# II.2.1. Tipuri de funcţii

Funcţiile Oracle sunt împărţite astfel:

-      **Funcţii singulare** – acestea operează la un moment dat asupra unei singure înregistrări. Aceste funcţii vor fi discutate în acest capitol

-      **Funcţiile de grup** – operează asupra unui grup de înregistrări şi returnează o singură valoare pentru întregul grup.

Funcţiile singulare pot fi folosite în:

-      clauza **SELECT**, pentru a modifica modul de afişare a datelor, pentru a realiza diferite calcule etc.

-      clauza **WHERE**, pentru a preciza mai exact care sunt înregistrările ce se afişează

-      clauza **ORDER BY**

Funcţiile singulare (single-row functions) pot fi la rândul lor împărţite în:

-      Funcţii care operează asupra şirurilor de caractere

-      Funcţii numerice

-      Funcţii pentru manipularea datelor calendaristice

-      Funcţii de conversie – care convertesc datele dintr-un tip în altul

-      Funcţii de uz general.

Unele funcţii, precum **TRUNC** şi **ROUND** pot acţiona asupra mai multor tipuri de date, dar cu semnificaţii diferite.

# II.2.2. Tabela DUAL

În cele ce urmează vom folosi tabela **DUAL** pentru a testa modul de operare a funcţiilor singulare.

Această tabela este una specială, care conţine o singură coloană numită ”**DUMMY**” şi o singură linie.

Tabela **DUAL** se foloseşte atunci când realizăm calcule, sau evaluăm expresii care nu derivă din nici o tabelă anume.

Fie de exemplu comanda

 **SELECT (5\*7-3)/2 FROM DUAL;**

Expresia evaluată în această comandă nu are în componenţă nicio coloană a vreunei tabele, motiv pentru care este nevoie să apelăm la tabela **DUAL**.

Putem privi tabela **DUAL** ca pe o variabilă în care memorăm rezultatele calculelor noastre.

Tabela **DUAL** este o facilitate specifică Oracle. Este echivalentul tabelei **SYSDUMMY1** din **DB2**, tabelă aflată în schema sistem **SYSIBM**. În **Microsoft SQL Server 2000** este permisă scrierea de interogări fără clauza **FROM**.

# II.2.3. Funcţii asupra şirurilor de caractere

Şirurile de caractere pot conţine orice combinaţie de litere, numere, spaţii, şi alte simboluri, precum semne de punctuaţie, sau caractere speciale. În Oracle există două tipuri de date pentru memorarea şirurilor de caractere:

-      **CHAR** – pentru memorarea şirurilor de caractere de lungime fixă

-      **VARCHAR2** – pentru memorarea şirurilor de caractere având lungime variabilă.

  **LOWER(sir)** – converteşte caracterele alfanumerice din şir în litere mici.

  **UPPER(sir)** – converteşte caracterele alfanumerice din şir în litere mari.

  **INITCAP(sir)** – converteşte la majusculă prima literă din fiecare cuvânt al şirului. Cuvintele sunt şiruri de litere separate prin orice caracter diferit de literă. Literele din interiorul cuvântului care erau scrise cu majuscule vor fi transformate în litere mici.

|  |  |
| --- | --- |
| **Exemplu** | **Rezultatul afişat** |
| **SELECT LOWER(first\_name)**  **FROM employees;** | afişează prenumele persoanelor din tabela **employees** scrise cu litere mici |
| **SELECT LOWER('abc123ABC')**  **FROM DUAL;** | **abc123abc** |
| **SELECT UPPER('abc123ABC')**  **FROM DUAL;** | **ABC123ABC** |
| **SELECT INITCAP('aBc def\*ghi')**  **FROM dual;** | **Abc Def\*Ghi**  ***Explicaţie*** şirul conţine **3** cuvinte **aBc def** şi **ghi** |

  **CONCAT(sir1, sir2)** – concatenează două şiruri de caractere

|  |  |
| --- | --- |
| **Exemplu** | **Rezultatul afişat** |
| **SELECT CONCAT('abc','def')**  **FROM dual;** | **abcdef**  ***Explicaţie*** comanda poate fi transcrisă folosind operatorul de concatenare astfel:  **SELECT 'abc'||'def'**  **FROM dual;**  **SELECT first\_name || ' ' || last\_name ”nume angajat” from emloyees;** |

  **SUBSTR(sir,poz,nr)** – extrage din **sir** cel mult **nr** caractere începând din poziţia **poz**.

***Observaţii***

-      dacă din poziţia **poz** până la sfârşitul şirului sunt mai puţin de **nr** caractere, se vor extrage toate caracterele de la poziţia **poz** până la sfârşitul şirului.

-      parametrul **poz** poate fi şi o valoare negativă, ceea ce înseamnă că poziţia de unde se va începe extragerea caracterelor din şir se va determina numărând caracterele din şir de la dreapta spre stânga (vezi ultimele 3 exemple de mai jos)

-      dacă **nr** nu este specificat, se va returna subşirul începând cu caracterul de pe poziţia **poz** din şir până la sfârşitul şirului.

| **Exemplu** | **Rezultatul afişat** |
| --- | --- |
| **select substr('abcdef',3,2)**  **from dual** | **cd** |
| **select substr('abcdef',3,7)**  **from dual** | **cdef**  ***Explicaţie.*** Chiar dacă din poziţia 3 până la sfârşitul şirului nu mai sunt 7 caractere se returnează caracterele rămase |
| **select substr('abcdef',3)**  **from dual** | **cdef**  ***Explicaţie.***Acelaşi rezultat ca mai sus dacă nu se specifică numărul de caractere ce se extrag |
| **select substr('abcdef',7,3)**  **from dual** | nu se va afişa nimic deoarece nu există poziţia **7** în şir, acesta având doar **5** caractere. |
| **select substr('abcdef',-4,2)**  **from dual** | **cd**  ***Explicaţie.***Se extrag două caractere începând cu al patrulea caracter din dreapta. |
| **select substr('abcdef',-4,7)**  **from dual** | **cdef** |
| **select substr('abcdef',-10,5)**  **from dual** | nu se va afişa nimic deoarece şirul conţine mai puţin de 10 caractere |

  **INSTR(sir,subsir,poz,k)** – returnează poziţia de început a celei de a **k**-a apariţii a subşirului **subsir** în şirul **sir**, căutarea făcându-se începând cu poziţia **poz** .

