Capitolul 8. NULLități, joncțiuni externe şi structuri alternative

Alăturarea celor trei subiecte în acest capitol are o doză substanțială de relativism. Totuşi, de joncțiuni externe nu putem sufla o vorbă dacă am fi ignoranți în materie de NULL, iar structurile de tip CASE, ca funcționalitate, pot fi legate destul de natural de celelalte două teme.

8.1. NULLi în tabele

Valoarea NULL a fost introdusă în capitolul 2, la explicarea noțiunilor modelului relațional, ca posibilitate de reprezentate a informațiilor… inexistente sau inaplicabile[1]. Raportul Interim 75-02-09 înaintat ANSI X3 (SPARC Study Group 1975) a delimitat 14 tipuri de date incomplete ce ar putea apărea ca rezultate ale unor operații sau valori ale atributelor, printre care: depăşiri ale capacității de stocare, diviziune la zero, trunchierea şirurilor de caractere, ridicarea lui zero la puterea zero şi alte erori computaționale, precum şi valori necunoscute.

8.1.1. Valori nule în bazele de date TRIAJ şi VÎNZĂRI

O parte dintre problemele nulității atributelor din tabelele celor două baze de date au fost discutate pe parcursul capitolelor 2 şi 3. Tipologia NULLităților în BD TRIAJ este destul de simplă. Tabela TRIAJ este singura în care apar, deocamdată, valori NULL (vezi listingurile de populare din cap. 3). Astfel, în primul caz din triaj (prima linie a tabelei TRIAJ) care a fost recepționat/investigat pe 3 ian. 2008, ora 7:18, valoarea atributului Tratament_Imediat este NULL, ceea ce înseamnă că nu i-a fost administrat, pe loc, nici un medicament sau vreo procedură. Pe a treia linie, cazul recepționat tot pe 3 ian. 2008, dar la ora 12:45, valoarea atributului Secție_Destinație este NULL, ceea ce ar putea însemna că starea pacientului nu era îngrijorătoare; după o scurtă medicație, pacientul şi-a revenit; nemaiexistând alte simptome, acesta a fost externat.

---

[1] Vezi şi [Fotache00-2]

În schimb, pentru tabelele bazei de date VÎNZĂRI, aveam trei situații majore:

- valoare necunoscută, cum ar fi cea a adresei Clientului 2 SA
- valoare inaplicabilă, cum ar valoarea unui eventual atributului Culoare în tabela PRODUSE, pentru sortimente precum Vodcă, Sare iodată, Cafea etc.
- valoare inexistentă – cum ar fi Ioan Vasile (tabela PERSOANE) care nu şi-a deschis, încă nici un cont de poştă electronică (e-mail).

De multe ori, nici nu ştim la care dintre aceste situații se referă (meta) valoarea NULL. Practic, nu ştim exact dacă Ioan Vasile este tehnofob şi nu vrea să audă de e-mail sau dacă şi-a deschis un cont între timp, dar n-am aflat noi (şi soția).

Cum pot apărea valorile nule într-o tabelă ? Simplu: fie la inserarea unei linii într-o tabelă (INSERT), fie la modificarea valorii unui atribut pe o linie dintr-o tabelă (UPDATE). Bun, ar mai fi o modalitate – prin proceduri stocate, dar nu ne interesează lucrurile astea acum (adică ne interesează, dar nu ne pricepem). Din capitolul 3 ştim că este posibil ca la inserare să se specifice numai valorile anumitor atribute:

```
INSERT INTO persoane (CNP, Nume, Prenume, Sex, CodPost, TelAcasa)
VALUES ('CNP10', 'Procopiu', 'Ivan', 'B' , '700505',  NULL) ;
```

Clauza VALUES conține numai valorile atributelor enumate după numele tabelei, adică CNP, Nume, Prenume, Sex, CodPost şi TelAcasa. Cum pentru atributele Adresa, TelBirou, TelMobil şi EMail nu am definit în capitolul 3 valori implicite (DEFAULT), în tabelă, pentru linia proaspăt inserată valorile acestora vor fi NULL (vezi figura 8.1). Dar şi clauza VALUES are un NULL, aşa încât şi atributul TelAcasă va avea aceeaşi soartă.

```
INSERT INTO persoane (CNP, Nume, Prenume, Sex, CodPost, TelAcasa)
              VALUES ('CNP10', 'Procopiu', 'Ivan', 'B' , '700505',  NULL) ;

SELECT * FROM persoane
```

Results | Messages

	CNP	Nume	Prenume	Adresa	Sex	CodPost	TelAcasa	TelBirou	TelMobil	Email
1	CNP1	Ioan	Vasile	I.L.Caragiale, 22	B	700505	123456	987654	094222222	NULL
2	CNP10	Procopiu	Ivan	NULL	B	700505	NULL	NULL	NULL	NULL
3	CNP2	Vasile	Ion	NULL	B	700505	234567	876543	094222223	Ion@a.ro
4	CNP3	Popovici	Ioana	V.Micle, Bl.I, Sc.B,Ap.2	F	701150	345678	NULL	094222224	NULL
5	CNP4	Lazar	Caraion	M.Eminescu, 42	B	706500	456789	NULL	094222225	NULL
6	CNP5	Iurea	Simion	I.Creanga, 44 bis	B	706500	567890	543210	NULL	NULL
7	CNP6	Vasc	Simona	M.Eminescu, 13	F	701150	NULL	432109	094222227	NULL
8	CNP7	Popa	Ioanid	I.Ion, Bl.H2, Sc.C, Ap.45	B	701900	789012	321098	NULL	NULL
9	CNP8	Bogacs	Ildiko	I.V.Viteazu, 67	F	705550	890123	210987	094222229	NULL
10	CNP9	Ioan	Vasilica	Garii, Bl.B4, Sc.A, Ap.1	F	701900	901234	109876	094222230	NULL

Figura 8.1. Valorile din baza de date pentru linia inserată

La modificare, lucrurile sunt relativ simple. Dacă aflăm că numărul de mobil al Ioanei Popovici (CNP3) nu este cel corect, pentru a evita deranjurile (telefonice), până la aflarea numărului corect putem să i-l „nulizăm” (un pic):

```
UPDATE persoane SET TelMobil = NULL WHERE CNP = 'CNP3'
```

În capitolul 3 am văzut că o parte dintre atribute – cele ale căror valori considerate obligatorii – pot scăpa de valorile NULL, şi, implicit, de eventuale necazuri pricinuite de acestea, cu ajutorul restricției NOT NULL declarată fie la crearea tabelei (CREATE TABLE), fie ulterior (ALTER TABLE). Dacă, în urma unei inserări/modificări, vreunui atribut declarat NOT NULL (ex. Judet din tabela JUDETE) i se încearcă a i se atribui această valoare:

```
UPDATE judete
SET Judet = NULL
WHERE Jud='IS'
```

SGBD-ul (Oracle, în cazul nostru) va reacționa cu o eroare ce blochează inserarea/modificarea respectivă – vezi figura 8.2.

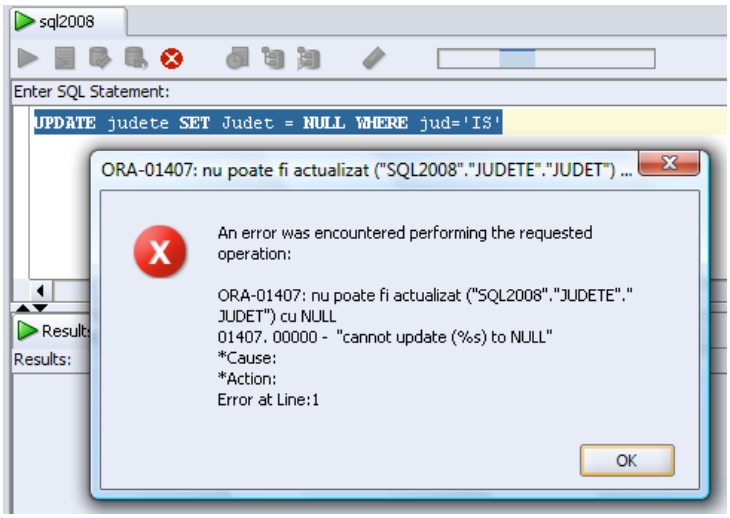
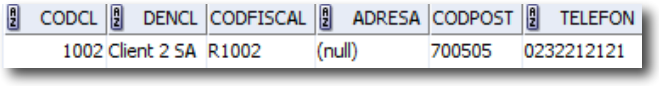


Figura 8.2. Modificare ce violează restricţia de nenulitate

În paragraful 5.5 am introdus un mic ghimpe NULList, întrucând, dacă introducem în lista operatorului NOT IN o valoare NULL, rezultatul nu va conține nicio linie. Probabil că ați rămas cu ideea că, fără negație, operatorul IN funcționează uns chiar dacă în listă strecurăm discret un NULL. Ei, bine, nu-i chiar aşa ! În tabela CLIENTI, pentru cei care au codurile 1001, 1004 şi 1006 nu cunoaştem numărul de telefon, prin urmare, pe cele trei linii, valorile atributului Telefon este NULL. Cu toate acestea, interogarea:

```
SELECT *
FROM clienti
WHERE Telefon IN ('0232212121', NULL)
```

extrage doar o linie – vezi figura 8.3 – cea a clientului 1002, deci nici urmă de vreun client cu numărul de telefon necunoscut, deşi în listă am inclus şi NULL.



CODCL	DENCL	CODFISCAL	ADRESA	CODPOST	TELEFON
1002	Client 2 SA	R1002	(null)	700505	0232212121

Figura 8.3. Valorile NULL nu pot fi extrase cu operatorul IN

Iar dacă am dori să extragem clienţii cu adresele ne-nule, în nici un caz nu folosim operatorul NOT IN, deoarece interogarea:

```
SELECT *
FROM clienti
WHERE Telefon NOT IN (NULL)
```

nu extrage nicio linie. După cum vom vedea în paragraful 8.2 în SQL putem folosi în predicatul de selecţie clauza IS NULL prin care aceste probleme sunt rezolvate destul de uşor.

Tot în paragraful 5.5 am văzut că DB2-ul nu acceptă includerea unui NULL într-o listă-argument a operatorului IN (figura 5.23)[2].

8.1.2. Două tabele suplimentare

Deşi baza de date VÎNZĂRI este alcătuită dintr-un număr considerabil de tabele şi atribute, introducem încă două tabele cu scopul de a gestiona o parte din datele privind drepturile băneşti (salariu negociat şi sporuri) ale angajaţilor firmei. Ambele tabele nu au nimic în comun cu vânzările, ci doar cu salarizarea. Prima tabelă se numeşte PERSONAL2 şi conţine date generale despre angajaţi: marcă; nume şi prenume; data naşterii; compartiment; marca şefului (direct); salariu tarifar. A doua, SPORURI, evidenţiază sporurile lunare primite de fiecare angajat: sporul de vechime (SporVechime), sporul pentru orele lucrate noaptea (SporNoapte), sporuri pentru condiţii deosebite (SporCD) şi sporuri diverse (AlteSpor).

Cu ajutorul valorii NULL se poate face diferenţa între angajaţii pentru care nu s-a calculat valoarea sporului pe luna curentă (şi care vor avea în câmpul corespunzător valoarea NULL) şi cei care nu au dreptul la un asemenea spor, adică valoarea este 0. Scriptul de creare a celor două tabele este prezentat în listingul 8.1.

Listing 8.1. Script DB2/PostgreSQL de creare a tabelelor SPORURI şi PERSONAL2[3]

---

[2] Mesajul de eroare DB2 este, în cazul lansării în execuţie a ultimelor două interogăru, *SQL0206N "NULL" is not valid in the context where it is used. SQLSTATE=42703*

```
DROP TABLE sporuri ;
DROP TABLE personal2 ;

CREATE TABLE personal2 (
        Marca NUMERIC(5) NOT NULL
                CONSTRAINT pk_personal2 PRIMARY KEY,
        NumePren VARCHAR(40),
        DataNast DATE,
        Compart VARCHAR(20),
        MarcaSef NUMERIC(5)
                CONSTRAINT fk_personal2 REFERENCES personal2(marca),
        SalTarifar NUMERIC(12,2)   ) ;

CREATE TABLE sporuri (
        An NUMERIC(4) NOT NULL,
        Luna NUMERIC(2) NOT NULL,
        Marca NUMERIC(5) NOT NULL
                CONSTRAINT fk_sporuri_pers2 REFERENCES personal2 (marca),
        SporVechime NUMERIC(12,2),
        SporNoapte NUMERIC(12,2),
        SporCD NUMERIC(12,2),
        AlteSpor NUMERIC(12,2),
        CONSTRAINT pk_sporuri PRIMARY KEY (an,luna,marca)  ) ;
```

Figurile 8.4 şi 8.5 ilustrează câteva înregistrări de test ale celor două tabele. Tabela PERSONAL2 conține 10 linii, deci, la momentul curent, firma are zece angajați. Pentru doi (angajații 4 şi 7) nu li se cunosc data naşterii. Cinci angajați (2, 4, 5, 6 şi 7) lucrează în compartimentul Financiar, trei (3, 8 şi 9) la compartimentul Marketing, iar unul (10) la Resurse umane. Există şi un angajat care lucrează în cadrul compartimentul Direcțiune. Bănuim că este vorba de directorul general, bănuială confirmată de faptul că, în linia corespunzătoare acestui angajat (prima), atributul MarcaŞef, care indică marca şefului direct al fiecărui angajat, are valoarea NULL. Cu alte cuvinte, Angajat 1 nu are şef.

---

[3] În Oracle, în loc de VARCHAR se foloseşte VARCHAR2, iar în SQL Server, în loc de DATE se foloseşte SMALLDATETIME.

MARCA	NUMEPREN	DATANAST	COMPART	MARCASEF	SALTARIFAR
1	ANGAJAT 1	01-07-1962	DIRECTIUNE	(null)	1600
2	ANGAJAT 2	11-10-1977	FINANCIAR	1	1450
3	ANGAJAT 3	02-08-1962	MARKETING	1	1450
4	ANGAJAT 4	(null)	FINANCIAR	2	1380
5	ANGAJAT 5	30-04-1965	FINANCIAR	2	1420
6	ANGAJAT 6	09-11-1965	FINANCIAR	5	1350
7	ANGAJAT 7	(null)	FINANCIAR	5	1280
8	ANGAJAT 8	31-12-1960	MARKETING	3	1290
9	ANGAJAT 9	28-02-1976	MARKETING	3	1410
10	ANGAJAT 10	29-01-1972	RESURSE UMANE	1	1370

Figura 8.4. Câteva înregistrări în tabela PERSONAL2

Dacă privim cu atenție şi figura 8.5, mai putem bănui că firma s-a înființat în aprilie 2007, când avea numai trei angajați. Lucrul acesta ne este sugerat de existența, pentru această lună, a numai trei angajați pentru care au fost introduse sporuri. La momentul curent (primul trimestru al anului 2008) sunt 10 angajați.

AN	LUNA	MARCA	SPORVECHIME	SPORNOAPTE	SPORCD	ALTESPOR
2007	4	1	160	0	0	132
2007	4	2	130	45	0	70
2007	4	3	145	156	420	157
2007	5	1	160	0	0	0
2007	5	2	80	45	0	70
2007	5	3	145	0	0	0
2007	5	10	137	0	0	430
2007	6	1	160	0	0	0
2007	6	2	80	0	0	150
2007	6	4	50	15	88	120
2007	6	5	130	15	0	20
2007	6	10	200	12	0	6
2007	7	1	160	0	(null)	(null)
2007	7	2	80	0	0	158
2007	7	3	145	0	0	0
2007	7	4	50	15	(null)	15
2007	7	5	130	0	0	120
2007	7	6	110	147	0	0
2007	7	7	60	210	0	0
2007	7	8	130	0	15	0
2007	7	9	140	100	77	0
2007	7	10	200	0	0	120

Figura 8.5. Câteva înregistrări în tabela SPORURI

Popularea celor trei tabele este conținută în listingul 8.2.

