Cofinanțat de Uniunea Europeană

Proiectul Wallachia eHUB (WEH)
ID proiect: EC/101083410 – WeH; POCIDIF/1147/2/1/161799

Str. Italiană nr. 28, sect. 2, București - România
weh@spiruharet.ro weh.spiruharet.ro

Categoria de servicii: ***WP 3. Servicii de dezvoltare a competențelor și de formare profesională (Skills Development and Training Services)***

Subcategoria de servicii: ***Sesiuni de instruire pe soluții digitale avansate (Training sessions on high performance digital solutions)***

SESIUNE DE INSTRUIRE PENTRU TEHNOLOGII CU BAZE DE DATE

- Baze de date relaționale și baze de date orientate obiect -

Parteneri WEH: Universitatea *Spiru Haret*

Trainer de coordonare a transformarii digitale:

Andronie Mihai

Str. Italiană nr. 28, sect. 2, București - România
weh@spiruharet.ro weh.spiruharet.ro

CUPRINS

1. Noțiuni fundamentale utilizate în organizarea datelor

1.1. Concepte de bază

Datele sunt stocate în memoria internă si memoria externă a oricărui sistem de calcul. Organizarea datelor se referă la procesul de definire si structurare a datelor în colecții de date, precum si la realizarea legăturilor între elementele unei colecții si între colecțiile de date. Organizarea datelor se proiectează în scopul regăsirii automate a acestora după diverse criterii.

Obiectivele organizării datelor sunt, în principal, următoarele:

- timp de acces minim la date;
- apariția o singură dată a datelor în sistem;
- spațiu de memorie internă si externă pentru date cât mai mic;
- reflectarea prin organizare a tuturor legăturilor dintre procesele economice pe care aceste date le reprezintă;
- posibilitatea modificării structurii datelor si a relațiilor dintre date fără a produce schimbări în programele care le gestionează.

Tehnicile de organizare a datelor în colecții de date sunt: fisierul de date si baza de date.

Fisierul de date reprezintă o colecție de date memorată pe un suport tehnic într-o succesiune de înregistrări. Accesul la o înregistrare din fisierul de date se obține prin parcurgerea înregistrărilor fisierului în secvența în care au fost stocate (acces secvențial) sau pe baza unei chei de identificare care să permită regăsirea rapidă a înregistrării (acces direct). Accesul direct se obține prin indexarea fisierelor, adică prin crearea unor tabele de indecsi care pentru fiecare valoare a atributului cheie primară (atribut care permite identificarea în mod unic a unei înregistrări din fisier) să conțină adresa corespunzătoare (în cadrul fisierului) a fiecărei înregistrări.

Această organizare a datelor în fisiere de date prezintă următoarele dezavantaje:

- redundanță mare (stocarea acelorasi date în mai multe fisiere);
- acces dificil la date; exploatarea multiutilizator a datelor necesita operații suplimentare de sortare, fuziune, ventilare etc.;
- izolarea datelor, adică nu pot fi realizate programe pe calculator care să acceseze datele într-o manieră globală;
- actualizarea datelor, prin adăugare, modificare, stergere, generează conflicte atunci când mai mulți utilizatori doresc să modifice simultan aceleasi date;

- dependența programelor față de date; deoarece datele se descriu în programe, modificările din structura datelor obligă la efectuarea de corecturi în programele pe calculator;
- problemele neprevăzute nu obțin răspunsuri rapide;
- fiecare dată este descrisă independent în toate fisierele în care apare; dacă într-un fisier se modifică formatul si valoarea unei date, acea modificare nu se transmite automat, pentru aceeasi dată, în toate fisierele de date; ca urmare, pentru aceeasi dată se creează posibilitatea apariției de valori diferite în fisiere diferite (inconsistența datelor);
- nu se menține integritatea datelor, atunci când fisierul este realizat cu limbaje diferite.

Cresterea necesarului de date, informații si cunostințe pentru agenții economici si progresele tehnologiilor informației si ale comunicațiilor (IT&C) au determinat organizarea datelor în baze de date.

Noțiunile fundamentale folosite în organizarea datelor sunt entitatea, atributul si valoarea. Între acestea există legături de interdependență astfel:

- o entitate are mai multe atribute, iar atributele au o anumită mulțime de valori;
- entitatea reprezintă un obiect concret sau abstract definit prin proprietățile sale;
- orice proprietate a unui obiect este exprimată printr-o pereche (ATRIBUT, VALOARE).

1.2 Relații între date

Între date există relații sau legături diferite. Între datele care aparțin unor tipuri de entități se pot realiza două feluri de legături:

- primă legătură se exprimă prin apartenența datelor la entitate;
- a doua legătură se defineste pentru entitățile de acelasi tip sau de tipuri diferite.

Exemple:

a) Dacă se notează cu SALARIATI mulțimea salariaților unei societăți comerciale, între datele a1 si a2 ce aparțin acestei mulțimi, se pot defini relații de tipul:

- a1 are aceeasi funcție de încadrare cu a2;
- a1 are acelasi salariu cu a2;
- a1 are aceeasi vârstă cu a2 etc.

b) Se consideră două clase de entități: PRODUSE_BANCARE si CLIENTI. Între datele acestor două clase de entități se pot defini relațiile:

- un produs bancar poate fi achiziționat de unul sau mai mulți clienți ai băncii;
- un client al băncii poate achiziționa unul sau mai multe produse bancare.

1.3. Structuri de date

Structura de date este o colecție de date între care s-au stabilit un ansamblu de relații pe baza cărora funcționează un mecanism de selecție si identificare a componentelor. Altfel exprimat, o structură de date reprezintă un anumit aranjament al datelor atunci când sunt stocate în memoria unui calculator. Datele din structurile de date pot fi manipulate cu ajutorul algoritmilor, în mai multe moduri, sortând datele sau căutând un anumit element. Structurile de date, în afara situației de instrumente de programare, servesc pentru stocarea si modelarea unor date din universul real.

Mulțimea de date, asociată structurii de date, poate cuprinde datele unui tip sau ale mai multor tipuri de entități. Componentele structurii se identifică prin nume sau prin poziția pe care o dețin în structură în raport cu ordinea specificată.

În situația în care pentru localizarea unei componente se parcurg toate celelalte componente dinaintea ei, structura are acces secvențial. În schimb, atunci când o componentă poate fi selectată fără a ține seama de celelalte, structura are acces direct.

Componentele unei structuri de date sunt date elementare sau sunt ele însele structuri de date.

Asupra unei structuri de date se pot efectua următoarele operații:

- crearea (înseamnă memorarea datelor inițiale pe suportul de stocare);
- actualizarea (schimbarea stării structurii prin adăugare, modificare sau stergere de elemente, modificarea valorii sau relațiilor dintre elemente);
- consultarea (accesarea componentelor structurii de date);
- sortarea (aranjarea elementelor unei structuri de date în conformitate cu criterii prestabilite);
- ventilarea (divizarea unei structuri de date în două sau mai multe structuri de date);
- fuzionarea (formarea unei structuri de date noi din două sau mai multe structuri de date) etc.

Structurile de date care prezintă aceeasi organizare si asupra cărora se execută aceleasi operații formează un anumit tip de structuri de date. Tipul de structură de date reprezintă o mulțime ordonată de date între care s-au stabilit anumite relații si pentru care realizarea operațiilor se efectuează cu un grup de operatori de bază care au o anumită semantică.

