Capitolul 12. Tabele virtuale

În aproape orice aplicație profesională cu baze de date există diferențieri între modul de stocare a datelor în bază şi modul de prezentare a acestora utilizatorilor. Pe de o parte, se încearcă furnizarea unei imagini unitare asupra datelor din punctul de vedere al funcțiunilor specifice sistemului informațional, iar, pe de altă parte, gestionarea persistenței se realizează în conformitatea cu opțiunile puse la dispoziție de instrumentele software folosite.

Urmare a acestei delimitări între stratul de prezentare (şi, implicit, acces) şi cel de stocare a apărut necesitatea folosirii tabelelor virtuale (view-uri) al căror scop primordial este de a oferi posibilitatea obținerii unui subset actualizabil de date din una sau mai multe tabele.

12.1. Elemente generale. Sintaxa de bază

Sintaxa comenzii pentru crearea unei tabele virtuale în Oracle este următoarea:

```
CREATE [ OR REPLACE ] VIEW  nume AS ( fraza Select SQL)
   [WITH CHECK OPTION]
```

Spre deosebire de tabelele clasice, pentru view-uri în dicționarul de date se păstrează fraza SELECT ca atare şi nu atributele definitorii ale relației rezultat. Setul de înregistrări corespunzător va fi generat ad-hoc pe baza datelor existente în tabele-sursă la momentul interogării tabelei virtuale. De obicei, scopul acestor structuri SQL este acela de a prezenta o *imagine (fereastră) actualizabilă* a datelor la un moment dat în timp.

Spre exemplu, având în vedere schema bazei de date utilizată pe parcursul acestei lucrări (vezi capitolul 4), se poate crea o tabelă virtuală cu scopul de a prezenta datele angajaților ce aparțin compartimentului *Producție* astfel:

```
CREATE VIEW pers_prod_view AS
          (SELECT * FROM Personal WHERE compart='PROD');
```

Pentru a obține informații despre definiția tabelelor virtuale interogăm tabela USER_VIEWS a dicționarului de date.

După cum specificam câteva rânduri mai sus, seturile de date obținute sunt actualizabile, orice modificare fiind trimisă automat în tabela sursă. În aces sens, figura 12.1 exemplifică cele trei operații DML (INSERT, UPDATE, DELETE) efectuate asupra datelor tabelei PERSONAL prin intermediul tabelei virtuale PERS_PROD_VIEW.

```
SQL> create view pers_prod_view as (select * from personal where compart='PROD');

View created.

SQL> select * from pers_prod_view;

     MARCA NUMEPREN                                  COMPA DATASV       SALORAR  SALORARCO C
---------- ---------------------------------------- ----- --------- ---------- ---------- -
     21111 Mandache Gigi                            PROD  15-AUG-82      21052      20000 N
     51111 Vasilescu Claudia                        PROD  15-DEC-77      29239      30000 D
     61111 Isaiescu Gina                            PROD  01-NOV-98      35087      34000 N

SQL> update pers_prod_view set salorar=30000 where marca=21111;

1 row updated.

SQL> insert into pers_prod_view values (11121,'Dumitrescu Fane','PROD',sysdate,56818,50000 ,

1 row created.

SQL> select * from pers_prod_view;

     MARCA NUMEPREN                                  COMPA DATASV       SALORAR  SALORARCO C
---------- ---------------------------------------- ----- --------- ---------- ---------- -
     21111 Mandache Gigi                            PROD  15-AUG-82      30000      20000 N
     51111 Vasilescu Claudia                        PROD  15-DEC-77      29239      30000 D
     61111 Isaiescu Gina                            PROD  01-NOV-98      35087      34000 N
     11121 Dumitrescu Fane                          PROD  19-MAR-03      56818      50000 N

SQL> commit;

Commit complete.

SQL> delete from pers_prod_view where marca=11121;

1 row deleted.
```

Figura 12.1 Interogarea şi actualizarea tabelei PERSONAL prin intermediul tabelei virtuale PERS_PROD_VIEW

În definiția tabelei virtuale poate fi adăugată clauza `WITH CHECK OPTION`. Scopul acesteia este de a restricționa orice încercare de actualizare a datelor care ar încălca condițiile prevăzute în clauza WHERE a frazei SELECT-definiție. Aplicată exemplului anterior, clauza `WITH CHECK OPTION` nu va permite modificarea compartimentului persoanelor si nici adăugarea unei persoane într-un alt serviciu decăt *Producție,* specificat prin definiție (vezi figura 12.2)

```
CREATE VIEW pers_prod_view AS
(SELECT * FROM Personal WHERE compart= 'PROD' )
 WITH CHECK OPTION;
```

```
SQL> update pers_prod_view set compart='FIN';
update pers_prod_view set compart='FIN'
       *
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation

SQL> insert into pers_prod_view values (72222, 'Pers Test', 'FIN', sysdate, 0,0,'N');
insert into pers_prod_view values (72222, 'Pers Test', 'FIN', sysdate, 0,0,'N')
            *
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation
```

