

## **CAPITOLUL 3. ELEMENTELE DE BAZĂ ALE LIMBAJULUI SQL\*PLUS**

### ***3.1. DESPRE LIMBAJ***

În 1974 a fost lansat proiectul System/R de către firma IBM. Tot în acest an a apărut limbajul structurat de programare SEQUEL (Structured English as Query Language) autori fiind Chamberlin și Boyce.

În 1976 apare un nou limbaj SEQUEL 2 care a fost declarat limbajul de interogare al SGBD System/R. Denumirea limbajului este schimbată de Chamberlin în SQL (Structured Query Language) în anul 1980.

Ulterior limbajul a fost perfecționat fiind considerat cel mai răspândit limbaj de interogare a bazelor de date relaționale.

Institutul Național pentru Standarde în anul 1982 a lansat un proiect de lucru pentru standardizarea limbajelor de interogare care a fost finalizat în 1986 apărând standardul ANSI SQL-86. Acesta definește comenzile de bază ale SQL, dar nu conține partea de actualizare și acordarea drepturilor de acces la o bază de date.

Prin revizuire acest limbaj apare în 1989 SQL-1 ca fiind limbajul fundamental al SGBD relaționale.

În 1992 apare versiunea SQL-2 care oferă noi facilități cum ar fi: joncțiune externă, implementarea restricției referențiale, modificarea schemei bazei de date, etc.

Cel mai recent standard este SQL-3 care a fost lansat în anul 1999, acesta este considerat un limbaj complet în vederea definirii și gestiunii obiectelor complexe. Se consideră că prin publicarea standardului propus în acest an a fost depășită bariera relaționalului, el fiind mult mai mult decât un instrument de consultare a bazelor de date.

### ***3.2. CONCEPTE UTILIZATE***

SQL\*PLUS este un limbaj neprocedural și operează asupra datelor normalizate. Conceptele necesare a fi cunoscute pentru lucrul cu acest limbaj sunt: tabelă, cheie primară, coloană, rând, viziune, index, sinonim, cluster, bază de date relațională, comanda, blocul, cererea, raportul etc.

*Tabela sau relația* este un ansamblu format din  $n$  coloane (atribute/subansambluri) și  $m$  rânduri (tupluri/linii) care respectă următoarele condiții minime: nu conține date la nivel agregat (valorile aflate la intersecția liniilor cu coloanele să fie la un nivel elementar); liniile sunt distincte unele față de altele; nu conține coloane repetitive în descriere.

*Cheia primară* este un atribut care are valori distincte. Deci, fiecare linie se identifică printr-o valoare distinctă. Două sau mai multe atribute care pot fi chei primare se numesc chei candidate.

*Coloana* tabelii este formată din valorile pe care le ia atributul în liniile tabelii respective.

*Rândul/tuplul/linia* este format din valorile coloanelor ce se referă la o entitate a tabelii.

*Baza de date relațională* este un ansamblu de tabele normalizate, grupate în jurul unui subiect, în principiu, bine definit. Într-o bază de date relațională, entitățile și legăturile sunt transpuse în tabele.

*Viziunea* este o tabela logică și reprezintă o fereastră la date, dintr-una sau mai multe tabele.

Pentru ca accesul la date să se facă mai rapid, se utilizează indexarea. Un *index* reprezintă o cheie pe una sau mai multe coloane. Indexarea este dinamică deoarece se pot adăuga sau șterge indecși oricând, fără ca datele memorate sau aplicațiile scrise să fie afectate.

Pentru realizarea unor operații sau pentru a utiliza în cereri nume mai scurte, se pot defini *sinonime* ale unor nume de tabele sau viziuni.

Un *cluster* reprezintă o anumită modalitate de grupare a rândurilor unei sau mai multor tabele. Această grupare mărește viteza de execuție a unor operații consumatoare de timp.

*Comanda* este o instrucțiune emisă din SQL\*Plus către o bază de date Oracle.

*Blocul* reprezintă un grup de instrucțiuni SQL și PL/SQL.

*Cererea* este o comandă SQL (SELECT) care regăsește date din baza de date. Rezultatul cererii îl formează datele regăsite din baza de date.

*Raportul* este rezultatul cererii formatat cu ajutorul comenzilor SQL\*Plus.

Numele unei baze de date, al unei tabeli, coloane sau variabile utilizator trebuie să aibă lungimea între 1 și 30 caractere. Un nume nu poate conține apostrofuri. Cu atât mai puțin, un nume utilizat într-o comandă nu va fi introdus între apostrofuri. Literele mici și mari sunt echivalente (nu se face distincția între literele mici și mari). Un nume trebuie să înceapă cu o literă, să conțină numai anumite caractere (A-Z, 0-9, \$, #, @, -), să nu duplicate numele unui alt obiect de același tip și să difere de un cuvânt rezervat ORACLE.

Cuvintele rezervate nu pot fi utilizate ca nume de tabele, coloane sau orice alte obiecte definite de utilizator. Ele sunt prezentate în Anexa 1.

### 3.3. FUNCȚII SQL

Funcțiile se apelează prin sintaxa:

**Nume\_funcție (argument1, argument2, ...)**

Funcțiile SQL sunt *cu un singur rând* sau scalare (returnează un singur rezultat pentru fiecare rând al cererii emise asupra unei tabele sau vederi) și *cu mai multe rânduri* numite funcții grup sau agregate (returnează un singur rezultat pentru un grup de rânduri regăsite dintr-o tabelă sau vedere).

#### A. Funcțiile SQL cu un singur rând (funcțiile scalare)

Acestea sunt funcții: numerice, caracter, de tip DATE, de conversie și alte funcții.

1) *Funcțiile numerice* acceptă valori numerice și returnează rezultate numerice și sunt prezentate în tabelul 3.1. Tabela **DUAL** folosită în exemple, este creată automat de către Oracle odată cu crearea dicționarului de date și se află în schema utilizatorului SYS, dar cu acest nume este accesibilă tuturor utilizatorilor unei baze de date Oracle. Ea are o singură coloană numită DUMMY cu tipul de date VARCHAR2(1) și un singur rând cu valoarea 'X'. Selecțiile din tabela DUAL sunt utile pentru calcularea unor expresii constante cu comanda SQL SELECT. Din cauză că are un singur rând, constanta este returnată o singură dată. Alternativ pentru aceeași activitate se poate selecta o constantă, pseudocoloană sau o expresie din orice tabelă.

**Tabelul 3.1 Funcțiile numerice**

<i>Funcția</i>	<i>Rolul funcției</i>	<i>Exemple</i>
<i>ABS(n)</i>	<i>Returnează valoarea absolută a numărului n.</i>	<i>SELECT ABS(-15) "Absolut"</i> <i>FROM DUAL;</i> <i>Absolut</i> ----- <i>15</i>
<i>ACOS(n)</i>	<i>Arc cosinus de n. Rezultatul este exprimat în radiani.</i>	<i>SELECT COS(.3) "Arc_Cosinus"</i> <i>FROM DUAL;</i> <i>Arc cosinus</i> ----- <i>1.26610367</i>
<i>ASIN(n)</i>	<i>Arc sinus de n.</i>	<i>SELECT ASIN(.3) "Arc_Sinus"</i> <i>FROM DUAL;</i> <i>Arc_Sinus</i> ----- <i>.304692654</i>
<i>ATAN</i>	<i>Arc tangentă de n.</i>	<i>SELECT ATAN(.3) "Arc_Tangentă"</i> <i>FROM DUAL;</i>

		<i>Arc_Tangentă</i> ----- .291456794
<i>ATAN2</i>	<i>Arc tangentă de n și m sau arc tangentă de n/m. Deci ATAN2(n,m) este identic cu ATAN2(n/m).</i>	SELECT ATAN2(.3,.2) <i>Arc_Tangentă2</i> " FROM DUAL; <i>Arc_Tangentă2</i> ----- .982793723
<i>CEIL(n)</i>	<i>Returnează numărul întreg cel mai mic care este mai mare sau egal cu n.</i>	SELECT CEIL(15.7) "NUMĂR" FROM DUAL; NUMĂR ----- 16
<i>COS(n)</i>	<i>Cosinus de n.</i>	SELECT COS(180 * 3.14/180) "Cosinus de 180 grade" FROM DUAL; Cosinus de 180 grade ----- -1
<i>COSH(n)</i>	<i>Cosinus hiperbolic de n.</i>	SELECT COSH(0) "Cosinus hiperbolic de 0" FROM DUAL; Cosinus hiperbolic de 0 ----- 1
<i>EXP(n)</i>	<i>Returnează o valoare egală cu e ridicat la puterea n, unde e = 2.71828183.</i>	SELECT EXP(4) "e la puterea 4" FROM DUAL; e la puterea 4 ----- 54.59815
<i>FLOOR(n)</i>	<i>Returnează numărul cel mai mare care este mai mic sau egal cu n.</i>	SELECT FLOOR(15.7) "Floor" FROM DUAL; Floor ----- 15
<i>LN(n)</i>	<i>Returnează logaritmul natural de n, unde n &gt; 0.</i>	SELECT LN(95) "Logaritm natural de 95" FROM DUAL; Logaritm natural de 95 ----- 4.55387689
<i>LOG(m,n)</i>	<i>Returnează logaritm în baza m de n. (LOG<sub>m</sub>n)</i>	SELECT LOG(10,100) "Log în baza 10 de 100" FROM DUAL; Log în baza 10 de 100 ----- 2
<i>MOD(m,n)</i>	<i>Returnează restul împărțirii</i>	SELECT MOD(11,4) "Modulo 4"

	<p>lui <math>m</math> la <math>n</math>. Dacă <math>n</math> este 0 returnează valoarea <math>m</math>. Această funcție se comportă diferit față de funcția modulo clasică din matematică, atunci când <math>m</math> este negativ. Având în vedere semnificația funcției MOD funcția modulo clasică din matematică se poate exprima cu formula : <math>m - n * \text{FLOOR}(m/n)</math></p>	<p>FROM DUAL; Modulo 4 ----- 3 2.</p>
POWER (n,m)	Returnează o valoare egală cu $m$ la puterea $n$ .	<p>SELECT POWER(3,2) "Putere" FROM DUAL; Putere ----- 9</p>
ROUND (n[,m])	<p>Returnează <math>n</math> rotunjit la un număr de <math>m</math> zecimale. Dacă <math>m</math> este omis se elimină zecimalele, iar dacă este negativ se face rotunjirea numărului din dreapta virgulei zecimale, după regula: <math>m = -1</math> rotunjire la nivel de zeci, <math>m = -2</math> rotunjire la nivel de sute și așa mai departe.</p>	<p>SELECT ROUND(15.193,1) "Rotunjire" FROM DUAL; Rotunjire ----- 15.2</p> <p>SELECT ROUND(15.193,-1) "Rotunjire " FROM DUAL; Rotunjire ----- 20</p> <p>Aici rotunjirea s-a făcut la nivel de zeci, căci scala negativă înseamnă rotunjirea valorii din dreapta semnului zecimal, iar scala pozitivă înseamnă rotunjirea numărului din dreapta semnului zecimal la ordinul de mărime cât este scala.</p>
SIGN(n)	Returnează semnul numărului $n$ , după regula: $n < 0$ returnează valoarea -1; $n = 0$ returnează valoarea 0; $n > 0$ returnează valoarea +1.	<p>SELECT SIGN(-15) "Semn" FROM DUAL; Semn ----- -1</p>
SIN(n)	Returnează sinus de $n$ în radiani.	<p>SELECT SIN(30 * 3.14159265359/180) "Sinus de 30 de grade" FROM DUAL; Sinus de 30 de grade ----- .5</p>
SINH	Returnează sinus hiperbolic	SELECT SINH(1) " Sinus hiperbolic de

	de n.	1" FROM DUAL; Sinus hiperbolic de 1 ----- 1.17520119
SQRT(n)	Returnează rădăcină pătrată din n.	SELECT SQRT(26) "Rădăcină pătrată" FROM DUAL; Rădăcină pătrată ----- 5.09901951
TAN(n)	Returnează tangentă de n.	SELECT TAN(135 * 3.14/180) "Tangentă de 135 grade" FROM DUAL; Tangentă de 135 grade ----- - 1
TANH(n)	Returnează tangentă hiperbolică de n.	SELECT TANH(.5) "Tangentă hiperbolic de 5" FROM DUAL; Tangentă hiperbolic de 5 ----- .462117157
TRUNC (n,m)	Returnează valoarea lui n trunchiată la un număr de zecimale egal cu m. Dacă m este omis se elimină valorile zecimale, iar dacă ia o valoare negativă trunchiere se aplică părții din stânga virgulei zecimale, după regula: m = -1 rotunjire la nivel de zeci, m= -2 rotunjire la nivel de sute și așa mai departe.	SELECT TRUNC(15.79,-1) "Trunc" FROM DUAL; Trunc ----- 10

2) *Funcțiile caracter* acceptă la intrare valori caracter și furnizează valori caracter sau numerice. În tabelul 3.2 sunt prezentate funcțiile caracter care *returnează caractere*, iar în tabelul 3.3 funcțiile caracter care *returnează valori numerice*. Valorile caracter returnate sunt de tipul VARCHAR2 dacă nu se specifică altfel.

