Capitolul 6. Interogări SQL

Pe tema importanței SQL-ului în lumea bazelor de date a curs multă cerneală şi toner, atât din stilouri, cât şi din imprimante; multe taste au fost apăsate şi bile de mouse rostogolite. Fără a exagera prea mult, cum deja începusem, se poate spune că Oracle este construit în jurul SQL. Dialectul SQL din Oracle, deşi pe alocuri diferit de standardele SQL-92 şi SQL:1999, este unul dintre cele mai puternice, prin comparație cu orice SGBD contemporan, medieval sau antic.

6.1. Sintaxa de bază a frazei SELECT

Nu o să intrăm în prea multe explicații despre fundamentele teoretice ale frazei SELECT şi despre corespondența cu algebra relațională, deoarece am face concurență neloială cărții de SQL scrisă de unul dintre autorii de față şi publicată tot la editura Polirom. Cum în mai toate capitolele precedente am presărat, din loc în loc, câte o interogare, ne mulțumim doar cu câteva exemple banale.

Care sunt angajații din compartimentul IT, ordonați alfabetic ?
Acest exemplu este dedicat celor care au deschis mai târziu aparatele de... SQL şi pune în evidență cele trei clauze cvasi-folosite (`SELECT`, `FROM` şi `WHERE`), plus cea de ordonare (`ORDER BY`) – figura 6.1.

```
SELECT *
FROM personal
WHERE compart = 'IT'
ORDER BY numepren
```

```
SQL> SELECT *
  2  FROM personal
  3  WHERE compart = 'IT'
  4  ORDER BY numepren
  5  /

  MARCA NUMEPREN            COMPART  DATASV       SALORAR  SALORARCO COLABORATOR
------- ------------------- -------- --------- ---------- ---------- -----------
    101 Angajat 1           IT       12-OCT-80      56000      55000 N
    103 Angajat 3           IT       02-JUL-76      67500      66000 N
    105 Angajat 5           IT       12-NOV-77      62500      62000 N
    109 Primul Angajat Nou  IT       10-JUL-03      55500          0 N

SQL>
```

Figura 6.1. Prima interogare (din acest capitol)

Primele două clauze sunt obligatorii. Prin comparație, există SGBD-uri, ca PostgreSQL în care clauza `FROM` este facultativă. În Oracle, atunci când valoarea ce se doreşte a fi afişată este o constantă sau, în orice caz, nu este extrasă dintr-o

tabelă anume, se recurge la o tabelă pusă la dispoziție în acest scop, DUAL, cu o singură coloană (`dummy`) şi o singura linie – vezi figura 6.2.

```
SQL> SELECT * FROM dual ;

DUMMY
-------
X
```

Figura 6.2. Impresionanta tabelă DUAL

Obținerea datei curente presupune "interogarea" acestei tabele – vezi figura 6.3. Formatul de afişare depinde de setările SQL*Plus (vezi capitolul 3) şi poate fi ameliorat folosind funcția `TO_CHAR`, despre care vom vorbi în paragraful următor:

```
SELECT SYSDATE FROM dual
```

```
SQL> SELECT SYSDATE FROM DUAL ;

SYSDATE
---------
13-MAY-03

SQL>
```

Figura 6.3. Afişarea datei curente

Predicatul de selecție (clauza `WHERE`) poate conține operatori de compație (>, ≥, <, ≤, =, ≠), dar şi operatori precum `BETWEEN`, `LIKE`, `IN`, `IS NULL` şi `EXISTS`. Primul serveşte la o mai facilă definire a unui interval numeric, calendaristic sau de tip şi de caractere. Al doilea la căutarea "inexactă" a unui şir de caractere, al treilea este legat de valorile nule, iar de ultimul ne vom ocupa abia în capitolul următor.

Care sunt angajații cu salariul orar cuprins între 55000 şi 65000 lei ?
Iată o soluție fără BETWEEN:

```
SELECT *
FROM personal
WHERE salorar >= 55000 AND salorar <= 65000
```

şi una cu:

```
SELECT *
FROM personal
WHERE salorar BETWEEN  55000 AND 65000
```

Rezultatul este cel din figura 6.4.

```
SQL> SELECT *
  2  FROM personal
  3  WHERE salorar BETWEEN  55000 AND 65000
  4  /

  MARCA NUMEPREN               COMPART  DATASV       SALORAR  SALORARCO COLABORATOR
------- ---------------------- -------- --------- ---------- ---------- -----------
    109 Primul Angajat Nou     IT       10-JUL-03      55500          0 N
    101 Angajat 1              IT       12-OCT-80      56000      55000 N
    102 Angajat 2              CONTA    12-NOV-78      57500      56000 N
    105 Angajat 5              IT       12-NOV-77      62500      62000 N
    107 Angajat 7              CONTA    21-NOV-91      61500      60000 N
```

Figura 6.4. Operatorul BETWEEN

Care sunt angajații cu nume cuprins între Calancea T. Vasile la Dumitriu F. Cezar ?
Întrebarea e un pic retorică, deoarece, prin străduniile din capitolul anterior, tabela PERSONAL conține nişte nume cu totul aiurea. Oricum, să prespunem că, la un moment dat, vom introduce date reale, iar răspunsul la întrebare va fi furnizat de interogarea:

```
SELECT *
FROM personal
WHERE numepren BETWEEN  'Calancea T. Vasile'
    AND 'Dumitriu F. Cezar'
```

Care sunt pontajele din perioada 1-6 iulie 2003 ?

```
SELECT *
FROM pontaje
WHERE data BETWEEN DATE'2003-07-01' AND DATE'2003-07-06'
```

Listați, în ordinea compartimentului şi a vechimii, personalul angajat la Contabilitate şi IT.
Atenție ! Se foloseşte operatorul `OR`, nu `AND`, chiar dacă enunțul este "...personalul angajat la CONTA **şi** IT":

```
SELECT *
FROM personal
WHERE compart = 'PROD' OR compart = 'IT'
ORDER BY compart, datasv
```

A doua soluție "se înfruptă" din operatorul `IN`:

```
SELECT *
FROM personal
WHERE compart IN ('PROD', 'IT')
ORDER BY compart, datasv
```

Numele căror persoane conține litera G pe a treia poziție ?

Acesta este un prim exemplu de căutare într-o tabelă a unui caracter/şir de caractere specificat. Cei doi specificatori multipli (jokeri) folosiți în predicate de tip LIKE sunt _ (underscore), ce substituie un singur caracter, şi % (procent) ce substituie un şir de caractere de lungime variabilă. Iată soluția pentru exemplul nostru, precum şi rezultatul (figura 6.5):

```
SELECT *
FROM personal
WHERE UPPER(numepren) LIKE '__G%'
```

```
SQL> SELECT *
  2  FROM personal
  3  WHERE UPPER(numepren) LIKE '__G%'
  4  /

  MARCA NUMEPREN              COMPART  DATASV       SALORAR  SALORARCO COLABORATOR
------- --------------------- -------- --------- ---------- ---------- -----------
    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.
```

Figura 6.5. Angajații cu litera G pe a treia poziție a numelui

Funcția UPPER asigură extragerea deopotrivă a liniilor pe care "G"-ul căutat este majusculă sau minusculă.

În numele căror persoane apare, măcar o dată, litera G (indiferent de poziție/poziții) ?

```
SELECT *
FROM personal
WHERE UPPER(numepren) LIKE '%G%'
```

De data aceasta, jokerul folosit este %, plasat atât înaintea, cât şi după litera G.

Lista persoanelor ce trebuie felicitate de Sfântul Ion

Trebuie avut în vedere ca, pe lângă "Ioni", să fie extrase şi persoanele cu prenume ca Ioan, Ioana, Ioanid, iar, pe de altă parte, nu trebuie luate în considerare persoanele cu prenumele Caraion şi Simion:

```
SELECT *
FROM PERSONAL
WHERE UPPER(Prenume) LIKE '% ION%' OR
    UPPER(Prenume) LIKE '% IOAN%'OR
    UPPER(Prenume) LIKE '%-ION%' OR
    UPPER(Prenume) LIKE '%-IOAN%'
```

6.2. Expresii şi funcții-sistem

Prelucrările la care sunt supuse informațiile extrase din baza de date reclamă folosirea a o serie de funcții şi expresii, din care câteva vor fi expuse în acest paragraf. Anexa A conține o prezentare succintă a celor mai importante dintre acestea.

6.2.1. Funcții pentru şiruri de caractere

Una dintre cele mai frecvente operațiuni cu şiruri de caractere este concatenarea, care se realizează fie prin operatorul ||, fie prin funcția CONCAT.

```
SELECT numepren || ' lucreaza in compartimentul ' ||
    compart AS text
FROM PERSONAL
```

sau

```
SELECT CONCAT(numepren, CONCAT
    (' lucreaza in compartimentul ', compart)) AS text
FROM PERSONAL
```

```
TEXT
--------------------------------------------------
Primul Angajat Nou lucreaza in compartimentul IT
Al Doilea Angajat Nou lucreaza in compartimentul PROD
Angajat 1 lucreaza in compartimentul IT
Angajat 2 lucreaza in compartimentul CONTA
Angajat 3 lucreaza in compartimentul IT
Angajat 4 lucreaza in compartimentul PROD
Angajat 5 lucreaza in compartimentul IT
Angajat 6 lucreaza in compartimentul CONTA
Angajat 7 lucreaza in compartimentul CONTA
Angajat 8 lucreaza in compartimentul PROD
```

Figura 6.6. Nişte concatenări

Concat are numai două argumente, aşa că în interogarea de mai sus au fost necesare două funcții. Interesant este că în Oracle pot fi concatenate şi date de alte tipuri decât şiruri de caractere. Spre exemplu, interogarea următoare:

```
SELECT numepren || ' lucreaza din ' || datasv ||
    ' si are salariul orar de '|| salorar || ' lei.'
   AS text
FROM PERSONAL
```

funcționează, chiar dacă NumePren este de tip şir de caractere, DataSV dată calendaristică, iar SalOrar numeric.

Tot în categoria funcțiilor folosite frecvente intră cele de conversie: UPPER (toate literele vor fi convertite în majuscule), LOWER (toate literele vor fi convertite în minuscule), INITCAP (prima literă din fiecare cuvânt este majusculă, iar restul litere mici). Iată o interogare lămuritoare şi un rezultat pe măsură – figura 6.7.

```
SELECT UPPER(numepren) , LOWER(numepren), INITCAP(numepren)
```

```
FROM PERSONAL
```

```
UPPER(NUMEPREN)        LOWER(NUMEPREN)        INITCAP(NUMEPREN)
---------------------- ---------------------- ----------------------
PRIMUL ANGAJAT NOU     primul angajat nou     Primul Angajat Nou
AL DOILEA ANGAJAT NOU  al doilea angajat nou  Al Doilea Angajat Nou
ANGAJAT 1              angajat 1              Angajat 1
ANGAJAT 2              angajat 2              Angajat 2
ANGAJAT 3              angajat 3              Angajat 3
ANGAJAT 4              angajat 4              Angajat 4
ANGAJAT 5              angajat 5              Angajat 5
ANGAJAT 6              angajat 6              Angajat 6
ANGAJAT 7              angajat 7              Angajat 7
ANGAJAT 8              angajat 8              Angajat 8
```

Figura 6.7. Litere mari, mici şi combinate

Completarea valorilor unui atribut/expresie cu un caracter până la atingerea unei lungimi specificate se realizează prin funcțiile `LPAD` (completare la stânga) şi `RPAD` (completare la dreapta). Interogarea următoare completează cu spații, la stânga şi la dreapta, valorile a atributului `Compart` – vezi figura 6.8. După lungimea pe care se va face completarea se poate specifica un alt caracter decât spațiul, pentru "umplutură". În premieră, numele coloanei (clauza `AS`) face apel la ghilimele pentru a putea incluse spații.

```
SELECT '*' || compart || '*' AS compart,
    '*' || LPAD(compart,15) || '*' AS "     L_PAD_15    ",
    '*' || RPAD(compart,15) || '*' AS "      R_PAD_15    "
FROM PERSONAL
```

```
SQL> SELECT '*' || compart || '*' AS compart,
  2  '*' || LPAD(compart,15) || '*' AS "     L_PAD_15    ",
  3  '*' || RPAD(compart,15) || '*' AS "      R_PAD_15    "
  4  FROM PERSONAL
  5  /

COMPART      L_PAD_15           R_PAD_15
------- ---------------- ----------------
*IT*    *             IT* *IT             *
*PROD*  *           PROD* *PROD           *
*IT*    *             IT* *IT             *
*CONTA* *          CONTA* *CONTA          *
*IT*    *             IT* *IT             *
*PROD*  *           PROD* *PROD           *
*IT*    *             IT* *IT             *
*CONTA* *          CONTA* *CONTA          *
*CONTA* *          CONTA* *CONTA          *
*PROD*  *           PROD* *PROD           *

10 rows selected.

SQL>
```

Figura 6.8. Completare cu spații, la stânga şi la dreapta

Operațiunea inversă, de eliminare ("tundere") a spațiilor de la stânga sau de la dreapta valorii unui atribut/expresie se realizează prin funcțiile `LTRIM` şi `RTRIM`. La prima am apelat în capitolul 4 când am definit o regulă de validare pentru

atributului `numepren` prin care interziceam ca primul caracter dintr-un nume să fie spațiu. Pentru eliminarea simultană a spațiilor la stânga şi la dreapta, se recomandă funcția `TRIM` cu opțiunea `BOTH`.

Atunci când se doreşte înlocuirea unui şir de caractere cu un altul într-o expresie de acest tip se poate recurge la funcția `REPLACE`, iar pentru a extrage o porțiune dintr-un şir `SUBSTR`. Interogarea următoare înlocuieşte toate "a"-urile cu "u"-uri din numele angajaților şi extrage din acelaşi atribut patru caractere, începând cu al cincilea:

```
SELECT numepren,
    REPLACE(numepren, 'a', 'u'),
    SUBSTR (numepren, 5, 4)
FROM PERSONAL
```

NUMEPREN	REPLACE(NUMEPREN,'A','U'	SUBSTR(NUMEPREN,5,4)
Primul Angajat Nou	Primul Angujut Nou	ul A
Al Doilea Angajat Nou	Al Doileu Angujut Nou	oile
Angajat 1	Angujut 1	jat
Angajat 2	Angujut 2	jat
Angajat 3	Angujut 3	jat
Angajat 4	Angujut 4	jat
Angajat 5	Angujut 5	jat
Angajat 6	Angujut 6	jat
Angajat 7	Angujut 7	jat
Angajat 8	Angujut 8	jat

Figura 6.9. Funcțiile REPLACE şi SUBSTR

Dintre funcțiile "de informare" despre un şir de caractere, ne oprim asupra celei care numără lungimea şirului şi o alta care depistează pe a câta poziție a unei expresii/atribut apare un caracter sau şir specificat. Astfel, interogarea:

```
SELECT numepren, LENGTH (numepren),
    INSTR (numepren, 'j'), INSTR (numepren, 'D')
FROM PERSONAL
```

determină, pentru numele fiecărui angajat, numărul de carctere care-l alcătuiesc şi pe ce poziție apar literele j şi D – vezi figura 6.10.

```
SQL> SELECT numepren, LENGTH (numepren), INSTR (numepren, 'j'), INSTR (numepren, 'D')
  2  FROM PERSONAL
  3  /
```

NUMEPREN	LENGTH(NUMEPREN)	INSTR(NUMEPREN,'J')	INSTR(NUMEPREN,'D')
Primul Angajat Nou	18	12	0
Al Doilea Angajat Nou	21	15	4
Angajat 1	9	5	0
Angajat 2	9	5	0
Angajat 3	9	5	0
Angajat 4	9	5	0
Angajat 5	9	5	0
Angajat 6	9	5	0
Angajat 7	9	5	0
Angajat 8	9	5	0

Figura 6.10. Funcțiile LENGTH şi INSTR

6.2.2. Funcții pentru valori numerice

Trecem peste funcțiile trigonometrice şi algebrice care în aplicațiile economice se regăsesc ceva mai rar. Ajugem rapid la câteva funcții de trunchiere/rotunjire:

- `CEIL(p)` – întoarce cel mai mic întreg mai mare sau egal ca argumentul (`p`);
- `FLOOR (p)`– întoarce cel mai mare întreg mai mic sau egal cu `p`;
- `ROUND (p,n)` – rotunjeşte rezultatul unei expresii (`p`) la un număr de poziții fracționare, dar `n` este pozitiv sau, dacă `n` este negativ, rotunjirea se face la ordinul zecilor, sutelor, miilor etc.
- `TRUNC (p,n)` – are efect similar funcției `ROUND`, numai că în loc de rotunjire (incrementare cu 1 a poziției de referință, dacă de poziția de ordin inferior are valoeare mai mare sau egală cu 5) se face trunchiere.

