Curs 12. Tipuri de baze de date

Concepte ale bazelor de date

1. Definiții și caracteristici ale bazelor de date

Bazele de date sunt fundamentale pentru gestionarea eficientă a informațiilor în era digitală. Comparativ cu sistemele tradiționale bazate pe fișiere, bazele de date oferă multiple avantaje, inclusiv reducerea redundanței, îmbunătățirea integrității și acces facil la date. Componentele unui sistem de baze de date – hardware, software, date, utilizatori și proceduri – colaborează pentru a asigura o administrare eficientă și sigură a datelor. Pe măsură ce tehnologiile evoluează, bazele de date devin din ce în ce mai complexe, dar și mai eficiente în gestionarea volumelor mari de informații.

1.1. Ce este o bază de date?

O bază de date este o colecție organizată de date care sunt stocate electronic și structurate într-un mod care permite accesul, gestionarea și actualizarea eficientă a acestora. Bazele de date sunt utilizate în aproape toate domeniile, inclusiv afaceri, sănătate, educație și cercetare, datorită capacității lor de a gestiona volume mari de informații și de a permite acces rapid la date relevante. Bazele de date sunt construite pe un model specific, cel mai utilizat fiind modelul relațional, care organizează datele în tabele interconectate. Alte modele includ modelul ierarhic, modelul rețea și modelul orientat pe obiecte. Prin utilizarea unui Sistem de Gestionare a Bazelor de Date (SGBD), utilizatorii pot crea, interoga și manipula date fără a avea nevoie de cunoștințe avansate de programare.

O bază de date trebuie să îndeplinească mai multe caracteristici esențiale:

- ✓ **Persistența datelor** – Datele sunt stocate permanent și nu se pierd după oprirea sistemului.
- ✓ **Consecvența** - Datele trebuie să respecte anumite reguli de integritate pentru a asigura validitatea acestora.
- ✓ **Concurența** - Mai mulți utilizatori pot accesa și modifica datele simultan fără a genera erori sau conflicte.
- ✓ **Securitatea** - Accesul la date este controlat prin permisiuni și mecanisme de autentificare.
- ✓ **Independența datelor** - Modificările structurale nu afectează aplicațiile care utilizează baza de date.

1.2. Diferența dintre baze de date și fișiere tradiționale

Înainte de apariția bazelor de date moderne, datele erau gestionate în sisteme bazate pe fișiere tradiționale. Aceste sisteme aveau limitări semnificative, iar bazele de date au fost dezvoltate pentru a rezolva aceste probleme.

1.2.1. Sisteme de fișiere tradiționale

În sistemele tradiționale bazate pe fișiere, datele sunt stocate în fișiere text sau binare, gestionate de aplicații software specifice. Fiecare aplicație trebuie să implementeze propriile metode de acces și manipulare a datelor.

1.2.2. **Limitări ale sistemelor tradiționale de fișiere:**

1. **Redundanță și inconsistență** - Aceleași date pot fi stocate în mai multe fișiere, ducând la inconsistențe și consum excesiv de spațiu.
2. **Lipsa integrității** - Nu există mecanisme automate pentru a asigura corectitudinea și validitatea datelor.
3. **Dificultăți în accesul la date** - Accesul la date necesită dezvoltarea de programe personalizate pentru fiecare tip de interogare.
4. **Lipsa securității** - Nu există mecanisme eficiente pentru restricționarea accesului la anumite date.
5. **Dificultăți în partajarea datelor** - Accesul simultan la fișiere de către mai mulți utilizatori este dificil de gestionat.

1.3. Bazele de date moderne

Bazele de date elimină majoritatea acestor probleme prin utilizarea unui sistem centralizat de gestionare a datelor. Acestea permit accesul eficient la informații, reduc redundanța și oferă mecanisme avansate de securitate și integritate.

