COS221
L02 - Database System Concepts and Architecture
(Chapter 2 - Editions 6 and 7)

Linda Marshall

23 February 2023

Evolution of the DBMS Architecture

There was a transition from monolithic systems to *client/server* architecture.
Client/server architecture comprises of two modules, the *client* and the *server*.
The client runs on the user workstations and provides the interface between the user and the database.
The server handles storage, access, searching etc. of the database.

Concepts Relating to the Study of Database Systems

- **Data abstraction** does not consider data organisation or storage. It provides a better understanding of the data. Users perceive data at a level of abstraction relevant to them.
- A **Data model** is a collection of concepts that can be used to describe the structure of the database.

Categories of Concepts used to describe the database (Data Models)

- High-level or **conceptual data model**s - relates to how the user perceives that data. Concepts include entities, attributes and relationships. e.g. Entity-relationship model
- Representational or **implementation data model**s - understood by the end-users, but not to far removed from how the data is organised. e.g. *Relational data model*, network model, hierarchical model, object data model, XML model
- Low-level or **physical data model**s - provides details on how data is stored. e.g. record formats, record ordering, access paths (indexes)
- **Self-describing data model**s (Schema and data values combined). e.g Key-value stores and NOSQL systems

Schemas, Instances, and Database State

The *description of a database* is called the **database schema,** which is specified during database design and is not expected to change frequently. Example of a **schema diagram:**

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

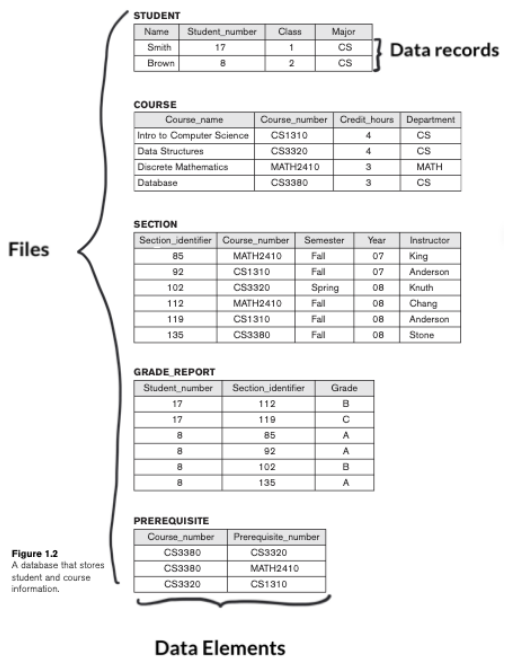
Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

Schemas, Instances, and Database State Cont...

- We call each object in the schema - such as STUDENT or COURSE - a **schema construct.**
- The data in the database at a particular moment in time is called a **database state** or **snapshot.**
- It is also called the *current* set of **occurrences** or **instances** in the database.
- The schema is sometimes called the **intension,** and a database state is called an **extension** of the schema.



STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

Data records

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Files

Data Elements

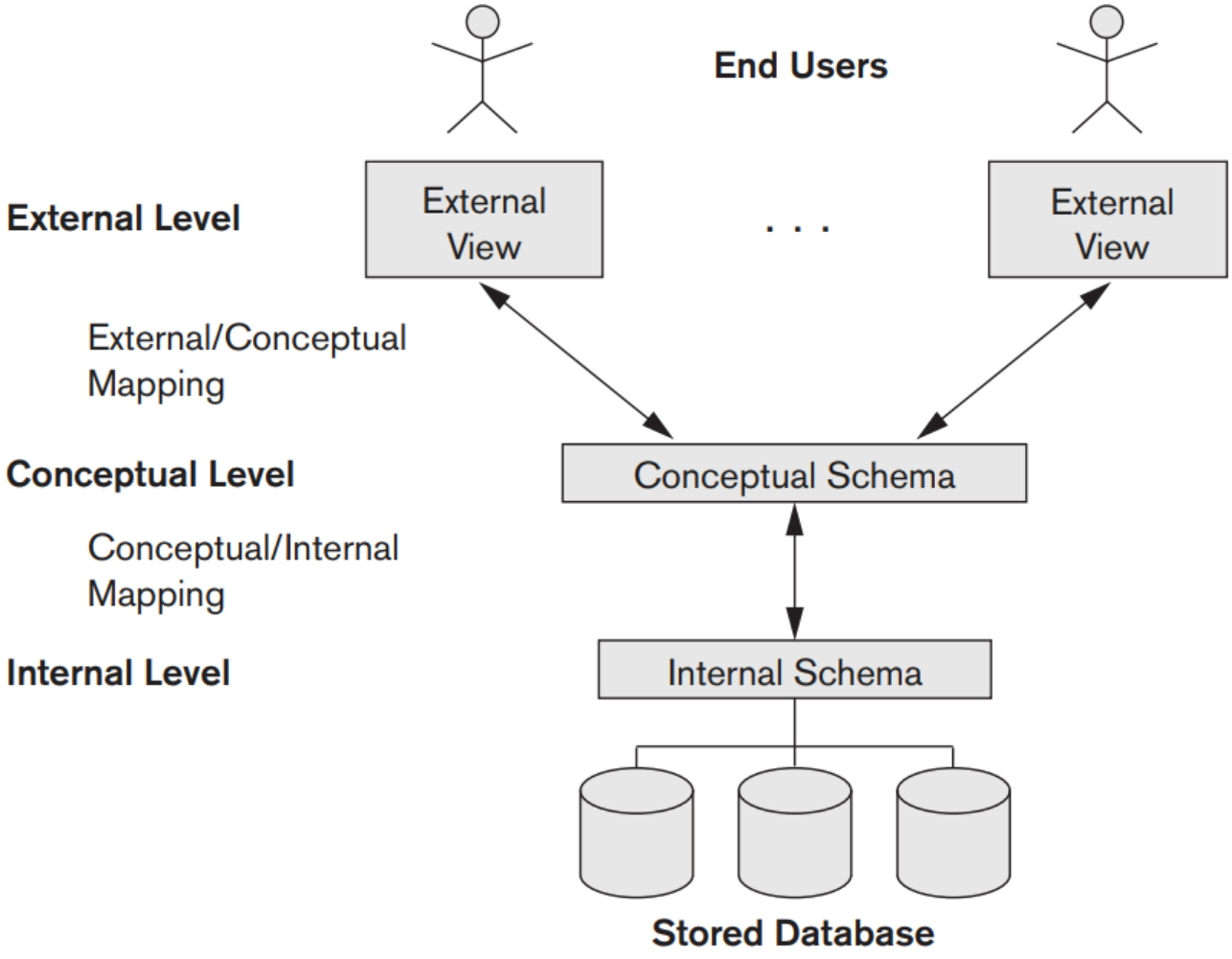
Figure 1.2
A database that stores student and course information.

The Three-Schema Architecture (ANSI/SPARC)

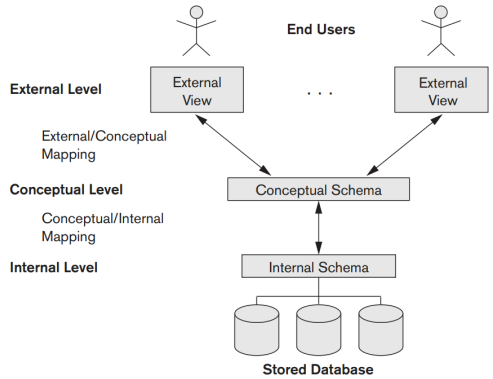
The three-schema architecture is used to:

- achieve multiple views
- separate programs and data, provides data abstraction
- provide a catalog to store the schema
- promote data independence

The Three-Schema Architecture (ANSI/SPARC)



The Three-Schema Architecture (ANSI/SPARC)



