

Capitolul 9. Subconsultări

Una dintre cele mai importante facilități ale interogărilor SQL constă în folosirea rezultatului unei consultări ca argument într-o clauză WHERE, HAVING, FROM sau SELECT a unei alte consultări. Se poate ajunge la o structură destul de complicată de interogări și subinterogări, dar și rezultatele sunt pe măsură. Ar mai trebui spus că titlul acestui capitol putea fi *subconsultări ne-corelate*, deoarece, după cum vom vedea în capitolul următor, o categorie aparte de subconsultări sunt cele simplu sau dublu corelate.

9.1. Subconsultări în clauza WHERE. Operatorul IN

Operatorul cel mai utilizat în materie de subconsultări este IN pe care l-am întâlnit deja într-un paragraful 5.5 într-o cu totul altă ipostază - testarea încadrării valorii unui atribut într-o listă de constante. Pentru cele ce urmează, domeniul de acțiune al operatorului va fi o tabelă (ad-hoc) obținută printr-o (sub)consultare.

9.1.1. Subconsultări în (sub)consultări în (sub)consultări în...

Revenim (iarăși) la exemplul 19 din algebra relațională (paragraful 4.4.4, figurile 4.24 și 4.25): *Ce facturi au fost emise în aceeași zi cu factura 1120 ?* Anterior a fost formulată o soluție bazată pe joncțiunea a două instanțe ale tabelului FACTURI. Iată însă o soluție mai simplă bazată pe subconsultări:

```
SELECT *  
FROM facturi  
WHERE DataFact IN  
    (SELECT DataFact  
     FROM facturi  
     WHERE NrFact=1120)
```

Execuția acestei interogări se derulează în doi timp. Mai întâi, se execută subconsultarea *SELECT DataFact FROM FACTURI WHERE NrFact=1120* obținându-se o tabelă intermediară cu o singură linie și o singură coloană (DataFact) - vezi partea stângă a figurii 9.1. În al doilea pas sunt selectate liniile tabelului FACTURI pentru care valoarea atributului DataFact este 7 august 2007.

| NRFACT | DATAFACT | CODCL | OBS |
|--------|------------|-------|--------|
| 1119 | 07-08-2007 | 1003 | (null) |
| 1120 | 07-08-2007 | 1001 | (null) |
| 1121 | 07-08-2007 | 1004 | (null) |
| 1122 | 07-08-2007 | 1005 | (null) |

Figura 9.1. Rezultatul subconsultării (stânga) și cel final

În rezultat a fost inclusă și factura de referință – 1120. Dacă se dorește excluderea acesteia, fraza SELECT se modifică astfel:

```
SELECT *
FROM facturi
WHERE DataFact IN
      (SELECT DataFact
       FROM facturi
       WHERE NrFact=1120)
AND NrFact <> 1120
```

Ce facturi au fost emise în alte zile decât cea a facturii 1120 ?

Acest exemplu necesită folosirea operatorului de negație – NOT IN:

```
SELECT NrFact
FROM facturi
WHERE DataFact NOT IN
      (SELECT DataFact
       FROM facturi
       WHERE NrFact=1120)
```

Care sunt clienții cărora li s-au trimis facturi în aceeași zi în care a fost întocmită factura 1120 ?

```
SELECT DenCl
FROM clienti
WHERE CodCl IN
      (SELECT CodCl
       FROM facturi
       WHERE DataFact IN
            (SELECT DataFact
             FROM facturi
             WHERE NrFact=1120)
      )
```

Figura 9.2 ilustrează modul ierarhic de execuție a interogării cu trei niveluri de consultare (fraza principală, o sub-consultare și o sub-sub-consultare).

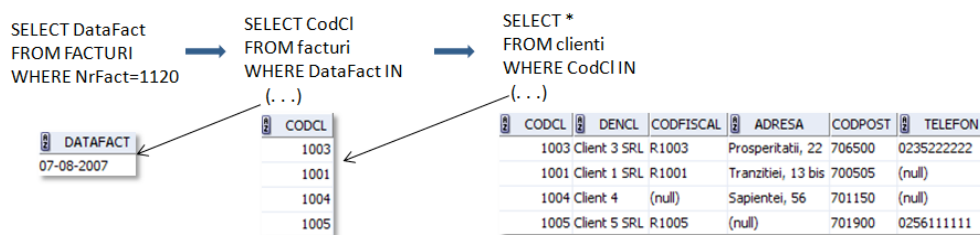


Figura 9.2. Mecanismul de execuție al interogării cu două niveluri de subconsultare

La fel de corectă este și varianta care folosește o joncțiune în locul unui nivel de subconsultare:

```
SELECT clienti.*
FROM clienti INNER JOIN fatturi      ON clienti.CodCl=fatturi.CodCl
WHERE DataFact IN
      (SELECT DataFact
       FROM fatturi
       WHERE NrFact=1120)
```

În ce județe s-a vândut produsul "Produs 2" ?

Am ales acest exemplu pentru a "vântura", prin subconsultări, cât mai multe
tabele ale bazei:

```

SELECT Judet
FROM judete
WHERE Jud IN
    (SELECT Jud
     FROM coduri_postale
     WHERE CodPost IN
         (SELECT CodPost
          FROM clienti
          WHERE CodCl IN
              (SELECT CodCl
               FROM facturi
               WHERE NrFact IN
                   (SELECT NrFact
                    FROM liniifact
                    WHERE CodPr IN
                        (SELECT CodPr
                         FROM produse
                         WHERE DenPr = 'Produs 2'
                        )
                   )
              )
          )
         )
    )

```

Revenim la tabela PERSONAL2 din figura 8.4: *Câți subordonați direcți are ANGAJAT 2* ? La această problemă (la care răspunsul este 2) formulăm, pentru comparație, două soluții. Soluția bazată pe joncțiune este:

```
SELECT COUNT(*) AS NrSubordonati
FROM personal2 SUBORDONATI INNER JOIN personal2 SEFI
ON SUBORDONATI.MarcaSef=SEFI.Marca
```

WHERE SEFI.NumePren='ANGAJAT 2'

Deși trebuia să facem lucrul acesta mai demult, întrucât nu este o chestiune legată de subconsultări, să vedem cum se prezintă execuția acestei interogări. Până acum, am mai joncționat două instanțe ale unei tabele după un câmp comun. Noutatea acestei probleme ține de faptul că tabela PERSONAL2, prin cele două atribute, Marca (cheia primară) și MarcaSef (cheie străină) conține structura ierarhică a firmei. Obținem numele șefului direct al fiecărui angajat astfel:

```
SELECT SUBORDONATI.*,  
SEFI.Marca AS "Sefi.Marca",  
SEFI.NumePren AS "Sefi.NumePren",  
SEFI.SalTarifar AS "Sefi.SalTarifar"  
FROM personal2 SUBORDONATI INNER JOIN personal2 SEFI  
ON SUBORDONATI.MarcaSef=SEFI.Marca
```

Din rezultat (figura 9.3) lipsește angajatul cu marca 1 deoarece el este singurul fără superiori ierarhici. Pe fiecare linie a rezultatului se află, deci, un angajat împreună cu șeful său direct.

| 1 | MARCA | 2 | NUMEPREN | 3 | DATANAST | 4 | COMPART | 5 | MARCASEF | 6 | SALTARIFAR | 7 | Sefi.Marca | 8 | Sefi.NumePren | 9 | Sefi.SalTarifar |
|---|-------|------------|----------|------------|----------|---------------|---------|---|----------|------|------------|---|------------|---|---------------|---|-----------------|
| | 3 | ANGAJAT 3 | | 02-08-1962 | | MARKETING | | 1 | | 1450 | | 1 | ANGAJAT 1 | | | | 1600 |
| | 10 | ANGAJAT 10 | | 29-01-1972 | | RESURSE UMANE | | 1 | | 1370 | | 1 | ANGAJAT 1 | | | | 1600 |
| | 2 | ANGAJAT 2 | | 11-10-1977 | | FINANCIAR | | 1 | | 1450 | | 1 | ANGAJAT 1 | | | | 1600 |
| | 4 | ANGAJAT 4 | | (null) | | FINANCIAR | | 2 | | 1380 | | 2 | ANGAJAT 2 | | | | 1450 |
| | 5 | ANGAJAT 5 | | 30-04-1965 | | FINANCIAR | | 2 | | 1420 | | 2 | ANGAJAT 2 | | | | 1450 |
| | 9 | ANGAJAT 9 | | 28-02-1976 | | MARKETING | | 3 | | 1410 | | 3 | ANGAJAT 3 | | | | 1450 |
| | 8 | ANGAJAT 8 | | 31-12-1960 | | MARKETING | | 3 | | 1290 | | 3 | ANGAJAT 3 | | | | 1450 |
| | 7 | ANGAJAT 7 | | (null) | | FINANCIAR | | 5 | | 1280 | | 5 | ANGAJAT 5 | | | | 1420 |
| | 6 | ANGAJAT 6 | | 09-11-1965 | | FINANCIAR | | 5 | | 1350 | | 5 | ANGAJAT 5 | | | | 1420 |

Figura 9.3. Șefii direcți ai fiecărui angajat

A doua soluție utilizează o subconsultare:

```
SELECT COUNT(Marca) AS NrSubordonati  
FROM personal2  
WHERE MarcaSef IN  
(SELECT Marca  
FROM personal2  
WHERE NumePren='ANGAJAT 2')
```

În paragraful 4.4.6 am prezentat, foarte pe scurt, un tip de joncțiune destul de discret - semijoncțiunea - prin care se extrag numai liniile dintr-o tabelă ce au corespondent (ca valoare a atributului de legătură) în a doua tabelă. Cu operatorul IN se pot formula lejer soluții ce corespund semijoncțiunii. Astfel, o tabelă precum cea din figura 4.29 se obține prin:

```
SELECT * FROM r1  
WHERE C IN (SELECT C FROM r2)
```

Completăm discuția și cu exemplul “practic” luat pentru ilustrarea semijoncțiunii (exemplul 22 din paragraful 4.4.6): *Care sunt localitățile (codul poștal, denumirea și indicativul județului) în care există măcar un client ?*

```
SELECT *
FROM coduri_postale
WHERE CodPost IN
      (SELECT CodPost
       FROM clienti)
```

Tot prin subconsultări putem realiza intersecția și diferența relațională. Raportându-ne la intersecția a două relații, R1 și R2, operațiunea se poate realiza în SQL și astfel:

```
SELECT *
FROM r1
WHERE (A,B,C) IN
      (SELECT C,D,E
       FROM r2)
```

Spre deosebire de interogările de până acum din acest capitol, care funcționează în toate cele patru servere BD, această variantă nu funcționează în SQL Server (dealtminteri, nici în Visual FoxPro sau Access), deoarece într-o subconsultare nu pot fi testate simultan trei valori (ale atributelor A, B și C), ci numai una. Astfel încât, pentru a-i înșela vigilența, se vor concatena cele trei atribute, dând „senzația” unuia singur:

```
SELECT *
FROM r1
WHERE CAST(A AS VARCHAR) + B + CAST (C AS VARCHAR) IN
      (SELECT CAST (C AS VARCHAR) + D + CAST (E AS VARCHAR)
       FROM r2)
```

Exemplul 17 din paragraful 4.4.4: *În ce zile s-au vândut și produsul cu denumirea “Produs 1” și cel cu denumirea “Produs 2” ?*

```
SELECT DISTINCT DataFact
FROM produse
      INNER JOIN liniifact ON produse.CodPr=liniifact.CodPr
      INNER JOIN facturi ON liniifact.NrFact=facturi.NrFact
WHERE DenPr = 'Produs 1' AND DataFact IN
      (SELECT DataFact
       FROM produse
              INNER JOIN liniifact ON produse.CodPr=liniifact.CodPr
              INNER JOIN facturi ON liniifact.NrFact=facturi.NrFact
       WHERE DenPr = 'Produs 2')
```

Și diferența relațională poate fi realizată cu ajutorul subconsultărilor – operatorul NOT IN. Diferența relațiilor R1 și R2 din paragraful 4.3.3 (figura 4.5) este rezultatul SELECT-ului:

```
SELECT *
FROM r1
WHERE (A,B,C) NOT IN
      (SELECT C,D,E
       FROM r2)
```

Firește, în SGBD-urile în care „subiect” al subconsultărilor nu poate fi un tuplu ad-hoc (două sau mai multe atribute), cum este cazul SQL Server-ului, vom aplica “trucul” concatenării, ca și la intersecție.

Apelăm iarăși la un exemplu din algebra relațională (exemplu 18 din paragraful 4.4.4): *Ce clienți au cumpărat și “Produs 2” și “Produs 3”, dar nu au cumpărat “Produs 5” ?*

```
SELECT DISTINCT DenCl
FROM produse
  INNER JOIN liniifact ON produse.CodPr=liniifact.CodPr
  INNER JOIN facturi ON liniifact.NrFact=facturi.NrFact
  INNER JOIN clienti ON facturi.CodCl=clienti.CodCl
WHERE DenPr = 'Produs 2' AND facturi.CodCl IN
      (SELECT CodCl
       FROM produse
         INNER JOIN liniifact ON produse.CodPr=liniifact.CodPr
         INNER JOIN facturi ON liniifact.NrFact=facturi.NrFact
        WHERE DenPr = 'Produs 3' AND facturi.CodCl NOT IN
              (SELECT CodCl
               FROM produse
                 INNER JOIN liniifact
                   ON produse.CodPr=liniifact.CodPr
                 INNER JOIN facturi
                   ON liniifact.NrFact=facturi.NrFact
                WHERE DenPr = 'Produs 5'
              )
        )
```

Până la apariția operatorilor LEFT OUTER JOIN, RIGHT OUTER JOIN și FULL OUTER JOIN în multe SGBD-uri joncțiunea externă era realizată prin reuniunea liniilor obținute din echi-joncțiune cu liniile unei tabele (completate cu zerouri/spații pentru atributele celeilalte tabele) ce nu au corespondent în cealaltă. Iată o interogare ce joncțiunează extern la stânga relațiile R1 și R2 prin atributul C:

```
SELECT *
FROM r1 INNER JOIN r2 ON r1.C = r2.C
```

```

UNION
SELECT A,B,C, NULL, NULL, NULL
FROM r1
WHERE C NOT IN
      (SELECT C FROM r2)

```

Dacă rândurile din rezultat sunt identice cu cele obținute prin subconsultare, în Oracle titulatura coloanelor este cel puțin interesantă la execuția acestui SELECT – vezi figura 9.4

| A | B | QCSJ_C000000000300000 | QCSJ_C000000000300001 | D | E |
|--------|---|-----------------------|-----------------------|--------|----|
| 20 XYZ | | 30 | 30 XXZ | | 40 |
| 30 XXZ | | 20 | (null) (null) | (null) | |
| 40 YYX | | 25 | 25 XYZ | | 30 |

Figura 9.4. Surprize, surprize în Oracle (alt episod)

În DB2 nu putem introduce direct NULL în lista coloanelor celui de-al doilea SELECT, ci vom apela la funcția CAST. În plus, pentru a asigura o titulatură dorită ar fi trebui să folosim clauza AS pentru fiecare coloană din ambele SELECT-uri.

```

SELECT *
FROM r1 INNER JOIN r2 ON r1.C = r2.C
UNION
SELECT A,B,C, CAST( NULL AS NUMERIC), CAST(NULL AS CHAR),
      CAST( NULL AS NUMERIC)
FROM r1
WHERE C NOT IN
      (SELECT C
      FROM r2)

```

Revenim (deși o să regretăm) la câteva exemple prezentate la joncțiunea externă pe care le rezolvăm prin noua “rețetă” – reuniune/valori vide.

Care sunt valorile facturate și încasate ale fiecărei facturi ?

Ultima variantă de rezolvare este cea din partea finală a paragrafului 8.5. Soluția de mai jos (redactată în sintaxa Oracle/PostgreSQL) reunește facturile care au măcar o tranșă de încasare cu cele neîncasate deloc:

```

SELECT lf.NrFact,
      TRUNC(
        SUM(Cantitate * PretUnit * (1+ProcTVA)) /
        COUNT(DISTINCT COALESCE(i.CodInc,0))
      ,0) AS "Facturat",
      TRUNC (
        SUM(COALESCE(Transa,0)) / MAX(lf.Linie)
      ,0) AS "Incasat",

```

```

        TRUNC(
            SUM(Cantitate * PretUnit * (1+ProcTVA)) /
            COUNT(DISTINCT COALESCE(i.CodInc,0))
        -
        SUM(COALESCE(Transa,0)) / MAX(lf.Linie)
        ,0) AS "Diferenta",
CASE
    WHEN SUM(Cantitate * PretUnit * (1+ProcTVA)) /
        COUNT(DISTINCT COALESCE(i.CodInc,0))
        >
        SUM(COALESCE(Transa,0)) / MAX(lf.Linie)
    THEN ' Incasata partial'
    ELSE ' *Incasata total*'
END AS "Situatiune"
FROM facturi f
    INNER JOIN liniifact lf ON f.NrFact = lf.NrFact
    INNER JOIN produse p ON lf.CodPr=p.CodPr
    INNER JOIN incasfact i ON lf.NrFact=i.NrFact
WHERE EXTRACT (YEAR FROM DataFact)=2007
    AND EXTRACT (MONTH FROM DataFact)=8
GROUP BY lf.NrFact
    UNION
SELECT lf.NrFact,
    TRUNC( SUM(Cantitate * PretUnit * (1+ProcTVA))),
    0,
    TRUNC( SUM(Cantitate * PretUnit * (1+ProcTVA))),
    ' Fara nici o incasare'
FROM facturi f
    INNER JOIN liniifact lf ON f.NrFact = lf.NrFact
    INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE EXTRACT (YEAR FROM DataFact)=2007

```



```

        AND EXTRACT (MONTH FROM DataFact)=8
        AND F.NrFact NOT IN
            (SELECT NrFact
             FROM incasfact)
GROUP BY lf.NrFact
ORDER BY 11

```

Să se obțină sporurile de noapte pentru al doilea trimestru al anului 2007, atât lunar, cât și cumulat.

Trebuie reunite persoanele care au sporul de noapte pe toate cele trei luni, cu persoanele care prezintă sporul numai pe câte două luni, cu persoanele cu sporul pe numai o singură lună, și cu persoanele cărora nu li s-a calculat spor de noapte pe niciuna dintre cele trei luni.

Dacă la momentul formulării soluției anterioare afirmam că fraza SELECT este supraponderală, prezenta soluție este pur și simplu pantagruelică. Cred că interogarea următoare demonstrează fără dubii că joncțiunea externă este o găselniță grozav de inteligentă și, pe de altă parte, că scrierea frazelor SQL poate deveni, pe alocuri, o treabă înfricoșătoare, dăunând, împreună cu excesul de sare, zahăr și alcool, grav sănătății:

– primul SELECT este al celor ce au sporuri de noapte

– pe toate cele trei luni - 4,5 și 6

```

SELECT personal2.Marca, NumePren,
       s1.SporNoapte AS Spor_Noapte_Aprilie,
       s2.SporNoapte AS Spor_Noapte_Mai,
       s3.SporNoapte AS Spor_Noapte_Iunie,
       s1.SporNoapte + s2.SporNoapte + s3.SporNoapte
       AS Spor_Noapte_Trim_II
FROM personal2
      INNER JOIN sporuri s1 ON personal2.Marca=s1.Marca
      AND s1.An=2007 AND 4=s1.Luna

```

¹ Sintaxa DB2 și SQL Server este diferită în privința funcției TRUNC și necesită substituirea funcției EXTRACT prin YEAR și MONTH.

```

        INNER JOIN sporuri s2 ON personal2.Marca=s2.Marca
        AND s2.An=2007 AND 5=s2.Luna
        INNER JOIN sporuri s3 ON personal2.Marca=s3.Marca
        AND s3.An=2007 AND 6=s3.Luna
    UNION
-- acest SELECT extrage pe cei care au avut spor de noapte
--      numai pe lunile 4 si 5
SELECT personal2.Marca, NumePren,
        s1.SporNoapte,
        s2.SporNoapte,
        0 ,
        s1.SporNoapte + s2.SporNoapte
FROM personal2
        INNER JOIN sporuri s1 ON personal2.Marca=s1.Marca
        AND s1.An=2007 AND 4=s1.Luna
        INNER JOIN sporuri s2 ON personal2.Marca=s2.Marca
        AND s2.An=2007 AND 5=s2.Luna
WHERE personal2.Marca NOT IN
        (SELECT Marca FROM sporuri WHERE An=2007 AND Luna=6)
    UNION
-- acest SELECT extrage pe cei care au avut spor de noapte
--      numai pe lunile 4 si 6
SELECT personal2.Marca, NumePren,
        s1.SporNoapte,
        0,
        s3.SporNoapte,
        s1.SporNoapte + s3.SporNoapte
FROM personal2
        INNER JOIN sporuri s1 ON personal2.Marca=s1.Marca
        AND s1.An=2007 AND 4=s1.Luna
        INNER JOIN sporuri s3 ON personal2.Marca=s3.Marca
        AND s3.An=2007 AND 6=s3.Luna
WHERE personal2.Marca NOT IN
        (SELECT Marca FROM sporuri WHERE An=2007 AND Luna=5)
    UNION
-- acest SELECT extrage pe cei care au avut spor de noapte
--      numai pe lunile 5 si 6
SELECT personal2.Marca, NumePren,
        0,
        s2.SporNoapte,

```

```
s3.SporNoapte,
s2.SporNoapte + s3.SporNoapte
FROM personal2
    INNER JOIN sporuri s2 ON personal2.Marca=s2.Marca
        AND s2.An=2007 AND 5=s2.Luna
    INNER JOIN sporuri s3 ON personal2.Marca=s3.Marca
        AND s3.An=2007 AND 6=s3.Luna
WHERE personal2.Marca NOT IN
    (SELECT Marca FROM sporuri WHERE An=2007 AND Luna=5)

UNION

-- acest SELECT extrage pe cei care au avut spor de noapte numai pe luna 4
SELECT personal2.Marca, NumePren,
    s1.SporNoapte,
    0,
    0,
    s1.SporNoapte
FROM personal2 INNER JOIN sporuri s1 ON personal2.Marca=s1.Marca
    AND s1.An=2007 AND 4=s1.Luna
WHERE personal2.Marca NOT IN
    (SELECT Marca
        FROM sporuri
        WHERE An=2007 AND Luna=5)
    AND personal2.Marca NOT IN
        (SELECT Marca FROM sporuri
            WHERE An=2007 AND Luna=6)

UNION

-- acest SELECT extrage pe cei care au avut spor de noapte numai pe luna 5
SELECT personal2.Marca, NumePren,
    0,
    s2.SporNoapte,
    0,
    s2.SporNoapte
FROM personal2 INNER JOIN sporuri s2 ON personal2.Marca=s2.Marca
    AND s2.An=2007 AND 5=s2.Luna
WHERE personal2.Marca NOT IN
    (SELECT Marca FROM sporuri WHERE An=2007 AND Luna=4)
    AND personal2.Marca NOT IN
        (SELECT Marca FROM sporuri
            WHERE An=2007 AND Luna=6)

UNION
```

```

-- acest SELECT extrage pe cei care au avut spor de noapte numai pe luna 6
SELECT personal2.Marca, NumePren,
       0,
       0,
       s3.SporNoapte,
       s3.SporNoapte
FROM personal2 INNER JOIN sporuri s3 ON personal2.Marca=s3.Marca
AND s3.An=2007 AND 6=s3.Luna
WHERE personal2.Marca NOT IN
      (SELECT Marca FROM sporuri
       WHERE An=2007 AND Luna=4)
AND personal2.Marca NOT IN
      (SELECT Marca FROM sporuri
       WHERE An=2007 AND Luna=5)

UNION

-- acest SELECT extrage pe cei care nu au avut spor de noapte pe nici o luna
SELECT personal2.Marca, NumePren,
       0,
       0,
       0,
       0
FROM personal2
WHERE personal2.Marca NOT IN
      (SELECT Marca FROM sporuri
       WHERE An=2007 AND Luna=4)
AND personal2.Marca NOT IN
      (SELECT Marca FROM sporuri
       WHERE An=2007 AND Luna=5)
AND personal2.Marca NOT IN
      (SELECT Marca FROM sporuri
       WHERE An=2007 AND Luna=6)

ORDER BY NumePren

```

O mângâiere (deși e mai mult masochism decât mângâiere) este că am scris cea mai lungă frază SELECT de până acum... Plus că sintaxa este comună celor patru servere BD.

