ANEXA C

TEORIA BAZELOR DE DATE

C1. GENERALITĂȚI. DEFINIȚII

Baza de date (BD, *Database - DB*) constituie o aplicație fundamentală în toate domeniile de activitate, civile sau de apărare: financiar, administrativ, educațional, informațional, de comunicații şi, nu în ultimul rând, cel al calculatoarelor.

Inițial a apărut necesitatea computerizării sistemului de îndosariere. O primă soluție a acestei probleme a constituit-o sistemul bazat pe fişiere, cu mai multe programe de aplicație care oferă diverse servicii utilizatorilor, printre care şi generarea de rapoarte. Deşi foarte folosit, acest sistem se dovedeşte a fi extrem de redundant şi greu de actualizat prin dublarea datelor şi complexitatea relativ mare. În acest context, bazele de date au oferit o modalitate eficientă de tratare distribuită a datelor, într-o resursă comună partajată în care se include şi descrierea acestora în aşa-numitul **catalog de sistem** sau **dicționar de date** sau **meta-date.** BD asigură independența program-date dar şi controlul accesului şi manipulării datelor.

Sistemele de baze de date, ca aplicație software, reprezintă cea mai importantă realizare din domeniul ingineriei programării pe calculator.

Complexitatea şi dimensiunea BD evoluează rapid, determinând reproiectarea algoritmilor de stocare şi acces al fişierelor şi chiar schimbarea unor principii de administrare a acestora.

Simultan este necesară pregătirea de personal specializat în domeniul bazelor de date:

- administratori
- proiectanți
- programatori.

Scopul întregii activități de proiectare, programare şi administrare a BD constă în punerea la dispoziția utilizatorului final a unor BD permanent actualizate, coerente şi uşor accesibile.

Cele patru categorii de persoane implicate în mediul BD necesită o anumită pregătire şi diferite abilități.

Sistemul de gestiune a bazelor de date (SGBD, *Database Management System - DBMS*) administrează şi controlează accesul la acestea.

În funcție de aplicația pe care o deserveşte, BD pot fi centralizate sau distribuite în rețea, iar modalitățile de gestionare diferă de la caz la caz.

Prin **bază de date** se înțelege orice colecție partajată de date, între care există relații logice, cu o descriere a datelor, proiectată pentru a satisface necesitățile informaționale ale unei organizații sau grup de utilizatori.

BD este gândită ca o resursă unică, utilizată simultan de mai mulți utilizatori, în care datele sunt integrate împreună cu o descriere a lor, cu o dublare minimă în scopul reducerii redundanței şi menținerii coerenței lor.

BD include o definiție internă a fiecărui obiect, folosită de programele de aplicație, şi o definiție externă a obiectului oferită utilizatorilor. Acest fapt reprezintă **procesul de abstractizare a datelor** în BD şi asigură transparența pentru utilizatorii finali, interesați doar de anumite vizualizări externe şi nu de modalitățile de stocare şi manipulare a datelor. Totodată este permisă modificarea structurii datelor sau a modalităților de stocare a lor fără a afecta vederile externe. Numai eliminarea anumitor elemente (câmpuri, atribute etc) poate deranja eventualele programe de aplicații care le utiliza la momentul respectiv. Acest neajuns este evitat prin tratarea corespunzătoare a tranzacțiilor în BD.

Pentru realizarea unei BD este necesară identificarea următoarelor elemente specifice:

- **entități**
- **atribute**
- **relații.**

O **entitate** este un obiect distinct inclus în BD (persoane, locuri, firme, documente, concepte etc). De exemplu, angajații unei organizații pot fi incluşi în BD ca o entitate distinctă denumită **personal.**

Un **atribut** este o proprietate care descrie un anumit aspect al unei entități. De exemplu, numele, prenumele, salariul constituie atribute ale entității **personal.**

O **relație** reprezintă o asociație între mai multe entități. De exemplu, entitatea **personal** gestionează entitatea **contracte** pentru o firmă.

BD poate fi modelată grafic sub forma unei diagrame **Entitate-Relație (**ER) în care entitățile sunt reprezentate, prin convenție, sub formă de dreptunghiuri, relațiile ca romburi iar atributele ca ovale, toate legate între ele sub forma unui graf neorientat.

Sistemul de programe software care permite definirea, crearea, administrarea şi accesarea bazei de date este numit **sistem de gestiune a bazei de date (SGBD).**

Dintre funcțiile unui SGBD se remarcă:

- asigurarea securității şi confidențialității accesării BD;
- personalizarea BD simultan cu realizarea independenței program-date;
- menținerea coerenței BD chiar şi în cazul efectuării unor tranzacții concurente.

În prezent, se dezvoltă SGBD multiutilizator pentru BD cu capacități mari de stocare pentru aplicații grafice, video, multimedia în general, cu interfețe grafice de utilizator (GUI – *Graphic User Interface*) atractive.

C2. LIMBAJELE BAZELOR DE DATE

Implementarea şi utilizarea BD includ mai multe aspecte:

- definirea bazei de date folosind un **limbaj de definire a datelor** (DDL – *Data Definition Language*) pentru descrierea BD în **schema bazei de date**;
- manipularea datelor din BD prin intermediul unui **limbaj de manipulare a datelor** (DML – *Data Manipulation Language*)**;**
- controlul accesului la BD, prin politica de securitate implementată, reguli de integritate, controlul concurenței şi al refacerii BD în cazul apariției unor defecțiuni hardware sau software.

DDL reprezintă un limbaj descriptiv utilizat pentru denumirea entităților BD şi a relațiilor logice dintre acestea, specificarea tipurilor şi structurilor de date precum şi a modurilor de vizualizare personalizate**.** Rezultatul compilării instrucțiunilor DDL este un set de tabele care se constituie în **catalogul de sistem.**

DML este folosit pentru efectuarea **operațiilor specifice de manipulare a datelor** din BD:

- **inserarea** de noi date;
- **modificarea** datelor existente;
- **extragerea** de date din BD;
- **ştergerea** unor date din BD.

Limbajele de manipulare a datelor din BD sunt de două tipuri:

- **DML procedurale** care definesc procedurile prin care se efectuează anumite operații;
- **DML neprocedurale** sau **declarative** care specifică numai ce operații trebuie efectuate asupra datelor.

Acea parte a unui limbaj DML utilizată pentru regăsirea datelor în BD se numeşte **limbaj de interogare**.

SQL este un exemplu de limbaj de interogare a BD neprocedural, care constituie limbajul standard pentru SGBD relaționale.

DDL şi DML sunt considerate **sublimbaje de date** care pot fi încorporate într-un limbaj de nivel înalt denumit şi **limbaj gazdă.**

Un alt limbaj de manipulare a datelor din BD este limbajul QBE (*Query by Example*). SQL şi QBE sunt limbaje din a patra generație (4GL – *Fourth Generation Language*) care definesc **ce** trebuie făcut şi nu **cum** se procedează. Limbajele din generația a treia sunt procedurale şi complicate sintactic.

Limbajele 4GL sunt de mai multe tipuri:

- **limbaje de prezentare** (de interogare, generatoare de rapoarte, generatoare grafice etc)
- **limbaje de specialitate** (de exemplu, pentru calcul tabelar)
- **generatoare de aplicații** (care construiesc aplicații folosind datele din BD)
- **limbaje de nivel foarte înalt** (care generează codul-sursă al aplicației).

C3. COMPONENTELE UNUI SGBD

SGBD se aplică într-un anumit mediu care include diferite componente hardware şi software, date, proceduri şi persoane.

Ca elemente **hardware**, se utilizează rețele de calculatoare şi servere dedicate acestor BD.

Componenta **software** cuprinde programele SGBD şi cele de aplicații, scrise într-un limbaj de nivel înalt, din generația a treia (Pascal, C, C++, Turbo C etc) sau a patra (SQL, MySQL şi altele), împreună cu sistemul de operare propriu-zis (OS – *Operating System*) şi componenta sa de rețea (NOS – *Network Operating System*), folosind modelul client-server.

Datele constituie componenta principală a oricărui sistem de BD. Structura BD se numeşte **schema BD.** Aceasta stochează în mai multe fişiere **tabelele** din BD. Pentru tabel se foloseşte şi termenul echivalent de **relație.** În tabel, apar mai multe **înregistrări** sau linii, fiecare cu un anumit număr de **câmpuri** sau coloane în care se introduc valorile **atributelor**.

BD include şi **meta-datele** care descriu datele propriu-zise, de exemplu, tipul datelor (întreg, şir de caractere etc), lungimea câmpului etc. Meta-datele sunt stocate în **catalogul de sistem** care conține:

- denumirile, tipurile şi dimensiunile articolelor din BD;
- denumirile relațiilor sau tabelelor;
- constrângerile de integritate impuse datelor;
- numele utilizatorilor autorizați să acceseze BD;
- indexurile şi structurile de stocare folosite.