**Tabelul 3.2 Funcțiile caracter care returnează caractere**

<b>Funcția</b>	<b>Rolul funcției</b>	<b>Exemple</b>
CHR(n)	Returnează caracterul care are valoarea binară n.	SELECT CHR(67)  CHR(65)  CHR(84) "Caractere" FROM DUAL; Caractere

		--- CAT
CONCAT (c1,c2)	Returnează o valoare formată din concatenarea caracterului c1 cu caracterul c2.	SELECT CONCAT( CONCAT (nume, ' este '), funcție) "Funcție" FROM tabl WHERE codfuncție = 7000; Funcție ----- Popescu este PROGRAMATOR
INITCAP (șir')	Returnează șirul de caractere 'șir' astfel încât fiecare cuvânt al șirului are prima literă în format literă mare.	SELECT INITCAP('cuvânt1 cuvânt2') "Litere mari" FROM DUAL; Litere mari ----- Cuvânt1 Cuvânt2
LOWER (șir')	Returnează șirul de caractere 'șir' astfel încât toate literele șirului sunt de format literă mică.	SELECT LOWER('BUCUREȘTI') "Literă mică" FROM DUAL; Literă mică. ----- bucurești
LPAD (c1',n,'c2')	Returnează șirul de caractere 'c1', pe lungime de n caractere, astfel încât partea din stânga șirului până la lungimea de n caractere este umplută cu secvențe de caractere egale cu 'c2'. Dacă șirul 'c1' este mai lung decât valoarea n, atunci se returnează partea dreaptă a șirului pe lungime de n caractere.	SELECT LPAD('Page.1',10,'*') "LPAD exemplu" FROM DUAL; LPAD exemplu ----- *.*.*Pag.1
LTRIM ('c1' [, 'c2'])	Returnează partea din șirul 'c1' care a mai rămas după ce au fost înlăturate toate caracterele din stânga acestuia care se regăsesc în setul de caractere 'c2'.	SELECT LTRIM('xyxXxyREST ȘIR', 'xy') "LTRIM exemplu" FROM DUAL; LTRIM exemplu ----- XxyREST ȘIR
NLSSORT (c1' [, 'nlsparams'])	Returnează un șir de caractere folosite pentru sortarea câmpurilor de tip caracter. Caracterul 'c1' definește un marcator de sortare, în sensul că toate valorile dintr-un câmp care sunt mai mari sau mai mici decât acesta vor fi afișate sau nu prin folosirea	SELECT nume FROM tabl WHERE NLSSORT(nume,'NLS_SORT=romanian')< NLSSORT('O','NLS_SORT=romanian') ORDER BY nume; NUME -----

	<p>clauzei ORDER BY.</p> <p>Parametrul 'nlsparams' definește valoare lingvistică după care să se facă sortare și se dă sub forma 'NLS_SORT = sort' în care sort definește limba după care dorim să se facă sortarea(germană, română etc).</p>	<p>IONESCU MARINESCU POPESCU OLGA</p> <p>Notă: Numele care încep cu o literă mai mare decât 'O' nu se vor afișa.</p>
<p>REPLACE (<i>'c1','c2','c3'</i>)</p>	<p>Returnează șirul 'c1' translatat, astfel încât în șirul 'c1' toate valorile egale cu șirul de căutare 'c2' sunt înlocuite cu șirul de înlocuire 'c3'.</p>	<p>SELECT REPLACE ('MASĂ și MUSCA','M','C') "REPLACE" FROM DUAL; REPLACE ----- CASĂ și CUȘCĂ</p>
<p>RPAD (<i>'c1',n[, 'c2']</i>)</p>	<p>Returnează șirul de caractere 'c1', pe lungime de n caractere, astfel încât partea din dreapta șirului până la lungimea de n caractere este umplută cu secvențe de caractere egale cu 'c2'. Dacă șirul 'c1' este mai lung decât valoarea n, atunci se returnează partea stângă a șirului pe lungime de n caractere.</p>	<p>SELECT RPAD('CLUJ',10,'ab') "RPAD exemplu" FROM DUAL; RPAD exemplu ----- CLUJababab</p>
<p>RTRIM (<i>'c1','c2'</i>)</p>	<p>Returnează partea din șirul 'c1' care a mai rămas după ce au fost înlăturate toate caracterele din dreapta acestuia care se regăsesc în setul de caractere 'c2'.</p>	<p>SELECT RTRIM('BUCUREȘTȳxXxy','xy') "RTRIM exemplu" FROM DUAL; RTRIM exemplu ----- BUCUREȘTȳxX</p>
<p>SUBSTR (<i>'c1',m[,n]</i>)</p>	<p>Returnează porțiunea din șirul 'c1' care începe de la al m-lea caracter pe lungime de n caractere.</p>	<p>SELECT SUBSTR ('ABCDEFG',3,1,4) Subșir1" FROM DUAL; Subșir1 ---- CDEF</p> <p>SELECT SUBSTR('ABCDEFG',-5,4) "Subșir2" FROM DUAL; Subșir2 ---- CDEF</p>



<i>TRANSLATE ('c1','c2','c3')</i>	<p>1. Translatează șirul 'c1' prin intermediul șirului 'c2' la valorile din șirul 'c3' după regula: fiecare caracter din șirul c1 este căutat în șirul 'c2', dacă este găsit valoarea acestuia este înlocuită cu caracterul din șirul 'c3' a cărui poziție corespunde cu poziția caracterului din șirul 'c2'.</p> <p>2. Dacă șirul 'c2' este mai lung decât șirul 'c3' caracterele ce nu pot fi traduse sunt eliminate din șirul 'c1'.</p>	<pre>SELECT TRANSLATE ('2KRB229', '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ '9999999999XXXXXXXXXXXXXXXXXXXXX XXXXXXXX') "TRANSLATE 1" FROM DUAL; TRANSLATE 1 ----- 9XXX999  SELECT TRANSLATE ('2KRW229', '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ RSTUVWX', '0123456789') " TRANSLATE 2" FROM DUAL; TRANSLATE 2 ----- 2229</pre>
<i>UPPER('c1')</i>	Returnează șirul 'c1' cu toate caracterele transformate în caractere mari.	<pre>SELECT UPPER('București') "LITERE MARI" FROM DUAL; LITERE MARI ----- BUCUREȘTI</pre>

Tabelul 3.3 Funcțiile caracter care returnează valori numerice

<b>Funcția</b>	<b>Rolul funcției</b>	<b>Exemple</b>
<i>ASCII ('c1')</i>	Returnează valoarea zecimală a primului caracter din șirul 'c1'.	<pre>SELECT ASCII('Q') FROM DUAL; ASCII('Q') ----- 81</pre>
<i>INSTR ('c1','c2' [,n[,m]])</i>	<p>1. Caută în șirul 'c1' începând cu al n-lea său caracter a m-a apariție a șirului 'c2' și returnează poziția acestuia în șirul 'c1' relativ la începutul șirului. Dacă șirul 'c2' nu este găsit în șirul 'c1' se returnează valoarea 0. Valorile asumate prin lipsă pentru n și m sunt 1.</p> <p>2. Dacă n este negativ căutarea se face invers de la sfârșitul șirului.</p>	<pre>SELECT INSTR ('CORPORATE FLOOR', 'OR', 3, 2) "INSTR" FROM DUAL; INSTR ----- 14  SELECT INSTR ('CORPORATE FLOOR', 'OR', -3, 2) "INSTR INVERS" FROM DUAL; INSTR INVERS ----- 2</pre>
<i>LENGTH</i>	Returnează lungimea în	<i>SELECT LENGTH('BUCUREȘTI')</i>

	caractere a şirului de caractere 'c1' de tip CHAR.	"LUNGIME ŞIR" FROM DUAL; LUNGIME ŞIR ----- 9
--	--	--

3) *Funcţiile de tip DATE* operează cu tipuri de date de tip DATE şi sunt prezentate în tabelul 3.4. Toate funcţiile de tip DATE returnează valori de tip DATE, mai puţin funcţia MONTH\_BETWEEN care furnizează o valoare numerică. Structurile formatului *fmt* de afişare a datelor pentru funcţiile de tip DATE sunt prezentate în tabelul 3.7.

Tabelul 3.4 Funcţiile de tip DATE

<b>Funcţia</b>	<b>Rolul funcţiei</b>	<b>Exemple</b>
<i>ADD_MONTHS(d,n)</i>	Returnează data <i>d</i> plus un număr de luni egal cu <i>n</i> .	Dacă în coloana <i>data1</i> aferentă numelui 'POPESCU' din tabela <i>tab1</i> avem data 17 septembrie2005 cu comanda de mai jos se va ob'ine data 17 octombrie 2005. SELECT TO_CHAR (ADD_MONTHS( <i>data1</i> ,1), 'DD-MON-YYYY') "Luna următoare" FROM <i>tab1</i> WHERE nume = 'POPESCU'; Luna următoare ----- 17-OCT-2005
<i>LAST_DAY (d)</i>	Returnează data ultimei zile din lună.	Cu această funcţie se poate determina numărul zilelor rămase din luna curentă. SELECT SYSDATE, LAST_DAY(SYSDATE) "ULTIMA",LAST_DAY(SYSDATE) - SYSDATE "ZILE RĂMASE" FROM DUAL; SYSDATE      ULTIMA      ZILE RĂMASE -----      -----      ----- 23-SEP-05   30-SEP-05      7  SELECT TO_CHAR(ADD_MONTHS(LAST_DAY( <i>data1</i> ), 5), 'DD-MON-YYYY') "Cinci luni" FROM <i>tab1</i> WHERE nume = 'POPESCU'; Cinci luni ----- 28-FEB-2006
<i>MONTHS_BETWEEN (d1, d2)</i>	Returnează numărul de luni dintre datele <i>d1</i>	SELECT MONTHS_BETWEEN (TO_DATE('02-09-2005','MM-DD-YYYY'), TO_DATE('01-08-2005','MM-DD-YYYY'))

	și d2. Dacă d1 și d2 sunt aceleși zile din lună sau sunt ultimele zile din lună rezultatul este un număr întreg, altfel Oracle calculează fracțiuni din lună bazat pe o lună cu 31 zile.	"Luni" FROM DUAL; Luni ----- 1.03225806
NEXT_DAY (d, 'c1')	Returnează data primei zile a săptămânii după ziua definită de șirul 'c1' și care este după data d.	În exemplul de mai jos se returnează data zilei care urmează zilei de Marți, după data de 15 martie 1999. SELECT NEXT_DAY('15-MAR-05','TUESDAY') "ZIUA URMĂTOARE" FROM DUAL; ZIUA URMĂTOARE ----- 22-MAR-05
ROUND (d,[fmt])	Returnează data d rotunjită la unitatea de timp specificată de către formatul <b>fmt</b> , conform specificațiilor de la sfârșitul tabelului.	SELECT ROUND (TO_DATE ('27-SEP-05'),'YEAR') "Noul an" FROM DUAL; Noul an ----- 01-JAN-06
SYSDATE	Returnează data și timpul curent.	SELECT TO_CHAR(SYSDATE, 'MM-DD-YYYY HH24:MI:SS') "Data și timpul curent" FROM DUAL; Data și timpul curent ----- 27-09-2005 20:27:11
TRUNC (d,[fmt])	Returnează data d fără timp trunchiată la o unitate specificată de formatul <b>fmt</b> , iar dacă este absent se face trunchierea la ziua cea mai	SELECT TRUNC(TO_DATE ('27-SEP-05','DD-MON-YY'), 'YEAR') "Anul nou" FROM DUAL; Anul nou ----- 01-JAN-05

	<i>apropiată.</i>	
--	-------------------	--

<i>Formatul <b>fnt</b> utilizat de funcțiile <b>ROUND</b> și <b>TRUNC</b></i>	
<b>Formatul fnt</b>	<b>Semnificația formatului fnt</b>
CC, SCC	Se rotunjește la nivel de secol (primii doi digiți ai anului exprimat pe patru digiți + 1) <i>Exemplu: 1898 se rotunjește la 1998.</i>
YYYY, YYYY, YEAR SYEAR, YYY, YY, Y	Se rotunjește la nivelul 01 ianuarie a anului din data care se rotunjește. <i>Exemplu: 27-sep-05 se rotunjește la 01-jan-05.</i>
Q	Rotunjire la nivel de trimestru (Se rotunjește în sus la a șasesprezecea zi a lunii a doua a trimestrului).
MONTH, MON, MM, RM	Rotunjire la nivel de lună (Se rotunjește în sus la a șasesprezecea zi a lunii).

