1. GENERALITĂŢI DESPRE BAZE DE DATE

1.1. Introducere

Baza de date este un ansamblu structurat de date coerente, fără redondanţă inutilă, astfel încât acestea pot fi prelucrate eficient de mai mulţi utilizatori într-un mod concurent.

Baza de date este o colecţie de date pe rsistente, care sunt folosite de către sistemele de aplicaţii ale unei anumite „întreprinderi“. Datele din baza de date persistă deoarece, după ce au fost acceptate de către sistemul de gestiune pentru introducerea în baza de date, ele pot fi şterse din bază numai printr-o cerere explicită adresată sistemului de gestiune.

Aici, termenul de „întreprindere“ este un cuvânt generic, utilizat pentru a desemna orice organizaţie independentă, de natură tehnică, comercială, ştiinţifică sau de alt tip. Întreprinderea poate fi, de exemplu, un spital, o bancă, o facultate, o fabrică, un aeroport etc. Fiecare întreprindere are **regulile proprii de funcţionare** şi conţine o **mulţime de date** referitoare la modul său de operare.

Datele din baza de date pot fi atît integrate, cât şi partajate. Noţiunea de **integrat** se referă la faptul că baza de date poate fi considerată ca o unificare a mai multor fişiere, iar prin **partajare** se înţelege că baza de date poate fi partajată concurent între diferiţi utilizatori.

Un **sistem de gestiune a bazelor de date** (SGBD – *Data Base Management System*) este un produs *software* care asigură interacţiunea cu o bază de date, permiţând definirea, consultarea şi actualizarea datelor din baza de date. Toate cererile de acces la baza de date sunt tratate şi controlate de către SGBD.

Organizarea datelor în baze de date constituie o formă de centralizare a acestora. Aceasta implică existenţa unui **administrator al bazei de date** (DBA – *Data Base Administrator*) care este o persoană sau un grup de persoane ce răspund de ansamblul activităţilor (analiză, proiectare, implementare, exploatare, întreţinere etc.) legate de baza de date. Atribuţiile unui administrator pot fi grupate în patru mari categorii: atribuţii de proiectare, atribuţii administrative, atribuţii operative şi atribuţii de coordonare.

Dicţionarul datelor (catalog de sistem), structurat şi administrat ca o bază de date (metabază de date), contine „date despre date“, furnizează descrierea tuturor obiectelor unei baze de date, starea acestor obiecte, diversele constrângeri de securitate şi de integritate etc. Dicţionarul poate fi interogat, la fel, ca orice altă bază de date.

1.2. Gestiunea bazelor de date

Un sistem de baze de date presupune următoarele componente principale, care definesc **arhitectura** acestuia:

- baza de date propriu-zisă în care se memorează datele;
- sistemul de gestiune a bazei de date, care realizează gestionarea şi prelucrarea complexă a datelor;
- un dicţionar al bazei de date (metabaza de date), ce conţine informaţii despre date, structura acestora, statistici, documentaţie;
- mijloace *hardware* (comune sau specializate);
- reglementări administrative destinate bunei funcţionări a sistemului;
- personalul implicat (utilizatori finali, administratorul datelor, administratorul bazei de date, proiectanţi, programatori de aplicaţii).

Se pot identifica patru categorii de persoane implicate în mediul bazelor de date:

- administratorii de date şi baze de date,
- proiectanţii (designeri) de baze de date,
- programatorii de aplicaţii,
- utilizatorii finali.

Administratorul de date (DA) este un manager, nu un tehnician, ce:

- decide care date trebuie stocate în baza de date;
- stabileşte regulile de întreţinere şi de tratare a acestor date după ce sunt stocate. De exemplu, o regulă ar putea fi aceea prin care se stabilesc pentru utilizatori privilegii asupra informaţiilor din baza de date, cu alte cuvinte o anumită politică de securitate a datelor.

Administratorul bazei de date (DBA) este responsabil cu implementarea deciziilor administratorului de date şi cu controlul general al sistemului, la nivel tehnic. El este un profesionist în domeniul IT, care:

- creează baza de date reală;
- implementează elementele tehnice de control;
- este responsabil cu asigurarea funcţionării sistemului la performanţe adecvate, cu monitorizarea performanţelor;
- furnizează diverse servicii tehnice etc.

Proiectanţii de baze de date pot acoperi atât aspectul fizic, cât şi cel logic al concepţiei.

Proiectantul de baze de date care abordează direcţia logică trebuie să posede o cunoaştere completă şi amănunţită a modelului real de proiectat şi a regulilor de funcţionare ale acestuia. Practic, acesta proiectează conceptual baza de date, iar modelul creat este independent de programele de aplicaţii, de limbajele de programare. De asemenea, va proiecta logic baza de date, proiectare care este îndreptată spre un anumit model de date (relaţional, orientat obiect, ierarhic etc.).

Proiectantul de baze de date fizice preia modelul logic de date şi stabileşte cum va fi realizat fizic. Acesta trebuie să cunoască funcţionalităţile SGBD-ului, avantajele şi dezavantajele fiecărei alternative corespunzătoare unei implementări. Practic, se face transpunerea modelului logic într-un set de tabele supuse unor constrângeri, se selectează structuri de stocare şi metode de acces specifice, astfel încât să se asigure performanţe, se iau măsuri privind securitatea datelor.

Utilizatorii finali sunt cei care accesează interactiv baza de date. Aceasta a fost proiectată, implementată, întreţinută pentru a satisface necesităţile informaţionale ale clienţilor. Utilizatorii finali pot fi utilizatori simpli, care nu cunosc nimic despre baza de date, despre SGBD, dar accesează baza prin intermediul unor programe de aplicaţie. În general, această clasă de utilizatori alege anumite opţiuni din meniul aplicaţiei. Există utilizatori finali sofisticaţi, care sunt familiarizaţi cu structura bazei de date. Ei pot utiliza limbaje speciale pentru a exploata posibilităţile oferite de baza de date.

Programatori de aplicaţii sunt responsabili de scrierea programelor aplicaţie ce conferă funcţionalitatea cerută de utilizatorii finali. Programele pot fi scrise în diferite limbaje de programare (*C++*, *PL/SQL*, *Java* etc.).

Cerinţe minimale care se impun unei baze de date:

- asigurarea unei redundanţe minime în date;
- furnizarea în timp util a informaţiilor solicitate (timpul de răspuns la o interogare);
- asigurarea unor costuri minime în prelucrarea şi întreţinerea informaţiei;
- capacitatea de a satisface, cu aceleaşi date, necesităţi informaţionale ale unui număr mare de utilizatori,
- posibilitatea de adaptare la cerinţe noi, răspunsuri la interogări neprevăzute iniţial (flexibilitate);
- exploatarea simultană a datelor de către mai mulţi utilizatori (sincronizare);
- asigurarea securităţii datelor prin mecanisme de protecţie împotriva accesului neautorizat (confidenţialitate);
- înglobarea unor facilităţi destinate validării datelor şi recuperării lor în cazul unor deteriorări accidentale, garantarea (atât cât este posibil) că datele din baza de date sunt corecte (integritate);
- posibilitatea de valorificare a eforturilor anterioare şi anticiparea nevoilor viitoare (compatibilitate şi expandabilitate);
- permisivitatea, prin ierarhizarea datelor după criteriul frecvenţei acceselor, a unor reorganizări (eventual dinamice) care sporesc performanţele bazei.

În cadrul unei baze de date putem vorbi de patru **niveluri de abstractizare şi de percepţie** a datelor: intern, conceptual, logic şi extern. Datele există doar la nivel fizic, iar celelalte trei niveluri reprezintă virtualizări ale acestora.

- Nivelul fizic (intern) este descris de schema fizică a datelor (bit, octet, adresă);
- Nivelul conceptual este descris de schema conceptuală a datelor (articol, înregistrare, zonă) şi reprezintă viziunea programatorilor de sistem asupra datelor;
- Nivelul logic este descris de una din schemele logice posibile ale datelor şi reprezintă viziunea programatorului de aplicaţie asupra datelor;
- Nivelul virtual (extern) reprezintă viziunea utilizatorului final asupra datelor.

Independenţa datelor cuprinde două aspecte fundamentale: o modificare a structurii fizice nu va afecta aplicaţia şi reciproc, modificări ale aplicaţiei vor lăsa nealterată structura fizică de date.

- **Independenţa fizică**: posibilitatea modificării schemei fizice a datelor fără ca aceasta să implice modificarea schemei conceptuale, a schemei logice şi a programelor de aplicaţie. Este vorba despre imunitatea programelor de aplicaţie faţă de modificările modului în care datele sunt stocate fizic şi accesate.
- **Independenţa logică**: posibilitatea modificării schemei conceptuale a datelor fără ca aceasta să implice modificarea schemei logice şi a programelor de aplicaţie. Prin independenţa logică a datelor se urmăreşte a se crea fiecărui utilizator iluzia că este singurul beneficiar al unor date pe care, în realitate, le foloseşte în comun cu alţi utilizatori.