Dacă se ia în considerare tipul componentelor, structurile de date se clasifică în:

- omogene (componentele sunt de acelasi tip);
- eterogene (componentele au tipuri diferite).

Str. Italiană nr. 28, sect. 2, București - România
weh@spiruharet.ro weh.spiruharet.ro

Când structura de date se descompune în structuri de date de acelasi tip, atunci structura obținută este denumită recursivă.

După nivelul de structurare al datelor, se deosebesc:

- structura fizică (structura de date care se referă la modul de implementare pe suporți tehnici informaționali);
- structura logică (modul de ordonare a datelor si modul de folosire a operatorilor de tratare a datelor).

Dacă se ia în considerare posibilitatea modificării valorilor si a structurilor, se identifică:

- structuri statice (pe tot parcursul existenței acestora prezintă acelasi număr de componente si în aceeasi ordine (adică au cardinalitate finită, prin cardinalitate înțelegând numărul elementelor mulțimii);
- structuri dinamice (permit modificarea valorilor si a structurii de date prin aplicarea unor operatori; aceste structuri de date au cardinalitate infinită deoarece prezintă un număr nelimitat de componente).

O structură logică poate fi implementată atât ca structură statică cât si ca structură dinamică. Există însă si structuri logice ce nu pot fi implementate static. În organizarea datelor trebuie definită atât structura logică, cât si cea fizică, deorece cele două nivele se condiționează reciproc.

Din punctul de vedere al tipului de structură de date, se deosebesc:

- structura de date punctuală (o entitate de grup izolată);
- structura de date liniară (când există o relație de ordine totală între elementele colecției de date; primul element nu are predecesori; ultimul element nu are succesori; între date se stabilesc relații de tipul "unu-la-unu"; când ultimul element coincide cu primul element, structura liniară devine structură circulară sau inelară);
- structura de date arborescentă (este denumită si structură de date ierarhică sau descendentă; acest tip de structură de date se defineste când există o relație de ordine între elementele colecției de date; există un element unic care este denumit nodul rădăcină (root node);orice nod diferit de nodul rădăcină prezintă un predecessor imediat unic; orice nod care nu este terminal prezintă un număr finit de succesori imediați; între noduri se stabilesc relații de tipul "unu-la-mulți);
- structura de date rețea (acest tip de structură de date se defineste când există o relație de preordine între elementele colecției de date; un nod prezintă mai mulți predecesori; un nod poate fi predecesor pentru propriul său predecesor; între elementele rețelei se stabilesc relații de tipul "mulți-la-mulți);

- structura de date relațională (acest tip de structură de date este formată din mai multe tabele, relații sau tablouri de date elementare).

Datele si structurile de date pot fi predefinite sau definite de utilizator.

2. Teoria bazelor de date si a sistemelor de gestiune a bazelor de date Organizarea datelor în baze de date. Sisteme de gestiune a bazelor de date.

Un sistem de calcul din compunerea sistemului informatic are organizate datele într-o ierarhie care începe cu biți si octeți (bytes) si continuă cu câmpuri, înregistrări, fisiere, baze de date si depozite de date. Sistemul bază de date se defineste ca fiind ansamblul de colecții organizate de date, împreună cu descrierea datelor si a relațiilor dintre ele, care reprezintă, complet, corect si coerent, universul real al organizație economice (compartimentului specializat al acesteia) prin caracteristicile relevante (reprezentative) ale elementelor sale, percepute de sistem prin semantica lor (semnificația lor reală) si prin legăturile dintre aceste caracteristici (fig. 1).

Conceptul de bază de date a fost introdus în anul 1969, cu prilejul prezentării primului raport CODASYL. Ulterior si alte grupuri de lucru specializate (IIBM, ANSI, DBTG) si-au adus contribuția la standardizarea conceptelor din teoria bazelor de date.

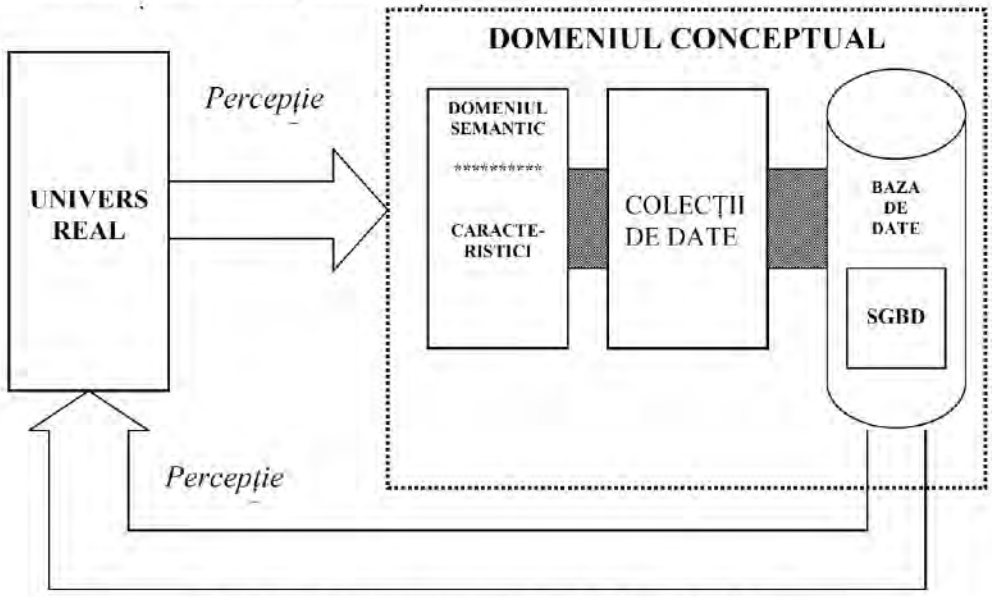


Fig. 1 Definirea conceptului de bază de date

Colecția de date, se defineste ca fiind mulțimea de valori (date) pe care le iau caracteristicile reprezentative ale unui element din universul real al organizației economice, dacă la fiecare moment

de timp se aplică asupra lor un predicat, o acțiune din realitatea organizației economice, împreună cu domeniile de definiție reale ale acestor caracteristici.

Într-un sistem bază de date, descrierea datelor constă în descrierea structurii de date a sistemului bază de date si în descrierea regulilor care asigură coerența datelor, în raport cu universul real al organizației economice reprezentat. Se reaminteste că tipurile de structuri logice de date sunt: punctuală, liniară, arborescentă, rețea, relațională, orientată pe obiecte (OO).

Structura de date a unui sistem bază de date este determinată de modelul abstract de reprezentare a datelor folosit, numit bază de date. În funcție de tipul stabilit pentru legăturile dintre datele din colecțiile de date (ierarhic, rețea, relațional, orientat pe obiecte), s-au realizat mai multe modele abstracte de reprezentare a datelor, dar fiecăruia îi corespunde o singură structură de date a sistemului bază de date. Din acest motiv s-a generalizat utilizarea conceptului de bază de date, BD sau DB (DataBase), care este folosit atât pentru denumirea structurii de date a unui sistem bază de date, cât si pentru denumirea modelului abstract de reprezentare a datelor care o determină. Mai mult chiar, conceptul de bază de date denumeste atât colecția organizată, cît si structura de date folosită pentru reprezentarea acesteia în sistemul bază de date.

Sistemul de gestiune a bazei de date, SGBD sau DBMS (Data-Base Management System) reprezintă un ansamblu complex de programe care asigură interfața dintre baza de date si utilizator.

