Capitolul 4

MODELUL RELAŢIONAL DE ORGANIZARE A BAZELOR DE DATE

Sumar:

- Câte ceva despre baze de date
- Prezentare generală a modelului relaţional
- Restricţii ale bazei de date
- Schema şi conţinutul unei baze de date relaţionale

4.1. CÂTE CEVA DESPRE BAZE DE DATE

Simplificând şi exagerând nepermis lucrurile, am putea spune că există doi poli între care poate fi poziţionată orice problemă informatică. Pe de o parte, cel al chestiunilor interesante, nu neapărat cu formule şi calcule complexe, cât o anume ingeniozitate în rezolvare - mult invocata şi aşteptata "fisă". Imaginea clasică a informaticianului mioritic (un informatician fără ochelari, adidaşi şi blugi este unul îndoielnic, dacă nu insuportabil !) este cea a unui programator preocupat permanent să găsească o funcţie recursivă care să-i rezolve o problemă neapărat de logică superioară.

Celălalt pol regrupează probleme în care complexitatea calculelor rareori depăşeşte nivelul celor patru operaţii aritmetice elementare - adunare, scădere şi încă două; în schimb, volumul informaţiilor şi zecile/ sutele moduri de regrupare şi agregare a lor este deconcertant.

Fără a face concurenţă vreunui manual de filosofie, putem spune că, din păcate, ca şi în viaţă, ponderea problemelor din a doua categorie – să le spunem plicticoase – este mult mai mare decât ponderea problemelor cu adevărat interesante. Aceasta ar fi vestea proastă. Vestea bună este că se câştigă enorm de mulţi bani din chestiunile plicticoase. O veste intermediară ar fi că, în majoritate, problemele pe care le are de rezolvat un informatician presupun elemente din ambele categorii. De fapt, cei doi poli de care vorbim au o existenţă virtuală, fiind utili mai degrabă din raţiuni didactice & pedagogice.

Cert este că, încă de la începuturile sale, informatica a fost confruntată nu numai cu efectuarea de calcule sofisticate, ştiinţifice, dar şi cu stocarea şi gestionarea unui volum de informaţii din ce în ce mai mare. Astfel încât apariţia unor instrumente software dedicate gestiunii şi prelucrării datelor a fost doar o problemă de timp.

4.1.1 La început a fost fişierul

Prima care a resimţit acut nevoia unor instrumente software dedicate administrării unor cantităţi imense de informaţii a fost armata SUA. În a doua parte a anilor '50 Departamentul Apărării al SUA a format un grup de specialişti pentru elaborarea unui limbaj destinat aplicaţiilor administrative, în care dificultatea majoră ţinea de volumul imens de resurse materiale şi financiare ce trebuia "chivernisit" şi pentru care erau necesare rapoarte dintre cele mai diverse. La acel moment apăruse FLOWMATIC, limbaj precursor celui care a fost considerat câteva decenii regele informaticii economice – COBOL (Common Business Oriented Language).

Arhitectura aplicaţiilor de acest tip – specifică nu numai COBOL-ului, ci multor limbaje din a III-a generaţie, denumită *flat-files architecture* – tradusă, într-o doară, în româneşte drept *fişiere independente* – este reprezentată în figura 4.1. Specific acestui mod de lucru, referit ca *file-based* sau *flat files* (fişiere independente), este faptul că fiecare dată (Data1, Data2,... Data*n*) este descrisă (nume, tip, lungime), autonom, în toate fişierele în care apare. Mai mult, descrierea fiecărui fişier de date (câmpurile care-l alcătuiesc, tipul şi lungimea fiecăruia, modul de organizare (secvenţial, indexat, relativ etc.)) este obligatorie în toate programele care îl “citesc” sau modifică. Între FIŞIER1, FIŞIER2, ... FIŞIER*n* nu există nici o relaţie definită explicit.

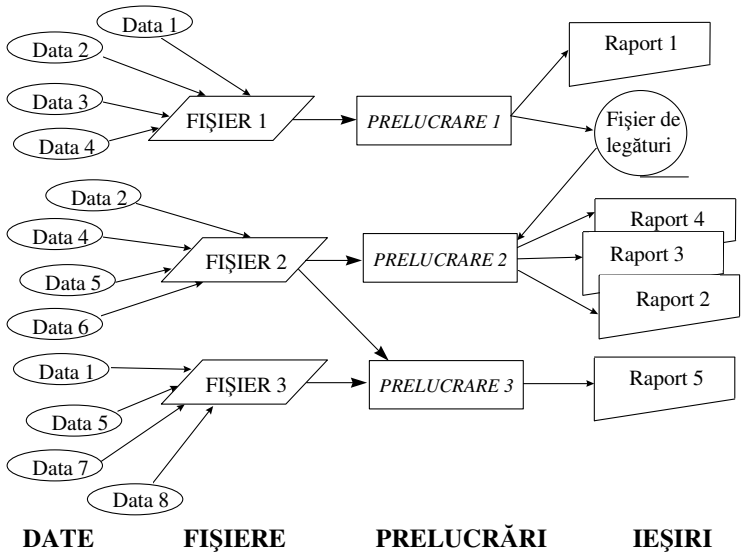

Figura 4.1. Sistem informatic bazat pe organizarea datelor în fişiere independente

Spre exemplu, Data2 este prezentă în două fişiere de date, FIŞIER1 şi FIŞIER2. Dacă, prin program, se modifică formatul sau valoarea acesteia în FIŞIER1, modificarea nu se face automat şi în FIŞIER2; prin urmare, o aceeaşi dată, Data2, va prezenta două valori diferite în cele două fişiere, iar necazurile bat la uşă: informaţiile furnizate de sistemul informatic sunt redundante şi prezintă un mare risc de pierdere a coerenţei.

Se poate proiecta un mecanism de menţinere a integrităţii datelor, astfel încât actualizarea unei date într-un fişier să atragă automat actualizarea tuturor fişierelor de date în care aceasta apare, însă, în sistemele mari, care gestionează volume uriaşe de informaţii, implementarea unui asemenea mecanism este extrem de complexă şi costisitoare. În plus, fişierele de date sunt uneori proiectate şi implementate la distanţe mari în timp, în formate diferite: de exemplu, FIŞIER1 este posibil să fi fost creat cu ajutorul limbajului COBOL, FIŞIER2 în FORTRAN iar FIŞIER3 în BASIC. În asemenea condiţii, punerea în operă a mecanismului de menţinere a integrităţii devine o utopie.

Chiar numai şi din cele prezentate mai sus, se pot desprinde câteva dezavantaje ale organizării datelor după modelul fişierelor independente:

Redundanţa şi inconsistenţa datelor: o aceeaşi dată apare în mai multe fişiere; în aceste cazuri există riscul modificării acesteia într-un fişier fără a face modificările şi în toate celelalte fişiere.

Dificultatea accesului. Într-o întreprindere, o aceeaşi informaţie este exploatată de mai mulţi utilizatori. Spre exemplu, pentru departamentul care se ocupă cu gestiunea stocurilor, intrările de materiale trebuie ordonate pe magazii (depozite) şi repere, în timp ce pentru departamentul care se ocupă cu decontările cu partenerii de afaceri ai întreprinderii, intrările trebuie ordonate pe furnizori ai materialelor. Or, fişierele tradiţionale nu facilitează accesarea datelor după mai multe criterii, specifice diferiţilor utilizatori sau grupuri de utilizatori.

