Baze de date-Anul 2
Laborator 4 SQL

Funcţii grup şi clauzele GROUP BY, HAVING. Operatorii ROLLUP şi CUBE.

I. [Funcţii grup şi clauza GROUP BY]

- Clauza *GROUP BY* este utilizată pentru a diviza liniile unui tabel în grupuri. Pentru a returna informaţia corespunzătoare fiecărui astfel de grup, pot fi utilizate funcţiile agregat. Ele pot apărea în clauzele:
    - *SELECT*
    - *ORDER BY*
    - *HAVING*.

Server-ul *Oracle* aplică aceste funcţii fiecărui grup de linii şi returnează un singur rezultat pentru fiecare mulţime.

- Dintre funcţiile grup definite în sistemul *Oracle*, se pot enumera: *AVG*, *SUM*, *MAX*, *MIN*, *COUNT*, *STDDEV*, *VARIANCE* etc. Tipurile de date ale argumentelor funcţiilor grup pot fi *CHAR*, *VARCHAR2*, *NUMBER* sau *DATE*.
    - Funcţiile *AVG*, *SUM*, *STDDEV* şi *VARIANCE* operează numai asupra valorilor numerice.
    - Funcţiile *MAX* şi *MIN* pot opera asupra valorilor numerice, caracter sau dată calendaristică.
- Absenţa clauzei GROUP BY conduce la aplicarea funcţiei grup pe mulţimea tuturor liniilor tabelului.
- Toate funcţiile grup, cu excepţia lui *COUNT*(*), ignoră valorile *null*. *COUNT*(*expresie*) returnează numărul de linii pentru care expresia dată nu are valoarea *null*. Funcţia *COUNT* returnează un număr mai mare sau egal cu zero şi nu întoarce niciodată valoarea *null*.
- Când este utilizată clauza *GROUP BY*, *server*-ul sortează implicit mulţimea rezultată în ordinea crescătoare a valorilor coloanelor după care se realizează gruparea.
- În clauza *GROUP BY* a unei cereri se pot utiliza operatorii *ROLLUP* şi *CUBE*. Aceşti operatori sunt disponibili începând cu versiunea *Oracle8i*.
- Expresiile din clauza *SELECT* a unei cereri care conţine opţiunea *GROUP BY* trebuie să reprezinte o proprietate unică de grup, adică fie un atribut de grupare, fie o funcţie de agregare aplicată tuplurilor unui grup, fie o expresie formată pe baza primelor două. Toate expresiile din clauza *SELECT*, cu excepţia funcţiilor de agregare, se trec în clauza *GROUP BY* (unde pot apărea cel mult 255 expresii).

II. [Clauza HAVING]

Opţiunea *HAVING* permite restricţionarea grupurilor de linii returnate, la cele care îndeplinesc o anumită condiţie.

Dacă această clauză este folosită în absenţa unei clauze *GROUP BY*, aceasta presupune că gruparea se aplică întregului tabel, deci este returnată o singură linie, care este reţinută în rezultat doar dacă este îndeplinită condiţia din clauza *HAVING*.

III. [Operatorul ROLLUP]

Acest operator furnizează valori agregat şi superagregat corespunzătoare expresiilor din clauza *GROUP BY*. Operatorul *ROLLUP* poate fi folosit pentru extragerea de statistici şi informaţii totalizatoare din mulţimile rezultate. Acest operator poate fi util la generarea de rapoarte, diagrame şi grafice.

Operatorul *ROLLUP* creează grupări prin deplasarea într-o singură direcție, de la dreapta la stânga, de-a lungul listei de coloane specificate în clauza *GROUP BY*. Apoi, se aplică funcția agregat acestor grupări. Dacă sunt specificate *n* expresii în operatorul *ROLLUP*, numărul de grupări generate va fi *n* + 1. Liniile care se bazează pe valoarea primelor *n* expresii se numesc linii obişnuite, iar celelalte se numesc linii superagregat.

