Noutăți SQL în Oracle 9i

Fiola de SQL (23)

– Marin Fotache

Oracle ne-a obișnuit, din frageda sa pruncie – iaca, acuși face 25 ani de la naștere – cu postura de fruntaș pe ramură, și nu numai la SQL. De altimteri, corporația are cuvânt greu în toate organismele de standardizare SQL, și, în multe privințe, deține un avans vizibil față de concurenți și uneori chiar și față de stardardul curent.

Cei mai în vârstă dintre dvs. își amintesc că prima fiolă de SQL, apărută în mai 2000, se numea *Trei diferențe SQL între Visual FoxPro și Oracle*. Ideea era simplă. Cei mai mulți specialiști din zona bazelor de date erau fox-iști. Dintre aceștia, o parte trebuiau să treacă pe servere de baze de date, iar Oracle era liderul mondial în această tehnologie. Socoteala n-a fost tocmai rea, deși serverele de baze de date comerciale sunt încă „un pod prea îndepărtat" pentru mare parte dintre firmele românești cufundate în fiscalitate și recesiune.

Versiunea Oracle a primelor fiole a fost 8. Apoi 8*i*2. La finele anului trecut compania a lansat Oracle 9*i* pentru Windows 2000 (variantele 9*i* pentru Linux și alte platforme non-Microsoft au ieșit cu destule luni mai devreme). Această nouă (9) versiune aduce cu sine elemente în premieră, dar și optimizează multe opțiuni și comenzi prezente mai de demult. Se spune, bunăoară, că administrarea bazelor de date în 9*i* este cu 45% mai ușoară prin comparație cu 8*i*. Noi, însă, suntem interesați doar de câteva dintre găselnițe SQL proaspăt apărute sau „recondiționate".

Pentru exemplele din acest material ne vom servi, pentru început, de două tabele expuse în premieră națională în numărul 94 - iulie 2000 - a *PC* (pe vreme aceea) *Report*-ului. A treia fiolă de SQL se numea *Joncțiunea externă și urmările ei* (vezi *http://www.agora.ro/pcrep/pcrep94/054.shtml* și *http://www.netreport.ro/surse/netreport114*) și poate servi drept element de comparație pentru cele ce urmează.

Joncțiune internă (echijoncțiune), adică, mai pe românește, INNER JOIN

N-am înțeles niciodată motivul pentru care cei de la Oracle s-au încăpățânat să nu implementeze operatorul `INNER JOIN` din SQL-92, astfel încât, până la 9*i*, joncțiunea internă era formulată exclusiv ca pe vremea lui SQL-89. Spre exemplu, sporurile de noapte ale salariaților, pentru luna iunie 2001 (figura 1), se obțineau prin interogarea:

```
SELECT personal2.*, spornoapte as sp_noapte_iunie
FROM personal2, sporuri
WHERE personal2.marca = sporuri.marca
AND an = 2001 AND luna = 6
```

Acum, în sfârșit, s-a produs alinierea. Începând cu Oracle 9*i* putem tasta liniștiți:

Figura 2. Semnalizarea omiterii condiției de joncțiune

```
SQL> SELECT personal2.*, spornoapte as sp_noapte_iunie
  2  FROM personal2 INNER JOIN sporuri
  3  WHERE an = 2001 AND luna = 6
  4
SQL> /
WHERE an = 2001 AND luna = 6
*
ERROR at line 3:
ORA-00905: missing keyword
```

```
SELECT personal2.*, spornoapte as sp_noapte_iunie
FROM personal2 INNER JOIN sporuri
ON personal2.marca = sporuri.marca
WHERE an = 2001 AND luna = 6
```

sau:

```
SELECT personal2.*, spornoapte as sp_noapte_iunie
FROM personal2 INNER JOIN sporuri
ON personal2.marca = sporuri.marca
AND an = 2001 AND luna = 6
```

Jonathan Gennick face, în numărul pe noiembrie/decembrie 2001 al *Oracle Magazine*, apologia acestui mod de notație, încât te întrebi de ce, totuși, așa de târziu s-au decis producătorii să se alinieze la standard. Într-adevăr, avantajele sunt evidente. Unul dintre cele mai importante îi ajută pe cei neatenți/obosiți/etc. Mie unuia mi s-a întâmplat în tinerețe să enumăr tabelele în `FROM` și să uit specificarea condiției de joncțiune în `WHERE`. De aici un ditamai produs cartezian care, la tabele mari, aproape că paralizează resursele. Iată, în figura 2, ce se întâmplă dacă se omite specificarea condiției (acum, obligatorie în `FROM`) în `INNER JOIN`.