Dacă parametrii **poz** şi **k** lipsesc, atunci se va returna poziţia primei apariţii a subşirului **subsir** în întregul şir **sir**.

Poziţia de unde începe căutarea poate fi precizată şi relativ la sfârşitul şirului, ca şi în cazul funcţiei **substr**, dacă parametrul **poz** are o valoare negativă.

| **Exemplu** | **Rezultatul afişat** |
| --- | --- |
| **select**  **instr('abcdabcdabc','cd')**  **from dual** | **3** |
| **select**  **instr('abcd','ef')**  **from dual** | **0** |
| **select instr('abcd','bce')**  **from dual** | **0** |
| **select instr('ababababababab','ab',4,2)**  **from dual** | **7**  ***Explicaţie.***Se începe căutarea din poziţia a patra, adică în zona subliniată cu o linie, şi se afişează poziţia de start a celei de a doua apariţii, (subşirul subliniat cu linie dublă) |
| **select**  **instr('abababababab','ab',-4,1)**  **from dual** | **9** |

  **LENGTH(sir)** – returnează numărul de caractere din şirul **sir**.

|  |  |
| --- | --- |
| **Exemplu** | **Rezultatul afişat** |
| **select length('abcd')**  **from dual** | **4** |

  **LPAD(sir1,nr,sir2)** – completează şirul **sir1** la stânga cu caracterele din şirul **sir2** până ce şirul obţinut va avea lungimea **nr**.

Dacă lungimea şirului **sir1** este mai mare decât **nr**, atunci funcţia va realiza trunchierea şirului **sir1**, ştergându-se caracterele de la sfârşitul şirului.

|  |  |
| --- | --- |
| **Exemplu** | **Rezultatul afişat** |
| **select lpad('abcd',3,'\*')**  **from dual** | **abc** |
| **select lpad('abcd',10,'\*.')**  **from dual** | **\*.\*.\*.abcd** |
| **select lpad('abc',10,'\*.')**  **from dual** | **\*.\*.\*.\*abc** |
| **select lpad('abc',5,'xyzw')**  **from dual** | **xyabc** |

  **RPAD(sir,nr,subsir)** – similară cu funcţia **LPAD**, completarea făcându-se la dreapta.

|  |  |
| --- | --- |
| **Exemplu** | **Rezultatul afişat** |
| **select rpad('abcd',3,'\*')**  **from dual** | **abc** |
| **select rpad('abcd',10,'\*.')**  **from dual** | **abcd\*.\*.\*.** |
| **select rpad('abc',10,'\*.')**  **from dual** | **abc\*.\*.\*.\*** |
| **select rpad('abc',5,'xyzw')**  **from dual** | **abcxy** |

  **TRIM(LEADING ch FROM sir)**

**TRIM(TRAILING ch FROM sir)**

**TRIM(BOTH ch FROM sir)**

**TRIM(sir)**

**TRIM(ch FROM sir)**

-      funcţia **TRIM** şterge caracterele **ch** de la începutul, sfârşitul sau din ambele părţi ale şirului **sir**.

-      în ultimele două formate ale funcţiei este subînţeleasă opţiunea **BOTH**.

-      dacă **ch** nu este specificat se vor elimina spaţiile inutile de la începutul, sfârşitul sau din ambele părţi ale şirului **sir**.

|  |  |
| --- | --- |
| **Exemplu** | **Rezultatul afişat** |
| **select**  **trim(leading 'a' from 'aaxaxaa')**  **from dual** | **xaxaa** |
| **select**  **trim(trailing 'a' from 'aaxaxaa')**  **from dual** | **aaxax** |
| **select**  **trim(both 'a' from 'aaxaxaa')**  **from dual** | **xax** |
| **select**  **trim('a' from 'aaxaxaa')**  **from dual** | **xax** |
| **select '\*'||trim('** **abc** **')||'\*'**  **from dual** | **\*abc\*** |

  **REPLACE(sir,subsir,sirnou)** - înlocuieşte toate apariţiile subşirului **subsir** din şirul **sir** cu şirul **sirnou**. Dacă nu este specificat noul şir, toate apariţiile subşirului **subsir** se vor elimina.

|  |  |
| --- | --- |
| **Exemplu** | **Rezultatul afişat** |
| **select replace('abracadabra','ab','xy')**  **from dual** | **xyracadxyra** |
| **select replace('abracadabra','ab','xyz')**  **from dual** | **xyzracadxyzra** |
| **select replace('abracadabra','a')**  **from dual** | **brcdbr** |

## Combinarea funcţiilor asupra şirurilor de caractere

Evident într-o expresie pot fi folosite două sau mai multe astfel de funcţii, imbricate ca în următorul exemplu.

**SELECT substr('abcabcabc',1,instr('abcabcabc','bc')-1)||**

 **'xyz' ||**

 **substr('abcabcabc',instr('abcabcabc','bc')+length('bc'))**

**FROM dual**

Să analizăm pe această comandă

**instr('abcabcabc','bc')**

retunează poziţia primei apariţii a şirului '**bc**' în şirul '**abcabcabc**', adică **2**. Primul apel al funcţiei **substr** este deci echivalent cu apelul

**substr('abcabcabc',1,1)**

adică extrage doar prima litera '**a**'. Al doilea apel al funcţiei substr este echivalent cu

**substr('abcabcabc',4)**

adică extrage toate caracterele de la poziţia 4 până la sfârşitul şirului, deci '**abcabc**'. Aşadar cele două apeluri extrag subşirul de dinaintea primei apariţii a lui '**bc**'  în şirul '**abcabcabc**', şi respectiv de după această apariţie. Cele două secvenţe se concatenează apoi între ele incluzându-se şirul '**xyz**'. În concluzie comanda înlocuieşte prima apariţie a şirului '**bc**' din şirul '**abcabcabc**' cu şirul '**xyz**'.

# II.2.4. Funcţii numerice

Aceste funcţii operează asupra valorilor numerice şi returnează un rezultat numeric. Funcţiile numerice oferite de Oracle sunt destul de puternice.

  **ABS(n)** – returnează valoarea absolută a argumentului.

|  |  |
| --- | --- |
| **Exemplu** | Rezultatul afişat |
| **select abs(-5.23) from dual** | **5.23** |
| **select abs(5) from dual** | **5** |

  **ACOS(n), ASIN(n), ATAN(n)** – sunt funcţiile trigonometrice inverse, cu semnificaţia din matematică. Valoarea returnată de aceste funcţii este exprimată în radiani.