Izolarea datelor: când datele sunt stocate în formate diferite, este dificil de scris programe care să realizeze accesul într-o manieră globală a tuturor celor implicate în derularea unei tranzacţii.

Complexitatea apăsătoare a actualizărilor. O actualizare presupune adăugarea, modificarea sau ştergerea unor informaţii din fişiere. Cum prelucrările se desfăşoară în timp real, de la mai multe terminale (în mediile multi-utilizator), pot apare situaţii conflictuale atunci când doi utilizatori doresc

modificarea simultană a unei aceleiaşi date. Rezolvarea acestui gen de conflicte presupune existenţa unui program-supervizor al prelucrărilor, care este greu de realizat cu o multitudine de fişiere, create la distanţă în timp şi, în formate diferite.

Problemele de *securitate* ţin de dificultatea creării unui mecanism care să protejeze pe deplin datele din fişiere de accesul neautorizat.

Probleme legate de *integritatea datelor*. Informaţiile stocate în fişiere sunt supuse la numeroase restricţii semantice. Toate aceste restricţii alcătuiesc mecanismul de integritate a datelor, deosebit de complex în mediile de lucru multi-utilizator şi eterogene.

Inabilitatea de a obţine răspunsuri rapide la probleme ad-hoc simple.

Costul ridicat se datorează gradului mare de redundanţă a datelor, eforturilor deosebite ce trebuie depuse pentru interconectarea diferitelor tipuri de fişiere de date şi pentru asigurarea funcţionării sistemului în condiţiile respectării unui nivel minim de integritate şi securitate a informaţiilor.

Inflexibilitatea faţă de schimbările ulterioare, ce sunt inerente oricărui sistem informaţional.

Modelarea inadecvată a lumii reale.

Aceste dezavantaje sunt mai mult decât convingătoare, încât vă puteţi întreba dacă au existat aşa inconştienţi care să-şi arunce banii pe apa... fişierelor independente. Ei bine, o serie de aplicaţii dezvoltate în anii '60 sau '70 au fost moştenite şi folosite până în zilele noastre. De ce ? Datorită consistentelor sume investite, care au putut fi amortizate (trecute pe costuri) doar în ani buni, chiar decenii. Un alt motiv a fost însă funcţionalitatea şi viteza unor asemenea aplicaţii. Şi, nu în ultimul rând, un mediu economic relativ "aşezat" (vă daţi seama că nu de mediul *nostru* economic este vorba), în care nu au fost necesare modificări majore ale procedurilor şi funcţiilor esenţiale.

4.1.2 Baze de date

Sintagma *bază de date* apare pentru prima dată în titlul unei conferinţe organizate la Santa Monica (California) în 1964 de System Development Corporation. Consacrarea definitivă a termenului este marcată de publicarea în anul 1969, de către CODASYL, în cadrul unei conferinţe dedicate limbajelor de gestiune a datelor, a primului raport tehnic în care este prezentat conceptul de bază de date. Faţă de modelul fişierelor independente, noutatea o constituie existenţa unui *fişier de descriere globală* a bazei, astfel încât să se poată asigura independenţa programelor faţă de date, după cum o arată şi figura 4.2.

Avantajele organizării informaţiilor în baze de date decurg tocmai din existenţa acestui fişier de descriere globală a bazei, denumit, în general, *dicţionar de date* (alte titulaturi: *repertoar* de date sau *catalog de sistem*). Extragerea şi modificarea datelor, altfel spus, lucrul cu fişierele de date, se derulează exclusiv prin intermediul dicţionarului în care se găsesc informaţii privitoare la structura datelor şi restricţiile îndeplinite de acestea.

O bază de date (BD) reprezintă un ansamblu structurat de fişiere, care grupează datele prelucrate în aplicaţiile informatice ale unei persoane, grup de persoane, întreprinderi, instituţii etc. Formal, *BD poate fi definită ca o colecţie de date aflate în interdependenţă, împreună cu descrierea datelor şi a relaţiilor dintre ele, sau ca o colecţie de date utilizată într-o organizaţie, colecţie care este automatizată, partajată, definită riguros (formalizată) şi controlată la nivel central*.

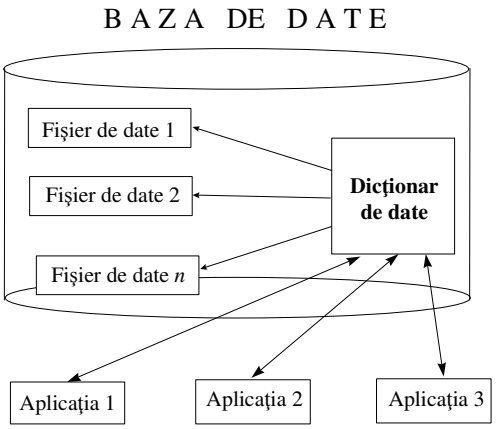

Figura 4.2. Schemă de principiu a unei baze de date

Atunci când vorbim despre o bază de date, trebuie avute în vedere două aspecte fundamentale aceste acesteia, *schema* şi *conţinutul*. Organizarea bazei de date se reflectă în *schema* sau *structura* sa, ce reprezintă un ansamblu de instrumente pentru descrierea datelor, a relaţiilor dintre acestea, a semanticii lor şi a restricţiilor la care sunt supuse. Ansamblul informaţiilor stocate în bază la un moment dat constituie *conţinutul* sau *instanţierea* sau *realizarea* acesteia. În timp ce volumul prezintă o evoluţie spectaculoasă în timp, schema unei baze rămâne relativ constantă pe tot parcursul utilizării acesteia.

Într-un sistem informatic ce utilizează BD, organizarea datelor poate fi analizată din mai multe puncte de vedere şi pe diferite paliere. De obicei, abordarea se face pe trei nivele: *fizic* sau *intern*, *conceptual* sau *global* şi *extern*.

Nivelul fizic (sau *intern*). Reprezintă modalitatea efectivă în care acestea sunt "scrise" pe suportul de stocare - disc magnetic, disc optic, bandă magnetică etc.

Nivelul conceptual (sau *global*). Este nivelul imediat superior celui fizic, datele fiind privite prin prisma semanticii lor; interesează conţinutul lor efectiv, ca şi relaţiile care le leagă de alte date. Reprezintă primul nivel de abstractizare a lumii reale observate. Obiectivul acestui nivel îl constituie modelarea realităţii considerate, asigurându-se independenţa bazei faţă de orice restricţie tehnologică sau echipament anume. Toţi utilizatorii îşi exprimă nevoile de date la nivel conceptual, prezentându-le administratorului bazei de date, acesta fiind cel care are o viziune globală necesară satisfacerii tuturor cerinţelor informaţionale.

Nivelul extern. Este ultimul nivel de abstractizare la care poate fi descrisă o bază de date. Structurile de la nivelul conceptual sunt relativ simple, însă volumul lor poate fi deconcertant. Iar dacă la nivel conceptual baza de date este abordată în ansamblul ei, în practică, un utilizator sau un grup de utilizatori lucrează numai cu o porţiune specifică a bazei, în funcţie de departamentul în care îşi desfăşoară activitatea şi de atribuţiile sale (lor). Simplificarea interacţiunii utilizatori-bază, precum şi creşterea securităţii bazei, sunt deziderate ale unui nivel superior de abstractizare, care este nivelul extern. Astfel, structura BD se prezintă sub diferite machete, referite, uneori şi ca sub-scheme, scheme externe sau imagini, în funcţie de nevoile fiecărui utilizator sau grup de utilizatori.

