

## LABORATOR 3:

### FUNCȚII SQL

### CERERI MULTIRELAȚIE (join)

#### *Funcții SQL*

Funcțiile SQL sunt predefinite în sistemul Oracle și pot fi utilizate în instrucțiuni SQL. Ele nu trebuie confundate cu funcțiile definite de utilizator, scrise în PL/SQL.

Dacă o funcție SQL este apelată cu un argument având un alt tip de date decât cel așteptat, sistemul convertește implicit argumentul înainte să evalueze funcția.

Dacă o funcție SQL este apelată cu un argument null, ea returnează automat valoarea null. Singurele funcții care nu urmează această regulă sunt CONCAT, NVL și REPLACE.

Principalele funcții SQL pot fi clasificate în următoarele categorii:

- Funcții single-row
- Funcții multiple-row (funcții agregat)

#### *Funcții single-row*

Funcțiile single row returnează câte o singură linie rezultat pentru fiecare linie a tabelului sau vizualizării interogate. Aceste funcții pot apărea în listele SELECT, clauzele WHERE, START WITH, CONNECT BY și HAVING. În ceea ce privește tipul argumentelor asupra cărora operează și al rezultatelor furnizate, funcțiile single row pot fi clasificate în clase corespunzătoare.

#### 1. **Funcțiile de conversie** cele mai importante sunt:

Funcție	Semnificație	Exemplu
TO_CHAR (expr_number_sau_date[, format][,nlsparameters])	Convertește o valoare de tip numeric sau dată calendaristică, la un șir de caractere conform cu formatul specificat sau cu setările naționale specificate (NLS - National Language Support). Dacă formatul sau parametrii lipsesc se utilizează formatul și parametrii impliciti. Formatul este case sensitive.	TO_CHAR('3') = '3' TO_CHAR(-12) = '-12' TO_CHAR(sysdate, 'DDMMYYYY') = '09122009'
TO_NUMBER (expr_char[, format][,nlsparameters])	Convertește o valoare de tip șir de caractere la o valoare numerică conform cu formatul specificat. Dacă formatul sau parametrii lipsesc se utilizează formatul și parametrii impliciti.	TO_NUMBER ('-12.22', 'S99.99') = -12.22
TO_DATE (expr_char[, format][,nlsparameters])	Convertește o valoare de tip șir de caractere la o valoare de tip dată calendaristică în conformitate cu formatul specificat. Dacă formatul sau parametrii lipsesc se utilizează formatul și parametrii impliciti.	TO_DATE ('15-feb-2006','dd-mon-yyyy')

Obs: Există două tipuri de conversii:

- implicite, realizate de sistem atunci când este necesar;
- explicite, indicate de utilizator prin intermediul funcțiilor de conversie.

**Exercițiul 1:** Să se testeze funcțiile prezentate utilizând comenzi de tipul :

```
SELECT apel_funcție FROM dual;
```

```
SELECT TO_NUMBER('100.00', '9G999D99') from dual;
```

```
SELECT TO_NUMBER('1,300.23', '9G999D999' ) FROM DUAL;
```

```
SELECT TO_CHAR (13598.98, '099G999D999' ) FROM DUAL;
```

```
SELECT TO_CHAR (13598.98, '099G999D999' , ' NLS_NUMERIC_CHARACTERS = ",." ' )  
FROM DUAL;
```