9.1.2. Diviziunea prin subconsultări în clauza WHERE

Tot cu ajutorul operatorului IN (și NOT IN) se poate aborda și "problema" diviziunii relaționale în SQL (parcă vă aud spunând: "Asta ne mai lipsea acum

!"). Succesiunea pașilor ilustrată în algebra relațională în figura 4.33 (paragraful 4.4.7) se realizează în SQL (dialectele DB2/PostgreSQL) astfel :

```
SELECT X
FROM rd1
EXCEPT
SELECT DISTINCT rd1.X FROM rd1 CROSS JOIN rd2
WHERE (rd1.X, rd2.Y) NOT IN
      (SELECT X, Y FROM rd1)
```

În Oracle se înlocuiește operatorul EXCEPT cu MINUS, iar SQL Server nu acceptă tupluri ad-hoc ca argumente ale operatorului IN, așa că recurgem la soluția concatenării.

```
SELECT DISTINCT X
FROM rd1
WHERE X NOT IN
      (
        SELECT DISTINCT rd1.X
        FROM rd1, rd2
        WHERE rd1.X + rd2.Y NOT IN
              (SELECT X + Y
               FROM rd1)
      )
```

sau

```
SELECT DISTINCT X
FROM rd1
WHERE X NOT IN (
      SELECT R1_1.X
      FROM (rd1 R1_1 INNER JOIN rd2 R2_1 ON 1=1)
            LEFT OUTER JOIN rd1 R1_2 ON R1_1.X=R1_2.X
            AND R2_1.Y=R1_2.Y
      WHERE R1_2.Y IS NULL
    )
```

Această ultimă comandă poate fi lansată în oricare dintre cele patru dialecte. Pseudo-joncțiunea internă este de fapt un produs cartezian, deoarece condiția de joncțiune este $1=1$ ². Subconsultarea extrage icșii care au “goluri”, iar fraza SELECT principală efectuează scăderea lor din icșii tablei RD1. Ultima interogare funcționează pe toate cele patru servere de baze de date.

Care sunt facturile ce conțin măcar produsele din factura 1117? - exemplul 27 din paragraful 4.4.7. Soluția DB2/Oracle/PostgreSQL este:

```
SELECT NrFact
FROM liniifact
EXCEPT3
SELECT DISTINCT rd1.NrFact
FROM liniifact rd1, liniifact rd2
WHERE rd2.NrFact=1117 AND (rd1.NrFact, rd2.CodPr) NOT IN
(SELECT NrFact, CodPr
FROM liniifact)
```

iar cea SQL Server:

```
SELECT NrFact
FROM liniifact
EXCEPT
SELECT DISTINCT rd1.NrFact
FROM liniifact rd1, liniifact rd2
WHERE rd2.NrFact=1117 AND
CAST (rd1.NrFact AS CHAR(8)) + CAST (rd2.CodPr AS CHAR(6))
NOT IN
(SELECT CAST (NrFact AS CHAR(8)) + CAST (CodPr AS CHAR(6))
FROM liniifact)
```

Care sunt clienții pentru care există cel puțin câte o factură emisă în fiecare zi cu vânzări din perioada 10-30 septembrie 2007?

² Era mai elegant să folosim *CROSS JOIN* în loc de ... *INNER JOIN*... *ON 1=1*.

³ MINUS (în loc de EXCEPT) în Oracle

Acesta este exemplul 23 din paragraful 4.4.7 (figura 4.31) pentru ilustrarea operatorului diviziune. Urmăm logica pe care tocmai am prezentat-o în Oracle/PostgreSQL și DB2 (dacă scoatem cuvintele DATE):

```

SELECT DenCl
FROM clienti
WHERE CodCl NOT IN
    (SELECT c.CodCl
     FROM clienti c CROSS JOIN facturi f
     WHERE DataFact BETWEEN DATE'2007-09-10' AND DATE'2007-09-30'
     AND (c.codcl, f.DataFact) NOT IN
        (
            SELECT CodCl, DataFact
            FROM facturi f
            WHERE DataFact BETWEEN DATE'2007-09-10'
            AND DATE'2007-09-30'
        )
    )

```

Rămâne ca temă pentru acasă elaborarea variantei funcționabile în MS SQL Server.

Ce produse au fost vândute tuturor clienților ?

Acesta este exemplul 26 din paragraful 4.4.7 pentru ilustrarea folosirii operatorului diviziune. După scrutarea tabelor PRODUSE, LINIIFACT, FACTURI și CLIENȚI este evident că numai Produs 2 a fost vândut tuturor clienților. Cu toate acestea, urmând logica funcțională în DB2, PostgreSQL și Oracle (înlocuind EXCEPT cu MINUS) pe care tocmai am prezentat-o:

```

SELECT DenPr
FROM produse
EXCEPT
    SELECT DISTINCT p1.DenPr
    FROM (produse p1
         INNER JOIN liniifact lf1 ON p1.CodPr=lf1.CodPr
         INNER JOIN facturi f1 ON lf1.NrFact=f1.NrFact)
         CROSS JOIN clienti rd2
    WHERE (p1.DenPr, rd2.CodCl) NOT IN
    (SELECT DISTINCT DenPr, CodCl
     FROM produse
     INNER JOIN liniifact ON produse.CodPr=liniifact.CodPr
     INNER JOIN facturi ON liniifact.NrFact=facturi.NrFact
    )

```

rezultatul conține două produse – vezi figura 9.5.

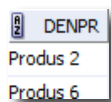


Figura 9.5. Rezultat eronat al diviziunii (ex.26 din paragraful 4.4.7)

Intrusul este produsul 6. Acesta nu apare în nicio factură. De aceea, la joncțiunea cu LINIIFACT și FACTURI, nu va fi inclus în produsul cartezian și, astfel, nu va fi identificat ca lipsind din combinații cu toți clienții (CodCl) – în fapt, el nu este în combinație cu nici un client. Nefiind identificat ca lipsă în combinație cu toți clienții, va fi considerat „prezent”.

Pentru a remedia situația, vom forța includerea în produsul cartezian a tuturor produselor, așa că vom apela la joncțiunea externă. În plus, clauza WHERE, care testează existența tuplurilor din produsul cartezian în „relația” ad-hoc cu atributele (DenPr, CodCl), va face apel la structura CASE:

```
SELECT DenPr
FROM produse
EXCEPT
    SELECT DISTINCT p1.DenPr
    FROM (produse p1 LEFT OUTER JOIN (
        liniifact lf1 INNER JOIN facturi f1 ON lf1.NrFact=f1.NrFact
        ) ON p1.CodPr=lf1.CodPr)
    CROSS JOIN clienti rd2
    WHERE (p1.DenPr,
        CASE WHEN lf1.CodPr IS NULL THEN 0
        ELSE rd2.CodCl
        END) NOT IN
        (SELECT DISTINCT DenPr, CodCl
        FROM produse
        INNER JOIN liniifact ON
        produse.CodPr=liniifact.CodPr
        INNER JOIN facturi ON liniifact.NrFact=facturi.NrFact
        )
    )
```

Iată și varianta SQL Server:

```
SELECT DenPr
FROM produse
EXCEPT
    SELECT DISTINCT p1.DenPr
    FROM (produse p1 LEFT OUTER JOIN (
        liniifact lf1 INNER JOIN facturi f1 ON lf1.NrFact=f1.NrFact
        ) ON p1.CodPr=lf1.CodPr)
    CROSS JOIN clienti rd2
```



```

WHERE (p1.DenPr + CAST( CASE WHEN lf1.CodPr IS NULL THEN 0
                        ELSE rd2.CodCl END AS CHAR(6))) NOT IN
      (SELECT DISTINCT DenPr + CAST (CodCl AS CHAR(6))
       FROM produse
        INNER JOIN liniifact ON produse.CodPr=liniifact.CodPr
        INNER JOIN facturi ON liniifact.NrFact=facturi.NrFact
       )

```

9.1.3. Subconsultări și NULLități

Am amenințat în câteva rânduri că prezența valorilor NULL în liste-argument ale operatorului IN poate crea probleme. Să luăm un exemplu. Dacă până acum ne-au interesat facturile emise în aceeași zi cu factura 1120, acum ne frământă ideea de a afla *facturile cu aceleași observații ca ale facturii 1120*. După cum bănuiați, interogarea:

```

SELECT * FROM facturi
WHERE obs IN (SELECT obs FROM facturi WHERE NrFact = 1120)

```

nu conține nici un rând, întrucât valoarea atributului Obs pentru factura-etalon este NULL. De aceea, trebuie reținut, printre sfaturile părintești (pre și post-maritale) din SQL că, la folosirea subconsultărilor care ar extrage valori NULL, să facem conversia folosind funcții COALESCE/NVL/VALUE:

```

SELECT * FROM facturi
WHERE COALESCE(Obs, '') IN
      (SELECT COALESCE(Obs, '')
       FROM facturi
        WHERE NrFact = 1120)

```

Deseori o subconsultare extrage valori nenule combinate cu cele nule. De exemplu, în interogarea următoare:

```

SELECT *
FROM facturi
WHERE Obs IN
      (SELECT Obs
       FROM facturi
        WHERE NrFact = 1112 OR NrFact=1111)

```

subconsultarea va extrage două linii, dintre care cea corespunzătoare facturii 1111 este NULL. Conform logicii predicatelor în care apar valori NULL, interogarea de mai sus afișează numai facturile care au observații nenule egale cu ale facturii 1112 (vezi figura 9.6).

| NRFACT | DATAFACT | CODCL | OBS |
|--------|------------|-------|-------------------------|
| 3112 | 01-09-2007 | 1005 | Probleme cu transportul |
| 2112 | 14-08-2007 | 1005 | Probleme cu transportul |
| 1112 | 01-08-2007 | 1005 | Probleme cu transportul |

Figura 9.6. Valori nule extrase prin subconsultări

Dacă se folosește NOT IN, însă:

```
SELECT *
FROM facturi
WHERE Obs NOT IN
(SELECT Obs
FROM facturi
WHERE NrFact = 1112 OR NrFact=1111)
```

rezultatul nu va conține nici un rând. Spectrul NULLității este, pentru mulți practicieni, suficient de stimulant pentru renunțarea la subconsultări în favoarea joncțiunii externe, însă, în situații precum aceasta, în care și atributul testat (nu numai valorile extrase prin subconsultare) poate prezenta valori NULL, cel mai sănătos e să folosim funcția COALESCE.

9.2.Subconsultări & comparații în clauza WHERE

Până acum subconsultările au fost conectate la fraza SELECT superioară exclusiv prin operatorul IN. În continuare vom vedea că pentru (sub)interogările comparative, pot fi întrebuințați ALL, SOME, ANY. Atunci când rezultatul unei subconsultări se concretizează într-o tabelă cu o singură coloană și o singură linie (i se spune interogare scalară), corelarea poate fi făcută cu operatorii de comparație obișnuiți: =, >, >=, <, <=. Vom ilustra această facilități prin câteva exemple.

Ce facturi au fost emise în ziua în care a fost întocmită factura 1120 ?

Dacă în locul operatorului IN din paragraful anterior folosim egal, nu este nici o problemă:

```
SELECT NrFact
FROM facturi
WHERE DataFact =
(SELECT DataFact
FROM facturi
WHERE NrFact=1120)
```

În schimb, dacă în interogarea pentru aflarea *codurilor poștale* (plus localitatea, indicativul județului) în care există măcar un client schimbăm IN cu =:

```
SELECT * FROM coduri_postale
WHERE CodPost = (SELECT CodPost FROM clienti)
```

recepționăm un mesaj de eroare în care ni se spune clar că subconsultarea conține mai mult de o linie – vezi figura 9.7.

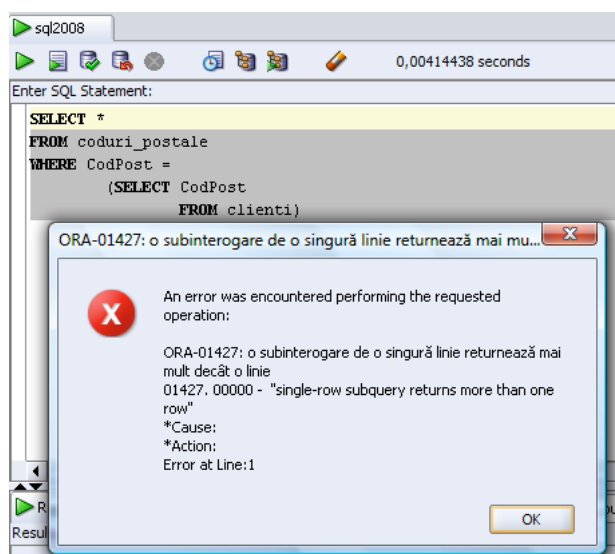


Figura 9.7. Folosirea eronată a semnului = pentru conexiunea cu subconsultarea

Care este cel mai mare preț unitar la care a fost vândut un produs, și care este produsul, precum și factura unde se înregistrează respectivul preț maxim ?

Este una din problemele care ne-au dat de furcă în paragraful 6.7.4. Renunțând la overdoza de improvizație din acel paragraf, cu ajutorul subconsultărilor obținem o soluție foarte simplă:

```
SELECT NrFact, DenPr, PretUnit
FROM liniifact lf INNER JOIN produse p
ON lf.CodPr=p.CodPr
WHERE PretUnit = (SELECT MAX(PretUnit) FROM liniifact)
ORDER BY 1
```

De data aceasta rezultatul (vezi figura 9.8) este complet, în sensul că vor fi afișate toate facturile și produsele în/pentru care se înregistrează prețul respectiv.

| NFACT | DENPR | PRETUNIT |
|-------|----------|----------|
| 1114 | Produs 5 | 7064 |
| 1121 | Produs 5 | 7064 |
| 2121 | Produs 5 | 7064 |

Figura 9.8. Facturile și produsele în/pentru care s-au înregistrat cel mai mare preț

Care sunt cele mai mari două prețuri unitare de vânzare, care sunt produsele și facturile pentru care se înregistrează respectivele prețuri maxime ?

Iată o variantă bazată pe subconsultări:

```

SELECT NrFact, DenPr, PretUnit
FROM liniifact lf INNER JOIN produse p
    ON lf.CodPr=p.CodPr
WHERE PretUnit >=
    (SELECT MAX(PretUnit)
     FROM liniifact
     WHERE PretUnit <
        (SELECT MAX(PretUnit)
         FROM liniifact
         )
    )
ORDER BY PretUnit DESC, NrFact

```

Pentru a înțelege mecanismul acestei interogări, pornim de la SELECT-ul “cel mai de jos”. *SELECT MAX(PretUnit) FROM liniifact* extrage prețul unitar maxim din tabela LINIIFACT. Subconsultarea superioară, (*SELECT MAX(PretUnit) FROM liniifact WHERE PretUnit < (...ultima subconsultare...)*), determină al doilea preț unitar din LINIIFACT. SELECT-ul principal afișează toate prețurile unitare mai mari sau egale cu penultimul. Figura 9.9 ilustrează acest mecanism.

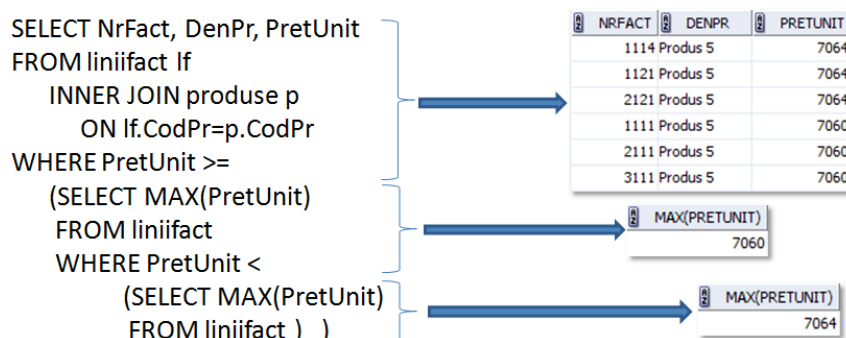


Figura 9.9. Cele mai mari două prețuri, facturile și produsele în/pentru care s-au înregistrat cel mai mare preț

Care sunt cele mai mari cinci prețuri unitare de vânzare, produsele și facturile în care apar cele cinci prețuri maxime ?

Aici voiam, de fapt, să ajungem:

```

SELECT NrFact, DenPr, PretUnit
FROM liniifact lf INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE PretUnit >
    (SELECT MAX(PretUnit)
     FROM liniifact
     WHERE PretUnit <

```

```

        (SELECT MAX(PretUnit)
         FROM liniifact
         WHERE PretUnit <
        (SELECT MAX(PretUnit)
         FROM liniifact
         WHERE PretUnit <
        (SELECT MAX(PretUnit)
         FROM liniifact
         WHERE PretUnit <
        (SELECT MAX(PretUnit)
         FROM liniifact
         WHERE PretUnit <
        (SELECT MAX(PretUnit)
         FROM liniifact
        )
    )
)
)
)
)
ORDER BY PretUnit DESC, NrFact

```

Logica este cea din soluția anterioară, iar rezultatul este prezentat în figura 9.10. Celor tari de înger le sugerez să încerce cu primele 10, 20 s.a.m.d. prețuri unitare.

| NRFACT | DENPR | PRETUNIT |
|---------------|-------|----------|
| 1114 Produs 5 | | 7064 |
| 1121 Produs 5 | | 7064 |
| 2121 Produs 5 | | 7064 |
| 1111 Produs 5 | | 7060 |
| 2111 Produs 5 | | 7060 |
| 3111 Produs 5 | | 7060 |
| 1119 Produs 5 | | 6300 |
| 2119 Produs 5 | | 6300 |
| 3119 Produs 5 | | 6300 |
| 1114 Produs 4 | | 1705 |
| 1119 Produs 4 | | 1410 |
| 2119 Produs 4 | | 1410 |
| 3119 Produs 4 | | 1410 |

Figura 9.10. Cele mai mari cinci prețuri unitare

În PostgreSQL o clauză de mare ajutor este LIMIT prin care se extrag primele n valori ale unei expresii dintr-un set de înregistrări. Astfel, soluția devine jenant de simplă:

```

SELECT NrFact, DenPr, PretUnit
FROM liniifact INNER JOIN produse ON liniifact.CodPr=produse.CodPr
WHERE PretUnit IN
    (SELECT DISTINCT PretUnit
     FROM liniifact
     ORDER BY PretUnit DESC
     LIMIT 5)

```

Bine, nu chiar jenant ! Subconsultarea trebuie neapărat să folosească clauza DISTINCT pentru ca să conțină cele mai mari cinci valori ale prețurilor unitare. La fel de simplă este clauza TOP din MS SQL Server (și Visual FoxPro):

```

SELECT TOP 5 NrFact, DenPr, PretUnit
FROM liniifact INNER JOIN produse ON liniifact.CodPr=produse.CodPr
ORDER BY PretUnit DESC

```

și clauza FETCH din DB2:

```

SELECT NrFact, DenPr, PretUnit
FROM liniifact INNER JOIN produse ON liniifact.CodPr=produse.CodPr
ORDER BY PretUnit DESC
FETCH FIRST 5 ROWS ONLY

```

După cum am văzut în figura 9.7, atunci când se folosește semnul = pentru a lega o subconsultare de (sub)consultarea „superioară”, este necesar ca rezultatul subconsultării să fie scalar (o sigură linie și o singură coloană). Lucrurile nu stau diferit nici cu ceilalți operatori de comparație <, >, <=, >=, < > sau #. Dacă, în cea mai mare parte a cazurilor de până acum, se compara un atribut (sau rezultatul unei expresii/funcții) cu o valoare scalară, prin clauzele ALL, SOME și ANY se compară valoarea atributului/funcției/expresiei cu un set de tupluri (absamblu de linii) extras printr-o subconsultare.

Care sunt produsele vândute la prețuri unitare superioare oricărui preț unitar la care a fost vândut 'Produs 1' ?

```

SELECT DISTINCT DenPr, PretUnit
FROM produse p INNER JOIN liniifact lf ON p.CodPr=lf.CodPr
WHERE PretUnit > ALL
    (SELECT DISTINCT PretUnit
     FROM produse p INNER JOIN liniifact lf ON p.CodPr=lf.CodPr
     WHERE DenPr ='Produs 1')
ORDER BY DenPr, PretUnit DESC

```

Ca orice interogare pe două niveluri, ostilitățile se derulează în doi pași (vezi figura 9.11). Mai întâi se execută subconsultarea și se obține o tabelă intermediară în care se găsesc toate prețurile unitare la care a fost vândut, în decursul istoriei, Produs 1.

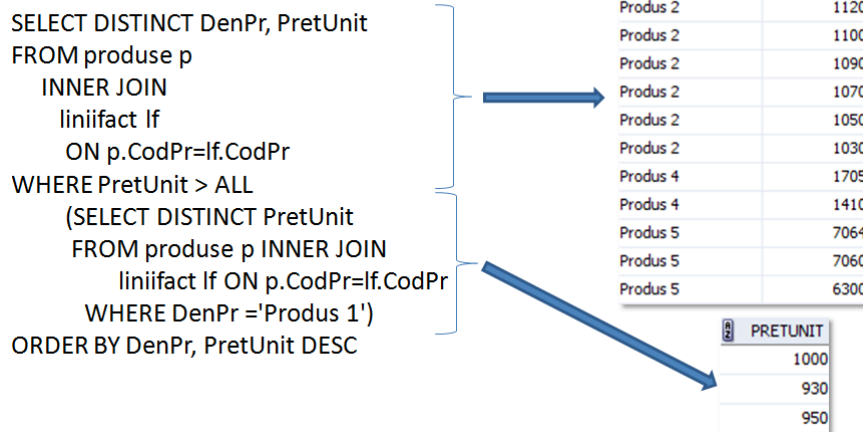


Figura 9.11. Produsele vândute la prețuri mai mari decât toate prețurile produsului 1

Cum operatorul de conexiune a frazei SELECT principale cu subconsultarea este > ALL, din joncțiunea tabelor PRODUSE și LINIIFACT vor fi extrase numai liniile care au valoarea atributului PretUnit mai mare decât *toate* valorile din dreapta-jos figurii 9.11.

