

## MODELUL RELAȚIONAL DE ORGANIZARE A BAZELOR DE DATE

---

*Sumar:*

- Câte ceva despre baze de date
- Prezentare generală a modelului relațional
- Restricții ale bazei de date
- Schema și conținutul unei baze de date relaționale

## 4.1. CÂTE CEVA DESPRE BAZE DE DATE

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

Celălalt pol regroupează probleme în care complexitatea calculelor rareori depășește nivelul celor patru operații aritmetice elementare - adunare, scădere și încă două; în schimb, volumul informațiilor și zecile/ sutele moduri de regroupare și agregare a lor este deconcertant.

Fără a face concurență vreunui manual de filosofie, putem spune că, din păcate, ca și în viață, 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 câștigă enorm de mulți 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 existență virtuală, fiind utili mai degrabă din rațiuni didactice & pedagogice.

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

### 4.1.1 La început a fost fișierul

Prima care a resimțit acut nevoia unor instrumente software dedicate administrării unor cantități imense de informații a fost armata SUA. În a doua parte a anilor '50 Departamentul Apărării al SUA a format un grup de specialiști pentru elaborarea unui limbaj destinat aplicațiilor administrative, în care dificultatea majoră ținea de volumul imens de resurse materiale și financiare ce trebuia "chivernisit" și 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 aplicațiilor de acest tip – specifică nu numai COBOL-ului, ci multor limbaje din a III-a generație, denumită *flat-files architecture* – tradusă, într-o doară, în românește drept *fișiere independente* – este reprezentată în figura 4.1. Specific acestui mod de lucru, referit ca *file-based* sau *flat files* (fișiere independente), este faptul că fiecare dată (Data1, Data2,... Datan) este descrisă (nume, tip, lungime), autonom, în toate fișierele în care apare. Mai mult, descrierea fiecărui fișier de date (câmpurile care-l alcătuiesc, tipul și lungimea fiecăruia, modul de organizare (secvențial, indexat, relativ etc.)) este obligatorie în toate programele care îl "citesc" sau modifică. Între FIȘIER1, FIȘIER2, ... FIȘIERn nu există nici o relație definită explicit.

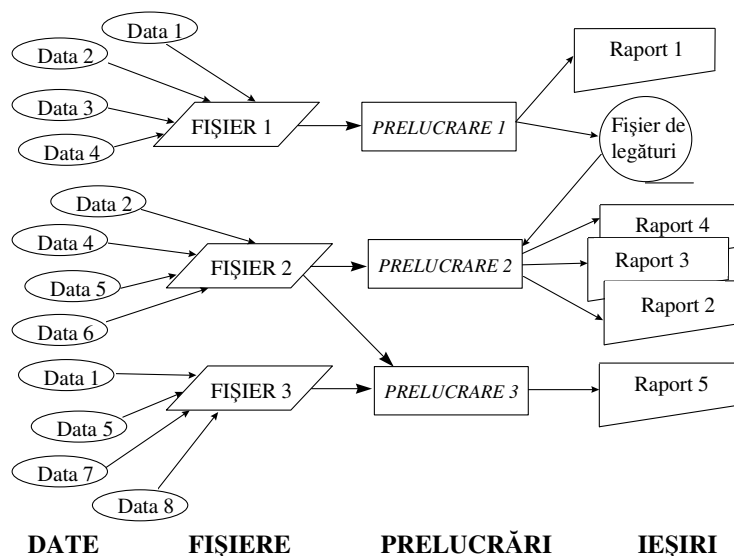


Figura 4.1. Sistem informatic bazat pe organizarea datelor în fișiere independente

Spre exemplu, Data2 este prezentă în două fișiere de date, FIȘIER1 și FIȘIER2. Dacă, prin program, se modifică formatul sau valoarea acesteia în FIȘIER1, modificarea nu se face automat și în FIȘIER2; prin urmare, o aceeași dată, Data2, va prezenta două valori diferite în cele două fișiere, iar necazurile bat la ușă: informațiile furnizate de sistemul informatic sunt redundante și prezintă un mare risc de pierdere a coerenței.

Se poate proiecta un mecanism de menținere a integrității datelor, astfel încât actualizarea unei date într-un fișier să atragă automat actualizarea tuturor fișierelor de date în care aceasta apare, însă, în sistemele mari, care gestionează volume uriașe de informații, implementarea unui asemenea mecanism este extrem de complexă și costisitoare. În plus, fișierele de date sunt uneori proiectate și implementate la distanțe mari în timp, în formate diferite: de exemplu, FIȘIER1 este posibil să fi fost creat cu ajutorul limbajului COBOL, FIȘIER2 în FORTRAN iar FIȘIER3 în BASIC. În asemenea condiții, punerea în operă a mecanismului de menținere a integrității devine o utopie.

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

*Redundanța și inconsistența datelor:* o aceeași dată apare în mai multe fișiere; în aceste cazuri există riscul modificării acesteia într-un fișier fără a face modificările și în toate celelalte fișiere.

*Dificultatea accesului.* Într-o întreprindere, o aceeași informație este exploatată de mai mulți utilizatori. Spre exemplu, pentru departamentul care se ocupă cu gestiunea stocurilor, intrările de materiale trebuie ordonate pe magazine (depozite) și 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, fișierele tradiționale nu facilitează accesarea datelor după mai multe criterii, specifice diferiților 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 tranzacții.

*Complexitatea apăsătoare a actualizărilor.* O actualizare presupune adăugarea, modificarea sau ștergerea unor informații din fișiere. Cum prelucrările se desfășoară în timp real, de la mai multe terminale (în mediile multi-utilizator), pot apare situații conflictuale atunci când doi utilizatori doresc

modificarea simultană a unei aceleiași date. Rezolvarea acestui gen de conflicte presupune existența unui program-supervizor al prelucrărilor, care este greu de realizat cu o multitudine de fișiere, create la distanță în timp și, în formate diferite.

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

Probleme legate de *integritatea datelor*. Informațiile stocate în fișiere sunt supuse la numeroase restricții semantice. Toate aceste restricții alcătuiesc mecanismul de integritate a datelor, deosebit de complex în mediile de lucru multi-utilizator și eterogene.

