

Capitolul 1. Despre bazele de date

Se spune, pe drept cuvânt, că primele semne ale procesului de ramolire apar atunci când începi să te repeți, confirmarea procesului este auto-citarea, iar desăvârșirea este auto-plagiarea (mai ales cea fără intenție). Deși tentative de a defini bazele de date apar în mai toate cărțile publicate, singur sau în colaborare, la Editurile Polirom ([Fotache 2001], [Fotache s.a. 2002], [Fotache s.a. 2003], [Fotache 2005]) și Junimea [Fotache 1997], din rațiuni populiste voi apela pentru început, cu precădere, la secvențe din două cursuri care încearcă cu disperare să atragă atenția studenților Facultății de Economie și Administrarea Afacerilor¹ la disciplinele *Instrumente software pentru afaceri* și *Baze de date I*. ([Grama s.a.2006], [Fotache 2007]). Sunt vizate importanța și noțiunile fundamentale privind lucrul cu baze de date.

1.1. Nevoia de baze de date

Folosim bazele de date deoarece avem memoria prea scurtă și stăm prost cu calculele. Ca și în alte privințe, avem nevoie de baze de date pentru că suntem niște limitați. Trăim într-o lume așezată pe un morman de hârtii & hârțoage, și ne este cu neputință să reconstituim ceea ce am făcut adineaori, darămite ieri, săptămâna trecută sau acum un an sau cinci. Necazul e că, de cele mai multe ori, trebuie să știm nu numai ce-am făcut noi, dar și câte ceva din ceea ce-au făcut colegii și partenerii de afaceri.

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, dar care reclamă un anumit grad de ingeniozitate în rezolvare - mult invocată și așteptată "fisă". 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.

¹ De la Universitatea A.I.I. Cuza Iași

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.

Prin urmare, avem nevoie de baze de date pentru a păstra, într-un format utilizabil, date și informații legate de evenimente, tranzacții etc. și, la nevoie, de a le regăsi și prelucra după cum ne cer împrejurările. Bazele de date nu reprezintă singurul instrument de stocare și prelucrare a informațiilor. Și într-un banal fișier .DOC (document Word, WordPerfect...), prezentare PowerPoint sau foaie de calcul (Excel, Lotus 1-2-3...) păstrăm date. Ca să nu mai vorbim de pagini Web. Problema este că în documente, foi de calcul, prezentări, fișiere HTML etc. datele sunt slab structurate. Pentru a localiza informațiile trebuie folosite instrumente de căutare care să depisteze prezența unor cuvinte cheie, eventual în preajma unor alte cuvinte cheie (cu o afacere de genul acesta s-au umplut de bani cei de la Google). Iar rezultatele acestui gen de căutare/localizare sunt, de multe ori, iritante prin ambiguitatea și volumul lor.

Încheiem acest mini-paragraf cu altă pereche de vești. Cea bună este că, la acest moment, bazele de date reprezintă cel mai bine structurat mod de păstrare și scotocire a informațiilor. Este motivul pentru care piața produselor de lucru cu bazele de date se exprimă în valori de ordinul miliardelor de dolari, nefiind prea multe semne că s-ar diminua. Vestea rea ține de faptul că, dintre toate datele și informațiile pe care le vehiculăm/gestionăm/prelucram pe hârtie, la telefon, pe bandă sau disc magnetic, optic etc., doar o mică parte sunt preluate și preluabile în bazele de date.

1.2. Cum ne folosim de bazele de date

Mare parte dintre noi suntem datornici, unii chiar redutabili, băncilor, așa că putem rememora împreună câteva „imagini” din plata unei rate. După tradiționala binețe, lucrătorul de la ghișeu băncii ne cere un document de identitate și apoi începe să se uite atent pe ecran la o serie de meniuri și ferestre, tastează numele sau codul numeric personal iar apoi ne comunică suma de plată (rata), eventual câte rate mai avem sau alte informații legate de credit².

Meniurile, ferestrele și rapoartele de pe ecran constituie *interfața* aplicației informatice a băncii, aplicație la care este conectat lucrătorul de la ghișeu – vezi figura 1.1. Ceea ce nu se vede pe ecran și este necunoscut lucrătorului (și multora

² Este adevărat că sunt bănci la care informațiile enumerate pot fi obținute doar printr-un efort traumatizant, însă de dragul exemplului să ne imaginăm că așa stau lucrurile.

dintre noi) este că informațiile afișate pe ecran sunt preluate dintr-un bazin de date aflat îndărătul uneia sau mai multor cutii de tablă numite uneori calculatoare.

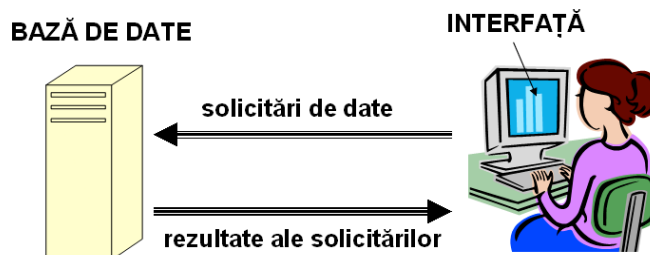


Figura 1.1. Schemă excesiv de simplistă a aplicațiilor ce utilizează baze de date

Figura 1.1 este simplistă până la irelevant. În realitate, baza de date poate fi pe același calculator pe care se afișează meniurile, ferestrele și rapoartele (lucru frecvent în aplicații mici și foarte mici realizate în Access sau FoxPro), dar, de cele mai multe ori, pe un alt calculator care se ocupă numai cu gestiunea datelor (serverul de baze de date). Pentru ca lucrurile să fie și mai interesante, între calculatorul lucrătorului și serverul pe care se află baza de date sunt interpusse alte *servere*, fiecare gestionând anumite module ale aplicației – vezi figura 1.2.

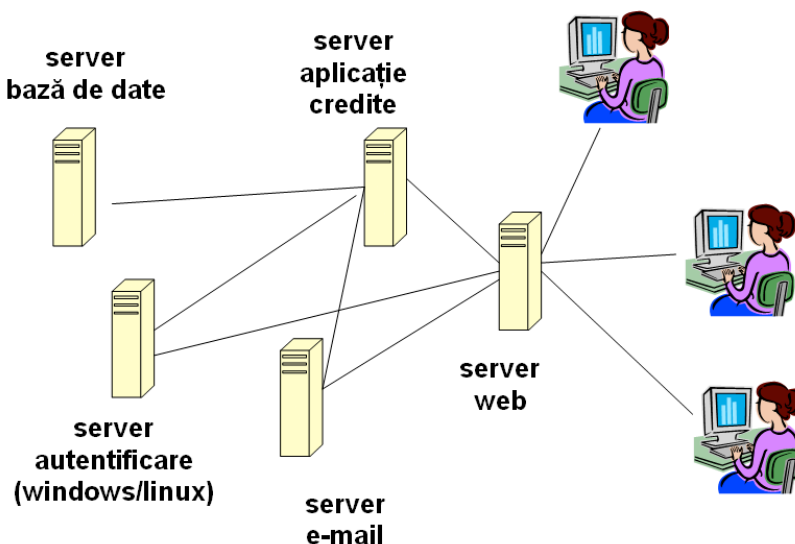


Figura 1.2. Schemă ceva mai complicată (dar tot inexactă) a aplicațiilor ce utilizează baze de date

Deși, poate, mai impresionantă, figura 1.2 conține destule inexactități. Astfel, pot exista și alte servere, în afara celor sugerate (de exemplu, servere pentru aplicații dedicate grupurilor de lucru (groupware), servere pentru gestiunea documentelor etc.). De fapt, aplicațiile se descompun pe procese și servicii, fiecare

server furnizând unul sau mai multe servicii. Apoi, trebuie spus că baza de date poate, la rândul său, să fie împrăștiată pe mai multe servere (vorbit de baze de date centralizate, distribuite, replicate). În al treilea rând, nu toți utilizatorii sunt de sex femeiesc și nu toate femeile din sistemul informatic al băncii au părul șaten & coadă/coc.