O bază de date trebuie să satisfacă următoarele condiții:

- structura bazei de date trebuie să asigure informațiile necesare si suficiente pentru îndeplinirea cerințelor de informare si decizie;
- să asigure o independență sporită a datelor față de programe si invers;
- să se realizeze o redundanță (cardinalitatea informațiilor colecțiilor de date) minimă si controlată a datelor memorate;
- accesul la datele stocate în baza de date să fie rapid si eficace.

O bază de date poate să fie exploatată, de regulă, în regim de prelucrare pe loturi (batch) si în regim conversațional. Accesarea bazei de date se realizează prin aplicații generale, programe de aplicație, limbaje de manipulare autonome (procedurale si neprocedurale), interfețe specializate cu limbajele de programare clasice etc., local sau de la distanță, prin utilizarea calculatoarelor singulare sau a rețelelor de calculatoare. Rezultatele interogărilor utilizatorilor se prezintă sub formă vizuală, listată, prin memorare pe diversi suporți tehnic de date, local sau la distanță.

Sistemul bază de date are rolul de organizare si stocare a unor volume mari de date, în vederea gestionării, prelucrării, distribuirii si utilizării multiple, folosind sistemele de calcul, programele utilitare si programele de aplicație.

Pornind de la funcția sa, un sistem bază de date este format, ca structură generală, din: colecții de date, baza de date, SGBD, programe de aplicație si utilitare, precum si utilizatori. Dacă conceptul de bază de date denumeste atât colecțiile de date cît si structura de date folosită pentru reprezentarea acesteia, atunci structura generală a sistemului bază de date este baza de date, SGBD, programe de utilizare, utilizatori.

Arhitectura unui sistem bază de date este prezentată în fig. 2. În conformitate cu specificațiile utilizatorilor finali (end-users), programatorii de aplicație, având la dispoziție utilitare (programe specializate de proiectare) si prin colaborarea cu administratorul bazei de date (acesta lucrează nemijlocit cu schema bazei de date), pun la punct programele de aplicație. Asa cum s-a precizat deja, interfața dintre baza de date si schema BD, utilitare si programele de aplicație este sistemul de gestiune a bazei de date, SGBD.

Obiectivele unui SGBD sunt, în principal, următoarele:

- asigurarea independenței datelor față de aplicație;
- asigurarea redundanței minime si controlate a datelor;
- asigurarea tuturor facilităților posibile de exploatare a datelor;
- asigurarea securității si protecției datelor împotriva accesului neautorizat (inclusiv prin criptarea datelor);
- asigurarea coerenței si integrității datelor împotriva stergerilor accidentasle sau intenționate;
- asigurarea partajării datelor (accesul concurent al utilizatorilor la baza de date);
- asigurarea nivelului de performanță globală (volum mare de date complexe gestionate cu un timp de răspuns acceptabil la adresarea cererilor de interogare din partea utilizatorilor multipli).

Funcțiile generale ale unui SGBD sunt:

1. descrierea datelor (definirea structurii bazei de date prin intermediul limbajului de definire a datelor);
2. manipularea datelor (încărcarea, actualizarea, prelucrarea si regăsirea datelor cu ajutorul limbajului de manipulare a datelor);
3. utilizarea bazei de date (de către toate categoriile de utilizatori);
4. administrarea bazei de date.

Fiecare grup de lucru pentru standardizarea bazelor de date (CODASYL si ANSI, în principal) a propus o arhitectură proprie a unui SGBD.

Comenzile SGBD (DBMS) pot fi grupate în trei categorii de limbaje:

a. limbajul de definire a datelor (DDL, Data Definition Language);
b. limbajul de manipulare a datelor (DML, Data Manipulation Language);

c. limbajul de descriere a stocării datelor (DSDL, Data Storage Description Language).

Limbajul de definire a datelor asigură, în principal:

- definirea tuturor tipurilor de înregistrări si de câmpuri de date, precum si asocierea corespondenței acestora cu nivelul conceptual;
- specificarea ordinii logice a câmpurilor de date;
- definirea câmpurilor ce vor fi folosite drept chei de căutare;
- definirea drepturilor de acces;
- definirea legăturilor între tipurile de înregistrări.

Limbajul de manipulare a datelor permite:

- parcurgerea structurilor si a legăturilor existente;
- accesul la înregistrări prin adresă sau prin conținutul acestora;
- actualizări ale înregistrărilor;
- reordonări ale câmpurilor de date;
- definirea tranzacțiilor si a condițiilor de eroare.

Limbajul de descriere a stocării datelor oferă posibilități de:

- asociere a fisierelor la programele de aplicație, a dispozitivelor fizice, alocare de spații de memorie;
- specificarea zonelor de lucru permanente si tranzitorii;
- definirea si izolarea datelor confidențiale;
- specificarea structurilor de memorare, a mecanismelor de adresare, a modului de translatare a înregistrării logice în înregistrare fizică;
- crearea indecsilor asociați cheilor de căutare.

Operațiile ce se execută asupra unei baze de date sunt:

- creare;
- încărcare (populare);
- consultare: căutare (selecție);
- actualizare: modificare, adăugare articole noi, stergerea unor articole, ordonare (sortare, indexare), prelucrare etc.

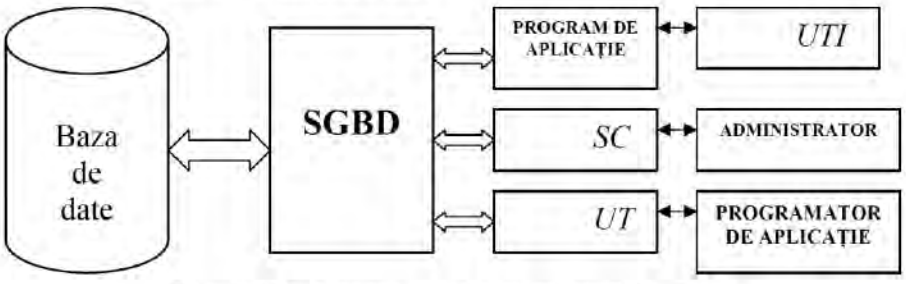


Fig. 2 Arhitectura unui sistem bază de date

Dicționarul de date (Data Dictionary) este un fisier care memorează definițiile datelor si caracteristicile lor ca: folosirea, reprezentarea fizică, proprietatea (cine este responsabil pentru întreținerea lor), autorizarea si securitatea. Prin faptul că reprezintă un inventar a datelor conținute într-o bază de date, dicționarul de date este un important instrument de management organizațional. În realizarea acestor dicționare de date se folosesc metadatele. Metadatele reprezintă date despre date (nume, conținut, semnificație, proprietar etc.).

O bază de date este compusă dintr-o mulțime de atribute (câmpuri, coloane) si are asociată o mulțime de date (linii, rânduri, înregistrări, articole). O înregistrare (record) reprezintă o asociere a valorilor pentru fiecare câmp (field) al bazei de date.

Cele trei nivele de organizare a datelor într-o bază de date sunt logic, virtual si fizic (fig. 3).

