Baze de date-Anul 2
Laborator 1 SQL

I. Introducere

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

Un sistem de gestiune a bazei de date (SGBD) este un produs software care asigură interacțiunea cu o bază de date, permițând definirea, consultarea şi actualizarea datelor din baza de date.

2. Ce este SQL?

SQL (*Structured Query Language*) este un limbaj neprocedural pentru interogarea şi prelucrarea informațiilor din baza de date.

Compilatorul limbajului *SQL* generează automat o procedură care accesează baza de date şi execută comanda dorită.

SQL permite atât definirea, prelucrarea şi interogarea datelor, cît şi controlul accesului la acestea. 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 recunoaşte instrucțiunile SQL şi le trimite server-ului Oracle pentru execuție.

Dintre funcționalitățile mediului *SQL*Plus*, se pot enumera:

- editarea, executarea, salvarea şi regăsirea instrucțiunilor *SQL* şi a blocurilor *PL/SQL*;
- calculul, stocarea şi afişarea rezultatelor furnizate de cereri;
- listarea structurii tabelelor;
- accesarea şi copierea de informații dintr-o bază de date în alta;
- administrarea bazei de date.

Tabelul următor evidențiază diferențele dintre instrucțiunile *SQL* şi cele *SQL*Plus*:

SQL	*SQL*Plus*
Este un limbaj de comunicare cu *server*-ul *Oracle* pentru accesarea datelor.	Recunoaşte instrucțiunile *SQL* şi le transferă *server*-ului *Oracle*.
Se bazează pe standardul *ANSI* pentru *SQL*.	Este o interfață specifică sistemului *Oracle* pentru execuția instrucțiunilor *SQL*.
Prelucrează date şi defineşte obiecte din baza de date.	Nu permite prelucrarea informațiilor din baza de date.
Nu are un caracter de continuare.	Acceptă „–“ drept caracter de continuare pentru comenzile scrise pe mai multe linii.
Instrucțiunile nu pot fi abreviate.	Comenzile pot fi abreviate.
Utilizează funcții pentru a efectua formatări.	Utilizează comenzi pentru formatarea datelor.
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*, şi 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.

Pe lângă comenzile care alcătuiesc aceste limbaje, *SQL* cuprinde:

- 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)? In instructiunea urmatoare 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 şi cap de coloană pentru expresia respectivă. Doar cele specificate între ghilimele sunt *case-sensitive*, celelalte fiind scrise implicit cu majuscule.

II. Exerciții

1. a) Consultați diagrama exemplu HR (Human Resources) pentru lucrul în cadrul laboratoarelor SQL.

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

2. Să se inițieze o sesiune SQL*Plus folosind *user ID*-ul şi parola 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, afişând 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 dă comanda:

SQL> /

sau

SQL> RUN

6. Listați structura tabelului *EMPLOYEES* (*DESC employees*) şi apoi dați comanda *RUN* (sau "/"). Ce observați? Comenzile *SQL*Plus* sunt păstrate în *buffer*?

7. Să se afişeze codul angajatului, numele, codul job-ului, data angajarii. Ce fel de operație este aceasta (selecție sau proiecție)? Salvați instrucțiunea *SQL* într-un fişier *p7l1.sql*.

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

```
SQL> SELECT employee_id, last_name, job_id, hire_date
       FROM employees;
SQL> SAVE h:\…\p7l1.sql
```

Precizarea extensiei *.SQL* a fişierului nu este obligatorie.

8. a) Executați cererea din fişierul *p7l1.sql*.

```
SQL> START h:\…\p7l1.sql
```

sau

```
SQL> @ h:\…\p7l1.sql
```

b) Editați fişierul *p7l1.sql*, astfel încât, la rulare, capetele coloanelor să aibă numele *cod*, *nume*, *cod job*, *data angajarii*.

```
SQL> EDIT h:\…\p7l1.sql
```

Cererea modificată va fi:

```
SELECT employee_id cod, last_name nume, job_id "cod job", hire_date "data angajarii"
FROM employees;
```

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

```
SQL> SELECT job_id FROM employees;
SQL> SELECT DISTINCT job_id FROM employees;
```

10. Să se afişeze numele concatenat cu job_id-ul, separate prin virgula si spatiu, si etichetati coloana "Angajat si titlu".

