1. TEORIE

Operatorul DIVISION

diviziune = identificarea tututor valorilor atributelor dintr-o relaţie care sunt corelate cu toate valorile din altă relaţie

Ex: Presupunand că un angajat poate lucra în mai multe departamente, dorim să aflăm toţi angajaţii care lucrează în toate departamentele

Operatorul DIVISION poate fi exprimat prin intermediul cuantificatorului (∀), dar deoarece acesta nu exista în SQL, el va fi simulat cu ajutorul lui (∃), astfel:

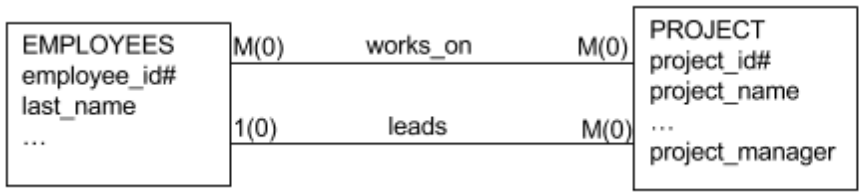
$$\forall x\ P(x) \equiv \neg \exists x \neg P(x)$$

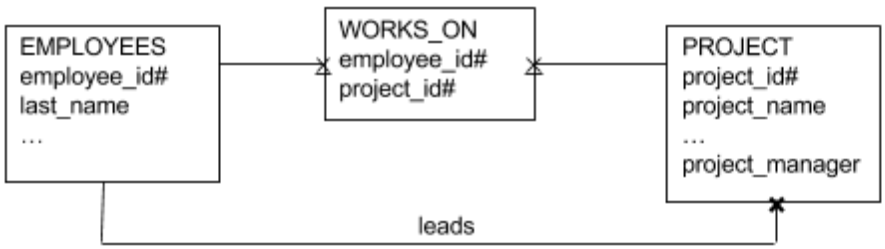
Ex: ∀ departamentul x, angajatul lucrează în cadrul lui ≡ nu există niciun departament x a.î angajatul nu lucrează în cadrul lui

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* astfel:

- o nouă entitate - ***PROJECT***
- 2 asocieri între entităţile *EMPLOYEES* şi *PROJECT*:
  - **„angajat lucreaza în cadrul unui proiect”** - relaţie *many-to-many*, care va conduce la apariţia unui tabel asociativ, numit ***WORKS_ON***
  - **„angajat conduce proiect”** -relaţie *one-to-many*





Pe baza următorului exemplu vor fi prezentate 4 metode de implementare a operatorului DIVISION:

Exemplu:

Să se obțină codurile salariaților ataşați 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 funcției *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);
```

Metoda 4 - B inclus în A => B \ A = Ø

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

Variabile de substituţie

➔ utile în crearea de comenzi / script-uri dinamice (care depind de niste valori pe care utilizatorul le furnizează la momentul rulării)

➔ se pot folosi pentru stocarea temporară de valori, transmiterea de valori între comenzi *SQL* etc.

➔ pot fi create prin:

❖ comanda *DEFINE* - crează o variabilă utilizator

```
DEFINE variabila = valoare;
```

OBS: Comanda DEFINE mai poate fi utilizată şi astfel:

1. `DEFINE variabila;` - afişează variabila, valoarea ei şi tipul de date
2. `DEFINE;` - afişează toate variabilele existente în sesiunea curentă, împreună cu valorile şi tipurile lor de date.

❖ Prefixarea cu &

- indică existenţa unei variabile într-o comandă *SQL*
- dacă variabila nu există, atunci *SQL*Plus* o creează

❖ Prefixarea cu &&

- indică existenţa unei variabile într-o comandă *SQL*
- dacă variabila nu există, atunci *SQL*Plus* o crează
- deosebirea fata de & este că, dacă se foloseşte &&, atunci referirea ulterioară cu & sau && nu mai cere ca utilizatorul să introducă de fiecare dată valoarea variabilei - este folosita valoarea dată la prima referire.

Exemplu:

Dorim să afişăm angajaţii care au salariul mai mare decat un număr citit de la tastatură:

```
DEFINE var = 10000;
SELECT *
FROM employees
WHERE salary > &var;

SELECT *
FROM employees
WHERE salary > &var;

SELECT *
FROM employees
WHERE salary > &&var;
```

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

```
UNDEFINE var;
```

Observatii:

- ➢ Variabilele de tip *DATE* sau *CHAR* trebuie să fie incluse între apostrofuri în comanda *SELECT*
- ➢ Odată definită, o variabilă rămâne până la eliminarea ei cu o comandă UNDEF sau până la terminarea sesiunii SQL*Plus respective
- ➢ Comandă `SET VERIFY ON | OFF` permite afişarea sau nu a comenzii înainte şi după înlocuirea variabilei de substituție.

Comenzi interactive în SQL*Plus

1. `ACC[EPT] variabila [tip] [PROMPT text]`
   - citeşte o linie de intrare şi o stochează într-o variabilă utilizator.
2. `PROMPT [text]`
   - afişează mesajul specificat sau o linie vidă pe ecranul utilizatorului.

---

2. EXERCIŢII

[Operatorul DIVISION]

1. Să se listeze informații despre angajații care au lucrat în toate proiectele demarate în primele 6 luni ale anului 2006. Implementați toate variantele.
2. Să se listeze informații despre proiectele la care au participat toți angajații care au deținut alte 2 posturi în firmă.
3. Să se obțină numărul de angajați care au avut cel puțin trei job-uri, luându-se în considerare şi job-ul curent.
4. Pentru fiecare țară, să se afişeze numărul de angajați din cadrul acesteia.
5. Să se listeze angajații (codul şi numele acestora) care au lucrat pe cel puțin două proiecte nelivrate la termen.
6. Să se listeze codurile angajaților şi codurile proiectelor pe care au lucrat. Listarea va cuprinde şi angajații care nu au lucrat pe nici un proiect.
7. Să se afişeze angajații care lucrează în acelaşi departament cu cel puțin un manager de proiect.
8. Să se afişeze angajații care nu lucrează în acelaşi departament cu nici un manager de proiect.
9. Se cer informații (nume, prenume, salariu, număr proiecte) despre managerii de proiect care au condus 2 proiecte.
10. Să se afişeze lista angajaților care au lucrat numai pe proiecte conduse de managerul de proiect având codul 102.
11.
    a. Să se obțină numele angajaților care au lucrat **cel puțin** pe aceleaşi proiecte ca şi angajatul având codul 200.
    b. Să se obțină numele angajaților care au lucrat **cel mult** pe aceleaşi proiecte ca şi angajatul având codul 200.

c. Să se obțină angajații care au lucrat pe aceleaşi proiecte ca şi angajatul având codul 200.

[Variabile de substitutie]

12. Să se afiseze codul, numele, salariul şi codul departamentului din care face parte pentru un angajat al carui cod este introdus de utilizator de la tastatura. 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;
```

13. Sa se afiseze numele, codul departamentului si salariul anual pentru toti angajatii care au un anumit job.
14. Sa se afiseze numele, codul departamentului si salariul anual pentru toti angajatii care au fost angajati dupa o anumita data calendaristica.
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.
16. Să se citească două date calendaristice de la tastatură şi să se afişeze zilele dintre aceste două date. Modificați cererea anterioară astfel încât să afişeze doar zilele lucrătoare dintre cele două date calendaristice introduse.
17. Să se determine departamentele având media salariilor mai mare decat un număr dat.