Baze de date - Anul 1

Laborator 9

Operatorul *DIVISION*. *SQL*Plus*

I. [Obiective]

- Implementarea operatorului *DIVISION* prin diferite metode.
- Prezentarea succintă a câtorva comenzi utile din mediul SQL*Plus, prezente și în *SQL Developer*.
- Exerciții recapitulative.

II. [Implementarea operatorului *DIVISION* în *SQL*]

Diviziunea este o operaţie binară care defineşte o relaţie ce conţine valorile atributelor dintr-o relaţie care apar **în toate** valorile atributelor din cealaltă relaţie.

Operatorul *DIVISION* este legat de cuantificatorul universal (∀) care nu există în *SQL*. Cuantificatorul universal poate fi însă simulat cu ajutorul cuantificatorului existenţial (∃) utilizând relaţia:

$\forall x\, P(x) \equiv \neg\, \exists\, x \neg\, P(x)$.

Prin urmare, operatorul *DIVISION* poate fi exprimat în *SQL* prin succesiunea a doi operatori *NOT EXISTS*. Alte modalităţi de implementare a acestui operator vor fi prezentate în exemplul de mai jos.

Extindem diagrama *HR* cu o nouă entitate, *PROJECT*, şi o nouă asociere: „angajat lucrează în cadrul unui proiect”, între entităţile *EMPLOYEES* şi *PROJECT*. Aceasta este o relaţie *many-to-many*, care va conduce la apariţia unui tabel asociativ, numit *WORKS_ON*.

O altă asociere între entităţile *EMPLOYEES* şi *PROJECT* este „angajat conduce proiect”. Aceasta este o relaţie *one-to-many*.

Noile tabele au următoarele scheme relaţionale:

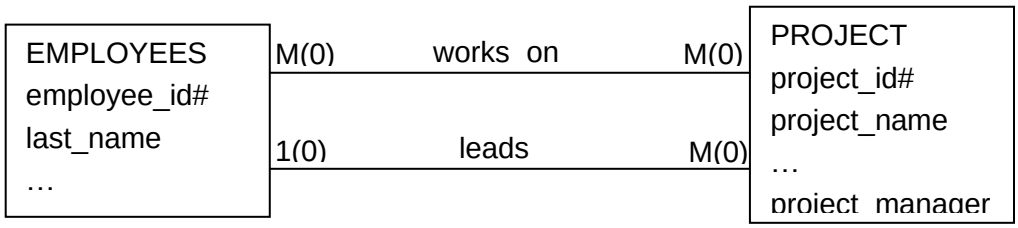
1) *PROJECT*(*project_id*#, *project_name*, *budget*, *start_date*, *deadline*, *delivery_date*, *project_manager*)
- *project_id* reprezintă codul proiectului şi este cheia primară a relaţiei *PROJECT*
- *project_name* reprezintă numele proiectului
- *budget* este bugetul alocat proiectului
- *start_date* este data demarării proiectului
- *deadline* reprezintă data la care proiectul trebuie să fie finalizat

- *delivery_date* este data la care proiectul este livrat efectiv
- *project_manager* reprezintă codul managerului de proiect şi este cheie externă. Pe cine referă această coloană ? Ce relaţie implementează această cheie externă?

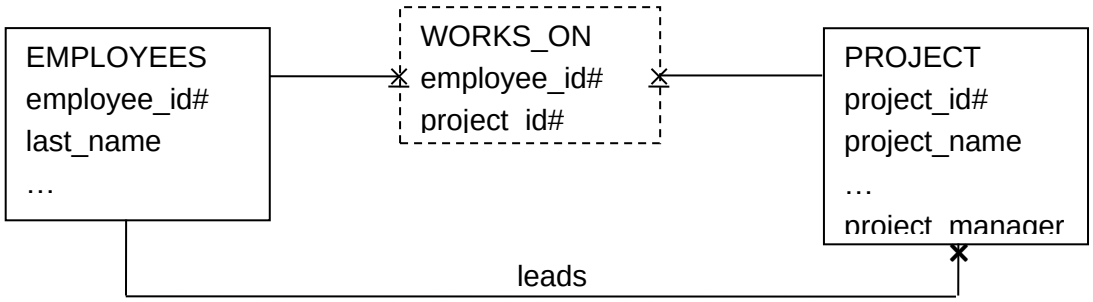
2) *WORKS_ON*(*project_id*#, *employee_id*#, *start_date*, *end_date*)
- cheia primară a relaţiei este compusă din atributele *employee_id* şi *project_id*.