Joncțiune naturală

Deși o predau de mulți ani studenților (începând din capitolul dedicat algebrei relaționale), mult timp nu am știut că joncțiunea naturală - operatorul `NATURAL JOIN` - face parte din SQL-92. Nefiind prea folosită, îmi permit să vă reamintesc că joncțiunea naturală este un caz particular al joncțiunii interne, „normale" (care e denumită științific echi-joncțiune - și trebuie să recunoașteți că sună bine). Rezultatul joncțiunii interne - `INNER JOIN` - poate conține ambele valori ale

Figura 1. Sporuri de noapte pe luna iunie 2001

MARCA	NUMEPREN	DATANAST	COMPART	MARCASEF	SALTARIFAR	DATA_SV	SP_NOAPTE_IUNIE
1	ANGAJAT 1	1-Jul-62	DIRECTIUNE		6000000	11-Jul-87	0
2	ANGAJAT 2	11-Oct-77	FINANCIAR	1	4500000	15-Jun-01	0
4	ANGAJAT 4		FINANCIAR	2	3800000	1-Oct-72	150000
5	ANGAJAT 5	30-Apr-65	FINANCIAR	2	4200000	31-Aug-93	150000
10	ANGAJAT 10	29-Jan-72	RESURSE UMANE	1	3700000	1-Jul-94	80000

Figura 3. Interzicea prefixării atributului de joncțiune internă

```
SQL> SELECT personal2.*, spornoapte as sp_noapte_iunie
  2  FROM personal2 INNER JOIN sporuri USING ( marca )
  3  WHERE an = 2001 AND luna = 6
  4  /
SELECT personal2.*, spornoapte as sp_noapte_iunie
                    *
ERROR at line 1:
ORA-25154: column part of USING clause cannot have
   qualifier
```

Figura 4. Eroarea prefixării atributului de joncțiune naturală

```
SQL> SELECT personal2.*, spornoapte as sp_noapte_iunie
  2  FROM personal2 NATURAL JOIN sporuri
  3  WHERE an = 2001 AND luna = 6
  4  /
SELECT personal2.*, spornoapte as sp_noapte_iunie
                    *
ERROR at line 1:
ORA-25155: column used in NATURAL join cannot have
   qualifier
```

atributului de legătură, deși sunt egale). În cazul de mai sus, dacă în clauza `SELECT` a interogării s-ar fi scris `SELECT *...`, atunci în rezultat apărea și `PERSONAL2.Marca` și `SPORURI.Marca`, deși este evident că cele două valori sunt egale în cazul joncțiunii interne.

Joncțiunea naturală extrage automat o singură dată fiecare dintre atributele de joncțiune. Aceasta atrage însă imposibilitatea prefixării, în clauzele `SELECT`, `WHERE` sau `GROUP BY / HAVING`, a atributului sau atributelor de joncțiune cu numele oricăreia dintre tabele.

Dar să începem ușor. Ultima consultare (funcțională) poate fi redactată, apelând la joncțiunea naturală astfel:

```
SELECT marca, numepren, datanast,
spornoapte as sp_noapte_iunie
FROM personal2 NATURAL JOIN sporuri
WHERE an = 2001 AND luna = 6
```

Gennick e destul de pornit pe joncțiunea naturală, includerea acesteia în stardardele SQL taxând-o drept greșeală a ANSI. Aceasta pe motiv că SQL-ul este orb, luând automat, ca și coloane de comparație, toate cele care au același nume. Or, spune Gennick, este posibil ca un utilizator să joncționeze natural două tabele fără a ști că, în afara atributului dorit, mai există și un altul care compromite rezultatul. Parafrazându-l pe Andrei Pleșu în răspunsul, publicat în Dilema, celebrei scrisori a lui G.M. Tamas, aș zice: *Dragă Jonathan, quand même !* Cum adică, utilizatorul nu știe atributele tabelelor pe care le joncționează ? Sau să existe vreun atribut „ascuns“ ? Haida-de ! (asta nu face parte din răspunsul lui A. Pleșu). Am văzut câteva capodopere de tabele cu cincizeci-șaizeci de atribute, dar acestea erau, de cele mai multe ori, improvizații disperate ale unor informaticeni copleșiți de schimbările de legislație și/sau pretențiile informaționale ale șefilor .

Așa că eu vă recomand (dumneavostră, nu lui Jonathan Gennick) să vă bucurați de câte ori aveți prilejul de sintaxa laconică a joncțiunii naturale.

Pentru operatorul `INNER JOIN`, ca alternativă la clauza `ON` poate fi folosită o alta - `USING`, caz în care atributului de joncțiune nu mai poate fi prefixat de numele tabelei. În figura 3 este prezentată o situație în care s-a recurs la `USING`, dar atributul `Marca` este prefixat (prefixarea este implicită, prin specificație `PERSONAL2.*`)

Același gen de problemă apare și la joncțiunea naturală, după cum bine se observă în figura 4.