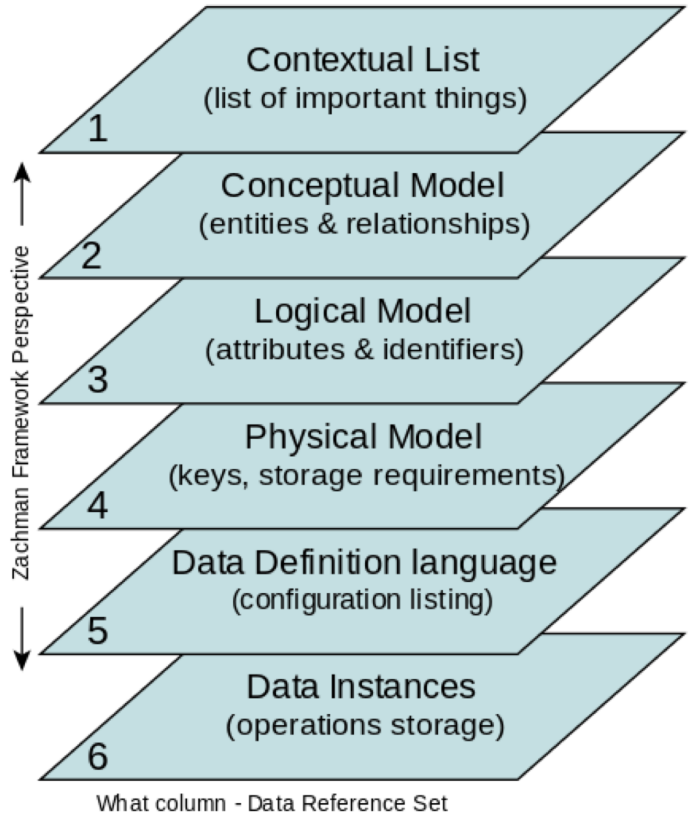
- the mappings transform requests and results between the levels
- the schemas
  - External - includes multiple user views in a high-level data model
  - Conceptual - describes entities, data types, relationships, user operations and constraints
  - Internal - describes the physical storage structure of the database and access paths in a physical data model

Data Independence

The capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

1. **Logical data independence** is the capacity to change the conceptual schema without having to change external schemas or application programs.
2. **Physical data independence** is the capacity to change the internal schema without having to change the conceptual schema.

Interesting fact, the *Zachman framework*, used in Enterprise Architecture Frameworks, evolved from the three schema architectural model.



DBMS Languages

- **Data Definition Language (DDL),** is used by the DBA and by database designers to define the conceptual schema. The DDL can also be used to define the internal schema if there is no strict separation of the schema levels in the DBMS.
- **Storage Definition Language (SDL),** is used to specify the internal schema if there is a clear separation of the schema levels. The DDL is used to specify the conceptual schema and the mappings to the internal schemas.
- **View Definition Language (VDL),** is used to specify user views and their mappings to the conceptual schema, but in most DBMSs the DDL is used to define both conceptual and external schemas.
- **Data Manipulation Language (DML),** is used for retrieval, insertion, deletion, and modification of the data that is stored in the database.

