Principles of Databases

Introduction

David Sinclair

Overview

Texts

Recommended:

- *Fundamentals of Database Systems*, Elmasri & Navathe, Addison-Wesley (Main Module Text)

Supplementary:

- *An Introduction to Database Systems*, C.J. Date, Addison-Wesley

What is a Database?

- A **database** is a collection of *related* data.
  - The data represents recorded facts of some aspect of the real world. This limited view of the real-world is called the **miniworld** or the **universe of discourse (UoD)**.
  - The data collected in the database has some *inherent* meaning.
- The database is built for use by an intended set of users with some predefined applications.
- The size and complexity of a database can very from a small number of records with a simple structure to massive databases with millions of records with a complex internal structure.
- A database can be generated and *manipulated* manually or by a computer-based system.
  - Manipulated $\Rightarrow$ store/retrieve/update/delete

Database Management Systems (DBMS)

- A **database management system (DBMS)** is a set of programs that allow a user to create and manipulate a database.
- DBMS = database (DB) + DBMS
- It provides an abstract view of the data in the database and describes, logically, how the data is stored, accessed and modified.
- The **schema** of a database defines the structure of the database in a formal language that is supported the the DBMS.
- A **DBMS catalog** contains a complete description of the structure of each database in the DBMS and the constraints on the data in each database.
- A **DB view** is a subset of the DB and/or data derived from the DB.

Program-Data Independence

- A DBMS provides the user with an abstract representation of the data. This representation is separate from:
  - how the data is stored; and
  - how the data is accessed.
- We can change how the data is physically stored without changing the operations that access the data as these operation use the logical data model.
- We can modify the structure of the data being stored, by adding additional data in each record or removing data from every record, and we will not need to modify the access operations that do not refer to the newly added or removed data.
- This characteristic of a DBMS is called **program-data independence**.

Information Systems & DBMS

Where does a DMBS fit into an Information System?

- Interactive query (i.e adhoc or canned) or host query (i.e. SQL in a host language)
- Facilitates unambiguous statement (data meets requirements)
- Facilitates precise query (i.e. comes back with single piece of data)
- Retrieved data is specifically stored or aggregated (in SQL views)
- Query is Boolean combination of predicates (i.e relation between entities)
- Exact matching of conditions (i.e select ... where in SQL)
- Provides a formal schema (structure of data not expected to change)

Information Systems & DBMS (2)

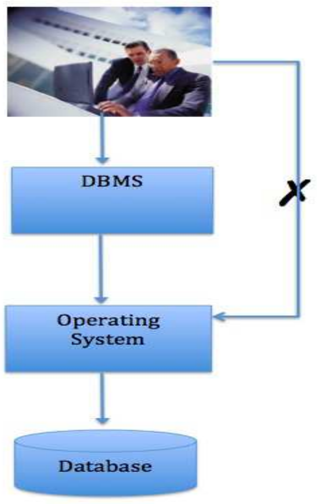
A DBMS also provides:

- Security (i.e no unauthorised access)
- Data independence
- Persistence (i.e complex data and schemas can be stored on termination)
- Concurrency (i.e updates by users visible by multiple users simultaneously)
- recovery and backup (i.e transfer of copied files from one location to another with operations on them)

DBMS Data

- A DBMS requires data that describes and enforces structure and constraints on the data.
- Has to allow for the DBMS running on a range of platforms.
- Can have single/multi-user, with shared access and integrity of data must be maintained.
- Users are concerned with overlapping subsets of total data (i.e. data perceived by different users in different ways)
    - Example: A university's data consists of:
        - Students have views consisting of . . .
        - Library has views consisting of . . .
        - Finance has views consisting of . . .
- An inherent feature of DBMS data is that is shared.

DBMS Software



- A DBMS is an application program that sits between the user(s) and the data.
- The DBMS handles all interactions between the user(s) and the data.
- The DBMS shields users from each other and from unauthorised access to the database.

DBMS Users

- *Database Admnistrators (DBA)* are the systems managers of the database and are responsible for:
  - the database itself;
  - the DBMS and related subsystems;
  - authorising access to the database;
  - monitoring the database's usage and performance;
  - updates; and
  - breaches of security
- *Database Designers* are responsible for:
  - talking to prospective customers to understand their requirements; and
  - identifying the data to be stored in the database, and choosing the appropriate structures to represent the data.