Ei bine, din toată această figură, în cartea de față ne interesează numai serverul de bază de date. Mai mult (de fapt, mai puțin !), dintre zecile de subiecte dedicate serverelor de baze de date, noi ne vom ocupa doar de câteva: cum creăm și folosim o bază de date, și, mai ales, cum o stocăm de informații ?

Există însă și persoane care folosesc baze de date fără a avea nevoie de formulare, meniuri și alte elemente de interfață. Se poate crea și gestiona o bază de date și în acest mod, mod căruia putem să-i spunem spartan (sau stoic). Însă asemenea gen de utilizatori trebuie să fie ași (sau măcar valeți) în ale bazelor de date. Or, o caracteristică esențială a bazelor de date este accesibilitatea. O bază de date este cu atât mai valoroasă cu cât pot avea acces la informațiile sale cât mai multe categorii de utilizatori (firește, fiecare cu drepturile și îndatoririle sale).

1.3. La început a fost fișierul (independent)

Schema din figura 1.2 este valabilă doar de câțiva ani încoace, însă mai toate aplicațiile informatice pentru organizații (firme, spitale, universități, bănci etc.) realizate în ultimii cincizeci de ani prezintă cel puțin două „straturi”: interfața și datele. Înainte de folosirea bazelor de date, datele erau organizate în fișiere independente gestionate prin programe scrise în limbaje precum COBOL, FORTRAN, C, Basic, Pascal etc.

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. Pornind de la precursorul FLOWMATIC, grupul cu pricina a redactat specificațiile 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 în românește drept *fișiere independente* – este reprezentată în figura 1.3.

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 (sercvențial, indexat, relativ etc.)) este obligatorie în toate programele care îl "citesc" sau modifică. Între FIȘIER1, FIȘIER2, ... FIȘIER m nu există nici o relație definită explicit.

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.

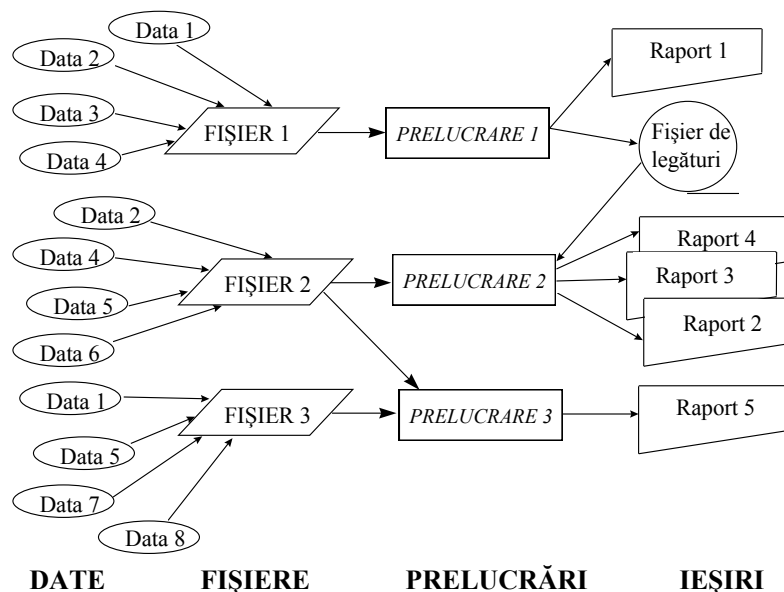


Figura 1.3. Sistem informatic bazat pe organizarea datelor în fișiere independente

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:

1. *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.
2. *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.

3. *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.
4. *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 aceleși 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.
5. Problemele de *securitate* țin de dificultatea creării unui mecanism care să protejeze pe deplin datele din fișiere de accesul neautorizat.
6. 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.
7. Inabilitatea de a obține răspunsuri rapide la *probleme ad-hoc*, atât de frecvente în lumea afacerilor.
8. *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.
9. *Inflexibilitatea* față de schimbările ulterioare, ce sunt inerente oricărui sistem informațional.
10. Modelarea indecvată a lumii reale.

Aceste dezavantaje sunt mai mult decât convingătoare, încât vă puteți întreba dacă au existat 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 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, precum și experiența acumulată de o largă categorie de profesioniști în ale IT-ului.

1.4.Ce este o bază de date ?

Bazele de date sunt, în general, percepute ca uriașe rezervoare informaționale în care sunt turnate (sau, după caz, aruncate) tot soiul de cifre, șiruri de caractere, ba chiar imagini, texte etc. în speranța că ar putea fi regăsite ulterior și (re)ordonate,

combinate și grupate, în funcție de nevoile utilizatorilor autorizați. În orice caz, o bază de date este *mare*, uneori imensă, *partajabilă* mai multor utilizatori și *persistentă*, adică poate fi stocată pe disc (sau orice suport) pe termen nelimitat³.

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 1.4.

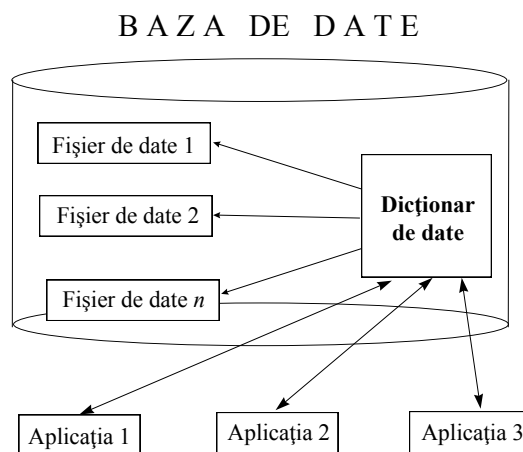


Figura 1.4. Schemă de principiu a unei baze de date

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 titlaturi: *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. Mai riguros, *BD poate fi definită ca o colecție de date aflate în interdependență, împreună cu descrierea datelor și a relațiilor dintre ele*, sau ca o

³ [Atzeni s.a. 1999] pp. 3-4

*colecție de date utilizată într-o organizație, colecție care este automatizată, partajată, definită riguros (formalizată) și controlată la nivel central*⁴.

Atunci când vorbim despre o bază de date, trebuie avute în vedere două aspecte fundamentale a 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. Corespunzător celor două aspecte complementare, schemă/conținut, limbajele de programare dedicate bazelor de date se împart în limbaje de definire a datelor (DDL – Data Definition Language) și limbaje de manipulare a datelor (DML – Data Manipulation Language). Limbajele uzuale dedicate bazelor de date, cum este SQL-ul, prezintă opțiuni atât pentru declararea structurii, cât și pentru editarea conținutului și consultarea/interogarea bazei.

1.5. Sisteme de gestiune a bazelor de date

Așa după cum fișierele de tip document au nevoie, pentru editare, de procesoare de texte (Word, WordPerfect etc.) sau după cum fișierele de tip foi de calcul (.XLS, WK1 etc.) au nevoie de programe de calcule tabelare (Excel, Lotus 1-2-3, Quattro Pro), și o bază de date necesită pentru creare, utilizare și administrare software specializat, și anume un *Sistem de Gestiune a Bazei de Date*.