4) *Funcțiile de conversie* convertesc o valoare de la un tip de dată la alt tip de dată. Aceste funcții au formatul general de forma **tip\_de\_dată TO tip\_de\_dată** și sunt prezentate în tabelul 3.5.

**Tabelul 3.5** Funcțiile de conversie

<b>Funcția</b>	<b>Semnificația</b>	<b>Exemple</b>
CONVERT('c1', set_destinație, [set_sursă] )	Convertește un șir de caractere de la un set de caractere la alt set de caractere. Setul set_sursă este setul de caractere din care fac parte caracterele șirului 'c1', iar set_destinație este setul de caractere în care se convertesc caracterele șirului 'c1'.	SELECT CONVERT('Groß', 'US7ASCII', 'WE8HP') "Conversie" FROM DUAL;  Conversie ----- Gross Seturile de caractere cele mai comune sunt: US7ASCII, WE8DEC, WE8HP, F7DEC, WE8EBCDIC500 , WE8PC850, WE8ISO8859P1 .
HEXTOROW('c1')	Convertește șirul 'c1' care conține digiți hexazecimali la tipul de date RAW.	INSERT INTO tab1 (raw_column) SELECT HEXTORAW('7D') FROM DUAL;
RAWTOHEX(raw)	Convertește digiți hexazecimali de tip RAW la șirul 'c1'.	
ROWIDTOCHAR(rowid)	Convertește valoarea ROWID la o valoare de tip VARCHAR". Rezultatul conversiei	SELECT ROWID FROM tab1 WHERE ROWIDTOCHAR(ROWID) LIKE '%Br1AAB%';

	este tot impul un șir de 18 caractere.	ROWID ----- AAAAZ6AABAAABr1AAB
TO_CHAR pentru conversie de caractere, are sintaxa: TO_CHAR (d [,fmt])	Convertește data d de tip DATE la o valoare de tip VARCHAR2 în formatul specificat.	SELECT TO_CHAR (data1, 'Month DD, YYYY') "Format nou" FROM tab1 WHERE nume = 'ION'; Format nou ----- May 01, 2005
TO_CHAR pentru conversie de numere, are sintaxa: TO_CHAR (n [,fmt])	Convertește numărul n de tip NUMBER la o valoare de tip VARCHAR2.	SELECT TO_CHAR(-10000,'L99G999D99MI') "Cantitate" FROM DUAL; Cantitate ----- \$10,000.00-
CHARTOROWID ('c1')	Convertește o valoare de tip CHAR sau VARCHAR2 la o valoare de tip ROWID	SELECT nume FROM tab1 WHERE ROWID = CHARTOROWID ('AAAAfZAABAAACp8AAO'); nume ----- POPESCU
TO_DATE ('c1' [,fmt])	Convertește șirul de caractere 'c1' de tip CHAR sau VARCHAR2 la o valoare de tip DATE în conformitate cu formatul fmt . Formatul fmt pentru datele de tip DATE este prezentat în tabelul 3.7.	INSERT INTO tab1 (data1) SELECT TO_DATE('January 15, 2005, 11:00 A.M.', 'Month dd, YYYY, HH:MI A.M.') FROM DUAL;
TO_NUMBER ('c1',[fmt])	Convertește șirul de caractere 'c1' de tip CHAR sau VARCHAR2 la o valoare numerică de tip NUMBER în conformitate cu formatul fmt. Forma acestui format este în tabelul 3.6.	UPDATE tab1 SET salariu = salariu + TO_NUMBER('100.00', '9G999D99') WHERE nume = 'ION';

**Tabelul 3.6** Structurile formatului **fmt** pentru datele de tip NUMBER

Elementul <i>fmt</i>	Exemple	Semnificația elementului <i>fmt</i>
9	9999	Nr. semnificativ de digiți care vor fi afișați
0	0999 9990	Afișează 0 în fața sau după digiții semnificativi

\$	\$999	Afișează semnul \$ în fața numărului
B	B999	Afișează valorile 0 ca blank-uri
MI	999MI	Afișează semnul “-” după numerele negative
S	S999	Afișează semnul “+” sau “-” în fața numerelor pozitive, respectiv negative
D	99D99	Afișează punctul zecimal în această poziție
G	99G999	Separator de grupuri
,	99,999,99	Virgula se afișează în pozițiile indicate
.	999.99	Afișează punctul zecimal în această poziție
RN sau rn	RN sau rn	Afișează cifre romane cu majuscule, respectiv cu caractere mici(minusculе)

Tabelul 3.7 Structurile formatului fmt pentru datele de tip DATE

<b>Elementul fmt</b>	<b>Se specifică în TO-DATE</b>	<b>Semnificația elementului fmt</b>
- / , . ; :	Da	Punctuații pentru dată
"text"	Da	Text reprodus în rezultat
AD sau A.D.	Da	Specificarea unui an din Era Noastră(E.N.)
BC sau B.C.	Da	Specificarea unui an Înaintea Ereii Noastre(Î.E.N.)
CC sau SCC	Nu	Secolul = cu primii doi digiți ai anului pe patru digiți + 1
D	Da	Ziua din săptămână(de la 1 la 7)
DAY	Da	Numele zilei
DD	Da	Ziua din lună(de la 1 la 31)
DDD	Da	Ziua din an(1 la 366)
HH sau HH12	Da	Ora din zi din intervalul 1 - 12
HH24	Da	Ora din zi din intervalul 1 - 24
J	Da	Ziua din calendarul Iulian cu valori începând cu 1 ianuarie 4712 BC. Trebuie să fi întreg.
MI	Da	Minute (0 la 59)
MM	Da	Luna (01 la 12)
MON	Da	Numele prescurtat al lunii
MONTH	Da	Numele întreg al lunii
PM sau P.M.	Nu	Indicator de meridian
RR sau RRRR	Da	Rotunjirea anului pe doi digiți sau patru digiți
SS	Da	Secunde în cadrul minutului(0 la 59)
SSSS	Da	Secunde de la miezul nopții în cadrul zilei(0 la 86399)
WW	Nu	Săptămâna din an (1 la 53)
W	Nu	Săptămâna în cadrul lunii(1 la 5)
YYYY,YYY,YY,Y	Da	Anul cu 4 , 3, 2, 1 digiți.

Alte funcții cu un singur rând sunt prezentate în tabelul 3.8.

**Tabelul 3.8 Alte funcții cu un singur rând**

<b>Funcția</b>	<b>Rolul funcției</b>	<b>Exemple</b>
<i>DUMP ('c1' [,return_format [,start_position [,length]]])</i>	<i>Returnează o valoare de tip VARCHAR2 conținând codul tipului de date, lungimea în bii și reprezentarea internă a expresiei expr return_format este codul sistemului de numerație în care este reprezentată valoarea returnată de funcție pentru șirul 'c1'; Dacă se furnizează ca o valoare egală cu codul sistemului de numerație + 1000 se returnează și numele sistemului de numerație. start_position determină poziția din șirul 'c1' de unde să înceapă DUMP-u length este lungimea DUMP-ului.</i>	<i>SELECT DUMP('abc', 1016) FROM DUAL; DUMP('ABC',1016) ----- Typ=96 Len=3 CharacterSet=WE8DEC: 61,62,63  SELECT DUMP(nume, 8, 3, 2) "OCTAL" FROM tabl WHERE nume = 'SCOTT'; OCTAL ----- Type=1 Len=5: 117,124 Sistemele de numerație sunt: 8 = sistemul octal; 10 = sistemul zecimal; 16 = sistemul hexazecimal; 17 = rezultatul este returnat sub forma de caractere singulare.</i>
<i>EMPTY_ [B C LOB]()</i>	<i>Returnează un pointer sau locator care poate fi folosit pentru inițializare unei variabile de tip LOB, într-un INSERT, într-un UPDATE pentru inițializarea unei coloane de tip LOB sau ca atribut EMPTY, care înseamnă că LOB-ul a fost inițializat dar nu a fost populat cu date.</i>	<i>INSERT INTO lob_tabl VALUES (EMPTY_BLOB());  UPDATE lob_tabl SET clob_col = EMPTY_BLOB();</i>
<i>BFILENAME ( 'director', 'nume_fișier')</i>	<i>Returnează un pointer sau locator asociat cu un LOB de tipul unui fișier binar de pe server. director este numele directorului unde se află fișierul LOB de tip binar. nume_fișier este numele fișierului.</i>	<i>INSERT INTO tabl VALUES (BFILENAME('lob_dir1','foto1.gif')) ;</i>
<i>GREATEST (expr [,expr] ...)</i>	<i>Returnează valoarea cea mai mare dint-o listă de valori.</i>	<i>SELECT GREATEST ('HARRY', 'HARRIOT', 'HAROLD') "Mare" FROM DUAL; Mare -----</i>

		<i>HARRY</i>
<i>LEAST (expr [,expr] ...)</i>	<i>Returnează valoarea cea mai mică dint-o listă de valori.</i>	<i>SELECT LEAST('HARRY','HARRIOT','HAR OLD') "Mică" FROM DUAL; Mică ----- HAROLD</i>
<i>NVL (expr1, expr2)</i>	<i>Dacă expr1 este NULL returnează expresia expr2, iar dacă expr1 nu este NULL returnează expr1. Argumentele expr1 și expr2 pot avea orice tip de date. Dacă sunt de tipuri diferite Oracle convertește expr2 la tipul expr1 înainte de a executa comparațiile. Valoarea returnată este totdeauna de tipul expr1, excepție făcând situația când expr1 este de tip caracter, caz în care rezultatul este VARCHAR2.</i>	<i>SELECT nume, NVL(TO_CHAR(comision), 'NOT APPLICABLE') "COMISION" FROM tabl WHERE codepart = 30; NUME COMISION ----- ALLEN 300 WARD 500 MARTIN 1400 TURNER 0 JAMES NOT APPLICABLE</i>
<i>UID</i>	<i>Returnează un întreg care identifică în mod unic utilizatorul curent.</i>	
<i>USER</i>	<i>Returnează identificatorul utilizatorului curent în format VARCHAR2.</i>	<i>SELECT USER, UID FROM DUAL; USER UID ----- SCOTT 19</i>
<i>USERENV (option)</i>	<i>Returnează informații despre sesiune curentă.</i>	<i>SELECT USERENV('LANGUAGE') "Limbaajul" FROM DUAL; Limbaajul ----- AMERICAN AMERICA.WE8DEC</i>
<i>VSIZE(expr)</i>	<i>Returnează lungimea în baiți a expresiei expr.</i>	<i>SELECT nume, VSIZE (nume) "BYTES" FROM tabl WHERE codepart = 10; NUME BYTES ----- CLARK 5 KING 4 MILLER 6</i>



## B.Funcțiile cu mai multe randuri (de grup)

Furnizează un rezultat bazat pe prelucrarea mai multor rânduri. Toate funcțiile de grup, mai puțin COUNT(\*) ignoră valorile NULL.