Procedurile sunt reprezentate de regulile şi instrucțiunile utilizate pentru implementarea, întreținerea şi accesarea BD:

- pornirea şi oprirea SGBD;
- deschiderea sesiunilor de lucru;
- utilizarea programelor de aplicație pentru accesarea BD;
- tratarea defecțiunilor hardware şi software prin identificarea componentei defecte, eventual depanarea automată a acesteia, şi refacerea BD;
- reorganizarea BD prin modificarea structurii tabelelor şi optimizarea modului de stocare a datelor prin arhivare în capacitatea de stocare secundară.

Persoanele implicate în mediul SGBD se împart în patru categorii:

- proiectanți;
- programatori;
- administratorii de date şi de BD;
- utilizatori.

Proiectanții BD se ocupă fie de proiectarea logică reprezentată de identificarea entităților, atributelor, relațiilor şi constrângerilor din BD, atât la nivel conceptual independent de modalitatea de implementare sau de orice considerent de ordin fizic, cât şi pe baza unui model de date relațional, în rețea, ierarhic sau orientat pe obiect, fie de proiectarea fizică, constând în transpunerea modelului de date în tabele şi stabilirea constrângerilor de integritate, alegerea structurilor de stocare şi a metodelor de acces şi impunerea măsurilor de securitate.

Programatorii realizează aplicații specifice care oferă diverse servicii utilizatorilor finali ai BD (interogare a BD, generare de rapoarte, formulare, de vizualizări grafice sau de aplicații).

Administratorii de date se ocupă de gestionarea datelor (planificare, dezvoltare şi întreținerea resurselor de date).

Administratorul BD sunt răspunzători de realizarea şi întreținerea fizică a acesteia, având o orientare mai tehnică decât administratorul de date.

Utilizatorii reprezintă clienții BD, beneficiarii datelor stocate în BD. Marea majoritate a acestora nu cunosc detaliile SGBD şi folosesc pentru accesarea BD diverse aplicații software cu meniu-uri accesibile. Utilizatorii specializați utilizează limbaje de interogare a BD de nivel înalt şi pot să îşi dezvolte propriile aplicații software.

C4. ARHITECTURA BAZELOR DE DATE

Inițial grupul DBTG (*Data Base Task Group*) a propus o arhitectură a sistemelor de BD cu două nivele: **schema** BD reprezentând nivelul inferior, de implementare şi întreținere, şi **subschema** BD pentru realizarea vederilor utilizatorilor.

Personalizarea acestor vederi în sistemele multiutilizator este posibilă prin introducerea unui al treilea nivel, intermediar, care să separe detaliile de implementare de cele

impuse în vizualizare. Astfel ANSI (*American National Standards Institute*) şi SPARC (*Standards Planning and Requirements Committee*) au propus arhitectura cu trei nivele ANSI/X3/SPARC sau ANSI-SPARC, care include:

- nivelul intern
- nivelul conceptual
- nivelul extern.

Nivelul extern este format din vederile utilizatorilor, fiecare incluzând anumite entități, relații şi atribute, eventual cu reprezentări diferite ale aceloraşi date, cu combinații ale acestora sau cu atribute derivate, pe baza unor scheme externe.

Nivelul conceptual reprezintă o vedere generală a BD şi descrie ce date şi relații sunt stocate în BD, într-o structură logică denumită şi schemă conceptuală. Aceasta nu depinde nici de modul de implementare a BD. nici de cerințele de vizualizare ale utilizatorilor. Schema conceptuală cuprinde toate entitățile, atributele, relațiile şi constrângerile datelor, informațiile semantice despre date, normele de securitate şi regulile de integritate.

Nivelul intern reprezintă implementarea fizică a BD, pe baza unei scheme interne cuprinzând structurile de date şi de organizare a fişierelor. Acest nivel este responsabil de alocarea spațiului de stocare a datelor şi indexurilor, de descrierea şi plasarea înregistrărilor în spațiul alocat, de codarea şi compresia datelor.

Nivelul intern interacționează direct cu nivelul inferior, cel fizic, gestionat de sistemul de operare.

Schema internă este legată de cea conceptuală printr-o interfață de transpunere conceptual/intern care transformă ca format cererea procesului client şi răspunsul procesului server între cele două nivele.

Schemele externe sunt deduse din cea conceptuală prin transpunerea extern/conceptual.

Informațiile incluse la un anumit moment în BD reprezintă **starea** sau **instanța** BD şi indiferent de cum se modifică datele, ele corespund aceleiaşi scheme a BD.

Imunitatea schemelor externe față de modificările efectuate în schema conceptuală reprezintă **independența logică de date** a BD.

Imunitatea schemei conceptuale față de schimbările intervenite în schema internă este denumită **independența fizică de date** a BD**.**

În figura C.1 sunt reprezentate cele trei nivele ale arhitecturii ANSI-SPARC pentru BD cu toate elementele componente.

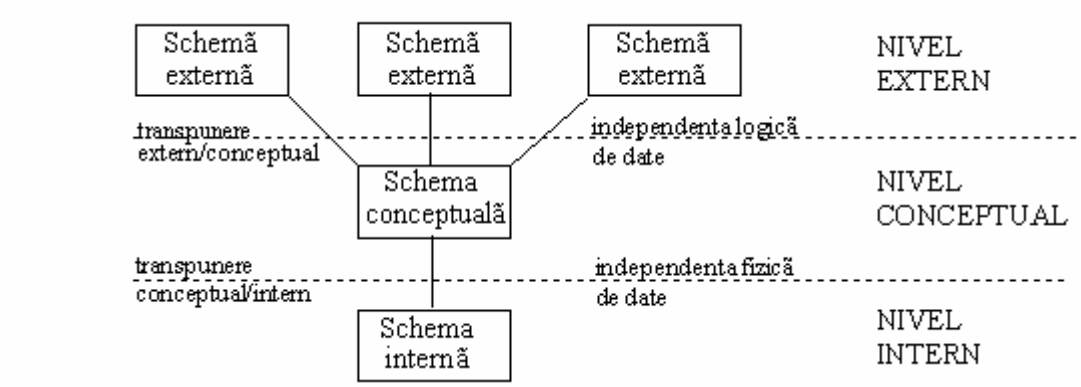

Figura C.1 Arhitectura ANSI-SPARC a BD

C5. MODELAREA BAZELOR DE DATE

Un model de date este alcătuit din totalitatea conceptelor utilizate pentru descrierea datelor din BD, a relațiilor dintre ele şi a constrângerilor impuse lor.

La nivel extern se utilizează modele logice de date bazate pe înregistrări:

- modelul de date relațional, bazat pe conceptul de relații matematice, reprezintă datele din BD şi relațiile dintre ele sub formă de tabele;
- modelul de date în rețea reprezintă datele ca o colecție de înregistrări (noduri) iar relațiile dintre acestea prin direcțiile sau muchiile unui graf.
- modelul de date ierarhic, similar modelului în rețea, foloseşte conceptul de nod-părinte şi permite unui nod din graf să posede numai un singur părinte, realizând o structură arborescentă.

Pe nivelul conceptual se folosesc modele de date bazate pe obiecte, dintre care cele mai utilizate sunt:

- modelul Entitate-Relație (ER), tehnică principală de proiectare conceptuală.
- modelul orientat pe obiecte, care extinde definiția entității prin luarea în considerare şi a comportamentului acesteia, pe lângă atributele care descriu starea ei.

Pe nivelul intern se folosesc modele fizice al BD care descriu modul de stocare a datelor în memoria calculatorului, structurile, ordinea şi căile de accesare ale înregistrărilor.

Diferit de această etapă de modelare logică pe cele trei nivele, procesul de modelare a datelor independent de modul de implementare, de SGBD țintă, programe aplicație, limbaje de programare sau aspecte fizice, reprezintă **modelarea conceptuală a BD.**

C6. MODELUL DE DATE RELAȚIONAL

SGBD relațional (SGBDR) reprezintă elementul dominant în procesul actual de prelucrare a datelor din BD, constituindu-se în cea de a doua generație de SGBD bazate pe modelul de date relațional propus în 1970 de E.F. Codd. Din prima generație fac parte SGBD ierarhice şi în rețea. Limbajul standard al SGBDR este SQL.

În modelul relațional, datele sunt structurate logic simplu, sub formă de **tabele (relații, fişiere).**

Această structurare a BD sub formă de tabele este valabilă numai pe nivelele superioare (extern, conceptual) nu şi pe nivelul intern.

Relația este un tabel cu coloane şi linii.