### **DATE, TIMESTAMP, INTERVAL**

```
SELECT TO_CHAR(sysdate, 'hh:mi:ss PM') FROM dual;
```

```
SELECT SYSTIMESTAMP FROM dual; --TIMESTAMP
```

```
SELECT CAST (SYSDATE AS TIMESTAMP) FROM dual;
```

```
SELECT CURRENT_TIMESTAMP FROM dual; --TIMESTAMP WITH TIME ZONE
```

```
SELECT CAST (SYSDATE AS TIMESTAMP WITH TIME ZONE) FROM dual;
```

```
SELECT CAST (SYSDATE AS TIMESTAMP WITH LOCAL TIME ZONE) FROM dual;
```

```
SELECT TO_TIMESTAMP('03-02-2014 10:30:50.45', 'dd-mm-yyyy hh24:mi:ss.ff') from dual;
```

```
SELECT TO_TIMESTAMP_TZ('03-02-2014 10:30:50.45 03:30', 'dd-mm-yyyy hh24:mi:ss.ff  
TZh:TzM') from dual;
```

```
SELECT  
FROM_TZ(TO_TIMESTAMP('03-02-2014 10:30:50.45', 'dd-mm-yyyy hh24:mi:ss.ff'), '2:00')  
from dual;
```

```
SELECT 'x'  
FROM DUAL  
WHERE FROM_TZ(TO_TIMESTAMP('03-02-2014 10:30:50.45', 'dd-mm-yyyy  
hh24:mi:ss.ff'), '2:00') =  
TO_TIMESTAMP_TZ('03-02-2014 09:30:50.45 01:00', 'dd-mm-yyyy hh24:mi:ss.ff TZh:TzM');
```

```
SELECT sysdate + INTERVAL '3-2' YEAR TO MONTH from dual;
```

```
SELECT TO_CHAR(sysdate + INTERVAL '1' YEAR - INTERVAL '1 1:05' DAY TO MINUTE,  
'dd/mm/yyyy hh24:mi')  
FROM DUAL;
```

```
SELECT systimestamp + INTERVAL '1 1:05:30.40' DAY TO SECOND FROM DUAL;
```

2. **Funcțiile pentru prelucrarea caracterelor** sunt prezentate în următorul tabel:

Funcție	Semnificație	Exemplu
LOWER (expresie)	Convertește un șir de caractere la minuscule.	LOWER ('AbCdE') = 'abcde'
UPPER (expresie)	Convertește un șir de caractere la majuscule.	UPPER ('AbCdE') = 'ABCDE'
INITCAP (expresie)	Convertește un șir de caractere la un șir care începe cu majusculă și continuă cu minuscule.	INITCAP ('AbCdE') = 'Abcde'
CONCAT (expr1, expr2)	Concatenează doua expresii de tip caracter. Echivalent cu operatorul de concatenare '  '.	CONCAT ('Ab', 'CdE') = 'AbCdE'
SUBSTR (expresie, m[, n])	Extrage din expresia de tip șir de caractere, $n$ caractere începând cu poziția $m$ . Dacă lipsește argumentul $n$ , atunci extrage toate caracterele până la sfârșitul șirului. Dacă $m$ este negativ numărătoarea pozițiilor începe de la sfârșitul șirului de caractere spre început.	SUBSTR ('AbCdE', 2, 2) = 'bC' SUBSTR ('AbCdE', 2) = 'bCdE' SUBSTR ('AbCdE', -3, 2) = 'Cd' SUBSTR ('AbCdE', -3) = 'CdE'
LENGTH (expresie)	Returnează numărul de caractere al expresiei.	LENGTH ('AbCdE') = 5
INSTR (expresie, expr1[, m][, n])	Returnează poziția la care se găsește a $n$ -a ocurență a expresiei 'expr1' în cadrul expresiei 'expresie', căutarea începând de la poziția $m$ . Dacă $m$ sau $n$ lipsesc, valorile implicite sunt 1 pentru ambele.	INSTR (LOWER('AbC aBcDe'), 'ab', 5, 2) = 0 INSTR (LOWER('AbCdE aBcDe'), 'ab', 5) = 7
LPAD (expresie, n[, expr1]) sau RPAD (expresie, n[, expr1])	Completează expresia caracter dată ca parametru (expresie), la stânga (LPAD), respectiv la dreapta (RPAD) cu caracterele specificate în expresia expr1, până la lungimea specificată de parametrul $n$ . Implicit, dacă lipsește, expr1 este ' ' un spațiu.	RPAD (LOWER('AbCdE'), 10, 'X') = 'abcdeXXXXX' LPAD (LOWER('AbCdE'), 10) = '    abcde'
LTRIM (expresie[, expr1]) sau RTRIM (expresie[, expr1])	Reversul funcțiilor LPAD, RPAD. Trunchează expresia caracter la stânga sau la dreapta prin eliminarea succesivă a caracterelor din expresia expr1. Implicit, dacă lipsește, expr1 este ' ' un spațiu.	RTRIM ('abcdeXXXX', 'X') = 'abcde' LTRIM ('    abcde') = 'abcde'
TRIM (LEADING   TRAILING   BOTH caractere_trim FROM expresie)	Permite eliminarea caracterelor specificate (caractere_trim) de la începutul (leading) , sfârșitul (trailing) sau din ambele părți, dintr-o expresie caracter data.	TRIM (LEADING 'X' FROM 'XXXabcdeXXX') = 'abcdeXXX' TRIM (TRAILING 'X' FROM 'XXXabcdeXXX') = 'XXXabcde'