Independenţă faţă de strategiile de acces permite programului să precizeze data pe care doreşte să o acceseze, dar nu modul cum accesează această dată. SGBD-ul va stabili drumul optim de acces la date.

În limbajele de programare uzuale declaraţiile şi instrucţiunile executabile aparţin aceluiaşi limbaj. În lumea bazelor de date, funcţiile de declarare şi de prelucrare a datelor sunt realizate cu ajutorul unor limbaje diferite, numite **limbaje pentru baze de date.**

- Limbaje pentru definirea datelor (LDD – *Data Description Language*). Descrierea concretă a unui LDD este specifică fiecărui sistem de gestiune, dar funcţiile principale sunt aceleaşi. La nivel conceptual, LDD realizează definirea entităţilor şi a atributelor acestora, sunt precizate relaţiile dintre date şi strategiile de acces la ele, sunt stabilite criterii diferenţiate de confidenţialitate şi de validare automată a datelor utilizate.
- Limbaje pentru prelucrarea datelor (LMD – *Data Manipulation Language*). Operaţiile executate în cadrul unei baze de date presupun existenţa unui limbaj specializat, în care comenzile se exprimă prin fraze ce descriu acţiuni asupra bazei. În general, o comandă are următoarea structură: operaţia (calcul aritmetic sau logic, editare, extragere, deschidere-închidere, adăugare, ştergere, căutare, reactualizare etc.), criterii de selecţie, mod de acces (secvenţial, indexat etc.), format de editare. Există limbaje LMD procedurale, care specifică **cum** se obţine rezultatul unei comenzi LMD şi limbaje neprocedurale, care descriu doar datele **ce** vor fi obţinute şi nu modalitatea de obţinere a acestora.

- Limbaje pentru controlul datelor (LCD – *Data Control Language*). Controlul unei baze de date se referă la asigurarea confidenţialităţii şi integrităţii datelor, la salvarea informaţiei în cazul unor defecţiuni, la obţinerea unor performanţe, la rezolvarea unor probleme de concurenţă.

Limbajele universale nu se utilizează frecvent pentru gestionarea unei baze de date, dar există această posibilitate. De exemplu, sistemul Oracle este dotat cu precompilatoare (*C*, *Pascal*, *ADA*, *Cobol*, *PL/1*, *Fortran*) care ajută la incorporarea de instrucţiuni *SQL* sau blocuri *PL/SQL* în programe scrise în alte limbaje, de nivel înalt, numite limbaje gazdă.

Sistemul de gestiune a bazelor de date interacţionează cu programele de aplicaţie ale utilizatorului şi cu baza de date, oferind o mulţime de facilităţi. Realizarea optimă a acestor facilităţi este asigurată de **obiectivele** fundamentale ale unui sistem de gestiune. Câteva dintre aceste obiective vor fi enumerate în continuare.

- Independenţa fizică. Obiectivul esenţial este acela de a permite realizarea independenţei structurilor de stocare în raport cu structurile de date din lumea reală. Se defineşte mulţimea de date indiferent de forma acesteia din lumea reală, ţinând seama doar de a realiza un acces simplu la date şi de a obţine anumite performanţe.
- Independenţa logică. Grupul de lucru care exploatează baza de date poate să utilizeze diferite informaţii de bază (nu aceleaşi), pentru a-şi construi entităţi şi relaţii. Fiecare grup de lucru poate să cunoască doar o parte a semanticii datelor, să vadă doar o submulţime a datelor şi numai sub forma în care le doreşte. Această independenţă asigură imunitatea schemelor externe faţă de modificările făcute în schema conceptuală.
- Prelucrarea datelor de către neinformaticieni. Neinformaticienii văd datele independent de implementarea lor şi pot exploata aceste date prin intermediul unui sistem de meniuri oferit de aplicaţia pe care o exploatează.
- Administrarea centralizată a datelor. Administrarea datelor presupune definirea structurii datelor şi a modului de stocare a acestora. Administrarea este în general centralizată şi permite o organizare coerentă şi eficace a informaţiei.
- Coerenţa datelor. Informaţia trebuie să satisfacă constrângeri statice sau dinamice, locale sau generale.

- Neredundanţa datelor. Fiecare aplicaţie posedă datele sale proprii şi aceasta conduce la numeroase dubluri. De asemenea, organizarea nejudicioasă a relaţiilor poate să genereze redundanţă în date. Administrarea coerentă a datelor trebuie să asigure neduplicarea fizică a datelor. Totuşi, nu sunt excluse nici cazurile în care, pentru a realiza performanţe referitoare la timpul de acces la date şi răspuns la solicitările utilizatorilor, să se accepte o anumită redundanţă a datelor.
- Partajabilitatea datelor. Aceasta permite ca aplicaţiile să partajeze datele din baza de date în timp şi simultan. O aplicaţie poate folosi date ca şi cum ar fi singura care le utilizează, fără a şti că altă aplicaţie, concurent, le poate modifica.
- Securitatea şi confidenţialitatea datelor. Datele trebuie protejate de un acces neautorizat sau rău intenţionat. Există mecanisme care permit identificarea şi autentificarea utilizatorilor şi există proceduri de acces autorizat care depind de date şi de utilizator. Sistemul de gestiune trebuie să asigure securitatea fizică şi logică a informaţiei şi să garanteze că numai utilizatorii autorizaţi pot efectua operaţii corecte asupra bazei de date.

Sistemele de gestiune a bazelor de date au, din nefericire, şi **dezavantaje** dintre care se remarcă:

- complexitatea şi dimensiunea sistemelor pot să crească considerabil, datorită necesităţii extinderii funcţionalităţilor sistemului;
- costul, care variază în funcţie de mediu şi funcţionalitatea oferită, la care se adugă cheltuieli periodice de întreţinere;
- costuri adiţionale pentru elemente de *hardware*;
- costul conversiei aplicaţiilor existente, necesară pentru ca acestea să poată funcţiona în noua configuraţie *hardware* şi *software*;
- impactul unei defecţiuni asupra aplicaţiilor, bazei de date sau sistemului de gestiune.

Structura unui sistem de gestiune a bazelor de date este de complexitate variabilă, iar nivelul real de funcţionalitate diferă de la produs la produs. În orice moment apar noi necesităţi, care cer o nouă funcţionalitate, astfel încât aceasta nu va putea deveni niciodată statică. În general, un SGBD trebuie să includă cel puţin cinci clase de module:

- programe de gestiune a bazei de date (PGBD), care realizează accesul fizic la date ca urmare a unei comenzi;

- module pentru tratarea limbajului de definire a datelor, ce permit traducerea unor informaţii (care realizează descrierea datelor, a legăturilor logice dintre acestea şi a constrângerilor la care sunt supuse), în obiecte ce pot fi apoi exploatate în manieră procedurală sau neprocedurală;
- module pentru tratarea limbajului de prelucrare a datelor (interpretativ, compilativ, generare de programe), care permit utilizatorilor inserarea, ştergerea, reactualizarea sau consultarea informaţiei dintr-o bază de date;
- module utilitare, care asigură întreţinerea, prelucrarea, exploatarea corectă şi uşoară a bazei de date;
- module de control, care permit controlul programelor de aplicaţie, asigurarea confidenţialităţii şi integrităţii datelor, rezolvarea unor probleme de concurenţă, recuperarea informaţiei în cazul unor avarii sau defecţiuni *hardware* sau *software* etc.

Modulele PGBD asigură accesul fizic la date ca urmare a unei comenzi. Cum lucrează aceste module?

- găsesc descrierea datelor implicate în comandă;
- identifică datele şi tipul acestora;
- identifică informaţii ce permit accesul la structurile fizice de stocare (fişiere, volume etc.);
- verifică dacă datele sunt disponibile;
- extrag datele, fac conversiile, plasează datele în spaţiul de memorie al utilizatorului;
- transmit informaţii de control necesare execuţiei comenzii, în spaţiul de memorie al utilizatorului;
- transferă controlul programului de aplicaţie.

Prin urmare, din punct de vedere conceptual:

- utilizatorul lansează o cerere de acces;
- SGBD-ul acceptă cererea şi o analizează;
- SGBD-ul inspectează pe rând, schema internă corespunzatoare utilizatorului, schema conceptuală, definiţia structurii de stocare şi corespondenţele corespunzătoare;
- SGBD-ul execută operaţiile necesare în baza de date stocată.

1.3. Arhitectura sistemelor de gestiune a bazelor de date

Asigurarea independenţei fizice şi logice a datelor impune adoptarea unei arhitecturi de baze de date organizată pe trei niveluri:

- **nivelul intern** (baza de date fizică);
- **nivelul conceptual** (modelul conceptual, schema conceptuală);
- **nivelul extern** (modelul extern, subschema, vizualizarea).