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

*Costul ridicat* se datorează gradului mare de redundanță a datelor, eforturilor deosebite ce trebuie depuse pentru interconectarea diferitelor tipuri de fișiere de date și pentru asigurarea funcționării sistemului în condițiile respectării unui nivel minim de integritate și securitate a informațiilor.

*Inflexibilitatea față de schimbările ulterioare*, ce sunt inerente oricărui sistem informațional.

*Modelarea inadecvată a lumii reale.*

Aceste dezavantaje sunt mai mult decât convingătoare, încât vă puteți întreba dacă au existat așa inconștienți care să-și arunce banii pe apa... fișierelor independente. Ei bine, o serie de aplicații dezvoltate în anii '60 sau '70 au fost moștenite și 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 funcționalitatea și viteza unor asemenea aplicații. Și, nu în ultimul rând, un mediu economic relativ "așezat" (vă dați seama că nu de mediul *nostru* economic este vorba), în care nu au fost necesare modificări majore ale procedurilor și funcțiilor esențiale.

### 4.1.2 Baze de date

Sintagma *bază de date* apare pentru prima dată în titlul unei conferințe 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 conferințe dedicate limbajelor de gestiune a datelor, a primului raport tehnic în care este prezentat conceptul de bază de date. Față de modelul fișierelor independente, noutatea o constituie existența unui *fișier de descriere globală* a bazei, astfel încât să se poată asigura independența programelor față de date, după cum o arată și figura 4.2.

Avantajele organizării informațiilor în baze de date decurg tocmai din existența acestui fișier de descriere globală a bazei, denumit, în general, *dicționar de date* (alte titulaturi: *repertor de date* sau *catalog de sistem*). Extragerea și modificarea datelor, altfel spus, lucrul cu fișierele de date, se derulează exclusiv prin intermediul dicționarului în care se găsesc informații privitoare la structura datelor și restricțiile îndeplinite de acestea.

*O bază de date (BD) reprezintă un ansamblu structurat de fișiere, care grupează datele prelucrate în aplicațiile informatice ale unei persoane, grup de persoane, întreprinderi, instituții etc. Formal, BD poate fi definită ca o colecție de date aflate în interdependență, împreună cu descrierea datelor și a relațiilor dintre ele, sau ca o colecție de date utilizată într-o organizație, colecție care este automatizată, partajată, definită riguros (formalizată) și controlată la nivel central.*

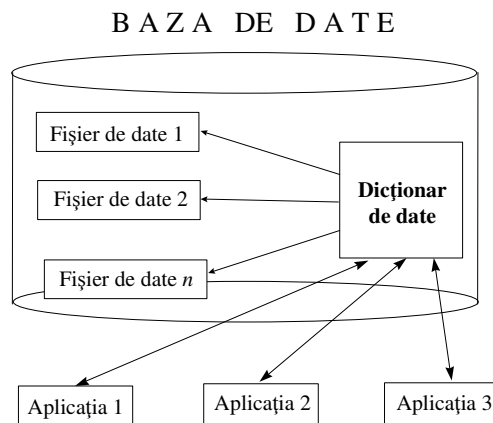


Figura 4.2. Schemă de principiu a unei baze de date

Atunci când vorbim despre o bază de date, trebuie avute în vedere două aspecte fundamentale acesteia, *schema* și *conținutul*. Organizarea bazei de date se reflectă în *schema* sau *structura* sa, ce reprezintă un ansamblu de instrumente pentru descrierea datelor, a relațiilor dintre acestea, a semanticii lor și a restricțiilor la care sunt supuse. Ansamblul informațiilor stocate în bază la un moment dat constituie *conținutul* sau *instanțierea* sau *realizarea* acesteia. În timp ce volumul prezintă o evoluție 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 și pe diferite paliere. De obicei, abordarea se face pe trei nivele: *fizic* sau *intern*, *conceptual* sau *global* și *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 conținutul lor efectiv, ca și relațiile care le leagă de alte date. Reprezintă primul nivel de abstractizare a lumii reale observate. Obiectivul acestui nivel îl constituie modelarea realității considerate, asigurându-se independența bazei față de orice restricție tehnologică sau echipament anume. Toți utilizatorii își exprimă nevoile de date la nivel conceptual, prezentându-le administratorului bazei de date, acesta fiind cel care are o viziune globală necesară satisfacerii tuturor cerințelor informaționale.

*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 porțiune specifică a bazei, în funcție de departamentul în care își desfășoară activitatea și de atribuțiile sale (lor). Simplificarea interacțiunii utilizatori-bază, precum și creșterea securității bazei, sunt deziderate ale unui nivel superior de abstractizare, care este nivelul extern. Astfel, structura BD se prezintă sub diferite machete, referite, uneori și ca sub-scheme, scheme externe sau imagini, în funcție 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.

