

II. Cunoștințe de specialitate

BAZE DE DATE

*Baze de date relaționale. Definiții, componente, niveluri de abstractizare.
(Mehedințu A., Baze de date, Note de curs, pag. 3-13)*

1.1 Definiții

Baze de date

O bază de date este o colecție de informații interrelaționate gestionate ca o singură unitate. Această definiție este intenționat foarte largă, deoarece există mari diferențe între concepțiile diferiților producători care pun la dispoziție sisteme de baze de date. De exemplu, Oracle Corporation definește o bază de date ca fiind o colecție de fișiere fizice gestionate de o singură instanță (copie) a produsului software pentru baze de date, în timp ce Microsoft definește o bază de date SQL Server ca fiind o colecție de date și alte obiecte. Un *obiect* al bazei de date este o structură de date denumită stocată în baza de date, cum ar fi un tabel, o vizualizare sau un index.

Sisteme de gestiune a bazelor de date

Un *sistem de gestionare a bazei de date (DBMS - database management system)* este un produs software furnizat de producătorul bazei de date. Produse software precum Microsoft Access, Microsoft SQL Server, Oracle Database, Sybase, DB2, INGRES, MySQL și PostgreSQL fac parte din categoria DBMS sau, mai corect, *DBMS relaționale (RDBMS)*.

Sistemul DBMS pune la dispoziție toate serviciile de bază necesare pentru organizarea și întreținerea bazei de date, inclusiv următoarele:

- Transferarea datelor în și din fișierele fizice de date, în funcție de cerințe.
- Gestionarea accesului concurențial la date al mai multor utilizatori, inclusiv prevenirea conflictelor care ar putea fi cauzate de actualizările simultane.
- Gestionarea tranzacțiilor, astfel încât toate modificările făcute asupra bazei de date printr-o tranzacție să fie executate ca o singură unitate. Cu alte cuvinte, dacă tranzacția reușește, toate modificările efectuate de tranzacție sunt înregistrate în baza de date; dacă tranzacția eșuează, nici una dintre modificări nu este înregistrată în baza de date. Totuși, unele sisteme RDBMS nu asigură suportul pentru tranzacții.
- Acceptă un *limbaj de interogare*, care reprezintă sistemul de comenzi folosit de utilizator pentru a obține date din baza de date. SQL este principalul limbaj folosit pentru sistemele DBMS relaționale.
- Funcții pentru salvarea bazei de date și pentru refacerea bazei de date în urma erorilor.
- Mecanisme de securitate pentru împiedicarea accesului neautorizat la date și modificarea acestora.

Baze de date relaționale

O *bază de date relațională* este o bază de date care respectă modelul relațional, dezvoltat de Dr. E. F. Codd. Modelul relațional prezintă datele sub forma familiarelor tabele bidimensionale, similar cu o foaie de calcul tabelar Excel. Spre deosebire de o foaie de calcul tabelar, nu este obligatoriu ca datele să fie stocate într-o formă tabelară, iar modelul permite și combinarea tabelor (*crearea uniunilor (joining)*, în terminologia relațională) pentru formarea vizualizărilor, care sunt prezentate tot ca tabele bidimensionale. Flexibilitatea extraordinară a bazelor de date relaționale este dată de posibilitatea de a folosi tabelele independent sau în combinații, fără nici o ierarhie sau secvență predefinită în care trebuie să se facă accesul la date.

1.2 Componente ale bazelor de date relaționale

1. Tabele

Unitatea primară de stocare a datelor într-o bază de date relațională este *tabelul*, care este o structură bidimensională compusă din rânduri și coloane. Fiecare tabel reprezintă o *entitate*, ceea

ce înseamnă o persoană, un loc, un lucru sau un eveniment care trebuie să fie reprezentat în baza de date, cum ar fi un client, un cont bancar sau o tranzacție bancară. Fiecare rând al tabelului reprezintă o apariție a entității.

Figura 1.1 reprezintă conținutul parțial al unui tabel numit **MOVIE** (Film).

Tabelul **MOVIE** este parte a unei baze de date pentru un magazin de produse video, folosită ca exemplu.

Tabelul **MOVIE** conține date care descriu filmele disponibile în magazinul de produse video. Fiecare rând din tabel reprezintă un film, iar fiecare coloană reprezintă o caracteristică a filmului respectiv, cum ar fi titlul filmului.