Apărute în anii '60, Sistemele de Gestiune a Bazelor de Date (prescurtat SGBD-uri) reprezintă un ansamblu de programe ce permit utilizatorilor să interacționeze cu o bază de date, în vederea creării, actualizării și interogării acesteia. SGBD-ul este cel care asigură și supervizează: introducerea de informații în baza de date; actualizarea și extragerea datelor din bază; autorizarea și controlul accesului la date; păstrarea independenței dintre structura bazei și programe⁵.

Obiectivul esențial al unui SGBD este furnizarea unui mediu eficient, adaptat utilizatorilor care doresc să consulte sau să actualizeze informațiile conținute în bază. Bazele de date sunt concepute pentru a prelucra un volum mare de informații. Gestiunea acestora impune nu numai o structurare riguroasă a datelor, dar și o raționalizare a procedurilor de acces și prelucrare.

Prin urmare, principalele funcțiuni ale unui SGBD vizează:

- descrierea ansamblului de date la nivel fizic și conceptual;

⁴ [Everest1986], p.11

⁵ [G. Dodescu s.a. 1987], p. 511

- crearea (inițializarea) și exploatarea (consultarea și actualizarea) bazei de date;
- controlul integrității bazei;
- confidențialitatea informațiilor conținute în bază;
- accesul simultan al mai multor utilizatori la informații;
- securitatea în funcționare;
- furnizarea unui set de comenzi și instrucțiuni, necesare atât utilizatorilor pentru consultarea directă a bazei, prin intermediul unui limbaj de manipulare, cât și programatorilor, pentru redactarea programelor de lucru cu baza de date;
- revizia și restructurarea bazei;
- monitorizarea performanțelor.

Există diferențe considerabile între SGBD-urile aflate azi pe piață, în privința performanțelor și prețului. Access sau Visual FoxPro, ambele produse Microsoft, sunt SGBD-uri mai modeste în privința dimensiunii bazei de date. Visual FoxPro nu are, nici măcar în versiunea 9, opțiuni de declarare a utilizatorilor și grupurilor de utilizatori, cu atât mai mult opțiuni avansate de administrare. În schimb, este foarte generos în realizarea de meniuri, formulare și rapoarte (interfața aplicațiilor), ceea ce l-a făcut mult timp preferatul informaticienilor în dezvoltarea de aplicații mici și medii. Astăzi numele grele în domeniul bazelor de date sunt Oracle (Oracle), DB2 (IBM) și SQL Server (Microsoft), iar cei mai serioși competitori ai acestora sunt SGBD-urile *open-source* cum ar fi PostgreSQL, MySQL etc. Vom reveni la acest subiect.

Într-un sistem informatic ce utilizează BD, bazele de date sunt privite împreună cu SGBD-urile. Organizarea datelor poate fi analizată din mai multe puncte de vedere și pe diferite paliere. De obicei, abordarea se face pe trei niveluri: fizic sau intern, conceptual sau global și extern – vezi figura 1.5.

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.

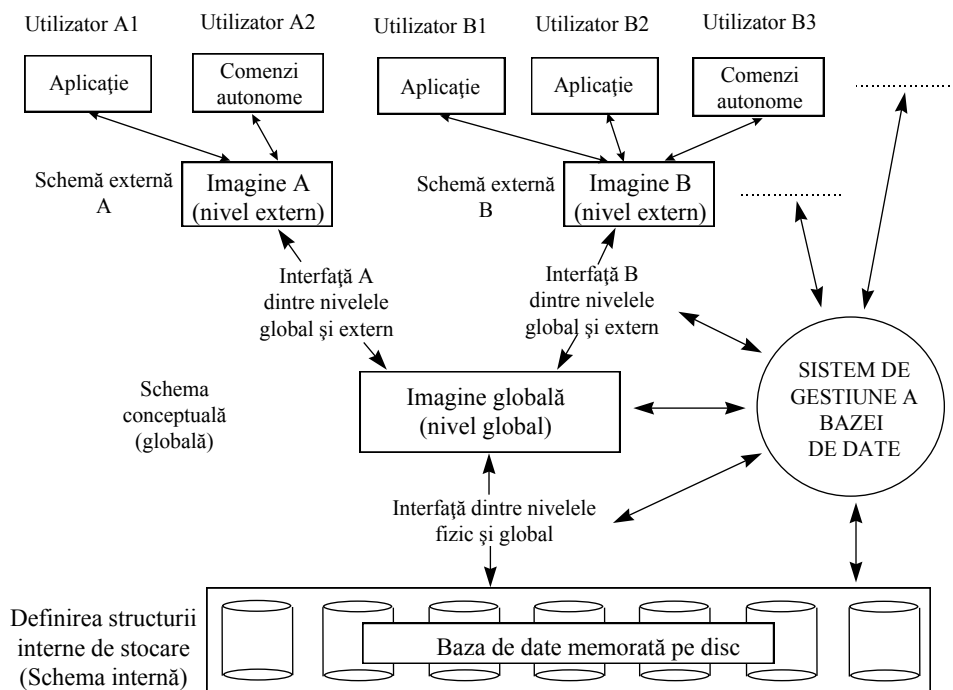


Figura 1.5. Schematizare a unui sistem de lucru cu o bază de date

Posibilitatea modificării structurii la un nivel, fără a afecta structura nivelului sau nivelurilor 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, mărimea 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.

Pe baza noilor elemente prezentate, putem să reluăm caracteristicile datelor stocate într-o BD⁶:

- partajabilitate – disponibilitate pentru un mare număr de utilizatori și aplicații;
- persistență – existență permanentă, din momentul preluării în bază până în momentul actualizării sau ștergerii (ștergere cu sau fără arhivare);
- 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, tranzacțiile pe care le reflectă aceste date;
- consistență – de multe ori, procesele/tranzacțiile sunt preluate în bază sub forma mai multor entități (sau atribute). Aceste entități/atribute trebuie să fie în concordanță unele cu celelalte, să respecte relațiile existente între aspectele proceselor reale pe care se reflectă;
- 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.

1.6.Module, limbaje și utilizatori ale SGBD-urilor

Realizarea unui SGBD reprezintă un demers extrem de complex, ținând seama de volumul datelor de stocat și prelucrat, numărul utilizatorilor conectați simultan la bază, viteza de onorare a solicitărilor informaționale, precum și cerințele de integritate și securitate. Majoritatea covârșitoare a celor ce lucrează cu bazele de date se descurcă onorabil fără a ști detaliile de organizare a SGBD-ului, ba chiar fără să fie conștienți de existența SGBD-ului. Alții, precum administratorii de baze de date, sunt obligați să cunoască destul de multe despre organizarea fizică, indecși, buffere, jurnal, drepturi ale utilizatorilor etc., iar alții chiar lucrează în echipe care “fabrică” SGBD-uri, adică le proiectează, realizează anumite module, le testează etc.

Pentru a avea o idee generală despre modulele esențiale ale unui SGBD apelăm la o schemă “clasică” prezentată în figura 1.6⁷. După spusele autorilor – figuri foarte cunoscute în lumea bazelor de date -, dreptunghiurile simple reprezintă module ale sistemului, dreptunghiurile duble - structuri de date rezidente în memorie, săgețile cu liniatură simplă indică fluxuri și de date și de control, în timp ce săgețile punctate reprezintă fluxuri (numai) de date.

⁶ Unii autori, precum [Atzeni s.a. 1999], adaugă explicit drept caracteristică dimensiunea foarte mare a bazei de date. În cea mai mare parte a lucrărilor dedicate bazele de date, această caracteristică este una implicită.