DBMS Users (2)

- *System Analysts* and *Application Programmers*
  - Analyse the requirements of end-users and develop "canned" transactions to meet these requirements.
  - Typically write C, Java, PHP etc. with embedded SQL commands.
  - Run online or in batch mode
- End-users
  - Casual/Occasional
    - Only use the database occasionally and the queries may be different each time.
    - Typically middle/high-level managers using a high-level language.
  - Naïve/Canned Transactions
    - Constant querying of the database.
    - Typically bank tellers, reservation staff for airlines/trains, etc.

DBMS Users (3)

- End-users (contd.)
  - Sophisticated
    - Need to thoroughly familiarise themselves with the DBMS facilities.
    - Typically engineers, scientists, etc. with complex requirements.
  - Stand-alone
    - Maintain personal database using “off the shelf” (COTS) software with menus or GUIs.
    - Example is a user of a tax package for their own personal use.

The “Behind the Scene” DBMS Workers

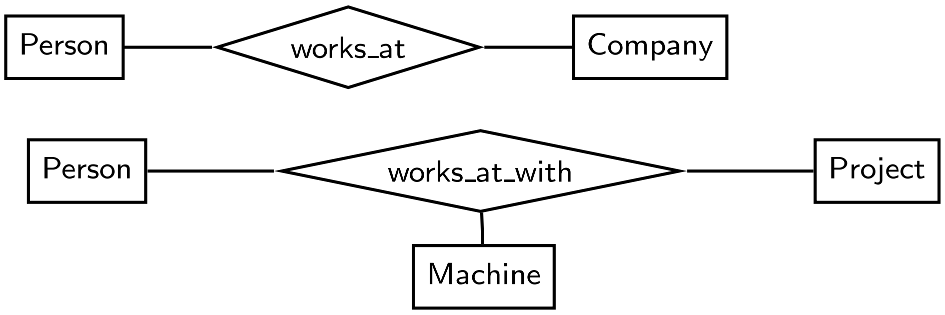
- These are typically associated with design, development and operation of the DBMS software and system environment.
- DBMS Designers and Developers
  - Design and implement DBMS modules/interfaces as software packages.
  - These modules implement catalog functions, query language, data access and security.
- Tool Developers
  - Develop optional software packages to facilitate database system design and use.
  - For example, Schema design, performance monitoring and optimisation, GUIs, etc.
- Operation and Maintenance Personnel
  - Responsible for the actual running and maintenance of hardware and software environment of the DBMS system.

DBMS Data Example: Entities and Relationships

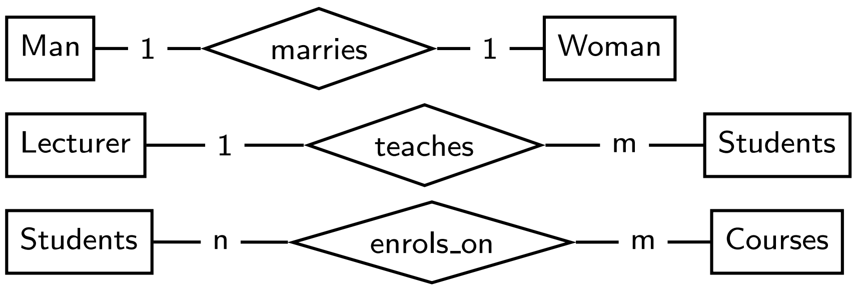
- A DBMS is used by any reasonably self-contained organisation (from a single individual to a large corporation) that wants to manage a large volume of information.
- For example: A university
  - Students, Lecturers, Courses, Modules, Books, Faculties, Schools, Lectures
  - All are *entities* or distinguishable objects in the real world.
  - There are *relationships* between real world *entities*
    - Schools **make up** Faculties
    - Schools **have** Students
    - Students **attend** Lectures **given by** Lecturers
    - Lectures **are part of** Modules
    - Modules **are part of** Courses
    - Courses **belong to** Schools
    - Students **borrow** Books
    - Lecturers **recommend** Books
    - Courses **can be composed of other** Courses

DBMS Data Example: Entities and Relationships (2)

- Most real-world relationships are binary, involving only 2 entities, but some are ternary involving 3 participating entities.



- Entity types may be linked in more than one way.