MOVIE_ID	MOVIE_GENRE_CODE	MPAA_RATING_CODE	MOVIE_TITLE	RETAIL_PRICE_VHS	RETAIL_PRICE_DVD	YEAR_PRODUCED
1	Drama	R	Mystic River	58.97	19,96	2003
2	ActAd	R	The Last Samurai	15,95	19,96	2003

Fig. 1.1 Structura tabelii Movie (Film)

2. Relații

Relațiile reprezintă asocierile dintre tabelele bazelor de date relaționale. Deși fiecare tabel relațional poate exista independent, esența bazelor de date este tocmai stocarea informațiilor între care există legături. De exemplu, pe lângă filmele propriu-zise, se pot stoca și informații despre categoriile folosite de magazin pentru organizarea inventarelor de filme. În același timp, puteți stoca și informații despre copiile fiecărui film, inclusiv data la care a fost primită copia și formatul acesteia, cum ar DVD sau VHS. Prin folosirea relațiilor, se pot asocia tabelele înrudite, într-un mod formal, ușor de folosit astfel încât să combinăm date din tabele multiple în aceeași interogare a bazei de date, dar păstrând flexibilitatea de a include numai informațiile care îl interesează pe utilizator. Posibilitatea de a selecta din baza de date numai informațiile care ne interesează ne permite să ajustăm informațiile din baza de date în funcție de cerințele specifice ale persoanelor sau aplicațiilor care au acces la baza de date.

Figura 1.2 prezintă patru tabele din baza de date a magazinului de produse video și relațiile dintre acestea, într-un format cunoscut sub numele de diagramă de relații a entităților (ERD - Entity Relationship Diagram). Diagramele ERD ne pun la dispoziție o modalitate de prezentare a proiectului general al unei baze de date relaționale, într-un format ușor de înțeles pentru utilizatorii bazei de date, indiferent dacă au sau nu cunoștințe tehnice. Fiecare dreptunghi din diagramă reprezintă un tabel relațional, cu numele tabelului scris deasupra liniei orizontale și coloanele tabelului enumerate pe verticală, în porțiunea principală a dreptunghiului.

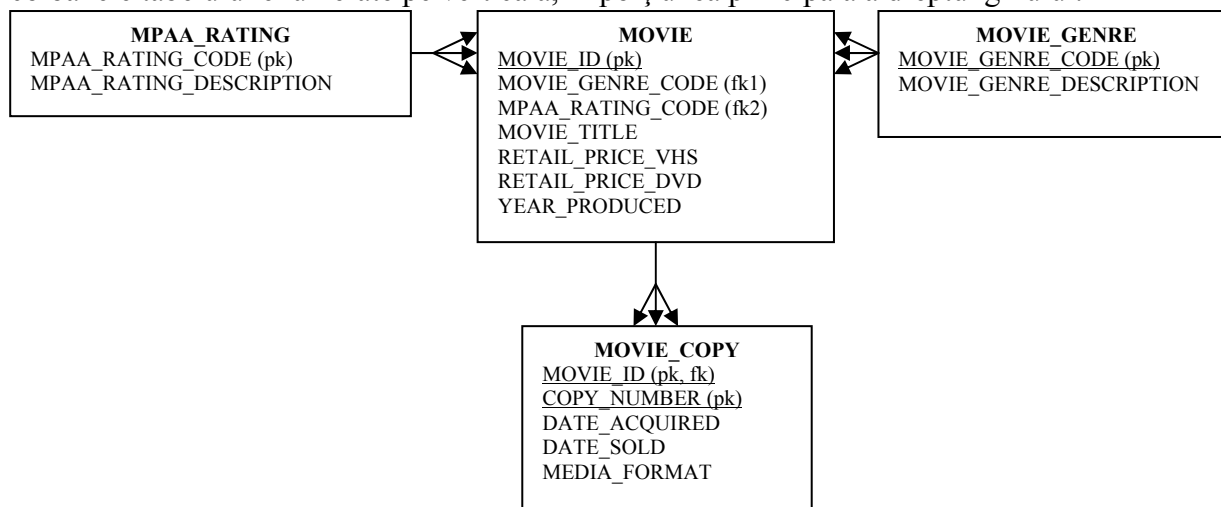


Fig. 1.2 Diagrama ERD a bazei de date pentru magazinul de produse video (prezentare parțială)

În funcție de numărul de elemente, între care se stabilesc relații, aparținând celor două

colecții, aceste relații pot fi de tip unu la unu, unu la mulți și mulți la mulți.

Relațiile de tipul 1→1 (unu la unu), care presupun că unui membru din colecția A îi corespunde un singur membru din colecția B.

Relațiile de tipul 1→m sau m→1 (unu la mulți sau mulți la unu), care presupun că unui membru din prima entitate A îi corespund mai mulți membri din a doua entitate B; astfel de relații se mai numesc și relații ierarhice.