Luând în considerare cele trei nivele de abstractizare, schematizarea unui sistem de lucru cu o bază de date se poate face ca în figura 4.3.

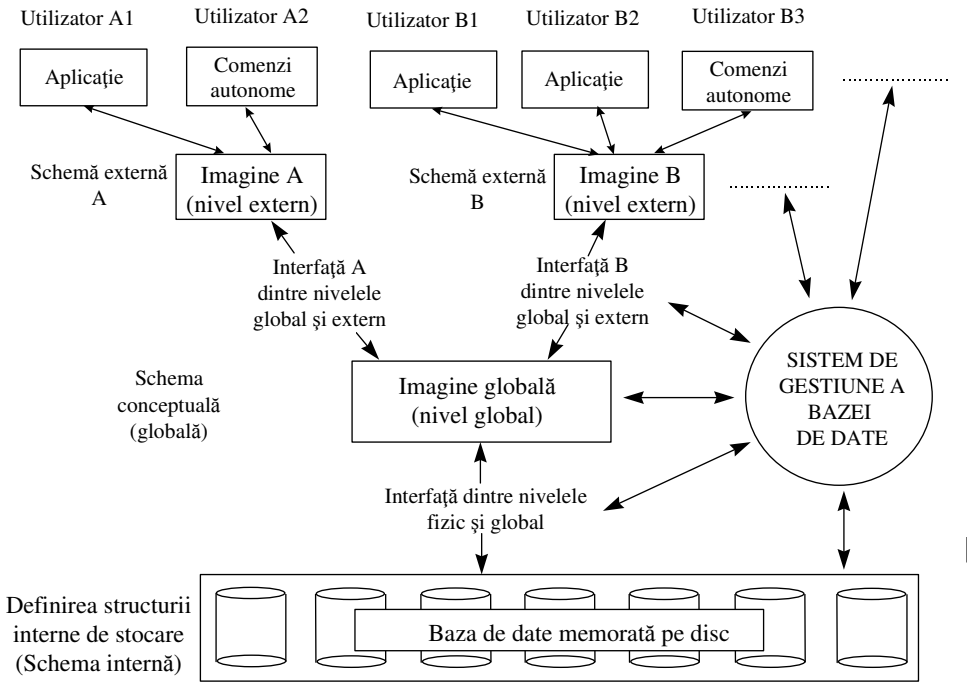

Figura 4.3. Schematizare a unui sistem de lucru cu o bază de date

Posibilitatea modificării structurii la un nivel, fără a afecta structura nivelului sau nivelelor superioare, se numeşte *autonomie* a datelor stocate în bază, analizabilă pe două paliere.

Autonomia fizică reprezintă posibilitatea modificării arhitecturii bazei la nivel intern, fără ca aceasta să necesite schimbarea schemei conceptuale şi rescrierea programelor pentru exploatarea bazei de date. Asemenea modificări sunt necesare uneori pentru ameliorarea performanţelor de lucru (viteză de acces, mărimea fişierelor etc.). Tot autonomia fizică este cea care asigură portarea bazei de date de pe un sistem de calcul pe altul fără modificarea schemei conceptuale şi a programelor.

Autonomia logică presupune posibilitatea modificării schemei conceptuale a bazei (modificare datorată necesităţii rezolvării unor noi cerinţe informaţionale) fără a rescrie programele de exploatare. Autonomia logică a datelor este mai greu de realizat decât autonomia fizică, deoarece programele de exploatare sunt dependente, în foarte mare măsură, de structura logică a datelor pe care le consultă şi actualizează, în ciuda existenţei dicţionarului de date. Fireşte, un element important îl reprezintă şi anvergura modificării schemei conceptuale.

Datele stocate într-o BD prezintă, într-o măsură mai mare sau mai mică, următoarele caracteristici:

- *partajabilitate* – disponibilitate pentru un mare număr de utilizatori şi aplicaţii;
- *persistenţă* – existenţă permanentă, din momentul preluării în bază până în momentul actualizării sau ştergerii;
- *securitate* – protejarea de accesul neautorizat, atât în ceea ce priveşte citirea şi copierea, cât şi modificarea şi ştergerea;

- *validitate* – referită şi ca integritate sau corectitudine – priveşte gradul de adecvare dintre datele din bază şi realitatea, procesele pe care le reflectă aceste date;
- *consistenţă* – ori de câte ori diverse aspecte ale proceselor sau fenomenelor reale sunt preluate în bază sub forma a doua sau mai multor entităţi sau atribute, aceste entităţi /atribute trebuie să fie în concordanţă unele cu celelalte, să respecte relaţiile existente între aspectele proceselor /fenomenelor reale;
- *nonredundanţă* - pe cât posibil, o entitate din realitate ar trebui să aibă un singur corespondent în baza de date;
- *independenţă* – priveşte autonomia logică şi fizică evocate mai sus.

4.1.3 Modele de organizare a datelor în baze

Nucleul unei baze de date îl reprezintă dicţionarul de date ce conţine structura bazei, structură care se materializează prin instrucţiuni scrise cu ajutorul unui limbaj de definire a datelor (DDL). Analiza, proiectarea şi implementarea structurii (schemei) bazei se realizează utilizând un model de date. Un asemenea model reprezintă un ansamblu de instrumente conceptuale care permit descrierea datelor, relaţiilor dintre ele, a semanticii lor, ca şi a restricţiilor la care sunt supuse.

Modelul datelor este o reprezentare a obiectelor lumii reale şi a evenimentelor asociate lor. Este o abstractizare care se concentrează pe aspectele esenţiale ale organizaţiei /aplicaţiei, furnizând conceptele de bază şi notaţiile care vor permite utilizatorilor bazelor de date să comunice clar şi rapid informaţiile şi cunoştinţele lor despre datele organizaţiei.

O grupare "tradiţională" a modelelor utilizate în bazele de date delimitează trei categorii: modele logice bazate pe obiect, modele logice bazate pe înregistrare şi modele fizice. Din punctul nostru de vedere, interesează numai nivelele conceptual şi extern de abstractizare a datelor; de aceea, vom prezenta, în linii mari, numai reprezentanţii principali ai primelor două categorii.

Modelul ierarhic. Primele produse-software (Sisteme de Gestiune a Bazelor de Date - SGBD) lucrau cu baze de date ierarhice. Structura datelor este prezentată sub forma unui arbore, partea superioară a arborelui fiind rădăcina (arborele este văzut "cu verdele în jos"). Un nod-tată poate avea mai multe noduri-fii. Un fiu nu poate exista independent de tatăl său. Legătura (reprezentată prin linie) se face exclusiv între tată şi fii.

Modelul reţea. Este o dezvoltare a modelului ierarhic, prin care se pot reprezenta şi situaţiile în care un fiu "posedă" mai mulţi taţi. Înregistrările sunt privite în BD ca o colecţie de grafuri.

Modelul relaţional a fost următorul în ordinea cronologică şi rămâne cel care domină copios piaţa bazelor de date şi la acest moment, motiv pentru care cea mai mare parte a acestui curs îi este dedicată.

Modelul obiectual. Începând cu anii ’60, în programare şi, ceva mai târziu, în analiză şi proiectare, orientarea pe obiecte (OO) a avut un succes uriaş, reuşind să depăşească metodologiile structurate. Pe baza acestui succes, s-a crezut că şi în materie de baze de date, modelul obiectual îl va surclasa pe cel relaţional. Rezultatele sunt însă deprimante pentru suporterii OO-ului, piaţa SGBD OO fiind sub 6% din valoarea totală a pieţei bazelor de date.