Dacă în clauza *GROUP BY* sunt specificate *n* coloane, pentru a produce subtotaluri fără operatorul *ROLLUP* ar fi necesare *n* + 1 instrucțiuni *SELECT* conectate prin *UNION ALL*. Aceasta ar face execuția cererii ineficientă pentru că fiecare instrucțiune *SELECT* determină accesarea tabelului. Operatorul *ROLLUP* determină rezultatele efectuând un singur acces la tabel şi este util atunci când sunt implicate multe coloane în producerea subtotalurilor.

Ilustrăm aplicarea acestui operator prin urmatorul exemplu.

Exemplu:

Pentru departamentele având codul mai mic decât 50, să se afişeze:

- pentru fiecare departament şi pentru fiecare an al angajării (corespunzător departamentului respectiv), valoarea totală a salariilor angajaților în acel an;
- valoarea totală a salariilor pe departamente (indiferent de anul angajării);
- valoarea totală a salariilor (indiferent de anul angajării şi de departament).

```
SELECT   department_id, TO_CHAR(hire_date, 'yyyy'), SUM(salary)
FROM    employees
WHERE   department_id < 50
GROUP BY ROLLUP(department_id, TO_CHAR(hire_date, 'yyyy'));
```

Instrucțiunea precedentă va avea un rezultat de forma:

DEPARTMENT_ID	TO_CHAR(hire_date,'yyyy')	SUM(SALARY)
10	1987	4400
10		4400
20	1996	13000
20	1997	6000
20		19000
30	1994	11000
30	1995	3100
30	1997	5700
30	1998	2600
30	1999	2500
30		24900
40	1994	6500
40		6500
		54800

În rezultatul prezentat anterior se pot distinge 3 tipuri de linii.

1) Prima linie reprezintă suma salariilor angajaților în 1987 din departamentul care are codul 10. În mod similar se interpretează liniile din rezultat care au toate coloanele completate.

2) Linia a doua conține valoarea totală a salariilor din departamentul al cărui cod este 10. La fel se interpretează toate liniile care se disting prin faptul că valoarea coloanei *TO_CHAR(hire_date, 'dd')* este *null*.

3) Ultima linie conține suma salariilor tuturor angajatiilor din departamentele al căror cod este mai mic decat 50. Întrucât această linie corespunde totalului general, ea conține valoarea *null* pe toate coloanele, cu excepția câmpului *SUM*(*salary*)**.**

IV. [Operatorul CUBE]

Operatorul *CUBE* grupează liniile selectate pe baza valorilor tuturor combinațiilor posibile ale expresiilor specificate şi returnează câte o linie totalizatoare pentru fiecare grup. Acest operator este folosit pentru a produce mulțimi de rezultate care sunt utilizate în rapoarte. În vreme ce *ROLLUP* produce subtotalurile doar pentru o parte dintre combinațiile posibile, operatorul *CUBE* produce subtotaluri pentru toate combinațiile posibile de grupări specificate în clauza *GROUP BY*, precum şi un total general.

Dacă există *n* coloane sau expresii în clauza *GROUP BY*, vor exista 2^n combinații posibile superagregat. Din punct de vedere matematic, aceste combinații formează un cub *n*-dimensional, de aici provenind numele operatorului. Pentru producerea de subtotaluri fără ajutorul operatorului *CUBE* ar fi necesare 2^n instrucțiuni *SELECT* conectate prin *UNION ALL*.

