

## Capitolul 5. Actualizarea tabelelor prin comenzi SQL

După crearea tabelelor și implementarea mecanismului de asigurare a integrității și coerenței bazei de date se poate trece la folosirea sa efectivă, prin preluarea datelor legate de obiectele, procesele și tranzacțiile care constituie substanța aplicației propriu-zise. Și în Oracle sistemul “clasic” de actualizare a tabelelor se bazează pe trei comenzi SQL, `INSERT`, `UPDATE` și `DELETE`. Formatul general al acestora este însă unul generos, cu atât mai mult cu cât fiecare poate folosi și subconsultări.

### 5.1. Adăugarea de înregistrări - `INSERT`

Cum e și normal, începem capitolul cu prezentarea comenzii prin care o tabelă poate fi populată cu înregistrări. Pe parcursul capitolului precedent am făcut apel în câteva rânduri la comanda `INSERT` pentru a pune în evidență modul în care Oracle reacționează la încălcarea restricțiilor declarate.

#### 5.1.1. Două formate

Cea mai frecventă formă a comenzii `INSERT` este:

```
INSERT INTO tabelă  
VALUES (val_atribut1, val_atribut_2, ..., val_atribut_n)
```

Dacă valorile enumerate sunt mai multe sau mai puține decât atributele tabelului `PERSONAL`, atunci se declanșează erorile din figura 5.1. În primul caz, apare o valoare zero inutilă, ce nu poate fi asociată nici unui atribut, iar în al doilea lipsește valoarea ultimului atribut (`Colaborator`).

De asemenea, ordinea enumerării valorilor trebuie să coincidă cu ordinea în care au fost declarate atributele la crearea tabelului. Pe parcursul capitolului precedent a fost folosit și un al doilea format, cel în care se declară atributele ce urmează a primi valori în clauza `VALUES`. Astfel, în urma comenzii `INSERT` următoare, în linia nou introdusă, atributele `Compart`, `SalOrar`, `SalOrarCO` și `Colaborator` vor primi, după caz, valorile specificate în clauzele `DEFAULT` sau valoarea `NULL`.

```
INSERT INTO personal (marca, numepren, datasv)  
VALUES (104, 'Angajat 4', DATE'1982-01-05') ;
```

Reamintim că fiecărui atribut declarat `NOT NULL`, și fără de valoare implicită (clauza `DEFAULT`), trebuie să i se precizeze o valoare în clauza `VALUES`.

---

```

SQL> INSERT INTO personal VALUES (101, 'Angajat 1', 'IT',
  2   TO_DATE('12/10/1980', 'DD/MM/YYYY'), 56000, 55000, 'N', 0)
  3 /
INSERT INTO personal VALUES (101, 'Angajat 1', 'IT',
                               *
ERROR at line 1:
ORA-00913: too many values

SQL> INSERT INTO personal VALUES (101, 'Angajat 1', 'IT',
  2   TO_DATE('12/10/1980', 'DD/MM/YYYY'), 56000, 55000)
  3 /
INSERT INTO personal VALUES (101, 'Angajat 1', 'IT',
                               *
ERROR at line 1:
ORA-00947: not enough values

SQL>

```

---

Figura 5.1. Cele două erori datorate necordanței numărului de valori și de atribute

### 5.1.2. Inserări și secvențe

Secvența este un obiect al unei baze de date care generează valori unice. Are un regim public, așa că în aplicațiile cu număr mare de utilizatori orice atribut de tip cheie surogat (așa cum este PERSONAL.Marca) poate beneficia de serviciile sale. Pentru exemplificare, creăm o secvență care va genera mărci, astfel încât fiecare marcă să fie unică:

```

CREATE SEQUENCE seq_marca
INCREMENT BY 1
MINVALUE 101 MAXVALUE 5555
NOCYCLE NOCACHE ORDER ;

```

Numele secvenței este SEQ\_MARCA. Valorile generate vor fi consecutive (INCREMENT BY 1), începând cu 101 și terminând cu 5555. Dacă valorile generate ating limita maximă, iar la crearea secvenței s-a folosit clauza NOCYCLE, la următorul apel al unei valori din secvență va fi generată o eroare (ORA-08004). Dacă s-ar folosi CYCLE, după atingerea limitei superioare, se reiau valorile, începând cu cea minimă. ORDER garantează generarea valorilor în ordinea în care secvența este apelată.

După crearea secvenței, o valoare nouă va fi generată folosind clauza NEXTVAL; valoarea curentă (ultima valoare generată) a secvenței poate fi obțină prin clauza CURRVAL – vezi figura 5.2.

---

```

SQL> SELECT seq_marca.NEXTVAL FROM dual ;

      NEXTVAL
-----
         101

SQL> SELECT seq_marca.NEXTVAL FROM dual ;

      NEXTVAL
-----
         102

SQL> SELECT seq_marca.CURRVAL FROM dual ;

      CURRVAL
-----
         102

SQL>

```

---

Figura 5.2. Generarea valorilor din secvență

**Atenție**, însă ! Clauza CURRVAL nu poate fi folosită decât dacă secvența fost inițializată (printr-un NEXTVAL) în sesiunea respectivă. Spre exemplu, în figura 5.3 se închide sesiunea Oracle. Apoi se deschide o nouă sesiune, după care, imediat, se lansează fraza SELECT prin care s-ar dori aflarea valorii curente din secvență. Eroarea declanșată are codul ORA-08002.

---

```

SQL> DISCONNECT
Disconnected from Personal Oracle9i Release 9.2.0.1.0 - Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.1.0 - Production
SQL>
SQL> CONNECT fotachem/fotache
Connected.
SQL>
SQL> SELECT seq_marca.CURRVAL FROM dual ;
SELECT seq_marca.CURRVAL FROM dual
*
ERROR at line 1:
ORA-08002: sequence SEQ_MARCA.CURRVAL is not yet defined in this session

SQL>

```

---

Figura 5.3. Clauza CURRVAL fără inițializarea secvenței

Există, însă, o soluție pentru aflarea valorii curente din secvență fără a recurge la NEXTVAL (ceea ce ar echivala cu pierderea unei valori). Dicționarul de date păstrează informații prețioase despre secvențele create în schema curentă în tabela virtuală USER\_SEQUENCES. Deoarece am declarat, la creare, că renunțăm la opțiunea CACHE, atributul CACHE\_SIZE va fi zero, așa încât câmpul LAST\_NUMBER din interogarea SELECT \* FROM user\_sequences va indica destul de precis ce valoare urmează a furniza secvența la prima solicitare - vezi figura 5.4.

```

SQL> DESC user_sequences
Name
-----
SEQUENCE_NAME
MIN_VALUE
MAX_VALUE
INCREMENT_BY
CYCLE_FLAG
ORDER_FLAG
CACHE_SIZE
LAST_NUMBER

SQL> SELECT * FROM user_sequences ;

SEQUENCE_N MIN_VALUE MAX_VALUE INCREMENT_BY C O CACHE_SIZE LAST_NUMBER
-----
SEQ_MARCA      101      5555          1 N Y          0      103

SQL>

```

Figura 5.4. Informații despre secvență extrase din dicționar

Succesiunea de comenzi `INSERT` din listingul 5.1 populează tabela `PERSONAL`, folosind toate variantele discutate în acest paragraf, inclusiv prin apelul la secvență. Pentru a evita duplicarea chei primare și producerea erorii cu pricina, scriptul începe prin ștergerea eventualelor înregistrări.

Listing 5.1. Script de populare a tabeli `PERSONAL`

```

DELETE FROM personal ;

INSERT INTO personal VALUES (101, 'Angajat 1', 'IT',
    TO_DATE('12/10/1980', 'DD/MM/YYYY'), 56000, 55000, 'N') ;
INSERT INTO personal VALUES (102, 'Angajat 2', 'CONTA',
    TO_DATE('12/11/1978', 'DD/MM/YYYY'), 57500, 56000, 'N') ;
INSERT INTO personal VALUES (seq_marca.NextVal, 'Angajat 3', 'IT',
    TO_DATE('02/07/1976', 'DD/MM/YYYY'), 67500, 66000, 'N') ;

-- se folosesc valorile implicite ale atributelor Compart, SalOrar, SalOrarCO si Colaborator
INSERT INTO personal (marca, numepren, datasv)
VALUES (seq_marca.NextVal, 'Angajat 4', DATE'1982-01-05') ;

INSERT INTO personal VALUES (seq_marca.NextVal, 'Angajat 5', 'IT',
    TO_DATE('12/11/1977', 'DD/MM/YYYY'), 62500, 62000, 'N') ;
INSERT INTO personal VALUES (seq_marca.NextVal, 'Angajat 6', 'CONTA',
    TO_DATE('11/04/1985', 'DD/MM/YYYY'), 71500, 70000, 'N') ;
INSERT INTO personal VALUES (seq_marca.NextVal, 'Angajat 7', 'CONTA',
    TO_DATE('21/11/1991', 'DD/MM/YYYY'), 61500, 60000, 'N') ;
INSERT INTO personal VALUES (seq_marca.NextVal, 'Angajat 8', 'PROD',
    TO_DATE('30/12/1994', 'DD/MM/YYYY'), 54500, 52000, 'N') ;

COMMIT ;

```

Ultima comandă, `COMMIT`, are drept scop finalizarea tranzacției și, în consecință, scrierea “definitivă” a liniilor în tabelă. Detalii despre mecanismul tranzacțional vă sunt oferite gratuit în alt paragraf. În urma execuției scriptului, conținutul tabeli este cel din figura 5.5.

```
SQL> SELECT * FROM personal ;
```

MARCA	NUMEPREN	COMPA	DATASU	SALORAR	SALORARCO	C
101	Angajat 1	IT	12-OCT-80	56000	55000	N
102	Angajat 2	CONTA	12-NOV-78	57500	56000	N
103	Angajat 3	IT	02-JUL-76	67500	66000	N
104	Angajat 4	PROD	05-JAN-82	75000	75000	N
105	Angajat 5	IT	12-NOV-77	62500	62000	N
106	Angajat 6	CONTA	11-APR-85	71500	70000	N
107	Angajat 7	CONTA	21-NOV-91	61500	60000	N
108	Angajat 8	PROD	30-DEC-94	54500	52000	N

8 rows selected.

```
SQL> |
```

Figura 5.5. Conținutul tabeli PERSONAL după primul apel al listingului

La drept vorbind, ideea generării mărcilor atât prin constante (la primii doi angajați), cât și prin secvență nu este prea inteligentă. Astfel, dacă lansăm în execuție a doua oară scriptul, avem toate șansele ca mărcile să fie cele din figura 5.6.

