Capitolul 2. Modelul relațional de organizare a datelor

Ca în atâtea alte domenii, şi în teoria şi practica bazelor de date se manifestă o dorință apăsată de a scăpa de bătrâni, de a încerca şi altceva. Modelul relațional este „bătrânul care nu vrea să se retragă". Comparația cu Ion Iliescu se opreşte aici, modelul relațional fiind infinit mai benefic pentru istoria recentă a omenirii. Apărut la începutul anilor '70, modelul s-a dovedit a avea oarecum soarta rock-ului aceloraşi ani, cu contestatari destul de vocali, dar cu înlocuitori mult mai palizi (ex. disco-ul, punk-ul), la propriu, la figurat şi mai ales la trecerea timpului.

Afirmația de mai sus este incorectă politic, imposibil de demonstrat ştiințific şi degrabă aducătoare de ranchiună. Aşa că nu are rost să vă enervați (mai bine vă puneți CD-ul cu albumul *London Calling* al celor de la The Clash). Lumea academică, în general, lumea „liber-schimbistă" în ale bazelor de date fremăta, spre finalul anilor 80 şi începutul 90, la ideea că relaționalul şi-a găsit naşul în persoana modelului obiectual. Statisticile indicau că în anul 2000, odată cu sfârşitul lumii, bazele de date orientate pe obiecte vor trece de 50% din piață. După cum spuneam în primul capitol, nici în prezent procentul BDOO nu ar trece de 5 sau 6 %.

N-aş vrea să înțelegeți că relaționalul nu va fi depăşit într-o zi de obiectual sau de alt model (sau post-model, cum este XML-ul). De asemenea, mărturisesc că nu am (deocamdată) contract cu modelul relațional. Afirmația la care subscriu este că modelul relațional s-a dovedit un model bun şi, din multe puncte de vedere, este încă valabil pentru gestiunea datelor în firme şi organizații, atât sub forma sa originală, cât mai ales îmbogățit cu tipuri de date definite de utilizator (vezi capitolul 19).

Modelul relațional de organizare a datelor s-a conturat în două articole publicate în 1969 şi 1970 de către E.F. Codd, matematician la centrul de cercetări din San Jose (California) al firmei IBM[1]. În acel moment, tehnologia bazelor de date era centrată pe modelele ierarhic şi rețea, modele ce depind într-o mai mare măsură de organizarea internă a datelor pe suportul de stocare (celebrii pointeri pe care i-am pomenit fără a-i arăta în primul capitol). Codd a propus o structură de date tabelară, independentă de tipul de echipamente şi software de sistem pe care

[1] Extrase din lucrarea [Codd 1970] se găsesc la adresa http://www.acm.org/classics/nov95/. De asemenea, o excelentă analiză a articolelor lui Codd este în [Date 1998].

este implementată, structură "înzestrată" cu o serie de operatori pentru extragerea datelor. Deşi puternic matematizat, modelul relaţional este relativ uşor de înţeles. Faţă de modelele ierarhice şi reţea, modelul relaţional prezintă câteva avantaje:

- propune structuri de date uşor de utilizat;
- ameliorează independenţa logică şi fizică;
- pune la dispoziţia utilizatorilor limbaje ne-procedurale;
- optimizează accesul la date;
- îmbunătăţeşte integritatea şi confidenţialitatea datelor;
- permite tratarea unei largi varietăţi de aplicaţii;
- abordează metodologic definirea structurii bazei de date (normalizarea).

Deşi puternic contestat, şi cu neajunsurile sale, modelul relaţional de organizare a bazelor de date rămâne cel mai utilizat. Cu foarte puţine excepţii, toate aplicaţiile software realizate pentru bănci, buticuri, universităţi (ordinea este, în ciuda aparenţelor, pur întâmplătoare) sunt realizate cu produse/instrumente ce gestionează baze de date relaţionale sau relaţional-obiectuale.

Un model de date are trei piloni: componenta *structurală*, adică modul în care, efectiv, la nivel logic, datele sunt stocate în bază, componenta de *integritate*, adică regulile/restricţiile ce pot fi declarate pentru datele din bază, şi o componentă *manipulatorie*, adică modul în care obţinem informaţii din bazele de date (ceea ce presupune o serie de operatori aplicabili uneia sau mai multor relaţii). Modelului relaţional îi este asociată teoria normalizării, care are ca scop prevenirea comportamentului aberant al relaţiilor în momentul actualizării lor, eliminarea datelor redundante şi înlesnirea înţelegerii legăturilor semantice dintre date[2]. Prezentul capitol va fi dedicat componentei structurale şi celei de integritate din modelul relaţional.

2.1. Structura

Modelul relaţional al datelor se poate defini printr-o serie de structuri de date (relaţii alcătuite din tupluri), operaţii aplicate asupra structurilor de date (selecţie, proiecţie, joncţiune etc.) şi reguli de integritate care să asigure consistenţa datelor (chei primare, restricţii referenţiale s.a.)[3].

---

[2] Pentru detalii privind normalizarea, vezi (şi) [Fotache2005], [Sitar-Tăut2006].

[3] Codd este cel care impune cele trei axe de discuţie ale unui model de date: structură, integritate şi manipulare. Vezi [Codd 1981].

La modul simplist, o bază de date relațională (BDR) poate fi definită ca un ansamblu de relații (tabele); fiecare tabelă (sau tabel), alcătuită din linii (tupluri), are un nume unic şi este stocată pe suport extern (de obicei disc). La intersecția unei linii cu o coloană se găseşte o valoare atomică (elementară, nedecompozabilă). O relație conține informații omogene legate de anumite entități, procese, fenomene: CĂRȚI, STUDENȚI, LOCALITĂȚI, PERSONAL, FACTURI etc. Spre exemplu, în figura 2.1 este reprezentată tabela CLIENȚI.

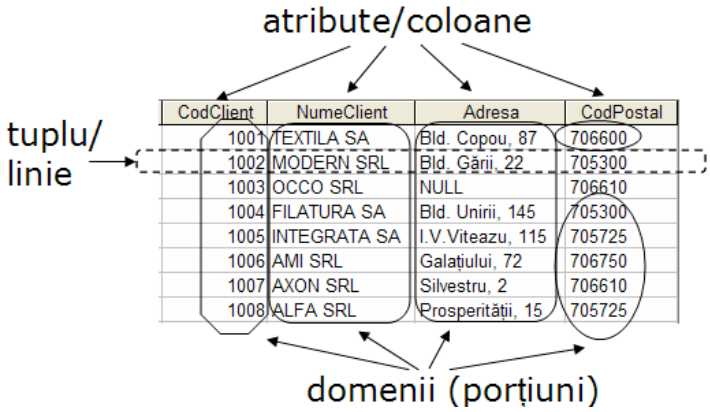

Figura 2.1. Relaţia (tabela) CLIENŢI

Teoria relațională foloseşte termenul *relație*. Practica, însă, a consacrat termenul *tabelă* (engl. *table*). Un *tuplu* sau o *linie* este o succesiune de valori de diferite tipuri. În general, o linie (re)grupează informații referitoare la un obiect, eveniment etc., altfel spus, informații referitoare la o entitate: o carte (un titlu sau un exemplar din depozitul unei biblioteci, depinde de circumstanțe), un/o student(ă), o localitate (oraş sau comună), un angajat al firmei, o factură emisă etc. Figura 2.2 conține al doilea tuplu din tabela CLIENȚI, tuplu referitor la firma MODERN SRL. Linia este alcătuită din patru valori ce desemnează: codul, numele, adresa şi codul poştal al localității şi adresei referitoare la clientul MODERN SRL.

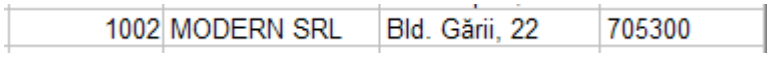

Figura 2.2. Un tuplu al tabelei CLIENŢI

Teoretic, orice tuplu reprezintă o *relație între clase de valori* (în cazul nostru, între patru clase de valori); de aici provine sintagma *baze de date relaționale*, în sensul matematic al relației, de asociere a două sau mai multe elemente. Fireşte, toate tuplurile relației au acelaşi format (structură), ceea ce înseamnă că în tabela CLIENȚI, din care a fost extrasă linia din figura 2.2, fiecare linie este constituită dintr-o valoare a codului, o valoare a numelui, o valoare a adresei, şi o valoare a codului poştal. Ordinea tuplurilor nu prezintă importanță din punctul de vedere al conținutului informațional al tabelei.

Fiecare atribut este caracterizat printr-un *nume* şi un *domeniu* de valori pe care le poate lua. Domeniul poate fi definit ca ansamblul valorilor acceptate (autorizate) pentru un element component al relației:

- într-o tabelă destinată datelor generale ale angajaților, pentru atributul Sex, domeniul este alcătuit din două valori: *Femeiesc* şi *Bărbătesc*;
- pentru atributul Regiune, domeniul, deşi limitat, este ceva mai mare; valorile autorizate pot fi: *Dobrogea*, *Banat*, *Transilvania*, *Oltenia*, *Muntenia*, *Moldova*.
- domeniul atributului Jud este alcătuit din indicativele auto (două caractere) ale fiecărui județ (plus Bucureşti).
- domeniul atributului Judet este alcătuit din numele fiecărui județ (plus Bucureşti).
- domeniul unui atribut precum PrețUnitar, care se referă la prețul la care a fost vândut un produs/serviciu, este cu mult mai larg, fiind alcătuit din orice valoarea cuprinsă între 1 şi 999999.99[4] lei (ceva mai noi).

Firește, în funcție de specificul bazei de date, domeniul poate fi extins sau restrâns după cerințe. De exemplu, la regiunile luate în discuție mai sus, mai pot fi adăugate şi provincii precum: Bucovina, Maramureş, Crişana.

Şi acum, câteva senzații tari (urmează un pic de matematică) ! Dacă notăm cu D_1 domeniul atributului CodClient, cu D_2 domeniul atributului NumeClient, cu D_3 domeniul pentru Adresa, şi cu D_4 domeniul atributul CodPostal, se poate spune că fiecare linie a tabelei CLIENȚI este un tuplu de patru elemente,

$$(v_1, v_2, v_3, v_4) \text{ unde } v_1 \in D_1, v_2 \in D_2, v_3 \in D_3, v_4 \in D_4$$

iar relația în ansamblu corespunde unui subansamblu din ansamblul tuturor tuplurilor posibile alcătuite din patru elemente, ansamblu care este produsul cartezian al celor patru domenii.

$$\bigotimes_{i=1}^{4} D_i$$

Generalizând, orice relație R poate fi definită ca un subansamblu al produsului cartezian de n domenii D_i:

[4] Punctul indică marca, zecimală, cei doi de nouă ce urmează punctului fiind asociați banilor (subdiviziunii leului).

$$R \subseteq D_1 \times D_2 \times D_3 \times \ldots \times D_n ,$$

n fiind *gradul* sau *ordinul* relației. Relațiile de grad 1 sunt unare, cele de grad 2 - binare, ..., cele de grad n - n-are. Această definiție pune în evidență aspectul constant al relației, de independență în timp (se spune că în acest caz relația este definită ca un predicat).

O a doua definiție abordează o relație R ca un ansamblu de m-uplete (m-tupluri) de valori:

$$R = \{ t_1, t_2, \ldots, t_k, \ldots, t_m \}, \quad \text{unde } t_k = (d_{k1}, d_{k2}, \ldots, d_{ki}, \ldots, d_{kn})$$

în care:

d_{k1} este o valoare în D_1, d_{k2} este o valoare în D_2, … , d_{kn} este o valoare în D_n;

n - reprezintă *ordinul* lui R;

m - cardinalitatea lui R.

Pentru un (mic) plus de claritate, vezi figura 2.3. Reprezentarea sub formă de tabelă, deci ca ansamblu de tupluri, pune în evidență aspectul dinamic, variabil al relației. Abordarea predicativă (prima definiție) sau ansamblistă (a doua definiție) reprezintă un criteriu important de delimitare a limbajelor de interogare a bazelor de date relaționale.

R	D_1	D_2	D_3		D_i		D_n
t_1	d_{11}	d_{12}	d_{13}	. . .	d_{1i}	. . .	d_{1n}
t_2	d_{21}	d_{22}	d_{23}	. . .	d_{2i}	. . .	d_{2n}
t_3	d_{31}	d_{32}	d_{33}	. . .	d_{3i}	. . .	d_{3n}
	. . .	. . .	. . .		. . .		. . .
t_k	d_{k1}	d_{k2}	d_{k3}	. . .	d_{ki}	...	d_{kn}
	. . .	. . .	. . .		. . .		. . .
t_m	d_{m1}	d_{m2}	d_{m3}	. . .	d_{mi}	...	d_{mn}

Figura 2.3. Ilustrarea celei de-a doua definiţii a unei relaţii

Având în vedere cele două definiții ale relației, se poate reformula conceptul de atribut, ca fiind elementul care determină ansamblul de valori luate de fiecare

domeniu al relației[5]. Precizarea fără ambiguitate a rolului jucat de fiecare domeniu în cadrul relației impune ca numele atributelor să fie diferite. O relație poate fi simbolizată şi astfel:

$$R(A_1, A_2, \ldots, A_n) \text{ sau } (A_1 : D_1, A_2 : D_2, \ldots, A_n : D_n)$$

unde A_i sunt atributele relație.

Reținem corespondența noțiunilor *relație-tabelă*, *tuplu-linie* şi *atribut-coloană*.

Numărul de tabele, atributele "adunate" în fiecare tabelă, domeniul fiecăruia dintre atribute prezintă diferențe majore de la o bază la alta, chiar dacă uneori reflectă acelaşi tip de procese. Intrăm astfel în sfera proiectării bazelor de date, a dependențelor şi normalizării.