⁷ Preluare din [Garcia-Molina s.a. 2002], p.11

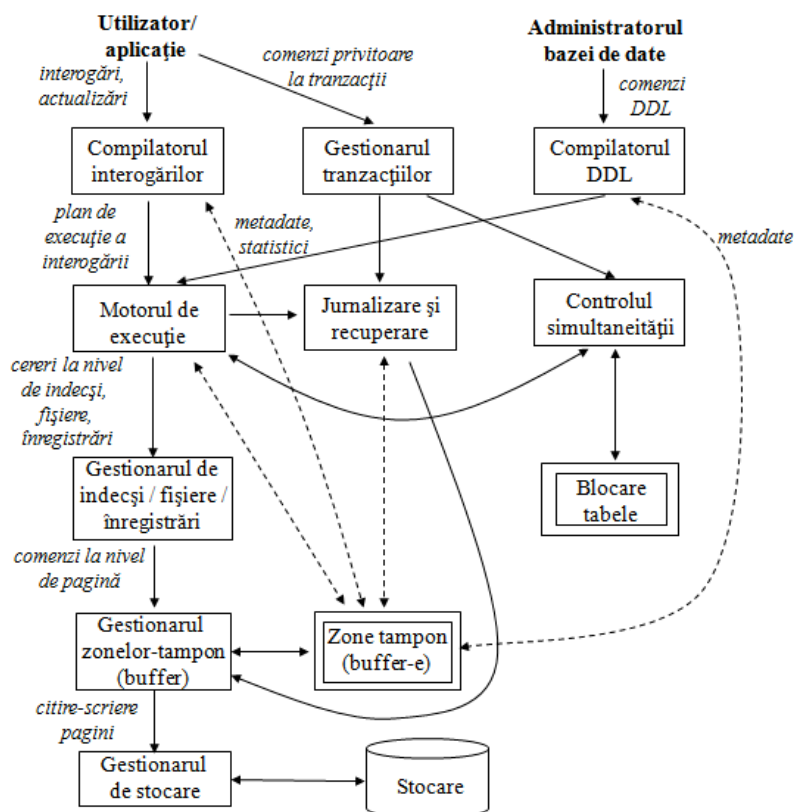


Figura 1.6. Structura unui sistem de lucru cu o bază de date

1.6.1. Module principale

Schema de mai sus este destul de impresionantă (cel puțin pentru mine), deși este vorba de o reprezentare simplificată a ceea ce întâmplă în "inima" SGBD-ului. Sistemul nu „scrie” datele din memorie pe disc chiar imediat după preluarea lor, ci la anumite intervale de timp. De aceea, o componentă importantă, pe lângă gestionarul stocării, este gestionarul buffer-elor. Accesul rapid la date este imposibil fără indecși, astfel încât în figură apare gestionarul indecșilor/fișierelor/înregistrărilor. Arbitrajul mai multor utilizatori/module ce lucrează simultan cu baza de date, și, implicit, rezolvarea eventualelor blocaje, este asigurat de modulul de control al simultaneității (concurenței).

Păstrarea integrității și coerenței bazei de date presupune gruparea operațiunilor în tranzații pe principiul „totul sau nimic”. Căderea uneia dintre operațiunile tranzației atrage în sine căderea întregii tranzații și revenirea (restaurarea) bazei de date „pe starea” dinaintea tranzației. Acest mecanism tranzații-

onal care presupune jurnalizarea cade sub incidența gestionarului de tranzacții și a modulului de jurnalizare și recuperare.

1.6.2. Limbaje de definire a datelor

Figura 1.6 pune în evidență cele două compilatoare care corespund categoriilor clasice de limbaje pentru lucrul cu bazele de date – DDL (Data Definition Language) care este un limbaj destinat gestionării structurii bazei și DML (Data Manipulation Language) care este un limbaj pentru gestionarea conținutului.

Arhitectura unei baze de date este specificată printr-o serie de definiții redactate sub formă de instrucțiuni scrise în limbajul de definire a datelor - DDL (Data Definition Language). Execuția acestor definiții se materializează într-un ansamblu de tabele, tabele virtuale, proceduri stocate, secvențe etc. care sunt memorate într-un fișier special, denumit *dicționar* (sau repertor) *de date*. Un dicționar de date conține deci *metadate*, adică date relative la alte date, fiind consultat înaintea oricărei citiri sau modificări a datelor din bază.

Principale funcțiuni ale DDL sunt:

- Descrierea logică a bazei de date și sub-schemelor proprii fiecărui grup de utilizatori.
- Identificarea schemei, sub-schemelor și diverselor agregări de date.
- Specificarea fișierelor de date și a legăturilor logice dintre acestea. Pe baza acestor specificații se poate realiza accesul la date chiar și în condițiile coexistenței mai multor modele de organizare într-o aceeași BD.
- Definirea restricțiilor semantice la care sunt supuse datele, restricții care se referă la ansamblul valorilor permise fiecărei date, eventual formula de calcul a unei date pe baza valorilor altor date. Respectarea acestor restricții asigură coerența bazei.
- Definirea cheilor de acces rapid și a cheilor confidențiale (parolelelor).
- Definirea metodelor de „exploatare” a fișierelor ce vor fi utilizate în aplicații pentru selectarea înregistrărilor.
- Definirea procedurilor speciale de criptare, în vederea generării cheilor de acces.
- Definirea modalităților de indexare și localizare ale entităților.
- Determinarea tipului unei date, *de bază* sau *derivată* (calculată printr-o expresie, pe baza valorilor altor date).

Dată fiind complexitatea structurilor de memorare și metodelor de acces la acestea, la nivel elementar fișierele de date și dicționarul de date sunt accesibile numai unui număr restrâns de conașseuri. În schimb, pentru descrierea schemei BD, marea majoritate a limbajelor de interogare prezintă comenzi adecvate, ce pot fi utilizate și de non-informaticieni.

Cel mai important limbaj dedicat bazelor de date, SQL – despre care vom discuta începând cu al treilea capitol, prezintă comenzi DDL cum ar fi: CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE/ALTER/DROP VIEW, CREATE/DROP INDEX, CREATE/DROP PROCEDURE, CREATE/DROP TRIGGER, CREATE/DROP SEQUENCE etc.

1.6.3. Limbaje de manipulare a datelor

Prin manipularea datelor se înțelege efectuarea uneia dintre următoarele operațiuni:

- extragerea unor date din bază (consultare);
- scrierea de noi date în bază (adăugare);
- ștergerea datelor perimate sau eronate (uneori chiar și a celor corecte);
- modificarea valorii unor date.

Un limbaj de manipulare a datelor (DML - Data Manipulation Language) este utilizat pentru a prelucra datele în funcție de structura lor. La nivel fizic, prin DML se vizează identificarea și implementarea unor algoritmi performanți de acces la date, în timp ce la nivel extern DML trebuie să faciliteze dialogul utilizatorului cu baza în vederea obținerii informațiilor dorite.

În mod curent, termenul *consultare* sau *interogare* desemnează acțiunea de căutare și identificare a datelor necesare, dintre cele aflate în bază. Ansamblul instrucțiunilor DML pentru căutarea și pentru identificarea datelor constituie limbajul de consultare. Revenind la SQL, cele mai cunoscute comenzi DML sunt: INSERT (pentru adăugarea de înregistrări într-o tabelă), DELETE (pentru ștergerea de linii dintr-o tabelă), UPDATE (pentru modificarea valorii unuia sau mai multor attribute, pe una sau mai multe înregistrări ale unei tabelă) și SELECT (pentru extragerea de informații din una sau mai multe tabelă).

1.6.4. Limbaje de control al datelor

Un element mai puțin sugerat în figura 1.6 este grupul de comenzi prin care se declară utilizatorii, grupurile de utilizatori (profilurile), precum și drepturile fiecărui utilizator/profil la obiecte din schema bazei (tabelă, tabelă virtuale, proceduri etc.). Aceste comenzi sunt arondate unui al treilea tip de limbaj destinat bazelor de date – DCL (Data Control Language). Dintre acestea, în standardul și dialectele SQL cele mai frecvente sunt: CREATE USER, DROP USER, CREATE/DROP ROLE, GRANT și REVOKE.