Exemplu:

Pentru departamentele având codul mai mic decât 50 să se afişeze:

- valoarea totală a salariilor corespunzătoare fiecărui an de angajare, din cadrul fiecărui departament;
- valoarea totală a salariilor din fiecare departament (indiferent de anul angajarii);
- valoarea totală a salariilor corespunzatoare fiecărui an de angajare (indiferent de departament);
- valoarea totală a salariilor (indiferent de departament si de anul angajarii).

```
SELECT   department_id, TO_CHAR(hire_date, 'yyyy'), SUM(salary)
FROM    employees
WHERE   department_id < 50
GROUP BY CUBE(department_id, TO_CHAR(hire_date, 'yyyy'));
```

În plus față de rezultatul corespunzător operației *ROLLUP*, operatorul *CUBE* va produce linii care reprezintă suma salariilor pentru fiecare an de angajare corespunzător unui departament având codul mai mic decât 50. Aceste linii se disting prin faptul că valoarea coloanei *department_id* este *null*.

V. [Exerciții – funcții grup şi clauzele GROUP BY, HAVING]

1. a) Functiile grup includ valorile *NULL* in calcule?
   b) Care este deosebirea dintre clauzele *WHERE* şi *HAVING*?

2. Să se afişeze cel mai mare salariu, cel mai mic salariu, suma şi media salariilor tuturor angajaților. Etichetați coloanele Maxim, Minim, Suma, respectiv Media. Sa se rotunjeasca rezultatele. Sa se salveze instructiunea SQL intr-un fisier *p2l4.sql*.

```
SELECT MAX(salary) Maxim, _____, _____, _____
FROM    employees;
```

3. Să se modifice fisierul *p1l4.sql* pentru a se afişa minimul, maximul, suma şi media salariilor pentru fiecare job. Salvati acest fisier ca *p2l4.sql*. Executați cererea.

```
SELECT job_id, MAX(salary) Maxim, _____, _____, _____
FROM    employees
```

GROUP BY job_id;

4. Să se afişeze numărul de angajați pentru fiecare job.

SELECT job_id, COUNT(*)
FROM employees
GROUP BY ____;

5. Să se determine numărul de angajați care sunt şefi. Etichetati coloana “Nr. manageri”.

SELECT COUNT(DISTINCT manager_id) “Nr. Manageri”
FROM employees;

? De ce am folosit cuvântul cheie DISTINCT? Ce am fi obținut dacă îl omiteam?

6. Să se afişeze diferența dintre cel mai mare si cel mai mic salariu. Etichetati coloana “Diferenta”.

7. Scrieți o cerere pentru a se afişa numele departamentului, locația, numărul de angajați şi salariul mediu pentru angajații din acel departament. Coloanele vor fi etichetate corespunzător.

SELECT d.department_name “Departament”, l.city “Locatie”,
COUNT(*) “Nr angajati”, AVG(salary) “Salariu mediu”
FROM locations l, departments d, employees e
WHERE l.location_id = d.location_id
AND d.department_id = e.department_id
GROUP BY department_name, l.city;

!!!Obs: În clauza *GROUP BY* se trec obligatoriu toate coloanele prezente în clauza *SELECT*, care nu sunt argument al funcțiilor grup (a se vedea ultima observație de la punctul I).

8. Să se afişeze codul şi numele angajaților care câstiga mai mult decât salariul mediu din firmă. Se va sorta rezultatul în ordine descrescătoare a salariilor.

SELECT employee_id, first_name, last_name
FROM employees
WHERE salary > (SELECT AVG(salary)
FROM employees)
ORDER BY salary DESC;

9. Pentru fiecare şef, să se afişeze codul său şi salariul celui mai prost platit subordonat. Se vor exclude cei pentru care codul managerului nu este cunoscut. De asemenea, se vor exclude grupurile în care salariul minim este mai mic de 1000$. Sortați rezultatul în ordine descrescătoare a salariilor.

SELECT manager_id, MIN(salary)
FROM employees
WHERE manager_id IS NOT NULL
GROUP BY manager_id
HAVING MIN(salary) > 1000
ORDER BY 2 DESC;

10. Pentru departamentele in care salariul maxim depăşeşte 3000$, să se obțină codul, numele acestor departamente şi salariul maxim pe departament.

11. Care este salariul mediu minim al job-urilor existente? Salariul mediu al unui job va fi considerat drept media arirmetică a salariilor celor care îl practică.