Modelul relaţional-obiectual. Este un model mai recent ce încearcă să valorifice deopotrivă atuurile relaţionalului cu orientarea pe obiecte. Deşi privit mai degrabă cu neîncredere în cercurile

teoreticienilor, acest model se impune încet-încet datorită marilor producători de software dedicat bazelor de date.

4.2 PREZENTARE GENERALĂ A MODELULUI RELAŢIONAL

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 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).

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.

În acest paragraf vom discuta aspectele structurale ale modelului (paragraful 4.2.2) şi pe cele de integritate (paragraful 4.3), urmând ca în paragraful 5.4 şi mai ales în capitolul 6 să ne ocupăm de manipularea datelor, altfel spus, de modalităţile în care "stoarcem" de informaţii baza de date.

4.2.1 Puţină istorie

Modelul relaţional 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 reţea, modele ce depind într-o mai mare măsură de organizarea internă a datelor pe suportul de stocare. Codd a propus o structură de date tabelară, independentă de tipul de echipamente şi software de sistem pe care este implementată, structură "înzestrată" cu o serie de operatori pentru extragerea datelor. Deşi puternic matematizat, modelul relaţional este relativ uşor de înţeles.

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.). Modelarea realităţii se concretizează în tabele de valori numite relaţii, avându-se în vedere că:

- o relaţie are un nume;
- o coloană reprezintă un atribut;
- o linie reprezintă un n-uplet (tuplu) de valori ale celor *n* atribute din relaţie;
- ordinea liniilor şi coloanelor în cadrul tabelei nu este relevantă pentru conţinutul informaţional.

Fiecare linie a tabelei reprezintă o entitate sau un fapt al realităţii, în timp ce o coloană reprezintă o proprietate a acestei entităţi sau fapt. Introducând un plus de rigoare, se cuvine de remarcat că entitatea poate fi nu numai un obiect concret/proces (o persoană, un lucru oarecare), dar şi o relaţie între obiecte (persoane, lucruri), cum ar fi: contractele de afaceri, căsătoriile (există totuşi câteva deosebiri între acestea şi precedentele), structura ierarhică a unei organizaţii etc. În plus, nu e întotdeauna evident care dintre informaţii sunt entităţi, care atribute şi care asociaţii (relaţii).

[1] Extrase din lucrarea [Codd70] se găsesc la adresa http://www.acm.org/classics/nov95/. De asemenea, o excelentă analiză a articolelor lui Codd este în [Date98].

Ansamblul valorilor stocate în tabelele reprezintă conţinutul bazei de date, conţinut ce poate fi modificat prin operaţiuni de actualizare: introducerea unor linii (tupluri) noi, ştergerea unor linii, modificarea valorii unor atribute.

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;
- ia în calcul o largă varietate de aplicaţii;
- abordează metodologic definirea structurii bazei de date.

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.

Deşi IBM a fost prima care a iniţial un proiect destinat elaborării unui SGBD relaţional (System/R, începând cu 1974), prima firmă care a lansat primul SGBDR comercial a fost Relational Software Inc., astăzi Oracle. Infiinţarea firmei a avut loc în 1977, iar lansarea produsului în 1979. Impunerea pe piaţă a SGBD-urilor grefate pe modelul relaţional a fost mult mai dificilă decât s-ar putea înţelege astăzi. Abia în deceniul 9, datorită vitezei ameliorate, securităţii sporite şi mai ales productivităţii dezvoltatorilor de aplicaţii, modelul relaţional a câştigat supremaţia.

Există o largă tipologie a SGBDR-urilor, aşa încât orice categorisire îşi are riscul său. Cele mai accesibile şi, implicit, cele mai utilizate sunt SGBD-urile dedicate iniţial uzului individual: Access, Paradox, Visual FoxPro. Astăzi, multe dintre acestea au caracteristici profesionale şi pot fi folosite la dezvoltarea aplicaţiilor pentru simultan a zeci de utilizatori.

Pentru aplicaţiile complexe din bănci, corporaţii, organizaţii şi instituţii de mari dimensiuni s-au impus SGBDR-urile de "categoria grea": Oracle, DB2 (IBM), Informix (DB2), Sybase, SQL Server (Microsoft). Acestea sunt mult mai robuste, mult mai fiabile, dar şi costisitoare.

În ultimul timp şi-au făcut apariţia, ca alternative la marii coloşi, aşa zisele *Free-DBMS*-uri, precum PostgreSQL, MySQL, Interbase etc. Acestea rulează, de obicei, pe sisteme de operare de tip Linux (foarte ieftine) şi se întrevăd ca adversari serioşi ai marilor producători.

4.2.2. Relaţii/tabele, domenii şi atribute

La modul simplist, o bază de date relaţională (BDR) poate fi definită ca un ansamblu de relaţii (tabele); fiecare tabelă (sau tabel), alcătuită din linii (tupluri), are un nume unic şi este stocată pe suport extern (de obicei disc). La intersecţia unei linii cu o coloană se găseşte o valoare atomică (elementară). O relaţie conţine informaţii omogene legate de anumite entităţi, procese, fenomene: CĂRŢI, STUDENŢI, LOCALITĂŢI, PERSONAL, FACTURI etc. Spre exemplu, în figura 4.4 este reprezentată tabela CLIENŢI.

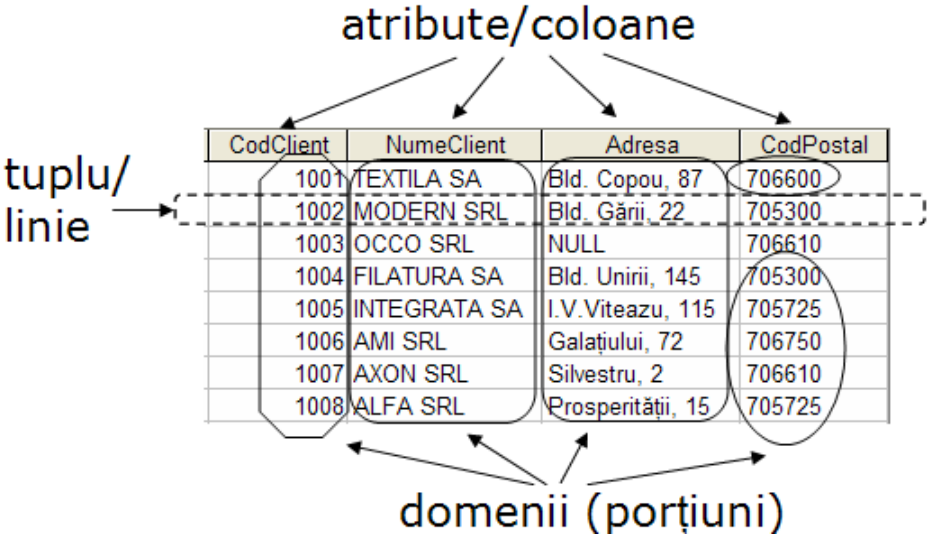

Figura 4.4. Relaţia (tabela) CLIENŢI

În teoria relaţională se foloseşte termenul *relaţie*. 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 regrupează informaţii referitoare la un obiect, eveniment etc., altfel spus, informaţii referitoare la o entitate: o carte (un titlu sau un exemplar din depozitul unei biblioteci, depinde de circumstanţe), un/o student(ă), o localitate (oraş sau comună), un angajat al firmei, o factură emisă etc. Figura 4.5 conţine al doilea tuplu din tabela CLIENŢI, tuplu referitor la forma MODERN SRL. Linia de mai jos este alcătuită din patru valori ce desemnează: codul, numele, adresa şi codul poştal al localităţii şi adresei referitoare la clientul MODERN SRL.