1.4.**Avantajele bazelor de date față de fișierele tradiționale:**

1. **Eliminarea redundanței** - Datele sunt normalizate și stocate într-un mod care minimizează duplicările.
2. **Îmbunătățirea integrității** - Se pot defini constrângeri care asigură coerența și validitatea datelor.
3. **Acces flexibil la date** - Utilizatorii pot efectua interogări complexe folosind limbaje precum SQL.
4. **Securitate sporită** - Se pot implementa niveluri diferite de acces pentru utilizatori.
5. **Partajare eficientă a datelor** - Mai mulți utilizatori pot accesa simultan baza de date fără a afecta integritatea informațiilor.

1.5. Componentele unui sistem de baze de date

Un sistem de baze de date este compus din mai multe elemente esențiale care asigură funcționarea și gestionarea eficientă a datelor. Aceste componente includ hardware, software, date, utilizatori și proceduri.

1. Hardware

Hardware-ul reprezintă infrastructura fizică necesară pentru operarea unei baze de date. Acesta include:

- ✓ Servere și stații de lucru
- ✓ Dispozitive de stocare (SSD, HDD, unități cloud)
- ✓ Rețele de comunicație
- ✓ Procesoare și memorie RAM pentru optimizarea performanței

2. Software

Software-ul este componenta care gestionează și controlează baza de date. Principalele elemente includ:

- ✓ **Sistemul de Gestionare a Bazelor de Date (SGBD)** - Software care permite crearea, manipularea și interogarea bazelor de date. Exemple de SGBD-uri: MySQL, PostgreSQL, Oracle, Microsoft SQL Server.
- ✓ **Sistemul de operare** - Asigură interacțiunea dintre hardware și software-ul bazei de date.
- ✓ **Aplicațiile utilizatorilor** - Programe care interacționează cu baza de date pentru a furniza servicii specifice.

3. Date

Datele reprezintă componenta centrală a oricărui sistem de baze de date. Acestea pot fi:

- ✓ **Date brute** - Informațiile primare stocate în tabele.
- ✓ **Metadate** - Informații despre structura bazei de date, precum tipurile de date și relațiile dintre tabele.
- ✓ **Indexuri** - Structuri utilizate pentru a accelera căutările și interogările în baze de date mari.

4. Utilizatori

Utilizatorii bazei de date pot fi clasificați în mai multe categorii:

- ✓ **Administratorii bazei de date (DBA)** - Responsabili pentru gestionarea și întreținerea bazei de date.
- ✓ **Dezvoltatorii de aplicații** - Creează software care interacționează cu baza de date.
- ✓ **Utilizatori finali** - Accesează baza de date prin aplicații sau interfețe specifice pentru a introduce sau interoga date.

5. Proceduri și politici

Procedurile și politicile definesc regulile și bunele practici pentru utilizarea bazei de date. Acestea includ:

- ✓ **Reguli de securitate** - Politici de acces și autentificare.
- ✓ **Proceduri de backup și recuperare** - Planuri pentru protejarea datelor împotriva pierderii accidentale.
- ✓ **Norme de utilizare** - Standardele stabilite pentru gestionarea datelor și interacțiunea utilizatorilor cu sistemul.

2. Tipuri de baze de date

Bazele de date pot fi clasificate în mai multe categorii în funcție de modelul de organizare și stocare a datelor. Fiecare tip are avantaje și dezavantaje specifice, iar alegerea unui model depinde de cerințele aplicației și de volumul de date gestionat.

2.1. Baze de date ierarhice

Definiție:
Bazele de date ierarhice organizează datele într-o structură de tip arbore, unde fiecare înregistrare are un singur părinte și poate avea mai mulți copii. Relațiile sunt de tipul părinte-copil.
Exemplu:
Un exemplu clasic este IBM Information Management System (IMS), folosit în instituții financiare și guvernamentale.
Avantaje:

- ✓ Acces rapid la date pentru interogări predefinite
- ✓ Structură clară și organizată

Dezavantaje:

- ✓ Lipsa flexibilității (dificil de modificat structura)
- ✓ Complexitate în gestionarea relațiilor între date