Pentru comparație, interogarea de mai jos foloseşte toate cele patru funcții, rotunjirea realizându-se atât cu argument pozitiv, cât şi negativ. Rezultatul din figura 6.11 este câte se poate de grăitor.

```
SELECT marca, salorar, salorar/168,
CEIL(salorar/168) AS CEIL , FLOOR(salorar/168) AS FLOOR,
ROUND(salorar/168,2) AS "ROUND+", ROUND(salorar/168,-2)
    AS "ROUND-",
TRUNC(salorar/168,2) AS "TRUNC+"
FROM PERSONAL
```

```
SQL> SELECT marca, salorar, salorar/168,
  2  CEIL(salorar/168) AS CEIL , FLOOR(salorar/168) AS FLOOR,
  3  ROUND(salorar/168,2) AS "ROUND+", ROUND(salorar/168,-2) AS "ROUND-",
  4  TRUNC(salorar/168,2) AS "TRUNC+"
  5  FROM PERSONAL
  6  /

 MARCA  SALORAR SALORAR/168       CEIL      FLOOR     ROUND+     ROUND-     TRUNC+
------ -------- ----------- ---------- ---------- ---------- ---------- ----------
   109    55500  330.357143        331        330     330.36        300     330.35
   110    50500  300.595238        301        300      300.6        300     300.59
   101    56000  333.333333        334        333     333.33        300     333.33
   102    57500  342.261905        343        342     342.26        300     342.26
   103    67500  401.785714        402        401     401.79        400     401.78
   104    75000  446.428571        447        446     446.43        400     446.42
   105    62500   372.02381        373        372     372.02        400     372.02
   106    71500  425.595238        426        425      425.6        400     425.59
   107    61500  366.071429        367        366     366.07        400     366.07
   108    54500  324.404762        325        324      324.4        300      324.4
```

Figura 6.11. Câteva funcții numerice

6.2.3. Date calendaristice

Oracle şi-a ameliorat, odată cu ultimele versiuni, nu numai funcțiile pentru lucrul cu date calendaristice, dar mai ales conformitatea cu standardele SQL. Dacă, tradițional, singura funcție ce furniza data sistemului era una "proprietară", SYSDATE, de la 9i apare şi CURRENT_DATE, preluată din SQL. Tot în Oracle 9i apar funcțiile CURRENT_TIMESTAMP şi SYSTIMESTAMP ce furnizează, deopotrivă, data curentă, ora exactă (atât cât de bine este reglat ceasul calculatorului), la fracțiuni de secundă ce pot fi specificate.

```
SELECT CURRENT_DATE AS " CURRENT_DATE" ,
    CURRENT_TIMESTAMP AS " Data si ora curenta "
FROM DUAL
```

```
SQL> SELECT CURRENT_DATE AS " CURRENT_DATE" ,
  2  CURRENT_TIMESTAMP AS " Data si ora curenta "
  3  FROM DUAL
  4  /

 CURRENT_  Data si ora curenta
--------- ---------------------------------------
17-MAY-03 17-MAY-03 08.44.56.325000 AM -07:00
```

Figura 6.12. Data şi ora exactă !

Pentru operații cu date calendaristice, fucțiile disponibile încă din Oracle 7 sunt:

- ADD_MONTHS(data) – adună un număr de luni la data argument;
- LAST_DAY(data) – furnizează ultima zi din luna în care se află data argument;
- NEXT_DAY(data, zi) – întoarce data primei zile (luni, marți...) ce urmează datei argument;

Astfel, interogarea următoarea afişează ziua curentă (lansării comenzii SELECT), data de peste două luni calendaristice, ultima zi a lunii curente şi data viitoare în care o cadă o zi de marți – vezi şi figura 6.13.

```
SELECT SYSDATE AS "Astazi",
    ADD_MONTHS (SYSDATE,2) AS "Peste doua luni",
    LAST_DAY(SYSDATE) "Ultima_zi_a_lunii_curente",
    NEXT_DAY(SYSDATE, 'TUESDAY') AS "Următoarea marți"
FROM DUAL
```

```
SQL> SELECT SYSDATE AS "Astazi", ADD_MONTHS (SYSDATE,2) AS "Peste doua luni",
  2  LAST_DAY(SYSDATE) "Ultima_zi_a_lunii_curente",
  3  NEXT_DAY(SYSDATE, 'TUESDAY') AS "Urmatoarea marti"
  4  FROM DUAL
  5  /

Astazi    Peste doua luni   Ultima_zi_a_lunii_curente   Urmatoarea marti
--------- ----------------- --------------------------- ----------------
17-MAY-03 17-JUL-03         31-MAY-03                   20-MAY-03
```

Figura 6.13. Funcțiile ADD_MONTHS, LAST_DAY şi NEXT_DAY

Cu funcțiile ROUND şi TRUNC am făcut cunoştință ceva mai înainte. E îmbucurător că rotunjirea şi trunchierea funcționază şi pentru date calendaristice, după logica acestora. Astfel, ROUND(data [, format]) rotunjeşte o dată calendaristică după un format specificat, iar TRUNC (data [, format]) trunchiază o dată calendaristică după formatul specificat. Spre exemplu, dacă la format se specifică YEAR (sau YYYY) şi data-argument este mai mică decât 1 iulie, ROUND şi TRUNC întorc anul curent; dacă însă data-argument este după 1 iulie, atunci ROUND întoarce anul următor, în timp ce TRUNC tot anul curent. Când formatul se referă la lună (MONTH, MON sau MM), atunci luna furnizată este luna curentă, însă dacă data "cade" în a doua jumătate (după 15 ale lunii), ROUND întoarce luna următoare:

```
SELECT SYSDATE AS “Astazi”,
    ROUND(SYSDATE,’YEAR’) AS Rot_AN,
    TRUNC(SYSDATE, ‘YYYY’) AS Trunc_AN,
    ROUND(SYSDATE,’MON’) AS Rot_LUNA,
    TRUNC(SYSDATE, ‘MONTH’) AS Trunc_LUNA,
    ROUND(SYSDATE,’DDD’) AS Rot_ZI,
    TRUNC(SYSDATE, ‘DDD’) AS Trunc_ZI
FROM DUAL
```

```
SQL> SELECT SYSDATE AS "Astazi",
  2  ROUND(SYSDATE,'YEAR') AS Rot_AN, TRUNC(SYSDATE, 'YYYY') AS Trunc_AN,
  3  ROUND(SYSDATE,'MON') AS Rot_LUNA, TRUNC(SYSDATE, 'MONTH') AS Trunc_LUNA,
  4  ROUND(SYSDATE,'DDD') AS Rot_ZI, TRUNC(SYSDATE, 'DDD') AS Trunc_ZI
  5  FROM DUAL
  6  /

Astazi    ROT_AN    TRUNC_AN  ROT_LUNA  TRUNC_LUN ROT_ZI    TRUNC_ZI
--------- --------- --------- --------- --------- --------- ---------
17-MAY-03 01-JAN-03 01-JAN-03 01-JUN-03 01-MAY-03 17-MAY-03 17-MAY-03
```

Figura 6.14. Diverse rotunjiri şi trunchieri

Dacă pentru extragerea anului/lunii/zilei dintr-o dată calendaristică versiunile mai vechi ale Oracle necesitau funcții de conversie precum TO_CHAR sau TO_NUMBER, în 8i şi 9i lucrurile s-au simplificat simțitor. Similar standardelor SQL, prin funcția EXTRACT putem afla orice componentă a unei date/ore:

```
SELECT EXTRACT (YEAR FROM SYSDATE) AS Anul,
```

```
    EXTRACT (MONTH FROM SYSDATE) AS Luna,
    EXTRACT (DAY FROM SYSDATE) AS Ziua,
    EXTRACT (HOUR FROM SYSTIMESTAMP) AS Ora,
    EXTRACT (MINUTE FROM SYSTIMESTAMP) AS Minutul,
    EXTRACT (SECOND FROM SYSTIMESTAMP) AS Secunda
FROM DUAL
```

Cum este şi firesc, argumentele `YEAR`, `MONTH`, `DAY` pot fi aplicate unui atribut, variabilă sau constantă de tip dată calendaristică, în timp ce `HOUR`, `MINUTE`, `SECOND` unei expresii de tip timp.

```
SQL> SELECT EXTRACT (YEAR FROM SYSDATE) AS Anul,
  2   EXTRACT (MONTH FROM SYSDATE) AS Luna,
  3   EXTRACT (DAY FROM SYSDATE) AS Ziua,
  4   EXTRACT (HOUR FROM SYSTIMESTAMP) AS Ora,
  5   EXTRACT (MINUTE FROM SYSTIMESTAMP) AS Minutul,
  6   EXTRACT (SECOND FROM SYSTIMESTAMP) AS Secunda
  7  FROM DUAL
  8  /

      ANUL       LUNA       ZIUA        ORA    MINUTUL    SECUNDA
---------- ---------- ---------- ---------- ---------- ----------
      2003          5         17         19         54     53.542
```

Figura 6.15. Funcția EXTRACT

În privința operațiunilor cu datele calendaristice, se cuvine să începem prin a reaminti că dacă o constantă sau o variabilă întreagă apare într-o expresie cu un atribut de tip dată, constanta/variabila se va considera a fi un număr de zile. Curios lucru, lucrurile stau identic pentru atributele de tip timp:

```
SELECT SYSDATE AS Azi,
    SYSDATE + 43 AS "Peste 43 de zile",
    SYSTIMESTAMP AS "Acum",
    SYSTIMESTAMP + 34 AS "Peste 34 de..."
FROM DUAL
```

Astfel, ne-am aştepta ca `SYSTIMESTAMP + 34` să calculeze ora exactă de peste 34 secunde sau minute, însă, după cum se observă în figura 6.16, 34 este considerat numărul de zile.

```
SQL> SELECT SYSDATE AS Azi, SYSDATE + 43 AS "Peste 43 de zile",
  2   SYSTIMESTAMP AS "Acum", SYSTIMESTAMP + 34 AS "Peste 34 de..."
  3  FROM DUAL
  4  /

AZI       Peste 43 de zile   Acum                                   Peste 34 de...
--------- ------------------ ------------------------------------- --------------
17-MAY-03 29-JUN-03          17-MAY-03 01.57.36.443000 PM -07:00   20-JUN-03
```

Figura 6.16. Incrementări ale unor valori de tip dată şi timp

Dacă dorim să incrementăm o dată cu un interval de ordinul lunilor/anilor, o funcţie de mare ajutor este `ADD_MONTHS` pe care am folosit-o ceva mai înainte. Astfel, raportat la ziua curentă (18 MAI 2003 – data execuţiei), peste 2 ani, 2 luni şi 12 zile va fi data obţinută prin expresia `ADD_MONTHS (SYSDATE, 26) + 12`, unde 26 este 2 ani * 12 luni + 2 luni. Similar standardelor SQL, ultimele versiuni Oracle au introdus un mult mai elegant tip de dată, `INTERVAL`, aşa încât putem formula aceeaşi condiţie mult mai explicit: `SYSDATE + INTERVAL '2' YEAR + INTERVAL '2' MONTH + INTERVAL '12' DAY`. Iată interogarea şi rezultatul acesteia – figura 6.17.

```
SELECT SYSDATE AS Azi,
    ADD_MONTHS (SYSDATE, 26) + 12 AS "Peste 2a_2l_12z",
    SYSDATE + INTERVAL '2' YEAR + INTERVAL '2' MONTH +
          INTERVAL '12' DAY AS Idem,
FROM DUAL
```

```
SQL> SELECT SYSDATE AS Azi,
  2  ADD_MONTHS (SYSDATE, 26) + 12 AS "Peste 2a_2l_12z",
  3  SYSDATE + INTERVAL '2' YEAR + INTERVAL '2' MONTH +
  4  INTERVAL '12' DAY AS Idem
  5  FROM DUAL
  6  /

AZI         Peste 2a_2l_12z     IDEM
----------- ------------------- ---------
18-MAY-03   30-JUL-05           30-JUL-05
```

Figura 6.17. Somewhere in time

Lucrurile pot fi aplicate similar şi pentru atribute/variabile/constante/expresii de tip timp. Astfel, pentru a afla ora exactă de peste 71 minute şi 100 de secunde de la momentul curent (cel al execuţiei interogării) se poate folosi fraza `SELECT` următoare – vezi figura 6.18.

```
SELECT SYSTIMESTAMP AS Ora_exacta,
    SYSTIMESTAMP + INTERVAL '71' MINUTE  +
    INTERVAL '100' SECOND AS Peste_71min_100sec
FROM DUAL
```

```
SQL> SELECT SYSTIMESTAMP AS Ora_exacta,
  2  SYSTIMESTAMP + INTERVAL '71' MINUTE  +
  3  INTERVAL '100' SECOND AS Peste_71min_100sec
  4  FROM DUAL
  5  /

ORA_EXACTA                                PESTE_71MIN_100SEC
----------------------------------------- ----------------------------------------
18-MAY-03 11.39.46.125000 AM -07:00       18-MAY-03 12.52.26.125000000 PM -07:00
```

Figura 6.18. Un interval

Mai mult, în Oracle 9i există două tipuri de date `INTERVAL`, `YEAR TO MONTH` şi `DAY TO SECOND` care pot fi asociate unor atribute, ceea ce uşurează sensibil gestionarea unor informații de tip perioadă calendaristică. Spre exemplu, echivalent celor două variante de mai sus, data care "cade" peste exact 2 ani, două luni şi 34 de zile se poate obține astfel:

```
SELECT SYSDATE AS Azi,
    SYSDATE + INTERVAL '2-2' YEAR TO MONTH + 34,
    SYSDATE + INTERVAL '2-2' YEAR(1) TO MONTH + 34
FROM DUAL
```

