LABORATOR 2:
FUNCŢII SQL
CERERI MULTIRELAŢIE (introducere)

Funcţii SQL

Funcţiile SQL sunt predefinite în sistemul Oracle şi pot fi utilizate în instrucţiuni SQL. Ele nu trebuie confundate cu funcţiile definite de utilizator, scrise în PL/SQL.

Dacă o funcţie SQL este apelată cu un argument având un alt tip de date decât cel aşteptat, sistemul converteşte implicit argumentul înainte să evalueze funcţia.

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

Principalele funcţii SQL pot fi clasificate în următoarele categorii:

- Funcţii single-row
- Funcţii multiple-row (funcţii agregat)

Funcţii single-row

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

1. **Funcţiile de conversie** cele mai importante sunt:

Funcţie	**Semnificaţie**	**Exemplu**
TO_CHAR (expr_number_sau_date[, format] [,nlsparameters])	Converteşte o valoare de tip numeric sau dată calendaristică, la un şir de caractere conform cu formatul specificat sau cu setările naţionale specificate (NLS - National Language Support). Daca formatul sau parametrii lipsesc se utilizează formatul şi parametrii impliciţi. Formatul este case sensitive.	TO_CHAR('3') = ' 3' TO_CHAR(-12) = '-12' TO_CHAR(sysdate, 'DDMMYYYY')= ' 09122009'
TO_NUMBER (expr_char[, format] [,nlsparameters])	Converteşte o valoare de tip şir de caractere la o valoare numerică conform cu formatul specificat. Dacă formatul sau parametrii lipsesc se utilizează formatul şi parametrii impliciţi.	TO_NUMBER ('-12.22', 'S99.99') = -12.22
TO_DATE (expr_char[, format] [,nlsparameters])	Converteşte o valoare de tip şir de caractere la o valoare de tip dată calendaristică în conformitate cu formatul specificat. Dacă formatul sau parametrii lipsesc se utilizează formatul şi parametrii impliciţi.	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 funcţiilor de conversie.

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

Funcţie	Semnificaţie	Exemplu
LOWER (expresie)	Converteşte un şir de caractere la minuscule.	LOWER ('AbCdE') = 'abcde'
UPPER (expresie)	Converteşte un şir de caractere la majuscule.	UPPER ('AbCdE') = 'ABCDE'
INITCAP (expresie)	Converteşte un şir de caractere la un şir care începe cu majusculă şi 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 şir de caractere, *n* caractere începând cu poziţia *m*. Dacă lipseşte argumentul *n*, atunci extrage toate caracterele până la sfârşitul şirului. Dacă *m* este negativ numărătoarea poziţiilor începe de la sfârşitul şirului 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 poziţia la care se găseşte a *n*-a ocurentă a expresiei 'expr1' în cadrul expresiei 'expresie', căutarea începând de la poziţia *m*. Daca *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 lipseşte, expr1 este ' ' un spaţiu.	RPAD (LOWER('AbCdE'), 10, 'X') = 'abcdeXXXXX' LPAD (LOWER('AbCdE'),10) = ' abcde'
LTRIM (expresie[, expr1]) sau RTRIM (expresie[, expr1])	Reversul funcţiilor LPAD, RPAD. Trunchează expresia caracter la stânga sau la dreapta prin eliminarea succesivă a caracterelor din expresia expr1. Implicit, daca lipseşte, expr1 este ' ' un spaţiu.	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) , sfârşitul (trailing) sau din ambele părţi, 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)	Înlocuieşte în prima expresie toate ocurentele şirului expr1 cu şirul expr2.	REPLACE ('%1%11','%','2') = '21211' REPLACE ('%1%11','%1','23') = '23231' REPLACE ('%1%11','%') = '111'
TRANSLATE(expr, expr1, expr2)	Fiecare caracter din şirul de caractere expr care apare şi în expr1 este transformat în caracterul corespunzător (aflat pe aceeaşi poziţie ca şi în expr1) din şirul de caractere expr2.	TRANSLATE('%1%11','%','2') = '21211' TRANSLATE('%1%111','%1','23') = ' 232333'
ASCII (expresie)	Returnează codul ASCII al primului caracter din şirul 'expresie'.	ASCII ('curs') = ASCII ('c') = 99
CHR(expresie)	Întoarce caracterul corespunzător codului ASCII specificat.	CHR(99)= 'c'

Exerciţiul 1: Să se testeze funcţiile prezentate utilizând comenzi de tipul :

SELECT apel_functie FROM dual;

Exemplul 2: Să se afişeze pentru angajaţii cu prenumele Steven, codul, numele şi codul departamentului în care lucrează. Căutarea trebuie să nu fie case-sensitive, iar eventualele blank-uri care preced sau urmează numelui trebuie ignorate.

```
SELECT employee_id, last_name, department_id
FROM employees
WHERE LTRIM(RTRIM(UPPER(first_name))) ='STEVEN';

SELECT employee_id, last_name, department_id
FROM employees
WHERE TRIM(BOTH FROM UPPER(first_name)) ='STEVEN';
```

Exerciţiul 3: Scrieţi o cerere care are următorul rezultat pentru fiecare angajat: <prenume angajat> <nume angajat> castiga <salariu> lunar dar doreste <salariu de 3 ori mai mare>. Etichetaţi coloana 'Salariu ideal'. Pentru concatenare, utilizaţi atât funcţia CONCAT cât şi operatorul ||.