Joncțiuni externe, COALESCE și CASE

Dacă tot s-au apucat să se alinieze la specificațiile standardului SQL:1999, „oracliștii“ s-au gândit să facă treaba până (aproape) la capăt. Și dacă tot s-au schimbat lucrurile, vreau să vă mărturisesc și eu ceva ce aveam de mult pe inimă. Nu mi-a plăcut niciodată vechiul (de-acum) stil de notație al joncțiunii externe din Oracle. Așa că bucuria mea este deplină în fața operatorilor `LEFT OUTER JOIN`, `RIGHT OUTER JOIN` și `FULL OUTER JOIN`.

După cum se observă din scriptul prezentat în listing 1, în `SPORURI` nu toți angajații prezintă înregistrări pe luna iunie 2001. Dorim, însă, ca în rezultat să fie să fie prezenți și cei „ne-prinși“ de joncțiunea internă, în dreptul lor fiind prezentă, pentru sporul de noapte, valoarea `NULL`. Astfel, apelăm la joncțiunea externă, care în Oracle pre-9*i* se redacta astfel:

```
SELECT personal2.*, spornoapte as sp_noapte_iunie
FROM personal2, sporuri
WHERE personal2.marca = sporuri.marca (+)
      AND 2001 = an (+) AND 6 = luna (+)
```

Rezultatul este cel din figura 5. Noua sintaxă a joncțiunii externe este întru totul conformă cu standardele SQL-92 și SQL:1999:

```
SELECT personal2.*, spornoapte as sp_noapte_iunie
FROM personal2 LEFT OUTER JOIN sporuri
ON personal2.marca = sporuri.marca
AND an = 2001 AND luna =6
```

Firește, în funcție de situație, poate fi folosită și joncțiunea externă la dreapta - `RIGHT OUTER JOIN`. Cât privește joncțiunea externă „plină“ - și la stânga și la dreapta - acest lucru nu era posibil direct, ci folosind o variantă de genul:

Figura 5. Joncțiune externă la stânga între PERSONAL2 și SPORURI

MARCA	NUMEPREN	DATANAST	COMPART	MARCASEF	SALTARIFAR	DATA_SV	SP_NOAPTE_IUNIE
1	ANGAJAT 1	1-Jul-62	DIRECTIUNE		6000000	11-Jul-87	0
2	ANGAJAT 2	11-Oct-77	FINANCIAR	1	4500000	15-Jun-01	0
3	ANGAJAT 3	22-Aug-62	MARKETING	1	4500000	1-Mar-79	
4	ANGAJAT 4		FINANCIAR	2	3800000	1-Oct-72	150000
5	ANGAJAT 5	30-Apr-65	FINANCIAR	2	4200000	31-Aug-93	150000
6	ANGAJAT 6	9-Nov-65	FINANCIAR	5	3500000	15-Sep-90	
7	ANGAJAT 7		FINANCIAR	5	2800000	17-May-96	
8	ANGAJAT 8	31-Dec-60	MARKETING	3	2900000	24-Sep-86	
9	ANGAJAT 9	28-Feb-76	MARKETING	3	4100000	1-Dec-00	
10	ANGAJAT 10	29-Jan-72	RESURSE UMANE	1	3700000	1-Jul-94	80000

```
SELECT personal2.*, spornoapte as sp_n_iun01
FROM personal2, sporuri
WHERE personal2.marca = sporuri.marca (+)
    AND 2001 = AN (+) AND 6 = luna (+)
UNION
 SELECT NULL, NULL, NULL, NULL, NULL,
 NULL, NULL, spornoapte
 FROM sporuri
 WHERE marca NOT IN
  (SELECT  marca
   FROM personal2)
```

Este drept, că exemplul e un pic forțat. Joncțiunea externă ar extrage și liniile din SPORURI pentru iunie 2001 care nu au echivalent în PERSONAL2, situație imposibilă câtă vreme există o restricție referențială declarată. Așa că, pentru un plus de interes, dezactivăm restricția referențială între SPORURI și PERSONAL2 și introducem o linie orfană în tabela copil:

```
ALTER TABLE sporuri
DISABLE CONSTRAINT SYS_C002718 ;
INSERT INTO SPORURI
  VALUES (2001, 6, 99, 100000, 400000, NULL, 0);
COMMIT ;
```

Rezultatul interogării de mai sus este cel din figura 6. Ultima linie apare ca urmare a înregistrării orfane din SPORURI.

