II.2.1. Tipuri de funcţii

Funcțiile Oracle sunt împărțite astfel:

- **Funcții singulare** – acestea operează la un moment dat asupra unei singure înregistrări. Aceste funcții vor fi discutate în acest capitol
- **Funcțiile de grup** – operează asupra unui grup de înregistrări și returnează o singură valoare pentru întregul grup.

Funcțiile singulare pot fi folosite în:

- clauza **SELECT**, pentru a modifica modul de afișare a datelor, pentru a realiza diferite calcule etc.
- clauza **WHERE**, pentru a preciza mai exact care sunt înregistrările ce se afișează
- clauza **ORDER BY**

Funcțiile singulare (single-row functions) pot fi la rândul lor împărțite în:

- Funcții care operează asupra șirurilor de caractere
- Funcții numerice
- Funcții pentru manipularea datelor calendaristice
- Funcții de conversie – care convertesc datele dintr-un tip în altul
- Funcții de uz general.

Unele funcții, precum **TRUNC** și **ROUND** pot acționa asupra mai multor tipuri de date, dar cu semnificații diferite.

II.2.2. Tabela DUAL

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

Această tabela este una specială, care conține o singură coloană numită "**DUMMY**" și o singură linie.

Tabela **DUAL** se folosește 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 componență 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

Șirurile de caractere pot conține orice combinație de litere, numere, spații, și alte simboluri, precum semne de punctuație, sau caractere speciale. În Oracle există două tipuri de date pentru memorarea șirurilor de caractere:

- **CHAR** – pentru memorarea șirurilor de caractere de lungime fixă
- **VARCHAR2** – pentru memorarea șirurilor de caractere având lungime variabilă.

👇**LOWER(sir)** – convertește caracterele alfanumerice din șir în litere mici.

👇**UPPER(sir)** – convertește caracterele alfanumerice din șir în litere mari.

👇**INITCAP(sir)** – convertește la majusculă prima literă din fiecare cuvânt al șirului. Cuvintele sunt șiruri 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 afișat
SELECT LOWER(first_name) **FROM employees;**	afișează 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** ***Explicație*** șirul conține **3** cuvinte **aBc def** și **ghi**

👇**CONCAT(sir1, sir2)** – concatenează două șiruri de caractere

Exemplu	Rezultatul afișat
SELECT CONCAT('abc','def') **FROM dual;**	**abcdef** ***Explicație*** 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 poziția **poz**.

Observații

- dacă din poziția **poz** până la sfârșitul șirului sunt mai puțin de **nr** caractere, se vor extrage toate caracterele de la poziția **poz** până la sfârșitul șirului.
- parametrul **poz** poate fi și o valoare negativă, ceea ce înseamnă că poziția de unde se va începe extragerea caracterelor din șir se va determina numărând caracterele din șir de la dreapta spre stânga (vezi ultimele 3 exemple de mai jos)
- dacă **nr** nu este specificat, se va returna subșirul începând cu caracterul de pe poziția **poz** din șir până la sfârșitul șirului.

Exemplu	Rezultatul afișat
`select substr('abcdef',3,2) from dual`	**cd**
`select substr('abcdef',3,7) from dual`	**cdef** ***Explicație.*** Chiar dacă din poziția 3 până la sfârșitul șirului nu mai sunt 7 caractere se returnează caracterele rămase
`select substr('abcdef',3) from dual`	**cdef** ***Explicație.*** Același rezultat ca mai sus dacă nu se specifică numărul de caractere ce se extrag
`select substr('abcdef',7,3) from dual`	nu se va afișa nimic deoarece nu există poziția **7** în șir, acesta având doar **5** caractere.
`select substr('abcdef',-4,2) from dual`	**cd** ***Explicație.*** 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 afișa nimic deoarece șirul conține mai puțin de 10 caractere

👉**INSTR(sir,subsir,poz,k)** – returnează poziția de început a celei de a **k**-a apariții a subșirului **subsir** în șirul **sir**, căutarea făcându-se începând cu poziția **poz** .

Dacă parametrii **poz** și **k** lipsesc, atunci se va returna poziția primei apariții a subșirului **subsir** în întregul șir **sir**.

Poziția de unde începe căutarea poate fi precizată și relativ la sfârșitul șirului, ca și în cazul funcției **substr**, dacă parametrul **poz** are o valoare negativă.