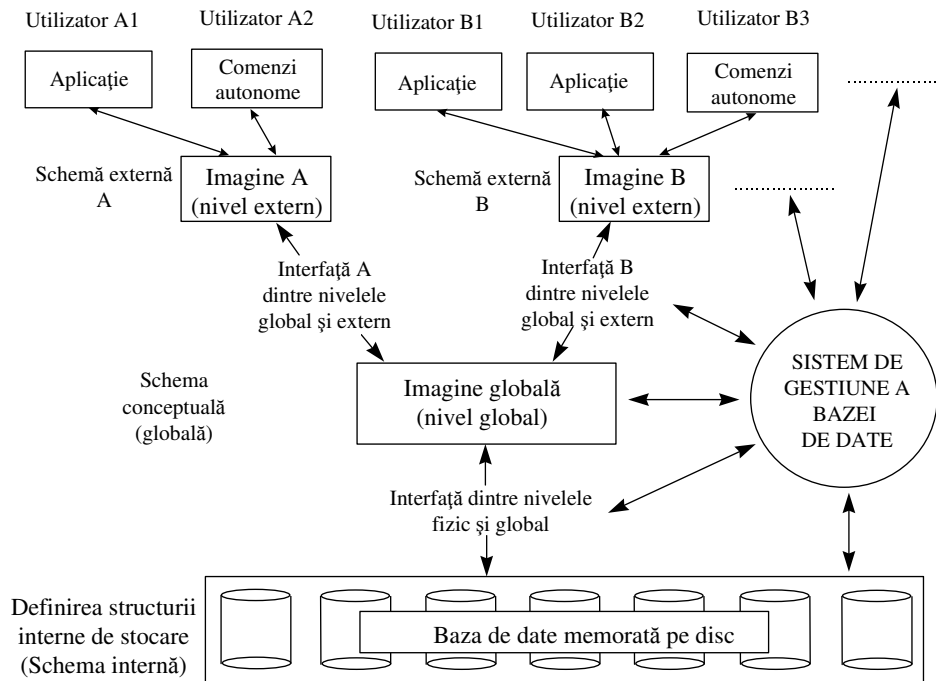


Figura 4.3. Schematizare a unui sistem de lucru cu o bază de date

Posibilitatea modificării structurii la un nivel, fără a afecta structura nivelului sau nivelelor superioare, se numește *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 și rescrierea programelor pentru exploatarea bazei de date. Asemenea modificări sunt necesare uneori pentru ameliorarea performanțelor de lucru (viteză de acces, mărirea fișierelor etc.). Tot autonomia fizică este cea care asigură portarea bazei de date de pe un sistem de calcul pe altul fără modificarea schemei conceptuale și a programelor.

*Autonomia logică* presupune posibilitatea modificării schemei conceptuale a bazei (modificare datorată necesității rezolvării unor noi cerințe informaționale) 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 și actualizează, în ciuda existenței dicționarului de date. Firește, un element important îl reprezintă și 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 și aplicații;
- *persistență* – existență permanentă, din momentul preluării în bază până în momentul actualizării sau ștergerii;
- *securitate* – protejarea de accesul neautorizat, atât în ceea ce privește citirea și copierea, cât și modificarea și ștergerea;

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

### 4.1.3 Modele de organizare a datelor în baze

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

Modelul datelor este o reprezentare a obiectelor lumii reale și a evenimentelor asociate lor. Este o abstractizare care se concentrează pe aspectele esențiale ale organizației /aplicației, furnizând conceptele de bază și notațiile care vor permite utilizatorilor bazelor de date să comunice clar și rapid informațiile și cunoștințele lor despre datele organizației.

O grupare "tradițională" a modelelor utilizate în bazele de date delimitează trei categorii: modele logice bazate pe obiect, modele logice bazate pe înregistrare și modele fizice. Din punctul nostru de vedere, interesează numai nivelele conceptual și extern de abstractizare a datelor; de aceea, vom prezenta, în linii mari, numai reprezentanții 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 și fii.

*Modelul rețea.* Este o dezvoltare a modelului ierarhic, prin care se pot reprezenta și situațiile în care un fiu "posedă" mai mulți tați. Înregistrările sunt privite în BD ca o colecție de grafuri.

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

*Modelul obiectual.* Începând cu anii '60, în programare și, ceva mai târziu, în analiză și proiectare, orientarea pe obiecte (OO) a avut un succes uriaș, reușind să depășească metodologiile structurate. Pe baza acestui succes, s-a crezut că și în materie de baze de date, modelul obiectual îl va surclasa pe cel relațional. Rezultatele sunt însă deprimante pentru suporterii OO-ului, piața SGBD OO fiind sub 6% din valoarea totală a pieței bazelor de date.

*Modelul relațional-obiectual.* Este un model mai recent ce încearcă să valorifice deopotrivă atuurile relaționalului cu orientarea pe obiecte. Deși 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 RELAȚIONAL

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 și o componentă *manipulatorie*, adică modul în care obținem informații din bazele de date (ceea ce presupune o serie de operatori aplicabili uneia sau mai multor relații).

Deși puternic contestat, și cu neajunsurile sale, modelul relațional de organizare a bazelor de date rămâne cel mai utilizat. Cu foarte puține excepții, toate aplicațiile software realizate pentru bănci, buticuri, universități (ordinea este, în ciuda aparențelor, pur întâmplătoare) sunt realizate cu produse/instrumente ce gestionează baze de date relaționale.

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

### 4.2.1 Puțină istorie

Modelul relațional de organizare a datelor s-a conturat în două articole publicate în 1969 și 1970 de către E.F. Codd, matematician la centrul de cercetări din San Jose (California) al firmei IBM<sup>1</sup>. În acel moment, tehnologia bazelor de date era centrată pe modelele ierarhic și rețea, 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 și software de sistem pe care este implementată, structură “înzestrată” cu o serie de operatori pentru extragerea datelor. Deși puternic matematizat, modelul relațional este relativ ușor de înțeles.

Modelul relațional al datelor se poate defini printr-o serie de structuri de date (relații alcătuite din tupluri), operații aplicate asupra structurilor de date (selecție, proiecție, joncțiune etc.) și reguli de integritate care să asigure consistența datelor (chei primare, restricții referențiale s.a.). Modelarea realității se concretizează în tabele de valori numite relații, avându-se în vedere că:

- o relație are un nume;
- o coloană reprezintă un atribut;
- o linie reprezintă un n-uplet (tuplu) de valori ale celor  $n$  atribute din relație;
- ordinea liniilor și coloanelor în cadrul tabelului nu este relevantă pentru conținutul informațional.

Fiecare linie a tabelului reprezintă o entitate sau un fapt al realității, în timp ce o coloană reprezintă o proprietate a acestei entități 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 și o relație între obiecte (persoane, lucruri), cum ar fi: contractele de afaceri, căsătoriile (există totuși câteva deosebiri între acestea și precedentele), structura ierarhică a unei organizații etc. În plus, nu e întotdeauna evident care dintre informații sunt entități, care atribute și care asociații (relații).

<sup>1</sup> 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 conținutul bazei de date, conținut ce poate fi modificat prin operațiuni de actualizare: introducerea unor linii (tupluri) noi, ștergerea unor linii, modificarea valorii unor atribute.

Față de modelele ierarhice și rețea, modelul relațional prezintă câteva avantaje:

- propune structuri de date ușor de utilizat;
- ameliorează independența logică și fizică;
- pune la dispoziția utilizatorilor limbaje ne-procedurale;
- optimizează accesul la date;
- îmbunătățește integritatea și confidențialitatea datelor;
- ia în calcul o largă varietate de aplicații;
- abordează metodologic definirea structurii bazei de date.

