2 MEDIUL BAZELOR DE DATE

2.1 Arhitectura bazei de date cu 3 nivele

Asigurarea independenţei fizice şi logice a datelor impune adoptarea unei arhitecturi organizată pe cel puțin 3 nivele (arhitectura ANSI-SPARC):

1. nivelul intern (baza de date fizică)
2. nivelul conceptual
3. nivelul extern

Obiectivul arhitecturii cu 3 nivele este **separarea vederii fiecărui utilizator** asupra bazei de date de modul în care este ea reprezentată fizic.

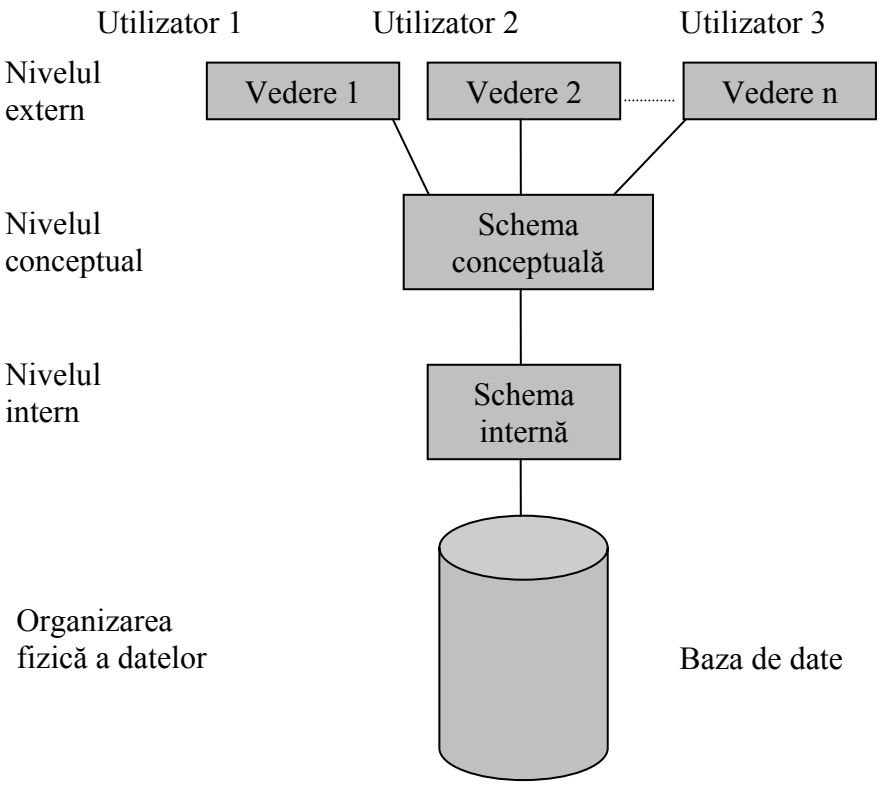


Fig. 2.1 Arhitectura bazei de date cu 3 nivele

Modul în care **utilizatorii** percep datele este numit **nivel extern**. Modul în care **SGBD şi sistemul de operare** percep datele este numit **nivel intern**. **Nivelul conceptual** realizează atât ***transpunerea*** cât şi ***independența*** dorită dintre nivelul extern şi cel intern.

2.1.1 Nivelul intern

Nivelul intern	Reprezentarea fizică a bazei de date pe calculator. Acest nivel descrie ***CUM*** sunt stocate datele în baza de date.

Nivelul intern (baza de date fizică) este o **colecție de fişiere** conținând ***datele fizice*** la care se adaugă diverse ***structuri auxiliare*** menite să asigure accesul operativ la date. Structurile auxiliare pot fi: directoare, indexuri, pointeri, tabele de dispersie. Modul de organizare a bazei de

date fizice este în mare măsură influiențat de **configurația echipamentelor hardware** care suportă baza de date şi de **sistemul de operare**. Schimbarea sistemului de operare sau modificări în configurația hardware pot atrage modificări ale bazei de date fizice. Dacă este satisfăcută condiția de **independență fizică**, aceste modificări în nivelul intern al bazei de date nu vor ataca nivelele superioare ale acesteia.

Nivelul intern tratează chestiuni cum ar fi:

- alocarea spațiului de stocare pentru date şi indexuri
- descrierea înregistrărilor pentru stocare (cu dimensiunile de stocare pentru date)
- plasarea înregistrărilor
- tehnici de comprimare a datelor şi de codificare a acestora

2.1.2 Nivelul conceptual

Nivelul conceptual	Este o vedere generală a bază de date. Acest nivel descrie ***CE*** date sunt stocate în bază de date şi ***RELAȚIILE*** dintre acestea.

Nivelul conceptual conține ***structura logică*** a bazei de date, aşa cum este ea văzută de administratorul bazei de date. Fiecare bază de date are un model conceptual propriu prin care sunt numite şi descrise toate entitățile logice din baza de date împreună cu legăturile dintre acestea. El reprezintă o imagine completă a cerințelor organizației privind datele.

Ex: în descrierea bazei de date a unei intreprinderi pot apărea concepte ca: angajat, produse, furnizor, beneficiar, etc.

Modelul conceptual **integrează viziunile tuturor utilizatorilor** asupra bazei de date, fiind rezultatul unui compromis între cerințele diferiților utilizatori. Nivelul conceptual reprezintă:

- toate entitățile, atributele şi relațiile dintre ele
- constrângeri asupra datelor
- informații semnatice asupra datelor
- informații privind securitatea şi integritatea

De reținut că **modelul conceptual este o descriere a conținutului de date** din baza de date şi **NU** cuprinde nici un fel de referire la modul de memorare a datelor sau la strategia de acces. De ex. descrierea unei entități trebuie să conțină numai tipurile de date (integer, real, character) şi lungimea lor (numărul maxim de cifre sau caractere) dar nici o informație privind stocarea, cum ar fi numărul de octeți ocupat.

2.1.3 Nivelul extern

Nivelul extern	Reprezintă ***vederea utilizatorului*** asupra bază de date. Acest nivel descrie acea parte a bazei de date care este relevantă pentru fiecare utilizator.