Fiecare tabel are o denumire, fiind format din mai multe **înregistrări (tuplu-uri, rânduri, linii)** fiecare cu mai multe **atribute (câmpuri, coloane).**

Tuplul este un rând sau o linie din tabel.

Ordinea tuplurilor nu este importantă, deşi poate afecta eficiența accesării tabelului.

Nu trebuie să existe dubluri ale tuplurilor.

Numărul de tupluri dintr-un tabel reprezintă **cardinalitatea relației.**

Atributul este o coloană a unei relații, având o denumire proprie, distinctă.

Ordinea atributelor nu este importantă.

Domeniul este mulțimea valorilor pe care le poate lua un atribut.

Numărul de atribute ale unei relații reprezintă **gradul relației.**

Baza de date relațională constă într-un set de relații normalizat.

Normalizarea este o operație complexă de prelucrare a relațiilor în vederea eliminării dublării datelor din BD.

Prima proprietate de normalizare impune ca fiecare celulă a unui tabel să conțină o singură valoare.

O relație care satisface această condiție este **în prima formă de normalizare.**

Fiecare tuplu este identificat unic prin atributele sale.

Un atribut sau un set de atribute care identifică în mod unic un tuplu dintr-o relație se numeşte **supercheie.**

O supercheie pentru care nici un subset de atribute nu este o supercheie se numeşte **cheie candidat.**

Se pot găsi mai multe chei candidat pentru aceeaşi relație.

Cheia candidat aleasă pentru identificarea unică a tuplurilor dintr-o relație se numeşte **cheie primară.**

Cheile candidat care nu au fost alese ca şi chei primare sunt considerate **chei alternative.**

O cheie formată din mai multe atribute se numeşte **cheie compusă.**

Setul de atribute care constituie o cheie candidat a altei relații se numeşte **cheie străină.**

Modelul sau schema conceptuală a bazei de date se reprezintă printr-o mulțime de relații specificate prin numele lor, urmat de atributele fiecăreia între paranteze, cheia primară fiind subliniată.

De exemplu, o bază de date a unei agenții imobiliare cuprinde date despre clienți, angajați şi proprietățile oferite:

Clienți (Număr_client, Nume, Prenume, Adresa, Telefon, Proprietate)

Personal (Număr_personal, Nume, Prenume, Funcție, Salariu, Data de naştere, Sex, CNP, Adresa, Telefon, Rudă)

Proprietăți (Număr_proprietate, Tip, Număr de camere, Preț, Proprietar, Adresă, Telefon).

APLICAȚIE: Dezvoltați modelul relațional al unei baze de date pentru o agenție imobiliară, cu un singur sediu, care intermediază vânzarea şi cumpărarea de case, apartamente şi spații comerciale într-o localitate. Identificați toate entitățile reprezentative cu atributele dorite. Precizați cheile primare. Eliminați eventualele atribute care ar lua valori multiple. Reprezentați grafic entitățile şi relațiile dintre acestea într-o diagramă ER.

Valoarea unui atribut care nu este cunoscută la un anumit moment sau nu este aplicabilă reprezintă un **nul.**

Nul-ul semnifică absența unei valori dintr-un câmp şi nu valoarea nulă.

Nul-urile elimină riscul introducerii unor date fictive în BD. De exemplu, pentru un client care nu are telefon se introduce un nul în câmpul *Telefon* din tabelul *Clienți.*

Nul-urile nu sunt permise în orice sistem de BD relațional.

Faptul că oricărui atribut i se dau valori dintr-un anumit domeniu reprezintă o **constrângere de domeniu.**

O relație cu o anumită denumire, corespunzătoare unei entități din schema conceptuală a BD, ale cărei tupluri sunt stocate fizic în BD, se numeşte **relație de bază.**

O relație produsă la cererea unui client pe baza relațiilor existente în BD se numeşte **vedere** şi reprezintă o **relație virtuală**.

Vederile sunt dinamice şi orice modificare în relațiile de bază se reflectă imediat în vederile externe.

Prin utilizarea vederilor se asigură securitatea BD şi se personalizează modelul fiecărui utilizator.

Totuşi nu toate vederile pot fi reactualizate.

Similar, dacă o vedere este reactualizată, atunci şi relația de bază care a generat-o trebuie reactualizată. Nu sunt permise reactualizările prin intermediul vederilor care implică relații de bază multiple sau operații de acumulare sau de grupare a atributelor.

Nici un atribut dintr-o cheie primară a unei relații de bază nu poate fi **nul** (lipsă valoare), aceasta reprezentând **regula de integritate a entităților.**

Într-o vedere sau relație virtuală, această regulă nu se aplică.

Valoarea unei chei străine trebuie să fie egală cu valoarea unei chei candidat din relația sa de bază sau un nul. Această condiție reprezintă **regula de integritate referențială.**

Alte reguli impuse de utilizatorii sau administratorii unei BD se numesc **constrângeri de întreprindere.**

Codd a enunțat o regulă fundamentală şi 12 reguli specifice pentru un SGBDR:

0. **Regula fundamentală**

Orice sistem care pretinde sau i se face reclama de a fi un SGBDR trebuie să fie capabil să gestioneze în întregime bazele de date prin capacitățile sale relaționale.

1. **Reprezentarea informațiilor**

La nivelul logic, toate informațiile dintr-o BD relațională sunt reprezentate explicit într-un singur mod – prin valorile din tabele.

2. **Accesul garantat**

Se garantează faptul că orice element – dată dintr-o BDR este accesibil din punct de vedere logic prin apelarea la o combinație de nume de tabel, valoare a cheii primare şi nume de coloană.

3. **Tratarea sistematică a valorilor nul**

Valorile nul sunt acceptate pentru a reprezenta informațiile lipsă şi pe cele care nu pot fi aplicate în mod sistematic, indiferent de tipul de date.

4. **Catalog dinamic on-line, bazat pe modelul relațional**

Descrierea BD este reprezentată la nivel logic în acelaşi mod ca şi datele obişnuite, astfel încât utilizatorii autorizați pot folosi pentru interogarea acesteia acelaşi limbaj relațional aplicat datelor curente.

5. **Sublimbaje de date cuprinzătoare**

Un sistem relațional poate accepta mai multe limbaje şi diverse moduri de utilizare a terminalelor. Totuşi trebuie să existe cel puțin un limbaj ale cărui instrucțiuni să poată exprima următoarele:

- *definirea datelor*
- *definirea vederilor*
- *manipularea datelor*
- *constrângerile de integritate*
- *autorizarea*
- *limitele tranzacțiilor (început, efectuare şi rulare înapoi).*

6. **Reactualizarea vederilor**

Toate vederile care sunt teoretic reactualizabile, pot fi reactualizate de către sistem.

8. **Operații de inserare, reactualizare şi ştergere de nivel înalt**

Capacitatea de tratare a unei relații de bază sau a unei relații derivate (vedere)ca pe un singur operand se aplică nu numai regăsirii datelor în BD, ci şi inserării, reactualizării şi ştergerii acestora.

9. **Independența fizică de date**

Programele de aplicații şi activitățile de la terminale rămân logic intacte ori de câte ori sunt făcute modificări, fie în metodele de stocare, fie în metodele de acces.

10. **Independența logică de date**

Programele de aplicații şi activitățile de la terminale rămân logic intacte ori de câte ori sunt făcute modificări în tabelele de bază cu păstrarea informațiilor sau deteriorarea acestora.

11. **Independența de integritate**

Constrângerile de integritate specifice unei anumite BDR trebuie să poată fi definite în sublimbajul relațional de date şi stocate în catalogul de sistem, nu în programele de aplicații.

12. **Independența de distribuție**

Sublimbajul de manipulare a datelor dintr-un SGBDR trebuie să permită programelor de aplicații şi interogărilor să rămână aceleaşi din punct de vedere logic dacă şi ori de câte ori datele sunt centralizate sau distribuite fizic.

13. **Regula de non-subversiune**

Dacă un sistem relațional are un limbaj de nivel jos (câte-o-înregistrare-o-dată), atunci acel nivel nu poate fi folosit pentru a submina sau a ocoli regulile de integritate şi constrângerile exprimate în limbajul relațional de nivel mai înalt (mai-multe-înregistrări-deodată).

Deşi regulile lui Codd au fost cauza multor controverse, se pare că ele sunt utile pentru analiza caracterului relațional al oricărui SGBD.

Regulile 0 şi 12 sunt esențiale. Un SGBD care nu le îndeplineşte este cu siguranță non-relațional.

