Introduction to Databases and Data Modelling

Oliver Gregory

2024-10-01

Table of contents

1 Basic Concepts

1.1 File based approach

A file based approach does not require database software. Instead, each user defines and implements the files needed for a specific application. This approach has several disadvantages:

- Duplicated data.
- Separation and isolation.
- Data is physically dependent on the application.

1.2 Databases

1.2.1 What is a database?

Data is a collection of known **facts** that can be recorded and that have implicit meaning.

No reason to keep data that is not related to what we are doing.

A database is a collection of **related** and **logically coherent** data, built for a specific purpose.

1.2.2 How do we design a database?

1.2.2.1 Start with a miniworld description

> A UNIVERSITY database for maintaining information about students, courses, and grades. Each student is referred by their name, unique number, class and major. Each course has a name, a unique code, the number of credits or hours, and the department to which it belongs. Some courses can have another course as a prerequisite. Courses are offered within sections, and each section contains a number, the course number, the semester, the year and the instructor's name. The students receive a grade report containing all the grades they completed across sections.

This is a high-level conceptual model where you identify the main entities, relationships, rules and constraints.

1.2.2.2 Simplify and identify

- Subjects and nouns → relations.
- Objects → attributes/keys.
- Predicates → relationships.
- Numerals → cardinalities.

1.2.2.3 Example

A university database for maintaining information about students, courses, and grades. Each student is referred by their name, unique number, class and major. Each course has a name, a unique code, the number of credits or hours, and the department to which it belongs. Some courses can have another course as a prerequisite. Courses are offered within sections, and each section contains a number, the course number, the semester, the year and the instructor's name. The students receive a grade report containing all the grades they completed across sections.

1.2.2.4 Schema diagram

Visual representation of the structure and organisation of the database. It shows how the elements relate to each other.

1.2.2.5 Database snapshot

A static image of the database at a given point in time, capturing the state of the data (or instances).

> **Note**
>
> You cannot have a database table that is does not have at least one foreign key to connect it to the other tables. The foreign keys should come from the miniworld description.

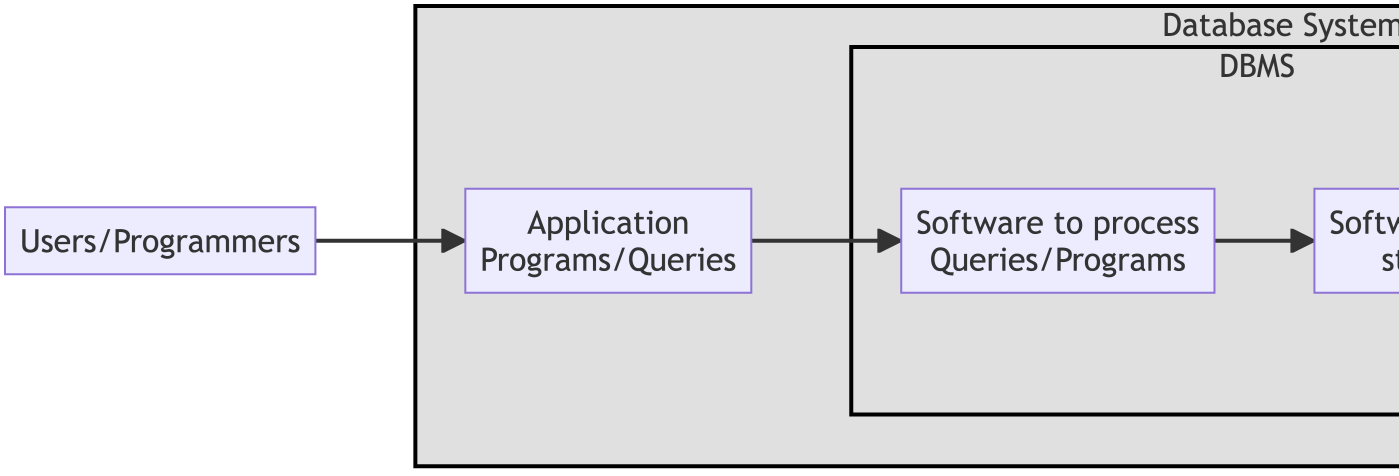
1.3 Database management systems

1.3.1 What is a DBMS?

The DBMS is a *general-purpose software system* that facilitates the processes of defining, constructing, manipulating, and sharing databases among users and applications.