Acum fraza de mai sus se poate rescrie folosind operatorul FULL OUTER JOIN. Atenție, însă, la modul de redactare ! Din cauza selectării înregistrărilor numai pentru luna iunie a lui 2001, interogarea următoare nu furnizează răspunsul corect:

```
SELECT personal2.*, spornoapte as sp_n_iun01
FROM personal2 FULL OUTER JOIN sporuri ON
   personal2.marca = sporuri.marca AND
sporuri.an = 2001 AND sporuri.luna = 6
```

Nici cea care urmează nu-i mai „brează“ (scuzați versificația):

```
SELECT personal2.*, spornoapte as sp_n_iun01
FROM personal2 FULL OUTER JOIN sporuri
 ON personal2.marca = sporuri.marca
 AND sporuri.an = 2001 AND sporuri.luna = 6
WHERE  sporuri.an = 2001 AND sporuri.luna = 6
```

Pentru a evita problemele, cel mai la îndemână truc ține de folosirea unei subconsultări în clauza FROM prin care, din SPORURI, sunt decupate numai liniile care interesează:

```
SELECT personal2.*, spornoapte as sp_n_iun01
FROM personal2 FULL OUTER JOIN
(SELECT *
 FROM sporuri
 WHERE an = 2001 AND luna = 6
) sp_iun
ON personal2.marca = sp_iun.marca
```

Ca și în celelate forme de joncțiune se poate folosi USING, însă fără specificarea alias-urilor:

```
SELECT marca, numepren, compart, saltarifar,
spornoapte as sp_n_iun01
FROM personal2 FULL OUTER JOIN
   (SELECT *
    FROM sporuri
    WHERE an = 2001 AND luna = 6
   ) sp_iun USING  (marca )
```

La completarea liniilor rezultatului cu valorile atributelor unei tabele ce nu au corespondent în cealaltă tabelă din joncțiune, valorile din oficiu sunt NULL. Pentru ca, pentru luna iunie 2001, valorile sporului de noapte ale persoanele care nu lucrau în firmă (care nu prezintă înregistrări în SPORURI) să prezinte valori 0, nu NULL, tradițional, se foloseau funcțiile DECODE sau NVL.

```
SELECT personal2.marca,
NVL(numepren, 'ANGAJAT FARA NUME !') AS Numepren,
   compart, saltarifar, NVL(spornoapte,0) as sp_n_iun01
FROM personal2 FULL OUTER JOIN
(SELECT * FROM sporuri
 WHERE an = 2001 AND luna = 6
) sp_iun
ON personal2.marca = sp_iun.marca
```

Rezultatul este cel din figura 7.
9*i* pune la dipoziție funcția COALESCE - consacrată de SQL-92 și DB2:

```
SELECT personal2.marca,
   COALESCE(numepren, 'ANGAJAT FARA NUME !') AS Numepren,
   compart, saltarifar,
   COALESCE(spornoapte,0) as sp_n_iun01
FROM personal2 FULL OUTER JOIN
(SELECT *
  FROM sporuri
  WHERE an = 2001 AND luna = 6
) sp_iun
ON personal2.marca = sp_iun.marca
```

Firește, se poate folosi și o structură de tip CASE apărută încă din Oracle 8*i*:

Figura 6. Joncțiune externă totală între PERSONAL2 și SPORURI

MARCA	NUMEPREN	DATANAST	COMPART	MARCASEF	SALTARIFAR	DATA_SV	SP_N_IUN01
1	ANGAJAT 1	1-Jul-62	DIRECTIUNE		6000000	11-Jul-87	0
2	ANGAJAT 2	11-Oct-77	FINANCIAR	1	4500000	15-Jun-01	0
3	ANGAJAT 3	22-Aug-62	MARKETING	1	4500000	1-Mar-79	
4	ANGAJAT 4		FINANCIAR	2	3800000	1-Oct-72	150000
5	ANGAJAT 5	30-Apr-65	FINANCIAR	2	4200000	31-Aug-93	150000
6	ANGAJAT 6	9-Nov-65	FINANCIAR	5	3500000	15-Sep-90	
7	ANGAJAT 7		FINANCIAR	5	2800000	17-May-96	
8	ANGAJAT 8	31-Dec-60	MARKETING	3	2900000	24-Sep-86	
9	ANGAJAT 9	28-Feb-76	MARKETING	3	4100000	1-Dec-00	
10	ANGAJAT 10	29-Jan-72	RESURSE UMANE	1	3700000	1-Jul-94	80000
							400000