MARCA	NUMEPREN	COMPA	DATASU	SALORAR	SALORARCO	C
109	Angajat 3	IT	02-JUL-76	67500	66000	N
110	Angajat 4	PROD	05-JAN-82	75000	75000	N
111	Angajat 5	IT	12-NOV-77	62500	62000	N
112	Angajat 6	CONTA	11-APR-85	71500	70000	N
113	Angajat 7	CONTA	21-NOV-91	61500	60000	N
114	Angajat 8	PROD	30-DEC-94	54500	52000	N
101	Angajat 1	IT	12-OCT-80	56000	55000	N
102	Angajat 2	CONTA	12-NOV-78	57500	56000	N

Figura 5.6. Conținutul tabeli PERSONAL după al doilea apel al listingului

Dacă Angajat 1 și Angajat 2 își “păstrează” marcele, celelalte valori urmează cursul secvenței. Cel mai sigur este să uzităm fie numai de constante, fie numai de secvență. În plus, scriptul prezentat în listing 5.2 re-crează secvența, astfel încât întotdeauna prima marcă va fi 101.

Listing 5.2. Noul script de populare a tabeli PERSONAL

```
DELETE FROM personal ;

DROP SEQUENCE seq_marca ;

CREATE SEQUENCE seq_marca INCREMENT BY 1
  MINVALUE 101 MAXVALUE 5555 NOCYCLE NOCACHE ORDER ;

INSERT INTO personal VALUES (seq_marca.NextVal, 'Angajat 1', 'IT',
  DATE'1980-10-12', 56000, 55000, 'N') ;
INSERT INTO personal VALUES (seq_marca.NextVal, 'Angajat 2', 'CONTA',
  DATE'1978-11-12', 57500, 56000, 'N') ;
INSERT INTO personal VALUES (seq_marca.NextVal, 'Angajat 3', 'IT',
```

```

DATE'1976-07-02',67500, 66000, 'N' );
INSERT INTO personal (marca, numepren, datasv) VALUES (seq_marca.NextVal, 'Angajat 4',
DATE'1982-01-05');
INSERT INTO personal VALUES (seq_marca.NextVal, 'Angajat 5', 'IT',
DATE'1977-11-12',62500, 62000, 'N' );
INSERT INTO personal VALUES (seq_marca.NextVal, 'Angajat 6', 'CONTA',
DATE'1985-04-11',71500, 70000, 'N' );
INSERT INTO personal VALUES (seq_marca.NextVal, 'Angajat 7', 'CONTA',
DATE'1991-11-21', 61500, 60000, 'N' );
INSERT INTO personal VALUES (seq_marca.NextVal, 'Angajat 8', 'PROD',
DATE'1994-12-30', 54500, 52000, 'N' );

COMMIT ;

```

### 5.1.3. Inserare prin subconsultare

Am epuizat cam fugitiv subiectul populării cu înregistrări a tabeli PERSONAL, așa că ne grăbim să continuăm cu cea mai stufoasă dintre tabele – PONTAJE. Firește, și aici putem apela la unul din formatele mai mult sau mai puțin clasice de mai sus. Însă mult mai atractiv ar fi să adăugăm dintr-o singură mișcare câte o linie pentru fiecare angajat.

Deși 1 iulie 2003 cade într-o marți și marțea nu e prea indicată pentru începuturi, comanda INSERT următoare va avea efectul scontat:

```

INSERT INTO pontaje
SELECT marca, DATE'2003-07-01', 8, 0, 0, 0
FROM personal ;

```

Celor neîncrezători le este dedicată figura 5.7.

```

SQL> INSERT INTO pontaje
2  SELECT marca, DATE'2003-07-01', 8, 0, 0, 0
3  FROM personal
4  /

8 rows created.

SQL> SELECT * FROM pontaje ;

```

MARCA	DATA	ORELUCRATE	ORECO	ORENOAPTE	OREABSNEH
101	01-JUL-03	8	0	0	0
102	01-JUL-03	8	0	0	0
103	01-JUL-03	8	0	0	0
104	01-JUL-03	8	0	0	0
105	01-JUL-03	8	0	0	0
106	01-JUL-03	8	0	0	0
107	01-JUL-03	8	0	0	0
108	01-JUL-03	8	0	0	0

```

8 rows selected.

SQL> |

```

Figura 5.7. Inserare & consultare

Începând cu 2 iulie intră în concediu angajații cu mărcile 102 și 107:

```
INSERT INTO pontaje (marca, data, orelucrate, oreco)
  SELECT marca, DATE'2003-07-02', 8, 0
  FROM personal
  WHERE marca NOT IN (102,107)
UNION
  SELECT marca, DATE'2003-07-02', 0, 8
  FROM personal
  WHERE marca IN (102,107) ;
```

Interesant este că numele unei tabeli poate fi substituit printr-o subconsultare. Astfel, echivalent comenzii de mai sus se poate scrie:

```
INSERT INTO
  (SELECT marca, data, orelucrate, oreco FROM pontaje)
  SELECT marca, DATE'2003-07-02', 8, 0
  FROM personal
  WHERE marca NOT IN (102,107)
UNION
  SELECT marca, DATE'2003-07-02', 0, 8
  FROM personal
  WHERE marca IN (102,107) ;
```

Prin scriptul din listingul 5.3 se introduc pontaje corespunzătoare zilelor de 1, 2, 3, 4, 7 și 8 iulie și pentru lunile august și septembrie 2003. Pe 1 iulie lucrează toți angajații. Pe 2, 3, 4, 7 și 8 iulie angajații cu mărcile 102 și 107 sunt în concediu. În data de 7 iulie angajatul cu marca 103 a lucrat 4 ore în regim de noapte, iar cel cu marca 105 a chiulit (și l-a aflat șeful) 2 ore. 8 iulie este ziua în care omul muncii cu marca 104 are 2 ore de noapte, iar cel cu marca 105 a mai tras o porție de chiul, este adevărat, numai de o oră.

Listing 5.3. Script de populare a tabeli PONTAJE

```
DELETE FROM pontaje ;

-- 1 iulie 2003
INSERT INTO pontaje
  SELECT marca, DATE'2003-07-01', 8, 0, 0, 0 FROM personal ;

-- 2 iulie 2003
INSERT INTO pontaje (marca, data, orelucrate, oreco)
  SELECT marca, DATE'2003-07-02', 8, 0 FROM personal WHERE marca NOT IN (102,107)
  UNION SELECT marca, DATE'2003-07-02', 0, 8 FROM personal WHERE marca IN (102,107) ;

-- 3 iulie 2003
INSERT INTO pontaje (marca, data, orelucrate, oreco)
  SELECT marca, DATE'2003-07-03', 8, 0 FROM personal WHERE marca NOT IN (102,107)
  UNION SELECT marca, DATE'2003-07-03', 0, 8 FROM personal WHERE marca IN (102,107) ;

-- 4 iulie 2003
INSERT INTO pontaje (marca, data, orelucrate, oreco)
  SELECT marca, DATE'2003-07-04', 8, 0 FROM personal WHERE marca NOT IN (102,107)
  UNION SELECT marca, DATE'2003-07-04', 0, 8 FROM personal WHERE marca IN (102,107) ;
```

```

-- 7 iulie 2003
INSERT INTO pontaje
  SELECT marca, DATE'2003-07-07',
    CASE WHEN marca IN (102,107) THEN 0 ELSE 8 END AS OreLucru,
    CASE WHEN marca IN (102, 107) THEN 8 ELSE 0 END AS OreCO,
    CASE marca WHEN 103 THEN 4 ELSE 0 END AS OreNoapte,
    CASE marca WHEN 105 THEN 2 ELSE 0 END AS OreAbsente
  FROM personal ;

-- 8 iulie 2003
INSERT INTO pontaje
  SELECT marca, DATE'2003-07-08', CASE WHEN marca=103 THEN 0 ELSE 8 END
    AS OreLucru,
    CASE WHEN marca=103 THEN 8 ELSE 0 END AS OreCO,
    CASE marca WHEN 104 THEN 2 ELSE 0 END AS OreNoapte,
    CASE marca WHEN 105 THEN 1 ELSE 0 END AS OreAbsente
  FROM personal ;

/* pe luna august introducem aceleași date ca și pe iulie, însă cu 0 la absențe nemotivate și
orenoapte; în plus, se evită pontajele pentru sâmbete și duminici */
INSERT INTO pontaje
  SELECT marca, ADD_MONTHS(data,1), orelucrate, oreco, 0 AS orenoapte, 0 AS absnem
  FROM pontaje
  WHERE EXTRACT (MONTH FROM data) = 7 AND
    RTRIM(TO_CHAR(ADD_MONTHS(data,1),'DAY')) NOT IN ('SATURDAY', 'SUNDAY') ;

/* pe luna septembrie, preluăm datele din iulie, ca și pentru august; în plus,
nimeni nu mai e în concediu */
INSERT INTO (SELECT marca, data FROM pontaje)
  SELECT marca, ADD_MONTHS(data,2)
  FROM pontaje
  WHERE EXTRACT (MONTH FROM data) = 7 AND
    RTRIM(TO_CHAR(ADD_MONTHS(data,2),'DAY')) NOT IN ('SATURDAY', 'SUNDAY') ;

COMMIT ;

