CMSC 206: Database Management Systems

Fundamental DBMS concepts

Thomas Laurent

07/09/2019

University of the Philippines Open University

Table of contents

Introduction

Introduction

This week we will *define* what a DataBase Management System (DBMS) is. We will discuss *what* they are and *why* we use them. We will see the *abstraction levels* offered by a DBMS . And finally we will see *examples* of modern DBMSes.

Take home points

Definition of DB and DBMS
DBMS levels of abstraction
ACID properties
Types of DBMS

Database and Database Management System

Database - Definition

A DB is a collection of **related** data, facts about a particular world or process, with **meaning**, a **purpose** and **target users**.

- Very broad definition
- Data takes many forms: numbers, text, images, sounds, video, ...
- Any random collection of data is **not** a DB

Figure 1: Classic representation of a database: three stacked cylinders

Database - Examples

- Personal address book containing one's contacts' information.
  Data: name, address, phone number
- A university's student records.
  Data: name, address, phone number, birth date, grades, ...
- A bank's records.
  Data: name, address, phone number, account balance, transactions, ...
- An online retailer's catalog.
  Data: product name, description, price, image, ...

DataBase Management System - Definition

A DBMS is a software system that enables users to create, maintain and exploit a database. It allows users to:

Define the DB: specify the structure of the database, the data that will be stored, its type and constraints. This constitutes the **meta-data** of our data.

Construct the DB: Insert the data in the database and store it durably

Manipulate the DB: Retrieve and update data from the database

Share the database: Allow multiple users to use the database concurrently

DataBase Management System - Examples

- Microsoft Access, SQL Server, MySQL, PostgreSQL, ...
- Redis, Amazon DynamoDB, ...
- MongoDB, Couchbase, ...
- Neo4J, Datastax Enterprise Graph, ...

- Spreadsheet software (e.g. Excel, LibreOffice calc, Google sheets, etc) are not a DBMS

DataBase System

Database + Database Management System = Database System

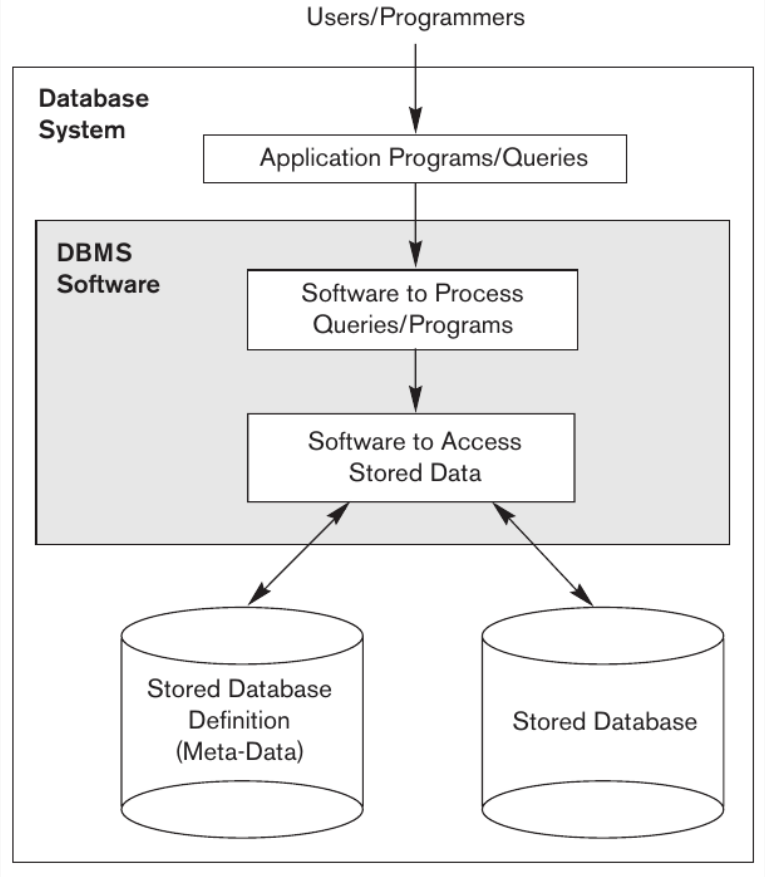

Figure 2: A simplified database system[1]

[1] Figure 1.1 Fundamentals of Database Systems, Elmasri and Navathe

DB without DBMS

Storing data with a file system - Example

If a DB is just a coherent collection of data, why do we need a DBMS? Why not just keep our data in files that we format in an ad-hoc way and organise through the folder tree and file names for example?

