Table of Contents

Sunt utilizate următoarele tabele:
MODEL(*cod_angajat*#, cod_agentie, inaltime, nr_pantof)
ANGAJAT_TEMP(cod_angajat#, nume, prenume, data_nastere, nationalitate, sex, info, cod_localizare, cod_info_acces, tip)

ORGANIZATOR(cod_organizator#, denumire, banca, cont, cod_info_acces, informatii, cod_localizare)
PREZENTARE(cod_prezentare#, denumire, data_start, data_final, cod_firm_sec, *cod_organizator*, cod_soc_asig, cod_locatie)
SPONSOR(cod_sponsor#, tip, nume, info, cod_localizare, cod_info_acces)
CREATOR(cod_creator#, nume, prenume, data_nastere, data_angajare, tip, mod_angajare, info, cod_casa_moda, cod_localizare, cod_info_acces)
VESTIMENTATIE(cod_vestimentatie#, denumire, valoare, descriere, cod_creator, *cod_model*, *cod_prezentare*)
ACCESORIU(*cod_vestimentatie*#, cod_accesoriu#, *cod_creator*, descriere, tip, valoare)
FINANTEAZA(*cod_sponsor*#, *cod_prezentare*#, suma, banca, cont_emitent, data_emitere, cod_ordin_plata)

Operatorul EXISTS

Operatorul *EXISTS* este proiectat pentru a fi utilizat numai în contextul subcererilor. Acest operator produce un rezultat simplu, de tip boolean. Condiţia obţinută cu ajutorul cuvântului cheie *EXISTS* este *true* dacă există cel puţin o linie returnată de către subcerere, şi este *false* dacă subcererea returnează o mulţime rezultat vidă.

Acest operator este util pentru a testa dacă valoarea recuperată de cererea externă există în mulţimea valorilor regăsite de o cerere internă sincronizată a sa.

Deoarece operatorul *EXISTS* nu face decât să verifice existenţa sau inexistenţa liniilor în rezultatul subcererii, aceasta poate conţine orice număr de coloane. Nefiind necesar ca subcererea să returneze o anumită valoare, se poate selecta o constantă. De altfel, din punct de vedere al performanţei, selectarea unei constante asigură mai mare rapiditate decât selectarea unei coloane.

Un avantaj al operatorului *EXISTS* este dat de faptul că nu mai este continuată căutarea în cererea internă după ce aceasta regăseşte o linie. Ca alternativă a lui *EXISTS*, poate fi utilizat operatorul *IN* într-o cerere nesincronizată.

Exemplu. Să se determine, în două moduri (cu şi fără sincronizare), codul şi denumirea organizatorilor care au coordonat **cel puţin** o prezentare.

```
SELECT   cod_organizator, denumire
FROM     organizator o
WHERE    EXISTS
        (SELECT   1
         FROM     prezentare          WHERE
cod_organizator = o.cod_organizator);
```

```
SELECT   cod_organizator, denumire
FROM     organizator
WHERE    cod_organizator IN (SELECT   cod_organizator
                             FROM     prezentare);
```

Exemplu. Să se determine, în două moduri, codurile modelelor care nu au participat la **nicio** prezentare.

```
SELECT   cod_angajat
FROM     model m
WHERE    NOT EXISTS (SELECT 'x'
                     FROM   vestimentatie
                     WHERE  m.cod_angajat = cod_model);
```

```
SELECT   cod_angajat
FROM     model m
WHERE    cod_angajat NOT IN
                    (SELECT NVL(cod_model, 0)
                     FROM   vestimentatie
                     WHERE  m.cod_angajat = cod_model);
```

Exemplu. Să se obţină numele şi prenumele modelelor care au defilat **cel puţin** în aceleaşi prezentări ca şi modelul având codul 144.

Pentru rezolvarea acestei cereri, trebuie selectate modelele pentru care este vidă lista prezentărilor la care a lucrat modelul 144 mai puţin lista prezentărilor la care au defilat acele modele.

```
SELECT nume, prenume
FROM   vestimentatie v, model m, angajat_temp a
WHERE  v.cod_model = m.cod_angajat
AND    m.cod_angajat = a.cod_angajat
AND    NOT EXISTS
             (SELECT  cod_prezentare
              FROM    vestimentatie
              WHERE   cod_model = 144
              MINUS
              SELECT  cod_prezentare
              FROM    vestimentatie
              WHERE   cod_model = m.cod_angajat);
```

Dacă problema este modificată în sensul că „cel puţin”este înlocuit prin „cel mult” atunci trebuie inversate interogările legate prin MINUS.

***Exemplu*.** Să se obţină numele şi prenumele modelelor care au lucrat **cel mult** la aceleaşi prezentări ca şi modelul având codul 144.

```
SELECT nume, prenume
FROM   vestimentatie v, model m, angajat_temp a
WHERE  v.cod_model = m.cod_angajat
AND    m.cod_angajat = a.cod_angajat
AND    NOT EXISTS
             (SELECT  cod_prezentare
              FROM    vestimentatie
              WHERE   cod_model = m.cod_angajat
              MINUS
              SELECT  cod_prezentare
              FROM    vestimentatie
              WHERE   cod_model = 144);
```