Literalul '2-2' indică un număr de ani egal cu 2, cratima este separatorul, iar cel de-al doilea 2 reprezintă numărul de luni. Al doilea format precizează, prin `YEAR(1)`, numărul de poziții pe care se reprezintă anii din interval. Dacă la acest gen de interval delimitatorul este cratima (-), la intervale orare se foloseşte ":". Interogarea următoare afişează, pe rând, următoarele momente: curent, cel de peste două zile, 8 ore şi 34 minute, cel de peste 17 ore şi 49 minute şi cel de peste 120 zile, 12 ore, 44 minute şi 45 secunde:

```
SELECT 'Acum ' AS "Ce se doreste",
  CURRENT_TIMESTAMP AS Momentul
FROM DUAL
      UNION
SELECT 'Peste 2 zile, 8 ore si 34 minute',
  CURRENT_TIMESTAMP + INTERVAL '2 8:34' DAY (1) TO MINUTE
FROM DUAL
      UNION
SELECT  'Peste 17 ore si 49 minute',
  CURRENT_TIMESTAMP + INTERVAL '17:49' HOUR TO MINUTE
FROM DUAL
      UNION
SELECT 'Peste 120 de zile, 12 ore, 44 minute si 45 secunde',
  CURRENT_TIMESTAMP + INTERVAL '120 12:44:45'
      DAY (3) TO SECOND
FROM DUAL
```

Rezultatul este cel din figura 6.19. Pentru un plus de fotogenie, s-a recurs la reuniunea a patru fraze `SELECT`, anticipând scurta discuție despre operatorul `UNION` pe care o vom purta peste câteva pagini.

```
SQL> SELECT 'Acum ' AS "Ce se doreste",
  2   CURRENT_TIMESTAMP AS Momentul FROM DUAL
  3  UNION
  4  SELECT 'Peste 2 zile, 8 ore si 34 minute',
  5  CURRENT_TIMESTAMP + INTERVAL '2 8:34' DAY (1) TO MINUTE  FROM DUAL
  6  UNION
  7  SELECT  'Peste 17 ore si 49 minute',
  8  CURRENT_TIMESTAMP + INTERVAL '17:49' HOUR TO MINUTE FROM DUAL
  9  UNION
 10  SELECT  'Peste 120 de zile, 12 ore, 44 minute si 45 secunde',
 11  CURRENT_TIMESTAMP + INTERVAL '120 12:44:45' DAY (3) TO SECOND FROM DUAL
 12  /

Ce se doreste                                      MOMENTUL
-------------------------------------------------- ---------------------------------------
Acum                                               19-MAY-03 09.54.58.699000000 AM -07:00
Peste 120 de zile, 12 ore, 44 minute si 45 secunde 16-SEP-03 10.39.43.699000000 PM -07:00
Peste 17 ore si 49 minute                          20-MAY-03 03.43.58.699000000 AM -07:00
Peste 2 zile, 8 ore si 34 minute                   21-MAY-03 06.28.58.699000000 PM -07:00
```

Figura 6.19. Alte intervale

Formatul de afişare al datei calendaristice poate fi modificat prin comanda ALTER SESSION:

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MM-YYYY HH24:MI:SS'
```

După această comandă, datele calendaristice vor fi afişate în formatul specificat – vezi figura 6.20.

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MM-YYYY HH24:MI:SS'
  2  /

Session altered.

SQL> SELECT SYSDATE FROM DUAL ;

SYSDATE
-------------------
19-05-2003 11:20:49
```

Figura 6.20. Modificarea formatului de afişare a datei

De reţinut că nu se pot aduna, şi nici scădea, două date calendaristice. Ne interesează capătul perioadei calendaristice în care jumătatea este ziua curentă (18 mai 2003, data execuţiei). Suntem tentaţi să folosim interogarea:

```
SELECT SYSDATE + SYSDATE
FROM DUAL
```

Execuţia sa se soldează cu mesajul de eroare din figura 6.21 care este destul de limpede în privinţa motivului declanşării erorii.

```
SQL> SELECT SYSDATE + SYSDATE
  2  FROM DUAL
  3  /
SELECT SYSDATE + SYSDATE
               *
ERROR at line 1:
ORA-00975: date + date not allowed
```

Figura 6.21. Eroare la adunarea a două date calendaristice

Pentru a rezolva problema, trebuie să transformăm data curentă în interval:

```
SELECT CURRENT_DATE AS Astazi,
    SYSDATE + INTERVAL '2002-05' YEAR(4) TO MONTH +
          INTERVAL '19' DAY AS Oale_si_ulcele
FROM DUAL
```

```
SQL> SELECT CURRENT_DATE AS Astazi,
  2  SYSDATE + INTERVAL '2002-05' YEAR(4) TO MONTH +
  3  INTERVAL '19' DAY AS Oale_si_ulcele
  4  FROM DUAL
  5  /

ASTAZI              OALE_SI_ULCELE
------------------- -------------------
19-05-2003 16:40:55 07-11-4005 16:40:55
```

Figura 6.22. Conversia unei date în interval

Tabela PERSONAL incorporează atributul `DataSV` ce reprezintă data de la care se calculează anii de vechime ai angajatului, ani în funcție de care se acordă sporul de vechime. De aceea, o importanță deosebită o va avea, pe parcursul multora dintre capitolele viitoare, calculul numărului întreg de ani scurşi de la data de calcul sporului de vechime până la 1 ale lunii curente (aprilie 2003, mai 2003 etc.). Funcția la care se poate apel din Oracle 7 încoace este `MONTHS_BETWEEN(data1, data2)` care calculează numărul de luni (cu tot cu poziții fracționare) dintre două date calendaristice. Fraza `SELECT` de mai jos determină numărul de ani cuprinşi între 1 mai 2003 şi data sporului de vechime. În prealabil modificăm formatul de afişare a datei calendaristice.

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MM-YYYY' ;

SELECT numepren, datasv,
    MONTHS_BETWEEN( DATE'2003-05-01', datasv) AS Nr_Luni,
    TRUNC(MONTHS_BETWEEN( DATE'2003-05-01', datasv)/12,0)
          AS Ani_Intregi
FROM personal ;
```

Întrucât la vechime interesează numărul **întreg** de ani, fără rotunjiri, câtul împărțirii funcției MONTHS_BETWEEN la 12 este trunchiat. Rezultatul este cel din figura 6.23.

```
SQL> SELECT numepren, datasv,
  2  MONTHS_BETWEEN( DATE'2003-05-01', datasv) AS Nr_Luni,
  3  TRUNC(MONTHS_BETWEEN( DATE'2003-05-01', datasv)/12,0) AS Ani_Intregi
  4  FROM personal ;

NUMEPREN             DATASV        NR_LUNI ANI_INTREGI
-------------------- ---------- ---------- -----------
Primul Angajat Nou   10-07-2003 -2.2903226           0
Al Doilea Angajat Nou 10-07-2003 -2.2903226           0
Angajat 1            12-10-1980 270.645161          22
Angajat 2            12-11-1978 293.645161          24
Angajat 3            02-07-1976 321.967742          26
Angajat 4            05-01-1982 255.870968          21
Angajat 5            12-11-1977 305.645161          25
Angajat 6            11-04-1985 216.677419          18
Angajat 7            21-11-1991 137.354839          11
Angajat 8            30-12-1994 100.064516           8
```

Figura 6.23. Funcția MONTHS_BETWEEN

De la Oracle 9i putem beneficia de una dintre "măreţele cuceriri ale nucleului SQL, şi anume lucrul cu intervale. Astfel, dacă dorim să vedem câți ani şi câte luni s-au scurs între două date calendaristice, până în Oracle 8i, am avea nevoie de număr de virtuozitare SQL. Acum, însă, pe baza tipului INTERVAL YEAR TO MONTH, lucrurile devin o banalitate: (DATE'2003-05-01' - datasv) YEAR(2) TO MONTH. Iar dacă anterior am folosit funcția EXTRACT pentru determinarea unei componente a datei calendaristice (ziua, luna sau anul), aceeaşi funcție poate fi aplicată şi intervalelor, aşa încât EXTRACT (YEAR FROM ((DATE'2003-05-01' - datasv) YEAR(2) TO MONTH)) va furniza numărul de ani pe care îi conține intervalul. Zis şi făcut:

```
SELECT numepren, datasv,
  MONTHS_BETWEEN( DATE'2003-05-01', datasv)/12
     AS Ani_Fractionari,
  (DATE'2003-05-01' - datasv) YEAR(2) TO MONTH
     AS Ani_si_Luni,
  EXTRACT (YEAR FROM ((DATE'2003-05-01' - datasv) YEAR(2)
     TO MONTH)) AS Ani
FROM personal
```

Tocmai pentru a păstra o paralelă cu interogarea anterioară, se afişează, pe rând: data de la care se calculează sporul de vechime, numărul de ani calculați cu parte fracționară (prin MONTHS_BETWEEN), intervalul anilor şi lunilor scurse între 1 mai 2003 şi data sporului de vechime şi, în fine, glazura de pe tort (sau bomboana de pe coliva, depinde cum priviți lucrurile), numărul de ani extras din interval – vezi figura 6.24.

```
SQL> SELECT numepren, datasv,
  2  MONTHS_BETWEEN( DATE'2003-05-01', datasv)/12 AS Ani_Fractionari,
  3  (DATE'2003-05-01' - datasv) YEAR(2) TO MONTH AS Ani_si_Luni,
  4  EXTRACT (YEAR FROM ((DATE'2003-05-01' - datasv) YEAR(2) TO MONTH)) AS Ani
  5  FROM personal
  6  /

NUMEPREN                 DATASV    ANI_FRACTIONARI ANI_SI_LUNI       ANI
------------------------ --------- --------------- --------------- -----
Primul Angajat Nou       10-JUL-03      -.19086022 -000000000-02       0
Al Doilea Angajat Nou    10-JUL-03      -.19086022 -000000000-02       0
Angajat 1                12-OCT-80      22.5537634 +000000022-07      22
Angajat 2                12-NOV-78      24.4704301 +000000024-06      24
Angajat 3                02-JUL-76      26.8306452 +000000026-10      26
Angajat 4                05-JAN-82      21.3225806 +000000021-04      21
Angajat 5                12-NOV-77      25.4704301 +000000025-06      25
Angajat 6                11-APR-85      18.0564516 +000000018-01      18
Angajat 7                21-NOV-91      11.4462366 +000000011-05      11
Angajat 8                30-DEC-94      8.33870968 +000000008-04       8
```

Figura 6.24. Intervale obținute prin scădere şi numărul de ani extras dintr-un interval

6.2.4. Conversii dintr-un tip în altul

Dacă s-ar fi făcut o statistică privind gradul de folosire a funcțiilor sistem, istoria Oracle ar fi cosemnat în unul din primele locuri: `TO_CHAR`, `TO_DATE` şi `TO_NUMBER`, adică funcțiile clasice de conversie între tipurile majore de date, numere, şiruri de caractere şi date calendaristice. Astfel, o constantă de tip dată calendaristică poate fi definită atât în maniera `DATE'2003-05-01'`, cât şi `TO_DATE('01/05/2003', 'DD/MM/YYYY')`. Prima variantă e mai scurtă, însă a doua mai flexibilă. Noi am ales şablonul `zi/lună/an`, dar cei curioşi pot consulta documentația Oracle pentru a se minuna de puzderia de şabloane pusă la dispoziție de nucleul SQL al acestui produs.

Operațiunea inversă, de transformare a unei date calendaristice în şir de caractere se realizează prin funcția `TO_CHAR`. Tot `TO_CHAR` se face conversia unui număr într-un şir de caractere, după un şablon specificat:

```
SELECT numepren,
   TO_DATE('01/05/2003', 'DD/MM/YYYY') AS "1_Mai_Muncitoresc",
   TO_CHAR(datasv, 'DD-MM-YYYY') AS "Data->Sir",
   TO_CHAR(salorar, '999999') AS "Numar->Sir",
   TO_NUMBER(TO_CHAR(datasv, 'MM'))
      AS "Luna (Data->Sir->Numar)"
FROM personal
```

În schimb, conversia unui şir de caractere într-un număr, cu sau fără şablon, se realizează prin funcția `TO_NUMBER`. Ultima coloană a interogării de mai sus (vezi figura 6.25) extrage luna din data sporului de vechime, prin `TO_CHAR`, iar şirul obținut este convertit apoi în număr.

```
SQL> SELECT numepren,
  2  TO_DATE('01/05/2003', 'DD/MM/YYYY') AS "1_Mai_Muncitoresc",
  3  TO_CHAR(datasv, 'DD-MM-YYYY') AS "Data->Sir",
  4  TO_CHAR(salorar, '999999') AS "Numar->Sir",
  5  TO_NUMBER(TO_CHAR(datasv, 'MM')) AS "Luna (Data->Sir->Numar)"
  6  FROM personal
  7  /

NUMEPREN             1_Mai_Muncitoresc  Data->Sir    Numar->Sir     Luna (Data->Sir->Numar)
-------------------- ------------------ ---------- -------------- -----------------------
Primul Angajat Nou   01-MAY-03          10-07-2003     55500                          7
Al Doilea Angajat Nou 01-MAY-03         10-07-2003     50500                          7
Angajat 1            01-MAY-03          12-10-1980     56000                         10
Angajat 2            01-MAY-03          12-11-1978     57500                         11
Angajat 3            01-MAY-03          02-07-1976     67500                          7
Angajat 4            01-MAY-03          05-01-1982     75000                          1
Angajat 5            01-MAY-03          12-11-1977     62500                         11
Angajat 6            01-MAY-03          11-04-1985     71500                          4
Angajat 7            01-MAY-03          21-11-1991     61500                         11
Angajat 8            01-MAY-03          30-12-1994     54500                         12
```

Figura 6.25. Conversii

Odată cu versiunea 9i, apar, printre altele, două funcții de conversie cu un cert grad de utilitate. Prima este NUMYMINTERVAL care converteşte un număr într-un interval YEAR TO MONTH. Exemplu:

```
SELECT '4 ani' AS Descriere,
    NUMTOYMINTERVAL (4, 'YEAR') AS "Interval"
FROM dual UNION
SELECT '7 luni',
    NUMTOYMINTERVAL (7, 'MONTH')
FROM dual UNION
SELECT '4 ani si 7 luni',
    NUMTOYMINTERVAL (4, 'YEAR') +
          NUMTOYMINTERVAL (7, 'MONTH')
FROM dual UNION
SELECT '4 ani fara 7 luni',
    NUMTOYMINTERVAL (4, 'YEAR') -
          NUMTOYMINTERVAL (7, 'MONTH')
FROM dual
```

După constanta numerică apare 'YEAR' sau 'MONTH', ceea ce indică semnificația numărului (este an sau lună). Utilizatea acestei funcții este dată de operațiunile în care poate fi angajată, dintre care câteva sunt exemplificate în interogarea precedentă, al cărei rezultat se prezintă în figura 6.26.