Care sunt produsele vândute la prețuri unitare superioare măcar unui preț unitar al 'Produsului 1' ?

Este genul de situații în care se folosește SOME sau ANY (sunt echivalente).

```
SELECT DISTINCT DenPr, PretUnit
FROM produse p INNER JOIN liniifact lf ON p.CodPr=lf.CodPr
WHERE PretUnit > ANY
(SELECT DISTINCT PretUnit
FROM produse p INNER JOIN liniifact lf ON p.CodPr=lf.CodPr
WHERE DenPr ='Produs 1')
ORDER BY DenPr, PretUnit DESC
```

Rezultatul este cel din figura 9.12; spre deosebire de ALL, și ANY și SOME selectează liniile pentru care prețul unitar este mai mare decât *măcar una* dintre valorile obținute prin subconsultare.

| DENPR | PRETUNIT |
|----------|----------|
| Produs 1 | 1000 |
| Produs 1 | 950 |
| Produs 2 | 1120 |
| Produs 2 | 1100 |
| Produs 2 | 1090 |
| Produs 2 | 1070 |
| Produs 2 | 1050 |
| Produs 2 | 1030 |
| Produs 2 | 1000 |
| Produs 2 | 975 |
| Produs 4 | 1705 |
| Produs 4 | 1410 |
| Produs 5 | 7064 |
| Produs 5 | 7060 |
| Produs 5 | 6300 |

Figura 9.12. Produse cu cel puțin un preț unitar superior măcar unui preț la care a fost al Produsului 1

De reținut că operatorul =ANY este echivalent cu IN.

Câți alți angajați au salariul tarifar egal cu cel al ANGAJATului 2 ?

Soluția:

```
SELECT COUNT(*) - 1 AS Nr
FROM personal2
WHERE SalTarifar IN
  (SELECT SalTarifar
   FROM personal2
   WHERE NumePren='ANGAJAT 2')
```

este echivalentă cu:

```
SELECT COUNT(*) - 1 AS Nr
FROM personal2
WHERE SalTarifar =ANY
  (SELECT SalTarifar
   FROM personal2
   WHERE NumePren='ANGAJAT 2')
```

Firește, folosirea unuia din cei trei operatori nu este obligatorie atunci când subconsultarea conține o funcție-agregat (ce întoarce o valoare dintr-un ansamblu de tupluri) - MIN, MAX, COUNT, SUM, AVG.

Care este ultima factură întocmită (factura cea mai recentă) și data în care a fost emisă?

Avem variante destule, chiar pe alese:

```
SELECT DataFact, NrFact AS UltimaFactura
FROM facturi
```



```
WHERE NrFact IN
      (SELECT MAX(NrFact)
       FROM facturi)
```

sau

```
SELECT DataFact, NrFact AS UltimaFactura
FROM facturi
WHERE NrFact =
      (SELECT MAX(NrFact)
       FROM facturi)
```

sau

```
SELECT DataFact, NrFact AS UltimaFactura
FROM facturi
WHERE NrFact = ANY
      (SELECT MAX(NrFact)
       FROM facturi)
```

sau

```
SELECT DataFact, NrFact AS UltimaFactura
FROM facturi
WHERE NrFact =ALL
      (SELECT MAX(NrFact)
       FROM facturi)
```

Fără funcția MAX, interogarea ar avea forma:

```
SELECT DataFact, NrFact AS UltimaFactura
FROM facturi
WHERE NrFact >= ALL
      (SELECT NrFact
       FROM facturi)
```

9.3.Subconsultări în clauza HAVING

Predicatele incluse în clauza HAVING ale interogărilor din capitolele anterioare comparau o expresie cu o constantă. În cele ce urmează vom ataca problema includerii în clauza HAVING a subconsultărilor.

9.3.1. Comparații și topuri la nivel de grupuri

Începem prin a reveni asupra unor probleme deja discutate, formulând câteva soluții noi. Nu o să mai insistăm cu ușoarele modificări de sintaxă necesare portării comenzilor în toate cele patru dialecte SQL.

Care sunt zilele în care s-au emis mai multe facturi decât pe 2 august 2007 ?

```
SELECT DataFact AS Zi, COUNT(NrFact) AS Nr_Facturilor
```

```

FROM facturi
GROUP BY DataFact
HAVING COUNT(NrFact) >
    (SELECT COUNT(NrFact)
     FROM facturi
     WHERE DataFact = DATE'2007-08-02')

```

Care este ziua în care s-au emis cele mai multe facturi ?

```

SELECT DataFact, COUNT(*) AS Nr_Facturilor
FROM facturi
GROUP BY DataFact
HAVING COUNT(*) >= ALL
    (SELECT COUNT(*)
     FROM facturi
     GROUP BY DataFact)

```

Subconsultarea calculează numărul de facturi corespunzător fiecărei zile. Predicatul clauzei HAVING compară numărul de facturi al fiecărei zile cu toate valorile extrase de subconsultare. Se obține tabela din figura 9.13.

| DATAFACT | NR_FACTURILOR |
|------------|---------------|
| 01-08-2007 | 4 |
| 07-08-2007 | 4 |

Figura 9.13. Zilele în care s-au emis cele mai multe facturi

În PostgreSQL ne putem folosi fără rușine (doar nu este prima dată!) de clauza LIMIT:

```

SELECT DataFact, COUNT(*) AS Nr_Facturilor
FROM facturi
GROUP BY DataFact
HAVING COUNT(*) =
    (SELECT COUNT(*)
     FROM facturi
     GROUP BY DataFact
     ORDER BY COUNT(*) DESC LIMIT 1)

```

Se cuvine de adăugat că numai Oracle permite și o soluție bazată pe incluziunea unei funcții în alta:

```

SELECT DataFact, COUNT(*) AS Nr_Facturilor
FROM facturi
GROUP BY DataFact
HAVING COUNT(*) =
    (SELECT MAX(COUNT(*))
     FROM facturi)

```

GROUP BY DataFact)

iar în SQL Server se poate formula o soluție elegantă bazată pe operatorul TOP:

```
SELECT TOP 1 DataFact, COUNT(*) AS Nr
FROM facturi
GROUP BY DataFact
ORDER BY Nr DESC
```

Surprinzător pentru unii (ca mine), soluția funcționează. ORDER BY se aplică grupurilor, iar ordonarea împreună cu opțiunea TOP extrag rezultatul corect, însă incomplet, întrucât chiar dacă sunt mai multe zile cu același număr maxim de facturi, interogarea furnizează doar unul. Așa că varianta corectă și completă este:

```
SELECT DataFact, COUNT(*) AS Nr_Facturilor
```

```
FROM facturi
```

```
GROUP BY DataFact
```

```
HAVING COUNT(*) =
```

```
(SELECT TOP 1 COUNT(*)
FROM facturi
GROUP BY DataFact
ORDER BY COUNT(*) DESC)
```

Adăugăm și soluția DB2:

```
SELECT DataFact, COUNT(*) AS Nr_Facturilor
```

```
FROM facturi
```

```
GROUP BY DataFact
```

```
HAVING COUNT(*) =
```

```
(SELECT COUNT(*)
FROM facturi
GROUP BY DataFact
ORDER BY COUNT(*) DESC
FETCH FIRST 1 ROW ONLY)
```

Care este clientul care a cumpărat cele mai multe produse ?

Analog problemei anterioare, formulăm o soluție generală:

```
SELECT DenCl, COUNT(DISTINCT CodPr) AS "CiteProduse"
```

```
FROM clienti c
```

```
INNER JOIN facturi f ON c.CodCl=f.CodCl
```

```
INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
```

```
GROUP BY DenCl
```

```
HAVING COUNT(DISTINCT CodPr) >= ALL
```

```
(SELECT COUNT(DISTINCT CodPr)
```

```
FROM facturi f INNER JOIN liniifact lf
```

```
ON f.NrFact= lf.NrFact
```

```
GROUP BY CodCl)
```

| DENCL | CiteProduce |
|--------------|-------------|
| Client 3 SRL | 4 |

Figura 9.14. Clientul care a cumpărat cele mai multe produse

una "Oracle only" (COUNT în MAX):

```
SELECT DenCl, COUNT(DISTINCT CodPr) AS "CiteProduce"
FROM clienti c INNER JOIN facturi f ON c.CodCl=f.CodCl
      INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
GROUP BY DenCl
HAVING COUNT(DISTINCT CodPr) =
      (SELECT MAX(COUNT(DISTINCT CodPr))
      FROM facturi f INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
      GROUP BY CodCl)
```

o alta VFP/SQL Server (clauza TOP) :

```
SELECT TOP 1 DenCl, COUNT(DISTINCT CodPr) AS CiteProduce
FROM clienti c INNER JOIN facturi f ON c.CodCl=f.CodCl
      INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
GROUP BY DenCl
ORDER BY CiteProduce DESC
```

una PostgreSQL (LIMIT):

```
SELECT DenCl, COUNT(DISTINCT CodPr) AS "CiteProduce"
FROM clienti c INNER JOIN facturi f ON c.CodCl=f.CodCl
      INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
GROUP BY DenCl
HAVING COUNT(DISTINCT CodPr) =
      (SELECT COUNT(DISTINCT CodPr)
      FROM facturi f INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
      GROUP BY CodCl
      ORDER BY COUNT(DISTINCT CodPr) DESC LIMIT 1)
```

iar în DB2 se înlocuiește ultima linie cu:

```
...
ORDER BY COUNT(DISTINCT CodPr) DESC FETCH FIRST 1 ROW ONLY)
```

Care este compartimentul cu cea mai bună medie a salariilor tarifare ?


Se exclude din discuție compartimentul DIRECȚIUNE în care apare, singur și ferice, directorul general (sintaxa nu este acceptată de SQL Server din cauza funcției TRUNC):

```
SELECT Compart, TRUNC(AVG(SalTarifar),0) AS Medie_Sal
```

```

FROM personal2
WHERE Compart <> 'DIRECTIUNE'
GROUP BY Compart
HAVING AVG(SalTarifar) >= ALL
      (SELECT AVG(SalTarifar)
       FROM personal2
       WHERE Compart <> 'DIRECTIUNE'
       GROUP BY Compart)

```



| COMPART | MEDIE_SAL |
|-----------|-----------|
| MARKETING | 1383 |

Figura 9.15. Compartimentul cu cea mai bună medie a salariilor tarifare

Pentru aducerea aminte a structurilor alternative, putem folosi și variantele:

```

SELECT Compart, TRUNC(AVG (
      CASE
      WHEN Compart <> 'DIRECTIUNE' THEN SalTarifar
      ELSE 0
      END),0) AS Medie_Sal
FROM personal2
GROUP BY Compart
HAVING AVG (CASE WHEN Compart <> 'DIRECTIUNE' THEN SalTarifar
      ELSE 0
      END) >= ALL
      (SELECT AVG (
      CASE
      WHEN Compart <> 'DIRECTIUNE'
      THEN SalTarifar
      ELSE 0
      END
      )
      FROM personal2
      GROUP BY Compart)

```

și o variantă bazată pe DECODE-ul „oraclisto-db2-ist”:

```

SELECT Compart, TRUNC(AVG (
      DECODE (Compart, 'DIRECTIUNE', 0, SalTarifar)
      ),0) AS Medie_Sal
FROM personal2
GROUP BY Compart
HAVING AVG (DECODE (Compart, 'DIRECTIUNE', 0, SalTarifar))

```

```

=> ALL
(SELECT AVG (DECODE (Compart, 'DIRECTIUNE', 0, SalTarifar))
FROM personal2
GROUP BY Compart)

```

Funcția TRUNC a fost necesară pentru afișarea numai a părții întregi din medie. De data aceasta (și următoarele), trecem peste varianta Oracle cu funcție inclusă în altă funcție, și versiunile MS SQL Server.

Care este județul în care berea s-a vândut cel mai bine ?

În tabela PRODUSE există un atribut care reprezintă grupa în care se încadrează produsul respectiv. Berea este una dintre grupele îndrăgite.

```

SELECT Judet, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari_Bere
FROM judete j
    INNER JOIN coduri_postale cp ON j.Jud=cp.Jud
    INNER JOIN clienti c ON cp.CodPost=c.CodPost
    INNER JOIN facturi f ON c.CodCl=f.CodCl
    INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
    INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE Grupa='Bere'
GROUP BY Judet
HAVING SUM(Cantitate * PretUnit * (1+ProcTVA))
=> ALL
(SELECT SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari_Bere
FROM judete j
    INNER JOIN coduri_postale cp ON j.Jud=cp.Jud
    INNER JOIN clienti c ON cp.CodPost=c.CodPost
    INNER JOIN facturi f ON c.CodCl=f.CodCl
    INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
    INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE Grupa='Bere'
GROUP BY Judet)

```

| JUDET | VINZARI_BERE |
|-------|--------------|
| Iasi | 1896545,5 |

Figura 9.16. Județul cu cea mai bună vânzare a produselor din grupa Bere

Pe baza exemplelor anterioare se pot redacta soluțiile „proprietary” DB2, Oracle, PostgreSQL și SQLServer.

Care sunt clienții cu valoarea vânzărilor peste medie ?

Noțiunea de medie este destul de alunecoasă. În acest caz media vânzărilor pe client se calculează împărțind vânzările totale la numărul clienților cărora li s-a trimis măcar o factură:

```
SELECT dencl AS client, TRUNC(SUM(cantitate *pretunit *(1 + proctva)),0) AS vinzari
FROM clienti c
      INNER JOIN facturi f ON c.CodCl=f.CodCl
      INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
      INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY dencl
HAVING SUM(cantitate *pretunit *(1 + proctva))      >= ALL
      (SELECT SUM(cantitate *pretunit *(1 + proctva)) / COUNT(DISTINCT f.codcl)
      FROM facturi f
      INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
      INNER JOIN produse p ON lf.CodPr=p.CodPr)
ORDER BY 1
```

| CLIENT | VINZARI |
|--------------|----------|
| Client 1 SRL | 15061538 |
| Client 3 SRL | 17413834 |
| Client 4 | 9479109 |

Figura 9.17. Clienți cu vânzări peste medie

Extrageți factura cu valoarea imediat peste cea medie.

Privitor la tipul de medie care ne interesează, vom împărți valoarea totală a vânzărilor la numărul de facturi emise:

```
SELECT f.NrFact, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Valoare
FROM liniifact lf INNER JOIN produse p ON p.CodPr = lf.CodPr
      INNER JOIN facturi f ON lf.NrFact = f.NrFact
GROUP BY f.NrFact
HAVING SUM(Cantitate * PretUnit * (1+ProcTVA))      <= ALL
      (SELECT SUM(Cantitate * PretUnit * (1+ProcTVA)) AS "Facturi_cu_val_peste_medie "
      FROM liniifact lf
      INNER JOIN produse p ON p.CodPr = lf.CodPr
      INNER JOIN facturi f ON lf.NrFact = f.NrFact
      GROUP BY f.NrFact
      HAVING SUM(Cantitate * PretUnit * (1+ProcTVA))      >
      (SELECT SUM(Cantitate * PretUnit * (1+ProcTVA)) /
      COUNT(DISTINCT f.NrFact) AS Media
      FROM liniifact lf
      INNER JOIN produse p ON p.CodPr = lf.CodPr
      INNER JOIN facturi f ON lf.NrFact = f.NrFact
```

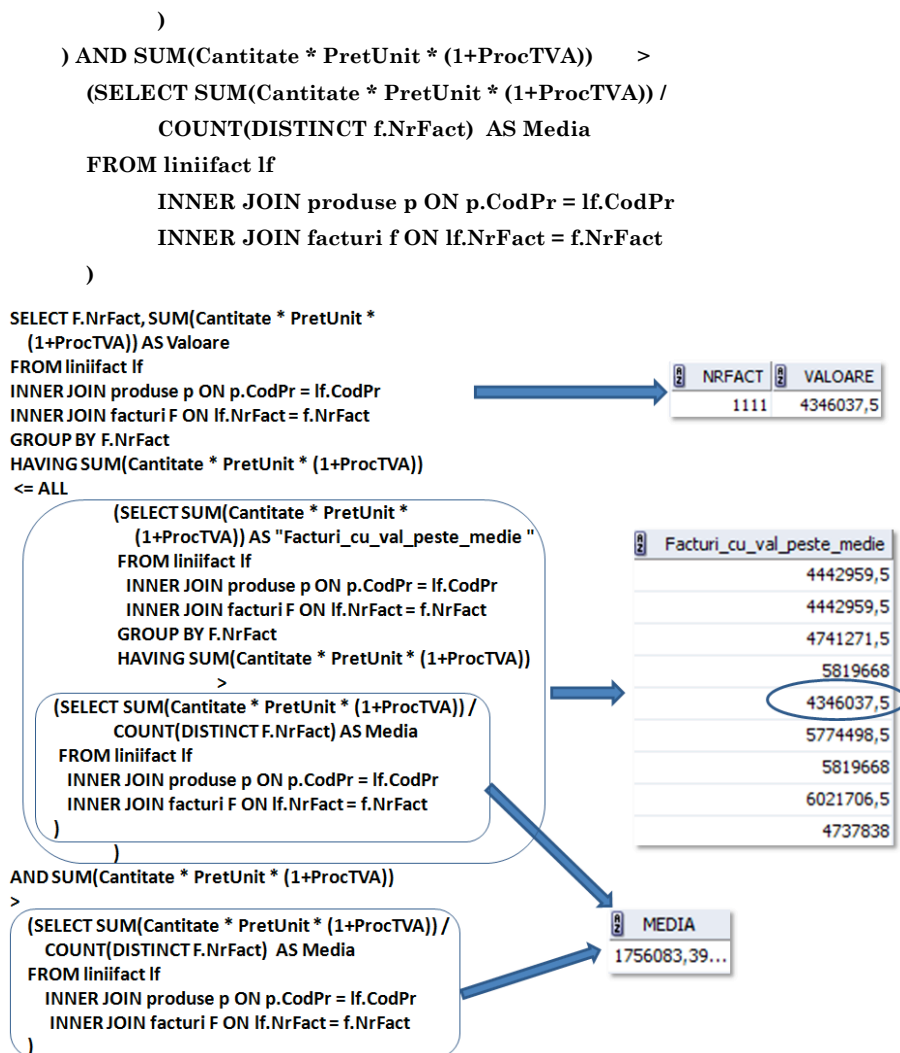


Figura 9.18. Factura cu cea mai mică valoare peste medie

Elementul de noutate al acestei interogări este includerea, într-o consultare ce prezintă clauza HAVING, a unei alte subconsultări în care apare, de asemenea, HAVING. Mecanismul de execuției este explicat în figura 9.18, iar sintaxa este comună celor patru dialecte.

Care este județul cu vânzări imediat superioare județului Neamț ?

```

SELECT Judet, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari
FROM judete j INNER JOIN coduri_postale cp ON j.Jud=cp.Jud
  INNER JOIN clienti c ON cp.CodPost=c.CodPost
  INNER JOIN facturi f ON c.CodCl=f.CodCl

```



```

        INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
        INNER JOIN produse p ON lf.CodPr=p.CodPr
    GROUP BY Judet
    HAVING SUM(Cantitate * PretUnit * (1+ProcTVA))
        <= ALL
        (SELECT SUM(Cantitate * PretUnit * (1+ProcTVA))
        FROM judete j INNER JOIN coduri_postale cp ON j.Jud=cp.Jud
        INNER JOIN clienti c ON cp.CodPost=c.CodPost
        INNER JOIN facturi f ON c.CodCl=f.CodCl
        INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
        INNER JOIN produse p ON lf.CodPr=p.CodPr
        GROUP BY Judet
        HAVING SUM(Cantitate * PretUnit * (1+ProcTVA))
            >
        (SELECT SUM(Cantitate * PretUnit * (1+ProcTVA))
        FROM judete j
        INNER JOIN coduri_postale cp ON j.Jud=cp.Jud
        INNER JOIN clienti c ON cp.CodPost=c.CodPost
        INNER JOIN facturi f ON c.CodCl=f.CodCl
        INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
        INNER JOIN produse p ON lf.CodPr=p.CodPr
        WHERE Judet='Neamt'
        )
    )
    AND SUM(Cantitate * PretUnit * (1+ProcTVA))
    >
    (SELECT SUM(Cantitate * PretUnit * (1+ProcTVA))
    FROM judete j
    INNER JOIN coduri_postale cp ON j.Jud=cp.Jud
    INNER JOIN clienti c ON cp.CodPost=c.CodPost
    INNER JOIN facturi f ON c.CodCl=f.CodCl
    INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
    INNER JOIN produse p ON lf.CodPr=p.CodPr
    WHERE Judet='Neamt')

```

| JUDET | VINZARI |
|--------|------------|
| Vaslui | 17413834,5 |

Figura 9.19. Județul cu vânzări imediat superioare Neamțului

9.3.2. Exagerări (vol. n+1)

Care este cel mai mare datornic dintre clienți ?

Este, cred, cea mai grea interogare de până acum. După cum am văzut atunci când am calculat pentru fiecare factură valorile facturate și încasate, la joncțiunea externă a "părții de facturare" cu "partea de încasare", fiecare tranșă de încasare se repetă în funcție de numărul de linii ce compun factura respectivă, iar fiecare linie a facturii se repetă în funcție de numărul de tranșe de încasare.

Trucul utilizat pentru a o scoate la capăt consta, pe de o parte, în împărțirea totalului tranșelor de încasare la numărul liniilor din factură, obținând astfel valoarea încasată a facturii și, pe de altă parte, împărțirea totalului valorilor liniilor din facturi la numărul tranșelor de încasare. Pentru problema de față trebuie găsită altă soluție, deoarece gruparea nu o facem la nivel de factură, ci la nivel de client. Numărul de linii din facturi și numărul de tranșe nu mai au nicio relevanță la gruparea după client.