Scriptul pentru crearea noilor tabele şi inserarea de date în acestea este *hr_project.sql*.

Diagrama entitate-relaţie corespunzătoare modelului *HR* va fi extinsă, pornind de la entitatea *EMPLOYEES*, astfel:



Partea din diagrama conceptuală corespunzătoare acestei extinderi a modelului este următoarea:



Exemplu: Să se obţină codurile salariaţilor ataşaţi tuturor proiectelor pentru care s-a alocat un buget egal cu 10000.

Metoda 1 (utilizând de 2 ori *NOT EXISTS*):

```
SELECT DISTINCT employee_id
FROM   works_on a
WHERE NOT EXISTS
      (SELECT   1
       FROM     project p
       WHERE    budget=10000
       AND NOT EXISTS
           (SELECT  'x'
```

```
            FROM   works_on b
            WHERE  p.project_id=b.project_id
            AND    b.employee_id=a.employee_id));
```

Metoda 2 (simularea diviziunii cu ajutorul funcţiei *COUNT*):

```
SELECT employee_id
FROM   works_on
WHERE  project_id IN
     (SELECT    project_id
      FROM      project
      WHERE     budget=10000)
GROUP BY   employee_id
HAVING COUNT(project_id)=
           (SELECT    COUNT(*)
            FROM      project
            WHERE     budget=10000);
```

Metoda 3 (operatorul *MINUS*):

```
SELECT employee_id
FROM   works_on
MINUS
SELECT employee_id from
    (SELECT employee_id, project_id
     FROM (SELECT DISTINCT employee_id FROM works_on) t1,
          (SELECT project_id FROM project WHERE budget=10000) t2
     MINUS
     SELECT employee_id, project_id
     FROM works_on
     ) t3;
```

Metoda 4 (A include B <=> B-A = Ø):

```
SELECT     DISTINCT employee_id
FROM       works_on a
WHERE NOT EXISTS (
 (SELECT   project_id
  FROM     project p
  WHERE    budget=10000)
  MINUS
 (SELECT   p.project_id
  FROM     project p, works_on b
  WHERE    p.project_id=b.project_id
  AND      b.employee_id=a.employee_id));
```

III. [Exerciții – *DIVISION* + alte cereri]

1. Să se listeze informaţii despre angajaţii care au lucrat în toate proiectele demarate în primele 6 luni ale anului 2016. Implementaţi toate variantele.

2. Să se listeze informaţii despre proiectele la care au participat toţi angajaţii care au deţinut alte 2 posturi în firmă.

3. Să se obţină numărul de angajaţi care au avut cel puţin trei job-uri, luându-se în considerare şi job-ul curent. Presupunem că pot exista valori *null* în coloana *job_id* din tabelul *EMPLOYEES*.

4. Pentru fiecare ţară, să se afişeze numele său și numărul de angajaţi din cadrul acesteia.

5. Să se listeze angajaţii (codul şi numele acestora) care au lucrat pe cel puţin două proiecte nelivrate la termen.

6. Să se listeze codurile angajaţilor şi codurile proiectelor pe care au lucrat. Listarea va cuprinde şi angajaţii care nu au lucrat pe niciun proiect.

7. Să se afişeze angajaţii care lucrează în acelaşi departament cu cel puţin un manager de proiect.

8. Să se afişeze angajaţii care nu lucrează în acelaşi departament cu niciun manager de proiect.

9. Să se determine departamentele având media salariilor mai mare decât un număr dat.

Observație: Este necesară o variabilă de substituţie. Apariţia acesteia este indicată prin caracterul „&”. O prezentare a variabilelor de substituţie va fi făcută în a doua parte a acestui laborator.

...

HAVING AVG(salary) > &p;

10. Se cer informaţii (nume, prenume, salariu, număr proiecte) despre managerii de proiect care au condus 2 proiecte.

11. Să se afişeze lista angajaţilor care au lucrat numai pe proiecte conduse de managerul de proiect având codul 102.

12. a) Să se obţină numele angajaţilor care au lucrat **cel puţin** pe aceleaşi proiecte ca angajatul având codul 200.

Observație: Incluziunea dintre 2 mulţimi se testează cu ajutorul proprietăţii „A inclus în B <=> A-B =Ø”. Cum putem implementa acest lucru în SQL?