Relațiile de tipul m→m (mulți la mulți), în care unui membru din entitatea A îi corespund mai multe date din colecția B și mai multor date din colecția A îi corespunde o singură dată din colecția B.

Relații de tip mulți la mulți se mai numesc și relații de tip rețea. O relație mulți la mulți se va descompune întotdeauna în două relații, o relație tip unu la mulți și respectiv o a doua relație de tip mulți la unu prin intermediul unei entități de legătură.

Fiecare relație este prezentată în diagrama ERD ca o linie ce conectează două tabele. Cele două capete ale liniei arată *cardinalitatea maximă* a relației, respectiv numărul maxim de rânduri dintr-un tabel care pot fi asociate cu un rând dat din tabelul aflat la celălalt capăt al relației.

Relațiile sunt implementate folosind coloane corespondente din cele două tabele participante. În diagrama ERD, coloana sau coloanele subliniate din fiecare tabel, având în dreapta notația „pk”, reprezintă cheia primară (primary key), adică o coloană sau un set de coloane care identifică în mod unic fiecare rând dintr-un tabel.

Un tabel poate avea o singură cheie primară. Totuși, o cheie primară poate fi compusă din mai multe coloane, dacă aceasta este calea de formare a unei chei unice. Dacă o cheie primară este folosită într-un alt tabel pentru stabilirea unei relații, poartă numele de cheie externă.

În fig. 1.2, observați coloanele cheie externă folosite în tabelul MOVIE pentru crearea relațiilor cu tabelele MOVIE_GENRE și MPAA_RATING și marcate cu identificatoarele „<fk1>” și „<fk2>” în dreapta numelui coloanei cheie externă.

Cheile primare și cheile externe sunt blocuri de construcție fundamentale ale modelului relațional, deoarece stabilesc relații și permit crearea legăturilor între date, atunci când este necesar. Trebuie să înțelegeți acest concept pentru a putea înțelege cum funcționează bazele de date relaționale.

3. Restricții

O restricție este o regulă specificată pentru un obiect al bazei de date (de obicei, un tabel sau o coloană), având rolul de a limita într-un mod oarecare domeniul de valori permise pentru obiectul respectiv al bazei de date

După ce sunt specificate, restricțiile sunt impuse automat de sistemul DBMS și nu pot fi ocolite decât dacă o persoană autorizată le dezactivează sau le șterge (le elimină). Fiecare restricție primește un nume unic, astfel încât să poată fi referită în mesajele de eroare și în comenzile folosite ulterior în baza de date. Este recomandabil ca proiectanții bazei de date să denumească restricțiile, deoarece numele generate automat de baza de date nu sunt foarte descriptive.

Există mai multe tipuri de restricții pentru baze de date:

- **Restricția NOT NULL.** Poate fi plasată pe o coloană pentru a împiedica folosirea valorilor nule. O valoare nulă (null) este o modalitate specială prin care sistemul RDBMS tratează valoarea unei coloane pentru a indica faptul că valoarea coloanei respective nu este cunoscută. O valoare nulă nu este același lucru cu un spațiu liber, un șir vid sau valoarea zero - este o valoare specială care nu este egală cu nimic altceva.
- **Restricția cheie primară (primary key).** Definită pe coloana (coloanele) cheie primară ale unui tabel pentru a garanta că valorile cheie primară sunt întotdeauna unice în întreg tabelul. Atunci când cheia primară este definită pe mai multe coloane, combinația valorilor acelor coloane trebuie să fie unică în tabel - o coloană care reprezintă doar o parte a cheii primare poate conține valori duplicate în tabel. Restricțiile cheie primară sunt aproape întotdeauna implementate de RDBMS prin folosirea unui index. Indexul este un tip special de obiect al

bazei de date care permite efectuarea căutărilor rapide în valorile coloanei. Atunci când în tabele sunt inserate rânduri noi, sistemul RDBMS verifică automat indexul pentru a se asigura că *pk* a noului rând nu este deja folosită în tabel și, dacă se întâmplă acest lucru, respinge cererea de inserare. Căutarea în indexuri se face mult mai repede decât căutarea în tabel; ca urmare, indexarea cheii primare este esențială pentru orice tabel, indiferent de dimensiunea acestuia, astfel încât căutarea cheilor duplicate la fiecare inserare să nu ducă la o reducere semnificativă a performanțelor. O caracteristică suplimentară a cheilor primare este faptul că nu pot fi definite decât pe coloane pentru care a fost definită și restricția NOT NULL.