Pentru încasări, ar fi o idee. Deoarece o tranșă se repetă pentru fiecare linie a facturii, putem elimina din SUM toate tranșele pentru care *Linie* > 1:

```
SELECT DenCl AS Client,
       SUM (
           CASE
             WHEN lf.Linie > 1 THEN NULL
             ELSE COALESCE(Transa,0)
           END
       ) AS Valoare_Incasata
FROM clienti c
     INNER JOIN facturi f ON c.CodCl=f.CodCl
     INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
     INNER JOIN produse p ON lf.CodPr=p.CodPr
     LEFT OUTER JOIN incasfact i ON f.NrFact=i.NrFact
GROUP BY DenCl
```

Rezultatul din figura 9.20 ne încredințează că ideea n-a fost chiar rea. Cu valoarea facturată, însă, lucrurile sunt mai complicate. Fiecare linie din factură se repetă de atâtea ori, în funcție de câte tranșe de încasare există pentru factura respectivă. Cum gruparea se face pe clienți, numărul tranșelor este irelevant. Trebuie însumate valorile distincte ale facturii și liniei.

| CLIENT | VALOARE_INCASATA |
|--------------|------------------|
| Client 1 SRL | 286,490.00 |
| Client 2 SA | 106,275.00 |
| Client 3 SRL | 0.00 |
| Client 4 | 0.00 |
| Client 5 SRL | 125,516.00 |
| Client 6 SA | 0.00 |
| Client 7 SRL | 0.00 |

Figura 9.20. Incasările pe clienți

Este adevărat, clauza DISTINCT poate fi folosită și cu SUM. Dar se poate întâmpla ca, la un moment dat, să existe într-o factură, două linii cu aceeași valoare. E musai de însumat valorile expresiei pentru combinații distincte (*NrFact*, *Linie*). Cofirmând vocația de popor de improvizatori (n-am zis cârpaci, să nu supăr adevărații patrioți), recurgem la un truc ieftin dar, vorba dlui. Graur (comentatorul), năucitor !

```

SELECT DenCl,
       SUM( DISTINCT
            1000000000000000 * If.NrFact +
            1000000000000 * CAST (If.Linie AS NUMERIC) +
            Cantitate * PretUnit * (1+ProcTVA))
AS Valoare_Facturata_Aiurea
FROM produse p
     INNER JOIN liniifact lf ON p.CodPr = lf.CodPr
     INNER JOIN facturi f ON lf.NrFact = f.NrFact
     INNER JOIN clienti c ON f.CodCl = c.CodCl
     LEFT OUTER JOIN incasfact i ON f.NrFact=i.NrFact
GROUP BY DenCl

```

Nu putem discuta detalii înainte de a vedea ce a putut fi obținut din interogare – vezi figura 9.21.

| DENCL | VALOARE_FACTURATA_AIUREA |
|--------------|---------------------------------|
| Client 1 SRL | 51,874,040,000,015,061,538.0000 |
| Client 2 SA | 6,339,003,000,000,361,335.0000 |
| Client 3 SRL | 25,428,030,000,017,413,834.5000 |
| Client 4 | 6,484,006,000,009,479,109.5000 |
| Client 5 SRL | 12,672,009,000,000,432,400.0000 |
| Client 6 SA | 3,342,006,000,006,021,706.5000 |
| Client 7 SRL | 6,348,003,000,000,400,411.5000 |

Figura 9.21. Improvizație pentru calculul valorii vânzărilor pe clienți

Expresia de calcul a valorii facturate introduce, pe lângă Cantitate, PretUnit și ProcTVA, și NrFact și Linie. Ultimele două attribute sunt înmulțite cu constante foarte mari (10^{15} , respectiv 10^{12}), suficient de mari pentru ca valoarea propriu-zisă a facturilor să nu fie “afectată”. Este sigur că valorile obținute pentru fiecare client au fost calculate luându-se în calcul o singură dată fiecare linie dintr-o factură. Iar

dacă privim în rezultat partea din dreapta fiecărui număr (după cele patru sau mai multe zerouri), observăm că acelea sunt chiar valorile facturate care ne interesează. Prin urmare, nu ne mai rămâne decât să extragem acea parte:

```
SELECT DenCl,
       SUM( DISTINCT
            1000000000000000 * If.NrFact +
            1000000000000 * CAST (If.Linie AS NUMERIC) +
            Cantitate * PretUnit * (1+ProcTVA)) AS Valoare_Umflata,
       CAST (
            RIGHT( CAST (SUM( DISTINCT
            1000000000000000 * If.NrFact +
            1000000000000 * CAST (If.Linie AS NUMERIC) +
            Cantitate * PretUnit * (1+ProcTVA)) AS CHAR(35))
            , 19)
            AS DECIMAL (16,4)) AS Valoare_Fact_Corect
FROM produse p
     INNER JOIN liniifact If ON p.CodPr = If.CodPr
     INNER JOIN facturi f ON If.NrFact = f.NrFact
     INNER JOIN clienti c ON f.CodCl = c.CodCl
     LEFT OUTER JOIN incasfact i ON f.NrFact=i.NrFact
GROUP BY DenCl
```

Obținem ceea ce doream, adică valoarea vânzărilor pe clienți, după cum o arată figura 9.22.

| DENCL | VALOARE_UMFLATA | VALOARE_FACT_CORECT |
|--------------|---------------------------------|---------------------|
| Client 1 SRL | 51,874,040,000,015,061,538.0000 | 15,061,538.0000 |
| Client 2 SA | 6,339,003,000,000,361,335.0000 | 361,335.0000 |
| Client 3 SRL | 25,428,030,000,017,413,834.5000 | 17,413,834.5000 |
| Client 4 | 6,484,006,000,009,479,109.5000 | 9,479,109.5000 |
| Client 5 SRL | 12,672,009,000,000,432,400.0000 | 432,400.0000 |
| Client 6 SA | 3,342,006,000,006,021,706.5000 | 6,021,706.5000 |
| Client 7 SRL | 6,348,003,000,000,400,411.5000 | 400,411.5000 |

Figura 9.22. Valoarea vânzărilor, pe clienți

În final, soluția problemei (și răspunsul în figura 9.23):

```
SELECT DenCl,
       CAST (
            RIGHT( CAST (SUM( DISTINCT 1000000000000000 * If.NrFact +
            1000000000000 * CAST (If.Linie AS NUMERIC) +
            Cantitate * PretUnit * (1+ProcTVA)
            ) AS CHAR(35))
            , 19)
            AS DECIMAL (16,4)) AS Valoare_Facturata,
```

```

SUM ( CASE WHEN If.Linie > 1 THEN NULL
          ELSE COALESCE(Transa,0) END
      ) AS Valoare_Incasata,
CAST (
    RIGHT( CAST (SUM( DISTINCT  10000000000000000 * If.NrFact
        + 1000000000000 * CAST (If.Linie AS NUMERIC) +
        Cantitate * PretUnit * (1+ProcTVA))
        AS CHAR(35))
        , 19)
    AS DECIMAL (16,4))
-
SUM ( CASE WHEN If.Linie > 1 THEN NULL
          ELSE COALESCE(Transa,0) END )
AS Rest_De_Incasat
FROM produse P
INNER JOIN liniifact If ON P.CodPr = If.CodPr
INNER JOIN facturi f ON If.NrFact = f.NrFact
INNER JOIN clienti c ON f.CodCl = c.CodCl
LEFT OUTER JOIN incasfact i ON f.NrFact=i.NrFact
GROUP BY DenCl
HAVING
    ( CAST (
        RIGHT( CAST (SUM( DISTINCT
            10000000000000000 * If.NrFact + 1000000000000 *
            CAST (If.Linie AS NUMERIC) +    Cantitate *
            PretUnit * (1+ProcTVA))
            AS CHAR(35))
            , 19)
        AS DECIMAL (16,4))
    -
    SUM ( CASE WHEN If.Linie > 1 THEN NULL ELSE COALESCE(Transa,0) END )
    )
    >= ALL
(SELECT CAST ( RIGHT( CAST (SUM( DISTINCT
    10000000000000000 * If.NrFact + 1000000000000 *
    CAST (If.Linie AS NUMERIC) +
    Cantitate * PretUnit * (1+ProcTVA))
    AS CHAR(35))
    , 19)
    AS DECIMAL (16,4))
    -

```

```

SUM ( CASE WHEN If.Linie > 1 THEN NULL ELSE COALESCE(Transa,0) END
)
FROM produse p
INNER JOIN liniifact lf ON p.CodPr = lf.CodPr
INNER JOIN facturi f ON lf.NrFact = f.NrFact
INNER JOIN clienti c ON f.CodCl = c.CodCl
LEFT OUTER JOIN incasfact i ON f.NrFact=i.NrFact
GROUP BY DenCl)

```

| DENCL | VALOARE_FACTURATA | VALOARE_INCASATA | REST_DE_INCASAT |
|--------------|-------------------|------------------|-----------------|
| Client 3 SRL | 17,413,834.5000 | 0.00 | 17,413,834.5000 |

Figura 9.23. Clientul ce are cel mai mare rest de plată

După atâta strădanie, este destul de greu de suportat vestea că soluția funcționează doar în DB2. Astfel, SQL Server nu face nazuri la execuție, dar afișează ce știe el – vezi figura 9.24. Necazul vine de la modul de extragere și conversie a șirului de caractere obținut prin expresia-artificiu pentru calcularea valorii facturate. Dacă înlocuim toate argumentele funcțiilor RIGHT din *RIGHT ...*, 19) în *RIGHT(..., 27)*, rezultatul este cel corect.

| DenCl | Valoare_Facturata | Valoare_Incasata | Rest_De_Incasat |
|--------------|-------------------|------------------|-----------------|
| Client 3 SRL | 3834.5000 | 0.00 | 3834.50 |

Figura 9.24. Rezultatul interogării de mai sus executată în SQL Server

Atât Oracle, cât și PostgreSQL nu au implementat funcția RIGHT; de aceea, este necesară înlocuirea acesteia cu SUBSTR (plus, pentru siguranță funcția LPAD). Dar am exagerat destul. Revenim în paragraful 9.4 cu soluție rezonabilă pentru această problemă.

9.3.3. Grupuri și diviziuni relaționale

Revenind la cazul teoretic din paragraful 4.4.7 (figura 4.30), câtul diviziunii relaționale a RD1 : RD2 (altfel spus, găsirea tuturor icșilor care sunt în RD1 în combinație cu toți igrecii din RD2), se calculează în SQL și astfel:

```

SELECT X
FROM rd1
GROUP BY X
HAVING COUNT(Y) =
(SELECT COUNT(Y)
FROM rd2)

```

Care sunt clienții pentru există cel puțin câte o factură emisă în fiecare zi cu vânzări din perioada 10-30 septembrie 2007? (exemplu 23, paragraf 4.4.7)?

```

SELECT DenCl, COUNT(DISTINCT DataFact)

```

```

FROM facturi f INNER JOIN clienti c ON f.CodCl=c.CodCl
WHERE DataFact BETWEEN DATE'2007-09-10' AND DATE'2007-09-30'
GROUP BY DenCl
HAVING COUNT(DISTINCT DataFact) =
    (SELECT COUNT (DISTINCT DataFact)
     FROM facturi
     WHERE DataFact BETWEEN DATE'2007-09-10' AND DATE'2007-09-30'
    )

```

Celelalte interogări din acest paragraf respectă sintaxa tuturor celor patru dialecte. În schimb aceasta necesită eliminarea cuvintelor DATE pentru a fi executabilă în DB2 și SQL Server.

Ce produse au fost vândute tuturor clienților ? (exemplu 26, paragraf 4.4.7)?

```

SELECT DenPr
FROM produse p
    INNER JOIN liniifact lf ON p.CodPr=lf.CodPr
    INNER JOIN facturi f ON lf.NrFact=f.NrFact
GROUP BY DenPr
HAVING COUNT(DISTINCT CodCl) =
    (SELECT COUNT (CodCl)
     FROM clienti)

```

Care sunt facturile ce conțin măcar produsele din factura 1117? - exemplul 27 din paragraful 4.4.7.

Problema pare mai dificilă decât precedentele, pentru că trebuie numărate, din fiecare factură, numai produsele care se găsesc în factura etalon 1117:

```

SELECT lf1.NrFact
FROM liniifact lf1
    INNER JOIN liniifact lf2 ON lf1.CodPr=lf2.CodPr AND lf2.NrFact=1117
GROUP BY lf1.NrFact
HAVING COUNT(DISTINCT lf1.CodPr) =
    (SELECT COUNT(DISTINCT CodPr)
     FROM liniifact
     WHERE NrFact=1117)

```

9.4.Subconsultări în clauza FROM

Posibilitatea de a defini tabele „ad-hoc”, în clauza FROM, este unul dintre cele mai importante daruri pe care SQL-ul îl oferă utilizatorilor, deopotrivă profesioniști și neprofesioniști în ale bazelor de date.

9.4.1. Subconsultări pentru filtrare, calcule și grupări

Ce facturi au fost emise în aceeași zi cu factura 1120 ?

Nu este cel mai de efect mod de a introduce în scenă subconsultările din clauza FROM, dar ce ziceți de soluția următoare ?

```
SELECT f.*
FROM facturi f INNER JOIN
    (SELECT *
     FROM facturi
     WHERE NrFact=1120
    ) f1120
ON f.DataFact=f1120.DataFact
```

Dacă până acum în clauza FROM jonționam numai tabele ale bazei, în această interogare jonționăm (după atributului DataFact) o tabelă (FACTURI) cu o tabelă temporară (F1120), definită ad-hoc printr-o subconsultare. După cum o arată și figura 9.25, tabela F1120 are o singură linie, cea corespunzătoare facturii etalon (1120), de aici și numele său.

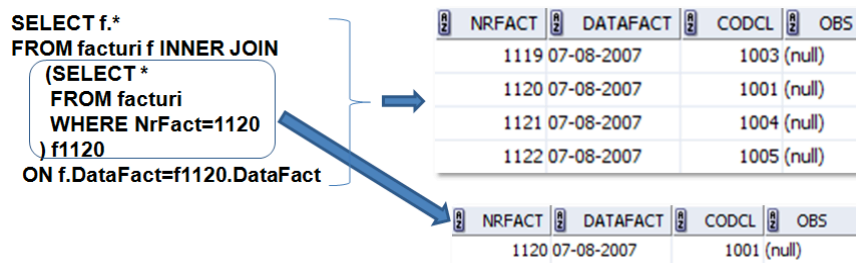


Figura 9.25. Joncțiunea unei tabele cu rezultatul unei subconsultări

În ce zile s-au vândut și produsul cu denumirea „Produs 1” și cel cu denumirea „Produs 2” ?

În clauza FROM vom defini două subconsultări, una care va extrage zilele în care s-a vândut „Produs 1” (ZILE_PROD1) și cealaltă destinată zilelor în care s-a vândut „Produs 2” (ZILE_PROD2). Rezultatele celor două subconsultări vor fi jonționate după atributul DataFact:

```
SELECT DISTINCT zile_prod1.DataFact
FROM
    (SELECT DataFact
     FROM produse p INNER JOIN liniifact lf ON p.CodPr=lf.CodPr
     INNER JOIN facturi f ON lf.NrFact=f.NrFact
     WHERE DenPr='Produs 1') zile_prod1
    INNER JOIN
    (SELECT DataFact
```



```

FROM produse p INNER JOIN liniifact lf ON p.CodPr=lf.CodPr
      INNER JOIN facturi f ON lf.NrFact=f.NrFact
WHERE DenPr='Produs 2') zile_prod2
      ON zile_prod1.DataFact = zile_prod2.DataFact
ORDER BY 1

```

Care dintre angajați au salariul tarifar egal cu cel al ANGAJAT2 ?

Lucrăm după același calapod:

```

SELECT NumePren FROM
      (SELECT NumePren, SalTarifar
      FROM personal2) temp1
      INNER JOIN
      (SELECT SalTarifar
      FROM personal2
      WHERE NumePren='ANGAJAT 2') temp2
      ON temp1.SalTarifar = temp2.SalTarifar

```

Adevărata forță a acestei opțiuni se arată însă în interogările complexe, în care liniile tabelor temporare reprezintă nu tupluri, ci grupuri de tupluri. Practic, în loc de a compara două subconsultări în clauza HAVING (clauză, care, la rândul său, presupune GROUP BY), includem gruparea în subconsultările din FROM, iar comparația o facem între tuplurile tabelor temporare ad-hoc.

Care sunt valorile facturate și încasate, precum și situația ("fără nici o încasare", "încasată parțial" sau "încasată total") pentru fiecare factură ?

```

SELECT VINZARI.NrFact, Facturat, COALESCE(Incasat,0) AS Incasat,
      Facturat - COALESCE(Incasat,0) AS Diferenta,
      CASE
      WHEN COALESCE(Incasat,0) = 0 THEN 'Fara nici o incasare'
      WHEN Facturat > COALESCE(Incasat,0) THEN 'Incasata partial'
      ELSE 'INCASATA TOTAL'
      END AS Situatiune
FROM (
      SELECT NrFact, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Facturat
      FROM liniifact lf INNER JOIN produse p ON lf.CodPr = p.CodPr
      GROUP BY NrFact
      ) VINZARI
LEFT OUTER JOIN
      (
      SELECT NrFact, SUM(Transa) AS Incasat
      FROM incasfact
      GROUP BY NrFact

```

) INCASARI

ON VINZARI.NrFact = INCASARI.NrFact

În clauza FROM a frazei SELECT principale au fost definite două tabele temporare, VINZARI și INCASARI. Prima conține valoarea totală a fiecărei facturi, în timp ce a doua valoarea încasată. Aceste două tabele sunt jonctionate extern (pentru a include și facturile fără nicio încasare) după atributul NrFact – vezi figura 9.26.

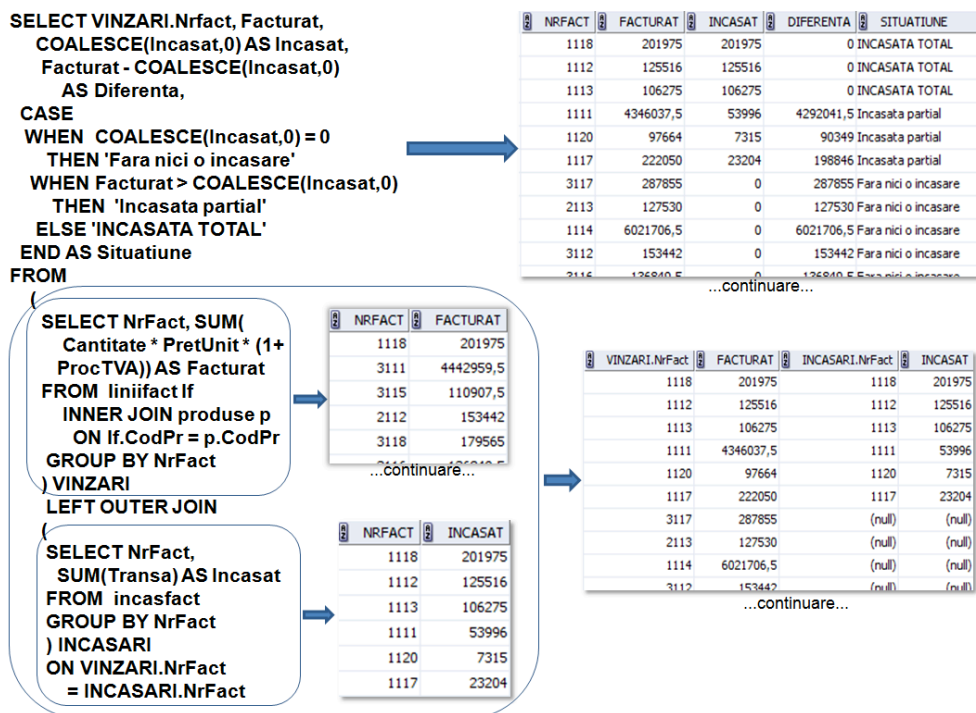


Figura 9.26. Subconsultări în clauza FROM

Să se obțină sporurile de noapte pentru al doilea trimestru al anului 2007, atât lunar, cât și cumulat.

SELECT sl1.Marca, NumePren,
COALESCE(sl1.SpN,0) **AS** Sp_N_Apr,
COALESCE(sl2.SpN,0) **AS** Spor_N_Mai,
COALESCE(sl3.SpN,0) **AS** Spor_N_Iun ,
COALESCE(sl1.SpN,0) + **COALESCE**(sl2.SpN,0)+**COALESCE**(sl3.SpN,0)
AS Sp_Noapte_Trim2
FROM
 (**SELECT** p.Marca, NumePren, SporNoapte **AS** SpN

```

FROM personal2 p LEFT OUTER JOIN sporuri s
    ON p.Marca=s.Marca AND An=2007 AND Luna=4
) s11 INNER JOIN
(SELECT p.Marca, SporNoapte AS SpN
FROM personal2 p LEFT OUTER JOIN sporuri s ON p.Marca=s.Marca
    AND An=2007 AND Luna=5
) s12 ON s11.Marca=s12.Marca
INNER JOIN
(SELECT p.Marca, SporNoapte AS SpN
FROM personal2 p
    LEFT OUTER JOIN sporuri s ON p.Marca=s.Marca
    AND An=2007 AND Luna=6
) s13 ON s11.Marca=s13.Marca
ORDER BY NumePren

```

Clauza FROM principală “calculează” trei tabele temporare ce conțin sporurile de noapte pe lunile aprilie (SL1), mai (SL2) și iunie (SL3) 2005 ale fiecărui angajat (indiferent de data angajării acestuia). Cele trei tabele sunt jonctionate după atributul Marca – vezi figura 9.27.

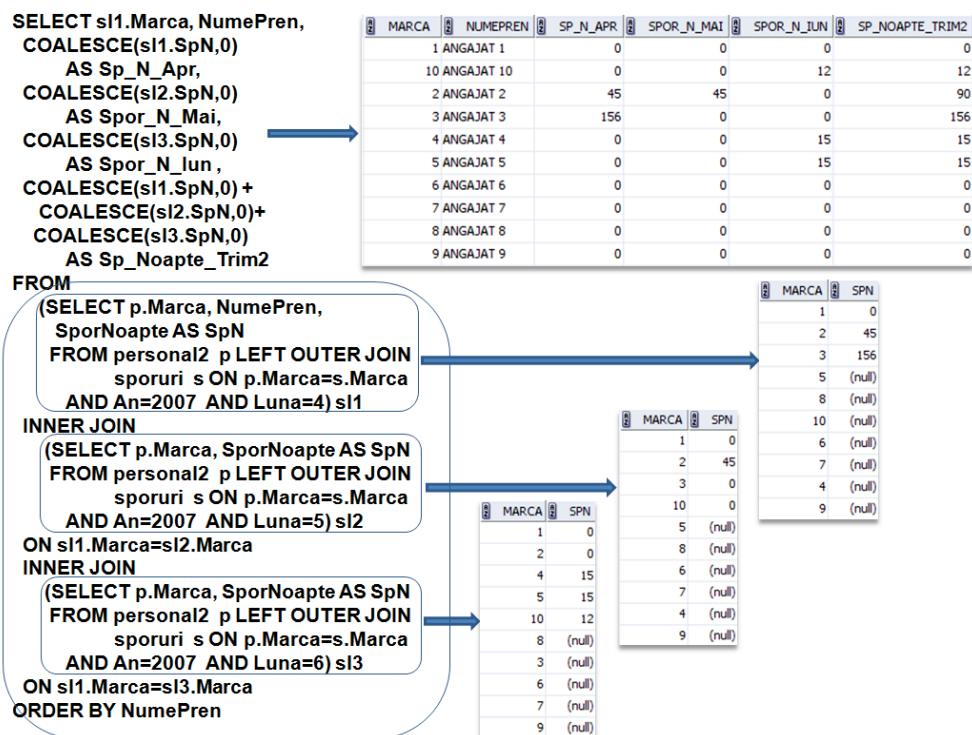


Figura 9.27. Sporuri de noapte pe trimestrul II în 2007

Să se afișeze câte facturi sunt: neîncasate deloc, încasate parțial și încasate total ?

```

SELECT
    CASE
        WHEN COALESCE(Incasat,0) = 0 THEN 'Fara nici o incasare'
        WHEN Facturat > COALESCE(Incasat,0) THEN 'Incasata partial'
        ELSE 'INCASATA TOTAL'
    END AS Situatiune, COUNT(*) AS Nr
FROM (SELECT NrFact, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Facturat
      FROM liniifact LF INNER JOIN produse P ON LF.CodPr = P.CodPr
      GROUP BY NrFact
    ) VINZARI
LEFT OUTER JOIN
    (SELECT NrFact, SUM(Transa) AS Incasat
     FROM incasfact
     GROUP BY NrFact
    ) INCASARI
ON VINZARI.NrFact = INCASARI.NrFact
GROUP BY
    CASE
        WHEN COALESCE(Incasat,0) = 0 THEN 'Fara nici o incasare'
        WHEN Facturat > COALESCE(Incasat,0) THEN 'Incasata partial'
        ELSE 'INCASATA TOTAL'
    END
END

```

Joncțiunea externă la stânga dintre VÎNZĂRI și ÎNCASĂRI este completată de o structură CASE.

