Database Systems

Introduction

H. Turgut Uyar Şule Öğüdücü

2002-2016

License

Topics

Problem

- store and process large amounts of data effectively
- add new data
- change existing data
- delete data
- query data: planned - ad hoc
- CRUD: create - read - update - delete

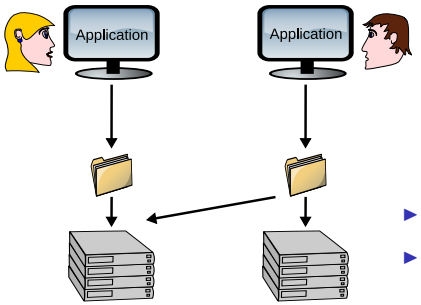
Data

- persistent data: data that must be stored due to the nature of the information
- temporary data
- output data: data that can be derived from persistent data (query results, reports, etc.)
- input data: unprocessed data that just entered the system

Example: University student data

- Student Affairs: student name, number, department, courses taken, internships, ...
- Library: student name, number, department, books lent, ...
- common data: student name, number, department, ...
- application specific data: courses, internships, books, ...

Record Files



- every application has its own data
- every application keeps its data in the files that it manages itself

Redundancy

- same data kept in multiple places
- waste of disk space

example

- names, numbers and departments of students are kept both in Student Affairs and in the Library

Inconsistency

- multiple copies of the same data can become different

example

- the name of the same student can be recorded
  as "Victoria Adams" in Student Affairs and
  as "Victoria Beckham" in the Library

Integrity

- it is difficult to keep the data correct

example

- "Control and Computer Engineering" department is closed
  but the department data of its students remains the same

Duplicated Work

- a lot of work must be duplicated for every new application

example

- a new application will be developed for the Scholarship Office

Policy Gaps

- no standards in the applications of the institution
- different paradigms, methods, programming languages
- data transfer between applications
- each department considers only its own requirements

Security

- hard to define detailed security permissions
- security depends only on the operating system

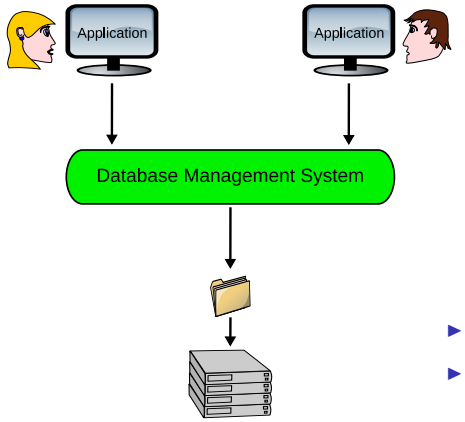
Data Dependence

- data dependence: application code depends on the organization of the data and the access method
- hard to make changes in the code

Data Dependence Example

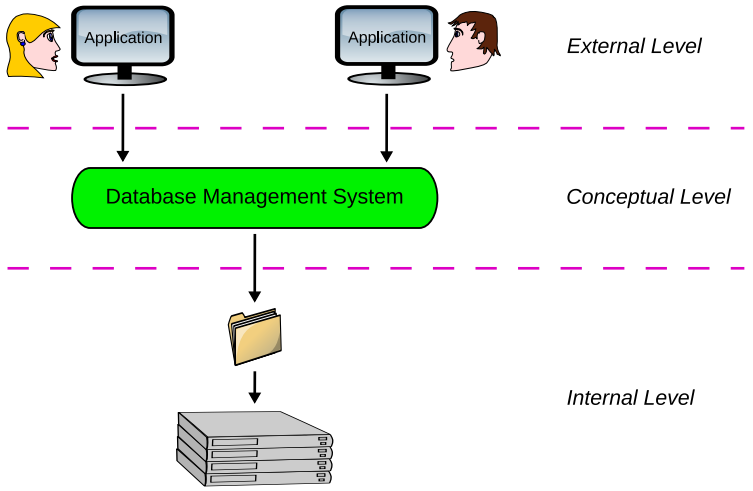
- student number is a string in Student Affairs but a number in the Library
- Student Affairs application keeps a B-tree index on the student number
- B-tree search algorithms are used for queries
- what if we decide to switch to a hashed index?

Database Management Systems



- data is kept in a shared system
- applications access data over a common interface

ANSI/SPARC Architecture



External Level

- external level from the end user's perspective:
- data needed by that end user
- interface of the application

- external level from the application programmer's perspective:
- programming language
- database extensions to this language: data sublanguage

Conceptual Level

- conceptual level: the entire data
- where data independence is achieved

- catalogue: definitions that describe the data
- databases
- data types, integrity constraints
- users, privileges, security constraints

Internal Level

- internal level: implementation details

- how the data is represented
- files, records

- how the data is accessed
- pointers, indexes, B-trees

Conversions

- conversions between levels for data independence

example: conceptual - external

- present the student number
  as a string to the Student Affairs application, and
  as a number to the Library application

example: conceptual - internal

- generate an index on the student number

Administrator Roles

- data administator: makes the decisions
- which data will be stored?
- who can access which data?

- database administrator: applies the decisions
- defines the conceptual - external/internal conversions
- adjusts system performance
- guarantees system availability

DBMS Functions

- data definition language
- data manipulation language

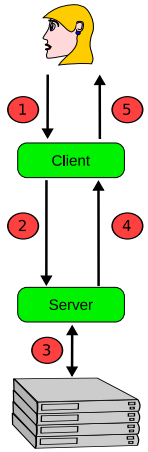
- checking data manipulation requests for security constraints
- checking data manipulation requests for integrity constraints
- processing simultaneous requests properly
- performance

Client / Server Architecture

- server: provides the DBMS functions

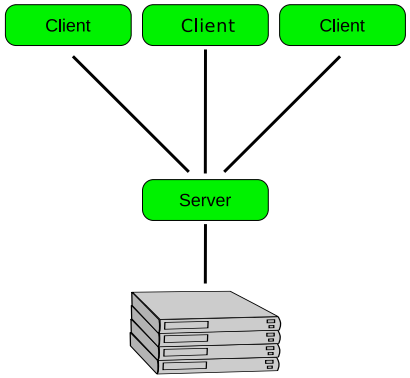
- client: provides the interaction between the user and the server
- vendor supplied tools (query processors, report generators, ...)
- applications developed by application programmers

Architecture



- client and server can be on the same computer or on different computers

Multiple Clients / Single Server



- many clients can connect to a single server
- server is a bottleneck
- replication

SQL

- Structured Query Language
- data definition language
- data manipulation language
- interaction with general purpose programming languages

- started by IBM in the 1970s
- standards: 1992, 1999, 2003

SQL Products

- Oracle, IBM DB2, MS-SQL, ...
- open source: PostgreSQL, MySQL, ...
- embedded: SQLite, ...

References

Required Reading: Date

- Chapter 1: An Overview of Database Management
  - 1.4. Why Database?
  - 1.5. Data Independence
- Chapter 2: Database System Architecture