Relația CLIENȚI conține informații despre firmele cărora compania noastră le vinde produsele pe care le producem şi/sau comercializăm. Fiecare linie se referă la un singur client. În figura 2.1 pe a treia linie a tabelei apare o valoarea curioasă notată NULL. Valoarea NULL este considerată o *metavaloare* şi indică faptul că, în acel loc, informația este *necunoscută*, *inexistentă* sau *inaplicabilă* (în acest caz nu cunoaştem adresa clientului Modern SRL), fiind valabilă indiferent de tipul atributului (numeric, şir de caractere, date calendaristice). Valoarea NULL este diferită, însă de valorile 0 sau spațiu. Uneori, importanța sa este (din păcate) majoră în expresii şi funcții, după cum ştiu cei cu oarecare experiență în limbajul SQL.

Principalele caracteristici ale unei relații pot fi, astfel, sistematizate după cum urmează:

- În cadrul unei baze de date, o relație prezintă un nume distinct de al celorlalte relații.
- Valoarea unui atribut într-un tuplu oarecare este simplă (atomică sau elementară).
- Fiecare atribut are un nume distinct.
- Orice valoare a unui atribut face parte din domeniul pe care a fost definit acesta.
- Ordinea dispunerii atributelor în relație nu prezintă importanță.
- Fiecare tuplu este distinct, adică nu pot exista două tupluri identice într-o aceeaşi relație.

---

[5] [Saleh 1994], p.29

- Ordinea tuplurilor nu influenţează conţinutul informaţional al relaţiei.

Structura unei baze de date mai cuprinde restricţii şi alte obiecte discutate în paragrafele 2.2 şi 2.3.

2.2. Restricţii

De ce ne interesează restricţiile într-o bază de date ? Termenul de *restricţie* (*regulă* sau *constrângere*) este oarecum iritant, atât pentru studenţi, cât şi pentru profesori, deoarece semnalează existenţa unor constrângeri instituite şi oarecum obligatorii, şi, de vreme ce sunt impuse, însemnă că nu sunt prea plăcute (decât, în cel mai bun caz, pentru cel care le-a instituit). Partea cea mai enervantă este că respectarea restricţiilor este (supra)vegheată de o anumită autoritate înzestrată cu anumite instrumente de constrângere, de la bastoane de cauciuc, la creşterea şi scăderea impozitelor, salariilor, banilor de buzunar etc.

Ei bine, în bazele de date, regulile sunt ceva mai acceptabile. Cei care lucrează cu bazele de date sunt foarte interesaţi în declararea restricţiilor, pentru că, odată definite, de respectarea lor se va îngriji SGBD-ul. Esenţial este că, ajutaţi de reguli, putem creşte gradul de corectitudine şi de încredere pentru datele din bază. În cele ce urmează vor fi prezentate, pe scurt, cele mai importante restricţii definibile într-o BD relaţională: de domeniu, de atomicitate, de unicitate, referenţiale şi restricţiile-utilizator.

2.2.1. Restricţia de domeniu

După cum am văzut în paragraful anterior, un atribut este definit printr-un nume şi un domeniu. Orice valoare a atributului trebuie să se încadreze în domeniul definit. Există mai multe moduri de percepţie a acestei restricţii.

O parte dintre informaticieni substituie domeniul *tipului* atributului: numeric, şir de caractere, dată calendaristică, logic (boolean) etc. şi, eventual, lungimii (numărul maxim de poziţii/caractere pe care se poate "întinde" un atribut). După cum se observă, este luat în calcul numai aspectul sintactic al domeniului. Faptul că indicativul auto al unui judeţ (vezi plăcuţele de înmatriculare) poate fi una din valorile: IS, TM, B etc. reprezintă o restricţie de comportament sau, mai simplu, o restricţie definită de utilizator.

Cea de-a doua categorie priveşte domeniul deopotrivă *sintactic* şi *semantic*. Astfel, domeniul sintactic al atributului Jud (indicativul judeţului) este un şir de două caractere, obligatoriu litere (sau o literă şi un spaţiu, pentru Bucureşti), şi, chiar mai restrictiv, literele sunt obligatoriu majuscule. Din punct de vedere semantic, indicativul poate lua una din valorile: IS, TM …

Majoritatea SGBD-urilor permit definirea tuturor elementelor ce caracterizează domeniul (sintactic şi semantic) atributului Jud prin declararea tipului şi lungimii atributului şi prin aşa-numitele reguli de validare la nivel de câmp (field validation rule). Sunt însă şi produse la care domeniul poate fi definit explicit, sintactic şi

semantic, dându-i-se un nume pe care vor fi definite atributele în momentul creării tabelelor.

2.2.2. Atomicitate

Conform teoriei bazelor de date relaționale, orice atribut al unei tabele oarecare trebuie să fie atomic, în sensul imposibilității descompunerii sale în alte atribute. Implicit, toate domeniile unei baze de date sunt musai atomice (adică elementare). În aceste condiții, se spune că baza de date se află în *prima formă normală* sau prima formă normalizată (1NF).

Astăzi, atomicitatea valorii atributelor a devenit o țintă predilectă a "atacurilor dușmănoase" la adresa modelului relațional, pe motiv de imposibilitate a gestionării unor structuri de date mai complexe, specifice unor domenii ca proiectare asistată de calculator, baze de date multimedia etc. Mulți autori, dintre care merită amintiți cu deosebire Chris J. Date şi Hugh Darwen, se opun ideii de atomicitate formulată de Codd.

La drept vorbind, singurul lucru cert despre atomicitate este relativitatea acesteia. Să reluăm un exemplu de atribut compus (non-atomic) - Adresa[6]. Fiind alcătuită din Stradă, Număr, Bloc, Scară, Etaj, Apartament, discuția despre atomicitatea adresei pare de prisos, iar descompunerea sa imperativă. Trebuie însă să ne raportăm la obiectivele bazei. Fără îndoială că pentru BD a unei filiale CONEL sau ROMTELECOM, sau pentru poliție, preluarea separată a fiecărui element constituent al adresei este foarte importantă. Pentru un importator direct, însă, pentru un mare en-grossist sau pentru o firmă de producție lucrurile stau într-o cu totul altă lumină. Partenerii de afaceri ai acestora sunt persoane juridice, iar adresa interesează numai la nivel general, caz în care atributul Adresa nu este considerat a fi non-atomic.

Alte exemple de atribute ce pot fi considerate, în funcție de circumstanțe, simple sau compuse: DataOperațiuniiBancare (Data + Ora), BuletinIdentitate (Serie + Număr), NrînmatriculareAuto (privit global, sau pe cele trei componente: număr, județ, combinație trei de litere).

O relație (tabelă) în 1NF nu trebuie să conțină atribute care se repetă ca grupuri (grupuri repetitive). Într-o altă formulare, toate liniile unei tabele trebuie să conțină acelaşi număr de atribute. Fiecare celulă a tabelei (intersecția unei coloane cu o linie), altfel spus, valoarea unui atribut pe o linie (înregistrare), trebuie să fie

[6] Este un exemplu dintre cele mai frecvente în majoritatea cărților despre baze de date.

atomică. Detalii despre atomicitate şi grupuri repetitive găsiţi (şi) în [Fotache 2005] şi [Sitar-Tăut 2006].

2.2.3. Nenulitate

Modelul relaţional prevede ca, atunci când nu se cunoaşte valoarea unui atribut pentru o anumite entitate, sau când pentru acel obiect, entitate, persoană etc. atributul este inaplicabil, să se folosească (meta)valoarea NULL. După cum am discutat, celui de-al treilea client din figura 2.1 nu i se cunoaşte adresa. Dacă am avea o tabelă PRODUSE cu atributele CodProdus, DenumireProdus, UM, Culoare, este posibil ca, pentru anumite sortimente, cum ar tricouri, cămăşi, jachete, atributul să fie important, în timp ce pentru altele, precum vodcă, cafea, lapte atributul Culoare să nu furnizeze nici o informaţie, adică să nu fie aplicabil. Iar dacă laptele poate fi doar alb, vodca chiar că nu are culoare (sunt foarte mulţi specialişti în acest domeniu, îi puteţi „interoga" !).

Într-o bază de date relaţională avem posibilitatea de a le impune unora dintre atribute (sau tuturora) să aibă întotdeauna valori specificate, altfel spus, le interzicem valorile nule, în timp ce altor atribute li se pot permite valori nule. Ca regulă, atributele importante, ce ţin de identificarea sau caracterizarea unei entităţi, proces, fenomen, precum şi cele implicitate în calculul unor informaţii importante, sunt declarate NOT NULL, iar atributele fără importanţă deosebită pot fi mai „relaxate".

2.2.4. Unicitate

O relaţie nu poate conţine două linii identice (două linii care prezintă aceleaşi valori pentru toate atributele). Mai mult, majoritatea relaţiilor prezintă un atribut, sau o combinaţie de atribute, care diferenţiază cu siguranţă un tuplu de toate celelalte tupluri ale relaţiei. *Cheia primară* a unei relaţii (tabele) este un atribut sau un grup de atribute care identifică fără ambiguitate fiecare tuplu (linie) al relaţiei (tabelei). După Codd, există trei restricţii pe care trebuie să le verifice cheia primară:

- *unicitate*: o cheie identifică un singur tuplu (linie) al relaţiei;
- *compoziţie minimală*: atunci când cheia primară este compusă, nici un atribut din cheie nu poate fi eliminat fără distrugerea unicităţii tuplului în cadrul relaţiei; în cazuri limită, o cheie poate fi alcătuită din toate atributele relaţiei;

- *valori non-nule*: valorile atributului (sau ale ansamblului de atribute) ce desemnează cheia primară sunt întotdeauna specificate, deci ne-nule şi, mai mult, nici un atribut din compoziția cheii primare nu poate avea valori nule; această a treia condiție se mai numeşte şi *restricție a entității*.

Joe Celko inventariază patru proprietăți dezirabile pentru o cheie[7]:

- Familiaritate: valorile cheii să fie uşor de înțeles pentru utilizatori.
- Stabilitate: valorile cheii nu trebuie să fie volatile.
- Minimalitate.
- Simplitate: sunt de preferat chei scurte şi simple.

Tot el atrage atenția ca cele patru proprietăți pot intra, uneori, în conflict. O cheie stabilă poate să se dovedească complexă şi dificil de gestionat. Identificarea cheii primare pentru o relație face parte (împreună cu stabilirea tabelelor prin gruparea atributelor, stabilirea celorlalte restricții) din procesul de proiectare a bazei de date. Uneori, precum în cazul tabelei JUDEȚE, stabilirea cheii primare nu ridică probleme. Fiecare județ are un indicativ auto unic, deci atributul Jud candidează la postul de cheie a relației. La fel se poate spune şi despre denumirea județului. Rezultă că relația JUDEȚE prezintă două chei candidate: Jud şi Judet. Alegerea unei dintre cheile candidat are în vedere criterii precum lungimea, uşurința în reținere şi/sau editare. Fiind mai scurt, desemnăm atributul Jud drept cheie primară, situație în care Judet devine cheie alternativă.

Tabela CLIENȚI are cheia primară simplă – CodClient. CodClient reprezintă un număr unic asociat fiecărei firme căreia i-am făcut vânzări (clienții noştri sunt exclusiv persoane juridice, adică firme). Există însă suficiente cazuri în care cheia primară este compusă din două, trei s.a.m.d. sau, la extrem, toate atributele relației. Să luăm spre analiză o relație PERSONAL care conține date generale despre angajații firmei. Fiecare tuplu al relației se referă la un angajat, atributele fiind: Nume, Prenume, DataNaşterii, Vechime, SalariuTarifar.

- Atributul Nume nu poate fi cheie, deoarece chiar şi într-o întreprindere de talie mijlocie, este posibil să existe doi angajați cu acelaşi nume.
- Dacă apariția a două persoane cu nume identice este posibilă, atunci apariția a două persoane cu acelaşi Prenume este probabilă.
- Nici unul din aceste atributele DataNaşterii, Vechime, SalariuTarifar nu poate fi "înzestrat" cu funcțiunea de identificator.

În acest caz, se încearcă gruparea a două, trei, patru s.a.m.d. atribute, până când se obține combinația care va permite diferențierea clară a oricărei linii de toate

7 [Celko1999], p.247

celelalte. Combinaţia Nume+Adresă pare, la primele două vederi, a îndeplini "cerinţele" de identificator. Ar fi totuşi o problemă: dacă în aceeaşi firmă (organizaţie) lucrează împreună soţul şi soţia ? Ambii au, de obicei, acelaşi nume de familie şi, tot de obicei, acelaşi domiciliu. Este adevărat, cazul ales nu este prea fericit. Dar este suficient pentru a "compromiterea" combinaţiei.

Următoarea tentativă este grupul Nume+Prenume+Adresă, combinaţie neoperantă dacă în organizaţie lucrează tatăl şi un fiu (sau mama şi o fiică) care au aceleaşi nume şi prenume şi domiciliul comun. Ar rămâne de ales una dintre soluţiile (Nume+Prenume+Adresă+Vechime) sau (Nume+Prenume+Adresa+DataNaşterii).

Oricare din cele două combinaţii prezintă riscul violării restricţiei de entitate, deoarece este posibil ca, la preluarea unui angajaţi în bază, să nu i se cunoască adresa sau data naşterii, caz în care atributul respectiv ar avea valoarea NULL. Dificultăţile de identificare fără ambiguitate a angajaţilor au determinat firmele ca, la angajare, să aloce fiecărei persoane un număr unic, număr denumit Marcă. Prin adăugarea acestui atribut la cele existente, pentru relaţia PERSONAL problema cheii primare este rezolvată mult mai simplu. Actualmente, pentru cetăţenii români sarcina este simplificată şi prin utilizarea codului numeric personal (CNP), combinaţie de 13 cifre care prezintă avantajul că rămâne neschimbată pe tot parcursul vieţii persoanei.