		TRIM ( BOTH 'X' FROM 'XXXabcdeXXX') = 'abcde' TRIM (' abcde ') = 'abcde'
REPLACE (expr, expr1, expr2)	Înlocuiește în prima expresie toate ocurențele șirului expr1 cu șirul expr2.	REPLACE ('%1%11', '%', '2') = '21211' REPLACE ('%1%11', '%1', '23') = '23231' REPLACE ('%1%11', '%') = '111'
TRANSLATE(expr, expr1, expr2)	Fiecare caracter din șirul de caractere expr care apare și în expr1 este transformat în caracterul corespunzător (aflat pe aceeași poziție ca și în expr1) din șirul de caractere expr2.	TRANSLATE('%1%11', '%', '2') = '21211' TRANSLATE('%1%111', '%1', '23') = ' 232333'
ASCII (expresie)	Returnează codul ASCII al primului caracter din șirul 'expresie'.	ASCII ('curs') = ASCII ('c') = 99
CHR(expresie)	Întoarce caracterul corespunzător codului ASCII specificat.	CHR(99)= 'c'

**Exercițiul 2:** Să se afișeze pentru angajații cu prenumele Steven, codul, numele și denumirea departamentului în care lucrează. Căutarea trebuie să nu fie case-sensitive, iar eventualele blank-uri care preced sau urmează numelui trebuie ignorate.

**Exercițiul 3:** Scrieți o cerere care are următorul rezultat pentru fiecare angajat: <titlu job> <prenume angajat> <nume angajat> castiga <salariu> lunar dar dorește <salariu de 3 ori mai mare>. Etichetați coloana 'Salariu ideal'. Pentru concatenare, utilizați atât funcția CONCAT cât și operatorul ||.

**Exercițiul 4:** Scrieți o cerere prin care să se afișeze prenumele salariatului cu prima litera majusculă și toate celelalte litere minuscule, numele acestuia cu majuscule și lungimea numelui, pentru angajații al căror nume începe cu J sau M sau care au a treia literă din nume A. Rezultatul va fi ordonat descrescător după lungimea numelui. Se vor eticheta coloanele corespunzător. Se cer 2 soluții (cu operatorul LIKE și funcția SUBSTR).

**Exercițiul 5:** Să se afișeze pentru toți angajații al căror nume se termină cu litera 'e', codul, numele, lungimea numelui și poziția din nume în care apare prima dată litera 'a'. Utilizați alias-uri corespunzătoare pentru coloane.

WHERE  
LOWER(SUBSTR(last\_name;-1))='e';

**Exercițiul 6:** Să se afișeze numele și data angajării salariaților precum și o coloană care va conține câte un caracter '\*' pentru fiecare 1000 zile lucrate.

### 3. Funcțiile aritmetice single-row

Pot opera

- asupra unei singure valori, și aceste funcții sunt: ABS (valoarea absolută), CEIL (partea întreagă superioară), FLOOR (partea întreagă inferioară), ROUND (rotunjire cu un număr speci.cat de zecimale), TRUNC (trunchiere cu un număr speci.cat de zecimale), EXP (ridicarea la putere a lui e), LN (logaritm natural), LOG (logaritm într-o bază speci.cată), MOD (restul împărțirii a două numere speci.cate), POWER (ridicarea la putere), SIGN(semnul unui număr), COS (cosinus), COSH (cosinus hiperbolic), SIN (sinus), SINH (sinus hiperbolic), SQRT (rădăcina pătrată), TAN (tangent), TANH (tangent hiperbolic);
- unei liste de valori, iar acestea sunt funcțiile LEAST și GREATEST, care întorc cea mai mică, respectiv cea mai mare valoare a unei liste de expresii.