If we take the previous example of a university, why not have:

- The student records system work with */school/program/year/student_name_grades.csv* files where each line would record a class taken and the grade obtained
- The financial office's system work with */school/program/year/student_name_fees.csv* files where each line would record a transaction on the account of the student
- The PR system work with */school/program/year.csv* files where each line would record a student name and their contact info

Storing data with a file system - Example

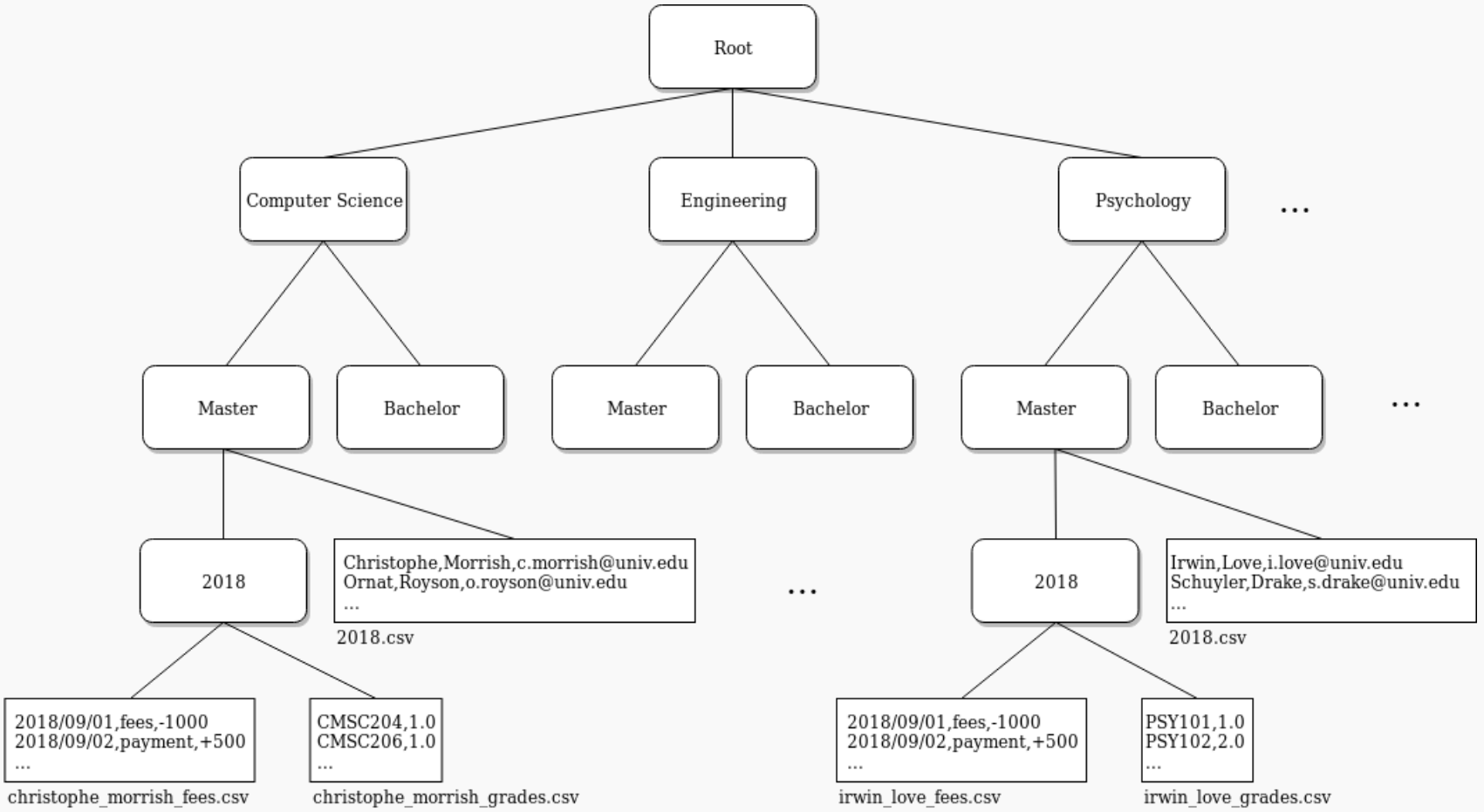

Figure 3: Example of a school managing everything through the file system

Storing data with a file system - Drawbacks

This example shows several drawbacks of the file system approach:

Data redundancy: student_name appeared in 3 different places. We are wasting storage.