Exemplu	Rezultatul afișat
`select instr('abcdabcdabc','cd')`	**3**

Exemplu	Rezultatul afișat
`from dual`	

`select`
`instr('abcd','ef')`
`from dual`
	`0`

`select instr('abcd','bce')` `from dual`	`0`

`select`
`instr('abababababab','ab',4,2)`
`from dual`
	`7`
	Explicație. Se începe căutarea din poziția a patra, adică în zona subliniată cu o linie, și se afișează poziția de start a celei de a doua apariții, (subșirul subliniat cu linie dublă)

`select`
`instr('abababababab','ab',-4,1)`
`from dual`
	`9`

👉**`LENGTH(sir)`** – returnează numărul de caractere din șirul **`sir`**.

Exemplu	Rezultatul afișat
`select length('abcd')` `from dual`	`4`

👉**`LPAD(sir1,nr,sir2)`** – completează șirul **`sir1`** la stânga cu caracterele din șirul **`sir2`** până ce șirul obținut va avea lungimea **`nr`**.

Dacă lungimea șirului **`sir1`** este mai mare decât **`nr`**, atunci funcția va realiza trunchierea șirului **`sir1`**, ștergându-se caracterele de la sfârșitul șirului.

Exemplu	Rezultatul afișat
`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 funcția **LPAD**, completarea făcându-se la dreapta.

Exemplu	Rezultatul afișat
`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)

- funcția **TRIM** șterge caracterele **ch** de la începutul, sfârșitul sau din ambele părți ale șirului **sir**.
- în ultimele două formate ale funcției este subînțeleasă opțiunea **BOTH**.
- dacă **ch** nu este specificat se vor elimina spațiile inutile de la începutul, sfârșitul sau din ambele părți ale șirului **sir**.

Exemplu	Rezultatul afișat

`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)`** - înlocuiește toate aparițiile subșirului **`subsir`** din șirul **`sir`** cu șirul **`sirnou`**. Dacă nu este specificat noul șir, toate aparițiile subșirului **`subsir`** se vor elimina.

Exemplu	Rezultatul afișat
`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 funcții, 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 poziția primei apariții a șirului '**bc**' în șirul '**abcabcabc**', adică **2**. Primul apel al funcției **`substr`** este deci echivalent cu apelul

`substr('abcabcabc',1,1)`

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

`substr('abcabcabc',4)`

adică extrage toate caracterele de la poziția 4 până la sfârșitul șirului, deci '**abcabc**'. Așadar cele două apeluri extrag subșirul de dinaintea primei apariții a lui '**bc**' în șirul '**abcabcabc**', și respectiv de după această apariție. Cele două secvențe se concatenează apoi între ele incluzându-se șirul '**xyz**'. În concluzie comanda înlocuiește prima apariție a șirului '**bc**' din șirul '**abcabcabc**' cu șirul '**xyz**'.

II.2.4. Funcţii numerice

Aceste funcții operează asupra valorilor numerice și returnează un rezultat numeric. Funcțiile 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 funcțiile trigonometrice inverse, cu semnificația din matematică. Valoarea returnată de aceste funcții este exprimată în radiani.

SIN(n), COS(n), TAN(n) – sunt funcțiile trigonometrice cu aceeași semnificație ca și la matematică. Argumentul acestor funcții trebuie precizat în radiani.

Exemplu	Rezultatul afișat
`select sin(3.1415/2) from dual`	`.99999999892691403749520608603434614537 4`
`select cos(3.1415/2) from dual`	`.00004632679488004835355670590049419594`

POWER(m,n) – calculează valoarea m^n.

Exemplu	Rezultatul afișat
`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`	`.59460355750136053335874998528023795765 1`

SQRT(x) – calculează rădăcina pătrată a argumentului. Apelul **SQRT(x)** returnează aceeași valoare ca și **POWER(x,0.5)**.

Exemplu	Rezultatul afișat
`select sqrt(3) from dual`	`1.73205080756887729352744634150587236694`

REMAINDER(x,y) – funcția determină mai întâi acel multiplu a lui **y** care este cel mai apropiat de **x**, și returnează apoi diferența dintre **x** și acel multiplu.

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

Exemplu	Rezultatul afișat
	lui **3.5** este**10.5**, iar **10-10.5=-0.5**.
select remainder(-10,3.5) **from dual**	**0.5** ***Explicație.*** 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 funcția returnează același rezultat ca și funcția **REMAINDER**, adică restul împărțirii lui **x** la **y**. Teorema împărțirii cu rest este extinsă de această funcție și pentru numerele reale. Adică se ține cont de relația

x=y * cât + rest

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

Exemplu	Rezultatul afișat
select mod(10,3) **from dual**	**1** ***Explicație.*** **10=3*3+1**.
select mod(5,3) **from dual**	**2** ***Explicație.*** **5=3*1+2**
select mod(10,3.5) **from dual**	**3** ***Explicație.*** **10=3.5*2+3**.
select mod(-10,3.5) **from dual**	**-3** ***Explicație.*** **-10=3.5*(-2)-3**.
select mod(-10,-3.5) **from dual**	**-3** ***Explicație.*** **-10=-3.5*2-3**.
select mod(10,-3.5) **from dual**	**3** ***Explicație.*** **10=-3.5*(-2)+3**.