Majoritatea funcțiilor de grup acceptă opțiunile: **DISTINCT** (determină luarea în calcul numai a valorilor distincte ale rândurilor din cadrul grupului) și **ALL** (determină luarea în calcul a tuturor valorilor grupului de rânduri).

Funcțiile de grup sunt prezentate în tabelul 3.9.

Tabelul 3.9 Funcțiile de grup

<i><b>Funcția</b></i>	<i><b>Semnificația</b></i>	<i><b>Exemple</b></i>
<i>AVG([DISTINCT ALL] n)</i>	<i>Returnează media celor n valori</i>	<i>SELECT AVG(salariu) "Medie"</i> <i>FROM tab1;</i> <i>Media</i> <i>-----</i> <i>2077343.21429</i>
<i>COUNT ({*   [DISTINCT ALL] expr})</i>	<i>Returnează toate rândurile cererii. Dacă avem argumentul = * se returnează toate rândurile indiferent de valoarea lor (NULL sau NOT NULL)</i>	<i>SELECT COUNT(*) "Total"</i> <i>FROM tab1;</i> <i>Total</i> <i>-----</i> <i>18</i>  <i>SELECT COUNT(job) "Count"</i> <i>FROM tab1;</i> <i>Count</i> <i>-----</i> <i>14</i>  <i>SELECT COUNT(DISTINCT job) "Jobs"</i> <i>FROM emp;</i> <i>Jobs</i> <i>-----</i> <i>5</i>
<i>MAX([DISTINCT ALL] expr)</i>	<i>Returnează maximumul din expresia expr.</i>	<i>SELECT MAX(salariu)</i> <i>"Maximum"</i> <i>FROM tab1;</i> <i>Maximum</i> <i>-----</i> <i>5000</i>
<i>MIN([DISTINCT ALL] expr)</i>		<i>SELECT MIN(data1) "Minim"</i> <i>FROM tab1;</i> <i>Minimum</i>

		----- 17-DEC-80
<i>SUM([DISTINCT ALL] n)</i>		SELECT SUM(salariu) "Total" FROM tabl; Total ----- 29081

Utilizatorii pot să-și scrie propriile funcții în PL/SQL pentru a executa activități neacoperite de către funcțiile SQL. Aceste funcții pot fi folosite în comenzile SQL la fel ca și cele standard.

De exemplu, funcțiile utilizator **pot fi folosite** în: lista de selecție a comenzii SELECT; condiția din clauza WHERE; clauzele CONNECT BY, START WITH, ORDER BY ȘI GROUP BY ; clauza VALUES a comenzii INSERT; clauza SET a comenzii UPDATE.

Funcțiile utilizator **nu pot** fi folosite în clauzele CONSTRAINT sau DEFAULT ale comenzilor CREATE TABLE sau ALTER TABLE și nici pentru actualizarea bazei de date. În funcțiile utilizator nu este permisă utilizarea parametrilor OUT sau IN OUT.

### 3.4. EXPRESII SQL

**Expresia** este o combinație de unul sau mai mulți operatori, operanzi (variabile, literal, coloane, funcții SQL) care se evaluează la o singură valoare.

**Operatorii** utilizați în comenzile SQL sunt: operatori SQL\*PLUS; operatori SQL; operatori aritmetici; operatori logici; operatori specifici în expresiile de cereri. Ei sunt prezentați în Anexa 2, în ordinea descrescătoare a priorității, cei de aceeași importanță fiind grupați între perechi de linii orizontale. Operatorii sunt evaluați de la stînga spre dreapta.

O expresie are în general același tip de date ca și componentele sale. Expresiile sunt folosite pentru construirea instrucțiunilor SQL și a unor liste de expresii.

**A.** Există cinci forme de furnizarea a expresiilor pentru **construirea instrucțiunilor SQL**.

1) *Forma I* este formată din coloană, pseudocoloană, constantă, secvență sau NULL și are sintaxa:

**nume\_schemă.tabelă | vedere. coloană | pseudocoloană | ROWLABEL**  
sau

**text | număr | nume\_secvență | nume secvență. CURRVAL | NEXTVAL**  
**| NULL**

Pentru *nume\_schemă* se poate folosi pe lângă numele schemei utilizatorului și valoarea "PUBLIC", care califică sinonimele publice pentru o tabelă sau o vedere, calificare care este suportată doar în instrucțiunile SQL pentru manipularea datelor de tip (DDL), nu și în cele de definire a datelor de tip DDL.

*Pseudocoloană* poate fi doar LEVEL, ROWID sau ROWNUM.

**Exemple:**

***Tab1.numecol\_1***  
***'acesta este un șir de caractere'***  
***10***  
***secventa1.CURRVAL***

2) *Forma II* este folosită pentru definirea unei variabile gazdă (host variable) cu sau fără indicator de variabilă. Expresiile din această formă se vor folosi doar în instrucțiunile SQL incluse în limbajele de programare gazdă sau în instrucțiunile prelucrate de programele OCI(Oracle Call Interface). Sintaxa este de forma:

**:variabilă\_gazdă INDICATOR :variabilă\_indicator**

**Exemple:**

***:nume\_angajat INDICATOR :indicator\_var\_nume\_angajat***  
***:nume\_persoană***

3) *Forma III* este folosită pentru apelul funcțiilor cu un singur rând și are sintaxa:

**funcție (DISTINCT | ALL expresie1, expresie2, ... )**

**Exemple :**

***LENGTH('BLAKE')***  
***ROUND(1234.567\*43)***  
***SYSDATE***

4) *Forma IV* este folosită pentru apelul funcțiilor de utilizator și are sintaxa:

**nume\_schemă . nume\_pachet. nume\_funcție**

**Exemple:**

***aria\_cercului(raza)***  
***calcul\_rate(nume\_angajat)***

5) Forma *V* este o combinație de mai multe expresii și are sintaxa:  
( *expresie* ) | + | - | PRIOR *expresie* | *expresie1* \* | / | - | || *expresie2*

**Exemple:**

*('IONESCU' || 'PETRE')*  
*LENGTH('BUCURESTI') \* 57*  
*SQRT(144) + 72*  
*funcție\_utilizator(TO\_CHAR(sysdate,'DD-MMM-YY'))*

Expresiile pentru decodificarea unor valori folosesc sintaxa specială de tip **DECODE** de forma:

**DECODE ( *expr*, *val1*, *rezultat1*, *val2*, *rezultat2*, .... , *valoare\_asumată* )**

*expr* va fi evaluată și apoi valoarea rezultată va fi comparată cu fiecare dintre valorile *val1*, *val2*, .... . Dacă valoarea expresiei este egală cu una din valorile de comparație se va furniza valoarea rezultat (*rezultat1*, *rezultat2*, ...) care corespunde valorii de comparație. Dacă valoarea expresiei *expr1* nu este egală cu nici una din valorile de comparație se furnizează valoarea *valoare\_asumată*. Dacă valoarea asumată este omisă atunci se furnizează valoarea **NULL**.

**Exemplu:**

*DECODE (cod\_funcție,10, 'programator',*  
*20, 'cercetător',*  
*30, 'vânzător',*  
*40, 'operatorr',*  
*'lipsă\_funcție')*

Exemplul de mai sus decodifică valoarea expresiei *cod\_funcție*. Astfel dacă valoarea expresiei *cod\_funcție* = 10 se furnizează funcția *programator*, dacă este = 20 se furnizează funcția *cercetător* și așa mai departe. În caz că expresia *cod\_funcție* are a valoare care nu este egală cu nici una din valorile 10, 20, 30 sau 40 se furnizează ca rezultat valoarea *lipsă\_funcție*.

**B. O listă de expresii** este o serie de expresii separate între ele prin virgulă și închisă între paranteze rotunde și poate conține pînă la maximum 1000 de expresii.

**Exemplu:**

(10, 20, 40)

('SCOTT', 'BLAKE', 'TAYLOR')

(LENGTH('MOOSE') \* 57, -SQRT(144) + 72, 69)

### 3.5. CONDIȚIILE

**Condiția** este o combinație de una sau mai multe expresii și operatori logici evaluată cu una din valorile TRUE, FALSE sau UNKNOWN. Condițiile se pot folosi în clauza WHERE a instrucțiunilor SQL **DELETE**, **SELECT** și **UPDATE** sau într-una din clauzele **WHERE**, **START WITH**, **CONNECT BY** sau **HAVING** ale instrucțiunii **SELECT**.

**Exemple:**

1) Expresia  $I=I$  este evaluată ca TRUE

2) Expresia  $NVL(sal, 0) + NVL(comm, 0) > 2500$  adună valoarea sal cu comm și evaluează rezultatul dacă este  $\geq$  cu 2500000. Dacă sal sau comm sunt NULL se vor înlocui cu valoarea zero.

Condițiile au opt forme de prezentare.

*Forma I* este folosită pentru compararea unei expresii cu altă expresie sau cu o subcerere și are sintaxa:

**expresie1** = | != | <> | > | < | >= | <= **expresie2** | (subcerere)

sau

**listă\_de\_expresii** = | != | <> (subcerere)

*Forma II* este folosită pentru compararea unei expresii sau a unei liste de expresii cu unul sau toți membrii unei liste de expresii sau ai unei subcereri și are sintaxa:

**expresie1** = | != | <> | > | < | >= | <= **ANY** | **SOME** | **ALL**

**listă\_de\_expresii** | (subcerere)

sau

**listă\_de\_exp** = | != | <> **ANY** | **SOME** | **ALL** (**listă\_expr1**,  
**listă\_expr2**, ...) | (subcerere1, subcerere2, .... )

*Forma III* este folosită pentru căutarea unei expresii sau a unei liste de expresii dacă figurează într-o listă de expresii sau într-o subcerere și are sintaxa:

**expresie1 NOT IN** **listă\_de\_expresii** | (subcerere)

sau

**listă\_de\_exp NOT IN ( listă\_expr1, listă\_expr2, ...) |  
(subcerere1, subcerere2, ....)**

*Forma IV* este folosită pentru testarea existenței sau inexistenței expresiei între două limite și are sintaxa:

**expresie1 NOT BETWEEN expr2 AND expr3**

*Forma V* este folosită pentru testarea valorii NULL și are sintaxa:

**expresie1 IS NOT NULL**

*Forma VI* este folosită pentru testarea existenței unui rând într-o subcerere și are sintaxa:

**EXISTS ( subcerere )**

*Forma VII* este folosită pentru testarea egalității unei șir de caractere cu un format anume, cu un alt șir de caracter și are sintaxa:

**șir\_ caracter1 NOT LIKE șir\_cacra2 ESCAPE  
'caracter\_de\_schimbare'**

*Forma VIII* este folosită pentru specificarea unei combinații de mai multe condiții și poate avea sintaxele:

**NOT condiție**

sau

**condiție1 NOT AND | OR condiție2**

### ***3.6. DESCHIDERA ȘI ÎNCHIDERA UNEI SESIUNI DE LUCRU SQL\*PLUS***

Pentru a avea acces la componentele unei baze de date (tabele, viziuni, clustere etc.), utilizatorul trebuie mai întâi să se conecteze la ea. La sfârșitul sau în timpul unei sesiuni SQL există posibilitatea deconectării de la baza de date. O primă conectare se face atunci când se începe lucrul cu SQL\*Plus. Efectul comenzii sau utilizării meniului sistem îl constituie deschiderea unei sesiuni de lucru sub SQL\*Plus.

Comanda are următoarele forme sintactice:

**SQLPLUS [nume-utilizator[/parolă]**

**[@nume-bază-de-date]**

**[@nume-fișier]**

**[-SILENT]**

**SQLPLUS /NOLOG [-SILENT]**

**SQLPLUS -?**

unde:

*nume-utilizator* și *parolă*: sunt numele și parola unui utilizator cu drept de acces la baza de date.