C7. ETAPELE CICLULUI DE VIAȚĂ AL UNEI APLICAȚII BAZĂ DE DATE

S-a constatat eficiența planificării activităților necesare creării unui produs BD performant:

Planificarea BD constă în activități administrative care vizează identificarea planurilor de afaceri şi a cerințelor sistemului informațional, evaluarea situației curente şi a

oportunităților IT, şi se bazează pe modelul general de date care include entitățile şi relațiile dintre ele reprezentate într-o diagramă ER în care se specifică şi legăturile cu diferitele zone funcționale ale întreprinderii.

Definirea sistemului constă în stabilirea scopului şi a limitelor aplicației BD, inclusiv domeniile sale de utilizare şi grupurile principale de utilizatori.

Colectarea şi analiza cerințelor se face prin chestionarea persoanelor reprezentative pentru toate domeniile de interes ale întreprinderii, observarea funcționării acesteia şi analiza documentelor utilizate pentru înregistrarea şi redarea informațiilor, observarea tranzacțiilor efectuate etc. Instrumentele CASE (*Computer-Aided Software Engineering*) sunt utile pentru specificarea completă şi coerentă a cerințelor (caracteristicilor) întreprinderii, .

Proiectarea BD începe cu realizarea unor modele de date care conțin entități şi relații de nivel înalt (esențiale pentru reprezentarea funcționalității întreprinderii în BD), după care se trece la o analiză detaliată pentru a identifica şi include în model toate entitățile, atributele şi relațiile de nivel jos.

Alegerea unui SGBD adecvat, care să accepte aplicația de tip BD, se poate face între fazele de proiectare logică şi conceptuală a BD. Se au în vedere costurile de achiziții software şi hardware, cele de instruire a personalului, precum şi performanțele sistemului.

Proiectarea programelor aplicație pentru utilizatori, cu interfețe prietenoase şi eficiente, cu posibilități de accesare a datelor şi efectuare a tranzacțiilor în BD, se face în paralel cu proiectarea propriu-zisă a BD.

Realizarea unui prototip al BD este opțională dar avantajoasă, întrucât pe baza unui model de lucru simplificat, cu costuri reduse, se testează funcționalitatea aplicației BD, se identifică eventualele probleme dar şi caracteristici noi care trebuie implementate.

Implementarea BD şi a aplicațiilor constă în realizarea fizică a proiectelor acestora folosind diverse limbaje. Implementarea BD se face pe baza DDL. Instrucțiunile DDL sunt compilate pentru a crea schema BD. Vederile utilizatorilor sunt definite în această etapă. Programele aplicație sunt implementate cu ajutorul altor limbaje, DML sau de nivel înalt Java, C, C++, Delphi, cu meniuri, formulare, rapoarte şi cu reguli de securitate şi de integritate.

Conversia şi încărcarea datelor constă în transferul în BD a datelor existente, cu eventuala conversie de format, folosind un utilitar de încărcare în BD a fişierelor existente.

Testarea aplicației BD constă în executarea programelor de aplicație cu scopul depistării erorilor de funcționare pe baza unor strategii de testare.

Întreținerea operațională prin monitorizare continuă, remedierea erorilor de funcționare prin reorganizarea BD şi eventual, implementarea unor cerințe noi. Este indicată folosirea noilor aplicații de tip BD în paralel cu cele vechi, dacă acestea există, pentru o anumită perioadă de timp în care să se observe şi să se rezolve disfuncționalitățile noului sistem.

C8. PROIECTAREA BAZELOR DE DATE

Proiectarea BD constă în realizarea unui model bine conceput al acesteia, structurat pe nivelele arhitecturii ANSI-SPARC.

Modelarea are ca scopuri:

- înțelegerea semnificației datelor;
- comunicarea nevoilor informaționale între utilizatori şi proiectanți;
- familiarizarea tuturor părților cu notațiile folosite în BD;
- separarea datelor pe departamente.

Un model de date este optim dacă întruneşte următoarele criterii:

- **validitate structurală** – modelul este consecvent față de modul de definire şi organizare a datelor în cadrul întreprinderii.
- **simplitate** – modelul este uşor de înțeles de către programatori şi administratori.
- **expresivitate** – în cadrul modelului se face o distincție clară între diferitele date, entități, relații şi constrângeri.
- **non-redundanță** – în model orice informație apare exact o singură dată iar informațiile neesențiale nu sunt incluse.
- **posibilitate de partajare** - BD poate fi utilizată de mai mulți utilizatori.
- **extensibilitate –** modelul permite implementarea de noi cerințe**.**
- **integritate –** modelul este consecvent în utilizarea şi administrarea datelor conform cerințelor întreprinderii.

- **reprezentare schematică –** modelul poate fi reprezentat schematic, folosind notații uşor de înțeles.

Proiectarea BD se face pe cele trei nivele ale arhitecturii ANSI-SPARC: logic (extern), conceptual şi intern (fizic).

Prima fază de proiectare se realizează pe nivelul conceptual.

Proiectarea conceptuală constă în crearea unui model conceptual al BD, independent de detaliile de implementare, de programele de aplicație, sistem de operare, elemente hardware etc.

Proiectarea logică reprezintă a doua fază de proiectare a BD şi constă în rafinarea modelului conceptual (normalizare) şi transpunerea acestuia într-un model de date logic, cunoscând tipul de SGBD țintă (relațional, ierarhic, în rețea sau orientat-obiect).

Cele două etape sunt esențiale pentru obținerea unui model al BD complet, care să permită definirea tuturor vederilor utilizatorilor şi menținerea integrității BD.

Prin integrarea în modelul logic a schemelor externe pentru vederile utilizatorilor, respectiv a modelelor de date logice locale, se obține **modelul de date logic global**. În această etapă apar probleme atunci când se utilizează aceiaşi termeni pentru obiecte diferite sau termeni diferiți pentru aceleaşi obiecte.

A treia etapă de proiectare, proiectarea fizică a BD, constă în descrierea modului de implementare a BD, a structurilor de stocare în capacitatea de stocare secundară şi a metodelor de acces la date.

Din modelul de date logic global se obțin tabelele relaționale, se deduc constrângerile impuse datelor şi necesitățile de securitate ale sistemului.

În paralel cu proiectarea BD, se face şi proiectarea aplicațiilor pentru accesarea acestora.

Aceasta constă în proiectarea tranzacțiilor din BD şi a interfeței cu utilizatorul.

Prin tranzacție se înțelege o acțiune sau o serie de acțiuni care accesează sau modifică BD.

Pentru SGBD, o tranzacție este un eveniment din „lumea reală”, prin care BD trece dintr-o stare coerentă în altă stare coerentă. SGBD este responsabil de menținerea coerenței BD chiar şi în cazul în care o tranzacție eşuează, fie din cauza unei defecțiuni tehnice, fie prin decizia utilizatorului de a renunța la acea tranzacție.

Proiectarea tranzacțiilor constă în crearea schemelor externe corespunzătoare tuturor vederilor externe ale utilizatorilor, capabile să efectueze orice tip de tranzacție:

- de regăsire a datelor în BD, pentru afişare sau crearea unui raport.
- de reactualizare a datelor, prin inserarea de noi date, ştergerea de date sau modificarea celor existente.
- **mixte**, de regăsire şi reactualizare.

Deşi din punctul de vedere al utilizatorului, o tranzacție constă într-o singură sarcină, în realitate aceasta poate fi realizată printr-un set complex de operații, fiecare reprezentând o tranzacție efectuată în BD.

Proiectarea interfeței cu utilizatorul se face pe baza unei machete, cu titlu semnificativ, cu terminologii inteligibile, cu o grupare logică şi secvențială a câmpurilor (dintr-un formular, raport etc), cu o folosire sugestivă a culorilor care să diferențieze câmpurile de introducere de date de cele de afişare sau opționale, cu mesaje de eroare care să indice şi domeniul de valori permise într-un câmp, precum şi cu mecanisme simple de mutare a cursorului între câmpuri (tastele TAB, săgeți sau indicatorul mouse-ului).

C9. INSTRUMENTE CASE

Pentru proiectarea BD, ca şi pentru orice proces de proiectare de software, se pot utiliza instrumente de proiectare software asistată de calculator (CASE) care includ:

- un dicționar de date, integrat şi coerent.
- instrumente pentru analiza datelor
- instrumente pentru realizarea modelului logic de date global
- instrumente pentru crearea prototipurilor aplicațiilor de tip BD.

Există trei categorii de instrumente CASE:

- **superioare**, pentru etapele inițiale de planificare, definire sistem, colectarea cerințelor şi proiectare sistemului de BD;
- **inferioare,** pentru etapele de implementare, testare până la întreținerea operațională a BD;
- **integrate,** care susțin toate etapele ciclului de viață a BD.

Instrumentele CASE permit proiectarea eficientă a aplicațiilor software de tip BD.