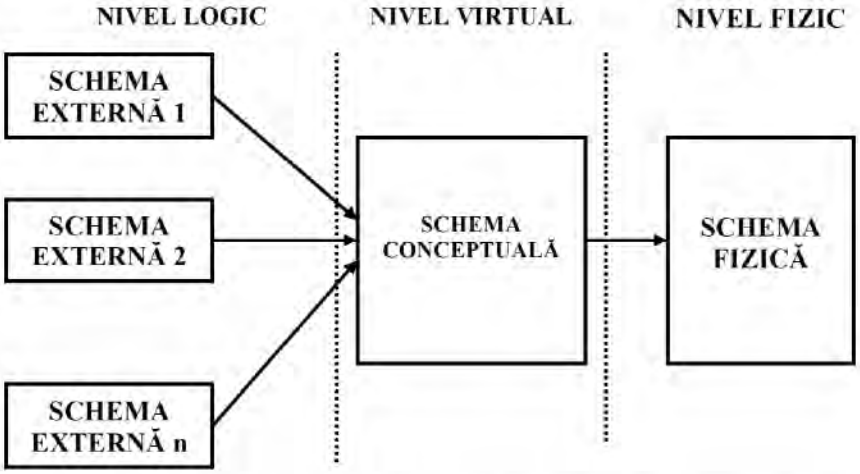


Fig. 3 Niveluri de organizare a datelor într-o bază de date

Nivelul logic sau extern (nivelul programatorului de aplicație) califică o structură de date ce are o realitate în planul semnificației sau utilizării, dar nu si în implementarea fizică; califică forma în care fiecare utilizator vede structurarea datelor, în funcție de aplicația pe care o foloseste sau în funcție de resursele de date pe care administratorul bazei de date i le pune la dispoziție. Nivelul virtual sau conceptual (nivelul administratorului bazei de date) se referă la definirea structurii datelor din baza de date astfel încât aceasta să îndeplinească cerințele tuturor utilizatorilor, în condiții de redundanță minimă si controlată a acesteia. Nivelul fizic (nivelul inginerului de sistem) priveste modul de stocare si de structurare a datelor pe suportul fizic de memorare a datelor (volum magnetic, cilindru, pistă, sector, bloc, octet si bit). Structura virtuală reprezintă schema bazei de date, iar structura logică este denumită subschema bazei de date (concepția CODASYL). Astfel, se poate concluziona că SGBD (DBMS) asigură legătura dintre nivelul conceptual (virtual) si nivelul fizic.

O înregistrare virtuală se poate prezenta sub forma a una sau mai multe înregistrări fizice si poate participa la construirea unei sau mai multor înregistrări logice.

Într-o bază de date ideală datele sunt definite o singură dată si folosite ori de câte ori este necesar.

În funcție de locul în care sunt memorate colecțiile de date ce formează baza de date, se deosebesc:

- *baze de date centralizate*, CDB (Centralized DataBases), în situația în care toate colecțiile care formează baza de date sunt stocate pe un singur calculator;
- *baze de date distribuite*, DDB (Distributed DataBases), în situația în care colecțiile care formează baza de date sunt răspândite în nodurile unei rețele de calculatoare si de comunicații. După orientare, bazele de date pot fi generalizate si specializate.

În cadrul DDBMS, accesarea bazelor de date distribuite, DDB se realizează, în principal, prin intermediul limbajului structurat de interogare, SQL (Structured Querry Language) si al arhitecturii Client/Server.

Realizarea unei baze de date se obține prin parcurgerea unor etape.

Conținutul acestor etape este dependent, de regulă, de tipul bazei de date si de domeniul în care este ea folosită. Activitatea de analiză a sistemului economic presupune:

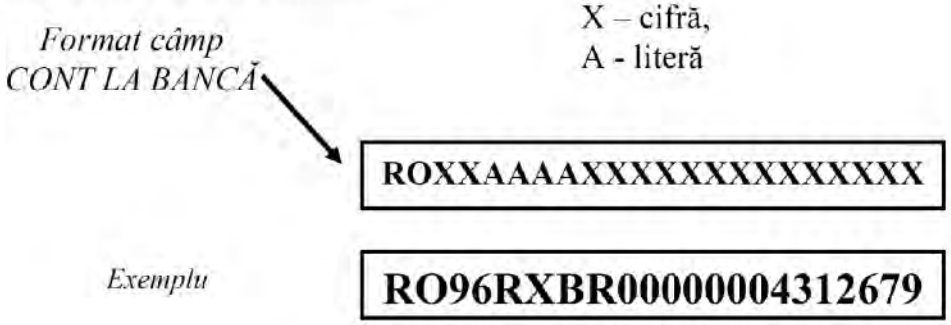
a. analiza componentelor sistemului si a legăturilor dintre acestea sau analiza structurală în urma căreia se defineste modelul structural sau static al sistemului economic;
b. analiza stărilor sistemului si a tranzacțiilor posibile între aceste stări în raport cu anumite evenimente. În urma acestei analize rezultă modelul dinamic sau temporal;
c. analiza cerințelor informaționale, în urma căreia se defineste modelul funcțional al sistemului economic;
d. integrarea modelelor sistemului economic (structural, dinamic si funcțional) în scopul corelării si completării lor.

Se face mențiunea că analiza funcțională a sistemului are ca scop determinarea transformărilor de date care se produc în cadrul sistemului în scopul satisfacerii cerințelor informaționale specifice acestui sistem. Transformările de date se vor prezenta sub forma unei diagrame de flux a prelucrărilor (modelul funcțional), în care nodurile reflectă procesele de prelucrare informațională si arcele fluxurile informaționale ale datelor în baza de date.

La proiectarea unei baze de date, procesul de normalizare ajută proiectantul bazei de date să creeze o structură a bazei de date care poate economisi spațiul de memorare a datelor si poate conduce la cresterea eficienței prelucrării datelor. Scopul normalizării este de a minimiza redundanța datelor.

Erorile de introducere a datelor pot să compromită precizia si validitatea bazei de date. Utilizatorul poate să fie uneori în situația de a nu constientiza faptul că el introduce date incorecte. Datele incorecte pot proveni, în primul rând, de la apăsarea gresită a unor taste. Cele mai multe SGBD-uri pot preveni, dar nu pot elimina introducerile incorecte de date. Informația furnizată de rutinele de prelucrare si de rapoarte este la fel de precisă si corectă în măsura în care aceste caracteristici sunt prezente si la datele stocate în bazele de date.

Formatul de câmp menține consistența datelor prin asigurarea unei structuri de introducere a datelor, asa cum se arată în figura următoare:



Formatul de câmp pentru *CONT LA BANCA*

Proiectantul bazei de date poate să preîntâmpine introducerea datelor incorecte prin:

- specificarea unui anumit format de câmp destinat pentru introducerea datelor (formatul de câmp este de fapt o imagine a cum trebuie să arate data atunci când această dată este introdusă în baza de date);
- utilizarea regulilor de validare a câmpurilor (acele specificații prin care se filtrează datele introduse într-un anumit câmp), a casetelor cu liste sau a formatelor predefinite.