Să se determine discountul financiar pentru fiecare tranșă de factură încasată, știind că acesta se acordă pentru clienții care plătesc rapid facturile, și se calculează pe tranșe, astfel:

- pentru tranșele încasate în maxim 9 zile de la data facturării, procentul este de 10%;
- pentru tranșele încasate în intervalul 10-12 zile de la data facturării, procentul este de 7%;
- pentru tranșele încasate în intervalul 13-15 zile de la data facturării, procentul este de 5%;
- pentru tranșele încasate în intervalul 16-18 zile de la data facturării, procentul este de 3%;
- pentru tranșele încasate după 19 zile nu se acordă niciun bonus.

Trebuie să recunosc că găsesc problema tentantă și pentru că pot exemplifica includerea, pe mai multe niveluri, a subconsultărilor în clauza FROM. În plus, pentru aceeași factură pot exista oricâte tranșe de încasare, fiecare tranșă fiind

subiect de discount financiar în funcție de momentul încasării. Începem cu o interogarea pregătitoare:

```

SELECT NrFact, DataFact, Facturat, DataInc, Incasat, NrZile, Procent_DsentFin,
       Incasat * Procent_DsentFin AS Discount_Fin
FROM
  (SELECT vinzari.NrFact, DataFact, Facturat,
        COALESCE(DataInc, CURRENT_DATE) AS DataInc
    ,COALESCE(Incasat,0) AS Incasat,
    TRUNC(COALESCE(DataInc, CURRENT_DATE) - DataFact,0)
        AS NrZile,
    CASE
    WHEN TRUNC(COALESCE(DataInc, CURRENT_DATE) - DataFact,0)
        < 10 THEN 0.10
    WHEN TRUNC(COALESCE(DataInc, CURRENT_DATE) - DataFact,0)
        < 13 THEN 0.07
    WHEN TRUNC(COALESCE(DataInc, CURRENT_DATE) - DataFact,0)
        < 16 THEN 0.05
    WHEN TRUNC(COALESCE(DataInc, CURRENT_DATE) - DataFact,0)
        < 19 THEN 0.03
    ELSE 0
    END AS Procent_DsentFin
  FROM
    (
    SELECT f.NrFact, DataFact,
          ROUND(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) AS Facturat
    FROM facturi f
      INNER JOIN liniifact lf ON f.NrFact=lf.NrFact
      INNER JOIN produse p ON lf.CodPr = p.CodPr
    GROUP BY f.NrFact, DataFact
    ) VINZARI
    LEFT OUTER JOIN

```

```

(
    SELECT NrFact, DataInc, ROUND(SUM(Transa),0) AS Incasat
    FROM incasfact INNER JOIN incasari ON incasfact.CodInc=incasari.CodInc
    GROUP BY NrFact, DataInc
) INCASARI
    ON VINZARI.NrFact = INCASARI.NrFact
ORDER BY NrFact, DataInc
) x4

```

Rezultatul este promițător (primele linii din rezultat se găsesc în figura 9.28). Atât tabela ad-hoc VINZARI, cât și INCASARI prezintă atributele dedicate datei facturării și datei încasării tranșei respective. Diferența dintre cele două date reprezintă numărul de zile în funcție de care se acordă discountul.

| NrFact | DataFact | Facturat | DataInc | Incasat | NrZile | Procent_DscntFin | Discount_Fin |
|--------|------------|----------|------------|---------|--------|------------------|--------------|
| 1111 | 01-08-2007 | 4346038 | 15-08-2007 | 53996 | 14 | 0,05 | 2699,8 |
| 1112 | 01-08-2007 | 125516 | 15-08-2007 | 125516 | 14 | 0,05 | 6275,8 |
| 1113 | 01-08-2007 | 106275 | 17-08-2007 | 106275 | 16 | 0,03 | 3188,25 |
| 1114 | 01-08-2007 | 6021707 | 12-03-2008 | 0 | 224 | 0 | 0 |
| 1115 | 02-08-2007 | 151238 | 12-03-2008 | 0 | 223 | 0 | 0 |
| 1116 | 02-08-2007 | 126713 | 12-03-2008 | 0 | 223 | 0 | 0 |
| 1117 | 03-08-2007 | 222050 | 16-08-2007 | 9754 | 13 | 0,05 | 487,7 |
| 1117 | 03-08-2007 | 222050 | 17-08-2007 | 9754 | 14 | 0,05 | 487,7 |
| 1117 | 03-08-2007 | 222050 | 18-08-2007 | 3696 | 15 | 0,05 | 184,8 |
| 1118 | 04-08-2007 | 201975 | 15-08-2007 | 101975 | 11 | 0,07 | 7138,25 |
| 1118 | 04-08-2007 | 201975 | 16-08-2007 | 100000 | 12 | 0,07 | 7000 |
| 1119 | 07-08-2007 | 5774499 | 12-03-2008 | 0 | 218 | 0 | 0 |
| 1120 | 07-08-2007 | 97664 | 16-08-2007 | 7315 | 9 | 0,1 | 731,5 |
| 1121 | 07-08-2007 | 433838 | 12-03-2008 | 0 | 218 | 0 | 0 |

Figura 9.28. Calcularea discountului financiar pe fiecare tranșă de încasare

Răspunsul exact la problema formulată presupune gruparea după numărul facturii și însumarea atât a tranșelor încasate, cât și a discounturilor financiare pentru fiecare tranșă. Atenție, însă, nu trebuie însumată și valoarea totală a facturii, așa încât interogarea va avea forma:

⁴ Sintaxa nu este acceptată de SQL Server din cauza funcțiilor CURRENT_DATE și TRUNC.

```

SELECT NrFact, DataFact, Facturat, SUM(Incasat) AS Total_Incasat,
      SUM(Discount_Fin) AS Total_Discount_Fin
FROM
(
  SELECT NrFact, DataFact, Facturat, DataInc, Incasat,
        NrZile, Procent_DscentFin,
        ROUND(Incasat * Procent_DscentFin,0) AS Discount_Fin
  FROM
    (SELECT vinzari.NrFact, DataFact, Facturat,
      COALESCE(DataInc, CURRENT_DATE) AS DataInc
    ,COALESCE(Incasat,0) AS Incasat,
      TRUNC(COALESCE(DataInc, CURRENT_DATE) - DataFact,0)
      AS NrZile,
      CASE
        WHEN TRUNC(COALESCE(DataInc, CURRENT_DATE) -
          DataFact,0) <= 9 THEN 0.10
        WHEN TRUNC(COALESCE(DataInc, CURRENT_DATE) -
          DataFact,0) <= 12 THEN 0.07
        WHEN TRUNC(COALESCE(DataInc, CURRENT_DATE) -
          DataFact,0) <= 15 THEN 0.05
        WHEN TRUNC(COALESCE(DataInc, CURRENT_DATE) -
          DataFact,0) <= 18 THEN 0.03
        ELSE 0
      END AS Procent_DscentFin
    FROM
      (
        SELECT f.NrFact, DataFact, ROUND(SUM(
          Cantitate * PretUnit * (1+ProcTVA)),0) AS Facturat
        FROM facturi f
          INNER JOIN liniifact lf ON f.NrFact=lf.NrFact
          INNER JOIN produse p ON lf.CodPr = p.CodPr
        GROUP BY f.NrFact, DataFact
      ) VINZARI
      LEFT OUTER JOIN
      (
        SELECT NrFact, DataInc, ROUND(SUM(Transa),0) AS Incasat
        FROM incasfact INNER JOIN incasari
          ON incasfact.CodInc=incasari.CodInc
        GROUP BY NrFact, DataInc
      ) INCASARI
    )
  )

```

ON VINZARI.NrFact = INCASARI.NrFact

ORDER BY NrFact, DataInc

) X

) Y

GROUP BY NrFact, DataFact, Facturat

ORDER BY 1

Ușoarele diferențe de la însumare (vezi figura 9.29 comparativ cu figura 9.28) se datorează rotunjirilor (funcția ROUND). Probabil că cea mai importantă factură din rezultat este 1117 care are trei tranșe de încasare, fiecare cu discountul său.

| RZ | NRFACT | RZ | DATAFACT | RZ | FACTURAT | RZ | TOTAL_INCASAT | RZ | TOTAL_DISCOUNT_FIN |
|----|--------|----|------------|----|----------|----|---------------|----|--------------------|
| | 1111 | | 01-08-2007 | | 4346038 | | 53996 | | 2700 |
| | 1112 | | 01-08-2007 | | 125516 | | 125516 | | 6276 |
| | 1113 | | 01-08-2007 | | 106275 | | 106275 | | 3188 |
| | 1114 | | 01-08-2007 | | 6021707 | | 0 | | 0 |
| | 1115 | | 02-08-2007 | | 151238 | | 0 | | 0 |
| | 1116 | | 02-08-2007 | | 126713 | | 0 | | 0 |
| | 1117 | | 03-08-2007 | | 222050 | | 23204 | | 1161 |
| | 1118 | | 04-08-2007 | | 201975 | | 201975 | | 14138 |
| | 1119 | | 07-08-2007 | | 5774499 | | 0 | | 0 |
| | 1120 | | 07-08-2007 | | 97664 | | 7315 | | 732 |
| | 1121 | | 07-08-2007 | | 4737838 | | 0 | | 0 |

Figura 9.29. Totalizarea discountului financiar pe fiecare factură

Cu ocazia aceasta descoperim că, dacă eram așa de generoși, ar fi trebuit să restituim niște bani unor clienți.

Să se calculeze penalizările pentru întârzierea la plată a facturilor, știind că și acestea se calculează pentru fiecare zi de întârziere, pe tranșe, astfel:

- pentru tranșele neîncasate în maxim 10 zile de la data facturării, nu se calculează penalizări;
- pentru tranșele neîncasate în intervalul 11-12 zile de la data facturării, procentul de penalitate este 0,05% din valoarea neîncasată, pentru fiecare zi de întârziere;
- pentru tranșele neîncasate în intervalul 13-14 zile de la data facturării, procentul de penalitate este 0,1% din valoarea neîncasată, pentru fiecare zi de întârziere;
- pentru tranșele încasate în intervalul 15-20 zile de la data facturării, procentul este de 0,2% din valoarea neîncasată pentru fiecare zi de întârziere;
- pentru tranșele încasate în intervalul 20-40 zile de la data facturării, procentul este de 0,3% din valoarea neîncasată pentru fiecare zi de întârziere;

- pentru tranșele încasate cu întârziere de peste 40 zile de la data facturării, procentul este de 0,5% din valoarea neîncasată pentru fiecare zi de întârziere;

După cum vedeți, acum am trecut în extrema cealaltă, a cărpănoșilor. Necazul este că, pentru această problemă nu suntem în stare să formulăm un răspuns corect, întrucât o factură poate avea mai multe tranșe de încasare, iar penalitatea depinde de restul de încasat din momentul fiecărei tranșe, rest calculat prin diferența dintre valoarea totală a facturii și valoarea cumulată a tranșelor de încasare pentru factura respectivă până în momentul respectiv. Vom avea ceva mai mult succes în capitolul următor (cap. 10) și în capitolul dedicat opțiunilor OLAP (cap. 11).

9.4.2. Un artificiu „proprietar” – ROWNUM din Oracle

Cu atâtea kilograme de SQL înghițite, putem să realizăm că interogarea:

```
SELECT DISTINCT PretUnit
FROM liniifact
ORDER BY PretUnit DESC
```

extrage în ordine descrescătoare toate valorile (distincte) ale prețului unitar la care au fost vândute produsele. Adăugăm, discret, o clauză proprietară Oracle, ROWNUM, care nu face mare lucru – atribuie un număr fiecărei linii din rezultat:

```
SELECT DISTINCT PretUnit, ROWNUM
FROM liniifact
ORDER BY PretUnit DESC
```

Din figura 9.30 aflăm o veste bună – fiecare linie prezintă un număr – și o veste rea – numărul respectiv nu ne folosește la nimic.

| PRETUNIT | ROWNUM |
|----------|--------|
| 7064 | 9 |
| 7064 | 21 |
| 7064 | 39 |
| 7060 | 3 |
| 7060 | 25 |
| 7060 | 43 |
| 6300 | 19 |
| 6300 | 38 |
| 6300 | 56 |
| 1705 | 8 |
| 1410 | 18 |
| 1410 | 37 |

Figura 9.30. Clauza ROWNUM (aplicată greșit)

Apelând la o subconsultare în clauza FROM, lucrurile devin ceva mai interesante:

```

SELECT t.*, ROWNUM
FROM
    (SELECT DISTINCT PretUnit
     FROM liniifact
     ORDER BY PretUnit DESC
    ) t

```

Tabela ad-hoc T obținută prin subconsultare conține valorile distincte ale prețului unitar, ordonate descrescător. Acum ROWNUM va alocă valoarea 1 pentru cel mai mare preț unitar, valoarea 2 pentru următorul s.a.m.d – vezi figura 9.31.

| PRETUNIT | ROWNUM |
|----------|--------|
| 7064 | 1 |
| 7060 | 2 |
| 6300 | 3 |
| 1705 | 4 |
| 1410 | 5 |
| 1120 | 6 |
| 1100 | 7 |
| 1090 | 8 |
| 1070 | 9 |
| 1050 | 10 |
| 1030 | 11 |
| 1000 | 12 |
| 975 | 13 |
| 950 | 14 |
| 930 | 15 |
| 925 | 16 |
| 750 | 17 |
| 700 | 18 |

Figura 9.31. Clauza ROWNUM (aplicată ceva mai bine)

În continuare, dacă dorim să aflăm cele mai mari cinci prețuri unitare din LINIIFACT, aplicăm condiția de filtrare asupra coloanei ROWNUM:

```

SELECT t.*, ROWNUM
FROM
    (SELECT DISTINCT PretUnit
     FROM liniifact
     ORDER BY PretUnit DESC) t
WHERE ROWNUM <= 5

```

Atenție, însă, mânați de dorință, am fi tentați să credem că al cincilea (în ordine descrescătoare) preț unitar s-ar putea afla eliminând din interogarea precedentă semnul <:

```

SELECT t.*, ROWNUM
FROM
    (SELECT DISTINCT PretUnit
    FROM liniifact
    ORDER BY PretUnit DESC) t
WHERE ROWNUM = 5

```

Rezultatul nu are nici o linie, întrucât atribuirea numărului de linie se face *după* filtrare. Necazul se păstrează și când includem interogarea de mai sus în clauza FROM a unei consultări:

```

SELECT *
FROM
    (SELECT t.*, ROWNUM
    FROM
        (SELECT DISTINCT PretUnit
        FROM liniifact
        ORDER BY PretUnit DESC) t
    )
WHERE ROWNUM=5

```

Ne amăgim în van, întrucât ROWNUM-ul din clauza WHERE se aplică la nivelul noii interogări (nu e ROWNUM-ul pentru T). Din fericire, reparația este minoră: în tabela ad-hoc T redenumim coloana *ROWNUM* în *Nr* și aplicăm condiția de filtrare la noul atribut „calculat”:

```

SELECT *
FROM (SELECT t.*, ROWNUM AS Nr
    FROM
        (SELECT DISTINCT PretUnit
        FROM liniifact
        ORDER BY PretUnit DESC) t
    )
WHERE Nr=5

```

Care sunt cele mai mari cinci prețuri unitare de vânzare, produsele și facturile în care apar cele cinci prețuri maxime ?

Această problemă la a cărei premieră am avut ceva frisoane (paragraful 9.2) se rezolvă „cu vorba bună” astfel:

```

SELECT NrFact, DenPr, PretUnit
FROM liniifact INNER JOIN produse ON liniifact.CodPr=produse.CodPr
WHERE PretUnit IN
    (SELECT PretUnit
    FROM
        (SELECT t.*, ROWNUM
        FROM

```

```

        (SELECT DISTINCT PretUnit
        FROM liniifact
        ORDER BY PretUnit DESC
        ) t
    WHERE ROWNUM <=5
)
)

```

9.4.3. Comparații mutate din HAVING în clauza FROM

Firește, ne putem gândi de acum cu mai multă simpatie și la problemele ce necesitau comparații în clauza HAVING.

Care sunt zilele în care s-au emis mai multe facturi decât pe 2 august 2007 ?

```

SELECT Zi, Nr_Facturilor
FROM
    (SELECT DataFact AS Zi, COUNT(NrFact) AS Nr_Facturilor
    FROM facturi
    GROUP BY DataFact ) fact_zile
INNER JOIN
    (SELECT COUNT(NrFact) AS NrFact_2Aug
    FROM facturi
    WHERE DataFact = DATE'2007-08-02') fact_2aug
ON Nr_Facturilor > NrFact_2Aug

```

Care este ziua în care s-au emis cele mai multe facturi ?

Din păcate (sau, din fericire !), soluția:

```

SELECT temp1.*
FROM
    (SELECT DataFact, COUNT(Nrfact) AS Nr
    FROM facturi
    GROUP BY DataFact) temp1,
    (SELECT DataFact, COUNT(Nrfact) AS Nr
    FROM facturi
    GROUP BY DataFact) temp2
WHERE temp1.Nr >= ALL (SELECT Nr FROM temp2)

```

nu funcționează – vezi figura 9.32. Aceasta înseamnă că o tabelă temporară definită pe baza unei subconsultări în clauza FROM nu este recunoscută într-o subconsultare din clauza WHERE sau HAVING. Se poate, totuși, utiliza cu succes varianta:

```

SELECT DataFact, COUNT(*) AS Nr_Facturi
FROM facturi

```

```

GROUP BY DataFact
HAVING COUNT(*) = (SELECT MAX(Nr)
                    FROM (   SELECT DataFact, COUNT(NrFact) AS Nr
                              FROM facturi
                              GROUP BY DataFact
                              ) TEMP1 )

```

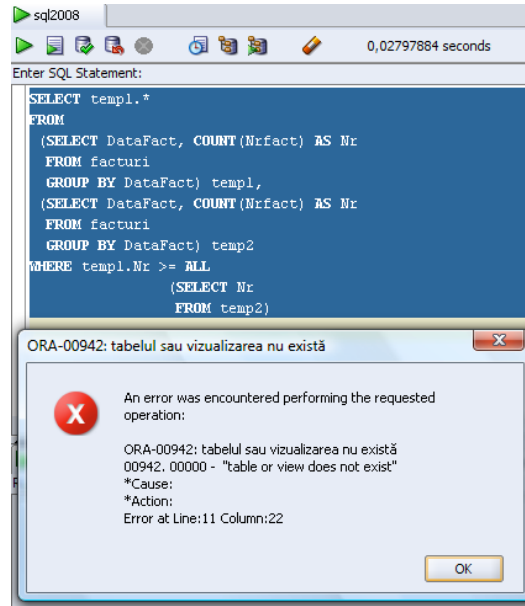


Figura 9.32. Referirea eronată a rezultatului unei subconsultări din clauza FROM

În subconsultarea din clauza HAVING a fost definită tabela „intermediară” TEMP1 al cărei atribut Nr este folosit în funcția MAX din clauza SELECT. Iar dacă tot vrem să ne complicăm viața, am putea folosi și:

```

SELECT *
FROM (SELECT DataFact, COUNT(NrFact) AS Nr
      FROM facturi
      GROUP BY DataFact) zile_fact
WHERE Nr =
      (SELECT MAX(zile_fact1.Nr)
      FROM (SELECT DataFact, COUNT(NrFact) AS Nr
            FROM facturi
            GROUP BY DataFact
            ) zile_fact1
      INNER JOIN
      (SELECT DataFact, COUNT(NrFact) AS Nr

```

```

FROM facturi
GROUP BY DataFact) zile_fact2
ON zile_fact1.Nr > zile_fact2.Nr )

```

Care este clientul care a cumpărat cele mai multe produse ?

```

SELECT DenCl, COUNT(DISTINCT CodPr) AS Nr_Produse
FROM clienti c
    INNER JOIN facturi f ON c.CodCl=f.CodCl
    INNER JOIN liniifact lf ON f.NrFact=lf.NrFact
GROUP BY DenCl
HAVING COUNT(DISTINCT CodPr) =
    (SELECT MAX(Nr)
    FROM (SELECT CodCl, COUNT(DISTINCT CodPr) AS Nr
    FROM facturi f INNER JOIN liniifact lf
    ON f.NrFact=lf.NrFact
    GROUP BY CodCl
    ) TEMP1
    )

```

Care este județul în care berea s-a vândut cel mai bine ?

Avem soluții pe-alese. Iată doar două:

```

SELECT Judet, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari_Bere
FROM judete j
    INNER JOIN coduri_postale cp ON j.Jud=cp.Jud
    INNER JOIN clienti c ON cp.CodPost=c.CodPost
    INNER JOIN facturi f ON c.CodCl=f.CodCl
    INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
    INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE Grupa='Bere'
GROUP BY Judet
HAVING SUM(Cantitate * PretUnit * (1+ProcTVA)) =
    (SELECT MAX(Vinzari)
    FROM (SELECT SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari
    FROM coduri_postale cp
        INNER JOIN clienti c ON cp.CodPost=c.CodPost
        INNER JOIN facturi f ON c.CodCl=f.CodCl
        INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
        INNER JOIN produse p ON lf.CodPr=p.CodPr
    WHERE Grupa='Bere'
    GROUP BY Jud)
    TEMP1)

```

și

```

SELECT bere_judete.*

```

```

FROM
    (SELECT Judet, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari_Bere
    FROM judete j
        INNER JOIN coduri_postale cp ON j.Jud=cp.Jud
        INNER JOIN clienti c ON cp.CodPost=c.CodPost
        INNER JOIN facturi f ON c.CodCl=f.CodCl
        INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
        INNER JOIN produse p ON lf.CodPr=p.CodPr
    WHERE Grupa='Bere'
    GROUP BY Judet
    ) Bere_Judete INNER JOIN
    (SELECT MAX(Vinzari) AS Max_Bere
    FROM
    (SELECT SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari
    FROM coduri_postale cp
        INNER JOIN clienti c ON cp.CodPost=c.CodPost
        INNER JOIN facturi f ON c.CodCl=f.CodCl
        INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
        INNER JOIN produse p ON lf.CodPr=p.CodPr
    WHERE Grupa='Bere'
    GROUP BY Jud)
    ) Bere_Jud_Max ON Vinzari_Bere = Max_Bere

```

Cea de-a doua soluție funcționează în DB2 și Oracle, însă în PostgreSQL și MS SQL Server provoacă o eroare, după cum se observă în figura 9.33. Din mesaj deducem că una dintre subconsultări ar trebui să aibă un alias. Este tocmai cea pe care am pus-o în evidență.