Pentru rezolvarea exerciţiului, trebuie selectaţi angajaţii pentru care este vidă lista proiectelor pe care a lucrat angajatul 200 minus lista proiectelor pe care au lucrat acei angajaţi.

b) Să se obţină numele angajaţilor care au lucrat **cel mult** pe aceleaşi proiecte ca angajatul având codul 200.

13. Să se obţină angajaţii care au lucrat pe aceleaşi proiecte ca angajatul având codul 200.
Obs: Egalitatea între două mulţimi se testează cu ajutorul proprietăţii „A=B <=> A-B=Ø şi B-A=Ø”.

14. Modelul HR conţine un tabel numit *JOB_GRADES*, care stochează grilele de salarizare ale companiei.

a) Afişaţi structura şi conţinutul acestui tabel.

b) Pentru fiecare angajat, afişaţi numele, prenumele, salariul şi grila de salarizare corespunzătoare. Ce operaţie are loc între tabelele din interogare?

15. Schimbaţi salariul şi comisionul celui mai prost plătit salariat din firmă, astfel încât să fie egale cu salariul si comisionul şefului său.

16. Să se modifice adresa de e-mail pentru angajaţii care câştigă cel mai mult în departamentul în care lucrează astfel încât acesta să devină iniţiala numelui concatenată cu prenumele. Dacă nu are prenume atunci în loc de acesta apare caracterul ‘.’. Anulaţi modificările.

17. Pentru fiecare departament să se mărească salariul celor care au fost angajaţi primii astfel încât să devină media salariilor din companie.

IV. [SQL*Plus]

Variabile de substituție

- Variabilele de substituție sunt utile în crearea de comenzi/script-uri dinamice (care depind de niște valori pe care utilizatorul le furnizează la momentul rulării).
- Variabilele de substituție se pot folosi pentru stocarea temporară de valori, transmiterea de valori între comenzi *SQL* etc. Ele pot fi create prin:
  - comanda *DEFINE*.(*DEFINE* variabila = valoare)
  - prefixarea cu & (indică existența unei variabile într-o comandă *SQL*; dacă variabila nu există, atunci *SQL*Plus* o creează).
  - prefixarea cu && (indică existența unei variabile într-o comandă *SQL*; dacă variabila nu există, atunci *SQL*Plus* o creează). Deosebirea față de & este că, dacă se folosește &&, atunci referirea ulterioară cu & sau && nu mai necesită ca

utilizatorul să introducă de fiecare dată valoarea variabilei. Este folosită valoarea dată la prima referire.

- Variabilele de substituție pot fi eliminate cu ajutorul comenzii *UNDEF*[*INE*]

Comanda *DEFINE*

Forma comenzii	**Descriere**
DEFINE variabila = valoare	Creează o variabilă utilizator cu valoarea de tip șir de caractere precizată.
DEFINE variabila	Afișează variabila, valoarea ei și tipul de date al acesteia.
DEFINE	Afișează toate variabilele existente în sesiunea curentă, împreună cu valorile și tipurile lor de date.

Observații:

- Variabilele de tip *DATE* sau *CHAR* trebuie să fie incluse între apostrofuri în comanda *SELECT*.
- Dupa cum numele sugerează, variabilele de sustitutie înlocuiesc/substituie în cadrul comenzii *SQL* variabila respectivă cu șirul de caractere introdus de utilizator.
- Variabilele de sustituție pot fi utilizate pentru a înlocui la momentul rulării:
  - condiții *WHERE*;
  - clauza *ORDER BY*;
  - expresii din lista *SELECT*;
  - nume de tabel;
  - o intreagă comandă *SQL*;
- Odată definită, o variabilă rămâne până la eliminarea ei cu o comanda *UNDEF* sau până la terminarea sesiunii *SQL*Plus* respective.
- Comanda *SET VERIFY ON | OFF* permite afișarea sau nu a formei comenzii înainte și după înlocuirea variabilei de substituție.

Comenzi interactive în SQL*Plus

Comanda	**Descriere**
ACC[*EPT*] *variabila* [*tip*]	Citeşte o linie de intrare şi o stochează într-o variabilă

[*PROMPT text*]	utilizator.
PAU[*SE*] [*text*]	Afişează o linie vidă, urmată de o linie conţinând text, apoi aşteaptă ca utilizatorul să apese tasta *return*. De asemenea, această comandă poate lista două linii vide, urmate de aşteptarea răspunsului din partea utilizatorului.
PROMPT [*text*]	Afişează mesajul specificat sau o linie vidă pe ecranul utilizatorului.