C10. CONCEPTELE MODELULUI ENTITATE-RELAȚIE

Se va folosi în continuare modelul conceptual Entitate-Relație (ER), dezvoltat inițial de Chen în 1976, care descrie structura BD şi tranzacțiile de regăsire şi reactualizare a datelor. Modelul ER se realizează independent de SGBD şi de platforma hardware pe care se va rula aplicația BD.

Modelul ER include următoarele concepte:

Tip de entitate – obiect (fizic) sau concept (abstract) inclus în BD, cu existență independentă.

Entitate – instanță a unui tip de entitate, unic identificabilă.

Tip de entitate slabă – tip de entitate a cărei existență depinde de alte tipuri de entități.

Tip de entitate tare – tip de entitate a cărei existență nu depinde de alte tipuri de entități.

Tip de relație – asociere între tipuri de entități

Relație – o instanță a unui tip de relație.

Gradul unei relații – numărul de entități implicate în acea relație (unară, binară, ternară etc)

Relație recursivă – relație în care o entitate participă de mai multe ori, cu diferite roluri.

Raportul de cardinalitate – descrie numărul de entități implicate de fiecare parte a unei relații (1:1, unu-la-mulți 1:M, mulți-la-mulți M:N).

Regulă de afaceri – regula pe baza căreia se stabileşte un raport de cardinalitate.

Atributul entității – proprietatea unui tip de entitate**.**

Atributul relației – proprietate a unei relații.

Atribut simplu – atribut cu o singură componentă**.**

Atribut compus – atribut cu mai multe componente, cu existențe independente.

Domeniul atributului – mulțimea valorilor pe care le poate lua un atribut.

Atribut cu o singură valoare – atribut care conține o singură valoare pentru o entitate.

Atribut cu valori multiple – atribut care conține mai multe valori în acelaşi câmp.

Atribut derivat – atribut dedus din valorile unui alt atribut sau set de atribute.

Cheie – articol de date care permite identificarea în mod unic instanța unui tip de entitate.

Cheie candidat – atribut sau set de atribute care identifică în mod unic instanța unui tip de entitate.

Cheie primară – cheia candidat aleasă pentru identificarea instanțelor unei entități.

Cheie alternativă – cheie candidat neutilizată ca şi cheie primară.

Cheie simplă – cheie candidat formată dintr-un singur atribut.

Cheie compusă – cheie candidat formată din mai multe atribute.

Asupra entităților şi relațiilor din modelul ER se pot defini **constrângeri**:

- **de cardinalitate**, exprimate prin raportul de cardinalitate şi impuse de regulile de afaceri ale firmei.
- **de participare**, exprimând dependența existenței unei entități de o altă entitate.

Participarea unei entități într-o relație poate fi:

- totală sau obligatorie;
- parțială sau opțională.

C.11 REPREZENTAREA GRAFICĂ A DIAGRAMEI ENTITATE-RELAȚIE

Modelul ER poate fi reprezentat grafic sub forma unei diagrame ER în care se aplică următoarele convenții (Fig.C.2):

- Entitățile sunt reprezentate ca dreptunghiuri etichetate cu numele acestora, cu chenar simplu pentru entitățile tari şi dublu pentru cele slabe.
- Atributele sunt reprezentate schematic sub forma unor elipse etichetate cu numele acestora şi legate de entitatea pe care o descriu prin segmente de dreaptă, sub forma unor raze. Denumirea cheii primare se subliniază. Conturul elipsei este continuu dacă

reprezintă atribute propriu-zise şi discontinuu pentru atributele derivate. Conturul elipsei este reprezentat printr-o linie dublă în cazul atributelor cu valori multiple.

- Relațiile sunt reprezentate sub formă de romburi, etichetate cu numele relației, cu chenar simplu între entități tari şi cu chenar dublu când în relație este implicată o entitate slabă.
- Rapoartele de cardinalitate se reprezintă ca etichete ale liniilor ce leagă entitățile implicate într-o relație.
- Liniile de legătură între simbolurile grafice ale entităților şi relațiilor sunt simple dacă participarea este parțială şi duble când participarea entității la acea relație este totală.
- Liniile de legătură pot fi etichetate şi cu valorile minimă şi maximă ale numărului de entități implicate într-o relație, scrise între paranteze.

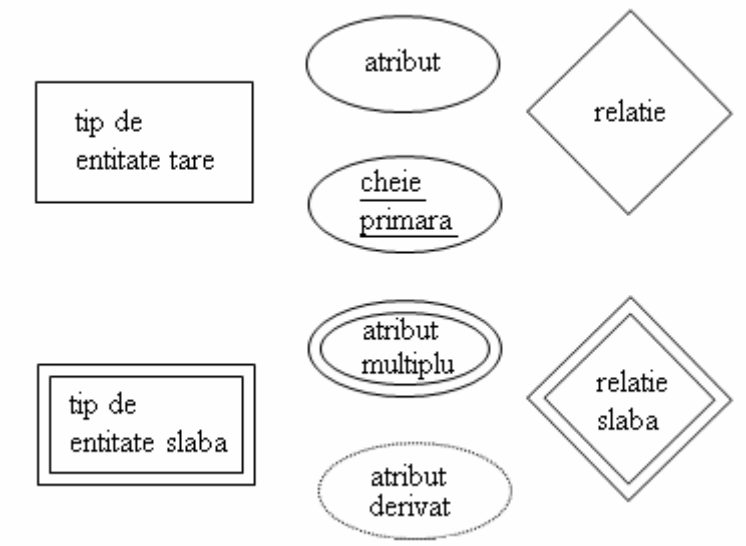

Fig. C.2 Reprezentarea grafică a elementelor modelului ER

C12. CAPCANE DE CONECTARE

În proiectarea modelului de date conceptual, pot să apară anumite ambiguități de interpretare a relațiilor care să conducă la imposibilitatea soluționării unor interogări asupra BD. Aceste probleme ale modelului ER se numesc **capcane de conectare** şi sunt de două tipuri:

Capcane în „evantai" – apar atunci când aceeaşi entitate este implicată în mai multe relații de tip 1:M şi căile dintre entități devin ambigue. De exemplu, din diagrama ER reprezentată în figura C.3 nu putem deduce exact ce proprietăți sunt administrate de un anumit membru de personal. Prin modificarea ordinii de reprezentare a entităților implicate în aceste relații se elimină această ambiguitate (Fig.C.4).

Capcane de întrerupere - sunt cauzate de absența reprezentării unor relații în modelul ER. De exemplu, în modelul din figura C.4 lipseşte relația dintre entitățile Departament şi Proprietăți necesară identificării departamentului care se ocupă de o anumită proprietate (Figura C.5).