Nivelul extern este cel mai apropiat utilizatorului. Este ceea ce vede acesta din baza de date, sau modul cum vede acesta baza de date. Modelul extern este derivat din cel conceptual dar poate prezenta deosebiri substanțiale față de acesta. Un termen deseori folosit pentru modelul extern este acela de ***vedere*** sau ***viziune***. Prin aceste viziuni, utilizatorii au acces doar la părți bine definite din baza de date, fiindu-le ascunse părțile care nu interesează. Prin modelul extern se

realizează **independența logică** a datelor. Fiecărei viziuni îi corespunde o descriere în termenii entităților logice din modelul conceptual.

Diferite vederi pot avea reprezentări diferite ale aceloraşi date. De ex, un utilizator poate vedea datele calendaristice în format an-lună-zi, altul le poate vedea ca zi-lună-an.Vederile pot include chiar date combinate sau derivate din entități diferite.

2.2 Limbajele bazelor de date

Limbajele bazelor de date sunt împărțite în 2 categorii: limbaje de definire a datelor (**DDL**) şi limbaje de manipulare a datelor (**DML**). DDL este utilizat pentru a specifica schema bazei de date, iar DML este utilizat pentru citirea şi reactualizarea bazei de date.

Aceste limbaje sunt numite **sublimbaje de date** deoarece ele nu includ construcții pentru toate necesitățile de calcul, cum sunt cele asigurate de limbajele de nivel înalt. Multe SGBD au o facilitate de încorporare a sublimbajului într-un limbaj de programare de nivel înalt, cum sunt COBOL, Pascal, C, etc. În acest caz, limbajul de nivel înalt se numeşte **limbaj gazdă**. Pentru a compila fişierul încorporat, mai întâi comenzile specifice sublimbajului de date sunt înlocuite prin apelări de funcții. Apoi fişierul preprocesat este compilat şi rezultatul este plasat într-un modul obiect, legat legat la o librărie care conține funcțiile ănlocuite.

2.2.1 Limbajul de definire a datelor (DDL)

DDL Este un **limbaj descriptiv**, care permite *administratorului* bazei de date sau *utilizatorului* **să descrie şi să denumească entitățile** cerute de aplicație şi **relațiile** care pot exista între diferitele entități.

Rezultatul compilării instrucțiunilor DDL este un set de tabele stocate în fişiere speciale, denumite global **catalog de sistem**. Acesta conține **meta-datele** – adică datele care descriu obiectele din baza de date.

2.2.2 Limbajul de manipulare a datelor (DML)

DML Asigură un set de procedee ce permit operații de bază pentru manipularea datelor din bază de date:

- inserarea de date noi
- modificări de date
- regăsirea datelor
- ştergerea de date

Limbajele DML pot fi de două tipuri: **procedurale** şi **neprocedurale**.

- **procedurale** specifică modul ***cum*** trebuie să fie obținut rezultatul unei instrucțiuni DML
- **neprocedurale** descriu numai ***ce*** rezultat trebuie obținut

De ex, SQL este un limbaj neprocedural.

Curs 3

2.3 Modele de date şi modelarea conceptuală

Model de date	Este o colecție integrată de concepte necesare ***descrierii*** datelor, ***relațiilor*** dintre date şi ***constrângerilor*** impuse datelor.

Ne putem imagina că un model de date are următoarele trei componente:

1. o parte **structurală**, constând dintr-un set de reguli conform cărora sunt construite bazele de date;
2. o parte de **manipulare**, definind tipurile de operații care sunt permise asupra datelor
3. un set de **reguli de integritate**, care garantează că datele sunt corecte.

Scopul unui model este să reprezinte datele şi să le facă înțelese.

Pentru arhitectura ANSI - SPARC a bazei de date, se pot identifica trei modele de date:

1. un model de date extern, pentru a reprezenta vederea fiecărui utilizator
2. un model de date conceptual, pentru a reprezenta vederea logică, generală, care este independentă de SGBD
3. un model de date intern, pentru a reprezenta schema conceptuală, în aşa fel încât să poată fi înțeleasă de SGBD

Modelele de date se pot clasifica în trei categorii principale:

1. modele de date bazate pe **obiecte**
2. modele de date bazate pe **înregistrări**
3. modele de date **fizice**

Ultima categorie, modelele de date fizice descriu **cum** sunt stocate datele pe calculator. Reprezintă informații despre **structura şi ordinea înregistrărilor şi căile de acces**. Nu există la fel de multe modele de date fizice ca cele logice, motiv pentru care vom prezenta mai amănunțit numai primele două categorii de modele de date.

2.3.1 Modele de date bazate pe obiecte

În modelele de date bazate pe obiecte se utilizează concepte ca: entitate, atribut, relație.

O **entitate** este un obiect distinct (persoană, loc, lucru, concept, eveniment) care va fi reprezentat în baza de date.

Un **atribut** este o proprietate care descrie un anumit aspect al obiectului pe care dorim să-l înregistrăm.

O **relație** este o asociere între entități.

Cele mai obişnuite modele de date bazate pe obiecte sunt:

- **Entitate – Relație** constituie una din tehnicile principale de proiectare conceptuală a bazelor de date
- **semantic**
- **funcțional**
- **orientat spre obiecte** extinde definiția unei entități, pentru a include şi atributele care descriu ***starea*** obiectului şi acțiunile acestuia, respectiv ***comportamentul***. Se spune că obiectul **încapsulează** atât starea cât şi comportamentul

2.3.2 Modele de date bazate pe înregistrări

Într-un astfel de model, baza de date constă dintr-un număr de înregistrări cu format fix, posibil de tipuri diferite. Fiecare tip de înregistrare defineşte un număr fix de câmpuri, fiecare având o lungime fixă.

