Baze de date-Anul 2
Laborator 9 SQL

- **[Operatorul DIVISION]**

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 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*.

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 DISTINCT employee_id
FROM    works_on
MINUS
SELECT employee_id from
   ( SELECT employee_id, project_id
     FROM (SELECT 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, A->mulțime proiecte angajat, B->mulțime proiecte cu buget 10000):

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

Exerciţii:

1. Să se listeze informaţii despre angajaţii care au lucrat în toate proiectele demarate în primele 6 luni ale anului 2006. Implementaţi toate variantele.
2. a) Să se obţină numele angajaţilor care au lucrat **cel puţin** pe aceleaşi proiecte ca şi angajatul având codul 200.

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

   c) 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”

Definirea vizualizărilor (*view*)

- **Vizualizările sunt tabele virtuale** construite pe baza unor tabele sau a altor vizualizări, denumite tabele de bază.
- Vizualizările nu conţin date, dar reflectă datele din tabelele de bază.
- Vizualizările sunt definite de o cerere *SQL*, motiv pentru care mai sunt denumite ***cereri stocate***.

- **Avantajele** utilizării vizualizărilor:
  - restricţionarea accesului la date;
  - simplificarea unor cereri complexe;
  - asigurarea independenţei datelor de programele de aplicaţii;
  - prezentarea de diferite imagini asupra datelor.

- **Crearea vizualizărilor** se realizează prin comanda *CREATE VIEW*, a cărei sintaxă simplificată este:

  CREATE [OR REPLACE] [FORCE | NOFORCE] VIEW
  nume_vizualizare [(alias, alias, ..)]
  AS *subcerere*
  [WITH CHECK OPTION [CONSTRAINT *nume_constrangere]]***
  [WITH READ ONLY [CONSTRAINT *nume_constrangere]];***

  - *OR REPLACE* se utilizează pentru a schimba definiţia unei vizualizări fără a mai reacorda eventualele privilegii.
  - Opţiunea *FORCE* permite crearea vizualizării inainte de definirea tabelelor, ignorând erorile la crearea vizualizării.
  - Subcererea poate fi oricât de complexă dar nu poate conţine clauza *ORDER BY*. Dacă se doreşte ordonare se utilizează *ORDER BY* la interogarea vizualizării.
  - *WITH CHECK OPTION* permite inserarea şi modificarea prin intermediul vizualizării numai a liniilor ce sunt accesibile vizualizării. Daca lipseşte numele constrângerii atunci sistemul asociază un nume implicit de tip SYS_C*n* acestei constrangeri (*n* este un număr astfel încât numele constrângerii să fie unic).
  - *WITH READ ONLY* asigură că prin intermediul vizualizarii nu se pot executa operaţii *LMD*.

- **Modificarea vizualizărilor** se realizează prin recrearea acestora cu ajutorul opţiunii *OR REPLACE*. Totuşi, începând cu *Oracle9i*, este posibilă utilizarea comenzii *ALTER VIEW* pentru adăugare de constrângeri vizualizării.

- **Suprimarea vizualizărilor** se face cu comanda *DROP VIEW* :
  DROP VIEW *nume_vizualizare;*

- Informaţii despre vizualizări se pot găsi în dicţionarul datelor interogând vizualizările: *USER_VIEWS*, *ALL_VIEWS* . Pentru aflarea informaţiilor despre coloanele actualizabile, este utilă vizualizarea *USER_UPDATABLE_COLUMNS*.

- Operaţii LMD asupra vizualizărilor
  - Vizualizările se pot împărţi în **simple** şi **complexe**. Această clasificare este importantă pentru că asupra vizualizărilor simple se pot realiza operaţii *LMD*, dar în cazul celor complexe acest lucru nu este posibil intotdeauna.
    - **Vizualizările *simple*** sunt definite pe baza unui singur tabel şi **nu conţin funcţii grup** sau **grupări de date**.
    - **Vizualizările *complexe*** sunt definite pe baza mai multor tabele sau conţin funcţii grup sau grupări de date.
  - **Nu se pot realiza operaţii *LMD*** în vizualizări ce conţin:
    - funcţii grup,
    - clauzele *GROUP BY*, *HAVING, START WITH, CONNECT BY*,
    - cuvântul cheie *DISTINCT*,
    - pseudocoloana *ROWNUM*,

- operatori pe mulțimi.

- Nu se pot actualiza**:**

II. coloane ale căror valori rezultă prin calcul sau definite cu ajutorul funcției *DECODE*,

III. coloane care nu respectă constrângerile din tabelele de bază.

- **Pentru vizualizările bazate pe mai multe tabele**, orice operație *INSERT*, *UPDATE* sau *DELETE* poate **modifica datele doar din unul din tabelele de bază**. Acest tabel este cel protejat prin cheie (***key preserved***). În cadrul unei astfel de vizualizări, un tabel de bază se numeşte *key-preserved* dacă are proprietatea că fiecare valoare a cheii sale primare sau a unei coloane având constrângerea de unicitate, este unică şi în vizualizare.

  Prima condiție ca o vizualizare a cărei cerere conține un *join* să fie modificabilă este ca instrucțiunea *LMD* să afecteze un singur tabel din operația de *join*.

- Reactualizarea tabelelor implică reactualizarea corespunzătoare a vizualizărilor!!!

Exerciţii

1. Pe baza tabelului *EMP_PNU*, să se creeze o vizualizare *VIZ_EMP30_PNU*, care conține codul, numele, email-ul şi salariul angajaților din departamentul 30.
   Inseraţi o linie prin intermediul acestei vizualizări; comentaţi.

2. Modificaţi *VIZ_EMP30_PNU* astfel încât să fie posibilă inserarea/modificarea conţinutului tabelului de bază prin intermediul ei. Inserați şi actualizați o linie (cu valoarea 300 pentru codul angajatului) prin intermediul acestei vizualizări.

Obs: Trebuie introduse neapărat în vizualizare coloanele care au constrângerea *NOT NULL* în tabelul de bază (altfel, chiar dacă tipul vizualizării permite operații *LMD*, acestea nu vor fi posibile din cauza nerespectării constrângerilor *NOT NULL*).

3. Să se creeze o vizualizare, *VIZ_EMPSAL50_PNU*, care contine coloanele *cod_angajat, nume, email, functie, data_angajare* si *sal_anual* corespunzătoare angajaților din departamentul 50. Analizați structura şi conținutul vizualizării.

- Inseraţi o linie prin intermediul vizualizării precedente. Comentaţi.
- Care sunt coloanele actualizabile ale acestei vizualizări? Verificați răspunsul în dicționarul datelor (*USER_UPDATABLE_COLUMNS*).

4. Să se creeze vizualizarea *VIZ_DEPT_SUM_PNU*, care conține codul departamentului şi pentru fiecare departament media salariilor. Ce fel de vizualizare se obține (complexa sau simpla)? Se poate actualiza vreo coloană prin intermediul acestei vizualizări?

5. Să se creeze o vizualizare *VIZ_SAL_PNU*, ce conține numele angajaților, numele departamentelor, salariile şi locațiile (oraşele) pentru toți angajații. Etichetați sugestiv coloanele. Considerați ca tabele de bază tabelele originale din schema HR. Care sunt coloanele actualizabe