Exerciţiul 4: Scrieţi o cerere prin care să se afişeze prenumele salariatului cu prima litera majusculă şi toate celelalte litere minuscule, numele acestuia cu majuscule şi lungimea numelui, pentru angajaţii 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 soluţii (cu operatorul LIKE şi funcţia SUBSTR).

```
WHERE SUBSTR(LOWER(last_name), 1, 1) ='j' OR
SUBSTR(LOWER(last_name), 3, 1) = 'a' OR
SUBSTR(LOWER(last_name), 1, 1) = 'm'
```

Exerciţiul 5: Să se afişeze pentru toţi angajaţii al căror nume se termină cu litera 'e', codul, numele, lungimea numelui şi poziţia din nume în care apare prima dată litera 'a'. Utilizaţi alias-uri corespunzătoare pentru coloane.

WHERE
LOWER(SUBSTR(last_name;-1)) ='e';

Exerciţiul 6: Să se listeze numele şi data angajării salariaţilor care câştigă comision. Să se eticheteze coloanele 'Nume angajat', 'Data angajarii'. Pentru a nu obţine alias-ul datei angajării trunchiat, utilizaţi funcţia RPAD.
RPAD(TO_CHAR(hire_date), 20,' ') "Data angajarii"

3. **Funcţiile aritmetice** single-row

Pot opera

- asupra unei singure valori, şi aceste funcţii 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 împărţirii 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 funcţiile LEAST şi GREATEST, care întorc cea mai mică, respectiv cea mai mare valoare a unei liste de expresii.

Exerciţiul 7: Să se afişeze codul salariatului, numele, salariul, salariul mărit cu 15%, exprimat cu două zecimale şi numărul de sute al salariului nou rotunjit la 2 zecimale. Etichetaţi ultimele două coloane 'Salariu nou', respectiv 'Numar sute'. Se vor lua în considerare salariaţii al căror salariu nu este divizibil cu 1000.

4. **Funcţiile pentru prelucrarea datelor calendaristice** sunt:

Funcţie	Semnificaţie	Exemplu
SYSDATE	Întoarce data şi timpul curent	
MONTHS_BETWEEN (date1, date2)	Returnează numărul de luni dintre data *date1* şi data *date2*. Rezultatul poate fi pozitiv sau negativ după cum *date1* este mai recentă sau nu faţă de *date2*. Zecimalele reprezintă parţi dintr-o luna!	ROUND(MONTHS_BETWEEN (SYSDATE + 31, SYSDATE)) = 1
ADD_MONTHS (date, n)	Adaugă *n* luni la o data specificată. Valoarea *n* 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 conţine 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, funcţiile întorc prima, respectiv ultima dată în ordine cronologică.	

Exemplul 8: 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'.
NEXT_DAY (ADD_MONTHS(___,___), '_____')

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

Operaţie	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 afişeze detalii despre salariaţii care au lucrat un număr întreg de săptămâni până la data curentă.

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

Exerciţiul 12: Să se afişeze data (numele lunii, ziua, anul, ora, minutul şi secunda) de peste 30 zile.

Exerciţiul 13: Să se afişeze numărul de zile rămase până la sfârşitul anului.

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

6. **Funcţii diverse:**

Funcţie	Semnificaţie	Exemplu
NVL (expr1, expr2)	Returnează expr1 dacă aceasta nu este NULL, expr2 în caz contrar. Cele 2 expresii trebuie să aibă acelaşi 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)	Daca expr1 = expr2 atunci funcţia 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 octeţi 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_cautarei atunci e returnată expr_rezultati. Dacă nu se potriveşte 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

Exemplul 15: Să se listeze numele, salariul şi comisionul tuturor angajaţilor al căror venit lunar depăşeşte 10000$.

SELECT last_name, salary, commission_pct, salary + salary NV L(commission_pct, 0) venit_lunar
FROM employees
WHERE salary + salary NV L(commission_pct, 0) > 10000;

Exerciţiul 16: Să se afişeze numele angajaţilor şi comisionul. Dacă un angajat nu câştigă comision, să se scrie 'Fara comision'. Etichetaţi coloana 'Comision'.