**Exercițiul 7:** Să se afișeze codul salariatului, numele, salariul, salariul mărit cu 15%, exprimat cu două zecimale și numărul de sute al salariului nou rotunjit la 2 zecimale. Etichetați ultimele două coloane 'Salariu nou', respectiv 'Numar sute'. Se vor lua în considerare salariații al căror salariu nu este divizibil cu 1000.

### 4. Funcțiile pentru prelucrarea datelor calendaristice sunt:

Funcție	Semnificație	Exemplu
SYSDATE	Întoarce data și timpul curent	
MONTHS_BETWEEN (date1, date2)	Returnează numărul de luni dintre data <i>date1</i> și data <i>date2</i> . Rezultatul poate fi pozitiv sau negativ după cum <i>date1</i> este mai recentă sau nu față de <i>date2</i> . Zecimalele reprezintă părți dintr-o luna!	ROUND(MONTHS_BETWEEN (SYSDATE + 31, SYSDATE)) = 1
ADD_MONTHS (date, n)	Adaugă <i>n</i> luni la o data specificată. Valoarea <i>n</i> trebuie să fie întreagă (pozitivă sau negativă).	MONTHS_BETWEEN (ADD_MONTHS(SYSDATE, 3), SYSDATE) = 3
NEXT_DAY (date, char)	Returnează data corespunzătoare primei zile a săptămânii specificate (char) care urmează după date.	NEXT_DAY('15-dec-2006','Monday') = '18-dec-2006' NEXT_DAY ('15-dec-2006',1) = '18-dec-2006'
LAST_DAY (date)	Returnează data corespunzătoare ultimei zile din luna calendaristică ce conține data specificată.	LAST_DAY ('15-feb-2006') = '28-feb-2006'
ROUND (date [, format])	Returnează data calendaristică rotunjită după formatul specificat. Valoarea implicită este 'DAY'.	ROUND (TO_DATE ('27-OCT-00'),'YEAR') = 01-JAN-01  TO_CHAR (ROUND (TO_DATE ('15-feb-2006 13:50','dd-mon-yyyy hh24:mi'), 'dd-mm-yyyy hh24:mi') = '16-02-2006 00:00'  TO_CHAR (ROUND (TO_DATE ('15-feb-2006 11:50','dd-mon-yyyy

		hh24:mi')), 'dd-mm-yyyy hh24:mi') = '15-02-2006 00:00'
TRUNC (date [, format])	Returnează data calendaristică trunchiată după formatul specificat. Valoarea implicită este 'DAY'.	TO_CHAR (TRUNC (TO_DATE ('15-feb-2006 13:50','dd-mon-yyyy hh24:mi')), 'dd-mm-yyyy hh24:mi') = '15-02-2006 00:00'
LEAST(d1, d2, ..., dn) GREATEST(d1, d2, ..., dn)	Dintr-o listă de date calendaristice, funcțiile întorc prima, respectiv ultima dată în ordine cronologică.	

**Exercițiul 9:** Să se afișeze numele și prenumele angajatului (într-o singură coloană), data angajării și data negocierii salariului, care este prima zi de Luni după 6 luni de serviciu. Etichetați această coloană 'Negociere'.

**Exercițiul 9:** Să se afișeze data următoarei zile de Vineri de peste 3 luni în formatul zi\_lună, denumire\_lună, zi\_saptamană, minut, ora.

**Exercițiul 10:** Pentru fiecare angajat să se afișeze numele și numărul de luni de la data angajării. Etichetați coloana 'Luni lucrate'. Să se ordoneze rezultatul după numărul de luni lucrate. Se va rotunji numărul de luni la cel mai apropiat număr întreg.