Figura 7. Joncțiune externă și folosirea funcției NVL

MARCA	NUMEPREN	COMPART	SALTARIFAR	SP_N_IUN01
1	ANGAJAT 1	DIRECTIUNE	6000000	0
2	ANGAJAT 2	FINANCIAR	4500000	0
4	ANGAJAT 4	FINANCIAR	3800000	150000
5	ANGAJAT 5	FINANCIAR	4200000	150000
10	ANGAJAT 10	RESURSE UMANE	3700000	80000
8	ANGAJAT 8	MARKETING	2900000	0
3	ANGAJAT 3	MARKETING	4500000	0
6	ANGAJAT 6	FINANCIAR	3500000	0
7	ANGAJAT 7	FINANCIAR	2800000	0
9	ANGAJAT 9	MARKETING	4100000	0
	ANGAJAT FARA NUME !			400000

```
SELECT personal2.marca,
    CASE
        WHEN numepren IS NULL
        THEN 'ANGAJAT FARA NUME !'
    ELSE numepren
    END as Numepren,
    compart, saltarifar,
    CASE
        WHEN spornoapte IS NULL THEN 0
        ELSE spornoapte
    END as sp_n_iun01
FROM personal2 FULL OUTER JOIN
(SELECT *
 FROM sporuri
 WHERE an = 2001 AND luna = 6
) sp_iun
ON personal2.marca = sp_iun.marca
```

Acest format al structurii de tip `CASE` nu este întru totul conform cu standardele SQL, deoarece condiția poate fi exprimată în întregime numai după un `WHEN`. Din 9*i*, atunci când `CASE`-ul are în vedere valoarea unui singur atribut/expresie, atunci numele acestuia sau expresia poate fi scris/scrisă imediat după `CASE`, în timp ce după fiecare `WHEN` este indicată o valoare etalon. Dar mai bine să vedem cum poate fi reformulată fraza `SELECT` anterioară:

```
SELECT personal2.marca,
    CASE numepren
        WHEN NULL THEN 'ANGAJAT FARA NUME !'
        ELSE numepren
    END as Numepren,
    compart, saltarifar,
    CASE spornoapte
```

Figura 9. Probleme la folosirea NVL

```
SQL> SELECT personal2.marca,
  2  NVL(numepren, CAST(personal2.marca AS VARCHAR2(15)),
  3  'ANGAJAT FARA NUME SI MARCA !') AS Numepren,
  4  compart, saltarifar, NVL(spornoapte,0) as sp_n_iun01
  5  FROM personal2 FULL OUTER JOIN
  6  (SELECT *
  7  FROM sporuri
  8  WHERE an = 2001 AND luna = 6
  9  ) sp_iun  ON personal2.marca = sp_iun.marca
 10  /
NVL(numepren, CAST(personal2.marca AS VARCHAR2(15)),
*
ERROR at line 2:
ORA-00909: invalid number of arguments
```

Figura 8. Rezultat defectuos în noua variantă CASE (Oracle 9i)

MARCA	NUMEPREN	COMPART	SALTARIFAR	SP_N_IUN01
1	ANGAJAT 1	DIRECTIUNE	6000000	0
2	ANGAJAT 2	FINANCIAR	4500000	0
4	ANGAJAT 4	FINANCIAR	3800000	150000
5	ANGAJAT 5	FINANCIAR	4200000	150000
10	ANGAJAT 10	RESURSE UMANE	3700000	80000
8	ANGAJAT 8	MARKETING	2900000	
3	ANGAJAT 3	MARKETING	4500000	
6	ANGAJAT 6	FINANCIAR	3500000	
7	ANGAJAT 7	FINANCIAR	2800000	
9	ANGAJAT 9	MARKETING	4100000	
				400000

```
        WHEN NULL THEN 0
        ELSE spornoapte
    END as sp_n_iun01
FROM personal2 FULL OUTER JOIN
(SELECT *
 FROM sporuri
 WHERE an = 2001 AND luna = 6
 ) sp_iun
  ON personal2.marca = sp_iun.marca
```

Ei bine, rezultatul nu este corect ! Pur și simplu, secvența de substituire a valorilor nule nu funcționează. Dacă nu mă credeți, priviți la figura 8. Asta e ! Legat de raportul `NVL - COALESCE`, în afară de simpla aliniere la standard, probabil că vă întrebați dacă ar fi vreun avantaj suplimentar al folosirii `COALESCE` în detrimentul primeia ?

Pentru ca exemplele următoare să fie mai de efect, mai întâi NULL-izăm nume angajatului cu marca 9.

```
UPDATE PERSONAL2
SET NUMEPREN=NULL
WHERE MARCA=9 ;
```

Acum ne propunem ca, în rezultatul joncțiunii totale, dacă numele angajatului este `NULL`, în locul său să apară marca, iar dacă și aceasta este `NULL`, pe linia respectivă, valoarea atributului `NumePren` să fie una care să semnalizeze că este vorba de o linie inclusă ca urmare a joncțiunii externe: `'ANGAJAT FARA NUME SI MARCA !'`. Practic, am dori ca interogarea să obțină un rezultat ca în figura 10. Pentru coloana corespunzătoare atributului `NumePren`, soluția la care suntem tentați să recurgem se bazează pe funcția `NVL` aplicată asupra acestui atribut. Problema ține de faptul că, atunci când `NumePren IS NULL`, trebuie continuată testarea nulității atributului `Marca`. Dacă `Marca` este nenulă, se va