  **SIN(n), COS(n), TAN(n)** – sunt funcţiile trigonometrice cu aceeaşi semnificaţie ca şi la matematică. Argumentul acestor funcţii trebuie precizat în radiani.

|  |  |
| --- | --- |
| **Exemplu** | **Rezultatul afişat** |
| **select sin(3.1415/2) from dual** | **.999999998926914037495206086034346145374** |
| **select cos(3.1415/2) from dual** | **.00004632679488004835355670590049419594** |

  **POWER(m,n)** – calculează valoarea http://paulierco.ro/wp-content/uploads/2008/02/sql_files/image026.gif.

|  |  |
| --- | --- |
| **Exemplu** | **Rezultatul afişat** |
| **select power(2,5) from dual** | **32** |
| **select power(2,0.5) from dual** | **1.41421356237309504880168872420969807855** |
| **select power(2,-1) from dual** | **.5** |
| **select power(2,-0.75) from dual** | **.594603557501360533358749985280237957651** |

  **SQRT(x)** – calculează rădăcina pătrată a argumentului. Apelul **SQRT(x)** returnează aceeaşi valoare ca şi **POWER(x,0.5)**.

|  |  |
| --- | --- |
| **Exemplu** | **Rezultatul afişat** |
| **select sqrt(3) from dual** | **1.73205080756887729352744634150587236694** |

  **REMAINDER(x,y)** – funcţia determină mai întâi acel multiplu a lui **y** care este cel mai apropiat de **x**, şi returnează apoi diferenţa dintre **x** şi acel multiplu.

| **Exemplu** | **Rezultatul afişat** |
| --- | --- |
| **select remainder(10,3)**  **from dual** | **1**  ***Explicaţie.***Cel mai apropiat  de **10** multiplu a lui **3** este **9**. **10-9=1**. |
| **select remainder(5,3)**  **from dual** | **-1**  ***Explicaţie.***Cel mai apropiat de **5** multiplu a lui **3** este  **6**, iar **5-6=-1**. |
| **select remainder(10,3.5)**  **from dual** | **-0.5**  ***Explicaţie.***Cel mai apropiat de **10** multiplu a lui **3.5** este**10.5**, iar **10-10.5=-0.5**. |
| **select remainder(-10,3.5)**  **from dual** | **0.5**  ***Explicaţie.***Cel mai apropiat de **-10** multiplu a lui **3.5** este**-10.5**, iar  **-10-(-10.5)=0.5**. |

  **MOD(x,y)** – dacă cei doi parametrii sunt numere întregi, atunci funcţia returnează acelaşi rezultat ca şi funcţia **REMAINDER**, adică restul împărţirii lui **x** la **y**. Teorema împărţirii cu rest este extinsă de această funcţie şi pentru numerele reale. Adică se ţine cont de relaţia

**x=y \* cât + rest**

unde restul trebuie să fie în modul strict mai mic decât **y**.

|  |  |
| --- | --- |
| **Exemplu** | **Rezultatul afişat** |
| **select mod(10,3)**  **from dual** | **1**  ***Explicaţie.*** **10=3\*3+1**. |
| **select mod(5,3)**  **from dual** | **2**  ***Explicaţie.*****5=3\*1+2** |
| **select mod(10,3.5)**  **from dual** | **3**  ***Explicaţie.*****10=3.5\*2+3**. |
| **select mod(-10,3.5)**  **from dual** | **-3**  ***Explicaţie.*****-10=3.5\*(-2)-3**. |
| **select mod(-10,-3.5)**  **from dual** | **-3**  ***Explicaţie.*****-10=-3.5\*2-3**. |
| **select mod(10,-3.5)**  **from dual** | **3**  ***Explicaţie.*****10=-3.5\*(-2)+3**. |

Se observă din exemplele anterioare că restul are întotdeauna acelaşi semn cu primul parametru.

  **SIGN(x)** – returnează semnul lui **x**, adică **1** dacă **x** este număr pozitiv, respectiv **-1** dacă **x** este număr negativ.

  **CEIL(x)** – returnează cel mai mic număr întreg care este mai mare sau egal decât parametrul transmis.

  **FLOOR(x)** – returnează cel mai mare număr întreg care este mai mic sau egal decât parametrul transmis.

| **Exemplu** | **Rezultatul afişat** |
| --- | --- |
| **select ceil(3) from dual** | **3** |
| **select ceil(-3) from dual** | **-3** |
| **select ceil(-3.7) from dual** | **-3** |
| **select ceil(3.7) from dual** | **4** |
| **select floor(3) from dual** | **3** |
| **select floor(-3) from dual** | **-3** |
| **select floor(-3.7) from dual** | **-4** |
| **select floor(3.7) from dual** | **3** |

  **ROUND(x,y)** – rotunjeşte valoarea lui **x** la un număr de cifre precizat prin parametrul **y**.

Dacă al doilea parametru este un număr pozitiv, atunci se vor păstra din **x** primele **y** zecimale, ultima dintre aceste cifre fiind rotunjită, în funcţie de următoarea zecimală.

Al doilea argument poate fi o valoare negativă, rotunjirea făcându-se la stânga punctului zecimal. Cifra a **|y|+1** din faţa punctului zecimal (numărând de la punctul zecimal spre stânga începând cu **1**) va fi rotunjită în funcţie de cifra aflată imediat la dreapta ei. Primele **|y|** cifre din stânga punctului zecimal vor deveni **0**.

Cel de al doilea argument este opţional, în cazul în care nu se precizează, este considerată implicit valoarea **0**.

|  |  |
| --- | --- |
| **Exemplu** | **Rezultatul afişat** |
| **select round(745.123,2) from dual** | **745.12** |
| **select round(745.126,2) from dual** | **745.13** |
| **select round(745.126,-1)**  **from dual** | **750** |
| **select round(745.126,-2)**  **from dual** | **700** |
| **select round(745.126,-3)**  **from dual** | **1000** |
| **select round(745.126,-4)**  **from dual** | **0** |
| **select round(745.126,0)**  **from dual** | **745** |
| **select round(745.826,0)**  **from dual** | **746** |
| **select round(745.826)**  **from dual** | **746** |

  **TRUNC(x)** – este asemănătoare cu funcţia **ROUND**, fără a rotunji ultima cifră.

|  |  |
| --- | --- |
| **Exemplu** | **Rezultatul afişat** |
| **select trunc(745.123,2) from dual** | **745.12** |
| **select trunc(745.126,2) from dual** | **745.12** |
| **select trunc(745.126,-1)**  **from dual** | **740** |
| **select trunc(745.126,-2)**  **from dual** | **700** |
| **select trunc(745.126,-3)**  **from dual** | **0** |
| **select trunc(745.126,-4)**  **from dual** | **0** |
| **select trunc(745.126,0)**  **from dual** | **745** |
| **select trunc(745.826,0)**  **from dual** | **745** |
| **select trunc(745.826) from dual** | **745** |

# II.2.5. Funcţii asupra datelor calendaristice

Una dintre caracteristicile importante ale Oracle este abilitatea de a memora şi opera cu date calendaristice. Tipurile de date calendaristice recunoscute de Oracle sunt:

  **DATE** - valorile având acest tip sunt memorate într-un format intern specific, care include pe lângă ziua, luna şi anul, de asemenea ora, minutul, şi secunda.

