

## Baze de date-Anul 1

### Laborator 1

---

#### 1. Introducere

1. Ce este o bază de date ? Dar un sistem de gestiune a bazelor de date? Dați exemple.

- **Baza de date** este un ansamblu structurat de date coerente, fără redundanță inutilă, care pot fi accesate în mod concurrent de către mai mulți utilizatori.
- Un **sistem de gestiune a bazelor de date** (SGBD) este un produs software care asigură interacțiunea cu o bază de date, permițând definirea, consultarea și actualizarea datelor din baza de date.

2. Ce este SQL?

- **SQL** (Structured Query Language) este un **limbaj** neprocedural pentru interogarea și prelucrarea informațiilor din baza de date.
  - Compilatorul limbajului SQL generează automat o procedură care accesează baza de date și execută comanda dorită.
  - SQL permite:
    - definirea datelor (LDD)
    - prelucrarea și interogarea datelor (LMD)
    - controlul accesului la date (LCD).
  - Comenzile SQL pot fi integrate în programe scrise în alte limbaje, de exemplu Cobol, C, C++, Java etc.

3. Ce este SQL\*Plus? Comenzile SQL\*Plus accesează baza de date ?

- **SQL\*Plus** este un **utilitar Oracle**, având comenzi proprii specifice, care recunoaște instrucțiunile SQL și le trimite server-ului Oracle pentru execuție.
  - Dintre funcționalitățile mediului SQL\*Plus, se pot enumera:
    - editarea, executarea, salvarea și regăsirea instrucțiunilor SQL și a blocurilor PL/SQL;
    - calculul, stocarea și afișarea rezultatelor furnizate de cereri;
    - listarea structurii tabelelor.
  - Tabelul următor evidențiază diferențele dintre instrucțiunile SQL și cele SQL\*Plus:

SQL	SQL*Plus
Este un <b>limbaj</b> de comunicare cu server-ul Oracle pentru accesarea datelor.	Recunoaște instrucțiunile SQL și le transferă server-ului Oracle.
Se bazează pe <b>standardul ANSI</b> pentru SQL.	Este o interfață specifică sistemului Oracle pentru execuția instrucțiunilor SQL.
Prelucrează date și definește obiecte din baza de date.	Nu permite prelucrarea informațiilor din baza de date.
Utilizează funcții pentru a efectua formatare.	Utilizează comenzi pentru a efectua formatare.
Instrucțiunile nu pot fi abreviate.	Comenzile pot fi abreviate.
Nu are un caracter de continuare a instrucțiunilor scrise pe mai multe linii.	Acceptă „-“ drept caracter de continuare pentru comenzile scrise pe mai multe linii.
Caracterul de terminare a unei comenzi este „.”	Nu necesită caracter de terminare a unei comenzi.

4. Comenzile SQL\*Plus acceptă abrevieri? Este necesar vreun caracter de încheiere a comenzii? (vezi tabelul de mai sus)
5. Care sunt limbajele SQL?
- În funcție de tipul acțiunii pe care o realizează, instrucțiunile SQL se împart în mai multe categorii. Datorită importanței pe care o au comenzile componente, unele dintre aceste categorii sunt evidențiate ca limbaje în cadrul SQL, și anume:
    - limbajul de definire a datelor (LDD) – comenzile *CREATE*, *ALTER*, *DROP*;
    - limbajul de prelucrare a datelor (LMD) – comenzile *INSERT*, *UPDATE*, *DELETE*, *SELECT*;
    - limbajul de control al datelor (LCD) – comenzile *COMMIT*, *ROLLBACK*, *SAVEPOINT*.
  - Pe lângă instrucțiunile care alcătuiesc aceste limbaje, SQL cuprinde și alte tipuri de instrucțiuni:
    - instrucțiuni pentru controlul sesiunii;
    - instrucțiuni pentru controlul sistemului;
    - instrucțiuni SQL încapsulate.

6. Analizați sintaxa simplificată a comenzii SELECT:

```
SELECT { [ { DISTINCT | UNIQUE } | ALL ] lista_campuri | *}
FROM [nume_schemă.]nume_obiect ]
      [, [nume_schemă.]nume_obiect ...]
[ WHERE condiție_clauza_where ]
[ START WITH condiție_clauza_start_with
CONNECT BY condiție_clauza_connect_by ]
[ GROUP BY expresie [, expresie ...]
  [ HAVING condiție_clauza_having ] ]
[ ORDER BY {expresie | poziție} [, {expresie | poziție} ...] ]
[ FOR UPDATE
  [ OF [ [nume_schemă.]nume_obiect.]nume_coloană
    [, [ [nume_schemă.]nume_obiect.]nume_coloană ...]
  [ NOWAIT | WAIT număr_întreg ] ];
```

Un element din *lista\_campuri* are forma: *expresie [AS] alias*.

