

1. GENERALITĂȚI DESPRE BAZE DE DATE

1.1. Introducere

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

Baza de date este o colecție de date persistente, care sunt folosite de către sistemele de aplicații 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 șterse din bază numai printr-o cerere explicită adresată sistemului de gestiune.

Aici, termenul de „întreprindere“ este un cuvânt generic, utilizat pentru a desemna orice organizație independentă, de natură tehnică, comercială, științifică 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 funcționare** și conține o **mulțime de date** referitoare la modul său de operare.

Datele din baza de date pot fi atât integrate, cât și partajate. Noțiunea de **integrat** se referă la faptul că baza de date poate fi considerată ca o unificare a mai multor fișiere, iar prin **partajare** se înțelege că baza de date poate fi partajată concurent între diferiți utilizatori.

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

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

Dicționarul datelor (catalog de sistem), structurat și administrat ca o bază de date (metabază de date), conține „date despre date”, furnizează descrierea tuturor obiectelor unei baze de date, starea acestor obiecte, diversele constrângeri de securitate și de integritate etc. Dicționarul poate fi interogată, 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 și prelucrarea complexă a datelor;
- un dicționar al bazei de date (metabaza de date), ce conține informații despre date, structura acestora, statistici, documentație;
- mijloace *hardware* (comune sau specializate);
- reglementări administrative destinate bunei funcționări a sistemului;
- personalul implicat (utilizatori finali, administratorul datelor, administratorul bazei de date, proiectanți, programatori de aplicații).

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

- administratorii de date și baze de date,
- proiectanții (designeri) de baze de date,
- programatorii de aplicații,
- utilizatorii finali.

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

- decide care date trebuie stocate în baza de date;
- stabilește regulile de întreținere și 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 informațiilor 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 și 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 funcționării sistemului la performanțe adecvate, cu monitorizarea performanțelor;
- furnizează diverse servicii tehnice etc.

Proiectanții de baze de date pot acoperi atât aspectul fizic, cât și cel logic al concepției.

Proiectantul de baze de date care abordează direcția logică trebuie să posede o cunoaștere completă și amănunțită a modelului real de proiectat și a regulilor de funcționare ale acestuia. Practic, acesta proiectează conceptual baza de date, iar modelul creat este independent de programele de aplicații, de limbajele de programare. De asemenea, va proiecta logic baza de date, proiectare care este îndreptată spre un anumit model de date (relațional, orientat obiect, ierarhic etc.).

Proiectantul de baze de date fizice preia modelul logic de date și stabilește cum va fi realizat fizic. Acesta trebuie să cunoască funcționalitățile SGBD-ului, avantajele și 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 și metode de acces specifice, astfel încât să se asigure performanțe, se iau măsuri privind securitatea datelor.

Utilizatorii finali sunt cei care accesează interactiv baza de date. Aceasta a fost proiectată, implementată, întreținută pentru a satisface necesitățile informaționale ale clienților. Utilizatorii finali pot fi utilizatori simpli, care nu cunosc nimic despre baza de date, despre SGBD, dar accesează baza prin intermediul unor programe de aplicație. În general, această clasă de utilizatori alege anumite opțiuni din meniul aplicației. Există utilizatori finali sofisticăți, care sunt familiarizați cu structura bazei de date. Ei pot utiliza limbaje speciale pentru a exploata posibilitățile oferite de baza de date.

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

Cerințe minime care se impun unei baze de date:

- asigurarea unei redundanțe minime în date;
- furnizarea în timp util a informațiilor solicitate (timpul de răspuns la o interogare);
- asigurarea unor costuri minime în prelucrarea și întreținerea informației;
- capacitatea de a satisface, cu aceleași date, necesități informaționale ale unui număr mare de utilizatori,
- posibilitatea de adaptare la cerințe noi, răspunsuri la interogări neprevăzute inițial (flexibilitate);
- exploatarea simultană a datelor de către mai mulți utilizatori (sincronizare);
- asigurarea securității datelor prin mecanisme de protecție împotriva accesului neautorizat (confidențialitate);
- înglobarea unor facilități destinate validării datelor și 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 și anticiparea nevoilor viitoare (compatibilitate și expandabilitate);
- permisivitatea, prin ierarhizarea datelor după criteriul frecvenței acceselor, a unor reorganizări (eventual dinamice) care sporesc performanțele bazei.

În cadrul unei baze de date putem vorbi de patru **niveluri de abstractizare și de percepție** a datelor: intern, conceptual, logic și 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) și reprezintă viziunea programatorilor de sistem asupra datelor;
- Nivelul logic este descris de una din schemele logice posibile ale datelor și reprezintă viziunea programatorului de aplicație asupra datelor;
- Nivelul virtual (extern) reprezintă viziunea utilizatorului final asupra datelor.

Independența datelor cuprinde două aspecte fundamentale: o modificare a structurii fizice nu va afecta aplicația și reciproc, modificări ale aplicației vor lăsa nealterată structura fizică de date.

- **Independența fizică:** posibilitatea modificării schemei fizice a datelor fără ca aceasta să implice modificarea schemei conceptuale, a schemei logice și a programelor de aplicație. Este vorba despre imunitatea programelor de aplicație față de modificările modului în care datele sunt stocate fizic și accesate.
- **Independența logică:** posibilitatea modificării schemei conceptuale a datelor fără ca aceasta să implice modificarea schemei logice și a programelor de aplicație. Prin independența logică a datelor se urmărește a se crea fiecărui utilizator iluzia că este singurul beneficiar al unor date pe care, în realitate, le folosește în comun cu alți utilizatori.