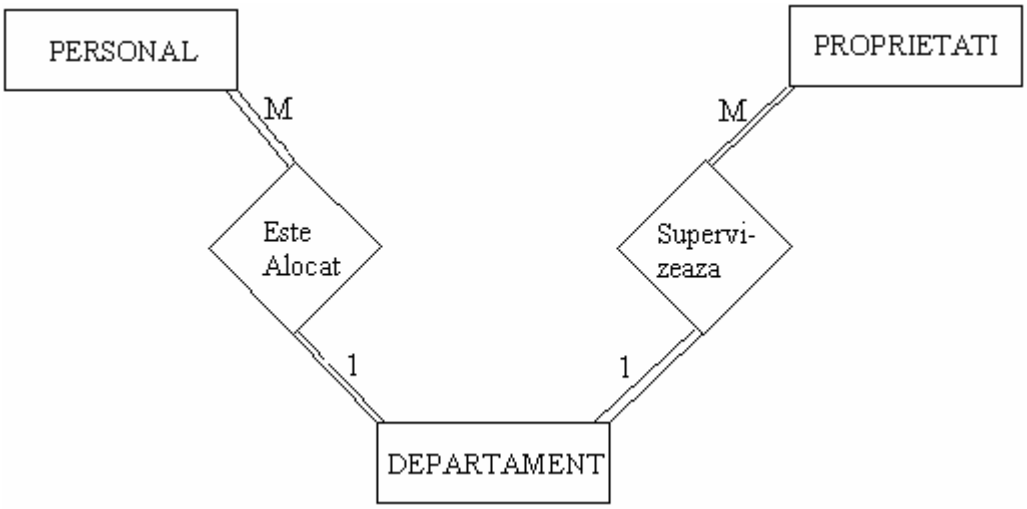

Fig.C.3 Model ER cu capcană în evantai

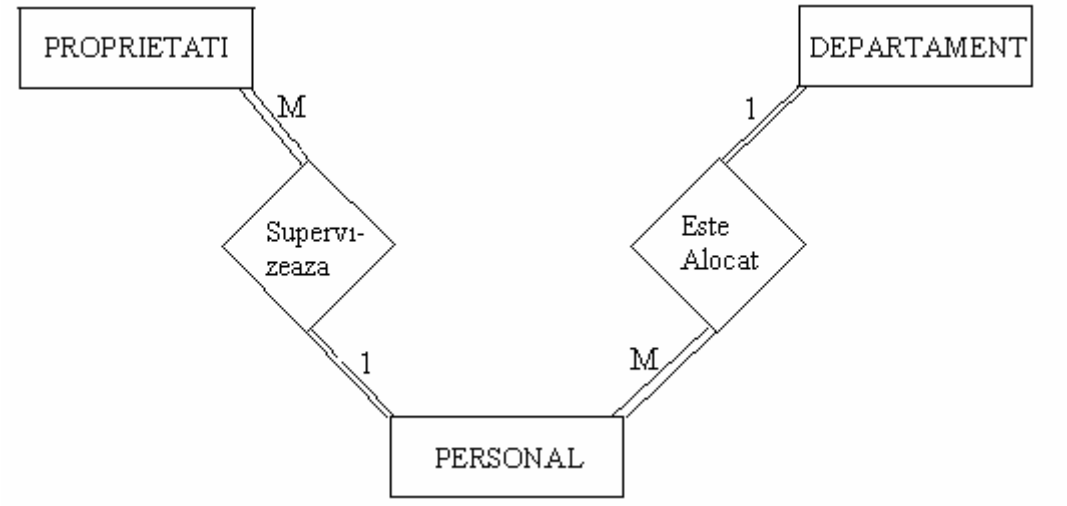

Fig.C.4 Model ER modificat pentru eliminarea capcanei în evantai

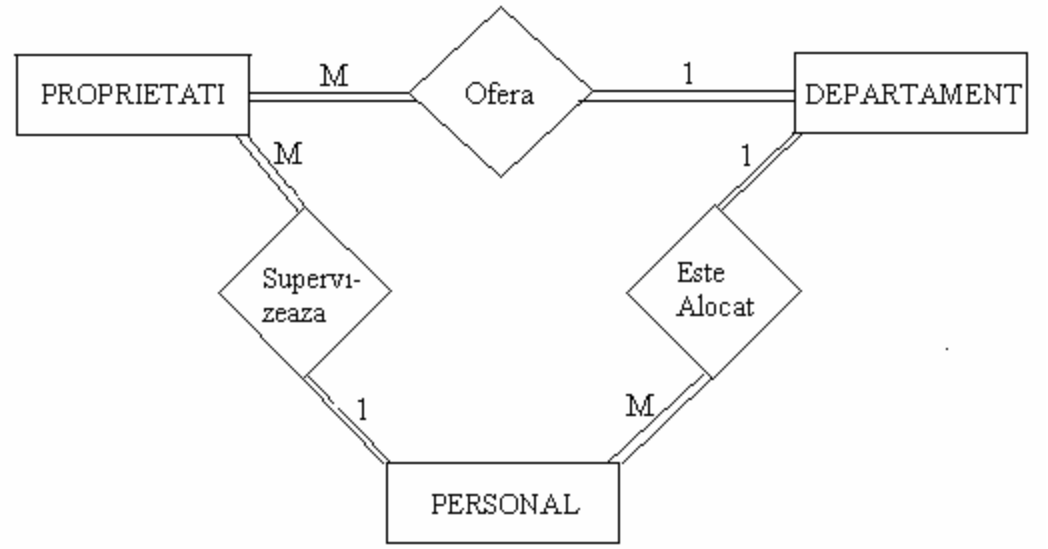

Fig.C.5 Model ER modificat pentru eliminarea capcanei de întrerupere

C.13 MODELUL ENTITATE-RELAȚIE EXTINS

Modelul ER completat cu concepte semantice adiționale de tip subclasă şi superclasă se numeşte **model Entitate-Relație Extins** (EER – *Extended Entity Relation*).

Un tip de entitate poate fi partajat în mai multe subclase cu rol distinct şi eventual cu atribute distincte.

Prin **subclasă** se înțelege un tip de entitate cu rol distinct în organizația respectivă care apare inclus într-un tip de entitate mai larg.

Mai multe subclase alcătuiesc o **superclasă**, adică un tip de entitate cu sens mai larg.

De exemplu, entitatea Clienți ca superclasă poate fi divizată în două subclase: Vânzători şi Cumpărători.

Subclasa moşteneşte atributele superclasei dar poate avea şi atribute proprii, distincte. În diagrama EER, atributele specifice unei subclase sunt ataşate direct la dreptunghiul care o reprezintă. Superclasei i se ataşează numai atributele comune.

Când se identifică un set de subclase este necesară stabilirea atributelor specifice fiecărei subclase şi a relațiilor dintre aceste subclase şi alte tipuri de entități.

Relațiile dintre subclase şi superclasă se reprezintă cu semnul de incluziune, specific operațiilor cu mulțimi ($\subset$).

Dacă o entitate nu poate fi membră decât într-o singură subclasă a unei superclase, acest fapt constituie o **constrângere de disjuncție**, reprezentată în diagramă prin litera **d,** plasată într-un cerculeț care leagă superclasa la subclasele respective. În toate celelalte cazuri, există **suprapuneri** ale subclaselor, fapt reprezentat prin litera **s** într-un cerculeț.

Procesul de maximizare a diferențelor dintre membrii unei entități prin identificarea caracteristicilor lor distincte se numeşte **proces de specializare.**

La rândul său, o subclasă poate avea alte subclase ceea ce determină o aşa-numită **ierarhie de specializare**.

Procesul de minimizare a diferențelor dintre entități, prin identificarea caracteristicilor lor comune, reprezintă un **proces de generalizare.** Rezultatul unui proces de generalizare este o superclasă. Generalizarea este procedeul invers specializării.

Dacă în procesul de specializare, fiecare entitate a superclasei este în mod obligatoriu membră a unei subclase spunem că s-a realizat o **specializare cu participare totală**. În caz contrar, se numeşte **specializare cu participare parțială**.

Grafic, participarea totală este marcată printr-o linie dublă între superclasă şi cerculețul de specializare din care derivă subclasele iar cea parțială este reprezentată cu o linie simplă.

De exemplu, tipul de entitate Clienți poate fi divizat în două subclase Cumpărători şi Vânzători, nu neapărat disjuncte, dar cu participare totală (Figura C.6).

O subclasă subordonată mai multor superclase distincte se numeşte **categorie.**

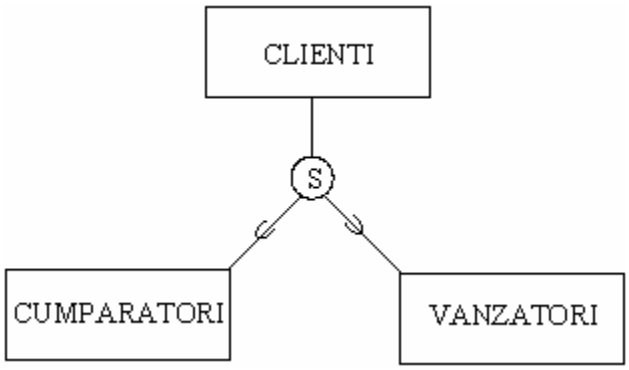

Figura C.6 Specializarea superclasei Clienți

Procesul de modelare a unei categorii este denumit **categorisire.**

O categorie are o moştenire selectivă a atributelor superclaselor cărora le este subordonată.

Categoria este legată printr-un cerculeț de categorisire, incluzând simbolul de reuniune a mulțimilor, de superclasele respective. Linia de legătură dintre categorie şi cerculeţul de categorisire este dublă în cazul participării totale (orice entitate din superclase este membră a categoriei= şi simplă pentru participare parțială.

De exemplu, tipul de entitate Clienți poate fi privit ca o categorie a tipurilor de entități Cumpărători, şi Vânzători, cu participare totală (Figura C.7).

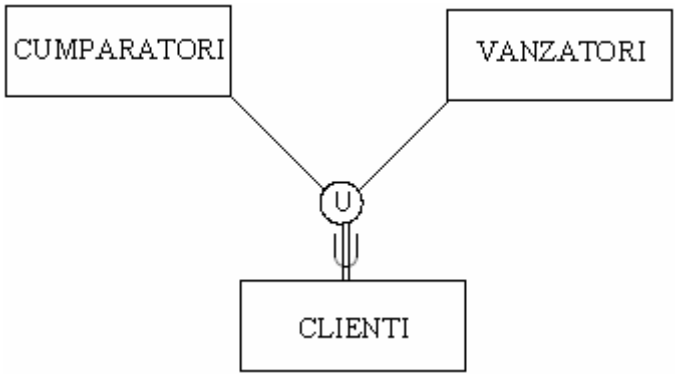

Figura C.7 Categoria Clienți

Alegerea procedeului de specializare sau de categorisire rămâne la latitudinea proiectantului. Este totuşi indicat ca în cazul partajării mai multor atribute, inclusiv a cheii primare, să se aplice procedeul de specializare/generalizare.