1.7. Utilizatorii bazelor de date

În figura 1.6 (stânga, sus) au acces la baza de date utilizatori/aplicații care pot lansa fie interogări și actualizări (adresate compilatorului de interogări), fie comenzi privitoare la tranzacții (COMMIT – care declară încheierea cu succes a unei tranzacții și ROLLBACK – care abandonează o tranzacție). Tot în partea de sus, dar în dreapta, este reprezentat personajul cheie în exploatarea unei baze de date – administratorul bazei de date. Cei mai puțini vizibili în figura 1.6 sunt cei care participă la realizarea unui SGBD, figura fiind interesată de cei care folosesc bazele de date și SGBD-ul din momentul „punerii în funcțiune” a aplicației cu baze de date.

1.7.1. Utilizatori mai mult sau mai puțin curenți

Cei mai numeroși utilizatori ai unei aplicații ce folosesc baze de date sunt și cei mai „nevinovați” în ale bazelor de date. Doamnele (și/sau domnișoarele) din figurile 1.1 și 1.2 au la dispoziție meniuri și formulare pentru a introduce operațiuni, și rapoarte pentru a extrage informațiile necesare. Fluxurile dintre ecran/claviatură/mouse și baza de date (locul unde se află, de fapt, informațiile) sunt invizibile acestor utilizatori.

Realizarea aplicațiilor care afișează pe ecran meniurile, formularele și rapoartele și care preiau datele introduse în baze cade în sarcina unei categorii profesionale destul de pestrițe denumită „dezvoltatori de aplicații cu baze de date”. Zicem că este pestriță deoarece aici intră de-a valma: analiști și proiectanți de sisteme informaționale, proiectanți de baze de date (pot fi aceeași sau diferiți de analiști/proiectanții de sisteme informaționale), programatori de interfețe .NET, Java, PHP etc., programatori de baze de date Oracle (PL/SQL), SQL Server (Transact-SQL), PostgreSQL (plpgSQL), DB2 (SQL PL) etc., testerii s.a.m.d. Firește că, pentru ceea ce discutăm în cartea de față, programatorii de baze de date reprezintă o țință privilegiată.

Dezvoltatorii de aplicații sunt implicați în fazele de realizare a aplicației (proiectare, programare, implementare/instalare). După darea în folosință, aplicațiile pot suferi modificări de mai mică sau mai mare amploare (datorate schimbărilor din legislația economică, regândirea modalităților de derulare a unor operațiuni în cadrul organizației), modificări ce cad tot în (co)responsabilitatea dezvoltatorilor. Deși sunt implicați în etapele de realizare a aplicației (unde pot formula cerințe, observații și proteste legate de modul în care funcționează modulele aplicației), utilizatorii curenți sunt destinatarii principali ai aplicației din momentul instalării (implementării) acesteia, cei care o vor folosi pentru a introduce operațiuni și a obține informații/rapoarte.

Între categoria „nevinovaților” și cea a dezvoltatorilor de aplicații cu baze de date există multe nuanțe de utilizatori ce dispun de suficiente cunoștințe care să le permită obținerea de informații din bază sau chiar să modifice o serie de date direct, fără mijlocirea aplicației. Prin urmare, există utilizatori care nu sunt nici administratori, nici dezvoltatori, dar care sunt în stare (și au drepturi suficiente) să lanseze interogări (fraze SELECT), comenzi de editare a înregistrărilor (INSERT/UPDATE/DELETE) sau să-și gestioneze „propriile” tabele, proceduri, tabele virtuale etc. Stilul acesta direct cu baza seamănă cu lucrul cu un ferăstrău electric fără apărătoare, deoarece chiar și în condițiile folosirii tranzațiilor, câteva minute de neatenție pot antrena săptămâni întregi de migrene.

1.7.2. Administratorul bazei de date

Folosind limbajul mafiot, se poate spune că administratorul unei baze de date este „il capo di tutti capi”, sau, în cel castrist, „il lider maximo”. Administratorul unei baze de date este persoana responsabilă de sistem în ansamblul său. Rolul acestuia este determinant în:

- Definirea arhitecturii bazei de date, realizată prin redactarea definițiilor care vor fi transformate de compilatorul DDL în tabele și alte obiecte stocate permanent în dicționarul de date;
- Definirea modalităților în care va fi structurată memoria externă și a metodelor de acces la date;
- Modificarea arhitecturii și organizării fizice a bazei de date;
- Autorizarea accesului la date se acordă fiecărui utilizator sau grup (rol) al bazei de date, administratorul fiind cel care decide asupra datelor ce pot fi consultate și actualizate de fiecare utilizator sau grup de utilizatori;
- Specificarea restricțiilor de integritate care sunt stocate pe disc și consultate de gestionarul bazei la fiecare actualizare.

În plus, administratorul bazei de date este cel care: asigură legătura cu utilizatorii; definește procedurile de verificare a drepturilor de acces și a procedurilor de validare a integrității datelor; definește strategia de salvare (înregistrarea copiilor de siguranță)/restaurare a bazei; monitorizează performanțele bazei și o adaptează la modificările ulterioare ale sistemului informațional.

Dat fiind că are mână liberă în crearea, utilizarea și optimizarea bazei, în acordarea de drepturi utilizatorilor, administratorul trebuie să fie nu numai un bun profesionist, dar și o persoană de caracter. Un supărăcios sau frustrat poate nenoroci ireversibil tot capitalul informațional al firmei. Pe de altă parte, și managerii ar trebui să se poarte cât mai frumos cu administratorul bazei de date (indiferent de ceea ce gândesc despre el).

1.8. Modele de organizare a datelor în BD

Nucleul unei baze de date îl reprezintă dicționarul de date ce conține structura bazei. Analiza, proiectarea și implementarea structurii (schemei) bazei se realizează utilizând un model de date, model ce 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 din BD este o reprezentare a obiectelor lumii reale și a evenimentelor asociate lor. Presupune un demers de abstractizare care se concentrează pe aspectele esențiale ale organizației/aplicației, furnizând conceptele 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 nivelurile 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-uri) lucrau cu baze de date ierarhice. Structura datelor este prezentată sub forma unui arbore - vezi figura 1.7 -, 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. Orice fiu poate fi și tată, deci poate avea, la rândul său, fii.

Astfel, în clasa a IX-a A (profil Uman) există doi elevi: Pop I. Vasile ce are numărul matricol 4545 și domiciliază pe strada Primăverii nr. 22, iar al doilea, Ion V. Viorel, are matricolul 4550 și locuiește pe strada Corupției nr. 13 bis. Vasile a luat la fizică (profesor Cernat Dan, născut pe 21 aprilie 1968) un 5 pe 8 februarie 2002 și un 9 pe 15 februarie. La istorie (profesor Pal Dana, născută pe 12 noiembrie 1975) stă ceva mai bine, având un 8 (căpătat pe 21 februarie), un 7 (pe 28 februarie) și un 8 (pe 7 martie). În același mod poate fi interpretată figura pentru Ion V. Viorel, cu mențiunea specială că istoria nu e punctul lui forte.

