-1)
       END AS impozit
FROM
    (
    SELECT venituri.marca, scutit_imp, venituri.saltarifar,
   venituri.venituri,
        venituri * .01 AS cfs, venituri * .1167 AS cas,
   venituri * .07 AS casa,
        1273000 AS ded_pers_baza, .15 * 1273000 AS ch_profes,
         NVL(coeficient,0) * 1273000 AS ded_pers_suplim,
        ROUND(GREATEST (venituri - venituri * .01 -
        venituri * .1167 - venituri * .07 -
            1273000 - .15 * 1273000 - NVL(coeficient,0) *
   1273000, 0),-3) AS baza_calcul
    FROM
        (SELECT p2.marca, p2.scutit_imp, p2.saltarifar,
   p2.saltarifar +
            NVL(sporvechime,0) + NVL(spornoapte,0) +
            NVL(sporcd,0) + NVL(altespor,0) AS sporuri,
            p2.saltarifar +     NVL(sporvechime,0) +
   NVL(spornoapte,0) +
            NVL(sporcd,0) + NVL(altespor,0) AS venituri
        FROM personal2 p2, sporuri
        WHERE p2.marca = sporuri.marca (+) AND sporuri.an
   (+) =2001  AND sporuri.luna (+) = 7
        ) venituri,
        (
        SELECT marca,
            LEAST( 2.5,
            SUM (CASE WHEN coeficient_ded = .35 AND nr <=2
   OR coeficient_ded <> .35
                           THEN coeficient_ded * nr
                           ELSE 2 * coeficient_ded + (nr -
   2) * .20
                END)
                ) AS Coeficient
        FROM
            (SELECT marca, categorie, coeficient_ded,
   COUNT(*) AS nr
              FROM pers_intretinere
              GROUP BY marca, categorie, coeficient_ded)
        GROUP BY marca ) ded_suplim
     WHERE  venituri.marca = ded_suplim.marca    (+)
     ) baza, barem_impozit
WHERE baza_calcul >= liminf AND baza_calcul <= limsup
```

CASE-ul din clauza SELECT principală asigură scutirea de impozit a personalului IT din firmă. Funcțiile ROUND asigură rotunjirea la zeci (când precizia de rotunjire este –1) și la mii (când al doilea argument este –3). Funcția GREATEST elimină riscul apariției valorilor negative pentru impozit și baza de calcul.

Ar fi fost mult mai elegant, și chiar vă recomand ca, în locul utilizării constantelor pentru deducerea personală de bază (1 273 000), contribuțiile la asigurările de șomaj s.a.m.d, să utilizați o tabelă specială, astfel încât, ori de câte ori se modifică sumele/procentele, doar tabela specială trebuie actualizată, iar interogările rămân neschimbate.

Acestea ar fi câteva considerente vis-à-vis de modul în care se poate calcula sporul de vechime și mai ales câteva dintre dările către prea-cinstitul și prea-ocrotitorul stat român. Din lipsă de spațiu (eternul alibi), nu am mai inclus variantele DB2, VFP sau calculul impozitului în PostgreSQL. Vă doresc toate bune și bronz uniform (pentru cei care se expun).

Marin Fotache este conferențiar la Catedra de Informatică Economică, UAIC Iași, Facultatea de Economie și Administrarea Afacerilor. Poate fi contactat pe email la: fotache@uaic.ro. ■ 76