- Instrucţiunea CASE

În funcţie 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 acelaşi 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 a.şeze numele, codul job-ului, salariul şi o coloană care să arate salariul după mărire. Se presupune că pentru IT_PROG are loc o mărire de 20%, pentru SA_REP creşterea este de 25%, iar pentru SA_MAN are loc o mărire de 35%. Pentru ceilalţi angajaţi 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;

Exerciţiul 18: Să se afişeze codul, numele şi oraşul pentru toate departamentele. Primele departamente afişate vor fi cele din Seattle, care vor aparea in lista ordonate alfabetic dupa nume. Restul departamentelor vor fi ordonate dupa numele oraşului, ordinea departamentelor din acelaşi oraş fiind cea alfabetica.
Modificţi cererea de mai sus astfel încât să se afişeze şi colona **manger**. Dacă pentru un departament nu este cunoscut managerul coloana manger va conţine textul “manager necunoscut”.

Exerciţiul 19: Să se afişeze numele angajatilor, titlul jobului (job_title), numele departamentului în care lucreaza si coloana "spor de vechime" calculată astfel:
daca x >0 si x< 100: x*10
daca x>100: x*5
x este un numar intreg care reprezinta numarul de luni lucrate pana la data de '01-01-2000'.
Dacă salariatul a fost angajat după această dată, sporul este 0.

Exerciţiul 20: Să se afişeze numele angajaților şi coloana id are se conţină codul angajatului concatenat cu codul departamentului dacă este cunoscut codul departamentului sau codul angajatului daca nu este ştiut departamentul.

Funcţii multiple-row

Funcţiile multiple-row (agregat) pot fi utilizate pentru a returna informaţia corespunzătoare fiecăruia dinte grupurile obţinute în urma divizării liniilor tabelului cu ajutorul clauzei GROUP BY. Ele pot apărea în clauzele SELECT, ORDER BY şi HAVING. Server-ul Oracle aplică aceste funcţii fiecărui grup de linii şi returnează un singur rezultat pentru fiecare mulţime.
Dintre funcţiile grup definite în sistemul Oracle, se pot enumera: AVG, SUM, MAX, MIN, COUNT, STDDEV, VARIANCE etc. Tipurile de date ale argumentelor funcţiilor grup pot .CHAR, VARCHAR2, NUMBER sau DATE.
Funcţiile AVG, SUM, STDDEV şi VARIANCE operează numai asupra valorilor numerice. Funcţiile MAX şi MIN pot opera asupra valorilor numerice, caracter sau dată calendaristică.
Toate funcţiile grup, cu excepţia lui COUNT(*), ignoră valorile null. COUNT(expresie) returnează numărul de linii pentru care expresia dată nu are valoarea null. Funcţia COUNT returnează un număr mai mare sau egal cu zero şi nu întoarce niciodată valoarea null.
Când este utilizată clauza GROUP BY, server-ul sortează implicit mulţimea rezultată în ordinea crescătoare a valorilor coloanelor după care se realizează gruparea.

Exemplul 21: Să se obţină care este cel mai mic dintre salariile angajaţilor.

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

Exerciţiul 22: Să se obţină media salariilor angajaţilor rotunjită la 2 zecimale pentru angajaţii care caştigă comision.

Cereri multitabel -- Join

Join-ul este operaţia de regăsire a datelor din două sau mai multe tabele, pe baza valorilor comune ale unor coloane. De obicei, aceste coloane reprezintă cheia primară, respectiv cheia externă a tabelelor.
Condiţia de join se scrie în clauza WHERE a instrucţiunii SELECT. Într-o instrucţiune SELECT care uneşte tabele prin operaţia de join, se recomandă ca numele coloanelor să fie precedate de numele sau alias-urile tabelelor pentru claritate şi pentru îmbunătăţirea timpului de acces la baza de date. Dacă acelaşi nume de coloană apare în mai mult de două tabele, atunci numele coloanei se prefixează obligatoriu cu numele sau alias-ul tabelului corespunzător. Pentru a realiza un join între n tabele, va finevoie de cel puţin n .1 condiţii de join.
Inner join (equijoin, join simplu) corespunde situaţiei în care valorile de pe coloanele ce apar în condiţia de join trebuie să fie egale.

Exemplul 23: Să se afişeze numele salariatului, codul şi numele departamentului pentru toţi angajaţii.

```
SELECT employee_id, last_name, department_name
FROM employees, departments
WHERE employees.department_id = departments.department_id;
```

```
SELECT employee_id, last_name, department_name
FROM employees e, departments d
WHERE e.department_id = d.department_id
```

Numele sau alias-urile tabelelor sunt obligatorii în dreptul coloanelor care au acelaşi nume în mai multe tabele. Altfel, nu sunt necesare dar este recomandată utilizarea lor pentru o mai bună claritate a cererii.

Exerciţiul 24: Să se listeze titlurile job-urilor atribuite angajaţilor care lucrează în departamentul 30.

Exerciţiul 25: Să se afişeze numele angajatului, numele departamentului şi locaţia pentru toţi angajaţii care câştigă comision.

Exerciţiul 26: Să se afişeze numele salariatului şi numele departamentului pentru toţi salariaţii care au litera A inclusă în nume.

Exerciţiul 27: Să se afişeze numele, job-ul, codul şi numele departamentului pentru toţi angajaţii care lucrează în Oxford.