Figura 12.2 Incercări de actualizare a unei tabele virtuale restricţionate prin clauza WHITH CHECK OPTION

Dacă între regulile definite de E.F. Codd pentru ca un SGBD să poată aspira la statutul "relaţional" se află şi cea cu privire la actualizarea tuturor tabelelor virtuale, în practică lucrurile sunt mai complicate, şi chiar, în unele situaţii, insolubile. În Oracle două condiţii trebuie obligatoriu îndeplinite pentru ca o tabelă virtuală să devină actualizabilă:

- fraza SQL ce defineşte tabela virtuală nu trebuie să conţină clauze `DISTINCT`, `GROUP BY`, operatori relaţionali (`UNION`, `INTERSECT` etc.) sau atribute calculate (generate în urma evaluării unei expresii);
- definiţia trebuie să includă toate atributele care nu acceptă valori nule şi pentru care nu au fost definite valori implicite la nivelul tabelei persistente sursă.

Ştergerea unei tabele virtuale din dicţionarul de date se poate realiza prin comanda `DROP VIEW nume_view`.

Important! Pentru o tabelă virtuală se pot specifica doar restricţii de tip `PRIMARY KEY`, `FOREIGN KEY` şi `UNIQUE`. Acestea sunt disponibile doar în mod `DISABLE NOVALIDATE`. Spre exemplu:

```
ALTER VIEW PERS_PROD_VIEW ADD PRIMARY KEY (marca)
                        DISABLE NOVALIDATE
```

Deşi, la prima vedere, pare o acţiune fără sens, operaţia de mai sus va determina stocarea restricţiei în dicţionarul bazei de date. Multe aplicaţii-client (după cum vom vedea şi în capitolul 16) apelează la dicţionarul de date pentru a afla informaţii despre restricţiile asociate tabelelor pe care apoi le utilizează în mediul propriu.

12.2. Tabele virtuale parametrizate

Tabela VIEW_PERS_PROD exemplificată mai devreme prezintă totuşi o deficienţă majoră: pentru a oferi o imagine a datelor angajaţilor din alte compartimente ar trebui să modificăm definiţia (fraza SELECT). Mult mai practic ar fi ca definiţia să fie suficient de flexibilă pentru a prelua codul compartimentului dorit la momentul interogării şi nu la momentul definirii tabelei virtuale.

La o primă vedere problema ar putea fi rezolvată prin intermediul variabilelor legate. Din păcate însă, Oracle nu permite această tehnică în frazele DDL (care au ca rezultat un obiect nou în schemă) şi, ca urmare, nu putem utiliza variabile legate nici la definirea unei tabele virtuale. Există totuşi o modalitate de a ocoli acest mecanism inflexibil: crearea unei funcţii-utilizator şi apelarea sa în fraza SELECT SQL. Funcţia respectivă poate returna valoarea dorită, dacă apelăm la o variabilă definită într-un pachet PL/SQL pentru a beneficia de "vizibilitatea" ei globală la nivelul sesiunii-utilizator curente (vezi şi capitolul 9) .

Listingul 12.1 prezintă definiția tabelei virtuale PERS_COMPART_VIEW, modificată față de exemplul anterior în sensul "parametrizării". Pachetul PL/SQL `PERS_PROD_VIEW_PARAMETERS` conține variabila `vcompart` a cărei valoare va fi returnată de funcția `getVcompart()` inclusă în clauza WHERE a frazei SELECT ce stă la baza definirii view-ului.