1002	MODERN SRL	Bld. Gării, 22	705300

Figura 4.5. Un tuplu al tabelei CLIENŢI

Teoretic, *orice tuplu reprezintă o relaţie între clase de valori* (în cazul nostru, între patru clase de valori); de aici provine sintagma *baze de date relaţionale*, în sensul matematic al relaţiei, de asociere a două sau mai multe elemente. Firește, toate tuplurile relaţiei au acelaşi format (structură), ceea ce înseamnă că în tabela CLIENŢI, din care a fost extrasă linia din figura 4.5, 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 importanţă din punctul de vedere al conţinutul informaţional 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 relaţiei:

- într-o tabelă destinată datelor generale ale angajaţilor, pentru atributul **Sex**, domeniul este alcătuit din două valori: *Femeiesc* şi *Bărbătesc*;
- domeniul atributului **Judet** este alcătuit din numele fiecărui judeţ (plus Bucureşti).
- domeniul unui atribut precum **PreţUnitar**, care se referă la preţul la care a fost vândut un produs/serviciu, este cu mult mai larg, fiind alcătuit din orice valoarea cuprinsă între 1 şi 99999999 lei (ceva mai noi).

Şi acum, câteva senzaţii tari (urmează un pic de matematică) ! Dacă notăm cu D1 domeniul atributului **CodClient**, cu D2 domeniul atributului **NumeClient**, cu D3 domeniul pentru **Adresa**, şi cu D4 domeniul atributul **CodPostal**, se poate spune că fiecare linie a tabelei CLIENŢI este un tuplu de patru elemente, iar relaţia în ansamblu corespunde unui subansamblu din ansamblul tuturor tuplurilor posibile alcătuite din patru elemente, ansamblu care este produsul cartezian al celor patru domenii.
În general, orice relaţie R poate fi definită ca un subansamblu al produsului cartezian de *n* domenii D_i:

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

n fiind denumit gradul sau ordinul relaţiei. Relaţiile de grad 1 sunt *unare*, cele de grad 2 - *binare*, ..., cele de grad n - *n-are*. Această definiţie pune în evidenţă aspectul constant al relaţiei, de independenţă în timp (se spune că în acest caz relaţia este definită ca un predicat).

O a doua definiţie abordează o relaţie R ca un *ansamblu* de *m*-uplete (*m*-tupluri) de valori: $R = \{ t_1, t_2, \ldots, t_k, \ldots, t_m\}$, 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 4.6.

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 4.6. Ilustrarea celei de-a doua definiţii a unei relaţii

Reprezentarea sub formă de tabelă, deci ca ansamblu de tupluri, pune în evidenţă aspectul dinamic, variabil al relaţiei. Abordarea predicativă (prima definiţie) sau ansamblistă (a doua definiţie) reprezintă un criteriu important de delimitare a limbajelor de interogare a bazelor de date relaţionale.

Reţinem corespondenţa noţiunilor *relaţie-tabelă*, *tuplu-linie* şi *atribut-coloană.*

Numărul de tabele pe care le conţine o bază de date, atributele “adunate” în fiecare tabelă, domeniul fiecăruia dintre atribute prezintă diferenţe majore de la o bază la alta, chiar dacă uneori reflectă acelaşi tip de procese. Intrăm astfel în sfera proiectării bazelor de date, a dependenţelor şi normalizării.

Relaţia CLIENŢI conţine informaţii 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 4.4 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, informaţia este necunoscută sau inaplicabilă. Valoarea *NULL* este diferită, însă de valorile 0 sau spaţiu. Uneori, importanţa sa este, din păcate, majoră în expresii şi funcţii, după cum ştiu cei cu oarecare experienţă în limbajului SQL.

4.3 RESTRICŢII ALE BAZEI DE DATE

De ce ne interesează restricţiile într-o bază de date? Termenul de *restricţie* 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, restricţiile 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 sistemul de gestiune a bazelor de date (adică programele de lucru cu bazele de date). Esenţial este că, ajutaţi de restricţii, putem creşte gradul de corectitudine şi de încredere al datelor din bază. În cele ce urmează vor fi prezentate pe scurt cele mai importante restricţii definibile într-o bază de date relaţională: restricţia de domeniu, de atomicitate, de unicitate, referenţială şi restricţiile-utilizator.

4.3.1 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 din 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 la care vor fi legate atributele în momentul creării tabelelor.

4.3.2 Atomicitate

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

Astăzi, atomicitatea valorii atributelor a devenit o ţintă predilectă a "atacurilor duşmănoase" la adresa modelului relaţional, datorită imposibilităţii înglobării unor structuri de date mai complexe, specifice unor domenii ca: proiectare asistată de calculator, baze de date multimedia etc. Mulţi autori, dintre care merită amintiţi cu deosebire Chris J. Date şi Hugh Darwen, se opun ideii de atomicitate formulată de Codd.

Primele vizate de "stigmatul" neatomicităţii sunt atributele compuse. Un exemplu de atribut compus (non-atomic) este Adresa. Fiind alcătuită din **Stradă**, **Număr**, **Bloc**, **Scară**, **Etaj**, **Apartament**, discuţia 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 poliţie, 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 producţie 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 funcţie de circumstanţe, simple sau compuse: **DataOperaţiuniiBancare** (**Data** + **Ora**), **BuletinIdentitate** (**Seria**+**Număr**), **NrÎnmatriculareAuto** (privit global, sau pe cele trei componente: număr, judeţ, combinaţie trei de litere).

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

4.3.3 Nenulitate

Modelul relaţional acceptă 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 foloseacă (meta)valoare NULL. Celui de-al treilea client din figura 4.4 nu i 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 cafea, vodcă, 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 întreba !).

Într-o bază de date relaţională avem posibilitatea de a le impune unor atribute să aibă întotdeauna valori specificate, altfel spus, le interzicem valorile nule, în timp cel altor atribute li se pot permite valori nule. Modul în care se declară şi se afişează o restricţie de nulitate depinde de la SGBD la SGBD. De exemplu, în Access mecanismul este de tipul celui din figura 4.7. Atributului NumeClient din tabela CLIENŢI i se interzic valorile nule, prin setarea opţiunii **Required** pe valoarea

Yes, în timp ce atributul Adresa din aceeaşi tabelă poate avea valori NULL, **Required** fiind setat pe valoarea *No*.