Listing 8.2. Script Oracle/PostgreSQL de populare a tabelelor SPORURI şi PERSONAL2[4]

```
DELETE FROM personal2 ;
INSERT INTO personal2 VALUES (1, 'ANGAJAT 1', DATE'1962-07-01',
        'DIRECTIUNE', NULL, 1600) ;
INSERT INTO personal2 VALUES (2, 'ANGAJAT 2', DATE'1977-10-11',
         'FINANCIAR', 1, 1450) ;
INSERT INTO personal2 VALUES (3, 'ANGAJAT 3', DATE'1962-08-02',
        'MARKETING', 1, 1450) ;
INSERT INTO personal2 VALUES (4, 'ANGAJAT 4', NULL,
        'FINANCIAR', 2, 1380) ;
INSERT INTO personal2 VALUES (5, 'ANGAJAT 5', DATE'1965-04-30',
        'FINANCIAR', 2, 1420) ;
INSERT INTO personal2 VALUES (6, 'ANGAJAT 6', DATE'1965-11-09',
        'FINANCIAR', 5, 1350) ;
INSERT INTO personal2 VALUES (7, 'ANGAJAT 7', NULL, 'FINANCIAR', 5, 1280) ;
INSERT INTO personal2 VALUES (8, 'ANGAJAT 8', DATE'1960-12-31',
        'MARKETING', 3, 1290) ;
INSERT INTO personal2 VALUES (9, 'ANGAJAT 9', DATE'1976-02-28',
        'MARKETING', 3, 1410) ;
INSERT INTO personal2 VALUES (10, 'ANGAJAT 10', DATE'1972-01-29',
        'RESURSE UMANE', 1, 1370) ;

DELETE FROM sporuri ;
INSERT INTO sporuri VALUES (2007, 4, 1, 160, 0, 0, 132) ;
INSERT INTO sporuri VALUES (2007, 4, 2, 130, 45, 0, 70) ;
INSERT INTO sporuri VALUES (2007, 4, 3, 145, 156, 420, 157) ;
INSERT INTO sporuri VALUES (2007, 5, 1, 160, 0, 0, 0) ;
INSERT INTO sporuri VALUES (2007, 5, 2, 80, 45, 0, 70) ;
INSERT INTO sporuri VALUES (2007, 5, 3, 145, 0, 0, 0) ;
INSERT INTO sporuri VALUES (2007, 5, 10, 137, 0, 0, 430) ;
INSERT INTO sporuri VALUES (2007, 6, 1, 160, 0, 0, 0) ;
INSERT INTO sporuri VALUES (2007, 6, 2,  80, 0, 0, 150) ;
INSERT INTO sporuri VALUES (2007, 6, 4, 50, 15, 88, 120) ;
```

[4] Versiunea DB2/SQL Server a scriptului se obține prin eliminarea cuvintelor DATE.

```
INSERT INTO sporuri VALUES (2007, 6, 5, 130, 15, 0, 20) ;
INSERT INTO sporuri VALUES (2007, 6, 10, 200, 12, 0, 6) ;
INSERT INTO sporuri VALUES (2007, 7, 1, 160, 0, NULL, NULL) ;
INSERT INTO sporuri VALUES (2007, 7, 2, 80, 0, 0, 158) ;
INSERT INTO sporuri VALUES (2007, 7, 3, 145, 0, 0, 0) ;
INSERT INTO sporuri VALUES (2007, 7, 4, 50, 15, NULL, 15) ;
INSERT INTO sporuri VALUES (2007, 7, 5, 130, 0, 0, 120) ;
INSERT INTO sporuri VALUES (2007, 7, 6, 110, 147, 0, 0) ;
INSERT INTO sporuri VALUES (2007, 7, 7, 60, 210, 0, 0) ;
INSERT INTO sporuri VALUES (2007, 7, 8, 130, 0, 15, 0) ;
INSERT INTO sporuri VALUES (2007, 7, 9, 140, 100, 77, 0) ;
INSERT INTO sporuri VALUES (2007, 7, 10, 200, 0, 0, 120) ;
```

8.2. Operatorul IS NULL

Valoarea NULL nu se confundă cu valoarea 0 - pentru atributele numerice - sau cu valoarea '' – spațiu - pentru atributele de tip şir de caractere. Este important de notat că, în vederea identificării valorilor nule, operatorul are forma *IS NULL* şi nu *=NULL*.

Pentru care dintre clienți nu se cunoaşte adresa ?

Soluția se bazează pe utilizarea operatorului IS NULL care extrage toate valorile NULL pentru un atribut (figura 8.6):

```
SELECT *
FROM clienti
WHERE Adresa IS NULL
```

CODCL	DENCL	CODFISCAL	ADRESA	CODPOST	TELEFON
1002	Client 2 SA	R1002	(null)	700505	0232212121
1005	Client 5 SRL	R1005	(null)	701900	0256111111

Figura 8.6. Clienţii „fără adresă”

Prin execuția frazei SQL:

```
SELECT *
FROM clienti
```

```
WHERE Adresa = NULL
```

se va obține o tabelă cu 0 (zero) linii sau chiar o eroare (în DB2[5]). Rezultatul evaluării **Adresa = NULL** va fi întotdeauna FALSE.

În unele SGBD-uri, precum Oracle, există o sursă de confuzie între şirul de caractere de lungime zero ('', adică două apostrofuri "lipite" (fără spațiu)) şi NULL-ul propriu zis. Astfel, dacă vrem să NULL-izăm adresa clientului cu codul 1007 putem recurge la varianta următoare a comenzii UPDATE:

```
UPDATE clienti
SET Adresa = ''
WHERE CodCl=1007
```

Atribuirea s-a făcut, deci, cu **Adresa = ''**, şi nu cu **Adresa = NULL**. Rezultatul este acelaşi, dacă examinăm linia acestui client după comanda UPDATE – vezi figura 8.7.

CODCL	DENCL	CODFISCAL	ADRESA	CODPOST	TELEFON
1007	Client 7 SRL	R1007	(null)	701900	0256121212

Figura 8.7. NULLitatea adresei clientului 1007 (în Oracle)

Prin urmare, dacă şirul vid '' (de lungime zero) este echivalentul valorii NULL, la atribuire, nu la fel stau lucrurile în clauza WHERE, întrucât interogarea Oracle:

```
SELECT *
FROM CLIENTI
WHERE Adresa = ''
```

„continuă" să nu extragă nici o linie. Lucrurile nu stau la fel în DB2 şi în MS SQL Server, întrucât interogarea extrage linia corespunzătoarea clientului 1007, valoarea adresei pentru acest client nefiind considerată nulă. În concluzie, dacă pentru DB2, PostgreSQL şi SQL Server, şirul de caractere de lungime zero - '' (două apostrofuri, nu ghilimele!) - este diferit de NULL, pentru Oracle uneori este diferit (în clauza WHERE), alteori acelaşi lucru (în clauza SET a comenzii UPDATE).

Care sunt persoanele şi lunile pentru care nu s-a calculat (nu se cunoaşte) sporul pentru condiții deosebite ?

Prin interogarea:

```
SELECT sporuri.Marca, NumePren, Compart, An, Luna
```

---

[5] Mesajul de eroare din DB2 la execuția comenzii este: *SQL0401N The data types of the operands for the operation "=" are not compatible. SQLSTATE=42818*

```
FROM personal2
INNER JOIN sporuri ON personal2.Marca=sporuri.Marca
WHERE SporCD IS NULL
```

se obține situația din figura 8.8.

MARCA	NUMEPREN	COMP...	AN	LUNA
1	ANGAJAT 1	DIRECTIUNE	2007	7
4	ANGAJAT 4	FINANCIAR	2007	7

Figura 8.8. Angajaţii pentru care nu s-au operat sporurile pentru condiţii deosebite

Care sunt angajații şi lunile în care aceştia nu au primit spor pentru condiții deosebite ?

Atât soluția cât şi rezultatul sunt sensibil diferite – vezi figura 8.9, întrucât acum ne interesează valorile zero (nu NULL) ale SporCD:

```
SELECT sporuri.Marca, NumePren, Compart, An, Luna
FROM personal2 INNER JOIN sporuri
        ON personal2.Marca=sporuri.Marca
WHERE SporCD = 0
ORDER BY An, Luna, NumePren
```

MARCA	NUMEPREN	COMPART	AN	LUNA
1	ANGAJAT 1	DIRECTIUNE	2007	4
2	ANGAJAT 2	FINANCIAR	2007	4
1	ANGAJAT 1	DIRECTIUNE	2007	5
10	ANGAJAT 10	RESURSE UMANE	2007	5
2	ANGAJAT 2	FINANCIAR	2007	5
3	ANGAJAT 3	MARKETING	2007	5
1	ANGAJAT 1	DIRECTIUNE	2007	6
10	ANGAJAT 10	RESURSE UMANE	2007	6
2	ANGAJAT 2	FINANCIAR	2007	6
5	ANGAJAT 5	FINANCIAR	2007	6
10	ANGAJAT 10	RESURSE UMANE	2007	7
2	ANGAJAT 2	FINANCIAR	2007	7
3	ANGAJAT 3	MARKETING	2007	7
5	ANGAJAT 5	FINANCIAR	2007	7
6	ANGAJAT 6	FINANCIAR	2007	7
7	ANGAJAT 7	FINANCIAR	2007	7

Figura 8.9. Angajaţii şi lunile pentru care SporCD este zero (neNULL)

Care dintre angajați sunt născuți înainte de 1 ianuarie 1970 şi care după această dată?

Persoanele născute *înainte* (figura 8.10) se obțin prin:

```
SELECT *
FROM personal2
WHERE DataNast < DATE'1970-01-01'
```

MARCA	NUMEPREN	DATANAST	COMPART	MARCASEF	SALTARIFAR
1	ANGAJAT 1	01-07-1962	DIRECTIUNE	(null)	1600
3	ANGAJAT 3	02-08-1962	MARKETING	1	1450
5	ANGAJAT 5	30-04-1965	FINANCIAR	2	1420
6	ANGAJAT 6	09-11-1965	FINANCIAR	5	1350
8	ANGAJAT 8	31-12-1960	MARKETING	3	1290

Figura 8.10. Persoane născute înainte de 1 ian. 1970

iar cele născute *după* 1 ianuarie 1970 (figura 8.11) prin:

```
SELECT *
FROM personal2
WHERE DataNast >= DATE'1970-01-01'
```

MARCA	NUMEPREN	DATANAST	COMPART	MARCASEF	SALTARIFAR
2	ANGAJAT 2	11-10-1977	FINANCIAR	1	1450
9	ANGAJAT 9	28-02-1976	MARKETING	3	1410
10	ANGAJAT 10	29-01-1972	RESURSE UMANE	1	1370

Figura 8.11. Persoane născute după 1 ian. 1970

Dacă reunim mulțimea angajaților născuți înainte de data fixată cu mulțimea celor născuți după această dată nu obținem relația inițială PERSONAL2 – vezi figura 8.12 !

```
SELECT * FROM personal2
WHERE DataNast < DATE'1970-01-01'
        UNION
SELECT * FROM personal2
WHERE DataNast >= DATE'1970-01-01'
```

MARCA	NUMEPREN	DATANAST	COMPART	MARCASEF	SALTARIFAR
1	ANGAJAT 1	01-07-1962	DIRECTIUNE	(null)	1600
2	ANGAJAT 2	11-10-1977	FINANCIAR	1	1450
3	ANGAJAT 3	02-08-1962	MARKETING	1	1450
5	ANGAJAT 5	30-04-1965	FINANCIAR	2	1420
6	ANGAJAT 6	09-11-1965	FINANCIAR	5	1350
8	ANGAJAT 8	31-12-1960	MARKETING	3	1290
9	ANGAJAT 9	28-02-1976	MARKETING	3	1410
10	ANGAJAT 10	29-01-1972	RESURSE UMANE	1	1370

Figura 8.12. Reuniunea persoanelor născute înainte de 1 ian. 1970 cu persoanele născute după această dată

Din tabela obținută în figura 8.12 lipsesc angajații care nu au precizată data naşterii, altfel spus, persoanele "fără vârstă". Pentru recompunerea tabelei PERSO-

NAL2, în reuniune mai trebuie adăugate şi liniile pentru care *DataNast IS NULL*, adică:

```
SELECT * FROM personal2 WHERE DataNast < DATE'1970-01-01'
        UNION
SELECT * FROM personal2 WHERE DataNast >= DATE'1970-01-01'
        UNION
SELECT * FROM personal2 WHERE DataNast IS NULL
ORDER BY 1
```

Revenind la exemplele din finalul paragrafului 5.5, interogarea *SELECT * FROM facturi WHERE DataFact IN (DATE'2007-08-01', DATE'2007-08-03', DATE'2007-08-07', NULL)* este echivalentă cu *SELECT * FROM facturi WHERE DataFact IN (DATE'2007-08-01', DATE'2007-08-03', DATE'2007-08-07')* în Oracle, PostgreSQL şi SQL Server (nu uitați să eliminați DATE-urile în SQL Server) deoarece logica execuției sale este, de fapt:

```
SELECT *
FROM facturi
WHERE DataFact = DATE'2007-08-01' OR DataFact = DATE'2007-08-03'
        OR DataFact = DATE'2007-08-07' OR DataFact =NULL
```

iar rezultatul unui *sau logic* (OR) dintre o valoare TRUE şi NULL este TRUE – vezi tabelul 8.1.

Tabel 8.1. Combinarea a două expresii utilizând operatorul logic OR

OR	*TRUE*	*FALSE*	*UNKNOWN*
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

În schimb, interogarea *SELECT * FROM facturi WHERE DataFact NOT IN (DATE'2007-08-01', DATE'2007-08-03', DATE'2007-08-07', NULL)* nu returnează nici o linie, întrucât este echivalentă cu:

```
SELECT *
FROM facturi
WHERE DataFact <> DATE'2007-08-01'  AND DataFact <> DATE'2007-08-03'
        AND DataFact <> DATE'2007-08-07' AND DataFact <> NULL
```

iar rezultatul unui *şi logic* (AND) dintre o valoare TRUE şi NULL este NULL – vezi tabelul 8.2. Discuția nu este aplicabilă în DB2, care nu tolerează NULL-uri în liste (ştim asta din capitolul 5).

Tabel 8.2. Combinarea a două expresii utilizând operatorul logic AND

AND	*TRUE*	*FALSE*	*UNKNOWN*
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE

UNKNOWN	UNKNOWN	FALSE	UNKNOWN

Pentru negația logică (NOT) evaluarea se face ca în tabelul 8.3. Probleme similare apar şi la interogări care folosesc subconsultări, subconsultări ce prezintă valori nule (vezi capitolul 9).

Tabel 8.3. Rezultatele utilizării operatorului logic NOT

NOT	***TRUE***	***FALSE***	***UNKNOWN***
	FALSE	TRUE	UNKNOWN

Încheiem paragraful abordând comportamentul SQL-ului în situațiile în care "subiectul predicatului" IS NULL nu este un singur atribut, ca până acum, ci două sau mai multe atribute. Problema a fost atacată de Melton şi Simon[6]. În SQL, interogarea:

```
SELECT *
FROM sporuri
WHERE (SporCD, AlteSpor) IS NULL
```

este echivalentă cu:

```
SELECT *
FROM sporuri
WHERE SporCD IS NULL AND AlteSpor IS NULL
```

rezultatul fiind cel din figura 8.13. Prin urmare, valorile ambelor atribute trebuie să fie, simultan, NULL.