Listing 12.1 Paşi necesari definirii unei tabele virtuale parametrizate

```
-- pachetul ce conţine variabila globală

CREATE OR REPLACE PACKAGE pers_prod_view_parameters AS
   vcompart personal.compart%TYPE;
END;

-- funcţia care returnează valoarea variabilei globale

CREATE OR REPLACE FUNCTION getVcompart() RETURN personal.compart%TYPE
AS
BEGIN
     RETURN pers_prod_view_parameters.vcompart;
END;

-- crearea tabelei virtuale cu funcţia anterioară în clauza WHERE

CREATE OR REPLACE VIEW pers_compart_view AS
     SELECT * FROM personal WHERE compart=getVcompart() WITH CHECK OPTION
```

```
SQL>  execute PERS_PROD_VIEW_PARAMETERS.vcompart:='IT';

PL/SQL procedure successfully completed.

SQL> select * from pers_compart_view;

     MARCA NUMEPREN                                  COMPA DATASV       SALORAR  SALORARCO C
---------- ---------------------------------------- ----- --------- ---------- ---------- -
     71111 Pandele Costel                           IT    01-OCT-95      23391      25000 D
     81111 Paparuga Ina                             IT    01-AUG-97      26315      25000 N
     91111 Fratila Geo                              IT    01-MAY-97      26315      30000 N

SQL> execute PERS_PROD_VIEW_PARAMETERS.vcompart:='CONTA';

PL/SQL procedure successfully completed.

SQL> select * from pers_compart_view;

     MARCA NUMEPREN                                  COMPA DATASV       SALORAR  SALORARCO C
---------- ---------------------------------------- ----- --------- ---------- ---------- -
     11111 Ionescu Fane                             CONTA 01-JAN-78      56818      50000 D
     31111 Adascalitei Matei                        CONTA 01-MAY-90      35087      35000 N

SQL> update pers_compart_view set salorar=60000 where marca=11111;

1 row updated.

SQL>  select * from pers_compart_view;

     MARCA NUMEPREN                                  COMPA DATASV       SALORAR  SALORARCO C
---------- ---------------------------------------- ----- --------- ---------- ---------- -
     11111 Ionescu Fane                             CONTA 01-JAN-78      60000      50000 D
     31111 Adascalitei Matei                        CONTA 01-MAY-90      35087      35000 N

SQL>
```

Figura 12.3 Tabelă virtuală parametrizată

După cum se observă şi în figura 12.3, prin modificarea variabilei globale `vcompart`, tabela virtuală PERS_COMPART_VIEW prezintă un comportament dinamic în ceea ce priveşte setul de înregistrări: în primă fază sunt angajaţi din compartimentul IT iar apoi cei de la Contabilitate. Reamintim că pentru variabilele definite într-un pachet PL/SQL se crează câte o instanţă separată la nivelul fiecărei sesiuni-client. Ca urmare, nu există probleme de acces concurent la valoarea acesteia.

12.3. Tabele virtuale create pe baza unei joncţiuni interne

Oracle oferă şi posibilitatea definirii unei tabele virtuale pe baza mai multor tabele sursă prin joncţiune internă sau externă. În ceea ce priveşte actualizarea acetui tip de tabele virtuale există însă restricţia conform căreia **doar una din tabelele sursă poate constitui destinaţia actualizărilor**.

Un prim exemplu în acest sens face apel la o joncţiune internă între tabelele PERSONAL şi PONTAJE. Conform celor prezentate în capitolul 4, actualizarea datelor privind pontajul pentru o anumită persoană necesită un "efort" suplimentar din partea utilizatorului în ceea ce priveşte identificarea mărcii persoanei respective. Pentru a elimina acest pas intermediar putem defini o tabelă virtuală care să prezinte o imagine atât a datelor din tabela `PERSONAL(NumePren, SalOrar` etc.) cât şi din tabela `PONTAJE`, astfel:

```
CREATE OR REPLACE VIEW pontaje_view AS
    SELECT numepren, salorar,salorarco, colaborator, po.*
    FROM personal p, pontaje po
    WHERE p.marca=po.marca AND compart='PROD' ;
```