***Exemplu*.** Să se obţină modelele care au lucrat **la aceleaşi** prezentări **ca şi** modelul având codul 144.

Rezolvarea problemei se bazează pe ideea: $A = B \Leftrightarrow A \subset B$ şi $B \subset A \Leftrightarrow (A-B) = \varnothing$ şi $(B-A) = \varnothing \Leftrightarrow$ A-B şi B-A nu furnizează nici un tuplu rezultat.

```
SELECT nume, prenume
FROM   vestimentatie v, model m, angajat_temp a
WHERE  v.cod_model = m.cod_angajat
AND    m.cod_angajat = a.cod_angajat AND
NOT EXISTS
             (SELECT  cod_prezentare
              FROM    vestimentatie
              WHERE   cod_model = 144
              MINUS
              SELECT  cod_prezentare
              FROM    vestimentatie
              WHERE   cod_model = m.cod_angajat)
AND     NOT EXISTS
             (SELECT  cod_prezentare
              FROM    vestimentatie
              WHERE   cod_model = m.cod_angajat
              MINUS
              SELECT  cod_prezentare
              FROM    vestimentatie
              WHERE   cod_model = 144)
AND     cod_model != 144;
```

Operatorul *NOT EXISTS* este util în implementarea operaţiei de diviziune din algebra relaţională. (exemple pe schema HR la http://193.226.51.37/down/sql/Laborator6_SQL_an2.pdf)

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* al algebrei relaţionale 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*.

O altă modalitate de implementare a operatorului *DIVISION* este cu ajutorul funcţiei *COUNT*.

Exemplu. Să se obţină codurile sponsorilor care au finanţat **toate** prezentările care au început în anul 2008. Se vor furniza două soluţii.

```
SELECT DISTINCT cod_sponsor
FROM   finanteaza f
WHERE  NOT EXISTS
      (SELECT  1
       FROM    prezentare p
       WHERE   TO_CHAR(data_start, 'YYYY') = 2008
       AND NOT EXISTS
           (SELECT 'x'
            FROM   finanteaza fi
            WHERE  p.cod_prezentare = fi.cod_prezentare
            AND    fi.cod_sponsor = f.cod_sponsor));
SELECT   cod_sponsor
FROM     finanteaza
WHERE    cod_prezentare IN
          (SELECT   cod_prezentare
           FROM     prezentare            WHERE
TO_CHAR(data_start, 'YYYY') = 2008)
GROUP BY cod_sponsor
HAVING   COUNT(cod_prezentare)=
          (SELECT  COUNT(*)
           FROM    prezentare
           WHERE   TO_CHAR(data_start, 'YYYY') = 2008);
```

Definirea blocurilor de cerere în clauza WITH

Atunci când o interogare face referinţă de mai multe ori la o aceeaşi subcerere, este utilă definirea blocului de cerere corespunzător înainte de a fi utilizat în interogarea respectivă. Acest lucru este posibil prin intermediul clauzei *WITH* şi este deosebit de util atunci când cererea astfel definită este complexă, conţinând operaţii de compunere şi funcţii agregat.

Clauza permite reutilizarea aceluiaşi bloc de cerere într-o instrucţiune *SELECT* complexă. Serverul *Oracle* regăseşte rezultatele unui bloc de cerere definit în clauza *WITH* şi le stochează în spaţiul tabel temporar al utilizatorului, fapt ce poate determina îmbunătăţirea performanţelor.

Intern, clauza *WITH* este tratată ca o vizualizare *inline* (subcerere în clauza *FROM*) sau ca un tabel temporar. Optimizorul alege decizia adecvată pe baza costului sau beneficiului stocării temporare a rezultatelor clauzei *WITH*.

Clauza *WITH* poate fi folosită numai pentru instrucţiuni *SELECT*. Un nume de bloc de cerere este vizibil tuturor cererilor din clauza *WITH* definite ulterior (inclusiv subcererilor acestora). De asemenea, un nume de bloc de cerere este vizibil cererii principale şi subcererilor acesteia. Atunci când un nume de cerere coincide cu numele unui tabel, numele de cerere are precedenţă asupra numelui tabelului. În clauza *WITH* se pot defini mai mult decât o singură cerere. În acest caz, cererile sunt separate prin virgule.

Exemplu. Să se obţină numele creatorilor de modă şi valoarea totală a obiectelor realizate de către aceştia. Se vor considera atât creatorii de vestimentaţii cât şi cei de accesorii, a căror valoare totală a obiectelor este mai mare decât media valorilor totale ale obiectelor tuturor creatorilor. Interogarea va utiliza cereri definite în clauza *WITH*.

```
WITH
val_creator_vestim AS
                (SELECT   nume, prenume,
                          SUM(valoare) AS total
                 FROM     vestimentatie v, creator c
                 WHERE    v.cod_creator = c.cod_creator
                 GROUP BY nume, prenume),
val_creator_accesorii AS
                (SELECT   nume, prenume,
                          SUM(valoare) AS total
                 FROM     accesoriu a, creator c
                 WHERE    a.cod_creator = c.cod_creator
                 GROUP BY nume, prenume),
val_medie_vestim  AS
                (SELECT   SUM(total)/COUNT(*) AS medie
                 FROM     val_creator_vestim)
val_medie_accesorii  AS
                (SELECT   SUM(total)/COUNT(*) AS medie
                 FROM     val_creator_accesorii)
```

```
SELECT   *
FROM     val_creator_vestim
WHERE    total > (SELECT   medie
                  FROM     val_medie_vestim)
UNION
SELECT   *
FROM     val_creator_accesorii
WHERE    total > (SELECT   medie
                  FROM     val_medie_accesorii)
ORDER BY nume;
```