

## Capitolul 7. Grupări

Limbajul SQL, ca și algebra relațională din care se trage, este orientat preponderent pe tupluri, lucrând cu seturi de linii ale tabelelor, și nu cu fiecare linie în parte. Funcțiile agregat pe care le-am discutat în paragraful anterior operau asupra tuturor liniilor (tuplurilor) tabelului din clauza FROM care îndeplineau condiția de filtrare din clauza WHERE (când în clauza FROM apar două sau mai multe tabele jonctionate, liniile sunt filtrate atât prin clauza WHERE, cât și prin clauza ON de la INNER JOIN).

Folosind clauza GROUP BY funcțiile agregat pot fi aplicate la nivel de grup de înregistrări, rezultatul având atâtea linii câte valori distincte are atributul/coloana de grupare. Mai mult, putem elimina o parte din grupuri folosind un predicat specificat într-o clauză specială – HAVING.

Dacă în precedentele capitole am fost destul de democrați în materie de sintaxă cu toate cele patru servere BD, în acesta vom privilegia variantele cele mai conforme cu standardul, adică cele Oracle și PostgreSQL. Pe baza discuțiilor de până acum, veți putea porta interogările pe sintaxa DB2 și SQL Server fără prea multe dureri de cap.

### 7.1. Gruparea după un criteriu

Clauza GROUP BY formează grupe (grupuri) de tupluri ale unei relații, pe baza valorilor comune ale unui atribut. Rezultatul unei fraze SELECT ce conține această clauză se obține prin regruparea tuturor liniilor din tabelele enumerate în FROM, extrăgându-se câte o apariție pentru fiecare valoare distinctă a coloanei/coloanelor de grupare. Cel mai simplu format este:

```
SELECT expresie 1, expresie 2, ..., expresie m
FROM tabelă
WHERE predicat
GROUP BY coloană-de-grupare
```

Să luăm câteva exemple pentru a limpezi discuția.

*Care este valoarea fără TVA a facturilor cu numere cuprinse între 1111 și 1119?*

Valoarea (fără TVA) a unei facturi se obține însumând rezultatele produsului *Cantitate \* PretUnit* pentru toate liniile din factura respectivă. Totalizarea se face pe facturi, așa încât atributul de grupare este NrFact:

```
SELECT NrFact, SUM(Cantitate*PretUnit) as ValFaraTVA
FROM liniifact
WHERE NrFact BETWEEN 1111 AND 1119
GROUP BY NrFact
```

Lansarea în execuție a acestei interogări declanșează următoarea succesiune de operații:

- Din tabela LINIIFACT se extrag numai tuplurile în care valorile atributului NrFact sunt cuprinse între 1111 și 1119.
- Se constituie câte un grup pentru fiecare valoare distinctă a NrFact; în cazul, deoarece sunt nouă valori distincte ale NrFact selectate, vor fi constituite nouă grupuri.
- În interiorul fiecărui grup (care are un număr de rânduri egal cu numărul liniilor din factura corespunzătoare grupului) se execută funcția-agregat -  $SUM(Cantitate * PretUnit)$ .
- Se obține rezultatul al cărui număr de linii coincide cu valorile distincte ale NrFact. Ordinea liniilor nu este garantată în lipsa clauzei ORDER BY.

Pașii b), c), și d) sunt schematizați (sau sugerați) în figura 7.1. Deși valorile atributului de grupare sunt ordonate crescător în figură, acest lucru nu este garantat automat la orice grupare.

	NrFact	Linie	CODPR	CANTITATE	PRETUNIT
1	1111	1	1	50	1000
	1111	2	2	75	1050
	1111	3	5	500	7060
2	1112	1	2	80	1030
	1112	2	3	40	750
3	1113	1	2	100	975
4	1114	1	2	70	1070
	1114	2	4	30	1705
	1114	3	5	700	7064
5	1115	1	2	150	925
6	1116	1	2	125	930
7	1117	1	2	100	1000
	1117	2	1	100	950
8	1118	1	2	30	1100
	1118	2	1	150	930
9	1119	1	2	35	1090
	1119	2	3	40	700
	1119	3	4	50	1410
	1119	4	5	750	6300

	NrFact	Val_fara_TVA
1	1111	3658750
2	1112	112400
3	1113	97500
4	1114	5070850
5	1115	138750
6	1116	116250
7	1117	195000
8	1118	172500
9	1119	4861650

Figura 7.1. Pașii b)-d) din execuția grupării

*Care este numărul de facturi emise în fiecare zi cu vânzări ?*

O zi cu vânzări este o dată în care avem măcar o factură întocmită. Întrucât în enunțul problemei apare *...în fiecare zi cu vânzări*, atributul care corespunde zilelor de vânzare este DataFact din FACTURI. Acesta va fi atributul de grupare. Numărul de facturi se obține cu ajutorul funcției COUNT:

```
SELECT DataFact AS "Zi", COUNT(*) AS "Numar facturi"
FROM facturi
```

**GROUP BY DataFact**

Zi	Numar facturi
01-08-2007	4
04-08-2007	1
17-09-2007	1
03-08-2007	1
01-09-2007	2
07-10-2007	1
02-08-2007	2
21-08-2007	2
07-08-2007	4
14-08-2007	3
15-08-2007	2
16-08-2007	2
02-09-2007	2
22-08-2007	1
10-09-2007	2

Figura 7.2. Numărul de facturi întocmite în fiecare zi cu vânzări

După cum se observă în figura 7.2, în Oracle ordinea extragerii liniilor în rezultat nu este cea a valorilor atributului de grupare. La fel stau lucrurile în PostgreSQL. De aceea, pentru a fi siguri de ordonare, trebuie să folosim clauza ORDER BY. Ne interesează ca zilele să apară în ordine crescătoare:

Zi	Numar facturi
01-08-2007	4
02-08-2007	2
03-08-2007	1
04-08-2007	1
07-08-2007	4
14-08-2007	3
15-08-2007	2
16-08-2007	2
21-08-2007	2
22-08-2007	1
01-09-2007	2
02-09-2007	2
10-09-2007	2
17-09-2007	1
07-10-2007	1

Figura 7.3. Ordonarea valorilor atributului de grupare

```
SELECT DataFact AS "Zi", COUNT(*) AS "Numar facturi"
FROM facturi GROUP BY DataFact ORDER BY DataFact
```

Rezultatul este cel din figura 7.3. DB2 și SQL Server asigură ordonarea liniilor în rezultat, chiar și fără clauza ORDER BY.

*Care este valoarea totală a vânzărilor pentru fiecare zi din luna septembrie 2007 în care s-au emis facturi (fig. 7.4)?*

Dacă atributul de grupare este tot DataFact, pentru că ne interesează valoarea zilnică, obținerea vânzărilor (cu tot cu TVA) reclamă joncționarea tabeli FACTURI cu LINIIFACT și apoi cu PRODUSE:

```
SELECT DataFact, TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) AS ValTotala
FROM liniifact lf
    INNER JOIN produse p ON lf.CodPr=p.CodPr
    INNER JOIN facturi f ON lf.NrFact=f.NrFact
WHERE EXTRACT (YEAR FROM DataFact) = 2007
    AND EXTRACT (MONTH FROM DataFact) = 9
GROUP BY DataFact
ORDER BY 11
```

R 2	DATAFACT	R 2	VALTOTALA
	01-09-2007		4596401
	02-09-2007		238437
	10-09-2007		424704
	17-09-2007		179565

Figura 7.4. Vânzările zilnice pentru luna septembrie 2007

*Care sunt, în luna august 2007, numărul de facturi și valoarea vânzărilor pentru fiecare client ?*

Gruparea trebuie făcută după client. În acest sens putem folosi oricare cheie primară sau alternativă a tabeli CLIENȚI. Firește, mai nimerită este denumirea clientului, așa încât clauza de grupare va fi *GROUP BY DenCl*. Extragerea vânzărilor corespunzătoare fiecărui client presupune „filiera” de joncționare CLIENTI-FACTURI-LINIIFACT-PRODUSE.

```
SELECT DenCl, COUNT(DISTINCT F.NrFact) AS NrFacturilor,
    TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) AS ValTotala
FROM clienti c
    INNER JOIN facturi f ON c.CodCl=f.CodCl
    INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
    INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE EXTRACT (YEAR FROM DataFact) = 2007
```

---

<sup>1</sup> Sintaxă PostgreSQL/Oracle.

```

AND EXTRACT (MONTH FROM DataFact) = 8
GROUP BY DenCl
ORDER BY 12

```

Ei bine, rezultatul are o ciudățenie. În partea stângă a figurii 7.5 avem interogarea de mai sus și rezultatul său. În partea dreaptă avem o altă interogare ce calculează, și ea, numărul facturilor emise fiecărui client în august 2007, dar fără a mai apela la LINIIFACT și PRODUSE, întrucât interesează numai numărul facturilor, nu și valoarea acestora.

```

SELECT DenCl,
COUNT(DISTINCT F.NrFact) AS NrFacturilor_v1,
TRUNC(
SUM(Cantitate * PretUnit * (1+ProcTVA))
,0) AS ValTotala
FROM clienti c
INNER JOIN facturi f ON c.CodCl=f.CodCl
INNER JOIN liniifact l ON l.NrFact=f.NrFact
INNER JOIN produse p ON l.CodPr=p.CodPr
WHERE EXTRACT (YEAR FROM DataFact) = 2007
AND EXTRACT (MONTH FROM DataFact) = 8
GROUP BY DenCl
ORDER BY 1

```

#	DENCL	#	NRFACTURILOR	#	VALTOTALA
1	Client 1 SRL	9			10040251
2	Client 2 SA	2			233805
3	Client 3 SRL	2			11594166
4	Client 4	2			9479109
5	Client 5 SRL	2			278958
6	Client 6 SA	1			6021706
7	Client 7 SRL	2			263562

```

SELECT DenCl,
COUNT(DISTINCT F.NrFact) AS NrFacturilor_v2
FROM clienti c
INNER JOIN facturi f ON c.CodCl=f.CodCl
WHERE EXTRACT (YEAR FROM DataFact) = 2007
AND EXTRACT (MONTH FROM DataFact) = 8
GROUP BY DenCl
ORDER BY 1

```

#	DENCL	#	NRFACTURILOR_V2
1	Client 1 SRL	9	
2	Client 2 SA	2	
3	Client 3 SRL	2	
4	Client 4	2	
5	Client 5 SRL	4	
6	Client 6 SA	1	
7	Client 7 SRL	2	

Figura 7.5. Facturi pierdute „pe drum”

Nedumerirea apare la numărul facturilor corespunzătoare Clientului 5 SRL. Dacă în prima variantă sunt 2, în a doua apar cu 2 în plus. Încercăm să localizăm problema, și să extragem numai facturile acestui client, prima dată doar prin joncțiunea CLIENTI-FACTURI, a doua oară prin joncțiunea CLIENTI-FACTURI-LINIIFACT – vezi figura 7.6.