```
DESCRIERE          Interval
------------------ --------------
4 ani              +000000004-00
4 ani fara 7 luni  +000000003-05
4 ani si 7 luni    +000000004-07
7 luni             +000000000-07
```

Figura 6.26. Funcția NUMTOYMINTERVAL

Pe acelaşi calapod este construită şi logica funcţiei NUMTODSINTERVAL, numai că în acest caz o constantă numerică poate fi convertită, după caz, în zile (DAY), ore (HOUR), minute (MINUTE) sau secunde (SECOND) – vezi figura 6.27:

```
SELECT '3 zile cu 7 minute si 5 secunde' AS Descriere,
    NUMTODSINTERVAL (3, 'DAY') +
    NUMTODSINTERVAL (7, 'MINUTE') +
    NUMTODSINTERVAL (5, 'SECOND') AS Interval
FROM dual UNION
SELECT '3 zile fara 7 minute',
    NUMTODSINTERVAL (3, 'DAY') -
    NUMTODSINTERVAL (7, 'MINUTE')
FROM dual
```

```
DESCRIERE                      INTERVAL
------------------------------ ------------------------------
3 zile cu 7 minute si 5 secunde +000000003 00:07:05.000000000
3 zile fara 7 minute           +000000002 23:53:00.000000000
```

Figura 6.27. Funcția NUMTODSINTERVAL

Interogarea următoare foloseşte cele două funcţii pentru a calcula datele şi orele exacte de peste *4 ani fără 7 luni* şi peste *3 zile fără 7 minute,* relativ la momentul execuției:

```
SELECT 'Astazi' AS Obiectiv,
    TO_CHAR(SYSDATE, 'DD-MM-YYYY HH:MI:SS') AS Data_Ora
FROM dual
UNION
SELECT 'Peste 4 ani fara 7 luni ',
    TO_CHAR(SYSDATE + NUMTOYMINTERVAL (4, 'YEAR') -
    NUMTOYMINTERVAL (7, 'MONTH') , 'DD-MM-YYYY HH:MI:SS')
FROM dual
UNION
SELECT 'Peste 3 zile fara 7 minute',
    TO_CHAR(SYSDATE + NUMTODSINTERVAL (3, 'DAY') -
    NUMTODSINTERVAL (7, 'MINUTE') , 'DD-MM-YYYY HH:MI:SS')
FROM dual
```

Ceea ce se obține în urma lansării în SQL*Plus seamănă izbitor cu figura 6.28.

```
OBIECTIV                   DATA_ORA
-------------------------- -------------------
Astazi                     20-05-2003 11:11:12
Peste 3 zile fara 7 minute 23-05-2003 11:04:12
Peste 4 ani fara 7 luni    20-10-2006 11:11:12
```

Figura 6.28. Expresii ce folosesc funcții de conversie număr-interval calendaristic

În comunitatea SQL funcția consacrată în privința conversiilor este CAST. Dintre multiplele posibilități, următoarea frază SELECT ilustrează următoarele

conversii: din număr în şir de caractere, din tip dată calendaristică în tip timp şi din dată calendaristică în şir de caractere:

```
SELECT CAST (marca AS CHAR(6)) AS Marca_Sir,
    CAST (CURRENT_DATE AS TIMESTAMP) AS Data_Timp,
    CAST (datasv AS VARCHAR2(40)) AS Data_Sir
FROM personal
```

Tipologia datelor rezultatelor în urma conversiei (figura 6.29) reiese şi din modul de aliniere. Astfel, coloana `Marca_Sir` este aliniată la stângă, ceea ce indică faptul că tipul său este şir de caractere.

MARCA_SIR	DATA_TIMP	DATA_SIR
109	20-MAY-03 04.30.47.000000 PM	10-JUL-03
110	20-MAY-03 04.30.47.000000 PM	10-JUL-03
101	20-MAY-03 04.30.47.000000 PM	12-OCT-80
102	20-MAY-03 04.30.47.000000 PM	12-NOV-78
103	20-MAY-03 04.30.47.000000 PM	02-JUL-76
104	20-MAY-03 04.30.47.000000 PM	05-JAN-82
105	20-MAY-03 04.30.47.000000 PM	12-NOV-77
106	20-MAY-03 04.30.47.000000 PM	11-APR-85
107	20-MAY-03 04.30.47.000000 PM	21-NOV-91
108	20-MAY-03 04.30.47.000000 PM	30-DEC-94

Figura 6.29. Folosirea funcției CAST

Adevărata forță a funcției `CAST` iese la iveală atunci când se dorește coversia dintr-un tip utilizator în altul, cu sau fără subconsultări care extrag seturi de înregistrări. Despre aceste facilități vom discuta cu altă ocazie.

6.2.5. Structuri alternative: DECODE şi CASE

În Oracle există două opțiuni pentru lucrul cu secvențe alternative: duncția `DECODE`, disponibilă încă din Oracle 7 şi structura `CASE` care e ceva mai "caldă" (din Oracle 8i şi 9i în PL/SQL). Tabela PERSONAL conține atributul `Colaborator` pentru a şti dacă persoana respectivă este angajată cu carte de muncă, caz în care valoarea atributului este 'N', sau este colaborator al fimei ('D'). În afara celor două valori, la un moment dat, valoarea poate să fie NULL, ceea ce înseamnă fie că situația angajatului este incertă, fie că, pur şi simplu, este vorba de o neglijența a operatorilor. Ne propunem să afişăm, în dreptul fiecărei linii din tabelă, un mesaj care să indice tipul persoanei respective. Folosind funcția `DECODE` se poate scrie:

```
SELECT marca, numepren, colaborator,
    DECODE (colaborator ,
    'N', 'Angajat cu norma intreaga',
    'D', 'Colaborator',
```

```
      'Nu este specificat !!!')  AS Tipul_Angajatului
  FROM personal
```

Logica este următoarea: primul argument al funcție, `colaborator`, este comparat cu prima valoare de referință, `'N'`; dacă cele două valori sunt egale, funcția returnează şirul `'Angajat cu norma intreaga'`; în caz că nu, se trece la verificarea egalității dintre `colaborator` şi a doua valoare etalon – `'D'`, iar atunci când cele două valori coincid se returnează `'Colaborator'`. Dacă valoarea atributului nu se potriveşte cu nici una dintre valorile specidicare, se returnează ultimul şir `'Nu este specificat !!!'`. Iată rezultatul - figura 6.30.

MARCA	NUMEPREN	COLABORATOR	TIPUL_ANGAJATULUI
109	Primul Angajat Nou	D	Colaborator
110	Al Doilea Angajat Nou		Nu este specificat !!!
101	Angajat 1	N	Angajat cu norma intreaga
102	Angajat 2	N	Angajat cu norma intreaga
103	Angajat 3	N	Angajat cu norma intreaga
104	Angajat 4	N	Angajat cu norma intreaga
105	Angajat 5	N	Angajat cu norma intreaga
106	Angajat 6	N	Angajat cu norma intreaga
107	Angajat 7	N	Angajat cu norma intreaga
108	Angajat 8	N	Angajat cu norma intreaga

Figura 6.30. DECODE şi CASE

Din Oracle 8i apare, pe linia alinierii cu SQL-92, structura `CASE`, aşa că, echivalent frazei de mai sus putem folosi şi alte două soluții:

```
  SELECT marca, numepren, colaborator,
      CASE colaborator
            WHEN 'N' THEN 'Angajat cu norma intreaga'
            WHEN 'D' THEN 'Colaborator'
            ELSE 'Nu este specificat !!!'
      END  AS Tipul_Angajatului
  FROM personal
```

sau

```
  SELECT marca, numepren, colaborator,
      CASE
      WHEN colaborator = 'N' THEN 'Angajat cu norma intreaga'
      WHEN colaborator = 'D' THEN 'Colaborator'
      ELSE 'Nu este specificat !!!'
      END  AS Tipul_Angajatului
  FROM personal
```

Prima este mai apropiată de logica funcției `DECODE` şi permite specificarea a o serie de condiții, relativ la un singur parametru sau expresie. A doua este mai flexibilă, fiecare ramură `WHEN` având propria condiție de evaluat. În oricare dintre variante, importantă este ordinea de enumerate a condiției/valorii-etalon,

deoarece odată identificată o egalitate, execuţia este direcţionată numai pe ramura respectivă, chiar dacă ar urma şi alte condiţii ce ar fi îndeplinite. Pe de altă parte, dacă lipseşte ramura ELSE (echivalenta OTHERWISE) şi există valori care nu se încadrează în nici una dintre condiţiile enumerate, valoarea returnată este NULL – vezi figura 6.31.

```
SQL> SELECT marca, numepren, colaborator,
  2  CASE
  3  WHEN colaborator = 'N' THEN 'Angajat cu norma intreaga'
  4  WHEN colaborator = 'D' THEN 'Colaborator'
  5  END  AS Tipul_Angajatului
  6  FROM personal
  7  /

    MARCA NUMEPREN                COLABORATOR TIPUL_ANGAJATULUI
--------- ---------------------- ----------- ------------------------
      109 Primul Angajat Nou     D           Colaborator
      110 Al Doilea Angajat Nou
      101 Angajat 1              N           Angajat cu norma intreaga
      102 Angajat 2              N           Angajat cu norma intreaga
      103 Angajat 3              N           Angajat cu norma intreaga
      104 Angajat 4              N           Angajat cu norma intreaga
      105 Angajat 5              N           Angajat cu norma intreaga
      106 Angajat 6              N           Angajat cu norma intreaga
      107 Angajat 7              N           Angajat cu norma intreaga
      108 Angajat 8              N           Angajat cu norma intreaga
```

Figura 6.31. Condiţie “netratată” într-o structură CASE

Luând în discuţie tabela PONTAJE, dorim ca în dreptul fiecărei înregistrări corespunzătoare lunii iulie 2003 să apară “DUMINICĂ”, “sâmbătă” sau “Zi lucrătoare”. Soluţia bazată pe CASE este:

```
SELECT pontaje.*,
    CASE  TO_CHAR(data, 'DAY')
    WHEN 'SUNDAY' THEN 'DUMINICA'
    WHEN 'SATURDAY' THEN 'simbata'
    ELSE 'Zi lucratoare'
    END  AS Categoria_Zilei
FROM pontaje
WHERE TO_CHAR(data, 'MM/YYYY') = '07/2003'
```

iar cea bazată pe DECODE:

```
SELECT pontaje.*,
DECODE (TO_CHAR(data, 'DAY'),
          'SUNDAY', 'DUMINICA',
          'SATURDAY', 'simbata',
          'Zi lucratoare' )AS Categoria_Zilei
FROM pontaje
WHERE TO_CHAR(data, 'MM/YYYY') = '07/2003'
```

De obicei, veteranii Oracle manifestă un ataşament (attachment) deosebit faţă de DECODE. La fel şi "bobocii" Oracle care vor să pară veterani. Când valorile etalon sunt punctuale, atunci poate că e de preferat DECODE. Situațiile complexe (intervale, spre exemplu) sunt cele în care CASE-ul îşi arată forţa. Spre exemplu, fiecărui angajat vrem să-i afişăm un mesaj care să indice intervalul în care se situează numărul anilor săi de vechime în muncă: sub 10 ani, între 10 şi 20 de ani şi peste 20 de ani. Cu o structură CASE lucrurile nu sunt prea complicate:

```
SELECT numepren, datasv,
EXTRACT (YEAR FROM ((DATE'2003-05-01' - datasv)
YEAR(2) TO MONTH)) AS Ani,
    CASE
    WHEN EXTRACT (YEAR FROM ((DATE'2003-05-01' - datasv)
                YEAR(2) TO MONTH)) < 10
          THEN 'Sub 10 ani de vechime'
    WHEN EXTRACT (YEAR FROM ((DATE'2003-05-01' - datasv)
                YEAR(2) TO MONTH)) BETWEEN 10 AND 20
          THEN 'Vechime intre 10 si 20 de ani'
    ELSE 'Peste 20 de ani de vechime'
    END AS Situatie_vechime
FROM personal
```

În schimb, deoarece DECODE lucrează pe bază de egalitate de valori, vom apela la un truc: convertim, prin TO_CHAR, numărul de ani într-un şir de caractere, apoi testam primul caracter din ani; în cazul în care anii sunt mai puțini de 10, vom avea situații de genul " 1", " 7" etc., deci primul caracter este spațiu; dacă primul caracter este "1", înseamnă că numărul este cuprins între 10 şi 19:

```
SELECT numepren, datasv,
    EXTRACT (YEAR FROM ((DATE'2003-05-01' - datasv)
    YEAR(2) TO MONTH)) AS Ani,
    DECODE (
    SUBSTR(TO_CHAR(EXTRACT (YEAR FROM
          ((DATE'2003-05-01' - datasv)
          YEAR(2) TO MONTH)),'99'), 2,1),
          ' ', 'Sub 10 ani de vechime',
          '1', 'Vechime intre 10 si 20 de ani',
          'Peste 20 de ani de vechime')
          AS Situatie_vechime
FROM personal
```

Rezultatul oricărei din cele două variante este reprezentat în figura 6.32.

NUMEPREN	DATASV	ANI	SITUATIE_VECHIME
Primul Angajat Nou	10-JUL-03	0	Sub 10 ani de vechime
Al Doilea Angajat Nou	10-JUL-03	0	Sub 10 ani de vechime
Angajat 1	12-OCT-80	22	Peste 20 de ani de vechime
Angajat 2	12-NOV-78	24	Peste 20 de ani de vechime
Angajat 3	02-JUL-76	26	Peste 20 de ani de vechime
Angajat 4	05-JAN-82	21	Peste 20 de ani de vechime
Angajat 5	12-NOV-77	25	Peste 20 de ani de vechime
Angajat 6	11-APR-85	18	Vechime intre 10 si 20 de ani
Angajat 7	21-NOV-91	11	Vechime intre 10 si 20 de ani
Angajat 8	30-DEC-94	8	Sub 10 ani de vechime

Figura 6.32. CASE/DECODE pe intervale

6.3. Joncțiuni interne

Întrucât ținta prezentei lucrări este una covârşitor practică, nu cheltuim prea mult spațiu cu fundamentarea teroretică a joncțiunilor interne, ca o combinație dintre produs cartezian şi selecție, şi nici cu detalii legate de theta-joncțiune, echi-joncțiune şi joncțiune naturală. Vom lucra, pentru început, la modul general, cu joncțiuni pe care le vom aplica pentru rezolvarea câtorva probleme.

Care sunt zilele din iulie 2003 în care Angajat 1 a fost la lucru ?
Informația care ne interesează, adică valorile atributului `data`, precum şi condiția "prezență la lucru" – `OreLucrate > 0`, se află în tabela PONTAJE. În schimb, numele celebrului angajat se află în tabela PERSONAL. Aşa că apelăm la joncțiune. Până la versiunea 9i, Oracle s-a încăpățânat să folosească exclusiv notația SQL-89:

```
SELECT data
FROM pontaje, personal
WHERE pontaje.marca = personal.marca AND
    EXTRACT(YEAR FROM data) = 2003 AND
    EXTRACT (MONTH FROM data) = 7 AND
    numepren = 'Angajat 1'
```

Din 9i, Oracle a revenit la sentimente mai bune față de SQL-92, aşa încât notația este mai elegantă:

```
SELECT data
FROM pontaje INNER JOIN personal
    ON pontaje.marca = personal.marca
WHERE EXTRACT(YEAR FROM data) = 2003 AND
    EXTRACT (MONTH FROM data) = 7 AND
    numepren = 'Angajat 1'
```

Ba, mai mult, când două tabele sunt joncționate după atribute cu nume identic, se poate apela cu succes la operatorul `NATURAL JOIN`. Astfel, joncționarea

tabelelor SPORURI şi SALARII, realizată prin intermediul celor trei atribute, `Marca`, `An` şi `Luna`, se poate realiza prin trei variante:

- tip SQL1:

```
SELECT *
FROM sporuri, salarii
WHERE sporuri.marca = salarii.marca AND
    sporuri.an = salarii.an AND sporuri.luna = salarii.luna
```

- cu `INNER JOIN`:

```
SELECT *
FROM sporuri INNER JOIN salarii
    ON sporuri.marca = salarii.marca AND
    sporuri.an = salarii.an AND sporuri.luna = salarii.luna
```

- cu `NATURAL JOIN`:

```
SELECT *
FROM sporuri NATURAL JOIN salarii
```

Care sunt colegii de compartiment ai lui Angajat 2 ?

Problema reclamă un artificiu SQL: joncționarea a două instanțe ale tabelei PERSONAL după atributul `compart` şi filtrarea liniilor pentru una din instanțe (`numepren = 'Angajat 2'`). Operațiunea este posibilă folosind două alias-uri pentru cele două instanțe.