- **Restricția de unicitate (unique).** Definită pe o coloană sau un set de coloane care trebuie să conțină valori unice în cadrul tabelului. Ca și în cazul cheilor primare, sistemul RDBMS folosește aproape întotdeauna un index ca modalitate de impunere eficientă a restricției. Totuși, spre deosebire de cheile primare, un tabel poate avea definite mai multe restricții de unicitate, iar coloanele care participă la o restricție de unicitate pot conține (în cele mai multe sisteme RDBMS) și valori nule.
- **Restricția referențială** (numita uneori *restricție de integritate referențială*). O restricție care impune o relație între două tabele dintr-o bază de date relațională. Prin „impunere”, se înțelege că sistemul RDBMS se asigură întotdeauna, în mod automat, că fiecărei valori a cheii externe îi corespunde o valoare a cheii primare în tabelul părinte. Pe scurt, restricția referențială garantează că relația dintre cele două tabele și valorile corespondente ale cheii primare și cheii externe își păstrează logica în orice moment.
- **Restricția CHECK.** Folosește o instrucțiune logică simplă (scrisă în SQL) pentru a valida valoarea unei coloane. Rezultatul instrucțiunii trebuie să fie o valoare logică de adevărat (true) sau fals (false), astfel încât un rezultat adevărat să permită inserarea în tabel a valorii coloanei, iar un rezultat fals să ducă la rejectarea valorii coloanei, cu mesajul de eroare corespunzător.

4. Vizualizări

O vizualizare (view) este o interogare stocată în baza de date care pune la dispoziția utilizatorului un subset personalizat al datelor din unul sau mai multe tabele ale bazei de date. Cu alte cuvinte, o vizualizare este un tabel virtual, deoarece arată ca un tabel și, în cele mai multe privințe, se comportă ca un tabel, dar nu stochează date (nu este stocată decât interogarea SQL care definește vizualizarea). Vizualizările au mai multe funcții utile:

- Maschează coloanele pe care utilizatorul nu este nevoie să le vadă (sau nu-i este permis să le vadă).
- Maschează rândurile pe care utilizatorul nu este nevoie să le vadă (sau nu-i este permis să le vadă).
- Maschează operațiile complexe efectuate în baza de date, cum ar fi uniunile de tabele (respectiv combinarea coloanelor din tabele multiple într-o singură interogare a bazei de date).
- Îmbunătățesc performanțele interogărilor (în unele sisteme RDBMS precum Microsoft SQL Server).

1.3 Niveluri de abstractizare a datelor

Într-un sistem informatic ce utilizează BD, organizarea datelor poate fi analizată din mai multe puncte de vedere și pe diferite niveluri. De obicei, abordarea se face pe trei niveluri: *intern*, *conceptual* și *extern* (figura 1.3).

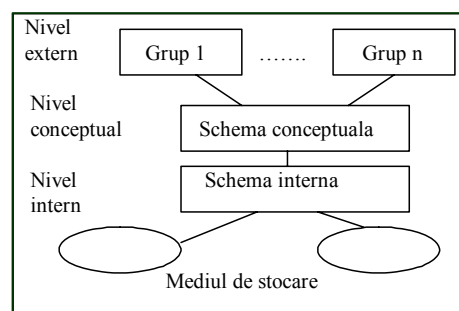


Fig. 1.3 Niveluri de abstractizare

Nivelul intern. Structura datelor este descrisă foarte detaliat, fiind accesibilă numai specialiștilor (ingineri de sistem, programatori în limbaje de asamblare sau alte limbaje apropiate de „mașină”). Cele două părți principale ale bazei la acest nivel sunt:

1) un set de programe care interacționează cu sistemul de operare pentru îmbunătățirea managementului bazei de date.

2) fișierele stocate în memoria externă a calculatorului.

Fișierele ce conțin datele propriu-zise sunt alcătuite din articole sau înregistrări cu format comun. La acest nivel, structura BD se concretizează în schema internă.

Nivelul conceptual. Este nivelul imediat superior celui fizic, datele fiind privite prin prisma semnificării lor; interesează conținutul lor efectiv, ca și relațiile care le leagă de alte date. Reprezintă primul nivel de abstractizare a lumii reale observate. Obiectivul acestui nivel îl constituie modelarea realității considerate, asigurându-se independența bazei față de orice restricție tehnologică sau echipament anume. Întreaga bază este descrisă prin intermediul unui număr restrâns de structuri. Toți utilizatorii își exprimă nevoile de date la nivel conceptual, prezentându-le administratorului bazei de date, acesta fiind cel care are o viziune globală necesară satisfacerii tuturor cerințelor informaționale. La acest nivel, structura BD se concretizează în schema conceptuală.