Modelului relațional îi este asociată teoria normalizării, care are ca scop prevenirea comportamentului aberant al relațiilor în momentul actualizării lor, eliminarea datelor redundante și înlesnirea înțelegerii legăturilor semantice dintre date.

Deși IBM a fost prima care a inițiat un proiect destinat elaborării unui SGBD relațional (System/R, începând cu 1974), prima firmă care a lansat primul SGBDR comercial a fost Relational Software Inc., astăzi Oracle. Înființarea firmei a avut loc în 1977, iar lansarea produsului în 1979. Impunerea pe piață a SGBD-urilor grefate pe modelul relațional a fost mult mai dificilă decât s-ar putea înțelege astăzi. Abia în deceniul 9, datorită vitezei ameliorate, securității sporite și mai ales productivității dezvoltatorilor de aplicații, modelul relațional a câștigat supremația.

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

Pentru aplicațiile complexe din bănci, corporații, organizații și instituții 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 și costisitoare.

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

#### 4.2.2. Relații/tabele, domenii și atribute

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 și este stocată pe suport extern (de obicei disc). La intersecția unei linii cu o coloană se găsește o valoare atomică (elementară). O relație conține informații omogene legate de anumite entități, procese, fenomene: CĂRȚI, STUDENȚI, LOCALITĂȚI, PERSONAL, FACTURI etc. Spre exemplu, în figura 4.4 este reprezentată tabela CLIEȚI.

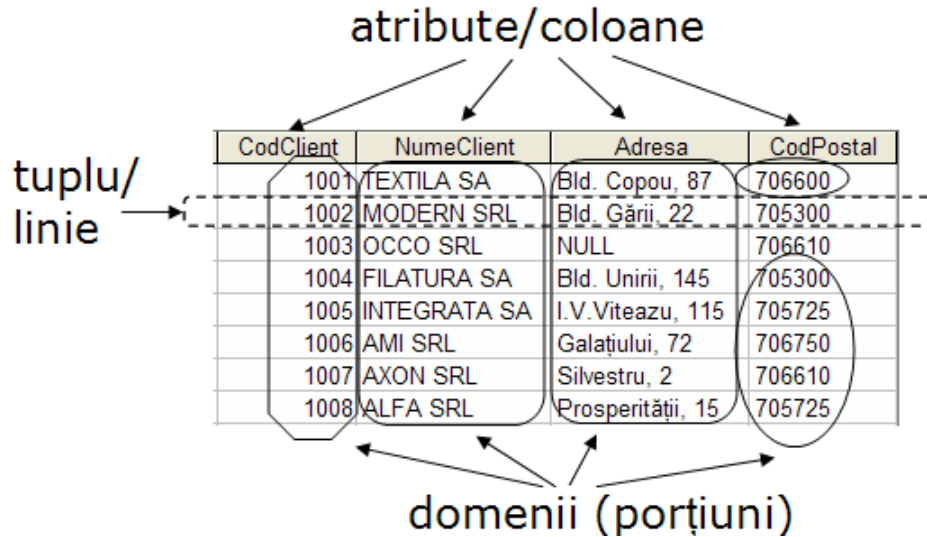


Figura 4.4. Relația (tabela) *CLIEŢI*

În teoria relațională se folosește 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 regroupează 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 (oraș sau comună), un angajat al firmei, o factură emisă etc. Figura 4.5 conține al doilea tuplu din tabela *CLIEŢI*, tuplu referitor la forma MODERN SRL. Linia de mai jos este alcătuită din patru valori ce desemnează: codul, numele, adresa și codul poștal al localității și adresei referitoare la clientul MODERN SRL.

1002	MODERN SRL	Bld. Gării, 22	705300
------	------------	----------------	--------

Figura 4.5. Un tuplu al tablei *CLIEŢI*

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. Firește, toate tuplurile relației au același format (structură), ceea ce înseamnă că în tabela *CLIEŢI*, 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, și o valoare a codului poștal. Ordinea tuplurilor nu prezintă importanță din punctul de vedere al conținutului informațional al tablei.

Fiecare atribut este caracterizat printr-un *nume* și 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* și *Bărbătesc*;
- domeniul atributului **Judet** este alcătuit din numele fiecărui județ (plus București).
- 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 valoare cuprinsă între 1 și 99999999 lei (ceva mai noi).

Și 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**, și cu  $D_4$  domeniul atributului **CodPostal**, se poate spune că fiecare linie a tabelului **CLIENTI** este un tuplu de patru elemente, 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.

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

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

$n$  fiind denumit 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  $m$ -uple (  $m$ -tupluri) de valori:  $R = \{ t_1, t_2, \dots, t_k, \dots, t_m \}$ , unde  $t_k = (d_{k1}, d_{k2}, \dots, d_{ki}, \dots, d_{kn})$

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 <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	...	D <sub>i</sub>	...	D <sub>n</sub>
t <sub>1</sub>	d <sub>11</sub>	d <sub>12</sub>	d <sub>13</sub>	...	d <sub>1i</sub>	...	d <sub>1n</sub>
t <sub>2</sub>	d <sub>21</sub>	d <sub>22</sub>	d <sub>23</sub>	...	d <sub>2i</sub>	...	d <sub>2n</sub>
t <sub>3</sub>	d <sub>31</sub>	d <sub>32</sub>	d <sub>33</sub>	...	d <sub>3i</sub>	...	d <sub>3n</sub>
	.	.	.		.		.
	.	.	.		.		.
	.	.	.		.		.
t <sub>k</sub>	d <sub>k1</sub>	d <sub>k2</sub>	d <sub>k3</sub>	...	d <sub>ki</sub>	...	d <sub>kn</sub>
	.	.	.		.		.
	.	.	.		.		.
	.	.	.		.		.
t <sub>m</sub>	d <sub>m1</sub>	d <sub>m2</sub>	d <sub>m3</sub>	...	d <sub>mi</sub>	...	d <sub>mn</sub>

Figura 4.6. Ilustrarea celei de-a doua definiții a unei relații

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.

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

Numărul de tabele pe care le conține o bază de date, atributele “adunate” în fiecare tabelă, domeniul fiecăruia dintre atribute prezintă diferențe majore de la o bază la alta, chiar dacă uneori reflectă același tip de procese. Intrăm astfel în sfera proiectării bazelor de date, a dependențelor și normalizării.