  **TIMESTAMP** – valorile având acest tip memorează data calendaristică, ora, minutul şi secunda dar şi fracţiunea de secundă.

  **TIMESTAMP WITH [LOCAL] TIME ZONE** – este similar cu **TIMESTAMP**, însă se va memora şi diferenţa de fus orar faţă de ora universală, a orei de pe serverul bazei de date, sau a aplicaţiei client, în cazul în care se include opţiunea **LOCAL**.

  **INTERVAL YEAR TO MONTH** – memorează o perioadă de timp în ani şi luni.

  **INTERVAL DAY TO SECOND** – memorează un interval de timp în zile, ore, minute şi secunde.

Să exemplificăm aceste tipuri de date creând o tabelă de test cu comanda:

**create table test3**

 **(data1 DATE,**

**data2 TIMESTAMP(5),**

 **data3 TIMESTAMP(5) WITH TIME ZONE,**

 **data4 TIMESTAMP(5) WITH LOCAL TIME ZONE)**

Vom insera acum o linie nouă în această tabelă:

**insert into test3**

 **values(sysdate,systimestamp,systimestamp,systimestamp)**

şi la afişarea tabelei

**select \* from test3**

vom obţine rezultatul din figura II.2.3.

|  |  |  |  |
| --- | --- | --- | --- |
| **DATA1** | **DATA2** | **DATA3** | **DATA4** |
| 27-FEB-07 | 27-FEB-07 05.49.35.02886 AM | 27-FEB-07 05.49.35.02886 AM -06:00 | 27-FEB-07 11.49.35.02886 AM |

**Figura II.2.3**

## Aritmetica datelor calendaristice

Oracle ştie să realizeze operaţii aritmetice asupra datelor calendaristice, astfel adăugarea valorii **1** la o dată calendaristică, va duce la obţinerea următoarei date calendaristice:

**SELECT sysdate, sysdate+5, sysdate-70 from dual**

|  |  |  |
| --- | --- | --- |
| **SYSDATE** | **SYSDATE+5** | **SYSDATE-70** |
| 21-APR-07 | 26-APR-07 | 10-FEB-07 |

**Figura II.2.4.** Adunarea unui număr întreg la o dată calendaristică

De asemenea se poate face diferenţa dintre două date calendaristice, obţinându-se numărul de zile dintre cele două date:

**SELECT first\_name, last\_name,**

**hire\_date, sysdate-hire\_date**

**FROM employees**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIRST\_NAME** | **LAST\_NAME** | **HIRE\_DATE** | **SYSDATE-HIRE\_DATE** |
| Steven | King | 17-JUN-87 | 7248.18565972222222222222222222222222222 |
| Neena | Kochhar | 21-SEP-89 | 6421.18565972222222222222222222222222222 |
| Lex | De Haan | 13-JAN-93 | 5211.18565972222222222222222222222222222 |
| Alexander | Hunold | 03-JAN-90 | 6317.18565972222222222222222222222222222 |
| … | … | … | … |

**Figura II.2.5.** Diferenţa dintre două date calendaristice

Deşi implicit o dată calendaristică de tip **DATE** nu este afişată în format complet (nu se afişează ora, minutul, secunda), în tabelă se memorează complet. De aceea poate fi uneori derutant rezultatul unor operaţii aritmetice cu date calendaristice, după cum se vede în figura II.2.6. în care diferenţa dintre ziua de astăzi şi cea de ieri este de **1.187997**….

**SELECT sysdate-TO\_DATE('20-APR-07','dd-MON-yy') FROM dual**

|  |
| --- |
| **SYSDATE-TO\_DATE('20-APR-07','DD-MON-YY')** |
| 1.18799768518518518518518518518518518519 |

**Figura II.2.6.**

De ce se obţine acest lucru? Simplu, data de **20 aprilie** a fost precizată fără oră, aşadar a fost considerată implicit ora **00:00**. Iar **sysdate** ne-a furnizat data curentă incluzând şi ora. Aşadar de ieri de la ora **00:00** până astăzi la ora **12:32** a trecut mai mult de o zi.

## Funcţii cu date calendaristice

Oracle oferă un număr foarte mare de funcţii care operează asupra datelor calendaristice, dar în cele ce urmează ne vom opri asupra celor mai importante dintre acestea.

  **SYSDATE** – returnează data şi ora curentă a serverului bazei de date.

  **CURRENT\_DATE** – returnează data şi ora curentă a aplicaţiei client. Aceasta poate să difere de data bazei de date.

  **SYSTIMESTAMP** – returnează data în formatul **TIMESTAMP**.

**select CURRENT\_DATE, sysdate, systimestamp**

**from dual**

|  |  |  |
| --- | --- | --- |
| **CURRENT\_DATE** | **SYSDATE** | **SYSTIMESTAMP** |
| 21-APR-07 | 21-APR-07 | 21-APR-07 04.33.32.445081 AM -05:00 |

**Figura II.2.7.** Funcţiile **SYSDATE**, **CURRENT\_DATE** şi **SYSTIMESTAMP**

  **ADD\_MONTHS(data,nrluni)** – adaugă un număr de luni la data curentă. Dacă al doilea parametru este un număr negativ, se realizează de fapt scăderea unui număr de luni din data precizată.

|  |  |
| --- | --- |
| **Exemplu** | **Rezultatul afişat** |
| **select sysdate,**  **ADD\_MONTHS(sysdate,2) from dual** | **27-FEB-07** **27-APR-07** |
| **select**  **sysdate, ADD\_MONTHS(sysdate,-2)**  **from dual** | **27-FEB-07** **27-DEC-07** |

  **MONTHS\_BETWEEN(data1,data2)** – determină numărul de luni dintre două date calendaristice precizate. Rezultatul returnat poate fi un număr real (vezi figura II.2.8). Dacă prima dată este mai mică (o dată mai veche) atunci rezultatul va un număr negativ.