3. Baze de date relaționale

Termenul de bază de date relațională (BDR) a fost introdus de E.F.Codd de la firma IBM în anul 1969. Modelul relațional este fundamentat pe reguli, structuri si operații. Regulile stabilesc modul de manipulare a datelor, structurile sunt obiecte definite ce conțin date si care sunt accesibile utilizatorului, iar operațiile reprezintă acțiuni prin care sunt manipulate datele sau obiectele schemei bazei de date. E.F.Codd a formulat în anul 1985 cele 13 reguli de bază care definesc o bază de date relațională si care sunt prezentate în tabelul următor:

Str. Italiană nr. 28, sect. 2, București - România
weh@spiruharet.ro weh.spiruharet.ro

Nr. regulii	Conținutul regulii	Comentarii
R0	SGBD gestionează datele la nivel de relație (exclusiv pe baza caracteristicilor relaționale).	Conceptul de bază este relația.
R1	Toate datele din baza de date relațională se reprezintă explicit sub forma unor valori într-un tabel (regula reprezentării logice a datelor).	Catalogul conține denumiri de tabele, coloane, domenii, restricții de integritate etc.
R2	Datele individuale dintr-un tabel sunt accesate prin specificarea numelui tabelului, a valorii cheii primare si a coloanei (regula garantării accesului la date)	Datele sunt accesibile prin numele tabelului, a liniei si a coloanei.
R3	Valorile NULL (inexistența datelor) sunt tipuri de date acceptate (regula referitoare la valorile NULL).	Valoarea NULL semnifică „nimic". A nu se confunda cu zero (0).
R4	Baza de date relațională reprezintă descrierea bazei de date în format logic simplificat sub formă de tabele (regula metadatelor).	Metadatele sunt date despre date. Regula nu face diferențieri între tratarea datelor si a metadatelor.
R5	Modelul relațional permite implementarea mai multor limbaje (regula de permisiune a limbajelor multiple).	SQL este limbajul de bază pentru realizarea interogărilor asupra bazei de date.
R6	Dacă vederea curentă reprezintă un tabel, toate vederile (view-urile) sunt actualizabile (regula actualizării vederilor).	Toate tabelele virtuale ce teoretic sunt posibil de actualizat, trebuie să fie în mod practic actualizate.
R7	În operațiile de schimbare a conținutului bazelor de date, se lucrează la un moment dat pe toată relația (regula referitoare la actualizări, inserări si stergeri în baza de date).	Modelul relațional abordează relațiile de bază si pe cele derivate ca pe un singur operand destinat operațiilor de actualizare (update), inserare (insert) si stergere (delete) efectuate asupra datelor.
R8	Structura logică a bazei de date se prezintă complet separată de structura fizică a bazei de date (regula referitoare la independența fizică a datelor).	Programele de aplicație nu trebuie să fie influențate de schimbările survenite în modul de reprezentare a datelor sau în metodele de acces.
R9	Atunci când bazei de date i se aduc modificări neconforme, datele se conservă (regula referitoare la independența logică a datelor).	Programele de aplicație nu trebuie să fie influențate de schimbările efectuate asupra relațiilor bazelor de date.
R10	Restricțiile de integritate sunt definte în limbajul folosit de sistem (regula referitoare la restricțiile de integritate).	Restricțiile de integritate sunt create în limbajul SQL si se stochează în dicționarul bazei de date si nu în aplicațiile individuale.

R11	Accesarea datelor pe server de către client se produce în mod continuu (regula referitoare la distribuirea geografică a datelor).	Este presupusă producerea unui proces de copiere a datelor dintr-o bază de date localizată la distanță.
R12	Regulile si restricțiile de integritate nu pot fi evitate de nici un limbaj de acces la date (regula referitoare la prelucrarea datelor la nivel de bază).	Nu trebuie folosit un limbaj de nivel scăzut orientat pe prelucrarea pe tupluri.

Trebuie precizat faptul că nici un SGBD actual nu respectă în totalitate cele 13 reguli ale lui Codd. Asa cum s-a arătat, o bază de date relațională reprezintă o colecție de relații (tabele în accepțiunea uzuală, memorate fizic în fisiere). Coloanele tabelului se numesc atribute, iar liniile se numesc tupluri.

Baza de date relațională (RDB) este compusă dintr-o mulțime de domenii si o mulțime de relații peste care se aplică o mulțime de asocieri. Domeniul este definit ca mulțimea obiectelor de acelasi tip. Relația este o mulțime rezultată ca urmare a agregării (corespondenței) a două sau mai multe mulțimi. O relație în accepțiunea bazelor de date pe domeniile Di constă dintr-un cap de tabel si un corp de tabel. Asocierea se realizează pe bază de atribute (din capul de tabel).

Un astfel de exemplu este tabelul (relația) referitor la MATERIALE:

Cod_material	Denumire_material	Cantitate	Pret_unitar
01212	Tablă	1200	180000
03214	Cornier	400	420000
04301	Cherestea	850	210000

Fiecare linie descrie un anumit material. Coloanele conțin etichete ce reprezintă nume ale atributelor (Cod_material, Denumire_material, Cantitate, Pret_unitar).

Domeniul ce reprezintă codurile materialelor este:

D1: {“01212”,”03214”,”04301”}.

iar domeniul pentru tipurile de materiale (delimitate prin denumire_material) este:

D2: {“TABLA”,”CORNIER”,”CHERESTEA”}.

Domeniul pentru cantitate este:

D3: {“1200”,”400”,”850”}.

Domeniul prețurilor unitare, în acest caz, este:

D4: {pret_unitar pret_unitarÎ[180000,420000]}.

Mulțimea tuplurilor este definită prin produsul cartezian al domeniilor D1 X D2 X …X Dn. Exemplu de tuplu: <”01212”,”TABLA”,”1200”,”180000”>.

Relația L se defineste prin tupluri corespunzătoare din tabel:

L: {<"01212","TABLA","1200","180000">, <"03214","CORNIER","400","420000">}.

Într-o relație, este necesar ca tuplurile să fie distincte (nu se permit valori duplicate). Ca urmare, se observă că relația este reprezentată prin tabelul bidimensional în care coloanele sunt domenii iar liniile sunt tupluri. Numărul tuplurilor unei relații este cardinalul relației. Numărul valorilor unui tuplu este gradul relației. Schema unei relații este formată din numele relației si lista atributelor (pentru fiecare atribut este necesară specificarea domeniului asociat).

Modelul relațional este format din două mulțimi de operatori pe relații: algebra relațională si calculul relațional.

E.F.Codd a definit algebra relațională ca o colecție de operații pe relații, astfel încât o anumită operație dispune de operanzi de tipul relație si are ca rezultat tot o relație. Tipurile de operații acceptate de algebra relațională sunt operații de bază (reuniunea, diferența, proiecția, produsul cartezian s.a.), operații derivate (intersecția si diviziunea) si operații suplimentare (selecția, splitarea unei relații, complementarea unei relații, închiderea tranzitivă, joncțiunea etc.). Algebra relațională permite derivarea procedurală a relațiilor.

Se prezintă ca exemplu reuniunea. Reuniunea (notată cu OR, È, APPEND sau UNION) este o operație definită pe relațiile Rel1 si Rel2 (cu aceeasi schemă), ce generează o nouă relație Rel3 (cu schema identică cu a relațiilor Rel1 si Rel2) care este formată din reuniunea tuplurilor relațiilor Rel1 si Rel2 din figura de mai jos.

Calculul relațional conține mulțimea operatorilor din modelul relațional si este o adaptare a calculului cu predicate (o relație este identificată cu un predicat) pentru domeniul BDR. Calculul relațional asigură definirea neprocedurală, declarativă a relațiilor. Relațiile sunt precizate prin proprietățile tuplurilor. Inițial, în BDR, variabilele definite asupra relațiilor aveau valori care reprezentau tupluri de relație (variabile tuplu), obținându-se calculul relațional orientat pe tuplu. Când variabilele operează asupra domeniilor – asa cum se petrec lucrurile în prezent – ele sunt variabile domeniu si determină calculul relațional orientat pe domeniu. Regulile de integritate sunt aserțiuni pe care datele ce formează baza de date trebuie să le satisfacă si sunt în număr de trei: unicitatea cheii (cheia primară trebuie să fie unică si minimală), integritatea entității (atributele cheii primare trebuie să fie diferite de null) si integritatea referirii (o cheie externă trebuie să fie null în întregime sau să corespundă unei valori a cheii primare asociate). Constrângerile structurale sunt de trei tipuri: de cheie, de referință si de entitate.