Problema unicităţii a suscitat o serie de discuţii în modelul relaţional. Clasicii modelului, Codd şi Date, s-au pronunţat împotriva repetării tuplurilor în cadrul unei relaţii, cerinţă pe care standadele SQL nu au luat-o în considerare. Există şi situaţii reale în care repetarea liniilor este inevitabilă. David Beech a prezentat într-o lucrare înaintată Consiliului de Standardizare a bazelor de date ANSI X3H2, şi mai apoi în revista Datamation[8], problema consacrată în literatura de specialitate drept "argumentul cutiilor cu mâncare pentru pisici"[9]: într-un supermagazin în care casele de marcat sunt legate la o bază de date, un client cumpără patru cutii Whiskas[10] pentru pisici care costă, fiecare (cutie), 10 lei; pe bon apare de patru ori "cutii Whiskas". Autorul afirmă, pe bună dreptate, că la nivelul conceptual la care se înregistrează informaţiile, nu există nici o valoare prin care cele patru rânduri să fie diferenţiate. Astfel încât, pentru a respecta canoanele modelului relaţional, este necesară introducerea unei coloane suplimentare fără prea mare relevanţă, dar care

8 Beech, D. – *New Life for SQL*, Datamation, Febr. 1, 1989

9 [Celko1999], p.45

10 Am rugămintea de a nu vă lăsa manipulaţi de reclame ieftine (ci numai de cele scumpe) !

să asigure unicitatea fiecărui tuplu. În acest sens, E.F. Codd a propus consiliului ANSI X3H2 o funcție specială care să măsoare gradul de repetabilitate a valorii unuia sau mai multor atribute, funcție care ar semăna întrucâtva cu funcția COUNT() din SQL folosită împreună cu GROUP BY[11].

Pe lângă noțiunile cheie-candidat, cheie primară, cheie alternativă, modelul relațional utilizează şi sintagma *supercheie*. O supercheie poate fi definită ca un set de coloane ce îndeplineşte condiția de cheie (unicitate), dar care conține cel puțin un subset care este, el-însuşi, cheie[12]. Cu alte cuvinte, din cele trei condiții ale cheii primare, o supercheie o îndeplineşte numai pe prima (unicitate), fără a avea însă compoziția minimală şi fără a pune problema restricției de entitate (valorile nule ale atributelor componente).

Domeniul unui atribut care este cheie primară într-o relație este denumit domeniu primar. Dacă într-o relație există mai multe combinații de atribute care conferă unicitate tuplului, acestea sunt denumite chei candidate. O cheie candidată care nu este identificator primar este referită drept *cheie alternativă*.

2.2.5. Restricția referențială

O bază de date relațională este alcătuită din relații (tabele) aflate în legătură. Stabilirea legăturii se bazează pe mecanismul cheii *străine* (*externe*) şi, implicit, a restricției referențiale. Figura 2.4 prezintă o legătură în care sunt implicate tabelele FACTURI şi CLIENȚI. Atributul CodClient joacă un rol de "agent de legătură" între tabelele CLIENȚI şi FACTURI. Pentru relația CLIENȚI, atributul CodClient este cheie *primară*, în timp ce în tabela FACTURI, CodClient reprezintă *coloana de referință* sau *cheia străină (externă)*, deoarece numai pe baza valorilor sale se poate face legătura cu relația părinte CLIENȚI.

Cheile străine, denumite şi *chei externe* sau *coloane de referință*, sunt deci atribute sau combinații de atribute care pun în legătură linii (tupluri) din relații diferite. Tabela în care atributul de legătură este cheie primară se numeşte *tabelă-părinte* (în cazul nostru, CLIENȚI), iar cealaltă *tabelă-copil*.

Legat de noțiunea de cheie străină apare conceptul de *restricție referențială*. O restricție de integritate referențială apare atunci când o relație face referință la o altă relație. C.J.Date defineşte restricția de integritate referențială astfel[13]: Fie D un

11 Vezi [Celko1999], p.49

12 [Celko1999], p.52

13 [Date1986], p.89

domeniu primar şi R1 o relaţie ce prezintă un atribut A definit pe D - notate R1 (..., A:D, ...). În orice moment, orice valoare a lui A în R1 trebuie să fie:

a. ori nulă,

b. ori egală cu o valoare a lui V, unde V este cheie primară sau candidată într-un tuplu al unei relaţii oarecare R2 (R1 şi R2 nu trebuie să fie neapărat distincte), cheie primară definită pe acelaşi domeniu primar D.

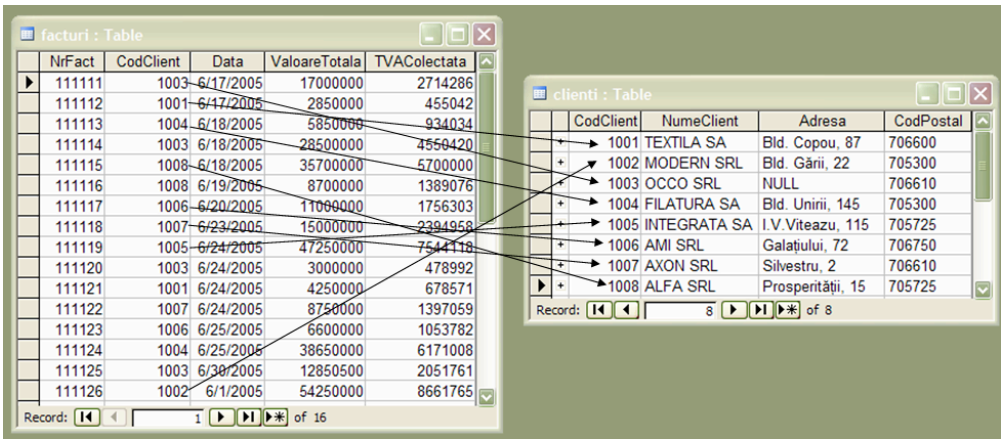

Figura 2.4. O restricţie referenţială între FACTURI (tabelă-copil) şi CLIENŢI (părinte)

Într-o altă formulare, se iau în discuţie două tabele (relaţii) T1 şi T2. Ambele au atributul sau grupul de atribute notat CH care pentru T1 este cheie primară, iar pentru T2 cheie străină. Dacă în T2 se interzice apariţia de valori nenule ale CH care nu există în nici un tuplu din T1, se spune că între T2 şi T1 s-a instituit o restricţie referenţială.

Instituirea restricţiei referenţiale între tabela CLIENŢI (părinte) şi FACTURI (copil) permite cunoaşterea, pentru fiecare factură, a denumirii şi adresei clientului căreia i-a fost adresată. Dacă în FACTURI ar exista vreo linie în care valoarea atributului CodClient ar fi, spre exemplu 9988, este clar că acea linie ar fi orfană (nu ar avea linie corespondentă în tabela părinte CLIENŢI).

Observaţii

- Pentru mulţi utilizatori şi profesionişti ai bazelor de date, denumirea de "relaţional" desemnează faptul că o bază de date este alcătuită din tabele puse în legătură (relaţie) prin intermediul cheilor externe şi primare. Aceasta este, de fapt, a doua accepţiune a termenului de *baze de date relaţionale* (şi cea eronată), prima, cea "clasică", având în vedere percepţia fiecărei linii dintr-o tabelă ca o relaţie între clase de valori.
- Majoritatea SGBD-urilor prezintă mecanisme de declarare şi gestionare automată a restricţiilor referenţiale, prin actualizări în cascadă şi interzicerea valorilor care ar încălca aceste restricţii.
- Respectarea restricţiilor referenţiale este una din cele mai complicate sarcini pentru dezvoltatorii de aplicaţii ce utilizează baze de date. Din

acest punct de vedere, tentația este de a "sparge" baza de date în cât mai puține tabele cu putință, altfel spus, de a avea relații cât mai "corpolente". Gradul de fragmentare al bazei ține de *normalizarea* bazei de date, care, ca parte a procesului de proiectare a BD, se bazează pe dependențele funcționale, multivaloare şi de joncțiune între atribute pe care, din fericire, nu le atacăm în această carte.

2.2.6. Restricții-utilizator

Restricțiile utilizator mai sunt denumite şi *restricții (reguli) de comportament* sau *restricții (reguli) ale organizației*. De obicei, iau forma unor *reguli la nivel de atribut* (field validation rules), *reguli la nivel de linie* (record validation rules), sau reguli complexe, inter-atribute/linii/tabele, implementabile prin *aserțiuni* (assertions), *reguli* (rules) sau *declanşatoare* (triggers).

Reguli la nivel de atribut

O restricție la nivel de atribut poate preveni introducerea în baza de date a unor valori din alte intervale decât cele autorizate, în alte formate decât cele permise etc. O astfel de regulă înlocuieşte (sau completează, mai bine zis) o restricție de domeniu. Forma clasică a unei restricții la nivel de atribut este o expresie în care apar constante, funcții-sistem şi, nu în ultimul rând, atributul respectiv. De obicei, expresia conține, pe lângă constante şi funcții sistem (nici funcțiile sistem nu sunt tolerate întotdeauna), numai atributul pentru care se defineşte restricția. De exemplu, pentru atributul CodCl din tabela CLIENȚI, se poate declara restricția: *CodCl BETWEEN 1001 AND 5678*. Sau, în tabela STUDENȚI, există atributul CicluStudii, care, în condițiile (oarecum) noului sistem universitar european[14] (Bologna), poate avea valorile 1, 2 sau 3, iar regula poate fi formulată în mai multe variante:

- *CicluStudii > 0 AND CicluStudii < 4*
- *CicluStudii >= 1 AND CicluStudii <= 3*
- *CicluStudii BETWEEN 1 AND 3*
- *CicluStudii IN (1, 2, 3)*

Numărul anilor de studii pe fiecare ciclu este însă uşor diferit. Ciclurile 1 (licență) şi 3 (doctorat) se întind pe durata a trei ani[15], în timp ce al doilea ciclu

[14] Afirmația este discutabilă, ca şi noul "sistem", dealtminteri.

[15] Regula se aplică în universitațile "clasice", universitățile de medicină şi politehnice având o altă structură.

(specializare & master) durează doi ani. Aşa că, pentru AnStudii regula este similară atributului CicluStudii:

- *AnStudii >= 1 AND AnStudii <= 3*
- *AnStudii BETWEEN 1 AND 3*
- *AnStudii IN (1, 2, 3)*

Dacă, însă, dorim să fim mai exacţi şi să ţinem cont că valoarea maximă a anului de studii pentru al doilea ciclu este 2, putem imagina o regulă mai complexă: *CASE WHEN CicluStudii IN (1,3) THEN AnStudii BETWEEN 1 AND 3 ELSE AnStudii BETWEEN 1 AND 2 END*. Ei bine, această regulă (expresie) conţine două atribute, nefiind una la nivel de atribut, ci *la nivel de înregistrare*.

La orice editare a atributului cu pricina (declanşată la inserarea unei linii în tabela din care face parte, sau la modificarea valorii sale), expresia este evaluată şi, dacă rezultatul evaluării are valoarea logică adevărat (TRUE), atunci inserarea/modificarea este permisă, iar dacă rezultatul este FALSE, atunci inserarea/modificarea este blocată. Astfel, dacă se inserează o înregistrare în tabela STUDENŢI, înregistrare ce corespunde unui student proaspăt înmatriculat (în urma admiterii sau transferului), iar valoarea specificată pentru atributul CicluStudii este 4, atunci expresia *CicluStudii BETWEEN 1 AND 3* are valoarea logică *FALS*, iar SGBD-ul nu va tolera inserarea. Ulterior, o altă comandă de inserare în care valoarea atributului ar fi cuprinsă între 1 şi 3, ar putea fi acceptată, în caz că nicio altă restricţie nu ar fi încălcată.

Reguli la nivel de înregistrare

După cum am văzut mai sus, expresia care defineşte o restricţie la nivel de înregistrare conţine două sau mai multe atribute: *CASE WHEN CicluStudii IN (1,3) THEN CASE WHEN AnStudii BETWEEN 1 AND 3 THEN TRUE ELSE FALSE END ELSE CASE WHEN AnStudii BETWEEN 1 AND 2 THEN TRUE ELSE FALSE END END*[16]. Restricţia la nivel de înregistrare este evaluată la inserarea unei linii sau la modificarea valorii a măcar unui atribut într-o înregistrare din tabelă. Dacă, teoretic (şi, uneori, practic) restricţia unui atribut se verifică imediat după modificarea valorii acestuia, restricţia la nivel de înregistrare se verifică după modificarea tuturor atributelor de pe linia respectivă. În practică, este greu de delimitat aceste momente, pentru că deseori se produc aproape simultan.

---

16 Este o altă variantă (care se vrea mai impresionantă) a restricţiei prin care anul de studii se sincronizează cu ciclul de studii.

Tabela FACTURI ar putea conține, printre altele, două atribute "valorice", unul pentru păstrarea valorii totale (inclusiv TVA) a facturii respective, şi un altul care stochează cuantumul TVA colectate pentru factură. Majoritatea produselor şi serviciilor comercializate la noi în țară au un procent al taxei pe valoarea adăugată de 19%. Există, însă, şi produse la care procentul poate fi 9% sau chiar 0%) De aceea, TVA colectată pentru o factură este mai mică sau cel mult egală decât/cu 19% din valoarea fără tva. Să încercăm o formulă: *ValoareaTotală = ValoareaFărăTVA + TVA*. Dacă factura conține numai produse cu 19%: *ValoareaTotală = ValoareaFărăTVA + 0.19 * ValoareaFărăTVA*, sau *ValoareaTotală = 1.19 * ValoareaFărăTVA*. Dar tabela noastră are doar atributele ValoareTotală şi TVAColectată, aşa că înlocuim ValoareaFărăTVA prin diferența celorlalte două. Scriem: ValoareTotală *= 1.19 * (ValoareTotală – TVAColectată)*, şi după calcule de matematică superioară, *TVAColectată * 1.19 = 0.19 * ValoareTotală*, altfel spus *TVAColectată = ValoareTotală * 0.19 / 1.19*[17]. Prin urmare, expresia restricției la nivel de înregistrare este: *TVAColectată BETWEEN 0 AND ValoareTotală * 0.19 / 1.19*[18].