*@nume-bază-de-date*: este numele unei baze de date cu care se lucrează în rețea (este alt nume decât cel al bazei de date implicite, de pe calculatorul la care se lucrează).

*@nume-fișier*: reprezintă numele unui fișier de comenzi SQL care va fi rulat de SQL\*Plus.

*SILENT*: are ca efect inhibarea facilității de afișare a tuturor informațiilor și mesajelor furnizate de SQL\*Plus

*/NOLOG*: lansează în execuție SQL\*Plus dar nu face conectarea la o bază de date.

*-?*: are ca efect afișarea versiunii curente a componentei SQL\*Plus, după care returnează controlul sistemului de operare

Dacă în linia de comandă se specifică *parola* și *@nume-bază-de-date* atunci între ele nu trebuie să existe spațiu. *Parola* sau *@nume-bază-de-date* vor fi separate printr-un spațiu de *@nume-fișier*.

### ***Conectarea utilizatorului la o bază de date***

Dacă în timpul unei sesiuni de lucru SQL\*PLUS se dorește conectarea la o altă bază de date decât cea deschisă inițial se poate folosi comanda **CONNECT**.

Sintaxa acestei comenzi este:

**CONN[ECT] [nume-utilizator[/parolă]]  
[@nume-bază-de-date];**

unde:

*nume-utilizator* și *parolă*: sunt numele unui utilizator cu drepturi de acces la baza de date și parola unui astfel de utilizator. Chiar dacă nu se specifică parola (în linia de comandă este opțională), ea este cerută de sistem printr-un mesaj explicit. Parola introdusă la mesajul sistemului va fi invizibilă.

*@nume-bază-de-date*: se specifică în cazul lucrului în rețea, când se dorește conectarea la o altă bază de date decât cea aflată pe calculatorul la care se lucrează.

De remarcat faptul că în timpul unei sesiuni de lucru cu SQL\*Plus se poate realiza conectarea la o bază de date, fără a mai fi necesară închiderea sesiunii.

Furnizarea parametrilor nume-utilizator și parolă asigură protecția bazei de date împotriva accesului neautorizat.

### ***Deconectarea utilizatorului de la baza de date***

Deconectarea utilizatorului de la o bază de date se realizează prin comanda **DISCONNECT**.

Sintaxa acestei comenzi este:

**DISC[ONNECT];**

Comanda are ca efect deconectarea de la baza de date curentă, fără a închide sesiunea de lucru SQL\*Plus.

### ***Închiderea sesiunii de lucru SQL\*Plus***

Închiderea sesiunii de lucru SQL\*PLUS și predarea controlului sistemului de operare al calculatorului gazdă se realizează cu una din comenzile: **QUIT**, **EXIT** sau **^Z**.

Sintaxa acestor comenzi este:

**QUIT;**

**EXIT:**

**^Z;**

## ***3.7. ELEMENTE DE LUCRU CU SQL\*PLUS***

### **A. Încărcarea și executarea comenzilor**

În exemplele ce se vor prezenta, referitor la utilizarea comenzilor SQL\*Plus, se va folosi tabela Oracle **pers100** cu structura de mai jos:

```
CREATE TABLE pers100
(CODPERS NUMBER(5),
NUME VARCHAR2(30),
PRENUME VARCHAR2(30),
ZINAST NUMBER(2),
LUNAST NUMBER(2),
ANAST NUMBER(4),
STRADA VARCHAR2(40),
NRSTR NUMBER(2),
SECTOR NUMBER(1),
LOCNAST VARCHAR(20),
FUNCTIA VARCHAR(15),
SALARIU NUMBER(8),
NUMAR_ACTIUNI NUMBER (6))
```



PCTFREE 5 PCTUSED 75;

Comenzile se pot introduce pe *una* sau *mai multe linii*.

**Exemplu:**

Comanda

***sql> select \* from pers100;***

este echivalentă cu:

***sql> select***

***2 \* from***

***3 pers100;***

Introducerea comentariilor se poate realiza folosind:

- Comanda REMARK din SQL\*Plus

**Exemplu:**

REMARK comentariu

Nu se vor introduce comentarii între liniile aceleași comenzi SQL.

- Delimitatorii de comentariu din SQL */\* ... \*/*

**Exemplu:**

*/\* comentariu \*/*

- comentariile tip PL/SQL prefixate cu *--*

**Exemplu:**

*-- comentariu*

Terminarea unei comenzi SQL se face cu: punct și virgulă (;), slash (/) sau blank. Primele două forme cer SQL\*Plus să execute imediat comanda, a treia formă nu. Comanda curentă poate fi rulată sau rerulată introducând comenzile RUN sau slash(/).

Se pot introduce și blocuri PL/SQL care să fie executate. La sfârșitul blocurilor PL/SQL se vor insera două linii una conținând un *punct*, iar cealaltă un slash(/).

**Exemplu:**

***declare***

***x number := 100;***

***begin***

***for i in 1 .. 10 loop***

***insert into pers100 values***

***(10, 'ionel', 'marin', 10,7,1970, 'ion manolescu',2, 6, 'suceava');***

***end loop;***

***end;***

***/***

Zona (aria) în care SQL\*Plus memorează comenzile curente se numește *buffer-ul SQL*.

Pe lângă comenzile SQL și PL/SQL se pot introduce și comenzi SQL\*Plus, care pot fi abreviate și la o literă. Comenzile foarte lungi pot fi întrerupte și continuate pe linia următoare. Întreruperea se marchează cu ‘-’. Oracle automat afișează semnul > (prompter) după care se introduce restul comenzii.

**Exemplu:**

```
sql> select * from -  
>pers100;
```

Controlul listării rapoartelor lungi se poate face utilizând tasta Cancel sau setând variabila PAUSE pe ON.

**Exemplu:**

```
set pause 'text de atenționare'  
set pause on
```

Această setare va determina ca sistemul să oprească afișarea la fiecare pagină și să afișeze textul text de atenționare. Cu SET PAUSE OFF se revine la starea anterioară.

**B. Editarea comenzilor SQL\*Plus**

Editarea în mod linie se realizează prin comenzile din tabelul 3.10.

**Tabelul 3.10. Comenzile de editare a comenzilor SQL\*Plus**

Comanda	Abreviația	Funcția
APPEND text	A text	Adaugă text la sfârșitul unei linii
CHANGE /old/new/	C /old/new/	<b>Schimbă un text cu altul</b>
CHANGE /text	C/text	Șterge textul unei linii
CLEAR BUFFER	CL BUFF	Curăță bufferul
DEL	Fără abreviație	Șterge o linie
INPUT	I	Adaugă una sau mai multe linii
INPUT text	I text	Adaugă o linie formată dintr-un text
LIST	L	Listează toate liniile din buffer
LIST n	L n sau n	Listează linia n
LIST *	L *	Listează linia curentă
LIST LAST	L LAST	Listează ultima linie
LIST m n	L m n	Listează liniile de la m la n

Editarea comenzilor cu editorul sistemului se realizează cu comanda **EDIT**. Apare fereastra Editorului, în care se vor tasta instrucțiunile SQL dorite. Se salvează fișierul cu **nume.SQL** și se execută cu comanda **@nume**.

**C. Crearea, regăsirea, rularea și modificarea fișierelor de comenzi.**

Crearea fișierelor de comenzi se pot realiza prin:

- Salvarea conținutului bufferului cu comanda **SAVE**.

**Exemplu:**

**SAVE nume\_fișier.SQL**

Înainte de salvare se va lista bufferul cu comanda **LIST** pentru a verifica dacă instrucțiunile ce vor fi salvate sunt corecte.

- Utilizarea comenzii **INPUT** în corelație cu **SAVE**

**Exemplu:**

**sql> clear buffer**

**sql> input**

**select \* from pers100**

**sql> save comand1.sql**

**sql> @comand1**

- Utilizarea editorului de sistem

**SQL> EDIT nume\_fișier**

Regăsirea (citirea) fișierelor de comenzi se face cu comanda **GET**.

**Exemplu:**

**SQL> GET edit3**

**SELECT \* FROM PERS100;**

Rularea fișierelor de comenzi se execută folosind comenzile:

**START nume\_fișier**

**@nume\_fișier**

Dacă avem mai multe fișiere de comenzi pe care vrem să le executăm secvențial, vom crea un fișier de comenzi conținând comenzile **START** corespunzătoare, după care pentru rulare vom activa acest ultim fișier.

**Exemplu:**

În fișierul **FILE2** introducem comenzile:

**START fiș1**

**START fiș2**

**START fiș3**

Apoi cu una din comenzile **START FILE2** sau **@FILE2** vom activa aceste comenzi.

Modificarea fișierelor de comenzi se poate realiza folosind comanda **EDIT nume\_fișier** sau comanda **GET** urmată de **SAVE nume\_fișier REPLACE**.

#### **D. Facilități pentru setarea fișierelor de comenzi**

Următoarele facilități fac posibilă setarea unor fișiere de comenzi care să permită utilizatorilor să-și introducă propriile valori pentru controlul

execuției comenzilor: definirea variabilelor de utilizator; ștergerea variabilelor de utilizator; substituirea valorilor în comenzi; folosirea comenzii START pentru furnizarea de valori în comenzi; crearea unor prompteri pentru introducerea valorilor.

1) *Definirea variabilelor de utilizator* se face explicit cu comanda **DEFINE** sau implicit prin utilizarea prefixării variabilelor cu două '&'.

Definirea, listarea și ștergerea unei variabile utilizator în mod explicit se fac cu comenzile:

**DEFINE** *nume\_variabilă* = "valoare variabila"

**Exemplu:**

**SQL > DEFINE** *variabila1* = "mihai"

2) *Ștergerea variabilelor de utilizator* se realizează prin utilizarea comenzii

**UNDEFINE** *nume\_variabilă*.

**Exemplu:**

**SQL> UNDEFINE** *variabila1*

3) *Substituirea variabilelor* este o tehnică prin care putem crea proceduri de lucru astfel încât să folosim același script (grup de instrucțiuni) pentru efectuarea unor funcții diferite pornind de la structura procedurii. Variabilele ce se substituie pot exista la momentul substituirii dacă anterior au fost create explicit cu comanda **DEFINE** sau implicit prin prefixare cu &. Există patru modalități de substituire a variabilelor: substituirea variabilelor *prefixate cu un '&'*; substituirea variabilelor *prefixate cu două '&'*; substituirea variabilelor de tip &n cu ajutorul comenzii **START**; substituirea variabilelor folosind comenzile **PROMPT**, **ACCEPT** și **PAUSE**

a) *Substituirea variabilelor prefixate cu un '&'*. Vom crea o procedură generalizată pentru calculul unor subgrupe statistice pe o coloană numerică.

**Exemplu:**

Să se execute selecția din baza de date a valorilor salariilor aparținând aceleași funcții, iar din aceste grupe afișarea celor care sunt cele mai mici din grupă, ordonate descrescător.