Relația **CLIENTI** conține informații despre firmele cărora compania noastră le vinde produsele pe care le producem și/sau comercializăm. Fiecare linie se referă la un singur client. În

figura 4.4 pe a treia linie a tabelului apare o valoare curioasă notată *NULL*. Valoarea *NULL* este considerată o metavaloare și indică faptul că, în acel loc, informația este necunoscută sau inaplicabilă. Valoarea *NULL* este diferită, însă de valorile 0 sau spațiu. Uneori, importanța sa este, din păcate, majoră în expresii și funcții, după cum știu cei cu oarecare experiență în limbajului SQL.

## 4.3 RESTRICȚII ALE BAZEI DE DATE

De ce ne interesează restricțiile într-o bază de date? Termenul de *restricție* este oarecum iritant, atât pentru studenți, cât și pentru profesori, deoarece semnalează existența unor constrângeri instituite și oarecum obligatorii, și, de vreme ce sunt impuse, înseamnă că nu sunt prea plăcute (decât, în cel mai bun caz, pentru cel care le-a instituit). Partea cea mai enervantă este că respectarea restricțiilor este (supra)vegheată de o anumită autoritate înzestrată cu anumite instrumente de constrângere, de la bastoane de cauciuc, la creșterea și scăderea impozitelor, salariilor, banilor de buzunar etc.

Ei bine, în bazele de date, restricțiile sunt ceva mai acceptabile. Cei care lucrează cu bazele de date sunt foarte interesați în declararea restricțiilor, 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). Esențial este că, ajutați de restricții, putem crește gradul de corectitudine și de încredere al datelor din bază. În cele ce urmează vor fi prezentate pe scurt cele mai importante restricții definibile într-o bază de date relațională: restricția de domeniu, de atomicitate, de unicitate, referențială și restricțiile-utilizator.

### 4.3.1 Restricția de domeniu

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

O parte din informaticieni substituie domeniul tipului atributului: *numeric*, *șir de caractere*, *dată calendaristică*, *logic* (boolean) etc. și, eventual, lungimii (numărul maxim de poziții/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 județ (vezi plăcuțele de înmatriculare) poate fi una din valorile: IS, TM, B etc. reprezintă o restricție de comportament sau, mai simplu, o restricție definită de utilizator.

Cea de-a doua categorie privește domeniul deopotrivă sintactic și semantic. Astfel, domeniul sintactic al atributului **Jud** (indicativul județului) este un șir de două caractere, obligatoriu litere (sau o literă și un spațiu, pentru București), și, 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 și semantic) atributului Jud prin declararea tipului și lungimii atributului și prin așa-numitele reguli de validare la nivel de câmp (field validation rule). Sunt însă și produse la care domeniul poate fi definit explicit, sintactic și semantic, dându-i-se un nume la care vor fi legate attributele în momentul creării tabelului.

### 4.3.2 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 țință predilectă a “atacurilor dușmănoase” la adresa modelului relațional, datorită imposibilității înglobării 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 și Hugh Darwen, se opun ideii de atomicitate formulată de Codd.

Primele vizate de “stigmatul” neatomicității sunt atributele compuse. Un exemplu de atribut compus (non-atomic) este Adresa. 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 (Seria+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 același număr de atribute. Fiecare celulă a tabelului (intersecția unei coloane cu o linie), altfel spus, valoarea unui atribut pe o linie (înregistrare), trebuie să fie atomică.

### 4.3.3 Nenulitate

Modelul relațional acceptă ca, atunci când nu se cunoaște valoarea unui atribut pentru o anumite entitate, sau când pentru acel obiect, entitate, persoană etc. atributul este inaplicabil, să se folosească (meta)valoarea NULL. Celui de-al treilea client din figura 4.4 nu i cunoaște adresa. Dacă am avea o tabelă PRODUSE cu atributele CodProdus, DenumireProdus, UM, Culoare, este posibil ca, pentru anumite sortimente, cum ar tricouri, cămăși, jachete, atributul să fie important, în timp ce pentru altele, precum cafea, vodcă, lapte atributul culoare să nu furnizeze nici o informație, adică să nu fie aplicabil. Iar dacă laptele poate fi doar alb, vodca chiar că nu are culoare (sunt foarte mulți specialiști în acest domeniu, îi puteți întreba !).

Într-o bază de date relațională avem posibilitatea de a le impune unor atribute să aibă întotdeauna valori specificate, altfel spus, le interzicem valorile nule, în timp ce altor atribute li se pot permite valori nule. Modul în care se declară și se afișează o restricție 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 CLIENȚI i se interzic valorile nule, prin setarea opțiunii **Required** pe valoarea

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

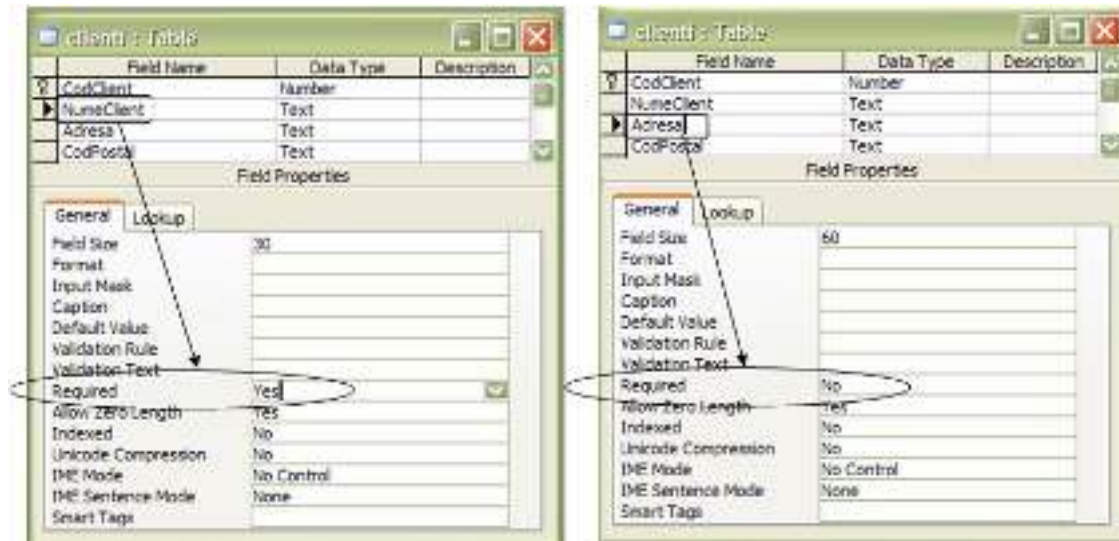


Figura 4.7. Un atribut “forțat” să nu primească valori NULL și un altul “relaxat”

Ca regulă, atributele importante, ce țin de identificarea sau caracterizarea unei entități, proces, fenomen, precum și cele implicate în calculul unor informații importante, sunt declarate NOT NULL, iar atributele fără importanță deosebită pot fi mai relaxate.