Figura 10. Utilizarea funcției COALESCE

MARCA	NUMEPREN	COMPART	SALTARIFAR	SP_N_IUN01
1	ANGAJAT 1	DIRECTIUNE	6000000	0
2	ANGAJAT 2	FINANCIAR	4500000	0
4	ANGAJAT 4	FINANCIAR	3800000	150000
5	ANGAJAT 5	FINANCIAR	4200000	150000
10	ANGAJAT 10	RESURSE UMANE	3700000	80000
8	ANGAJAT 8	MARKETING	2900000	0
3	ANGAJAT 3	MARKETING	4500000	0
6	ANGAJAT 6	FINANCIAR	3500000	0
7	ANGAJAT 7	FINANCIAR	2800000	0
9	9	MARKETING	4100000	0
	ANGAJAT FARA NUME SI MARCA !			400000

afișa ca atare (ca valoarea a NumePren); în caz contrar, trebuie afișat mesajul lămuritor de acum câteva rânduri.

Varianta bazată pe NVL din figura 9 este sortită eșecului.

Eroarea ține de faptul că NVL poate testa nulitatea unui singur atribut/expresie. Ceea ce nu este cazul funcției COALESCE, un soi de generalizare a NVL:

```
SELECT personal2.marca,
 COALESCE (numepren, CAST(personal2.marca AS VARCHAR2 (15)),
'ANGAJAT FARA NUME SI MARCA !') AS Numepren,
 compart, saltarifar, NVL(spornoapte,0) as sp_n_iun01
FROM personal2 FULL OUTER JOIN
(SELECT *
 FROM sporuri
 WHERE an = 2001 AND luna = 6
) sp_iun  ON personal2.marca = sp_iun.marca
```

După numele funcției, pot fi enumerate pe rând oricâte atribute/expresii, fiind extrasă în rezultat prima valoare nenulă dintre atributele/expresiile indicate (în ordinea specificării).

Interogări scalare

O altă facilitate proaspătă în Oracle, dar prezentă în alte produse de câțiva ani (cum ar fi DB2-ul IBM-ului) o reprezintă interogărilor scalare specificate chiar în clauza SELECT a consultării. Dar, înainte de a trece la exemplul lămuritor, readucem la „normal" numele angajatului cu marca 9:

```
UPDATE PERSONAL2
SET NUMEPREN='ANGAJAT 9'
WHERE MARCA = 9 ;
COMMIT ;
```

Pentru a ilustra această facilitate, ne propunem să calculăm procentul sporului de noapte pe luna iunie 2001 al fiecărui angajat, din suma totală a sporurilor de noapte acordate pe această lună. În lipsa interogărilor scalare, e necesar să folosim subconsultări plasate în clauza FROM a interogării principale:

```
SELECT personal2.marca,
   COALESCE(numepren, 'ANGAJAT FARA NUME !') AS Numepren,
   compart, saltarifar,
   COALESCE(spornoapte,0) as sp_n_iun01,
   total_sn,
ROUND (COALESCE(spornoapte,0) / total_sn, 4) * 100
                                        AS Procent_sn
FROM
(personal2 FULL OUTER JOIN
 (SELECT * FROM sporuri
  WHERE an = 2001 AND luna = 6
 ) sp_iun
ON personal2.marca = sp_iun.marca)
   INNER JOIN
(SELECT SUM(spornoapte) AS total_sn
FROM sporuri
WHERE an = 2001 AND luna = 6
) total_sn ON 1=1
```

Figura 11. Procentele individuale din totalul sporului de noapte pe iunie 2001

MARCA	NUMEPREN	COMPART	SAL-TARIFAR	SP_N_IUN01	TOTAL_SN	PRO-CENT_SN
1	ANGAJAT 1	DIRECTIUNE	6000000	0	780000	0
2	ANGAJAT 2	FINANCIAR	4500000	0	780000	0
4	ANGAJAT 4	FINANCIAR	3800000	150000	780000	19.23
5	ANGAJAT 5	FINANCIAR	4200000	150000	780000	19.23
10	ANGAJAT 10	RESURSE UMANE	3700000	80000	780000	10.26
8	ANGAJAT 8	MARKETING	2900000	0	780000	0
3	ANGAJAT 3	MARKETING	4500000	0	780000	0
6	ANGAJAT 6	FINANCIAR	3500000	0	780000	0
7	ANGAJAT 7	FINANCIAR	2800000	0	780000	0
9	ANGAJAT 9	MARKETING	4100000	0	780000	0
	ANGAJAT FARA NUME!			400000	780000	51.28