Să ne imaginăm şi alte asemenea restricții:

O tabelă INCASĂRI (vezi paragraful 2.3.2), ar putea conține un atribut Dataîncasării, care să indice ziua în care ne-au intrat banii în cont (sau casierie) şi un altul, DatăDocument. Ideea este că un client ne poate plăti o sumă printr-un ordin de plată pe care îl întocmeşte pe 9 septembrie 2008 (aceasta este valoarea atributului DatăDocument), iar banii să ajungă în contul nostru bancar pe 12 septembrie 2008 (aceasta este valoarea atributului Dataîncasării). Regula s-ar referi la precedența celor două valori: *CASE WHEN DataIncasarii IS NOT NULL THEN Dataîncasării >= DatăDocument END*.

Într-o altă tabelă – GĂRZI – din baza de date a unui spital de urgență, ar putea exista un atribut pentru evidențierea momentului începutului (DatăOrăîncputGardă) şi sfârşitului (DatăOrăSfârşitGardă) unei gărzi. Restricția la nivel de înregistrare ar privi tot precedența celor două valori: *DatăOrăîncputGardă < DatăOrăSfârşitGardă*.

Cele două tipuri de restricții pot lucra în tandem. Astfel, pentru tabela STUDENȚI putem declara:

- O restricție pentru atributul CicluStudii: *CicluStudii IN (1, 2, 3)*

[17] Această expresie reprezintă chintesența desăvârşirii mele matematice.

[18] În economia lucrării, după acest paragraf, pentru autor a urmat o pauză de jumătate de zi datorată epuizării intelectuale.

- O restricție pentru atributul AnStudii: *AnStudii IN (1, 2, 3)*
- O regulă la nivel de înregistrare care ar avea o expresie mult simplicată: *CASE WHEN CicluStudii = 2 THEN AnStudii <= 2 END*.

Reguli complexe

Există şi alte tipuri de restricții a căror validare presupune verificarea valorilor (unuia sau mai multor atribute) de pe două sau mai multe linii (înregistrări) ale unei aceleaşi tabele, sau chiar aflate în tabele diferite. Iată câteva exemple mai mult sau mai puțin realiste:

- o factură poate avea maximum 30 de linii (produse vândute);
- se interzice emiterea unei noi facturi (o nouă vânzare) dacă datoriile firmei client sunt mai mari de 20000 lei (iar directorul acesteia nu este membru în partidul/partidele de guvernământ);
- pentru o bibliotecă universitară, numărul maxim de cărți împrumutate, la orice moment dat, depinde de categoria cititorului: 5 pentru studenții din ciclul 1, 7 pentru studenții din ciclurile 2 şi 3, şi 10 pentru profesori.

Modalitățile prin care aceste restricții pot fi declarate/implementate sunt destul de eterogene în standardele SQL şi în practică (SGBD-uri): aserțiuni, reguli şi, probabil, cele mai frecvente, declanşatoarele (triggerele) pe care le vom discuta în detaliu mai către finalul cărții.

2.3. Schema şi conținutul unei baze de date relaționale

Aşa cum era prezentat în primul capitol, există două aspecte complementare de abordare a bazelor de date: *schema* (structura, intensia) şi *conținutul* (instanțierea, extensia). Pentru o bază de date relațională (BDR) *conținutul* unei relații este reprezentat de ansamblul tuplurilor care o alcătuiesc la un moment dat. *Schema* unei baze de date conține denumiri ale tabelelor, numele, tipul şi lungimea atributelor, restricții de unicitate, de non-nulitate, restricții la nivel de atribut şi de linie, şi alte eventuale tipuri de restricții de comportament, precum şi restricții referențiale. La acestea se adaugă drepturile utilizatorilor, definițiile şi restricțiile tabelelor virtuale, indecşilor etc. Foarte importante în lumea profesioniştilor dezvoltării de aplicații cu bazele de date sunt procedurile stocate – programe de forma funcțiilor, procedurilor, pachetelor şi declanşatoarelor (triggerelor) – care, după cum le spune şi numele, sunt memorate în schema bazei de date şi fac parte integrantă din aceasta. Mai sunt şi alte ingrediente ale unei scheme de BDR, însă le vom prezenta pe parcurs.

2.3.1. Reprezentarea schemei unei baze de date relaționale

Schema unei BDR poate fi (şi chiar trebuie) reprezentată grafic. Există o mulțime de reprezentări, mai sugestive sau mai criptice, fiind greu de făcut o clasificare din punctul de vedere al lizibilității/comprehensibilității lor. Oricum, ierahizarea ar fi, din start, contaminată de preferințele şi experiența autorului.

Prima ediție a cărții a recurs la un formalism „neutru”, bazat pe următoarele reguli simpliste:[19]:

- O tabelă se reprezintă pe două linii, pe prima fiind scris numai numele relației iar pe cea de-a doua numele atributelor; ordinea prezentării atributelor nu are importanță, însă, pentru lizibilitate, coloana sau coloanele care desemnează cheia primară se dispun succesiv, la început (stânga); tot alăturat se dispun şi atributele care constituie o cheie străină.
- Numele coloanelor ce alcătuiesc cheia primară se subliniază cu o linie continuă, iar cele care alcătuiesc cheile alternative se subliniază cu linie punctată.
- Numele coloanelor facultative (atributele ce pot avea valori NULL) sunt scrise între paranteze.
- Dacă o cheie străină este alcătuită din mai multe coloane, se poate utiliza acolada pentru a le grupa.
- O restricție referențială este reprezentată printr-o săgeată care pleacă din cheia externă (sau de la vârful acoladei, în cazul unei chei străine compuse) şi are vârful înfipt în cheia primară a tabelei părinte.

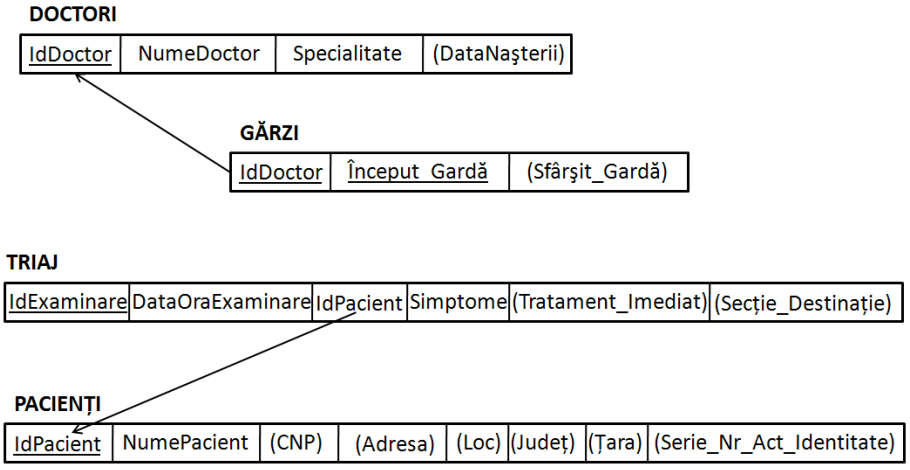

Figura 2.5. Una dintre reprezentările BD TRIAJ

[19] Preluare din [Hainaut 1994], pp. 24-25

Astfel, o bază de date destul de simplistă dedicată unui spital de urgență, BD denumită TRIAJ ar putea avea schema reprezentată ca în figura 2.5. Faptul că există o tabelă cu acelaşi nume ca al bazei de date nu e grozav, dar nici toxic.

Un formalism frecvent printre non-profesionişti este cel din Access – vezi figura 2.6. Principalul avantaj este că, datorită modului de dispunere a atributelor în tabele, reprezentarea relațiilor cu multe atribute nu ridică probleme precum formalismul anterior. Există, însă, şi câteva dezavantaje majore:

- nu se pot vizualiza cheile alternative (alte atribute/grupuri de atribute cu valori unice în tabela respectivă);
- nu se pot deosebi atributele cu valori obligatorii (NOT NULL) de cele pentru care se pot accepta valori nule;
- restricțiile referențiale trebuie ghicite după numele atributelor, întrucât linia care leagă tabela copil de tabela părinte nu este trasată (decât accidental) în dreptul atributelor cheie externă şi cheie primară;
- pentru Access, conectarea unei tabele părinte cu una copil nu înseamnă automat şi că restricția referențială a fost instituită.

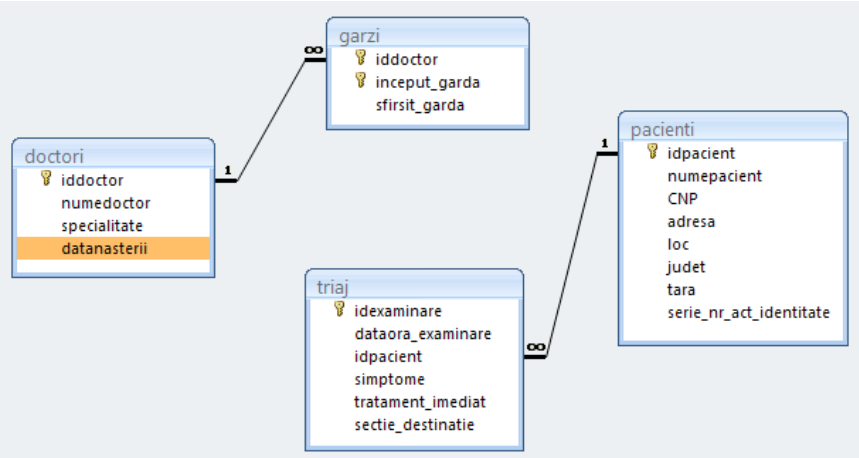

Figura 2.6. Reprezentarea din Access a mini-BD TRIAJ

În această ediție nu vom lucra (prea mult) cu MS Access. Dintre cele patru SGBD-uri alese (vezi paragraful 2.4), SQL Server (produs de Microsoft) pune la dispoziție un mecanism de reprezentare la fel de simplu precum cel din Access – vezi figura 2.7, formalismul „moştenind", de fapt, atât avantajele, cât şi dezavantajele celui din Access.

2.3.2. Scheme folosite pentru ilustrarea interogărilor în algebra relațională şi SQL

Pe parcursul următoarelor capitole, ne vom folosi, în principal, de două baze de date, dintre care prima este, mai degrabă, mini sau micro bază de date.

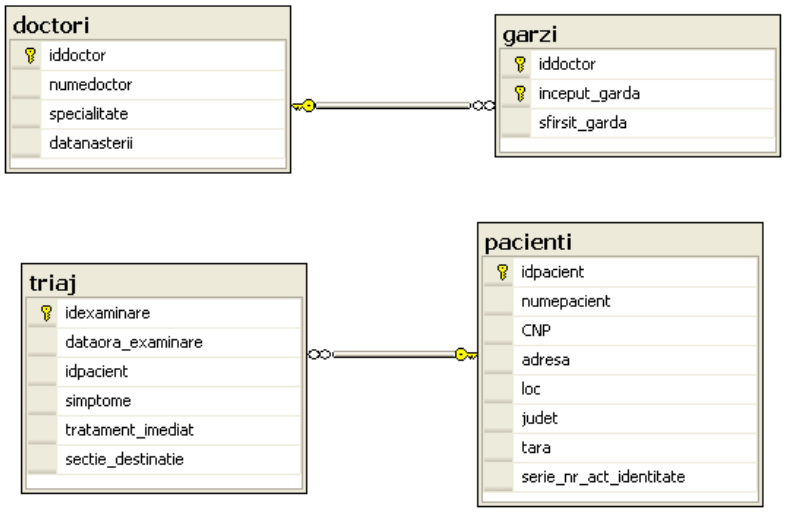

Figura 2.7. Reprezentarea în SQL Server a mini-BD TRIAJ

Baza de date TRIAJ

Deşi nu este cea mai importantă în economia lucrării, începem prin a detalia BD TRIAJ. Aceasta conține patru tabele, DOCTORI, GĂRZI, PACIENȚI şi TRIAJ care stochează informații generale despre cazurile recepționate la camera de gardă în decursul unei perioade de câțiva ani.

Tabela DOCTORI păstrează, pe fiecare linie (înregistrare), date generale despre fiecare medic al clinicii (spitalului), atributele sale fiind:

- IdDoctor – este un număr întreg care identifică fiecare doctor (un fel de marcă sau cod);
- NumeDoctor – şir de caractere al cărui rol este evident;
- Specialitate – şir de caractere ce identifică domeniul în care este pregătit doctorul (boli interne, chirurgie cardio-vasculară etc);
- DataNaşterii – o dată calendaristică asupra căreia nu mai are rost să insistăm.

Cheia primară este IdDoctor, iar pentru ultimele două atribute se pot accepta valori nule.

Tabela GĂRZI conține programarea medicilor în gărzile clinicii. Fiecare linie se referă la un medic care face o gardă. Atribute:

- IdDoctor – este un număr întreg care identifică medicul de gardă;
- Început_Gardă – o combinație dată + oră (inclusiv minut, secundă), să-i zicem moment, în care medicul intră în gardă;
- Sfârşit_Gardă – momentul (ziua şi ora) la care medicul a ieşit sau va ieşi din gardă;

Cheia primară a tabelei este combinația (IdDoctor, Început_Gardă), ceea ce înseamnă, printre altele, că, în spitalul cu pricina, într-o aceeaşi gardă pot fi programați doi sau mai mulți medici (în perioadele aglomerate). Fiind componente ale cheii primare, aceste două atribute nu pot accepta valori nule. Ultimul, însă, poate avea valori NULL, ceea ce ar însemna că este vorba de gardă în curs, ce nu s-a terminat.

IdDoctor este în tabela GĂRZI cheie străină, cheia primară corespondentă fiind atributul cu acelaşi nume din tabela DOCTORI.

Tabela PACIENȚI „oferă" câte o linie pentru fiecare bolnav, fericit sau nefericit, care a trecut pragul spitalului (şi a fost înregistrat). Atribute:

- IdPacient – este un număr întreg care identifică fiecare persoană-pacient;
- NumePacient – şir de caractere cu funcționalitate evidentă;
- CNP – Codul Numeric Personal al pacientului;
- Adresa – şir de caractere ce indică domiciliul (din cartea de identitate) pacientului;
- Loc – localitatea în care îşi are domiciliul pacientul (este separată de adresă);
- Județ – județul în se care se găseşte localitatea;
- Țară – țara în care îşi are domiciliul pacientul;
- Serie_Nr_Act_Identitate – seria şi numărul actului de identitate al pacientului.