```

Din rațiuni de pură comoditate, atât cât de pură poate să fie comoditatea, pe lunile august preluăm înregistrările din iulie. Pe august, angajații 102 și 107 sunt tot în concediu, în timp ce pentru septembrie toată lumea e la lucru. Pentru ambele luni nu sunt nici ore lucrate în regim de noapte, și nici absențe nemotivate. În plus, dacă 2 iulie cade într-o miercuri, 2 august este o sâmbătă. Așa că, pentru dintre aceste două luni se vor elimina zilele care ar cădea în weekend (sâmbete și duminici).

Din “punct de vedere” SQL, în interogări apar destule noutăți: structuri CASE, funcțiile ADD\_MONTHS, RTRIM, TO\_CHAR, EXTRACT și alte bunătăți ce vor fi expuse pe parcursul capitolelor 6 și 7.

Analog pot fi populate tabelele SPORURI, RETINERI și SALARII pentru o lună dată. Astfel, corespunzător lunii iulie 2003 se lansează interogările:

```

INSERT INTO sporuri (marca, an, luna)
  SELECT marca, 2003, 7
  FROM personal ;

INSERT INTO retineri (marca, an, luna)

```



```
SELECT marca, 2003, 7
FROM personal ;

INSERT INTO salarii (marca, an, luna)
SELECT marca, 2003, 7
FROM personal ;
COMMIT ;
```

#### 5.1.4. Inserări în tabele multiple

Comanda `INSERT` are și un format de-a dreptul exotic, prin care, simultan, pot fi introduse linii în două sau mai multe tabele, pe baza unor condiții aplicate rezultatului unei fraze `SELECT`. Astfel, presupunem că se dorește arhivarea separată, pe luni calendaristice, a pontajelor. De aceea, periodic, din tabela `PONTAJE` se vor salva înregistrările corespunzătoare în tabele de genul `PONTAJE_2003_1`, `PONTAJE_2003_3` etc. Crearea acestor tabele nu ridică probleme. La începutul anului 2003 se lansează fabulosul script din listing 5.4.

Listing 5.4. Crearea tabelelor de arhivare

```
CREATE TABLE pontaje_2003_ianuarie AS SELECT * FROM pontaje WHERE 1=2 ;
CREATE TABLE pontaje_2003_februarie AS SELECT * FROM pontaje WHERE 1=2 ;
CREATE TABLE pontaje_2003_martie AS SELECT * FROM pontaje WHERE 1=2 ;
CREATE TABLE pontaje_2003_aprilie AS SELECT * FROM pontaje WHERE 1=2 ;
CREATE TABLE pontaje_2003_mai AS SELECT * FROM pontaje WHERE 1=2 ;
CREATE TABLE pontaje_2003_iunie AS SELECT * FROM pontaje WHERE 1=2 ;
CREATE TABLE pontaje_2003_iulie AS SELECT * FROM pontaje WHERE 1=2 ;
CREATE TABLE pontaje_2003_august AS SELECT * FROM pontaje WHERE 1=2 ;
CREATE TABLE pontaje_2003_septembrie AS SELECT * FROM pontaje WHERE 1=2 ;
CREATE TABLE pontaje_2003_octombrie AS SELECT * FROM pontaje WHERE 1=2 ;
CREATE TABLE pontaje_2003_noiembrie AS SELECT * FROM pontaje WHERE 1=2 ;
CREATE TABLE pontaje_2003_decembrie AS SELECT * FROM pontaje WHERE 1=2 ;
```

Presupunând că am ajuns în luna octombrie și ultima arhivarea a fost făcută pentru luna iunie, următoarea comandă `INSERT` "umple" dintr-o singură mișcare tabelele corespondente lunilor iulie, august și septembrie:

```
INSERT ALL
  WHEN EXTRACT (YEAR FROM data) = 2003 AND
    EXTRACT (MONTH FROM data) = 7 THEN
    INTO pontaje_2003_iulie
  WHEN EXTRACT (YEAR FROM data) = 2003 AND
    EXTRACT (MONTH FROM data) = 8 THEN
    INTO pontaje_2003_august
  WHEN EXTRACT (YEAR FROM data) = 2003 AND
    EXTRACT (MONTH FROM data) = 9 THEN
    INTO pontaje_2003_septembrie
SELECT * FROM pontaje
```

## 5.2. Ștergerea liniilor - DELETE

Operațiunea de ștergere este mult mai simplă și, statistic vorbind, este cea care se deprime cel mai ușor, uneori funcționând chiar mai bine decât intenționa cel care a lansat-o. În plus, ca și INSERT, DELETE este deja folosită de câteva capitole, așa că vom cheltui puțin timp și spațiu cu discutarea sa. Prima variantă șterge toate liniile din tabela PONTAJE\_2003\_SEPTEMBRIE în care marca este 101 și data 1 septembrie:

```
DELETE FROM pontaje_2003_septembrie
WHERE marca = 101 AND data = DATE'2003-09-01' ;
```

Pentru cazul nostru, o singură linie îndeplinește predicatul specificat în clauza WHERE. Pentru zăpăciți, cel mai neplăcut la comanda DELETE este că, atunci când se omite specificarea clauzei WHERE, se șterg toate liniile din tabelă. Astfel,

```
DELETE FROM pontaje_2003_septembrie ;
```

va șterge, neîndoindu-se, toate înregistrările tabelii PONTAJE\_2003\_SEPTEMBRIE. Noroc cu mecanismul tranzacțional și comanda ROLLBACK, altminteri, rata sinuciderilor în rândul informaticienilor era cu mult mai mare.

Trecând la lucruri un pic mai dificile, ne propunem ca, odată făcută salvarea pontajelor pe luna august 2003 (în tabela PONTAJE\_2003\_AUGUST), să ștergem liniile respective din tabela sursă – PONTAJE. Comanda de ștergere face apel la o subconsultare prin care se extrag angajații și zilele ce prezintă pontaje pentru această lună și care au fost salvate în tabela-arhivă:

```
DELETE FROM pontaje WHERE (marca, data) IN
(SELECT marca, data FROM pontaje_2003_august) ;
```

Pentru plăcerea de a SQL-ui (se pronunță *eschiuêlui*), folosim și o variantă exotică ce are cam același efect ca al ultimei comenzi:

```
DELETE FROM (SELECT * FROM pontaje ) po1
WHERE EXISTS
(SELECT 1
FROM pontaje_2003_iulie po2
WHERE po2.marca=po1.marca AND po2.data=po1.data) ;
```

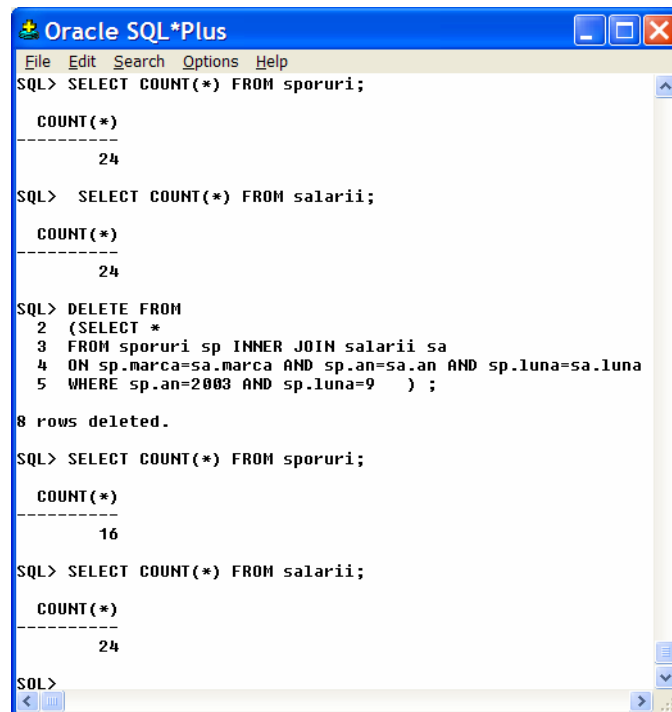
Două elemente rețin atenția în mod deosebit. Mai întâi, faptul că în locul numelui tabelii PONTAJE s-a folosit o frază SELECT. Al doilea element ține de recursul la o interogare corelată (pentru detalii, vezi capitolul 7).