C14. ETAPELE PROIECTĂRII CONCEPTUALE A BAZELOR DE DATE. NORMALIZAREA

Etapele de construire a modelului EER reprezintă procesul de proiectare conceptuală a BD care constă în:

- identificarea tipurilor de entități
- identificarea tipurilor de relații
- determinarea cardinalității şi constrângerilor de participare
- identificarea atributelor şi asocierea lor cu tipurile de entități sau de relații

- determinarea cheilor candidat şi alegerea cheilor primare
- specializarea sau generalizarea tipurilor de entități (dacă este cazul)
- categorisirea tipurilor de entități (dacă este cazul)
- reprezentarea grafică a diagramei EER.

Normalizarea este un procedeu de identificare a setului optim de relații, bazat pe dependențele funcționale dintre atribute, care are ca scop crearea unui model logic de date valid, neredundant, care să elimine riscurile de apariție a anomaliilor de reactualizare a datelor în baza de date.

Dependența funcțională dintre atribute semnifică faptul că unul sau mai multe atribute sunt determinate în mod unic de un alt atribut sau set de atribute, care constituie **determinantul**.

De exemplu, toate atributele unei relații sunt dependente funcțional de cheia primară, cu excepția atributelor care o formează.

Datele redundante sau „dublate" pot conduce la reactualizări eronate ale BD. **Anomaliile de reactualizare** se clasifică în trei categorii:

- **anomalii de inserare** de noi date sau înregistrări cauzate de incoerența unor relații din schema relațională. De exemplu, dacă se defineşte o relație Personal_Departament, în care sunt incluşi toți membrii de personal din agenție şi fiecare departament, cu atributele sale, apare de mai multe ori în tabel, pe rândurile corespunzătoare reprezentanților săi, se produce o dublare a datelor. În cazul în care se doreşte înființarea unui nou departament care inițial nu are angajați, este necesară introducerea de nul-uri în cheia primară a relației ceea ce nu este admis. Dacă se angajează un nou membru de personal trebuie să introducem corect datele referitoare la departamentul în care va fi repartizat, aşa cum apar ele şi în celelalte înregistrări ale personalului aferent aceluiaşi birou. Reprezentarea distinctă a tipurilor de entități în două relații Personal şi Departamente elimină această dublare a datelor şi anomaliile de inserare. Fiecare membru de personal este asociat numai cu numărul de identificare a departamentului, atributele acestuia fiind incluse în alt tabel.
- **anomalii de modificare** sunt cauzate de dublarea datelor în tabele. De exemplu, modificarea unui atribut al unui departament, impune schimbarea valorii datelor din mai multe rânduri. Dacă nu se reactualizează toate înregistrările corespunzătoare, se vor genera rapoarte incorecte, cu date depreciate.

- **anomalii de ştergere** determină pierderea unor date din BD. De exemplu, dacă în relația Personal_Departament se elimină singurul membru al unui departament, se şterg şi datele referitoare la acesta.

Pentru a evita anomaliile de reactualizare este necesară descompunerea unor relații în mai multe relații care să preia seturi parțiale din atributele inițiale astfel încât într-o relație să nu existe date dublate.

Forma nenormalizată (UNF – *Unnormalized Form***)** a unui tabel din BD include date repetitive. Un astfel de tabel nu este considerat relație în BD.

Normalizarea este o tehnică de analiză şi testare a relațiilor, pe baza cheilor primare şi candidat, respectiv a dependențelor funcționale.

Normalizarea se realizează în mai mulți paşi, denumiți şi forme normalizate:

1. prima formă normalizată (1NF) se obține atunci când la intersecția fiecărei linii cu fiecare coloană din orice relație a schemei apare numai **o singură valoare**. Este necesară identificarea şi eliminarea grupurilor repetitive din tabel, prin introducerea de noi rânduri şi definirea unei chei primare compuse din mai multe atribute. Dacă un atribut al cheii primare compuse se repetă, aceasta nu constituie o dublare a datelor, dar poate crea probleme la reactualizarea BD.

2. a doua formă normalizată (2NF) se aplică relațiilor cu chei compuse şi constă în faptul că fiecare atribut care nu aparține cheii primare este total dependent funcțional de aceasta. **Dependența funcțională totală** apare atunci când atributul dependent nu depinde de un subset de atribute al unei chei şi eliminarea oricărei componente din determinant conduce la dispariția dependenței. În caz contrar, se consideră că este o **dependență funcțională parțială**. O relație cu cheie primară singulară (cu un singur atribut) este în 2NF. Pentru o relație cu cheie primară compusă, trecerea în 2NF se face prin eliminarea dependențelor funcționale parțiale. Pentru aceasta, fiecare atribut dependent funcțional parțial este copiat într-un nou tabel, împreună cu o copie a determinantului. De exemplu, într-o relație Client_Proprietate cu cheie primară compusă (Nr_client, Nr_proprietate), atributul Nume_client depinde parțial de cheia primară. Este necesară definirea unei noi relații Client în care să se includă toate atributele clienților împreună cu determinantul Nr_client ca şi cheie primară. Normalizarea în a doua formă se face prin identificarea şi eliminarea tuturor dependențelor funcționale parțiale de cheia primară, eventual şi cheile candidat.

3. a treia formă normalizată (3NF) se obține prin eliminarea dependențelor tranzitive din model. Prin definiție, atunci când un atribut A determină atributul B, iar B determină atributul C, apare **dependența tranzitivă** a atributului C de atributul determinant A prin intermediul atributului B. Dependențele funcționale tranzitive dintre atributele unei relații pot cauza erori de reactualizare. O relație aflată în 1NF şi 2NF se găseşte în 3NF dacă nici un atribut din afara cheii primare nu este dependent tranzitiv de cheia primară, prin intermediul altui atribut care nu intră în componența cheii primare. Pentru obținerea formei a treia de normalizare, orice atribut cu dependență tranzitivă se plasează într-o nouă relație împreună cu copia determinantului (determinanților). De exemplu, în relația Proprietate_Proprietar, cu cheia primară simplă Nr_proprietate, atributul Nume_proprietar depinde funcțional de atributul Nr_proprietar care la rândul său este determinat de cheia primară ceea ce arată apariția unei dependențe funcționale tranzitorii. Eliminarea ei se face prin descompunerea relației inițiale în două relații, Proprietăți şi Proprietari, în care Nr_proprietar devine cheie primară care determină funcțional total toate celelalte atribute (Nume_proprietar, Adresa, Telefon etc) iar în relația Proprietăți apare cheia străină Nr_proprietar ca singur atribut care descrie proprietarul unei proprietăți.

4. forma normală Boyce-Codd (BCNF) ia în considerare şi **dependențele funcționale de** eventualele **chei candidat** dintr-o relație şi se obține după crearea formei 3NF, atunci când toți determinanții sunt chei candidat. Pentru o relație cu o singură cheie candidat care se utilizează implicit ca şi cheie primară, formele 3NF şi BCNF sunt echivalente. De exemplu, într-o relație Proprietate_Vizitare cu cheia primară (Nr_proprietate, Nr_client) apar mai multe dependențe funcționale care sunt admise dacă determinantul lor este cheie candidat. Atributele Data_vizitei, Ora_vizitei, Nr_Agent, Maşină, Comentarii sunt determinate funcțional total de cheia primară. Dar atributul Maşină depinde funcțional şi de setul de atribute (Data_vizitei, Ora_vizitei, Nr_agent) care însă constituie o cheie candidat. Prin analiza tuturor dependențelor din relație se stabileşte dacă aceasta este sau nu în forma BCNF.

5. a patra formă normalizată (4NF) se obține din BCNF prin eliminarea **dependențelor funcționale multivalorice (MVD) netriviale** (nu determină integral tabelul), care apar în procesul de generare a primei forme de normalizare din cauza relațiilor de tip 1:M independente dintre tipurile de entități. De exemplu, există astfel de relații 1:M între entitățile Departament şi Personal sau Departament şi Clienți. Într-o relație

Departament_Personal_Clienți există o singură cheie candidat (Nr_departament, Nr_agent, Nr_client) deci relația este în forma BCNF. Pentru a evita apariția atributelor cu valori multiple, pentru același departament se combină în mai multe rânduri numele agenților cu numele clienților pe care îi deservesc. Deci numele unui agent poate corespunde la mai mulți clienți iar combinația (Nr_agent, Nr_departament) apare redundant, de mai multe ori. Similar, apar repetate combinațiile (Nr_client, Nr_departament). Pentru eliminarea acestor dependențe multivalorice, se descompune relația inițială în două relații, de exemplu, Departament_Personal și Departament_Clienți.