```
SELECT p1.numepren
FROM personal p1, personal p2
WHERE p1.compart = p2.compart AND p2.numepren = 'Angajat 2'
```

sau

```
SELECT p1.numepren
FROM personal p1 INNER JOIN personal p2
    ON p1.compart = p2.compart
WHERE p2.numepren = 'Angajat 2'
```

sau

```
SELECT p1.numepren
FROM personal p1 INNER JOIN personal p2
    ON p1.compart = p2.compart AND
    p2.numepren = 'Angajat 2'
```

Dincolo de notația diferită, cele trei soluții au acelaşi principiu de funcționare. În prima fază se joncționează instanța P1 a tabelei PERSONAL cu instanța P2 a aceleaşi tabele, atributul de joncționare fiind `compart`. Rezultatul este cel din figura 6.33. SQL*Plus este un pic derutatant, deoarece în figură nu se poate face distincție între atributele din P1 şi cele din P2. Intuiția brucaniană ne spune că primele şapte atribute sunt ale P1, iar celelalte şapte ale P2.

MARCA	NUMEPREN	COMPA	DATASV	SALORAR	SALORARCO	C	MARCA	NUMEPREN	COMPA	DATASV	SALORAR	SALORARCO	C
102	Angajat 2	CONTA	12-NOV-78	57500	56000	N	102	Angajat 2	CONTA	12-NOV-78	57500	56000	N
106	Angajat 6	CONTA	11-APR-85	71500	70000	N	102	Angajat 2	CONTA	12-NOV-78	57500	56000	N
107	Angajat 7	CONTA	21-NOV-91	61500	60000	N	102	Angajat 2	CONTA	12-NOV-78	57500	56000	N
102	Angajat 2	CONTA	12-NOV-78	57500	56000	N	106	Angajat 6	CONTA	11-APR-85	71500	70000	N
106	Angajat 6	CONTA	11-APR-85	71500	70000	N	106	Angajat 6	CONTA	11-APR-85	71500	70000	N
107	Angajat 7	CONTA	21-NOV-91	61500	60000	N	106	Angajat 6	CONTA	11-APR-85	71500	70000	N
102	Angajat 2	CONTA	12-NOV-78	57500	56000	N	107	Angajat 7	CONTA	21-NOV-91	61500	60000	N
106	Angajat 6	CONTA	11-APR-85	71500	70000	N	107	Angajat 7	CONTA	21-NOV-91	61500	60000	N
107	Angajat 7	CONTA	21-NOV-91	61500	60000	N	107	Angajat 7	CONTA	21-NOV-91	61500	60000	N
109	Primul Angajat Nou	IT	10-JUL-03	55500	0	D	109	Primul Angajat Nou	IT	10-JUL-03	55500	0	D
101	Angajat 1	IT	12-OCT-80	56000	55000	N	109	Primul Angajat Nou	IT	10-JUL-03	55500	0	D
103	Angajat 3	IT	02-JUL-76	67500	66000	N	109	Primul Angajat Nou	IT	10-JUL-03	55500	0	D
105	Angajat 5	IT	12-NOV-77	62500	62000	N	109	Primul Angajat Nou	IT	10-JUL-03	55500	0	D
109	Primul Angajat Nou	IT	10-JUL-03	55500	0	D	101	Angajat 1	IT	12-OCT-80	56000	55000	N
101	Angajat 1	IT	12-OCT-80	56000	55000	N	101	Angajat 1	IT	12-OCT-80	56000	55000	N
103	Angajat 3	IT	02-JUL-76	67500	66000	N	101	Angajat 1	IT	12-OCT-80	56000	55000	N
105	Angajat 5	IT	12-NOV-77	62500	62000	N	101	Angajat 1	IT	12-OCT-80	56000	55000	N
109	Primul Angajat Nou	IT	10-JUL-03	55500	0	D	103	Angajat 3	IT	02-JUL-76	67500	66000	N
101	Angajat 1	IT	12-OCT-80	56000	55000	N	103	Angajat 3	IT	02-JUL-76	67500	66000	N
103	Angajat 3	IT	02-JUL-76	67500	66000	N	103	Angajat 3	IT	02-JUL-76	67500	66000	N
105	Angajat 5	IT	12-NOV-77	62500	62000	N	103	Angajat 3	IT	02-JUL-76	67500	66000	N
109	Primul Angajat Nou	IT	10-JUL-03	55500	0	D	105	Angajat 5	IT	12-NOV-77	62500	62000	N
101	Angajat 1	IT	12-OCT-80	56000	55000	N	105	Angajat 5	IT	12-NOV-77	62500	62000	N
103	Angajat 3	IT	02-JUL-76	67500	66000	N	105	Angajat 5	IT	12-NOV-77	62500	62000	N
105	Angajat 5	IT	12-NOV-77	62500	62000	N	105	Angajat 5	IT	12-NOV-77	62500	62000	N
110	Al Doilea Angajat Nou	PROD	10-JUL-03	50500	0		110	Al Doilea Angajat Nou	PROD	10-JUL-03	50500	0	
108	Angajat 8	PROD	30-DEC-94	54500	52000	N	110	Al Doilea Angajat Nou	PROD	10-JUL-03	50500	0	
104	Angajat 4	PROD	05-JAN-82	75000	75000	N	110	Al Doilea Angajat Nou	PROD	10-JUL-03	50500	0	
110	Al Doilea Angajat Nou	PROD	10-JUL-03	50500	0		108	Angajat 8	PROD	30-DEC-94	54500	52000	N
108	Angajat 8	PROD	30-DEC-94	54500	52000	N	108	Angajat 8	PROD	30-DEC-94	54500	52000	N
104	Angajat 4	PROD	05-JAN-82	75000	75000	N	108	Angajat 8	PROD	30-DEC-94	54500	52000	N
110	Al Doilea Angajat Nou	PROD	10-JUL-03	50500	0		104	Angajat 4	PROD	05-JAN-82	75000	75000	N
108	Angajat 8	PROD	30-DEC-94	54500	52000	N	104	Angajat 4	PROD	05-JAN-82	75000	75000	N
104	Angajat 4	PROD	05-JAN-82	75000	75000	N	104	Angajat 4	PROD	05-JAN-82	75000	75000	N

Figura 6.33. Joncțiunea a două instanțe ale tabelei PERSONAL prin atributul `compart`

Rezultatul final al interogării este cel din figura 6.34. Pentru eliminarea Angajatului 2 din rezultat ar trebui ca în `WHERE` să fie inclusă şi condiția `p1.numepren <> 'Angajat 2'`.

```
SQL> SELECT p1.numepren
  2  FROM personal p1 INNER JOIN personal p2
  3  ON p1.compart = p2.compart AND p2.numepren = 'Angajat 2'
  4  /

NUMEPREN
----------------------
Angajat 2
Angajat 6
Angajat 7
```

Figura 6.34. Colegii de compartiment al lui Angajat 2 (plus el-însuşi)

Care sunt zilele în care au lucrat simultan Angajat 1 şi Primul Angajat Nou ?

Deşi ne este destul de greu, să presupunem că nu ştim nimic de existența operatorului `INTERSECT` pe care-l vom pomeni în paragraful viitor. O soluție dragă Fox-iştilor (în Visual FoxPro nu există operatorul `INTERSECT`) constă în joncționarea a două... joncțiuni PONTAJE-PERSONAL:

```
SELECT po1.data AS Ziua
FROM (pontaje po1 INNER JOIN personal pe1 ON
    po1.marca = pe1.marca AND numepren = 'Angajat 1')
INNER JOIN
    (pontaje po2 INNER JOIN personal pe2 ON
      po2.marca = pe2.marca AND
      pe2.numepren = 'Primul Angajat Nou')
    ON po1.data = po2.data
```

Rezultatul este cel din figura 6.35.

```
  1  SELECT po1.data AS Ziua
  2  FROM (pontaje po1 INNER JOIN personal pe1 ON
  3  po1.marca = pe1.marca AND numepren = 'Angajat 1')
  4  INNER JOIN
  5  (pontaje po2 INNER JOIN personal pe2 ON
  6  po2.marca = pe2.marca AND pe2.numepren = 'Primul Angajat Nou')
  7*    ON po1.data = po2.data
SQL> /

ZIUA
---------
09-JUL-03
10-JUL-03
11-JUL-03
```

Figura 6.35. Intersecție prin joncțiune

6.4. Reuniune, intersecție, diferență

Discuția din acest paragraf este, în mare parte, de prisos, deoarece, până în acest paragraf, am recurs de câteva ori şi la reuniune şi la intersecție.

Care sunt angajații compartimentelor Contabilitate şi IT ?
La cele două soluții formulate în primul paragraf al capitolului (operatorii SAU logic şi `IN`), putem adăuga una ce utilizează `UNION`:

```
SELECT * FROM personal WHERE compart = 'CONTA'
UNION
SELECT * FROM personal WHERE compart = 'IT'
```

Care sunt zilele din iulie 2003 în care a lucrat măcar unul dintre oamenii muncii (de la oraşe şi sate) Angajat 1 şi Primul Angajat Nou ?
Dacă se foloseşte operatorul `UNION`, rezultatul ar conține nouă linii, corespunzător celor nouă zile în care măcar unul dintre cei doi a venit la serviciu:

```
SELECT Data
FROM pontaje po INNER JOIN personal pe
    ON po.marca=pe.marca
WHERE numepren = 'Angajat 1' AND
```

```
    TO_CHAR(data, 'MM/YYYY')='07/2003'
UNION
SELECT Data
FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
WHERE numepren = 'Primul Angajat Nou'
    AND TO_CHAR(data, 'MM/YYYY')='07/2003'
```

În schimb, operatorul `UNION ALL` extrage 12 linii, neeliminând duplicatele, astfel încât din analiza figurii 6.36 putem deduce că pe 9, 10 şi 11 iulie ambii angajaţi au fost la lucru, în timp ce în restul zilelor numai Angajat 1 a fost prezent.

```
SELECT Data
FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
WHERE numepren = 'Angajat 1' AND
    TO_CHAR(data, 'MM/YYYY')='07/2003'
UNION ALL
SELECT Data
FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
WHERE numepren = 'Primul Angajat Nou' AND
    TO_CHAR(data, 'MM/YYYY')='07/2003'
```

```
SQL> SELECT Data
  2  FROM pontaje po INNER JOIN personal pe
  3  ON po.marca=pe.marca
  4  WHERE numepren = 'Angajat 1' AND
  5  TO_CHAR(data, 'MM/YYYY')='07/2003'
  6  UNION
  7  SELECT Data
  8  FROM pontaje po INNER JOIN personal pe
  9  ON po.marca=pe.marca
 10  WHERE numepren = 'Primul Angajat Nou' AND
 11  TO_CHAR(data, 'MM/YYYY')='07/2003'
 12  /

DATA
---------
01-JUL-03
02-JUL-03
03-JUL-03
04-JUL-03
07-JUL-03
08-JUL-03
09-JUL-03
10-JUL-03
11-JUL-03

9 rows selected.
```

```
SQL> SELECT Data
  2  FROM pontaje po INNER JOIN personal pe
  3  ON po.marca=pe.marca
  4  WHERE numepren = 'Angajat 1'
  5  AND TO_CHAR(data, 'MM/YYYY')='07/2003'
  6  UNION ALL
  7  SELECT Data
  8  FROM pontaje po INNER JOIN personal pe
  9  ON po.marca=pe.marca
 10  WHERE numepren = 'Primul Angajat Nou'
 11  AND TO_CHAR(data, 'MM/YYYY')='07/2003'
 12  /

DATA
---------
01-JUL-03
02-JUL-03
03-JUL-03
04-JUL-03
07-JUL-03
08-JUL-03
09-JUL-03
10-JUL-03
11-JUL-03
09-JUL-03
10-JUL-03
11-JUL-03

12 rows selected.
```

Figura 6.36. Diferența dintre UNION şi UNION ALL

Efectul `UNION ALL` este similar folosirii operatorului `OR` sau soluției bazate pe `IN`:

```
SELECT Data
FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
WHERE numepren = ('Angajat 1' OR
    numepren='Primul Angajat Nou') AND
```

```
    TO_CHAR(data, 'MM/YYYY')='07/2003'
```

Care sunt zilele în care au lucrat simultan Angajat 1 şi Primul Angajat Nou ?
Acum putem recurge la operatorul `INTERSECT`:

```
SELECT data AS Ziua
FROM pontaje po INNER JOIN personal pe ON
    po.marca = pe.marca AND numepren = 'Angajat 1'
INTERSECT
SELECT data
FROM pontaje po INNER JOIN personal pe ON
    po.marca = pe.marca AND numepren = 'Primul Angajat Nou'
```

Care sunt zilele din iulie 2003 în care a lucrat Angajat 1, dar NU a lucrat Primul Angajat Nou ?
Soluția se bazează pe diferență, însă, spre deosebire de standardul SQL în care operatorul folosit este `EXCEPT`, în Oracle se întrebuințează `MINUS`:

```
SELECT data AS Ziua
FROM pontaje po INNER JOIN personal pe ON
    po.marca = pe.marca AND
    numepren = 'Angajat 1' AND
    TO_CHAR(data, 'MM/YYYY')='07/2003'
MINUS
SELECT data
FROM pontaje po INNER JOIN personal pe
    ON po.marca = pe.marca
    AND numepren = 'Primul Angajat Nou'
```

6.5. Joncțiuni externe, valori nule

Joncțiunea externă are o largă utilizare datorită posibilității extragerii, din cele două tabele joncționate, atât a liniilor corespondente, cât şi a celor fără corespondent. Standardul SQL-92 introduce operatorii necesari joncțiunii externe:

- `LEFT OUTER JOIN` pentru joncțiune externă la stânga,
- `RIGHT OUTER JOIN` pentru joncțiune externă la dreapta,
- `FULL OUTER JOIN` pentru joncțiune externă totală (în ambele direcții).

Pentru luna iulie 2003 toți cei zece angajați prezintă pontaje; în schimb, în august, doi dintre ei, cei cu mărcile 109 şi 110, sunt totalmente absenți. Beneficiind de suportul comenzii `MERGE` discutată în ultima parte a capitolului precedent, actualizăm tabela SALARII pentru august 2003:

```
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) = 8
 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)
```

Care sunt orele lucrate şi de concediu pentru fiecare angajat pe luna iulie 2003 ?
Folosind joncţiunea, se obţine rezultatul din figura 6.37 în care sunt excluşi angajaţii 109 şi 110, deoarece liniile lor din PERSONAL nu au corespondent în PONTAJE.