AN	LUNA	MARCA	SPORVECHIME	SPORNOAPTE	SPORCD	ALTESPOR
2007	7	1	160	0	(null)	(null)

Figura 8.13. Predicat IS NULL aplicat asupra a două atribute

Genul de predicat în care argumentul operatorului IS NULL este un tuplu (alcătuit din două sau mai multe atribute) „ad-hoc" nu funcționează nici în Oracle, nici în MS SQL Server, nici în DB2, însă, suprinzător (pentru un server BD *open-source*) merge în PostgreSQL! De aceea, pentru cele trei servere BD ultima interogare reprezintă o soluție viabilă.

---

[6] [Melton & Simon 2002], pp.188-189

Analog discuție de mai sus,

```
SELECT *
FROM sporuri
WHERE (SporCD, AlteSpor) IS NOT NULL
```

este echivalent cu:

```
SELECT *
FROM sporuri
WHERE SporCD IS NOT NULL AND AlteSpor IS NOT NULL
```

(vezi figura 8.14) pentru că încearcă să extragă toate liniile în care valorile ambelor atribute sunt simultam nenule, şi, tot la fel de analog, prima variantă nu funcționează în DB2, Oracle şi SQL Server.

	an numeric(4,0)	luna numeric(2,0)	marca numeric(5,0)	sporvechime numeric(12,2	spornoapte numeric(12,2	sporcd numeric(12,2	altespor numeric(12,2
1	2007	4	1	160.00	0.00	0.00	132.00
2	2007	4	2	130.00	45.00	0.00	70.00
3	2007	4	3	145.00	156.00	420.00	157.00
4	2007	5	1	160.00	0.00	0.00	0.00
5	2007	5	2	80.00	45.00	0.00	70.00
6	2007	5	3	145.00	0.00	0.00	0.00
7	2007	5	10	137.00	0.00	0.00	430.00
8	2007	6	1	160.00	0.00	0.00	0.00
9	2007	6	2	80.00	0.00	0.00	150.00
10	2007	6	4	50.00	15.00	88.00	120.00
11	2007	6	5	130.00	15.00	0.00	20.00
12	2007	6	10	200.00	12.00	0.00	6.00
13	2007	7	2	80.00	0.00	0.00	158.00
14	2007	7	3	145.00	0.00	0.00	0.00
15	2007	7	5	130.00	0.00	0.00	120.00
16	2007	7	6	110.00	147.00	0.00	0.00
17	2007	7	7	60.00	210.00	0.00	0.00
18	2007	7	8	130.00	0.00	15.00	0.00
19	2007	7	9	140.00	100.00	77.00	0.00
20	2007	7	10	200.00	0.00	0.00	120.00

Figura 8.14. Predicatul *(SporCD, AlteSpor) IS NOT NULL*

Lucrurile se complică un pic în PostgreSQL (în celelalte trei SGBD-uri luăm, momentan, o pauză) dacă negația este plasată înaintea tuplului "ad-hoc". Să începem cu interogarea:

```
SELECT *
FROM sporuri
WHERE NOT (SporCD, AlteSpor) IS NULL
```

Rezultatul din figura 8.15 conține 21 de linii. Față de figura 8.14, linia în plus (a 15-a) este cea în care valoarea atributului SporCD este nulă iar cea a atributului AlteSpor nenulă. Motivul este că interogarea de mai sus este echivalentă cu:

```
SELECT *
FROM sporuri
```

```
WHERE (NOT SporCD IS NULL) OR (NOT AlteSpor IS NULL)
```

care, mai departe, înseamnă:

```
SELECT *
FROM sporuri
WHERE (SporCD IS NOT NULL) OR (AlteSpor IS NOT NULL)
```

(parantezele sunt pur explicative).

	an numeric(4,0)	luna numeric(2,0)	marca numeric(5,0)	sporvechime numeric(12,2	spornoapte numeric(12,2	sporcd numeric(12,2	altespor numeric(12,2
1	2007	4	1	160.00	0.00	0.00	132.00
2	2007	4	2	130.00	45.00	0.00	70.00
3	2007	4	3	145.00	156.00	420.00	157.00
4	2007	5	1	160.00	0.00	0.00	0.00
5	2007	5	2	80.00	45.00	0.00	70.00
6	2007	5	3	145.00	0.00	0.00	0.00
7	2007	5	10	137.00	0.00	0.00	430.00
8	2007	6	1	160.00	0.00	0.00	0.00
9	2007	6	2	80.00	0.00	0.00	150.00
10	2007	6	4	50.00	15.00	88.00	120.00
11	2007	6	5	130.00	15.00	0.00	20.00
12	2007	6	10	200.00	12.00	0.00	6.00
13	2007	7	2	80.00	0.00	0.00	158.00
14	2007	7	3	145.00	0.00	0.00	0.00
15	2007	7	4	50.00	15.00		15.00
16	2007	7	5	130.00	0.00	0.00	120.00
17	2007	7	6	110.00	147.00	0.00	0.00
18	2007	7	7	60.00	210.00	0.00	0.00
19	2007	7	8	130.00	0.00	15.00	0.00
20	2007	7	9	140.00	100.00	77.00	0.00
21	2007	7	10	200.00	0.00	0.00	120.00

Figura 8.15. Predicatul **NOT (SporCD, AlteSpor) IS NULL**

Pentru ca ameţeala să fie completă, mai trebuie să adaugăm că:

```
SELECT *
FROM sporuri
WHERE NOT (SporCD, AlteSpor) IS NOT NULL
```

(figura 8.16) este echivalentă:

```
SELECT *
FROM sporuri
WHERE (NOT SporCD IS NOT NULL) OR
(NOT AlteSpor IS NOT NULL)
```

care, la rândul său, este echivalentă cu mult mai simpla versiune:

```
SELECT *  FROM sporuri
WHERE (SporCD IS NULL) OR (AlteSpor IS NULL)
```

	an numeric(4,0)	luna numeric(2,0)	marca numeric(5,0)	sporvechime numeric(12,2	spornoapte numeric(12,2	sporcd numeric(12,2	altespor numeric(12,2
1	2007	7	1	160.00	0.00		
2	2007	7	4	50.00	15.00		15.00

Figura 8.16. Predicatul *NOT (SporCD, AlteSpor) IS NOT NULL*

Negaţia la nivel de tuplu este, de cele mai multe ori, un simplu exerciţiu de preţiozitate, mai ales când discutăm despre negarea negaţiei (nici o legătură cu Engels!, deşi recenta criză mondială a adus o nesperată culoare în obraji (neo) marxiştilor).

8.3. Transformarea NULLilor

Am văzut că un prim aspect iritant al NULL-ilor în SQL se manifestă la includerea acestei valori într-o listă precedată de operatorul NOT IN. Probabil, însă că iritarea o să crească dacă luăm câteva exemple ce conţin expresii în care se strecoară câte un NULL.

Care este totalul sporurilor fiecărui angajat pe luna iulie 2007 ?

```
SELECT sporuri.Marca, NumePren, Compart,
       SporVechime + SporNoapte + SporCD + AlteSpor
       AS TotalSporuri
FROM personal2 INNER JOIN sporuri
       ON personal2.Marca=sporuri.Marca
WHERE An = 2007 AND Luna = 7
```

Din păcate, rezultatul este incorect – vezi figura 8.17 – deoarece, din prezentarea conţinutului tabelei SPORURI (figura 8.5), reiese că, pe luna iulie 2007, ANGAJAT 1 are calculat spor de vechime, iar ANGAJAT 4 are, pe aceaşi lună şi spor de vechime, şi de noapte şi alte sporuri, iar aceste sporuri nu au fost luate în calcul la însumare.

MARCA	NUMEPREN	COMPART	TOTALSPORURI
1	ANGAJAT 1	DIRECTIUNE	(null)
2	ANGAJAT 2	FINANCIAR	238
3	ANGAJAT 3	MARKETING	145
4	ANGAJAT 4	FINANCIAR	(null)
5	ANGAJAT 5	FINANCIAR	250
6	ANGAJAT 6	FINANCIAR	257
7	ANGAJAT 7	FINANCIAR	270
8	ANGAJAT 8	MARKETING	145
9	ANGAJAT 9	MARKETING	317
10	ANGAJAT 10	RESURSE UMANE	320

Figura 8.17. Rezultatul unei expresii în care cel puţin un operand este NULL

Explicaţia este simplă: dacă, într-o expresie, unul dintre operanzi este NULL, atunci rezultatul evaluării întregii expresii este NULL. Fac excepţie funcţiile agregat (paragraful 6.7). Dacă, spre exemplu, vrem să calculăm *totalul sporurilor de noapte acordate pentru luna iulie 2007*, fraza:

```
SELECT SUM(SporNoapte) AS Total_SporuriNoapte_Luna_7
```

```
FROM sporuri
WHERE An = 2007 AND Luna=7
```

calculează corect rezultatul (figura 8.18).

TOTAL_SPORURINOAPTE_LUNA_7
472

Figura 8.18. Funcţia SUM (SporNoapte) nu este afectată de eventualele valori NULL

Revenim la problema din figura 8.17. Pentru a asigura corectitudinea totalului, ar trebui ca, în expresie, orice valoare nulă să fie considerată zero. Lucru realizabil, deoarece încă din SQL92 a apărut o funcție în acest sens – COALESCE:

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

Sumele obținute sunt în acest caz cele din figura 8.19.

MARCA	NUMEPREN	COMPART	TOTALSPORURI
1	ANGAJAT 1	DIRECTIUNE	160
2	ANGAJAT 2	FINANCIAR	238
3	ANGAJAT 3	MARKETING	145
4	ANGAJAT 4	FINANCIAR	80
5	ANGAJAT 5	FINANCIAR	250
6	ANGAJAT 6	FINANCIAR	257
7	ANGAJAT 7	FINANCIAR	270
8	ANGAJAT 8	MARKETING	145
9	ANGAJAT 9	MARKETING	317
10	ANGAJAT 10	RESURSE UMANE	320

Figura 8.19. Conversia valorilor nule în zero şi evaluarea corectă a expresiei

Este important de reținut că funcția nu se aplică la nivel de expresie, ci fiecărui operand susceptibil de nulitate. Completăm SELECT-ul anterior cu o coloană în care funcția se aplică la nivelul întregii expresii:

```
SELECT sporuri.Marca, NumePren, Compart,
        COALESCE(SporVechime,0) + COALESCE (SporNoapte,0) +
        COALESCE (SporCD,0) + COALESCE (AlteSpor,0)
                AS TotalSporuri_Corect,
        COALESCE(SporVechime + SporNoapte + SporCD +
                AlteSpor,0) AS TotalSporuri_Incorect
FROM personal2 INNER JOIN sporuri
        ON personal2.Marca=sporuri.Marca
```

WHERE An = 2007 AND Luna = 7

Rezultatul din figura 8.20 arată destul de limpede diferența pentru angajații 1 şi 4, ei fiind cei care prezintă sporuri cu valori NULL pentru iulie 2007.

Am aplicat funcția COALESCE după nevoile exemplului. Formatul său general este însă ceva mai generos. Să presupunem că pentru luna iulie 2007, în urma unor discuții aprinse, s-a hotărât ca, la persoanele cărora nu li s-a calculat sporul pentru condiții deosebite (SporCD), valoarea acestuia să fie egală cu alte sporuri (AlteSpor); dacă nici alte sporuri nu s-au calculat, atunci valoarea luată în considerare să fie zero.

MARCA	NUMEPREN	COMPART	TOTALSPORURI_CORECT	TOTALSPORURI_INCORECT
1	ANGAJAT 1	DIRECTIUNE	160	0
2	ANGAJAT 2	FINANCIAR	238	238
3	ANGAJAT 3	MARKETING	145	145
4	ANGAJAT 4	FINANCIAR	80	0
5	ANGAJAT 5	FINANCIAR	250	250
6	ANGAJAT 6	FINANCIAR	257	257
7	ANGAJAT 7	FINANCIAR	270	270
8	ANGAJAT 8	MARKETING	145	145
9	ANGAJAT 9	MARKETING	317	317
10	ANGAJAT 10	RESURSE UMANE	320	320

Figura 8.20. Funcţia COALESCE aplicată la nivel de operand şi la nivelul întregii expresii

Figura 8.21 limpezeşte un pic lucrurile. Cu excepția rândurilor corespunzătoare angajaților cu mărcile 1 şi 4, valorile din coloana a şasea (COALESCE....) sunt egale cu valori SporCD. Pentru angajatul cu marca 4, întrucât SporCD este nul, valoarea de pe coloana a şasea va prelua valoarea atributului AlteSpor (15). La angajatul cu marca 1 şi SporCd şi AlteSpor sunt nule, aşa încât pe a şasea coloană apare valoarea zero.

MARCA	SpVech	SpNoapte	SPORCD	ALTESPOR	COALESCE(SPORCD,ALTESPOR,0)	Total_Nou
1	160	0	(null)	(null)	0	160
2	80	0	0	158	0	238
3	145	0	0	0	0	145
4	50	15	(null)	15	15	95
5	130	0	0	120	0	250
6	110	147	0	0	0	257
7	60	210	0	0	0	270
8	130	0	15	0	15	145
9	140	100	77	0	77	317
10	200	0	0	120	0	320

Figura 8.21. Când valoarea SporCD e NULL, se preia valoarea AlteSpor; când şi aceasta e NULL, se consideră zero

Acest rezultat se obține tot cu funcția COALESCE, dar cu sintaxa următoare:

```
SELECT Marca,
        SporVechime AS "SpVech",
        SporNoapte AS "SpNoapte",
        SporCD,
        AlteSpor,
        COALESCE (SporCD, AlteSpor, 0),
        COALESCE(SporVechime,0) + COALESCE (SporNoapte,0) +
        COALESCE (SporCD, AlteSpor, 0) +
        COALESCE (AlteSpor,0)
                        AS "Total_Nou"
FROM sporuri
WHERE An = 2007 AND Luna = 7
```

Prin urmare, COALESCE poate avea două sau mai multe argumente, returnând prima valoare nenulă întâlnită (parcurgerea are loc în ordinea enumerării). Toate interogările din acest paragraf sunt corecte în toate cele patru dialecte.

După cum valoarea NULL este una universal aplicabilă (pot avea valori NULL şi atribute numerice, şi atribute de şir, dată calendaristică etc.), tot aşa funcţia COALESCE poate fi aplicată asupra oricărui de argument. Revenim la un exemplu din paragraful 5.4.