Cheia primară a tabelei este IdPacient. Interesant este că acesta este singurul atribut al tabelei care nu poate accepta valori nule, ceea ce pare destul de curios. Motivul este că la clinică pot sosi victime neînsoțite ale unor accidente rutiere care nu au asupra lor niciun act de identitate, şi, până în momentul în care devin conştiente şi li se recuperează documentele, trebuie preluate în baza de date fără a se şti mai nimic despre ele.

Tabela TRIAJ conține, pe fiecare dintre linii, un „caz", adică sosirea unei persoane în stare mai bună mai mai rea. În decursul anilor, o persoană care se regăseşte în tabela PACIENȚI pe o singură linie/înregistrare, poate avea mai multe linii din TRIAJ corespondente. Astfel, un ghinionist poate a avut pe 13 mai 2005 o criză renală, pe 13 ianuarie 2007 a alunecat pe gheață şi şi-a rupt un picior, iar pe 13 iunie 2008 a avut un accident rutier în care şi-a rupt o coastă. Toate cele trei bucurii s-au petrecut în preajma spitalului, care îl poate considera client fidel. Atribute:

- IdExaminare – este un număr întreg care identifică orice sosire a unui pacient în camera de gardă, sosire care devine „caz";
- DataOra_Examinare – momentul (data/ora) sosirii/examinării pacientului;
- IdPacient – numărul întreg care identifică pacientul;
- Simptome – şir de caractere în care medicul de gardă descrie cum se prezenta pacientul în momentul sosirii;

- Tratament_Imediat – medicamente, masaje cardiace şi alte operaţiuni pe care medicul de gardă le-a administrat/aplicat pacientului, ţinând seama de simptome;
- Secţie_Destinaţie – şir de caractere care indică eventuala secţie în care a fost transferat pacientul, după administrarea tratamentului imediat.

Cheia primară a tabelei este IdExaminare. Acest atribut, împreună cu DataOra_Examinare şi IdPacient, nu pot accepta valori nule. IdPacient este cheie străină (tabela părinte fiind PACIENŢI).

Baza de date VÂNZĂRI

Capitolele următoare vor utiliza cu precădere o bază de date "martor", denumită VÂNZĂRI, a cărei schemă simplificată este cea din figura 2.8.

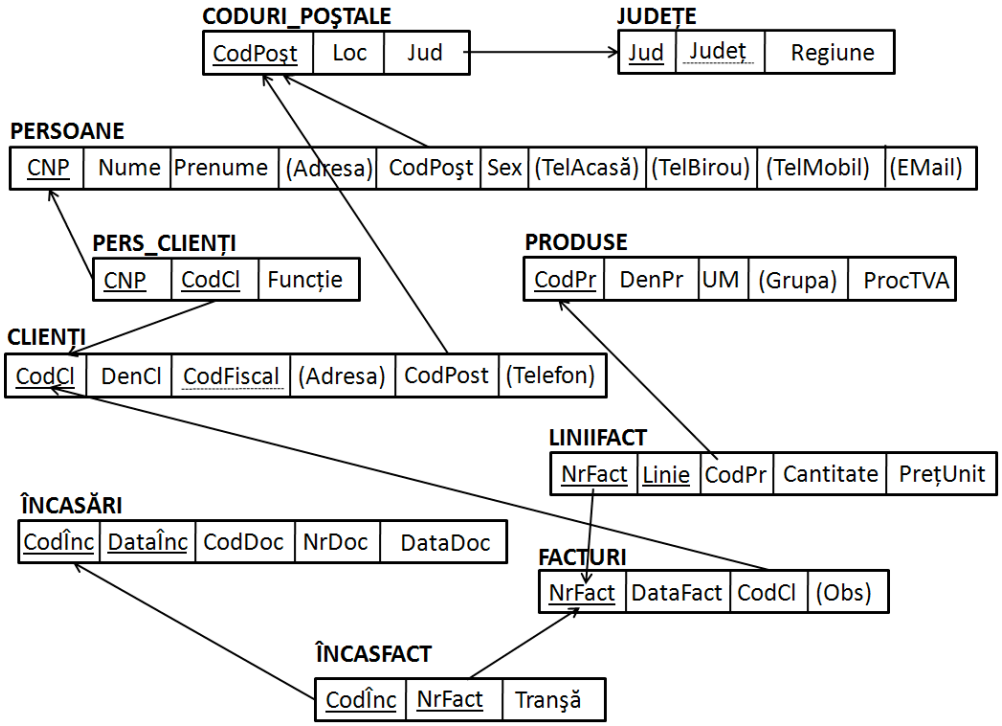

Figura 2.8. Schema simplificată a bazei de date VÂNZĂRI

Tabela JUDEŢE conţine câte o linie pentru fiecare judeţ în care sunt clienţi. Atribute:

- Jud – indicativul auto al judeţului (alcătuit din două litere, majuscule);
- Judeţ – denumirea judeţului;
- Regiune – regiunea istorică (provincia) din care face parte judeţul.

Cheia primară este atributul Jud, iar atributul Judeţ este cheie alternativă. Niciunul dintre atribute nu acceptă valori nule. Restricţii-utilizator:

- Valoarea atributului Jud să fie alcătuită numai din majuscule (eventual, un spațiu, pentru Bucureşti);
- Fiecare cuvânt din valoarea atributului Județ să înceapă cu majusculă; restul literelor să fie mici (minuscule);
- Valoarea atributului Regiune să înceapă cu o majusculă; restul literelor să fie mici;
- Regiune poate avea numai una din valorile: Banat, Dobrogea, Muntenia, Oltenia, Transilvania, Moldova.

Tabela CODURI_POŞTALE conține câte o linie pentru fiecare cod poştal dintr-un oraş sau comună:
- CodPoşt – codul poştal;
- Loc – denumirea comunei/oraşului;
- Jud – indicativul auto al județului în care se află oraşul/comuna.

Cheia primară este atributul CodPoşt. Atributul Jud este cheie străină, tabela părinte fiind JUDEȚE. Nici CodPost, nici Loc nu trebuie să accepte valori NULL. Restricții-utilizator:
- Valoarea lui Jud este alcătuită numai din majuscule (eventual, un spațiu, pentru Bucureşti);
- Fiecare cuvânt din Localitate începe cu majusculă; restul literelor sunt mici.

Tabela CLIENȚI păstrează date generale ale clienților firmei pentru care s-a constituit baza de date (o linie – un client). Atribute:
- CodCl – codul clientului;
- DenCl – denumirea clientului (persoană juridică);
- CodFiscal – codul fiscal;
- Adresa – adresa sediului firmei client;
- CodPoşt – codul poştal al comunei sau oraşului;
- Telefon – telefonul (principal) al clientului.

Cheia primară este atributul CodCl, iar CodFiscal este cheie alternativă întrucât clienții sunt exclusiv persoane juridice (firme). Atributul CodPoşt este cheie străină, tabela părinte fiind CODURI_POŞTALE. DenCl şi Adresa încep cu majuscule. Numai Adresa şi Telefon pot avea valori NULL. De asemenea, valoarea minimă acceptată pentru atributul CodCl este 1001.

Tabela PERSOANE stochează câteva date despre persoanele cheie de la firmele client: directori generali, directori financiari, şefi ai compartimentelor comerciale (aprovizionare şi/sau vânzări) etc. Probabil că vi se pare ciudat, dar ceea ce intenționăm cu această tabelă (şi următoarea) este să sprijinim fidelizarea clientului. Nu costă (mai) nimic dacă de Sfântul Ion se trimite, din partea firmei noastre, câte o felicitare tuturor Ionilor şi Ioanelor. Iar pentru ca lucrurile să fie şi mai bine puse la punct, ar fi trebuit să preluăm şi informații precum: data naşterii, starea civilă, numele şi vârsta copiilor, pasiuni în materie de muzică, arte pastice, literatură, sport etc. Schema luată în considerare s-a oprit la următoarele atribute:
- CNP – codul numeric al persoanei;

- Nume;
- Prenume;
- Adresa – adresa domiciliului stabil (din cartea de identitate);
- Sex;
- CodPoşt – codul poştal al adresei;
- TelAcasă – numărul telefonului de acasă;
- TelBirou – numărul telefonului (fix) de la birou;
- TelMobil – numărul "mobilului";
- Email – adresa e-mail.

Deşi este un număr compus din 13 poziţii, este recomandabil ca CNP să fie declarat de tip şir de caractere. Aceasta deoarece în firmă există rezidenţi din alte ţări care au poziţii de vârf în firme româneşti (sau reprezentanţe ale unor firme străine). Ori, codul identificator din paşaport poate fi alcătuit din litere şi cifre.

Cheia primară este atributul CNP (să presupunem că firma le poate obţine, deşi e puţin plauzibil). Fiind vorba de persoane „exterioare" companiei, un atribut precum Marca este mai puţin indicat. Atributul CodPost este cheie străină, tabela părinte fiind CODURI_POŞTALE. Alte restricţii:

- fiecare cuvânt din Nume şi Prenume începe cu majusculă; restul literelor sunt mici;
- adresa începe cu majusculă;
- atributul Sex poate lua numai două valori: B de la Bărbătesc şi F de la Femeiesc.

Cu ajutorul tabelei PERSCLIENŢI cunoaştem funcţia deţinută de fiecare persoană la unul (sau mai multe, deşi aceste cazuri sunt rare) firme-client. O linie din această tabelă reflectă o funcţie deţinută de o persoană într-o firmă. O persoană poate avea mai multe funcţii, chiar la aceeaşi firmă (cumul de funcţii). Cele trei atribute sunt:

- CNP – codul numeric personal al persoanei care deţine funcţia;
- CodCl – firma la care persoana deţine funcţia;
- Funcţie – funcţia deţinută.

Cheia primară este compusă din toate cele trei atribute, (CNP, CodCl, Funcţie), deoarece, pe de o parte, o persoană poate cumula două sau mai multe funcţii la aceiaşi firmă, iar, pe de altă parte, o persoană poate, în unele cazuri să lucreze la două sau mai multe firme. CNP şi CodCl sunt chei străine, tabelele părinte fiind PERSOANE, respectiv CLIENŢI. Atributul Funcţie începe cu o majusculă, restul literelor fiind mici.

Tabela PRODUSE reprezintă nomenclatorul de produse şi servicii pe care le comercializează firma (produsele sunt obţinutele prin manufacturare sau revânzare). Atribute:

- CodPr – codul produsului;
- DenPr – denumirea;
- UM – unitatea de măsură a produsului;

- Grupa – grupa de mărfuri (sortimente) în care se încadrează; acest atribut este important pentru analiza vânzărilor;
- ProcTVA – procentul TVA care se aplică la prețul de vânzare (preț care este fără TVA); pare de prisos în condițiile actuale, dacă toate produsele vândute de firma noastră ar avea un singur procent, 19%; există, însă şi produse la vânzarea cărora se aplică doar 9%, şi, în plus, nimeni nu poate garanta cum vor gândi promoțiile viitoare de guvernanți "relaxarea fiscală";

Cheia primară este atributul CodPr. DenPr şi Grupa încep cu majusculă, restul literelor din valoarea acestor atribute fiind mici.

Tabela pare una "inofensivă", dar de modul său de organizare depinde rezolvarea unor probleme sensibile. Este important de stabilit regimul în care se va lucra cu produsele care se comercializează sub mai multe forme (cu unități de măsură diferite. Atributul DenPr nu a fost declarat cheie alternativă. De ce ? Să luăm un exemplu absolut la întâmplare: o firmă de comerț en-gross vinde, printre altele, produsul Vodcă X şi la cutii de 1 litru şi la peturi (sticle de plastic) de 250 ml.

PRODUSE

CodPr	DenPr	UM	Grupa	ProcTVA
...				
1234	Vodca X - cutie 1 l	buc	Spirtoase	0.19
...				
1273	Vodca X - pet 250 ml	buc	Spirtoase	0.19
...				

varianta 1:

Cheie primară: CodPr

Cheie alternativă: DenPr

Figura 2.9. Specificarea, în valoarea DenPr, a tipului cutiei

Deşi este acelaşi produs, pentru o evidență corectă a vânzărilor, este necesar ca în tabela PRODUSE să existe două linii afectate vodcii Scandic, fie ca în figura 2.9, fie ca în figura 2.10.

PRODUSE

CodPr	DenPr	UM	Grupa	ProcTVA
...				
1234	Vodca X	cutie 1 l	Spirtoase	0.19
...				
1273	Vodca X	pet 250 ml	Spirtoase	0.19
...				

varianta 2:

Cheie primară: CodPr

Cheie alternativă: (DenPr, UM)

Figura 2.10. Diferenţierea (în afară de cod) numai prin UM

Fiecare variantă are avantajele şi dezavantajele sale.

Tabela FACTURI conține câte o linie pentru fiecare factură emisă, factură ce reflectă o vânzare (către un client). Atribute:

- NrFact – numărul facturii;
- DataFact – data întocmirii facturii;
- CodCl – codul clientului căruia i s-au vândut produsele/serviciile consemnate în factură;
- Obs – observații; e folosit relativ rar, pentru a introduce eventuale detalii sau probleme care au apărut în legătură cu o factură.

Cheia primară este atributul NrFact. Explicația este una simplă: gestionarea facturilor emise este, conform legii, strictă. Nu pot exista două facturi (care reflectă două vânzări) cu un acelaşi număr[20]. CodCl este cheie străină (tabela părinte este CLIENȚI). Ca restricție utilizator suplimentară, să zicem că interzicem preluarea facturilor întocmite înainte de 1 ianuarie 2007 sau după 31 decembrie 2015 (*DataFact* >= *DATE'2007-01-01' AND DataFact* <= *DATE'2015-12-31'*).

Tabela LINIIFACT „detaliază" tabela precedentă. Un tuplu se referă la un produs/serviciu vândut şi consemnat într-o factură emisă. Pentru fiecare factură vor fi atâtea linii câte produse/servicii au fost consemnate la vânzarea respectivă. Atribute:

- NrFact – numărul facturii;
- Linie – numărul liniei din factura respectivă (1, 2, 3, ...);
- CodPr – codul produsului/serviciului vândut;
- Cantitate – cantitatea vândută;
- PretUnit – prețul unitar (fără TVA) la care s-a făcut vânzarea.