```
SELECT p.marca, numepren, compart, orelucrate
    AS Ore_Lucr_Aug, oreco AS Ore_CO_Aug
FROM personal p  INNER JOIN salarii s ON p.marca = s.marca
    AND an=2003 and luna=8
```

MARCA	NUMEPREN	COMPA	ORE_LUCR_AUG	ORE_CO_AUG
101	Angajat 1	IT	32	0
102	Angajat 2	CONTA	16	16
103	Angajat 3	IT	24	8
104	Angajat 4	PROD	32	0
105	Angajat 5	IT	32	0
106	Angajat 6	CONTA	32	0
107	Angajat 7	CONTA	16	16
108	Angajat 8	PROD	32	0

Figura 6.37. Lista incompletă, datorată joncţiunii interne

Rezolvarea problemei vine de la operatorul joncţiune externă la stânga, care, pe lângă liniile joncţiunii interne, include în rezultat şi înregistrările tabelei din stânga (PERSONAL) în care valorilor atributului `Marca` nu se regăsesc în SALARII pentru anul 2003, luna 8:

```
SELECT p.marca, numepren, compart, orelucrate
    AS Ore_Lucr_Aug, oreco AS Ore_CO_Aug
FROM personal p  LEFT OUTER JOIN salarii s
    ON p.marca = s.marca AND an=2003 and luna=8
```

Ultimele două rânduri din figura 6.38 corespund celor doi angajați fără nici un pontaj pe august 2003, lucru indicat de spațiile ce apar în dreptul coloanelor referitoare la orele lucrate şi orele de concediu. În SQL*Plus, spațiile respective semnifică, pentru acest gen de situații, valori NULL.

MARCA	NUMEPREN	COMPA	ORE_LUCR_AUG	ORE_CO_AUG
101	Angajat 1	IT	32	0
102	Angajat 2	CONTA	16	16
103	Angajat 3	IT	24	8
104	Angajat 4	PROD	32	0
105	Angajat 5	IT	32	0
106	Angajat 6	CONTA	32	0
107	Angajat 7	CONTA	16	16
108	Angajat 8	PROD	32	0
109	Primul Angajat Nou	IT		
110	Al Doilea Angajat Nou	PROD		

Figura 6.38. Joncțiune externă la stânga PERSONAL-SALARII

Interesant este că, până la versiunea 9i, Oracle nu a avut implementate cele trei clauze dedicate joncțiunii externe, în acest scop folosindu-se semnul `+`. Astfel joncțiunea externă la stânga se realiza de maniera următoare:

```
SELECT p.marca, numepren, compart, orelucrate
    AS Ore_Lucr_Aug, oreco AS Ore_CO_Aug
FROM personal p, salarii s
WHERE p.marca = s.marca  (+) AND 2003=an (+) and 8=luna (+)
```

Semnul `(+)` se plasează la atributul din dreapta semnului egal, atribut care trebuie să fie, obligatoriu, din tabela în care nu se găsesc înregistrări corespondente valorilor tabelei din stânga.

Care sunt angajații fără nici un pontaj pe august 2003 ?
Până la a ajunge la subconsultări, putem valorifica faptul că liniile fără corespondent prezintă în rezultatul unei joncțiuni externe valoarea NULL. Aşa că interogarea următoare funcționează:

```
SELECT p.marca, numepren
FROM personal p, salarii s
WHERE p.marca = s.marca  (+) AND 2003=an (+) and 8=luna (+)
    AND s.marca IS NULL
```

Transformând fraza `SELECT` pe logica joncțiunii externe la dreapta obținem:

```
SELECT p.marca, numepren
FROM salarii s, personal p
WHERE s.marca (+) = p.marca  AND an(+) = 2003 AND
    luna (+) = 8 AND s.marca IS NULL
```

sau, după noul format conform cu SQL-92:

```
SELECT p.marca, numepren
FROM salarii s RIGHT OUTER JOIN personal p
    ON s.marca = p.marca  AND an = 2003 AND luna = 8
WHERE s.marca IS NULL
```

Fireşte, aceeaşi informaţie putea fi obţinută din PONTAJE astfel:

```
SELECT pe.marca, numepren
FROM personal pe LEFT OUTER JOIN pontaje po
    ON pe.marca = po.marca  AND
TO_CHAR(data, 'MM/YYYY') = '08/2003'
WHERE po.marca IS NULL
```

Care sunt orele lucrate de fiecare angajat în lunile iulie şi august 2003, atât pe luni, cât şi însumat ?

Se joncţionează extern tabela PERSONAL cu două instanţe ale tabelei SALARII, una pentru luna iulie şi o alta pentru august:

```
SELECT p.marca, numepren, s1.orelucrate AS Ore_Lucr_Iul,
    s2.orelucrate AS Ore_Lucr_aug,
    s1.orelucrate + s2.orelucrate AS "Ore_Lucr_Iul-Aug"
FROM personal p
LEFT OUTER JOIN salarii s1
    ON p.marca = s1.marca  AND s1.an=2003 AND s1.luna=7
LEFT OUTER JOIN salarii s2
    ON p.marca = s2.marca  AND s2.an=2003 AND s2.luna=8
```

Din păcate, totalul se calculează numai pentru cei care au lucrat în ambele luni (figura 6.39), deoarece o valoare NULL care apare într-o expresie atrage după sine NULL-itatea expresiei, indiferent de valorile celorlalţi termeni.

MARCA	NUMEPREN	ORE_LUCR_IUL	ORE_LUCR_AUG	Ore_Lucr_Iul-Aug
101	Angajat 1	72	32	104
102	Angajat 2	40	16	56
103	Angajat 3	40	24	64
104	Angajat 4	72	32	104
105	Angajat 5	72	32	104
106	Angajat 6	72	32	104
107	Angajat 7	40	16	56
108	Angajat 8	72	32	104
109	Primul Angajat Nou	24		
110	Al Doilea Angajat Nou	24		

Figura 6.39. Propagarea valorii NULL

În asemenea situaţii Oracle pune la dispoziţie funcţia `NVL` care transformă o eventuală valoare nulă într-o alta specificată:

```
SELECT p.marca, numepren,
    NVL(s1.orelucrate,0) AS Ore_Lucr_Iul,
    NVL(s2.orelucrate,0) AS Ore_Lucr_aug,
```

```
    NVL(s1.orelucrate,0) + NVL(s2.orelucrate,0)
          AS "Ore_Lucr_Iul-Aug"
FROM personal p
    LEFT OUTER JOIN salarii s1
          ON p.marca = s1.marca  AND s1.an=2003
                AND s1.luna=7
    LEFT OUTER JOIN salarii s2
          ON p.marca = s2.marca  AND s2.an=2003
                AND s2.luna=8
```

Cum este şi normal, pentru însumarea orelor pe iulie şi august, valorile nule vor fi substituite prin zero, ceea ce conduce la un calcul corect – vezi figura 6.40.

MARCA	NUMEPREN	ORE_LUCR_IUL	ORE_LUCR_AUG	Ore_Lucr_Iul-Aug
101	Angajat 1	72	32	104
102	Angajat 2	40	16	56
103	Angajat 3	40	24	64
104	Angajat 4	72	32	104
105	Angajat 5	72	32	104
106	Angajat 6	72	32	104
107	Angajat 7	40	16	56
108	Angajat 8	72	32	104
109	Primul Angajat Nou	24	0	24
110	Al Doilea Angajat Nou	24	0	24

Figura 6.40. Substituirea valorilor NULL

Trebuie să recunoaştem, fără a cunoaşte funcția `NVL`, problema este rezolvabilă. Cel mai la îndemână e o structură `CASE`:

```
SELECT p.marca, numepren,
    CASE WHEN s1.orelucrate IS NOT NULL
          THEN s1.orelucrate ELSE 0 END AS Ore_Lucr_Iul,
    CASE WHEN s2.orelucrate IS NOT NULL
          THEN s2.orelucrate ELSE 0 END AS Ore_Lucr_aug,
    CASE WHEN s1.orelucrate IS NOT NULL
          THEN s1.orelucrate ELSE 0 END +
                CASE WHEN s2.orelucrate IS NOT NULL
                THEN s2.orelucrate ELSE 0
                END
    AS "Ore_Lucr_Iul-Aug"
FROM personal p
LEFT OUTER JOIN salarii s1
    ON p.marca = s1.marca  AND s1.an=2003 AND s1.luna=7
LEFT OUTER JOIN salarii s2
    ON p.marca = s2.marca  AND s2.an=2003 AND s2.luna=8
```

sau o funcție `DECODE`:

```
SELECT p.marca, numepren,
    DECODE (s1.orelucrate, NULL, 0, s1.orelucrate)
```

```
          AS Ore_Lucr_Iul,
    DECODE (s2.orelucrate, NULL, 0, s2.orelucrate)
          AS Ore_Lucr_aug,
    DECODE (s1.orelucrate, NULL, 0, s1.orelucrate) +
          DECODE (s2.orelucrate, NULL, 0, s2.orelucrate)
          AS "Ore_Lucr_Iul-Aug"
FROM personal p
    LEFT OUTER JOIN salarii s1
          ON p.marca = s1.marca  AND s1.an=2003
                AND s1.luna=7
    LEFT OUTER JOIN salarii s2
          ON p.marca = s2.marca
                AND s2.an=2003 AND s2.luna=8
```

Oracle 9i a preluat din standardele SQL funcția `COALESCE` care, pentru problema de față, se poate aplica identic:

```
SELECT p.marca, numepren,
    COALESCE (s1.orelucrate,0) AS Ore_Lucr_Iul,
    COALESCE (s2.orelucrate,0) AS Ore_Lucr_aug,
    COALESCE(s1.orelucrate,0) + COALESCE (s2.orelucrate,0)
          AS "Ore_Lucr_Iul-Aug"
FROM personal p
    LEFT OUTER JOIN salarii s1
          ON p.marca = s1.marca  AND s1.an=2003
                AND s1.luna=7
    LEFT OUTER JOIN salarii s2
          ON p.marca = s2.marca  AND s2.an=2003
                AND s2.luna=8
```

Formatul general este mai puternic decât cel al funcției `NVL`, în sensul că argumentul funcției este o listă de oricâte valori, returnându-se prima valoare diferită de NULL. De asemea, nu de mult timp a apărut funcția `NVL2` ce evaluează nulitatea unei expresii şi, returnează o valoare a unei alte expresii. Sintaxa este: `NVL2(expresie_de_test,` valoare retunată când `expresie_de_test` este nenulă, valoare returnată când `expresie_de_test` este nulă). Astfel, calculăm salariul de bază pe lunile iulie şi august, ca produs între orele lucrate şi salariul tarifar orar:

```
SELECT p.marca, numepren,
    COALESCE (s1.orelucrate,0) AS Ore_L_Iul,
    NVL2(s1.orelucrate, s1.orelucrate * salorar, 0)
          AS Sal_Baza_Iul,
    COALESCE (s2.orelucrate,0) AS Ore_L_aug,
    NVL2(s2.orelucrate, s2.orelucrate  * salorar, 0)
          AS Sal_Baza_Aug,
    COALESCE(s1.orelucrate,0) + COALESCE (s2.orelucrate,0)
          AS "Ore_Iul-Aug",
    NVL2(s1.orelucrate, s1.orelucrate * salorar, 0) +
```

```
        NVL2(s2.orelucrate, s2.orelucrate  * salorar, 0)
            AS "SalBaza_Iul-Aug"
FROM personal p
    LEFT OUTER JOIN salarii s1
          ON p.marca = s1.marca  AND s1.an=2003
                AND s1.luna=7
    LEFT OUTER JOIN salarii s2
          ON p.marca = s2.marca
                AND s2.an=2003 AND s2.luna=8
```

Pentru prima funcție `NVL2` din interogarea de mai sus logica este următoarea: când valoarea `s1.orelucrate` este nenulă, funcția returnează produsul `s1.orelucrate * salorar`; în caz contrar, zero – vezi figura 6.41.

MARCA	NUMEPREN	ORE_L_IUL	SAL_BAZA_IUL	ORE_L_AUG	SAL_BAZA_AUG	Ore_Iul-Aug	SalBaza_Iul-Aug
101	Angajat 1	72	4032000	32	1792000	104	5824000
102	Angajat 2	40	2300000	16	920000	56	3220000
103	Angajat 3	40	2700000	24	1620000	64	4320000
104	Angajat 4	72	5400000	32	2400000	104	7800000
105	Angajat 5	72	4500000	32	2000000	104	6500000
106	Angajat 6	72	5148000	32	2288000	104	7436000
107	Angajat 7	40	2460000	16	984000	56	3444000
108	Angajat 8	72	3924000	32	1744000	104	5668000
109	Primul Angajat Nou	24	1332000	0	0	24	1332000
110	Al Doilea Angajat Nou	24	1212000	0	0	24	1212000

Figura 6.41. Funcția NVL2

6.6. Funcții-agregat: COUNT, SUM, AVG, MIN, MAX

Intrăm abrupt în câteva exemple.

Câți angajați numără firma ?

Funcția `COUNT` contorizează valorile nenule ale unei coloane sau numărul de linii dintr-un rezultat al unei interogări, altfel spus, în rezultatul unei consultări, `COUNT` numără câte valori diferite de NULL are o coloană specificată sau câte linii sunt în tabela (tabelele) enunțate în clauza `FROM` şi, eventual, "filtrate" în `WHERE`:

```
SELECT COUNT(*) AS Nr_Angajati
FROM personal
```

Prezența asteriscului ca argument al funcției `COUNT` are ca efect numărarea liniilor tabelei – figura 6.42.

```
SQL> SELECT COUNT(*) AS Nr_Angajati
  2  FROM personal
  3  /

NR_ANGAJATI
-----------
         10
```

Figura 6.42. Numărul angajaților

Tabela PERSONAL are drept cheie primară atributul `Marca`, ce nu poate avea valori nule; de aceea, la fel corectă este şi soluția:

```
SELECT COUNT(marca) AS Nr_Angajati
FROM personal
```

Ba chiar locul atributului `Marcă` poate fi luat de orice atribut declarat cu restricția `NOT NULL` la creare.

Câte linii are produsul cartezian al tabelelor PERSONAL şi PONTAJE ?
Simpla enumerare a celor două tabele în cauza `FROM` determină efectuarea produsului cartezian a acestora, funcția `COUNT` numărându-i liniile:

```
SELECT COUNT(*)
FROM personal , pontaje
```

Câți angajați au lucrat pe 10 iulie 2003 ?

```
SELECT COUNT(*)
FROM pontaje
WHERE data = DATE'2003-07-10'
```

Câte compartimente are firma ?
După cum se observă în partea stângă a figurii 6.43, interogarea:

```
SELECT COUNT(compart)
FROM personal
```

furnizează eronat răspunsul, deoarece se numără toate valorile nenule ale atributului `compart`, în timp ce problema priveşte numărul de valorile **distincte** nenule ale acestui atribut, lucru posibil prin întrebuințarea clauzei `DISTINCT` (partea dreaptă a figurii 6.43):

```
SELECT COUNT(DISTINCT compart)
FROM personal
```

```
SQL> SELECT COUNT(compart)
  2  FROM personal
  3  /

COUNT(COMPART)
--------------
            10

SQL> SELECT COUNT(DISTINCT compart)
  2  FROM personal
  3  /

COUNT(DISTINCTCOMPART)
----------------------
                     3
```

Figura 6.43. Funcția COUNT, fără şi cu DISTINCT

Câte ore de noapte are pe luna iulie 2003 angajatul cu marca 104 ?
Intră în scenă funcția `SUM` (rezultatul în figura 6.44):

```
SELECT SUM (orenoapte)
FROM pontaje INNER JOIN personal
    ON pontaje.marca=personal.marca
WHERE TO_CHAR(data, 'MM/YYYY')='07/2003' AND
    numepren = 'Angajat 4'
```

```
SQL> SELECT SUM (orenoapte)
  2  FROM pontaje INNER JOIN personal
  3  ON pontaje.marca=personal.marca
  4  WHERE TO_CHAR(data, 'MM/YYYY')='07/2003' AND
  5  numepren = 'Angajat 4'
  6  /

SUM(ORENOAPTE)
--------------
             5
```

Figura 6.44. Funcția SUM

Ca şi în cazul funcției precedente, funcția `SUM` poate lua în calcul o singură dată fiecare valoare distinctă – vezi figura 6.45.