Nivelul central este **nivelul conceptual**. Acesta corespunde structurii canonice a datelor ce caracterizează procesul de modelat, adică structura semantică a datelor fără implementarea pe calculator. Schema conceptuală permite definirea tipurilor de date ce caracterizează proprietăţile elementare ale entităţilor, definirea tipurilor de date compuse care permit regruparea atributelor pentru a descrie entităţile modelului şi legăturile între aceste entităţi, definirea regulilor pe care trebuie să le respecte datele etc.

Nivelul intern corespunde structurii interne de stocare a datelor. Schema internă permite descrierea datelor unei baze sub forma în care sunt stocate în memoria calculatorului. Sunt definite fişierele care conţin aceste date, articolele din fişiere, drumurile de acces la aceste articole etc.

La nivel conceptual sau intern, schemele descriu o bază de date. La **nivel extern** schemele descriu doar o parte din date care prezintă interes pentru un utilizator sau un grup de utilizatori. Schema externă reprezintă o descriere a unei părţi a bazei de date ce corespunde viziunii unui program sau unui utilizator. Modelul extern folosit este dependent de limbajul utilizat pentru prelucrarea bazei de date. Schema externă permite asigurarea unei securităţi a datelor. Un grup de lucru va accesa doar datele descrise în schema sa externă, iar restul datelor sunt protejate împotriva accesului neautorizat sau rău intenţionat.

Pentru o bază de date particulară există o singură schemă internă, o singură schemă conceptuală, dar există mai multe scheme externe.

În afară de aceste trei niveluri, arhitectura presupune şi anumite corespondenţe dintre acestea:

- corespondenţa conceptual-intern defineşte relaţia dintre nivelul conceptual şi baza de date stocată, specificând modul în care înregistrările şi câmpurile conceptuale sunt reprezentate la nivel intern;
- corespondenţa extern-conceptual defineşte relaţia dintre o anumită vizualizare externă şi nivelul (vizualizarea) conceptual, reprezentând cheia independenţei logice de date;

- corespondenţa extern-extern permite definirea unor vizualizări externe în funcţie de altele, fără a necesita o definiţie explicită a corespondenţei cu nivelul conceptual.

Arhitectura funcţională de referinţă propusă de grupul de lucru ANSI/X3/SPARC este axată pe dicţionarul datelor şi cuprinde două părţi:

- prima, permite descrierea datelor (compoziţia dicţionarului datelor);
- a doua, permite prelucrarea datelor (interogarea şi reactualizarea bazei de date).

În fiecare parte se regăsesc cele trei niveluri: intern, conceptual şi extern. Acestea nu sunt neapărat distincte pentru orice SGBD.

Interfeţele numerotate din figura 1.1, ce descriu arhitectura de referinţă a unui SGBD, corespund următoarelor transformări:

a) Limbaj de descriere a datelor conceptuale, format sursă – permite administratorului întreprinderii să definească schema conceptuală, format sursă.

b) Limbaj de descriere a datelor conceptuale, format obiect – se obţine din compilarea celui precedent şi permite aranjarea schemei obiect în dicţionarul datelor.

c) Limbaj de descriere a datelor conceptuale, format editare – permite administratorilor aplicaţiilor şi a bazelor să consulte schema conceptuală pentru a defini reguli de corespondenţă.

d) Limbaje de descriere a datelor externe, format sursă – permit administratorilor aplicaţiilor să definească scheme externe corespunzând schemei conceptuale. Deoarece sistemele de gestiune pot suporta mai multe modele externe, pot exista mai multe limbaje de descriere a datelor externe.

e) Limbaje de descriere a datelor externe, format obiect – corespund formelor compilate ale celor precedente şi permit aranjarea schemelor externe (obiect) în dicţionarul datelor.

f) Limbaj de descriere a datelor interne, format sursă – permite administratorului bazei de date să definească schema internă şi regulile de corespondenţă cu schema conceptuală.

g) Limbaj de descriere a datelor interne, format obiect – corespunde formei compilate a celui precedent şi permite aranjarea schemei interne (obiect) în dicţionarul datelor.

h) Limbaje de prelucrare a datelor externe, format sursă – permit programatorilor de aplicaţii sau utilizatorilor neinformaticieni să manipuleze date externe (*view*).

i) Limbaje de prelucrare a datelor externe, format obiect – corespund formelor compilate ale celor precedente.

j) Limbaj de prelucrare a datelor conceptuale, format obiect – produs de procesorul de transformare extern/ conceptual pentru a manipula datele externe.

k) Limbaj de prelucrare a datelor interne, format obiect –produs de procesorul de transformare conceptual/intern pentru a gestiona datele interne.

l) Limbaj de stocare a datelor, format obiect – corespunde interfeţei cu sistemul de stocare a datelor.

m) Interfaţa cu memoria secundară – permite efectuarea de intrări/ieşiri în/din unitatea de memorie secundară.

n) Interfaţa de acces la dicţionarul datelor – permite diverselor procesoare de transformare să acceseze scheme obiect şi reguli de corespondenţă.

Procesoarele din figura 1.1 au următoarele funcţii:

- procesorul schemei conceptuale compilează schema conceptuală şi dacă nu sunt erori depune schema compilată în dicţionarul datelor;
- procesorul schemei externe compilează schemele externe şi regulile de corespondenţă externă şi dacă nu sunt erori aranjează schema compilată şi regulile de corespondenţă în dicţionarul datelor;
- procesorul schemei interne are rol similar pentru schema internă;
- procesorul de transformare extern/conceptual transformă manipulările externe în manipulări conceptuale şi invers, datele conceptuale în date externe;
- procesorul de transformare conceptual/intern transformă manipulările conceptuale în manipulări interne şi invers, datele interne în date conceptuale;
- procesorul de transformare intern/stocare transformă manipulările interne în primitive ale sistemului de stocare şi invers, eliberează datele stocate într-un format corespunzător schemei interne.

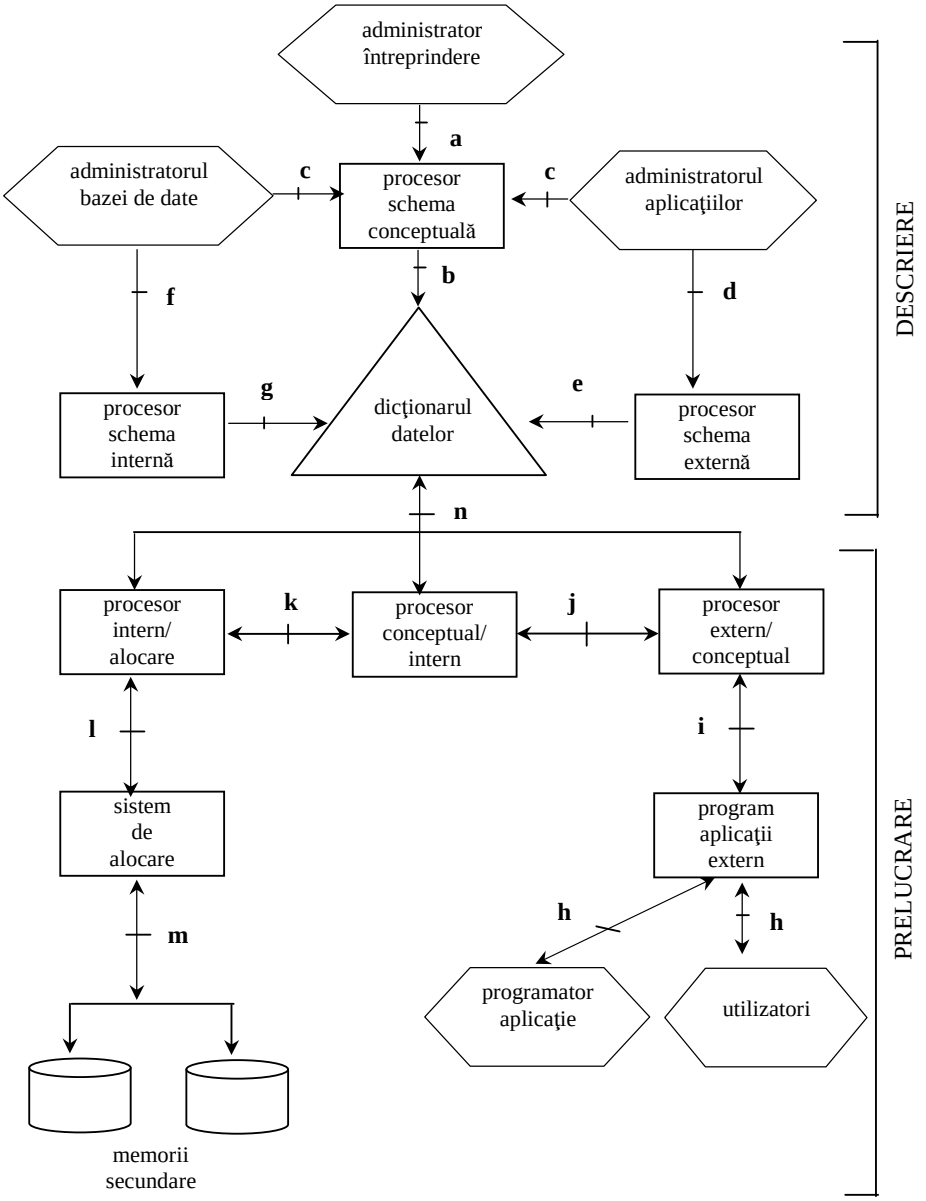

Fig. 1.1. Arhitectura de referinţă a unui SGBD.