Consultarea de după clauza FROM nu pare a fi de prea mare efect. Cu totul altfel stau, însă, lucrurile când aceasta va conține o joncțiune. Astfel, să

presupunem că, deoarece am făcut arhivarea pentru septembrie 2003 atât a pontajelor cât și a tabelor de sporuri, rețineri și salarii, dorim să ștergem liniile din SALARII și SPORURI corespunzătoare acestei luni. Să încercăm un DELETE mai special:

```
DELETE FROM
  (SELECT *
   FROM sporuri sp INNER JOIN salarii sa
    ON sp.marca=s.marca AND sp.an=s.an
    AND sp.luna=s.luna
   WHERE sp.an=2003 AND sp.luna=9 ) ;
```

Nu știm la ce v-ați așteptat, însă comanda de mai sus șterge înregistrări, dar numai din tabela SPORURI, după cum se observă în figura 5.8. Înainte de ștergere, ambele tabele jonctionate numărau 24 de linii, în timp ce după ștergere SPORURI are 16 linii, iar SALARII 24.



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> SELECT COUNT(*) FROM sporuri;

COUNT(*)
-----
        24

SQL> SELECT COUNT(*) FROM salarii;

COUNT(*)
-----
        24

SQL> DELETE FROM
  2  (SELECT *
  3  FROM sporuri sp INNER JOIN salarii sa
  4  ON sp.marca=s.marca AND sp.an=s.an AND sp.luna=s.luna
  5  WHERE sp.an=2003 AND sp.luna=9 ) ;

8 rows deleted.

SQL> SELECT COUNT(*) FROM sporuri;

COUNT(*)
-----
        16

SQL> SELECT COUNT(*) FROM salarii;

COUNT(*)
-----
        24

SQL>
```

Figura 5.8. Joncțiunea, ca argument al unei comenzi de ștergere – caz 1

În schimb, dacă modificăm ordinea tabelor la joncționare, SALARII este cea din care se va face ștergerea:

```
DELETE FROM
  (SELECT *
```

```

FROM salarii sa INNER JOIN sporuri sp
  ON sa.marca=sp.marca AND sa.an=sp.an
  AND sa.luna=sp.luna
WHERE sa.an=2003 AND sa.luna=9 ) ;

```

după cum reiese din figura 5.9.

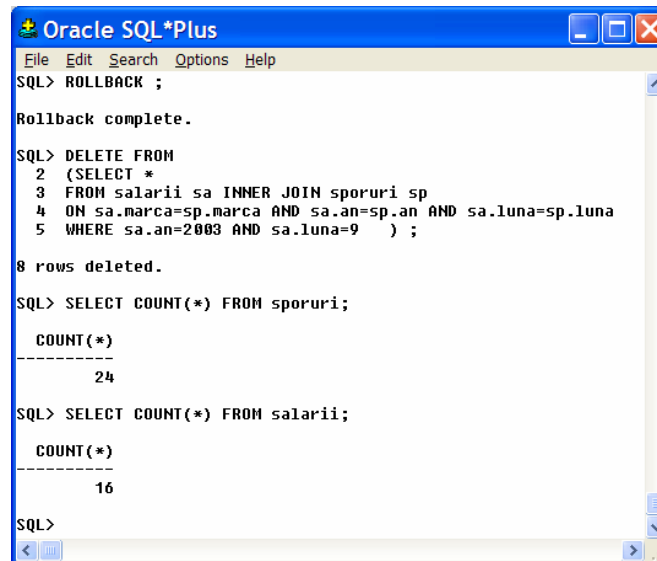


Figura 5.9. Joncțiunea ca argument al unei comenzi de ștergere – caz 2

### 5.3. Modificarea valorilor - UPDATE

Nici comanda `UPDATE` nu are un format supărător de complicat, chestiunile de finețe fiind legate de apelul la subconsultările corelate. Dacă, spre exemplu, conducerea firmei ia eroica decizie de a mări salariile orare tuturor angajaților cu 10%, incrementarea atributului `SalOrar` din tabela `PERSONAL` s-ar realiza prin comanda:

```
UPDATE personal SET salorar = salorar * 1.10 ;
```

Deoarece nu apare clauza `WHERE`, modificarea precizată în clauza `SET` va afecta toate liniile tabelului, inclusiv cele care se referă la colaboratori. Excluderea colaboratorilor de la această binevenită operațiune se realizează astfel:

```

UPDATE personal
SET salorar = salorar * 1.10
WHERE colaborator <> 'N' ;

```

Presupunem că procentul de creștere este diferențiat pe compartimente: 10% pentru producție, 8% pentru contabilitate și 12% pentru IT (dacă autorii acestei cărți erau contabili, fiți convinși că altele erau procentele !). Cea mai banală soluție este formată din trei comenzi UPDATE:

```
UPDATE personal
  SET salorar = salorar * 1.10 WHERE compart = 'PROD' ;
UPDATE personal
  SET salorar = salorar * 1.08 WHERE compart = 'CONTA' ;
UPDATE personal
  SET salorar = salorar * 1.12 WHERE compart = 'IT' ;
```

Nu putem, totuși, lua o mină de cunoscători SQL cu așa variantă primitivă. Astfel stau lucrurile dacă folosim o structură de tip CASE:

```
UPDATE personal
  SET salorar = salorar *
    CASE compart
      WHEN 'PROD' THEN 1.10
      WHEN 'CONTA' THEN 1.08
      WHEN 'IT' THEN 1.12
      ELSE 1
    END
```

Putem chiar exagera, folosind o interogare corelată:

```
UPDATE personal p1
  SET salorar = salorar *
    (SELECT CASE compart
      WHEN 'PROD' THEN 1.10
      WHEN 'CONTA' THEN 1.08
      WHEN 'IT' THEN 1.12
      ELSE 1 END
     FROM personal p2 WHERE p1.marca=p2.marca) ;
```

În general, însă, interogările corelate reprezintă prea mult pentru o asemenea problemă. Altfel stau lucrurile pentru cazurile următoare. Câmpurile OreLucrate și OreCO din SALARII centralizează orele lucrate și de concediu pentru un angajat pe o lună. Dacă dorim să actualizăm cele două attribute pentru iulie 2003, și presupunem că pentru această lună există câte o linie corespunzătoare fiecărui angajat, putem încerca o comandă de genul:

```
UPDATE salarii sa
  SET orelucrate = (
    SELECT SUM (orelucrate)
    FROM pontaje po
    WHERE EXTRACT (YEAR FROM data)=sa.an AND
          EXTRACT (MONTH FROM data) = sa.luna
```

```

        AND po.marca=sa.marca
    ),
    oreco = (
    SELECT SUM (oreco)
    FROM pontaje po
    WHERE EXTRACT (YEAR FROM data)=sa.an AND
          EXTRACT (MONTH FROM data) = sa.luna
          AND po.marca=sa.marca
    )
WHERE an = 2003 AND luna = 7 ;

```

Comanda poate fi simplificată, prin calcularea simultană a celor două atribute folosind o singură interogare.

```

UPDATE salarii sa
SET (orelucrate, oreco) = (
    SELECT SUM (orelucrate), SUM(oreco)
    FROM pontaje po
    WHERE EXTRACT (YEAR FROM data)=sa.an AND
          EXTRACT (MONTH FROM data) = sa.luna AND
          po.marca=sa.marca
    )
WHERE an = 2003 AND luna = 7 ;

```

Oricare ar fi varianta, rezultatul ar trebui să fie cel din figura 5.10.

---

```

SQL> UPDATE salarii sa
2  SET (orelucrate, oreco) = (
3  SELECT SUM (orelucrate), SUM(oreco)
4  FROM pontaje po
5  WHERE EXTRACT (YEAR FROM data)=sa.an AND
6  EXTRACT (MONTH FROM data) = sa.luna AND po.marca=sa.marca
7  )
8  WHERE an = 2003 AND luna = 7 ;

8 rows updated.

SQL> SELECT marca, an, luna, orelucrate, oreco FROM salarii WHERE luna=7
2 /

```

MARCA	AN	LUNA	ORELUCRATE	ORECO
101	2003	7	48	0
102	2003	7	16	32
103	2003	7	40	8
104	2003	7	48	0
105	2003	7	48	0
106	2003	7	48	0
107	2003	7	16	32
108	2003	7	48	0

```

8 rows selected.