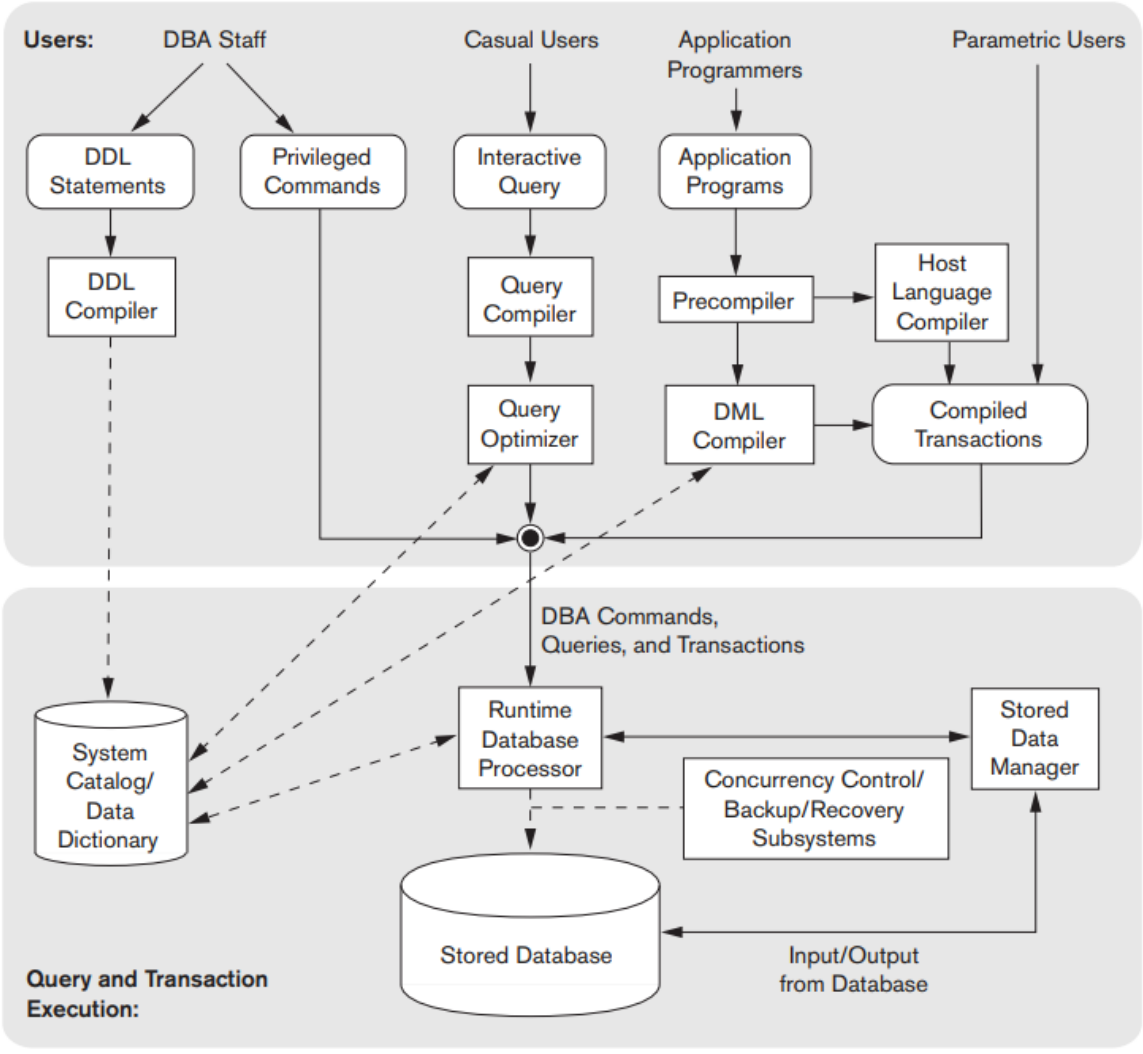
Two types of Data Manipulation Language (DML) exist:

- high-level (non-procedural, or set-at-a-time, also referred to as a query language e.g. SQL) DML's specify *what* is to be retrieved rather than *how*. These fall into the declarative paradigm of computer languages.
- low-level (procedural, record-at-a-time)

DBMS Interfaces

- Menu-based Interfaces for Web Clients or Browsing
- Apps for Mobile Devices
- Forms-based Interfaces
- Graphical User Interfaces
- Natural Language Interfaces
- Keyword-based Database Search
- Speech Input and Output
- Interfaces for Parametric Users
- Interfaces for the DBA

DBMS Component Modules



Database System Utilities

- **Loading.** A loading utility is used to load existing data files - such as text files or sequential files - into the database.
- **Backup.** A backup utility creates a backup copy of the database, usually by dumping the entire database onto tape or other mass storage medium.
- **Database storage reorganisation.** This utility can be used to reorgansze a set of database files into different file organisations and create new access paths to improve performance.
- **Performance monitoring.** Such a utility monitors database usage and provides statistics to the DBA.