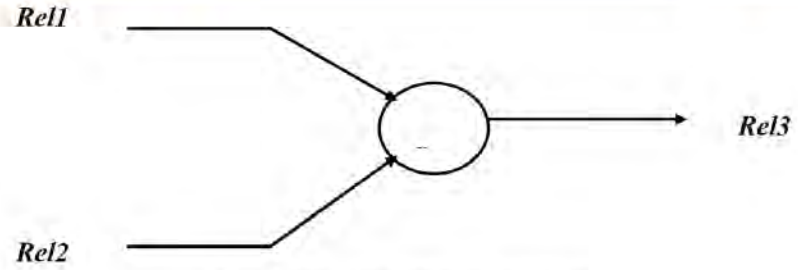


Diagrama operației reuniune

Cheia unei relații reprezintă o mulțime minimală de atribute ale căror valori identifică în mod unic un tuplu într-o relație. Diferitele chei posibile se numesc cheicandidat.

Cheia candidat aleasă pentru a identifica efectiv tupluri se numeste cheie primară.

Conceptele folosite pentru descrierea formală, uzuală si fizică a elementelor de bază ale organizării datelor în baze de date relaționale sunt prezentate în tabelul 2.

Tabelul 2

Formal	**Uzual**	**Fizic**
Relație	tablou	Fisier
Tuplu	linie	Inregistrare
Atribut	coloană	Câmp
Domeniu	tip de dată	tip de dată

Definirea proprietăților structurale ale relațiilor se realizează prin tehnica normalizării. Se afirmă că o relație se găseste într-o formă normală particulară dacă îndeplineste un număr specificat de restricții. Normalizarea se obține printr-un număr de pasi succesivi, în cadrul unui proces reversibil, până la realizarea formei dorite. Forma normală a unei relații este necesară deoarece formele normale nu produc anomalii în actualizarea datelor unei baze de date relaționale. Tipurile de restricții folosite la formele normale ale relațiilor sunt restricțiile asupra valorilor atributelor, restricțiile referitoare la dependența atributelor secundare de chei, restricțiile cu privire la dependența atributelor principale de toate atributele. Se folosesc următoarele forme normale:

- 1NF (forma normală 1). O relație se găseste în 1NF dacă domeniile pe care sunt definite atributele relației sunt formate numai din valori elementare. Un tuplu nu trebuie să conțină atribute sau grupuri de atribute repetitive (nu se admit duplicate).
- 2NF (forma normală 2). O relație se află în 2NF dacă este în 1NF si oricare dintre atributele non-cheie este dependent funcțional complet de cheia primară a relației.

- 3NF (forma normală 3). O relație se găseste în 3NF dacă este în 2NF si atributele non-cheie nu sunt dependente tranzitiv de cheia primară a relației.
- BCNF (forma normală Boyce-Codd). O relație este în BCNF dacă dependențele funcționale netriviale ce pot apare în cadrul relației conțin, în partea stângă, ca determinant, o cheie-candidat.
- 4NF (forma normală 4). O relație se găseste în 4NF dacă în cadrul acesteia nu se manifestă mai mult decât o dependență multivaloare.
- 5NF (forma normală 5). O relație L se găseste în 5NF dacă fiecare dependență joncțiune este implicată printr-o cheie-candidat a lui L.

Bazele de date relaționale conțin structuri de date simple si intuitive. Ele prezintă avantaje legate de existența unui ansamblu integrat de utilitare bazat pe un limbaj evoluat de programare (generatoare de meniuri, generatoare de forme, generatoare de aplicații, generatoare de etichete), de existența unor limbaje speciale de definire si de manipulare a datelor, precum si de independența completă în descrierea logică a datelor (în termeni de relații) si în descrierea fizică a datelor (în termen de fisiere). Dintre dezavantajele bazelor de date relaționale, se menționează imposibilitatea utilizării obiectelor complexe si dinamice, a administrării datelor distribuite si a cunostințelor.

4. Baze de date orientate obiect

Bazele de date orientate pe obiecte, OODB (Object-Oriented DataBase) si SGBD asociate asigură crearea de obiecte complexe formate din componente simple, fiecare prezentând atribute si comportament propriu. Aceste sisteme se mai numesc si sisteme de obiecte, cu originea în limbajele de programare orientate pe obiecte, OOP. Prin aceste tipuri de baze de date se ridică nivelul de abstractizare. Se face mențiunea că între partea de limbaje de programare si partea de baze de date există multe elemente comune; cu toate acestea, aceste părți sunt diferite:

- un program pe calculator este gândit să rezolve o anumită problemă;
- o bază de date este realizată pentru a rezolva o multitudine de probleme, inclusiv cu elemente de pornire nedeterministe.

Pentru un program, obiectele complexe simplifică problema, în timp ce în situația bazelor de date orientate pe obiecte, de regulă, problemele se complică. Ca urmare, se cuvine să se judece în mod nuanțat atunci când se încearcă reliefarea avantajelor utilizării OOP si OODB.

Sistemul de gestiune al bazelor de date orientate pe obiect (SGBD-OO sau OODBMS) are ca principale obiective:

1. Modelarea superioară a datelor, ceea ce semnifică dezvoltarea de noi aplicații; extinderea posibilităților de generalizare si agregare a relațiilor; evoluția către multimedia si hipermedia (sunet, imagine, texte).
2. Capacitatea de deducție superioară (ierarhie de clase, mostenire);
3. Îmbunătățirea interfeței cu utilizatorul:
4. Capacitatea de tratare dinamică, concomitent cu integrarea descrierii structurale si comportamentale.

Modelul de dată-obiect asigură reprezentarea unor structuri de date complexe si a unor ierarhii model, creind posibilitatea de definire a unor tipuri de date care combină atât structura de date cât si definirea procedurii. Un model de date orientat pe obiecte are la bază noțiunea de entitate conceptuală si defineste un obiect ca o colecție de proprietăți care descriu entitatea. O comparație între noțiunile clasice si cele asociate bazelor de date orientate pe obiecte este prezentată, după Date, în tabelul următor:

Nr.crt	Noțiunea specifică obiectelor	Noțiunea clasică de comparație
1	Obiect nemutabil (care nu se poate muta)	Valoare
2	Obiect mutabil (care se poate muta)	Variabilă
3	Clasa de obiecte	Tip
4	Metoda	Operator
5	Mesaj	Invocarea de operator

Obiectul reprezintă conceptual o unitate identificabilă cu conținut propriu, care se deosebeste de ceea ce o înconjoară. Fiecare obiect dispune de un identificator unic, denumit ID al obiectului, OID (Object ID). Două obiecte cu OID diferiți sunt diferite, chiar dacă sunt identice sub toate aspectele transparente utilizatorului.

Desi tentația inițială este de a considera obiecte doar unitățile ce se pot muta, prin obiecte se desemnează atât unitățile fixe cât si cele mutabile. Fiecare obiect posedă un tip care semnifică o clasă de obiecte. Instanța unui obiect reprezintă un obiect individual. Obiectele sunt încapsulate. Structura obiectului si modul de acțiune al metodelor sale nu pot fi accesate si actualizate direct de un agent extern, dar pot fi modificate indirect prin intermediul mesajelor. Această caracteristică ascunsă a obiectului se numeste încapsulare. Încapsularea presupune independența fizică de date. Astfel, prin încapsulare, reprezentarea internă a unui obiect poate să fie modificată fără a fi nevoie ca aplicațiile care utilizează obiectul să fie rescrise.