**select** *funcția* , **min**(*salariu*) **minimum**

**from** *pers100*

**group by** *funcția*

**order by** **min**(*salariu*) **desc**;

În fraza **SELECT** de mai sus *funcția*, *min*, *salariu* și *desc* pot fi definite ca variabile, ceea ce va permite ca să putem utiliza pentru grupare și altă coloană, pentru calcul și valorile max sau sum, iar pentru ordonare vom putea folosi și valoarea ascendent. Pentru executarea acestei comenzi vom crea fișierul de comenzi **CALC&.SQL**, cu structura de mai jos:

```

/* - -----
--*/
/* &v1_col_grup = nume coloană din tabelă după valorile căreia se
va face gruparea
/* &v2_tip_calc = tipul calculului: min, max, sum pentru un anumit
grup de valori numerice
/* &v3_col_calc = numele coloanei de tip numeric după care se va
face calculul
/*&v4_nume_col_calculat = numele coloanei, în listă, pe care se
vor afișa valorile calculate
/* &v5_tip_sort = tipul ordonării(sortării), desc(descending) sau
asc(ascending)
SELECT &v1_col_grup,
        &v2_tip_calc(&v3_col_calc),
        &v4_nume_col_calculată
FROM   pers100
GROUP BY &v1_col_grup
ORDER BY &v2_tip_calc(&v3_col_calc,) &v5_tip_sort;
/* - -----
--*/

```

După apelul fișierului cu comanda @calc& sistemul ne va cere succesiv să furnizăm valorile dorite pentru variabilele definite astfel:

*Enter value for v1\_col\_grup: nume*

*Enter value for v2\_tip\_calc: min*

*Enter value for v3\_col\_calc: salariu*

*Enter value for v4\_nume\_col\_calculat: MINIMUM*

*Enter value for v1\_col\_grup: nume*

*Enter value for v2\_tip\_calc: min*

*Enter value for v3\_col\_calc: salariu*

*Enter value for v5\_tip\_sort: desc*

Iar rezultatul după executare comenzii va arăta ca mai jos:

<i>NUME</i>	<i>MINIMUM</i>
-------------	----------------

<i>ionescu</i>	<i>3500000</i>
----------------	----------------

<i>petrescu</i>	<i>2500000</i>
-----------------	----------------

<i>mihai</i>	<i>1500000</i>
--------------	----------------

b) Substituirea variabilelor prefixate cu două '&'. Pentru exemplificare vom crea, pornind de la fișierul de comenzi CALC&.SQL, fișierul de comenzi CALC&&.SQL în care variabilele vor fi prefixate cu doua caractere &, ca mai jos:

```

/* -----
*/
/* &&v1_col_grup = nume coloană din tabelă după valorile căreia
se va face gruparea
/* &&v2_tip_calc = tipul calculului: min, max, sum pentru un
anumit grup de valori numerice
/* &&v3_col_calc = numele coloanei de tip numeric după care se
va face calculul
/*&&v4_nume_col_calculat = numele coloanei, în listă, pe care se
vor afișa valorile calculate
/* &&v5_tip_sort = tipul ordonării(sortării), desc(descending) sau
asc(ascending) */
SELECT &&v1_col_grup,
           &&v2_tip_calc(&&v3_col_calc),
           &&v4_nume_col_calculat
FROM pers100
GROUP BY &&v1_col_grup
ORDER BY &&v2_tip_calc(&&v3_col_calc), &&v5_tip_sort
/* -----*/

```

La momentul executării fișierului de comenzi CALC&&.Sql sistemul va cere să introducem valorile dorite pentru variabilele definite la fel ca la apelul precedent cu deosebirea că valorile ce le vom furniza vor fi memorate de fiecare dată astfel că indiferent de câte ori apare o variabilă pentru ea se va furniza valoarea o singură dată. Orice execuție ulterioară a unui fișier de comenzi în care apare una din variabilele create anterior (definite implicit ca variabile utilizator) se vor folosi aceste valori. Execuția acestui fișier de comenzi va produce același rezultat ca și execuția fișierului CALC&.SQL. Rulând comanda DEFINE vom găsi în sistem pe lângă alte variabile utilizator și variabilele create prin execuția fișierului CALC&.SQL.

**Exemplu:**

```

SQL> DEFINE
DEFINE _EDITOR      = "Notepad" (CHAR)
DEFINE _RC          = "1" (CHAR)
DEFINE V1_COL_GRP   = "nume" (CHAR)
DEFINE V2_TIP_CALC  = "min" (CHAR)
DEFINE V3_COL_CALC  = "salariu" (CHAR)
DEFINE V4_NUME_COL_CALCULAT = "minimum" (CHAR)
DEFINE V5_TIP_SORT  = "desc" (CHAR)

```

Dacă vom dori rularea procedurii create anterior dar dând variabilelor alte valori, va trebui întâi să ștergem aceste variabile cu comanda UNDEFINE.

c) *Substituirea variabilelor de tip &n cu ajutorul comenzii START.*  
 Pentru a nu mai furniza interactiv valori pentru variabilele utilizator la momentul execuției, le putem defini pe acestea sub forma **&n**, în care *n* ia valori începând cu **1**, iar valorile lor vor fi furnizate ca parametrii de apel ai instrucțiunii **START**.

Astfel vom crea fișierul de comenzi **CALCSTART.SQL** de forma:

```
/* - -----
---
/*  &1 = nume coloană din tabeli după care se face gruparea
/*  &2 = tipul calculului: min, max, sum pentru un anumit grup
/*  &3 = numele coloanei de tip numeric după care se va face
calculul
/*  &4 = numele coloanei pe care se vor afișa valorile calculate
/*  &5 = tipul ordonării(sortării), descending sau ascending*/
select &1, &2(&3) &4
from pers100
group by &1
order by &2(&3) &5;
/* - -----*/
```

Pentru execuție vom apela fișierul de comenzi **CALCSTART.SQL** cu comanda **START** în care vom furniza ca parametrii valorile dorite pentru cele 5 variabile de tip **&n** definite.

**Exemplu:**

**SQL> START CALCSTART nume min salariu MINIMUM desc**

Această execuție va produce același rezultat ca și execuțiile fișierelor de comenzi **CALC&.SQL** și **CALC&&.SQL**.

Deosebirea este că nu se mai creează variabilele utilizator astfel că putem executa în mod generalizat procedura **CALCSTART** dând variabilelor orice alte valori logic acceptabile.

d) *Crearea unor modalități interactive de comunicare cu calculatorul* se realizează cu comenzile **PROMPT**, **ACCEPT** și **PAUSE**, care sunt de fapt operații de intrare/ieșire standard. **PROMPT** permite trimiterea (scrierea) de mesaje la utilizator. **ACCEPT** permite preluarea (citirea) răspunsurilor de la utilizator. **PAUSE** permite introducerea unui moment de pauză pentru ca utilizatorul să citească mesajul și să-și formuleze răspunsul.

Aceste comenzi se folosesc în conjuncție cu **INPUT** și **SAVE**, pentru a introduce valorile dorite cu comenzile de mai sus, și respectiv pentru a le salva într-un fișier de comenzi, care să fie ulterior executat.

**Exemplu:**

**SQL> Clear buffer**

**SQL> INPUT**

**PROMPT** Introduceți un titlu format din maxim 30 caractere  
**PAUSE** urmează introducerea titlului, apăsați **RETURN**  
**ACCEPT** **TITLUL\_MEU** **PROMPT** 'TITLUL: '  
**PAUSE** urmează comanda de centrare a titlului, apăsați **RETURN**  
**TTITLE** **CENTER** **TITLUL\_MEU** **SKIP** 2  
**PAUSE** urmează selectarea din baza de date, apăsați **RETURN**  
**SELECT** **codpers, nume , prenume** **from** **pers100;**  
**SQL> SAVE cmdprompt**  
**SQL> @cmdprompt**

*Rezultatul este:*

Introduceți un titlu format din maxim 30 caractere  
urmează introducerea titlului, apăsați **RETURN**  
titlu:-----**SELECTARE DIN BAZA DE DATE** **codpers, nume și prenume** ----  
urmează comanda de centrare a titlului, apăsați **RETURN**  
urmează selectarea din baza de date, apăsați **RETURN**  
-----**SELECTARE DIN BAZA DE DATE** **codpers, nume și prenume** ----

<b>CODPERS</b>	<b>NUME</b>	<b>PRENUME</b>
1	petrescu	ion
2	petrescu	florea
3	ionescu	ion
4	ionescu	dumitru
5	mihai	florea
6	mihai	ion

6 rows selected.

*e) Utilizarea comenzilor **PROMPT** și **ACCEPT** în conjuncție cu substituirea variabilelor*

În exemplele anterioare când s-au utilizat fișierele de comenzi **CALC&.SQL** și **CALC&&.SQL** s-a văzut că sistemul pentru fiecare variabilă a creat câte un prompter de forma:

**Enter value for nume\_variabilă :**

Ca atare se poate înlocui prompter-ul sistemului cu propriu prompter utilizând pentru aceasta comenzile **PROMPT** și **ACCEPT** în fișierul de comenzi unde vrem să introducem o variabilă pentru care vom construi alt prompter decât cel de sistem.

**Exemplu:**

**SQL> Clear buffer**

**SQL> INPUT**

**PROMPT** Introduceți o valoare numerică pentru salariu

**PROMPT** De exemplu: 1500000, 2500000

**ACCEPT** **var\_salariu** **NUMBER** **PROMPT** 'valoare salariu: '



```

SELECT codpers, nume, salariu from pers100
WHERE salariu = &var_salariu
SQL> save cmd2prompt
SQL> @cmd2prompt

```

Rezultatul este:

Introduceți o valoare numerică pentru salariu

De exemplu: 1500000, 2500000

valoare salariu: aaaa

“aaaa” is not a valid number

valoare salariu: 3500000

old 1: **SELECT** codpers, nume, salariu from pers100 **WHERE** salariu =  
&var\_salariu

new 1: **SELECT** codpers, nume, salariu from pers100 **WHERE** salariu =  
3500000

CODPERS	NUME	SALARIU
4	ionescu	3500000

### 3.8. FORMATAREA REZULTATELOR

Limbajul SQL\*PLUS permite proiectarea și formatarea diverselor situații de ieșire. Aceste operații sunt posibile prin utilizarea unor comenzi pentru tratarea întreruperilor, comenzi pentru definirea titlurilor, definirea coloanelor, realizarea calculelor și stabilirea diverselor opțiuni pentru poziționarea unor arii de lucru pe ecran.

#### A. Tratarea întreruperilor

Întreruperea este un eveniment care se produce în timpul execuției unei comenzi **SELECT**, cum ar fi, de exemplu, apariția sfârșitului de pagină sau schimbarea valorii unei expresii.

Pentru a specifica evenimentele care determină o întrerupere și acțiunea corespunzătoare SQL care se execută, se utilizează comanda **BREAK**. Ea poate specifica mai multe evenimente care generează o întrerupere. Evenimentele sunt reținute într-o ordine numită "ierarhie de întrerupere". La un moment dat, se poate executa doar o singură comandă **BREAK**.

Comanda **BREAK** are următoarele forme sintactice:

```

BRE[AK] ON {expr | ROW | PAG[E] | REPORT}
[SKI[P] n | [SKIP]PAGE]
[NODUP[LICATES] | DUP[LICATES]];
BRE[AK];

```

unde:

**ON** expr determină o întrerupere când se schimbă valoarea expresiei expr; expr este fie o expresie care implică una sau mai multe coloane dintr-o tabelă, fie o etichetă atașată unei coloane declarată în comanda **SELECT** sau **COLUMN**.

Dacă **ON** expr apare de mai multe ori în comandă, atunci expresiile respectă "ierarhia de întrerupere", aceasta fiind chiar ordinea în care sunt specificate expresiile. Când se folosește **ON** expr, trebuie utilizată și clauza **ORDER BY**, pentru a ordona rândurile din "ierarhia de întrerupere". Ordinea de apariție a expresiilor expr în comanda **BREAK ON** trebuie să fie aceeași cu cea prezentă în **SELECT...ORDER BY**:

**ON ROW** determină o întrerupere când se selectează un rând cu **SELECT**. Această întrerupere este reținută la sfârșitul ierarhiei de întrerupere;

**ON PAGE** determină o întrerupere la sfârșitul fiecărei pagini;

**ON REPORT** determină o întrerupere la sfârșitul raportului sau cererii, întrerupere care este reținută la începutul ierarhiei de întrerupere.

**SKIP** determină saltul peste n linii, înainte de a se tipări linia asociată întreruperii respective.

**PAGE** sau **SKIP PAGE** determină saltul la o pagină nouă înainte de a tipări linia asociată respectivei întreruperi.

**NODUPPLICATES** determină tipărirea de spații când valorile de JJS coloana de întrerupere sunt identice. **DUPLICATES** determină tipărirea valorii coloanei de întrerupere în fiecare rând selectat. Valoarea **NODUPPLICATES** este implicită.

Comanda **BREAK** fără clauze indică poziția întreruperii curente.

### Exemple:

1) Să fie definită o întrerupere generată de schimbarea valorilor coloanei FUNCT. La apariția acestui eveniment să fie afișate două linii vide.

