Curs 14. Proiectarea bazelor de date relaționale

1. Procesul de proiectare

Proiectarea unei baze de date relaționale implică mai multe etape esențiale care asigură structura optimă a datelor și eficiența utilizării acestora.

1.1 Etapele proiectării bazelor de date

1. **Identificarea cerințelor** - Colectarea cerințelor utilizatorilor și analiza acestora.
2. **Modelarea conceptuală** - Crearea unui model abstract al bazei de date folosind diagrame ER (Entity-Relationship).
3. **Modelarea logică** - Transformarea modelului conceptual într-un model relațional.
4. **Modelarea fizică** - Optimizarea modelului logic pentru implementare în SGBD.
5. **Normalizarea** - Aplicarea regulilor pentru a elimina redundanța și asigura integritatea datelor.

Exemplu de **diagramă ER** pentru o bază de date universitară:
Entități: Student, Profesor, Curs
Relații: Student se înscrie la Curs, Profesor predă Curs

1.2 Exemplu de modelare conceptuală

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

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

CREATE TABLE Curs (
   Cod INT PRIMARY KEY,
   Denumire VARCHAR(100),
   ProfesorID INT,
   FOREIGN KEY (ProfesorID) REFERENCES Profesor(ID)
);

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

2. Integritatea datelor în proiectare

Pentru a asigura consistența bazei de date, sunt utilizate mai multe tipuri de constrângeri de integritate:

1. **Cheia primară (PRIMARY KEY)** - Asigură unicitatea fiecărei înregistrări.
2. **Cheia străină (FOREIGN KEY)** - Asigură referențialitatea corectă între tabele.
3. **Constrângeri CHECK** - Restricționează valorile acceptate într-o coloană.
4. **NOT NULL** - Impune ca un atribut să nu fie gol.
5. **UNIQUE** - Asigură valori unice pentru un atribut.

Exemplu de aplicare a integrității datelor:

```
ALTER TABLE Student ADD CONSTRAINT chk_email CHECK (Email LIKE '%@%.%');
```

3. Studii de caz despre proiectarea bazelor de date

3.1. Sistem de gestionare a comenzilor în e-commerce

O companie de comerț electronic trebuie să gestioneze produse, clienți și comenzi.

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

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

3.2. Sistem bancar - Proiectare bază de date pentru conturi

```
CREATE TABLE ContBancar (
   ID INT PRIMARY KEY,
   Titular VARCHAR(50),
   Sold DECIMAL(15,2) CHECK (Sold >= 0)
);

CREATE TABLE Tranzactie (
   ID INT PRIMARY KEY,
   ContID INT,
   Suma DECIMAL(15,2),
   Tip VARCHAR(10) CHECK (Tip IN ('Depunere', 'Retragere')),
   DataTranzactie DATE,
   FOREIGN KEY (ContID) REFERENCES ContBancar(ID)
);
```

3.4 Optimizarea bazei de date

Pentru a îmbunătăți performanța unei baze de date relaționale, se folosesc diferite tehnici de optimizare:

1. **Indexarea** - Crearea de indexuri pentru căutări rapide.
2. **Partiționarea tabelelor** - Împărțirea unei tabele mari în părți mai mici pentru acces mai rapid.
3. **Denormalizarea parțială** - Îmbunătățirea performanței prin eliminarea anumitor reguli de normalizare în cazuri specifice.
4. **Utilizarea procedurilor stocate** - Optimizarea execuției interogărilor frecvente.

Exemplu de creare a unui index pentru îmbunătățirea căutărilor:

```
CREATE INDEX idx_nume_client ON Client(Nume);
```

3.5 Scenarii reale de proiectare a bazelor de date

Scenariul 1: Sistem de gestionare a rezervărilor pentru un hotel
Descriere:
Un hotel dorește să își gestioneze rezervările printr-o bază de date care să permită urmărirea camerelor disponibile, a rezervărilor făcute de clienți și a detaliilor despre clienți.
Modelul relațional propus:

```
CREATE TABLE Client (
   ID INT PRIMARY KEY,
   Nume VARCHAR(50),
   Telefon VARCHAR(15),
   Email VARCHAR(50)
);
```

```
CREATE TABLE Camera (
   ID INT PRIMARY KEY,
   Tip VARCHAR(50),
   Pret DECIMAL(10,2),
   Disponibilitate BOOLEAN
);

CREATE TABLE Rezervare (
   ID INT PRIMARY KEY,
   ClientID INT,
   CameraID INT,
   DataStart DATE,
   DataEnd DATE,
   FOREIGN KEY (ClientID) REFERENCES Client(ID),
   FOREIGN KEY (CameraID) REFERENCES Camera(ID)
);
```