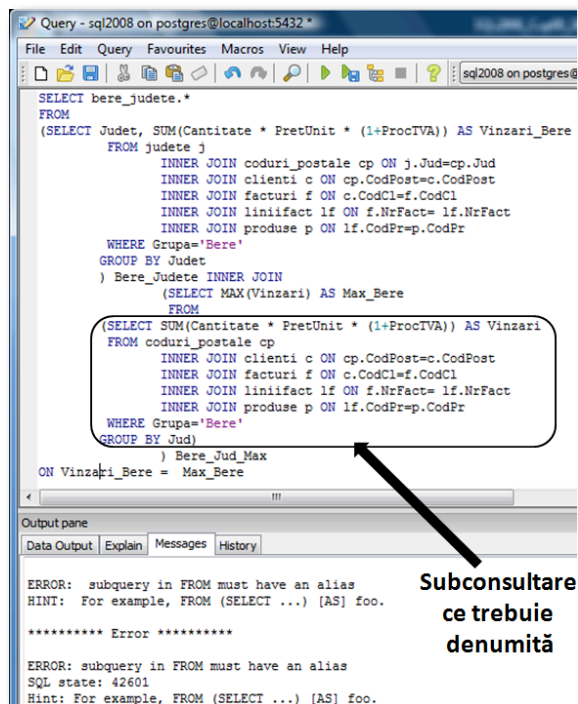


Figura 9.33. O situație în care aliasul unei subconsultări este obligatoriu în PostgreSQL

Acelei subconsultări îi dăm numele SUBCONSULTARE_DENUMITĂ:

SELECT bere_judete.*

FROM (SELECT Judet, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari_Bere

FROM judete j

INNER JOIN coduri_postale cp ON j.Jud=cp.Jud

INNER JOIN clienti c ON cp.CodPost=c.CodPost

INNER JOIN facturi f ON c.CodCl=f.CodCl

INNER JOIN liniifact lf ON f.NrFact= lf.NrFact

INNER JOIN produse p ON lf.CodPr=p.CodPr

WHERE Grupa='Bere'

GROUP BY Judet

) bere_judete **INNER JOIN**

(SELECT MAX(Vinzari) AS Max_Bere

FROM

(SELECT SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari

FROM coduri_postale cp

INNER JOIN clienti c ON cp.CodPost=c.CodPost

INNER JOIN facturi f ON c.CodCl=f.CodCl

INNER JOIN liniifact lf ON f.NrFact= lf.NrFact


```

        INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE Grupa='Bere'
GROUP BY Jud) Subconsultare_Denumita
) Bere_Jud_Max

```

```
ON Vinzari_Bere = Max_Bere
```

și lucrurile sunt acum în regulă. Cu atât mai mult cu cât interogarea funcționează și în celelalte trei servere BD. Nu ne rămâne decât să ne întrebăm de ce am dat peste eroarea aceasta așa de târziu. Răspunsul ține de faptul că abia acum ne-am găsit să nu denumim o subconsultare în clauza FROM.

Care sunt clienții cu valoarea vânzărilor peste medie ?

După cum am stabilit în paragraful 9.3.1, ne interesează media vânzărilor pe client calculată prin împărțirea vânzărilor totale la numărul clienților cărora li s-a trimis măcar o factură.

Soluția 1:

```

SELECT DenCl, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari
FROM clienti c
        INNER JOIN facturi f ON c.CodCl=f.CodCl
        INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
        INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY DenCl
HAVING SUM(Cantitate * PretUnit * (1+ProcTVA))
    >=
    (SELECT Vinzari / NrCienti
FROM
        (SELECT SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari
FROM liniifact lf INNER JOIN produse p
        ON lf.CodPr=p.CodPr
        ) temp1,
    (SELECT COUNT(DISTINCT CodCl) AS NrCienti
FROM facturi
        ) temp2
    )

```

Soluția 2:

```

SELECT DenCl, VINZ_CL.Vinzari
FROM
        (SELECT DenCl, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari
FROM clienti c
        INNER JOIN facturi f ON c.CodCl=f.CodCl
        INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
        INNER JOIN produse p ON lf.CodPr=p.CodPr

```

```

        GROUP BY DenCl
    ) VINZ_CL
INNER JOIN
    (SELECT DISTINCT Vinzari / NrClienti AS Medie_Vinz
    FROM
        (SELECT SUM(Cantitate * PretUnit * (1+ProcTVA))
        AS Vinzari
        FROM liniifact lf, produse p
        WHERE lf.CodPr=p.CodPr) X,
        (SELECT COUNT(DISTINCT CodCl) AS NrClienti
        FROM facturi) Y
    ) MEDIE_VINZ
ON VINZ_CL.Vinzari >= MEDIE_VINZ.Medie_Vinz

```

Care este factura cu cea mai mică valoare peste cea medie ?

```

SELECT NrFact, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS ValFact
FROM liniifact lf
    INNER JOIN produse p ON p.CodPr = lf.CodPr
GROUP BY NrFact
HAVING SUM(Cantitate * PretUnit * (1+ProcTVA)) =
    (SELECT MIN(ValFact)
    FROM (
        SELECT SUM(Cantitate * PretUnit * (1+ProcTVA)) AS ValFact
        FROM liniifact lf
            INNER JOIN produse p ON p.CodPr = lf.CodPr
        GROUP BY NrFact
    ) TEMP1
WHERE ValFact >
    (SELECT Vinzari / NrFacturi AS ValMedie
    FROM
        (SELECT SUM(Cantitate * PretUnit * (1+ProcTVA))
        AS Vinzari,
        COUNT(DISTINCT NrFact) AS NrFacturi
        FROM liniifact lf
            INNER JOIN produse p ON p.CodPr = lf.CodPr
        ) TEMP1
    )
)

```

Apelăm și la o soluție care să afișeze toate facturile cu valoarea peste medie, în ordinea crescătoare a acestei valori. Prima factură din figura 9.34 - va fi răspunsul la întrebarea formulată.

```

SELECT NrFact, ValFact, ValMedie
FROM
    (SELECT NrFact, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS ValFact
     FROM liniifact lf INNER JOIN produse p ON p.CodPr = lf.CodPr
     GROUP BY NrFact
    ) TEMP1
INNER JOIN
    (SELECT ROUND(Vinzari / NrFacturi,0) AS ValMedie
     FROM
        (SELECT SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari,
         COUNT(DISTINCT NrFact) AS NrFacturi
         FROM liniifact lf
         INNER JOIN produse p ON p.CodPr = lf.CodPr
        ) MEDIE1
    ) MEDIE1
ON ValFact > ValMedie
ORDER BY ValFact

```

| R | NRFACT | R | VALFACT | R | VALMEDIE |
|---|--------|---|-----------|---|----------|
| 1 | 1111 | 2 | 4346037,5 | 3 | 1756083 |
| 2 | 2111 | 2 | 4442959,5 | 3 | 1756083 |
| 3 | 3111 | 2 | 4442959,5 | 3 | 1756083 |
| 4 | 1121 | 2 | 4737838 | 3 | 1756083 |
| 5 | 2121 | 2 | 4741271,5 | 3 | 1756083 |
| 6 | 1119 | 2 | 5774498,5 | 3 | 1756083 |
| 7 | 3119 | 2 | 5819668 | 3 | 1756083 |
| 8 | 2119 | 2 | 5819668 | 3 | 1756083 |
| 9 | 1114 | 2 | 6021706,5 | 3 | 1756083 |

Figura 9.34. Facturile cu valori peste medie

Pentru a răspunde exact la întrebarea formulată trebuie să extragem doar prima linie. Trecem sub tăcere variantele bazate pe LIMIT 1 sau TOP 1 și ne oprim doar la una generală:

```

SELECT NrFact, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS ValFact
FROM liniifact lf INNER JOIN produse p ON p.CodPr = lf.CodPr
GROUP BY NrFact
HAVING SUM(Cantitate * PretUnit * (1+ProcTVA)) =
    (SELECT MIN(ValFact)
     FROM
        (SELECT NrFact, ValFact, ValMedie
         FROM
            (SELECT NrFact, SUM(Cantitate * PretUnit * (1+ProcTVA))

```

```

        AS ValFact
    FROM liniifact lf INNER JOIN produse p
        ON p.CodPr = lf.CodPr
    GROUP BY NrFact ) TEMP1
    INNER JOIN
        (SELECT ROUND(Vinzari / NrFacturi,0) AS ValMedie
        FROM
            (SELECT SUM(Cantitate * PretUnit *
                (1+ProcTVA)) AS Vinzari,
                COUNT(DISTINCT NrFact)
                AS NrFacturi
            FROM liniifact lf INNER JOIN produse p
                ON p.CodPr = lf.CodPr
            ) MEDIE1
        ) MEDIE2
        ON ValFact > ValMedie
    ) interogarea_dinainte )

```

Care este județul cu vânzări imediat superioare județului Neamț ?

```

SELECT Judet, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari
FROM judete j
    INNER JOIN coduri_postale cp ON j.Jud=cp.Jud
    INNER JOIN clienti c ON cp.CodPost=c.CodPost
    INNER JOIN facturi f ON c.CodCl=f.CodCl
    INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
    INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY Judet
HAVING SUM(Cantitate * PretUnit * (1+ProcTVA)) =
    (SELECT MIN ( VINZ_JUD1.Vinzari)
    FROM
        (SELECT Judet, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari
        FROM judete j
            INNER JOIN coduri_postale cp ON j.Jud=cp.Jud
            INNER JOIN clienti c ON cp.CodPost=c.CodPost
            INNER JOIN facturi f ON c.CodCl=f.CodCl
            INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
            INNER JOIN produse p ON lf.CodPr=p.CodPr
        GROUP BY Judet
        ) VINZ_JUD1
    INNER JOIN
        (SELECT Judet, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari

```

```

FROM judete j
    INNER JOIN coduri_postale cp ON j.Jud=cp.Jud
    INNER JOIN clienti c ON cp.CodPost=c.CodPost
    INNER JOIN facturi f ON c.CodCl=f.CodCl
    INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
    INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY Judet          ) VINZ_JUD2
ON VINZ_JUD1.Vinzari > VINZ_JUD2.Vinzari
AND VINZ_JUD2.Judet='Neamt')

```

Este posibil să nu vă fi revenit după trauma din paragraful 9.3.2 pricinuită de soluția problemei *Care este clientul cel mai datornic (ce are cel mai mare rest de plată) ?*

De aceea, mă grăbesc în a formula o soluție care mai diluează din depresie:

```

SELECT DenCl, Vinzari, Incasari, DeIncasat
FROM
    (SELECT FACTURAT.CodCl, Vinzari, COALESCE(Incasari, 0) AS Incasari,
        Vinzari - COALESCE(Incasari, 0) AS DeIncasat
    FROM
        (SELECT CodCl, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari
        FROM facturi f
            INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
            INNER JOIN produse p ON lf.CodPr=p.CodPr
        GROUP BY CodCl
        )
        FACTURAT
        LEFT OUTER JOIN
        (SELECT CodCl, SUM(Transa) AS Incasari
        FROM facturi f
            INNER JOIN incasfact i ON f.NrFact=i.NrFact
        GROUP BY CodCl
        ) INCASAT
        ON FACTURAT.CodCl=INCASAT.CodCl
    ) TEMP1 INNER JOIN
    (SELECT MAX (Vinzari - COALESCE(Incasari, 0)) AS DifMax
    FROM (
        SELECT CodCl, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari
        FROM facturi f
            INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
            INNER JOIN produse p ON lf.CodPr=p.CodPr
        GROUP BY CodCl
        )
        FACTURAT LEFT OUTER JOIN
        (SELECT CodCl, SUM(Transa) AS Incasari

```

```

        FROM facturi f INNER JOIN incasfact i ON f.NrFact=i.NrFact
        GROUP BY CodCl
    ) INCASAT
    ON FACTURAT.CodCl=INCASAT.CodCl
) TEMP2
    ON TEMP1.DeIncasat=TEMP2.DifMax
INNER JOIN clienti
    ON TEMP1.CodCl= clienti.CodCl

```

În interogare obținem o tabelă temporară cu diferența de încasat pe clienți (TEMP1), și o alta ce conține cea mai mare diferență de încasat pentru un client (TEMP2). Cele două tabele sunt jonctionate după diferență și, în final, pentru a afla denumirea clientului, adăugăm în joncțiune tabela CLIEȚI.

9.4.4. Diviziunea relațională *strikes back*

O parte dintre soluțiile SQL de până acum formulate pentru problemele încadrabile diviziunii relaționale (introdusă în paragraful 4.4.7) pot fi simplificate prin folosirea subconsultărilor în clauza FROM. Să începem cu succesiunea pașilor din figura 4.33). Cel puțin două variante putem adăuga la cele existente:

```

SELECT DISTINCT X
FROM rd1
WHERE X NOT IN
    (SELECT DISTINCT PRODUS_CARTEZIAN.X
     FROM (SELECT DISTINCT rd1.X, rd2.Y
           FROM rd1,rd2
          ) PRODUS_CARTEZIAN
     LEFT OUTER JOIN rd1 ON PRODUS_CARTEZIAN.X=rd1.X
                      AND PRODUS_CARTEZIAN.Y=rd1.Y
     WHERE rd1.X IS NULL
    )

```

și:

```

SELECT DISTINCT X
FROM
    (SELECT X, COUNT(Y) AS Nr
     FROM rd1
     GROUP BY X
    ) TEMP1
    INNER JOIN
    (SELECT COUNT(Y) AS Nr
     FROM rd2
    ) TEMP2
    ON TEMP1.Nr=TEMP2.Nr

```

În ultima interogare, tabela temporară ad-hoc TEMP1 conține numărul valorilor lui Y pentru fiecare X din R1, iar TEMP2, numai numărul total al valorilor lui Y din R2 – figura 9.35.

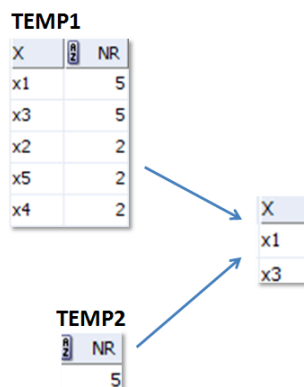


Figura 9.35. Tabelele temporare TEMP1 și TEMP2

Ce produse au fost vândute tuturor clienților ? (paragraful 4.4.7, exemplul 26)

Dintre cele două variante de lucru propuse la începutul acestui paragraf, apelăm (probabil, pe nedrept) la cea de-a doua:

```
SELECT DISTINCT DenPr
FROM (SELECT DenPr, COUNT(DISTINCT CodCl) AS Nr
      FROM produse p INNER JOIN liniifact lf ON p.CodPr=lf.CodPr
      INNER JOIN facturi f ON lf.NrFact=f.NrFact
      GROUP BY DenPr
    ) TEMP1 INNER JOIN (SELECT COUNT(CodCl) AS Nr FROM clienti) TEMP2
ON TEMP1.Nr=TEMP2.Nr
```

În ce zile s-au vândut și produsul cu denumirea "Produs 1" și cel cu denumirea "Produs 2" ? (exemplul 23 din paragraful 4.4.7).

```
SELECT DISTINCT DataFact
FROM
  (SELECT F.DataFact, COUNT(DISTINCT LF.CodPr) AS Nr
   FROM facturi f
   INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
   INNER JOIN produse p ON lf.CodPr=p.CodPr
   WHERE DenPr IN ('Produs 1', 'Produs 2')
   GROUP BY F.DataFact) TEMP1
WHERE Nr = 2
```

TEMP1 conține, pentru fiecare dată calendaristică, numărul de produse, dintre Produs 1 și Produs 2 (1 sau 2) care au fost vândute în ziua respectivă.

Care sunt clienții pentru care există cel puțin câte o factură emisă în fiecare zi cu vânzări din perioada 10-30 septembrie 2007? (exemplu 23, paragraf 4.4.7)

Soluția 1:

```

SELECT DISTINCT DenCl
FROM (
    SELECT clienti.*
    FROM clienti INNER JOIN facturi ON clienti.CodCl=facturi.CodCl
    WHERE DataFact BETWEEN DATE'2007-09-10' AND DATE'2007-09-30'
    ) un_alias
WHERE CodCl NOT IN
(SELECT DISTINCT PRODUS_CARTEZIAN.CodCl
 FROM (
    SELECT DISTINCT rd1.CodCl, rd2.DataFact
    FROM (
        SELECT *
        FROM facturi
        WHERE DataFact BETWEEN DATE'2007-09-10'
        AND DATE'2007-09-30'
    ) rd1 CROSS JOIN (SELECT DISTINCT DataFact
        FROM facturi
        WHERE DataFact BETWEEN
        DATE'2007-09-10' AND
        DATE'2007-09-30') rd2
    ) PRODUS_CARTEZIAN
    LEFT OUTER JOIN (SELECT CodCl, DataFact
        FROM facturi
        WHERE DataFact BETWEEN DATE'2007-09-10'
        AND DATE'2007-09-30' ) rd1
    ON PRODUS_CARTEZIAN.CodCl=rd1.CodCl AND
        PRODUS_CARTEZIAN.DataFact=rd1.DataFact
    WHERE rd1.CodCl IS NULL
    )
)

```

Soluția 2:

```

SELECT DISTINCT DenCl
FROM
    (SELECT CodCl, COUNT(DISTINCT DataFact) AS Nr
    FROM facturi
    WHERE DataFact BETWEEN DATE'2007-09-10' AND DATE'2007-09-30'
    GROUP BY CodCl
    ) TEMP1

```



```

INNER JOIN
    (SELECT COUNT(DISTINCT DataFact) AS Nr
     FROM facturi
     WHERE DataFact BETWEEN DATE'2007-09-10'
                        AND DATE'2007-09-30'
     ) TEMP2
ON TEMP1.Nr=TEMP2.Nr
INNER JOIN clienti
    ON TEMP1.CodCl=clienti.CodCl

```

Ce facturi conțin măcar produsele din factura 1117 ? (exemplu 27, paragraf 4.4.7)

```

SELECT DISTINCT NrFact
FROM
    (
        SELECT NrFact, COUNT(*) AS NrProd
        FROM liniifact
        WHERE CodPr IN
            (SELECT CodPr
             FROM liniifact
             WHERE NrFact = 1117)
        GROUP BY NrFact
    ) T1 INNER JOIN
        ( SELECT COUNT(CodPr) AS NrP1117
          FROM liniifact
          WHERE NrFact = 1117
        ) T2
    ON T1.NrProd = T2.NrP1117

```

T2 conține numărul produselor din factura 1117. T1 conține, pentru fiecare factură, câte produse sunt comune acesteia și facturii 1117. Prin joncțiunea tabelor „ad-hoc” T1 și T2 prin condiția $T1.NrProd = T2.NrP1117$, se extrag acele linii din T1 care au același număr de produse prezente în factura 1117 ca și aceasta (adică factura 1117).

9.4.5. Tabele temporare create ad-hoc

În paragraful 7.3 am formulat o problemă care viza numărul de produse vândute, pe intervale ale prețurilor unitare de ordinul miilor. Schimbăm enunțul astfel:

Să se afle numărul de produse vândute, pe următoarele intervale ale prețului unitar de vânzare:

- între 0 și 500 RON;

- între 501 și 750 RON;
- între 751 și 1000 RON;
- între 1001 și 1500 RON;
- între 1501 și 5000 RON;
- între 5001 și 6000 RON;
- între 6001 și 7000 RON;
- peste 7001 RON.

