Curs 13. Modelul relational de reprezentare a bazelor de date

1. Definiția modelului relațional

Modelul relațional este o abordare a organizării datelor bazată pe tabele (relații), unde fiecare tabel conține un set de atribute și tuple (înregistrări). Relațiile dintre tabele sunt definite prin chei primare și chei străine, asigurând integritatea datelor.

Caracteristici principale ale modelului relațional:
- Datele sunt organizate în tabele bidimensionale.
- Utilizarea cheilor primare pentru a identifica unic fiecare înregistrare.
- Cheile străine definesc relațiile între tabele.
- Independența datelor față de aplicațiile care le folosesc.
- Limbajul de interogare SQL este standardizat.

2. Algebra relațională

Algebra relațională este un set de operații matematice utilizate pentru a manipula datele din bazele de date relaționale.

2.1. Operațiile fundamentale:

1. Selecția (σ): Filtrează rândurile care îndeplinesc o anumită condiție.
   - Ex.: σ (Nume='Popescu') (Student)
   - SQL echivalent: SELECT * FROM Student WHERE Nume = 'Popescu';

2. Proiecția (π): Selectează anumite coloane dintr-un tabel.
   - Ex.: π (Nume, Prenume) (Student)
   - SQL echivalent: SELECT Nume, Prenume FROM Student;

3. Reuniunea (∪): Combină două relații cu aceleași atribute.
   - Ex.: Curs1 ∪ Curs2
   - SQL echivalent: SELECT * FROM Curs1 UNION SELECT * FROM Curs2;

4. Intersecția (∩): Returnează elementele comune între două relații.
   - Ex.: Curs1 ∩ Curs2
   - SQL echivalent: SELECT * FROM Curs1 INTERSECT SELECT * FROM Curs2;

5. Diferența (−): Returnează elementele dintr-o relație care nu se află în cealaltă.
   - Ex.: Curs1 - Curs2
   - SQL echivalent: SELECT * FROM Curs1 EXCEPT SELECT * FROM Curs2;

6. Produsul cartezian (×): Combină fiecare rând din prima relație cu fiecare rând din a doua.
   - Ex.: Student × Curs
   - SQL echivalent: SELECT * FROM Student, Curs;

7. Îmbinarea (JOIN): Conectează două relații printr-o condiție comună.
   - Ex.: Student ⋈ StudentID = Inscriere.StudentID Inscriere

- SQL echivalent: SELECT * FROM Student JOIN Inscriere ON Student.ID = Inscriere.StudentID;

2.2 Studiu de caz: Utilizarea algebrei relaționale într-o companie de livrări

O companie de livrări utilizează o bază de date relațională pentru a gestiona comenzile, clienții și livratorii. Relațiile cheie sunt:

```
CREATE TABLE Client (
   ID INT PRIMARY KEY,
   Nume VARCHAR(50)
);

CREATE TABLE Comanda (
   ID INT PRIMARY KEY,
   ClientID INT,
   Status VARCHAR(20),
   FOREIGN KEY (ClientID) REFERENCES Client(ID)
);

CREATE TABLE Livrator (
   ID INT PRIMARY KEY,
   Nume VARCHAR(50)
);
```

Interogări pentru monitorizarea livrărilor:

```
SELECT Client.Nume, Comanda.Status
FROM Client
JOIN Comanda ON Client.ID = Comanda.ClientID;
```

2.3. SQL - Limbajul interogărilor

Limbajul SQL (Structured Query Language) este standardul pentru interacțiunea cu bazele de date relaționale.