Independența față de strategiile de acces permite programului să precizeze data pe care dorește 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 declarațiile și instrucțiunile executabile aparțin aceluiași limbaj. În lumea bazelor de date, funcțiile de declarare și 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 funcțiile principale sunt aceleași. La nivel conceptual, LDD realizează definirea entităților și a atributelor acestora, sunt precizate relațiile dintre date și strategiile de acces la ele, sunt stabilite criterii diferențiate de confidențialitate și de validare automată a datelor utilizate.
- Limbaje pentru prelucrarea datelor (LMD – *Data Manipulation Language*). Operațiile executate în cadrul unei baze de date presupun existența unui limbaj specializat, în care comenzile se exprimă prin fraze ce descriu acțiuni asupra bazei. În general, o comandă are următoarea structură: operația (calcul aritmetic sau logic, editare, extragere, deschidere-închidere, adăugare, ștergere, căutare, reactualizare etc.), criterii de selecție, mod de acces (secvențial, indexat etc.), format de editare. Există limbaje LMD procedurale, care specifică **cum** se obține rezultatul unei comenzi LMD și limbaje neprocedurale, care descriu doar datele **ce** vor fi obținute și nu modalitatea de obținere a acestora.

- Limbaje pentru controlul datelor (LCD – *Data Control Language*). Controlul unei baze de date se referă la asigurarea confidențialității și integrității datelor, la salvarea informației în cazul unor defecțiuni, la obținerea unor performanțe, la rezolvarea unor probleme de concurență.

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/I*, *Fortran*) care ajută la incorporarea de instrucțiuni *SQL* sau blocuri *PL/SQL* în programe scrise în alte limbaje, de nivel înalt, numite limbaje gazdă.

Sistemul de gestiune a bazelor de date interacționează cu programele de aplicație ale utilizatorului și cu baza de date, oferind o mulțime de facilități. Realizarea optimă a acestor facilități este asigurată de **obiectivele** fundamentale ale unui sistem de gestiune. Câteva dintre aceste obiective vor fi enumerate în continuare.

- Independența fizică. Obiectivul esențial este acela de a permite realizarea independenței structurilor de stocare în raport cu structurile de date din lumea reală. Se definește mulțimea de date indiferent de forma acesteia din lumea reală, ținând seama doar de a realiza un acces simplu la date și de a obține anumite performanțe.
- Independența logică. Grupul de lucru care exploatează baza de date poate să utilizeze diferite informații de bază (nu aceleași), pentru a-și construi entități și relații. Fiecare grup de lucru poate să cunoască doar o parte a semanticii datelor, să vadă doar o submulțime a datelor și numai sub forma în care le dorește. Această independență asigură imunitatea schemelor externe față de modificările făcute în schema conceptuală.
- Prelucrarea datelor de către neinformaticieni. Neinformaticienii văd datele independent de implementarea lor și pot exploata aceste date prin intermediul unui sistem de meniuri oferit de aplicația pe care o exploatează.
- Administrarea centralizată a datelor. Administrarea datelor presupune definirea structurii datelor și a modului de stocare a acestora. Administrarea este în general centralizată și permite o organizare coerentă și eficace a informației.
- Coerența datelor. Informația trebuie să satisfacă constrângeri statice sau dinamice, locale sau generale.

- Neredundanța datelor. Fiecare aplicație posedă datele sale proprii și aceasta conduce la numeroase dubluri. De asemenea, organizarea nejudicioasă a relațiilor poate să genereze redundanță în date. Administrarea coerentă a datelor trebuie să asigure neduplicarea fizică a datelor. Totuși, nu sunt excluse nici cazurile în care, pentru a realiza performanțe referitoare la timpul de acces la date și răspuns la solicitările utilizatorilor, să se accepte o anumită redundanță a datelor.
- Partajabilitatea datelor. Aceasta permite ca aplicațiile să partajeze datele din baza de date în timp și simultan. O aplicație poate folosi date ca și cum ar fi singura care le utilizează, fără a ști că altă aplicație, concurent, le poate modifica.
- Securitatea și confidențialitatea datelor. Datele trebuie protejate de un acces neautorizat sau rău intenționat. Există mecanisme care permit identificarea și autentificarea utilizatorilor și există proceduri de acces autorizat care depind de date și de utilizator. Sistemul de gestiune trebuie să asigure securitatea fizică și logică a informației și să garanteze că numai utilizatorii autorizați pot efectua operații corecte asupra bazei de date.

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

- complexitatea și dimensiunea sistemelor pot să crească considerabil, datorită necesității extinderii funcționalităților sistemului;
- costul, care variază în funcție de mediu și funcționalitatea oferită, la care se adugă cheltuieli periodice de întreținere;
- costuri adiționale pentru elemente de *hardware*;
- costul conversiei aplicațiilor existente, necesară pentru ca acestea să poată funcționa în noua configurație *hardware* și *software*;
- impactul unei defecțiuni asupra aplicațiilor, bazei de date sau sistemului de gestiune.

Structura unui sistem de gestiune a bazelor de date este de complexitate variabilă, iar nivelul real de funcționalitate diferă de la produs la produs. În orice moment apar noi necesități, care cer o nouă funcționalitate, astfel încât aceasta nu va putea deveni niciodată statică. În general, un SGBD trebuie să includă cel puțin 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 definiție a datelor, ce permit traducerea unor informații (care realizează descrierea datelor, a legăturilor logice dintre acestea și 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, ștergerea, reactualizarea sau consultarea informației dintr-o bază de date;
- module utilitare, care asigură întreținerea, prelucrarea, exploatarea corectă și ușoară a bazei de date;
- module de control, care permit controlul programelor de aplicație, asigurarea confidențialității și integrității datelor, rezolvarea unor probleme de concurență, recuperarea informației în cazul unor avarii sau defecțiuni *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 și tipul acestora;
- identifică informații ce permit accesul la structurile fizice de stocare (fișiere, volume etc.);
- verifică dacă datele sunt disponibile;
- extrag datele, fac conversiile, plasează datele în spațiul de memorie al utilizatorului;
- transmit informații de control necesare execuției comenzii, în spațiul de memorie al utilizatorului;
- transferă controlul programului de aplicație.