Gardarin a propus o arhitectură funcţională apropiată de arhitectura sistemelor de gestiune actuale care are la bază doar două niveluri:

- schema, care corespunde integrării schemelor interne şi conceptuale;
- vizualizarea, care este o schemă externă.

Sistemul de gestiune gestionează un **dicţionar de date** care este alimentat prin comenzi de definire a schemei şi prin comenzi de definire a vizualizărilor.

Aceste comenzi, precum şi cererile de prelucrare sunt analizate şi tratate de un procesor numit **analizor**. Analizorul realizează analiza sintactică şi semantică a cererii şi o traduce în format intern. O cerere în format intern care face referinţă la o vizualizare este tradusă în una sau mai multe cereri care fac referinţă la obiecte ce există în baza de date (modificarea cererilor).

În cadrul acestei arhitecturi există un procesor, numit **translator**, care realizează modificarea cererilor, asigură controlul drepturilor de acces şi controlul integrităţii în cazul reactualizărilor.

Componenta cheie a sistemului de gestiune este procesorul **optimizor** care elaborează un plan de acces optim pentru a trata cererea. Acest procesor descompune cererea în operaţii de acces elementare şi alege o ordine de execuţie optimală. De asemenea, evaluează costul planului de acces înaintea execuţiei sale.

Planul de acces ales şi elaborat de optimizor este executat de un procesor numit **executor**. La acest nivel este gestionat controlul concurenţei.

1.4. Evoluţia bazelor de date

Istoria bazelor de date şi a sistemelor de gestiune a bazelor de date poate fi rezumată în trei generaţii:

- sisteme ierarhice şi reţea,
- sisteme relaţionale,
- sisteme avansate (orientate obiect, relaţionale orientate obiect, deductive, distribuite, multimedia, multibaze, active, temporale, decizionale, magazii de date etc.).

Baze de date ierarhice şi reţea

Pentru modelele ierarhice şi reţea, datele sunt reprezentate la nivel de articol prin legături ierarhice (arbore) sau de tip graf. Slaba independenţă fizică a datelor complică administrarea şi prelucrarea acestora. Limbajul de prelucrare a datelor impune programatorului să specifice drumurile de acces la date.

Baze de date relaţionale

A doua generaţie de SGBD-uri este legată de apariţia modelelor relaţionale (1970), care tratează entităţile ca nişte relaţii. Piaţa actuală de baze de date este acoperită în majoritate de sisteme **relaţionale**. Bazele de date relaţionale sunt caracterizate de:

- structuri de date simple, intuitive,
- inexistenţa pointerilor vizibili pentru utilizator,
- constrângeri de integritate,
- operatori aplicaţi relaţiilor care permit definirea, căutarea şi reactualizarea datelor.

Dezvoltarea unei aplicaţii riguroase utilizand o bază de date relaţionale necesită cunoaşterea a trei niveluri de instrumente, eterogene din punct de vedere conceptual:

- nivelul instrumentelor grafice (interfaţa);
- nivelul aplicaţie, cu limbajele sale de dezvoltare;
- nivelul SGBD, cu standardul *SQL* (*Structured Query Language*) ce permite definirea, prelucrarea şi controlul bazei de date.

Baze de date orientate obiect

Bazele de date relaţionale nu folosesc însă obiecte complexe şi dinamice, nu realizează gestiunea datelor distribuite şi nici gestiunea cunoştinţelor. A treia generaţie de SGBD-uri, sistemele avansate, încearcă să depăşească aceste limite ale sistemului relaţional.

Suportul obiectelor complexe şi dinamice şi prelucrarea acestora este dificilă pentru sistemele relaţionale, deoarece tipul datelor este limitat la câteva domenii alfanumerice, iar structura datelor este simplă. Sistemele relaţionale nu modelează obiecte complexe ca grafuri, liste etc. Un obiect complex poate să fie descompus în relaţii, dar apar dificultăţi atât la descompunerea, cât şi la refacerea acestuia prin compunere. De asemenea, limbajele modelului relaţional permit prelucrarea cu dificultate a obiectelor complexe.

Un sistem relaţional nu suportă obiecte dinamice care incorporează atât partea de date (informaţii) efective, cât şi o parte relativă la tratarea acestora.

Îmbinarea tehnicii limbajelor orientate obiect cu a bazelor de date a permis realizarea bazelor de date orientate obiect. Acestea permit organizarea coerentă a obiectelor partajate între utilizatori concurenţi. Sistemele de gestiune de baze de date orientate obiect (SGBDOO) prezintă o serie de avantaje:

- realizează o modelare superioară a informaţiei,
- furnizează posibilităţi superioare de deducţie (ierarhie de clase, moştenire),
- permit luarea în considerare a aspectelor dinamice şi integrarea descrierii structurale şi comportamentale,
- asigură îmbunătăţirea interfeţei cu utilizatorul.

Cu toate avantajele incontestabile oferite de SGBDOO-uri, impunerea lor pe piaţa bazelor de date nu a fost uşoară. Câteva motivaţii a acestei situaţii:

- absenţa unei fundamentări teoretice face imposibilă definirea unui SGBDOO de referinţă;
- gestiunea obiectelor complexe este mai dificilă decât accesul la relaţii prin cereri *SQL*;
- utilizatorii au investit sume uriaşe în sistemele relaţionale şi nu le pot abandona cu uşurinţă. Trecerea la noua tehnologie orientată obiect implică investiţii mari şi nu păstrează aproape nimic din vechile soluţii.

Baze de date relaţionale orientate obiect

Simplitatea modelului relaţional, combinată cu puterea tehnologiei orientate obiect a generat un domeniu nou şi promiţător în lumea bazelor de date, şi anume bazele de date relaţionale orientate obiect.

Construcţia unui sistem de gestiune de baze de date relaţionale orientate obiect (SGBDROO) trebuie să pornească de la cele existente. Aceasta se poate realiza în două moduri: dezvoltând un sistem relaţional prin adăugarea caracteristicilor obiectuale necesare sau pornind de la un sistem orientat obiect şi adăugând caracteristicile relaţionale.

Baze de date deductive

O relaţie este o mulţime de înregistrări ce reprezintă fapte. Cunoştinţele sunt aserţiuni generale şi abstracte asupra faptelor. Cunoştinţele permit să raţionezi, ceea ce permite deducerea de noi fapte, plecând de la fapte cunoscute. Un SGBD relaţional suportă o formă limitată de cunoştinţe, şi anume constrângerile de integritate, iar restul trebuie integrate în programele de aplicaţie. Aceasta generează probleme deoarece cunoştinţele trebuie codificate în programe şi apare imposibilitatea de a partaja cunoştinţe între utilizatori. Totul se complică dacă există un volum mare de fapte.

Bazele de date deductive, utilizând programarea logică, gestionează cunoştinţe relativ la baze de date care, în general, sunt relaţionale. Bazele de date deductive permit deducerea de noi informaţii, plecând de la informaţiile stocate în baza de date. Un SGBD deductiv posedă:

- un limbaj de definire a datelor care permite definirea structurii predicatelor sub formă de relaţii şi constrângeri de integritate asociate;
- un limbaj de prelucrare a datelor care permite efectuarea reactualizărilor asupra datelor şi formularea unor cereri;
- un limbaj de reguli de deducţie care permite ca, plecând cu predicatele definite anterior, să se specifice cum pot fi construite predicate derivate.

Baze de date distribuite

Un sistem distribuit este un ansamblu de maşini ce sunt interconectate printr-o reţea de comunicaţie şi utilizate într-un scop global. Administrarea şi prelucrarea datelor distribuite, situate pe diferite calculatoare şi exploatate de sisteme eterogene este obiectivul fundamental al bazelor de date distribuite.

Bazele de date distribuite sunt sisteme de baze de date cooperante care rezidă pe maşini diferite, în locuri diferite. Această mulţime de baze de date este exploatată de utilizator ca şi cum ar fi o singură bază de date. Programul de aplicaţie care exploatează o bază de date distribuite poate avea acces la date rezidente pe mai multe maşini, fără ca programatorul să cunoască localizarea datelor.