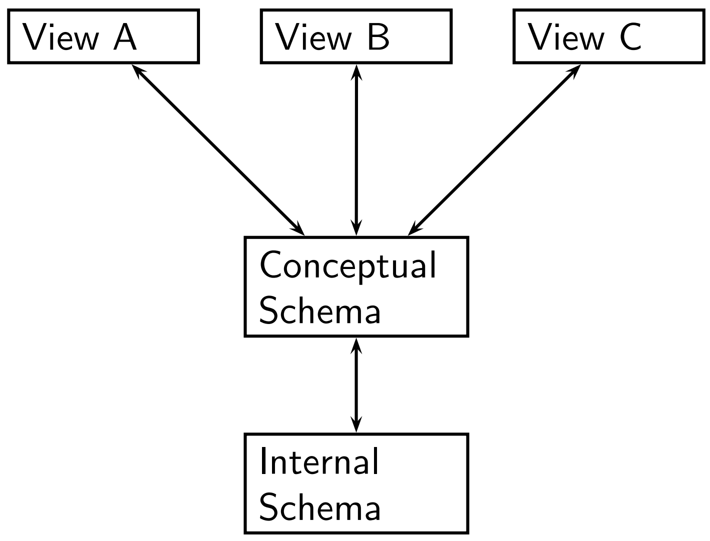
Some Definitions

- View
  - A user's perspective of the database that may be a subset of the database or contain virtual data derived from the database but not stored in it.
  - Example: Students, library, finance have different views of the database.
- Entity
  - A real-world object or concept.
- Attribute
  - A property of interest that further describes an entity.
- Relationship
  - A relationship between 2 or more entities represents an interaction between them.
  - Example: `works_on` between `employee` and `project`.

Why use a DBMS?

- The logical organisation gives a clear picture and helps the programmers develop application programs faster.
- Handles the low-level file maintenance.
- Centralising the data has the following advantages:
  - Redundancy is avoided.
  - Inconsistency is avoided.
  - Data is shared.
  - Standards are enforced (e.g. naming and data representation standards)
  - Security is applied.
  - *Referential integrity* ensures that relationships between tables remain consistent and ensures that changes made to the linked table are reflected in the primary table.
- Can provide *data independence* where the data organisation is not built into the application programs.
- A database administrator can change the structure of the database without the users noticing, except with respect to performance.

Three Level Architecture of a DBMS



External Level:

- User view
- Defined by user or programmer in consultation with DBA

Conceptual Level:

- Defined by DBA

Internal Level:

- Defined by DBA for optimisation

3 Level Architecture: External Level

- Users use a language incorporating:
  - *Data Definition Language* (DDL) for defining the External Schema.
  - *Data Manipulation Language* (DML) for data insertion/update/retrieval/deletion.
- An individual user's view is an external view. It can include multiple occurrences of multiple types of external records.
- Each view describes the part of the individual user is interested in, hiding the rest of the data.
- Views are defined by an External Schema defined in the DDL by the DBA, e.g. `CREATE VIEW` command in SQL.

3 Level Architecture: Conceptual Level

- The Conceptual Level describes the whole database.
- It hides the details of the data storage. It describes the data stored and the relationships between the data.
- It may be different or similar to the external views, i.e. it may require the joining of two tables.
- It presents the data as is; multiple occurrences of multiple types of conceptual records.
- Conceptual Schema:
  - Defined by the DBA using conceptual DDL.
  - Includes security and integrity constraints not present in the external levels.
  - Compiled by DBMS and stored in *data dictionary*, also known as *MetaData Repository* containing names and descriptions of tables and fields in the database.
- It is no more than a union of individual external schemas with security and integrity added on.