Intervalele sunt, după cum vedem, neregulate, astfel încât artificiile care ar utiliza funcții TRUNC/ROUND sunt mai greu de folosit. Beneficiind de minunăția includerii subconsultărilor în clauza FROM, putem încerca o variantă intermediară în Oracle astfel:

```
SELECT *
FROM
    (SELECT 0 AS LimInf, 500 AS LimSup FROM dual UNION
    SELECT 501, 750 FROM dual UNION
    SELECT 751, 1000 FROM dual UNION
    SELECT 1001, 5000 FROM dual UNION
    SELECT 5001, 6000 FROM dual UNION
    SELECT 6001, 7000 FROM dual UNION
    SELECT 7001, 99999999 FROM dual
    ) intervale
LEFT OUTER JOIN liniifact ON PretUnit BETWEEN LimInf AND LimSup
ORDER BY LimInf, LimSup, PretUnit
```

Fiecare interval este definit printr-un SELECT în care apar două coloane – limitele inferioare (LimInf) și superioare (LimSup) ale acestuia. Cele șapte SELECT-uri sunt conectate prin operatorul UNION, tabela temporară ad-hoc INTERVALE fiind cea din figura 9.36.

| LIMINF | LIMSUP |
|--------|----------|
| 0 | 500 |
| 501 | 750 |
| 751 | 1000 |
| 1001 | 5000 |
| 5001 | 6000 |
| 6001 | 7000 |
| 7001 | 99999999 |

Figura 9.36. Tabelele temporară (ad-hoc) INTERVALE

Această tabelă este (theta)jonționată extern cu tabela LINIIFACT, în vederea încadrării fiecărui preț unitar în intervalul corespunzător – vezi figura 9.37 (figură care conține doar o parte din liniile rezultatului).

| 1 | LIMINF | 2 | LIMSUP | 3 | NRFACT | 4 | LINIE | 5 | CODPR | 6 | CANTITATE | 7 | PRETUNIT |
|---|--------|---|--------|---|--------|---|--------|---|--------|---|-----------|---|----------|
| | 0 | | 500 | | (null) | | (null) | | (null) | | (null) | | (null) |
| | 501 | | 750 | | 2119 | | 2 | | 3 | | 40 | | 700 |
| | 501 | | 750 | | 3119 | | 2 | | 3 | | 40 | | 700 |
| | 501 | | 750 | | 1119 | | 2 | | 3 | | 40 | | 700 |
| | 501 | | 750 | | 2112 | | 2 | | 3 | | 65 | | 750 |
| | 501 | | 750 | | 1112 | | 2 | | 3 | | 40 | | 750 |
| | 501 | | 750 | | 3112 | | 2 | | 3 | | 65 | | 750 |
| | 751 | | 1000 | | 2115 | | 1 | | 2 | | 110 | | 925 |
| | 751 | | 1000 | | 1115 | | 1 | | 2 | | 150 | | 925 |
| | 751 | | 1000 | | 3115 | | 1 | | 2 | | 110 | | 925 |
| | 751 | | 1000 | | 1116 | | 1 | | 2 | | 125 | | 930 |
| | 751 | | 1000 | | 3118 | | 2 | | 1 | | 120 | | 930 |
| | 751 | | 1000 | | 1118 | | 2 | | 1 | | 150 | | 930 |
| | 751 | | 1000 | | 2118 | | 2 | | 1 | | 120 | | 930 |
| | 751 | | 1000 | | 3116 | | 1 | | 2 | | 135 | | 930 |

Figura 9.37. Încadrarea prețurilor unitare pe intervale (fragment)

În continuare, nu ne rămâne decât să facem gruparea după intervale. Argumentul funcției COUNT trebuie să fie, neapărat, un atribut din tabela LINIIFACT, pentru că, altminteri, la intervalele fără nici un preț ar apărea eronat valoarea 1:

```

SELECT LimInf, LimSup, COUNT(liniifact.PretUnit) AS Nr_produce
FROM
  (SELECT 0 AS LimInf, 500 AS LimSup FROM dual UNION
   SELECT 501, 750 FROM dual UNION
   SELECT 751, 1000 FROM dual UNION
   SELECT 1001, 5000 FROM dual UNION
   SELECT 5001, 6000 FROM dual UNION
   SELECT 6001, 7000 FROM dual UNION
   SELECT 7001, 99999999 FROM dual
  ) intervale
LEFT OUTER JOIN liniifact ON PretUnit BETWEEN LimInf AND LimSup
GROUP BY LimInf, LimSup
ORDER BY LimInf, LimSup

```

Figura 9.38 conține informațiile solicitate.

| LimInf | LimSup | Nr_Produse |
|--------|----------|------------|
| 0 | 500 | 0 |
| 501 | 750 | 6 |
| 751 | 1000 | 21 |
| 1001 | 5000 | 20 |
| 5001 | 6000 | 0 |
| 6001 | 7000 | 3 |
| 7001 | 99999999 | 6 |

Figura 9.38. Numărul corect de prețuri unitare, pe fiecare interval

În DB2 trebuie nu numai ca DUAL să fie înlocuit cu SYSIBM.SYSDUMMY1 (sau SYSIBM.DUAL), dar în toate cele șapte SELECT-uri conectate prin UNION trebuie folosită clauza AS pentru ca atributele să „păstreze” titulaturile *LimInf* și *LimSup*:

```
SELECT LimInf, LimSup, COUNT(liniiifact.PretUnit) AS Nr_produce
FROM
    (SELECT 0 AS LimInf, 500 AS LimSup FROM sysibm.sysdummy1 UNION
     SELECT 501 AS LimInf, 750 AS LimSup FROM sysibm.sysdummy1 UNION
     SELECT 751 AS LimInf, 1000 AS LimSup FROM sysibm.sysdummy1 UNION
     SELECT 1001 AS LimInf, 5000 AS LimSup FROM sysibm.sysdummy1
    UNION
     SELECT 5001 AS LimInf, 6000 AS LimSup FROM sysibm.sysdummy1
    UNION
     SELECT 6001 AS LimInf, 7000 AS LimSup FROM sysibm.sysdummy1
    UNION
     SELECT 7001 AS LimInf, 99999999 AS LimSup FROM sysibm.sysdummy1
    ) intervale
LEFT OUTER JOIN liniifact
ON PretUnit BETWEEN LimInf AND LimSup
GROUP BY LimInf, LimSup
ORDER BY LimInf, LimSup
```

Pentru portarea soluției în PostgreSQL și MS SQL Server nu ne rămâne decât ca din sintaxa Oracle să eliminăm *FROM dual*. PostgreSQL-ul și DB2-ul pun la dispoziție, însă, și o soluție bazată pe folosirea clauzei VALUES pentru a defini o „tabelă-constantă” (INTERVALE) chiar în clauza FROM, ceea ce economisește mult din lungimea interogării (vezi și paragrafele 3.4.4 și 3.4.6):

```
SELECT LimInf, LimSup, COUNT(liniiifact.PretUnit) AS Nr_produce
FROM
    (VALUES (0, 500), (501, 750), (751, 1000), (1001, 5000),
     (5001, 6000), (6001, 7000), (7001, 99999999)
    ) AS intervale (LimInf, LimSup)
LEFT OUTER JOIN liniifact
ON PretUnit BETWEEN LimInf AND LimSup
GROUP BY LimInf, LimSup
```

ORDER BY LimInf, LimSup

Să se afle numărul de facturi emise, pe următoarele intervale ale valorilor totale :

- între 0 și 100000 RON;
- între 100001 și 200000 RON;
- între 200001 și 500000 RON;
- între 500001 și 1000000 RON;
- peste 1000001 RON.

Singura diferență majoră față problema anterioară ține de faptul că valoarea unei facturi nu este un atribut la nivel de tuplu (cum era prețul unitar), ci un atribut la nivel de grup. Persistăm în soluții oracliste:

```
SELECT LimInf, LimSup, COUNT(facturi.ValFact) AS Nr_Facturi
FROM
  (SELECT 0 AS LimInf, 100000 AS LimSup FROM dual UNION
   SELECT 100001, 200000 FROM dual UNION
   SELECT 200001, 500000 FROM dual UNION
   SELECT 500001, 1000000 FROM dual UNION
   SELECT 1000001, 99999999 FROM dual
  ) intervale
LEFT OUTER JOIN
  (SELECT f.NrFact,
         SUM(Cantitate * PretUnit * (1+ProcTVA)) AS ValFact
   FROM facturi f INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
         INNER JOIN produse p ON lf.CodPr=p.CodPr
   GROUP BY f.NrFact
  ) facturi
ON ValFact BETWEEN LimInf AND LimSup
GROUP BY LimInf, LimSup
ORDER BY LimInf, LimSup
```

| R 2 | LIMINF | R 2 | LIMSUP | R 2 | NR_FACTURI |
|--------|---------|--------|----------|--------|------------|
| | 0 | | 100000 | | 1 |
| | 100001 | | 200000 | | 14 |
| | 200001 | | 500000 | | 4 |
| | 500001 | | 1000000 | | 0 |
| | 1000001 | | 99999999 | | 9 |

Figura 9.39. Numărul facturi, pe intervale de valori

În PostgreSQL și SQL Server trebuie doar să eliminăm *FROM dual*-urile; în DB2 *dual*-urile trebuie înlocuit cu *sysibm.DUAL*-uri, iar clauza *AS* trebuie folosită pentru coloanele tuturor *SELECT*-urilor – interval. Ca bonus, același rezultat se obține în DB2 și PostgreSQL fie prin înlocuirea/eliminarea din interogarea de mai sus a

secvențelor *FROM dual* (obținem, astfel, și versiunea SQL Server), fie cu ajutorul clauzei *VALUES* folosită ca în precedentul exemplu:

```
SELECT LimInf, LimSup, COUNT(facturi.ValFact) AS Nr_Facturi
FROM
    (VALUES (0, 100000) , (100001, 200000), (200001, 500000),
            (500001, 1000000), (1000001, 99999999)
    ) intervale (LimInf, LimSup)
LEFT OUTER JOIN
    (SELECT f.NrFact,
            SUM(Cantitate * PretUnit * (1+ProcTVA)) AS ValFact
    FROM facturi f INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
            INNER JOIN produse p ON lf.CodPr=p.CodPr
    GROUP BY f.NrFact
    ) facturi
    ON ValFact BETWEEN LimInf AND LimSup
GROUP BY LimInf, LimSup
ORDER BY LimInf, LimSup
```

Care sunt vânzările pentru toate cele zece zile cuprinse în intervalul 1-10 septembrie 2007 ?

De data aceasta, se dorește includerea în rezultat și a zilelor fără nicio factură întocmită, ca în figura 9.40.

Soluțiile se redactează pe calapodul precedentelor, astfel:

- în Oracle:

```
SELECT Zi, TRUNC(COALESCE(SUM(facturi.ValFact),0)) AS Nr_Facturi
FROM
    (SELECT DATE'2007-09-01' AS Zi FROM dual UNION
    SELECT DATE'2007-09-02' FROM dual UNION
    SELECT DATE'2007-09-03' FROM dual UNION
    SELECT DATE'2007-09-04' FROM dual UNION
    SELECT DATE'2007-09-05' FROM dual UNION
    SELECT DATE'2007-09-06' FROM dual UNION
    SELECT DATE'2007-09-07' FROM dual UNION
    SELECT DATE'2007-09-08' FROM dual UNION
    SELECT DATE'2007-09-09' FROM dual UNION
    SELECT DATE'2007-09-10' FROM dual
    ) zile
LEFT OUTER JOIN
    (SELECT f.NrFact, DataFact,
            SUM(Cantitate * PretUnit * (1+ProcTVA)) AS ValFact
    FROM facturi f INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
```

```

    INNER JOIN produse p ON lf.CodPr=p.CodPr
    GROUP BY f.NrFact, DataFact ) facturi
    ON Zi=TRUNC(DataFact)
GROUP BY Zi
ORDER BY Zi

```

| Zi | NR_FACTURI |
|------------|------------|
| 01-09-2007 | 4596401 |
| 02-09-2007 | 238437 |
| 03-09-2007 | 0 |
| 04-09-2007 | 0 |
| 05-09-2007 | 0 |
| 06-09-2007 | 0 |
| 07-09-2007 | 0 |
| 08-09-2007 | 0 |
| 09-09-2007 | 0 |
| 10-09-2007 | 424704 |

Figura 9.40. Vânzările din fiecare zi din intervalul 1-10 sept. 2007

- în MS SQL Server:

```

SELECT Zi, ROUND(COALESCE(SUM(facturi.ValFact),0),0,1) AS Nr_Facturi
FROM
  (SELECT '2007-09-01' AS Zi UNION SELECT '2007-09-02' UNION
   SELECT '2007-09-03' UNION SELECT '2007-09-04' UNION
   SELECT '2007-09-05' UNION SELECT '2007-09-06' UNION
   SELECT '2007-09-07' UNION SELECT '2007-09-08' UNION
   SELECT '2007-09-09' UNION SELECT '2007-09-10'
  ) zile
LEFT OUTER JOIN
  (SELECT f.NrFact, DataFact,
    SUM(Cantitate * PretUnit * (1+ProcTVA)) AS ValFact
  FROM facturi f INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
    INNER JOIN produse p ON lf.CodPr=p.CodPr
    GROUP BY f.NrFact, DataFact ) facturi
  ON Zi=DataFact
GROUP BY Zi
ORDER BY Zi

```

- în DB2/PostgreSQL:

```

SELECT Zi, TRUNC(COALESCE(SUM(facturi.ValFact),0),0) AS Nr_Facturi

```

```

FROM (VALUES
    (CAST ('2007-09-01' AS DATE)), (CAST ('2007-09-02' AS DATE)),
    (CAST ('2007-09-03' AS DATE)), (CAST ('2007-09-04' AS DATE)),
    (CAST ('2007-09-05' AS DATE)), (CAST ('2007-09-06' AS DATE)),
    (CAST ('2007-09-07' AS DATE)), (CAST ('2007-09-08' AS DATE)),
    (CAST ('2007-09-09' AS DATE)), (CAST ('2007-09-10' AS DATE))
) AS zile (Zi)
LEFT OUTER JOIN
    (SELECT f.NrFact, DataFact,
        SUM(Cantitate * PretUnit * (1+ProcTVA)) AS ValFact
    FROM facturi f INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
        INNER JOIN produse p ON lf.CodPr=p.CodPr
    GROUP BY f.NrFact, DataFact ) facturi
    ON Zi=DataFact
GROUP BY Zi
ORDER BY Zi

```

Tot în PostgreSQL există funcția-tabelă `GENERATE_SERIES` care crează un set de linii corespunzătoare unui interval și unui pas/increment. De exemplu, dacă dorim să generăm o tabelă în care fiecare linie să conțină o zi din luna septembrie 2007, se poate folosi interogarea:

```

SELECT DATE'2007-09-01' + Nr AS Zi
FROM GENERATE_SERIES(0,29,1) AS serie (Nr)

```

În aceste condiții, putem simplifica (oareccum) interogarea PostgreSQL pentru problema noastră:

```

SELECT Zi, TRUNC(COALESCE(SUM(facturi.ValFact),0)) AS Nr_Facturi
FROM
    (SELECT DATE'2007-09-01' + Nr AS Zi
    FROM GENERATE_SERIES(0,9,1) Serie (Nr)
    ) Zile
    LEFT OUTER JOIN
        (SELECT f.NrFact, DataFact,
            SUM(Cantitate * PretUnit * (1+ProcTVA)) AS ValFact
        FROM facturi f
            INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
            INNER JOIN produse p ON lf.CodPr=p.CodPr
        GROUP BY f.NrFact, DataFact ) facturi
        ON Zi=DataFact
GROUP BY Zi
ORDER BY Zi

```


În paragraful 8.4 am făcut cunoștință cu o funcție proprietară PostgreSQL, CROSSTAB. O folosim din nou, la o problemă care să pună în valoare și GENERATE_SERIES. Dorim să comparăm, vânzările din primele 7 zile pentru lunile august și septembrie 2007, ca în figura 9.41.

| luna integer | 01/luna/2007 numeric | 02/luna/2007 numeric | 03/luna/2007 numeric | 04/luna/2007 numeric | 05/luna/2007 numeric | 06/luna/2007 numeric | 07/luna/2007 numeric |
|-----------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|
| 8 | 10599535.0000 | 277950.0000 | 222050.0000 | 201975.0000 | | | 10610000.5000 |
| 9 | 4596401.5000 | 238437.5000 | | | | | |

Figura 9.41. Folosirea funcțiilor GENERATE_SERIES și CROSSTAB

Interogarea PostgreSQL se prezintă astfel:

```
SELECT *
FROM crosstab(
    'SELECT EXTRACT (MONTH FROM DataFact) AS Luna,
        EXTRACT (DAY FROM DataFact) AS Ziua, Vinzari
    FROM (
        SELECT DataFact, SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari
        FROM facturi
            INNER JOIN liniifact ON facturi.NrFact=liniifact.NrFact
            INNER JOIN produse ON liniifact.CodPr=produse.CodPr
        WHERE EXTRACT (YEAR FROM DataFact)=2007 AND
            EXTRACT (MONTH FROM DataFact) BETWEEN 8 AND 9 AND
            EXTRACT (DAY FROM DataFact) BETWEEN 1 AND 7
        GROUP BY DataFact
    ) x
    ORDER BY 1,2',
    'SELECT Nr FROM generate_series (1,7) Nr '
        ) AS (
        Luna INT,
        "01/luna/2007" NUMERIC,
        "02/luna/2007" NUMERIC,
        "03/luna/2007" NUMERIC,
        "04/luna/2007" NUMERIC,
        "05/luna/2007" NUMERIC,
        "06/luna/2007" NUMERIC,
        "07/luna/2007" NUMERIC
    );
```

9.5.Subconsultări scalare în clauza SELECT

Subconsultările scalare sunt cele care obțin un rezultat alcătuit dintr-o singură linie și o singură coloană. Dacă subconsultările incluse în clauzele WHERE, HAVING și FROM puteau obține un rezultat tabelar oarecare, în clauza SELECT suntem constrânși să includem doar interogări scalare.

Care sunt totalurile vânzărilor și încasărilor ?

Ca de obicei, începem cu o problemă slabă, pentru care formulăm o soluție curioasă, mai întâi în PostgreSQL/SQL Server:

```
SELECT
    (SELECT SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari
     FROM liniifact lf INNER JOIN produse p
       ON lf.CodPr=p.CodPr
    ) AS Facturat,
    (SELECT SUM (Transa) FROM incasfact) AS Incasat
```

apoi echivalenta sa în Oracle:

```
SELECT
    ...
FROM dual
```

și cea din DB2:

```
SELECT
    ...
FROM SYSIBM.SYSDUMMY1
```

Fraza SELECT conține două subconsultări scalare, una care extrage totalul vânzărilor, cealaltă, totalul încasărilor. În PostgreSQL și SQL Server clauza FROM poate lipsi, în timp ce în Oracle nu, așa că, fiind nevoie de o tabelă cu o singură linie, s-a folosit tabela DUAL (creată automat în Oracle la instalare) și SYSIBM.SYSDUMMY1 care, după cum am văzut încă din capitolul 5, are o singură linie și un singură coloană. Indiferent de variantă, cele două rezultate ale interogărilor scalare au valorile din figura 9.42.

| FACTURAT | INCASAT |
|----------|---------|
| 49170335 | 518281 |

Figura 9.42. Două interogări scalare în clauza SELECT

Care sunt totalurile salariilor tarifare ale angajaților și ale sporurilor pe luna iulie 2007 pentru întreaga firmă ?

Atributul SalTarifar se află în tabela PERSONAL2, iar sporurile în tabela SPORURI. Până în acest paragraf am putea folosi câteva variante, precum:

```
SELECT SUM(SalTarifar) AS Total_Sal_Tarifar, SUM ( COALESCE(SporVechime,0) +
      COALESCE(SporNoapte,0) + COALESCE(SporCD,0) + COALESCE(AlteSpor,0) )
      AS Total_Sporuri_Iulie
FROM personal2 INNER JOIN sporuri
      ON personal2.Marca=sporuri.Marca WHERE An=2007 AND Luna=7
```

sau

```
SELECT *
FROM (
      SELECT SUM(SalTarifar) AS Total_Sal_Tarifar
      FROM personal2
      ) un_total,
(SELECT SUM ( COALESCE(SporVechime,0) + COALESCE(SporNoapte,0)
      + COALESCE(SporCD,0) + COALESCE(AlteSpor,0) )
      AS Total_Sal_Tarifar
      FROM sporuri
      WHERE An=2007 AND Luna=7
      ) alt_total
```

Fiind vorba de două valori scalare, putem include două interogări în clauza SELECT, iar în clauza FROM a interogării principale folosim, după caz, DUAL (în Oracle), SYSIBM.SYSDUMMY1 (în DB2) sau nimic (în PostgreSQL și MS SQL Server):

```
SELECT (
      SELECT SUM(SalTarifar) FROM personal2
      ) AS Total_Sal_Tarifar,
      (
      SELECT SUM ( COALESCE(SporVechime,0) +
      COALESCE(SporNoapte,0) + COALESCE(SporCD,0) +
      COALESCE(AlteSpor,0) )
      FROM sporuri
      WHERE An=2007 AND Luna=7
      ) AS Total_Sal_Tarifar
FROM dual
```

Curios lucru, interogarea:

```
SELECT SUM(SalTarifar) AS Total_Sal_Tarifar,
      (SELECT SUM ( COALESCE(SporVechime,0) +
      COALESCE(SporNoapte,0)+
      COALESCE(SporCD,0)+ COALESCE(AlteSpor,0))
```

FROM sporuri

WHERE An=2007 AND Luna=7) AS Total_Sporuri_Iulie

FROM personal2

funcționează în DB2, PostgreSQL și SQL Server, dar nu și în Oracle – vezi figura 9.43.

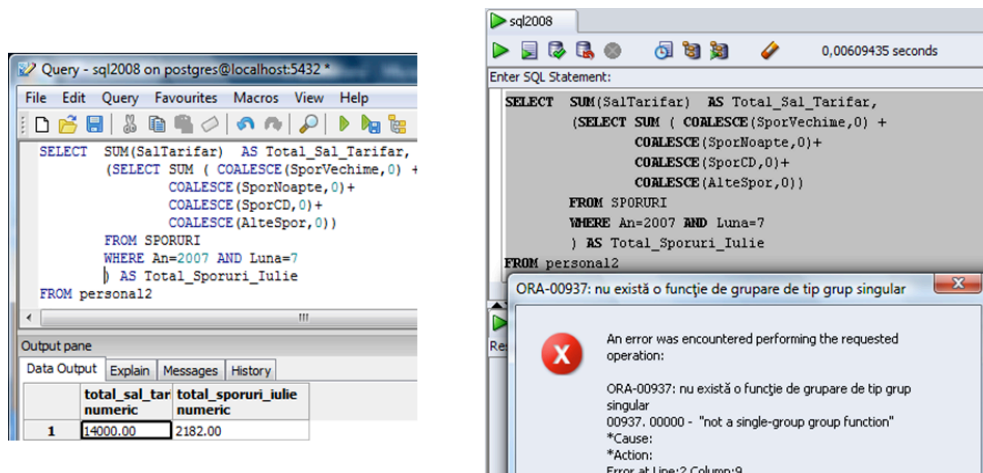


Figura 9.43. O interogare funcționabilă în PostgreSQL și SQL Server, dar nu și în Oracle

Pentru cât la sută dintre clienți s-au întocmit, zilnic, facturi ?

```
SELECT DataFact, COUNT(DISTINCT CodCl) AS Nr_Clienti,
      (SELECT COUNT(*)
       FROM clienti
      ) AS Nr_Total_Clienti,
      ROUND((COUNT(DISTINCT CodCl) * 100) / (
        SELECT COUNT(*)
        FROM clienti
      ),0) AS Procent
FROM facturi
GROUP BY DataFact
ORDER BY 1
```

Pentru a obține procentul care ne interesează se împarte rezultatul calculat de funcția COUNT pentru fiecare grup (zi calendaristică) la valoarea extrasă de interogarea scalară – vezi figura 9.44. Soluția este valabilă pentru toate cele patru dialecte.

| DataFact | Nr_Clienti | Nr_Total_Clienti | Procent |
|---------------------|------------|------------------|---------|
| 2007-08-01 00:00:00 | 4 | 7 | 57 |
| 2007-08-02 00:00:00 | 2 | 7 | 28 |
| 2007-08-03 00:00:00 | 1 | 7 | 14 |
| 2007-08-04 00:00:00 | 1 | 7 | 14 |
| 2007-08-07 00:00:00 | 4 | 7 | 57 |
| 2007-08-14 00:00:00 | 3 | 7 | 42 |
| 2007-08-15 00:00:00 | 2 | 7 | 28 |
| 2007-08-16 00:00:00 | 1 | 7 | 14 |
| 2007-08-21 00:00:00 | 2 | 7 | 28 |
| 2007-08-22 00:00:00 | 1 | 7 | 14 |
| 2007-09-01 00:00:00 | 2 | 7 | 28 |
| 2007-09-02 00:00:00 | 2 | 7 | 28 |
| 2007-09-10 00:00:00 | 2 | 7 | 28 |
| 2007-09-17 00:00:00 | 1 | 7 | 14 |
| 2007-10-07 00:00:00 | 1 | 7 | 14 |

Figura 9.44. Procentajul zilnic al clienților pentru care există facturi

Care este contribuția (procentuală) a fiecărui produs la totalul vânzărilor ?