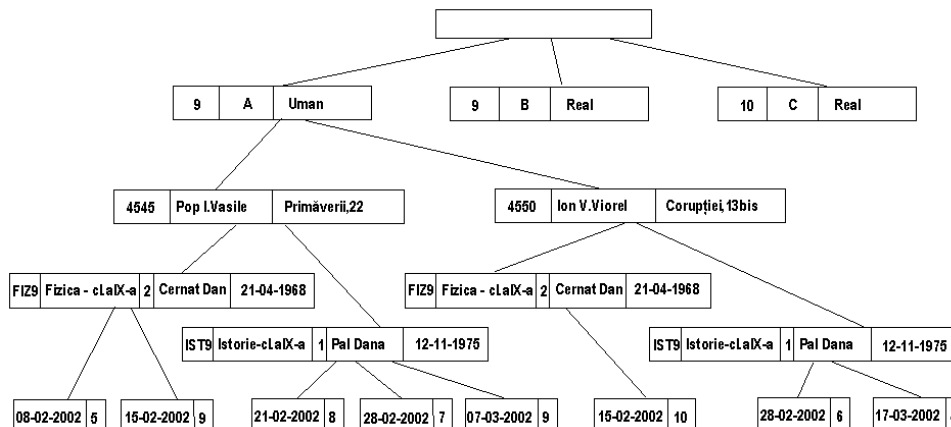


Figura 1.7. Arbore de structură specific modelului ierarhic

Modul în care înregistrările se înlanțuiesc presupune folosirea unor *pointeri* (un soi de adrese fizice) ce nu sunt reprezentați în figură (din considerente umanitare). Aflarea informațiilor din baza de date presupune navigarea între înregistrări cu ajutorul pointerilor. În plus, modificarea structurii bazei de date presupune actualizarea înlanțuirii pointerilor, ceea ce este extrem de laborios. A altă problemă a acestui model o reprezintă imposibilitatea reprezentării cazului în care un copil este, pardon, "rezultatul" a mai mulți tați. Situația nu este atât de scandaloaasă cum v-ați închipuit, și aceasta doarece modelul ierarhic nu face nici o referire la mama fiului.

Figura 1.7 este relevantă pentru gradul de redundanță impus de reprezentarea ierarhică. Codul, denumirea, numărul de ore ale fiecărei discipline, precum și datele despre profesor, apar pentru fiecare elev, deși acestea sunt comune la nivel de an de studii (numele disciplinei, numărul de ore) sau la nivel de clasă (profesorul titular).

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 cu o structură asemănătoare celei din figura 1.8. Navigarea se face tot prin pointeri.

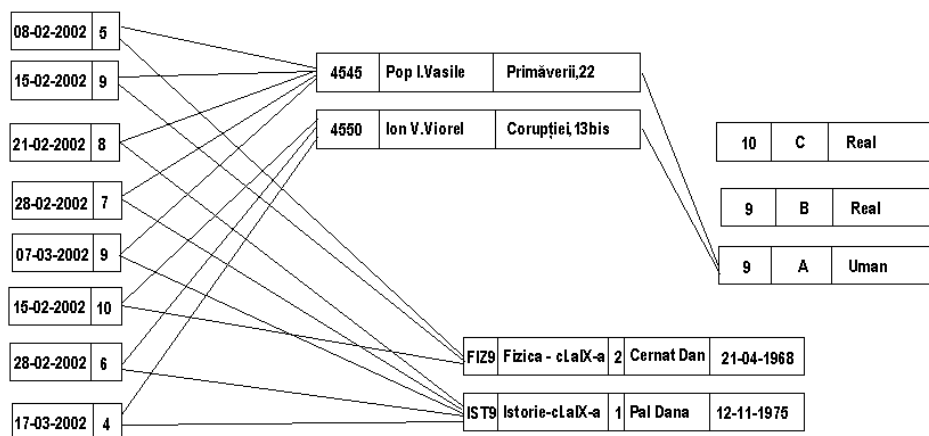


Figura 1.8. Graf specific reprezentării utilizând modelul rețea

Modelul rețea elimină o serie de probleme specifice modelului ierarhic. Spre exemplu, cazul ilustrat în figura 1.7 poate fi reprezentat după logica modelului rețea ca în figura 1.8. Se observă că fiecare disciplină (și profesorul care o predă la clasa respectivă) apare o singură dată, fiind legată prin linii de toate notele obținute de fiecare dintre elevi. Legăturile dintre înregistrări se fac tot cu pointeri, iar modificarea înlănțuirii pointerilor rămâne o operațiune foarte migăloasă.

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 îi rezervăm un capitol special, următorul.

Modelul obiectual. În programare, orientarea pe obiecte își are începuturile în anii '60, consacrarea în anii '80, iar gloria după anii '90. Începând cu anii '90, orientarea pe obiecte capătă importanță majoră și în analiză și proiectare, reușind să se constituie ca alternativă viabilă metodologiilor structurate. Pe baza acestui succes, s-a crezut că și în materie de baze de date, modelul obiectual îl va surclasa pe cel relațional până în 2000. Rezultatele sunt însă deprimante pentru suporterii OO-ului, piața SGBD OO fiind sub 6% din valoarea totală a pieței bazelor de date⁸.

Modelul relațional-obiectual. Este un model mai recent ce încearcă să valorifice deopotrivă atuurile relaționalului și ale orientării 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.

Pe lângă aceste modele de organizare efectivă a datelor în baze, există și modele pur teoretice, utile analiștilor și proiectanților de sisteme informaționale și aplicații,

⁸ Acest procent a rămas valabil de la prima ediție a cărții, nefiind exclus ca mărimea sa să fie, de fapt, supraevaluată.

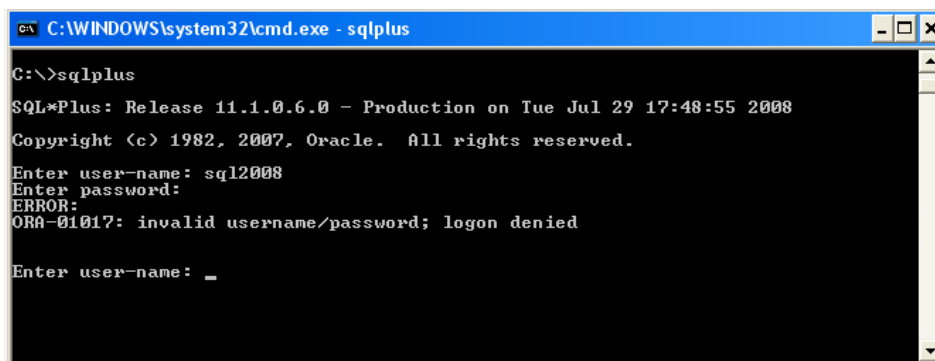
dar care nu au fost încă implementate. Probabil cel mai cunoscut reprezentant al acestei categorii de *modele semantice* este modelul *entitate-relație*.

1.9. Noțiuni preliminare în lucrul cu bazele de date

Încercam să vă conving în paragraful 1.5 că nu putem lucra cu bazele de date decât prin intermediul sistemelor de gestiune a bazelor de date (SGBD), de care ne vom ocupa și în paragraful 2.4 (dedicat SGBD-urilor construite pe logica modelului relațional). În continuare, dorim să lămurim câteva noțiuni legate de practica folosirii unui SGBD, și anume: conexiune, sesiune, script, modul de program și tranzacție.