Tabela ad-hoc obținută în clauza FROM a frazei principale a fost denumită SP_IUN (de la SPORURI_IUNIE) și conține toți angajații firmei și suma pe care au primit-o drept spor de noapte pe luna iunie 2001. A doua tabelă ad-hoc este mai mult ad-hoc decât tabelă, deoarece conține o singură coloană (total_sn) și o singură linie, de fapt, suma sporurilor de noapte pe luna cu pricina. Știind că TOTAL_SN (pseudo-tabela are același nume ca și singura coloană ce o alcătuiește) are o singură linie, pentru corectitudinea rezultatului am apelat la un truc ieftin: joncționarea SP_IUN cu TOTAL_SN, condiția de joncțiune fiind nici mai mult, nici mai puțin, 1 = 1, ceea ce echivalează cu un produs cartezian al celor două tabele ad-hoc. Iată și rezultatul în figura 11.

Beneficiind de serviciile subconsultărilor scalare, varianta de mai sus poate fi rescrisă astfel:

```
SELECT personal2.marca,
   COALESCE(numepren, 'ANGAJAT FARA NUME !') AS Numepren,
   compart, saltarifar,
   COALESCE(spornoapte,0) as sp_n_iun01,
   ROUND( COALESCE(spornoapte,0) /
      (SELECT SUM(spornoapte) AS total_sn
       FROM sporuri
       WHERE an = 2001 AND luna = 6
      ), 4   ) * 100 AS Procent_sn
FROM (personal2 FULL OUTER JOIN
 (SELECT *
  FROM sporuri
  WHERE an = 2001 AND luna = 6
  ) sp_iun   ON personal2.marca = sp_iun.marca     )
```

Practic, subconsultatea care calculează totalul sporurilor de noapte pentru luna de referință a fost mutată din clauza FROM în SELECT.

MERGE

Despre comanda MERGE nu prea am auzit discutându-se în „cetatea" SQL-ului; primul exemplu l-am văzut de curând, citind articolul dedicat PL/SQL din Oracle 9*i*, publicat de Steven Feuerstein în numărul ianuarie/februarie 2002 al Oracle Magazine. Pentru a fi mai elocvenți, să luăm un exemplu razant cu realitatea. Într-o firmă mare, cu multe secții/filiale, salariile se calculează pe baza pontajelor, iar fiecare secție are pontajul său. Tabela corespunzătoare poate avea o schemă de genul:

Figura 12. Centralizarea pontajelor secției 1

AN	LUNA	MARCA	ORELUCRATE	ORENOAPTE
2001	6	1	24	0
2001	6	2	24	0
2001	6	3	24	0
2001	6	4	24	0
2001	6	6	24	0
2001	6	7	16	5
2001	6	8	8	0
2001	6	9	16	8

```
CREATE TABLE pontaje (
   sectie INTEGER,
   marca INTEGER CONSTRAINT fk_pontaje_personal2
      REFERENCES personal2(marca),
   data DATE,
   orelucrate INTEGER,
   orenoapte INTEGER,
   CONSTRAINT pk_pontaje PRIMARY KEY (sectie, marca, data)
   ) ;
```

Mulți dezvoltatori de aplicații dedicate salarizării obișnuiesc să recurgă la denormalizare, folosind o tabelă centralizatoare, actualizată prin declanșatoare sau proceduri speciale, tabelă extrem de utilă atunci când interesează retrospectiv (cu o lună, un an etc. în urmă) date sintetice privind salariile angajaților. O asemenea tabelă ar putea avea structura:

```
CREATE TABLE salarizare (
   an INTEGER,
   luna INTEGER,
   marca INTEGER CONSTRAINT fk_salarizare_personal2
REFERENCES personal2(marca),
   orelucrate INTEGER,
   orenoapte INTEGER,
   sal_baza INTEGER,
   sporuri INTEGER,
   retineri INTEGER,
   rest_pl INTEGER ,
   CONSTRAINT pk_salarizare PRIMARY KEY (an, luna, marca)
   ) ;
```

Conform procedurilor de utilizare a aplicației, mai întâi se preiau pontajele (ne-am oprit numai la zilele 1, 2 și 3 iunie 2001) secției 1 – pe Web.