Starea unui obiect este exprimată prin valorile atributelor sale. Colecția de atribute trebuie aleasă astfel încât să descrie entitatea, adică să cuprindă atribute pe care utilizatorul trebuie să le

cunoască. Metoda reprezintă un program care manipulează obiectul sau indică starea sa. Ea este asociată unei clase, iar specificarea metodei se numeste „semnătură".

Comportamentul unui obiect reprezintă un set de metode sau operații care acționează asupra atributelor sale. Obiectele se clasifică în:

- obiecte elementare ca: întreg, boolean, sir de caractere;
- obiecte compuse ca: nume, adresă;
- obiecte complexe ca: autoturism, angajat.

Un obiect înglobează următoarele elemente:

a. structura de date;
b. specificarea operațiilor;
c. implementarea operațiilor.

Structura unui obiect si operațiile (metodele) permise pentru acel obiect sunt definite împreună.

Metodele si atributele nu sunt vizibile din „exteriorul" obiectului. Un obiect răspunde la mesaje care reprezintă cereri adresate obiectului pentru a returna o valoare sau pentru a-si schimba starea.

Un obiect este divizat în interfață publică si în memorie privată. Interfața publică este compusă din definițiile interfețelor (corespunzătoare semnăturilor specificației). Interfața publică nu face parte din obiectul corespunzător. Această interfață publică este inclusă în obiectul de definire a clasei, CDO (Class-Defining Object). CDO este obiectul ce defineste clasa pentru care obiectul considerat reprezintă o instanță (este similar unui descriptor). Memoria privată este compusă din variabile de instanță (atribute sau membri) ale căror valori reprezintă starea internă a obiectului. Deoarece sistemele baze de date orientate pe obiecte reale nu sunt „pure" (cu variabile instanță care sunt netransparente utilizatorului), variabilele de instanță apar ca transparente utilizatorului. Se deosebesc variabile de instanță publice (transparente utilizatorului) si variabile de instanță private (cele netransparente utilizatorului).

Persistența este o proprietate a datelor sau a obiectelor care presupune existența lor pe o durată mai mare în comparație cu aceea a procesului care le-a generat. Persistența reprezintă proprietatea prin care starea bazei de date asigură execuția unui proces pentru a fi refolosit ulterior în alt proces. Deoarece face parte integrantă din obiect, codul aferent metodelor este stocat în baza de date (ca si starea obiectului).

Tipuri si clase

Obiectele care prezintă acelasi fel de atribute si acelasi comportament fac parte din acelasi tip sau clasă. În raport cu această caracteristică există două categorii de sisteme orientate pe obiecte:

a. sisteme care admit ca noțiune de bază clasa, cum ar fi: VISION, ORION, GBASSE;
b. sisteme care admit ca noțiune de bază tipul, cum sunt: C++, Simula, O2.

Într-un sistem orientat pe obiecte, tipul sintetizează elementele comune ale unui set de obiecte cu aceleasi caracteristici. Acest sistem are ca si componente, interfața si implementarea. Interfața este partea vizibilă pentru utilizator si constă într-o listă de operații. Implementarea presupune descrierea structurii interne a datelor obiectului si realizarea procedurilor de implementare a operațiilor interfeței.

Un tip este construit recursiv, începând cu tipurile de bază: caracter, întreg, real, sir de caractere, boolean. Constructorii de tipuri sunt: tuplul, lista, setul si clasa. O listă este o colecție ordonată de obiecte ale aceleiasi clase sau de valori ale aceluiasi tip. Elementele unei liste sunt accesibile direct prin rangul lor.

Operațiile permise asupra listelor sunt: afectarea (=), comparația (= =), concatenarea (+), accesul direct ([i]), apartenența (in), sublista ([i ; j]), numărarea (count ( )), înlocuirea, stergerea (list ( )), inserarea ([:i]+=), iterația ({ }).

Noțiunea de clasă, desi are aceeasi specificație cu cea de tip, este asociată cu faza de execuție. Ea presupune generarea de obiecte prin operația „new" aplicată unei clase si stocarea setului de obiecte care reprezintă instanțele clasei. Descrierea clasei serveste ca sablon pentru crearea obiectele noi.

O clasă este un tip abstract de date care defineste atât structura obiectelor din clasa respectivă, cât si mulțimea metodelor existente pentru aceste obiecte. Astfel, obiectele din aceeasi clasă prezintă aceleasi atribute si aceleasi metode si răspund la acelasi mesaj.

Mostenirea

Într-o bază de date orientată pe obiecte, clasele sunt aranjate într-o ierarhie în care fiecare clasă mosteneste toate atributele si metodele superclasei din care face parte. Mostenirea conduce la reutilizarea codului. Mostenirea reprezintă mecanismul de realizare a definirii unei clase în care derivă variabilele de instanță si metodele din altă definire de clasă. Când o clasă mosteneste, ea este considerată ca subclasă. Conceptele de subclasă si superclasă sunt analoge conceptelor de generalizare si specializare.

Obiectele, clasele si mostenirea formează baza modelului de date orientat pe obiecte si presupune următoarele aspecte:

- obiectele sunt entități de bază care înglobează structuri de date si operații;
- fiecare obiect are asociat un identificator care este unic si asigurat de sistem;
- clasele descriu tipuri generice de obiecte, toate obiectele sunt membrii unei clase;
- clasele sunt înrudite prin mostenire;
- definirea unei clase este mecanismul de specificare a schemei bazei de date;

- definirea unei clase poate include variabile de instanță, având tipuri de date definite de sistem sau de utilizator;
- schema bazei de date poate fi extinsă dinamic prin definirea de noi clase.

Operațiile modelului de date orientat pe obiecte

Operațiile se pot grupa în modul următor:

a. obiectele comunică între ele prin mesaje;
b. un mesaj poate fi trimis instanțelor mai multor clase;
c. metodele pot fi definite, sterse sau modificate;
d. clasele pot fi definite si actualizate prin operații de creare, stergere si modificare;
e. instanța unei clase poate fi actualizată prin metode care modifică valorile variabilelor propriei instanțe, aceasta modificând starea internă a obiectului.

Într-o serie de implementări, definirile de clasă sunt ele însele obiecte, numite obiecte de clasă. Obiectele clasă sunt instanțe ale unei clase generice sau ale unei metaclase. Operațiile de creare, modificare si stergere ale definirilor de clasă pot fi si implementate ca mesaje. În modelul de date orientat pe obiecte, regulile de integritate reprezintă o consecință a structurii modelului si a următoarelor operații:

- toate obiectele trebuie să respecte protocolul specificat de definirile lor de clasă;
- obiectele sunt încapsulate, acest lucru presupunând accesul limitat la obiecte prin folosirea protocolului de mesaje definit pentru clasa obiectului;
- identificatorul obiectului asigură integritatea referirii la un obiect. Ca atare, un obiect nu există fără să aibă asignat un identificator. Dacă un obiect este sters sau mutat, identificatorul său trebuie si el sters sau mutat.

O schemă completă a unei baze de date orientată pe obiecte poate consta din una sau mai multe ierarhii de clasă, împreună cu relațiile structurale.