Nivelul extern. Este ultimul nivel de abstractizare la care poate fi descrisă o bază de date. Structurile de la nivelul conceptual sunt relativ simple, însă volumul lor poate fi deconcertant. Iar dacă la nivel conceptual baza de date este abordată în ansamblul ei, în practică un utilizator sau un grup de utilizatori lucrează numai cu o porțiune specifică a bazei, în funcție de departamentul în care își desfășoară activitatea și de atribuțiile sale (lor). Simplificarea interacțiunii utilizatori-BD, precum și creșterea securității BD sunt deziderate ale unui nivel superior de abstractizare, care este nivelul extern. Astfel, structura BD se prezintă sub diferite machete, referite uneori și ca *sub-scheme*, scheme externe sau imagini (view-uri), în funcție de nevoile fiecărui utilizator sau grup de utilizatori.

Etape de proiectare a bazelor de date relaționale. Normalizarea. ***(Mehedințu A., Baze de date, Note de curs, pag. 16-19)***

Proiectarea unei baze de date este o activitate laborioasă și necesită parcurgerea următoarelor etape:

- formularea problemei;
- analiza cerințelor informaționale și definirea datelor de ieșire și a datelor de intrare;
- definirea tabelor, a structurii acestora și a relațiilor dintre tablele;
- optimizarea structurii bazei de date.

Odată ce acest proces a fost finalizat se continuă cu:

- proiectarea procedurilor tehnologice, pentru prelucrarea bazei de date;
- elaborarea programelor;
- testarea programelor;
- definitivarea documentației.

Toate aceste activități necesită, pentru proiectele reale complexe, o muncă în echipă pe baza unei metodologii riguroase, cunoscută ca metodologia de analiză și proiectare a sistemelor informatice. În cadrul unui sistem informatic baza de date reprezintă elementul central în jurul căruia se concentrează celelalte componente ale sistemului.

Formularea problemei presupune stabilirea obiectivelor aplicației informatice care asigură actualizarea și exploatarea bazei de date în concordanță cu cerințele managementului activității economice pentru care este proiectată baza de date. Obiectivele unei aplicații informatice sunt legate de asigurarea informațională a desfășurării proceselor decizionale specifice actului de conducere. Deci, noi trebuie să ne gândim că, prin existența unei baze de date, să asigurăm fondul de informații, într-o structură și de o calitate corespunzătoare cu cerințele managementului firmei. Baza de date trebuie să permită atât obținerea unor informații

de detaliu, elementare, cât și calculul și prezentarea unor indicatori sintetici, agregați. Dacă am lua doar două obiective: reducerea costurilor și creșterea productivității muncii într-o firmă, atunci o bază de date trebuie să furnizeze informații despre consumul factorilor de producție, costurile medii și globale, despre personalul muncitor și producția realizată, despre cheltuielile salariale etc. Aceste informații vor servi conducerii la identificarea căilor de reducere a costurilor și adoptarea celor mai adecvate măsuri pentru reducerea acestor costuri. După aplicarea măsurilor în practică, informațiile stocate în baza de date trebuie să permită de data aceasta și o analiză comparată a costurilor noi cu cele vechi, de exemplu, o analiză a dinamicii costurilor pe baza indicilor statistici. Am ales acest mic exemplu didactic pentru a accentua încă o dată complexitatea procesului de proiectare a bazei de date.

Analiza cerințelor informaționale, pornind de la obiectivele formulate anterior, se concentrează asupra a două probleme:

- indicatorii, rapoartele, listele și datele de ieșire care trebuie obținute;
- datele de intrare necesare pentru obținerea datelor de ieșire.

Acestea sunt cerințele informaționale care înglobează atât cerințele pentru datele de intrare pe baza cărora se creează și se actualizează baza de date, cât și cerințele pentru datele de ieșire folosite pentru urmărirea, controlul și dirijarea activității economice. Datele de intrare se culeg, de regulă, din documentele primare care circulă în cadrul fluxului informațional al firmei. Datele finale se vor integra în ansamblul de rapoarte, liste, situații cu rezultate pe care le furnizează sistemul informațional compartimentelor de conducere. Pentru indicatorii incluși în rapoartele finale, în general pentru oricare din indicatorii de ieșire, trebuie să fie foarte clar modul în care sunt obținuți prin prelucrarea datelor de intrare. În consecință, acolo unde este cazul, se precizează algoritmi de calcul, regulile de totalizare, sau alte reguli de obținere a fiecărei coloane, sau totaluri din rapoartele finale.

Aceasta are ca punct de plecare inventarierea câmpurilor prezente în situațiile finale și apoi gruparea lor în tabele. Gruparea câmpurilor pe tabele se realizează prin diverse metode. Dintre aceste metode, două sunt cele mai utilizate:

- analiza concordanței IEȘIRI – INTRĂRI;
- analiza semnificației semantice a datelor.