Modelul relaţional a rămas instrumentul principal prin care se realizează prelucrarea datelor distribuite.

Câteva dintre argumentele pentru a justifica această afirmaţie sunt:

- bazele relaţionale oferă flexibilitate de descompunere în vederea distribuirii;
- operatorii relaţionali pot fi folosiţi pentru combinaţii dinamice ale informaţiilor descentralizate;
- limbajele sistemelor relaţionale sunt concise şi asigură o economie considerabilă a transmiterii datelor. Ele fac posibil, pentru un nod oarecare al reţelei, să analizeze intenţia unei tranzacţii, să o descompună rapid în componente ce pot fi realizate local şi componente ce pot fi transportate altor noduri.

Calculatoare şi maşini baze de date

Soluţia pentru a descentraliza prelucrarea datelor, în scopul evitării saturării memoriei şi a procesoarelor calculatorului central, a fost apariţia calculatoarelor baze de date şi a maşinilor baze de date. Descentralizarea presupune transferarea unei părţi din funcţiile unui SGBD către un calculator periferic (calculator *backend*) adică deplasarea algoritmilor de căutare şi a celor de actualizare a datelor mai aproape de memoria secundară. Acest calculator periferic permite utilizarea optimă a resurselor şi realizarea paralelismului în tratarea cererilor de informaţii.

Calculatorul periferic poate fi un calculator clasic, dar cu un *software* specific de SGBD (calculator bază de date) sau poate fi o maşină cu *hardware* specializat în gestiunea bazelor de date (maşină bază de date). Maşinile baze de date sunt înzestrate cu arhitecturi paralele special adaptate pentru gestionarea unui volum mare de date. Tratarea paralelă a cererilor permite reducerea timpului total de execuţie a acestora.

O execuţie în paralel solicită, fie descompunerea unui program în module, care pot fi executate în paralel (descompunere funcţională), fie descompunerea datelor în subgrupe, astfel încât toate procesoarele să execute acelaşi lucru, dar pe date diferite. Performanţele tratării paralele depind de modul în care sunt efectuate descompunerile.

Multibaze de date

Diferite departamente ale unei organizaţii mai mari pot folosi diferite sisteme de gestiune. Cu toate că fiecare sistem este dezvoltat pentru a satisface nevoile propriului său departament, informaţiile cu care lucrează pot fi utile şi altor departamente. De aceea, pentru ca organizaţia să funcţioneze bine, trebuie să existe o modalitate globală da a vedea datele din fiecare sistem. Există două caracteristici ale unor astfel de sisteme care fac acceasarea datelor în acest mediu integrat greoaie, uneori chiar imposibilă:

- autonomie – fiecare SGBD are o autonomie completă, ceea ce înseamnă că fiecare *manager* are control deplin asupra sistemului;
- eterogenitate – sistemele pot opera pe diferite platforme, cu diferite modele de date şi limbajele de interogare.

Una dintre soluţiile folosite pentru a depăşi dificultăţile întâmpinate în respectarea autonomiei şi a eterogenităţii este utilizarea sistemelor multibaze de date.

Un sistem multibaze de date (SMB) este alcătuit din mai multe sisteme de baze de date privite integrat, în care se construiesc una sau mai multe scheme globale pe baza schemelor fiecărei baze de date componente, astfel încât să se poată realiza accesul uniform şi integrat la fiecare din bazele de date componente. Fiecare schemă globală este construită pe baza unui model particular de date. De exemplu, se poate construi o schemă globală ce are la bază modelul relaţional pentru utilizatorii care sunt familiarizaţi cu acest model, dar se poate construi o schemă globală bazată pe modelul orientat obiect pentru utilizatorii bazelor de date orientate obiect.

Pentru o schemă globală dată, un sistem multibaze de date constă din sistemele componente împreună cu un sistem *front-end*, care suportă un singur model de date şi un singur limbaj de interogare. Principalele sarcini ale sistemului *front-end* sunt gestionarea schemei globale şi procesarea cererilor globale.

Un avantaj major al acestui model, faţă de altele, este faptul că o singură interogare poate accesa date din mai multe baze de date într-un mod integrat, fără să afecteze nici o aplicaţie care este scrisă utilizând una dintre bazele de date componente.

Baze de date cu suport decizional

Sistemele informatice, în particular bazele de date, au ajuns la maturitate. Marile companii au acumulat o mare cantitate de informaţii din domeniul lor de activitate, pe care le păstrează în tabele istorice şi sunt nefolositoare sistemelor operaţionale ale companiei, care funcţionează cu date curente. Analizate, aceste date ar putea oferi informaţii despre tendinţe şi evoluţii care ar putea interesa compania. Pentru a putea analiza aceste mari cantităţi de date este nevoie de tehnologii şi instrumente speciale.

Ideea de a analiza colecţii de date provenind din sistemele operaţionale ale companiei sau din surse externe pentru a le folosi ca suport în procesul de decizie nu aparţine ultimului deceniu, dar baze de date care să funcţioneze eficient după aceste criterii au fost studiate şi implementate în ultimii ani. Principalul scop al acestor baze de date a fost de a întâmpina nevoile sistemelor operaţionale, a căror natură este inerent tranzacţională.

Sistemele tranzacţionale sunt interesate, în primul rând, să controleze la un moment dat o singură tranzacţie. De exemplu, într-un sistem bancar, atunci când clientul face un depozit, sistemul operaţional bancar este responsabil de a înregistra tranzacţia într-un tabel al tranzacţiilor şi de a creşte nivelul curent al contului clientului, stocat în alt tabel.

Un sistem operaţional tipic operează cu evenimente predefinite şi, datorită naturii lor, necesită acces rapid la date. Uzual, fiecare tranzacţie operează cu cantităţi mici de date.

De-a lungul timpului, nevoile sistemelor operaţionale nu se schimbă mult. Aplicaţia care înregistrează tranzacţia, ca şi cea care controlează accesul utilizatorului la informaţie (partea de raportare a sistemului bancar), nu se modifică prea mult. În acest tip de sistem, informaţia necesară în momentul în care un client iniţiază o tranzacţie trebuie sa fie actuală. Înainte ca o bancă să aprobe un împrumut este nevoie să se asigure de situaţia financiară stabilă a clientului în acel moment, şi nu cu un an înainte.

În ultimii ani s-au pus la punct principii şi tehnologii noi care să servească procesului de analiză şi administrare a datelor. O bază de date optimizată în acest scop defineşte o ***Data Warehouse*** (magazie de date), iar principiul pe care îl urmează este cunoscut sub numele de procesare analitică (OLAP – *On Line Analytical Processing*). Spre deosebire de acesta, principiul pe care se bazează sistemele tranzacţionale a fost numit procesare tranzacţională (OLTP – *On Line Transactional Processing*).

Aplicaţiile unei *Data Warehouse* trebuie să ofere răspunsuri unor întrebări de tipul: „Care zi din săptămână este cea mai aglomerată?“ „Ce clienţi, cu care avem relaţii intense, nu au beneficiat de reduceri de preţuri?“. O caracteristică a bazelor de date analitice este că interogările la care acestea trebuie să răspundă sunt *ad-hoc*, nu sunt predefinite, iar baza de date trebuie optimizată astfel încât să fie capabilă să răspundă la orice fel de întrebare care poate implica mai multe tabele.

În această abordare, organele generale de decizie necesită accesul la toate datele organizaţiei, oriunde s-ar afla acestea. Pentru o analiză corespunzătoare a organizaţiei, afacerilor, cerinţelor, tendinţelor este necesară nu numai accesarea valorilor curente din baza de date, ci şi a datelor istorice. Prin urmare, pentru a facilita acest tip specific de analiză a informaţiei a fost creată magazia de date, care conţine **informaţii extrase din diverse surse, întreţinute de diverse unităţi operative, împreună cu istoricul şi rezumatul tranzacţiilor**.

Sursele de date pentru o magazie cuprind:

- date operaţionale, păstrate în baze de date ierarhice, de prima generaţie;
- date departamentale, păstrate în sisteme de fişiere patentate;

- date cu caracter personal, păstrate pe staţii de lucru şi servere personale;
- sisteme externe (baze de date comerciale, Internet etc.)

Data warehouse este o colecţie de date:

- orientate spre subiect (principalele subiecte ale modelului sunt clienţii, produsele, vânzările, în loc de domeniile de activitate),
- nevolatile (datele nu sunt reactualizate, înlocuite în timp real, ci sunt adăugate ca un supliment al bazei),
- integrate (transpunerea datelor provenite din diverse surse de informaţii se face într-un format consistent),
- variabile în timp (concentrarea depozitului de date se face asupra schimbărilor care au loc în timp).