**select sysdate, hire\_date,**

 **MONTHS\_BETWEEN(sysdate, hire\_date),**

 **MONTHS\_BETWEEN(hire\_date, sysdate)**

**from employees**

|  |  |  |  |
| --- | --- | --- | --- |
| **SYSDATE** | **HIRE\_DATE** | **MONTHS\_BETWEEN(SYSDATE,HIRE\_DATE)** | **MONTHS\_BETWEEN(HIRE\_DATE,SYSDATE)** |
| 21-APR-07 | 17-JUN-87 | 238.135216173835125448028673835125448029 | -238.135216173835125448028673835125448029 |
| 21-APR-07 | 21-SEP-89 | 211 | -211 |
| 21-APR-07 | 13-JAN-93 | 171.264248431899641577060931899641577061 | -171.264248431899641577060931899641577061 |
| 21-APR-07 | 03-JAN-90 | 207.586829077060931899641577060931899642 | -207.586829077060931899641577060931899642 |
| 21-APR-07 | 21-MAY-91 | 191 | -191 |
| … | … | … | … |

**Figura II.2.8.** Funcţia **MONTHS\_BETWEEN**

  **LEAST(data1,data2,…)** – determină cea mai veche (cea mai mică) dată dintre cele transmise ca parametru.

  **GREATEST(data1,data2,…)** – determină cea mai recentă (cea mai mare) dată dintre cele transmise ca parametru.

**select hire\_date,sysdate,**

 **least(hire\_date,sysdate),greatest(hire\_date,sysdate)**

**from employees**

|  |  |  |  |
| --- | --- | --- | --- |
| **HIRE\_DATE** | **SYSDATE** | **LEAST(HIRE\_DATE,SYSDATE)** | **GREATEST(HIRE\_DATE,SYSDATE)** |
| 17-JUN-87 | 21-APR-07 | 17-JUN-87 | 21-APR-07 |
| 21-SEP-89 | 21-APR-07 | 21-SEP-89 | 21-APR-07 |
| 13-JAN-93 | 21-APR-07 | 13-JAN-93 | 21-APR-07 |
| 03-JAN-90 | 21-APR-07 | 03-JAN-90 | 21-APR-07 |
| 21-MAY-91 | 21-APR-07 | 21-MAY-91 | 21-APR-07 |
| … | … | … | … |

**Figura II.2.9.** Funcţiile **LEAST** şi **GEATEST**

  **NEXT\_DAY(data, 'ziua')** – returnează următoarea dată de **'ziua'** de după data transmisă ca parametru, unde **'ziua'** poate fi **'Monday'**, **'Tuesday'** etc. În exemplele care urmează data curentă este considerată  ziua de marţi, 27 februarie 2007.

  **LAST\_DAY(data)** – returnează ultima zi din luna din care face parte data transmisă ca parametru.

|  |  |
| --- | --- |
| **Exemplu** | **Rezultatul afişat** |
| **select next\_day(sysdate,'Friday')**  **from dual** | **02-MAR-07** |
| **select next\_day(sysdate,'TUESDAY')**  **from dual** | **06-MAR-07**  ***Explicaţie.***Chiar dacă ziua curentă este o zi de marţi, funcţia va returna următoarea zi de marţi. |
| **select last\_day(sysdate)**  **from dual** | **28-FEB-07** |
| **select last\_day(sysdate+20)**  **from dual** | **31-MAR-07** |
| **select last\_day(ADD\_MONTHS(sysdate,12))**  **from dual** | **29-FEB-07**  ***Explicaţie.***Ziua returnată de **sysdate** este **27-FEB-07**, la care adăugăm **12** luni, deci obţinem data de **27-FEB-08**, iar anul **2008** este un an bisect de aceea ultima zi din lună este **29-FEB-08**. |

  **ROUND(data,'format')** – dacă nu se precizează formatul, funcţia rotunjeşte data transmisă ca parametru la cea mai apropiată oră **12 AM**, adică dacă ora memorată în **data** este înainte de miezul zilei atunci se va returna ora **12 AM** a datei transmise. Dacă ora memorată în **data** este după miezul zilei se va returna ora **12 AM** a zilei următoare.

**select to\_char(sysdate,'dd-MON-YY hh:mi AM'),**

 **round(sysdate) from dual**

|  |  |
| --- | --- |
| **TO\_CHAR(SYSDATE,'DD-MON-YYHH:MIAM')** | **ROUND(SYSDATE)** |
| 21-APR-07 04:41 AM | 21-APR-07 |

**Figura II.2.10.** Funcţia **ROUND**

În cazul în care este specificat formatul, data va fi rotunjită conform formatului indicat. Câteva dintre formatele cele mai uzuale sunt:

                      **y**, **yy**, **yyyy**, **year** – se rotunjeşte data la cea mai apropiată dată de 1 Ianuarie. Dacă data este înainte de 1 iulie, se va returna data de 1 ianuarie a aceluiaşi an. Dacă data este după data de 1 iulie se va returna data de 1 ianuarie a anului următor.

                      **mm**, **month** – rotunjeşte data la cel mai apropiat început de lună. Orice dată calendaristică aflată după data de **16**, inclusiv, este rotunjită la prima zi a lunii următoare.