Modificarea schemei presupune:

1. Definirea unei taxonomii si a unui model al schimbărilor. Taxonomia defineste un set de schimbări semnificative ale schemei, iar modelul furnizează o bază pentru specificarea semanticilor schimbărilor schemei;
2. Implementarea schimbărilor schemei. Aceste schimbări pot fi:
    a. schimbări referitoare la modul de definire al unei clase. Acestea includ schimbările atributelor si metodelor definite pentru o clasă, cum ar fi: schimbarea numelui sau domeniul unui atribut, adăugarea, stergerea unui atribut sau a unei metode;

b. schimbări referitoare la structura ierarhiei de clase care includ adăugarea sau stergerea unei clase si schimbarea relațiilor superclasă/subclasă dintre o pereche de clase.

Proiectarea bazei de date orientată pe obiecte

Pentru proiectarea unei baze de date orientată pe obiecte se foloseste tehnica topdown care constă în identificarea componentelor după care se stabilesc corelațiile între ele si se rafinează succesiv în „cascadă" componentele sale. Se poate utiliza si metoda bottom-up prin care mai întâi se identifică componentele funcționale pe baza cărora se vor identifica, în colecțiile existente, obiectele, care pot fi reutilizate pentru noul proiect. Componentele care nu există vor fi create ca subclase ale unor clase existente. O dată creată o ierarhie potrivită, se testează componentele specifice.

Sistemul de gestiune al bazelor de date orientate pe obiecte (SGBD-OO sau OODBMS) conține structuri si reguli orientate către lucrul cu obiecte, incluzând:

- un sistem de date abstracte pentru construirea de noi tipuri de date;
- un constructor de tip sir;
- un constructor de tip secvență;
- un constructor de tip înregistrare;
- un constructor de tip set;
- funcții;
- un constructor de tip reuniune;
- o compunere recursivă a elementelor anterioare.

În proiectarea SGBD-OO se au în vedere următoarele:

Principiul 1. SGBD-OO utilizează funcții care conțin metode si proceduri ale bazei de date, cu restricția ca acestea să fie cât mai compacte, încapsulate, ermetizate. Încapsularea funcțiilor îl ajută pe programatorul de aplicație să asocieze funcțiile pe care si le creează cu colecțiile utilizate.

Principiul 2. SGBD-OO si în general SGBD–urile din generația a treia vor prelua avantajele SGBD–urilor din generația a doua. În plus, se caută o modalitate de acces la o înregistrare existentă într-o colecție oarecare si aceasta se poate realiza prin utilizarea unui sistem de pointeri către identificatorii de obiecte.

Principiul 3. SGBD-OO trebuie să poată conecta si limbaje din generația a patra. Un SGBD-OO lucrează cu obiecte complexe, obiecte care se obțin prin aplicarea de constructori asupra obiectelor simple.

Identitatea obiectelor. Orice obiect există independent de valorile atributelor sale, ceea ce conduce la două relații posibile:

- identitatea a două obiecte, adică sunt unul si acelasi obiect;
- egalitatea a două obiecte, adică au aceeasi valoare.

Arhitectura SGBD-OO cuprinde trei componente:

1. Gestionarul de obiecte (Object Manager) furnizează interfața dintre procesele externe si SGBD-OO.
2. Server-ul de obiecte (asigură gestiunea tranzacției si gestiunea stocului de obiecte);
3. Stocul rezident de obiecte.

Gestionarul de obiecte asigură implementarea completă a modelului de dateobiecte pentru utilizatorul extern. Acest lucru include posibilitatea de a defini structurile si de a executa operațiile specificate prin model. El primeste cereri de creare de definiri de clase, de modificare a definirilor de clase deja existente, de manipulare a mesajelor generate de un program de aplicație în execuție.

Server-ul de obiecte asigură refacerea, inserția, stergerea si actualizarea obiectelor în stocul rezident de obiecte. Un singur server poate manipula tranzacții transmise de la mai mulți gestionari de obiecte.

Limbajul de definire a datelor este realizat prin mecanismul de transmitere a mesajelor. Limbajul pentru cereri ad-hoc se bazează pe transmitere de mesaj pentru selectarea si regăsirea obiectelor.

Prelucrarea mesajelor. Gestionarul de obiecte asigură interfața dintre procesele externe si SGBD-OO. El primeste mesaje pentru obiecte individuale, realizează legături dinamice si operații de verificare a tipului si expediază cerința externă pentru obiecte, către server-ul de obiecte.

Transmiterea de mesaje si prelucrarea cererii poate fi reprezentată astfel:

- controlul sesiunii (menținerea spațiului local de lucru al utilizatorului extern pentru operații efectuate asupra bazei de date);
- legătura dinamică (selectarea unei metode pentru un mesaj trimis unui obiect în momentul execuției);
- crearea de noi obiecte sau instanțe de clasă trebuie inițiată de gestionarul de obiecte;
- transmiterea cerințelor obiectului si actualizare acestuia;
- transmiterea cererii. Cererile pot fi translatate în planuri de execuție în care selecția si regăsirea obiectelor sunt realizate prin transmiterea de mesaje. Aceasta presupune că protocolul de mesaje al clasei obiectului este definit pentru a permite accesul la variabilele de instanță necesare pentru a selecta obiectul. Obiectele, definirile de clasă si metodele cerute de gestionarul de obiecte sunt regăsite de server-ul de obiecte din stocul rezident de obiecte.

Definirea si modificarea schemei constă din următoarele etape:

Str. Italiană nr. 28, sect. 2, București - România
weh@spiruharet.ro weh.spiruharet.ro

1. Asigurarea accesului la definirile de clasă existente. Definirile tuturor claselor asigurate de SGBD-OO, ca si a claselor create de utilizatorii umani, pot fi stocate permanent în SRO, într-o bibliotecă de clasă sau într-un dicționar de date.
2. Extensibilitatea schemei bazei de date. Aceasta include prelucrarea declarațiilor limbajului de baza de date, specificând crearea, mutarea sau identificarea definirilor de clasă.
3. Redefinirea dinamică a clasei (evoluției schemei).

Gestionarul de obiecte trimite cerințe pentru regăsirea si actualizarea definirilor de clasă, server-ului de obiecte. Gestiunea tranzacțiilor este asigurată de server-ul de obiecte. Gestiunea stocului de obiecte se referă la menținerea nivelului fizic de organizare a bazei de date obiect (ODB) si la asigurarea căilor de acces necesare realizării accesului eficient la stocul de obiecte.

Funcțiile de bază ale stocului de date-obiect se caracterizează ca fiind:

1. Suport pentru rezidență, adică obiectele create si adăugate trebuie reținute si după ce se încheie sesiunea.
2. Suport pentru obiecte mari. SGBD-OO trebuie să poată suporta stocarea si manipularea obiectelor de lungime variabilă si de orice dimensiune.
3. Facilități de arhivare si asigurare de rezerve (dubluri).

Caracteristicile care asigură regăsirea si actualizarea obiectelor stocate pot fi:

a. suport pentru căi de acces care este necesar pentru a asigura regăsirea si actualizarea eficientă a datelor stocate în baze de date mari. Aceasta include indexarea obiectelor pentru a permite regăsirea eficientă a obiectelor individuale, dar si indexarea obiectelor prin valorile variabilelor de instanță, pentru regăsirea subseturilor de obiecte pentru satisfacerea cererilor;
b. tipuri de indecsi specializați pentru obiecte;
c. gruparea obiectelor în acelasi sector de stoc secundar.
d. segmentarea obiectelor stocate.