Figura 12.4 prezintă situaţia iniţială a tabelelor sursă, diferită, din punctul de vedere al conţinutului, de cea din capitolele anterioare.

```
SQL> select * from personal;

     MARCA NUMEPREN                                 COMPA DATASV       SALORAR  SALORARCO C
---------- ---------------------------------------- ----- --------- ---------- ---------- -
     11111 Ionescu Fane                             CONTA 01-JAN-78      56818      50000 D
     21111 Mandache Gigi                            PROD  15-AUG-82      30000      20000 N
     31111 Adascalitei Matei                        CONTA 01-MAY-90      35087      35000 N
     41111 Profitoru Nico                           FIN   01-SEP-92      25695      25000 N
     51111 Vasilescu Claudia                        PROD  15-DEC-77      29239      30000 D
     71111 Pandele Costel                           IT    01-OCT-95      23391      25000 D
     81111 Paparuga Ina                             IT    01-AUG-97      26315      25000 N
     91111 Fratila Geo                              IT    01-MAY-97      26315      30000 N
     61111 Isaiescu Gina                            PROD  01-NOV-98      35087      34000 N
     11121 Dumitrescu Fane                          PROD  19-MAR-03      56818      50000 N

10 rows selected.

SQL> select * from pontaje;

     MARCA DATA      ORELUCRATE       ORECO  ORENOAPTE  OREABSNEM
---------- --------- ---------- ---------- ---------- ----------
     11111 03-MAR-03          8          0          0          0
     31111 03-MAR-03          8          0          0          0
     41111 03-MAR-03          8          0          0          0
     51111 03-MAR-03         12          0          0          0
     71111 03-MAR-03          8          0          0          0
     81111 03-MAR-03          8          0          0          0
     91111 03-MAR-03          8          0          0          0
     61111 03-MAR-03         12          0          0          0
     11111 04-MAR-03          8          0          0          0
     31111 04-MAR-03          8          0          0          0
     41111 04-MAR-03          8          0          0          0
     51111 04-MAR-03         12          0          0          0
     71111 04-MAR-03          8          0          0          0
     81111 04-MAR-03          8          0          0          0
     91111 04-MAR-03          8          0          0          0
     61111 04-MAR-03         12          0          0          0
     11111 05-MAR-03          8          0          0          0
     31111 05-MAR-03          8          0          0          0
     41111 05-MAR-03          8          0          0          0
     51111 05-MAR-03         12          0          0          0
     71111 05-MAR-03          8          0          0          0
     81111 05-MAR-03          8          0          0          0
     91111 05-MAR-03          8          0          0          0
     61111 05-MAR-03         12          0          0          0
```

Figura 12.4 Situaţia iniţială pentru tabelele sursă ale view-ului PONTAJE_VIEW

La interogarea tabelei virtuale PONTAJE_VIEW vom obţine rezultatul din figura 12.5

```
SQL> column numepren format a20
SQL> select * from pontaje_view;

NUMEPREN             SALORAR  SALORARCO C      MARCA DATA      ORELUCRATE       ORECO  ORENOAPTE  OREABSNEM
-------------------- ---------- ---------- - ---------- --------- ---------- ---------- ---------- -
Isaiescu Gina             35087      34000 N      61111 03-MAR-03         12          0          0          0
Isaiescu Gina             35087      34000 N      61111 04-MAR-03         12          0          0          0
Isaiescu Gina             35087      34000 N      61111 05-MAR-03         12          0          0          0
Vasilescu Claudia         29239      30000 D      51111 03-MAR-03         12          0          0          0
Vasilescu Claudia         29239      30000 D      51111 04-MAR-03         12          0          0          0
Vasilescu Claudia         29239      30000 D      51111 05-MAR-03         12          0          0          0

6 rows selected.
```

Figura 12.5 Tabela virtuală PONTAJE_VIEW

Problema în acest moment constă în identificarea atributelor modificabile Răspunsul este ilustrat de figura 12.6

```
SQL> update pontaje_view set orelucrate =0, oreco=8 where marca=61111 and data=to_date('03-03-2003',
'dd-mm-yyyy');

1 row updated.

SQL>  update pontaje_view set numepren='Isaiescu Gina Claudia' where numepren='Isaiescu Gina'
  2  /
 update pontaje_view set numepren='Isaiescu Gina Claudia' where numepren='Isaiescu Gina'
                                                          *
ERROR at line 1:
ORA-01779: cannot modify a column which maps to a non key-preserved table


SQL> select * from user_updatable_columns where table_name='PONTAJE_VIEW';

OWNER                          TABLE_NAME                     COLUMN_NAME                    UPD INS DEL
------------------------------ ------------------------------ ------------------------------ --- --- ---
LIVIU                          PONTAJE_VIEW                   NUMEPREN                       NO  NO  NO
LIVIU                          PONTAJE_VIEW                   SALORAR                        NO  NO  NO
LIVIU                          PONTAJE_VIEW                   SALORARCO                      NO  NO  NO
LIVIU                          PONTAJE_VIEW                   COLABORATOR                    NO  NO  NO
LIVIU                          PONTAJE_VIEW                   MARCA                          YES YES YES
LIVIU                          PONTAJE_VIEW                   DATA                           YES YES YES
LIVIU                          PONTAJE_VIEW                   ORELUCRATE                     YES YES YES
LIVIU                          PONTAJE_VIEW                   ORECO                          YES YES YES
LIVIU                          PONTAJE_VIEW                   ORENOAPTE                      YES YES YES
LIVIU                          PONTAJE_VIEW                   OREABSNEM                      YES YES YES

10 rows selected.

SQL> |
```

