Baze de date-Anul 1
Laborator 6 SQL

I. Operatorul *DIVISION*.
II. *SQL*Plus*

I. 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 ($\forall$) care nu există în *SQL*. Cuantificatorul universal poate fi însă simulat cu ajutorul cuantificatorului existenţial ($\exists$) 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 lucreaza 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*.

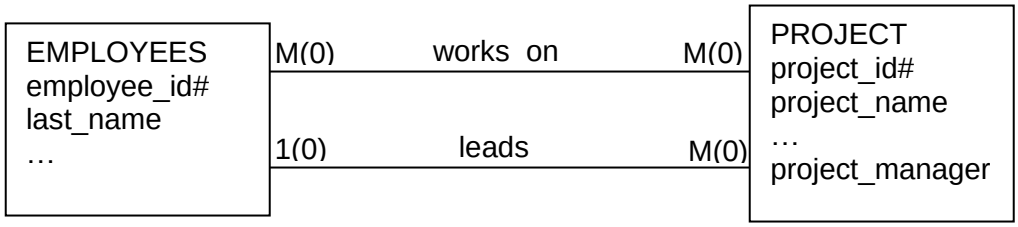
Noile tabele au următoarele scheme relaţionale:

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?

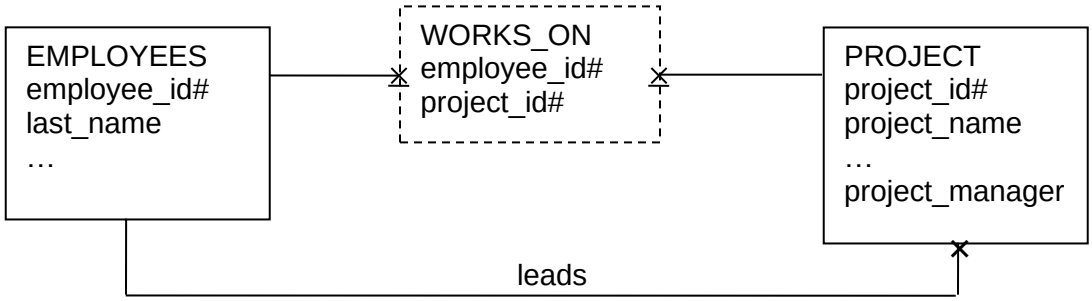
2) ***WORKS_ON***(project_id#, employee_id#, start_date, end_date)
- **cheia primară** a relaţiei este compusă din atributele **employee_id** şi **project_id**.