Care dintre clauze sunt obligatorii?

7. Care sunt regulile de scriere a comenzilor SQL (acceptă abrevieri, e nevoie de caracter de terminare)? În instrucțiunea următoare sunt 3 erori. Care sunt acestea?

```
SQL> SELECT employee_id, last_name
      2          salary x 12 ANNUAL SALARY
      3          FROM employees;
```

**Obs:** *ANNUAL SALARY* este un alias pentru câmpul reprezentând salariul anual.

- Dacă un alias conține *blank*-uri, el va fi scris obligatoriu între ghilimele. Altfel, ghilimelele pot fi omise.
- *Alias*-ul apare în rezultat, ca și cap de coloană pentru expresia respectivă. Doar cele specificate între ghilimele sunt *case-sensitive*, celelalte fiind scrise implicit cu majuscule.

## 2. Exerciții

1. a) Consultați diagrama exemplu *HR* (Human Resources) pentru lucrul în cadrul laboratoarelor de baze de date.

b) Identificați cheile primare și cele externe ale tabelelor existente în schemă, precum și tipul relațiilor dintre aceste tabele.

2. Să se inițieze o sesiune *SQL\*Plus / SQL Developer* folosind informațiile de conectare indicate.
3. Să se listeze **structura** tabelelor din schema *HR* (*EMPLOYEES*, *DEPARTMENTS*, *JOBS*, *JOB\_HISTORY*, *LOCATIONS*, *COUNTRIES*, *REGIONS*), observând tipurile de date ale coloanelor.

**Obs:** Se va utiliza comanda *DESC[RIBE] nume\_tabel*.

4. Să se listeze **conținutul** tabelelor din schema considerată, afișând valorile tuturor câmpurilor.

**Obs:** *SELECT \* FROM nume\_tabel;*

5. Să se obțină încă o dată rezultatul cererii precedente, fără a rescrie cererea.

**Obs:** Ultima cerere *SQL* lansată de către client este păstrată în buffer-ul *SQL*. Pentru rularea *buffer*-ului, se executa comanda:

*SQL> /*

sau

*SQL> RUN*

6. Listați structura tabelului *EMPLOYEES* (*DESC employees*) și apoi dați comanda *RUN* (sau */*). Ce observați? Comenzile *SQL\*Plus* sunt păstrate în *buffer*?
7. Să se afișeze codul angajatului, numele, codul job-ului, data angajării. Ce fel de operație este aceasta (selecție sau proiecție)? Salvați instrucțiunea *SQL* într-un fișier *p711.sql*.

**Obs:** Se utilizează comanda *SAVE* pentru salvarea *buffer*-ului într-un fișier.

*SQL> SELECT employee\_id, last\_name, job\_id, hire\_date*

*FROM employees;*

*SQL> SAVE d:\...\p711.sql*

Precizarea extensiei *.sql* a fișierului nu este obligatorie.

8. a) Executați cererea din fișierul *p711.sql*.

*SQL> START d:\...\p711.sql*

sau

*SQL> @ d:\...\p711.sql*

b) Editați fișierul *p711.sql*, astfel încât, la rulare, capetele coloanelor să aibă numele *cod*, *nume*, *cod job*, *data angajării*.

*SQL> EDIT d:\...\p711.sql*

Cererea modificată va fi:

*SELECT employee\_id cod, last\_name nume, job\_id "cod job", hire\_date "data angajării"*

*FROM employees;*

9. Să se listeze, cu și fără duplicate, codurile job-urilor din tabelul *EMPLOYEES*.

*SQL> SELECT job\_id FROM employees;*

*SQL> SELECT DISTINCT job\_id FROM employees;*

10. Să se afișeze numele concatenat cu *job\_id*-ul, separate prin virgula și spațiu. Etichetați coloana "Angajat si titlu".

**Obs:** Operatorul de concatenare este *||*. Șirurile de caractere se specifică între apostrofuri (NU ghilimele, caz în care ar fi interpretate ca *alias*-uri).

*SQL> SELECT last\_name|| ' , ' || job\_id "Angajat si titlu"*

*FROM employees;*

11. Creați o cerere prin care să se afișeze toate datele din tabelul *EMPLOYEES* pe o singură coloană. Separați fiecare coloană printr-o virgulă. Etichetați coloana "Informatii complete".

12. Să se listeze numele și salariul angajaților care câștigă mai mult de 2850.

```
SQL> SELECT last_name, salary
      FROM employees
      WHERE salary > 2850;
```

```
SQL> SAVE p12/1.sql
```

```
SQL> @p12/1.sql sau START p12/1.sql
```

13. Să se creeze o cerere pentru a afișa numele angajatului și numărul departamentului pentru angajatul având codul 104.

14. Să se modifice cererea de la problema 12 pentru a afișa numele și salariul angajaților al căror salariu nu se află în intervalul [1500, 2850].