Comparând cele două jumătăți ale figurii găsim dezlegarea enigmei: două dintre facturi, 1122 și 2122, nu conțin nici o linie. Astfel, la joncționarea cu LINIIFACT, aceste facturi dispar și, de aici, diferența de 2 din figura 7.5. Așadar, în figura 7.5, ambele numere furnizate sunt corecte, numai că reprezintă lucruri

<sup>2</sup> Sintaxă PostgreSQL/Oracle.

diferite: primul (cel din stânga) furnizează numărul facturilor „corecte”, al doilea numărul total de facturi, indiferent dacă acestea au sau nu vreo linie.

SELECT f.\*
FROM clienti c
INNER JOIN facturi f ON c.CodCl=f.CodCl
WHERE EXTRACT (YEAR FROM DataFact) = 2007
AND EXTRACT (MONTH FROM DataFact) = 8
AND DenCl='Client 5 SRL'
ORDER BY NrFact

NrFact	DataFact	CodCl	OBS
1112	01-08-2007	1005	Probleme cu transportul
1122	07-08-2007	1005	(null)
2112	14-08-2007	1005	Probleme cu transportul
2122	22-08-2007	1005	(null)

SELECT If.\*
FROM clienti c
INNER JOIN facturi f ON c.CodCl=f.CodCl
INNER JOIN liniifact If ON If.NrFact=f.NrFact
WHERE EXTRACT (YEAR FROM DataFact) = 2007
AND EXTRACT (MONTH FROM DataFact) = 8
AND DenCl='Client 5 SRL'
ORDER BY NrFact

NrFact	Linie	CodPr	Cantitate	PretUnit
1112	1	2	80	1030
1112	2	3	40	750
2112	1	2	85	1030
2112	2	3	65	750

Figura 7.6. Două facturi nu ai nici o linie

*Care este restul de încasat al fiecărei facturi ?*

Așa cum reiese din figura 7.7, soluția următoare oferă un rezultat incomplet. Lipsesc facturile care nu au nici o tranșă de încasare. Pentru rezolvarea cazului avem nevoie de clauza HAVING, dar și de joncțiunea externă, după cum vom vedea în capitolul următor.

```

SELECT f.NrFact, SUM(Cantitate * PretUnit * (1+ProcTVA)) /
      COUNT(DISTINCT i.CodInc) AS Facturat,
      SUM(Transa) / MAX(If.Linie) AS Incasat
FROM clienti c
  INNER JOIN facturi f ON c.CodCl=f.CodCl
  INNER JOIN liniifact If ON If.NrFact=f.NrFact
  INNER JOIN produse p ON If.CodPr=p.CodPr
  INNER JOIN incasfact i ON f.NrFact=i.NrFact
GROUP BY f.NrFact3

```

NrFact	Facturat	Incasat
1111	4346037,5	53996
1112	125516	125516
1113	106275	106275
1117	222050	23204
1118	201975	201975
1120	97664	7315

Figura 7.7. Valorile facturată și încasată pentru fiecare factură – rezultat incomplet

<sup>3</sup> Sintaxă valabilă în toate cele patru servere BD.

*Care sunt vânzările, cantitativ și valoric, pentru fiecare produs ?*

De data aceasta, gruparea se face după valorile atributului DenPr, iar în joncțiune nu avem nevoie decât de LINIIFACT și PRODUSE:

```
SELECT DenPr,
       SUM(Cantitate) AS Cantitativ,
       TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) AS Valoric
FROM liniifact lf
     INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY DenPr4
```

Rezultatul este cel din figura 7.8.

R 2	DENPR	R 2	CANTITATIV	R 2	VALORIC
	Produs 3		290		251685
	Produs 2		2554		2784023
	Produs 4		190		301657
	Produs 1		874		988533
	Produs 5		5580		44844436

Figura 7.8. Vânzările cantitative și valorice pe produse

## 7.2. Gruparea după două sau mai multe criterii

O informație esențială care lipsește din figura 7.8 este unitatea de măsură, fără de care nu putem fi siguri dacă totalul cantitativ se referă la cutii, sticle, pachete, baxuri etc. Din păcate, varianta:

```
SELECT DenPr, UM,
       SUM(Cantitate) AS Cantitativ,
       TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) AS Valoric
FROM liniifact lf
     INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY DenPr
```

nu funcționează (vezi figura 7.9), deoarece unitatea de măsură (UM) apare separat de atributul de grupare, nefiind inclus în funcția/funcțiile care se execută la nivelul grupului.

---

<sup>4</sup> Sintaxă DB2, Oracle și PostgreSQL.

Există însă două remedii cât se poate de simple. Mai întâi, se poate concatena unitatea de măsură cu denumirea produsului, și efectua gruparea după această coloană-șir:

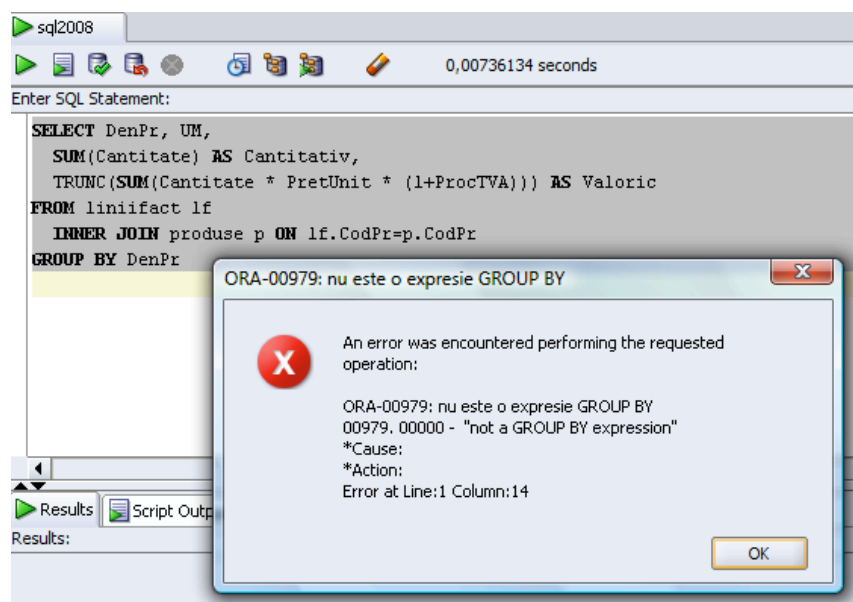


Figura 7.9. Eroare de grupare

```
SELECT DenPr || ' (' || UM || ')' AS "Produs (UM)",
       SUM(Cantitate) AS Cantitativ,
       TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) AS Valoric
FROM liniifact lf
       INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY DenPr || ' (' || UM || ')'
ORDER BY 15
```

cu rezultatul în figura 7.10.

Produs (UM)	CANTITATIV	VALORIC
Produs 1 (buc)	874	988533
Produs 2 (kg)	2554	2784023
Produs 3 (kg)	290	251685
Produs 4 (l)	190	301657
Produs 5 (buc)	5580	44844436

<sup>5</sup> Sintaxă DB2, Oracle și PostgreSQL.



Figura 7.10. Gruparea după șirul obținut prin concatenarea denumirii cu unitatea de măsură

Atenție ! Expresia din GROUP BY -- trebuie să fie identică cu cea din clauza SELECT, altminteri va fi declanșată o eroare (sunt tolerate câteva „deviații”, dar, pentru moment, luăm de bună această poruncă).

Al doilea remediu ține de gruparea după două atribute, DenPr și UM, cele două atribute fiind incluse și în clauza SELECT – vezi și figura 7.11:

```
SELECT DenPr, UM,
       SUM(Cantitate) AS Cantitativ,
       TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) AS Valoric
FROM liniifact lf
      INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY DenPr, UM
ORDER BY 16
```

DENPR	UM	CANTITATIV	VALORIC
Produs 1	buc	874	988533
Produs 2	kg	2554	2784023
Produs 3	kg	290	251685
Produs 4	l	190	301657
Produs 5	buc	5580	44844436

Figura 7.11. Gruparea după două atribute

Numărul de grupuri din figura 7.11 este același cu cel din figura 7.8, întrucât DenPr este cheie alternativă în tabela PRODUSE. Tot datorită faptului că DenPr este cheie alternativă, putem ca, la grupare, să schimbăm ordinea celor două atribute (ne ajută și faptul că ordonarea se face după denumirea produsului):

```
SELECT DenPr, UM,
       SUM(Cantitate) AS Cantitativ,
       TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) AS Valoric
FROM liniifact lf
      INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY UM, DenPr
ORDER BY 17
```

*Care este situația vânzărilor pe clienți și zile în luna septembrie 2007 ?*

<sup>6</sup> Sintaxă DB2, Oracle și PostgreSQL.

<sup>7</sup> Sintaxă DB2, Oracle și PostgreSQL.

Interesează valoarea facturilor emise pe clienți, și, pentru fiecare client, cuantumul zilnic al acestora. Este momentul să folosim o veritabilă grupare după două atribute (interogarea funcționează în Oracle):

```
SELECT DenCl AS DenumireClient, DataFact AS Data,
       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 lf.NrFact=f.NrFact
      INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE EXTRACT (YEAR FROM DataFact) = 2007
      AND EXTRACT (MONTH FROM DataFact) = 9
GROUP BY DenCl, DataFact
ORDER BY 1,28
```

Nu cred că mai are rost în a insista asupra faptului că ordinea atributelor este decisivă asupra modul de grupare și, implicit, a valorilor calculate.