Există 3 tipuri principale de modele de date bazate pe înregistrări

- **relațional**
- **în rețea**
- **ierarhic**

} au fost realizate cu aproape 10 ani înaintea celui relațional, așa încât legăturile lor cu conceptele tradiționale de prelucrare a fișierelor sunt mai evidente

Modelul de date relațional

- Se bazează pe conceptul de **relații matematice**
- Datele şi relațiile sunt reprezentate sub formă de **tabele**, fiecare având un număr de coloane cu o denumire unică

Observăm ca modelul de date relațional necesită ca utilizatorul să perceapă baza de date ca fiind formată din tabele. Dar această percepție se aplică numai la structura logică (*Nivelul Extern* şi *Nivelul Conceptual* din arhitectura pe 3 nivele a bazei de date), nu şi structurii fizice, care poate fi implementată utilizând o varietate de modalități de stocare.

Majoritatea sistemelor moderne sunt bazate pe modelul relațional.

Exemplu: presupunem că din baza de date a unei organizații cu mai multe filiale alegem să reprezentăm datele despre filiale şi personalul angajat prin două tabele, cu următoarea structură:

Filiale

NrFil	*Adresa*	*Orasul*	*CodPostal*	*Telefon*	*Fax*
F3	Rozelor 25	Timişoara	1700	121212	121222
F4	Stejeriş 19	Braşov	2200	232323	232333
F5	Eroilor 35	Timişoara	1700	434343	434333
F6	Unirii 10	Focşani	1500	454545	454555

Angajați

NrMarca	*Nume*	*Prenume*	*Adresa*	*Orasul*	*Functia*	*Salariul*	*NrFil*
214	Burcea	Ion	Lalelelor 12	Timişoara	manager	5000	F3
215	Gheorghe	Alina	Cetăți 21	Timişoara	contabil	4000	F3
216	Turcea	Elena	Varte 8	Braşov	secretara	3000	F4
217	Vasile	Valentin	Gării 32	Timişoara	portar	200	F5

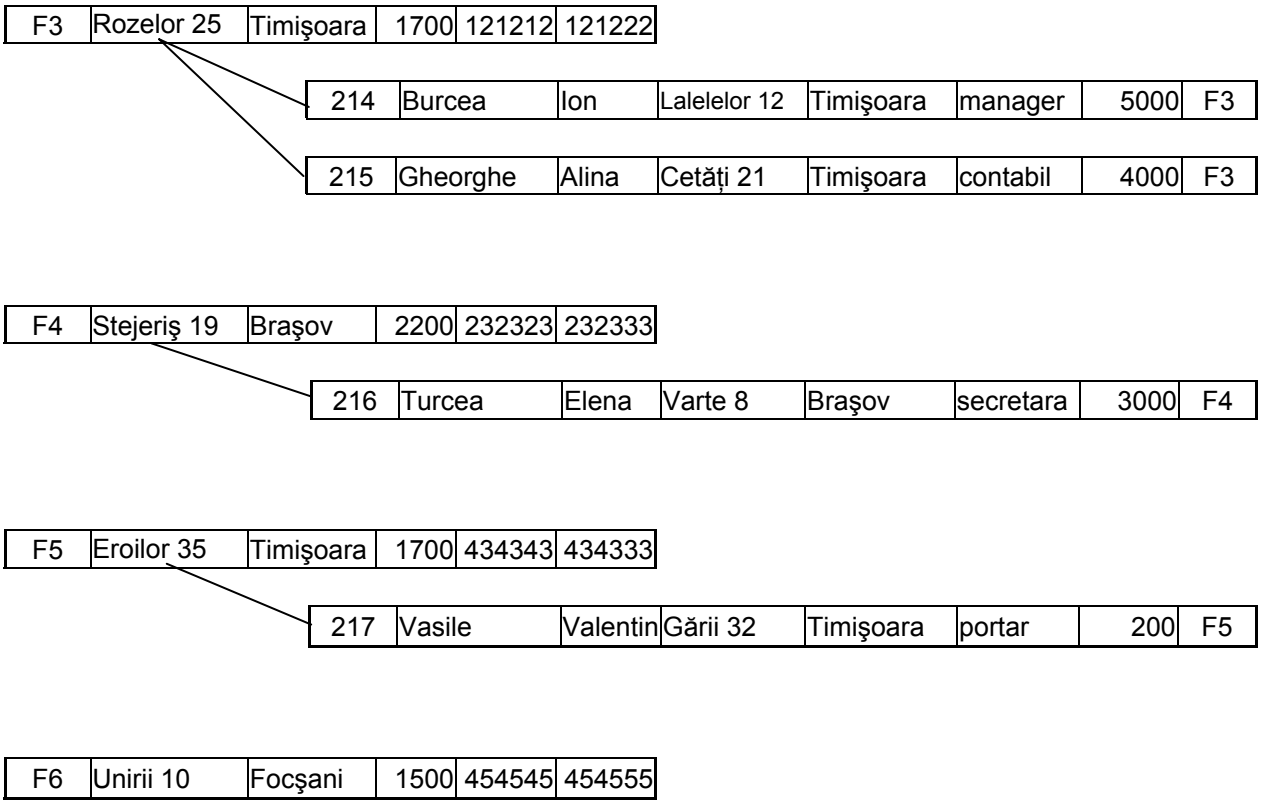
Modelul de date în rețea

- Datele sunt reprezentate printr-o **colecție de înregistrări**
- Relațiile sunt reprezentate prin **direcții**

Spre deosebire de modelul relațional, aici relațiile sunt modelate explicit prin direcții, care devin pointeri în implementrea propriu-zisă.

Modelul poate fi asemănat cu o structură de **grafuri**, cu înregistrările reprezentate ca **noduri** şi direcțiile ca **muchii**.