SQL> |

```

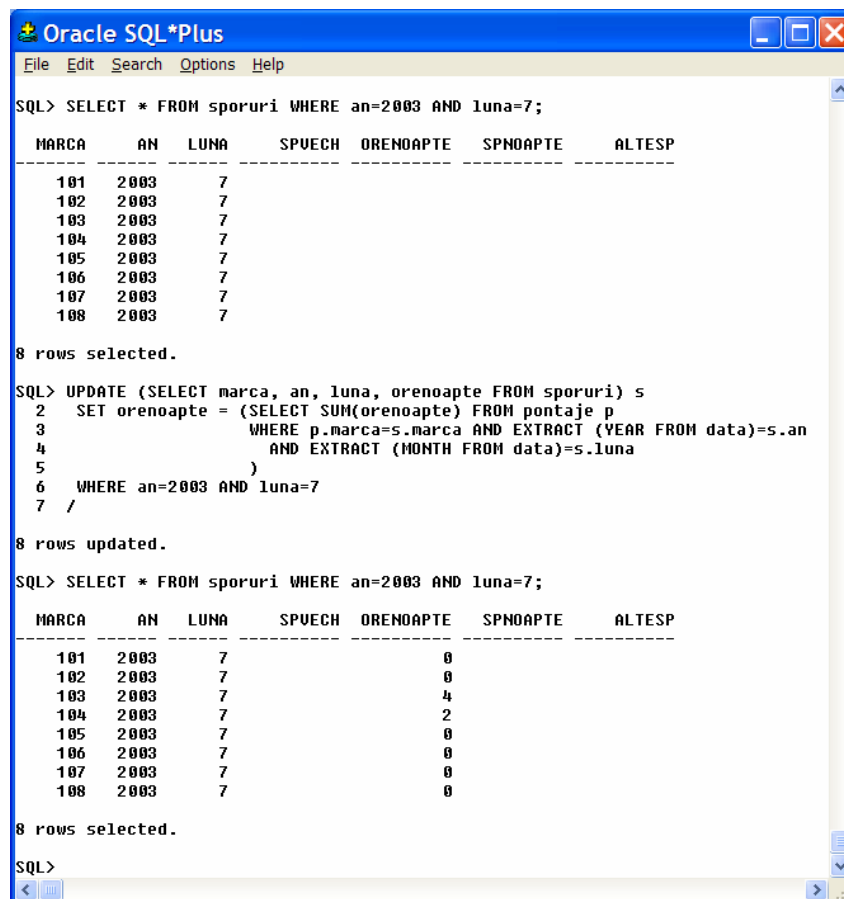
---

Figura 5.10. Actualizarea atributelor SALARII.OreLuCrate și SALARII.OreCo

Și în cazul comenzii UPDATE, în locul numelui de tabelă se poate specifica o subconsultare. Spre exemplu, dacă în tabela SPORURI se dorește centralizarea orelor de noapte pentru luna iulie 2003 se poate recurge (și) la soluția:

```
UPDATE (SELECT marca, an, luna, orenoapte FROM sporuri) s
SET orenoapte =
(SELECT SUM(orenoapte)
FROM pontaje p
WHERE p.marca=s.marca AND
      EXTRACT (YEAR FROM data)=s.an
      AND EXTRACT (MONTH FROM data)=s.luna
)
WHERE an=2003 AND luna=7
```

Valorile înregistrărilor din tabela SPORURI dinaintea și după comanda UPDATE sunt cele din figura 5.11.



```
SQL> SELECT * FROM sporuri WHERE an=2003 AND luna=7;
```

MARCA	AN	LUNA	SPVECH	ORENOAPTE	SPNOAPTE	ALTESP
101	2003	7				
102	2003	7				
103	2003	7				
104	2003	7				
105	2003	7				
106	2003	7				
107	2003	7				
108	2003	7				

```
8 rows selected.

SQL> UPDATE (SELECT marca, an, luna, orenoapte FROM sporuri) s
2 SET orenoapte = (SELECT SUM(orenoapte) FROM pontaje p
3 WHERE p.marca=s.marca AND EXTRACT (YEAR FROM data)=s.an
4 AND EXTRACT (MONTH FROM data)=s.luna
5 )
6 WHERE an=2003 AND luna=7
7 /

8 rows updated.

SQL> SELECT * FROM sporuri WHERE an=2003 AND luna=7;
```

MARCA	AN	LUNA	SPVECH	ORENOAPTE	SPNOAPTE	ALTESP
101	2003	7		0		
102	2003	7		0		
103	2003	7		4		
104	2003	7		2		
105	2003	7		0		
106	2003	7		0		
107	2003	7		0		
108	2003	7		0		

```
8 rows selected.

SQL>
```

Figura 5.11. "Actualizarea" unei subconsultări

Atunci când subconsultarea ce substituie numele unei tabele conține o joncțiune, Oracle devine mult mai reticent. Astfel, dacă am dori să calculăm, dintr-o singură mișcare, pentru luna iulie 2003, atât SPORURI.orenoapte, cât și SALARII.orelucrate și SALARII.oreco, am putea fi tentați să lansăm comanda:

```
UPDATE
  SELECT sporuri.marca, sporuri.an, sporuri.luna,
         orenoapte, orelucrate, oreco
  FROM sporuri INNER JOIN salarii ON
         sporuri.an=salarii.an AND
         sporuri.luna=salarii.luna
         AND sporuri.marca=salarii.marca) s
SET (orelucrate, oreco, orenoapte) =
  (SELECT SUM(orelucrate), SUM (oreco), SUM(orenoapte)
   FROM pontaje p WHERE p.marca=s.marca AND
    EXTRACT (YEAR FROM data)=s.an
    AND EXTRACT (MONTH FROM data)=s.luna
   )
WHERE an=2003 AND luna=7
```

Mesajul recepționat - vezi figura 5.12 - este unul clar: Orace nu are nimic împotriva dacă subconsultarea prezintă o joncțiune, însă de modificat nu se poate decât una dintre tabele.

```
SQL> UPDATE (SELECT sporuri.marca, sporuri.an, sporuri.luna,
2  orenoapte, orelucrate, oreco
3  FROM sporuri INNER JOIN salarii ON
4  sporuri.an=salarii.an AND sporuri.luna=salarii.luna AND sporuri.marca=salarii.marca) s
5  SET (orelucrate, oreco, orenoapte) = (SELECT SUM(orelucrate), SUM (oreco), SUM(orenoapte)
6  FROM pontaje p WHERE p.marca=s.marca AND EXTRACT (YEAR FROM data)=s.an
7  AND EXTRACT (MONTH FROM data)=s.luna
8  )
9  WHERE an=2003 AND luna=7
10 /
SET (orelucrate, oreco, orenoapte) = (SELECT SUM(orelucrate), SUM (oreco), SUM(orenoapte)
*
```

```
ERROR at line 5:
ORA-01776: cannot modify more than one base table through a join view
```

```
SQL>
```

Figura 5.12. Tentativă (nereușită) de actualizare simultană a două tabele

Probabil că până acum nici una dintre comenzi nu a creat frisoane, așa că încercăm altceva: actualizarea tabelei SPORURI pentru luna iulie 2003 pe baza datelor din PONTAJE, PERSONAL și TRANSE\_SV, ultima pentru calculul sporului de vechime. În prealabil, populăm tabela TRANSE\_SV – listing 5.5.

Listing 5.5. Înregistrările tabelei TRANSE\_SV

```
INSERT INTO transe_sv VALUES (0,3,0);
INSERT INTO transe_sv VALUES (3,6,5);
INSERT INTO transe_sv VALUES (6,10,10);
INSERT INTO transe_sv VALUES (10,15,14);
```



```

INSERT INTO transe_sv VALUES (15,20,18);
INSERT INTO transe_sv VALUES (20,99,25);

COMMIT ;

```

Liniile acestea vor să spună că, pentru angajații cu un număr de ani de vechime cuprins între 0 și 3, procentul sporului de vechime este zero (prima înregistrare), cei cu numărul între 3 și 6 beneficiază de 5% spor de vechime s.a.m.d. Acest procent este aplicat venitului de bază pe luna respectivă, calculat ca sumă între venit din lucrul efectiv ( $OreLucrate * SalOrar$ ) și venitul din concedii de odihnă ( $OreCO * SalOrarCO$ ).

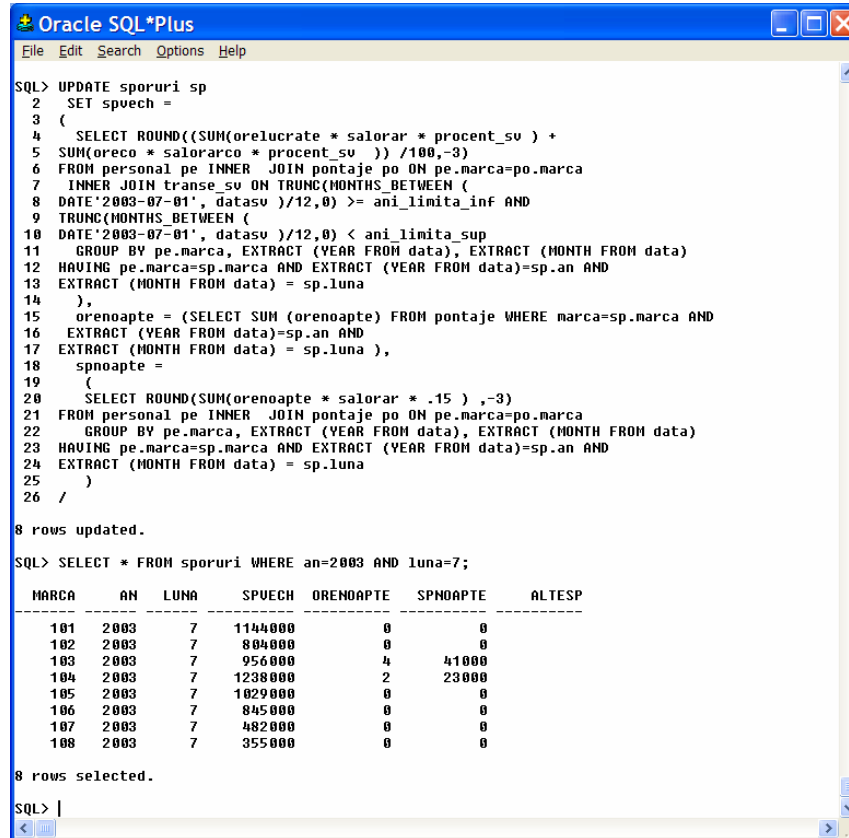
Ținând seama că sporul pentru orele de noapte este de 15%, actualizăm liniile tabelii SPORURI astfel:

```

UPDATE sporuri sp
SET spvech =
(
SELECT ROUND((SUM(orelucrate * salorar * procent_sv ) +
SUM(oreco * salorarco * procent_sv )) /100,-3)
FROM personal pe
INNER JOIN pontaje po ON pe.marca=po.marca
INNER JOIN transe_sv ON TRUNC(MONTHS_BETWEEN (
DATE'2003-07-01', datasv )/12,0) >=
ani_limita_inf AND TRUNC(MONTHS_BETWEEN (
DATE'2003-07-01', datasv )/12,0) <
ani_limita_sup
GROUP BY pe.marca, EXTRACT (YEAR FROM data),
EXTRACT (MONTH FROM data)
HAVING pe.marca=sp.marca AND
EXTRACT (YEAR FROM data)=sp.an AND
EXTRACT (MONTH FROM data) = sp.luna
),
orenoapte =
(SELECT SUM (orenoapte) FROM pontaje
WHERE marca=sp.marca AND
EXTRACT (YEAR FROM data)=sp.an AND
EXTRACT (MONTH FROM data) = sp.luna ),
spnoapte =
(
SELECT ROUND(SUM(orenoapte * salorar * .15 ) ,-3)
FROM personal pe INNER JOIN pontaje po
ON pe.marca=po.marca
GROUP BY pe.marca, EXTRACT (YEAR FROM data),
EXTRACT (MONTH FROM data)
HAVING pe.marca=sp.marca AND
EXTRACT (YEAR FROM data)=sp.an AND
EXTRACT (MONTH FROM data) = sp.luna
)
)

```

Execuția și rezultatul SQL\*Plus pentru această comandă sunt cele din figura 5.13.



```

Oracle SQL*Plus
File Edit Search Options Help

SQL> UPDATE sporuri sp
2 SET spvech =
3 (
4 SELECT ROUND((SUM(orelucrate * salorar * procent_sv ) +
5 SUM(oreco * salorarco * procent_sv )) /100,-3)
6 FROM personal pe INNER JOIN pontaje po ON pe.marca=po.marca
7 INNER JOIN transe_sv ON TRUNC(MONTHS_BETWEEN (
8 DATE'2003-07-01', datasv )/12,0) >= ani_limita_inf AND
9 TRUNC(MONTHS_BETWEEN (
10 DATE'2003-07-01', datasv )/12,0) < ani_limita_sup
11 GROUP BY pe.marca, EXTRACT (YEAR FROM data), EXTRACT (MONTH FROM data)
12 HAVING pe.marca=sp.marca AND EXTRACT (YEAR FROM data)=sp.an AND
13 EXTRACT (MONTH FROM data) = sp.luna
14 ),
15 orenoapte = (SELECT SUM (orenoapte) FROM pontaje WHERE marca=sp.marca AND
16 EXTRACT (YEAR FROM data)=sp.an AND
17 EXTRACT (MONTH FROM data) = sp.luna ),
18 spnoapte =
19 (
20 SELECT ROUND(SUM(orenoapte * salorar * .15 ) , -3)
21 FROM personal pe INNER JOIN pontaje po ON pe.marca=po.marca
22 GROUP BY pe.marca, EXTRACT (YEAR FROM data), EXTRACT (MONTH FROM data)
23 HAVING pe.marca=sp.marca AND EXTRACT (YEAR FROM data)=sp.an AND
24 EXTRACT (MONTH FROM data) = sp.luna
25 )
26 /

8 rows updated.

SQL> SELECT * FROM sporuri WHERE an=2003 AND luna=7;

MARCA AN LUNA SPVECH ORENOAPTE SPNOAPTE ALTESP
-----
101 2003 7 1144000 0 0
102 2003 7 804000 0 0
103 2003 7 956000 4 41000
104 2003 7 1238000 2 23000
105 2003 7 1029000 0 0
106 2003 7 845000 0 0
107 2003 7 482000 0 0
108 2003 7 355000 0 0

8 rows selected.

SQL> |

```

Figura 5.13. Un UPDATE ceva mai pretențios

## 5.4. O comandă recentă: MERGE

Nu, nu e nici o trimitere la “Omul recent” al lui H.R. Patapievici (mai ales că această comandă nu are nimic postmodern în ea) și nici la puseurile belicoase ale lui I.B. Lefter vis-a-vis de cartea cu pricina. Pur și simplu, cei de la Oracle s-au gândit (n-avem idee dacă au fost primii sau nu) la o comandă care să fie o combinată INSERT-UPDATE.

Comenzile UPDATE prin care calculăm orele lucrate, de concediu sau de noapte funcționează corect numai dacă în prealabil s-a făcut inserarea de înregistrări corespunzătoare în SALARII și SPORURI. Pe de altă parte, comanda INSERT este mai nimerită după scenariul “totul sau nimic”, o recalculare a atributelor centralizatoare reclamând un DELETE prealabil.

Să luăm un exemplu. Scriptul din listing 5.3 conține secvența de populare a tabelului PONTAJE pentru lunile iulie, august și septembrie. Luna de referință este iulie 2003. În paragraful anterior am calculat totalul orelor lucrate și de concediu în iulie pentru fiecare angajat. După această operațiune, brusc ne dăm seama că mai trebuie introduse pontajele pentru 10 și 11 iulie. Mai mult, începând cu data de 10 iulie avem și doi “boboci” la firmă, trecuți pe nedrept cu vederea.

Listing 5.6. Doi angajați și alte trei zile de pontaje

```

INSERT INTO personal VALUES (seq_marca.NextVal, 'Primul Angajat Nou', 'IT',
    DATE'2003-07-10', 55500, 0, 'N' );
INSERT INTO personal VALUES (seq_marca.NextVal, 'Al Doilea Angajat Nou', 'PROD',
    DATE'2003-07-10', 50500, 0, 'N' );

-- 9 iulie 2003
INSERT INTO pontaje
SELECT marca, DATE'2003-07-09', CASE WHEN marca=103 THEN 0 ELSE 8 END AS OreLucru,
    CASE WHEN marca=103 THEN 8 ELSE 0 END AS OreCO,
    CASE marca WHEN 104 THEN 3 ELSE 0 END AS OreNoapte,
    CASE marca WHEN 105 THEN 1 ELSE 0 END AS OreAbsente
FROM personal ;

-- 10 iulie 2003
INSERT INTO pontaje
SELECT marca, DATE'2003-07-10', CASE WHEN marca=103 THEN 0 ELSE 8 END AS OreLucru,
    CASE WHEN marca=103 THEN 8 ELSE 0 END AS OreCO,
    CASE marca WHEN 110 THEN 2 ELSE 0 END AS OreNoapte,
    0
FROM personal ;

-- 11 iulie 2003
INSERT INTO pontaje
SELECT marca, DATE'2003-07-11', CASE WHEN marca=103 THEN 0 ELSE 8 END AS OreLucru,
    CASE WHEN marca=103 THEN 8 ELSE 0 END AS OreCO,
    CASE marca WHEN 110 THEN 3 ELSE 0 END AS OreNoapte,
    0
FROM personal ;
COMMIT ;

```

Aceste noi înregistrări trebuie cumulate datelor deja existente, ceea ce presupune, pe de o parte, incrementarea orelor lucrate și de concediu cu orele celor trei zile de pontaje pentru “vechii” angajați, în timp ce, pentru noii veniți, trebuie inserată câte o înregistrare în SALARII. Operațiunea, complicată în aparență, poate fi tranșată printr-o comandă MERGE:

```

MERGE INTO salarii SA USING
(SELECT EXTRACT (YEAR FROM data) AS an,
    EXTRACT (MONTH FROM data) AS luna,
    marca, SUM(orelucrate) AS orelucrate,
    SUM(oreco) AS oreco
FROM pontaje
WHERE EXTRACT (YEAR FROM data) = 2003 AND
    EXTRACT (MONTH FROM data) = 7
    AND EXTRACT (DAY FROM data) >= 9

```

```
GROUP BY EXTRACT (YEAR FROM data),  
EXTRACT (MONTH FROM data), marca  
) PO  
ON  
    (SA.an = PO.an AND SA.luna=PO.luna AND  
    SA.marca = PO.marca )  
WHEN MATCHED THEN  
    UPDATE SET SA.orelucrate = SA.orelucrate +  
        PO.orelucrate,  
        SA.oreco = SA.oreco + PO.oreco  
WHEN NOT MATCHED THEN  
    INSERT (SA.an, SA.luna, SA.marca, SA.orelucrate,  
        SA.oreco) VALUES (PO.an, PO.luna, PO.marca,  
        PO.orelucrate, PO.oreco)
```

Tabela specificată după clauza INTO este cea în care vor fi efectuate actualizările - în cazul nostru SALARII (sinonim local SA). După clauza USING este plasată fraza SELECT care centralizează orele lucrate și de concediu pentru fiecare angajat.