Cheia primară este combinația (NrFact, Linie). NrFact şi CodPr sunt chei străine. Pentru a determina valoarea de încasat a unei facturi (inclusiv TVA), la valoarea fără TVA pentru fiecare linie (obținută prin produsul *Cantitate * PretUnit*) trebuie adăugată TVA colectată, obținută prin aplicarea procentului de TVA al produsului/serviciului (atributul ProcTVA din PRODUSE) la valoarea fără TVA.

Tabela INCASĂRI reprezintă „nomenclatorul" încasărilor. Printr-o încasare, un client îşi stinge una sau mai multe obligații de plată, adică achită una sau mai

[20] În ultimii ani, legislația s-a mai domolit în privința numerotării facturilor. Companiile pot refolosi numere de facturi în fiecare an (prima factură din orice an poate avea, spre exemplu, numărul 100001), însă noi vom menține restricția de unicitatate a valorilor NrFact în tabela FACTURI.

multe facturi. Documentul primar pe baza căruia se consemnează încasarea poate fi ordinul de plată, cecul, chitanța etc. Atributele tabelei sunt:

- CodInc – codul încasării - un număr intern, util pentru a diferenția o încasare de celelalte;
- DataInc – data încasării - data la care banii au intrat în contul sau casieria firmei;
- CodDoc – codul documentului justificativ al încasării: OP – ordin de plată, CHIT – chitanță, CEC – filă cec etc.;
- NrDoc – numărul documentului justificativ;
- DataDoc – data la care a fost întocmit documentul justificativ; din momentul întocmirii documentului de plată până la data la care banii ajung efectiv în cont/casierie trec câteva zile datorită circuitului documentelor între firme şi bănci, sau intervalul specificat în biletul la ordin sau în cambie.

Cheia primară este atributul CodÎnc. Ca restricții utilizator pot fi instituite:

- data încasării nu o poate precede pe cea a întocmirii documentului (*DataÎnc* >= *DataDataDoc*);
- data documentului de încasare şi data încasării efective trebuie să fie în intervalul de 1 ianuarie 2007 - 31 decembrie 2015 (*DataDoc* >= *DATE'2007-01-01' AND DataDoc* <= *DATE'2015-12-31'*);
- codul documentului se scrie numai cu majuscule.

Tabela INCASFACT detaliază tabela precedentă şi indică ce facturi (tranşe din facturi) sunt achitate prin fiecare încasare. Un client poate plăti mai multe facturi odată (printr-un singur document). Pe de altă parte, orice factură poate fi plătită într-una sau mai multe tranşe, în funcție de banii de care dispune clientul la momentul întocmirii documentului de plată. Atributele tabelei sunt:

- CodÎnc – codul încasării;
- NrFact – factura pentru care se încasează valoarea integrală sau numai o tranşă;
- Tranşă – tranşa din factură (sau întreaga valoare) care se încasează prin documentul primar ce stă la baza încasării.

Cheia primară este combinația (CodÎnc, NrFact), deoarece la o încasare se pot achita mai multe facturi, iar, pe de altă parte, o factură poate fi plătită în mai multe tranşe. CodÎnc şi NrFact sunt chei străine. Respectarea restricției de entitate presupune ca NrFact să nu poată avea valori nule, ceea ce atrage după sine un serios neajuns al acestei tabele – imposibilitatea de a prelua încasările în avans (înaintea întocmirii facturilor).

2.3.3. Alte obiecte din schema unei baze de date relaționale

Pe lângă tabele, atribute şi restricții, dicționarul de date al unei BDR conține multe alte tipuri de obiecte, dintre care vom zăbovi foarte puțin asupra tabelelor temporare şi virtuale (view-uri), procedurilor stocate şi secvențelor. Cele mai „vechi", în ordinea apariției lor în teoria relațională sunt tabelele persistente (de

bază, „clasice") şi tabelele virutale. Unele, precum procedurile stocate şi secvenţele, nu fac parte din modelul relaţional şi au apărut în SGBD-uri din raţiuni practice, fiind încorporate ulterior în stardardul SQL.

Tabele temporare

O caracteristică fundamentală a oricărei BD este persistenţa. Tabelele prezentate până acum sunt cele care asigură această caracteristică, întrucât, după crearea lor (şi, implicit, stocarea schemei lor în dicţionarul de date), informaţiile introduse şi modificate sunt păstrate pe o durată nedeterminată, până la ştergerea lor explicită sau până la vreun nedorit dezastru hardware sau software.

Mai mult, în condiţiile în care la baza de date are acces un mare număr de utilizatori, modificările efectuate de un utilizator (şi confirmate prin încheierea eventualei tranzacţii) în conţinutul unei tabele sunt vizibile celorlalţi utilizatori. Există însă şi posibilitatea de a declara tabele a căror schemă să fie stocată în dicţionar, dar pentru care conţinutul să fie volatil. Astfel, pot fi create tabele al căror conţinut să fie vizibil doar *sesiunii* de lucru care face apel la tabela respectivă, pe perioada întregii sesiuni. Acestea sunt tabelele *temporare globale*. Structura tabelelor *temporare locale* este, de asemenea, păstrată în dicţionarul de date, însă vizibilitatea conţinutului lor este limitată doar la *un singur modul/bloc* apelant în cadrul sesiunii. La ce ar putea folosi tabele ale căror conţinut ar fi aşa de efemer ? Să luăm un exemplu: dorim să analizăm structura vânzărilor pe o regiune din care face parte utilizatorul (sau o regiune aleasă), după mai multe criterii, pe ultimele 12 luni. Datele fiind „istorice", iar BD folosită de mulţi utilizatori/aplicaţii, în loc să consultăm repetat tabelele persistente, le putem „menaja" creând una sau mai multe tabele temporare pe care să le stoarcem (pe îndelete) de informaţii. Pe toată durata „vieţii", liniile tabelelor temporare rămân neschimbate, deci rezultatele sunt corecte (consistente). În capitolul 14 vom vedea modul în care pot fi create în SQL tabelele temporare.

Tabele virtuale

În engleză sintagma pentru acest tip de tabelă este *view*, adică *vedere* sau *imagine*. De aceea, în lucrările româneşti pot fi întâlnite mai multe titulaturi. O tabelă virtuală stabileşte o legătură semantică (printr-o expresie relaţională sau interogare) între tabele persistente şi/sau alte tabele virtuale, neavând tupluri proprii, ca o relaţie de bază (statică). Conţinutul (instanţierea) tabelei virtuale depinde, la un moment oarecare dat, de conţinutul tabelelor de bază din care derivă. Pentru a înţelege mai bine diferenţa, explicaţia se poate rezuma astfel: *tabela virtuală este cea pentru care pe disc se memorează numai schema, nu şi conţinutul*.

Ca şi tabelele temporare (şi cele „obişnuite"), definiţia unei tabele virtuale se păstrează pe disc. Înregistrările unei tabele temporare sunt stocate până la finalizarea modulului/blocului de cod (cazul celor locale) sau sesiunii (cazul celor globale), în timp ce înregistrările unei tabele virtuale nu se păstrează. Şi tabelele temporare şi cele virtuale pot fi actualizate. Actualizarea nu ridică nicio problemă în cazul tabelelor temporare. În schimb, actualizarea unei tabele virtuale trebuie să se propage în tabelele persistente din care provind tuplurile acesteia. Sunt situaţii

destul de frecvente în care o modificare a unei linii într-o tabelă virtuală nu poate fi „tradusă" cu exactitate în modificări ale tabelelor de bază, cazuri în care, de obicei, aceste modificări sunt respinse de către SGBD.

Tabelele virtuale oferă oricărui utilizator al unei baze de date posibilitatea prezentării datelor în funcţie de nevoile sale specifice. De asemenea, raţiuni de securitate şi confidenţialitate a anumitor informaţii pot conduce la izolarea unor date faţă de utilizatorii neautorizaţi, lucru deplin posibil prin intermediul view-urilor. Pornind de la aceleaşi tabele de bază, se pot crea oricâte tabele virtuale, în funcţie de situaţie. Tabelele virtuale constituie suportul creării schemelor externe. Odată definită, o tabelă virtuală poate fi referită ca o tabelă de bază oarecare.

Proceduri stocate

O procedură stocată este o secvenţă de program (cod) a cărei definiţie/semnătură face parte integrantă din baza de date. Avantajele utilizării procedurilor stocate decurg din faptul că acestea sunt parte din structura (schema) bazei, fiind păstrate în dicţionarul de date (catalogul sistem). Există mai multe tipuri de proceduri stocate: funcţii pentru calculul unor valori implicite, proceduri/funcţii de validare la nivel de linie sau înregistrare, funcţii/proceduri de calcul a unor expresii complexe etc. Începând cu versiunea 1999 a standardului, SQL-ul prezintă o componentă importantă – PSM (Persistent Stored Modules) care încearcă să instituie un etalon în redactarea procedurilor stocate. Unificarea PSM este o sarcină delicată. SQL-ul a apărut şi s-a consacrat la limbaj *neprocedural*, care nu lucrează cu module de program clasice. Aşa că fiecare producător de SGBD-uri şi-a perfecţionat unul (sau chiar mai multe) limbaje procedurale. Alinierea la specificaţiile PSM este un proces de durată şi oarecum de uzură (dar nu pentru noi).

Declanşatoare

Declanşatorul (trigger) a apărut ca un tip special de procedură stocată care este executată automat când un eveniment predefinit (inserare, actualizare sau ştergere) modifică o tabelă. Utilitatea declanşatoarelor este evidentă la formularea unor restricţii mai complexe decât "suportă" comenzile CREATE/ALTER TABLE, cum ar fi: actualizarea automată a unor atribute calculate, jurnalizarea modificărilor suferite de baza de date, păstrarea integrităţii referenţiale etc.

Există diferenţe sensibile între SGBD-uri şi în ceea ce priveşte sintaxa declanşatoarelor, deoarece acestea sunt redactate în extensiile procedurale ale SQL, care prezintă diferenţieri majore de la un producător la altul. În ultimii zece ani, au apărut şi alte evenimente-declanşatoare în afara celor trei, cum ar fi: conectarea la baza de date care echivalează cu deschiderea unei sesiuni de lucru, închiderea sesiunii, crearea unui obiect în BD, pornirea sau oprirea bazei de date etc.

Secvenţe

Secvenţa este un alt tip de obiect al unei BDR care nu face parte, propriu-zis, din modelul relaţional. A apărut din necesitatea asigurării unicităţii cheilor primare/alternative. Să luăm cazul atributului CodCl din tabela CLIENŢI. La

inserarea unui client, valoarea sa pentru atributul CodCl ar trebui să fie superioară celei a precedentului client introdus, sau, în orice caz, dacă nu superioară, măcar diferită. Dacă ar fi un singur utilizator cu drept de introducere a firmelor client în BD, lucrurile ar fi relativ simple, corectitudinea valorilor CodCl depinzând de atenția sa. Acest scenariu este cu totul improbabil în aplicațiile pentru afaceri, în care pot exista zeci, chiar sute sau mii de utilizatori ce lucrează simultan cu baza, iar, mai ales, în primele săptămâni de la darea în folosință a aplicației, clienții pot fi introduşi de către mai mulți utilizatori (uneori chiar simultan).

Secvența este un obiect care, la fiecare invocare a sa, generează o valoare unică, indiferent de câte cereri/invocări simultane sunt. Crearea unei secvențe presupune, în afara numelui său, precizarea minimului, maximului şi a incrementului valorilor generate.

2.4. SGBDR-uri şi servere de baze de date relaționale

Până acum am folosit cei doi termeni, *SGBD* şi *server de baze de date*, fără a avea prea mare grijă în a preciza ceea ce le aseamănă şi ceea ce le diferențiază. Nici de acum încolo nu vom fi prea grijulii, stabilind însă, din capul locului, că orice server de bază de date (server BD) este un SGBD, însă nu orice SGBD este server BD.

2.4.1. Scurtă istorie „veche" a SGBDR-urilor

În prima ediție a lucrării am evocat principalele repere cronologice în realizarea celor mai importante SGBD-uri relaționale. Acum o să scurtăm serios povestea. Lucrările fondatoare ale lui E.F.Codd au fost primite cu entuziam reținut de către firma la care lucra acesta – IBM. Rezerva IBM era legată nu numai de faptul că era vorba de bazele unei tehnologii noi, şi nici cei mai buni prezicători nu aveau cum să ghicească dacă relaționalul se va dovedi viabil sau va fi un eşec. IBM-ul avea, la acel moment, unul dintre cele mai puternice SGBD-uri de pe piață – IMS (care gestiona baze de date ierarhice), în care investise mulți bani şi oameni. Ori, afirmațiile lui Codd la adresa IMS-ului, făcute deseori în prezența clienților, nu erau dintre cele mai măgulitoare, ceea ce a iritat o serie de „fețe bisericeşti" ale companiei[21].

[21] Evocarea are un uşor aer de poveste. Eu le-am auzit de la Fabian Pascal, care le preluase de la Chris Date.

După câţiva ani, IBM se hotărăşte să aloce bani într-un proiect care să se materialezeze într-un SGBD relaţional - System/R. Proiectul s-a derulat în laboratorul Santa Teresa din San Jose, California. Prima fază - anii 1974 şi 1975 - s-a concretizat într-un prototip al SGBDR-ului, prototip care a fost rescris în anii 1976 şi 1977 pentru a permite interogarea simultană a mai multor tabele şi accesul multi-utilizator. IBM a distribuit produsul pe parcursul anilor 1978 şi 1979 la câţiva colaboratori ai săi pentru evaluări, dar în 1979 firma abandonează proiectul System/R, considerându-l nefezabil[22]. System/R a fost însă punctul de plecare pentru realizarea unuia dintre cele mai cunoscute SGBDR-uri, DB2.