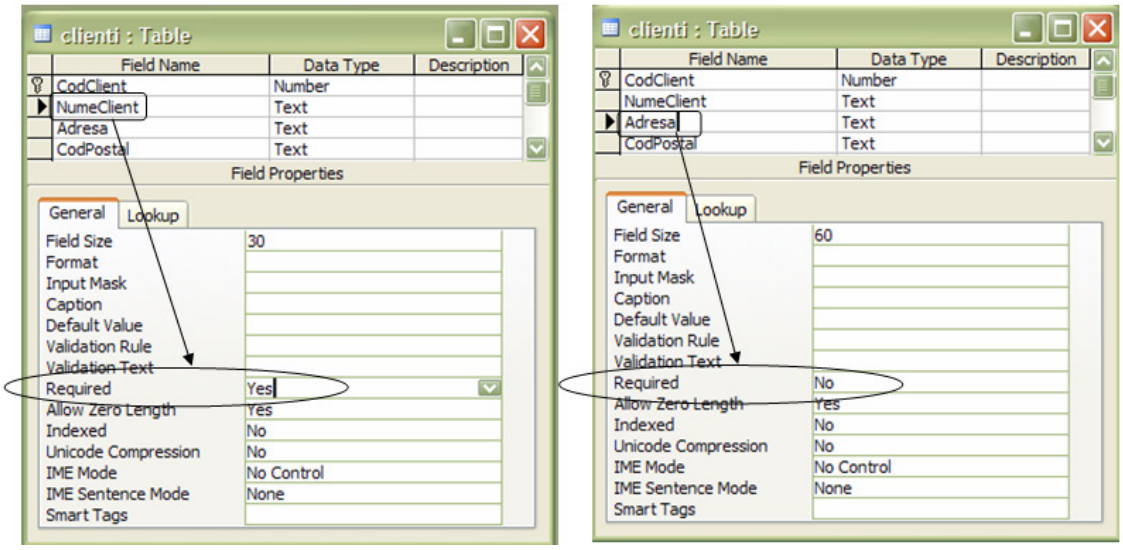

Figura 4.7. Un atribut "forţat" să nu primească valori NULL şi un altul "relaxat"

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.

4.3.4 Restricţia de unicitate

Într-o relaţie nu pot exista 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 compoziţia cheii primare nu poate avea valori nule; această a treia condiţie se mai numeşte şi *restricţie a entităţii*.

Domeniul unui atribut care este cheie primară într-o relaţie este denumit domeniu primar. Dacă într-o relaţie există mai multe combinaţii de atribute care conferă unicitate tuplului, acestea sunt

denumite *chei candidate*. O cheie candidată care nu este identificator primar este referită ca şi *cheie alternativă*.

În tabela CLIENŢI cheia primară este simplă - **CodClient**, **CodClient** reprezintă un număr unic asociat fiecărei firme căreia i-am făcut vânzări. Există însă suficiente cazuri în care cheia primară este compusă din două, trei s.a.m.d. sau, la extrem, toate atributele relaţiei. Să luăm spre analiză o relaţie PERSONAL care conţine date generale despre angajaţii firmei. Fiecare tuplu al relaţiei 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 angajaţi cu acelaşi nume.

- dacă apariţia a două persoane cu nume identice este posibilă, atunci apariţia a două persoane cu acelaşi Prenume este probabilă.

- nici unul din aceste atributele DataNaşterii, Vechime, SalariuTarifar nu poate fi "înzestrat" cu funcţiunea de identificator.

În acest caz, se încearcă gruparea a două, trei, patru s.a.m.d. atribute, până când se obţine combinaţia care va permite diferenţierea clară a oricărei linii de toate 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 "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 angajai 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, 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.

Revenim la tabela CLIENŢI. Dacă în ţara noastră s-ar respecta regula: *nu pot exista două firme cu aceeaşi denumire*, atunci tabela mai are, pe lângă CodClient, o cheie candidat – NumeClient. Întrucât CodClient a fost preferată drept cheie primară, NumeClient va fi cheie alternativă. În Access un atribut de tip cheie primară are în dreptul său o cheie (de yală) – vezi figura 4.8. Cheile alternative sunt declarate prin setarea opţiunii *Indexed* pe valoarea *Yes (NoDuplicates)*.

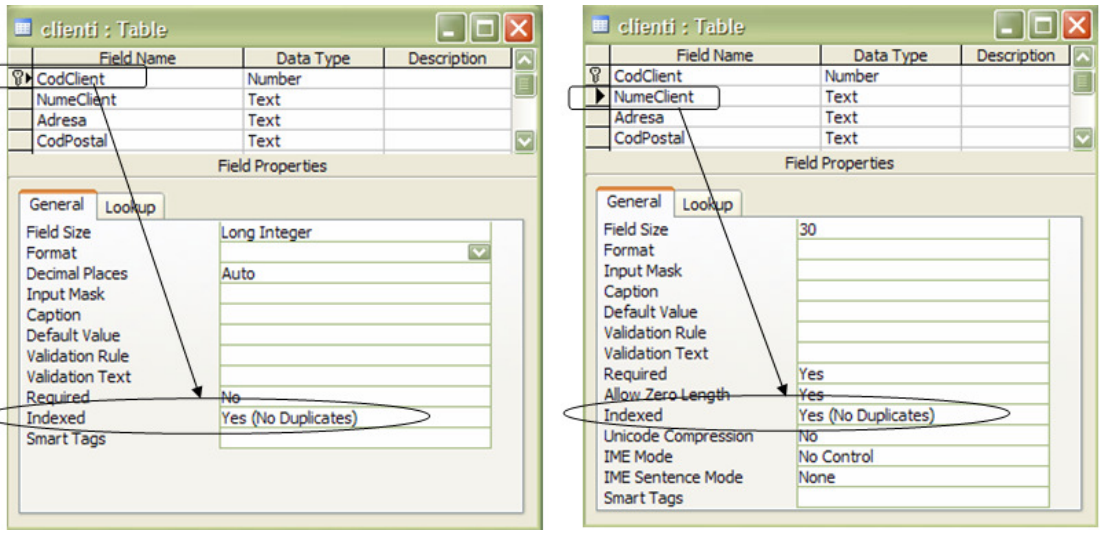

Figura 4.8. O cheie primară şi o cheie alternativă

4.3.5 Restricţia referenţială

O bază de date relaţională este alcătuită din relaţii (tabele) aflate în legătură. Stabilirea legăturii se bazează pe mecanismul cheii străine şi, implicit, a restricţiei referenţiale. Figura 5.5 prezintă o relaţie în care sunt implicate tabelele FACTURI şi CLIENŢI.

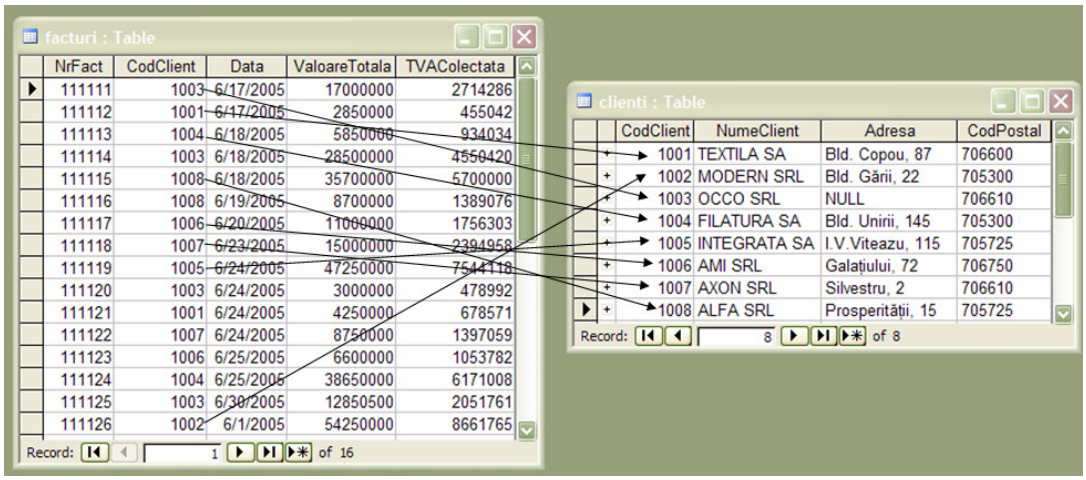

Figura 4.9. O restricţie referenţială între FACTURI (tabelă-copil) şi CLIENŢI (părinte)

Atributul **CodClient** joacă un rol de "agent de legătură" între tabelele CLIENŢI şi FACTURI. Pentru relaţia CLIENŢI, atributul **CodClient** este cheie primară, în timp ce în tabela FACTURI, **CodClient** reprezintă coloana de referinţă sau cheia străină, deoarece numai pe baza valorilor sale se poate face legătura cu relaţia părinte CLIENŢI.