Risk of inconsistency: what happens if I am recorded as Thomas in my student records but as Tomas in the financial system?
What happens if my PC crashes while I am modifying/saving a file?

No concurrency control: what happens if two people modify my fees records at the same time? Will one's modification override someone else's?

Low flexibility: the format of the data can not be easily changed. Similarly, we can only use ad-hoc programs to process the data as there are no standard formats.

We want ACID

What we are missing, and that DBMS provide to different degrees, is the ACID properties:

Atomicity: Every operation, or group of operations - called *transaction* - should be completely executed or not be executed at all. i.e. one or nothing principle, we can not take money from a bank account and not credit it somewhere else.

Consistency: Every transaction on the DB will bring it from a valid state to another valid state with regards to the constraints put on the data.

Isolation: If transactions are executed concurrently on the DB they will have the same effect as if they were run in sequence, and not interfere with one another.

Durability: Once a transaction is committed, it will stay committed, even if after a system failure.

Abstraction through the DBMS

Levels of abstraction - Definition

To allow us to achieve these ACID properties, the DBMS provides different abstractions. There are three levels of abstraction offered by the DBMS, which should work independently. i.e. we could change the way we implement one level without impacting the others.

Physical level: how the data is stored on disk. e.g. file formats, encoding, etc. Usually handled 100% by the DBMS

Logical/Conceptual level: What data is being stored? What is the type of the data? How are different data items related (e.g. the ”lecturer” item in the record for a class must be a ”person” record) ? Who can access what part of the data?

View level: The details of the data types are hidden. Different views are created depending on your accessible data. Your boss may view your salary but you may not view his for example.

Levels of abstraction

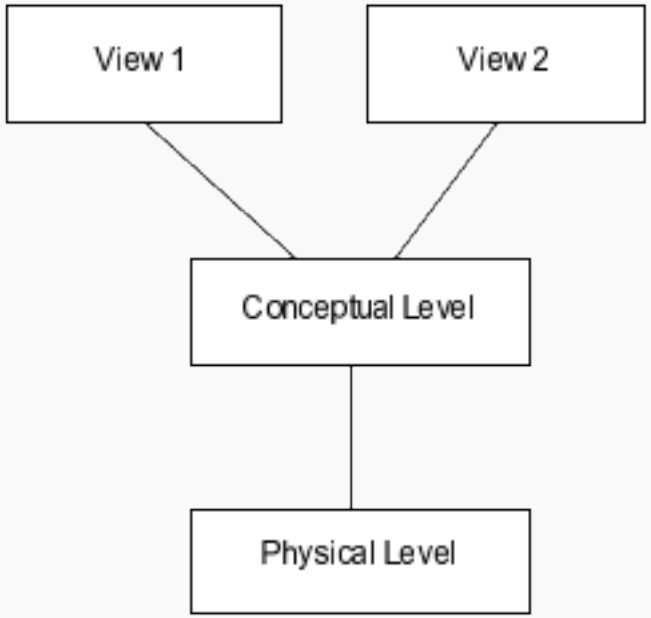

Figure 4: The three levels of abstraction

Each level is a *higher level* abstraction of the data. Each level is of concern to different people. The DBMS sysadmin is interested in how the files are saved to disk. A programmer building something on top of the DB is interested in the data types, but not in how the data is actually stored. The end user is just interested in browsing the data.

Different DBMS types

There are different types of DBMS that structure the data differently.

Relational (RDBMS): the focus of this course. The data is very structured into tables and columns with strong constraints on them. ACID properties are strongly enforced

Object oriented (OODB): simplifies the interaction between Object-Oriented programming languages and relational databases. Objects can be saved into and retrieved directly from the DB.

NoSQL: Emerged with big data. Much faster than RDBMS but ACID is sometimes relaxed. e.g. Key/Value stores, column stores, document stores, graph databases, ...