#### 4.3.4 Restricția de unicitate

Într-o relație nu pot exista două linii identice (două linii care prezintă aceleași valori pentru toate atributele). Mai mult, majoritatea relațiilor prezintă un atribut, sau o combinație de atribute, care diferențiază cu siguranță un tuplu de toate celelalte tupluri ale relației.

Cheia primară a unei relații (tabele) este un atribut sau un grup de atribute care identifică fără ambiguitate fiecare tuplu (linie) al relației (tabelei). După Codd, există trei restricții pe care trebuie să le verifice cheia primară:

- *unicitate*: o cheie identifică un singur tuplu (linie) al relației.
- *compoziție minimală*: atunci când cheia primară este compusă, nici un atribut din cheie nu poate fi eliminat fără distrugerea unicității tuplului în cadrul relației; în cazuri limită, o cheie poate fi alcătuită din toate atributele relației.
- *valori non-nule*: valorile atributului (sau ale ansamblului de atribute) ce desemnează cheia primară sunt întotdeauna specificate, deci ne-nule și, mai mult, nici un atribut din compoziția cheii primare nu poate avea valori nule; această a treia condiție se mai numește și *restricție a entității*.

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 ca și *cheie alternativă*.

În tabela **CLIENTI** 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 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**, **DataNașterii**, **Vechime**, **SalariuTarifar**.

- atributul **Nume** nu poate fi cheie, deoarece chiar și într-o întreprindere de talie mijlocie, este posibil să existe doi angajați cu același nume.
- dacă apariția a două persoane cu nume identice este posibilă, atunci apariția a două persoane cu același Prenume este probabilă.
- nici unul din aceste atributele **DataNașterii**, **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 celelalte.

- Combinația **Nume+Adresă** pare, la primele două vederi, a îndeplini "cerințele" de identificator. Ar fi totuși o problemă: dacă în aceeași firmă (organizație) lucrează împreună soțul și soția ? Ambii au, de obicei, același nume de familie și, tot de obicei, același domiciliu. Este adevărat, cazul ales nu este prea fericit. Dar este suficient pentru "compromiterea" combinației.
- Următoarea tentativă este grupul **Nume+Prenume+Adresă**, combinație neoperantă dacă în organizație lucrează tatăl și un fiu (sau mama și o fiică) care au aceleași nume și prenume și domiciliul comun.
- Ar rămâne de ales una dintre soluțiile (**Nume+Prenume+Adresă+Vechime**) sau (**Nume+Prenume+Adresa+DataNașterii**).

Oricare din cele două combinații prezintă riscul violării restricției de entitate, deoarece este posibil ca, la preluarea unui angajat în bază, să nu i se cunoască adresa sau data nașterii, caz în care atributul respectiv ar avea valoarea **NULL**. Dificultățile de identificare fără ambiguitate a angajaților au determinat firmele ca, la angajare, să aloce fiecărei persoane un număr unic, număr denumit **Marcă**. Prin adăugarea acestui atribut la cele existente, pentru relația **PERSONAL** problema cheii primare este rezolvată mult mai simplu. Actualmente, sarcina este simplificată și prin utilizarea codului numeric personal (**CNP**), combinație de 13 cifre care prezintă avantajul că rămâne neschimbată pe tot parcursul vieții persoanei.

Revenim la tabela **CLIENTI**. Dacă în țara noastră s-ar respecta regula: *nu pot exista două firme cu aceeași 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 opțiunii *Indexed* pe valoarea *Yes (NoDuplicates)*.

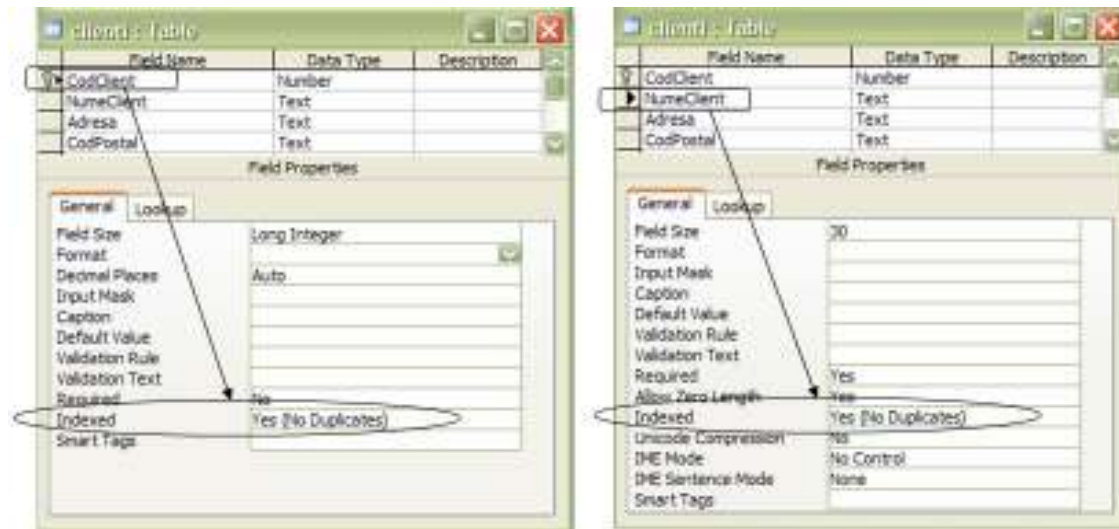


Figura 4.8. O cheie primară și o cheie alternativă

### 4.3.5 Restricția referențială

O bază de date relațională este alcătuită din relații (tabele) aflate în legătură. Stabilirea legăturii se bazează pe mecanismul cheii străine și, implicit, a restricției referențiale. Figura 5.5 prezintă o relație în care sunt implicate tabelele FACTURI și CLIENȚI.