DENUMIRECLIENT	DATA	VINZARI
Client 1 SRL	01-09-2007	4442959
Client 1 SRL	02-09-2007	110907
Client 1 SRL	10-09-2007	287855
Client 1 SRL	17-09-2007	179565
Client 2 SA	02-09-2007	127530
Client 5 SRL	01-09-2007	153442
Client 7 SRL	10-09-2007	136849

Figura 7.12. Vânzările zilnice pentru fiecare client în luna septembrie 2007

*Care este situația vânzărilor fiecărui produs pe fiecare regiune ?*

Cele două atribute de grupare sunt DenPr din tabela PRODUSE și Regiune din tabela JUDEȚE. Clauza FROM este ceva mai impresionantă întrucât trebuie jonctionate nu mai puțin de șase tabele: PRODUSE, LINIIFACT, FACTURI, CLIENTI, CODURI\_POSTALE și JUDEȚE:

```
SELECT DenPr, Regiune,
       TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) 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
```

<sup>8</sup> Sintaxă Oracle și PostgreSQL.

```

INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY DenPr, Regiune
ORDER BY 1,29

```




 DENPR	 REGIUNE	 VINZARI
Produs 1	Moldova	988533
Produs 2	Banat	681086
Produs 2	Moldova	2102937
Produs 3	Banat	151725
Produs 3	Moldova	99960
Produs 4	Moldova	301657
Produs 5	Moldova	44844436

Figura 7.13. Vânzările pe regiuni ale fiecărui produs

*Să se obțină situația vânzărilor pe produse, regiuni și zile în luna septembrie 2007*

Exemplul de față este o bună ocazie de a folosi gruparea după trei atribute, și, implicit, o analiză a vânzărilor pe cele axe tradiționale: sortimentală, teritorială și calendaristică – vezi figura 7.14.

```

SELECT DenPr, Regiune, DataFact AS Zi,
       TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) 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 lf.NrFact=f.NrFact
     INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE EXTRACT (YEAR FROM DataFact) = 2007
      AND EXTRACT (MONTH FROM DataFact) = 9
GROUP BY DenPr, Regiune, DataFact
ORDER BY 1,2,3

```

Fără îndoială, interogările de mai sus constituie un ajutor esențial în analiza vânzărilor oricărei firme (chiar și de stat). Ceea ce lipsește este un ingredient important al oricărui raport de acest gen – subtotalul. Vom recurge, așa cum v-ați obișnuit, la improvizații. Reluăm una din problemele anterioare, modificându-i ușor enunțul:

---

<sup>9</sup> Sintaxă Oracle și PostgreSQL

Să se obțină situația vânzărilor pe clienți și zile în luna septembrie 2007, afișându-se câte un subtotal la nivel de client și un total general.

Ideea improvizației este să "alipim", pentru fiecare client, după vânzările zilnice, câte o linie care să conțină subtotalul. Pentru cele două tipuri de rânduri ale raportului scriem două interogări pe care le conectăm prin UNION:

DENPR	REGIUNE	ZI	VINZARI
Produs 1	Moldova	01-09-2007	67830
Produs 1	Moldova	10-09-2007	124355
Produs 1	Moldova	17-09-2007	132804
Produs 2	Banat	01-09-2007	95429
Produs 2	Banat	10-09-2007	136849
Produs 2	Moldova	01-09-2007	90415
Produs 2	Moldova	02-09-2007	238437
Produs 2	Moldova	10-09-2007	163500
Produs 2	Moldova	17-09-2007	46761
Produs 3	Banat	01-09-2007	58012
Produs 5	Moldova	01-09-2007	4284714

Figura 7.14. Gruparea vânzărilor pe produse, regiuni și zile

```

SELECT DenCl AS DenumireClient,
       DataFact AS Data,
       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 lf.NrFact=f.NrFact
      INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE EXTRACT (YEAR FROM DataFact) = 2007
      AND EXTRACT (MONTH FROM DataFact) = 9
GROUP BY DenCl, DataFact
UNION
SELECT DenCl || ' - SUBTOTAL',
       NULL,
       TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0)
FROM clienti c
      INNER JOIN facturi f ON c.CodCl=f.CodCl
      INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
      INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE EXTRACT (YEAR FROM DataFact) = 2007
      AND EXTRACT (MONTH FROM DataFact) = 9

```

**GROUP BY DenCl****ORDER BY 1,2**

Pentru uni-compatibilitatea rezultatelor celor două SELECT-uri am folosit, în cel de-al doilea, un NULL drept corespondent al DataFact din primul SELECT. Clauza ORDER BY trebuie adăugată la finalul ultimului SELECT.

DENUMIRECLIENT	DATA	VINZARI
Client 1 SRL	01-09-2007	4442959
Client 1 SRL	02-09-2007	110907
Client 1 SRL	10-09-2007	287855
Client 1 SRL	17-09-2007	179565
Client 1 SRL - SUBTOTAL	(null)	5021287
Client 2 SA	02-09-2007	127530
Client 2 SA - SUBTOTAL	(null)	127530
Client 5 SRL	01-09-2007	153442
Client 5 SRL - SUBTOTAL	(null)	153442
Client 7 SRL	10-09-2007	136849
Client 7 SRL - SUBTOTAL	(null)	136849

Figura 7.15. Subtotaluri la nivel de clienți

Dacă trucul funcționează în Oracle, PostgreSQL și, prin schimbarea TRUNC-ului în ROUND și a EXTRACT-ului în DATEPART, și în MS SQL Server, în DB2 avem o problemă. Interogarea următoare:

```
SELECT DenCl AS DenumireClient, DataFact AS Data,
       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 lf.NrFact=f.NrFact
      INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE YEAR(DataFact) = 2007 AND MONTH(DataFact) = 9
GROUP BY DenCl, DataFact
      UNION
SELECT DenCl || ' - SUBTOTAL', NULL,
       TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0)
FROM clienti c
      INNER JOIN facturi f ON c.CodCl=f.CodCl
      INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
      INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE YEAR(DataFact) = 2007 AND MONTH(DataFact) = 9
GROUP BY DenCl ORDER BY 1,2
```

se soldează cu eroarea din figura 7.16, eroare datorată, aparent, „folosirii inadecvate” a NULL-ului.

```

-----
SELECT DenCl || ' - SUBTOTAL', NULL, TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0)
f.NrFact INNER JOIN produse p ON lf.CodPr=p.CodPr WHERE YEAR(DataFact) = 2007 AND MONTH(DataFact) = 9
SQL0206N "NULL" is not valid in the context where it is used.  SQLSTATE=42703
-----

```

Figura 7.16. Eroare DB2 la folosirea NULL-ului în a doua interogare (cea de subtotal)

Soluția este mai simplă decât vă așteptați: se înlocuiește, doar, NULL cu *CAST* (*NULL AS DATE*). În plus, se folosește clauza *AS* în ambele interogări, astfel încât antetul rezultatului este corect<sup>10</sup> – vezi figura 7.17:

DENUMIRECLIENT	DATA	VINZARI
Client 1 SRL	01.09.2007	4.442.959,0000
Client 1 SRL	02.09.2007	110.907,0000
Client 1 SRL	10.09.2007	287.855,0000
Client 1 SRL	17.09.2007	179.565,0000
Client 1 SRL - SUBTOTAL		5.021.287,0000
Client 2 SA	02.09.2007	127.530,0000
Client 2 SA - SUBTOTAL		127.530,0000
Client 3 SRL	07.09.2007	5.819.668,0000
Client 3 SRL - SUBTOTAL		5.819.668,0000
Client 5 SRL	01.09.2007	153.442,0000
Client 5 SRL - SUBTOTAL		153.442,0000
Client 7 SRL	10.09.2007	136.849,0000
Client 7 SRL - SUBTOTAL		136.849,0000

Figura 7.17. Obținerea subtotalurilor și totalului general în DB2

```

SELECT DenCl AS DenumireClient,
       DataFact AS Data,
       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 lf.NrFact=f.NrFact
       INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE YEAR(DataFact) = 2007 AND MONTH(DataFact) = 9
GROUP BY DenCl, DataFact
UNION
SELECT DenCl || ' - SUBTOTAL' AS DenumireClient,
       CAST (NULL AS DATE) Data,
       TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) AS Vinzari
FROM clienti c

```

<sup>10</sup> Reținem, deci, ideea: când, în DB2, o interogare constă în mai multe SELECT-uri reunite, singurul mod de a obține un antet dorit pentru fiecare coloană este să se folosească clauza *AS* în fiecare SELECT.

```

INNER JOIN facturi f ON c.CodCl=f.CodCl
INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE YEAR(DataFact) = 2007 AND MONTH(DataFact) = 9
GROUP BY DenCl, CAST (NULL AS DATE)
ORDER BY 1,2

```

La cele două SELECT-uri mai adăugăm un al treilea pentru afișarea totalului general. Pentru rândul de total general trebuie să avem în vedere ca valoarea coloanei DenumireClient să fie ultima la ordonare. Este motivul pentru care la început vom insera un z mic (oricum, prin restricțiile declarate în capitolul 3, denumirea fiecărui client trebuie să înceapă cu o majusculă) urmat de trei asteriscuri (pur decorative). Un alt element de noutate ține de afișarea subtotalului la începutul fiecărui client, ca în figura 7.18.

z	DENUMIRECLIENT	z	DATA	z	VINZARI
	Client 1 SRL		subtotal (toate zilele)		5021287
	Client 1 SRL		01-09-2007		4442959
	Client 1 SRL		02-09-2007		110907
	Client 1 SRL		10-09-2007		287855
	Client 1 SRL		17-09-2007		179565
	Client 2 SA		subtotal (toate zilele)		127530
	Client 2 SA		02-09-2007		127530
	Client 5 SRL		subtotal (toate zilele)		153442
	Client 5 SRL		01-09-2007		153442
	Client 7 SRL		subtotal (toate zilele)		136849
	Client 7 SRL		10-09-2007		136849
z***	TOTAL GENERAL				5439108

Figura 7.18. Subtotaluri la început de client și un total general

Pentru această formă a rezultatului interogarea anterioară se modifică și în SELECT-ul dedicat rândurilor „curente” - DataFact se va concatena cu spații pentru a se putea afișa, pentru rândurile de subtotal, mesajul „ subtotal (toate zilele)”, mesaj care începe cu un spațiu tocmai pentru a fi primul la ordonare:

```

SELECT DenCl AS DenumireClient, DataFact || ' ' AS Data,
       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 lf.NrFact=f.NrFact
INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE EXTRACT (YEAR FROM DataFact) = 2007
      AND EXTRACT (MONTH FROM DataFact) = 9
GROUP BY DenCl, DataFact