Să vedem **exemplul** de mai sus transpus într-un model de date în rețea:



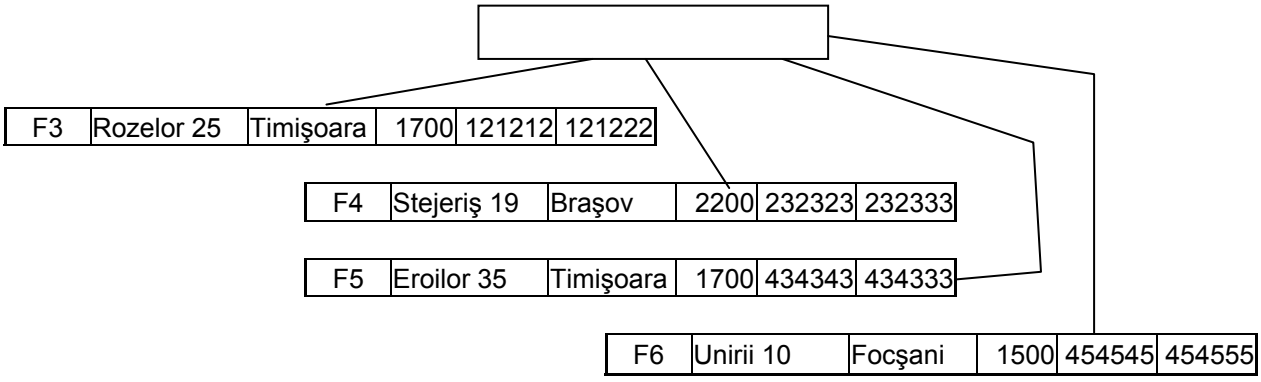
Modelul de date ierarhic

Constituie un tip restrâns de model în rețea.

- Datele sunt reprezentate printr-o **colecție de înregistrări**
- Relațiile sunt reprezentate prin **direcții**
- Permite ca un **nod să posede numai un singur părinte**

Poate fi reprezentat ca un graf de tip arbore.

Să vedem acelaşi **exemplu** de mai sus transpus într-un model de date ierarhic:



Modele de date bazate pe înregistrări sunt utilizate pentru a specifica **structura generală a bazei de date şi o descriere de nivel superior a implementării acesteia**. Principalul lor dezavantaj este că nu pun la dispoziție facilități de specificare explicită a constrângerilor asupra datelor.

Modelele relaționale adoptă o abordare ***declarativă*** pentru procesarea bazei de date (adică specifică ***ce*** date vor fi regăsite), iar **modelele în rețea şi ierarhice** adoptă o abordare ***navigațională*** (specificând ***cum*** vor fi regăsite datele).

2.3.3 Modelarea conceptuală

Printr-o examinare a arhitecturii cu trei nivele, se poate observa că schema conceptuală este ”**inima**” bazei de date. Ea suportă toate vederile externe şi este suportată de schema internă. Totuşi, schema internă este doar o implementare fizică a schemei conceptuale. Schema conceptuală trebuie să fie o **implementare completă şi corectă a cerinţelor companiei** (organizaţiei) privind datele. Dacă acest lucru nu este realizat, o serie de informaţii despre companie vor lipsi sau vor fi reprezentate incorect, ceea ce va crea dificultăţi în implementarea completă a uneia sau mai multor vederi externe.

Modelarea conceptuală sau proiectarea conceptuală a bazei de date este procesul de construire a unui model de informaţii utilizate într-o companie, care este independent de detaliile de implementare, cum ar fi sistemul SGBD, programele aplicaţie, limbajele de programare sau orice alte tipuri de consideraţii fizice. Acest model de date se numeşte **model de date conceptual**.

2.4 Funcţiile unui SGBD

Au fost stabilite 8 servicii pe care trebuie să le furnizeze un SGBD complet.

1. ***stocarea, regăsirea şi reactualizarea datelor.*** Aceasta este funcţia fundamentală a unui SGBD. Pentru asigurarea ei, sistemul SGBD trebuie să ascundă faţă de utilizator detaliile privind implementarea fizică internă (organizarea fişierelor şi structurile de stocare).
2. ***catalog accesibil utilizatorului***
3. ***asigurarea tranzacţiilor***

Un SGBD trebuie să furnizeze un mecanism care să garanteze că sunt efectuate **toate** reactualizările corespunzătoare unei anumite tranzacţii sau că nu se efectuează **nici una**. O **tranzacţie** constă într-o serie de acţiuni realizate de un singur utilizator sau un program aplicaţie, prin care se accesează sau se schimbă conţinutul bazei de date.

4. ***servicii de control concurente***

Un SGBD trebuie să furnizeze un mecanism care să garanteze că baza de date este corect reactualizată, atunci când *mai mulţi utilizatori* efectuează simultan astfel de operaţii

5. ***servicii de reconstituire***

Un SGBD trebuie să furnizeze un mecanism de reconstituire a bazei de date dacă aceasta este deteriorată într-un fel oarecare.

6. ***servicii de autorizare***

Un SGBD trebuie să furnizeze un mecanism care să garanteze că numai utilizatorii autorizaţi pot accesa baza de date.

7. ***suport pentru comunicarea datelor***

Un SGBD trebuie să poată fi integrat unui software de comunicaţie.

8. ***servicii de integritate***

Un SGBD trebuie să furnizeze mijloace care să asigure că, atât datele din baza de date, cât şi modificările acestora respectă anumite reguli. Integritatea se referă la **corectitudinea** şi **coerenţa** datelor stocate.

2.5 Componentele software ale unui SGBD

Din punct de vedere software, sistemele SGBD sunt foarte complexe şi sofisticate, deoarece modulele software din componenţa unui SGBD trebuie să permită furnizarea tuturor serviciilor analizate în paragraful precedent. Structura componentelor software ale unui SGBD nu poate fi generalizată, deoarece ea variază foarte mult de la un sistem de gestiune la altul.

Totuşi este util să încercăm o trecere în revistă a componentelor soft şi a relaţiilor dintre ele. În acest scop vom prezenta o posibilă arhitectură pentru un SGBD.

Un SGBD este partiţionat în diverse componente software (module), responsabile de câte o operaţie specifică. Câteva dintre funcţiile SGBD sunt susţinute de sistemul de operare. Dar sistemul de operare oferă numai serviciile de bază, iar SGBD trebuie construit peste acesta. Principalele componente software ale unui mediu SGBD sunt reprezentate în Fig. 2.2. Să explicităm semnificaţia următoarelor componente:

- *Procesorul de interogare* transformă interogările într-o serie de instrucţiuni de nivel jos, adresate administratorului bazei de date.