Prin urmare, din punct de vedere conceptual:

- utilizatorul lansează o cerere de acces;
- SGBD-ul acceptă cererea și o analizează;
- SGBD-ul inspectează pe rând, schema internă corespunzătoare utilizatorului, schema conceptuală, definiția structurii de stocare și corespondențele corespunzătoare;
- SGBD-ul execută operațiile necesare în baza de date stocată.

1.3. Arhitectura sistemelor de gestiune a bazelor de date

Asigurarea independenței fizice și 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 proprietățile elementare ale entităților, definirea tipurilor de date compuse care permit regruparea atributelor pentru a descrie entitățile modelului și legăturile între aceste entități, 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 fișierele care conțin aceste date, articolele din fișiere, 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 părți 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 securități 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 intenționat.

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 și anumite corespondențe dintre acestea:

- corespondența conceptual-intern definește relația dintre nivelul conceptual și baza de date stocată, specificând modul în care înregistrările și câmpurile conceptuale sunt reprezentate la nivel intern;
- corespondența extern-conceptual definește relația dintre o anumită vizualizare externă și nivelul (vizualizarea) conceptual, reprezentând cheia independenței logice de date;

- corespondența extern-extern permite definirea unor vizualizări externe în funcție de altele, fără a necesita o definiție explicită a corespondenței cu nivelul conceptual.

Arhitectura funcțională de referință propusă de grupul de lucru ANSI/X3/SPARC este axată pe dicționarul datelor și cuprinde două părți:

- prima, permite descrierea datelor (compoziția dicționarului datelor);
- a doua, permite prelucrarea datelor (interogarea și reactualizarea bazei de date).

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

Interfețele numerotate din figura 1.1, ce descriu arhitectura de referință 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 obține din compilarea celui precedent și permite aranjarea schemei obiect în dicționarul datelor.
- c) Limbaj de descriere a datelor conceptuale, format editare – permite administratorilor aplicațiilor și a bazelor să consulte schema conceptuală pentru a defini reguli de corespondență.
- d) Limbaje de descriere a datelor externe, format sursă – permit administratorilor aplicațiilor 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 și permit aranjarea schemelor externe (obiect) în dicționarul datelor.
- f) Limbaj de descriere a datelor interne, format sursă – permite administratorului bazei de date să definească schema internă și regulile de corespondență cu schema conceptuală.
- g) Limbaj de descriere a datelor interne, format obiect – corespunde formei compilate a celui precedent și permite aranjarea schemei interne (obiect) în dicționarul datelor.

- h) Limbaje de prelucrare a datelor externe, format sursă – permit programatorilor de aplicații 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 interfeței cu sistemul de stocare a datelor.
- m) Interfața cu memoria secundară – permite efectuarea de intrări/ieșiri în/din unitatea de memorie secundară.
- n) Interfața de acces la dicționarul datelor – permite diverselor procesoare de transformare să acceseze scheme obiect și reguli de corespondență.

Procesoarele din figura 1.1 au următoarele funcții:

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

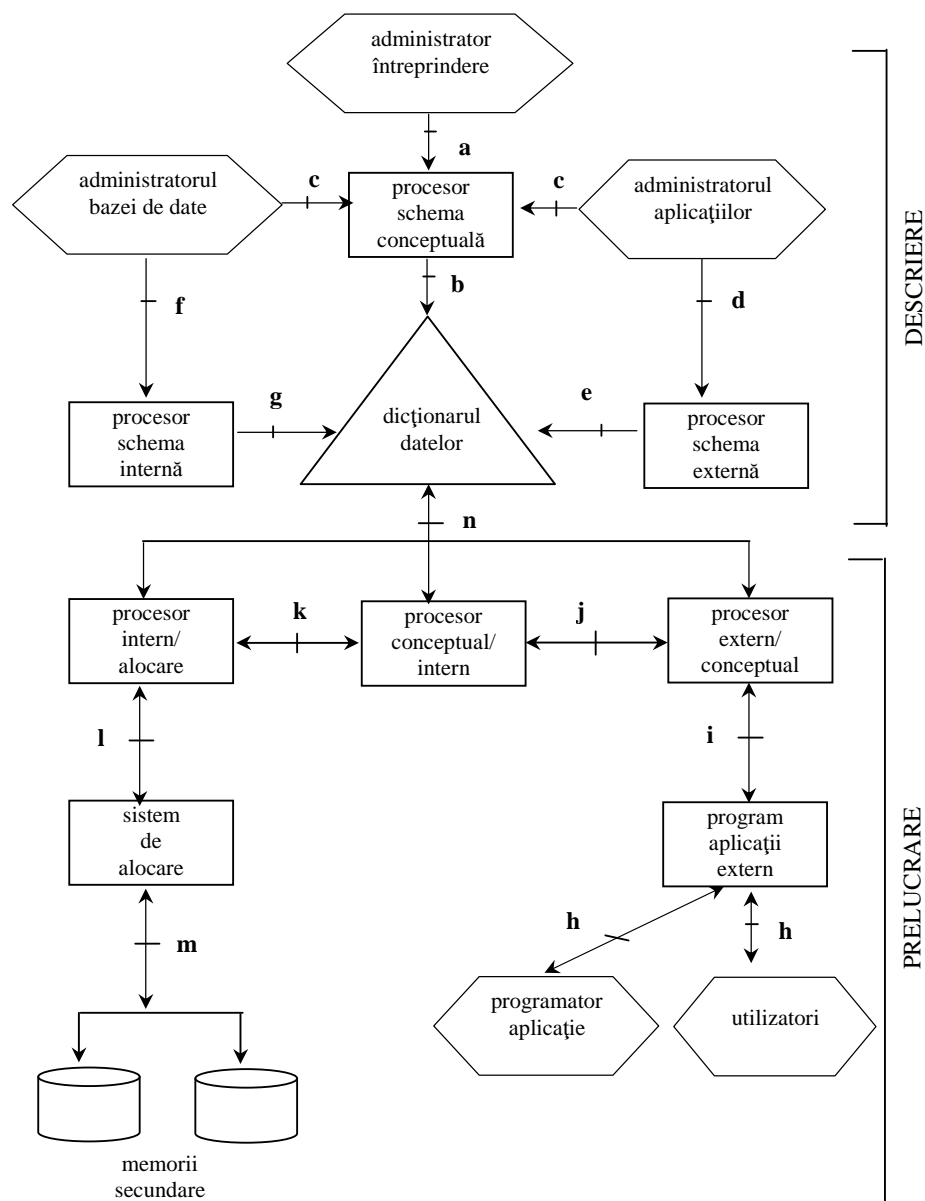


Fig. 1.1. Arhitectura de referință a unui SGBD.

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

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

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

Aceste comenzi, precum și cererile de prelucrare sunt analizate și tratate de un procesor numit **analizor**. Analizorul realizează analiza sintactică și semantică a cererii și o traduce în format intern. O cerere în format intern care face referință la o vizualizare este tradusă în una sau mai multe cereri care fac referință 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 și controlul integrității 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 operații de acces elementare și alege o ordine de execuție optimă. De asemenea, evaluează costul planului de acces înaintea execuției sale.

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

1.4. Evoluția bazelor de date

Istoria bazelor de date și a sistemelor de gestiune a bazelor de date poate fi rezumată în trei generații:

- sisteme ierarhice și rețea,
- sisteme relaționale,
- sisteme avansate (orientate obiect, relaționale orientate obiect, deductive, distribuite, multimedia, multibaze, active, temporale, decizionale, magazii de date etc.).

Baze de date ierarhice și rețea

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

Baze de date relaționale

A doua generație de SGBD-uri este legată de apariția modelelor relaționale (1970), care tratează entitățile ca niște relații. Piața actuală de baze de date este acoperită în majoritate de sisteme **relaționale**. Bazele de date relaționale sunt caracterizate de:

- structuri de date simple, intuitive,
- inexistența pointerilor vizibili pentru utilizator,
- constrângeri de integritate,
- operatori aplicați relațiilor care permit definirea, căutarea și reactualizarea datelor.

Dezvoltarea unei aplicații riguroase utilizând o bază de date relaționale necesită cunoașterea a trei niveluri de instrumente, eterogene din punct de vedere conceptual:

- nivelul instrumentelor grafice (interfața);
- nivelul aplicație, cu limbajele sale de dezvoltare;
- nivelul SGBD, cu standardul *SQL (Structured Query Language)*

ce permite definirea, prelucrarea și controlul bazei de date.

Baze de date orientate obiect

Bazele de date relaționale nu folosesc însă obiecte complexe și dinamice, nu realizează gestiunea datelor distribuite și nici gestiunea cunoștințelor. A treia generație de SGBD-uri, sistemele avansate, încearcă să depășească aceste limite ale sistemului relațional.

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

Un sistem relațional nu suportă obiecte dinamice care încorporează atât partea de date (informații) efective, cât și 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 concurenți. Sistemele de gestiune de baze de date orientate obiect (SGBDOO) prezintă o serie de avantaje:

- realizează o modelare superioară a informației,
- furnizează posibilități superioare de deducție (ierarhie de clase, moștenire),
- permit luarea în considerare a aspectelor dinamice și integrarea descrierii structurale și comportamentale,
- asigură îmbunătățirea interfeței cu utilizatorul.

Cu toate avantajele incontestabile oferite de SGBDOO-uri, impunerea lor pe piața bazelor de date nu a fost ușoară. Câteva motivații a acestei situații:

- absența unei fundamentări teoretice face imposibilă definirea unui SGBDOO de referință;
- gestiunea obiectelor complexe este mai dificilă decât accesul la relații prin cereri *SQL*;
- utilizatorii au investit sume uriașe în sistemele relaționale și nu le pot abandona cu ușurință. Trecerea la noua tehnologie orientată obiect implică investiții mari și nu păstrează aproape nimic din vechile soluții.

Baze de date relaționale orientate obiect

Simplitatea modelului relațional, combinată cu puterea tehnologiei orientate obiect a generat un domeniu nou și promițător în lumea bazelor de date, și anume bazele de date relaționale orientate obiect.

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

Baze de date deductive

O relație este o mulțime de înregistrări ce reprezintă fapte. Cunoștințele sunt aserțiuni generale și abstracte asupra faptelor. Cunoștințele permit să raționezi, ceea ce permite deducerea de noi fapte, plecând de la fapte cunoscute. Un SGBD relațional suportă o formă limitată de cunoștințe, și anume constrângerile de integritate, iar restul trebuie integrate în programele de aplicație. Aceasta generează probleme deoarece cunoștințele trebuie codificate în programe și apare imposibilitatea de a partaja cunoștințe între utilizatori. Totul se complică dacă există un volum mare de fapte.