```

Oracle SQL*Plus
File Edit Search Options Help
SQL> SELECT marca, orelucrate, oreco FROM salarii WHERE an=2003 AND luna=7 ;

  MARCA ORELUCRATE      ORECO
-----
    101         48         0
    102         16        32
    103         40         8
    104         48         0
    105         48         0
    106         48         0
    107         16        32
    108         48         0

8 rows selected.

SQL> MERGE INTO salarii SA USING
 2 (SELECT EXTRACT (YEAR FROM data) AS an, EXTRACT (MONTH FROM data) AS luna,
 3 marca, SUM(orelucrate) AS orelucrate, SUM(oreco) AS oreco
 4 FROM pontaje
 5 WHERE EXTRACT (YEAR FROM data) = 2003 AND EXTRACT (MONTH FROM data) = 7
 6 AND EXTRACT (DAY FROM data) >= 9
 7 GROUP BY EXTRACT (YEAR FROM data), EXTRACT (MONTH FROM data), marca
 8 ) PO
 9 ON (SA.an = PO.an AND SA.luna=PO.luna AND SA.marca = PO.marca )
10 WHEN MATCHED THEN
11 UPDATE SET SA.orelucrate = SA.orelucrate + PO.orelucrate, SA.oreco = SA.oreco + PO.oreco
12 WHEN NOT MATCHED THEN
13 INSERT (SA.an, SA.luna, SA.marca, SA.orelucrate, SA.oreco)
14 VALUES (PO.an, PO.luna, PO.marca, PO.orelucrate, PO.oreco)
15 /

10 rows merged.

SQL> SELECT marca, orelucrate, oreco FROM salarii WHERE an=2003 AND luna=7 ;

  MARCA ORELUCRATE      ORECO
-----
    109         24         0
    110         24         0
    101         72         0
    102         40        32
    103         40        32
    104         72         0
    105         72         0
    106         72         0
    107         40        32
    108         72         0

10 rows selected.

```

Figura 5.14. Funcționarea comenzii MERGE

Atenție ! Se iau în considerare numai noile pontaje, de după 9 iulie, pentru că celelalte sunt deja centralizate. Prin urmare, rezultatul subconsultării, care va fi referit drept PO, conține câte o linie pentru fiecare angajat ce are măcar o zi lucrată sau de concediu în perioada 9-31 iulie 2003. Pentru raportare la SALARII (SA), în PO au fost incluse și coloanele An și Luna.

Comanda UPDATE de pe ramura WHEN MATCHED se va executa numai pentru angajații care au deja, la momentul curent, o înregistrare în SALARII pentru anul și luna de referință. Practic, se vor incrementa cele două câmpuri (OreLucrate și OreCO) cu valorile calculate de subconsultare (PO). Liniile subconsultării ce nu se găsesc în SALARII (în sensul că nu există înregistrare corespondentă pentru care valorile atributelor An, Luna și Marca să fie identice) vor fi adăugare tabeli prin intermediul comenzii INSERT de pe ramura WHEN NOT MATCHED.

Conținutul tabelii SALARII, înainte și după lansarea comenzii MERGE constituie subiectul figuri 5.14.

## 5.5. Tranzacții

Unul din lucrurile considerate „agasante” de către primii veniți în lumea bazelor de date Oracle este faptul că după o sesiune „fructuoasă” cu SQL\*PLUS (sau alt client cu o „față” mai prietenoasă), sesiune încheiată prin „super”-comanda EXIT, aceștia constată că la reconectare toate datele „atent” introduse sau modificate anterior nu se mai regăsesc ... sunt dispărute și din păcate definitiv pierdute. Să fie un server de baze de date de calibrul Oracle atât de „neatent” cu datele noastre ???

Și totuși, câteva pagini mai înainte întâlnim o frază tulburătoare „ ... ultima comandă, COMMIT, are drept scop finalizarea *tranzacției* și, în consecință, *scrierea „definitivă” a liniilor în tabelă*”. Pentru a elimina, în cele din urmă, frustrările mai sus menționate, a venit vremea să dezvăluim „tâlcul” acestor cuvinte, și mai cu seamă a termenului „tranzacție”.

Pe scurt, o *tranzacție* reprezintă modalitatea prin care se pot împacheta într-o singură unitate de lucru o secvență de operații privind datele dintr-o bază de date. Această unitate se remarcă în principal prin *atomicitate*, sau, cu alte cuvinte, fie vor fi păstrate permanent în baza de date toate modificările operate (cazul COMMIT), fie se va renunța la toate acestea (cazul ROLLBACK).

Unitatea de lucru care se suprapune peste o tranzacție este determinată (și delimitată mai ales) de prelucrările din sistemele de aplicații. Astfel, o tranzacție poate însemna introducerea unei facturi, ceea ce se materializează de regulă prin introducerea liniei corespunzătoare într-o tabelă FACTURI urmată de introducerea liniilor corespunzătoare produselor facturate într-o tabelă DETALII\_FACTURI și apoi, în fine, calculul valorii totale a facturii în câmpul corespunzător. Toată această secvență de operații are sens numai dacă se derulează în totalitate; este nedorit cazul care s-ar finaliza numai cu linia din FACTURI (eventual și cu totalul calculat) și fără liniile din DETALII\_FACTURII, ca să nu mai vorbim de cazul invers în care am încălca și eventuala restricție referențială. De asemenea, dacă la un moment dat ne-am hotărî să renunțăm la detaliile unei facturi, ar fi foarte normal să renunțăm și la „antetul” facturii respective.

### 5.5.1. COMMIT și ROLLBACK

Revenind la cazul nostru concret (despre pontaje și salarii), o secvență *tranzacțională* de operațiuni ar putea fi următoarea: consemnăm orele lucrate de către un angajat în tabela PONTAJE, apoi calculăm sau recalculăm sporurile (tabela SPORURI) și în cele din urmă actualizăm și tabela SALARII.

Înainte să trecem la „lucru”, trebuie neîntârziat menționat că o tranzacție va începe fie imediat după conectarea la serverul de baze de date (de exemplu printr-o sesiune SQL\*PLUS), fie după o comandă care a încheiat tranzacția precedentă (adică ROLLBACK sau COMMIT).

Prin urmare, presupunem mai întâi următoarea situație în baza de date (vezi figura 5.15).

```
SQL> ROLLBACK;

Rollback complete.

SQL>
SQL> SELECT * FROM personal;

  MARCA NUMEPREN                                COMPA DATASU    SALORAR  SALORARCO C
-----
  2001 Angajat 21                                CONTA 15-JUL-91    33000    28000 N
  2002 Angajat 22                                CONTA 17-MAY-93    43000    30000 N
  2003 Angajat 23                                FIN    03-MAY-99    89000    85000 N
  2004 Angajat 24                                FIN    21-FEB-85    70000    65000 N
  2005 Angajat 25                                MARK   17-MAY-93    45000    40000 N

SQL> SELECT * FROM pontaje ORDER BY marca;

  MARCA DATA    ORELUCRATE    ORECO    ORENOAPTE    OREABSNEM
-----
  2001 07-FEB-03      6          0          0          0
  2001 10-FEB-03     10          0          2          0
  2001 11-FEB-03      4          0          4          0
  2002 07-FEB-03      6          0          0          0
  2002 10-FEB-03     10          0          2          0
  2003 07-FEB-03      6          0          0          0
  2004 07-FEB-03      6          0          0          0
  2004 11-FEB-03      4          0          4          0
  2005 07-FEB-03      6          0          0          0

9 rows selected.

SQL> SELECT * FROM sporuri;

  MARCA      AN      LUNA    SPVECH    ORENOAPTE    SPNOAPTE    ALTESP
-----
  2001      2003        2
```

```
SQL> SELECT * FROM salarii;

no rows selected
```

Figura 5.15 Situația inițială în tabelele ce urmează „tranzacționate”

Tranzacția noastră se va derula în felul următor:

*Pasul 1* - Inserăm un nouă linie în PONTAJE, linie care consemnează orele de noapte lucrate de către salariatul 2001 în data de 12/02/2003.