- *Administratorul bazei de date* realizează interfaţa bazei de date cu programele aplicaţie şi interogările lansate de utilizatori.

- *Administratorul de fişiere* manipulează **fişierele de stocare** aflate la bază şi administrează alocarea **spaţiului de stocare** pe disc. El stabileşte şi menţine lista de structuri şi indexuri definite în schema internă. El nu gestionează direct intrările şi ieşirile de date, ci transmite cererea către o *metodă de acces* corespunzătoare, care fie citeşte datele din *bufferul sistemului*, fie le scrie în acesta.

- *Preprocesorul DML*. Acest modul converteşte instrucţiunile DML încorporate într-un program aplicaţie în apelări de funcţii standard din limbajul gazdă. Preprocesorul DML trebuie să interacţioneze cu *procesorul de interogare*, pentru a genera codul corespunzător.

- *Compilatorul DDL* transformă instrucţiunile DDL într-un set de tabele care conţin meta-datele. Aceste tabele sunt ulterior stocate în catalogul de sistem, iar informaţiile de control sunt stocate în anteturile fişierelor de date.

- *Administratorul de catalog* gestionează accesul şi întreţinerea catalogului de sistem. Catalogul de sistem este accesat de majoritatea componentelor sistemului SGBD.

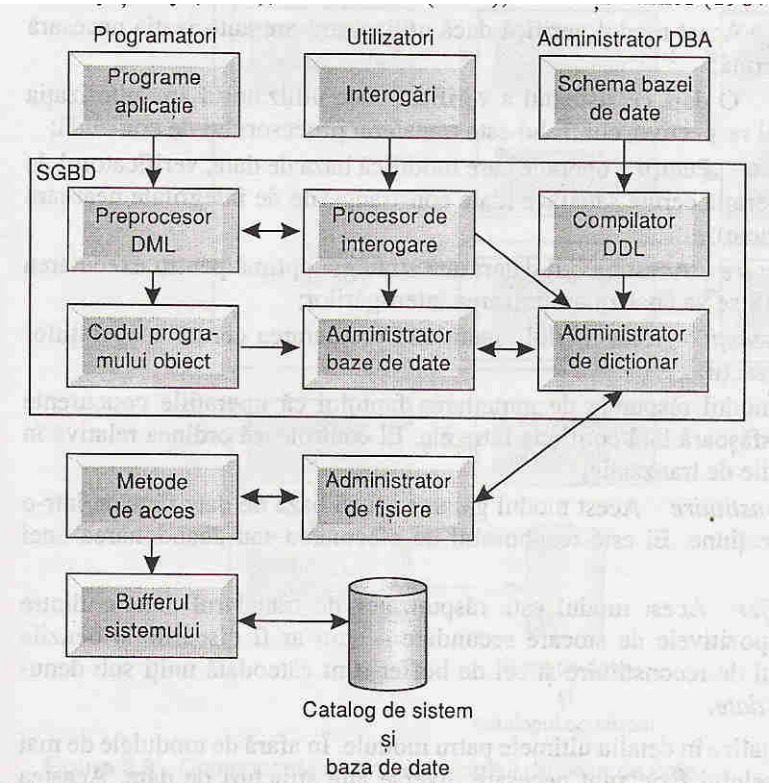


Fig. 2.2 Componentele principale ale unui SGBD

2.6 Arhitecturi multiutilizator

Pentru implementarea **SGBD multiutilizator**, uzual se folosesc trei tipuri de arhitecturi, pe care le vom prezenta în continuare.

2.6.1 Teleprocesarea

Arhitectura tradiţională pentru sistemele multiutilizator a fost teleprocesarea, la care există un calculator cu o singură unitate CPU şi un număr de terminale, aşa cum este ilustrat în Fig. 2.3. **Toată procesarea este efectuată pe acelaşi calculator.** De obicei, terminalele utilizatorilor sunt ”neştiutoare”, incapabile să funcţioneze singure. Ele sunt legate la calculatorul central. Prin intermediul subsistemului de control al comunicaţiilor din sistemul de operare, terminalele trimit mesaje la programele aplicaţie ale utilizatorilor, care la rândul lor, utilizează serviciile sistemului SGBD. În acelaşi mod, mesajele sunt transmise înapoi la terminalul utilizatorului. Din păcate, această arhitectură a plasat **o povară teribilă** asupra calculatorului central, care, pe lângă rularea programelor aplicaţie PA şi a sistemului SGBD, trebuie să preia şi o cantitate semnificativă de muncă din partea terminalelor (cum ar fi formatarea datelor pentru afişarea pe ecran).

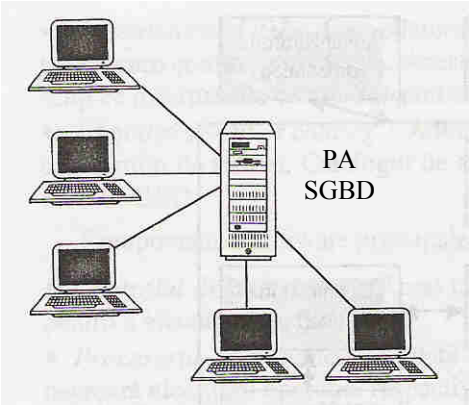


Fig. 2.3 Teleprocesarea

O dată cu dezvoltarea calculatoarelor personale de înaltă performanţă şi cu dezvoltarea reţelelor de comunicaţie dintre calculatoare, în industrie există o tendinţă majoră spre **miniaturizare**, care înseamnă înlocuirea calculatoarelor mainframe scumpe cu reţele de calculatoare personale - mai avantajoase ca preţ - care obţin rezultate identice sau chiar superioare. Această tendinţă a dat naştere la următoarele două arhitecturi - **fişier-server** şi **client-server** - care vor fi analizate în continuare.