```

```

UNION
    SELECT DenCl ,          ' subtotal (toate zilele) ',
           TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0)
    FROM clienti c
           INNER JOIN facturi f ON c.CodCl=f.CodCl
           INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
           INNER JOIN produse p ON lf.CodPr=p.CodPr
    WHERE EXTRACT (YEAR FROM DataFact) = 2007
           AND EXTRACT (MONTH FROM DataFact) = 9
    GROUP BY DenCl
UNION
    SELECT 'z*** TOTAL GENERAL', ' ',
           TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0)
    FROM facturi f
           INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
           INNER JOIN produse p ON lf.CodPr=p.CodPr
    WHERE EXTRACT (YEAR FROM DataFact) = 2007
           AND EXTRACT (MONTH FROM DataFact) = 9
    ORDER BY 1,2

```

În DB2 trebuie să modificăm, doar, pe ici pe colo, interogarea precedentă (Oracle/PostgreSQL):

```

SELECT ...,
       CAST(DataFact AS CHAR(10)) || ' ' AS Data,
...
FROM ...
WHERE YEAR(DataFact) = 2007 AND MONTH(DataFact) = 9
GROUP BY DenCl, DataFact
UNION
SELECT DenCl AS DenumireClient,
       ' subtotal (toate zilele) ' AS Data,
       TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) AS Vinzari
FROM ...
WHERE YEAR(DataFact) = 2007 AND MONTH(DataFact) = 9
GROUP BY DenCl
UNION
SELECT 'z*** TOTAL GENERAL' AS DenumireClient,
       ' ' AS Data,
       TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) AS Vinzari
FROM ...
WHERE YEAR(DataFact) = 2007 AND MONTH(DataFact) = 9

```



**ORDER BY 1,2**

Ei bine, simțul artistic ne roade rău de tot și ne întrebăm dacă nu am putea delimita începutul datelor despre un client, subtotalul unui client și totalul general ca în figura 7.19. Aveți dreptate ! Am luat-o razna rău de tot (unde mai puneți că în loc de egaluri voiam să pun inițial inimioare). Fraza SELECT (sintaxa Oracle) este sfâșietoare:

DENUMIRECLIENT	DATA	VINZARI
Client 1 SRL	=====	=====
Client 1 SRL	01-09-2007	4442959
Client 1 SRL	02-09-2007	110907
Client 1 SRL	10-09-2007	287855
Client 1 SRL	17-09-2007	179565
Client 1 SRL	s-----	s-----
Client 1 SRL	subtotal (toate zilele)	5021287
Client 2 SA	=====	=====
Client 2 SA	02-09-2007	127530
Client 2 SA	s-----	s-----
Client 2 SA	subtotal (toate zilele)	127530
Client 5 SRL	=====	=====
Client 5 SRL	01-09-2007	153442
Client 5 SRL	s-----	s-----
Client 5 SRL	subtotal (toate zilele)	153442
Client 7 SRL	=====	=====
Client 7 SRL	10-09-2007	136849
Client 7 SRL	s-----	s-----
Client 7 SRL	subtotal (toate zilele)	136849
z=====	=====	=====
z*** TOTAL GENERAL	+++++	5439108

Figura 7.19. Briz-brizuri pe la subtotaluri

**SELECT DenCl AS DenumireClient,** -- acesta este SELECT-ul pentru rindurile normale

**' ' || DataFact AS Data,**

**LPAD( TRIM (TRAILING FROM**

**CAST(TRUNC(SUM(Cantitate \* PretUnit \* (1+ProcTVA))) AS CHAR(14))**

**), 14, ' ') AS Vinzari**

**FROM clienti c**

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

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

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

**WHERE EXTRACT (YEAR FROM DataFact) = 2007**

**AND EXTRACT (MONTH FROM DataFact) = 9**

```

GROUP BY DenCl, DataFact
      UNION
SELECT DenCl, -- acesta este SELECT-ul pentru subtotaluri
      'subtotal (toate zilele) ',
      LPAD(
        TRIM (TRAILING FROM
        CAST(TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA))) AS CHAR(14))
        ), 14, ' ')
FROM clienti c
      INNER JOIN facturi f ON c.CodCl=f.CodCl
      INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
      INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE EXTRACT (YEAR FROM DataFact) = 2007
      AND EXTRACT (MONTH FROM DataFact) = 9
GROUP BY DenCl
      UNION
SELECT DenCl, -- acesta este SELECT-ul pentru rindul de la inceputul unui client
      ' =====',
      ' ====='
FROM clienti c
      INNER JOIN facturi f ON c.CodCl=f.CodCl
      INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
      INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE EXTRACT (YEAR FROM DataFact) = 2007
      AND EXTRACT (MONTH FROM DataFact) = 9
GROUP BY DenCl
      UNION
SELECT DenCl, -- acesta este SELECT-ul pentru rindul dinaintea unui subtotal
      ' s----- ',
      ' s----- '
FROM clienti c
      INNER JOIN facturi f ON c.CodCl=f.CodCl
      INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
      INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE EXTRACT (YEAR FROM DataFact) = 2007
      AND EXTRACT (MONTH FROM DataFact) = 9
GROUP BY DenCl
      UNION
SELECT 'z*** TOTAL GENERAL', -- acesta este SELECT-ul pentru rindul de total
      general
      ' ++++++',

```

```

        LPAD( TRIM (TRAILING FROM
        CAST(TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA))) AS CHAR(14))
        ), 14, ' ')
FROM facturi f
    INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
    INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE EXTRACT (YEAR FROM DataFact) = 2007
    AND EXTRACT (MONTH FROM DataFact) = 9
UNION
SELECT 'z=====', -- SELECT-ul pentru rindul dinaintea totalui general
    '=====', '====='
FROM dual ORDER BY 1,2

```

La drept vorbind, ceea ce impresionează/intimidează la aceasta interogare este dimensiunea, nu că ar îngloba prea multă inteligență. Acesta este motivul pentru care nici nu ne mai sinchisim să o convertim în sintaxa celorlalte trei SGBD-uri.

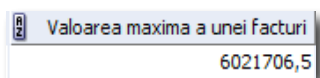
*Care este cea mai mare valoare a unei facturi ?*

Această interogare este prea pretențioasă pentru nivelul de SQL la care ne situăm în prezent. Cu toate acestea, Oracle prezintă o facilități grozavă prin care putem redacta un răspuns: includerea unei funcții în alta (rezultatul este prezentat în figura 7.20):

```

SELECT MAX(SUM(Cantitate * PretUnit * (1+ProcTVA)))
    AS "Valoarea maxima a unei facturi"
FROM liniifact lf INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY NrFact

```



Valoarea maxima a unei facturi
6021706,5

Figura 7.20. Valoarea maximă a unei facturi (Oracle)

Din păcate, majoritatea SGBD-urilor nu „suportă” această facilități – vezi „cazul” PostgreSQL în figura 7.21.

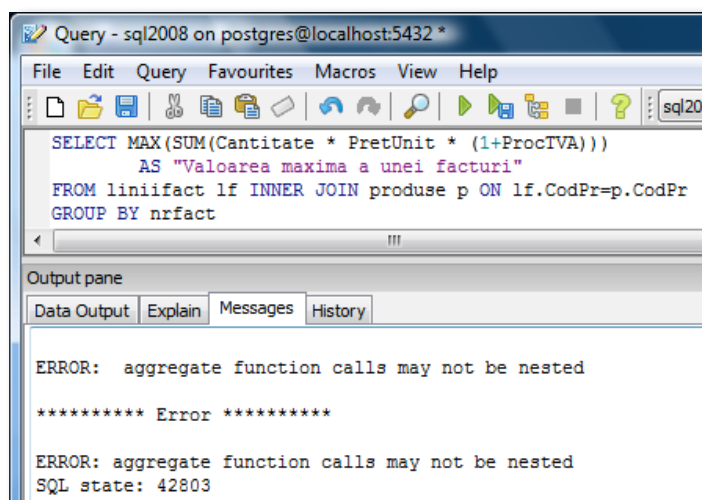


Figura 7.21. Imposibilitatea (în PostgreSQL) includerii unei funcții agregat în altă funcție agregat

De asemenea, cu opțiunile SQL de până acum, doar în Oracle putem afla și numărul uneia (este posibil să fie mai multe) dintre facturile cu această valoare maximă:

```
SELECT MAX(
  TO_CHAR(
    SUM(Cantitate * PretUnit * (1+ProcTVA)),
    '9999999999.99')
  || ', factura nr. ' || NrFact
) AS "Valoarea maxima a unei facturi"
FROM liniifact lf INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY nrfact
```

Valoarea maxima a unei facturi
6021706.50, factura nr. 1114

Figura 7.22. Valoarea maximă pentru o factură (și numărul acesteia sau uneia (dacă sunt mai multe)) – soluție Oracle

### 7.3. Gruparea după funcții și expresii

A doua interogare din paragraful anterior - cea în care clauza GROUP BY are forma *GROUP BY DenPr || ' (' || UM || ')'* - ne-a sugerat că gruparea se poate realiza nu numai pentru attribute, ci și după valorile anumitor expresii. Este și ceea ce vrem să dezvoltăm în cele ce urmează.

*Care este valoarea vânzărilor din fiecare zi a săptămânii ?*

Cu alte cuvinte, ne interesează o statistică prin care să putem compara cât s-a vândut luna cu cât s-a vândut marțea s.a.m.d., adică ceva de genul situației din figura 7.23.