Figura 4.9. O restricție referențială între FACTURI (tabelă-copil) și CLIENȚI (părinte)

Atributul **CodClient** joacă un rol de “agent de legătură” între tabelele CLIENȚI și FACTURI. Pentru relația CLIENȚI, atributul **CodClient** este cheie primară, în timp ce în tabela FACTURI, **CodClient** reprezintă coloana de referință sau cheia străină, deoarece numai pe baza valorilor sale se poate face legătura cu relația părinte CLIENȚI.

Cheile străine sau coloanele 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 primară se numește *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. Când două tabele (relații), T1 și T2, prezintă atributul sau grupul de attribute notat CH, care, pentru T1, este cheie primară, iar pentru T2 cheie străină, *dacă în T2 se interzice apariția de valori nenule ale CH care nu există în nici un tuplu din T1*, se spune că între T2 și T1 s-a instituit o restricție referențială.

Instituirea restricției referențiale între tabela CLIEŢI (părinte) și 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 CLIEŢI).

### Observații

1. Pentru mulți utilizatori și profesioniști ai bazelor de date, denumirea de "relațional" desemnează faptul că o bază de date este alcătuită din tabele puse în legătură prin intermediul cheilor străine. Aceasta este, de fapt, a doua accepțiune a termenului de BDR, prima, cea "clasică", având în vedere percepția fiecărei linii dintr-o tabelă ca o relație între clase de valori.
2. Majoritatea SGBD-urilor prezintă mecanisme de declarare și gestionare automată a restricțiilor referențiale, prin actualizări în cascadă și interzicerea valorilor care ar încălca aceste restricții.
3. Respectarea restricțiilor referențiale este una din cele mai complicate sarcini pentru dezvoltatorii de aplicații 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 "corpulente". 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 și de joncțiune între attribute.

## 4.3.6 Restricții-utilizator

Restricțiile utilizator mai sunt denumite și restricții de comportament sau restricții ale organizației. De obicei, aceste restricții iau forma unor reguli de validare la nivel de atribut, la nivel de linie/tabelă sau a unor reguli implementate prin declanșatoare (triggere).

### 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 stabilite, în alte formate decât cele acceptate etc. Forma clasică a unei restricții la nivel de atribut este o expresie în care apar constante, funcții-sistem și, nu în ultimul rând, atributul respectiv. La orice editare a atributului cu pricina (declanșată în cazul inserării unei linii în tabela din care face parte, sau la modificarea sa) expresia este evaluată și 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 CLIEŢI valorile atributului CodClient trebuie să fie mai mari decât 1000. Rubrica **Validation Rule** este cea în care apare expresia-restricție - *[CodClient]>1000* - (observați 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 restricția - *Cel mai mic cod de client acceptat este 1001 !*.

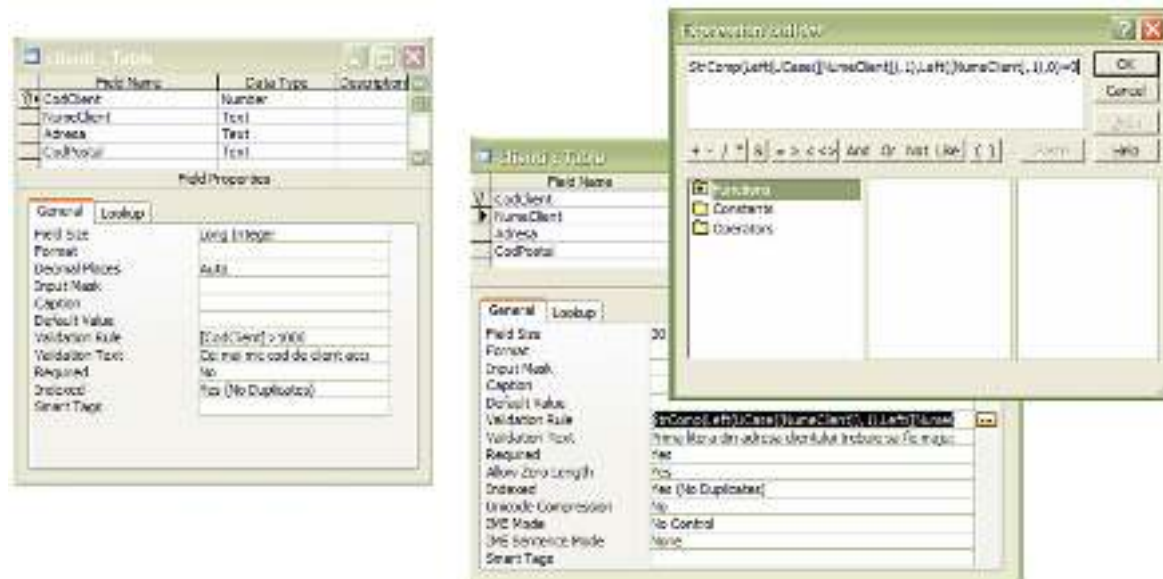


Figura 4.10. Două restricții la nivel de atribut

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 -  $StrComp(Left(UCase([NumeClient]),1),Left([NumeClient],1),0) = 0$  – și nu vom strica vraja până în capitolul următor.

### Reguli la nivel de înregistrare

Expresia care definește o restricție la nivel de înregistrare poate conține două sau mai multe atribute și este evaluată la inserarea sau modificarea oricărei linii din tabelă. Tabela FACTURI conține, printre altele două atribute “valorice”, unul pentru păstrarea valorii totale (inclusiv TVA) facturii respective și un altul care indică “doar” quantumul TVA colectate pentru factură. Majoritatea produselor și serviciilor comercializate la noi în țară au un procent al taxei pe valoarea adăugată de 19%. Există, însă, și 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 conține numai produse cu 19%:  $ValoareaTotală = ValoareaFărăTVA + 0.19 * ValoareaFărăTVA$ , sau  $ValoareaTotală = 1.19 * ValoareaFărăTVA$ . Dar tabela noastră are doar atributele ValoareTotală și TVACollectată, așa că înlocuim ValoareaFărăTVA prin diferența celorlalte două. Scriem:  $ValoareTotală = 1.19 * (ValoareTotală - TVACollectată)$ , și după calcule de matematică superioară,  $TVACollectată * 1.19 = 0.19 * ValoareTotală$ , altfel spus  $TVACollectată = ValoareTotală * 0.19 / 1.19$ .