Cheile străine sau coloanele de referinţă sunt deci atribute sau combinaţii de atribute care pun în legătură linii (tupluri) din relaţii diferite. Tabela în care atributul de legătură este primară se numeşte *tabelă-părinte* (în cazul nostru, CLIENŢI), iar cealaltă *tabelă-copil*.

Legat de noţiunea de cheie străină apare conceptul de restricţie referenţială. O restricţie de integritate referenţială apare atunci când o relaţie face referinţă la o altă relaţie. Când două tabele (relaţii), T1 şi T2, prezintă 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 client, a denumirii localităţii şi a judeţului în care-şi are sediul. Dacă în FATURI 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

1. 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 prin intermediul cheilor străine. Aceasta este, de fapt, a doua accepţiune a termenului de BDR, prima, cea "clasică", având în vedere percepţia fiecărei linii dintr-o tabelă ca o relaţie între clase de valori.
2. 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.
3. Respectarea restricţiilor referenţiale este una din cele mai complicate sarcini pentru dezvoltatorii de aplicaţii ce utilizează baze de date. Din acest punct de vedere, tentaţia este de a “sparge” baza de date în cât mai puţine tabele cu putinţă, altfel spus, de a avea relaţii cât mai “corpolente”. Gradul de fragmentare al bazei ţine de *normalizarea* bazei de date, care, ca parte a procesului de proiectare a BD, se bazează pe dependenţele funcţionale, multivaloare şi de joncţiune între atribute.

4.3.6 Restricţii-utilizator

Restricţiile utilizator mai sunt denumite şi restricţii de comportament sau restricţii ale organizaţiei. De obicei, aceste restricţii iau forma unor reguli de validare la nivel de atribut, la nivel de linie/tabelă sau a unor reguli implementate prin declanşatoare (triggere).

Reguli la nivel de atribut

O restricţie la nivel de atribut poate preveni introducerea în baza de date a unor valori din alte intervale decât cele stabilite, în alte formate decât cele acceptate etc. Forma clasică a unei restricţii la nivel de atribut este o expresie în care apar constante, funcţii-sistem şi, nu în ultimul rând, atributul respectiv. La orice editare a atributului cu pricina (declanşată în cazul inserării unei linii în tabela din care face parte, sau la modificarea sa) expresia este evaluată şi dacă rezultatul evaluării este TRUE (adevărat), atunci inserarea/modificarea este permisă, iar dacă rezultatul este FALSE, atunci inserarea/modificarea este blocată.

În partea stângă a figurii 4.10 este ilustrată o regulă de validare conform căreia în tabela CLIENŢI valorile atributului CodClient trebuie să fie mai mari decât 1000. Rubrica **Validation Rule** este cea în care apare expresia-restricţie - *[CodClient]>1000* - (observaţi că numele atributului este scris între paranteze unghiulare), iar în rubrica **Validation Text** se indică mesajul care va apărea pe ecran atunci când se încalcă restricţia - *Cel mai mic cod de client acceptat este 1001 !*.

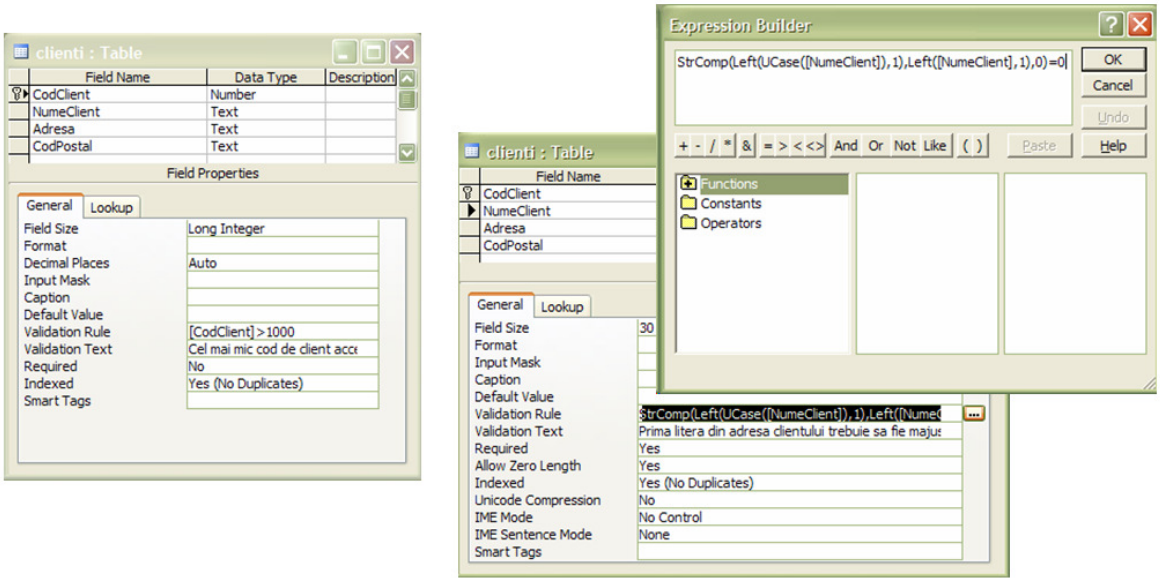

Figura 4.10. Două restricţii la nivel de atribut

În dreapta figurii apare o regulă ceva mai impresionantă, prin care literele din valorile atributului NumeClient sunt obligatoriu majuscule. Expresia este de-a dreptul impresionantă - *StrComp(Left(UCase([NumeClient]),1),Left([NumeClient],1),0) = 0* – şi nu vom strica vraja până în capitolul următor.

Reguli la nivel de înregistrare

Expresia care defineşte o restricţie la nivel de înregistrare poate conţine două sau mai multe atribute şi este evaluată la inserarea sau modificarea oricărei linii din tabelă. Tabela FACTURI conţine, printre altele două atribute "valorice", unul pentru păstrarea valorii totale (inclusiv TVA) facturii respective şi un altul care indică "doar" cuantumul TVA colectate pentru factură. Majoritatea produselor şi serviciilor comercializate la noi în ţară au un procent al taxei pe valoarea adăugată de 19%. Există, însă, şi produse la care procentul poate fi 9% sau chiar scutite de TVA (0%). De aceea, TVA colectată pentru o factură poate fi egală sau mai mică decât 19% din valoarea *fără* tva. Să punem sub formă de formulă: ValoareaTotală = ValoareaFărăTVA + TVA. Dacă factura conţine 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 diferenţa 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.

Prin urmare, regula este TVAColectată <= ValoareTotală * 0.19 / 1.19, iar în Access declararea sa este ilustrată în figura 4.11.