ZI_SAPTAMINA	VINZARI_ZI_SAPT
luni	604269
marți	25894871
miercuri	10847292
joi	745370
vineri	6041718
sâmbătă	4798376
duminică	238437

Figura 7.23. Vânzările pe zile ale săptămânii

Expresia de grupare are la bază discuția din paragrafele 6.3 și 6.5 (figura 6.37). În Oracle/PostgreSQL putem folosi varianta:

```
SELECT TO_CHAR(DataFact, 'day') AS Zi_Saptamina,
       TRUNC(
         SUM(Cantitate * PretUnit * (1+ProcTVA))
       ,0) AS Vinzari_Zi_Sapt
FROM facturi f
      INNER JOIN liniifact lf ON f.Nrfact=lf.NrFact
      INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY TO_CHAR(DataFact, 'day')
```

Rezultatul este încurajator – vezi figura 7.24. Entuziasmul ne trece repede, întrucât realizăm că nu prea ne pricepem cum să ordonăm zilele. Cu ceea ce știm cu siguranță este că, în acest moment, nu suntem în stare să încropim o clauză ORDER BY corespunzătoare.

ZI_SAPTAMINA	VINZARI_ZI_SAPT
miercuri	10847292
duminica	6058105
marti	25894871
joi	745370
vineri	222050
sâmbata	4798376
luni	604269

Figura 7.24. Zilele săptămânii ordonate anapoda

Ceva speranțe ar fi în PostgreSQL, întrucât, așa după cum am văzut în paragraful 6.3, în funcția EXTRACT poate fi folosit argumentul DOW – Day of the Week:

```
SELECT EXTRACT (DOW FROM DataFact) AS Zi_Saptamina,
       TRUNC(
           SUM(Cantitate * PretUnit * (1+ProcTVA))
       ,0) AS Vinzari_Zi_Sapt
FROM facturi f
     INNER JOIN liniifact lf ON f.Nrfact=lf.NrFact
     INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY EXTRACT (DOW FROM DataFact)
ORDER BY EXTRACT (DOW FROM DataFact)
```

Ceea ce obținem la execuția acestei interogări - figura 7.25 - ne duce cu gândul la celebrele bancuri cu vestea bună și vestea rea. Vestea bună este că zilele sunt ordonate, iar vestea proastă este că nu dispunem de numele fiecărei zile, ci de numărul său – 0 pentru duminică, 1 pentru luni, ... și 6 pentru sâmbătă.

zi_saptamina double precis	vinzari_zi_sapt numeric
0	6058105
1	604269
2	25894871
3	10847292
4	745370
5	222050
6	4798376

Figura 7.25. Zilele săptămânii ordonate bine, dar cu număr în loc de denumire

În afară de a ne manifesta nemulțumirea pentru inexistența unei clauze (de tip CDOW) care să furnizeze denumirea zilei din săptămână, suntem tentați la o improvizație în care să facem gruparea după denumirea zilei din săptămână, dar ordonarea să se facă după numărul acesteia:

```
SELECT TO_CHAR(DataFact, 'day') AS Zi_Saptamina,
       TRUNC(
           SUM(Cantitate * PretUnit * (1+ProcTVA))
       ,0) AS Vinzari_Zi_Sapt
FROM facturi f INNER JOIN liniifact lf ON f.Nrfact=lf.NrFact
     INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY TO_CHAR(DataFact, 'day')
ORDER BY EXTRACT (DOW FROM DataFact)
```

Din păcate, PostgreSQL-ul tratează cu ostilitate improvizația noastră - vezi figura 7.26, așa că nu ne rămâne decât să așteptăm până în paragraful 8.5, unde vom folosi o structură CASE care ne va face ordonarea după cum vom porunci.

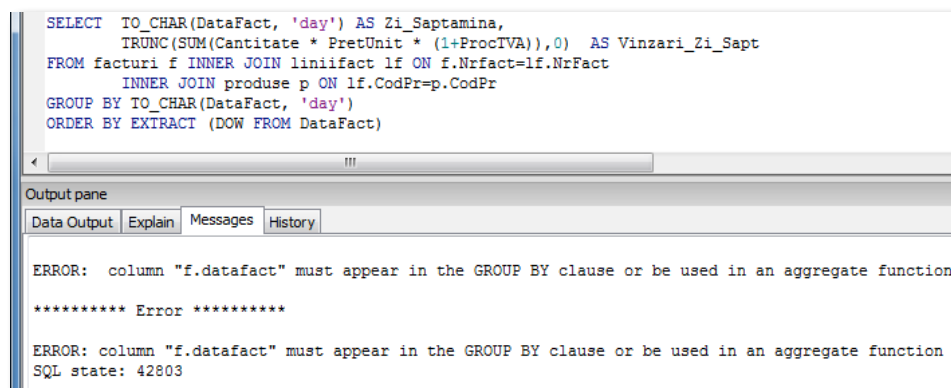


Figura 7.26. Improvizație ratată în PostgreSQL

Problema se prezintă similar și în MS SQL Server, întrucât interogarea:

```

SELECT DATENAME(WEEKDAY, DataFact) AS Zi_Saptamina,
       ROUND(
           SUM(Cantitate * PretUnit * (1+ProcTVA))
           ,0,1) AS Vinzari_Zi_Sapt
FROM facturi f INNER JOIN liniifact lf ON f.NrFact=lf.NrFact
       INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY DATENAME(WEEKDAY, DataFact)

```

obține rezultatul din figura 7.27, iar încercarea de a modifica ordonarea după dispunerea firească a zilelor din săptămână:

Zi_Saptamina	Vinzari_Zi_Sapt
Friday	222050.0000
Monday	604269.0000
Saturday	4798376.0000
Sunday	6058105.0000
Thursday	745370.0000
Tuesday	25894871.0000
Wednesday	10847292.0000

Figura 7.27. Zilele săptămânii ordonate implicit în MS SQL Server

```

SELECT DATENAME(WEEKDAY, DataFact) AS Zi_Saptamina, ROUND(
           SUM(Cantitate * PretUnit * (1+ProcTVA)) ,0,1) AS Vinzari_Zi_Sapt
FROM facturi f INNER JOIN liniifact lf ON f.NrFact=lf.NrFact
       INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY DATENAME(WEEKDAY, DataFact)
ORDER BY DATEPART(WEEKDAY, DataFact)

```

are soarta din PostgreSQL. La fel stau lucrurile și în DB2 (funcțiile folosite aici fiind DAYOFWEEK\_ISO și DAYNAME).

*Să se afle numărul de produse vândute, pe următoarele intervale ale prețului unitar de vânzare: sub 1000 RON, între 1000 și 1999, 2000 și 2999 s.a.m.d.*

Chiar dacă enunțul nu este dintre cele mai fericite, genul acesta de problemă este unul întâlnit în practică, dificultatea ținând de modul grupare. Aici ne interesează pe câte linii, din toate facturile, prețul unitar a fost între 0 și 999 RON, între 1000 și 1999, între 2000 și 2999 s.a.m.d., ca în figura 7.28.

Pret unitar peste ... (RON)	Nr.aparitii
0	21
1000	26
6000	3
7000	6

Figura 7.28. Grupare pe intervale ale prețului unitar

Cheia soluției stă, după cum vă așteptați, în expresia de grupare care este *TRUNC(PretUnit,-3)*. Argumentul -3 aplicat funcției TRUNC asigură, după cum am văzut în paragraful 6.1, trunchierea la ordinul miilor:

```
SELECT TRUNC(PretUnit,-3) AS "Pret unitar peste ... (RON)",
       COUNT(*) AS "Nr.aparitii"
FROM liniifact
GROUP BY TRUNC(PretUnit,-3)
ORDER BY 1 11
```

Singura nemulțumire pe care o putem avea este legată de modul de precizare a intervalelor, dar nu trebuie să fim prea tipicari.

*Să se afle numărul de facturi întocmite în fiecare lună (calendaristică).*

Deși noi am populat tabelele FACTURI și LINIIFACT cu câteva înregistrări doar pentru lunile august și septembrie 2007, trebuie avut în vedere că, în practică, baza de date se folosește ani și chiar decenii întregi, așa că trebuie să asigurăm diferențierea datelor din septembrie 2007, de cele din septembrie 2006, septembrie 2005 etc.

Pe scurt, am avea cel puțin două soluții de grupare. În prima, clauza GROUP BY va conține două expresii/funcții care vor extrage anul și luna (interogarea următoare este radactată pe sintaxa Oracle/PostgreSQL):

```
SELECT EXTRACT (YEAR FROM DataFact) AS An,
```

<sup>11</sup> Sintaxă DB2, Oracle și PostgreSQL. Varianta corectă SQL Server este: *SELECT ROUND(PretUnit,-3,1) AS "Pret unitar peste ... (RON)", COUNT(\*) AS "Nr.aparitii" FROM liniifact GROUP BY ROUND(PretUnit,-3,1) ORDER BY 1*



```

        EXTRACT (MONTH FROM DataFact) AS Luna,
        COUNT(*) AS Nr_facturi
    FROM facturi
    GROUP BY EXTRACT (YEAR FROM DataFact),
             EXTRACT (MONTH FROM DataFact)
    ORDER BY 1,2

```

A doua (tot Oracle/PostgreSQL) va folosi o funcție de extragere a lunii din data facturii, care poate fi TO\_CHAR(DataFact, 'YYYY-MM'):

```

    SELECT TO_CHAR(DataFact, 'YYYY-MM') AS An_Luna, COUNT(*) AS
    Nr_facturi
    FROM facturi
    GROUP BY TO_CHAR(DataFact, 'YYYY-MM')
    ORDER BY 1,2

```

sau TRUNC(DataFact,'MONTH') în Oracle, respectiv DATE\_TRUNC( 'MONTH', DataFact) în PostgreSQL:

```

    SELECT TRUNC(DataFact,'MONTH') AS "Dupa...", COUNT(*) AS Nr_facturi
    FROM facturi
    GROUP BY TRUNC(DataFact,'MONTH')
    ORDER BY 1

```

Diferențele dintre cele trei variante sunt doar de fațadă – vezi figura 7.29.