Explicație:

- Tabelul Client reține informațiile despre clienți.
- Tabelul Camera păstrează detalii despre camere și disponibilitatea acestora.
- Tabelul Rezervare conectează clienții cu camerele rezervate.
- Se utilizează chei străine pentru a menține integritatea datelor.

Exemplu de interogare:

```
SELECT C.Nume, R.DataStart, R.DataEnd, Cam.Tip
FROM Rezervare R
JOIN Client C ON R.ClientID = C.ID
JOIN Camera Cam ON R.CameraID = Cam.ID
WHERE Cam.Disponibilitate = TRUE;
```

Această interogare returnează toate rezervările active cu detaliile clienților și tipurile de camere rezervate.

Scenariul 2: Sistem de evidență a împrumuturilor dintr-o bibliotecă

Descriere:

O bibliotecă dorește un sistem care să monitorizeze împrumuturile cărților și să asigure că fiecare carte este returnată la timp.

Modelul relațional propus:

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

CREATE TABLE Carte (
   ID INT PRIMARY KEY,
```

```
    Titlu VARCHAR(100),
    Autor VARCHAR(100),
    ExemplareDisponibile INT
);

CREATE TABLE Imprumut (
    ID INT PRIMARY KEY,
    CititorID INT,
    CarteID INT,
    DataImprumut DATE,
    DataReturnare DATE,
    FOREIGN KEY (CititorID) REFERENCES Cititor(ID),
    FOREIGN KEY (CarteID) REFERENCES Carte(ID)
);
```

Explicație:

- Cititor conține informații despre cititori.
- Carte include titlul, autorul și numărul de exemplare disponibile.
- Imprumut leagă cititorii cu cărțile împrumutate și datele asociate acestora.

Exemplu de interogare:

```
SELECT C.Nume, B.Titlu, I.DataImprumut, I.DataReturnare
FROM Imprumut I
JOIN Cititor C ON I.CititorID = C.ID
JOIN Carte B ON I.CarteID = B.ID;
```

Această interogare returnează detaliile despre împrumuturile active din bibliotecă.

Scenariul 3: Sistem de gestionare a angajaților într-o companie

Descriere:

O companie dorește să își gestioneze angajații, împreună cu departamentele și salariile acestora.

Cerințe pentru implementare:

1. Crearea unui tabel Angajat care să conțină ID-ul, numele, poziția și salariul fiecărui angajat.
2. Crearea unui tabel Departament pentru a organiza angajații în echipe specifice.
3. Crearea unui tabel Plata pentru a înregistra salariile plătite fiecărui angajat.

Scenariul 4: Sistem de gestionare a comenzilor într-un restaurant

Descriere:

Un restaurant are nevoie de o bază de date pentru a monitoriza comenzile clienților și preparatele din meniu.

Cerințe pentru implementare:

1. Crearea unui tabel Client care să rețină detalii despre clienți.

2. Crearea unui tabel Comanda care să stocheze comenzile făcute de clienți.
3. Crearea unui tabel Preparat care să conțină lista de produse disponibile.
4. Crearea unui tabel ComandaPreparat pentru a lega comenzile cu preparatele comandate.

Scenariul 5: Sistem de gestionare a pacienților într-un spital
Descriere:

Un spital are nevoie de un sistem care să țină evidența pacienților internați, a doctorilor și tratamentelor administrate.

Cerințe pentru implementare:

1. Crearea unui tabel Pacient pentru a înregistra informațiile personale ale pacienților.
2. Crearea unui tabel Doctor pentru a stoca detalii despre doctori.
3. Crearea unui tabel Internare care să înregistreze perioada internării pacienților.
4. Crearea unui tabel Tratament care să urmărească medicamentele administrate pacienților.

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