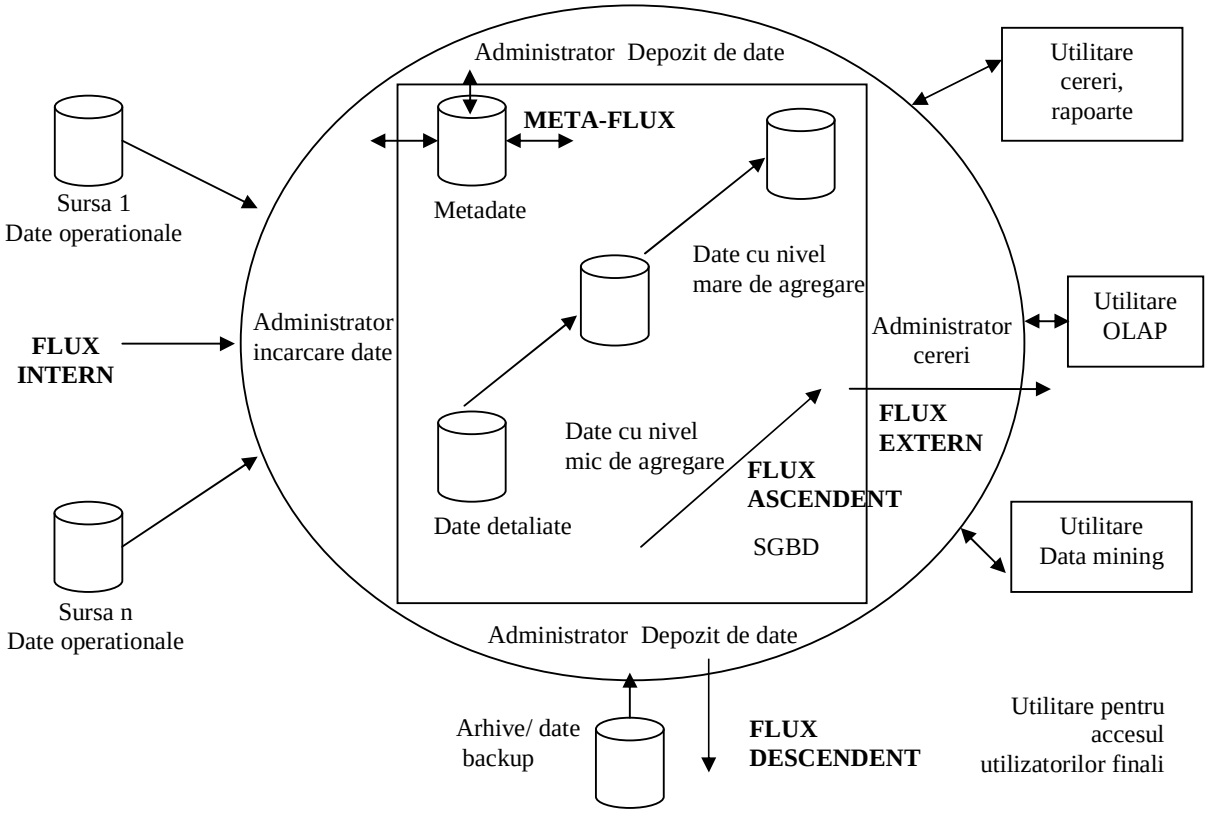

Fig 1.2. Arhitectura unui depozit de date

Înmagazinarea datelor se concentrează asupra gestionării a cinci fluxuri de informaţii:

- fluxul intern, care reprezintă procesele asociate extragerii şi încărcării datelor din fişierele sursă în magazia de date;
- fluxul ascendent, care reprezintă procesele asociate adăugării de valoare datelor din magazie, cu ajutorul împachetării şi distribuirii;
- fluxul descendent, care reprezintă procesele asociate arhivării, salvării, refacerii datelor din magazie;
- fluxul extern, care reprezintă procesele asociate punerii la dispoziţie a datelor pentru utilizatorii finali;
- meta-fluxul, care reprezintă procesele asociate gestionării meta-datelor (date despre date).

În arhitectura depozitului de date intervin câteva componente specifice acestei structuri.

- Administratorul pentru încărcarea datelor (componenta *front-end*) realizează toate operaţiile asociate cu obţinerea (extragerea) şi încărcarea datelor operaţionale într-un depozit de date.
- Administratorul depozitului de date realizează toate operaţiile legate de administrarea datelor din depozit. Operaţiile realizate de componenta de administrare a depozitului de date includ: analiza datelor pentru a asigura consistenţa acestora; transformarea şi mutarea datelor sursă din structurile temporare de stocare în tabelele depozitului de date; crearea de indecşi şi vizualizări asupra tabelelor de bază; generarea denormalizării (dacă este necesar); generarea agregărilor; crearea arhivelor şi a *backup*-urilor.
- Administratorul cererilor (componenta *back-end*) realizează toate operaţiile legate de administrarea cererilor utilizator. Această componentă este construită folosind utilitarele de acces la date disponibile utilizatorilor finali, utilitarele de monitorizare a depozitului de date, facilităţile oferite de sistemul de baze de date şi programele personalizate.
- În zona ce include date agregate sunt stocate toate agregările predefinite de date, pe diferite niveluri. Scopul, menţinerii acestora, este de a mări performanţa cererilor care necesită agregări. Datele agregate sunt actualizate permanent, pe măsură ce sunt încărcate noi informaţii în depozit.

- Scopul principal al depozitelor de date este de a oferi informaţii necesare utilizatorilor pentru luarea deciziilor strategice de marketing. Aceşti utilizatori interacţionează cu depozitul de date prin diferite utilitare de acces (utilitare pentru rapoarte şi cereri, utilitare pentru dezvoltarea aplicaţiilor, utilitare pentru procesarea analitică *on-line* (OLAP), utilitare *data mining*) etc.

Instrumentele de acces pentru utilizatorii finali ai magaziilor de date:

- prelucrarea analitică *on-line*;
- extensiile limbajului *SQL*;
- instrumentele de extragere a datelor.

Prelucrarea analitică *on-line* (OLAP) reprezintă sinteza, analiza şi consolidarea dinamică a unor volume mari de date multidimensionale. Serverele de baze de date OLAP utilizează structuri multidimensionale pentru stocarea datelor şi a relaţiilor dintre date. Aceste structuri pot fi vizualizate prin cuburi de date şi cuburi în cadrul cuburilor etc. Fiecare latură a cubului reprezintă o dimensiune. Serverele de baze de date OLAP multidimensionale acceptă operaţiile analitice uzuale: consolidarea (gruparea), parcurgerea în jos (inversul consolidării), tranşarea, tăierea. OLAP necesită o modalitate de agregare a datelor conform mai multor grupări diferite, în număr foarte mare, iar utilizatorii trebuie să le aibă în vedere pe toate.

Instrumentele OLAP presupun organizarea informaţiei într-un model multidimensional care este susţinut de o bază de date:

- multidimensională (MOLAP), în care datele sunt stocate conceptual în celulele unui tablou multidimensional;
- relaţională (ROLAP), proiectată pentru a permite interogări multidimensionale.

În acest context, a devenit o necesitate extinderea limbajului *SQL* prin operaţii puternice, necesare pentru rezolvarea noului tip de abordare. Au fost introduse noi funcţii numerice (limita inferioară, limita superioară etc.), noi funcţii statistice (distribuţie, distribuţie inversă, corelaţie etc.), noi operatori de agregare, extensii ale clauzei *GROUP BY* etc.

De exemplu, *RISQL* (*Red Brick Intelligent SQL*), proiectat pentru analiştii din domeniul afacerilor, permite: ordonare după rang (pe diferite niveluri - de exemplu, gruparea filialelor în trei categorii pe baza venitului generat în anul precedent), partajarea pieţei, compararea anului curent cu cel precedent etc.

O altă problemă esenţială este extragerea datelor şi utilizarea acestora pentru luarea de decizii cruciale în domeniul afacerilor. Descoperirea unor noi corelaţii, tipare, tendinţe, prin extragerea unor cantităţi mari de date folosind strategia inteligenţei artificiale este una din modalităţile de rezolvare.

Extragerea datelor presupune capacitatea de a construi modele mai degrabă previzibile, decât retrospective. Modelarea predictivă utilizează informaţii pentru a forma un model al caracteristicilor importante ale unui fenomen.

Tehnicile asociate operaţiilor fundamentale de extragere sunt:

- modelarea predictivă (clasificarea cu ajutorul unei reţele neurale sau al unei inducţii de tip arbore şi previziunea valorilor, utilizând tehnici de regresie);
- segmentarea bazei de date (comasarea demografică şi comasarea neurală care se deosebesc prin metodele uilizate pentru a calcula distanţa dintre înregistrări, prin intrările de date permise);
- analiza legăturilor (descoperirea asocierilor, descoperirea tiparelor, descoperirea secvenţelor de timp similare);
- detectarea deviaţiilor (statistici şi vizualizări pentru identificarea împrăştierii datelor, utilizând tehnici moderne de vizualizare grafică). În această clasă pot fi considerate, de exemplu, detectarea fraudelor privind utilizarea cărţilor de credit, pretenţiile de despăgubire ale asiguraţilor etc.