Să se afişeze facturile din luna septembrie 2007 în ordinea alfabetică a observaţiilor (valorile atributul *Obs*):

În Oracle rezolvam problema ordonării cu ajutorul opţiunilor NULLS FIRST sau NULLS LAST. Spuneam atunci că PostgreSQL-ul (până la versiunea 8.3), MS SQL Server şi DB2 fac nazuri, neavând implementate aceste opţiuni. În acest moment suntem în măsură să formulăm soluţii bazate pe COALESCE care transformă valorile NULL în spaţii, spaţii ce apar la ordonare înaintea literelor (mari sau mici):

```
SELECT *
FROM facturi
WHERE datafact >= DATE'2007-09-01' AND datafact <= DATE'2007-09-30'
ORDER BY COALESCE (obs,'')  -- NULLS FIRST
```

Aceeaşi interogare, din care scoatem cele două cuvinte DATE, rezolvă problema şi în DB2 şi în MS SQL Server. Dacă am fi intenţionat ca liniile cu valori NULL ale atributului Obs să apară la finalul rezultatului (NULLS LAST) conversia trebuia făcută într-un şir de caractere (*zzz*) care, garantat, apare la sfârşit:

```
SELECT * FROM facturi
WHERE datafact >= DATE'2007-09-01' AND datafact <= DATE'2007-09-30'
ORDER BY COALESCE (obs, 'zzz') -- NULLS LAST
```

Ne interesează, în continuare, să analizăm observaţiile legate de facturi, în sensul numărării de câte ori apare o anumită observaţie. Tabela care ne interesează este FACTURI, iar gruparea se realizează după valorile atributului Obs:

```
SELECT Obs, COUNT(*)
FROM facturi
GROUP BY Obs
ORDER BY Obs NULLS LAST
```

Rezultatul (Oracle) se prezintă ca în partea stângă a figurii 8.22. Pentru grupul facturilor fără observaţii, vrem ca în loc de NULL să apară, tot la final, indicaţia * *fără observaţii** (vezi partea dreaptă a figurii 8.22), se foloseşte funcţia COALESCE de maniera următoare:

OBS	COUNT(*)
Pretul propus initial a fost modificat	3
Probleme cu transportul	3
(null)	24

COALESCE(OBS,'*FARAOBSERVATII*')	COUNT(*)
Pretul propus initial a fost modificat	3
Probleme cu transportul	3
* fara observatii *	24

Figura 8.22. Folosirea COALESCE pentru un argument şir de caractere

```
SELECT COALESCE(Obs,'* fara observatii *'), COUNT(*)
FROM facturi
GROUP BY Obs
ORDER BY Obs NULLS LAST
```

sau, pentru a funcţiona şi PostgreSQL pre8.3, DB2 şi SQL Server:

```
SELECT COALESCE(Obs,'* fara observatii *'), COUNT(*)
FROM facturi
GROUP BY Obs
ORDER BY COALESCE(Obs,'zzz fara observatii *')
```

Dacă am dori să extragem din tabela PERSONAL2 persoanele născute înainte de 1 ianuarie 1970 şi cele pentru care nu cunoaştem data naşterii, ordonate după data naşterii, până în acest paragraf puteam formula doar o soluţie bazată pe UNION:

```
SELECT *
FROM personal2
WHERE DataNast < DATE'1970-01-01'
        UNION
SELECT *
FROM personal2
WHERE DataNast IS NULL
ORDER BY 3 NULLS FIRST
```

Acum însă putem renunţa la UNION beneficiind de COALESCE, interogarea funcţionând şi în PostgreSQL (şi SQL Server, dacă eliminăm DATE):

```
SELECT *
FROM personal2
WHERE COALESCE (DataNast,DATE'1950-01-01' ) < DATE'1970-01-01'
```

```
ORDER BY COALESCE (DataNast, DATE'1950-01-01')
```

În unele dialecte SQL funcționalitatea funcției COALESCE este asigurată de VALUE (ex. DB2) şi NVL (Oracle, Visual FoxPro şi, începând cu versiunea 9.5, şi în DB2). Funcția NVL din Oracle are numai două argumente, valoarea returnată fiind a primului argument, dacă acesta este nenul, sau a celui de-al doilea (când valoarea primului este NULL). Atunci când numărul argumentelor este trei, se poate folosi o funcție soră (vitregă) – NVL2 care funcționează pe acelaşi principiu. NVL şi COALESCE sunt sinonime în DB2, atât ca funcționalitate, cât şi ca sintaxă.

Un alt element interesant legat de valorile nule ține de conversia în sens invers, dintr-o valoare oarecare, în NULL.

Să se determine totalul sporurilor de vechime pentru luna iulie 2007, dar, în rezultat, să nu fie luate în calcul valorile (valoarea, dacă apare o singură dată) 130 Lei.

Soluția "clasică" este:

```
SELECT SUM(SporVechime)
FROM sporuri
WHERE  An = 2007 AND Luna=7 AND SporVechime <> 130
```

O variantă ceva mai elegantă se redactează prin utilizarea funcției NULLIF prezentă încă din SQL-92 care, în interogarea de mai jos, converteşte orice apariție a valorii în NULL[7]:

```
SELECT SUM(NULLIF(SporVechime,130))
FROM sporuri
WHERE  An = 2007 AND Luna=7
```

Pentru acest gen de probleme se pot folosi şi structuri de tip CASE (paragraful 8.5), sau funcții „propietare", cum ar fi REPLACE sau DECODE în Oracle[8]:

```
SELECT SUM(REPLACE(SporVechime,130,NULL))
FROM sporuri
WHERE  An = 2007 AND Luna=7
```

---

[7] Ambele soluții returnează, pentru conținutul curent al tabelelor, valoarea 945 (prin comparație cu 1205 care este totalul sporurilor de vechime pe luna iulie 2007)

[8] Funcția DECODE a fost introdusă şi în DB2, versiunea 9.5, însă nu funcționează în acest exemplu, din cauza prezenței NULL în lista-argument.

8.4. Joncțiunea externă

Standardul SQL-92 a introdus operatorii necesari joncțiunii externe pe care am prezentat-o în paragraful 4.4.5 (figura 4.27):

- LEFT OUTER JOIN pentru joncțiune externă la stânga,
- RIGHT OUTER JOIN pentru joncțiune externă la dreapta,
- FULL OUTER JOIN pentru joncțiune externă totală (în ambele direcții)[9].

Dacă ne raportăm la exemplul teoretic din algebra relațională, atunci joncțiunile externe la stânga, la dreapta şi totală dintre relațiile R1 şi R2 se transcriu în SQL astfel:

- Joncțiune externă la stânga:

```
SELECT *
FROM r1 LEFT OUTER JOIN r2 ON r1.C=r2.C
```

- Joncțiune externă la dreapta:

```
SELECT *
FROM r1 RIGHT OUTER JOIN r2 ON r1.C=r2.C
```

- Joncțiune externă totală:

```
SELECT *
FROM r1 FULL OUTER JOIN r2 ON r1.C=r2.C
```

Următorul exemplu este desprins tot din algebra relațională (paragraful 4.4.5, exemplul 21):

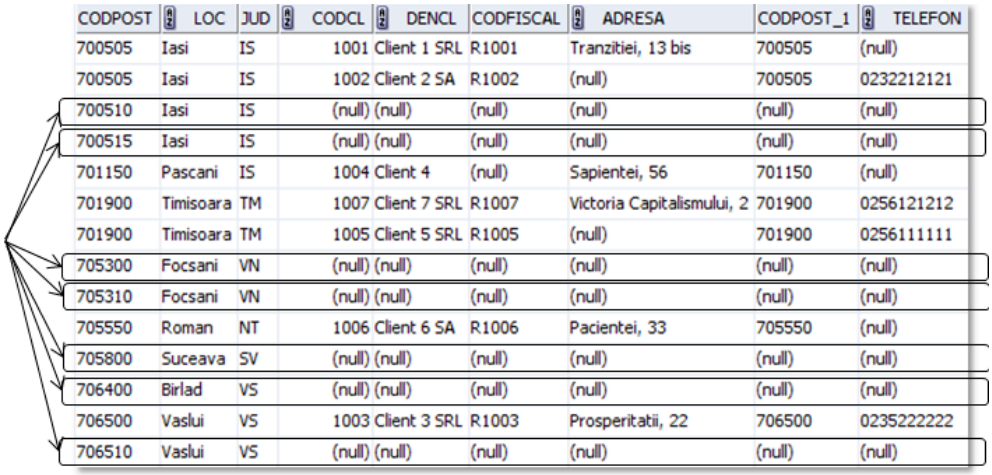
Care sunt codurile poştale în care nu avem nici un client ?
Prin joncțiunea externă la stânga dintre CODURI_POSTALE şi CLIENTI:

```
SELECT *
FROM coduri_postale LEFT OUTER JOIN clienti
        ON coduri_postale.CodPost = clienti.CodPost
ORDER BY 1
```

se obține tabela din figura 8.23. Primele trei coloane corespund tabelei CODURI_POSTALE, iar următoarele (ultimele) şase, tabelei CLIENTI. Liniile puse în

[9] Toate cele patru servere BD acceptă renunțarea la cuvântul OUTER pentru orice tip de joncțiune externă: LEFT JOIN, RIGHT JOIN, FULL JOIN. Noi, însă, vom folosi titulatura completă, pentru claritate.

evidență țin strict de joncționarea externă la stânga, deoarece prezintă valori nenule numai pentru primele trei atribute (din CODURI_POSTALE), completate cu valori NULL pentru toate atributele preluate din CLIENTI. Joncțiunea internă (echi- sau naturală) ar fi extras numai liniile nemarcate.



CODPOST	LOC	JUD	CODCL	DENCL	CODFISCAL	ADRESA	CODPOST_1	TELEFON
700505	Iasi	IS	1001	Client 1 SRL	R1001	Tranzitiei, 13 bis	700505	(null)
700505	Iasi	IS	1002	Client 2 SA	R1002	(null)	700505	0232212121
700510	Iasi	IS	(null)	(null)	(null)	(null)	(null)	(null)
700515	Iasi	IS	(null)	(null)	(null)	(null)	(null)	(null)
701150	Pascani	IS	1004	Client 4	(null)	Sapientei, 56	701150	(null)
701900	Timisoara	TM	1007	Client 7 SRL	R1007	Victoria Capitalismului, 2	701900	0256121212
701900	Timisoara	TM	1005	Client 5 SRL	R1005	(null)	701900	0256111111
705300	Focsani	VN	(null)	(null)	(null)	(null)	(null)	(null)
705310	Focsani	VN	(null)	(null)	(null)	(null)	(null)	(null)
705550	Roman	NT	1006	Client 6 SA	R1006	Pacientei, 33	705550	(null)
705800	Suceava	SV	(null)	(null)	(null)	(null)	(null)	(null)
706400	Birlad	VS	(null)	(null)	(null)	(null)	(null)	(null)
706500	Vaslui	VS	1003	Client 3 SRL	R1003	Prosperitatii, 22	706500	0235222222
706510	Vaslui	VS	(null)	(null)	(null)	(null)	(null)	(null)

Figura 8.23. Joncţiunea externă la stânga dintre CODURI_POSTALE şi CLIENTI

Pentru că în rezultat interesează numai codurile poştale în care nu îşi are sediul niciun client, se va folosi o clauză WHERE în care unul dintre atributele (de preferință cheia primară sau, oricum, un atribut NOT NULL) preluate din CLIENTI este NULL:

```
SELECT *
FROM coduri_postale
        LEFT OUTER JOIN clienti ON coduri_postale.CodPost =
clienti.CodPost
WHERE clienti.CodPost IS NULL
ORDER BY 1
```

CODPOST	LOC	JUD	CODCL	DENCL	CODFISCAL	ADRESA	CODPOST_1	TELEFON
700510	Iasi	IS	(null)	(null)	(null)	(null)	(null)	(null)
700515	Iasi	IS	(null)	(null)	(null)	(null)	(null)	(null)
705300	Focsani	VN	(null)	(null)	(null)	(null)	(null)	(null)
705310	Focsani	VN	(null)	(null)	(null)	(null)	(null)	(null)
705800	Suceava	SV	(null)	(null)	(null)	(null)	(null)	(null)
706400	Birlad	VS	(null)	(null)	(null)	(null)	(null)	(null)
706510	Vaslui	VS	(null)	(null)	(null)	(null)	(null)	(null)

Figura 8.24. Codurile poştale la care nu există adrese de clienţi

Care este situația încasării facturii 1114 ?

Revenim la exemple din paragrafele 6.7.2 şi 6.7.4 (de la funcţiile agregat SUM şi MAX). Spuneam, la momentele respective, că soluţiile formulate sunt valabile numai pentru facturile cu măcar o tranşă de încasare. Beneficiind de avantajele joncţiunii externe, se poate redacta o soluţie ameliorată care va funcţiona în orice situaţie. Am ales, pentru început, factura 1114 pentru că aceasta nu are nici o tranşă de încasare. Dacă vom joncţiona extern la stânga rezultatul joncţiunii dintre LINIIFACT şi PRODUSE cu tabela INCASFACT:

```
SELECT *
FROM (liniifact lf
        INNER JOIN produse p ON lf.CodPr=p.CodPr )
        LEFT OUTER JOIN incasfact i ON lf.NrFact=i.NrFact
WHERE lf.NrFact = 1114
```

se va obţine rezultatul din figura 8.25.

NRFACT	LINIE	CODPR	CANTITATE	PRETUNIT	CODPR_1	DENPR	UM	GRUPA	PROCTVA	CODINC	NRFACT_1	TRANSA
1114	1	2	70	1070	2	Produs 2	kg	Bere	0,09	(null)	(null)	(null)
1114	2	4	30	1705	4	Produs 4	l	Dulciuri	0,09	(null)	(null)	(null)
1114	3	5	700	7064	5	Produs 5	buc	Tigari	0,19	(null)	(null)	(null)

Figura 8.25. Folosirea joncţiunii externe pentru a afla situaţia încasării facturii 1114

Răspunsul la întrebare se află destul de simplu (vezi cele două valori în figura 8.26):

```
SELECT SUM(Cantitate * PretUnit * (1+ProcTVA)) AS "Facturat",
        SUM(COALESCE(Transa,0)) AS "Incasat"
FROM (liniifact lf
        INNER JOIN produse p ON lf.CodPr=p.CodPr )
        LEFT OUTER JOIN incasfact i ON lf.NrFact=i.NrFact
WHERE lf.NrFact = 1114
```

Facturat	Incasat
6021706,5	0

Figura 8.26. Valorile (corecte) facturate şi încasate pentru factura 1114

Soluţia nu este însă corectă pentru toate facturile, după cum vom vedea imediat.

Care este situaţia încasării facturii 1117 ?

Spre deosebire de 1114, această factură prezintă trei tranşe de încasare, rezultatul joncţiunii externe (identic cu al joncţiunii interne):

```
SELECT *
FROM (liniifact lf
        INNER JOIN produse p ON lf.CodPr=p.CodPr )
        LEFT OUTER JOIN incasfact i ON lf.NrFact=i.NrFact
WHERE lf.NrFact = 1117
```

fiind cel din figura 8.27.