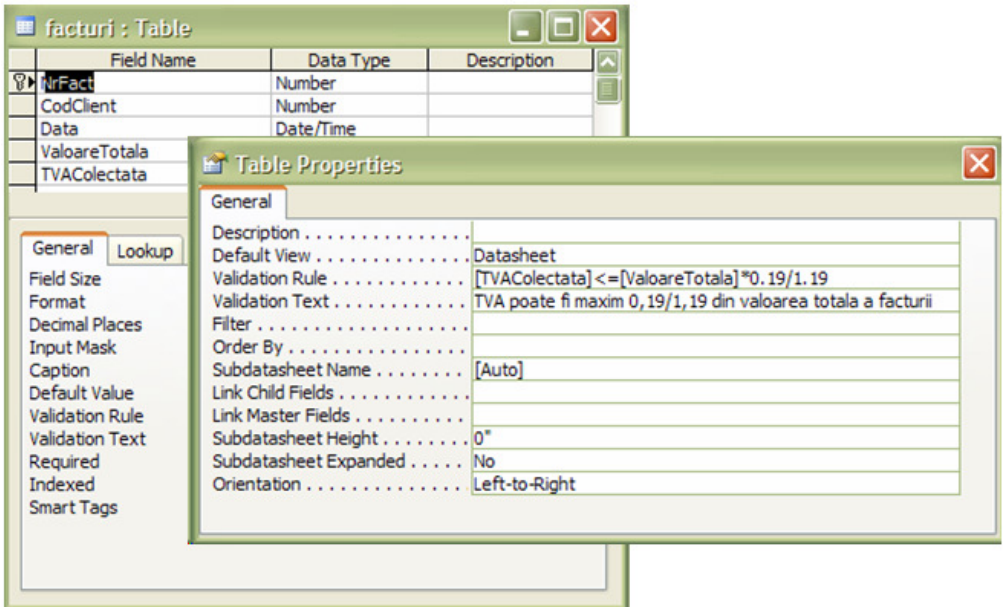

Figura 4.11. O restricţie la nivel de înregistrare

Alte tipuri de restricţii utilizator

Există şi alte tipuri de restricţii a căror validare presupune "citirea" unor date aflate în tabele diferite. Spre exemplu, se poate institui o regulă care interzice emiterea unei noi facturi (o nouă vânzare) dacă datoriile firmei client sunt mai mari de 200000 RON, iar directorul acesteia nu este membru în partidul/partidele de guvernământ. Acestea reclamă întrebuinţarea unor proceduri speciale, numite declanşatoare (triggere). Curioşii n-au decât să urmeze specializarea Informatică Economică.

4.4 SCHEMA ŞI CONŢINUTUL UNEI BAZE DE DATE RELAŢIONALE

Există două aspecte complementare de abordare a bazelor de date relaţionale: *schema* (structura, intensia) şi *conţinutul* (instanţierea, extensia).

Conţinutul unei relaţii este reprezentat de ansamblul tuplurilor ce o alcătuiesc la un moment dat. Pe parcursul exploatării bazei, conţinutul poate creşte exponenţial, în funcţie de volumul şi complexitatea operaţiunilor consemnate.

Schema unei baze de date conţine denumiri ale tabelelor, numele, tipul şi lungimea atributelor, restricţii de unicitate, de non-nulitate, restricţii la nivel de atribut, linie şi alte eventuale tipuri de restricţii de comportament, precum şi restricţii referenţiale. La acestea se adaugă cele privind drepturile utilizatorilor, definiţia şi restricţiile tabelelor virtuale, indecşi etc. Foarte importante în lumea profesioniştilor dezvoltării de aplicaţii cu bazele de date sunt procedurile stocate – programe de forma funcţiilor, procedurilor, pachetelor şi mai ales declanşatoarelor (triggerelor) – care, după cum le spune şi numele sunt memorate în schema bazei de date şi fac parte integrantă din aceasta. Schema este independentă de timp şi reprezintă componenta permanentă a relaţiilor.

Atenţie ! În literatura de popularizare a bazelor de date (mai ales a Accessului) apar şi o serie de bazaconii, precum că rapoartele, formularele, meniurile ar fi incluse în schema unei baze de date.

Nu este adevărat ! În Visual FoxPro, Accesss şi alte SGBD-uri, pentru a gestiona mai uşor o aplicaţie se creează un *project* în care sunt incluse bazele de date (tabele, tabele virtuale, proceduri stocate), interogări, formulare, rapoarte, meniuri etc. Ori, confuzia care se face uneori este între proiect şi bază de date.

În capitolele care vor urma, va fi utilizată cu precădere o bază de date "martor", denumită VÎNZĂRI, cu schema din figura 4.12.

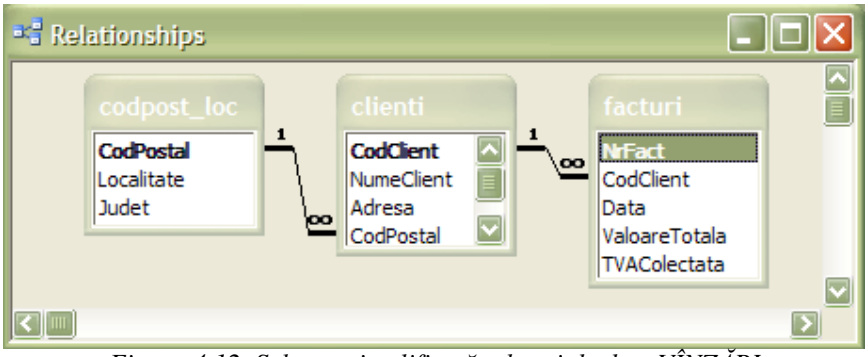

Figura 4.12. Schema simplificată a bazei de date VÎNZĂRI

Conţinutul celor trei tabele, ce vor constitui suportul editărilor şi interogărilor din capitolele următoare sunt prezentate în figura 4.13.

codPost_loc : Table

CodPostal	Localitate	Judet
705300	Focsani	Vrancea
705725	Pascani	Iasi
706600	Iasi	Iasi
706610	Iasi	Iasi
706750	Tg.Frumos	Iasi

Record: 1 of 5

clienti : Table

CodClient	NumeClient	Adresa	CodPostal
1001	TEXTILA SA	Bld. Copou, 87	706600
1002	MODERN SRL	Bld. Gării, 22	705300
1003	OCCO SRL		706610
1004	FILATURA SA	Bld. Unirii, 145	705300
1005	INTEGRATA SA	I.V.Viteazu, 115	705725
1006	AMI SRL	Galaţiului, 72	706750
1007	AXON SRL	Silvestru, 2	706610
1008	ALFA SRL	Prosperităţii, 15	705725

Record: 1 of 8

facturi : Table

NrFact	CodClient	Data	ValoareTotala	TVAColectata
111112	1001	6/17/2005	2850.00	455.04
111113	1004	6/18/2005	5850.00	934.03
111114	1003	6/18/2005	2850.00	455.04
111115	1008	6/18/2005	35700.00	5700.00
111117	1006	6/20/2005	1100.00	0.00
111118	1007	6/23/2005	15000.00	1238.53
111119	1005	6/24/2005	4720.00	753.61
111120	1003	6/24/2005	3000.00	478.99
111121	1001	6/24/2005	4250.00	678.57
111122	1007	6/24/2005	8750.00	722.48
111126	1002	6/1/2005	54250.00	8661.76

Record: 1 of 11

Figura 4.13. Conţinutul celor trei tabelei ale bazei de date VÎNZĂRI