**Obs:** Pentru testarea apartenenței la un domeniu de valori se poate utiliza operatorul *[NOT] BETWEEN valoare1 AND valoare2*.

15. Să se afișeze numele, job-ul și data la care au început lucrul salariații angajați între 20 Februarie 1987 și 1 Mai 1989. Rezultatul va fi ordonat crescător după data de început.

```
SQL> SELECT __, __, __
      FROM __
      WHERE __ BETWEEN '20-FEB-1987' __ '1-MAY-1989'
      ORDER BY __;
```

16. Să se afișeze numele salariaților și codul departamentelor pentru toți angajații din departamentele 10 și 30 în ordine alfabetică a numelor.

```
SQL> SELECT __, __
      FROM __
      ____ department_id IN (10, 30)
      ____;
```

**Obs:** Apartenența la o mulțime finită de valori se poate testa prin intermediul operatorului *IN*, urmat de lista valorilor (specificate între paranteze și separate prin virgule):  
*expresie IN (valoare\_1, valoare\_2, ..., valoare\_n)*

17. Să se modifice cererea de la problema 14 pentru a lista numele și salariile angajaților care câștigă mai mult de 1500 și lucrează în departamentul 10 sau 30. Se vor eticheta coloanele drept *Angajat* și *Salariu lunar*.

18. Care este data curentă? Afișați diferite formate ale acesteia.

**Obs:**

➤ Funcția care returnează data curentă este *SYSDATE*. Pentru completarea sintaxei obligatorii a comenzii *SELECT*, se utilizează tabelul *DUAL*:

```
SQL> SELECT SYSDATE
      FROM dual;
```

➤ Datele calendaristice pot fi formate cu ajutorul funcției *TO\_CHAR(data, format)*, unde formatul poate fi alcătuit dintr-o combinație a următoarelor elemente:

Element	Semnificație
D	Numărul zilei din săptămâna (duminică=1; luni=2; ...sâmbătă=6)
DD	Numărul zilei din lună.
DDD	Numărul zilei din an.
DY	Numele zilei din săptămână, printr-o abreviere de 3 litere (MON, THU etc.)
DAY	Numele zilei din săptămână, scris în întregime.
MM	Numărul lunii din an.
MON	Numele lunii din an, printr-o abreviere de 3 litere (JAN, FEB etc.)
MONTH	Numele lunii din an, scris în întregime.
Y	Ultima cifră din an
YY, YYYY, YYYY	Ultimele 2, 3, respectiv 4 cifre din an.

YEAR	Anul, scris în litere (ex: <i>two thousand four</i> ).
HH12, HH24	Orele din zi, între 0-12, respectiv 0-24.
MI	Minutele din oră.
SS	Secundele din minut.
SSSSS	Secundele trecute de la miezul nopții.

19. Să se afișeze numele și data angajării pentru fiecare salariat care a fost angajat în 1987. Se cer 2 soluții: una în care se lucrează cu formatul implicit al datei și alta prin care se formatează data.

**Varianta1:**

.....  
 WHERE hire\_date LIKE ('%87%');

**Varianta 2:**

.....  
 WHERE TO\_CHAR(hire\_date, 'YYYY')='1987';

Sunt obligatorii ghilimelele de la șirul '1987'? Ce observați?

20. Să se afișeze numele și job-ul pentru toți angajații care nu au manager.

SQL> SELECT \_\_\_\_\_, \_\_\_\_\_  
 FROM \_\_\_\_\_  
 WHERE manager\_id IS NULL;

21. Să se afișeze numele, salariul și comisionul pentru toți salariații care câștigă comision. Să se sorteze datele în ordine descrescătoare a salariilor și comisioanelor.

SQL> SELECT \_\_\_\_, \_\_\_\_, \_\_\_\_  
 FROM \_\_\_\_\_  
 WHERE \_\_\_\_\_  
 ORDER BY salary DESC, commission\_pct DESC;

22. Eliminați clauza WHERE din cererea anterioară. Unde sunt plasate valorile NULL în ordinea descrescătoare?

23. Să se listeze numele tuturor angajaților care au a treia literă din nume 'A'.

**Obs:** Pentru compararea șirurilor de caractere, împreună cu operatorul LIKE se utilizează caracterele wildcard:

- % - reprezentând orice șir de caractere, inclusiv șirul vid;
- \_ (underscore) – reprezentând un singur caracter și numai unul.

24. Să se listeze numele tuturor angajaților care au 2 litere 'L' în nume și lucrează în departamentul 30 sau managerul lor este 102.

25. Să se afișeze numele, job-ul și salariul pentru toți salariații al căror job conține șirul "CLERK" sau "REP" și salariul nu este egal cu 1000, 2000 sau 3000. (operatorul NOT IN)

26. Să se afișeze numele departamentelor care nu au manager.