Figura 12.6 Atributele actualizabile ale tabelei virtuale `PONTAJE_VIEW`

După cum se observă, pot fi actualizate doar atributele ce provin din tabela PONTAJE. Încercarea actualizării atributului `numepren` (aparţinând tabelei persistente PERSONAL) va genera mesajul de eroare `ORA-01779`. Acest comportament este determinat de un concept Oracle fundamental pentru înţelegerea view-urilor actualizabile: *key-preserved table* – adică acea tabelă a cărei cheie primară poate fi considerată şi cheie a rezultatului joncţiunii. Caracterul de tabelă cu cheie "conservată" (păstrată) nu este dat de selecţia atributelor-cheie primară în cadrul joncţiunii şi nici de datele efectiv stocate în relaţiile-sursă. Este mai degrabă o caracteristică ce decurge din schema joncţiunii şi din definiţiile celor două (sau mai multe) tabele implicate.

Dat fiind conceptul descris anterior şi analizând definiţia view-ului PONTAJE_VIEW, putem observa transferul de cheie primară (`marca` şi `data`) a tabelei PONTAJE în rezultatul joncţiunii, ceea ce-i conferă acesteia caracterul de tabelă cu cheie rezervată. Iată de ce putem actualiza prin intermediul view-ului tabela PONTAJE şi nu PERSONAL (a cărei cheie primară `marca` nu poate fi cheie a view-ului)

Pentru a afla care sunt atributele actualizabile poate fi interogată tabela USER_UPDATABLE_COLUMNS din dicţionarul de date. Figura 12.7 ilustrează două operaţii de actualizare a tabelei PONTAJE prin intermediul tabelei virtuale PONTAJE_VIEW.

```
SQL> insert into pontaje_view ( MARCA,data,orelucrate,oreco,orenoapte,oreabsnem)
  2  values (61111, to_date('06/03/2003','dd/mm/yyyy'), 8, 0,4,0);

1 row created.

SQL> delete from pontaje_view where marca=61111 and data=to_date('03/03/2003','dd/mm/yyyy');

1 row deleted.

SQL> column NUMEPREN format a30
SQL> select * from pontaje_view where marca=61111;

NUMEPREN                       SALORAR  SALORARCO C      MARCA DATA      ORELUCRATE      ORECO  ORENOAPTE
------------------------------ ---------- ---------- - ---------- --------- ---------- ---------- --
Isaiescu Gina                       35087      34000 N      61111 04-MAR-03         12          0          0
Isaiescu Gina                       35087      34000 N      61111 05-MAR-03         12          0          0
Isaiescu Gina                       35087      34000 N      61111 06-MAR-03          8          0          4

SQL>
```

Figura 12.7 Adăugare şi ştergere de înregistrăriîn PONTAJE_VIEW

12.4. Declanşatoare de tip INSTEAD-OF

Deşi nu este evidentă la prima vedere, în aplicațiile cu baze de date apare de multe ori necesitatea unei structuri de date actualizabilă creată pe baza joncțiunii externe. Pentru a înțelege mai bine acest aspect vom exemplifica o operație uzuală în sistemele de salarizare: acordarea diferențiată a sporurilor. Pentru că unii salariați vor beneficia de mai multe sporuri decât alții iar, pe parcurs, se poate renunța la unele sporuri sau pot apărea altele noi, baza de date trebuie astfel proiectată încât să ofere suficientă flexibilitate în această privință. O modalitate de a răspunde acestei situații ar fi extinderea schemei prin crearea a două noi tabele: una pentru tipuri de sporuri iar alta pentru sporurile efectiv acordate fiecărui angajat într-o anumită lună (în listingul 11.2 tabela NOMALTESPORURI respectiv ALTESPORURI). Valorile atributului `SporProc` din tabela NOMALTESPORURI reprezintă procentul sporului ce se aplică la salariul orar pentru orele lucrate corespunzător acelui spor (`Altesporuri.orespor`) de angajat în luna respectivă. Atributul `SpSuma` stochează suma de bani aferentă sporului respectiv.