Bazele de date deductive, utilizând programarea logică, gestionează cunoștințe relativ la baze de date care, în general, sunt relaționale. Bazele de date deductive permit deducerea de noi informații, plecând de la informațiile stocate în baza de date. Un SGBD deductiv posedă:

- un limbaj de definire a datelor care permite definirea structurii predicatelor sub formă de relații și constrângeri de integritate asociate;
- un limbaj de prelucrare a datelor care permite efectuarea reactualizărilor asupra datelor și formularea unor cereri;
- un limbaj de reguli de deducție 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 mașini ce sunt interconectate printr-o rețea de comunicație și utilizate într-un scop global. Administrarea și prelucrarea datelor distribuite, situate pe diferite calculatoare și 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 mașini diferite, în locuri diferite. Această mulțime de baze de date este exploatată de utilizator ca și cum ar fi o singură bază de date. Programul de aplicație care exploatează o bază de date distribuite poate avea acces la date rezidente pe mai multe mașini, fără ca programatorul să cunoască localizarea datelor.

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

Câteva dintre argumentele pentru a justifica această afirmație sunt:

- bazele relaționale oferă flexibilitate de descompunere în vederea distribuirii;
- operatorii relaționali pot fi folosiți pentru combinații dinamice ale informațiilor descentralizate;
- limbajele sistemelor relaționale sunt concise și asigură o economie considerabilă a transmiterii datelor. Ele fac posibil, pentru un nod oarecare al rețelei, să analizeze intenția unei tranzacții, să o descompună rapid în componente ce pot fi realizate local și componente ce pot fi transportate altor noduri.

Calculatoare și mașini baze de date

Soluția pentru a descentraliza prelucrarea datelor, în scopul evitării saturării memoriei și a procesoarelor calculatorului central, a fost apariția calculatoarelor baze de date și a mașinilor baze de date. Descentralizarea presupune transferarea unei părți din funcțiile unui SGBD către un calculator periferic (calculator *backend*) adică deplasarea algoritmilor de căutare și a celor de actualizare a datelor mai aproape de memoria secundară. Acest calculator periferic permite utilizarea optimă a resurselor și realizarea paralelismului în tratarea cererilor de informații.

Calculatorul periferic poate fi un calculator clasic, dar cu un *software* specific de SGBD (calculator bază de date) sau poate fi o mașină cu *hardware* specializat în gestiunea bazelor de date (mașină bază de date). Mașinile 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 execuție a acestora.

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

Multibaze de date

Diferite departamente ale unei organizații mai mari pot folosi diferite sisteme de gestiune. Cu toate că fiecare sistem este dezvoltat pentru a satisface nevoile propriului său departament, informațiile cu care lucrează pot fi utile și altor departamente. De aceea, pentru ca organizația să funcționeze bine, trebuie să existe o modalitate globală de 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 și limbajele de interogare.

Una dintre soluțiile folosite pentru a depăși dificultățile întâmpinate în respectarea autonomiei și a eterogenității 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 și 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 relațional pentru utilizatorii care sunt familiarizați 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 și un singur limbaj de interogare. Principalele sarcini ale sistemului *front-end* sunt gestionarea schemei globale și procesarea cererilor globale.

Un avantaj major al acestui model, față 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 aplicație 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 informații din domeniul lor de activitate, pe care le păstrează în tabele istorice și sunt nefolositoare sistemelor operaționale ale companiei, care funcționează cu date curente. Analizate, aceste date ar putea oferi informații despre tendințe și evoluții care ar putea interesa compania. Pentru a putea analiza aceste mari cantități de date este nevoie de tehnologii și instrumente speciale.

Ideea de a analiza colecții de date provenind din sistemele operaționale ale companiei sau din surse externe pentru a le folosi ca suport în procesul de decizie nu aparține ultimului deceniu, dar baze de date care să funcționeze eficient după aceste criterii au fost studiate și implementate în ultimii ani. Principalul scop al acestor baze de date a fost de a întâmpina nevoile sistemelor operaționale, a căror natură este inherent tranzacțională.

Sistemele tranzacționale sunt interesate, în primul rând, să controleze la un moment dat o singură tranzacție. De exemplu, într-un sistem bancar, atunci când clientul face un depozit, sistemul operațional bancar este responsabil de a înregistra tranzacția într-un tabel al tranzacțiilor și de a crește nivelul curent al contului clientului, stocat în alt tabel.

Un sistem operațional tipic operează cu evenimente predefinite și, datorită naturii lor, necesită acces rapid la date. Uzual, fiecare tranzacție operează cu cantități mici de date.

De-a lungul timpului, nevoile sistemelor operaționale nu se schimbă mult. Aplicația care înregistrează tranzacția, ca și cea care controlează accesul utilizatorului la informație (partea de raportare a sistemului bancar), nu se modifică prea mult. În acest tip de sistem, informația necesară în momentul în care un client inițiază o tranzacție trebuie să fie actuală. Înainte ca o bancă să aprobe un împrumut este nevoie să se asigure de situația financiară stabilă a clientului în acel moment, și nu cu un an înainte.

În ultimii ani s-au pus la punct principii și tehnologii noi care să servească procesului de analiză și administrare a datelor. O bază de date optimizată în acest scop definește 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 tranzacționale a fost numit procesare tranzacțională (OLTP – *On Line Transactional Processing*).

Aplicațiile unei *Data Warehouse* trebuie să ofere răspunsuri unor întrebări de tipul: „Care zi din săptămână este cea mai aglomerată?” „Ce clienți, cu care avem relații intense, nu au beneficiat de reduceri de prețuri?”. 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 organizației, oriunde s-ar afla acestea. Pentru o analiză corespunzătoare a organizației, afacerilor, cerințelor, tendințelor este necesară nu numai accesarea valorilor curente din baza de date, ci și a datelor istorice. Prin urmare, pentru a facilita acest tip specific de analiză a informației a fost creată magazia de date, care conține **informații extrase din diverse surse, întreținute de diverse unități operative, împreună cu istoricul și rezumatul tranzacțiilor**.