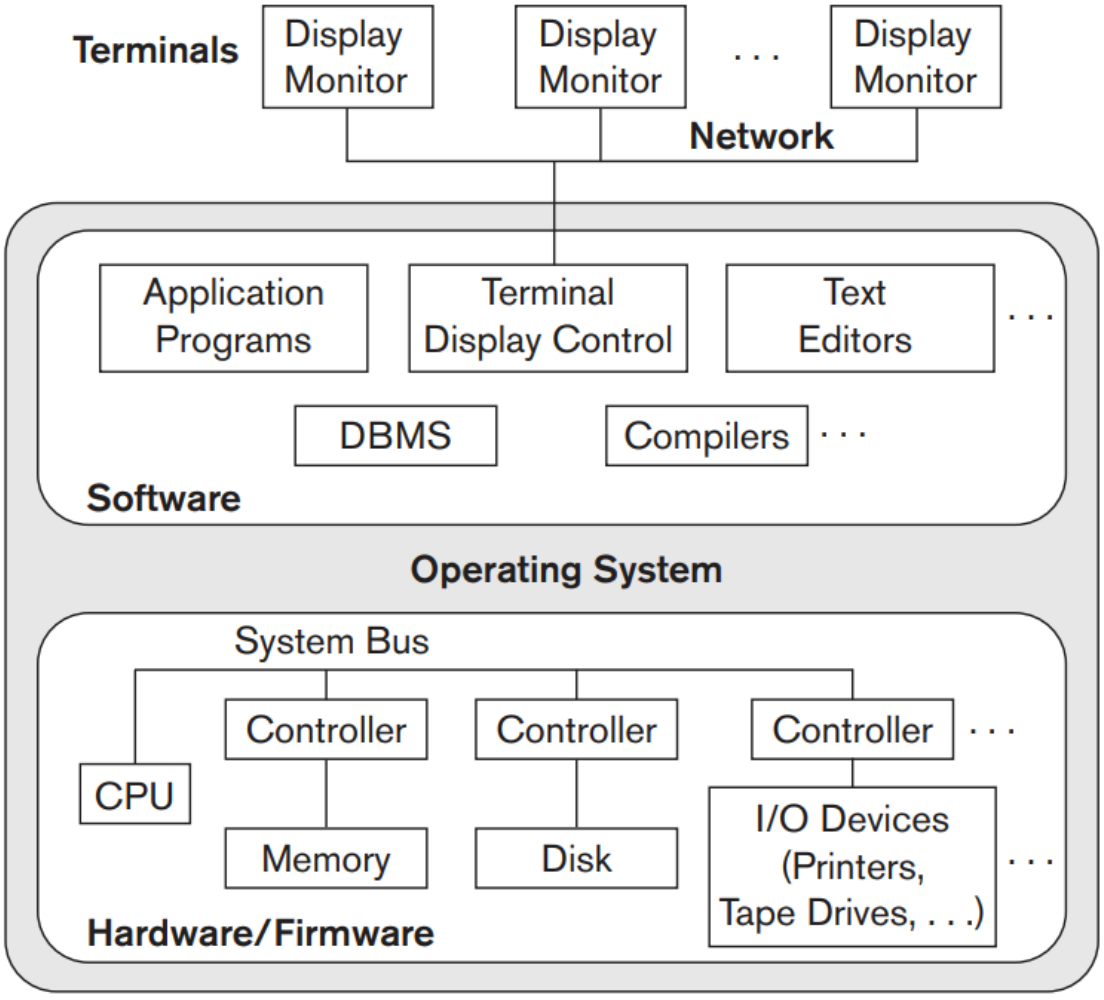
Classification of Database Management Systems

- Based on the data model: relational, object, object-relational, NOSQL, key-value, hierarchical, network, and others.
- Based on the number of users: single-user systems, multi-user systems.
- Based on the number of sites over which the database is distributed: centralised, distributed, etc.
- Based on cost: open source, etc.
- Based on types of access paths: inverted file structures, etc.
- Based on purpose: general purpose and special purpose.