Se observă din exemplele anterioare că restul are întotdeauna același 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 afișat
select ceil(3) from dual	**3**
select ceil(-3) from dual	**-3**

Exemplu	Rezultatul afișat
`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)`** – rotunjește 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 funcție de următoarea zecimală.

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

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

Exemplu	Rezultatul afișat
`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 funcția **ROUND**, fără a rotunji ultima cifră.

Exemplu	Rezultatul afișat
`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 și 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 și anul, de asemenea ora, minutul, și secunda.
- **TIMESTAMP** – valorile având acest tip memorează data calendaristică, ora, minutul și secunda dar și fracțiunea de secundă.
- **TIMESTAMP WITH [LOCAL] TIME ZONE** – este similar cu **TIMESTAMP**, însă se va memora și diferența de fus orar față de ora universală, a orei de pe serverul bazei de date, sau a aplicației client, în cazul în care se include opțiunea **LOCAL**.
- **INTERVAL YEAR TO MONTH** – memorează o perioadă de timp în ani și luni.
- **INTERVAL DAY TO SECOND** – memorează un interval de timp în zile, ore, minute și 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)
```

și la afișarea tabelei

```
select * from test3
```

vom obține 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 știe să realizeze operații aritmetice asupra datelor calendaristice, astfel adăugarea valorii **1** la o dată calendaristică, va duce la obținerea 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 diferența dintre două date calendaristice, obținându-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. Diferența dintre două date calendaristice

Deși implicit o dată calendaristică de tip **DATE** nu este afișată în format complet (nu se afișează ora, minutul, secunda), în tabelă se memorează complet. De aceea poate fi uneori derutant rezultatul unor operații aritmetice cu date calendaristice, după cum se vede în figura II.2.6. în care diferența dintre ziua de astăzi și 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')
1.187997685185185185185185185185185519

Figura II.2.6.

De ce se obține acest lucru? Simplu, data de **20 aprilie** a fost precizată fără oră, așadar a fost considerată implicit ora **00:00**. Iar **sysdate** ne-a furnizat data curentă incluzând și ora. Așadar 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 funcții care operează asupra datelor calendaristice, dar în cele ce urmează ne vom opri asupra celor mai importante dintre acestea.

- **SYSDATE** – returnează data și ora curentă a serverului bazei de date.
- **CURRENT_DATE** – returnează data și ora curentă a aplicației 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. Funcțiile **SYSDATE**, **CURRENT_DATE** și **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 afișat
`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. Funcția **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. Funcțiile **LEAST** și **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 marți, 27 februarie 2007.

👉**LAST_DAY(data)** – returnează ultima zi din luna din care face parte data transmisă ca parametru.

Exemplu	Rezultatul afișat
`select next_day(sysdate,'Friday')` `from dual`	**02-MAR-07**
`select next_day(sysdate,'TUESDAY')` `from dual`	**06-MAR-07** ***Explicație.*** Chiar dacă ziua curentă este o zi de marți, funcția va returna următoarea zi de marți.

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** ***Explicație.*** Ziua returnată de **sysdate** este **27-FEB-07**, la care adăugăm **12** luni, deci obținem 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, funcția rotunjește 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. Funcția **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 rotunjește data la cea mai apropiată dată de 1 Ianuarie. Dacă data este înainte de 1 iulie, se va returna data de 1 ianuarie a aceluiași an. Dacă data este după data de 1 iulie se va returna data de 1 ianuarie a anului următor.
- **mm**, **month** – rotunjește 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 rotunjește 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 afișat
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'),	27-FEB-07 01-MAR-07 01-MAR-07

round(sysdate+17,'mm') from dual	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 aceleași formate ca și în cazul funcției **ROUND**.

Exemplu	Rezultatul afișat
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 funcții care vă permit să transformați o valoare dintr-un tip de dată în altul.

Transformarea din dată calendaristică în şir de caractere

Transformarea unei date calendaristice în șir de caractere se poate realiza cu ajutorul funcției **TO_CHAR**. Această operație se poate dovedi utilă atunci când dorim obținerea unor rapoarte cu un format precis. Sintaxa acestei funcții 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 conține mai mulți parametrii care pot afecta modul în care va arăta șirul returnat. Câțiva din acești parametrii sunt prezentați în continuare.

Aspect	Parametru	Descriere	Exemplu
Secolul	**CC**	Secolul cu două cifre	**21**
