Baze de date

Utilizarea Sistemelor de Operare

Paul Irofti

Universitatea din București
Facultatea de Matematică și Informatică
Department de Informatică
Email: paul.irofti@fmi.unibuc.ro

Date

- adesea programele conțin algoritmi care procesează date
- modul în care sunt citite datele de intrare și scrise datele de ieșire ale unui algoritm au un impact direct asupra timpului de execuție
- date de intrare
  - trebuie găsite pe disc
  - aduse în memorie
  - parcurse de program
  - traduse în structurile proprii de date
  - trimise mai departe algoritmului
- date de ieșire
  - parcurse de program
  - traduse în formatul de pe disk
  - scrise pe disk
- avem nevoie de o structură eficientă a datelor

Structură

- frecvent datele au în spate o structură intrinsecă
- reprezentarea structurată aduce multe beneficii
- stocare – cu cât știm mai multe informații despre date, cu atât le putem împărții și organiza eficient pe disk
- parcurgere
  - extragem strict informația structurală care ne interesează
  - traversare rapidă
  - nu este necesar să încărcăm în memorie tot setul de date
- căutare eficientă
  - cu cât structura este mai laborioasă cu atât putem căuta mai eficient după informații specifice
  - citim mimul de informație necesară verdictului: găsit sau nu
  - exemple tehnici: indexare, caching, hashing
- reprezentare – generare de rapoarte, grafice, tabele care prezintă o idee, o tendință, pe care vrem să o evidențiem

Modelul entitate-asociere

- o entitate este o chestie care poate fi identificată în mod unic
- exemple: persoană, casă, mașină, artist
- o entitate are unul sau mai multe atribute
- exemplu: presoană – sex, vârstă, educație, ocupație
- o entiate are una sau mai multe relații cu alte entități
- exemplu: o persoană are doi părinți, patru bunici, câțiva copii
- o relație poate avea de asemenea una sau mai multe atribute
- exemplu: o relație de prietenie are o dată de început și una de sfârșit
- gramatical entitea este substantivul, reația verbul și atributul adjectivul sau adverbul

Exemplu: Sistemul de coordonate carteziene

- entitatea este punctul
- atributele sale sunt coordonatele în fiecare dimensiune
- relațiile între puncte formează formele geometrice

Tabele

- modul cel mai des întâlnit de structurare a datelor: linii și coloane
- fiecare tabelă reprezintă o entitate (ex. studenți)
- fiecare linie reprezintă o intrare, un element de acel tip (ex. studentul Alex)
- fiecare coloană reprezintă un atribut al entității (ex. nume, grupă, serie, note)
- relațiile reprezentate cu ajutorul coloanelor: o coloană poate indica către o altă intrare dintr-un tabel

Exemplu tabele: relație artist-melodie

ID	Artist	ID Melodie
1	Filarmonica George Enescu	512
13	Metallica	365
42	Mile Davis	88

ID	Melodie
88	So What
365	Frantic
512	Mahler 4

Modelul cheie-valoare

- nu există entități
- doar chei (atribute) și valori
- exemple sintaxă: `cheie = valoare`, `cheie :  valoare`
- folosit pentru fișiere de configurare (ex. `/etc/hosts`)

Format INI

Format fișier tip cheie-valoare

- proprietăți `cheie=valoare`
- opțional separate în secțiuni `[section-name]`
- linile comentate încep cu `;`
- cele mai răspândite în Windows
- folosit de PHP (ex. `php.ini`)

```
; modificat la data de 01.05.2018
[student]
prenume=Alex
nume=Alexandrescu
grupa=151

[uso]
laborator=32
proiect=15
verificare=25
```

Format CSV

Comma-separated values

- stochează date tabulare în fișiere text
- fiecare linie din fișier reprezintă o intrare în tabel
- valoarea de pe fiecare coloană este separată prin virgulă
- toate intrările trebuie să aibă același număr de proprietăți

Tabela artist devine

```
1,Filarmonica George Enescu,512
13,Metallica,365
42,Mile Davis,88
```

Format XML

- format text de stocare a informațiilor
- urmează modelului entitate-asociere
- folosit pentru transportul datelor în rețea
- tag – desemnează începutul și sfârșitul unei entități
- element – conținutul, intrarea
- atribut – definește o proprietate a entității; se află în interiorul tag-ului
- formatul HTML este un subset al formatului XML

Exemplu XML

```
<artists>
<artist>
        <id>1</id>
        <name>Filarmonica George Enescu</name>
        <melodie>512</melodie>
</artist>
<artist>
        <id>13</id>
        <name>Metallica</name>
        <melodie>365</melodie>
</artist>
<artist>
        <id>42</id>
        <name>Mile Davis</name>
        <melodie>88</melodie>
</artist>
</artists>
```