Între timp, au fost realizate alte SGBD-uri, fie în stadiu de prototip, fie finisate de-a binelea: INGRES (Interactive Graphics and Retrieval System) - al Universităţii Berkeley, QUERY BY EXAMPLE - Centrul de cercetări T.J. Watson, PRTV (Peterlee Relationnel Test Vehicle) - Centrul Ştiinţific Peterlee (Marea Britanie) patronat de firma IBM, RDMS (Relational Data Management System) - General Motors, INFORMIX (denumită iniţial Relational Data Systems) etc.

Ca reper, reţinem şi anul 1977, când un grup de ingineri de la Menlo Park, California, a fondat compania Relational Software Inc., cu scopul declarat de a dezvolta un SGBD bazat pe limbajul SQL. Produsul, denumit Oracle, a fost lansat în 1979 şi a reprezentat primul SGBDR comercial. Dincolo de faptul că Oracle precede cu doi ani primul SGBDR comercial al firmei IBM, orientarea sa a fost extrem de bine gândită, rulând pe minicalculatoarele VAX (Digital Equipment Corporation), sensibil mai ieftine decât mainframe-urile IBM, şi, deci, cu un segment de piaţă mai mare. Astfel, succesul a fost considerabil. Astăzi, firma, redenumită Oracle Corporation, este liderul produselor software dedicate mediilor de lucru cu bazele de date.

Primul SGBDR comercializat de IBM a fost SQL/DS (DS - Data System) anunţat în 1981 şi comercializat în 1982. În 1983 IBM a lansat DB2, dedicat iniţial mainframe-urilor sale, dar care în timp a fost portat pe toate sistemele de operare importante.

Până la jumătatea anilor '80, SGBD-urile relaţionale au stat în umbra celor ierarhice şi reţea. Motivul principal l-a constituit viteza mai mică, parametru esenţial în sistemele "confruntate" cu accesul simultan a sute sau chiar mii de utilizatori. În a doua parte a anilor '80 performanţele SGBDR-urilor au fost net ameliorate. Câştigul de viteză, care s-a adăugat atuului principal, interogarea ad-

[22] Pentru amănunte privind odiseea System R, vezi şi documentele de la adresa web: http://www.mcjones.org/System_R/SQL_Reunion_95

hoc prin utilizarea limbajelor de generaţia a IV-a (SQL, QBE, Quel), a lărgit continuu segmentul ocupat de SGBDR-uri.

Odată cu apariţia PC-urilor, tehnologia relaţională a devenit accesibilă unei largi categorii de utilizatori. Din anii '80 Încoace, produse precum dBase (Ashton-Tate, Borland), Rbase (Microrim), Clipper (Nantucket, Computer Associates), DBFast (Computer Associates), Foxbase-FoxPro (Fox Software, Microsoft), Paradox (Borland, Corel), Approach (Lotus, IBM), Access (Microsoft) au fost folosite de milioane de deţinători de calculatoare personale, de la ignoranţi, până la profesionişti în dezvoltarea de aplicaţii cu baze de date.

Este drept, deşi categoria începătorilor în ale bazelor de date ne interesează – nu numai ca posibili cumpărători ai cărţii, cât mai ales ca doritori de cunoştinţe pentru a-şi folosi cât mai bine bazele de date proprii (şi mai ales ca să-şi dorească mai mult de la acest domeniu) -, în capitolele care urmează nu vom zăbovi asupra dialectelor SQL ale acestor SGBD-uri, ci ale patru dintre cele mai importante servere BD actuale.

2.4.2. SGBD-uri non-servere BD

Cele mai simple şi ieftine SGBDR-uri sunt cele pentru uz individual sau aplicaţii de tip file-server. Exemplele clasice sunt Visual FoxPro, Paradox şi Access. Visual FoxPro-ul a avut în ţara noastră un mare succes, comparabil doar cu cel din câteva ţări est-europene. Modul în care calculatoarele şi reţelele de calculatoare au pătruns şi s-au extins la noi explică adoptarea sa în multe aplicaţii de întreprindere şi predarea sa în şcoli, licee şi universităţi. Generaţia mea de truditori în ale bazelor de date este, în bună parte, generaţia Foxpro[23]. Astăzi termenul este, frecvent, unul peiorativ, dar la timpul său (la începutul anilor ‘90), cu greu se putea găsi la noi ceva mai bun, ieftin (şi uşor de copiat).

Astăzi Visual FoxPro (vezi figura 2.11), ajuns la versiunea 9, a ieşit la pensie. Microsoft-ul s-a hotărât să oprească producţia şi suportul pentru acest SGBD venerabil. Semnele rele erau vizibile de mai bine de zece ani. Microsoft nu a mai venit cu modificări importante legate de gestiunea bazelor de date în FoxPro de la versiunea Visual FoxPro 5 lansată 1994. Versiunile ulterioare au mai corectat din erori, au mai introdus câteva opţiuni privitoare la nucleul SQL (mai ales la versiunea 9), şi… cam atât. Este adevărat, VFP nu este doar un SGBD, ci un mediu integrat de dezvoltare de aplicaţii, cu foarte bune generatoare de rapoarte, meniuri

23 Vezi [Fotache s.a.2002]

şi formulare, şi un limbaj de programare care a fost orientat pe obiecte cu mult înaintea Visual Basic-ului.

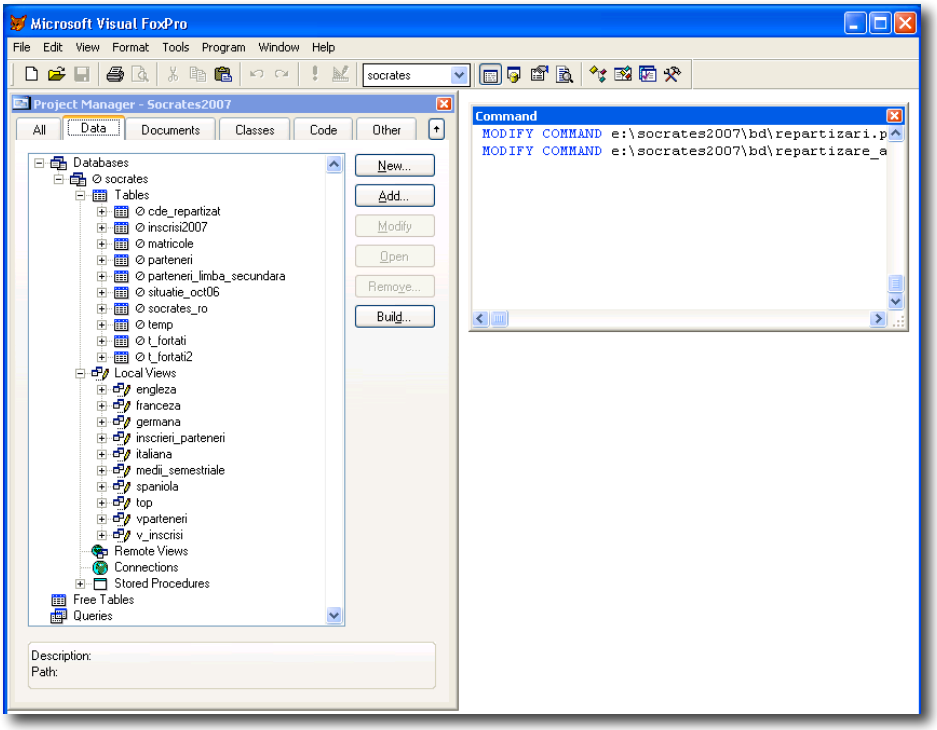

Figura 2.11. Afişarea conţinutului unui proiect (inclusiv schema BD) În Visual FoxPro

Totuşi, pentru a intra în lumea bună a serverelor BD, VFP trebuia înzestrat cu opţiuni de creare a utilizatorilor[24], a profilurilor, de alocare şi revocare de drepturi pe toate obiectele bazei de date. Mai mult, ar fi trebuit să permită "nativ" arhitecturi client/server şi web[25], module de arhivare, restaurare şi replicare, dar mai ales să se descurce cu recuperarea bazei în cazuri de dezastre (întreruperi de curent, blocări ale Windows-ului). Or, responsabilii de aplicaţii complexe realizate

24 Până şi Access-ul are, de câţiva ani, opţiuni de creare de utilizatori şi chiar de replicare.

25 Există opţiuni de adaptare a VFP la arhitecturi apropiate de cea c/s şi web (vezi [Fotache s.a.2002], însă efortul dezvoltatorilor de aplicaţii este considerabil.

în VFP, cu zeci de utilizatori conectați simultan la BD, știu ce este adrenalina când "pică nişte indecşi"[26].

Deşi i-a fost anunțat sfârşitul, mai am o speranță pentru VFP, şi anume ca, odată ce Microsoft îşi va lua mâinile de pe produs, ar putea intra în lumea open-source, şi cunoaşte o nouă viață (ar fi, cred, a treia…).

Cel mai celebru SGBD din categoria non-serverelor BD este Access, copilul pe care Microsoftul l-a preferat în dauna Visual FoxPro-ului[27]. Access-ul are toate atuurile care l-au făcut aşa de popular, de la uşurința folosirii, până la integrarea cu celelalte aplicații din Office şi migrarea naturală la SQL Server. Şi Access este mai mult decât un SGBD – vezi figura 2.12.

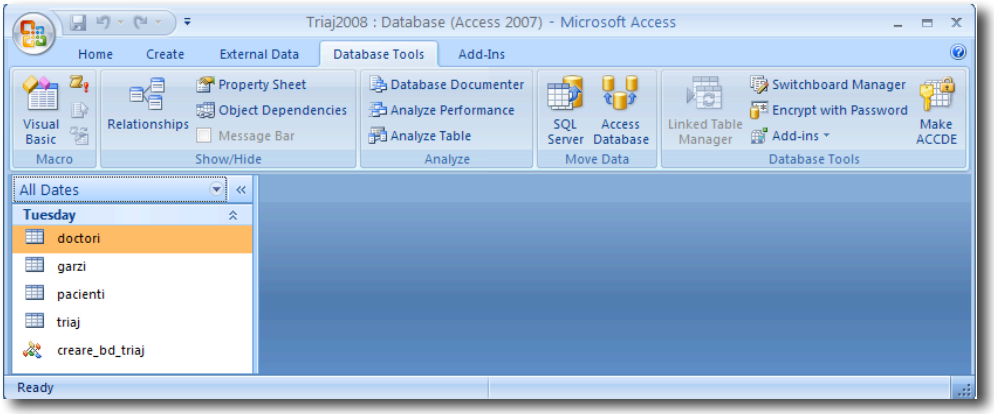

Figura 2.12. Afişarea conţinutului unui proiect (inclusiv tabelele BD) în Access

Însă, deşi i-au fost adăugate funcțiuni redutabile, Access nu este (încă) un server BD, iar dialectul său SQL este unul rezonabil, dar nu impresionant, în niciun caz superior celui din VFP.

---

[26] Vorbesc din propria experiență. Cea mai mare jenă profesională (de până acum) o am din urmă cu vreo zece ani, când, împreună cu alți colegi din Catedra de Informatică Economică, am încercat să modificăm modul de înscriere la concursul de admitere al FEAA Iaşi. Ghinionul a fost că decanul a avut încredere în noi, dar, contrar testelor preliminare, în primele patru ore de la începerea înscrierilor (se adunaseră vrei două sute de persoane) ne-au picat indecşii de vreo zece ori. Ulterior, am aflat că dintre cele douăzeci de stații (PC-uri) de înscriere, două erau "mufate" un pic greşit (şi, absolut aleatoriu, pierdeau din când în când legătura cu serverul pentru câteva fracțiuni de secundă).

[27] În fine, poate că exprimarea e cam melodramatică, dar ca fox-ist (şi ofticos pe deasupra) n-am putut să mă abțin.

2.4.3. Servere de baze de date

Aplicațiile complexe din bănci, corporații, organizații şi instituții gestionează volume uriaşe de informații şi sunt operaționale prin efortul a mii de utilizatori ce lucrează simultan cu baza de date. Cerințele de viteză şi fiabilitate sunt maxime. Practic, multe aplicații nu au niciun moment de pauză, orice întrerupere fiind foarte gravă. Vorbim de lumea *aplicațiilor critice*, cu regulile ei foarte dure, iar SGBD-urile care constituie parte din infrastructura acestor aplicații sunt cele de categorie grea: Oracle (Oracle), DB2 (IBM), Informix (IBM), Sysbase (Sybase), SQL Server (Microsoft) etc. Fiind mult mai robuste, fiabile, sunt şi mult mai costisitoare.

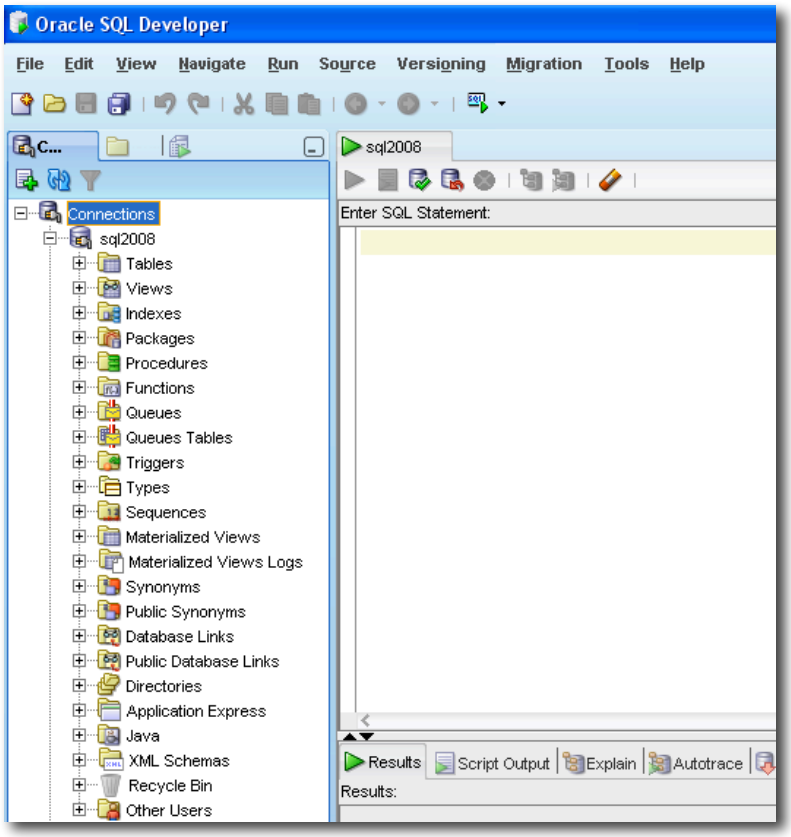

Figura. 2.13. Obiecte dintr-o bază de date Oracle (afişate de „clientul" SQL Developer)

Titlul figurii 1.6 făcea trimitere la structura unui SGBD. De fapt, numai serverele BD sunt construite pe arhitectura respectivă. Conform standardului SQL [ISO/IEC 2007-1], un server SQL:

- gestionează sesiunile SQL care se derulează între server şi un client prin intermediul unei conexiuni SQL;
- execută comenzile SQL primite de la client, recepționează şi transmite datele cerute;

- menține starea sesiunilor SQL, inclusiv a identificatorului de autorizare şi altor parametri impliciți ai sesiunii.