                      **ww**, **week** – se rotunjeşte data la cel mai apropiat început de săptămână. Prima zi a săptămânii este considerată lunea. Pentru datele aflate după ziua de joi, inclusiv, se va returna ziua de luni a săptămânii următoare.

|  |  |
| --- | --- |
| **Exemplu** | **Rezultatul afişat** |
| select sysdate,         round(sysdate,'year'),  round(ADD\_MONTHS(sysdate,5),'year')  from dual | 27-FEB-07  01-JAN-07  01-JAN-08 |
| select sysdate,         round(sysdate,'mm'),         round(sysdate+16,'mm'),         round(sysdate+17,'mm')  from dual | 27-FEB-07  01-MAR-07  01-MAR-07  01-APR-07 |
| select sysdate,         round(sysdate,'ww'),         round(sysdate+1,'ww'),         round(sysdate+2,'ww')  from dual | 27-FEB-07  26-FEB-07  26-FEB-07  05-FEB-07 |

  **TRUNC(data,'format')** – trunchează data specificată conform formatului specificat. Se pot folosi aceleaşi formate ca şi în cazul funcţiei **ROUND**.

|  |  |
| --- | --- |
| **Exemplu** | **Rezultatul afişat** |
| select sysdate,         trunc(sysdate,'year'),  trunc(ADD\_MONTHS(sysdate,5),'year')  from dual | 27-FEB-07  01-JAN-07  01-JAN-07 |
| select sysdate,         trunc(sysdate,'month'),         trunc(sysdate+16,'month'),         trunc(sysdate+17,'month')  from dual | 27-FEB-07  01-FEB-07  01-MAR-07  01-MAR-07 |
| select sysdate,         trunc(sysdate,'ww'),         trunc(sysdate+1,'ww'),         trunc(sysdate+2,'ww')  from dual | 27-FEB-07  26-FEB-07  26-FEB-07  26-FEB-07 |

# II.2.6. Funcţii de conversie

Oracle oferă un set bogat de funcţii care vă permit să transformaţi o valoare dintr-un tip de dată în altul.

## Transformarea din dată calendaristică în şir de caractere

Transformarea unei date calendaristice în şir de caractere se poate realiza cu ajutorul funcţiei **TO\_CHAR**. Această operaţie se poate dovedi utilă atunci când dorim obţinerea unor rapoarte cu un  format precis. Sintaxa acestei funcţii este:

**TO\_CHAR (dt, format)**

**dt** poate avea unul din tipurile pentru date calendatistice (**DATE**, **TIMESTAMP**, **TIMESTAMP WITH TIME ZONE**, **TIMESTAMP WITH LOCAL TIME ZONE**, **INTERVAL MONTH TO YEAR**, or **INTERVAL DAY TO SECOND**). Formatul poate conţine mai mulţi parametrii care pot afecta modul în care va arăta şirul returnat. Câţiva din aceşti parametrii sunt prezentaţi în continuare.

| **Aspect** | **Parametru** | **Descriere** | **Exemplu** |
| --- | --- | --- | --- |
| Secolul | **CC** | Secolul cu două cifre | **21** |
| Trimestrul | **Q** | Trimestrul din an în care se găseşte data | **3** |
| Anul | **YYYY**, **RRRR** | Anul cu patru cifre. | **2006** |
|  | **YY**, **RR** | Ultimele două cifre din an. | **06** |
|  | **Y** | Ultima cifră din an | **6** |
|  | **YEAR**, **Year** | Numele anului | **TWO THOUSAND-SIX,**  **Two Thousand-Six** |
| Luna | **MM** | Luna cu două cifre | **02** |
|  | **MONTH**, **Month** | Numele complet al lunii. | **JANUARY**,  **January** |
|  | **MON**, **Mon** | Primele trei litere ale denumirii lunii. | **JAN, Jan** |
|  | **RM** | Luna scrisă cu cifre romane. | **IV** |
| Săptămâna | **WW** | Numărul săptămânii din an. | **35** |
|  | **W** | Ultima cifră a numărului săptămânii din an. | **2** |
| Ziua | **DDD** | Numărul zilei din cadrul anului. | **103** |
|  | **DD** | Numărul zilei în cadrul lunii | **31** |
|  | **D** | Numărul zilei în cadrul săptămânii. | **5** |
|  | **DAY**, **Day** | Numele complet al zilei din săptămână | **SATURDAY,** **Saturday** |
|  | **DY**, **Dy** | Prescurtarea denumirii zilei din săptămână. | **SAT**, **Sat** |
| Ora | **HH24** | Ora în formatul cu **24** de ore. | **23** |
|  | **HH** | Ora în formatul cu **12** ore. | **11** |
| Minutele | **MI** | Minutele cu două cifre | **57** |
| Secundele | **SS** | Secundele cu două cifre | **45** |
| Sufixe | **AM** sau **PM** | **AM** sau **PM** după cum e cazul. | **AM** |
|  | **A.M.** sau **P.M.** | **A.M.** sau **P.M.** după cum e cazul. | **P.M.** |
|  | **TH** | Sufix pentru numerale (**th** sau **nd** sau **st**) |  |
|  | **SP** | Numerele sunt scrise în cuvinte. |  |

În cadrul formatului se pot folosi oricare dintre următorii separatori

**- / , . ; :**

Dacă în şirul returnat dorim să includem şi anumite texte acestea se vor include între ghilimele.

Iată în continuare şi câteva exemple de folosire a acestei funcţii.

| **Exemplu** | **Rezultatul afişat** |
| --- | --- |
| **select sysdate,**  **to\_char(sysdate,'MONTH DD, YYYY')**  **to\_char(sysdate,'Month DD, YYYY')**  **to\_char(sysdate,'Mon DD, YYYY')**  **from dual** | **28-FEB-07**  **FEBRUARY 28, 2007**  **February 28, 2007**  **Feb 28, 2007** |
| **select**  **to\_char(sysdate,'"Trimestrul "Q "al anului " Year')**  **from dual** | **Trimestrul 1 al anului Two Thousand Seven** |
| **select**  **to\_char(sysdate,'"Secolul "CC')**  **from dual** | **Secolul 21** |
| **select**  **to\_char(sysdate,'Day, dd.RM.YYYY') from dual** | **Wednesday, 28.II.2007** |
| **select**  **to\_char(sysdate,'Dy, D, DD, DDD')**  **from dual** | **Wed, 4, 28, 059** |
| **select to\_char(sysdate,'HH24:MI/HH:MI AM')**  **from dual** | **21:53/09:53 PM** |
| **select to\_char(sysdate+1,'ddth')**  **from dual** | **01st** |
| **select to\_char(sysdate+1,'ddspth')**  **from dual** | **First** |
| **select to\_char(sysdate+2,'Ddspth')**  **from dual** | **Second** |
| **select to\_char(sysdate+10,'DDspth')**  **from dual** | **TENTH** |
| **select to\_char(sysdate,'mmsp')**  **from dual** | **Two** |

## Transformarea din şir de caractere în dată calendaristică

Folosind funcţia **TO\_DATE** se poate transforma un şir de caractere precum **'May 26, 2006'** într-o dată calendaristică. Sintaxa funcţiei este:

**TO\_DATE(sir,format)**

Formatul nu este obligatoriu, însă dacă nu este precizat, şirul trebuie să respecte formatul implicit al datei calendaristice **DD-MON-YYYY** sau **DD-MON-YY**. Formatul poate folosi aceiaşi parametrii de format ca şi funcţia **TO\_CHAR**.

| **Exemplu** | **Rezultatul afişat** |
| --- | --- |
| **select**  **to\_date('7.4.07', 'MM.DD.YY')**  **from dual;** | **04-JUL-07** |
| **select to\_date('010101','ddmmyy')**  **from dual** | **01-JAN-01** |

## Formatul RR şi formatul YY

Aşa cum s-a precizat anterior în formatarea unei date calendaristice se pot folosi pentru an atât **YY** (respectiv **YYYY**) cât şi **RR** (respectiv **RRR**). Diferenţa dintre aceste două formate este modul în care ele interpretează anii aparţinând de secole diferite. Oracle memorează toate cele patru cifre ale unui an, dar dacă sunt transmise doar două din aceste cifre, Oracle va interpreta secolul diferit în cazul celor două formate.