Serializare

- transformarea datelor într-un format de stocat pe disk
- formatul poate fi unul dintre cele descrise mai sus sau altele
- aspecte importante
  - portabilitate – formatul să fie standardizat pentru ca datele să fie accesate și de alte programe
  - stocare eficientă – alegerea unui format cât mai succint pentru a minimiza spațiul de disk
  - transport – format text sau binar în funcție de cum se transmit datele
- operația inversă se numește deserializare

Baze de date relaționale

- soluții software de stocare a datelor
- urmează modelul entitate-asociere
- rezolvă problema stocării: propriu format de fișier de stocare
- serializare și deserializare transparentă
- asigură integritatea stocării și manipulării datelor
- datele sunt organizate în tabele
- soluțiile folosesc un limbaj comun: Structured Query Language (SQL)
  - expresii – produc valori în tabele sau tabele
  - clauze – specifică condițiile în care să se aplice comanda
  - predicate – verifică valoarea de adevăr a unei expresii
  - interogare – extrage date în funcție de criteriile date

SQL: baze de date

- creare: `CREATE DATABASE DatabaseName;`
- afișare: `SHOW DATABASES;`
- selectare: `USE DatabaseName;`
- eliminare: `DROP DATABASE DatabaseName;`
- exemplu:

```
CREATE DATABASE AristMelodie;
USE ArtistMelodie;
```

SQL: tabele

- creare:

```
CREATE TABLE TableName(
    column1 datatype,
    column2 datatype,
    column3 datatype,
    .....
    columnN datatype,
    PRIMARY KEY( one or more columns )
);
```

- afișare: `DESC TableName;`
- eliminare: `DROP TABLE TableName;`
- exemplu:

```
CREATE TABLE artist(
    ID int,
    artist varchar(255),
    ID_melodie int,
    PRIMARY KEY(ID)
);
```

SQL: Adăugare date

- comanda:

```
INSERT INTO TableName (column1, column2, ...columnN)
       VALUES (value1, value2, ...valueN);
```

- corespondență coloană-valoare
- nu trebuie specificate coloanele dacă adăugăm valori în toate
- ordinea este importantă în ambele cazuri
- exemplu:

```
INSERT INTO artist (ID, artist, ID_meldie)
       VALUES (1, Filarmonica George Enescu, 512);
```

SQL: interogare

- comanda:

```
SELECT column1, column2, columnN
  FROM TableName
  WHERE [condition];
```

- clauza `WHERE` este opțională
- exemplu: `SELECT ID,artist FROM artist;`

```
+----+-----------------------------+
| ID | NAME                        |
+----+-----------------------------+
|  1 | Filarmonica George Enescu   |
| 13 | Metallica                   |
| 42 | Mile Davis                  |
+----+-----------------------------+
```

- `WHERE` conține expresii logice care ajută filtrarea
  - `WHERE artist='Metallica'`
  - `WHERE ID < 10`
  - `WHERE artist='Metallica' AND ID_melodie='88'`

SQL: modificare date

- actualizare:

```
UPDATE TableName
  SET column1 = value1 ,... columnN = valueN
  WHERE [condition];
```

- eliminare:

```
DELETE FROM TableName
  WHERE [condition];
```

SQL: utilizatori

- nou:
  - mysql: `CREATE USER 'jeffrey'@'localhost' IDENTIFIED BY 'password';`
  - psql: `$ createuser joe`
  - sqlite: nu are utilizatori
- roluri:

```
CREATE [OR REPLACE] ROLE [IF NOT EXISTS] role
   [WITH ADMIN
   {CURRENT_USER | CURRENT_ROLE | user | role}]
```

- acordarea drepturilor se face cu comanda `GRANT`; sintaxă specifică soluției software
- exemplu MySQL drepturi depline

```
GRANT ALL ON *.* TO 'admin'@'localhost'
 IDENTIFIED BY 'password'
 WITH GRANT OPTION;
```

Dump

- salvarea informațiilor dintr-o bază de date se face diferit în funcție de soluția software
- mysql: `$ mysqldump DatabaseName > dbname.sql`
- psql: `$ pg_dump DatabaseName > dbname.sql`
- sqlite:

```
sqlite> USE DatabaseName;
sqlite> .output dbname.sql
sqlite> .dump
sqlite> .quit
```

- rezultatul final este un fișier text cu o înșiruire de comenzi SQL care crează baza de date, tabelele, și adaugă datele în tabele