Sursele de date pentru o magazie cuprind:

- date operaționale, păstrate în baze de date ierarhice, de prima generație;
- date departamentale, păstrate în sisteme de fișiere patentate;

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

Data warehouse este o colecție de date:

- orientate spre subiect (principalele subiecte ale modelului sunt clienții, 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 informații se face într-un format consistent),
- variabile în timp (concentrarea depozitului de date se face asupra schimbărilor care au loc în timp).

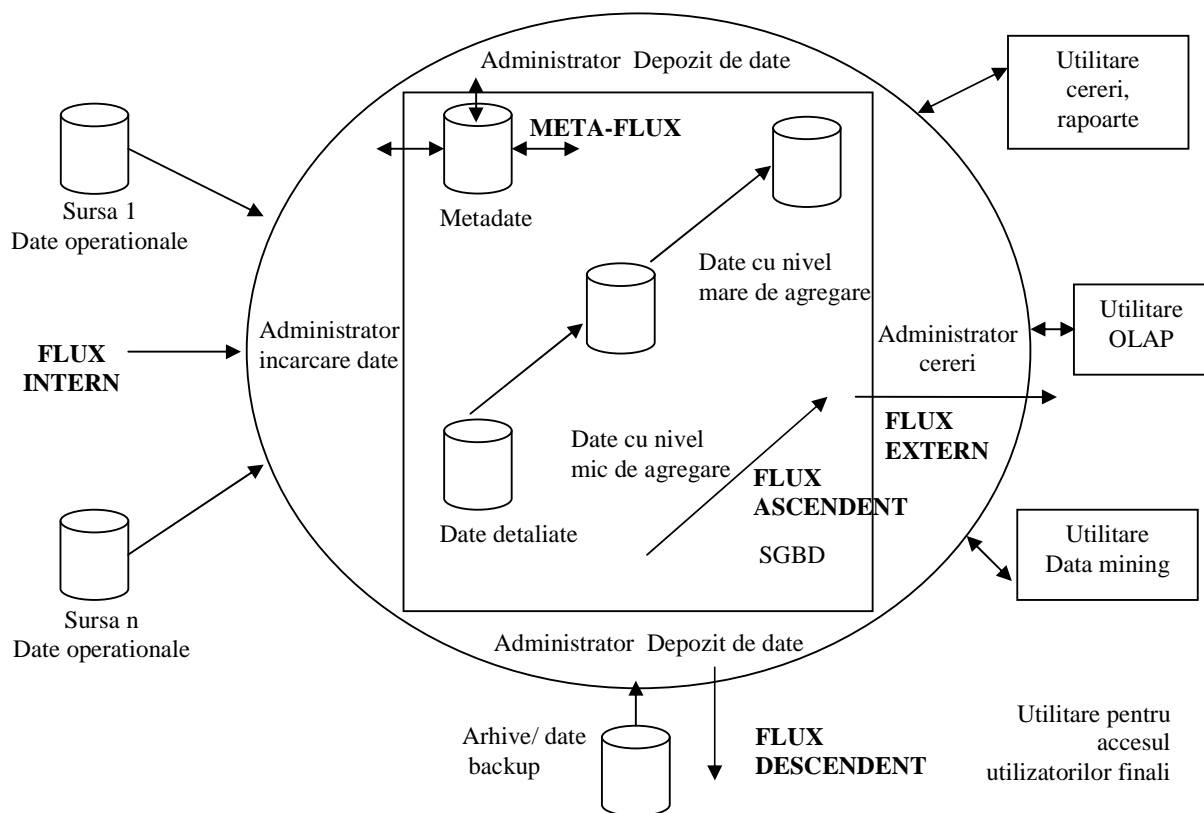


Fig 1.2. Arhitectura unui depozit de date

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

- fluxul intern, care reprezintă procesele asociate extragerii și încărcării datelor din fișierele sursă în magazia de date;
- fluxul ascendent, care reprezintă procesele asociate adăugării de valoare datelor din magazie, cu ajutorul împachetării și distribuirii;
- fluxul descendent, care reprezintă procesele asociate arhivării, salvării, refacerii datelor din magazie;
- fluxul extern, care reprezintă procesele asociate punerii la dispoziție 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 operațiile asociate cu obținerea (extragerea) și încărcarea datelor operaționale într-un depozit de date.
- Administratorul depozitului de date realizează toate operațiile legate de administrarea datelor din depozit. Operațiile realizate de componenta de administrare a depozitului de date includ: analiza datelor pentru a asigura consistența acestora; transformarea și mutarea datelor sursă din structurile temporare de stocare în tabelele depozitului de date; crearea de indecși și vizualizări asupra tabelelor de bază; generarea denormalizării (dacă este necesar); generarea agregărilor; crearea arhivelor și a *backup*-urilor.
- Administratorul cererilor (componenta *back-end*) realizează toate operațiile 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, facilitățile oferite de sistemul de baze de date și programele personalizate.
- În zona ce include date agregate sunt stocate toate agregările predefinite de date, pe diferite niveluri. Scopul, menținerii acestora, este de a mări performanța cererilor care necesită agregări. Datele agregate sunt actualizate permanent, pe măsură ce sunt încărcate noi informații în depozit.

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

Instrumentele de acces pentru utilizatorii finali ai magaziiilor de date:

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

Prelucrarea analitică *on-line* (OLAP) reprezintă sinteza, analiza și consolidarea dinamică a unor volume mari de date multidimensionale. Serverele de baze de date OLAP utilizează structuri multidimensionale pentru stocarea datelor și a relațiilor dintre date. Aceste structuri pot fi vizualizate prin cuburi de date și cuburi în cadrul cuburilor etc. Fiecare latură a cubului reprezintă o dimensiune. Serverele de baze de date OLAP multidimensionale acceptă operațiile analitice uzuale: consolidarea (gruparea), parcurgerea în jos (inversul consolidării), tranșarea, 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 informației într-un model multidimensional care este susținut de o bază de date:

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

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

De exemplu, *RISQL* (*Red Brick Intelligent SQL*), proiectat pentru analiștii 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 pieței, compararea anului curent cu cel precedent etc.