```
INSERT INTO pontaje(marca, data, orelucrate, orenoapte)
VALUES (2001, TO_DATE('12/02/2003', 'dd/mm/yyyy'), 8, 4);
```

*Pasul 2* - Actualizăm linia din tabela SPORURI corespunzătoare salariatului mai sus menționat pentru luna februarie. Ne interesează (deocamdată) sporul de noapte și totalul orelor noapte:

```
UPDATE sporuri SET spnoapte = .15 *
(SELECT salorar FROM personal WHERE marca = 2001) *
```

```

        (SELECT SUM(orenoapte) FROM pontaje
        WHERE marca = 2001 AND TO_CHAR(data, 'mm') = 2
        AND TO_CHAR(data, 'yyyy') = 2003),
orenoapte =
        (SELECT SUM(orenoapte) FROM pontaje
        WHERE marca = 2001 AND TO_CHAR(data, 'mm') = 2
        AND TO_CHAR(data, 'yyyy') = 2003)
WHERE marca = 2001 AND luna = 2 AND an = 2003;

```

*Pasul 3* – Ardem de nerăbdare să aducem și la „cunoștința” liniei corespunzătoare (marca 2001, luna februarie din 2003) din tabela SALARII modificările intervenite în tabela PONTAJE. Însă, după cum se observă din figura 5.15, această înregistrare nu există. Ca urmare, mai întâi o creăm:

```

INSERT INTO salarii (marca, an, luna)
VALUES (2001, 2003, 2);

```

apoi o vom actualiza:

```

UPDATE salarii
SET orelucrate =
        (SELECT SUM(orelucrate) FROM pontaje
        WHERE marca = 2001 AND TO_CHAR(data, 'yyyy') = 2003
        AND TO_CHAR(data, 'mm') = 2),
oreco = (SELECT SUM(oreco) FROM pontaje
        WHERE marca = 2001 AND TO_CHAR(data, 'yyyy') = 2003
        AND TO_CHAR(data, 'mm') = 2)
WHERE marca = 2001 AND an = 2003 AND luna = 2;

```

(ca să pară o tranzacție mai lungă, am recurs la două comenzi SQL, deși operația anterioară se putea face foarte bine printr-una singură)

În fine, putem verifica situația la acest moment (vezi figura 5.16). Dacă starea bazei de date se va păstra așa cum este prezentată în figura 5.16 depinde de modul în care vom încheia tranzacția:

- prin COMMIT, astfel încât după mesajul „Commit complete” modificările vor deveni permanente în baza de date și, de asemenea, vizibile și către ceilalți utilizatori;
- prin ROLLBACK, și atunci vom reveni la starea din figura 5.15



```
SQL> SELECT * FROM pontaje ORDER BY marca;
```

MARCA	DATA	ORELUCRATE	ORECO	ORENOAPTE	OREABSNEH
2001	07-FEB-03	6	0	0	0
2001	10-FEB-03	10	0	2	0
2001	11-FEB-03	4	0	4	0
2001	12-FEB-03	8	0	4	0
2002	07-FEB-03	6	0	0	0
2002	10-FEB-03	10	0	2	0
2003	07-FEB-03	6	0	0	0
2004	07-FEB-03	6	0	0	0
2004	11-FEB-03	4	0	4	0
2005	07-FEB-03	6	0	0	0

10 rows selected.

```
SQL> SELECT * FROM sporuri;
```

MARCA	AN	LUNA	SPVECH	ORENOAPTE	SPNOAPTE	ALTESP
2001	2003	2		10	49500	

```
SQL> SELECT * FROM salarii;
```

MARCA	AN	LUNA	ORELUCRATE	ORECO	VENITBAZA	SPORURI	IMPOZIT	RETINERI
2001	2003	2	28	0				

Figura 5.16 Situația în tabelele bazei de date după derularea operațiilor tranzacționale

### 5.5.2. Puncte de salvare: SAVEPOINTS

Ce s-ar întâmpla însă dacă după ce am actualizat tabela SPORURI realizăm că am greșit (spre exemplu, am actualizat înregistrarea altui salariat). După cum s-a derulat tranzacția în momentul anterior am avea două alternative: fie vom mai executa un UPDATE, sau mai multe, de corecție (dacă știm exact ce avem de corectat și cunoaștem în detaliu consecințele acestei operațiuni, consecințe pe care ni le asumăm), fie recurgem la ROLLBACK și începem tranzacția de la capăt. Dacă de la începutul tranzacției nu s-au derulat prea multe operațiuni, atunci deranjul n-ar fi prea mare, dar dacă am executat măcar vreo 10 comenzi, atunci situația se complică.

Din fericire există posibilitatea segmentării unei tranzacții principale în sub-tranzacții mai mici care să poată fi anulate individual în ordinea începerii lor (nu există și posibilitatea comiterii lor individuale: primul COMMIT-ul va încheia întreaga tranzacție, adică suma segmentelor tranzacționale începute până în acel moment). Comanda care ne scoate astfel la „liman” este SAVEPOINT.

Ca exemplu, să reluăm secvența de operații anterioare din starea inițială, descrisă în figura 5.17.

*Pasul 1* – Inserăm un nouă linie în PONTAJE

```
INSERT INTO pontaje(marca, data, orelucrate, orenoapte)
VALUES (2001, TO_DATE('12/02/2003', 'dd/mm/yyyy'), 8, 4);
```

*Pasul 2* – Prevăzători (anticipând „fatalitatea” pasului următor) declarăm un savepoint:

```
SAVEPOINT act_pontaje;
```

```
SQL> SELECT * FROM pontaje ORDER BY marca;
```

MARCA	DATA	ORELUCRATE	ORECO	ORENOAPTE	OREABSNEH
2001	07-FEB-03	6	0	0	0
2001	10-FEB-03	10	0	2	0
2001	11-FEB-03	4	0	4	0
2002	07-FEB-03	6	0	0	0
2002	10-FEB-03	10	0	2	0
2003	07-FEB-03	6	0	0	0
2004	07-FEB-03	6	0	0	0
2004	11-FEB-03	4	0	4	0
2005	07-FEB-03	6	0	0	0

9 rows selected.

```
SQL> SELECT * FROM sporuri;
```

MARCA	AN	LUNA	SPUECH	ORENOAPTE	SPNOAPTE	ALTESP
2001	2003	2				
2002	2003	2				

```
SQL> SELECT * FROM salarii;
```

no rows selected

Figura 5.17 Situația inițială în bazei de date

*Pasul 3* – Actualizăm linia din tabela SPORURI însă, din greșeală (neprovocată), luăm în considerare salariatul 2002

```
UPDATE sporuri SET spnoapte = .15 *
  (SELECT salorar FROM personal WHERE marca = 2001) *
  (SELECT SUM(orenoapte) FROM pontaje
   WHERE marca = 2001 AND TO_CHAR(data, 'mm') = 2
   AND TO_CHAR(data, 'yyyy') = 2003),
  orenoapte = (SELECT SUM(orenoapte) FROM pontaje
   WHERE marca = 2001 AND TO_CHAR(data, 'mm') = 2
   AND TO_CHAR(data, 'yyyy') = 2003)
WHERE marca = 2002 AND luna = 2 AND an = 2003;
```

Inconsistența este dată de faptul că linia salariatului cu marca 2002 din tabela SPORURI este actualizată cu datele salariatului cu marca 2001.

*Pasul 4* – Persistăm în eroare și actualizăm și tabela SALARII, însă fără a reflecta din nefericire modificările intervenite în tabela PONTAJE. Însă, după cum se observă din figura 5.15, această înregistrare nu există. Ca urmare, mai întâi o creăm:

```
INSERT INTO salarii (marca, an, luna)
VALUES (2002, 2003, 2);
```

apoi o vom actualiza:

```
UPDATE salarii SET orelucrate =
  (SELECT SUM(orelucrate) FROM pontaje
   WHERE marca = 2001 AND TO_CHAR(data, 'yyyy') =
     2003 AND TO_CHAR(data, 'mm') = 2),
  oreco = (SELECT SUM(oreco) FROM pontaje
```

```

WHERE marca = 2001 AND TO_CHAR(data,
'yyyy') = 2003 AND TO_CHAR(data, 'mm') = 2)
WHERE marca = 2002 AND an = 2003 AND luna = 2;

```

În fine, verificăm rezultatul tranzacției noastre și constatăm neconcordanțele față de rezultatul scontat (vezi figura 5.18).

```
SQL> SELECT * FROM sporuri;
```

MARCA	AN	LUNA	SPUECH	ORENOAPTE	SPNOAPTE	ALTESP
2001	2003	2				
2002	2003	2		10	49500	

```
SQL> SELECT * FROM salarii;
```

MARCA	AN	LUNA	ORELUCRATE	ORECO	VENITBAZA	SPORURI	IMPOZIT	RETINERI
2002	2003	2	28	0				

Figura 5.18. Relevarea inconsistențelor introduse

Dacă am fost prevăzători și am trecut prin pasul 2, atunci putem lansa comanda  
 ROLLBACK TO SAVEPOINT act\_pontaje;

și vom anula efectul operațiilor din pașii 3-4, păstrând ceea ce am muncit în pasul 1 (vezi figura 5.19). După care putem relua, cu atenție, șirul modificărilor în tabelele SPORURI și SALARII.

```
SQL> ROLLBACK TO SAVEPOINT act_pontaje;
```

Rollback complete.

```
SQL> SELECT * FROM pontaje ORDER BY marca;
```

MARCA	DATA	ORELUCRATE	ORECO	ORENOAPTE	OREABSNE
2001	07-FEB-03	6	0	0	0
2001	10-FEB-03	10	0	2	0
2001	11-FEB-03	4	0	4	0
2001	12-FEB-03	8	0	4	0
2002	07-FEB-03	6	0	0	0
2002	10-FEB-03	10	0	2	0
2003	07-FEB-03	6	0	0	0
2004	07-FEB-03	6	0	0	0
2004	11-FEB-03	4	0	4	0
2005	07-FEB-03	6	0	0	0

10 rows selected.

```
SQL> SELECT * FROM sporuri;
```

MARCA	AN	LUNA	SPUECH	ORENOAPTE	SPNOAPTE	ALTESP
2001	2003	2				
2002	2003	2				

```
SQL> SELECT * FROM salarii;
```

no rows selected

Figura 5.19 Efectul anulării unei sub-tranzacții marcate printr-un SAVEPOINT