Analiza concordanței IEȘIRI – INTRĂRI este o tehnică specifică proiectării sistemelor informatice care identifică documentele primare din care se preiau datele de intrare folosite în calculul datelor de ieșire. Aceste documente vor constitui sursele de creare și actualizare a tabelelor bazei de date.

Tehnica este utilă în cazul în care proiectantul bazei de date este familiarizat cu sistemul informațional existent.

Un indicator prezent într-o situație finală se poate obține astfel:

- prin preluarea directă din documentul primar, respectiv din tabelul bazei de date;
- prin aplicarea unui algoritm de calcul.

Tabelele se compun prin gruparea câmpurilor pe principiul apartenenței acestora la anumite documente primare care circulă în cadrul sistemului informațional.

Analiza semantică se practică atunci când proiectantul bazei de date nu are la dispoziție un set de documente primare și apare în cazul sistemelor informaționale care se proiectează pentru firmele noi.

Definirea tabelelor și relațiilor dintre tabele este etapa următoare în proiectarea bazei de date. Analiza cerințelor informaționale și a proceselor de prelucrare va conduce la identificarea datelor ce vor trebui stocate și care vor alcătui tabelele bazei de date. O tabelă va păstra datele fie despre toate caracteristicile unei colecții de date, fie numai pentru o parte dintre aceste caracteristici. Aici intervine spiritul analitic al proiectantului. Structura unei tabele este reprezentată de lista câmpurilor asociate tablei împreună cu descrierea atributelor fiecărui câmp (natură, lungime, număr de zecimale etc.). În structura unui tabel se regăsesc următoarele categorii de câmpuri:

- câmpuri de identificare (chei primare și chei condiționate);

- câmpuri tip dată calendaristică;
- câmpuri cantitativ-valorice;
- câmpuri de legătură cu alte tabele;
- câmpuri de stare care păstrează informații privind ultimele operații de prelucrare care au fost efectuate pe înregistrările din tabel.

Relațiile dintre tabele se caracterizează prin plasarea unor câmpuri comune în structura fiecăruia dintre tabelele aflate în relație directă. Pe baza acestor câmpuri, chei, fiecare sistem de gestiune a bazelor de date își construiește un mecanism propriu de accesare a înregistrărilor de date. Aceste mecanisme sunt transparente pentru utilizatorul obișnuit. Totuși este bine de reținut că nu orice câmp poate fi folosit la stabilirea unei relații, a unei legături între două tabele. Numai câmpurile de tip cheie candidat, care au proprietatea de a identifica în mod unic o înregistrare dintr-o tabelă, pot fi folosite în acest scop. Cheile candidate se mai numesc și indecși. Operația prin care se construiește sistemul de legături pentru ordonarea în vederea regăsirii înregistrărilor într-o tabelă se numește indexare. Prin indexare, fiecărei tabele principale de date i se va asocia o tabelă index corespunzătoare cheilor de indexare.

Optimizarea structurii bazei de date este un proces prin care se urmărește:

- reducerea redundanței datelor;
- eliminarea anomaliilor de actualizare.

Reducerea redundanței datelor până la un nivel minim și controlat urmărește eliminarea duplicării inutile a unor câmpuri în mai multe tabele sau eliminarea câmpurilor obținute prin calcul pe baza câmpurilor atomice. Un anumit nivel de redundanță, însă, trebuie admis pentru a nu denatura realitatea reflectată de date. De exemplu, câmpul VALOAREA_CONTRACTULUI, se calculează după relația: VALOAREA_CONTRACTULUI=CANTITATE*PRET

Nu este recomandată eliminarea acestui câmp pe considerentul că el se obține automat prin calcul.

De exemplu, în cazul în care după un anumit interval de timp, prețurile suportă o majorare globală, cum se practică foarte des, atunci automat se vor modifica și valorile contractelor încheiate anterior datei de majorare a prețurilor ori acest lucru nu este corect, contractul odată perfectat nu-și poate modifica prețul convenit prin negociere.

Anomaliile de actualizare se referă la anomaliile de ștergere, respectiv de modificare. De exemplu, se consideră o firmă care derulează lunar mii de contracte de aprovizionare, pentru un nomenclator foarte mare de produse agroalimentare, dar care operează numai cu câțiva furnizori. Dacă în tabela CONTRACTE sunt incluse alături de codul furnizorului și denumirea furnizorului, contul său bancar și denumirea băncii, atunci va apărea următoarea anomalie de actualizare, în cazul schimbării băncii și a contului bancar de către furnizor. Acest lucru necesită parcurgerea tuturor înregistrărilor în care există aceste valori și actualizarea acestora (figura 2.1).