2.3.1 Crearea bazelor de date și a tabelelor

```
CREATE DATABASE Universitate;
USE Universitate;

CREATE TABLE Student (
   ID INT PRIMARY KEY,
   Nume VARCHAR(50),
   Prenume VARCHAR(50),
   An INT
);

CREATE TABLE Curs (
```

```
    Cod INT PRIMARY KEY,
    Denumire VARCHAR(100)
);

CREATE TABLE Inscriere (
    StudentID INT,
    CursCod INT,
    PRIMARY KEY (StudentID, CursCod),
    FOREIGN KEY (StudentID) REFERENCES Student(ID),
    FOREIGN KEY (CursCod) REFERENCES Curs(Cod)
);
```

2.3.2 Inserarea datelor

```
INSERT INTO Student (ID, Nume, Prenume, An) VALUES (1, 'Popescu', 'Ion', 2);
INSERT INTO Curs (Cod, Denumire) VALUES (101, 'Baze de date');
INSERT INTO Inscriere (StudentID, CursCod) VALUES (1, 101);
```

2.3.3 Interogarea bazelor de date

```
-- Selectarea tuturor studenților
SELECT * FROM Student;

-- Selectarea studenților de anul 2
SELECT * FROM Student WHERE An = 2;

-- Afișarea cursurilor la care este înscris un student
SELECT Curs.Denumire
FROM Curs
JOIN Inscriere ON Curs.Cod = Inscriere.CursCod
WHERE Inscriere.StudentID = 1;
```

3. Integritatea datelor

Integritatea datelor asigură corectitudinea și consistența acestora în cadrul bazei de date. Tipurile principale sunt:

1. Integritatea entităților: Fiecare tabel trebuie să aibă un identificator unic.
2. Integritatea referențială: Relațiile dintre tabele trebuie să fie menținute corect prin chei primare și străine.
3. Integritatea domeniului: Valorile din coloane trebuie să respecte tipul de date definit.

4. Normalizarea bazelor de date

4.1. Introducere în normalizarea bazelor de date

Normalizarea este procesul de organizare a datelor într-o bază de date pentru a minimiza redundanța și dependențele nedorite. Aceasta implică aplicarea mai multor **forme normale (NF)**, fiecare având un set de reguli specifice care îmbunătățesc structura bazei de date.

Obiectivele normalizării:

1. Eliminarea redundanței datelor.
2. Îmbunătățirea integrității și consistenței bazei de date.
3. Creșterea eficienței interogărilor și a actualizărilor datelor.

4.2. Prima formă normală (1NF)

Reguli:

1. Fiecare coloană trebuie să conțină **doar valori atomice** (nu liste sau seturi de valori).
2. Fiecare coloană trebuie să conțină **doar un singur tip de date**.

Exemplu de tabel neconform cu 1NF:

```
CREATE TABLE Student (
   ID INT PRIMARY KEY,
   Nume VARCHAR(50),
   Telefoane VARCHAR(100)
);
```

În această structură, coloana Telefoane poate conține mai multe numere de telefon pentru un singur student, ceea ce încalcă principiile 1NF.

Tabel conform cu 1NF:

```
CREATE TABLE Telefon (
   StudentID INT,
   Numar VARCHAR(15),
   FOREIGN KEY (StudentID) REFERENCES Student(ID)
);
```

4.3. A doua formă normală (2NF)

Reguli:

1. Baza de date trebuie să fie în **1NF**.
2. Toate atributele necheie trebuie să depindă în întregime de cheia primară.

Exemplu de tabel neconform cu 2NF:

```
CREATE TABLE Inscriere (
   StudentID INT,
   CursCod INT,
   Profesor VARCHAR(50),
   PRIMARY KEY (StudentID, CursCod)
);
```

În acest exemplu, Profesor nu depinde de cheia compusă (StudentID, CursCod), ci doar de CursCod.

Tabel conform cu 2NF:

```
CREATE TABLE Curs (
   Cod INT PRIMARY KEY,
   Denumire VARCHAR(100),
   Profesor VARCHAR(50)
);

CREATE TABLE Inscriere (
   StudentID INT,
   CursCod INT,
   PRIMARY KEY (StudentID, CursCod),
   FOREIGN KEY (CursCod) REFERENCES Curs(Cod)
);
```

4.4. A treia formă normală (3NF)

Reguli:

1. Baza de date trebuie să fie în **2NF**.
2. Nicio coloană non-cheie nu trebuie să depindă tranzitiv de cheia primară.

Exemplu de tabel neconform cu 3NF:

```
CREATE TABLE Student (
   ID INT PRIMARY KEY,
   Nume VARCHAR(50),
   Oras VARCHAR(50),
   CodPostal VARCHAR(10)
);
```

În această structură, CodPostal depinde de Oras, nu de ID, ceea ce încalcă regula 3NF.