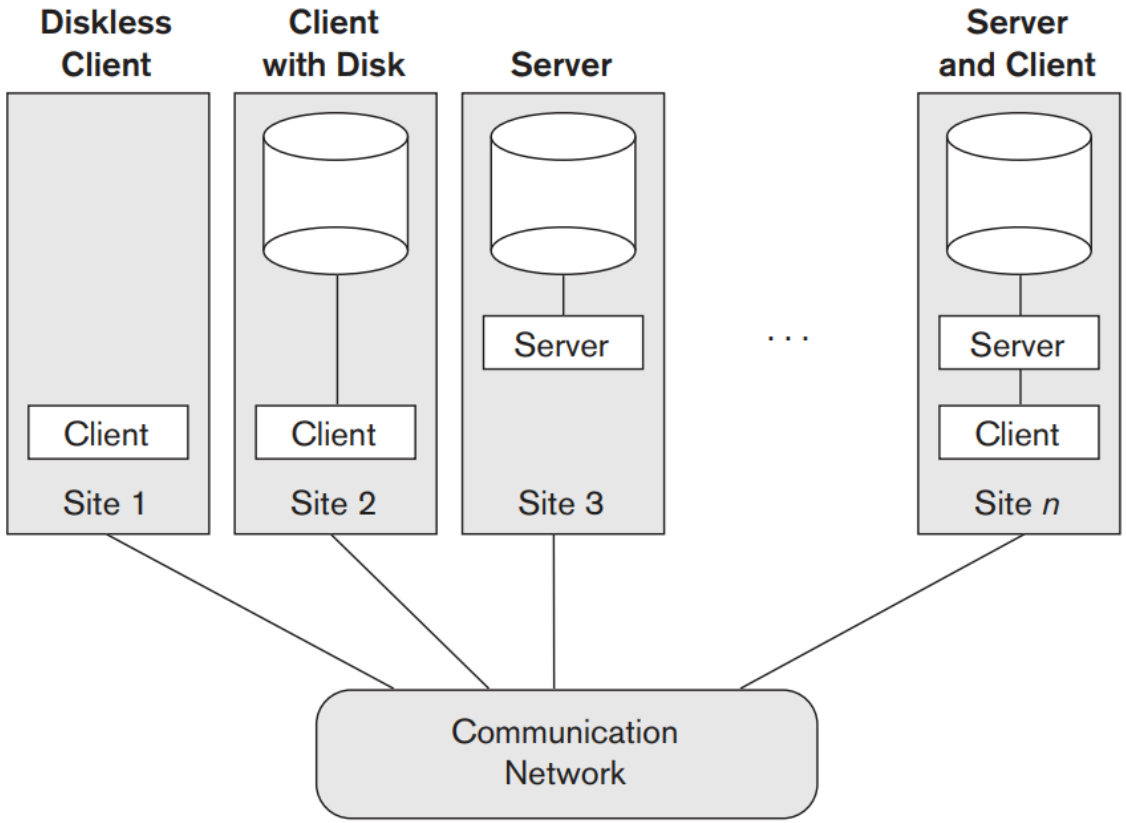
Architectures other than the three-tier C/S architecture

- Centralised DBMSs Architecture
- Two-Tier Client/Server Architectures for DBMSs
- Three-Tier and n-Tier Architectures for Web Applications

Centralised DBMSs Architecture



Two-Tier Client/Server Architectures for DBMSs



Three-Tier and n-Tier Architectures for Web Applications