Listing 11.2 Tabelele NOMALTESPORURI şi ALTESPORURI

```
CREATE TABLE NomAlteSporuri (
     sporid INTEGER PRIMARY KEY,
     sporden VARCHAR2 (50),
     sporProc NUMBER(5,2));

CREATE TABLE Altesporuri (
     marca INTEGER CONSTRAINT fk_altesp_personal REFERENCES personal(marca),
     an NUMBER (4),
     luna NUMBER (2),
     sporid INTEGER  CONSTRAINT fk_altesp_nomsp REFERENCES nomaltesporuri(sporid) ,
     orespor NUMBER (3),
     spsuma NUMBER (16,2),
      CONSTRAINT pk_altesporuri PRIMARY KEY (marca,an,luna,sporid);
```

Figura 12.8 prezintă o situație inițială a celor două noi tabele. Se observă diferențiarea sporurilor de la un angajat la altul.

```
SQL>  select * from nomaltesporuri;

    SPORID SPORDEN                         SPORPROC
---------- ------------------------------ ----------
       100 conditii grele de munca               .15
       101 nocivitate                             .1
       102 ore suplimentare                        1
       105 sef echipa de lucru                   .25
       104 spor de stres                         .06
       106 conducere                             .35

5 rows selected.

SQL> select * from altesporuri;

     MARCA         AN      LUNA     SPORID    ORESPOR     SPSUMA
---------- ---------- ---------- ---------- ---------- ----------
     21111       2003          3        102        100    3000000
     21111       2003          3        104        100     180000
     21111       2003          3        100        100     450000
     11111       2003          3        100        110     937497
     11111       2003          3        102        110    6249980
     11111       2003          3        104         90  306817.2
     11111       2003          3        106         90    1789767

7 rows selected.
```

Figura 12.8 Înregistrările initiale ale tabelelor `Nomaltesporuri` si `Altesporuri`

Pentru a înlesni operaţia de preluare a numărului de ore-spor specifice fiecărui angajat se poate costrui o tabelă virtuală care să prezinte date de identificare ale angajatului şi toate tipurile de sporuri posibile, astfel încât utilizatorul să completeze doar numărul de ore corespunzător unui spor sau altul (figura 12.9).

```
     MARCA     SPORID SPORDEN                         AN      LUNA    ORESPOR     SPSUMA
---------- ---------- ------------------------------ ---------- ---------- ---------- ----------
     21111        100 conditii grele de munca              2003          3        100     450000
     21111        101 nocivitate                           2003          3          0          0
     21111        102 ore suplimentare                     2003          3        100    3000000
     21111        104 spor de stres                        2003          3        100     180000
     21111        105 sef echipa de lucru                  2003          3          0          0
     21111        106 conducere                            2003          3          0          0
```

Figura 12.9 O imagine ameliorată a datelor ce descriu sporurile angajaţilor

O asemenea imagine s-ar putea obţine printr-o joncţiune externă la stânga între tabelele NOMALTESPORURI şi ALTESPORURI (plus, eventual, o joncţiune internă cu tabela PERSONAL dacă ar fi necesară evidenţierea şi a altor informaţii despre angajat în afară de marca acestuia).

O tabelă virtuală creată în acest sens este prezentată în listingul 12.3 alături de celelate elemente necesare (variabile globale şi funcţiile corespunzătoare) pentru a fi parametrizată după `marcă`, `an` şi `lună`. Joncţiunea externă este definită prin semnul `(+)` conform sintaxei versiunilor Oracle pre-9i2. Pentru a elimina valorile `NULL` ce rezultă pentru înregistrările din tabela din stânga (NOMALTESPORURI) ce nu au corespondent în cea din dreapta (ALTESPORURI) apelăm la funcţia `NVL`.

Listing 12.3 Tabela virtuală parametrizată bazată pe joncţiune externă la stânga

```
-- variabilele globale
    CREATE OR REPLACE PACKAGE ALTESP_VIEW_PARAMETERS
    AS
        Vluna  Altesporuri.luna%TYPE;
        Van    Altesporuri.an%TYPE;
```

```
        Vmarca Altesporuri.marca%TYPE;
    END;

-- funcţiile ce returnează valorile variabilelor globale
    CREATE OR REPLACE FUNCTION getVluna RETURN Altesporuri.luna%TYPE IS
    BEGIN
        RETURN ALTESP_VIEW_PARAMETERS.Vluna;
    END;

    CREATE OR REPLACE FUNCTION getVan RETURN Altesporuri.an%TYPE IS
    BEGIN
        RETURN ALTESP_VIEW_PARAMETERS.Van;
    END;

    CREATE OR REPLACE FUNCTION getVmarca RETURN Altesporuri.marca%TYPE IS
    BEGIN
        RETURN ALTESP_VIEW_PARAMETERS.Vmarca;
    END;

-- tabela virtuală parametrizată cu joncţiune externă la stânga
    CREATE OR REPLACE VIEW ALTESP_VIEW AS
            SELECT NVL(marca,getVmarca()) as marca,
                    NVL(a.sporid,n.sporid) as sporid,
                   sporden,
                   NVL(an,getVan()) as an,
                   NVL(luna,getVluna()) as luna,
                   NVL(orespor,0) as orespor,
                   NVL(spsuma,0) as spsuma
            FROM  nomaltesporuri n, altesporuri a  WHERE
               n.sporid=a.sporid(+) AND marca (+)=getVmarca()
               AND luna (+)=getVluna() AND an(+)=getVan();
```