În concluzie, spre deosebire de un sistem OLTP, *Data Warehouse* este o bază de date a cărei structură este proiectată pentru a facilita analiza datelor. Un sistem de suport decizional urmăreşte, în primul rând, obţinerea de informaţii din baza de date, în timp ce unul OLTP urmăreşte introducerea de informaţii în baza de date. Datorită acestor diferenţe, structura optimă a unei *Data Warehouse* este radical diferită de cea a unui sistem OLTP.

Depozitele de date şi sistemele OLTP sunt supuse unor cerinţe diferite, dintre care cele mai semnificative se referă la operaţii, actualizarea datelor, proiectare, operaţii tipice şi date istorice.

- **Operaţii**. Depozitele sunt create pentru a permite interogări *ad hoc*. Ele trebuie să fie suficient de flexibile pentru a putea răspunde interogărilor spontane ale utilizatorilor. Sistemele OLTP suportă numai operaţii predefinite. Aplicaţiile pot fi optimizate sau create special numai pentru acele operaţii.

- **Actualizarea datelor**. Utilizatorii finali ai unui depozit de date nu fac în mod direct actualizări ale depozitului. În sistemele OLTP, utilizatorii realizează, de obicei, în mod individual procedurile de modificare a bazei de date. În acest fel, baza de date OLTP este întotdeauna la zi şi reflectă starea curentă a fiecărei tranzacţii.
- **Proiectare**. Depozitele de date folosesc, în mod uzual, scheme denormalizate, în timp ce sistemele OLTP folosesc scheme normalizate pentru a optimiza performanţele operaţiilor.
- **Operaţii tipice**. O simplă interogare a depozitului de date poate scana mii sau chiar milioane de înregistrări (de exemplu, cererea „Care sunt vânzările totale către toţi clienţii din luna trecută?“), în timp ce o operaţie tipică OLTP accesează doar o parte mai mică din înregistrări.
- **Date istorice**. Depozitele de date stochează datele pe o perioadă lungă de timp, luni sau ani. Acest lucru oferă suport pentru analiza istorică a informaţiei. Sistemele OLTP reţin date istorice atât timp cât este necesar pentru a îndeplini cu succes cerinţele tranzacţiilor curente.

Sistemele OLTP	*Data Warehouse*
Păstrează date curente	Păstrează date istorice
Stochează date detaliate	Stochează date detaliate, agregate uşor sau puternic
Datele sunt dinamice	Datele sunt în mare măsură statice
Prelucrare repetitivă	Prelucrare *ad-hoc*, nestructurată şi euristică
Nivel înalt de transfer al tranzacţiilor	Nivel mediu sau scăzut de transfer al tranzacţiilor
Tipar de utilizare previzibil	Tipar de utilizare imprevizibil
Conduse prin tranzacţii	Conduse prin analiză
Susţin deciziile de zi cu zi	Susţin deciziile strategice
Deservesc un număr mare de utilizatori	Deservesc un număr relativ redus de utilizatori din administraţie
Orientate spre aplicaţii	Orientate spre subiect

Fig. 1.3. OLTP *versus Data Warehouse*

O bază de date OLAP poate fi relaţională, dar datorită naturii ei orientate spre dimensiuni, au fost dezvoltate pentru gestionarea acestor baze de date construcţii multidimensionale, mai potrivite pentru o raportare flexibilă. Spre deosebire de bazele de date relaţionale, structura unei baze de date multidimensionale nu implică tabele, linii şi coloane, ci obiecte de următoarele tipuri: variabile, dimensiuni, niveluri, ierarhii, atribute.

Data Warehouse, care cuprinde de obicei informaţii despre o întreagă companie, poate fi subîmpărţită în baze de date departamentale, numite rafturi de date (*Data Marts*). De exemplu, poate exista un *Data Mart* al departamentului financiar, un altul al departamentului vânzări şi un altul pentru departamentul marketing.

Construcţia unei astfel de baze de date poate fi abordată în două moduri.

- O primă abordare este de a construi mai întâi un schelet al bazei de date la care se lipesc ulterior rafturile de date. Aceasta necesită o analiză prealabilă a întregului şi o delimitare a blocurilor componente. Ea cere un timp mai lung de dezvoltare, dar rezultatul este o bază de date unitară.
- A doua metodă este de a construi mai întâi rafturi specifice, efortul constând, în acest caz, în asocierea acestora. Această soluţie oferă mai rapid, aplicaţii funcţionale utilizatorilor, dar au de suferit unitatea şi portabilitatea aplicaţiilor finale.

Utilizatorii trebuie să-şi schimbe optica asupra bazelor de date pentru a fi capabili să folosească puterea şi flexibilitatea instrumentelor analitice de care dispun. Instrumentele OLAP au evoluat ca o modalitate de a rezolva interogările complicate necesare procesului de analiză a datelor. Combinaţia între bazele de date multidimensionale şi instrumentele analitice prietenoase face uşoară analiza, sinteza şi consolidarea datelor.

În ultimii ani, marii producători de sisteme de gestiune a bazelor de date relaţionale, precum *Oracle*, au introdus în produsele lor construcţii care să faciliteze accesul la datele din sistemele fundamentale pentru luarea de decizii. Astfel, noile versiuni de SGBD-uri ale firmelor mari prevăd o modalitate mai inteligentă de a realiza operaţia de compunere între două sau mai multe tabele, metode de indexare noi, potrivite pentru marile cantităţi de date statice cu care operează sistemele *Data Warehouse*, capacitatea de a detecta şi optimiza interogări de un tip special, posibilitatea de a folosi mai multe procesoare pentru a rezolva o interogare.

Un sistem *Data Warehouse* are un efect fundamental asupra utilizatorilor. Ei pot manevra mult mai flexibil sistemul, au posibilităţi elevate pentru interogarea datelor, dar ei trebuie să ştie cum să prelucreze şi să vizualizeze datele şi cum să le folosească în procesul de decizie.

Un efort ce trebuie făcut pentru construirea unui sistem de suport pentru decizii (DSS – *Decision Support System*) constă în procesul de descoperire a informaţiilor utile din baza de date. Acest proces, numit ***Data Mining*** sau *Knowledge Discovery in Databases* (KDD), procesează mari cantităţi de date, a căror corelare nu este neapărat evidentă, în vederea descoperirii de tendinţe şi tipare.

1.5. Arhitecturi *multi-user* pentru sisteme de gestiune a bazelor de date

Arhitecturile uzuale care sunt utilizate pentru implementarea sistemelor de gestiune a bazelor de date *multi-user* sunt: teleprocesarea, arhitectura fişier-*server* arhitectura *client-server*.

Teleprocesarea este arhitectura tradiţională, ce cuprinde un calculator cu o singură unitate CPU şi un numar de terminale care sunt incapabile să funcţioneze singure. Terminalele trimit mesaje la programele aplicaţie ale utilizatorilor, care la rândul lor, utilizează serviciile SGBD.

Această arhitectură a plasat o greutate teribilă asupra calculatorului central, care pe lângă rularea programelor de aplicaţii şi ale SGBD-ului, mai trebuie să preia şi din munca terminalelor (de exemplu, formatarea datelor pentru afişarea pe ecran).

Arhitectura fişier-server, presupune deja că procesarea este distribuită în reţea (de obicei o reţea locală LAN). Arhitectura cuprinde fişierele cerute de aplicaţii şi SGBD-ul. Aplicaţiile şi funcţiile SGBD sunt executate pe fiecare staţie de lucru, solicitând când este nevoie fişiere de pe *server*-ul de fişiere. Dintre dezavantaje se remarcă:

- existenţa unui trafic intens pe reţea;
- necesitatea unei copii complete a SGBD-ului pe fiecare staţie de lucru;
- acelaşi fişier poate fi accesat de mai multe SGBD-uri, ceea ce implică un control complex al integrităţii, simultaneităţii, reconstituirii.

Arhitectura *client-server* se referă la modul în care interacţionează componentele *software* pentru a forma un sistem. Există un proces *client*, care necesită resurse şi un proces *server*, care oferă resurse.

În arhitectura *client-server*, clientul (*front end*) emite, prin intermediul reţelei locale, o cerere *SQL* care este executată pe *server* (*back-end*); acesta trimite ca răspuns ansamblul înregistrărilor rezultat. Într-o astfel de interacţiune maşinile sunt eterogene, iar protocoalele de reţea pot fi distincte.