„Vărsarea“ acestor înregistrări, ce corespund lunii iunie 2001 și secției 1, în tabela SALARIZARE, care în acest moment nu are nici o înregistrare, se poate realiza prin comanda MERGE astfel:

```
MERGE INTO salarizare S  USING
(SELECT TO_NUMBER(TO_CHAR(data,'YYYY')) AS an,
 TO_NUMBER(TO_CHAR(data, 'MM')) AS luna,
 marca, SUM(orelucrate) AS orelucrate,
 SUM(orenoapte) AS orenoapte
  FROM pontaje
  WHERE TO_NUMBER(TO_CHAR(data,'YYYY')) = 2001
  AND TO_NUMBER(TO_CHAR(data, 'MM')) = 6
  AND sectie = 1
  GROUP BY TO_NUMBER(TO_CHAR(data,'YYYY')),
         TO_NUMBER(TO_CHAR(data, 'MM')), marca
) P
ON (S.an = P.an AND S.luna=P.luna AND S.marca = P.marca )
WHEN MATCHED THEN
 UPDATE SET S.orelucrate = S.orelucrate + P.orelucrate,
            S.orenoapte = S.orenoapte + P.orenoapte
WHEN NOT MATCHED THEN
 INSERT (S.an, S.luna, S.marca, S.orelucrate, S.orenoapte)
 VALUES (P.an, P.luna, P.marca, P.orelucrate, P.orenoapte)
```

Noul conținut al primelor cinci atribute din tabela SALARIZARE este cel din figura 12.

Înainte de a explica mecanismul comenzii, trecem la secția 2, ale cărei pontaje se preiau ulterior. Pentru simplificare, în listing 3 ne oprim numai la trei linii (zile de pontaj).

Elementul de interes ține de faptul că angajații 7 și 9 au lucrat și la secția 1, deci noile ore lucrate și de noapte trebuie să se cumuleze. Însă angajatul cu marca 10 nu are nici o zi lucrată in secția 1, prin urmare în

Figura 13. „Vărsarea“ pontajelor secției 2

AN	LUNA	MARCA	ORELUCRATE	ORENOAPTE
2001	6	10	8	5
2001	6	1	24	0
2001	6	2	24	0
2001	6	3	24	0
2001	6	4	24	0
2001	6	6	24	0
2001	6	7	24	9
2001	6	8	8	0
2001	6	9	24	8

SALARIZARE îi trebuie adăugată o înregistrare specială. În urmă re-execuției comenzii MERGE, schimbând numai una din condițiile de selecție pentru tabela PONTAJE - sectie = 2 – conținutul tabelei-centralizatoare este cel din figura 13.

Care este, deci, mecanismul de funcționare a comenzii MERGE ? Tabela specificată după clauza INTO este SALARIZARE, ceea ce înseamnă că acualizările se vor face în această tabelă căreia i s-a atribuit sinonimul S. După USING, printr-o frază SELECT, a fost specificată o tabelă ad-hoc cu aliasul P, tabelă ce conține liniile din PONTAJE referitoare la luna iunie 2001 și secția 1 (la prima lansare), respectiv 2 (la a doua lansare). Pentru a putea face comparația tabelelor cu aliasurile S și P, această din urmă tabelă conține, pe lângă suma orelor lucrate și de noapte pentru fiecare angajat, coloanele An, Luna și Marcă. Primele două sunt obținute din conversii ale atributului Data.

Predicatul de comparație dintre cele două alias-uri este S.an = P.an AND S.luna=P.luna AND S.marca = P.marca. La prima execuție a comenzii, tabela SALARIZARE este vidă, așa că se va execută comanda de pe „ramura“ WHEN NOT MATCHED, inserându-se câte o înregistrare pentru fiecare linie din P.

La a doua execuție (când secție = 2), pentru angajații 7 și 9 există deja linii în S (SALARIZARE), așa că se va urma ramura WHEN MATCHED care incrementează orele lucrate și cele de noapte. Pentru angajatul 10, nefiind înregistrare corespondentă, va fi executată ramura WHEN MATCHED.

Acestea ar fi doar câteva noutăți SQL din Oracle 9*i*. Ar mai merita amintite măcar:

- noua funcție care întoarce data sistem - CURRENT_DATE, și alte câteva opțiuni funcții de lucru cu intervale calendaristice: INTERVAL, EXTRACT s.a.;
- tipul de dată TIMESTAMP, și grozav de util la jurnalizarea operațiunilor;
- suport pentru tipuri de date XML etc.

Pentru acestea, ca și pentru restul, deocamdată nu e nici spațiu, nici timp, nici energie...

Notă: Listingurile 1-3 sunt disponibile pe web la: http://www.netreport.ro/surse.

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. ■ 10

Bibliografie

- Jonathan Gennick – **ANSI Standard SQL Joins**, Oracle Magazine, Nov-Dec 2001
- Steven Feuerstein - **Oracle 9i PL/SQL Transforms and Performs**, Oracle Magazine, Jan-Feb 2002
- Oracle Corporation - **Oracle 9i SQL Reference**, Release 1 (9.0.1), June 2001