Informația este una esențială pentru orice firmă. Ne vom servi de o subconsultare scalară pentru a determina, pe fiecare linie (corespunzătoare unui produs) totalul vânzărilor și a obține, astfel, raportul care interesează⁵.

```

SELECT DenPr AS Produs,
       SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari_Produs,
       (
         SELECT SUM(Cantitate * PretUnit * (1+ProcTVA))
         FROM liniifact lf
              INNER JOIN produse p ON lf.CodPr=p.CodPr
       ) AS Total_Vinzari,
       TRUNC(
         (SUM(Cantitate * PretUnit * (1+ProcTVA)) * 100) /
         (SELECT SUM(Cantitate * PretUnit * (1+ProcTVA))

```

⁵ Cele patru servere BD prezintă diferențe în modul de afișare a pozițiilor fracționare. Figura redă rezultatul din PostgreSQL.

```

FROM liniifact lf INNER JOIN produse p
ON lf.CodPr=p.CodPr
)
, 2 ) AS Procent
FROM liniifact lf INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY DenPr
ORDER BY 1

```

| produs character var | vinzari_produ numeric | total_vinzari numeric | procent numeric |
|-------------------------|--------------------------|--------------------------|--------------------|
| Produs 1 | 988533.0000 | 49170335.0000 | 2.01 |
| Produs 2 | 2784023.5000 | 49170335.0000 | 5.66 |
| Produs 3 | 251685.0000 | 49170335.0000 | 0.51 |
| Produs 4 | 301657.5000 | 49170335.0000 | 0.61 |
| Produs 5 | 44844436.0000 | 49170335.0000 | 91.20 |

Figura 9.45. Contribuția fiecărui produs la cifra de afaceri

Care este ziua în care s-au emis cele mai multe facturi ?

```

SELECT DataFact, COUNT(*) AS Nr_Facturi,
(SELECT MAX(NrF)
FROM
(SELECT COUNT(*) AS NrF
FROM facturi
GROUP BY DataFact) T1 ) AS NrMax
FROM facturi
GROUP BY DataFact
HAVING COUNT(*) >=
(SELECT MAX(NrF)
FROM
(SELECT COUNT(*) AS NrF
FROM facturi
GROUP BY DataFact) T2 )

```

Valorile coloanelor Nr_Facturi și NrMax din figura 9.46 sunt furnizate de două subconsultări scalare, una care calculează numărul zilnic al facturilor, iar a doua numărul maxim de facturi emise într-o zi.

| RZ | DATAFACT | RZ | NR_FACTURI | RZ | NRMAX |
|----|------------|----|------------|----|-------|
| | 01-08-2007 | | 4 | | 4 |
| | 07-08-2007 | | 4 | | 4 |

Figura 9.46. Zilele în care s-au întocmit cele mai multe facturi

Din păcate, coloana calculată printr-o interogare scalară nu poate fi folosită în alte clauze (WHERE, HAVING), nici măcar în aceeași clauză SELECT. Astfel, ne-am fi imaginat că la problema *Ce facturi au fost emise în aceeași zi cu factura 1120 ?* am putea formula și o soluție de genul:

```
SELECT facturi.*,
      (SELECT DataFact FROM facturi WHERE NrFact=1120
       ) AS Data_F_1120
FROM facturi
WHERE DataFact = Data_F_1120
```

Aceasta ar fi vestea rea – vezi figura 9.47. Vestea bună este că subconsultările scalare din clauza SELECT pot fi corelate cu tabelele din clauza FROM, după cum vom vedea în capitoul următor.

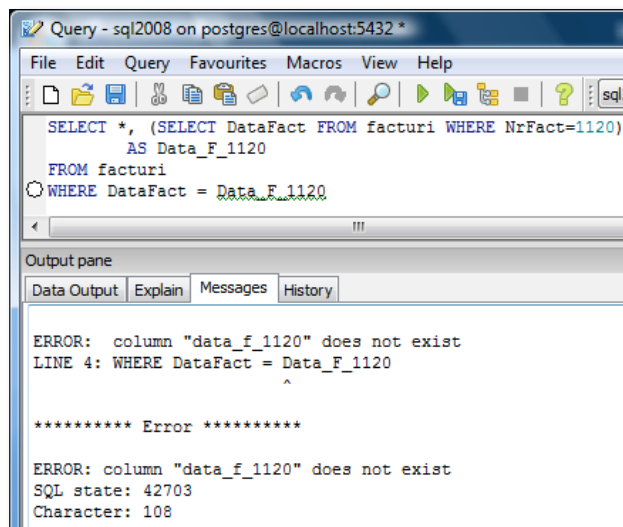


Figura 9.47. Subconsultarea din clauza SELECT nu este “recunoscută” în celelalte clauze ale interogării

9.6. Expresii – tabelă

În prima ediție a cărții prezentam eleganta opțiune a expresiilor tabele doar în DB2. Între timp și Oracle și SQL Server le-au implementat. Revenim (pentru a treia oară) la exemplul 19 din algebra relațională: *Ce facturi au fost emise în aceeași zi cu factura 1120 ?*

Iată și soluția DB2/Oracle/SQL Server bazată pe expresii-tabele:

```
WITH Zi_1120 AS
  (SELECT DataFact
   FROM facturi
   WHERE NrFact=1120)
```

```

SELECT NrFact
FROM facturi, Zi_1120
WHERE facturi.DataFact=Zi_1120.DataFact

```

ZI_1120 este o expresie-tabelă cu o singură coloană – DataFact - și o singură linie care conține data întocmirii facturii 1120, fiind folosită în fraza SELECT principală ca o tabelă obișnuită. Aici e marele avantaj, prin comparație cu subconsultările din clauza FROM: tabela expresie este recunoscută în toate clauzele și subconsultările „implicate” în interogarea principală și subconsultările ei.

Continuăm cu trei probleme de „maxim” rezolvate cu noua găselniță.

- *Care este ziua în care s-au emis cele mai multe facturi ?*

```

WITH facturi_zile AS
    (SELECT DataFact, COUNT(*) AS Nr_Facturilor
     FROM facturi
     GROUP BY DataFact)
SELECT *
FROM facturi_zile
WHERE Nr_Facturilor =
    (SELECT MAX(Nr_Facturilor)
     FROM facturi_zile)

```

- *Care este clientul care a cumpărat cele mai multe produse ?*

```

WITH clienti_produce AS
    (SELECT DenCl, COUNT(DISTINCT CodPr) AS Nr_Prod
     FROM clienti c
      INNER JOIN facturi f ON c.CodCl=f.CodCl
      INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
     GROUP BY DenCl)
SELECT *
FROM clienti_produce
WHERE Nr_Prod =
    (SELECT MAX(Nr_Prod)
     FROM clienti_produce)

```

- *Care este județul în care berea s-a vândut cel mai bine ?*

```

WITH judete_bere AS
    (SELECT Judet, SUM(Cantitate*PretUnit*(1+ProcTVA)) AS Vnz_Bere
     FROM judete j
      INNER JOIN coduri_postale cp ON j.Jud=cp.Jud
      INNER JOIN clienti c ON cp.CodPost=c.CodPost
      INNER JOIN facturi f ON c.CodCl=f.CodCl
      INNER JOIN liniifact lf ON f.NrFact= lf.NrFact)

```



```

        INNER JOIN produse p ON lf.CodPr=p.CodPr
    WHERE Grupa='Bere'
    GROUP BY Judet)

SELECT *
FROM judete_bere
WHERE Vnz_Bere =
    (SELECT MAX(Vnz_Bere)
     FROM judete_bere)

```

Într-o interogare putem declara două sau mai multe expresii-tabelă pe care să le folosim în toate clauzele, în aceeași manieră în care folosim celelalte tabele ale bazei de date. Să luăm un exemplu folosit de-a lungul și de-a latul ultimelor capitole pentru ilustrarea atât a intersecției, cât și a diviziunii:

În ce zile s-au vândut și produsul cu denumirea "Produs 1" și cel cu denumirea "Produs 2" ?

De această dată au fost definite două expresii-tabele, ZILE_PRODUS1 ce conține zilele în care s-a vândut produsul 1, și ZILE_PRODUS2 care „adăpostește” zilele în care s-a vândut cel de-al doilea produs. Fraza SELECT principală îndeplinește o simplă formalitate – intersecția celor două tabele ad-hoc:

```

WITH zile_produs1 AS
    (SELECT DISTINCT DataFact
     FROM facturi f
      INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
      INNER JOIN produse p ON lf.CodPr=p.CodPr
     WHERE DenPr = 'Produs 1'),
zile_produs2 AS
    (SELECT DISTINCT DataFact
     FROM facturi f
      INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
      INNER JOIN produse p ON lf.CodPr=p.CodPr
     WHERE DenPr = 'Produs 2')

SELECT *
FROM zile_produs1
    INTERSECT
SELECT *
FROM zile_produs2

```

Interogările din paragraful 9.4.5 – dedicat tabelor temporare ad-hoc declarate în clauza FROM pot fi, uneori, vizibil simplificate. Să luăm doar unul dintre aceste exemple, deși economia interogării nu se ameliorează spectaculos. *Să se afle numărul de produse vândute, pe următoarele intervale ale prețului unitar de vânzare: între*

0 și 500 RON; între 501 și 750 RON; între 751 și 1000 RON...peste 7001 RON. Enunțul este cel al primului exemplu din paragraful 9.4.5 (vezi figura 9.38). Iată soluția bazată pe expresii tabelă, redactată după sintaxa SQL Server:

WITH intervale AS

(SELECT 0 AS LimInf, 500 AS LimSup UNION

SELECT 501, 750 UNION

SELECT 751, 1000 UNION

SELECT 1001, 5000 UNION

SELECT 5001, 6000 UNION

SELECT 6001, 7000 UNION

SELECT 7001, 99999999

)

SELECT LimInf, LimSup, COUNT(liniiifact.PretUnit) AS Nr_produce

FROM intervale

LEFT OUTER JOIN liniifact ON PretUnit BETWEEN LimInf AND LimSup

GROUP BY LimInf, LimSup

ORDER BY LimInf, LimSup

Ce produse au fost vândute tuturor clienților ? (paragraful 4.4.7, exemplul 26)

WITH produse_clienti AS

(SELECT DenPr, COUNT(DISTINCT CodCl) AS Nr

FROM produse p

INNER JOIN liniifact lf ON p.CodPr=lf.CodPr

INNER JOIN facturi f ON lf.NrFact=f.NrFact

GROUP BY DenPr),

nr_clienti AS

(SELECT COUNT(CodCl) AS NrClienti

FROM clienti

)

SELECT DenPr

FROM produse_clienti INNER JOIN nr_clienti ON Nr=NrClienti

Care sunt clienții pentru care există cel puțin câte o factură emisă în fiecare zi cu vânzări din perioada 10-30 septembrie 2007? (exemplu 23, paragraf 4.4.7)

WITH clienti_zile AS (

SELECT DenCl, COUNT(DISTINCT DataFact) AS NrZileCl

FROM facturi INNER JOIN clienti

ON facturi.CodCl=clienti.CodCl

WHERE DataFact BETWEEN DATE'2007-09-10' AND DATE'2007-09-30'

GROUP BY DenCl

),

```
zile AS (  
    SELECT COUNT(DISTINCT DataFact) AS NrZile  
    FROM facturi  
    WHERE DataFact BETWEEN DATE'2007-09-10' AND DATE'2007-09-30'  
    )  
SELECT DenCl  
FROM clienti_zile INNER JOIN zile ON NrZileCl=NrZile6
```

Ce facturi conțin măcar produsele din factura 1117 ? (exemplu 27, paragraf 4.4.7)

```
WITH facturi_produse1117 AS (  
    SELECT lf1.NrFact, lf1.CodPr  
    FROM liniifact lf1 INNER JOIN liniifact lf2 ON  
        lf1.CodPr=lf2.CodPr AND lf2.NrFact=1117  
    ),  
nrproduse_1117 AS (  
    SELECT COUNT(DISTINCT CodPr) AS Nr  
    FROM liniifact  
    WHERE NrFact=1117)  
SELECT NrFact  
FROM facturi_produse1117  
GROUP BY NrFact  
HAVING COUNT(DISTINCT CodPr) =  
    (SELECT Nr  
     FROM nrproduse_1117  
    )
```

Aici am folosit în premieră o expresie tabelă tocmai în subconsultarea din clauza HAVING, ceea ce nu era chiar evident. În general, orice interogare care ni s-a părut prea lungă/laborioasă poate fi simplificată cu ajutorul expresiilor tabelă.

⁶ Nu uitați să eliminați cuvintele DATE în variantele DB2 și SQL Server.

Care este contribuția (procentuală) a fiecărui produs la totalul vânzărilor ?

```

WITH vinzari_produce AS
(
    SELECT DenPr AS Produs,
           SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Vinzari
FROM liniifact lf INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY DenPr
),
total_vinzari AS
(
    SELECT SUM(Cantitate * PretUnit * (1+ProcTVA)) AS Total
FROM liniifact lf
    INNER JOIN produse p ON lf.CodPr=p.CodPr
)
SELECT Produs, Vinzari, Total,
       CAST (Vinzari * 100 / Total AS NUMERIC (4,2)) AS Procent
FROM vinzari_produce, total_vinzari

```

Pot apărea ușoare diferențe față de figura 9.45, datorită folosirii funcției CAST în loc de TRUNC (din rațiuni de portabilitate).

Care sunt clienții cu valoarea vânzărilor peste medie ?

```

WITH
vinzari_clienti AS
    (SELECT DenCl AS client,
           ROUND(SUM(Cantitate * PretUnit * (1 + ProcTVA)),0) AS vinzari
FROM clienti c
    INNER JOIN facturi f ON c.CodCl=f.CodCl
    INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
    INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY DenCl),
vinzari_medii AS
    (SELECT SUM(Cantitate * PretUnit * (1 + ProcTVA)) /
           COUNT(DISTINCT f.CodCl) AS medie
FROM facturi f
    INNER JOIN liniifact lf ON f.NrFact= lf.NrFact
    INNER JOIN produse p ON lf.CodPr=p.CodPr )
SELECT *
FROM vinzari_clienti
WHERE vinzari > (SELECT medie FROM vinzari_medii)

```

Care este restul de plată al fiecărui client ?

WITH

facturat AS

(SELECT DenCl, c.CodCl, COALESCE(SUM(Cantitate *
PretUnit * (1+ProcTVA)),0) AS Facturat

FROM clienti c

INNER JOIN facturi f ON c.CodCl=f.CodCl

INNER JOIN liniifact lf ON f.NrFact=lf.NrFact

INNER JOIN produse p ON lf.CodPr=p.CodPr

GROUP BY DenCl, c.CodCl),

incasat AS

(SELECT CodCl, SUM(Transa) AS Incasat

FROM facturi f INNER JOIN incasfact i ON f.NrFact=i.NrFact

GROUP BY CodCl)

SELECT DenCl, Facturat, COALESCE(Incasat,0) AS Incasat,

Facturat - COALESCE(Incasat,0) AS De_Incasat

FROM facturat f LEFT OUTER JOIN incasat i ON f.CodCl=i.CodCl

ORDER BY DenCl

Care este clientul cu cel mai mare rest de plată ?

Bine mai zice proverbul cu dai un deget și ți se ia o mână. Forțând cu nerușinare SQL-ul, încercând încadrarea unei clauze WITH în altă clauză WITH:

WITH sinteza AS

(WITH facturat AS (SELECT DenCl, c.CodCl, COALESCE(SUM(Cantitate *
PretUnit * (1+ProcTVA)),0) AS Facturat

FROM clienti c INNER JOIN facturi f ON c.CodCl=f.CodCl

INNER JOIN liniifact lf ON f.NrFact=lf.NrFact

INNER JOIN produse p ON lf.CodPr=p.CodPr

GROUP BY DenCl, c.CodCl),

incasat AS (SELECT CodCl, SUM(Transa) AS Incasat

FROM facturi f INNER JOIN incasfact i ON f.NrFact=i.NrFact

GROUP BY CodCl)

SELECT DenCl, Facturat, COALESCE(Incasat,0) AS Incasat,

Facturat - COALESCE(Incasat,0) AS De_Incasat

FROM facturat f LEFT OUTER JOIN incasat i ON f.CodCl=i.CodCl

ORDER BY DenCl)

SELECT *

FROM sinteza

WHERE De_Incasat = (

SELECT MAX(De_Incasat) FROM sinteza

)

Reacția Oracle la execuția acestei interogări este surprinsă în figura 9.48. Nici DB2 sau SQL Server nu sunt mai cooperante. Nu putem bănuî prea multă iritare în mesajul de eroare, deși am fi meritat un pic de săpuneală. Reținem, deci, că nu putem include o expresie-tabelă în alta.

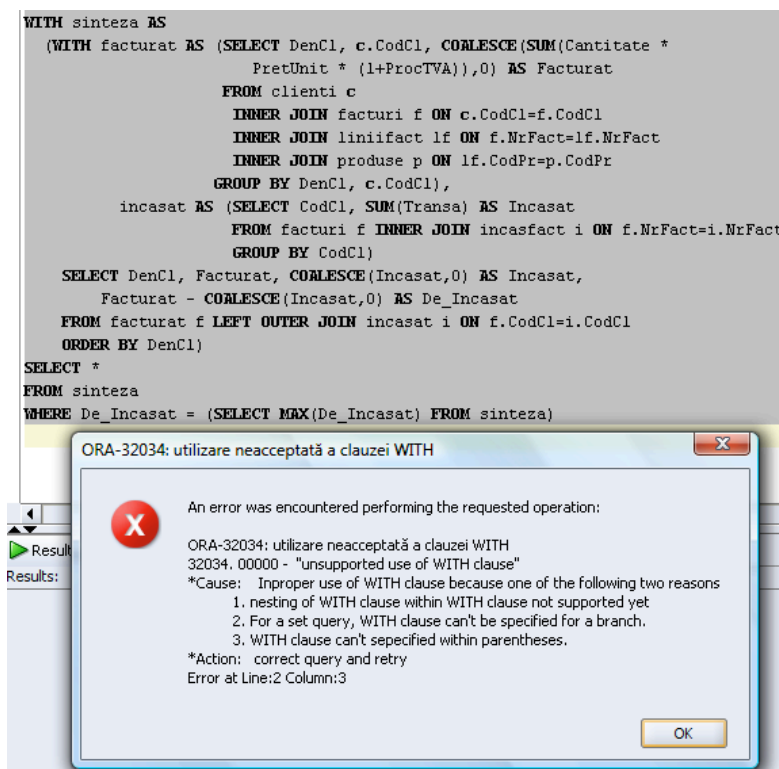


Figura 9.48. Expresiile tabelă nu se pot include una-n alta

Cu toate acestea, soluțiile bazate pe expresii-tabelă pentru această problemă nu lipsesc:

```
WITH situatie_clienti AS (
    SELECT CodCl,
           COALESCE(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) AS Facturat,
           SUM(COALESCE(Transa,0)) AS Incasat
    FROM facturi f
           INNER JOIN liniifact lf ON f.NrFact=lf.NrFact
           INNER JOIN produse p ON lf.CodPr=p.CodPr
           LEFT OUTER JOIN incasfact i ON f.NrFact=i.NrFact
    GROUP BY CodCl)
SELECT DenCl, Facturat, Incasat, Facturat-Incasat AS RestPlata
FROM clienti c LEFT OUTER JOIN situatie_clienti sc ON c.CodCl=sc.CodCl
```

```

WHERE Facturat-Incasat =
      (SELECT MAX(Facturat-Incasat) FROM situatie_clienti)

sau:
WITH facturat AS
      (SELECT DenCl, c.CodCl, COALESCE(SUM(Cantitate *
            PretUnit * (1+ProcTVA)),0) AS Facturat
      FROM clienti c
            INNER JOIN facturi f ON c.CodCl=f.CodCl
            INNER JOIN liniifact lf ON f.NrFact=lf.NrFact
            INNER JOIN produse p ON lf.CodPr=p.CodPr
      GROUP BY DenCl, c.CodCl),
incasat AS
      (SELECT CodCl, SUM(Transa) AS Incasat
      FROM facturi f INNER JOIN incasfact i ON f.NrFact=i.NrFact
      GROUP BY CodCl)
SELECT DenCl, Facturat, COALESCE(Incasat,0) AS Incasat,
      Facturat - COALESCE(Incasat,0) AS De_Incasat
FROM facturat f LEFT OUTER JOIN incasat i ON f.CodCl=i.CodCl
WHERE Facturat - COALESCE(Incasat,0) =
      (SELECT MAX(Facturat - COALESCE(Incasat,0))
      FROM facturat f LEFT OUTER JOIN incasat i ON f.CodCl=i.CodCl)

```

Care sunt numerele de facturi nefolosite ?

Numerele facturilor emise sunt, de obicei, consecutive, datorită regimului acestui tip de documente. Totuși, în tabela FACTURI avem câteva “găuri” (gap-uri, în romgleză) pe care ne propunem să le aflăm acum.

O soluție la această problemă este de a genera ad-hoc o tabelă în care un rând să fie o cifră de la 0 la 9. Vom aplica produsul cartezian asupra a patru instanțe ale acestei tabele ah-hoc știind că, spre exemplu, $1123 = 1 * 1000 + 1 * 100 + 2 * 10 + 3$:

```

WITH cifre AS (
      SELECT 0 AS Cifra FROM dual UNION SELECT 1 FROM dual
            UNION SELECT 2 FROM dual UNION SELECT 3 FROM dual
            UNION SELECT 4 FROM dual UNION
            SELECT 5 FROM dual UNION SELECT 6 FROM dual
            UNION SELECT 7 FROM dual UNION SELECT 8 FROM dual
            UNION SELECT 9 FROM dual
      )
SELECT c1000.Cifra * 1000 + c100.Cifra * 100 + c10.Cifra * 10 + c1.Cifra AS
Numar
FROM cifre c1000
      CROSS JOIN cifre c100

```

CROSS JOIN cifre c10

CROSS JOIN cifre c1

ORDER BY 1 DESC

Rezultatul acestei interogări (vezi primele 7,25 linii din stânga figura 9.49) conține 10000 de linii, ordonate de la 9999 la 0.

| NUMAR | Nr_nefolosit |
|-------|--------------|
| 9999 | 1123 |
| 9998 | 1124 |
| 9997 | 1125 |
| 9996 | 1126 |
| 9995 | 1127 |
| 9994 | 1128 |
| 9993 | 1129 |
| 9992 | 1130 |

Figura 9.49. Numerele întregi decrescătoare de la 9999 la 0 (stânga) și numere nefolosite în tabela FACTURI (dreapta)

Acum nu ne mai rămâne decât să folosim această tabelă drept argument al unui predicat în care vom căuta valorile care nu se regăsesc între numerele de factură minime și maxime din tabela FACTURI (vezi dreapta figurii 9.49):

WITH cifre AS (

```

SELECT 0 AS Cifra FROM dual UNION SELECT 1 FROM dual UNION
SELECT 2 FROM dual UNION SELECT 3 FROM dual UNION
SELECT 4 FROM dual UNION
SELECT 5 FROM dual UNION SELECT 6 FROM dual UNION
SELECT 7 FROM dual UNION SELECT 8 FROM dual UNION
SELECT 9 FROM dual
)
```

SELECT Numar AS "Nr_nefolosit"

```

FROM (SELECT c1000.Cifra * 1000 + c100.Cifra * 100 + c10.Cifra * 10 + c1.Cifra
      AS Numar
```

```

      FROM cifre c1000 CROSS JOIN cifre c100 CROSS JOIN cifre c10
      CROSS JOIN cifre c1)
```

```

WHERE Numar BETWEEN (SELECT MIN(NrFact) FROM facturi) AND
      (SELECT MAX(NrFact) FROM facturi)
      AND Numar NOT IN (SELECT NrFact FROM facturi)
```

ORDER BY 1

Sintaxa interogării pentru celelalte trei servere BD rămâne ca temă pentru acasă.