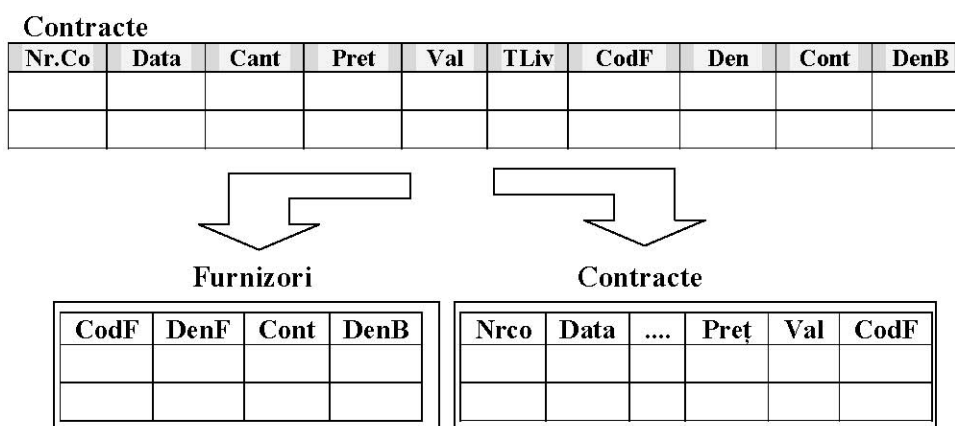


Fig. 2.1 Eliminarea anomaliilor de actualizare

Soluția este de a crea o tabelă separat, care are în structură: codul furnizorului, denumirea, contul și banca acestuia, în tabela CONTRACTE păstrându-se doar codul furnizorului ca element

de legătură, ceea ce previne pierderea de informații prin spargerea unui tabel în două. În acest fel modificarea se realizează numai asupra unei singure înregistrări.

Aplicarea procesului de normalizare

Prima formă normală: eliminarea datelor repetate

O relație este în *prima formă normală* atunci când nu conține atribute cu valori multiple (atribute multivaloare), adică atribute care au mai multe valori pentru același rând de date. Într-o relație, orice intersecție a unui rând cu o coloană trebuie să conțină *cel mult* o valoare pentru ca relația să fie în prima formă normală. Pentru transformarea relațiilor ne-normalizate în prima formă normală, trebuie mutate atributele multivaloare și grupurile repetitive în noi relații.

Procedura de mutare a unui atribut multivaloare sau a unui grup repetitiv într-o nouă relație constă în următoarele etape:

1. Creați o nouă relație, cu un nume sugestiv. Deseori, este bine să includeți numele relației originale, parțial sau în întregime, în numele noii relații.

2. Copiați identificatorul unic din prima relație în noua relație. Datele depind de acest identificator în relația originală, așa că trebuie să depindă de aceeași cheie și în noua relație. Identificatorul copiat va deveni cheie externă în noua relație.

3. Mutați grupul repetitiv sau atributul multivaloare în noua relație.

4. Formați un identificator unic în noua relație, adăugând atribute la identificatorul unic copiat din relația originală. Ca întotdeauna, asigurați-vă ca identificatorul unic nou format conține numai numărul minim de atribute necesar pentru a-l face unic. Dacă mutați un atribut multivaloare, care, în esență, este un grup repetitiv cu un singur atribut, este adăugat atributul respectiv pentru formarea identificatorului unic. Poate părea ciudat la prima vedere, dar identificatorul unic copiat din relația originală nu este doar o cheie externă, ci, de obicei, și o parte a identificatorului unic (cheia primară) a noii relații. Acest lucru este absolut normal. De asemenea, este perfect acceptabil să avem o relație în care toate atributele fac parte din identificatorul unic (adică nu există atribute care să nu facă parte din cheie).

5. Opțional, puteți să înlocuiți cheia primară cu un singur atribut surrogat pentru cheie. Dacă faceți acest lucru, trebuie să păstrați și atributele care compun cheia primară naturală, formată la pașii 2 și 4.

A doua formă normală: eliminarea dependențelor parțiale

Înainte de a explora a doua formă normală, trebuie să înțelegem conceptul *de dependență funcțională*.

Pentru această definiție, vom folosi două atribute arbitrare, inteligent denumite „A” și „B”. Atributul B este *dependent funcțional* de atributul A dacă în nici un moment nu există mai mult de o valoare a atributului B asociată cu o valoare dată a atributului A.