**SQL> BREAK ON FUNCT SKIP 2;**

**SQL> SELECT MARCA,NUME,CODD,**

**2 SALA, VENS**

**3 FROM SALARIAȚI**

**4 ORDER BY FUNCT;**

MARCANUME	FUNCT	CODD	SALA	VENS
7000 ION ION	DIRECTOR	100000	45000	40000
2550 FRINCU ION	SEF DEP	160000	36000	37000
1000 COMAN RADU		130000	35000	2500
2500 VLAD VASILE		160000	36500	1500
4000 PAUL ȘTEFAN		160000	35000	5600
3755 DORU DAN		130000	36500	5500

1111	AVRAM ION VINZATOR	100000	21200	1000
1680	RADU ION	130000	20750	3000
3700	MÂNU DAN	160000	27500	2500
2553	AILENEI FLORIN	120000	25000	400
3760	SANDU ION	130000	25600	0
3770	CARMEN ANA	130000	26500	
2554	DARIAN GEO	120000	26000	2000
3759	ALEXE IOAN	160000	25700	
3500	DAN ION	160000	24500	350
2650	VLAD ION	120000	25060	3500
1222	BARBU DAN	120000	20750	2000

2) Să fie definită o întrerupere la apariția unei schimbări a valorii coloanei ODS din tabela SALARIAȚI. În momentul realizării evenimentului să se realizeze salt la pagină nouă.

***SQL> SET PAGESIZE 11***

***SQL> BREAK ON CODS SKIP PAGE;***

***SQL> SELECT \* FROM SALARIAT  
2 ORDER BY CODS;***

MARCA	NUME	FUNCT	CODD	SALA	VENS	CODS
1111	AVRAM ION	VINZATOR	100000	21200	1000	1000
2650	VLAP ION	VINZATOR	120000	25060	3500	
1222	BARBU DAN	VINZATOR	120000	20750	2000	

MARCA	NUME	FUNCT	CODD	SALA	VENS	CODS
2550	FRINCU ION	SEF DEP	160000	36000	37000	2500
3500	DAN ION	VINZATOR	160000	24500	350	
1680	RADU ION	VINZATOR	130000	20750	3000	

MARCA	NUME	FUNCT	CODD	SALA	VENS	CODS	VENS	CODS
2553	AILENEI FLOR	VINZATOR	120000	120000	250000	2000		
2554	DARIAN GEO	VINZATOR	260000	4000				

MARCA	NUME	FUNCT	CODD	SALA	VENS	CODS
7000	ION ION	DIRECTOR	100000	45000	40000	8000

*9records selected.*

## **B. Tipărirea titlurilor**

Pentru afișarea unui titlu la sfârșitul sau la începutul fiecărei pagini se utilizează comanda BTITLE, respectiv TTITLE. Comanda BTITLE are următoarele sintaxe:

**BTI[TLE] [COL[UMN] n1 [SKIP [n]] [TAB n] [LEFT | RIGHT | CENTER] [FORMAT char] [char | var]...;**

**BTI[TLE] {OFF | ON};**  
**BTI[TLE] text;**  
**BTI[TLE]**

unde:

COL[UMN] n determină saltul la coloana n a liniei curente.

Prin SKIP n se determină saltul la începutul liniei curente, de n ori. Dacă n este omis, se sare o singură dată, iar dacă n este zero, se realizează întorcerea la începutul liniei curente.

TAB n are ca efect saltul a n coloane (prin coloană înțelegându-se nu o coloană de tabelă, ci poziția cursorului) înainte dacă n este pozitiv sau înapoi dacă n este negativ.

Clauzele LEFT, RIGHT, CENTER determină alinierea la stânga, la dreapta respectiv afișarea centrata a datelor din linia curentă. Următoarele date sunt aliniate ca un grup, de la începutul până la sfârșitul comenzii PRINT sau la următorul LEFT, CENTER, RIGHT sau COLUMN. CENTER și RIGHT folosesc valoarea returnată de comanda SET LINESIZE pentru a calcula poziția următorului articol.

FORMAT char specifică un model de format pentru articolul de date care urmează; acest format se menține până la întâlnirea unei alte clauze FORMAT sau până la sfârșitul comenzii. De reținut că doar un singur model de format poate fi activ la un moment dat. Dacă nici un format potrivit nu are efect, atunci valorile sunt tipărite în conformitate cu formatul specificat în SET NUMFORMAT iar dacă SET UNFORMAT nu a fost utilizat, valorile vor fi tipărite cu formatul implicit.

Pentru specificarea titlurilor pot fi folosite constante (char) și variabile (var), ele fiind poziționate și formate așa cum se specifică în clauzele comenzii.

Existența unui separator indică începutul unor linii noi, iar doi separatori pe același rând introduc o linie vidă. Inițial, caracterul de separare este "!", însă el poate fi schimbat cu SET HEADSEP.

SQL\*PLUS interpretează comanda BTITLE în forma nouă dacă primul cuvânt după numele comenzii este numele unei clauze valide (LEFT, SKIP, COL etc.).

Clauzele ON și OFF au ca efect apariția (ON) sau nu (OFF) a titlului.

BTITLE text afișează centrat textul specificat.

Comanda BTITLE fără clauze specifică titlul curent.

**Exemple:**

1) Să se afișeze la sfârșitul unui raport final cu privire la produsele din depozitul cu codul 100000, în partea stângă șirul de caractere "Data:" iar în partea dreaptă "Semnătura:".

**SQL> SET PAGESIZE 11**

**SQL> BTITLE LEFT Data: RIGHT Semnătura;;**

**SQL> SELECT • FROM PRODUSE**

**2 WHERE CODD=100000;**

CODD	CODP	DENP	STOC	DATA CRT	UM
100000	D4	SCAUN	36	10-SEP-05	BUC
100000	A3	FOTOLIU	27	15-SEP-05	BUC
100000	A7	MASA	23	05-SEP-05	BUC

Data: Semnătura:

2) Să se afișeze la sfârșitul unui raport privind situația produselor din depozitul 100000, începând din coloana 11, șirul de caractere "OBSERVATIT.

**SQL> BTITLE COL 11 OBSERVAȚII;**

**SQL> SELECT \* FROM PRODUSE**

**2 WHERE CODD=100000;**

CODD	CODP	DENP	STOC	DATA CRT	UM
100000	166666	PLACAJ 2/2	100	12-JUL-05	MP
100000	144444	SCAUN D4	36	12-JUL-05	BUC
100000	111111	MESE 15/20	7	27-JUN-05	BUC
100000	122222	FOTOLIU A3	12	01-JUL-05	BUC
100000	133333	CANAPEA A7	6	18-JUL-05	BUC
100000	155555	BIROU C6X4	9	29-JUL-05	BUC

OBSERVAȚII

3) Să se afișeze centrat, la sfârșitul unui raport privind produsele din depozitul cu codul 100000, șirul de caractere "Depozitul\_MOBILA/100000".

**SQL> BTITLE CENTER Depozitul\_MOBILA/100000**

**SQL> SELECT CODD "Cod depozit",**

**2 DENP "Denumire",**

**3 CODP "Cod produs",**

**4 STOC "Stoc", DATA CRT "Data",**

**5 UM**

**6 FROM PRODUSE**

**7 WHERE CODD=100000;**

Cod dep	Denumire	Cod produs	Stoc	Data	UM
---------	----------	------------	------	------	----

100000	PLACAJ 2/2	166666	100	12-JUL-05	MP
100000	SCAUN D4	144444	36	12-JUL-05	BUC
100000	MESE 15/20	111111	7	27-JUN-05	BUC
100000	FOTOLIU A3	122222	12	01-JUL-05	BUC
100000	CANAPEA A7	133333	6	18-JUL-05	BUC
100000	BIROU C6X4	155555	9	29-JUL-05	BUC

Depozitul MOBILA/100000

4) Să se afișeze specificațiile curente pentru BTITLE. Ultima comandă primită în sistem se prezintă astfel:

***SQL> BTITLE COL 40 Total RIGHT Semnătura;***

***SQL> BTITLE;***

bttitle ON and is the following 28 characters: COL 40 Total RIGHT Semnătura

Comanda TTITLE determină afișarea unui titlu la începutul paginii și are următoarele sintaxe:

***TTI[TLE] [COL[UMN] n] [SKIP [1 | n] [TAB n]***

***[LEFT | RIGHT | CENTER]***

***[FORMAT char] [char | var]***

***TTI[TLEI text;***

***TTI[TLE];***

Clauzele comenzii TTITLE au aceeași semnificație ca la BTITLE.

Comanda TTITLE fără clauze determină afișarea titlului curent.

***Exemplu:***

Să se afișeze la începutul unui raport privind comenzile cu date mai mică de 30 septembrie 2005, următoarele șiruri de caractere: "SITUAȚIA COMENZILOR" și "LA DATA DE 30-SEP-05". Cele două șiruri se vor afișa centrat, pe două linii.

***SQL> SET PAGESIZE 11 SQL> SET LIN 45***

***SQL> COLUMN NRCOM FORMAT 99999999***

***SQL> COLUMN NRCOM JUSTIFY CENTER***

***SQL> COLUMN CODP FORMAT 99999999***

***SQL> COLUMN CODP JUSTIFY CENTER***

***SQL> COLUMN VALOARE FORMAT 9999999999***

***SQL> COLUMN VALOARE JUSTIFY CENTER***

***SQL> SET SPACE 7***

***SQL> TTITLE 'SITUAȚIA COMENZILOR LA  
DATA DE 30-SEP-05'***

***SQL> SELECT***

***2 NRCOM,CODP,CANT\*PRET VALOARE***

**3 FROM COMENZI**  
**4 WHERE DATAL<='30- SEP-05;**

SITUATIA COMENZILOR LA DATA DE 30- SEP-05

NRCOM	CODP	VALOARE
211111	233333	244444
255566	133333	144444
166666	222222	320000
27000	375000	282000

**Anularea opțiunilor**

În vederea anulării opțiunilor se utilizează comanda:

**CL[EAR] option**

Este anulată opțiunea specificată prin option, după cum urmează:

**BRE[AKS]** anulează întreruperea indicată prin comanda BREAK;

**BUFF[ER]** determină ștergerea textului din buffer-ul curent;

**COL[UMNS]** anulează opțiunile indicate în comanda COLUMN;

**COMP[UTES]** anulează opțiunile indicate prin comanda COMPUTE;

**SCR[REEN]** determină ștergerea ecranului, iar SQL determină ștergerea buffer-ului SQL;

**TIMI[NG]** șterge toate zonele create cu comanda TIMING.

**Exemple:**

1) Să se anuleze întreruperile.

**SQL> CLEAR BREAKS;**

2) Să se anuleze definițiile de coloană.

**SQL> CLEAR COLUMNS;**

**Descrierea coloanelor**

Pentru specificarea modului de formare a unei coloane și a capului de coloană într-o situație, se utilizează comanda COLUMN. Sintaxa ei este:

**COL[UMN] (col | expr) [ALI[AS] sinonim] CLE[AR] |**  
**DEF[AULT]] [COLOR {culoare| variabila-culoare}]**  
**[FORMAT] format] [HEA[DING] text**  
**[JUS[TIFY] {L[EFT | C[ENTER] | R[IGHT]] [LIKE {expr |**  
**etichetă}] [LINEAPP {LINE | MARK | BOTH}]**  
**[NEW\_VALUE variabila] [NU[LL] char] [NOPRI[NT] |**  
**PRINT]] [OLD\_VALUE variabila]**  
**[ON | OFF]**  
**[PATTERN {număr-de-model | variabila-model}**  
**[WRA[PPED]! WOR[D,... WRAPPED] |**  
**TRU[NCATED]]**  
**...;**

unde:

col sau expr au rolul de a identifica coloana sau expresia la care se referă comanda. Comanda COLUMN trebuie să se refere la o coloană sau la o expresie utilizat-a în comanda SELECT. Dacă comanda COLUMN se referă la o expresie, aceasta trebuie să fie specificată în același mod în care a fost specificată în SELECT. De exemplu, dacă în SELECT expresia este 'A.+B', nu se poate folosi, în COLUMN, T+A'. Dacă sunt selectate din tabele diferite coloane cu același nume, atunci comanda COLUMN se va aplica tuturor tabelor ce conțin acea coloană.