AN	LUNA	NR_FACTURI	AN_LUNA	NR_FACTURI	Dupa...	NR_FACTURI
2007	8	22	2007-08	22	01-08-2007	22
2007	9	7	2007-09	7	01-09-2007	7
2007	10	1	2007-10	1	01-10-2007	1

Figura 7.29. Trei soluții la o problemă simplă de grupare pe luni calendaristice

O variantă funcțională în MS SQL Server este:

```

    SELECT DATEPART(YEAR, DataFact) AS An,
           DATEPART(MONTH, DataFact) AS Luna, COUNT(*) AS Nr_facturi
    FROM facturi
    GROUP BY DATEPART(YEAR, DataFact), DATEPART(MONTH, DataFact)
    ORDER BY 1,2

```

iar în DB2:

```

    SELECT YEAR(DataFact) AS An, MONTH(DataFact) AS Luna,
           COUNT(*) AS Nr_facturi
    FROM facturi
    GROUP BY YEAR(DataFact), MONTH(DataFact)
    ORDER BY 1,2

```

*Să se calculeze vânzările săptămânale.*

Neavând prea multe înregistrări în tabele, am renunțat la filtrarea pe luni, trimestre, semestre sau ani. Pentru a onora problema, trebuie să găsim o soluție prin care să determinăm începutul unei săptămâni (după aceea, aflarea sfârșitului săptămânii e floare la o ureche oarecare). Funcția TRUNC care, încet-încet, ne devine chiar simpatică, ne pune la dispoziția opțiunile 'D', 'DAY' și 'DY' care furnizează data de început a unei săptămâni (însă trebuie corectată cu o zi, întrucât săptămâna începe în Oracle cu duminică):

```
SELECT TRUNC(DataFact, 'd') + INTERVAL '1' DAY AS "De la...",
       TRUNC(DataFact, 'd') + INTERVAL '7' DAY AS "Pana la...",
       TRUNC(
           SUM(Cantitate * PretUnit * (1+ProcTVA))
           ,0) AS Vinzari_Sapt
FROM facturi f
     INNER JOIN liniifact lf ON f.Nrfact=lf.NrFact
     INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY TRUNC(DataFact, 'd') + INTERVAL '1' DAY,
         TRUNC(DataFact, 'd') + INTERVAL '7' DAY
ORDER BY 1
```

Soluția funcționează în Oracle (vezi în figura 7.30), însă în PostgreSQL trebuie să folosim, ca și mai sus, funcția DATE\_TRUNC (fără a mai avea nevoie de corecția de o zi):

RZ	De la...	RZ	Pana la...	RZ	VINZARI_SAPT
	31-07-2007		06-08-2007		11301510
	07-08-2007		13-08-2007		10610000
	14-08-2007		20-08-2007		5439108
	21-08-2007		27-08-2007		10560939
	28-08-2007		03-09-2007		4834839
	11-09-2007		17-09-2007		424704
	18-09-2007		24-09-2007		179565
	02-10-2007		08-10-2007		5819668

Figura 7.30. Vânzări săptămânale

```
SELECT DATE_TRUNC('week',DataFact) AS "De la...",
       DATE_TRUNC('week',DataFact) + INTERVAL '6 DAYS'
       AS "Pana la...",
       TRUNC(
           SUM(Cantitate * PretUnit * (1+ProcTVA))
           ,0) AS Vinzari_Sapt
FROM facturi f
     INNER JOIN liniifact lf ON f.Nrfact=lf.NrFact
     INNER JOIN produse p ON lf.CodPr=p.CodPr
```

```
GROUP BY DATE_TRUNC('week', DataFact), DATE_TRUNC('week', DataFact) +
        INTERVAL '6 DAYS'
ORDER BY 1
```

Pentru a furniza răspunsul la această problemă în MS SQL SERVER trebuie să ținem cont că nu beneficiem de elegantele funcții TRUNC/DATE\_TRUNC, ci doar de DATEPART și ADDDATE. Astfel, putem afla săptămâna anului în care pică fiecare factură, însă aflarea datei de început și de sfârșit a fiecărei săptămâni necesită adunarea la data de 1 ianuarie (dată obținută prin concatenarea anului facturii cu '-01-01' (1 ianuarie) și transformarea șirului în dată calendaristică) a numărului săptămânii determinat cu DATEPART(WEEK, DataFact). Datorită faptului că, prin adunarea la prima zi din an a unui număr de săptămâni, se determină prima zi a săptămânii viitoare (din cauza setării curente a primei zile din săptămână), „poziționarea” zilei de început și sfârșit a fiecărei săptămâni necesită un „deplasament: de 7 zile, respectiv 1 zi „înainte” (vezi figura 7.31):

```
SELECT NrFact, DataFact, DATEPART(WEEK, DataFact) AS Saptamina_Nr,
        CAST (STR(YEAR(DataFact),4)+'-01-01' AS SMALLDATETIME)
        AS Zi_Inceput_An,
        DATEADD (WEEK, DATEPART(WEEK, DataFact),
        CAST (STR(YEAR(DataFact),4)+'-01-01' AS SMALLDATETIME)) - 7
        AS Prima_zi_a_sapt,
        DATEADD (WEEK, DATEPART(WEEK, DataFact),
        CAST (STR(YEAR(DataFact),4)+'-01-01' AS SMALLDATETIME)) - 1
        AS Ultima_zi_a_sapt
FROM facturi f
```

NrFact	DataFact	Saptamina_Nr	Zi_Inceput_An	Prima_zi_a_sapt	Ultima_zi_a_sapt
1111	2007-08-01 00:00:00	31	2007-01-01 00:00:00	2007-07-30 00:00:00	2007-08-05 00:00:00
1112	2007-08-01 00:00:00	31	2007-01-01 00:00:00	2007-07-30 00:00:00	2007-08-05 00:00:00
1113	2007-08-01 00:00:00	31	2007-01-01 00:00:00	2007-07-30 00:00:00	2007-08-05 00:00:00
1114	2007-08-01 00:00:00	31	2007-01-01 00:00:00	2007-07-30 00:00:00	2007-08-05 00:00:00
1115	2007-08-02 00:00:00	31	2007-01-01 00:00:00	2007-07-30 00:00:00	2007-08-05 00:00:00
1116	2007-08-02 00:00:00	31	2007-01-01 00:00:00	2007-07-30 00:00:00	2007-08-05 00:00:00
1117	2007-08-03 00:00:00	31	2007-01-01 00:00:00	2007-07-30 00:00:00	2007-08-05 00:00:00
1118	2007-08-04 00:00:00	31	2007-01-01 00:00:00	2007-07-30 00:00:00	2007-08-05 00:00:00
1119	2007-08-07 00:00:00	32	2007-01-01 00:00:00	2007-08-06 00:00:00	2007-08-12 00:00:00
1120	2007-08-07 00:00:00	32	2007-01-01 00:00:00	2007-08-06 00:00:00	2007-08-12 00:00:00
1121	2007-08-07 00:00:00	32	2007-01-01 00:00:00	2007-08-06 00:00:00	2007-08-12 00:00:00
1122	2007-08-07 00:00:00	32	2007-01-01 00:00:00	2007-08-06 00:00:00	2007-08-12 00:00:00

Figura 7.31. “Manevre săptămânale” în SQL Server

Pe baza acestor aranjamente obținem aceleași vânzări săptămânale ca în figura 7.30 folosind interogarea:

```
SELECT DATEPART(WEEK, DataFact) AS Saptamina_Nr,
        DATEADD (WEEK, DATEPART(WEEK, DataFact),
        CAST (STR(YEAR(DataFact),4)+'-01-01' AS SMALLDATETIME)) - 7
```

```

        AS Prima_zi_a_sapt,
DATEADD (WEEK, DATEPART(WEEK, DataFact),
        CAST (STR(YEAR(DataFact),4)+'-01-01' AS SMALLDATETIME)) -1
        AS Ultima_zi_a_sapt,
ROUND(
        SUM(Cantitate * PretUnit * (1+ProcTVA))
        ,0,1) AS Vinzari_Sapt
FROM facturi f
        INNER JOIN liniifact lf ON f.Nrfact=lf.NrFact
        INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY DATEPART(WEEK, DataFact),
        DATEADD (WEEK, DATEPART(WEEK, DataFact),
        CAST (STR(YEAR(DataFact),4)+'-01-01' AS SMALLDATETIME)) - 7,
        DATEADD (WEEK, DATEPART(WEEK, DataFact),
        CAST (STR(YEAR(DataFact),4)+'-01-01' AS SMALLDATETIME)) -1

```

Analog, putem face grupări și pe trimestre, opțiunea-șablon din funcția Oracle TRUNC fiind 'Q' (de la Quarter – trimestru)<sup>12</sup>:

```

SELECT TRUNC(DataFact, 'Q') AS "Data_inceput_Trimestru",
        TRUNC(DataFact, 'Q') + INTERVAL '3' MONTH -
        INTERVAL '1' DAY AS "Data_sfirsit_Trimestru",
TRUNC(
        SUM(Cantitate * PretUnit * (1+ProcTVA))
        ,0) AS Vinzari_Sapt
FROM facturi f INNER JOIN liniifact lf ON f.Nrfact=lf.NrFact
        INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY TRUNC(DataFact, 'Q'), TRUNC(DataFact, 'Q') +
        INTERVAL '3' MONTH - INTERVAL '1' DAY
ORDER BY 1

```

Din păcate, plăcerea ne este stricată de faptul că, la acest moment, avem date doar pentru trimestrele III și IV din 2007 – vezi figura 7.32.