Exemplu 1 - Structura unei baze de date pentru o organizație:

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

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

Inserare date de test:

```
INSERT INTO Departament (ID, Nume) VALUES (1, 'IT'), (2, 'HR');
INSERT INTO Angajat (ID, Nume, DepartamentID) VALUES (1, 'Ion Popescu', 1), (2, 'Maria Ionescu', 2);
```

Interogare pentru angajații unui departament:

```
SELECT Angajat.Nume FROM Angajat
JOIN Departament ON Angajat.DepartamentID = Departament.ID
WHERE Departament.Nume = 'IT';
```

Exemplu 2 - Structura pentru o bază de date educațională:

```
CREATE TABLE Scoala (
   ID INT PRIMARY KEY,
   Nume VARCHAR(100)
```

```
);

CREATE TABLE Clasa (
   ID INT PRIMARY KEY,
   Nume VARCHAR(50),
   ScoalaID INT,
   FOREIGN KEY (ScoalaID) REFERENCES Scoala(ID)
);
```

Inserare date de test:
```
INSERT INTO Scoala (ID, Nume) VALUES (1, 'Liceul National'), (2, 'Colegiul Tehnic');
INSERT INTO Clasa (ID, Nume, ScoalaID) VALUES (1, '10A', 1), (2, '11B', 2);
```

Interogare pentru clasele dintr-o anumită școală:
```
SELECT Clasa.Nume FROM Clasa
JOIN Scoala ON Clasa.ScoalaID = Scoala.ID
WHERE Scoala.Nume = 'Liceul National';
```

2.2. Baze de date rețea

Definiție: Bazele de date de tip rețea permit relații multiple între înregistrări, spre deosebire de structura ierarhică strictă.

Exemplu: Modelul CODASYL este un exemplu de bază de date de rețea, folosit pentru aplicații complexe de gestionare a datelor.

Avantaje:
- ✓ Mai multă flexibilitate în interconectarea datelor
- ✓ Performanță crescută pentru structuri complexe

Dezavantaje:
- ✓ Necesită administrare complexă
- ✓ Mai dificil de învățat și utilizat

Exemplu 1 - Structura pentru un sistem de bibliotecă:
```
CREATE TABLE Carte (
   ID INT PRIMARY KEY,
   Titlu VARCHAR(100)
);

CREATE TABLE Autor (
   ID INT PRIMARY KEY,
   Nume VARCHAR(100)
);

CREATE TABLE Carte_Autor (
   CarteID INT,
   AutorID INT,
```

```
    PRIMARY KEY (CarteID, AutorID),
    FOREIGN KEY (CarteID) REFERENCES Carte(ID),
    FOREIGN KEY (AutorID) REFERENCES Autor(ID)
);
```

Inserare date de test:

```
INSERT INTO Carte (ID, Titlu) VALUES (1, 'Programare în C'), (2, 'Algoritmi și structuri de date');
INSERT INTO Autor (ID, Nume) VALUES (1, 'John Smith'), (2, 'Jane Doe');
INSERT INTO Carte_Autor (CarteID, AutorID) VALUES (1, 1), (2, 2);
```

Interogare pentru cărțile unui autor:

```
SELECT Carte.Titlu FROM Carte
JOIN Carte_Autor ON Carte.ID = Carte_Autor.CarteID
JOIN Autor ON Carte_Autor.AutorID = Autor.ID
WHERE Autor.Nume = 'John Smith';
```

2.3. Baze de date relaționale (RDBMS)

Definiție: Bazele de date relaționale utilizează tabele pentru stocarea datelor, iar relațiile dintre acestea sunt definite prin chei primare și chei străine.
Exemplu: Sisteme populare: MySQL, PostgreSQL, Oracle, SQL Server.
Avantaje:

- ✓ Ușurință în utilizare și interogare cu SQL
- ✓ Scalabilitate și integritate a datelor

Dezavantaje:

- ✓ Performanță redusă pentru date foarte mari sau nestructurate
- ✓ Necesită schema rigidă

2.4. Baze de date NoSQL

Definiție: Bazele de date NoSQL sunt concepute pentru a gestiona date nestructurate și semi-structurate, fără o schemă rigidă.