6. a cincia formă normalizată (5NF) se obține din 4NF prin descompunerea unei relații în după sau mai multe relații independente care elimină **dependențele de tip uniune fără pierderi.** care garantează faptul că prin uniunea naturală a mai multor tabele rezultate din descompunerea altuia nu se generează date sau corespondențe false.

C15. PROIECTAREA LOGICĂ A BAZELOR DE DATE RELAȚIONALE

Proiectarea logică a BD continuă procesul de proiectare conceptuală, prin construirea modelului logic global, independent de SGBD care va fi utilizat, precum și de orice considerații de implementare fizică a BD.

Se construiesc modelele logice locale pe baza vederilor fiecărui tip de utilizator al BD.

Modelul logic global include toate modelele logice de date locale.

Construcția acestor modele logice locale în mod coerent implică:

- eliminarea relațiilor de tip M:N
- eliminarea relațiilor complexe
- eliminarea relațiilor recursive
- eliminarea atributelor cu valori multiple
- eliminarea relațiilor cu atribute prin definirea de noi tipuri de entități
- eliminarea atributelor cu valori multiple
- eliminarea relațiilor redundante
- normalizarea modelelor

- verificarea posibilităților de efectuare a tranzacțiilor cerute de fiecare potențial utilizator
- impunerea constrângerilor de integritate asupra datelor, domeniilor atributelor, entităților, referențială şi de întreprindere (reguli de afaceri)
- impunerea constrângerilor de existență prin definirea condițiilor în care o cheie candidat sau străină poate fi inserată, reactualizată sau ştearsă şi precizarea acțiunilor permise (NO ACTION, CASCADE, SET NULL, SET DEFAULT, NO CHECK)
- elaborarea documentației aferente BD, care să includă schema relațională şi dicționarul de date.

Realizarea modelului logic global impune:

- revizuirea denumirilor entităților, relațiilor şi cheilor primare
- îmbinarea entităților din vederile locale şi includerea entităților unice care apar în modelele locale
- îmbinarea relațiilor din vederile locale şi includerea relațiilor unice
- identificarea entităților şi relațiilor lipsă
- verificarea cheilor străine
- verificarea constrângerilor de integritate
- desenarea diagramei modelului logic global
- reactualizarea documentației
- validarea modelului prin evaluarea capacității acestuia de a efectua orice tranzacție cerută de utilizatori.

APLICAȚIE

Pe baza modelului BD din aplicația anterioară, reprezentați grafic fiecare entitate şi atributele acesteia în diagrame separate, respectând convențiile de reprezentare enumerate mai sus.
Grupați toate informațiile în tabelele următoare conform vederii managerului firmei:

Tabel C1. Documentarea tipurilor de entități

Tip de entitate	Descriere	Aliasuri	Detalii de interpretare

Tabel C2. Documentarea relațiilor

Tip de entitate 1	Tip de relație	Tip de entitate 2	Raport de cardinalitate	Participare (T, P)

Tabel C3. Documentarea atributelor

Tip de entitate	Denumirile atributelor	Descriere	Tip şi lungime a câmpurilor	Chei	Valoare prestabilită	Alias	Nul admis?	Derivat?

Tabel C4. Documentarea domeniilor atributelor

Denumirea domeniului	Caracteristicile domeniului	Exemple de valori admise

Reprezentați într-o diagramă ER toate entitățile şi relațiile dintre acestea, în care specificați rapoartele de cardinalitate şi constrângerile de participare din fiecare caz, precum şi cheile primare utilizate. Identificați posibilele capcane din model şi eliminați-le. Dezvoltați modelul ER extins identificând eventualele subclase sau superclase de entități, cu sau fără specializare/generalizare sau categorisire.

Descrieți în limbaj DBDL schema relațională globală, specificând entitățile cu atributele, cheile primare, alternative şi/sau străine asociate inclusiv constrângerile impuse cheilor străine.

De exemplu:

Personal (Nr_personal, Nume, Prenume, Funcție, Salariu, Data_de_naştere, CNP, Adresă, Telefon, Data_de_angajare, Nr_departament)

Cheie primară Nr_personal

Cheie alternativă Nume, Prenume, CNP

Cheie străină Nr_departament **se referă la** Departamente(Nr_departament)

C16. PROIECTAREA FIZICĂ A BAZELOR DE DATE

Pe baza modelului de date logic global se realizează o schemă relațională de bază cunoscând SGBD țintă.

Se transformă relațiile de bază identificate în modelul de date logic global având în vedere caracteristicile SGBD care va fi utilizat şi se întocmeşte documentația referitoare la proiectarea relațiilor de bază pentru un anumit SGBD.

Pasul următor constă în proiectarea constrângerilor întreprinderii pentru SGBD şi documentarea acestora.

Pentru reprezentarea fizică a BD, se determină modalitățile de organizare afişierelor în BD şi metodele de acces care vor fi utilizate pentru stocarea schemei relaționale în capacitatea de memorie secundară, prin:

- analiza tranzacțiilor care pot fi executate în BD pe baza schemei relaționale de lucru.
- determinarea modalității optime de organizare a fiecărui fişier din BD.
- opțional se pot introduce indexuri secundare dacă acestea îmbunătățesc performanțele sistemului de tip BD.
- se analizează oportunitatea introducerii unei redundanțe controlate a datelor prin dublarea unor date şi atribute, eventual uniunea unor relații.
- estimarea spațiului necesar pe disc pentru stocarea BD.

Urmează proiectarea mecanismelor de securitate, a vederilor utilizatorilor şi a regulilor de acces la BD şi se elaborează documentația pentru acestea.

Pe măsură ce BD este utilizată şi monitorizată, se pot observa şi corecta deciziile de proiectare inadecvate şi se pot efectua eventualele modificări necesare unor cerințe noi ale utilizatorilor.

Un exemplu de SGBD tipic bazat pe PC este MS Access care permite crearea de tabele, formulare, rapoarte şi interogări pentru aplicații BD personalizate, rulate pe un singur calculator sau într-o rețea de calculatoare cu utilizatori multipli.

MS Access are anumite particularități:

- nu face deosebire între şirurile de caractere cu lungime fixă şi variabilă, folosind în acest caz numai tipul de date Text.

- nu admite ca o cheie străină sa facă referire la mai mult de o relație-părinte pe baza unei constrângeri referențiale. Este necesară eventuala specializare a entității-părinte pentru diferențierea mai multor chei străine cu un singur părinte, care însă admit nul-uri.
- oferă pentru operațiile de reactualizare şi ştergere de date din BD numai opțiunile CASCADE şi NO ACTION (implicit), fiind necesară reproiectarea tranzacțiilor pentru menținerea integrității referențiale a BD.
- permite crearea tabelelor cu specificarea câmpurilor acestora şi a proprietăților lor: *Field Size* pentru date de tip *Text, Number* şi *Autonumber*; Format pentru specificarea modului de afişare a datei, orei şi textului; *Decimal Places*; *Input Mask* pentru exprimarea opțiunilor de formatare datelor (\P – afişează în clar litera respectivă; L – semnifică o literă introdusă de la tastatură; >L – determină afişarea ca majusculă a literei care se introduce de la tastatură pe poziția marcată în mască cu semnul „mai mare"; 0 – specifică necesitatea introducerii unei cifre; 9 – reprezintă cifre care se introduc opțional); *Caption* include denumirea dorită pentru o anumită coloană din tabel, diferită de numele folosit în schema relațională pentru atributul respectiv; *Default Value*; *Validation Rule* – pentru exprimarea unor constrângeri asupra valorilor atributelor; *Validation Text* – specifică mesajul de avertizare care se afişează atunci când utilizatorul introduce date în mod incorect; *Required* – prin opțiunea *No* specifică admiterea nul-urilor pentru acel atribut; *Indexed* – folosirea indexurilor pe unele câmpuri pentru accelerarea procesului de interogare.
- permite folosirea listelor de căutare şi de valori (*Lookup/Value*). Pentru crearea unei liste de căutare se poate crea un tabel suplimentar cu coduri şi descrierea acestora, urmând ca în tabelul inițial să se folosească pentru lista de căutare acest tabel, cu câmpul de coduri ca o cheie străină. Listele de valori includ un set finit de valori permise pentru un anumit câmp. Modificarea valorilor din listă nu se reflectă în înregistrările anterioare.
- permite definirea relațiilor dintre tabele (în fereastra *Relationships*) pe baza cheilor străine, cu stabilirea regulilor de reactualizare şi ştergere.