Prin urmare, regula este  $TVACollectată \leq ValoareTotală * 0.19 / 1.19$ , iar în Access declararea sa este ilustrată în figura 4.11.

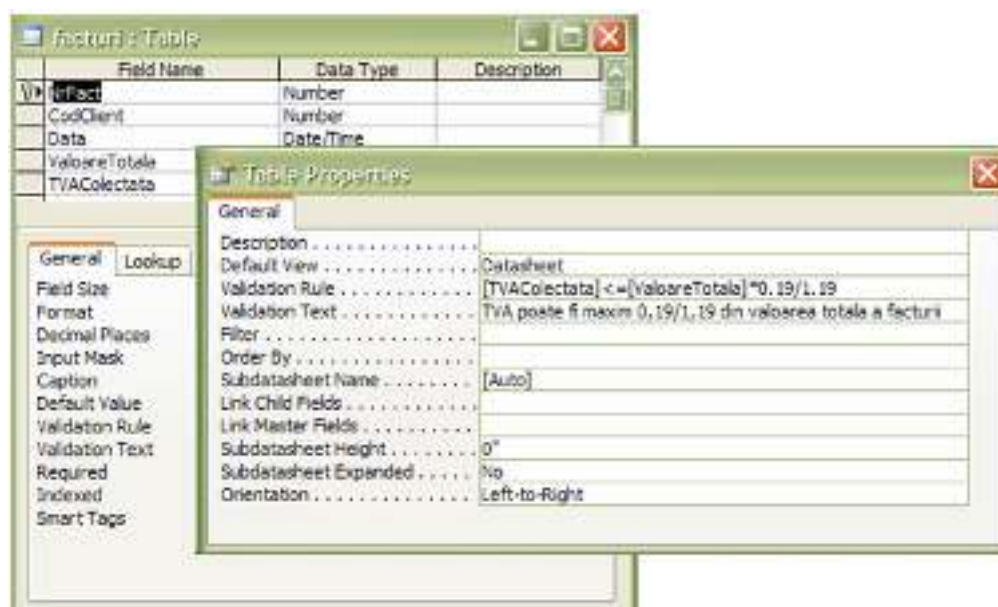


Figura 4.11. O restricție la nivel de înregistrare

#### Alte tipuri de restricții utilizator

Există și alte tipuri de restricții 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 acestuia nu este membru în partidul/partidele de guvernământ. Acestea reclamă întrebuițarea unor proceduri speciale, numite declanșatoare (triggere). Curioșii n-au decât să urmeze specializarea Informatică Economică.

## 4.4 SCHEMA ȘI CONȚINUTUL UNEI BAZE DE DATE RELAȚIONALE

Există două aspecte complementare de abordare a bazelor de date relaționale: *schema* (structura, intensia) și *conținutul* (instanțierea, extensia).

*Conținutul* unei relații este reprezentat de ansamblul tuplurilor ce o alcătuiesc la un moment dat. Pe parcursul exploatării bazei, conținutul poate crește exponențial, în funcție de volumul și complexitatea operațiunilor consemnate.

*Schema* unei baze de date conține denumiri ale tabelor, numele, tipul și lungimea atributelor, restricții de unicitate, de non-nulitate, restricții la nivel de atribut, linie și alte eventuale tipuri de restricții de comportament, precum și restricții referențiale. La acestea se adaugă cele privind drepturile utilizatorilor, definiția și restricțiile tabelor virtuale, indecși etc. Foarte importante în lumea profesioniștilor dezvoltării de aplicații cu bazele de date sunt procedurile stocate – programe de forma funcțiilor, procedurilor, pachetelor și mai ales declanșatoarelor (triggerelor) – care, după cum le spune și numele sunt memorate în schema bazei de date și fac parte integrantă din aceasta. Schema este independentă de timp și reprezintă componenta permanentă a relațiilor.

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

Nu este adevărat ! În Visual FoxPro, Access și alte SGBD-uri, pentru a gestiona mai ușor o aplicație 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 și 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.

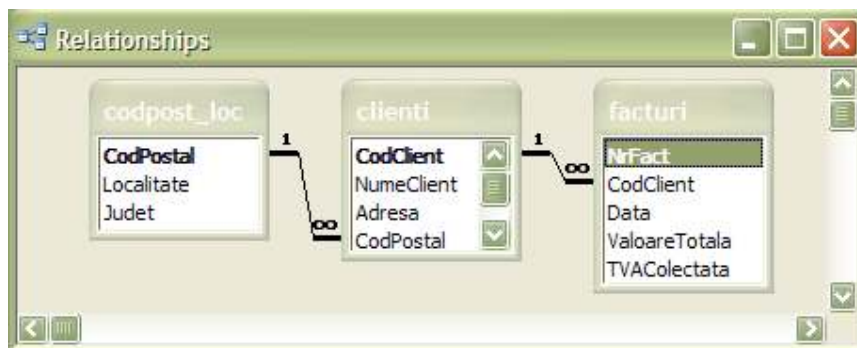


Figura 4.12. Schema simplificată a bazei de date VÎNZĂRI

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

CodPostal	Localitate	Judet
706300	Focsani	Vrancea
706725	Pascari	Iasi
706600	Iasi	Iasi
706610	Iasi	Iasi
706750	Tg Frumos	Iasi

CodClient	NumeClient	Adresa	CodPostal
1001	TEXTILA SA	Bld. Copou, 87	706600
1002	MODERN SRL	Bld. Găni, 22	706300
1003	OCCO SRL		706610
1004	FLATURA SA	Bld. Uniri, 145	706300
1005	INTEGRATA SA	I V Viteazu, 115	706725
1006	AMI SRL	Galajului, 72	706750
1007	AXON SRL	Silvestru, 2	706610
1008	ALFA SRL	Prospentati, 15	706725

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	1238.93
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

Figura 4.13. Conținutul celor trei tabele ale bazei de date VÎNZĂRI