Conexiuni, sesiuni & clienți

Dacă vrem să adăugăm, modificăm sau ștergem informații dintr-o bază de date, sau chiar și dacă numai vrem să „aruncăm un ochi” la câteva informații aflate în bază, este necesar să stabilim o conexiune la baza de date, prin intermediul SGBD-ului. Dacă vom reuși, vom fi *clienții* bazei de date, calculatorul pe care lucrăm fiind *platforma client*. SGBD-ul cu ajutorul căruia vom avea acces la bază este *serverul bazei de date*. Comanda folosită în multe SGBD-uri este chiar CONNECT. Pentru ca să fim luați în serios, la conectare trebuie să introducem un nume de utilizator și o parolă pe care serverul să le recunoască, altfel tocim claviatura degeaba – vezi figura 1.9.



```
C:\WINDOWS\system32\cmd.exe - sqlplus

C:\>sqlplus

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Jul 29 17:48:55 2008
Copyright (c) 1982, 2007, Oracle. All rights reserved.

Enter user-name: sql2008
Enter password:
ERROR:
ORA-01017: invalid username/password; logon denied

Enter user-name: _
```

Figura 1.9. Tentativă eșuată de conectare la serverul Oracle (din SQL*Plus) 11g

Mă grăbesc să diluez câteva dintre eventualele malentendu-uri. Mai întâi, SGBD-urile actuale nu au interfața atât de neguroasă. Doar administratorii/dezvoltatorii, din dorința de a impresiona, sau masochiștii (din alte dorințe) mai lucrează azi în „mod text” (uneori cele două categorii se intersectează în bună măsură). Restul oamenilor nu-prea-pricepuți și/sau normali preferă interfețe mai prietenoase, după cum o să vedem nu peste mult timp. În al doilea rând, o conexiune la baza de date nu înseamnă nicidecum accesul la toată baza de date, ci doar la unele obiecte, pe care utilizatorul le poate „vedea” și, eventual, modifica.

Prin conexiune, avem acces la o fereastră a bazei. Conexiunea presupune ca, în prealabil, cineva să creeze utilizatorul, să-i atribuie o parolă, și să-i confere anumite drepturi la obiecte ale bazei de date. Acest (semi)demiurg creator ia, de cele mai multe ori, forma dușmănoasă a administratorului bazei de date.

Între utilizatorul care se chinuie să se conecteze la bază și baza propriu-zisă poate fi o distanță de zeci, sute, mii ... de calculatoare (uneori și kilometri). Calculatorul pe care lucrează utilizatorul va adresa cererea de access, iar SGBD-ul dispune de un serviciu de ascultare (numit de obicei chiar *listener* - ascultător) care preia cererea, iar apoi verifică corectitudinea numelui și parolei, și, dacă lucrurile sunt în regulă, se alocă un spațiu în memoria serverului special pentru această conexiune în care utilizatorul urmează să se dezlănțuie. În plus, arhitecturile web actuale prezintă servere de aplicații care acționează ca intermediari între utilizatori (clienți) și serverul de bază de date.

```

C:\WINDOWS\system32\cmd.exe - sqlplus

Table dropped.

SQL> -- acum incepem o sesiune <de lucru>
SQL> SELECT CURRENT_DATE FROM dual ;

CURRENT_D
-----
30-JUL-08

SQL> CREATE TABLE sterge1 (x NUMERIC(4), y VARCHAR2(15)) ;

Table created.

SQL> INSERT INTO sterge1 VALUES (12, 'Prima linie');

1 row created.

SQL> INSERT INTO sterge1 VALUES (14, 'A doua linie');

1 row created.

SQL> -- acum incheiem tranzactia <cu succes>
SQL> COMMIT ;

Commit complete.

SQL> SELECT * FROM sterge1 ;

      X Y
-----
    12 Prima linie
    14 A doua linie

SQL> -- acum terminam sesiunea, prin deconectare de la server
SQL> DISCONNECT
Disconnected from Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL>

```

Figura 1.10. Exemplu simplist de conexiune/tranzacție/comenzi Oracle 11g (SQL*Plus)

De-conectarea și re-conectarea la baza de date se face prin comenzi succesive DISCONNECT/CONNECT – vezi figura 1.10. O *sesiune* de lucru începe din momentul în care un client se conectează la un server de bază de date și se termină în momentul de-conectării de la server – vezi figura 1.10. Într-o sesiune pot fi lansate în execuție comenzi și module de program.

După cum spuneam mai sus, ar fi destul de traumatizant să lucrăm cu o bază de date beneficiind de o interfață în mod text, cum este cea din figurile 1.9 și 1.10 de

lucru. De aceea, fiecare server de baze de date dispune de interfețe grafice îmbietoare. Unele sunt parte integrantă a software-ului pentru baza de date. Este cazul MS SQL Server (*Management Studio*) – vezi figura 1.11.

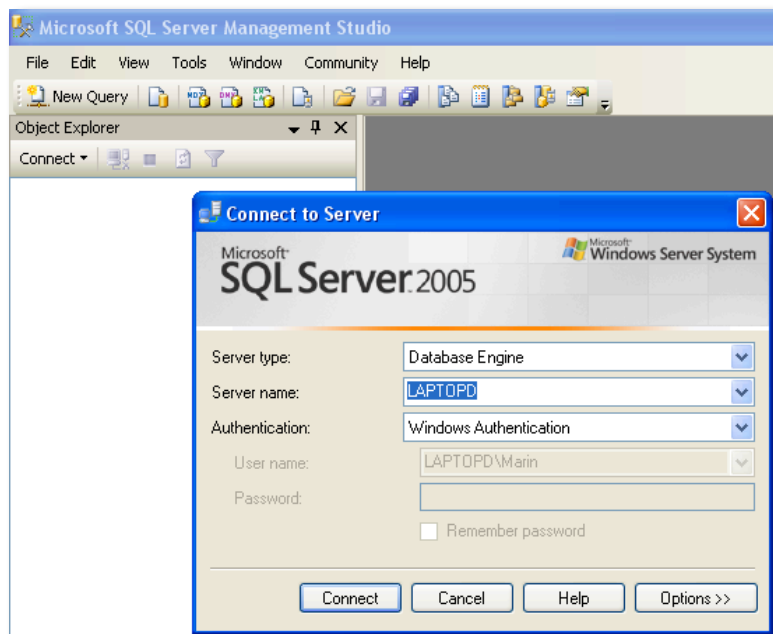


Figura. 1.11. Conectarea la MS SQL Server (prin Management Studio)

Similar, IBM DB2 dispune de un modul numit *Control Center* prin care se poate realiza conectarea la baza de date – vezi figura 1.12. Alte aplicații-client pot fi instalate separat, cum ar fi *pgAdmin* dedicat PostgreSQL (vezi figura 1.13) sau *SQL Developer* pentru Oracle.

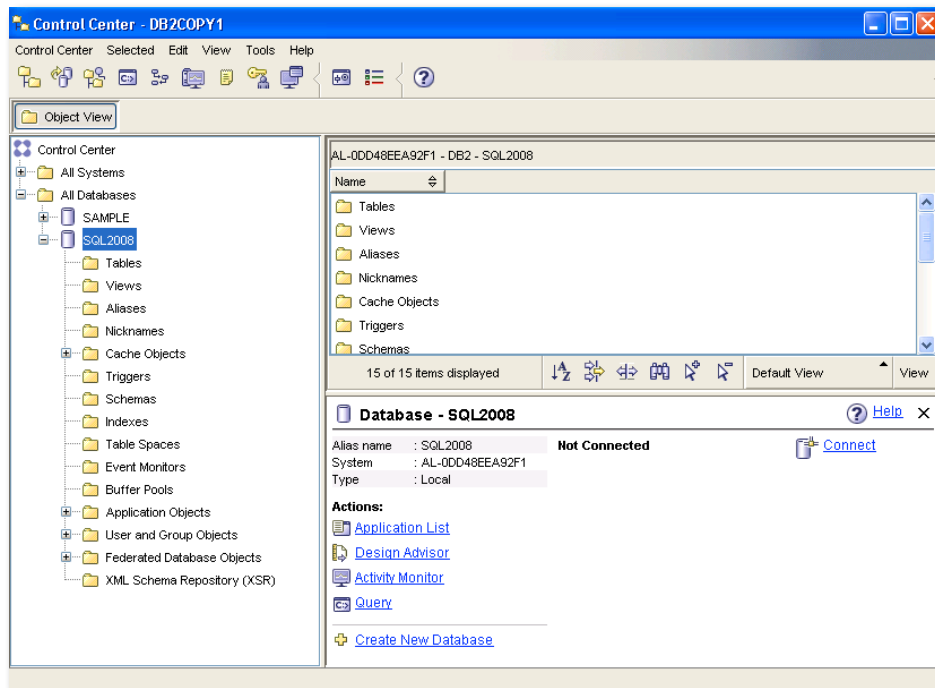


Figura. 1.12. Conectarea la IBM DB2 (prin Control Center)