Trimestrul	**Q**	Trimestrul din an în care se găsește 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.	

Aspect	Parametru	Descriere	Exemplu

În cadrul formatului se pot folosi oricare dintre următorii separatori

- / , . ; :

Dacă în șirul returnat dorim să includem și anumite texte acestea se vor include între ghilimele.

Iată în continuare și câteva exemple de folosire a acestei funcții.

Exemplu	Rezultatul afișat
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')	Second

Exemplu	Rezultatul afișat
`from dual`	
`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 funcția **TO_DATE** se poate transforma un șir de caractere precum **'May 26, 2006'** într-o dată calendaristică. Sintaxa funcției este:

TO_DATE(sir,format)

Formatul nu este obligatoriu, însă dacă nu este precizat, șirul trebuie să respecte formatul implicit al datei calendaristice **DD-MON-YYYY** sau **DD-MON-YY**. Formatul poate folosi aceiași parametrii de format ca și funcția **TO_CHAR**.

Exemplu	Rezultatul afișat

`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

Așa cum s-a precizat anterior în formatarea unei date calendaristice se pot folosi pentru an atât **YY** (respectiv **YYYY**) cât și **RR** (respectiv **RRR**). Diferența dintre aceste două formate este modul în care ele interpretează anii aparținând 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** și **RR**

Se observă modul diferit de interpretare a anului.

Dacă utilizați formatul **YY** și anul este specificat doar prin două cifre, se presupune că anul respectiv face parte din același 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 folosiți formatul **RR** și anul transmis este de două cifre, primele două cifre ale anului transmis este determinat în funcție de cele două cifre transmise și 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** și **49**, și ultimele două cifre ale anului curent sunt între **00** și **49** atunci secolul este același 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** și **99** iar anul curent este între **00** și **49** atunci secolul este secolul prezent minus **1**. De exemplu dacă transmiteți **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** și **99**, secolul este considerat secolul prezent plus 1. De exemplu dacă ați transmis anul **15** iar anul curent este **1987**, anul transmis este considerat ca fiind anul **2015**.

Regula 4: Dacă anul transmis este între **50** și **99**, iar anul curent este între **50** și **99**, secolul este același cu a anului curent. De exemplu, dacă transmiteți 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 șir de caractere, se folosește funcția **TO_CHAR**, cu următoarea sintaxă:

TO_CHAR(numar,format)

Formatul poate conține unul sau mai mulți parametrii de formatare dintre cei prezentați în tabelul următor.

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

Vom exemplifica în continuare câteva dintre aceste formate.

Exemplu	Rezultatul afișat
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')	**$0001.237**

Exemplu	Rezultatul afișat
`from dual`	
`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 șir de caractere într-o valoare numerică se realizează cu ajutorul funcției **TO_NUMBER**:

TO_NUMBER(sir,format)

Parametrii de formatare ce se pot folosi sunt aceeași ca în cazul funcției **TO_CHAR**. Iată câteva exemple.

Exemplu	Rezultatul afișat
`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 funcțiile care controlează modul de formatare sau conversie al datelor, Oracle oferă câteva funcții de uz general, care specifică modul în care sunt tratate valorile **NULL**.

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

👉**`NVL2(val1,val2,val3)`** – dacă valoarea **`val1`** nu este nulă atunci funcția va returna valoarea **`val2`**, iar dacă **`val1`** are valoarea **`NULL`** atunci funcția 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 Funcția **NVL2**

👉**`NULLIF(expr1,expr2)`** – dacă cele două expresii sunt egale, funcția returnează **`NULL`**. Dacă valorile celor două expresii sunt diferite atunci funcția 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 Funcția **NULLIF**

👉**`COALESCE(expr1, expr2, ..., exprn)`** – funcția 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 Funcția **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 funcția va returna valoarea **vali2**. Dacă funcția nu este egală cu nici una din valorile **vali1**, atunci funcția 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 afișa mesajul "**Ea este Maria**" însă următoarea comandă va afișa "**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 funcției **DECODE** se poate folosi expresia condițională **CASE**. Funcția **CASE** utilizează cuvintele cheie **when**, **then**, **else**, și **end** pentru a indica ramura selectată. În general orice apel al funcției **DECODE** poate fi transcris folosind funcția **CASE**. Chiar dacă o expresie folosind **CASE** este mai lungă decât expresia echivalentă care folosește funcția **DECODE**, varianta cu **CASE** este mult mai ușor de citit și greșelile sunt depistate mai ușor. În plus varianta **CASE** este compatibilă **ANSI-SQL**.

  Cele două comenzi de mai sus por fi transcrise cu ajutorul funcției **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
```