2.6.2 Arhitectura fişier – server

Într-un mediu fişier-server, **procesarea este distribuită în reţea**, de obicei, o reţea locală (LAN). Arhitectura fişier-server cuprinde fişierele cerute de programele aplicaţii PA şi sistemul SGBD. Totuşi, **aplicaţiile şi sistemul SGBD sunt executate pe fiecare staţie de lucru**, solicitând, când este necesar, fişiere de la serverul de fişiere, aşa cum este ilustrat în Fig. 2.4.

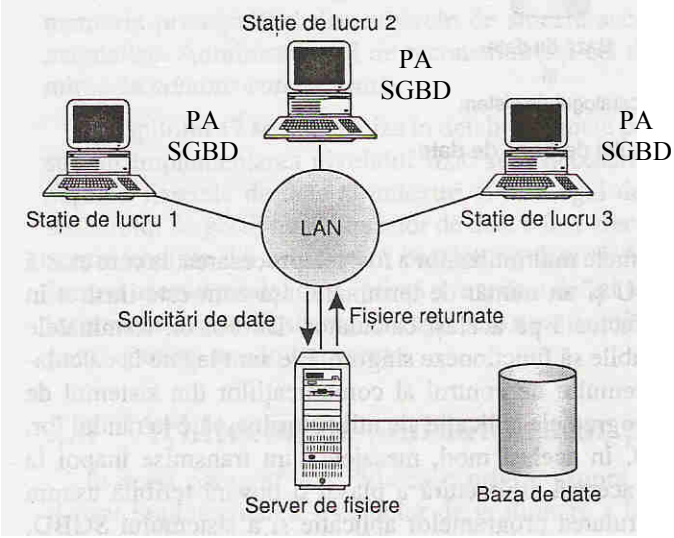


Fig. 2.4 Arhitectura fişier - server

In acest mod, serverul de fişiere acţionează pur şi simplu **ca o unitate de hard-disc partajată**. Sistemul SGBD de pe fiecare staţie de lucru trimite serverului de fişiere cererile pentru toate datele necesare, care sunt stocate pe disc. Această abordare poate genera un trafic intens pe reţea, ceea ce poate duce la apariţia unor probleme privind performanţele. De exemplu, să considerăm cererea unui utilizator, care conţine numele membrilor personalului care lucrează la filiala din 163 Main St. În limbajul SQL (a se vedea Capitolul 13) această cerere este exprimată astfel:

SELECT Prenume, Nume **FROM** Filiala b, Personal s **WHERE** b.nrFil=s.nrPer **AND** b.Strada='163 Main St';

Cum serverul de fişiere nu cunoaşte limbajul SQL, sistemul SGBD trebuie să-i ceară fişierele corespunzătoarere relaţiilor Filiala şi Personal în locul numelor membrilor personalului care satisfac interogarea.

Rezultă că arhitectura fisier-server are trei dezavantaje principale:

1. Există un **trafic intens pe reţea**;
2. Este necesară o copie completă a sistemului **SGBD pe fiecare staţie de lucru**;
3. Controlul **simultaneităţii**, **reconstituirii** şi **integrităţii** este mult mai complex, deoarece acelaşi fisier poate fi accesat de mai multe sisteme SGBD.

2.6.3 Arhitectura client –server

Arhitectura client - server a fost dezvoltată pentru a depăşi dezavantajele primelor două abordări. Arhitectura client - server se referă la modul în care interacţionează componentele de software pentru a forma un sistem: există un proces **client, care necesită câteva resurse**, şi un **server, care oferă resurse**. Nu există nici o cerinţă ca atât clientul, cât şi serverul să se afle pe acelaşi calculator. In practică, se obişnuieşte să se plaseze serverul pe un sit din reţeaua locală şi clienţii pe alte situri. In Fig. 2.5 se ilustrează arhitectura client - server, iar în Fig. 2.6 sunt prezentate câteva combinaţii posibile ale topologiei client – server.

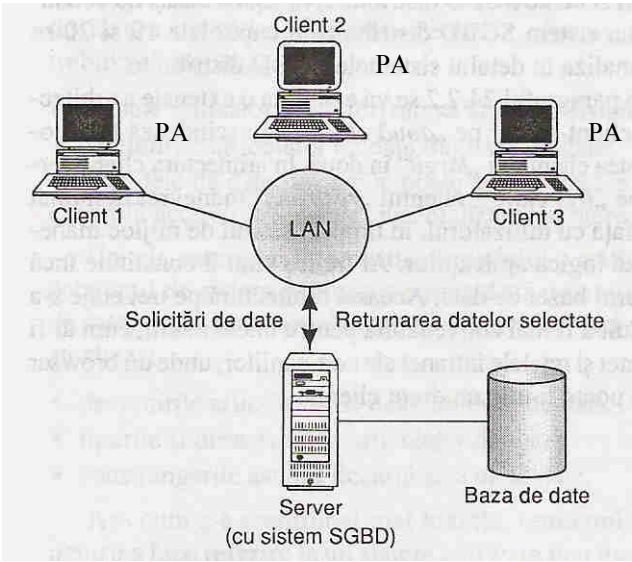


Fig. 2.5 Arhitectura client – server

În contextul bazelor de date, **clientul administrează interfaţa cu utilizatorul şi logica aplicaţiei**, acţionând ca o *staţie de lucru*, sofisticată, pe care sunt executate aplicaţiile bazei de date. Clientul preia cererea utilizatorului, verifică sintaxa şi generează cererea pentru baza de date în limbajul SQL sau în alt limbaj de baze de date, adecvat logicii aplicaţiei. Pe urmă, transmite mesajul serverului, aşteaptă un răspuns şi îl formatează pentru utilizatorul final. **Serverul acceptă şi procesează cererea** pentru baza de date, după care transmite rezultatul înapoi clientului. **Procesarea** implică:

1. verificarea autorizării
2. asigurarea integrităţii
3. menţinerea catalogului de sistem
4. execuţia proceselor de interogare şi reactualizare
5. asigurarea controlului simultaneităţii şi reconstituirii.

Operațiile clientului şi serverului sunt prezentate în Tabelul 2.1.