Tabel conform cu 3NF:

```
CREATE TABLE Oras (
   ID INT PRIMARY KEY,
   Nume VARCHAR(50),
   CodPostal VARCHAR(10)
);

CREATE TABLE Student (
   ID INT PRIMARY KEY,
   Nume VARCHAR(50),
   OrasID INT,
   FOREIGN KEY (OrasID) REFERENCES Oras(ID)
);
```

4.5 Forma normală Boyce-Codd (BCNF)
Reguli:

1. Baza de date trebuie să fie în **3NF**.
2. Fiecare determinant trebuie să fie o cheie candidat.

Exemplu de tabel neconform cu BCNF:

```
CREATE TABLE Angajat (
   ID INT PRIMARY KEY,
   Departament VARCHAR(50),
   SefDepartament VARCHAR(50)
);
```

În această structură, Departament determină SefDepartament, dar nu este o cheie primară.
Tabel conform cu BCNF:

```
CREATE TABLE Departament (
   ID INT PRIMARY KEY,
   Nume VARCHAR(50),
   SefID INT,
   FOREIGN KEY (SefID) REFERENCES Angajat(ID)
);

CREATE TABLE Angajat (
   ID INT PRIMARY KEY,
   Nume VARCHAR(50),
   DepartamentID INT,
   FOREIGN KEY (DepartamentID) REFERENCES Departament(ID)
);
```

4.6. Studii de caz aplicate pentru normalizare

Studiu de caz 1: Gestionarea vânzărilor într-un magazin online
Problema:
Un magazin online stochează detalii despre produse și comenzi într-un tabel unic, generând redundanță și anomalii la actualizare.
Soluție:

- Crearea unui tabel Produs care conține informații unice despre produse.
- Crearea unui tabel Comanda care conține informații despre fiecare comandă.
- Crearea unui tabel ComandaProdus pentru relația dintre comenzi și produse.

Studiu de caz 2: Sistem de gestionare a studenților
Problema:
Un sistem universitar păstrează informațiile studenților și ale cursurilor într-un singur tabel, cauzând inconsistențe.
Soluție:

- Separarea entităților Student, Curs și Inscriere în tabele distincte conform regulilor normalizării.

4.7. Scenarii reale de normalizare a bazelor de date

Scenariul 1: Sistem de gestionare a comenzilor într-un magazin online

Descriere:
Un magazin online gestionează comenzile clienților într-un singur tabel, ceea ce duce la redundanță și probleme de actualizare.
Tabel inițial (Neconform cu 1NF, 2NF și 3NF):

```
CREATE TABLE Comenzi (
   ID INT PRIMARY KEY,
   Client VARCHAR(50),
   Produs VARCHAR(100),
   Cantitate INT,
   Pret DECIMAL(10,2),
   AdresaLivrare VARCHAR(255)
);
```

Probleme:

1. Coloana Produs permite mai multe produse în aceeași celulă (încalcă 1NF).
2. Pret depinde de Produs, nu de ID (încalcă 2NF).
3. AdresaLivrare depinde de Client, nu de cheia primară (încalcă 3NF).