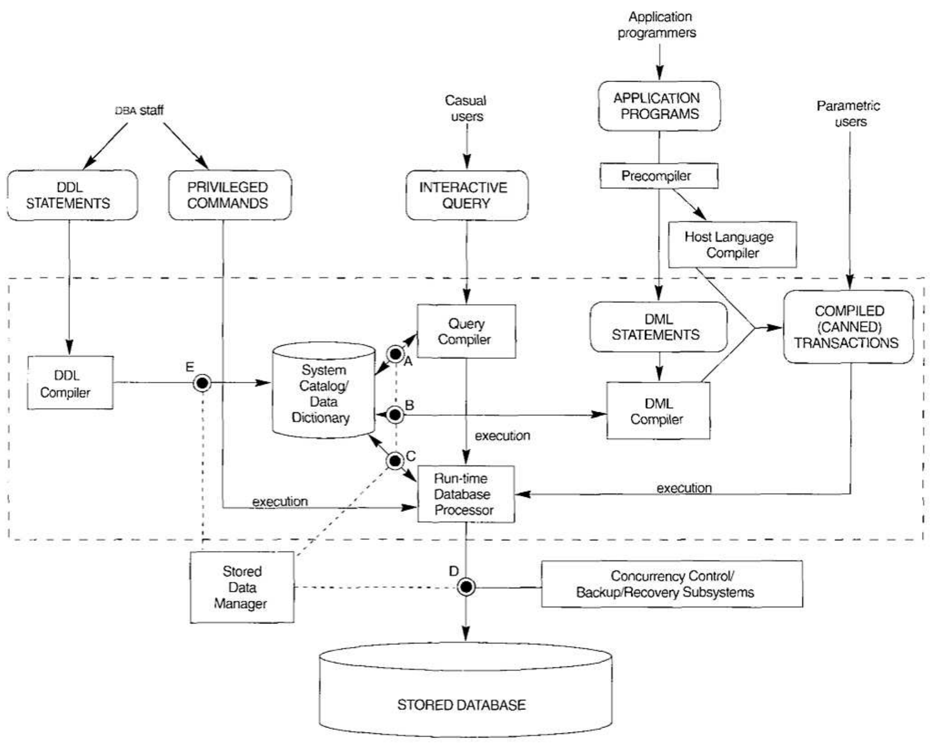
3 Level Architecture: Internal Level

- It describes the physical storage of the database.
- It managed by the Operating Systems under the direction of the DBMS.
- It defines the types of the stored records, indices, how fields are stored etc., i.e. how the data is stored at the internal level.
- It is defined using an internal DDL, i.e. a subset of the DDL (e.g. `CREATE INDEX` in SQL).
- Programs that access this level are dangerous as they bypass the security and integrity checks.
- Mappings exist between the three different levels of the three level architecture and the DBA is responsible for maintaining the correct mapping between the levels.
  - Changes to the internal level do not require changes to the conceptual level, just changes to the mapping between the levels, e.g. index change
  - Changes to the conceptual level map onto the external views.

An Alternate View Of What Makes Up A DBMS

1. *Query Compiler*: Handles high-level queries entered interactively; does parsing, analysis and generates calls to . . .
2. *Run-time Database Processor*: Handles access routines to database in the form of retrieval/update on database; access through . . .
3. *Stored Data Manager*: A high-level stored data manager controls access to DBMS data (database and database catalog) store on disk.
4. *Precompiler*: Extracts DML from application programs in the host language for sending to . . .
5. *DML Compiler*: Generates object code to access the database.
6. *DDL Compiler*: Used by DBA to process schema definitions in DDL for storage of metadata in a catalog.

An Alternate View Of What Makes Up A DBMS (2)



Source: Fundamentals of Database Systems (Elmasri & Navathe)

The Role of a Database Administrator (DBA)

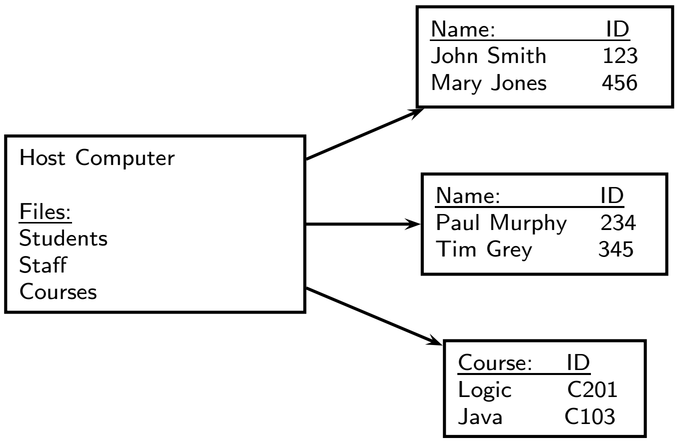
- The DBA has an essential role in any DBMS and has overall control of the DBMS.
- Decides on the information content and the logical and conceptual database design schema.
- Decides on the storage structure (index, hash) and access using the DDL.
- Liases with users and helps them design external schema using DDL.
- Defines security and integrity checks.
- Defines backup and recovery strategies.
- Monitors the performance of the DBMS, responding to changing requirements using statistical analysis routines.