În contextul bazelor de date, ***client***-ul:

- administrează interfaţa cu utilizatorul şi logica aplicaţiei;
- acceptă şi verifică sintaxa intrărilor utilizatorilor;
- procesează aplicaţiile;
- generează cerinţele pentru baza de date şi le trimite *server*-ului;
- transmite răspunsul înapoi la utilizator.

În contextul bazelor de date, ***server***-ul:

- primeşte şi procesează cerinţele clienţilor pentru baza de date;
- verifică autorizarea;
- garantează respectarea constrângerilor de integritate;
- efectuează procesarea interogare-reactualizare şi trimite clientului răspunsul;
- realizează optimizarea interogărilor;
- asigură controlul concurenţei dintre mai multi clienţi care se ignoră (mecanisme de blocare);
- intreţine dictionarul datelor;
- oferă acces simultan la baza de date;
- asigură robusteţea în cazul defecţiunilor;
- oferă controlul reconstituirii etc.

Arhitectura tradiţională *client-server* pe „două etaje (straturi)“ presupune:

- *client*-ul – responsabil, în primul rand, de prezentarea datelor către client;
- *server*-ul – responsabil, în primul rand, de furnizarea serviciilor către client.

Arhitectura *client-server* pe „trei etaje“ presupune trei straturi, fiecare fiind rulat, potenţial, pe o platformă diferită.

- stratul (*client*) format din interfaţa cu utilizatorul, care este rulat pe calculatorul utilizatorului final;

- stratul (*server* de aplicaţie), ce manevrează logica aplicaţiilor şi prelucrării datelor, şi care poate servi mai mulţi clienţi (conectare la celelalte două straturi se face prin reţele locale LAN sau de mare suprafaţă WAN);
- stratul (*server*-ul de baze de date), care se ocupă cu validarea datelor şi accesarea bazei de date (stochează date necesare stratului din mijloc).

Arhitectura se potriveşte natural mediului *Web*. Un *browser Web* acţionând drept *client* şi un *server Web* fiind *server* de aplicaţie.

Middleware este un strat, evident *software*, între aplicaţia postului *client* şi *server*-ul de baze de date, constituit dintr-o interfaţă de programare a aplicaţiilor (API - *Application Programming Interface*) şi un protocol de reţea.

API descrie tipul de interacţiune dintre o aplicaţie *client* şi un *server* la distanţă, *via* un protocol de comunicaţie şi de formatare a datelor. Scopul existenţei interfeţei de programare a aplicaţiilor este de a oferi o interfaţă unică mai multor *server*-e de baze de date.

Este convenabil ca sistemele de baze de date să fie considerate ca fiind formate dintr-un *server* (sistemul SGBD însăşi) şi un set de clienţi (aplicaţiile). Frecvent, clienţii şi *server*-ul pot fi rulate pe calculatoare diferite, realizându-se un tip simplu de procesare distribuită. În general, fiecare *server* poate deservi mai multi clienţi, iar fiecare client poate accesa mai multe *server*-e. Dacă sistemul oferă transparenţă totală (fiecare client se poate comporta ca şi cum ar comunica cu un singur *server*, de pe un singur calculator) atunci este vorba despre un sistem de baze de date distribuite.

1.6. Tehnologia *Web* şi sistemele SGBD

Internet = o colecţie mondială de reţele de calculatoare.

Intranet = un sit *Web* sau un grup de sit-uri care aparţin unei organizaţii, accesibil numai pentru membrii acesteia.

Extranet = o reţea intranet care este parţial accesibilă utilizatorilor externi autorizaţi.

Reţea Web (*World Wide Web*) = un sistem bazat pe hipermedii care pune la dispoziţie un mijloc simplu, de tip „indicare-clic“ de răsfoire a informaţiilor pe Internet, folosind hiperlegăturile.

HTTP = protocolul utilizat pentru a transfera pagini *Web* prin intermediul Internetului.

HTML = limbajul de formatare a documentelor utilizat în proiectarea majorităţii paginilor *Web*.

Adresa *URL* = şir de caractere alfanumerice care reprezintă adresa unei resurse pe Internet şi modul în care trebuie accesată resursa.

Interfaţa de poartă comună (*CGI*) = defineşte modul în care scripturile comunică cu *server*-ul Web. Este tehnica de bază de integrare a bazelor de date în reţeaua *Web*.

În mediul *Web* funcţionează modelul *three tier* format din:

- un strat de interfaţă cu utilizatorul (client),
- un strat de logică a afacerii şi prelucrare a datelor (*server* de aplicaţii),
- un sistem SGBD (*server* de baze de date) distribuit pe calculatoare diferite.

Avantajele reţelei Web ca platformă de baze de date:

- avantagele SGBD;
- simplitate;
- independenţa de platformă;
- interfaţa grafică cu utilizatorul;
- acces transparent în reţea;
- standardizare (HTML standard *de facto*).

Arhitectura de calcul în reţea a sistemului *Oracle* (NCA *Network Computing Architecture*) se axează în principal pe furnizarea extensibilităţii pentru mediile distribuite. Arhitectura este construită pe baza tehnologiei *CORBA* pentru manipularea obiectelor. Este o structură *three tier* care se bazează pe utilizarea de:

- cartuşe de *software* care permit utilizatorilor să adauge funcţionalităţi individuale în aplicaţii (cartuşele pot fi construite în *Java*, *C/C++*, *Visual Basic*, *SQL* şi pot fi conectate la oricare din cele 3 straturi);
- protocoale deschise şi interfeţe standardizate care permit comunicarea între cartuşe (distribuite într-o reţea) prin intermediul unui program magistrală (ICX);
- clienţi extensibili, *server*-e de aplicaţie, *server*-e de baze de date;
- dezvoltarea şi administrarea integrată a cartuşelor.

Un cartus utilizeaza un limbaj de definire a interfetelor (IDL) pentru a putea fi identificat de alte obiecte intr-un sistem distribuit. De exemplu, PL/*SQL* este un astfel de cartus.

Arhitectura *multitier* a sistemului *Oracle9i*

Arhitectura cu mai multe niveluri (*multitier*) conţine următoarele elemente:

- unul sau mai mulţi *client*-i care iniţiază operaţii;
- unul sau mai multe *server*-e de aplicaţii care execută părţi ale operaţiilor;
- un *server* de baze de date care stochează datele folosite de operaţii.

Client-ul, care poate fi un *browser Web* sau un proces *user*, iniţiază o cerere pentru a executa o operaţie referitoare la informaţiile stocate în baza de date. Conectarea la *server*-ul bazei de date se face printr-unul sau mai multe *server*-e de aplicaţii.

Server-ul de aplicaţii constituie interfaţa dintre *client*-i şi *server*-ul bazei de date, asigurând accesul la informaţii. De asemenea, el include un nivel adiţional pentru securitate. *Server*-ul de aplicaţii îşi asumă identitatea *client*-ului, atunci când execută, pe *server*-ul de baze de date, operaţiile solicitate de acesta.

Arhitectura *multitier* permite folosirea unui *server* de aplicaţii pentru acreditarea *client*-ului, conectarea la *server*-ul de baze de date şi execuţia operaţiilor iniţiate de *client*. Privilegiile *server*-ului de aplicaţii sunt limitate pentru a preveni execuţia operaţiilor nedorite sau inutile în timpul unei operaţii *client*.

Server-ul de baze de date pune la dispoziţia *server*-ului de aplicaţii informaţiile necesare pentru soluţionarea operaţiilor lansate de către *client*. De asemenea, acesta face distincţia între operaţiile pe care *server*-ul de aplicaţii le cere în favoarea *client*-ului şi cele pe care le solicită în nume propriu.

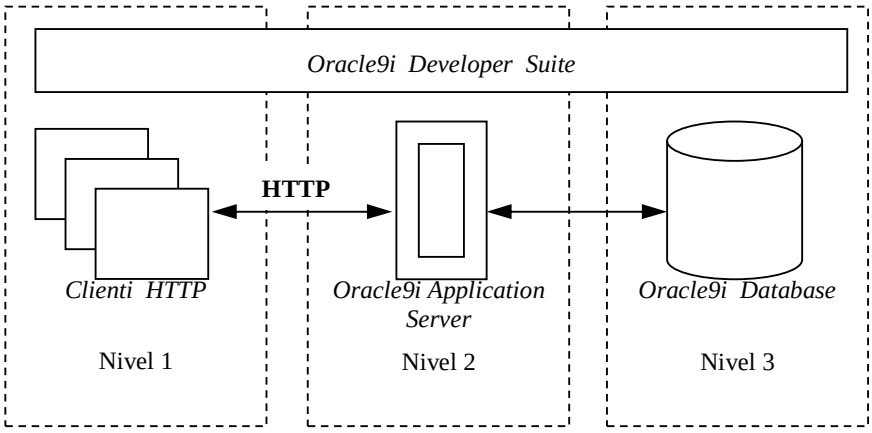

Fig. 1.4. Arhitectura *three-tier* a sistemului *Oracle9i*