5. **Operațiile care se pot efectua asupra datelor calendaristice** sunt următoarele:

Operație	Tipul de date al rezultatului	Descriere
date +/- number	Date	Scade/Adaugă un număr de zile dintr-o / la o dată.
date1 - date2	Number	Returnează numărul de zile dintre două date calendaristice.
date +/- number/24	Date	Scade/Adaugă un număr de ore la o / dintr-o dată calendaristică.

**Exemplul 11:** Să se afișeze detalii despre salariații care au lucrat un număr întreg de săptămâni până la data curentă.

```
SELECT employee_id, last_name, salary
FROM employees
WHERE MOD(ROUND(SYSDATE – hire_date), 7) = 0;
```

**Exercițiul 12:** Să se afișeze data (numele lunii, ziua, anul, ora, minutul și secunda) de peste 30 zile.

**Exercițiul 13:** Să se afișeze numărul de zile rămase până la sfârșitul anului.

**Exercițiul 14:** Să se afișeze data  
a) de peste 12 ore (12/24)  
b) data de peste 5 minute. (1/288).

## 6. Funcții diverse:

Funcție	Semnificație	Exemplu
NVL (expr1, expr2)	Returnează expr1 dacă aceasta nu este NULL, expr2 în caz contrar. Cele 2 expresii trebuie să aibă același tip sau expr2 să permită conversia implicită la tipul expresiei expr1.	NVL(NULL, 1) = 1 NVL(2, 1) = 2 NVL('c', 1) = 'c'  -- face conversie NVL(1, 'c') -- eroare --nu face conversie
NVL2 (expr1, expr2, expr3)	Dacă expr1 este nenulă atunci returnează expr2, altfel Returnează expr3	NVL2 (1, 2, 3) = 2 NVL2 (NULL, 2, 3) = 3
NULLIF (expr1, expr2)	Dacă expr1 = expr2 atunci funcția returnează NULL, altfel returnează expresia expr1. Echivalent cu CASE WHEN expr1 = expr2 THEN NULL ELSE expr1 END	NULLIF (1, 2) = 1 NULLIF (1,1) = NULL
COALESCE (expr1, expr2, ... , exprn)	Returnează expr1 dacă nu este NULL, altfel expr2, dacă expr2 nu este NULL, altfel... exprn , dacă expr2 nu este NULL.	COALESCE (1, 2, 3) = 1 COALESCE (NULL, 2, 3) = 2 COALESCE (NULL, NULL, 3) = 3
UID, USER	Întorc ID-ul, respectiv username-ul utilizatorului ORACLE curent	
VSIZE(expresie)	Întoarce numărul de octeți ai unei expresii de tip DATE, NUMBER sau VARCHAR2	SELECT VSIZE(hire_date) FROM employees WHERE employee_id=104;
DECODE (expr, expr_cautare1, expr_rezultat1, [expr_cautare2, expr_rezultat2, .. expr_cautaren, expr_rezultatn, ] [rezultat_implicit])	Decodifică valoarea expresiei. Dacă valoarea expresiei este expr_cautare1 atunci e returnată expr_rezultat1. Dacă nu se potrivește nici o expresie de căutare atunci e returnat rezultat_implicit.	DECODE (1, 1, 2, 3) = 2 DECODE (2, 1, 2, 3) = 3 DECODE (3, 1, 2, 3) = 3

**Exercițiul 15:** Să se listeze numele, salariul și comisionul tuturor angajaților al căror venit lunar depășește 10000\$.

**Exercițiul 16:** Să se afișeze numele angajaților și comisionul. Dacă un angajat nu câștigă comision, să se scrie 'Fara comision'. Etichetați coloana 'Comision'.