```
SQL> SELECT SUM (orenoapte) FROM pontaje
  2  WHERE TO_CHAR(data, 'MM/YYYY')='07/2003'
  3  /

SUM(ORENOAPTE)
--------------
            14

SQL> SELECT SUM (DISTINCT orenoapte) FROM pontaje
  2  WHERE TO_CHAR(data, 'MM/YYYY')='07/2003'
  3  /

SUM(DISTINCTORENOAPTE)
----------------------
                     9
```

Figura 6.45. Funcția SUM, cu şi fără DISTINCT

Care este salariul mediu al angajaților din compartimentul IT ?

Funcţia cea mai numerită este AVG (de la AVeraGe, adică medie aritmetică). Pentru comparabilitate, interogarea următoare (figura 6.46) extrage şi numărul angajaţilor IT şi suma salariilor orare ale acestora:

```
SELECT COUNT(*) AS Nr_IT, SUM(salorar) AS Suma_IT,
    AVG(salorar) AS Medie_Sal_IT
FROM personal WHERE compart = 'IT'
```

```
SQL>  SELECT COUNT(*) AS Nr_IT, SUM(salorar) AS Suma_IT,
  2      AVG(salorar) AS Medie_Sal_IT
  3   FROM personal
  4   WHERE compart = 'IT'
  5  /

     NR_IT    SUMA_IT MEDIE_SAL_IT
---------- ---------- ------------
         4     241500        60375
```

Figura 6.46. Funcţia AVG

Funcţia AVG ia în considerare numai valorile nenule. Pentru a sesiza diferenţa, interogarea următoare calculează o medie a orelor lucrate pe luna iulie 2003 în două moduri: prin împărţirea sumei orelor lucrate la produsul dintre numărul zilelor lucratoare din iulie şi cel al angajaţilor (coloana Medie1_Ore), iar, în al doilea mod, se foloseşte funcţia AVG (AVG_Ore):

```
SELECT COUNT(DISTINCT Data) AS Nr_Zile,
    COUNT(DISTINCT Marca) AS Nr_Angajati,
    SUM(orelucrate) AS Suma_Ore,
    SUM(orelucrate) /
       (COUNT(DISTINCT Data) * COUNT(DISTINCT marca))
      AS Medie1_Ore,
    AVG(orelucrate) AS AVG_Ore
FROM pontaje
WHERE TO_CHAR(data, 'MM/YYYY')='07/2003'
```

Diferenţa dintre valorile celor două medii (vezi figura 6.47) se datorează faptului că pentru calculul Medie1_Ore suma orelor lucrate (528) se împarte la produsul nr_zile * nr_angajaţi (90), rezultatul fiind 5,8666..., în timp ce AVG_Ore reprezintă raportul dintre 528 şi 78, deoarece o parte dintre angajaţi nu au pontaje pentru toate zilele lucrătoare din iulie.

```
 NR_ZILE NR_ANGAJATI   SUMA_ORE MEDIE1_ORE    AVG_ORE
-------- ----------- ---------- ---------- ----------
       9          10        528 5.86666667 6.76923077
```

Figura 6.47. Două medii

În funcţie de ceea ce se doreşte efectiv a se calcula, se poate alege una din cele două metode.

Care este cel mai mare şi cel mai mic salariu orar ?

A sosit momentul folosirii funcțiilor `MAX` şi `MIN`:

```
SELECT MAX(SalOrar) AS Sal_MAX, MIN(SalOrar) AS Sal_MIN
FROM personal
```

SAL_MAX	SAL_MIN
75000	50500

Figura 6.48. Funcțiile MAX şi MIN

Care este ultima zi lucrătoare a lunii iulie 2003 ?

```
SELECT MAX(Data) AS Ultima_Zi_Lucratoare_Iul
FROM pontaje
WHERE TO_CHAR(data, 'MM/YYYY')='07/2003'
```

Care sunt cele mai mari două salarii orare ?

Cu nivelul de SQL de care dispunem putem formula o soluție interesantă prin joncționarea a două instanțe ale tabelei PERSONAL, după condiția: salariul orar din prima tabelă să fie mai mare decât cel din a doua. Rezultatul joncțiunii conține 45 de linii (dacă nu ne credeți, calculați !), însă grație funcției `MAX` va fi selectată numai prima, cea care ne furnizează răspunsul (vezi figura 6.49):

```
SELECT MAX (
    'Pe locul I ' ||
    TO_CHAR(P1.SalOrar, '9999999')  ||
    ', iar pe locul al II-lea ' ||
    TO_CHAR(P2.SalOrar, '9999999')
          ) AS Tip_Top_Mini_Top
FROM personal p1 INNER JOIN personal p2
    ON p1.salorar > p2.salorar
```

TIP_TOP_MINI_TOP
Pe locul I 75000, iar pe locul al II-lea 71500

Figura 6.49. Primele două salare orare

6.7. Gruparea tuplurilor. GROUP BY şi HAVING

Clauza `GROUP BY` formează grupe (grupuri) de tupluri ale unei relații, pe baza valorilor comune ale unui atribut, iar `HAVING` permite selectarea anumitor *grupuri de tupluri* ce îndeplinesc un criteriu, criteriu valabil numai la nivel de grup (nu şi la nivel de linie).

Câți angajați lucrează în fiecare compartiment ?

Atributul de grupare este `Compart`, iar funcţia ce determină numărul angajaţilor este `COUNT` (vezi şi figura 6.50):

```
SELECT compart, COUNT(*) AS Nr_Angajati
FROM personal
GROUP BY compart
```

COMPART	NR_ANGAJATI
CONTA	3
IT	4
PROD	3

Figura 6.50. Numărul angajaților din fiecare compartiment

Să se afişeze situaţia orelor lucrate de fiecare angajat pe luna iulie 2003. Ordinea afişării este cea a compartimentelor, în cadrul fiecărui compartiment angajaţii fiind ordonaţi alfabetic.

De data aceasta, avem două atribute de grupare, `Compart` şi `NumePren`, plus funcţia `SUM`. Dispunerea liniilor în rezultat este asigurată prin clauza `ORDER BY`:

```
SELECT compart, numepren, SUM(orelucrate) AS Nr_Ore_L
FROM personal pe LEFT OUTER JOIN pontaje po
    ON pe.marca=po.marca AND
          TO_CHAR(data,'MM/YYYY')='07/2003'
GROUP BY compart, numepren
ORDER BY compart, numepren
```

Iata şi rezultatul – figura 6.51.

COMPART	NUMEPREN	NR_ORE_L
CONTA	Angajat 2	40
CONTA	Angajat 6	72
CONTA	Angajat 7	40
IT	Angajat 1	72
IT	Angajat 3	40
IT	Angajat 5	72
IT	Primul Angajat Nou	24
PROD	Al Doilea Angajat Nou	24
PROD	Angajat 4	72
PROD	Angajat 8	72

Figura 6.51. Orele lucrate de fiecare angajat pe luna iulie 2003

Să se afişeze venitul de bază corespunzător orelor lucrate de fiecare angajat pe luna iulie 2003, calculându-se subtotaluri pe compartimente, precum şi un total general

Practic, rezultatul conţine trei tipuri de înregistrări:

- pentru fiecare angajat datele sunt obţinute ca în interogarea anterioară, prin gruparea după atributele `Compart` şi `Numepren`;

- pentru fiecare compartiment funcția `SUM` este identică, însă gruparea se realizează numai după un atribut – `Compart`;
- totalul general presupune aplicarea funcției `SUM` fără grupare.

Iată interogarea:

```
  SELECT compart, numepren,
      SUM(orelucrate * salorar) AS Venit_Baza
  FROM personal pe LEFT OUTER JOIN pontaje po
      ON pe.marca=po.marca
            AND TO_CHAR(data,'MM/YYYY')='07/2003'
  GROUP BY compart, numepren
UNION
  SELECT compart, RPAD(CHR(123) || ' Subtotal - compart  ' ||
      compart,32,'-') ,
      SUM(orelucrate * salorar)
  FROM personal pe LEFT OUTER JOIN pontaje po
      ON pe.marca=po.marca
            AND TO_CHAR(data,'MM/YYYY')='07/2003'
  GROUP BY compart, CHR(254) || 'Subtotal - compart '
      || compart
UNION
  SELECT RPAD(CHR(123), 8,'='),
      RPAD(CHR(123) || ' TOTAL General ' , 32, '=') ,
      SUM(orelucrate * salorar)
  FROM personal pe LEFT OUTER JOIN pontaje po
      ON pe.marca=po.marca
            AND TO_CHAR(data,'MM/YYYY')='07/2003'
  ORDER BY 1, 2
```

Celor trei tipuri de înregistrări le corespund trei fraze `SELECT` conectate prin operatorul `UNION`. Clauza `ORDER BY` se plasează după ultimul `SELECT`, coloanele fiind indicate prin poziția (numărul) lor. Pentru a asigura ordonarea corespunzătoare, s-a recurs la caracterul ce are codul ASCII imediat superior lui `z`, adică 123. Rezultatul este cel din figura 6.52.

COMPART	NUMEPREN	VENIT_BAZA
CONTA	Angajat 2	2300000
CONTA	Angajat 6	5148000
CONTA	Angajat 7	2460000
CONTA	{ Subtotal - compart CONTA-----	9908000
IT	Angajat 1	4032000
IT	Angajat 3	2700000
IT	Angajat 5	4500000
IT	Primul Angajat Nou	1332000
IT	{ Subtotal - compart IT--------	12564000
PROD	Al Doilea Angajat Nou	1212000
PROD	Angajat 4	5400000
PROD	Angajat 8	3924000
PROD	{ Subtotal - compart PROD------	10536000
{=======	{ TOTAL General ================	33008000

Figura 6.52. Subtotaluri şi total general

Care este cel mai mare venit pe bază pe luna iulie 2003 ?
În Oracle se poate include funcţia `SUM` în `MAX`, aşa încât interogarea următoare este funcţională (vezi figura 6.53):

```
SELECT MAX(SUM(orelucrate * salorar)) AS Venit_Baza_MAX
FROM personal pe LEFT OUTER JOIN pontaje po
ON pe.marca=po.marca AND TO_CHAR(data,'MM/YYYY')='07/2003'
GROUP BY numepren
```

VENIT_BAZA_MAX
5400000

Figura 6.53. Includerea unei funcţii agregat în alta

În ce zile din iulie au fost prezenţi la lucru mai mult de 8 angajaţi ?
Condiţia (8 angajaţi la lucru) nu se poate aplica la nivel de linie, ci la nivel de grup, grup constituit la nivelul unei zile prin numărarea angajaţilor cu ore lucrate în data respectivă. Pentru acest gen de probleme a fost creată clauza `HAVING`:

```
SELECT data, COUNT(*) AS Nr_Pers
FROM pontaje
WHERE TO_CHAR(data,'MM/YYYY')='07/2003' AND orelucrate > 0
GROUP BY data
HAVING COUNT(*) > 8
```

rezultatul fiind cel din figura 6.54.

DATA	NR_PERS
09-JUL-03	9
10-JUL-03	9
11-JUL-03	9

Figura 6.54. Clauza HAVING

Care sunt angajații cu un număr de zile lucrate peste cel al Primului Angajat Nou ?
Ideea soluției care urmează ține de efectuarea produsului cartezian a două instanțe ale cuplului PERSONAL-PONTAJE (joncționate intern, selectându-se, totuşi, numai zilele pentru care `OreLucrate > 0`). Rezultatul acestui produs cartezian este impresionant – 17689 de linii – vezi figura 6.55.

```
SQL> SELECT COUNT(*)
  2  FROM (pontaje po1 INNER JOIN personal pe1 ON po1.marca=pe1.marca),
  3  (pontaje po2 INNER JOIN personal pe2 ON po2.marca=pe2.marca)
  4  WHERE po1.orelucrate > 0 AND po2.orelucrate > 0
  5  /

  COUNT(*)
----------
     17689
```

Figura 6.55. Numărul de linii ale produsului cartezian

Din produsul cartezian de mai sus se extrag numai liniile care, în a doua instanță PERSONAL-PONTAJE, privesc Primul Angajat Nou, şi apoi se face o grupare după marca şi numele primei instanțe şi marca celei de-a doua instanțe. Clauza `HAVING` asigură extragerea în rezultat numai a grupurilor (angajaților) pentru care numărul zilelor de pontaj este mai mare decât cel al Primului Angajat Nou.

```
SELECT po1.marca, pe1.numepren,
    COUNT(DISTINCT po1.data) AS Zile_Lucrate,
    COUNT(DISTINCT po2.data)  AS Zile_Lucrate_PAN
FROM (pontaje po1 INNER JOIN personal pe1
    ON po1.marca=pe1.marca),
    (pontaje po2 INNER JOIN personal pe2
    ON po2.marca=pe2.marca)
WHERE po1.orelucrate > 0 AND po2.orelucrate > 0 AND
    pe2.numepren = 'Primul Angajat Nou'
GROUP BY po1.marca, pe1.numepren, po2.marca
HAVING COUNT(DISTINCT po1.data) > COUNT(DISTINCT po2.data)
```

Atributele afişate (vezi figura 6.56) sunt: marca, numele, numărul zilelor lucrate de angajatul de pe linia (grupul) respectiv, şi numărul zilelor lucrate de persoana etalon.