Figura 12.10 ilustrează două situaţii ale sporurilor deţinute de doi angajaţi, imagini obţinute prin modificarea parametrilor şi interogarea tabelei virtuale ALTESP_VIEW. Analizând şi situaţia iniţială (figura 12.8) observăm că angajatului cu marca 61111 nu i-a fost atribuit nici un spor deocamdată. În scopul simplificării exemplului s-a evitat încă o joncţiune internă cu tabela `PERSONAL`, deşi ar fi soft necesar pentru a include în tabelă şi numele angajaţilor.

Deşi tabela virtuală ALTESP_VIEW oferă o imagine mult ameliorată a situaţiei sporurilor deţinute de diverse persoane, orice încercare de actualizare a datelor este sortită eşecului (interogarea tabelei `USER_UPDATABLE_COLUMNS` din dicţionarul de date va fi mai mult decât concludentă). Caracterul „read-only" al atributelor provine, pe de o parte, din faptul că s-a utilizat funcţia `NVL()` în definiţia view-ului, iar, pe de altă parte, este datorat imposibilităţii identificării unei tabele sursă cu cheie rezervată (concept descris ceva mai devreme).

Din fericire Oracle oferă posibilitatea înlocuirii comportamentului standard al comenzilor DML prin intermediul declanşatoarelor de tip `INSTEAD-OF` (la vcare s-a făcut trimitere în capitolul 11). Definiţia unui astfel de declanşator va indica motorului SQL acţiunea ce trebuie executată în cazul lansării unei instrucţiuni UPDATE, DELETE sau INSERT pentru tabela sau view-ul pentru care a fost declarat.

```
SQL> execute ALTESP_VIEW_PARAMETERS.Vmarca:=11111;

PL/SQL procedure successfully completed.

SQL> execute ALTESP_VIEW_PARAMETERS.Van:=2003;

PL/SQL procedure successfully completed.

SQL> execute ALTESP_VIEW_PARAMETERS.Vluna:=03;

PL/SQL procedure successfully completed.

SQL> select * from altesp_view;

     MARCA     SPORID SPORDEN                          AN       LUNA    ORESPOR     SPSUMA
---------- ---------- ------------------------------ ---------- ---------- ---------- ----------
     11111        100 conditii grele de munca              2003          3        110     937497
     11111        101 nocivitate                           2003          3          0          0
     11111        102 ore suplimentare                     2003          3        110    6249980
     11111        104 spor de stres                        2003          3         90  306817.2
     11111        105 sef echipa de lucru                  2003          3          0          0
     11111        106 conducere                            2003          3         90    1789767

6 rows selected.

SQL> execute ALTESP_VIEW_PARAMETERS.Vmarca:=61111;

PL/SQL procedure successfully completed.

SQL> select * from altesp_view;

     MARCA     SPORID SPORDEN                          AN       LUNA    ORESPOR     SPSUMA
---------- ---------- ------------------------------ ---------- ---------- ---------- ----------
     61111        100 conditii grele de munca              2003          3          0          0
     61111        101 nocivitate                           2003          3          0          0
     61111        102 ore suplimentare                     2003          3          0          0
     61111        104 spor de stres                        2003          3          0          0
     61111        105 sef echipa de lucru                  2003          3          0          0
     61111        106 conducere                            2003          3          0          0
```

Figura 12.10 Vizualizarea sporurilor pentru doi angajaţi (cu marca 11111 şi 61111).

În cazul tabelei virtuale ALTESP_VIEW, o instrucţiune UPDATE ar trebui să funcţioneze numai pentru atributul `orespor` şi ar putea să însemne:

- modificarea valorii atributului `orespor` şi a câmpului calculat `spsuma` de pe înregistrarea corespunzătoare din tabela-sursă `altesporuri`, numai dacă această înregistrare există;
- adăugarea unei noi înregistrări în tabela-sursă ALTESPORURI atunci când înregistrarea din tabela virtuală ALTESP_VIEW nu are un corespondent în tabela sursă ci este doar o imagine creată prin definiţia view-ului.