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
        WHEREbudget = 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
                   )
               );
```

Exerciţii (*DIVISION* + alte cereri):

1. Să se listeze **informaţii despre angajaţii** care au lucrat în **toate** proiectele demarate în primele 6 luni ale anului 2006.

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.

4. Pentru fiecare ţară, să se afişeze **numărul de angajaţi** din cadrul acesteia.

5. Să se listeze **codurile angajaţilor** şi **codurile proiectelor** pe care au lucrat. Listarea va cuprinde şi angajaţii care nu au lucrat pe nici un proiect.

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

7. Să se afişeze **angajaţii** care **nu lucrează** în acelaşi departament cu nici un manager de proiect.

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

Obs : 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;

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

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

    Obs: 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 mai puţin lista proiectelor pe care au lucrat acei angajaţi.

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

11. Să se obţină **angajaţii** care au lucrat pe **aceleaşi proiecte** ca şi angajatul având codul 200.

    Obs: Egalitatea între două mulţimi se testează cu ajutorul proprietăţii „A=B => A-B=Ø şi B-A=Ø”.

12. Modelul HR conţine un tabel numit **JOB_GRADES**, care conţine 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.

II. *SQL*Plus*

Variabile de substitutie

- Variabilele de substitutie sunt utile in crearea de comenzi/script-uri dinamice (care depind de niste valori pe care utilizatorul le furnizeaza la momentul rularii).
- Variabilele de substitutie se pot folosi pentru stocarea temporara de valori, transmiterea de valori intre comenzi *SQL* etc. Ele pot fi create prin:
    - comanda *DEFINE*.(*DEFINE* variabila = valoare)
    - Prefixarea cu & (indica existenta unei variabile intr-o comanda *SQL*, daca variabila nu exista, atunci *SQL*Plus* o creeaza).
    - Prefixarea cu && (indica existenta unei variabile intr-o comanda *SQL*, daca variabila nu exista, atunci *SQL*Plus* o creeaza). Deosebirea fata de & este ca, daca se foloseste &&, atunci referirea ulterioara cu & sau && nu mai cere ca utilizatorul sa introduca de fiecare data valoarea variabilei. Este folosita valoarea data la prima referire.

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

Comanda *DEFINE*

Forma comenzii	Descriere
DEFINE variabila = valoare	Creeaza o variabila utilizator cu valoarea de tip sir de caracter precizata.
DEFINE variabila	Afiseaza variabila, valoarea ei si tipul de data al acesteia.
DEFINE	Afiseaza toate variabilele existente in sesiunea curenta, impreuna cu valorile si tipurile lor de date.

Observatii:

- Variabilele de tip *DATE* sau *CHAR* trebuie sa fie incluse intre apostrofuri in comanda *SELECT*.
- Dupa cum le spune si numele, variabilele de substitutie inlocuiesc/substituie in cadrul comenzii *SQL* variabila respectiva cu sirul de caractere introdus de utilizator.
- Variabilele de substitutie pot fi utilizate pentru a inlocui la momentul rularii:

- conditii *WHERE*;
- clauza *ORDER BY*;
- expresii din lista *SELECT*;
- nume de tabel;
- o intreaga comanda *SQL*;

- Odata definita, o variabila ramane pana la eliminarea ei cu o comanda *UNDEF* sau pana la terminarea sesiunii *SQL*Plus* respective.
- Comanda *SET VERIFY ON | OFF* permite afisarea sau nu a comenzii inainte si dupa inlocuirea variabilei de substitutie.

Comenzi interactive in SQL*Plus

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

Cum se creeaza un fisier script ?

De obicei, un fişier script constă în comenzi *SQL*Plus* şi cel puţin o instrucţiune *SELECT*. Crearea unui fişier script simplu se poate realiza urmând etapele expuse în continuare.

1) Se redactează instrucţiunea *SELECT* la *prompt*-ul *SQL* sau în regiunea de editare din *iSQL*Plus*.
2) Se salvează instrucţiunea *SELECT* într-un fişier *script*.
3) Se editează fişierul *script*, adăugându-se comenzile *SQL*Plus* corespunzătoare.
4) Se verifică dacă instrucţiunea *SELECT* este urmată de un caracter pentru execuţie („;“ sau „/“).
5) Se salvează fişierul *script*.

Se execută fişierul *script* (prin comenzile @ sau *START*). În *SQL Developer*, se încarcă fişierul şi se acţionează butonul *Run Script*.

Exercitii (*SQL*Plus*)

13. 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
```

14. Sa se afiseze codul, numele, salariul si codul departamentului din care face parte pentru un angajat al carui cod este introdus de utilizator de la tastatura. Analizati diferentele dintre cele 4 posibilitati prezentate mai jos :

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

II.
DEFINE p_cod;
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;
```

15. Sa se afiseze o coloana aleasa de utilizator, dintr-un tabel ales de utilizator, ordonand dupa aceeasi coloana care se afiseaza. De asemenea, este obligatorie precizarea unei conditii WHERE.

```
SELECT   &&p_coloana  -- && determina ca valoarea lui p_coloana sa nu mai
            --fie ceruta si pentru clauza ORDER BY, urmand sa
            --fie utilizata valoarea introdusa aici pentru toate
            --aparitiile ulterioare ale lui &p_coloana
FROM     &p_tabel
WHERE    &p_where
ORDER BY &p_coloana;
```

16. Să se realizeze un script (fişier SQL*Plus) 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 calendaristica MM/DD/YY.

```
ACCEPT data_inceput PROMPT 'Introduceti data de inceput'
ACCEPT data_sfarsit PROMPT 'Introduceti data de sfarsit'
SELECT last_name, job_id, hire_date
FROM employees
WHERE hire_date BETWEEN TO_DATE('&data_inceput', 'mm/dd/yy')
                    AND TO_DATE('&data_sfarsit', 'mm/dd/yy');
```