NRFACT	LINIE	CODPR	CANTITATE	PRETUNIT	CODPR_1	DENPR	UM	GRUPA	PROCTVA	CODINC	NRFACT_1	TRANSA
1117	2	1	100	950	1	Produs 1	buc	Tigari	0,19	1236	1117	9754
1117	1	2	100	1000	2	Produs 2	kg	Bere	0,09	1236	1117	9754
1117	2	1	100	950	1	Produs 1	buc	Tigari	0,19	1237	1117	9754
1117	1	2	100	1000	2	Produs 2	kg	Bere	0,09	1237	1117	9754
1117	2	1	100	950	1	Produs 1	buc	Tigari	0,19	1239	1117	3696
1117	1	2	100	1000	2	Produs 2	kg	Bere	0,09	1239	1117	3696

Figura 8.27. Folosirea joncţiunii externe pentru a afla situaţia încasării facturii 1117

După cum am convenit spre finalul paragrafului 6.7.4, pentru obținerea valorii facturate trebuie împărțită suma totală la numărul de tranşe de încasări (repetarea se producea din cauza joncțiunii). Atunci când nu există nici o tranşă, împărțirea trebuie să se facă la 1, astfel încât expresia valorii totale a facturii va fi: *SUM (Cantitate * PretUnit * (1 + ProcTVA)) / COUNT (DISTINCT VALUE (i.CodInc, 1))*. În schimb, totalul tranşelor încasate (0 dacă nu există nici o tranşă) trebuie împărțit, pentru fiecare factură, la numărul de linii din factură. De aici expresia: *SUM (VALUE (Transa, 0)) / MAX (lf.Linie)*. Prin urmare, soluția valabilă pentru toate facturile pare a fi:

```
SELECT SUM(Cantitate * PretUnit * (1+ProcTVA)) /
                    COUNT(DISTINCT I.CodInc) AS "Facturat",
       SUM(Transa) / MAX(LF.Linie) AS "Incasat"
FROM (liniifact lf
      INNER JOIN produse p ON lf.CodPr=p.CodPr)
      LEFT OUTER JOIN incasfact i ON lf.NrFact=i.NrFact
WHERE lf.NrFact = 1117
```

Cele două valori sunt identice cu cele din figura 6.61 (paragraful 6.7.2). Supriză, însă ! Dacă revenim cu soluția aceasta la factura 1114, ne alegem cu un mesaj de eroare – vezi figura 8.28. Ruşinea pe care am pățit-o se datorează faptului că, dacă factura nu are nicio tranşă de încasare, numitorul expresiei *SUM(Cantitate * PretUnit * (1+ProcTVA)) / COUNT(DISTINCT i.CodInc)* este zero. Din fericire, putem rezolva problema folosindu-ne de funcția COALESCE în două locuri:

- pentru conversia eventualelor valori NULL ale CodInc în zero, astfel încât COUNT să le ia în considerare;
- pentru conversia eventualelor valori NULL ale atributului Transa în zero, astfel cât *SUM(COALESCE(Transa,0)) / MAX(LF.Linie)* să returneze zero.

Ecce rezolvarea !

```
SELECT SUM(Cantitate * PretUnit * (1+ProcTVA)) /
       COUNT(DISTINCT COALESCE(i.CodInc,0)) AS "Facturat",
       SUM(COALESCE(Transa,0)) / MAX(lf.Linie) AS "Incasat"
FROM (liniifact lf
      INNER JOIN produse p ON lf.CodPr=p.CodPr)
      LEFT OUTER JOIN incasfact i ON lf.NrFact=i.NrFact
```

```
WHERE lf.NrFact = 1114
```

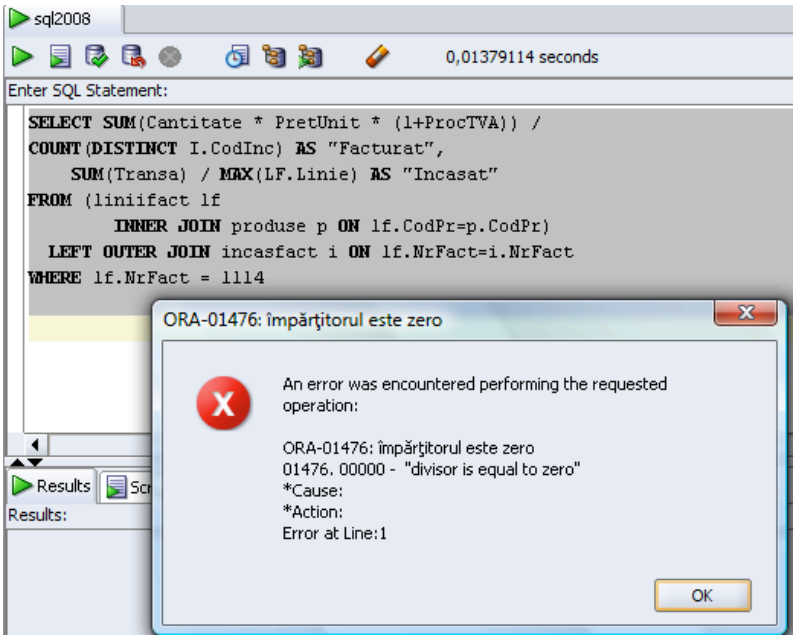


Figura 8.28. Eroarea la execuţia interogării pentru factura 1114

Care sunt valorile facturate şi încasate ale fiecărei facturi emise în luna august 2007 ?

Şi acesta este un caz rezolvat incomplet în paragraful 7.1 (figura 7.7). Beneficiind de experienţa exemplului anterior, formulăm soluţia:

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

```
WHERE EXTRACT (YEAR FROM DataFact)=2007
      AND EXTRACT (MONTH FROM DataFact)=8
GROUP BY lf.NrFact
ORDER BY 1
```

Rezultatul pare a fi unul corect – vezi figura 8.29. Toate soluțiile din acest paragraf, cu excepția celei de mai sus, sunt acceptate de cele patru dialecte. Aceasta, însă, este funcțională doar în Oracle şi PostgreSQL[10].

NRFACT	Facturat	Incasat
1111	4346037,5	53996
1112	125516	125516
1113	106275	106275
1114	6021706,5	0
1115	151237,5	0
1116	126712,5	0
1117	222050	23204
1118	201975	201975
1119	5774498,5	0
1120	97664	7315
1121	4737838	0
2111	4442959,5	0
2112	153442	0
2113	127530	0
2115	110907,5	0
2116	136849,5	0
2117	287855	0
2118	179565	0
2119	5819668	0
2121	4741271,5	0

Figura 8.29. Valorile facturate şi încasate ale fiecărei facturi

Care au fost sporurile de noapte acordate angajaților pe lunile mai şi iunie 2007 ?

Situația obținută se referă la două luni. Există însă angajați care nu au acest spor pe mai sau pe iunie, sau chiar pe nicio lună (din cele două). Cu interogarea:

---

[10] În DB2 şi SQL Server cele două funcții EXTRACT trebuie înlocuite cu YEAR şi MONTH.

```
SELECT An, Luna, personal2.Marca, NumePren, SporNoapte
FROM personal2 INNER JOIN sporuri
        ON personal2.Marca=sporuri.Marca
WHERE An=2007 AND Luna IN (5,6)
ORDER BY NumePren, An, Luna
```

rezultatul arată ca în figura 8.30.

AN	LUNA	MARCA	NUMEPREN	SPORNOAPTE
2007	5	1	ANGAJAT 1	0
2007	6	1	ANGAJAT 1	0
2007	5	10	ANGAJAT 10	0
2007	6	10	ANGAJAT 10	12
2007	5	2	ANGAJAT 2	45
2007	6	2	ANGAJAT 2	0
2007	5	3	ANGAJAT 3	0
2007	6	4	ANGAJAT 4	15
2007	6	5	ANGAJAT 5	15

Figura 8.30. Sporurile de noapte pe mai şi iunie – varianta 1 de afişare

Pe noi ne interesează, însă, o formă mai elegantă de prezentare, ca în figura 8.31.

MARCA	NUMEPREN	SPORNOAPTE_MAI	SPORNOAPTE_IUNIE
1	ANGAJAT 1	0	0
10	ANGAJAT 10	0	12
2	ANGAJAT 2	45	0
3	ANGAJAT 3	0	(null)
4	ANGAJAT 4	(null)	15
5	ANGAJAT 5	(null)	15
6	ANGAJAT 6	(null)	(null)
7	ANGAJAT 7	(null)	(null)
8	ANGAJAT 8	(null)	(null)
9	ANGAJAT 9	(null)	(null)

Figura 8.31. Sporurile de noapte pe mai şi iunie – rezultatul dorit

Dacă am joncţiona intern tabelele PERSONAL2 cu SPORURI pentru luna mai 2007, rezultatul ar conţine atâtea linii câti angajaţi au primit spor de noapte în această lună. Pot fi toţi, dar, la fel de bine, nu. Pentru a forţa includerea în rezultat a tuturor angajaţilor, apelăm la joncţiunea externă la stânga, obţinând tabela din figura 8.32.

```
SELECT personal2.Marca, NumePren,
        SporNoapte AS SporNoapte_Mai
FROM personal2
        LEFT OUTER JOIN sporuri ON personal2.Marca = sporuri.Marca
                AND An=2007 AND Luna=5
ORDER BY NumePren
```

MARCA	NUMEPREN	SPORNOAPTE_MAI
1	ANGAJAT 1	0
10	ANGAJAT 10	0
2	ANGAJAT 2	45
3	ANGAJAT 3	0
4	ANGAJAT 4	(null)
5	ANGAJAT 5	(null)
6	ANGAJAT 6	(null)
7	ANGAJAT 7	(null)
8	ANGAJAT 8	(null)
9	ANGAJAT 9	(null)

Figura 8.32. Sporurile de noapte pe luna mai (pentru toţi angajaţii)

Persoanele care nu erau angajate în această perioadă prezintă valori NULL pentru atributele An şi Luna. Pentru afişarea pe coloane separate a sporurilor de noapte pe lunile mai şi iunie (ca în figura 8.31) sunt necesare două joncţiuni externe ale tabelei PERSONAL2 cu două instanţe ale tabelei SPORURI (instanţe care vor avea alias-urile s1 şi s2).

```
SELECT personal2.Marca, NumePren,
        s1.SporNoapte AS SporNoapte_Mai,
        s2.SporNoapte AS SporNoapte_Iunie
FROM personal2
        LEFT OUTER JOIN sporuri s1 ON personal2.Marca=s1.Marca
                AND s1.An=2007 AND s1.Luna=5
        LEFT OUTER JOIN sporuri s2 ON personal2.Marca=s2.Marca
AND s2.An=2007 AND s2.Luna=6
ORDER BY NumePren
```

Elementul-cheie îl constituie prezenţa operatorului joncţiunii externe în dreptul atributelor An şi Luna. Prin această se includ în rezultat şi liniile din tabela PERSONAL2 care nu prezintă corespondenţă după atributul Marca cu tabela SPORURI pentru cele două luni.

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

Sunt necesare trei instanţe ale tabelei SPORURI (notate s1, s2 şi s3), fraza SELECT devenind un pic mai ponderală:

```
SELECT personal2.Marca, NumePren,
        COALESCE (s1.SporNoapte,0) AS SpNoapte_Aprilie,
        COALESCE (s2.SporNoapte,0) AS SpNoapte_Mai,
        COALESCE (s3.SporNoapte,0) AS SpNoapte_Iunie,
        COALESCE (s1.SporNoapte,0) + COALESCE (s2.SporNoapte,0)
                + COALESCE (s3.SporNoapte,0) AS Sp_Noapte_Trim2
FROM personal2
```

```
            LEFT OUTER JOIN sporuri s1 ON personal2.Marca=s1.Marca
AND s1.An=2007 AND s1.Luna=4
            LEFT OUTER JOIN sporuri s2 ON personal2.Marca=s2.Marca
AND s2.An=2007 AND s2.Luna=5
            LEFT OUTER JOIN sporuri s3 ON personal2.Marca=s3.Marca
AND s3.An=2007 AND s3.Luna=6
ORDER BY NumePren
```

Rezultatul este asemănător celui din figura 8.33.

MARCA	NUMEPREN	SPNOAPTE_APRILIE	SPNOAPTE_MAI	SPNOAPTE_IUNIE	SP_NOAPTE_TRIM2
1	ANGAJAT 1	0	0	0	0
10	ANGAJAT 10	0	0	12	12
2	ANGAJAT 2	45	45	0	90
3	ANGAJAT 3	156	0	0	156
4	ANGAJAT 4	0	0	15	15
5	ANGAJAT 5	0	0	15	15
6	ANGAJAT 6	0	0	0	0
7	ANGAJAT 7	0	0	0	0
8	ANGAJAT 8	0	0	0	0
9	ANGAJAT 9	0	0	0	0

Figura 8.33. Sporurile de noapte pe trimestrul al II-lea, pe luni şi cumulat

În PostgreSQL, pentru asemenea gen de problemă putem folosi o funcție „proprietară" care „orizontalizează verticalitățile"[11] - CROSSTAB:

```
SELECT *
FROM crosstab (
            'SELECT Marca, Luna, SporNoapte
             FROM sporuri
             WHERE An=2007 AND luna BETWEEN 4 AND 6
             ORDER BY 1,2,3'
                  )
                  AS SpNoapte_Orizontalizat (
                        Marca NUMERIC,
```

[11] Titulatura este preluată din Fiola de SQL nr. 4 din "îndepărtatul" NetReport nr. 95 din luna august 2000, şi inspirată din fotbal (vezi explicațiile din fiolă).

```
SpNoapte_Apr NUMERIC,
SpNoapte_Mai NUMERIC,
SpNoapte_Iun NUMERIC
)
```

Argumentul funcţiei este o interogare SQL în care specificăm că liniile care ne interesează în rezultat sunt în tabela SPORURI cu valorile anului 2007, iar luna să fie cuprinsă între 4 şi 6[12], iar rezultatul tabelar să aibă numele SPNOAPTE_ORIZONTALIZAT, titulatura şi tipologia coloanelor acestuia fiind precizate între parantezele ce urmează clauzei AS. Ceea ce se obţine în urma execuţiei acestei interogări PostgreSQL seamănă leit cu figura 8.34.

marca numeric	spnoapte_apr numeric	spnoapte_mai numeric	spnoapte_iun numeric
1	0.00	0.00	0.00
2	45.00	45.00	0.00
3	156.00	0.00	
4	15.00		
5	15.00		
10	0.00	12.00	

Figura 8.34. Funcţia PostgreSQL CROSSTAB()

Astfel, în PostgreSQL putem obţine o listă similară celei din figura 8.33 joncţionând extern la dreapta rezultatul funcţiei CROSSTAB cu tabela PERSONAL2:

```
SELECT p2.Marca, p2.NumePren,
        COALESCE(SpNoapte_Apr,0.00) AS SpNoapte_Apr,
        COALESCE(SpNoapte_Mai,0.00) AS SpNoapte_Mai,
        COALESCE(SpNoapte_Iun,0.00) AS SpNoapte_Iun,
        COALESCE(SpNoapte_Apr,0.00) + COALESCE(SpNoapte_Mai,0) +
                COALESCE(SpNoapte_Iun,0) AS Sp_Noapte_Trim2
FROM crosstab(
        'SELECT Marca, Luna, SporNoapte
         FROM sporuri
         WHERE An=2007 AND luna BETWEEN 4 AND 6
```

[12] Pentru detalii privind sintaxa şi modul de funcţionale al funcţiei, vezi documentaţia PostgreSQL 8.3, la pagina: http://www.postgresql.org/docs/8.3/interactive/tablefunc.html

```
        ORDER BY 1,2,3')
                AS sp_tab (
                        Marca NUMERIC,
                        SpNoapte_Apr NUMERIC,
                        SpNoapte_Mai NUMERIC,
                        SpNoapte_Iun NUMERIC
                        )
         RIGHT OUTER JOIN personal2 p2 ON sp_tab.Marca=p2.Marca
ORDER BY 2
```

Să se afişeze, pentru toţi clienţii; valoarea vânzărilor pe lunile august şi septembrie 2007, şi totalul vânzărilor pe cele două luni; în plus, pe acelaşi raport să se afişeze totalul vânzărilor pe fiecare din cele două luni (pentru toţi clienţii) şi pe ambele.