Data Models

- Models are used to hide superfluous details while highlighting details relevant to the applications at hand.
- The data model is a mechanism to allows conceptualisation of associations of entities and their attributes.
- Main models are *Hierarchical*, *Network* and *Relational*.
- Almost all non-relational models have been extended to have relational front-ends.
- *Entity-Relational* representations can be mapped into Relational databases.

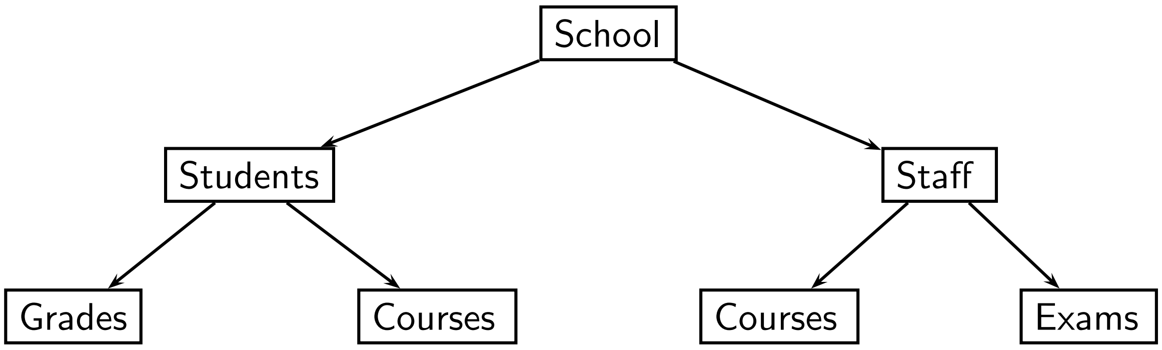
Flat File Model

- Dates from the 1950s
- Single, sequential 2D array of data
- Ok for storing data (small lists etc.)
- Very inefficient for searching (may have to look at all entries)
- No support for concurrency
- Generally contains a lot of redundant information



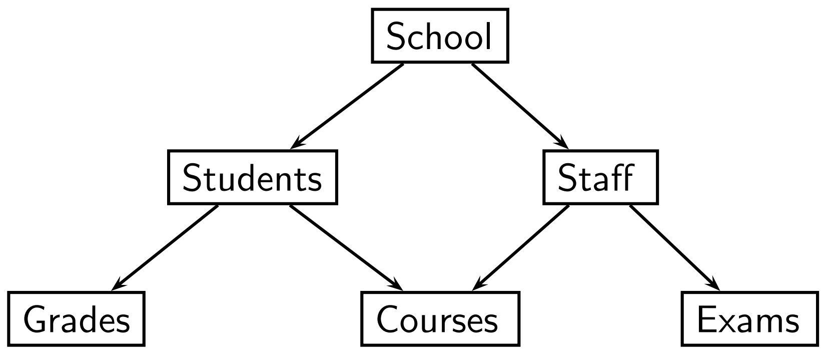
Hierarchical Model

- Dates from late 1960s
- Parent-child relationships
- Only 1:1 relations
- Works for Table of Contents type data
- Problems with deletion, duplication, navigation and changes may need to made multiple times



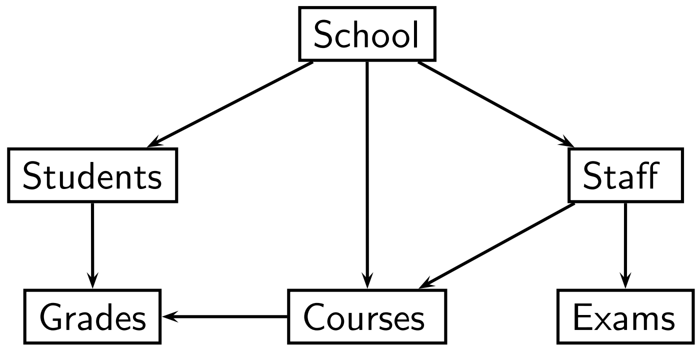
Network Model

- Dates from early 1970s
- Different from Hierarchical model
  - Supports N:M relationships
  - Navigation can start anywhere
  - Requires familiarity with structure to change and optimise the database



Relational Model

- Developed by EF Codd, IBM 1970
- Aims to “protect” user from database structure.
- Data independence means data location is not important.
- Data stored in *Relations/Tables* resulting in data independence.
- Tabular view results in conceptual simplicity, easy to design, implement and use. Ad hoc queries based on SQL.
- Increased hardware and software overhead.


School
Students
Staff
Grades
Courses
Exams