```
SQL> SELECT po1.marca, pe1.numepren,
  2  COUNT(DISTINCT po1.data) AS Zile_Lucrate,
  3  COUNT(DISTINCT po2.data)  AS Zile_Lucrate_PAN
  4  FROM (pontaje po1 INNER JOIN personal pe1 ON po1.marca=pe1.marca),
  5  (pontaje po2 INNER JOIN personal pe2 ON po2.marca=pe2.marca)
  6  WHERE po1.orelucrate > 0 AND po2.orelucrate > 0 AND
  7  pe2.numepren = 'Primul Angajat Nou'
  8  GROUP BY po1.marca, pe1.numepren, po2.marca
  9  HAVING COUNT(DISTINCT po1.data) > COUNT(DISTINCT po2.data)
 10  /

 MARCA NUMEPREN                         ZILE_LUCRATE ZILE_LUCRATE_PAN
------ ------------------------------- ------------ ----------------
   101 Angajat 1                                 18                3
   102 Angajat 2                                 12                3
   103 Angajat 3                                 13                3
   104 Angajat 4                                 18                3
   105 Angajat 5                                 18                3
   106 Angajat 6                                 18                3
   107 Angajat 7                                 12                3
   108 Angajat 8                                 18                3
```

Figura 6.56. Persoanele cu mai multe zile lucrate decât Primul Angajat Nou

Care sunt zilele în care au lucrat simultan Angajat 1 şi Primul Angajat Nou ?
Constituim un grup pentru fiecare zi lucrătoare, extrăgând numai liniile referitoare la unul dintre cei doi angajaţi. Datele care interesează corespund grupurilor ce conţin două linii:

```
SELECT data AS Ziua
FROM pontaje po INNER JOIN personal pe
    ON po.marca = pe.marca
WHERE numepren IN ( 'Angajat 1', 'Primul Angajat Nou')
GROUP BY Data
HAVING COUNT(*) = 2
```

6.8. Subconsultări. Operatorul IN

Nucleul SQL din Oracle este unul deosebit de generos şi în privinţa subconsultărilor şi, când spunem acest lucru, ne gândim în primul rând la numărul aproape nelimitat de nivele pe care pot fi incluse frazele `SELECT`. Marea majoritate a interogărilor incluse fac apel la operatorul `IN`.

Care sunt colegii de compartiment ai lui Angajat 2 ?
Dacă în paragraful dedicat joncţiunii interne am apelat la artificiul joncţionării a două instanţe ale tabelei PERSONAL după atributul `compart` şi filtrarea liniilor pentru una din instanţe (`numepren = 'Angajat 2'`), lucrurile pot fi simplificate recurgând la o subconsultare:

```
SELECT numepren
FROM personal
WHERE numepren <> 'Angajat 2' AND compart IN
    (SELECT compart
     FROM personal
```

```
      WHERE numepren = 'Angajat 2')
```

Mai întâi se execută subconsultarea, obținându-se un rezultat intermediar ce conține o singură linie şi o singură coloană – compartimentul Angajatului 2. În pasul al doilea se execută `SELECT`-ul principal care extrage din PERSONAL liniile în care `NumePren` este diferit de cel al angajatului etalon, iar valoarea atributului `Compart` este una din rezultatul intermediar.

Care sunt zilele în care au lucrat simultan Angajat 1 şi Primul Angajat Nou ?
Celor trei soluții bazate pe joncțiune, intersecție şi grupare le adăugăm o alta care foloseşte o subconsultare:

```
SELECT data AS Ziua
FROM pontaje po INNER JOIN personal pe
    ON po.marca = pe.marca
WHERE numepren = 'Angajat 1' AND data IN
    (SELECT data
     FROM pontaje po INNER JOIN personal pe
          ON po.marca = pe.marca
     WHERE numepren = 'Primul Angajat Nou')
```

Interesant este că subconsultarea poate substitui de multe ori joncțiunea. Astfel, echivalent cu fraza `SELECT` de mai sus se poate scrie:

```
SELECT data AS Ziua
FROM pontaje
WHERE marca IN
    (SELECT marca
     FROM personal
     WHERE numepren = 'Angajat 1'
     )
    AND data IN
          (SELECT data
           FROM pontaje
           WHERE marca IN
    (SELECT marca
     FROM personal
     WHERE numepren = 'Primul Angajat Nou'
    )     )
```

Care sunt zilele din iulie 2003 în care a lucrat Angajat 1, dar NU a lucrat Primul Angajat Nou ?
Dacă varianta precedentă de rezolvare a acestei probleme făcea apel la operatorul diferență (`EXCEPT/MINUS`), folosind o subconsultare lucrurile stau aproape identic interogării de mai sus, numai că `IN` se înlocuieşte cu `NOT IN`:

```
SELECT data AS Ziua
FROM pontaje po INNER JOIN personal pe
    ON po.marca = pe.marca
WHERE numepren = 'Angajat 1'
```

```
AND TO_CHAR(data, 'MM/YYYY')='07/2003'
AND data NOT IN
      (SELECT data
       FROM pontaje po INNER JOIN personal pe
            ON po.marca = pe.marca
       WHERE numepren = 'Primul Angajat Nou')
```

Care este angajatul (angajaţii) cu cel mai mare salariul orar ?
Una dintre greşelile cele mai frecvente ale începătorilor în SQL este folosirea unei soluţii de genul celei din figura 6.57, în care clauza `SELECT` conţine numele angajatului şi funcţia `MAX`.

```
SQL> SELECT numepren, MAX(salorar)
  2  FROM personal
  3  /
SELECT numepren, MAX(salorar)
       *
ERROR at line 1:
ORA-00937: not a single-group group function
```

Figura 6.57. Tentativă eşuată de aflare a angajatului cel mai bine plătit

Varianta corectă presupune utilizarea funcţiei într-o subconsultare. Informaţia furnizată este corectă, după cum reiese din figura 6.58.

```
SELECT numepren, salorar
FROM personal
WHERE salorar IN
    (SELECT MAX(salorar)
     FROM personal)
```

NUMEPREN	SALORAR
Angajat 4	75000

Figura 6.58. Angajatul ce are cel mai mare salariu orar

Care sunt angajaţii cu cele mai mari două salarii orare ?
Trei fraze `SELECT` incluse de maniera următoare rezovă problema:

```
SELECT numepren, salorar
FROM personal
WHERE salorar >=
    (SELECT MAX(salorar)
     FROM personal
     WHERE salorar <
          (SELECT MAX(salorar)
           FROM personal )        )
```

Mai greu este cu explicatul. De aceea, figura 6.59 conţine cele trei etape ale execuţiei. `SELECT`-ul "cel mai de jos" extrage cel mai mare salariu orar, în timp ce `SELECT`-ul intermediar determină penultimul salariu orar.

```
SQL> SELECT MAX(salorar)
  2   FROM personal
  3  /

MAX(SALORAR)
------------
       75000

SQL> SELECT MAX(salorar)
  2    FROM personal
  3  WHERE salorar <
  4   (SELECT MAX(salorar)
  5   FROM personal )
  6  /

MAX(SALORAR)
------------
       71500

SQL> SELECT numepren, salorar
  2  FROM personal
  3  WHERE salorar >=
  4   (SELECT MAX(salorar)
  5    FROM personal
  6  WHERE salorar <
  7   (SELECT MAX(salorar)
  8   FROM personal )  )
  9  /

NUMEPREN                       SALORAR
------------------------- ----------
Angajat 4                       75000
Angajat 6                       71500

SQL>
```

Figura 6.59. Angajații cu cele mai mare două salarii

Fraza principală îndeplineşte o simplă formalitate, extrăgând toți angajații cu salariul mai mare sau egal cu penultima valoare furnizată de subconsultarea mediană.

6.9. Subconsultări în clauza HAVING. Operatorii ALL, SOME, ANY

Posibilitatea folosirii subconsultărilor ca termeni ai clauzei `HAVING` este un mare atu în rezolvarea unor probleme informaționale cu un grad de dificultate ceva mai pronunțat. Predicatele clauzei conțin, pe lângă clasicii operatori de comparație, şi trei ceva mai speciali, `ALL`, `SOME` şi `ANY`.

Care sunt angajații cu un număr de zile lucrate peste cel al Primului Angajat Nou ?
Prin grupare, fiecărei persoane i se numără zilele de lucru. Clauza `HAVING` asigură extragerea numai a acelor grupuri la care numărul zilelor este mai mare decât al angajatului etalon, număr calculat printr-o subconsultare:

```
SELECT po.marca, numepren, COUNT(DISTINCT data)
    AS Zile_Lucrate
```

```
  FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
  WHERE orelucrate > 0
  GROUP BY po.marca, numepren
  HAVING COUNT(DISTINCT data) >
(SELECT  COUNT(DISTINCT data)
 FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
 WHERE orelucrate > 0 AND numepren = 'Primul Angajat Nou' )
```

Figura 6.60 ilustrează modul de derulare al interogării.

```
SQL> SELECT  COUNT(DISTINCT data)
  2   FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
  3   WHERE orelucrate > 0 AND numepren = 'Primul Angajat Nou'
  4  /

COUNT(DISTINCTDATA)
-------------------
                  3

SQL> SELECT po.marca, numepren, COUNT(DISTINCT data) AS Zile_Lucrate
  2  FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
  3  WHERE orelucrate > 0
  4  GROUP BY po.marca, numepren
  5  HAVING COUNT(DISTINCT data) >
  6   (SELECT  COUNT(DISTINCT data)
  7   FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
  8   WHERE orelucrate > 0 AND numepren = 'Primul Angajat Nou'
  9  )
 10  /

     MARCA NUMEPREN                                   ZILE_LUCRATE
---------- ---------------------------------------- ------------
       101 Angajat 1                                          18
       102 Angajat 2                                          12
       103 Angajat 3                                          13
       104 Angajat 4                                          18
       105 Angajat 5                                          18
       106 Angajat 6                                          18
       107 Angajat 7                                          12
       108 Angajat 8                                          18

8 rows selected.

SQL>
```

Figura 6.60. Persoanele cu mai multe zile lucrate decât Primul Angajat Nou – varianta 2

Care dintre angajați a fost la lucru în toate zilele lucrătoare ?

Lucrurile nu sunt atât de complicate cum par la prima vedere. Avem nevoie de o subconsultare care să calculeze numărul zilelor lucrătoare. Fraza `SELECT` principală va calcula numărul zilelor lucrate de fiecare angajat pe care îl va compara cu rezultatul subconsultării – vezi figura 6.61.

```
SELECT po.marca, numepren, COUNT(data) AS Zile_Lucrate
FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
WHERE orelucrate > 0
GROUP BY po.marca, numepren
HAVING COUNT(data) =
    (SELECT  COUNT(DISTINCT data)
     FROM pontaje
     WHERE orelucrate > 0 )
```

```
SQL> SELECT  COUNT(DISTINCT data)
  2   FROM pontaje
  3   WHERE orelucrate > 0
  4  /

COUNT(DISTINCTDATA)
-------------------
                 18

SQL> SELECT po.marca, numepren, COUNT(data) AS Zile_Lucrate
  2  FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
  3  WHERE orelucrate > 0
  4  GROUP BY po.marca, numepren
  5  HAVING COUNT(data) =
  6   (SELECT  COUNT(DISTINCT data)
  7   FROM pontaje
  8   WHERE orelucrate > 0 )
  9  /

     MARCA NUMEPREN             ZILE_LUCRATE
---------- -------------------- ------------
       101 Angajat 1                      18
       104 Angajat 4                      18
       105 Angajat 5                      18
       106 Angajat 6                      18
       108 Angajat 8                      18

SQL> |
```

Figura 6.61. Persoanele prezente la muncă în toate zilele lucrătoare

Care este angajatul (sau angajații) cu cel mai mare venit (obținut din orele lucrate, plus eventualele concedii de odihnă) ?

Ne gândim la o subconsultare care să calculeze veniturile de bază pentru fiecare angajat, subconsultare pe care să o folosim ca element de comparație într-un `SELECT` principal ce grupează veniturile pe angajați. Zis şi făcut:

```
SELECT po.marca, numepren,
SUM (orelucrate * salorar + oreco * salorarco)
    AS Venit_Baza
FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
GROUP BY po.marca, numepren
HAVING SUM (orelucrate * salorar + oreco * salorarco)
    >= ALL
    (SELECT  SUM (orelucrate * salorar + oreco * salorarco)
     FROM pontaje po INNER JOIN personal pe
          ON po.marca=pe.marca
     GROUP BY po.marca)
```

După cum se observă şi în figura 6.62, subconsultarea extrage zece linii (câte una pentru fiecare angajat) şi o singură coloană – venitul de bază. Spre deosebire de interogările precedente, pentru funcționarea corespunzătoare a clauzei `HAVING` este necesară folosirea operatorului `ALL`, astfel încât rezultatul va conține numai

linia/liniile cu valori mai mare sau egale tuturor valorilor furnizate de subconsultare.

```
SQL> SELECT  SUM (orelucrate * salorar + oreco * salorarco)
  2   FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
  3   GROUP BY po.marca
  4  /

SUM(ORELUCRATE*SALORAR+ORECO*SALORARCO)
---------------------------------------
                                8064000
                                8208000
                                9660000
                               10800000
                                9000000
                               10296000
                                8784000
                                7848000
                                1332000
                                1212000

10 rows selected.

SQL> SELECT po.marca, numepren,
  2  SUM (orelucrate * salorar + oreco * salorarco) AS Venit_Baza
  3  FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
  4  GROUP BY po.marca, numepren
  5  HAVING SUM (orelucrate * salorar + oreco * salorarco) >= ALL
  6   (SELECT  SUM (orelucrate * salorar + oreco * salorarco)
  7   FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
  8   GROUP BY po.marca)
  9  /

     MARCA NUMEPREN             VENIT_BAZA
---------- -------------------- ----------
       104 Angajat 4                10800000

SQL>
```

Figura 6.62. Persoana cu venitul cel mai mare

Prezența operatorului `ALL` este indispensabilă deoarece rezultatul subconsultării este multi-linie. În lipsa acestuia, mesajul de eroare este cel din figura 6.63.

```
SQL> SELECT po.marca, numepren,
  2  SUM (orelucrate * salorar + oreco * salorarco) AS Venit_Baza
  3  FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
  4  GROUP BY po.marca, numepren
  5  HAVING SUM (orelucrate * salorar + oreco * salorarco) >=
  6   (SELECT  SUM (orelucrate * salorar + oreco * salorarco)
  7   FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
  8   GROUP BY po.marca)
  9  /
 (SELECT  SUM (orelucrate * salorar + oreco * salorarco)
  *
ERROR at line 6:
ORA-01427: single-row subquery returns more than one row
```

Figura 6.63. Clauză HAVING incorectă în condițiile unei subconsultări multi-linie

Ce persoane au venitul de bază superior măcar unui angajat al compartimentului Contabilitate ?

De data aceasta, pentru ca un angajat să fie inclus în rezultat, venitul său trebuie să fie superior sau egal măcar unuia dintre oamenii muncii de la contabilitate, ceea ce reclamă folsirea operatorului `ANY`:

```
SELECT po.marca, numepren, compart,
    SUM (orelucrate * salorar + oreco * salorarco)
    AS Venit_Baza
FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
WHERE compart <> 'CONTA'
GROUP BY po.marca, numepren, compart
HAVING SUM (orelucrate * salorar + oreco * salorarco) >=ANY
    (SELECT  SUM (orelucrate * salorar + oreco * salorarco)
     FROM pontaje po INNER JOIN personal pe
          ON po.marca=pe.marca
     WHERE compart = 'CONTA'
     GROUP BY po.marca)
```

Lucrurile ar decurge identic şi dacă locul operatorului `ANY` ar fi luat de un altul înrudit – `SOME`. Cum la contabilitate sunt trei angajați, subconsultarea se va materializa în trei valori (linii) – vezi figura 6.64 – din care cea minimă este 8 208 000. Cele trei persoane extrase în rezultatul final au măcar veniturile de 8 208 000 lei, aşa că se poate formula o variantă şi fără operatorul `ANY`:

```
SELECT po.marca, numepren, compart,
    SUM (orelucrate * salorar + oreco * salorarco)
          AS Venit_Baza
FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
WHERE compart <> 'CONTA'
GROUP BY po.marca, numepren, compart
HAVING SUM (orelucrate * salorar + oreco * salorarco) >=
    (SELECT  MIN(SUM (orelucrate * salorar + oreco
          * salorarco))
     FROM pontaje po INNER JOIN personal pe
          ON po.marca=pe.marca
     WHERE compart = 'CONTA'
     GROUP BY po.marca)
```

```
SQL> SELECT  SUM (orelucrate * salorar + oreco * salorarco)
  2   FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
  3   WHERE compart = 'CONTA'
  4   GROUP BY po.marca
  5  /

SUM(ORELUCRATE*SALORAR+ORECO*SALORARCO)
---------------------------------------
                                8208000
                               10296000
                                8784000

SQL> SELECT po.marca, numepren, compart,
  2  SUM (orelucrate * salorar + oreco * salorarco) AS Venit_Baza
  3  FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
  4  WHERE compart <> 'CONTA'
  5  GROUP BY po.marca, numepren, compart
  6  HAVING SUM (orelucrate * salorar + oreco * salorarco) >= ANY
  7   (SELECT  SUM (orelucrate * salorar + oreco * salorarco)
  8   FROM pontaje po INNER JOIN personal pe ON po.marca=pe.marca
  9   WHERE compart = 'CONTA'
 10   GROUP BY po.marca)
 11  /

     MARCA NUMEPREN             COMPA VENIT_BAZA
---------- -------------------- ----- ----------
       103 Angajat 3            IT       9660000
       104 Angajat 4            PROD    10800000
       105 Angajat 5            IT       9000000
```

Figura 6.64. Operatorul ANY