Dacă interogarea anterioară era cam ponderală, ce să mai spunem de aceasta?:

```
SELECT DenCl,
        ROUND(COALESCE(SUM(lf1.Cantitate * lf1.PretUnit *
                (1+p1.ProcTVA)),0)) AS Vinz_Aug2007,
        ROUND(COALESCE(SUM(lf2.Cantitate * lf2.PretUnit *
                (1+p2.ProcTVA)),0)) AS Vinz_Sep2007,
        ROUND(SUM(lf1.Cantitate * lf1.PretUnit *
                (1+p1.ProcTVA))) + ROUND(COALESCE(SUM(lf2.Cantitate *
                        lf2.PretUnit * (1+p2.ProcTVA)),0))
                AS Vinz_AugSep2007
FROM clienti c
        LEFT OUTER JOIN
                (
                facturi f1 INNER JOIN liniifact lf1 ON f1.NrFact=lf1.NrFact
                        AND EXTRACT (YEAR FROM DataFact)=2007
                        AND EXTRACT (MONTH FROM DataFact)=8
              INNER JOIN produse p1 ON lf1.CodPr=p1.CodPr
                ) ON c.CodCl=f1.CodCl
        LEFT OUTER JOIN
        (
        facturi f2 INNER JOIN liniifact lf2 ON f2.NrFact=lf2.NrFact
                AND EXTRACT (YEAR FROM DataFact)=2007
                AND EXTRACT (MONTH FROM DataFact)=9
        INNER JOIN produse p2 ON lf2.CodPr=p2.CodPr
        ) ON c.CodCl=f2.CodCl
GROUP BY DenCl
UNION
SELECT 'T O T A L ',
```

```
        ROUND(SUM(lf1.Cantitate * lf1.PretUnit *
                (1+p1.ProcTVA))) AS Vinz_Aug2007,
        ROUND(COALESCE(SUM(lf2.Cantitate * lf2.PretUnit *
                (1+p2.ProcTVA)),0)) AS Vinz_Sep2007,
        ROUND(SUM(lf1.Cantitate * lf1.PretUnit * (1+p1.ProcTVA))) +
                ROUND(COALESCE(SUM(lf2.Cantitate *
                lf2.PretUnit * (1+p2.ProcTVA)),0)) AS Vinz_AugSep2007
FROM clienti c
        LEFT OUTER JOIN
        (
        facturi f1 INNER JOIN liniifact lf1 ON f1.NrFact=lf1.NrFact
                        AND EXTRACT (YEAR FROM DataFact)=2007
                AND EXTRACT (MONTH FROM DataFact)=8
        INNER JOIN produse p1 ON lf1.CodPr=p1.CodPr
        ) ON c.CodCl=f1.CodCl
        LEFT OUTER JOIN
        (
        facturi f2 INNER JOIN liniifact lf2 ON f2.NrFact=lf2.NrFact
                        AND EXTRACT (YEAR FROM DataFact)=2007
                AND EXTRACT (MONTH FROM DataFact)=9
    INNER JOIN produse p2 ON lf2.CodPr=p2.CodPr
        ) ON c.CodCl=f2.CodCl
ORDER BY DenCl
```

Rândul de total având tipicul său, interogarea conține două fraze SELECT conectate prin UNION. În prima, cea corespunzătoare rândurilor „curente" din raport se joncționează extern la stânga tabela CLIENTI cu două instanțe ale joncțiunii FACTURI-LINIIFACT-PRODUSE; fiecare cele două instanțe corespunde uneia dintre cele două luni calendaristice. Probabil că rezultatul din figura 8.35 merită efortul. Sintaxa este una Oracle/PostgreSQL, dar trecerea la varianta DB2/SQL Server nu mai ridică, de acum, probleme deosebite, singurele vizate de schimbări fiind funcțiile EXTRACT şi ROUND.

DENCL	VINZ_AUG2007	VINZ_SEP2007	VINZ_AUGSEP2007
Client 1 SRL	80322008	85361879	165683887
Client 2 SA	233805	255060	488865
Client 3 SRL	46376666	46557344	92934010
Client 4	9479110	0	9479110
Client 5 SRL	557916	613768	1171684
Client 6 SA	6021707	0	6021707
Client 7 SRL	263562	273699	537261
T O T A L	143254773	133061750	276316523

Figura 8.35. Totalizări pe orizontală şi verticală

8.5. Structuri de control alternative

Deşi sună apetisant, în acest paragraf nu vom vorbi de programare propriu-zisă, ci de evaluarea unei condiţii (predicat) şi, în funcţie de variantele testate, returnarea unor valori. Ca multe alte generalizări, nici propoziţia anterioară nu spune mare lucru. Mai precis (cu 7%!), în clauza SELECT (cu precădere) a unei interogări vom introduce uneori o serie de condiţii, cu ajutorul unei structuri CASE sau similare, prin care valoarea extrasă în rezultat să depindă de rezultatul evaluării expresiei pentru linia respectivă.

În SQL structura alternativă înseamnă CASE, automat structură „alternativă multiplă"[13]. Unii vor fi nemulţumiţi de inxistenţa structurii simple – IF sau IIF (IF imediat). Dacă ne gândim la păstrarea simplităţii sintaxei SQL, existenţa doar a structurii CASE este acceptabilă.

Care este sexul fiecărui pacient, cunoscând CNP-ul ?

După cum am discutat şi în paragraful 6.5, folosind funcţii de extragere a unor porţiuni dintr-un şir de caractere, putem afla câteva informaţii pe baza CNP-ului. Dacă prima cifră dintr-un cod numeric personal este 1 sau 5, atunci sexul este bărbătesc, iar dacă este 2 sau 6 este femeiesc.

O structură de tip CASE este delimitată de cuvintele CASE şi END. Între acestea, sunt descrise diferite variante testare (WHEN...) şi valoarea returnată la îndeplinirea condiţiei testate (THEN...). Nefiind prea mult de testat, CASE-ul nostru seamănă mai degrabă cu un IF.

```
SELECT NumePacient, CNP,
        CASE
                WHEN SUBSTR(CAST (CNP AS CHAR(13)), 1, 1)
                        IN ('1', '5') THEN 'Barbatesc'
                ELSE 'Femeiesc'
        END AS Sex
```

[13] În româneşte, „structură alternativă multiplă" sună rău, ca să nu spun că-i pleonasm în toată regula, atâta vreme cât alternativa presupune alegerea unei variante din două posibile. Pentru cei încruntaţi, aş vrea să arunc pisica către alte barbarisme, informatice şi non informatice, gen *setare*, *customizare*, *focusare*, *target* etc.

```
FROM pacienti
ORDER BY NumePacient
```
[14]

Atunci când toate variantele de test iau în calcul valori punctuale dintr-un acelaşi atribut (sau expresie), atributul (sau expresia) poate fi plasat imediat după CASE:

```
SELECT NumePacient, CNP,
        CASE SUBSTR(CAST (CNP AS CHAR(13)), 1, 1)
                WHEN '1'THEN 'Barbatesc'
                WHEN '5'THEN 'Barbatesc'
                ELSE 'Femeiesc'
        END AS Sex
FROM pacienti
ORDER BY NumePacient
```

Ambele variante obţin rezultatul din figura 8.36.

NUMEPACIENT	CNP	SEX
Bagdasar Adela	2601002250611	Femeiesc
Buzatu Corneliu	1650512370514	Barbatesc
Cazan Ana Maria	2690202200345	Femeiesc
Ionascu Ionelia	6001122390199	Femeiesc
Popescu Maria-Mirabela	2721231300888	Femeiesc
Spineanu Marius	5010101380625	Barbatesc
Stroe Mihaela	2601228390834	Femeiesc
Stroescu Mihaela Oana	6020719120545	Femeiesc

Figura 8.36. "Obţinerea" sexului din CNP-ul pacienţilor

Din păcate, în această din urmă variantă după WHEN se pot scrie numai valori, nefiind permise (în Oracle, PostgreSQL sau SQL Server) liste sau intervale. Aşa încât ambele variante următoare sunt eronate – vezi figura 8.37:

```
SELECT NumePacient, CNP,
        CASE SUBSTR(CAST (CNP AS CHAR(13)), 1, 1)
```

[14] Varianta SQL Server a acestei interogări este: *SELECT numepacient, CNP, CASE WHEN SUBSTRING(STR(CNP,13), 1, 1) IN ('1', '5') THEN 'Barbatesc' ELSE 'Femeiesc' END AS Sex FROM pacienti ORDER BY numepacient*, adică se schimbă SUBSTR cu SUBSTRING, iar numele atributelor trebuie să fie scrise cu acelaşi combinaţii litere mari-literei mici ca în CREATE TABLE.

```
                    WHEN '1' OR '5' THEN 'Barbatesc'
                    ELSE 'Femeiesc'
                END AS Sex
        FROM pacienti
```

şi

```
        SELECT NumePacient, CNP, CASE SUBSTR(CAST (CNP AS CHAR(13)), 1, 1)
        WHEN IN ('1', '5') THEN 'Barbatesc' ELSE 'Femeiesc'  END AS Sex
        FROM pacienti
```

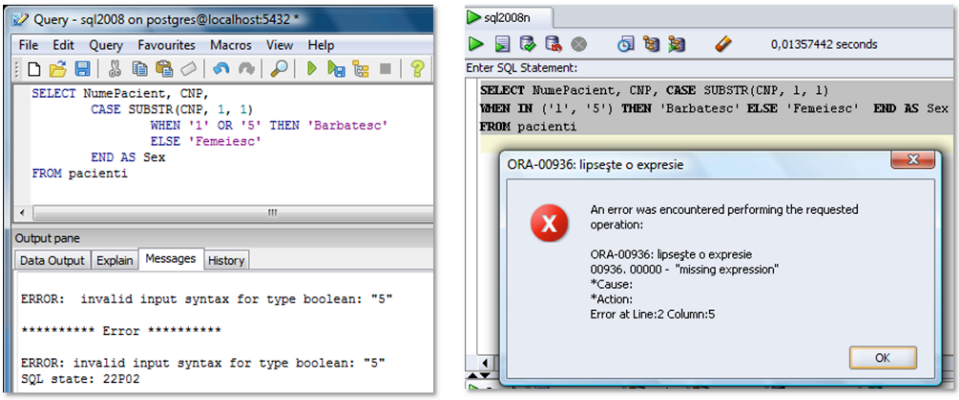


Figura 8.37. Limite ale CASE-ului

Chiar şi celor două variante corecte le putem reproşa că atunci când prima cifră a CNP-ului este eronată (nu e nici una dintre valorile 1, 2, 5 şi 6), sexul returnat va fi Femeiesc, datorită ramurii ELSE. Drept urmare variantele corecte sunt:

```
        SELECT NumePacient, CNP,
          CASE
        WHEN SUBSTR(CAST (CNP AS CHAR(13)), 1, 1) IN ('1', '5') THEN 'Barbatesc'
        WHEN SUBSTR(CAST (CNP AS CHAR(13)), 1, 1) IN ('2', '6') THEN 'Femeiesc'
                ELSE 'Sex necunoscut!!!'
          END AS Sex
        FROM pacienti
        ORDER BY NumePacient
```

şi

```
        SELECT NumePacient, CNP,
          CASE SUBSTR(CAST (CNP AS CHAR(13)), 1, 1)
                WHEN '1' THEN 'Barbatesc'
                WHEN '2' THEN 'Femeiesc'
                WHEN '5' THEN 'Barbatesc'
                WHEN '6' THEN 'Femeiesc'
                ELSE 'Sex necunoscut!!!'
```

```
 END AS Sex
FROM pacienti
ORDER BY NumePacient
```

Putem privi acum funcţiile de conversie ale valorilor NULL ca pe nişte „scurtături" ale structurilor de tip CASE. Spre exemplu, interogarea pentru *aflarea totalului sporurilor pentru fiecare angajat pe luna iulie 2007 ?* (vezi paragraful 8.3, figura 8.19) poate fi rescrisă astfel:

```
SELECT sporuri.Marca, NumePren, Compart,
        CASE WHEN SporVechime IS NULL THEN 0 ELSE SporVechime END
        + CASE WHEN SporNoapte IS NULL THEN 0 ELSE SporNoapte END
        + CASE WHEN SporCD IS NULL THEN 0 ELSE SporCD END
        + CASE WHEN AlteSpor IS NULL THEN 0 ELSE AlteSpor END
        AS TotalSporuri
FROM personal2 INNER JOIN sporuri
        ON personal2.Marca=sporuri.Marca
WHERE An = 2007 AND Luna = 7
```

Chiar dacă efortul intelectual este staţionar, lungimea acestei versiuni, altminteri corectă, este sensibil mai mare, deci avantajul este de partea funcţiilor de conversie a NULLităţilor. În plus, ajungem, ca în zilele bune, la consensul sintaxei celor patru dialecte SQL.

Să se afişeze facturile din luna septembrie 2007 în ordinea alfabetică a observaţiilor

Aceaşi problemă din DB2, PostgreSQL (pre 8.3) şi MS SQL Server (datorată inexistenţei opţiunilor NULLS FIRST şi NULLS LAST) pe care o soluţionam printr-o funcţie COALESCE, poate fi rezolvată printr-un CASE în clauza ORDER BY, după cum urmează:

```
SELECT *
FROM facturi
WHERE DataFact >= '2007-09-01' AND DataFact <= '2007-09-30'
ORDER BY CASE WHEN Obs IS NULL THEN ' ' ELSE Obs END  -- NULLS FIRST
```

şi:

```
SELECT * FROM facturi
WHERE DataFact >= '2007-09-01' AND DataFact <= '2007-09-30'
ORDER BY CASE WHEN Obs IS NULL THEN 'zzz' ELSE Obs END  -- NULLS LAST
```

Care este data naşterii fiecărui pacient, cunoscând CNP-ul ?

În acelaşi celebru paragraf 6.5 am afişat, pe baza CNP-ului, data naşterii pacienţilor, dar numai pentru cei născuţi fie înainte de 2000 fie după 2000 (figurile 6.33 şi 6.35). Cu acea ocazie dibuiam o ciudăţenie a lucrului cu intervale în Oracle (figura 6.34). Beneficiind de structurile de control de tip CASE, şi de experienţa exemplului de acum două pagini şi jumătate, în Oracle putem rezolva problema afişării corecte a datei naşterii, „dintr-o suflare", întrucât cei cu cifra „sexului" 1

sau 2 sunt născuți înainte de 2000, iar cei cu cifra 5 sau 6 după 2000 – vezi figura 8.38.

```
SELECT NumePacient, cnp,
        CASE
                WHEN SUBSTR(cnp, 1, 1) IN ('1','2')
                        THEN ADD_MONTHS(TO_DATE(SUBSTR(cnp, 2, 6),
                                'YYMMDD'), -1200)
                ELSE TO_DATE(SUBSTR(cnp, 2, 6), 'YYMMDD')
        END AS Data_Nasterii_CNP
FROM pacienti
ORDER BY NumePacient
```

NUMEPACIENT	CNP	DATA_NASTERII_CNP
Bagdasar Adela	2601002250611	02-10-1960
Buzatu Corneliu	1650512370514	12-05-1965
Cazan Ana Maria	2690202200345	02-02-1969
Ionascu Ionelia	6001122390199	22-11-2000
Popescu Maria-Mirabela	2721231300888	31-12-1972
Spineanu Marius	5010101380625	01-01-2001
Stroe Mihaela	2601228390834	28-12-1960
Stroescu Mihaela Oana	6020719120545	19-07-2002

Figura 8.38. Obţinerea datei naşterii din CNP-ul pacienţilor

Insistând cu erorile/ciudățeniile din Oracle, am ajuns să fim nedrepți cu „ghiduşiile" din alte dialecte. Încercăm să mai dregem din busuioc, prezentându-vă un mic intermezzo PostgreSQL. Interogarea de mai sus nu funcționează din cauza funcției ADD_MONTHS care este proprietară Oracle. Schimbând ADD_MONTHS cu scăderea unui interval de 100 de ani:

```
SELECT NumePacient, cnp, SUBSTR(CAST (cnp AS CHAR(13)), 1, 1),
        CASE
        WHEN SUBSTR(CAST (cnp AS CHAR(13)), 1, 1) IN ('1','2')
                THEN TO_DATE(SUBSTR(CAST(cnp AS CHAR(13)), 2, 6), 'YYMMDD') –
                        INTERVAL '100 YEAR'
        ELSE TO_DATE(SUBSTR(CAST (cnp AS CHAR(13)), 2, 6), 'YYMMDD')
        END AS Data_Nasterii_CNP
```

FROM pacienti

ORDER BY NumePacient

se obține rezultatul din figura 8.39. Ceea ce intrigă cel mai mult este data de pe linia 5, care este, nici mai mult, nici mai puțin, 31 dec. 1872 !!! Cu această ocazie aflăm că în PostgreSQL, atunci când anul dintr-o dată nu cuprinde şi secolul[15], dacă anul este înainte de anii '70 atunci ca fi „interpretat" ca fiind din secolul 21 (2001-2069), iar dacă e după '70 e considerat ca fiind din secolul 20 (1970-2000).