Data_inceput_Trimestru	Data_sfirsit_Trimestru	VINZARI_SAPT
01-07-2007	30-09-2007	43350667
01-10-2007	31-12-2007	5819668

Figura 7.32. Vânzări trimestriale

<sup>12</sup> Dacă înlocum TRUNC(DataFact, 'Q') cu DATE\_TRUNC('quarter', DataFact) obținem varianta funcțională în PostgreSQL

În SQL Server varianta corectă este:

```
SELECT DATEPART(QUARTER, DataFact) AS Saptamina_Nr,
       DATEADD (QUARTER, DATEPART(QUARTER, DataFact)-1,
               CAST (STR(YEAR(DataFact),4)+'-01-01' AS SMALLDATETIME))
               AS Prima_zi_a_trimestrului,
       DATEADD (QUARTER, DATEPART(QUARTER, DataFact),
               CAST (STR(YEAR(DataFact),4)+'-01-01' AS SMALLDATETIME)) -1
               AS Ultima_zi_a_trimestrului,
       ROUND(
               SUM(Cantitate * PretUnit * (1+ProcTVA))
               ,0,1) AS Vinzari_Semestriale
FROM facturi f
       INNER JOIN liniifact lf ON f.NrFact=lf.NrFact
       INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY DATEPART(QUARTER, DataFact),
       DATEADD (QUARTER, DATEPART(QUARTER, DataFact)-1,
               CAST (STR(YEAR(DataFact),4)+'-01-01' AS SMALLDATETIME)),
       DATEADD (QUARTER, DATEPART(QUARTER, DataFact),
               CAST (STR(YEAR(DataFact),4)+'-01-01' AS SMALLDATETIME)) -1
```

Sintaxa DB2 de obținere a rezultatului din figura 7.30 este:

```
SELECT DATE(CAST (YEAR(DataFact) AS CHAR(4)) || '-01-01') +
       ((WEEK_ISO(DataFact)-1) * 7) DAYS AS "De la...",
       DATE(CAST (YEAR(DataFact) AS CHAR(4)) || '-01-01') +
       ((WEEK_ISO(DataFact)) * 7) DAYS AS "Pina la...",
       ROUND(
               SUM(Cantitate * PretUnit * (1+ProcTVA))
               ,0) AS Vinzari_Sapt
FROM facturi f
       INNER JOIN liniifact lf ON f.NrFact=lf.NrFact
       INNER JOIN produse p ON lf.CodPr=p.CodPr
GROUP BY YEAR(DataFact),WEEK_ISO(DataFact),
       DATE(CAST (YEAR(DataFact) AS CHAR(4)) || '-01-01') +
       WEEK_ISO(DataFact) DAYS,
       DATE(CAST (YEAR(DataFact) AS CHAR(4)) || '-01-01') +
       ((WEEK_ISO(DataFact)) * 7) DAYS
```

## 7.4. Clauza HAVING

Cea mai simplă definiție ar suna cam așa: clauza HAVING este WHERE-ul ce operează la nivel de grupuri (un fel de a zice că „HAVING este pentru noi... un fel de WHERE doi”). Dacă WHERE acționează la nivel de *tuplu*, selectând acele linii care îndeplinesc o condiție specificată, HAVING permite specificarea unor condiții

de selecție care se aplică *grupurilor* de linii create prin clauza GROUP BY. Formatul general al unei interogări ce conține clauza HAVING este:

**SELECT** coloană 1, coloană 2, .... , coloană m

**FROM** tabelă

**WHERE** predicat-pentru-tupluri

**GROUP BY** coloană-de-grupare

**HAVING** predicat-pentru-grupuri

*Care sunt zilele în care s-au întocmit cel puțin trei facturi ?*

În al doilea exemplu din paragraful anterior (și figurile 7.2 și 7.3) am văzut mecanismul de grupare care ne permitea să aflăm numărul de facturi întocmit în fiecare zi cu vânzări. La pașii derulați într-o operațiune de grupare se mai adaugă unul prin care, dintre toate grupurile, se extrag în rezultatul final doar cele care îndeplinesc condiția specificată în clauza HAVING (vezi figura 7.33):

**SELECT** DataFact AS "Zi", COUNT(\*) AS "Numar facturi"

**FROM** facturi

**GROUP BY** DataFact

**HAVING** COUNT(\*) >= 3

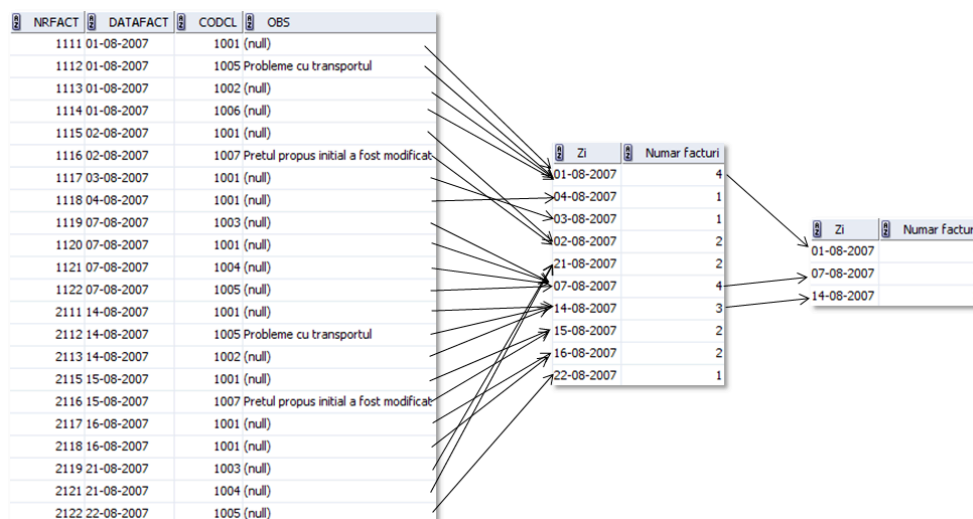


Figura 7.33. Logica execuției clauzelor GROUP BY și HAVING

În lipsa clauzei ORDER BY nu putem fi siguri de ordinea în care vor apărea rândurile în rezultat, cu atât mai puțin ordinea de constituire a grupurilor.

*Care sunt zilele din luna septembrie 2007 în care valoarea (cu TVA a) vânzărilor este mai mare ca 300000 RON ?*

Rezultatul interogării care urmează (valabilă în Oracle și PostgreSQL) ar trebui să conțină una sau mai multe linii dintre cele din figura 7.4:

```

SELECT DataFact, TRUNC(
    SUM(Cantitate * PretUnit * (1+ProcTVA))
    ,0) AS ValTotala
FROM liniifact lf
    INNER JOIN produse p ON lf.CodPr=p.CodPr
    INNER JOIN facturi f ON lf.NrFact=f.NrFact
WHERE EXTRACT (YEAR FROM DataFact) = 2007
    AND EXTRACT (MONTH FROM DataFact) = 9
GROUP BY DataFact
HAVING TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) > 300000
ORDER BY 1

```

Așa și este, dacă examinăm figura 7.34.

DATAFACT	VALTOTALA
01-09-2007	4596401
10-09-2007	424704

Figura 7.34. Zilele cu vânzări peste 300000 RON

*Pentru care clienți și în ce zile din luna septembrie 2007 vânzările au fost mai mari de 150000 RON ?*

Faptul că avem o grupare după două atribute (DenCl și DataFact) nu influențează cu nimic clauza HAVING. Ne oprim doar la sintaxa Oracle/PostgreSQL a interogării la execuția căreia obținem figura 7.35.

```

SELECT DenCl AS DenumireClient,
    DataFact AS Data,
    TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA))) AS Vinzari
FROM clienti c
    INNER JOIN facturi f ON c.CodCl=f.CodCl
    INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
    INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE EXTRACT (YEAR FROM DataFact) = 2007
    AND EXTRACT (MONTH FROM DataFact) = 9
GROUP BY DenCl, DataFact
HAVING TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA))) > 150000
ORDER BY 1,2

```

DENUMIRECLIENT	DATA	VINZARI
Client 1 SRL	01-09-2007	4442959
Client 1 SRL	10-09-2007	287855
Client 1 SRL	17-09-2007	179565
Client 5 SRL	01-09-2007	153442

Figura 7.35. Clienții și zilele din septembrie 2007 pentru care vânzările au fost peste pragul de 150000

Unul dintre reproșurile aduse SQL-ului din „tabăra” relaționalului este că, pentru aceeași problemă există o sumedenie de soluții, ceea ce ar genera confuzii (în termeni elevați, se vorbește de *supraîncărcare semantică*). Eu, unul, nu împărtășesc această supărare, ba, din contra, sunt mulțumit că am de ales. Să luăm un exemplu simplu pe care să formulăm două soluții echivalente, din punctul de vedere al rezultatului.

Să se afișeze vânzările zilnice din luna septembrie 2007 corespunzătoare Clientului 1 SRL, rezultatul având forma propusă în figura 7.36.

DENUMIRECLIENT	DATA	VINZARI
Client 1 SRL	01-09-2007	4442959
Client 1 SRL	02-09-2007	110907
Client 1 SRL	10-09-2007	287855
Client 1 SRL	17-09-2007	179565

Figura 7.36. Vânzările zilnice către Client 1 SRL pe luna septembrie 2007

Ne oprim tot la sintaxa PostgreSQL/Oracle.

Soluția 1 – filtrarea clientului în clauza WHERE și grupare după (*DenCl*, *DataFact*):

```
SELECT DenCl AS DenumireClient,
       DataFact AS Data,
       TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA))) AS Vinzari
FROM clienti c
     INNER JOIN facturi f ON c.CodCl=f.CodCl
     INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
     INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE EXTRACT (YEAR FROM DataFact) = 2007 AND EXTRACT (MONTH
       FROM DataFact) = 9 AND DenCl = 'Client 1 SRL'
GROUP BY DenCl, DataFact
ORDER BY 1,2
```

Soluția 2 – grupare tot după (*DenCl*, *DataFact*), însă filtrarea clientului se face în clauza HAVING:

```
SELECT DenCl AS DenumireClient,
       DataFact AS Data,
       TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA))) AS Vinzari
FROM clienti c
     INNER JOIN facturi f ON c.CodCl=f.CodCl
     INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
```



```

INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE EXTRACT (YEAR FROM DataFact) = 2007
      AND EXTRACT (MONTH FROM DataFact) = 9
GROUP BY DenCl, DataFact
HAVING DenCl = 'Client 1 SRL'
ORDER BY 1,2

```

Chiar dacă cele două soluții sunt echivalente ca rezultat, consumul lor de resurse este unul diferit, și putem spune că, într-o aplicație „aglomerată” (de mare complexitate și cu mulți utilizatori ce lucrează, direct sau indirect, cu baza de date) alegerea uneia în detrimentul celeilalte nu este întâmplătoare.