Fișiere *script*

De obicei, un fişier script constă în comenzi *SQL*Plus* şi cel puţin o instrucţiune *SELECT*. În mediul *SQL*Plus*, fișierul se execută prin comenzile @ sau *START*. În *SQL Developer*, se încarcă fişierul şi se acţionează butonul *Run Script*.

V. [Exerciţii – *SQL*Plus*]

18. Ce comenzi SQL*Plus ați utilizat în laboratoarele precedente?

19. (Exercițiu care nu se rezolvă dacă mediul de lucru este SQL Developer, ci doar în situația în care lucrăm în linie de comandă) Care sunt setările actuale pentru dimensiunea paginii şi a liniei în interfaţa SQL*Pus? Setaţi dimensiunea liniei la 100 de caractere şi pe cea a paginii la 24 de linii.

```
SHOW LINESIZE
SHOW PAGESIZE

SET LINESIZE 100
SET PAGESIZE 24
```

20. Să se afișeze codul, numele, salariul și codul departamentului din care face parte un angajat al cărui cod este introdus de utilizator de la tastatură. Analizați diferențele dintre cele 4 posibilități prezentate mai jos :

I.

```
SELECT employee_id, last_name, salary, department_id
FROM   employees
WHERE  employee_id = &p_cod;
```

II.

```
DEFINE p_cod; // Ce efect are?
SELECT employee_id, last_name, salary, department_id
FROM   employees
WHERE  employee_id = &p_cod;
UNDEFINE p_cod;
```

III.

```
DEFINE p_cod=100;
SELECT employee_id, last_name, salary, department_id
FROM   employees
WHERE  employee_id = &&p_cod;
UNDEFINE p_cod;
```

IV.

```
ACCEPT p_cod PROMPT "cod= ";
SELECT employee_id, last_name, salary, department_id
FROM   employees
WHERE  employee_id = &p_cod;
```

21. Să se afișeze numele, codul departamentului și salariul anual pentru toți angajații care au un anumit job.

22. Să se afișeze numele, codul departamentului și salariul anual pentru toate persoanele care au fost angajate după o anumită dată calendaristică.

23. Să se afișeze o coloană aleasă de utilizator, dintr-un tabel ales de utilizator, ordonând după aceeași coloană care se afișează. De asemenea, este obligatorie precizarea unei condiții în clauza WHERE.

```
SELECT   &&p_coloana
FROM     &p_tabel
WHERE    &p_where
ORDER BY &p_coloana;
```

24. Să se realizeze un script (fişier de comenzi) prin care să se afişeze numele, job-ul şi data angajării salariaţilor care au început lucrul între 2 date calendaristice introduse de utilizator. Să se concateneze numele şi job-ul, separate prin spaţiu şi virgulă, şi să se eticheteze coloana "Angajati". Se vor folosi comanda ACCEPT şi formatul pentru data calendaristică MM/DD/YY.

25. Să se realizeze un script pentru a afișa numele angajatului, codul job-ului, salariul și numele departamentului pentru salariații care lucrează într-o locație dată de utilizator. Va fi permisă cautarea case-insensitive.

26. Să se creeze un fişier (*script file*) care să permită introducerea de înregistrări în tabelul *EMP_PNU* în mod interactiv. Se vor cere utilizatorului: codul, numele, prenumele si salariul angajatului. Câmpul *email* se va completa automat prin concatenarea primei litere din prenume şi a primelor 7 litere din nume.

REM setari
REM comenzi ACCEPT
INSERT INTO emp_pnu
VALUES (&…);
REM suprimarea variabilelor utilizate
REM anularea setarilor, prin stabilirea acestora la valorile implicite

Executați script-ul pentru a introduce 2 înregistrări în tabel.

27. Creaţi un script prin intermediul căruia sa fie posibilă actualizarea în mod interactiv de înregistrări ale tabelului *dept_pnu*. Se va cere codul departamentului care urmează a fi actualizat, se va afişa linia respectivă, iar apoi se vor cere valori pentru celelalte câmpuri.

28. Să se creeze un fişier *script* prin care se cere un cod de angajat din tabelul *EMP_PNU*. Se va lista inregistrarea corespunzatoare acestuia, iar apoi linia va fi suprimată din tabel.