O altă problemă esențială este extragerea datelor și utilizarea acestora pentru luarea de decizii cruciale în domeniul afacerilor. Descoperirea unor noi corelații, tipare, tendințe, prin extragerea unor cantități mari de date folosind strategia inteligenței artificiale este una din modalitățile de rezolvare.

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

Tehnicile asociate operațiilor fundamentale de extragere sunt:

- modelarea predictivă (clasificarea cu ajutorul unei rețele neurale sau al unei inducții de tip arbore și previziunea valorilor, utilizând tehnici de regresie);
- segmentarea bazei de date (comasarea demografică și comasarea neurală care se deosebesc prin metodele utilizate pentru a calcula distanța dintre înregistrări, prin intrările de date permise);
- analiza legăturilor (descoperirea asocierilor, descoperirea tiparelor, descoperirea secvențelor de timp similare);
- detectarea deviațiilor (statistici și vizualizări pentru identificarea împrăștierii datelor, utilizând tehnici moderne de vizualizare grafică). În această clasă pot fi considerate, de exemplu, detectarea fraudelor privind utilizarea cărților de credit, pretențiile de despăgubire ale asiguraților 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 urmărește, în primul rând, obținerea de informații din baza de date, în timp ce unul OLTP urmărește introducerea de informații în baza de date. Datorită acestor diferențe, structura optimă a unei *Data Warehouse* este radical diferită de cea a unui sistem OLTP.

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

- **Operații.** 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 operații predefinite. Aplicațiile pot fi optimizate sau create special numai pentru acele operații.

- **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 și reflectă starea curentă a fiecărei tranzacții.
- **Proiectare.** Depozitele de date folosesc, în mod uzual, scheme denormalizate, în timp ce sistemele OLTP folosesc scheme normalizate pentru a optimiza performanțele operațiilor.
- **Operații tipice.** O simplă interogare a depozitului de date poate scana mii sau chiar milioane de înregistrări (de exemplu, cererea „Care sunt vânzările totale către toți clienții din luna trecută?“), în timp ce o operație 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 informației. Sistemele OLTP rețin date istorice atât timp cât este necesar pentru a îndeplini cu succes cerințele tranzacțiilor curente.

Sistemele OLTP	<i>Data Warehouse</i>
Păstrează date curente	Păstrează date istorice
Stochează date detaliate	Stochează date detaliate, agregate ușor sau puternic
Datele sunt dinamice	Datele sunt în mare măsură statice
Prelucrare repetitivă	Prelucrare <i>ad-hoc</i> , nestructurată și euristică
Nivel înalt de transfer al tranzacțiilor	Nivel mediu sau scăzut de transfer al tranzacțiilor
Tipar de utilizare previzibil	Tipar de utilizare imprevizibil
Conduse prin tranzacții	Conduse prin analiză
Susțin deciziile de zi cu zi	Susțin deciziile strategice
Deservesc un număr mare de utilizatori	Deservesc un număr relativ redus de utilizatori din administrație
Orientate spre aplicații	Orientate spre subiect

Fig. 1.3. OLTP *versus* Data Warehouse

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

Data Warehouse, care cuprinde de obicei informații despre o întreagă companie, poate fi subîmpărțită 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 și un altul pentru departamentul marketing.

Construcția 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 și 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 soluție oferă mai rapid, aplicații funcționale utilizatorilor, dar au de suferit unitatea și portabilitatea aplicațiilor finale.

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

În ultimii ani, marii producători de sisteme de gestiune a bazelor de date relaționale, precum *Oracle*, au introdus în produsele lor construcții 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 operația de compunere între două sau mai multe tabele, metode de indexare noi, potrivite pentru marile cantități de date statice cu care operează sistemele *Data Warehouse*, capacitatea de a detecta și 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 posibilități elevate pentru interogarea datelor, dar ei trebuie să știe cum să prelucreze și să vizualizeze datele și 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 informațiilor utile din baza de date. Acest proces, numit **Data Mining** sau *Knowledge Discovery in Databases* (KDD), procesează mari cantități de date, a căror corelare nu este neapărat evidentă, în vederea descoperirii de tendințe și 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 fișier-server arhitectura *client-server*.

Teleprocesarea este arhitectura tradițională, ce cuprinde un calculator cu o singură unitate CPU și un număr de terminale care sunt incapabile să funcționeze singure. Terminalele trimit mesaje la programele aplicație 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 aplicații și ale SGBD-ului, mai trebuie să preia și din munca terminalelor (de exemplu, formatarea datelor pentru afișarea pe ecran).

Arhitectura fișier-server, presupune deja că procesarea este distribuită în rețea (de obicei o rețea locală LAN). Arhitectura cuprinde fișierele cerute de aplicații și SGBD-ul. Aplicațiile și funcțiile SGBD sunt executate pe fiecare stație de lucru, solicitând când este nevoie fișiere de pe *server*-ul de fișiere. Dintre dezavantaje se remarcă:

- existența unui trafic intens pe rețea;
- necesitatea unei copii complete a SGBD-ului pe fiecare stație de lucru;
- același fișier poate fi accesat de mai multe SGBD-uri, ceea ce implică un control complex al integrității, simultaneității, reconstituirii.

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

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

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

- administrează interfața cu utilizatorul și logica aplicației;
- acceptă și verifică sintaxa intrărilor utilizatorilor;
- procesează aplicațiile;
- generează cerințele pentru baza de date și le trimite *server*-ului;
- transmite răspunsul înapoi la utilizator.