Acest tip de arhitectură are următoarele avantaje:

- permite un **acces mai larg la bazele de date** existente;
- are performanțe crescute - dacă clienții şi serverul se află pe calculatoare diferite, atunci diferite unități CPU **pot procesa aplicații în paralel**; **reglarea serverului** este mai uşor de efectuat, dacă singura sa sarcină este de a efectua prelucrarea bazei de date;
- **reduce costurile dispozitivelor hardware** - numai serverul necesită o capacitate de stocare şi o putere de prelucrare suficiente pentru a stoca şi gestiona baza de date;
- **reduce costurile comunicațiilor** - aplicațiile execută o parte din operații la client, care trimite prin rețea numai cererea de acces la baza de date, ceea ce face ca pe rețea să circule mai putine date;
- **măreşte coerența** - serverul poate trata verificările de integritate, deoarece **constrângerile trebuie definite şi validate într-un singur loc**, fără să fie necesar ca fiecare program aplicație să execute propriile verificări;
- se transpune destul de natural într-o arhitectură de sisteme deschise.

Observație Cu toate că arhitectura client-server poate fi utilizată pentru a oferi sisteme SGBD distribuite, totusi ea însăşi nu constituie un sistem SGBD distribuit. Sistemele SGBD distribuite vor fi ulterior analizate în detaliu. Atunci se va examina o extensie a arhitecturii client-server pe *"două etaje"* care scindează funcționalitatea clientului *"lărgit"* în două. În arhitectura client-server pe *"trei etaje",* clientul *"restrâns"* manevrează numai interfața cu utilizatorul, în timp ce stratul de mijloc manevrează logica aplicațiilor. Al treilea strat îl constituie serverul bazei de date. Această arhitectură pe trei etaje s-a dovedit a fi mai convenabilă pentru unele medii, cum ar fi Internet şi rețelele intranet ale companiilor, unde un browser Web poate fi utilizat drept client.

Tabelul 2.1 Sumar al funcțiilor client - server

Client	***Server***
Administrează interfața cu utilizatorul	Primeşte şi procesează cerințele clienților pentru baza de date
Acceptă şi verifică sintaxa intrărilor utilizatorilor	Verifică autorizarea
Procesează aplicațiile	Asigură respectarea contrângerilor de integritate
Generează cerințele pentru baza de date şi le transmite serverului	Efectuează procesarea interogare/reactualizare şi transmite clientului răspunsul
Transmite răspunsul înapoi la utilizator	Întreține catalogul de sistem
	Oferă acces simultan la baza de date
	Oferă controlul reconstituirii

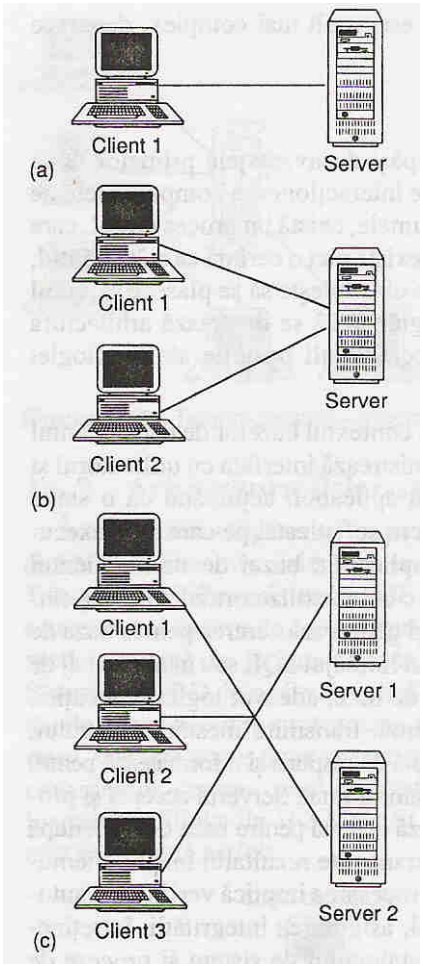


Fig. 2.6 Configurații posibile client – server

a) un singur client şi un singur server

b) mai mulți clienți şi un singur server

c) mai mulți clienți şi mai multe servere

2.7 Catalogul de sistem

În paragraful 2.4 s-a stabilit că un sistem SGBD trebuie să posede un **catalog de sistem** sau un **dicționar de date** accesibil utilizatorilor. În încheierea acestui capitol despre mediul bazelor de date, vom examina mai detaliat cataloagele de sistem.

Catalogul de sistem Este un depozit de informații care descriu datele din baza de date-adică meta-datele sau "datele despre date".

Catalogul de sistem SGBD este una din componentele de bază ale sistemului. Volumul de date conținut şi modul în care sunt utilizate informațiile variază de la sistem la sistem. De regulă, catalogul de sistem stochează:

- denumirile, tipurile şi dimensiunile articolelor de date
- denumirile relațiilor
- constrângerile de integritate asupra datelor
- numele utilizatorilor autorizați care au acces la date
- articolele de date pe care le poate accesa fiecare utilizator şi tipurile de acces

permise
- schemele externe, conceptuale şi interne şi transpunerile dintre ele
- statistica utilizării, cum ar fi *frecvenţa tranzacţiilor* şi contorizarea numărului de accesări ale obiectelor din baza de date

Atenţie: termenul de **dicţionar de date** este adesea utilizat pentru a face referire la un sistem software mai general decât catalogul unui sistem SGBD. Un sistem de dicţionare de date poate să fie activ sau pasiv. Un sistem **activ** este întotdeauna coerent cu structura bazei de date, deoarece este întreţinut automat de sistem. În schimb, un sistem **pasiv** poate să nu fie coerent cu baza de date, deoarece schimbările sunt iniţiate de utilizatori. Dacă dicţionarul de date este o parte a sistemului SGBD, atunci va fi folosită denumirea de dicţionar de date **integrat**. Un dicţionar de date **autonom** are propriul său sistem SGBD specializat. Un dioţionar de date autonom poate fi preferabil în stadiile iniţiale ale proiectării, deoarece acesta întârzie cât mai mult posibil decizia asupra unui anumit sistem SGBD al organizaţiei. Totusi, există dezavantajul că, o dată ce sistemul SGBD a fost selectat si baza de date implementată, este mult mai difiicl să se menţină dicţionarul de date autonom coerent cu baza de date. Această problemă ar putea fi minimizată dacă ar fi posibilă transferarea dicţionarului de date din proiectare în catalogul de sistem SGBD.