În relația Movie, putem să ne dăm seama cu ușurința că atributul Movie_Title este dependent funcțional de atributul Movie_ID, deoarece, în orice moment, poate exista o singură valoare Movie_Title pentru o valoare Movie_ID dată. Chiar faptul că valoarea Movie_ID definește în mod unic valoarea Movie_Title în relație înseamnă că Movie_Title este dependent funcțional de Movie ID.

Se spune că o relație este în *a doua formă normală* dacă îndeplinește următoarele criterii:

- Relația este în prima formă normală.
- Toate atributele non-cheie sunt dependente funcțional de identificatorul unic (cheia primară), *luat ca întreg*.

A doua formă normală se aplică numai relațiilor care au identificatoare unice concatenate (adică formate din atribute multiple). Într-o relație care are un singur atribut ca identificator unic, este imposibil ca un alt atribut să depindă de o parte a identificatorului unic, deoarece acesta, fiind format dintr-un singur atribut, nu are părți componente. **Ca urmare, orice relație în prima formă normală care are cheia primară formată dintr-un singur atribut este automat în a doua formă normală.**

A treia formă normală: eliminarea dependențelor tranzitive

Pentru a înțelege a treia formă normală, trebuie să înțelegem mai întâi conceptul de dependență tranzitivă.

Despre un atribut care depinde de un atribut care nu este identificator unic (cheie primară) a relației se spune că *este dependent tranzitiv*.

Uitându-ne la relația Movie, observăm că atributul Genre_Description depinde de atributul Genre_Code, iar MPAA_Rating_Description depinde de MPAA_Rating_Code. Pericolul păstrării acestor descrieri în relația Movie este faptul că, în final, cele două attribute ajung să depindă de înregistrarea unui film, ceea ce duce la toate cele trei anomalii de date prezentate mai devreme în acest capitol.

Se spune că o relație este în *a treia formă normală* dacă îndeplinește următoarele două criterii:

- Relația este în a doua formă normală.
- Nu există dependențe tranzitive (cu alte cuvinte, toate attributele non-cheie depind *numai* de identificatorul unic).

Pentru a aduce la a treia formă normală o relație aflată în a doua formă normală, mutăm attributele dependente tranzitiv în relații în care depind numai de cheia primară

Avem grijă să lăsăm atributul de care depind acestea în relația originală, cu rolul de cheie externă. Va trebui apoi să reconstruim vizualizarea originală printr-o uniune. Ca efect secundar, toate attributele ușor de calculat sunt eliminate ca încălcări ale criteriilor celei de-a treia forme normale.

De exemplu, într-o bază de date pentru vânzări, Suma Totală este obținută înmulțind Cantitatea Cumpărată cu Prețul Unitar; așa cum se observă cu ușurință, Suma Totală este dependentă de Cantitatea Cumpărată și de Prețul Unitar. Presupunând că toate cele trei attribute sunt dependente de identificatorul unic al relației care le conține, este ușor de văzut că Suma Totală (rezultatul calculat) este, de fapt, *dependentă tranzitiv* de celelalte două attribute.

PROGRAMARE ORIENTATĂ OBIECT

Avantajele programării orientate obiect

(Șoavă, G., Programarea orientată obiect, Note de curs, pag. 9-10)

Orientarea spre obiecte aduce avantaje decisive cum ar fi: modelarea obiectelor aplicațiilor, modularitatea, reutilizabilitatea și extensibilitatea codului care conduc la o mai mare productivitate, și dezvoltarea unei mari calități a aplicațiilor.

Deoarece utilizatorii doresc ca programele lor să se comporte într-o manieră fiabilă și previzibilă, este important să se înțeleagă modul în care programarea orientată spre obiecte și condusă de evenimente va contribui la corecta structurare și modularizare a programului.

Într-o manieră globală, un program orientat spre obiecte este un ansamblu de obiecte care printr-un schimb de mesaje declanșează anumite operații sau metode, facilitându-le stările lor interne și returnându-le parametrii.

Prin crearea în mod vizual a unei singure linii de cod, se obține un program funcțional care, deși banal, demonstrează toate elementele unei corecte proiectări orientate spre obiecte și conduse de evenimente.

Conceperea este fără îndoială primul domeniu în care obiectele ușurează munca. Obiectele permit reprezentarea în mod direct a entităților lumii reale și a relațiilor dintre entități. Reprezentat în mod grafic cu ajutorul grafurilor, în care nodurile reprezintă clasele iar arcele legăturile generale sau de agregare a entităților, face posibilă vizualizarea unui model de aplicație deosebit de clar.

Se definesc concepte noi, ca tip, clasă, moștenire, obținându-se o însumare a avantajelor sistemelor de gestiune a bazelor de date, care interoghează eficient datele, și a limbajelor procedurale care permit calcule complexe.