Vom începe printr-un exemplu:

**select to\_char(to\_date('05-FEB-95','DD-MON-YY'),**

 **'DD-MON-YYYY') as "YY Format",**

 **to\_char(to\_date('05-FEB-95','DD-MON-RR'),**

 **'DD-MON-RRRR') as "RR Format"**

**from dual**

|  |  |
| --- | --- |
| **YY Format** | **RR Format** |
| 05-FEB-2095 | 05-FEB-1995 |

**Figura II.2.11.** Formatele **YY** şi **RR**

Se observă modul diferit de interpretare a anului.

Dacă utilizaţi formatul **YY** şi anul este specificat doar prin două cifre, se presupune că anul respectiv face parte din acelaşi secol cu anul curent. De exemplu, dacă anul transmis este **15**iar anul curent este **2007**, atunci anul transmis este interpretat cu **2015**. De asemenea **75** interpretat ca **2075**.

**select to\_char(to\_date('15','yy'),'yyyy'),**

 **to\_char(to\_date('75','yy'),'yyyy')**

**from dual**

|  |  |
| --- | --- |
| **TO\_CHAR(TO\_DATE('15','YY'),'YYYY')** | **TO\_CHAR(TO\_DATE('75','YY'),'YYYY')** |
| 2015 | 2075 |

**Figura II.2.12.** Formatul **YY**

Dacă folosiţi formatul **RR** şi anul transmis este de două cifre, primele două cifre ale anului transmis este determinat în funcţie de cele două cifre transmise şi de ultimele două cifre ale anului curent. Regulile după care se determină secolul datei transmise sunt următoarele:

**Regula 1:**Dacă anul transmis este între **00** şi **49**, şi ultimele două cifre ale anului curent sunt între **00** şi **49** atunci secolul este acelaşi cu secolul anului curent. De exemplu dacă anul transmis este **15** iar anul curent este **2007**, anul transmis este interpretat ca fiind **2015**.

**Regula 2:**Dacă anul transmis este între **50** şi **99** iar anul curent este între **00** şi **49** atunci secolul este secolul prezent minus **1**. De exemplu dacă transmiteţi **75** iar anul curent este**2007**, anul transmis este interpretat ca fiind **1975**.

**Regula 3:**Dacă anul transmis este între **00** and **49** iar anul prezent este între **50** şi **99**, secolul este considerat secolul prezent plus 1. De exemplu dacă aţi transmis anul **15** iar anul curent este **1987**, anul transmis este considerat ca fiind anul **2015**.

**Regula 4:**Dacăanul transmis este între **50** şi **99**, iar anul curent este între **50** şi **99**, secolul este acelaşi cu a anului curent. De exemplu, dacă transmiteţi anul **55** iar anul prezent ar fi**1987**, atunci anul transmis este considerat ca fiind anul **1955**.

**select to\_char(to\_date('04-JUL-15','DD-MON-RR'),**

 **'DD-MON-YYYY') as dt1,**

 **to\_char(to\_date('04-JUL-75','DD-MON-RR'),**

 **'DD-MON-YYYY') as dt2**

**from dual**

|  |  |
| --- | --- |
| **DT1** | **DT2** |
| 04-JUL-2015 | 04-JUL-1975 |

**Figura II.2.13.** Formatul **RR**

## Transformarea din număr în şir de caractere

Pentru a transforma un număr într-un şir de caractere, se foloseşte funcţia **TO\_CHAR**, cu următoarea sintaxă:

**TO\_CHAR(numar,format)**

Formatul poate conţine unul sau mai mulţi parametrii de formatare dintre cei prezentaţi în tabelul următor.

| **Parametru** | **Exemplu de format** | **Descriere** |
| --- | --- | --- |
| **9** | **999** | Returnează cifrele numărului din poziţiile specificate, precedat de semnul minus dacă numărul este negativ |
| **0** | **0999** | Completează cifrele numărului cu zerouri în faţă |
| **.** | **999.99** | Specifică poziţia punctului zecimal |
| **,** | **9,999** | Specifică poziţia separatorului virgulă |
| **$** | **$999** | Afişează semnul dolar |
| **EEEE** | **9.99EEEE** | Returnează scrierea ştiinţifică a numărului. |
| **L** | **L999** | Afişează simbolul monetar. |
| **MI** | **999MI** | Afişează semnul minus după număr dacă acesta este negativ. |
| **PR** | **999PR** | Numerele negative sunt închise între paranteze unghiulare. |
| **RN**  **rn** | **RN**  **rn** | Afişează numărul în cifre romane. |
| **V** | **99V99** | Afişează numărul înmulţit cu **10** la puterea **x**, şi rotunjit la ultima cifră, unde **x** este numărul de cifre **9** de după **V**. |
| **X** | **XXXX** | Afişează numărul în baza **16**.. |

Vom exemplifica în continuare câteva dintre aceste formate.

| **Exemplu** | **Rezultatul afişat** |
| --- | --- |
| **select to\_char(123.45,'9999.99')**  **from dual** | **123.45** |
| **select to\_char(123.45,'0000.000')**  **from dual** | **0123.450** |
| **select to\_char(123.45,'9.99EEEE')**  **from dual** | **1.23E+02** |
| **select to\_char(-123.45,'999.999PR')**  **from dual** | **<123.450>** |
| **select to\_char(1.2373,'99999V99') from dual** | **124** |
| **select to\_char(1.2373,'L0000.000')**  **from dual** | **$0001.237** |
| **select to\_char(4987,'XXXXXX')**  **from dual** | **137B** |
| **select to\_char(498,'RN') from dual** | **CDXCVIII** |