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

- primește și procesează cerințele clienților pentru baza de date;
- verifică autorizarea;
- garantează respectarea constrângerilor de integritate;
- efectuează procesarea interogare-reactualizare și trimite clientului răspunsul;
- realizează optimizarea interogărilor;
- asigură controlul concurenței dintre mai mulți clienți care se ignoră (mecanisme de blocare);
- întreține dicționarul datelor;
- oferă acces simultan la baza de date;
- asigură robustețea în cazul defecțiunilor;
- oferă controlul reconstituirii etc.

Arhitectura tradițională *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, potențial, pe o platformă diferită.

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

- stratul (*server* de aplicație), ce manevrează logica aplicațiilor și prelucrării datelor, și care poate servi mai mulți clienți (conectare la celelalte două straturi se face prin rețele locale LAN sau de mare suprafață WAN);
- stratul (*server*-ul de baze de date), care se ocupă cu validarea datelor și accesarea bazei de date (stochează date necesare stratului din mijloc).

Arhitectura se potrivește natural mediului *Web*. Un *browser Web* acționând drept *client* și un *server Web* fiind *server* de aplicație.

Middleware este un strat, evident *software*, între aplicația postului *client* și *server*-ul de baze de date, constituit dintr-o interfață de programare a aplicațiilor (API - *Application Programming Interface*) și un protocol de rețea.

API descrie tipul de interacțiune dintre o aplicație *client* și un *server* la distanță, *via* un protocol de comunicație și de formatare a datelor. Scopul existenței interfeței de programare a aplicațiilor este de a oferi o interfață 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 însăși) și un set de clienți (aplicațiile). Frecvent, clienții și *server*-ul pot fi rulate pe calculatoare diferite, realizându-se un tip simplu de procesare distribuită. În general, fiecare *server* poate deservi mai mulți clienți, iar fiecare client poate accesa mai multe *server*-e. Dacă sistemul oferă transparență totală (fiecare client se poate comporta ca și 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* și sistemele SGBD

Internet = o colecție mondială de rețele de calculatoare.

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

Extranet = o rețea intranet care este parțial accesibilă utilizatorilor externi autorizați.

Rețea Web (*World Wide Web*) = un sistem bazat pe hipermedii care pune la dispoziție un mijloc simplu, de tip „indicare-clic” de răsfoire a informațiilor pe Internet, folosind hiperlegăturile.

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

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

Adresa URL = șir de caractere alfanumerice care reprezintă adresa unei resurse pe Internet și modul în care trebuie accesată resursa.

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

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

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

Avantajele rețelei *Web* ca platformă de baze de date:

- avantajele SGBD;
- simplitate;
- independența de platformă;
- interfața grafică cu utilizatorul;
- acces transparent în rețea;
- standardizare (HTML standard *de facto*).

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

- cartușe de *software* care permit utilizatorilor să adauge funcționalități individuale în aplicații (cartușele pot fi construite în *Java*, *C/C++*, *Visual Basic*, *SQL* și pot fi conectate la oricare din cele 3 straturi);
- protocoale deschise și interfețe standardizate care permit comunicarea între cartușe (distribuite într-o rețea) prin intermediul unui program magistrală (ICX);
- clienți extensibili, *server*-e de aplicație, *server*-e de baze de date;
- dezvoltarea și administrarea integrată a cartușelor.

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

Arhitectura *multitier* a sistemului *Oracle9i*

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

- unul sau mai mulți *client*-i care inițiază operații;
- unul sau mai multe *server*-e de aplicații care execută părți ale operațiilor;
- un *server* de baze de date care stochează datele folosite de operații.

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

Server-ul de aplicații constituie interfața dintre *client*-i și *server*-ul bazei de date, asigurând accesul la informații. De asemenea, el include un nivel adițional pentru securitate. *Server*-ul de aplicații își asumă identitatea *client*-ului, atunci când execută, pe *server*-ul de baze de date, operațiile solicitate de acesta.

Arhitectura *multitier* permite folosirea unui *server* de aplicații pentru acreditarea *client*-ului, conectarea la *server*-ul de baze de date și execuția operațiilor inițiate de *client*. Privilegiile *server*-ului de aplicații sunt limitate pentru a preveni execuția operațiilor nedorite sau inutile în timpul unei operații *client*.

Server-ul de baze de date pune la dispoziția *server*-ului de aplicații informațiile necesare pentru soluționarea operațiilor lansate de către *client*. De asemenea, acesta face distincția între operațiile pe care *server*-ul de aplicații le cere în favoarea *client*-ului și cele pe care le solicită în nume propriu.

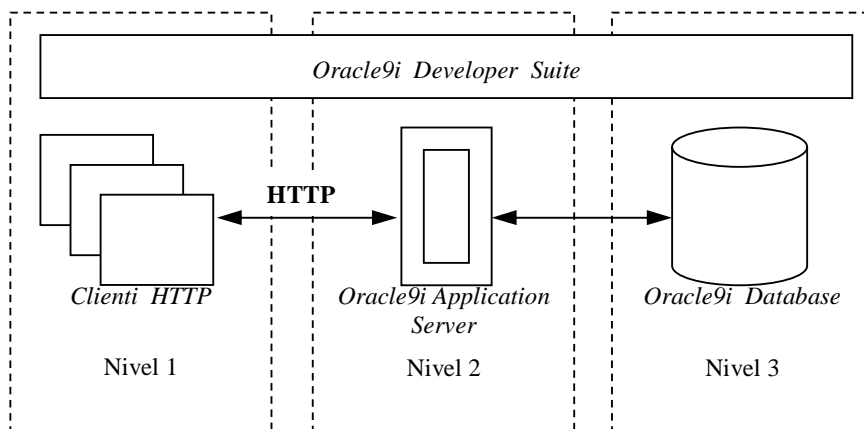


Fig. 1.4. Arhitectura *three-tier* a sistemului *Oracle9i*