Tipuri de baze de date NoSQL:

- ✓ **Bazele de date orientate pe documente:** Stochează datele sub formă de documente JSON, BSON (ex. MongoDB, CouchDB).
- ✓ **Bazele de date de tip key-value:** Stochează perechi cheie-valoare (ex. Redis, DynamoDB).
- ✓ **Bazele de date de tip columnar:** Optimizate pentru interogări analitice pe seturi mari de date (ex. Apache Cassandra, HBase).
- ✓ **Bazele de date graf:** Model specializat pentru date interconectate (ex. Neo4j, ArangoDB).

Avantaje:

- ✓ Scalabilitate orizontală mare

- ✓ Flexibilitate în modelarea datelor
- ✓ Performanță ridicată pentru date mari și distribuție geografică

Dezavantaje:

- ✓ Lipsa standardizării interogărilor
- ✓ Unele implementări nu oferă suport pentru tranzacții complexe

2.5. Compararea tipurilor de baze de date

Tip de bază de date	Structură	Scalabilitate	Flexibilitate	Cazuri de utilizare
Ierarhică	Arbore	Redusă	Scăzută	Sistemele bancare vechi
Rețea	Grafic	Medie	Medie	Aplicații complexe
Relațională	Tabelară	Scalabilitate limitată	Medie	CRM, ERP, eCommerce
NoSQL	Diversă	Mare	Mare	Big Data, IoT, social media

3. Baze de date NoSQL

Bazele de date NoSQL sunt o categorie modernă de sisteme de gestionare a bazelor de date care elimină limitările sistemelor relaționale pentru a permite o stocare și o interogare mai eficiente a datelor nestructurate și semi-structurate.

3.1. Utilizări comune

- ✓ **Big Data**: MongoDB, Cassandra
- ✓ **Caching și sesiuni web**: Redis, Memcached
- ✓ **Aplicații distribuite**: DynamoDB, CouchDB
- ✓ **Rețele sociale și grafică de date**: Neo4j, ArangoDB

3.2. Avantajele bazelor de date NoSQL

- ✓ **Scalabilitate orizontală:** Pot gestiona volume mari de date prin distribuire pe mai multe servere.
- ✓ **Flexibilitate:** Nu impun o schemă strictă a datelor, ceea ce permite schimbări rapide.
- ✓ **Performanță ridicată:** Se adaptează bine la cerințele aplicațiilor moderne, unde latența redusă este esențială.

1.3.3 Dezavantajele bazelor de date NoSQL

- ✓ **Lipsa unui standard unificat:** Fiecare sistem are propriul model de interogare și administrare.
- ✓ **Consistența datelor:** Majoritatea bazelor NoSQL sacrifică consistența în favoarea disponibilității și performanței (modelul CAP).

Exemplu 1 - MongoDB (bază de date orientată pe documente):

```
{
   "_id": ObjectId("60c72b2f5f1b2c6d88fdf301"),
   "nume": "Maria Ionescu",
   "email": "maria@example.com",
   "adresa": {
      "oras": "București",
      "strada": "Bd. Unirii"
   }
}
```

Interogare MongoDB:

```
{ "nume": "Maria Ionescu" }
```

Exemplu 2 - Redis (bază de date key-value):

```
SET user:1000 "Ion Popescu"
GET user:1000
```

Proiectarea unei baze de date NoSQL:

Proiectarea unei baze de date NoSQL trebuie să țină cont de cerințele aplicației. De exemplu, pentru o aplicație de rețele sociale, un model de bază de date NoSQL ar trebui să permită stocarea rapidă a mesajelor și a relațiilor între utilizatori fără a impune constrângerile unei baze de date relaționale.

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