To qualify as a database system, the software must be able to:

- Find (**retrieve**) data.
- Add (**insert**) new data.
- **Delete** unwanted data.
- Change (**update**) data.



1.3.2 Characteristics of a DBMS

1.3.2.1 Metadata

A complete definition (or description) of the database structure and constraints.

The metadata is used by the DBMS software and the users to obtain information about the database structure.

NoSQL databases do not need to rely on metadata: they can use self-described data (data item + data values).

1.3.2.2 Program-data independence and data abstraction

A DBMS provides users with a conceptual representation of data that does not include details on how the data is stored or how the operations are implemented.

Changes in the structure of the database do not affect the applications that use it.

1.3.2.3 Mutliple views of the data

A view may be a subset of the database and can contain virtual data that is derived from the database files but is not explicitly stored.

If a view is accessed frequently by multiple users, it can be stored as a virtual table.

1.3.2.4 Multiuser transaction processing and data sharing

A multiuser DBMS must allow multiple users to access the database at the same time.

The DBMS must include concurrency control routines to ensure that several users trying to update the same data do so in a controlled manner => support for online transaction processing (OLTP).

The concept of a transaction is a key element for ensuring ACID properties.

Atomicity	Consistency	Isolation	Durability
Transactions are all or nothing.	Only valid data is saved.	Transactions do not affect each other.	Written data will not be lost.

1.3.3 Database Actors

1.3.3.1 Database steward/curator

- Administering the organisation's data.
- Applying policies and standards.
- Ensuring data quality.

1.3.3.2 Database administrator (DBA)

- Managing the database and DBMS.
- Authorising access.
- Monitoring and coordinating the use.
- Acquiring software and hardware.
- Ensuring security and performance.

1.3.3.3 Database designers

- Identifying data to be stored.
- Defining data structures.
- Eliciting requisites with database users.
- Designing the database, including views.

1.3.3.4 Database users

Accessing the database for querying, updating and generating reports.

They can be:

- Casual end users: high-level query interface.
- Parametric end users: frequent use of standard types of queries and updates (e.g., mobile banking).
- Sophisticated end users: familiarise themselves with DBMS facilities to design complex applications (e.g. engineers).

1.3.4 Three-Schema Architecture

Three-schema architecture aims to separate user applications from the physical database.

Schemas can be defined at three levels:

- External view: uses a set of external schemas to describe part of the database that a specific group of users is interested in. Representational data models can be used to implement external views.
- Conceptual schema: describes entities, data types, relationships, user operations and constraints. A representational data model is used to describe the conceptual schema for a community of users.
- Internal schema: physical data model describing details of data storage and access paths for the database.

Mappings are used to transform requests and results between levels.

Data independence: the capacity to change the schema at one level of the database system without having to change the schema at the next higher level.

It can be **logical** (changes in the conceptual schema do not affect external views or applications) or **physical** (changes in the internal schema do not affect the conceptual schema).

1.3.5 Modules and interactions

Data definition language (DDL): used by the database administrator to define the database structure and schema. (Example: CREATE TABLE)

Data manipulation language (DML): used by database users to query and update the database. It can be interactive such as through a terminal or embedded in a python script for example. (Example: SELECT)

1.3.6 Architecture of a DBMS

1.3.6.1 Centralised architecture

DBMS software is installed on a single computer (e.g. mainframes), which acts as the server. Users access the server through terminals or workstations. (ATM machines, for example).

1.3.6.2 Client-server architecture

With 2 tier architecture, the application runs on the client machine and the DBMS and data are on the server machine.

With 3 to n tier architecture, there are intermediary layers between the client and the server such as those that authorise access and manage transactions, etc.

1.3.6.3 Distributed architecture

With distributed architecture, the database is distributed across multiple machines. These machines serve a part of the database while being a client to the other machines that serve a different part of the database.

Most NoSQL databases and cloud databases are distributed.