SELECT MIN(AVG(salary))
FROM employees

GROUP BY job_id;

***Obs**:* Într-o imbricare de funcții agregat, criteriul de grupare specificat în clauza *GROUP BY* se referă doar la funcția agregat cea mai interioară. Astfel, într-o clauză SELECT în care există funcții agregat imbricate nu mai pot apărea alte expresii.

12. Să se afişeze codul, numele departamentului şi suma salariilor pe departamente.
13. Să se afişeze maximul salariilor medii pe departamente.
14. Sa se obtina codul, titlul şi salariul mediu al job-ului pentru care salariul mediu este minim.
15. Să se afişeze salariul mediu din firmă doar dacă acesta este mai mare decât 2500. (clauza *HAVING* fără *GROUP BY*)
16. Să se afişeze suma salariilor pe departamente şi, în cadrul acestora, pe job-uri.

```
SELECT department_id, job_id, SUM(salary)
FROM     employees
GROUP  BY department_id, job_id;
```

! Modificați această cerere astfel încât să se afişeze numele departamentelor şi titlul job-urilor.

17. Să se afişeze numele departamentului si cel mai mic salariu din departamentul avand cel mai mare salariu mediu.
18. Sa se afiseze codul, numele departamentului si numarul de angajati care lucreaza in acel departament pentru:

a) departamentele in care lucreaza mai putin de 4 angajati;
b) departamentul care are numarul maxim de angajati.

```
a) SELECT e.department_id, d. department_name, COUNT(*)
   FROM     departments d JOIN employees e
                  ON (d.department_id = e.department_id )
  WHERE e.department_id IN (SELECT       department_id
                                  FROM         employees
                                  GROUP BY department_id
                                  HAVING      COUNT(*) < 4)
  GROUP BY  e.department_id, d.department_name;
```

Sau

```
SELECT      e.department_id, d.department_name, COUNT(*)
FROM         employees  e JOIN  departments d
             ON (d.department_id = e.department_id )
GROUP BY  e.department_id, d.department_name
HAVING       COUNT(*)<4;
```

19. Sa se afiseze salariatii care au fost angajati în aceeaşi zi a lunii în care cei mai multi dintre salariati au fost angajati.

```
SELECT employee_id, last_name, TO_CHAR(hire_date, 'dd')
FROM     employees
WHERE   TO_CHAR(hire_date,'dd') IN
          (SELECT       TO_CHAR(hire_date,'dd')
            FROM         employees
            GROUP BY TO_CHAR(hire_date,'dd')
            HAVING COUNT(*)=(SELECT MAX(COUNT(*))
                                 FROM employees
```

GROUP BY TO_CHAR(hire_date,'dd')));

20. Să se obțină numărul departamentelor care au cel puțin 15 angajați.

```
SELECT      COUNT(COUNT(department_id))
FROM        employees
GROUP BY department_id
HAVING      COUNT(*) > 15;
```

Ce reprezintă rezultatul returnat de cererea:

```
SELECT      COUNT(department_id)
FROM        employees
GROUP BY department_id
HAVING      COUNT(*) > 15;
```

21. Să se obțină codul departamentelor şi suma salariilor angajaților care lucrează în acestea, în ordine crescătoare. Se consideră departamentele care au mai mult de 10 angajați şi al căror cod este diferit de 30.
22. Sa se afiseze codul, numele departamentului, numarul de angajati si salariul mediu din departamentul respectiv, impreuna cu numele, salariul si jobul angajatilor din acel departament. Se vor afişa şi departamentele fără angajați (outer join).
23. Scrieti o cerere pentru a afisa, pentru departamentele avand codul > 80, salariul total pentru fiecare job din cadrul departamentului. Se vor afisa orasul, numele departamentului, jobul si suma salariilor. Se vor eticheta coloanele corespunzator.