Obs: Operatorul de concatenare este "||". Şirurile 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. Creati o cerere prin care sa se afiseze toate datele din tabelul *EMPLOYEES*. Separați fiecare coloană printr-o virgulă. Etichetati coloana "Informatii complete".

12. Sa se listeze numele si salariul angajaților care câştigă mai mult de 2850 $. Salvați instrucțiunea *SQL* într-un fişier numit *p12l1.sql*. Să se ruleze acesta.

```
SQL> SELECT last_name, salary
          FROM    employees
          WHERE salary > 2850;
SQL> SAVE p12l1.sql
SQL> @p12l1.sql      sau      START p12l1.sql
```

13. Să se creeze o cerere pentru a afişa numele angajatului şi numărul departamentului pentru angajatul nr. 104.

```
SQL> SELECT  last_name, department_id
      FROM     employees
      WHERE   employee_id =104;
```

14. Să se modifice *p12l1.sql* pentru a afişa numele şi salariul pentru toți angajații al căror salariu nu se află în domeniul 1500-2850$. Salvați din nou instrucțiunea într-un fişier numit *p14l1.sql*. Executați cererea.

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

15. Să se afişeze numele, job-ul şi data la care au început lucrul salariații angajați între 20 Februarie 1987 şi 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 afişeze numele salariaților şi codul departamentelor pentru toti angajații din departamentele 10 şi 30 în ordine alfabetică a numelor.

```
SQL> SELECT __, __
       FROM  ___
        ____ deptno IN (10, 30)
        _____ _____;
```

Obs: Apartenența la o mulțime finită de valori se poate testa prin intermediul operatorului IN, urmat de lista valorilor între paranteze şi separate prin virgule:

expresie IN (valoare_1, valoare_2, …, valoare_n)

17. Să se modifice *p14l1.sql* pentru a lista numele şi salariile angajatilor care câştigă mai mult de 1500 $ şi lucrează în departamentul 10 sau 30. Se vor eticheta coloanele drept *Angajat* si *Salariu lunar*. Salvați noua instructiune *SQL* într-un fişier numit *p17l1.sql*. Executati cererea.

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

Obs:

- Pseudocoloana 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 formatate 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 (duminica=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, YYY, YYYY	Ultimele 2, 3, respectiv 4 cifre din an.
YEAR	Anul, scris în litere (ex: *two thousand four*).
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. Sa se afiseze numele şi data angajării pentru fiecare salariat care a fost angajat in 1987. Se cer 2 soluții: una în care se lucrează cu formatul implicit al datei şi alta prin care se formatează data.

Varianta1:

```
SQL> SELECT first_name, last_name, hire_date
     FROM employees
     WHERE hire_date LIKE ('%87%');
```

Varianta 2:

```
SQL> SELECT first_name, last_name, hire_date
     FROM employees
     WHERE  TO_CHAR(hire_date, 'YYYY')='1987';
```

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

20. Să se afişeze numele şi job-ul pentru toți angajații care nu au manager.

```
SQL> select last_name, job_id
     FROM employees
     WHERE manager_id IS NULL;
```

21. Sa se afiseze numele, salariul si comisionul pentru toti salariatii care castiga comisioane. Sa se sorteze datele in ordine descrescatoare a salariilor si comisioanelor.

```
SQL> SELECT ___, ____, ____
     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 angajatilor care au a treia literă din nume 'A'.

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

- % - reprezentând orice şir de caractere, inclusiv şirul vid;
- _ (*underscore*) – reprezentând un singur caracter şi numai unul.

```
SQL>  SELECT DISTINCT last_name
      FROM employees
```

WHERE LOWER(last_name) LIKE '__a%';

24. Să se listeze numele tuturor angajatilor care au 2 litere 'L' in nume şi lucrează în departamentul 30 sau managerul lor este 7782.

25. Să se afiseze numele, job-ul si salariul pentru toti salariatii al caror job conține şirul "clerk" sau "rep" si salariul nu este egal cu 1000, 2000 sau 3000 $. (operatorul NOT IN)

26. Sa se modifice *p17l1.sql* pentru a afisa numele, salariul si comisionul pentru toti angajatii al caror salariu este mai mare decat comisionul (*salary*commission_pct*) marit de 5 ori. Executati din nou cererea . Salvati cererea ca *p26l1.sql*.