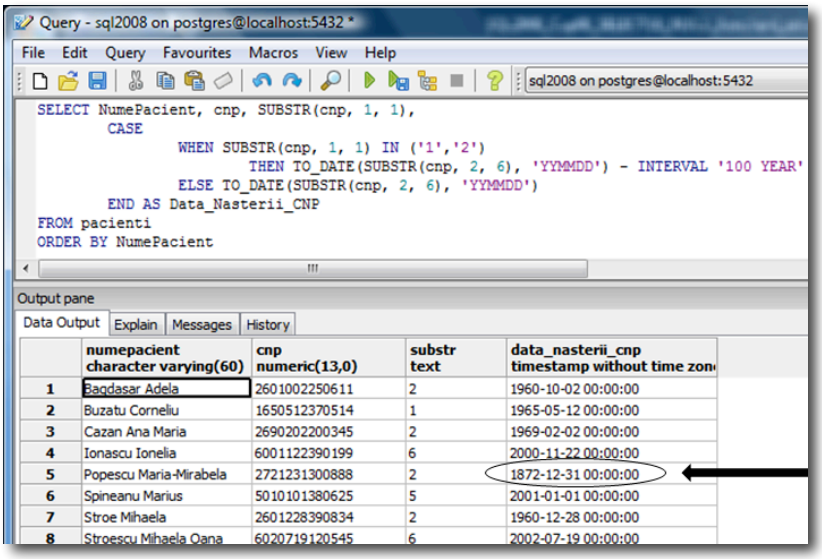


Figura 8.39. O dată calendaristică tulburătoare în PostgreSQL

Cunoscând aceste intimități, suntem în măsură să redactăm versiunea corectă:

```
SELECT NumePacient, cnp,
        TO_DATE(SUBSTR(CAST (cnp AS CHAR(13)), 2, 6), 'YYMMDD'),
        CASE
        WHEN SUBSTR(CAST (cnp AS CHAR(13)), 1, 1) IN ('1','2') THEN
                CASE
                WHEN CAST (SUBSTR(CAST (cnp AS CHAR(13)), 2, 2) AS NUMERIC)
```

[15] Să ne amintim că de la acest gen de reprezentări ale datelor calendaristice a pornit şi celebra *problemă a anului 2000* (Y2K)

```
                    < 70 THEN TO_DATE(SUBSTR(CAST (cnp AS CHAR(13)), 2, 6),
                        'YYMMDD') - INTERVAL '100 YEAR'
                ELSE TO_DATE(SUBSTR(CAST (cnp AS CHAR(13)), 2, 6), 'YYMMDD')
                END
        ELSE TO_DATE(SUBSTR(CAST (cnp AS CHAR(13)), 2, 6), 'YYMMDD')
        END AS Data_Nasterii_CNP
FROM pacienti
ORDER BY NumePacient
```

Adevărul este că ne-am complicat degeaba, pentru că putem renunța la operațiunile cu ani, secolul fiind specificat direct în şablon:

```
SELECT NumePacient, cnp, SUBSTR(CAST (cnp AS CHAR(13)), 1, 1),
        CASE
        WHEN SUBSTR(CAST (cnp AS CHAR(13)), 1, 1) IN ('1','2')
        THEN TO_DATE(('19' || SUBSTR(CAST (cnp AS CHAR(13)), 2, 6)),
        'YYYYMMDD')
        ELSE TO_DATE(('20' || SUBSTR(CAST (cnp AS CHAR(13)), 2, 6)), 'YYYYMMDD')
        END AS Data_Nasterii_CNP
FROM pacienti
ORDER BY NumePacient
```

Interogarea funcționează fără modificări şi în DB2. În SQL Server, dacă anul este specificat pe două poziții, pentru intervalul 00-49 conversia se va face în 2000-2049, în timp ce anii din intervalul 50-99 vor fi considerați 1950-1999. Noi, însă, nu ne vom bizui pe acest lucru, ci vor ține cont doar de cifra sexului (ca în interogarea PostgreSQL de mai sus):

```
SELECT numepacient, CNP,
        CASE
        WHEN SUBSTRING(STR(CNP,13), 1, 1) IN ('1','2') THEN
                CAST (('19'+SUBSTRING(STR(CNP,13),2,2)+'-'
                        +SUBSTRING(STR(CNP,13),4,2) +
                        '-'+SUBSTRING(STR(CNP,13),6,2)) AS SMALLDATETIME)
        ELSE
                CAST ('20'+SUBSTRING(STR(CNP,13),2,2) +'-'
                        +SUBSTRING(STR(CNP,13),4,2) +
                        '-'+SUBSTRING(STR(CNP,13),6,2) AS SMALLDATETIME)
        END AS Data_Nasterii_CNP
FROM pacienti
ORDER BY numepacient
```

Pare (şi chiar este) un efort nejustificat, însă putem rescrie şi interogarea pentru realizarea theta-joncțiunii - exemplul 20 din finalul paragrafului 4.4.4, cea cu afişarea numelui medicului care era de gardă în momentul sosirii fiecărui pacient

în triaj. Echivalent soluției propuse în paragraful 5.7 (rezultatul este cel din figura 5.41), se poate redacta următoarea interogare SQL (sintaxă Oracle):

```
SELECT DataOra_Examinare, NumePacient, NumeDoctor
FROM triaj
        INNER JOIN garzi ON DataOra_Examinare =
                        CASE
                        WHEN DataOra_Examinare BETWEEN Inceput_Garda
                                AND Sfirsit_Garda THEN DataOra_Examinare
                        ELSE DataOra_Examinare + INTERVAL '1' DAY
                        END
        INNER JOIN doctori ON garzi.IdDoctor = doctori.IdDoctor
        INNER JOIN pacienti ON triaj.IdPacient = pacienti.IdPacient
ORDER BY 1
```

De data aceasta theta-joncțiunea este, de fapt, echi-joncțiune. Egalitatea testată este cea între Data_Ora_Examinare şi rezultatul furnizat de CASE. Sintaxa DB2 solicită schimbarea constantei interval din *INTERVAL '1' DAY* în *1 DAY*, iar în PostgreSQL forma aceleaşi constante este *INTERVAL '1 DAY'*. În SQL Server toate numele atributelor din cele patru tabele se scriu cu litere mici, iar expresia-interval *INTERVAL '1' DAY* se înlocuieşte cu *DATEADD(DAY, 1, dataora_examinare)*.

Să se calculeze procentul de discount acordat fiecărei facturi din luna septembrie 2007, în situația în care acesta ar fi acordat pe trei tranşe de valori (totale) ale facturilor, după cum urmează:

- sub 15000 RON – 0%;
- între 15000 şi 199999 – 3%;
- între 199999 şi 399999 – 5%;
- între 400000 şi 600000 – 6%;
- peste 600000 – 7%.

Problema se rezolvă aplicând un CASE pentru fiecare linie (care se referă la o factură) a rezultatului. Ținând seama de faptul că parcurgerea fiecărei ramuri de test este secvențială, predicatul din clauza WHEN nu trebuie să conțină un interval, ci o simplă valoare:

```
SELECT f.NrFact, DataFact,
        TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) AS ValFact,
        CASE
                WHEN TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) < 150000
                        THEN 0
                WHEN TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) < 200000
                        THEN 0.03
                WHEN TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) < 400000
                        THEN 0.05
                WHEN TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) < 600000
```

```
                THEN 0.06
            ELSE 0.07
        END * 100 || '%' AS ProcentDiscount
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)=9
GROUP BY f.NrFact, DataFact
ORDER BY 1
```

Rezultatul (Oracle) este cel din figura 8.40, interogarea funcționând fără migrene şi în PostgreSQL.

NRFACT	DATAFACT	VALFACT	PROCENTDISCOUNT
3111	01-09-2007	4442959	7%
3112	01-09-2007	153442	3%
3113	02-09-2007	127530	0%
3115	02-09-2007	110907	0%
3116	10-09-2007	136849	0%
3117	10-09-2007	287855	5%
3118	17-09-2007	179565	3%

Figura 8.40. Procentul de discount acordat fiecărei facturi din sept. 2007

Mai ciudat se comportă DB2-ul. Dacă dorim să afişăm procentul sub formă de număr:

```
SELECT f.NrFact, DataFact,
        TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) AS ValFact,
        CASE
                WHEN TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) < 150000
                        THEN 0
                WHEN TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) < 200000
                        THEN 0.03
                WHEN TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) < 400000
                        THEN 0.05
                WHEN TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) < 600000
                        THEN 0.06
                ELSE 0.07
        END * 100 AS ProcentDiscount
FROM facturi f INNER JOIN liniifact lf ON f.Nrfact=lf.NrFact
        INNER JOIN produse p ON lf.CodPr=p.CodPr
WHERE YEAR (DataFact)=2007 AND MONTH (DataFact)=9
GROUP BY f.NrFact, DataFact
ORDER BY 1
```

rezultatul este corect – vezi stânga figurii 8.41. Dacă, însă, concatenăm rezultatul CASE-ului cu simbolul % pentru a obține o listă ca în figura 8.40:

NRFACT	DATAFACT	VALFACT	PROCENTDISCOUNT
3111	Sep 1, 2007	4,442,959.0000	7.00
3112	Sep 1, 2007	153,442.0000	3.00
3113	Sep 2, 2007	127,530.0000	0.00
3115	Sep 2, 2007	110,907.0000	0.00
3116	Sep 10, 2007	136,849.0000	0.00
3117	Sep 10, 2007	287,855.0000	5.00
3118	Sep 17, 2007	179,565.0000	3.00

NRFACT	DATAFACT	VALFACT	PROCENTDISCOUNT
3111	Sep 1, 2007	4,442,959.0000	0000000000%
3112	Sep 1, 2007	153,442.0000	0000000000%
3113	Sep 2, 2007	127,530.0000	0000000000%
3115	Sep 2, 2007	110,907.0000	0000000000%
3116	Sep 10, 2007	136,849.0000	0000000000%
3117	Sep 10, 2007	287,855.0000	0000000000%
3118	Sep 17, 2007	179,565.0000	0000000000%

Figura 8.41. Nazuri DB2

```
SELECT f.NrFact, DataFact,
        TRUNC(SUM(Cantitate * PretUnit * (1+ProcTVA)),0) AS ValFact,
        CAST ((CASE
        ...
        END * 100) AS CHAR(10)) || '%' AS ProcentDiscount
FROM
        ...
```

DB2-ul devine năzuros – vezi dreapta figurii 8.41.

Câți dintre clienți sunt din localitatea Iaşi şi câți din afara Iaşului ?

Această problemă ne este de folos în a exemplifică folosirea unui CASE la grupare, sintaxa fiind valabilă în toate cele patru servere BD:

```
SELECT CASE Loc
                WHEN 'Iasi' THEN 'Din Iasi'
                ELSE 'Din afara Iasului'
        END AS Pozitionare,
        COUNT(*) AS NrClienti
FROM clienti INNER JOIN coduri_postale
        ON clienti.CodPost = coduri_postale.CodPost
GROUP BY  CASE Loc  WHEN 'Iasi' THEN 'Din Iasi'
        ELSE 'Din afara Iasului' END
```

POZITIONARE	NRCLIENTI
Din Iasi	2
Din afara Iasului	5

Figura 8.42. Numărul clienţilor ieşeni şi al celor din afara Ieşilor

Câți angajați au primit, pe luna iulie 2007, spor pentru condiții deosebite şi câți nu ?

Prin clauza WHERE sunt filtrate înregistrările din SPORURI, astfel încât vor fi luate în considerare numai datele din iulie 2007, iar gruparea se realizează printr-un CASE aplicat asupra valorii SporCD (vezi figura 8.43), sintaxă acceptată de toate cele patru dialecte SQL:

```
SELECT CASE
            WHEN SporCD > 0 THEN 'Au spor CD'
            ELSE 'Nu au spor CD'
       END AS "Situatie",
       COUNT(*) AS Nr
FROM sporuri
WHERE An=2007 AND Luna=7
GROUP BY  CASE WHEN SporCD > 0
       THEN 'Au spor CD' ELSE 'Nu au spor CD' END
```

Situatie	NR
Nu au spor CD	8
Au spor CD	2

Figura 8.43. Numărul angajaţilor cu şi fără spor pentru condiţii deosebite în iulie 2007

Scadența fiecărei facturi emise este 20 de zile. Dacă însă data limită cade într-o sâmbătă sau duminică, atunci scadența se mută în lunea următoare. Care este, în aceste condiții, data scadentă a fiecărei facturi emise în luna septembrie 2007?

Bănuim că trebuie să aplicăm un CASE expresiei care ne returnează ziua săptămânii în care pică data scadentă. Din păcate, funcția ce returnează ziua din săptămână este nestandardizată. Să începem cu soluția Oracle:

```
SELECT NrFact,
       DataFact AS "Data Emiterii",
       DataFact + INTERVAL '20' DAY AS "Data Scadenta",
       TO_CHAR(DataFact + INTERVAL '20' DAY, 'DAY') AS "Zi scadenta",
       CASE
       WHEN RTRIM(TO_CHAR( DataFact + INTERVAL '20' DAY,
            'DAY'))  IN ('SATURDAY', 'SÂMBĂTĂ')
            THEN DataFact + INTERVAL '22' DAY
       ELSE
            CASE
            WHEN RTRIM(TO_CHAR(
                 DataFact + INTERVAL '20' DAY, 'DAY'))  IN
                 ('SUNDAY', 'DUMINICĂ')
            THEN DataFact + INTERVAL '21' DAY
                 ELSE DataFact + INTERVAL '20' DAY
            END
       END AS "Data scad.corectata",
       TO_CHAR(
            CASE
            WHEN RTRIM(TO_CHAR(DataFact + INTERVAL '20' DAY,
```

```
                'DAY'))  IN ('SATURDAY', 'SÂMBĂTĂ')
            THEN DataFact + INTERVAL '22' DAY
            ELSE
                CASE
                WHEN RTRIM(TO_CHAR(DataFact +
                        INTERVAL '20' DAY, 'DAY'))  IN
                        ('SUNDAY', 'DUMINICĂ')
                THEN DataFact + INTERVAL '21' DAY
                ELSE DataFact + INTERVAL '20' DAY
                END
            END
        , 'DAY')  AS "Zi scad.corectata"
FROM facturi
WHERE TO_CHAR(datafact, 'MM-YYYY') = '09-2007'
```

Aflarea zilei săptămânii presupune, după cum am văzut în paragraful 6.5, folosirea funcției TO_CHAR cu argumentul 'DAY'. Eventualele spații rezultate în urma conversiei sunt eliminate cu RTRIM(...). Întrucât este posibil ca sistemul de operare şi serverul Oracle să fie setate pe varianta engleză, dar, în unele cazuri (al meu, de ex.) şi română am testat numele zilei de sâmbătă sau duminică atât cu titulatura în engleză ('SATURDAY'/'SUNDAY'), cât şi cu cea română ('SÂMBĂTĂ'/'DUMINICĂ'). Când scadența (data facturii + 20 de zile) pică într-o sâmbătă, o corectăm cu două zile (data facturii + 22), când pică într-o duminică, o corectăm cu o zi (data facturii + 21), iar când nu e nici sâmbătă, nici duminică, o păstrăm aşa (data facturii + 20). Lungimea exagerată a frazei SELECT se datorează repetării secvenței CASE necesară afişării denumirii zilei. Rezultatul este cel din figura 8.44.