Obs: Plasați condiția *department_id* > 80, pe rând, în clauzele *WHERE* şi *HAVING*. Testați în fiecare caz. Ce se observă? Care este diferența dintre cele două abordări?

24. Care sunt angajatii care au mai avut cel putin doua joburi?
25. Să se calculeze comisionul mediu din firmă, luând în considerare toate liniile din tabel.

Obs: Funcțiile grup ignoră valorile *null*. Prin urmare, instrucțiunea

```
SELECT AVG(commission_pct)
FROM    employees;
```

va returna media valorilor pe baza liniilor din tabel pentru care există o valoare diferită de *null*. Astfel, reiese că suma valorilor se împarte la numărul de valori diferite de *null*. Calculul mediei pe baza tuturor liniilor din tabel se poate realiza utilizând funcțiile *NVL*, *NVL2* sau *COALESCE*:

```
SELECT AVG(NVL(commission_pct, 0))
FROM    employees;
```

O altă variantă este dată de o cerere de forma:

```
SELECT SUM(commission_pct)/COUNT(*)
FROM employees;
```

VI. [Exerciții – ROLLUP şi CUBE]

26. Analizați cele 2 exemple prezentate mai sus (III – IV), referitor la operatorii *ROLLUP* şi *CUBE*.

VII. [Exerciții – DECODE, subcereri în clauza SELECT]

27. Scrieți o cerere pentru a afişa job-ul, salariul total pentru job-ul respectiv pe departamente si salariul total pentru job-ul respectiv pe departamentele 30, 50, 80. Se vor eticheta coloanele corespunzător. Rezultatul va apărea sub forma de mai jos:

```
Job          Dep30        Dep50     Dep80            Total
-------------------------------------------------------------------------
 ……
 ……
```

```
SELECT job_id, SUM(DECODE(department_id, 30, salary)) Dep30,
          SUM(DECODE(department_id, 50, salary)) Dep50,
          SUM(DECODE(department_id, 80, salary)) Dep80,
          SUM(salary) Total
FROM    employees
GROUP BY job_id;
```

Metoda 2: *(cu subcereri corelate în clauza SELECT)*

```
SELECT job_id, (SELECT SUM(salary)
                  FROM     employees
                  WHERE  department_id = 30
                  AND job_id = e.job_id) Dep30,
                 (SELECT SUM(salary)
                  FROM     employees
                  WHERE  department_id = 50
                  AND job_id = e.job_id)  Dep50,
                 (SELECT SUM(salary)
                  FROM     employees
                  WHERE   department_id = 80
                  AND       job_id = e.job_id) Dep80,
          SUM(salary) Total
FROM    employees e
GROUP BY job_id;
```

28. Să se creeze o cerere prin care să se afişeze numărul total de angajați şi, din acest total, numărul celor care au fost angajați în 1997, 1998, 1999 si 2000. Denumiti capetele de tabel in mod corespunzator.

29. Rezolvați problema 22 cu ajutorul subcererilor specificate în clauza *SELECT*.

VIII. [Exerciții – subcereri în clauza FROM]

Obs: Subcererile pot apărea în clauza SELECT, WHERE sau FROM a unei cereri. O subcerere care apare în clauza FROM se mai numeşte *view in-line*.

30. Să se afişeze codul, numele departamentului şi suma salariilor pe departamente.

```
SELECT d.department_id, department_name,a.suma
FROM departments d,   (SELECT department_id ,SUM(salary) suma
                          FROM     employees
                          GROUP BY department_id) a
WHERE d.department_id =a.department_id;
```

31. Să se afişeze numele, salariul, codul departamentului si salariul mediu din departamentul respectiv.

32. Modificați cererea anterioară, pentru a determina şi listarea numărului de angajați din departamente.

33. Pentru fiecare departament, să se afişeze numele acestuia, numele şi salariul celor mai prost plătiți angajați din cadrul său.
34. Rezolvați problema 22 cu ajutorul subcererilor specificate în clauza *FROM*.