Deşi au existant contradicții între cifrele avansate de diferite firme de estimare a pieții bazelor de date, astăzi majoritatea lucrărilor indică drept lider firma Oracle şi produsul său cu acelaşi nume. Figura 2.13 afişează imaginea schemei BD aşa cum apare în „clientul" (de-acum „clasic") SQL Developer. Serverul BD Oracle (cea mai recenta versiune a sa fiind Oracle 11g) este un etalon în măsurarea puterii şi funcționalității unui SGBD, fiind folosit în mai toate tipurile de organizații, de la aplicații pentru firme de mărime medie până la aplicații pentru corporații multinaționale. Tradițional un produs scump, sub presiunea concurenților şi serverelor BD open-source, Oracle dispune astăzi de o politică de licențiere mult mai flexibilă.

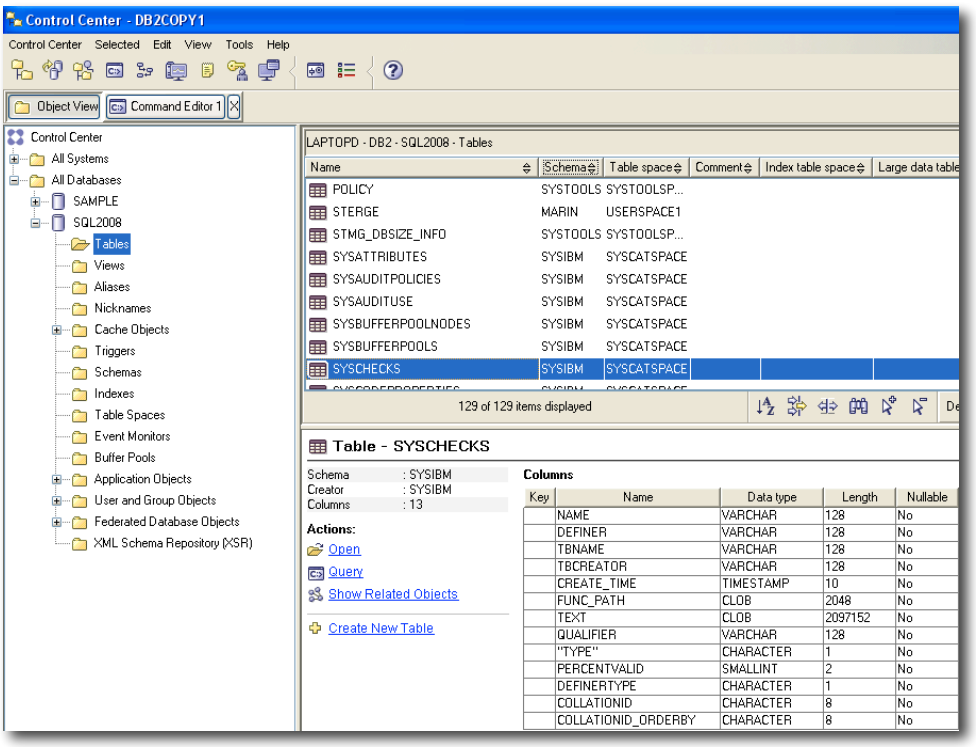

Figura. 2.14. Obiecte dintr-o bază de date DB2

Cel mai important concurent al Oracle pe piața serverelor BD este produsul DB2 al firmei IBM, ajuns la versiunea 9.5. Deşi are o imagine foarte bună şi un număr impresionant de instalări în SUA şi alte țări dezvoltate, în România prezența DB2 este mai degrabă discretă. Figura 2.14 ilustrează modul în care sunt afişate în DB2 (modulul Control Center) obiectele unei baze de date.

Pentru noi, DB2 prezintă importanță grozavă nu atât datorită pieței acoperite, cât rolului imens pe care l-a jucat firma IBM în procesul de standardizare a SQL-

ului, cel puțin primele două standarde majore, SQL-89 şi SQL-92 fiind, în mare măsură, copii ale dialectului SQL din DB2, aşa cum se prezenta la acea dată.

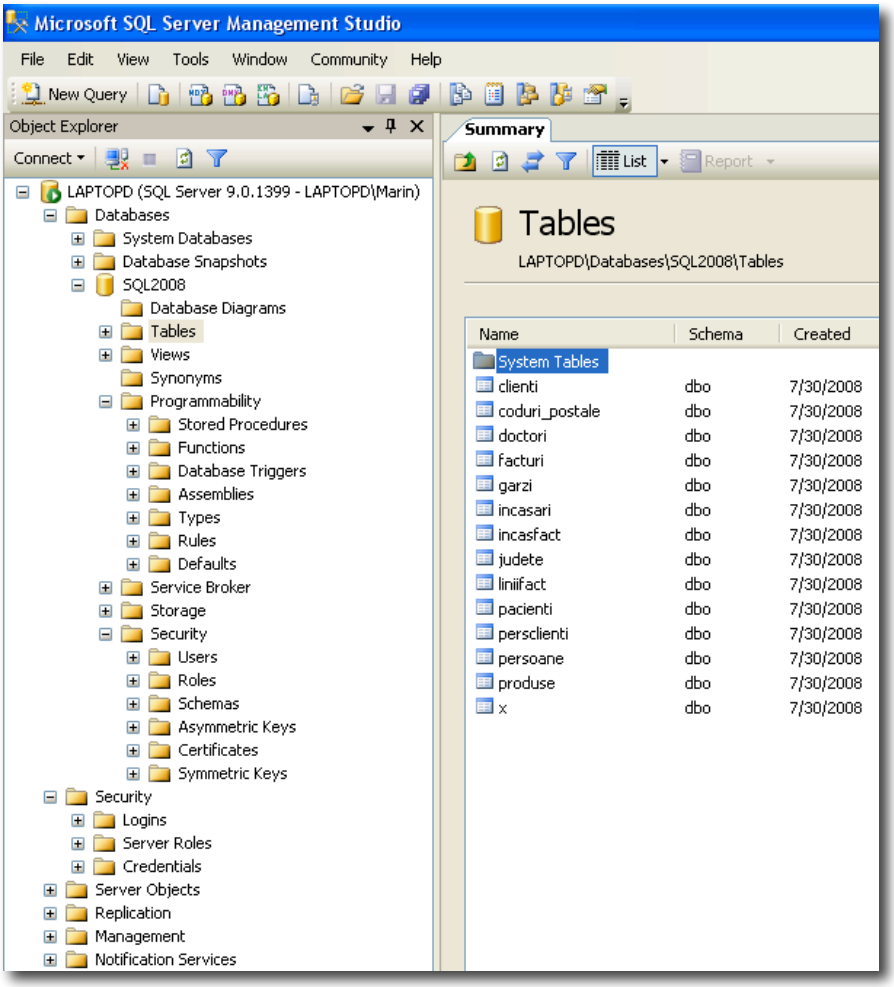

Figura. 2.15. Obiecte dintr-o bază de date MS SQL Server

În ultimul deceniu, dintre serverele „proprietare” cea mai dinamică evoluție nu a avut-o nici DB2, nici Oracle, ci produsul Microsoft denumit chiar SQL Server –

vezi figura 2.15. Titulara produsului este un pic nepoliticoasă (probabil că cea mai decentă denumire ar fi fost *One of the SQL Servers*), dar pe piață a prins[28].

Dinamica a fost asigurată şi de faptul că produsul a apărut spre mijlocul anilor 90 (baza fiind deci foarte mică). Spre deosebire de giganții DB2/Oracle, MS SQL Server a pornit-o „de jos", ca server BD pentru grupuri de lucru, ideal pentru aplicații folosite de grupuri mici şi mijlocii de utilizatori. Prețul a fost, de la început, extrem de tentant şi continuă să fie mult sub cele ale DB2 şi Oracle. Cu fiecare versiune, ultima fiind 2008[29], SQL Server a continuat să câştige în funcționalitate şi fiabilitate, astfel încât astăzi poate fi folosit şi la aplicații de mari dimensiuni.

În materie de opțiuni SQL, produsul Microsoft a prezentat destul de multe diferențe față de standard. Pe parcursul capitolelor următoare o să observăm multe funcții şi opțiuni „proprietare". Tendința de aliniere la standardele recente (SQL:1999, SQL:2003, SQL:2008) ale SQL este vizibilă, însa, cu fiecare nouă versiune, şi poate fi una dintre explicațiile succesului comercial al SQL Server-ului.

Ultimii cincisprezece ani au consacrat un segment de piață care se constituie că o alternativă apetisantă la primadonele serverelor BD – serverele BD *open-source*. Inspirate din succesul sistemelor de operare de tip open source, şi, ulterior, serverelor web, serverele BD din această categorie au atras atenția multor dezvoltatori de aplicații, mai ales în condițiile ubicuității arhitecturilor pe trei sau mai multe straturi.

Este greu de spus care este cel mai bun produs din această categorie. Dacă ar fi după numărul de utilizatori, probabil că cel mai bine plasat este MySQL[30]. Concurenții sunt, însă, foarte numeroşi şi puternici. Amintim doar pe PostgreSQL[31], mini SQL (sau mSQL)[32], Ingres[33], MaxDB[34], Firebird[35]. Paradoxul este că MySQL-ul a devenit liderul pieții serverelor BD open source după 1995 în

28 SQL Server a fost dezvoltat, inițial, printr-un parteneriat cu firma Sybase.

29 Comenzile SQL prezentate în carte au fost executate pe SQL 2005, deci produsul Microsoft a fost oarecum discriminat în privința ultimei versiuni.

30 www.mysql.com

31 www.postgresql.org

32 www.hughes.com.au

33 www.ingres.com

34 www.sdn.sap.com/irj/sdn/maxdb

35 www.firebirdsql.org

condiţiile în care avea implementat unul dintre cele mai slabe dialecte SQL, iar procedurile stocate erau evocate doar, din când în când, pe grupurile de discuţii. De la versiunea 5, MySQL-ul este mai apropiat de un nivel decent de opţiuni specifice serverelor BD.

Nivelul jenant (în materie de SQL) al MySQL-ului a fost unul dintre motivele alegerii (cu câţiva ani în urmă) a PostgreSQL-ului ca server BD open-source reprezentativ pentru lucrul cu studenţii. Deşi nescutit de probleme tehnice în a doua parte a anilor 90, astăzi PostgreSQL-ul, ajuns la versiunea 8.3, poate fi folosit în aplicaţii sofisticate, având un nivel de funcţionalitate redutabil. Mai mult, în materie de aliniere la standardele SQL, PostgreSQL merită toate laudele.

Figura 2.16 ilustrează modul în care sunt prezentate obiectele unei baze date în, probabil, cel mai popular „client" PostgreSQL – şi anume *pgAdmin* – client pe care îl vom folosi şi noi pentru exemplificare comenzilor SQL şi modulelor de proceduri stocate.

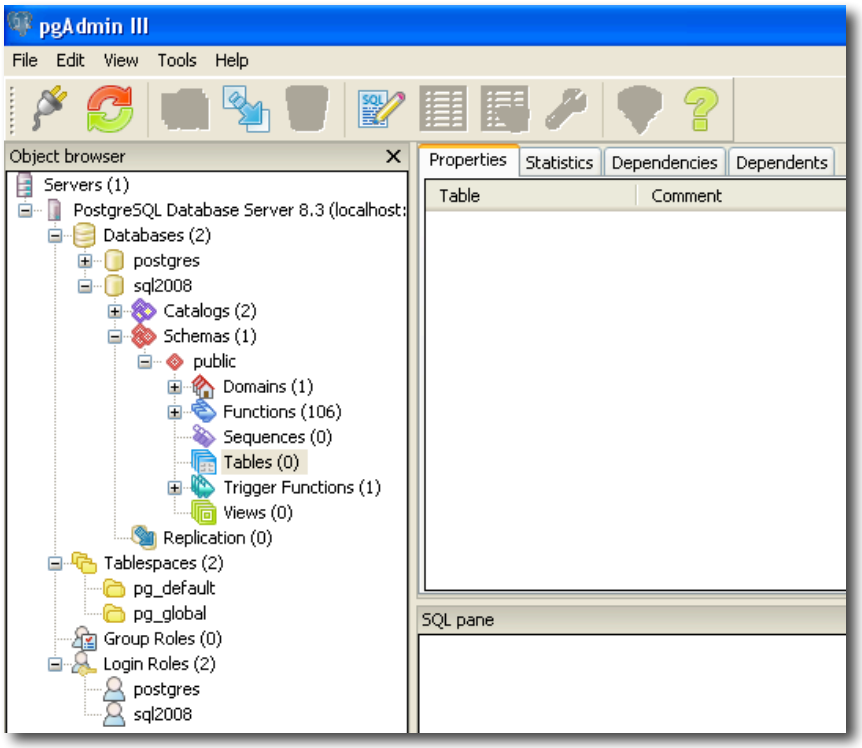

Figura. 2.16. Obiecte dintr-o bază de date PostgreSQL afişate în clientul pgAdmin