## Transformarea şir de caractere în număr

Transformarea inversă din şir de caractere într-o valoare numerică se realizează cu ajutorul funcţiei **TO\_NUMBER**:

**TO\_NUMBER(sir,format)**

Parametrii de formatare ce se pot folosi sunt aceeaşi ca în cazul funcţiei **TO\_CHAR**. Iată câteva exemple.

| **Exemplu** | **Rezultatul afişat** |
| --- | --- |
| **select to\_number('970.13') + 25.5**  **FROM dual** | **995.63** |
| **select**  **to\_number('-$12,345.67','$99,999.99')**  **from dual;** | **-12345.67** |

# II.2.7. Funcţii de uz general

Pe lângă funcţiile care controlează modul de formatare sau conversie al datelor, Oracle oferă câteva funcţii de uz general, care specifică modul în care sunt tratate valorile **NULL**.

  **NVL(val1,val2)** – funcţia returnează valoarea **val1**, dacă aceasta este nenulă, iar dacă **val1** este **NULL** atunci va returna valoarea **val2**. Funcţia **NVL** poate lucra cu date de tip caracter, numeric sau dată calendaristică, însă este obligatoriu ca cele două valori să aibă acelaşi tip.

**select first\_name, commission\_pct, NVL(commission\_pct,0.8)**

**from employees**

**where employee\_id between 140 and 150**

rezultatul returnat de această comandă este cel din figura II.2.14.

|  |  |  |
| --- | --- | --- |
| **FIRST\_NAME** | **COMMISSION\_PCT** | **NVL(COMMISSION\_PCT,0.8)** |
| Trenna | - | .8 |
| Curtis | - | .8 |
| Randall | - | .8 |
| Peter | - | .8 |
| Eleni | .2 | .2 |

**Figura II.2.14.** Funcţia **NVL**

  **NVL2(val1,val2,val3)** – dacă valoarea **val1** nu este nulă atunci funcţia va returna valoarea **val2**, iar dacă **val1** are valoarea **NULL** atunci funcţia va returna valoarea **val3**(vezi figura II.2.15.).

**select first\_name, commission\_pct,**

**NVL2(commission\_pct,'ARE','NU ARE')**

**from employees where employee\_id between 140 and 150**

|  |  |  |
| --- | --- | --- |
| **FIRST\_NAME** | **COMMISSION\_PCT** | **NVL2(COMMISSION\_PCT,'ARE','NUARE')** |
| Trenna | - | NU ARE |
| Curtis | - | NU ARE |
| Randall | - | NU ARE |
| Peter | - | NU ARE |
| Eleni | .2 | ARE |

**Figura II.2.15** Funcţia **NVL2**

  **NULLIF(expr1,expr2)** – dacă cele două expresii sunt egale, funcţia returnează **NULL**. Dacă valorile celor două expresii sunt diferite atunci funcţia va returna valoarea primei expresii (vezi figura II.2.16.).

**select employee\_id, first\_name, last\_name,**

 **NULLIF(length(first\_name),length(last\_name))**

**from employees where employee\_id between 103 and 142**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **FIRST\_NAME** | **LAST\_NAME** | **NULLIF(LENGTH(FIRST\_NAME),LENGTH(LAST\_NAME))** |
| 103 | Alexander | Hunold | 9 |
| 104 | Bruce | Ernst | - |
| 107 | Diana | Lorentz | 5 |
| 124 | Kevin | Mourgos | 5 |
| 141 | Trenna | Rajs | 6 |
| 142 | Curtis | Davies | - |
| … | … | … | … |

**Figura II.2.16** Funcţia **NULLIF**

  **COALESCE(expr1, expr2, ..., exprn)** – funcţia returnează valoarea primei expresii nenule (vezi figura II.2.17).

**select coalesce(null, null, '33', 'test') from dual**

|  |
| --- |
| **COALESCE(NULL,NULL,'33','TEST')** |
| 33 |

**Figura II.2.17** Funcţia **COALESCE**

# II.2.8 Funcţii şi expresii condiţionale

Oracle SQL oferă posibilitatea de a construi expresii alternative asemănătoare structurilor **IF-THEN-ELSE** prezente în alte limbaje.

  **DECODE(expresie, val11, val12, val21, val22, ..., valn1, valn2, val)** **–** această compară valoarea expresiei cu valorile **val11**, **val21**, ..., **valn1**. Dacă valoarea expresiei este egală cu valoarea **vali1**, atunci funcţia va returna valoarea **vali2**. Dacă funcţia nu este egală cu nici una din valorile **vali1**, atunci funcţia va returna valoarea **val**.

**select DECODE('Maria' ,'Dana', 'Ea este Ana' ,**

 **'Maria','Ea este Maria' ,**

 **'Nu e nici Ana nici Maria')**

**from dual**

această comandă va afişa mesajul “**Ea este Maria**” însă următoarea comandă va afişa “**Nu e nici Ana nici Maria**”.

**select DECODE('Valeria' ,'Dana', 'Ea este Ana' ,**

 **'Maria','Ea este Maria' ,**

 **'Nu e nici Ana nici Maria')**

**from dual**

  În locul funcţiei **DECODE** se poate folosi expresia condiţională **CASE**. Funcţia **CASE** utilizează cuvintele cheie **when**, **then**, **else**, şi **end** pentru a indica ramura selectată. În general orice apel al funcţiei **DECODE** poate fi transcris folosind funcţia **CASE**. Chiar dacă o expresie folosind **CASE** este mai lungă decât expresia echivalentă care foloseşte funcţia **DECODE**, varianta cu **CASE** este mult mai uşor de citit şi greşelile sunt depistate mai uşor. În plus varianta **CASE** este compatibilă **ANSI-SQL**.

Cele două comenzi de mai sus por fi transcrise cu ajutorul funcţiei **CASE** astfel:

**select CASE 'Maria'**

 **WHEN 'Dana' THEN 'Ea este Ana'**

 **WHEN 'Maria' THEN 'Ea este Maria'**

 **ELSE 'Nu e nici Ana nici Maria'**

 **END**

**from dual**

**select CASE 'Valeria'**

 **WHEN 'Dana' THEN 'Ea este Ana'**

 **WHEN 'Maria' THEN 'Ea este Maria'**

 **ELSE 'Nu e nici Ana nici Maria'**

### END

**from dual**