Să analizăm pe scurt un standard pentru dicţionarul de date.

2.7.1 Sistemul de informatii al dicţionarului de resurse (IRDS)[1]

În multe sisteme, dicţionarul de date este o componentă internă a sistemului SGBD, care stochează numai infonnaţiile direct legate de baza de date. Totuşi, datele conţinute de sistemul SGBD reprezintă, de obicei, numai o parte a cerinţelor informaţionale totale ale unei organizaţii. De regulă, există informaţii adiţionale conţinute în alte instrumente, cum ar fi CASE, instrumentele de documentare şi instrumentele de configurare şi administrare a proiectului. Fiecare dintre aceste instrumente va avea propriul dicţionar de date intern, care poate fi ascesat de alte instrumente externe. Din păcate, nu a existat nici o modalitate generală de partajare a acestor seturi diferite de informaţii între diversele grupuri de utilizatori sau aplicaţii.

Recent, s-a făcut o tentativă de a standardiza interfaţa la dicţionarele de date, pentru a le face mai accesibile şi cu posibilităţi superioare de partajare. Aceasta a condus la dezvoltarea sistemului de informaţii al dicţionarului de resurse (IRDS). Un sistem IRDS este un instrument software care poate fi utilizat pentru a controla şi documenta o resursă de informaţii a unei organizaţii. El oferă o definiţie pentru tabelele care conţin dicţionarul de date şi operaţiile care pot fi utilizate pentru accesarea acestor tabele. Operaţiile oferă o metodă coerentă de accesare a dictionarului de date şi o modalitate de transferare a definiţiilor datelor de la un dicţionar la altul. De exemplu, informaţiile stocate într-un dicţionar de date de tip IRDS al unui sistem DB2 pot fi transferate la un dicţionar de date de acelaşi tip al unui sistem ORACLE sau pot fi accesate de o aplicaţie DB1, utilizând serviciile IRDS.

Unul dintre punctele forte ale sistemului IRDS îl reprezintă extensibilitatea dicţionarului de date. Astfel, dacă un utilizator al unui sistem SGBD doreşte să stocheze definiţiile corespunzătoare unui nou tip de infonnaţie într-un instrument - de exemplu rapoartele de administrare a proiectelor dintr-un sistem SGBD, - atunci sistemul IRDS al SGBD poate fi extins pentru a include această informaţie. Sistemul IRDS a fost adoptat ca standard de către Organizaţia Internaţională pentru Standardizare (ISO).

Standardele IRDS definesc un set de reguli referitoare la modul în care sunt stocate şi accesate infonnaţiile în dicţionarul. de date. Sistemul IRDS are trei obiective:

- extensibilitatea datelor;

[1] Acronim pentru Information Resource Dictionary System. 2 International Organization for Standardization

- integritatea datelor;
- accesul controlat la date.

Sistemul IRDS se bazează pe o interfață de servicii, alcătuită dintr-un set de funcții care pot fi apelate pentru a accesa dicționarul de date. Interfața de servicii poate fi invocată de următoarele componente ale interfeței cu utilizatorul:

- tablou;
- limbaj de comandă;
- fişiere de export/import;
- programe aplicație.

Interfața cu tabloul constă într-un set de tablouri saa ecrane, fiecare dintre ele oferind acces la un set prescris de servicii. Această interfață poate fi similară limbajului QBE (Query-by-Example) şi permite utilizatorului să răsfoiască şi să modifice datele din dicționar.

Interfața cu limbajul de comandă (CLI[2]) constă într-un set de comenzi sau instrucțiuni, care permit utilizatorului să efectueze operații asupra datelor din dicționar. Interfața CLI poate fi invocată interactiv de la un terminal sau poate fi încorporată într-un limbaj de programare de nivel înalt. Interfața export/import generează un fişier care poate fi mutat între sistemele de tip IRDS. Standardul defineşte un format general pentru interschimbarea de informații. Standardul nu necesită ca baza de date a dicționarului să se conformeze unui anumit model de date, astfel încât interfața cu serviciile IRDS poate să conecteze sisteme SGBD heterogene (vezi fig. Fig. 7).

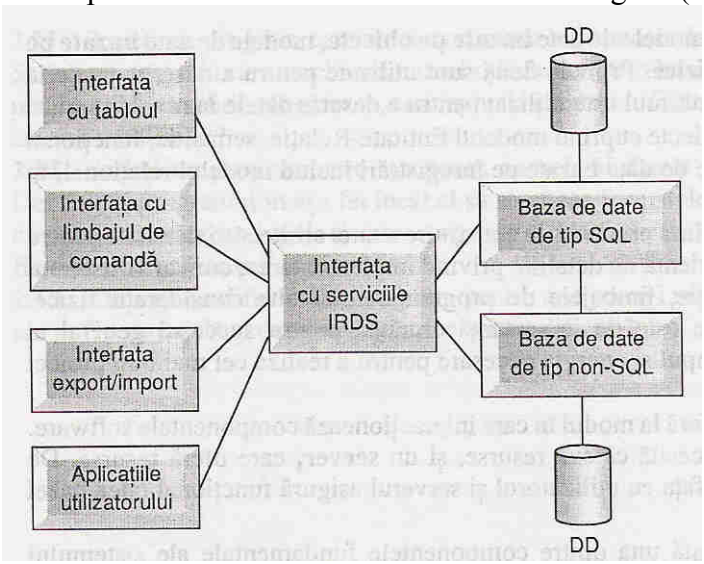


Fig. 7 Interfața cu serviciile IRDS.

[2] Acronim pentru Command Language Interface.