### ● Instrucțiunea CASE

În funcție de valoarea expresiei expr returnează valoare\_i corespunzătoare primei clauze WHEN .. THEN pentru care expr = expresie\_i; dacă nu corespunde cu nici o clauză WHEN atunci returnează valoarea din ELSE. Nu se poate specifica NULL pentru toate valorile de returnat. Toate valorile trebuie să aibă același tip de date.

```

CASE expr
WHEN expr_1 THEN valoare_1
[WHEN expr_2 THEN valoare_2
...
WHEN expr_n THEN valoare_n]
[ELSE valoare]
END

```

**Exemplul 17:** Să se afișeze numele, codul job-ului, salariul și o coloană care să arate salariul după mărire. Se presupune că pentru IT\_PROG are loc o mărire de 20%, pentru SA\_REP creșterea este de 25%, iar pentru SA\_MAN are loc o mărire de 35%. Pentru ceilalți angajați nu se acordă mărire. Să se denumească coloana "Salariu renegociat".

```

SELECT last_name, job_id, salary,
DECODE(job_id, 'IT_PROG', salary * 1.2,
'SA_REP', salary * 1.25,
'SA_MAN', salary * 1.35,
salary) "Salariu negociat"
FROM employees;

```

```

SELECT last_name, job_id, salary,
CASE job_id WHEN 'IT_PROG' THEN salary * 1.2
WHEN 'SA_REP' THEN salary * 1.25
WHEN 'SA_MAN' THEN salary * 1.35
ELSE salary
END "Salariu negociat"
FROM employees;

```

**Exercițiul 18:** Să se afișeze codul, numele și orașul pentru toate departamentele. Primele departamente afișate vor fi cele din Seattle, care vor apărea în lista ordonate alfabetic după nume. Restul departamentelor vor fi ordonate după numele orașului, ordinea departamentelor din același oraș fiind cea alfabetică.

Modifică cererea de mai sus astfel încât să se afișeze și coloana **manger**. Dacă pentru un departament nu este cunoscut managerul coloana manger va conține textul “manager necunoscut”.

**Exercițiul 19:** Să se afișeze numele angajaților, titlul jobului (job\_title), numele departamentului în care lucrează și coloana "spor de vechime" calculată astfel:

dacă  $x > 0$  și  $x < 100$ :  $x * 10$

dacă  $x > 100$ :  $x * 5$

$x$  este un număr întreg care reprezintă numărul de luni lucrate până la data de '01-01-2000'.

Dacă salariatul a fost angajat după această dată, sporul este 0.

**Exercițiul 20:** Să se afișeze numele angajaților și coloana id care să conțină codul angajatului concatenat cu codul departamentului dacă este cunoscut codul departamentului, sau codul angajatului dacă nu este știut departamentul.



## *Funcții multiple-row*

Funcțiile multiple-row (agregat) pot fi utilizate pentru a returna informația corespunzătoare fiecăruia dintre grupurile obținute în urma divizării liniilor tabelului cu ajutorul clauzei GROUP BY. Ele pot apărea în clauzele SELECT, ORDER BY și HAVING. Server-ul Oracle aplică aceste funcții fiecărui grup de linii și returnează un singur rezultat pentru fiecare mulțime.

Dintre funcțiile grup definite în sistemul Oracle, se pot enumera: AVG, SUM, MAX, MIN, COUNT, STDDEV, VARIANCE etc. Tipurile de date ale argumentelor funcțiilor grup pot .CHAR, VARCHAR2, NUMBER sau DATE.

Funcțiile AVG, SUM, STDDEV și VARIANCE operează numai asupra valorilor numerice. Funcțiile MAX și MIN pot opera asupra valorilor numerice, caracter sau dată calendaristică.

Toate funcțiile grup, cu excepția lui COUNT(\*), ignoră valorile null. COUNT(expresie) returnează numărul de linii pentru care expresia dată nu are valoarea null. Funcția COUNT returnează un număr mai mare sau egal cu zero și nu întoarce niciodată valoarea null.

Când este utilizată clauza GROUP BY, server-ul sortează implicit mulțimea rezultată în ordinea crescătoare a valorilor coloanelor după care se realizează gruparea.

**Exemplul 21:** Să se obțină care este cel mai mic dintre salariile angajaților.

```
SELECT min(salary)
FROM employees;
```

**Exercițiul 22:** Să se obțină media salariilor angajaților rotunjită la 2 zecimale pentru angajații care câștigă comision.