Tabele normalizate:

```
CREATE TABLE Client (
   ID INT PRIMARY KEY,
   Nume VARCHAR(50),
   AdresaLivrare VARCHAR(255)
);

CREATE TABLE Produs (
   ID INT PRIMARY KEY,
   Nume VARCHAR(100),
   Pret DECIMAL(10,2)
);

CREATE TABLE Comanda (
   ID INT PRIMARY KEY,
   ClientID INT,
   DataComanda DATE,
   FOREIGN KEY (ClientID) REFERENCES Client(ID)
);

CREATE TABLE ComandaProdus (
   ComandaID INT,
   ProdusID INT,
   Cantitate INT,
```

```
    PRIMARY KEY (ComandaID, ProdusID),
    FOREIGN KEY (ComandaID) REFERENCES Comanda(ID),
    FOREIGN KEY (ProdusID) REFERENCES Produs(ID)
);
```

Explicație:

- Client conține informații despre clienți.
- Produs stochează detalii despre produse și prețuri.
- Comanda înregistrează comenzile plasate de clienți.
- ComandaProdus leagă comenzile de produsele comandate, eliminând redundanța.

Scenariul 2: Sistem de evidență a studenților și cursurilor

Descriere:
O universitate gestionează înscrierile studenților la cursuri într-un tabel neconform cu regulile normalizării.
Tabel inițial (Neconform cu 1NF și 2NF):

```
CREATE TABLE Inscriere (
    Student VARCHAR(50),
    Curs VARCHAR(100),
    Profesor VARCHAR(50),
    Sala VARCHAR(20)
);
```

Probleme:

1. Profesor depinde doar de Curs, nu de întregul rând (încalcă 2NF).
2. Sala depinde de Curs, nu de Student (încalcă 3NF).

Tabele normalizate:

```
CREATE TABLE Student (
    ID INT PRIMARY KEY,
    Nume VARCHAR(50)
);

CREATE TABLE Curs (
    ID INT PRIMARY KEY,
    Denumire VARCHAR(100),
    Profesor VARCHAR(50),
    Sala VARCHAR(20)
);

CREATE TABLE Inscriere (
    StudentID INT,
    CursID INT,
    PRIMARY KEY (StudentID, CursID),
    FOREIGN KEY (StudentID) REFERENCES Student(ID),
```

```
    FOREIGN KEY (CursID) REFERENCES Curs(ID)
);
```

Explicație:

- Student și Curs sunt entități separate, evitând redundanța datelor.
- Inscriere leagă studenții de cursuri prin referințe, menținând integritatea datelor.

Scenariul 3: Sistem de gestionare a rezervărilor la un hotel
Descriere:
Un hotel stochează toate rezervările într-un singur tabel care include informații despre clienți, camere și durata ședерii.
Cerințe pentru normalizare:

1. Crearea unui tabel Client pentru a reține informațiile clienților.
2. Crearea unui tabel Camera pentru a stoca detalii despre camere.
3. Crearea unui tabel Rezervare pentru a înregistra fiecare rezervare și a evita redundanțele.

Scenariul 4: Sistem de gestionare a angajaților și departamentelor într-o companie
Descriere:
O companie păstrează într-un tabel unic date despre angajați, departamente și salarii, ceea ce duce la redundanță și anomalii la actualizare.
Cerințe pentru normalizare:

1. Separarea angajaților și departamentelor în tabele distincte.
2. Crearea unui tabel Plata pentru a înregistra salariile și a menține consistența.

Scenariul 5: Sistem de gestionare a pacienților într-un spital
Descriere:
Un spital înregistrează pacienți, doctori și tratamente într-un singur tabel, ceea ce creează probleme de integritate a datelor.
Cerințe pentru normalizare:

1. Crearea unui tabel Pacient pentru a stoca datele pacienților.
2. Crearea unui tabel Doctor pentru informațiile despre medici.
3. Crearea unui tabel Consultatie care să lege pacienții, doctorii și tratamentele administrate.

Referințe și resurse web

1) Connolly, T. & Begg, C. (2014). **Database Systems: A Practical Approach to Design, Implementation, and Management**. Pearson.
2) Silberschatz, A., Korth, H., & Sudarshan, S. (2020). **Database System Concepts**. McGraw-Hill.
3) Elmasri, R., & Navathe, S. (2016). **Fundamentals of Database Systems**. Pearson.
4) W3Schools SQL Tutorial
5) MongoDB Documentation
6) PostgreSQL Documentation
7) Redis Documentation