NRFACT	Data Emiterii	Data Scadenta	Zi scadenta	Data scad.corectata	Zi scad.corectata
3111	01-09-2007	21-09-2007	VINERI	21-09-2007	VINERI
3112	01-09-2007	21-09-2007	VINERI	21-09-2007	VINERI
3113	02-09-2007	22-09-2007	SÂMBATA	24-09-2007	LUNI
3115	02-09-2007	22-09-2007	SÂMBATA	24-09-2007	LUNI
3116	10-09-2007	30-09-2007	DUMINICA	01-10-2007	LUNI
3117	10-09-2007	30-09-2007	DUMINICA	01-10-2007	LUNI
3118	17-09-2007	07-10-2007	DUMINICA	08-10-2007	LUNI

Figura 8.44. Scadenţa rectificată a facturilor

Secvența CASE este prea complicată datorită prezenței a două structuri CASE incluse. Structura seamănă mai degrabă cu un IF tradițional decât cu un CASE. Aşa că pentru varianta PostgresSQL o simplificăm. În plus, pentru aflarea zilei din săptămână vom apela la EXTRACT (DOW...) descrisă în capitolul 6:

```
SELECT NrFact,
        DataFact AS "Data Emiterii",
```

```
        DataFact + INTERVAL '20 DAYS' AS "Data Scadenta",
        TO_CHAR(DataFact + INTERVAL '20 DAYS', 'DAY') AS "Zi scadenta",
        CASE EXTRACT (DOW FROM DataFact + INTERVAL '20 DAYS')
                WHEN 6 THEN DataFact + INTERVAL '22 DAYS'  -- simbata
                WHEN 0 THEN DataFact + INTERVAL '21 DAYS'  -- duminica
                ELSE DataFact + INTERVAL '20 DAYS'  -- luni-vineri
        END AS "Data scad.corectata",
        TO_CHAR (
                CASE EXTRACT (DOW FROM DataFact + INTERVAL '20 DAYS')
                        WHEN 6 THEN DataFact + INTERVAL '22 DAYS'  -- simbata
                        WHEN 0 THEN DataFact + INTERVAL '21 DAYS'  -- duminica
                        ELSE DataFact + INTERVAL '20 DAYS'  -- luni-vineri
                END
                , 'DAY') AS "Zi scad.corectata"
FROM FACTURI
WHERE TO_CHAR(datafact, 'MM-YYYY') = '09-2007'
```

Soluţia DB2 se bazează pe funcţia DAYOFWEEK, care întoarce 1, dacă *data-argument* se referă la duminică, 2 pentru luni… 6 pentru sâmbătă. DAYNAME afişează numele zilei:

```
        SELECT NrFact,
                DataFact AS "Data Emiterii",
                DataFact + 20 DAYS AS "Data Scadenta",
                DAYNAME(DataFact + 20 DAYS) AS "Zi scadenta",
                CASE DAYOFWEEK_ISO (DataFact + 20 DAYS)
                        WHEN 6 THEN DataFact + 22 DAYS  -- simbata
                        WHEN 7 THEN DataFact + 21 DAYS  -- duminica
                        ELSE DataFact + 20 DAYS  -- luni-vineri
                END AS "Data scad.corectata",
                DAYNAME(CASE DAYOFWEEK_ISO (DataFact + 20 DAYS)
                        WHEN 6 THEN DataFact + 22 DAYS  -- simbata
                        WHEN 7 THEN DataFact + 21 DAYS  -- duminica
                        ELSE DataFact + 20 DAYS  -- luni-vineri
                        END) AS "Zi scad.corectata"
        FROM FACTURI
        WHERE MONTH(DataFact)=9 AND YEAR(DataFact)=2007
```

În fine, pentru echilibru, vom prezentat şi o variantă MS SQL Server:

```
SELECT NrFact,
        DataFact AS "Data Emiterii",
        DataFact + 20  AS "Data Scadenta",
```

```
        DATEPART(WEEKDAY, DataFact + 20 ) AS "Nr Zi scadenta",
        DATENAME(WEEKDAY, DataFact + 20 ) AS "Nume Zi scadenta",
        CASE DATEPART (WEEKDAY, DataFact + 20 )
                WHEN 7 THEN DataFact + 22   -- simbata
                WHEN 1 THEN DataFact + 21   -- duminica
                ELSE DataFact + 20   -- luni-vineri
        END AS "Data scad.corectata",
        DATENAME(WEEKDAY,
                        CASE DATEPART (WEEKDAY, DataFact + 20 )
                                WHEN 7 THEN DataFact + 22   -- simbata
                                WHEN 1 THEN DataFact + 21   -- duminica
                                ELSE DataFact + 20   -- luni-vineri
                        END ) AS "Zi scad.corectata"
FROM facturi
WHERE DATEPART(MONTH,DataFact)=9 AND DATEPART (YEAR,DataFact)=2007
```

Care este valoarea vânzărilor din fiecare zi a săptămânii ?

Acum cred că suntem în măsură să rezolvăm onorabil problema din paragraful 7.3. Ordinea dispunerii zilelor săptâmânii în rezultat ca în figura 7.20 se obține apelând la o strucrură CASE în clauza ORDER BY. Iată o variantă comună Oracle/PostgreSQL:

```
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
        CASE
        WHEN RTRIM(TO_CHAR(DataFact, 'day')) IN ('luni', 'monday') THEN 1
        WHEN RTRIM(TO_CHAR(DataFact, 'day')) IN ('marți', 'tuesday') THEN 2
        WHEN RTRIM(TO_CHAR(DataFact, 'day')) IN ('miercuri', 'wednesday') THEN 3
        WHEN RTRIM(TO_CHAR(DataFact, 'day')) IN ('joi', 'thursday') THEN 4
        WHEN RTRIM(TO_CHAR(DataFact, 'day')) IN ('vineri', 'friday') THEN 5
        WHEN RTRIM(TO_CHAR(DataFact, 'day')) IN ('sâmbătă', 'saturday') THEN 6
        WHEN RTRIM(TO_CHAR(DataFact, 'day')) IN ('duminică', 'sunday') THEN 7
        END
```

pe cea MS SQL Server:

```
        SELECT DATENAME(WEEKDAY, DataFact) AS Zi_Saptamina,
```

```
        SUM(Cantitate * PretUnit * (1+ProcTVA))  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
        CASE  LOWER(DATENAME(WEEKDAY, DataFact))
                WHEN 'monday' THEN 1
                WHEN 'tuesday' THEN 2
                WHEN 'wednesday' THEN 3
                WHEN 'thursday' THEN 4
                WHEN 'friday' THEN 5
                WHEN 'saturday' THEN 6
                WHEN 'sunday' THEN 7
        END
```

şi pe cea DB2:

```
SELECT DAYNAME(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 DAYNAME(DataFact)
ORDER BY
        CASE
        WHEN DAYNAME(DataFact) IN ('luni', 'monday') THEN 1
        WHEN DAYNAME(DataFact) IN ('marți', 'tuesday') THEN 2
        WHEN DAYNAME(DataFact) IN ('miercuri', 'wednesday') THEN 3
        WHEN DAYNAME(DataFact) IN ('joi', 'thursday') THEN 4
        WHEN DAYNAME(DataFact) IN ('vineri', 'friday') THEN 5
        WHEN DAYNAME(DataFact) IN ('sâmbătă', 'saturday') THEN 6
        WHEN DAYNAME(DataFact) IN ('duminică', 'sunday') THEN 7
        END
```

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

La momentul rezolvării acestei probleme (paragraful 7.3), cârteam la adresa figurii 7.28 în legătură cu modul de precizare a intervalelor. Cu ajutorul unei structuri CASE putem să facem rezultatul mai elegant – vezi figura 8.45:

```
        SELECT CASE TRUNC(PretUnit,-3)
                        WHEN 0 THEN '   <   1000'
```

```
                ELSE '>= ' || TO_CHAR( TRUNC(PretUnit,-3), '999999')
        END AS "Pret unitar ...",
        COUNT(*) AS "Nr.aparitii"
FROM liniifact
GROUP BY    CASE TRUNC(PretUnit,-3)
            WHEN 0 THEN '   <   1000'
            ELSE '>= ' || TO_CHAR( TRUNC(PretUnit,-3), '999999')
            END
ORDER BY 1
```

Pret unitar ...	Nr.aparitii
< 1000	21
>= 1000	26
>= 6000	3
>= 7000	6

Figura 8.45. Ameliorarea rezultatului din figura 7.28

Sintaxa este comună – Oracle/PostgreSQL. Funcția TO_CHAR, precum şi spațiile de la primul interval - '   <   1000', ne asigură ordonarea corespunzătoare, indiferent de prețurile unitare din facturi. Acelaşi rezultat poate fi obținut în SQL Server cu interogarea:

```
SELECT CASE ROUND(PretUnit,-3,1)
                    WHEN 0 THEN '   <   1000'
        ELSE '>= ' + STR( ROUND(PretUnit,-3,1), 6)
        END AS "Pret unitar ...",
        COUNT(*) AS "Nr.aparitii"
FROM liniifact
GROUP BY CASE ROUND(PretUnit,-3,1)
                    WHEN 0 THEN '   <   1000'
        ELSE '>= ' + STR( ROUND(PretUnit,-3,1), 6)
            END
ORDER BY 1
```

iar în DB2 astfel:

```
SELECT CASE TRUNC(PretUnit,-3)
        WHEN 0 THEN '   <   1000'
        ELSE '>= ' || CAST( TRUNC(PretUnit,-3) AS CHAR(10))
        END AS "Pret unitar ...",
        COUNT(*) AS "Nr.aparitii"
FROM liniifact
GROUP BY
        CASE TRUNC(PretUnit,-3)
```

```
            WHEN 0 THEN '   <  1000'
            ELSE '>= ' || CAST( TRUNC(PretUnit,-3) AS CHAR(10))
        END
ORDER BY 1
```

Să se afişeze în dreptul fiecărei facturi emise în luna august 2007: valoarea facturată, valoarea încasată, diferența de încasat, precum şi un text din care să reiasă dacă factura este fără nici o încasare, încasată parțial sau încasată total (şi cele trei valori) ?

Raportul dorit este o dezvoltare a celui din paragraful 8.5 (figura 8.29). Practic, se doreşte o situație cum este cea din figura 8.46. Fireşte, ultima coloană presupune folosirea unei structuri CASE:

```
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(COALESCE(Transa,0)) / MAX(LF.Linie) = 0
                THEN ' Fara nici o incasare'
        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)
        LEFT OUTER JOIN incasfact i ON lf.NrFact=i.NrFact
WHERE EXTRACT (YEAR FROM DataFact)=2007
        AND EXTRACT (MONTH FROM DataFact)=8
```

```
GROUP BY lf.NrFact
ORDER BY 1
```

NRFACT	Facturat	Incasat	Diferenta	Situatiune
1111	4346037	53996	4292041	Incasata partial
1112	125516	125516	0	*Incasata total*
1113	106275	106275	0	*Incasata total*
1114	6021706	0	6021706	Fara nici o incasare
1115	151237	0	151237	Fara nici o incasare
1116	126712	0	126712	Fara nici o incasare
1117	222050	23204	198846	Incasata partial
1118	201975	201975	0	*Incasata total*
1119	5774498	0	5774498	Fara nici o incasare
1120	97664	7315	90349	Incasata partial
1121	4737838	0	4737838	Fara nici o incasare
2111	4442959	0	4442959	Fara nici o incasare
2112	153442	0	153442	Fara nici o incasare
2113	127530	0	127530	Fara nici o incasare
2115	110907	0	110907	Fara nici o incasare
2116	136849	0	136849	Fara nici o incasare
2117	287855	0	287855	Fara nici o incasare
2118	179565	0	179565	Fara nici o incasare
2119	5819668	0	5819668	Fara nici o incasare
2121	4741271	0	4741271	Fara nici o incasare

Figura 8.46. Un nou raport privind încasarea facturilor din august 2007

În DB2 tot ce rămâne de făcut este să se înlocuiască EXTRACT cu funcţiile YEAR şi MONTH. La aceste modificări, în SQL Server mai trebuie adăugate cele care privesc sintaxa funcţiei TRUNCATE.

Oracle nu avea, până la versiunea 8, structura CASE… WHEN… În schimb, putea fi folosită funcţia DECODE, care are o logică similară. Interesant este că această funcţie apare şi în DB2, din versiunea 9.5. DECODE întoarce o valoare în funcţie de conţinutul unui argument:

```
DECODE (atribut,
        valoare_testată1, valoare_returnată1,
        valoare_testată2, valoare_returnată2,
            ...
        valoare_returnatăN)
```

Valoare_retunatăN este echivalenta ELSE-ului dintr-o structură de tip CASE. Pentru a obţine sumara listă din figura 8.42, o altă soluţie Oracle/DB2 (post-9.5) ar fi:

```
SELECT DECODE (Loc,'Iasi', 'Din Iasi', 'Din afara Iasului') AS Pozitionare,
    COUNT(*) AS NrClienti
```

```
FROM clienti INNER JOIN coduri_postale
        ON clienti.CodPost = coduri_postale.CodPost
GROUP BY  DECODE (Loc,'Iasi', 'Din Iasi', 'Din afara Iasului')
```

Câți angajați au primit, pe luna iulie 2007, spor pentru condiții deosebite şi câți nu ?
Alăturăm „soluției CASE":

```
SELECT CASE
      WHEN SporCD > 0 THEN 'Au spor CD'
      ELSE 'Nu au spor CD'
     END AS Explicatii,
        COUNT(*) AS Nr
FROM sporuri
WHERE An=2007 AND Luna=7
GROUP BY  CASE
       WHEN SporCD > 0 THEN 'Au spor CD'
       ELSE 'Nu au spor CD'
      END
```

o alta „proprietară" Oracle/DB2 bazată pe DECODE:

```
SELECT DECODE(SIGN( NVL(SporCD,0)-1), 1,
        'Au spor CD', 'Nu au spor CD')
              AS Pozitionare,
        COUNT(*) AS NrClienti
FROM SPORURI
WHERE An=2007 AND Luna=7
GROUP BY DECODE(SIGN( NVL(SporCD,0)-1), 1,
        'Au spor CD', 'Nu au spor CD')
```

Aveam nevoie de un mic truc pentru ca în DECODE să transformăm intervalele în listă. De aceea se scade 1 din *NVL(SporCD,0)* şi, dacă rezultatul are semnul +, înseamnă că s-a acordat sporul respectiv, iar în caz contrar, că nu s-a acordat.