*În ce zile s-au emis mai multe facturi decât pe 2 august 2007 ?*

Problema este ceva mai exotică, iar soluția este una pe măsură:

```

SELECT F1.DataFact AS Zi1, COUNT(DISTINCT F1.NrFact) AS Nr_Facturilor1,
      F2.DataFact AS Zi2, COUNT(DISTINCT F2.NrFact) AS Nr_Facturilor2
FROM facturi F1, facturi F2
WHERE F2.DataFact = DATE'2007-08-02'
GROUP BY F1.DataFact, F2.DataFact
HAVING COUNT(DISTINCT F1.NrFact) > COUNT(DISTINCT F2.NrFact)

```

	F1.NrFact	F1.DataFact	F2.NrFact	F2.DataFact
Grup 1	1111	01-08-2007	1115	02-08-2007
	1111	01-08-2007	1116	02-08-2007
	1112	01-08-2007	1115	02-08-2007
	1112	01-08-2007	1116	02-08-2007
	1113	01-08-2007	1115	02-08-2007
	1113	01-08-2007	1116	02-08-2007
	1114	01-08-2007	1115	02-08-2007
	1114	01-08-2007	1116	02-08-2007
Grup 2	1115	02-08-2007	1115	02-08-2007
	1115	02-08-2007	1116	02-08-2007
	1116	02-08-2007	1115	02-08-2007
	1116	02-08-2007	1116	02-08-2007
Grup 3	1117	03-08-2007	1115	02-08-2007
	1117	03-08-2007	1116	02-08-2007
Grup 4	1118	04-08-2007	1115	02-08-2007
	1118	04-08-2007	1116	02-08-2007
...	...	...	...	...

Figura 7.37. Produs cartezian al două instanțe ale tabelului FACTURI, urmat de selectarea, în instanța a doua, numai a facturilor de pe 2 august (parțial)

Să zăbovim asupra logicii acestei interogări<sup>13</sup>. Mai întâi se efectuează produsul cartezian pentru două instanțe (F1 și F2) ale tabelului FACTURI, eliminându-se pentru F2 liniile în care data este alta decât 2 august 2007 (vezi), adică:

```
SELECT F1.NrFact AS "F1.NrFact", F1.DataFact AS "F1.DataFact",
       F2.NrFact AS "F2.NrFact", F2.DataFact AS "F2.DataFact"
FROM facturi F1, facturi F2
WHERE F2.DataFact = DATE'2007-08-02'
ORDER BY F1.DataFact, F1.NrFact, F2.DataFact, F2.NrFact
```

O parte din rezultatul acestui SELECT „intermediar” este afișată în figura 7.37. Se constituie apoi grupuri pentru fiecare combinație de valori (F1.DataFact, F2.DataFact). F2.DataFact are aceeași valoare, 02-08-2007, prin urmare, grupurile se constituie în funcție de F1.DataFact. Figura conține liniile primelor patru grupuri.

Z	ZI1	Z	NR_FACTURILOR1	Z	ZI2	Z	NR_FACTURILOR2
	01-08-2007		4		02-08-2007		2
	02-08-2007		2		02-08-2007		2
	03-08-2007		1		02-08-2007		2
	04-08-2007		1		02-08-2007		2
	07-08-2007		4		02-08-2007		2
	14-08-2007		3		02-08-2007		2
	15-08-2007		2		02-08-2007		2
	16-08-2007		2		02-08-2007		2
	21-08-2007		2		02-08-2007		2
	22-08-2007		1		02-08-2007		2
	01-09-2007		2		02-08-2007		2
	02-09-2007		2		02-08-2007		2
	10-09-2007		2		02-08-2007		2
	17-09-2007		1		02-08-2007		2
	07-10-2007		1		02-08-2007		2

Figura 7.38. Gruparea liniilor din figura 7.37 după cele două instanțe ale atributului DataFact

În continuare, pe liniile rezultatului din figura 7.38 se aplică gruparea:

```
SELECT F1.DataFact AS Zi1, COUNT(DISTINCT F1.NrFact) AS Nr_Facturilor1,
       F2.DataFact AS Zi2, COUNT(DISTINCT F2.NrFact) AS Nr_Facturilor2
FROM facturi F1, facturi F2
WHERE F2.DataFact = DATE'2007-08-02'
```

<sup>13</sup> În DB2 și SQL Server trebuie eliminat cuvântul DATE din interogare, ceea ce este valabil și pentru următoarele trei interogări.

**GROUP BY F1.DataFact, F2.DataFact**

**ORDER BY 1**

După cum se observă în figura 7.38 rezultatul are două jumătăți necorelate în vreun fel. Cea din stânga conține numărul de facturi pentru fiecare zi cu vânzări (pentru ca funcția COUNT să calculeze corect numărul facturilor dintr-o zi, este necesară clauza DISTINCT), iar în dreapta se repetă linia corespunzătoare datei-etalon, 2 august 2007.

Acum nu mai rămâne decât să aplicăm condiția de filtrare la nivel de grup și să extragem în rezultat doar coloanele din prima „instanță” din FACTURI:

```
SELECT F1.DataFact AS Zi1, COUNT(DISTINCT F1.NrFact) AS Nr_Facturilor1
FROM facturi F1, facturi F2
WHERE F2.DataFact = DATE'2007-08-02'
GROUP BY F1.DataFact, F2.DataFact
HAVING COUNT(DISTINCT F1.NrFact) > COUNT(DISTINCT F2.NrFact)
ORDER BY 1
```

Pentru conținutul tabeli FACTURI din acest moment, există trei zile cu mai mult de două facturi (doi fiind numărul facturilor emise pe 2 august 2007), 1, 7 și 14 august 2007 – vezi figura 7.39.

ZI1	NR_FACTURILOR1
01-08-2007	4
07-08-2007	4
14-08-2007	3

Figura 7.39. Zilele cu facturi mai multe decât 2 august 2007

*Care este restul de încasat al fiecărei facturi ?*

Așa cum reiese din figura 7.40, soluția următoare (Oracle/PostgreSQL) oferă un rezultat incomplet. Lipsesc facturile care nu au nici o tranșă de încasare. Pentru rezolvarea cazului avem nevoie de clauza HAVING, dar și de joncțiunea externă, după cum vom vedea în capitolul următor.

```
SELECT f.NrFact,
       TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)) /
       COUNT(DISTINCT i.CodInc)) AS Facturat,
       TRUNC(SUM(Transa) / MAX(lf.Linie)) AS Incasat,
       TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)) /
       COUNT(DISTINCT i.CodInc)) -
       TRUNC(SUM(Transa) / MAX(lf.Linie)) AS Rest_de_incasat
FROM clienti c
      INNER JOIN facturi f ON c.CodCl=f.CodCl
      INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
      INNER JOIN produse p ON lf.CodPr=p.CodPr
```

INNER JOIN incasfact i ON f.NrFact=i.NrFact  
GROUP BY f.NrFact

NRFACT	FACTURAT	INCASAT	REST_DE_INCASAT
1111	4346037	53996	4292041
1112	125516	125516	0
1113	106275	106275	0
1117	222050	23204	198846
1118	201975	201975	0
1120	97664	7315	90349

Figura 7.40. Tentativă eșuată de a afla restul de încasat din fiecare factură

*Care sunt facturile încasate măcar în proporție de 10% ?*

Gruparea o facem pe facturi (NrFact), iar extragerea celor care ne interesează se realizează prin clauza HAVING:

```
SELECT f.NrFact,
       TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)) /
COUNT(DISTINCT i.CodInc)) AS Facturat,
       TRUNC(SUM(Transa) / MAX(lf.Linie)) AS Incasat,
       TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)) /
COUNT(DISTINCT i.CodInc)) -
       TRUNC(SUM(Transa) / MAX(lf.Linie)) AS Rest_de_incasat
FROM clienti c
      INNER JOIN facturi f ON c.CodCl=f.CodCl
      INNER JOIN liniifact lf ON lf.NrFact=f.NrFact
      INNER JOIN produse p ON lf.CodPr=p.CodPr
      INNER JOIN incasfact i ON f.NrFact=i.NrFact
GROUP BY f.NrFact
HAVING TRUNC(SUM(Transa) / MAX(lf.Linie)) >=
       (TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)) /
COUNT(DISTINCT i.CodInc))) * 0.10
```

Sintaxa este Oracle/PostgreSQL. După se observă în figura 7.41, sunt patru facturi în această situație, dintre care trei sunt încasate în totalitate.

NRFACT	FACTURAT	INCASAT	REST_DE_INCASAT
1112	125516	125516	0
1113	106275	106275	0
1117	222050	23204	198846
1118	201975	201975	0

Figura 7.41. Facturile încasate în proporție de minimum 10%

## 7.5.Diviziunea relațională

Câteva situații ce reclamă diviziunea relațională pot fi soluționate elegant cu ajutorul clauzelor GROUP BY și HAVING. În exemplul următor (exemplul 24 din paragraful 4.4.7) este vorba de o intersecție “simulată” printr-un mecanism apropiat de diviziune.

*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 IN ('Produs 1', 'Produs 2')
GROUP BY DataFact
HAVING COUNT(DISTINCT liniifact.CodPr) = 2
ORDER BY DataFact
```

Sintaxa este operațională pentru toate cele patru servere BD. Prin joncțiunea celor trei tabele și selectarea liniilor care se referă strict la unul dintre cele două produse, obținem o corespundență zile-produse ca în stânga figurii 7.42.

**Zile și produsele 1 sau 2**

DATAFACT	DENPR
01-08-2007	Produs 1
01-08-2007	Produs 2
02-08-2007	Produs 2
03-08-2007	Produs 1
03-08-2007	Produs 2
04-08-2007	Produs 1
04-08-2007	Produs 2
07-08-2007	Produs 2
14-08-2007	Produs 1
14-08-2007	Produs 2
15-08-2007	Produs 2
16-08-2007	Produs 1
16-08-2007	Produs 2
21-08-2007	Produs 2
01-09-2007	Produs 1
01-09-2007	Produs 2
02-09-2007	Produs 2
07-09-2007	Produs 2
10-09-2007	Produs 1
10-09-2007	Produs 2
17-09-2007	Produs 1
17-09-2007	Produs 2

DATAFACT
01-08-2007
03-08-2007
04-08-2007
14-08-2007
16-08-2007
01-09-2007
10-09-2007
17-09-2007

Figura 7.42. Zilele în care s-au vândut și Produs 1 și Produs 2

Un grup se asociază unei zile de vânzări în care s-a vândut fie numai Produs 1, fie numai Produs 2, fie ambele (singure sau în combinație cu alte produse). Pentru fiecare grup se numără câte produse, dintre Produs 1 și Produs 2, au fost facturate. Funcția COUNT() din clauza HAVING poate "întoarce" maxim valoarea 2, caz în care data respectivă se încadrează în zilele căutate.

Celelalte probleme de diviziune relațională reclamă includerea în predicatul clauzei HAVING a uneia sau mai multor consultări (subconsultări). Dar despre această facilitate vom vorbi în capitolul 9 (paragraful 9.3).