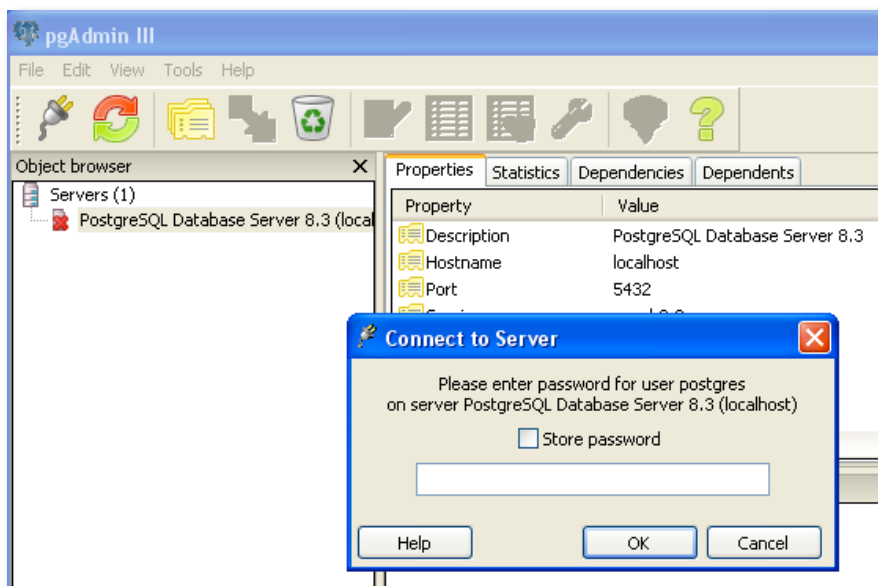


Figura. 1.13. Conectarea la PostgreSQL (prin pgAdmin)

Deși produs de firma Oracle, deci destinat cu precădere acestui server, SQL Developer-ul este ceva mai ambițios, fiind în stare să realizeze conexiuni și cu baze de date create cu Microsoft Access – vezi figura 1.14⁹. Există și clienți „independenți”, unii dedicați unui singur SGBD (ex. *SQL Navigator* produs de Quest Software, *SQLDetective* produs de Conquest Software, *PL/SQL Developer* produs de Allround Automations sunt destinate dezvoltării de aplicații cu baze de date Oracle), iar alții asigură conexiunea la două sau mai multe dintre SGBD-urile actuale, cum ar fi *TOAD* (Quest Software).

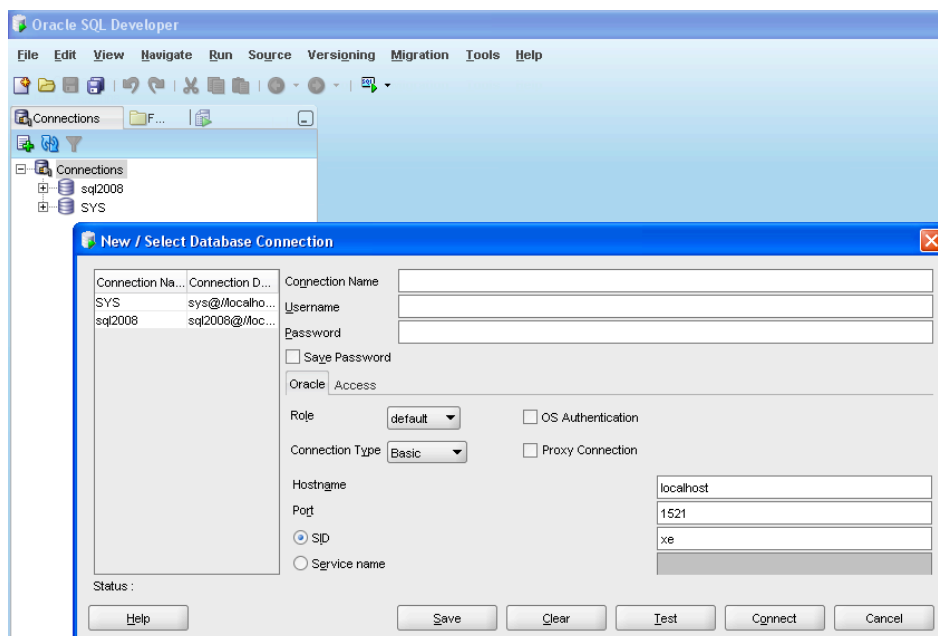


Figura. 1.14. SQLDeveloper - un „client” pentru conectarea la baze de date Oracle sau Access

Comenzi, scripturi și module de program

Odată conectat la bază, un utilizator poate crea, modifica sau șterge obiecte – celebrele comenzi DDL (cum este CREATE TABLE în figura 1.10) -, sau consulta/modifica conținutul bazei de date – comenzi DML (cum sunt comenzile INSERT și SELECT din figura 1.10). Ar mai fi și comenzile DCL prin care se creează utilizatori /grupuri și se acordă/revocă drepturi la obiecte din bază, dar acestea sunt pentru administratori. Comenzile pot fi lansate direct, rând pe rând, sau pot fi grupate în module de program. În această carte vom folosi, de câteva ori, și termenul *script* și pe cel de *program/modul*.

⁹ Nu cunosc niciun utilizator de Access care să folosească SQL Developer, însă atunci când se dorește trecerea bazei de date din Access în Oracle, probabil că SQL Developer-ul este de real folos.

Pentru lucrarea de față, un script este un fișier (de obicei ASCII/text) care adună, una după alta, comenzi DDL și/sau DML (sau chiar DCL). Un script *nu* conține structuri de control alternative sau repetitive (IF-uri, LOOP-uri, etc.), structuri ce constituie substanța unui modul de program. Un modul de program este redactat într-un limbaj anume (C, Java, Basic, PL/SQL, T-SQL etc.), în timp ce un script nu are nicio legătură cu un limbaj anume. Important este ca sintaxa comenzilor din script să fie acceptată de SGBD-ul pe care se lansează în execuție.

Tranzacții

Și acesta este un subiect asupra căruia vom reveni pe parcursul emisiunilor. Pentru început, să convenim că o tranzacție este o succesiune de comenzi (de obicei, DML) care funcționează pe principiul toți-pentru-unul, unul-pentru-toți sau, mai bine zis, toul sau nimic. De exemplu, grupăm într-o tranzacție 50 de comenzi DML (inserare/modificare/ștergere), comenzi lansate individual, dintr-unul sau mai multe scripturi, și/sau module de program. 49 merg strună, dar a 50-a generează o eroare în baza de date (o restricție încălcată, să zicem). Din cauza acestei ultime comenzi buclușe, toate celelalte 49 de comenzi pot fi anulate, din rațiuni de solidaritate, printr-o comandă specială, ROLLBACK. Dacă și a 50-a funcționează cum trebuie, declarăm printr-o altă comandă - COMMIT - că tranzacția s-a încheiat cu succes, iar modificările pot fi consemnate definitiv în bază. Lucrul cu tranzacții este foarte important în păstrarea corectitudinii/integrității bazei de date, după cum vom vedea în câteva dintre capitolele viitoare.