În acelaşi timp modificarea celorlalte atribute (altele decât `orespor`) nu trebuie să fie posibilă la fel ca şi adăugarea sau şteregerea de înregistrări. Pentru aceste cazuri comportamentul natural al tabelei virtuale este suficient. Listingul 12.4 prezintă un declanşator de tip `INSTEAD OF UPDATE` definit pentru tabela virtuală ALTESP_VIEW şi care se va comporta conform celor de mai sus.

Listing 12.4 Declanşator INSTEAD-OF pentru actualizarea sporurilor

```
CREATE OR REPLACE TRIGGER INST_UPD_ALTESP_VIEW INSTEAD OF UPDATE ON
Altesp_view FOR EACH ROW
 DECLARE
        salariuorar_ personal.salorar%TYPE;
        procentspor_ nomaltesporuri.sporproc%TYPE;
        rec_   altesporuri%ROWTYPE;
```

```
BEGIN
   -- preluăm datele necesare calculului sumei aferente sporului

    SELECT sporproc INTO  procentspor_ FROM nomaltesporuri WHERE sporid=:new.sporid;
    SELECT salorar INTO salariuorar_ FROM personal WHERE marca=:new.marca;

    UPDATE Altesporuri SET orespor=:new.orespor,
           spsuma=procentspor_*salariuorar_*:new.orespor
      WHERE marca=:new.marca AND an=:new.an AND luna=:new.luna AND sporid=:new.sporid;

    IF SQL%NOTFOUND THEN  -- update-ul nu a găsit înregistr. coresp. în ALTESPORURI

        INSERT INTO Altesporuri VALUES (:new.marca, :new.an, :new.luna,
                                        :new.sporid,:new.orespor,
                                        procentspor_*salariuorar_*:new.orespor);
   END IF;
END;
```

```
SQL> execute ALTESP_VIEW_PARAMETERS.Vmarca:=61111;

PL/SQL procedure successfully completed.

SQL> select * from altesporuri where marca=61111 and an=2003 and luna=3;

no rows selected

SQL>  select * from altesp_view;

     MARCA     SPORID SPORDEN                          AN       LUNA    ORESPOR     SPSUMA
---------- ---------- ------------------------------ ---------- ---------- ---------- ----------
     61111        100 conditii grele de munca              2003          3          0          0
     61111        101 nocivitate                           2003          3          0          0
     61111        102 ore suplimentare                     2003          3          0          0
     61111        104 spor de stres                        2003          3          0          0
     61111        105 sef echipa de lucru                  2003          3          0          0
     61111        106 conducere                            2003          3          0          0

6 rows selected.

SQL> update altesp_view set orespor=120 where sporid in (106,104);

2 rows updated.

SQL> select * from altesporuri where marca=61111 and an=2003 and luna=3;

     MARCA         AN       LUNA     SPORID    ORESPOR     SPSUMA
---------- ---------- ---------- ---------- ---------- ----------
     61111       2003          3        104        120   252626.4
     61111       2003          3        106        120    1473654

SQL>  select * from altesp_view;

     MARCA     SPORID SPORDEN                          AN       LUNA    ORESPOR     SPSUMA
---------- ---------- ------------------------------ ---------- ---------- ---------- ----------
     61111        100 conditii grele de munca              2003          3          0          0
     61111        101 nocivitate                           2003          3          0          0
     61111        102 ore suplimentare                     2003          3          0          0
     61111        104 spor de stres                        2003          3        120   252626.4
     61111        105 sef echipa de lucru                  2003          3          0          0
     61111        106 conducere                            2003          3        120    1473654
```

Figura 12.11 Tabela virtuală joncţionată extern devine actualizabilă prin implementarea declanşatorului din listing 12.4

În figura 12.11 vom regăsi o demonstrație a funcționării acestui trigger astfel:

1. angajatul cu marca 61111 nu are nici un spor preluat pentru luna martie 2003, aşă încât tabela virtuală afişează situaţia cu toate sporurile posibile şi valoare 0 pe câmpurile `orespor` şi `spsuma`;
2. utilizatorul modifică `orespor=120` pentru sporul de conducere şi cel de stres (cod 106 respectiv 104);
3. declanşatorul definit anterior pentru a implementa comportamentul de actualizare în tabelele de bază va adăuga cele două sporuri, cu tot cu suma aferentă, în tabela ALTESPORURI.