Clauza ALIAS pune în corespondență numele coloanelor cu sinonimul specificat.

Clauza CLEAR are ca efect ștergerea în totalitate a definiției coloanei.

Clauza FORMAT specifică formatul de apariție al coloanei. Lățimea implicită a coloanei este chiar lățimea coloanei definite în baza de date. Ea poate fi schimbată cu valoarea n, folosind "FORMAT An". Lățimea unei coloane numerice este implicit valoarea data de NUMWIDTH, ea putând fi schimbată cu ajutorul clauzei FORMAT.

Clauza HEADING definește capul coloanei, care implicit este col sau expr. Dacă textul din definirea coloanei conține spații sau semne de punctuație, acestea trebuie puse între ghilimele. Fiecare semn "I" din text are ca efect începerea unei linii noi.

Clauza JUSTIFY aliniază capul coloanei. Implicit, alinierea se face la dreapta pentru o coloană numerică și la stînga pentru alte tipuri de coloane.

Clauza LIKE determină copierea specificațiilor altei coloane sau expresii (specificații deja definite prin altă comandă COLUMN).

Prin folosirea clauzei NEWLINE se trece la o linie nouă, înainte de a afișa valorile coloanei.

Clauzele PRINT și NOPRINT au ca efect tipărirea sau nu a coloanei.

Dacă se utilizează clauza NULL, textul va fi afișat chiar dacă conține o valoare nulă. Valoarea implicită este un șir de blankuri.

Clauzele OFF sau ON determină folosirea, respectiv nefolosirea formatului specificației de ieșire pentru o coloană, fără a-i afecta conținutul.

Clauza WRAPPED are ca efect scrierea pe linia următoare a caracterelor care nu au încăput pe prima linie.

Clauza WORD\_WRAPPED are efect asemănător cu cel al clauzei WRAPPED, cu deosebirea că determină scrierea întregului cuvânt pe linia următoare și nu pe două rânduri ca la WRAPPED.

Clauza TRUNC determină trunchierea valorii.

Clauza COLOR specifică culoarea utilizată pentru afișarea valorilor coloanei, într-o reprezentare liniară sau prin benzi. Variabila CLR n (n=1,60)



se referă la valoarea curentă a culorii. Setarea se face prin comanda SET CLR n.

Clauza LINEAPP stabilește modul de subliniere a coloanei. Utilizând LINE dreapta apare continuă. Folosind MARK se schițează doar puncte, iar BOTH realizează ambele modalități: linie și puncte.

Clauza PATTERN specifică modelul pe care-l va avea dreptunghiul/banda într-un grafic de tip benzi sau sectorul într-un grafic de tip cerc. număr-de-model este cuprins între 1 și 16 și reprezintă un model. Variabila PAT n, unde ns[1,30] se referă la poziția curentă a modelului. Semnificația valorilor din număr-de-model și a valorilor PAT n sunt precizate în documentație.

Numărul de comenzi COLUMN folosite indică faptul că se lucrează cu diverse coloane sau expresii utilizate în comanda SELECT. Pentru o coloană sau expresie pot exista mai multe comenzi COLUMN. În cazul în care mai multe comenzi COLUMN folosesc aceeași clauză pentru aceeași coloană, se va lua în considerare doar ultima comandă.

Comanda COLUMN fără clauze are ca efect tipărirea definițiilor coloanei curente. Dacă această comandă are doar clauzele col și expr, efectul ei constă în indicarea definirii coloanei existente.

Trebuie precizat că în prima comandă COLUMN, expresia trebuie să aibă aceeași formă ca și în SELECT. În caz contrar, SQL\*Plus nu va putea executa comanda COLUMN pentru coloana respectivă.

### **Example:**

1) Să se definească coloana NUME pe o lățime de 25 de caractere iar capul de coloană să fie scris pe două rânduri, astfel: NUME PRODUS

```
SQL> COLUMN NUME FORMAT A25 HEADING 'NUME  
PRODUS';
```

```
SQL> SELECT DISTINCT(DENP) NUME  
2 FROM PRODUSE;
```

NUME PRODUS

MESE 15/20

FOTOLIU

2 records selected.

2) Să se afișeze sinonimul SALARIU pentru coloana definită de expresie aritmetică SALA+VENS. Coloana va fi scrisă cu formatul \$99,999.99.

```
SQL> COLUMN SALA+VENS ALIAS SALARIU
```

```
SQL> COLUMN SALARIU FORMAT $99,999.99
```

```
SQL> SELECT SALA+VENS SALARIU,  
2 MARCA, NUME  
3 FROM SALARIAȚI;
```

SALARIU	MARCA	NUME
\$22,200.00	1111	AVRAM ION
\$22,750.00	1222	BARBU DAN

2 records selected.

3) Să se definească coloana DENP de 16 caractere alfanumerice, pentru care se specifică sinonimul DENUMIRE . In capul de coloană se va afișa textul "Denumire produs".

```
SQL> COLUMN DENP FORMAT A16  
SQL> COLUMN DENP ALIAS DENUMIRE  
SQL> COLUMN DENUMIRE HEADING "Denumire produs"  
SQL> SELECT * FROM PRODUSE;
```

CODD	CODP	Denumire produs	STOC	DATA CRT	UM
100000	166666	PLACAJ 2/2	100	12-JUL-92	MP
100000	122222	FOTOLIU A3	7	27-JUN-92	BUC
100000	133333	CANAPEA A7	36	12-JUL-92	BUC
100000	155555	BIROU C6X4	12	01-JUL-92	BUC
100000	144444	SCAUN D4	6	18-JUL-92	BUC
100000	111111	MESE 15/20	9	29-JUL-92	BUC

6 records selected.

4) Să se definească coloana DENUM2 cu un format alfanumeric de trei caractere. Specificațiile pentru această coloană sunt copiate pentru coloana DENP.

```
SQL> COLUMN DENUM2 FORMAT A3  
SQL> COLUMN DENP LME DENUM2  
SQL> SELECT * FROM PRODUSE 2 WHERE CODP<138333;
```

CODD	CODP	DEN	STOC	DATA CRT	UM
100000	111111	MES	7	27-JUN-05	BUC

5) Să se definească coloana ADRESA pe 21 de caractere, utilizând clauza WRAPPED.

```
SQL> COLUMN FORMAT A21 ADRESA WRAPPED  
SQL> SELECT STR11 '-' 11NR11 '-' 11 LOC ADRESA 2  
FROM CLIENTI;
```

ADRESA  
MOȘILOR- 104-BUCUREȘTI  
DOROBANȚI- 18-BUCUREȘTI  
GOLENTINA-221-BUCUREȘTI  
EMINESCU-44-BUCUREȘTI

4 records selected.

## D.Realizarea de calcule

SQL\*Plus permite realizarea unor calcule cu rândurile selectate din tabele. Pentru aceasta se utilizează comanda COMPUTE, care are următoarea sintaxă:

**COMP[UTE] [AVG | COU[NT] | MAX[IMUM] |  
MIN[IMUM] NUMBER) ISDT | SUM | VAR[iance}]  
OF { expresiei etichetă},...  
ON [expresiei etichetă IPAGE1 REPORT! ROW];**

Semnificația clauzelor este:

Clauza	Semnificația	Tipul coloanei Tipuri de date
AVG	Valoare medie	Numeric
COUNT	Contorizare valori nule	Toate tipurile
MAXIMUM	Valoare maximă	Numeric și caracter
MINIMUM	Valoare minimă	Numeric și caracter
NUMBER	Contorizare rânduri	Toate tipurile
STD	Abatere standard	Numeric
SUM	Suma valorilor nenule	Numeric
VARIANCE	Dispersia	Numeric

Dacă se specifică mai multe funcții, nu trebuie separate între ele prin virgulă.

**OF** precizează coloana sau expresia ale căror valori se vor supune calculului. Acestea trebuie să apară în comanda SELECT, altfel comanda COMPUTE le va ignora. Dacă se dorește ca valoarea calculată să nu fie afișată pe ecran se va utiliza clauza **NON PRINT** atașată coloanei respective cu comanda **SELECT**.

**ON** specifică un element care va fi utilizat ca generator de întrerupere. El poate fi: expresie, etichetă, pagină (PAGE), raport (REPORT), linie (ROW). De câte ori se va schimba valoarea acestuia, se va declanșa automat recalcularea funcției definite de **COMPUTE**.

În cazul în care se dă comanda fără nici o funcție, se vor afișa specificațiile de calcul definite.

Secvența de instrucțiuni care se va utiliza, în general, va fi:

**SQL\* BREAK ON expresiei  
SQL> COMPUTE clauza OF expresie2 ON expresiei;  
SQL> SELECT**

**Exemple:**

1) Să se editeze situația finală cu următorul format:

SITUAȚIE SUM(SALA) FUNCȚIE

**SQL> SET PAGESIZE 22**

**SQL> RUN**

**1 SELECT SUM(SALA),FUNCT FROM SALARIAȚI**

## **2\* GROUP BY FUNCT**

SITUAȚIE	
SUM(SALA)	FUNCT
45000	DIRECTOR
179000	SEF DEP
268560	VINZATOR

2) Să se editeze o situație finală cu salariile grupate pe funcții și depozite.

```
SQL> RUN  
1 SELECT FUNCT,  
2 SUM(DECODE(CODD,100000,SALA,0)) "DEP 10",  
3 SUM(DECODE(CODD,130000,SALA,0)) "DEP 13",  
4 SUM(DECODE(CODD,160000,SALA,0)) "DEP 16"  
5 FROM SALARIAȚI  
6 GROUP BY FUNCT;
```

### SITUAȚIE

FUNCT	DEP	DEP	DEP
DIRECTOR	45000	0	0
SEF DEP	0	71500	107500
VINZATOR	21200	72850	77700

Data:

Semnătura:

3) Să se afișeze o situație finală cu salariile grupate pe funcții și depozite.  
De asemenea, să se introducă o coloană cu suma salariilor pe fiecare funcție.

```
SQL>RUN  
1 SELECT FUNCT,  
2 SUM(DECODE(CODD,100000,SALA,0)) "DEP 10",  
3 SUM(DECODE(CODD,130000,SALA,0)) "DEP 13",  
4 SUM(DECODE(CODD,160000,SALA,0)) "DEP 16",  
5 SUM(SALA)  
6 FROM SALARIAȚI  
7 GROUP BY FUNCT;
```

### SITUAȚIE

FUNCT	DEP 10	DEP 13	DEP 16	SUM(SALA)
DIRECTOR	45000	0	0	45000
SEF DEP	0	71500	107500	179000
VINZATOR	21200	72850	77700	268500

Data:

Semnătura:

4) Să se afișeze o situație finală cu salariile grupate pe funcții și depozite. De asemenea, să se facă totalul salariilor pe fiecare depozit și fiecare funcție.

```
SQL> BREAK ON DUMMY;
SQL> COMPUTE SUM OF "DEP 100000" ON DUMMY;
SQL> COMPUTE SUM OF "DEP 130000" ON DUMMY;
SQL> COMPUTE SUM OF "DEP 160000" ON DUMMY;
SQL> COMPUTE SUM OF "TOTAL" ON DUMMY;
SQL> COLUMN DUMMY NOPRINT;
SQL> TTITLE CENTER "SITUAȚIA SALARIILOR"
SKIP CENTER "PE DEPOZITE / FUNCȚII"
SKIP CENTER " "
SQL> RUN
1 SELECT FUNCT,
2 SUM(DECODE(CODB,IG0000,SALA,0)) "DEP 10",
3 SUM(DECODE(CODD,130000,SALA,0)) "DEP 13",
4 SUM(DECODE(CODD,160000,SALA,0)) "DEP 16",
5 SUM(SALA,)
6 SUM(0) DUMMY
7 FROM SALARIAȚI
8 GROUP BY FUNCT
```

SITUAȚIA SALARIILOR PE DEPOZITE / FUNCȚII				
FUNCT	DEP 10	DEP 13	DEP 16	SUM(SALA)
DIRECTOR	45000	0	0	45000
SEF DEP	0	71500	107500	179000
VINZATOR	21200	72850	77700	268500
	66200	144350	185200	

Data:

Semnătura: