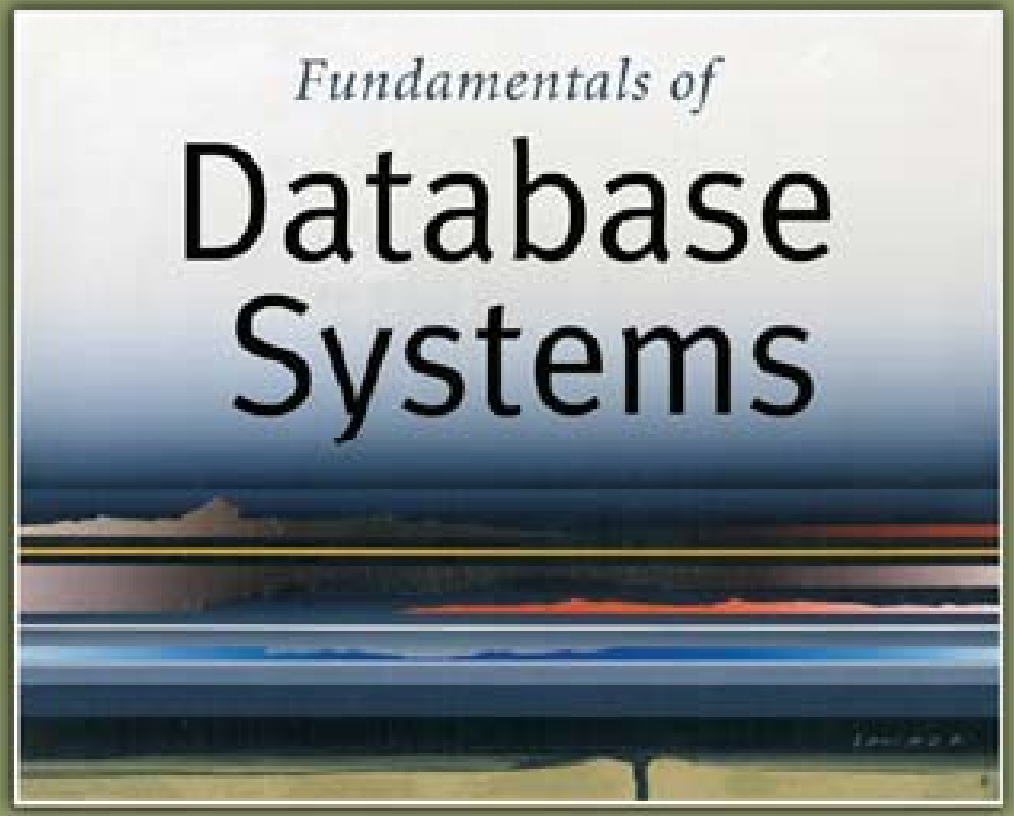
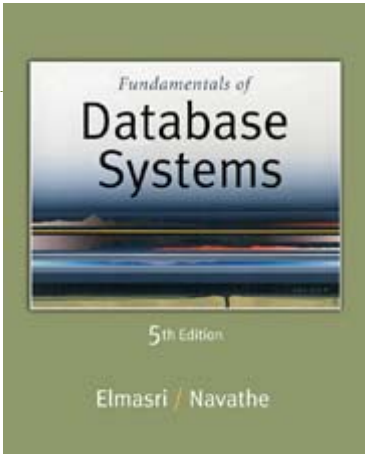


5th Edition

Elmasri / Navathe

Chapter 2

Database System Concepts and Architecture



Outline

- Data Models and Their Categories
- Schemas, Instances, and States
- Three-Schema Architecture
- Data Independence
- DBMS Languages and Interfaces
- Database System Utilities and Tools
- Classification of DBMSs
- History of Data Models

Data Models

- **Data Abstraction:**
  - The suppression of details of data organization and storage and the highlighting of essential features for better understanding
- **Data Model:**
  - A set of concepts to describe the ***structure*** of a database, and certain ***constraints*** that the database should obey.
  - Data model helps to achieve data abstraction.

Categories of Data Models

- **Conceptual (high-level, semantic) data models:**
  - Provide concepts that are close to the way many users perceive data.
    - Also called ***entity-based*** or ***object-based*** data models.
- **Physical (low-level, internal) data models:**
  - Provide concepts that describe details of how data is stored in the computer.
- **Representational (Implementation) data models:**
  - Provide concepts that may be understood by end users but that are not too different from the way data is organized within the computer
  - Hide some details of data storage but can be implemented on a computer system in a direct way
  - Used by many commercial DBMS implementations (e.g. relational data models used in many commercial systems).

Schemas versus Instances

- Database Schema:
  - The ***description*** of a database.
  - Includes descriptions of the database structure, data types, and the constraints on the database.
- Schema Diagram:
  - An ***illustrative*** display of a database schema.
- Schema Construct:
  - A ***component*** of the schema or an object within the schema, e.g., STUDENT, COURSE.
- Database State:
  - The actual data stored in a database at a ***particular moment in time***. This includes the collection of all the data in the database.
  - Also called database instance (or occurrence or snapshot).

Database Schema vs. Database State

- Database State:
  - Refers to the ***content*** of a database at a moment in time.
- Initial Database State:
  - Refers to the database state when it is initially loaded into the system.
- Valid State:
  - A state that satisfies the structure and constraints of the database.
- Schema and State Distinction
  - The ***database schema*** changes very infrequently. The ***database state*** changes every time the database is updated.
  - **Schema** is also called **intension** whereas **State** is also called **extension**.

Example of a Database Schema

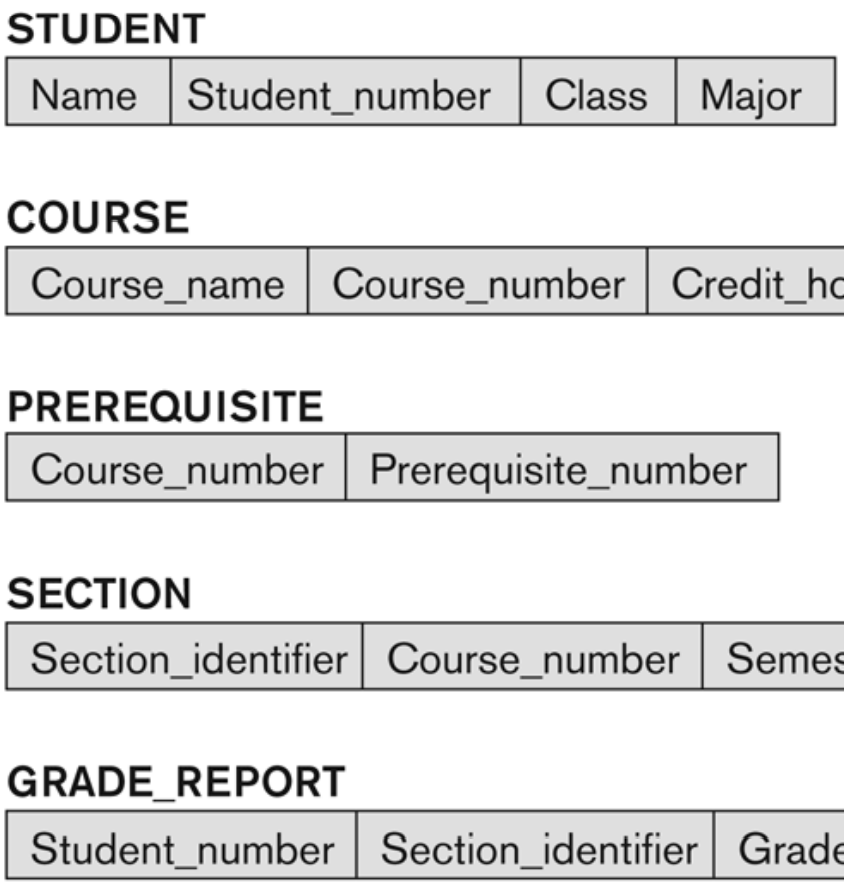


Figure 2.1
Schema diagram for the database in Figure 1.2.

Example of a database state

STUDENT

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

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

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

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

PREREQUISITE

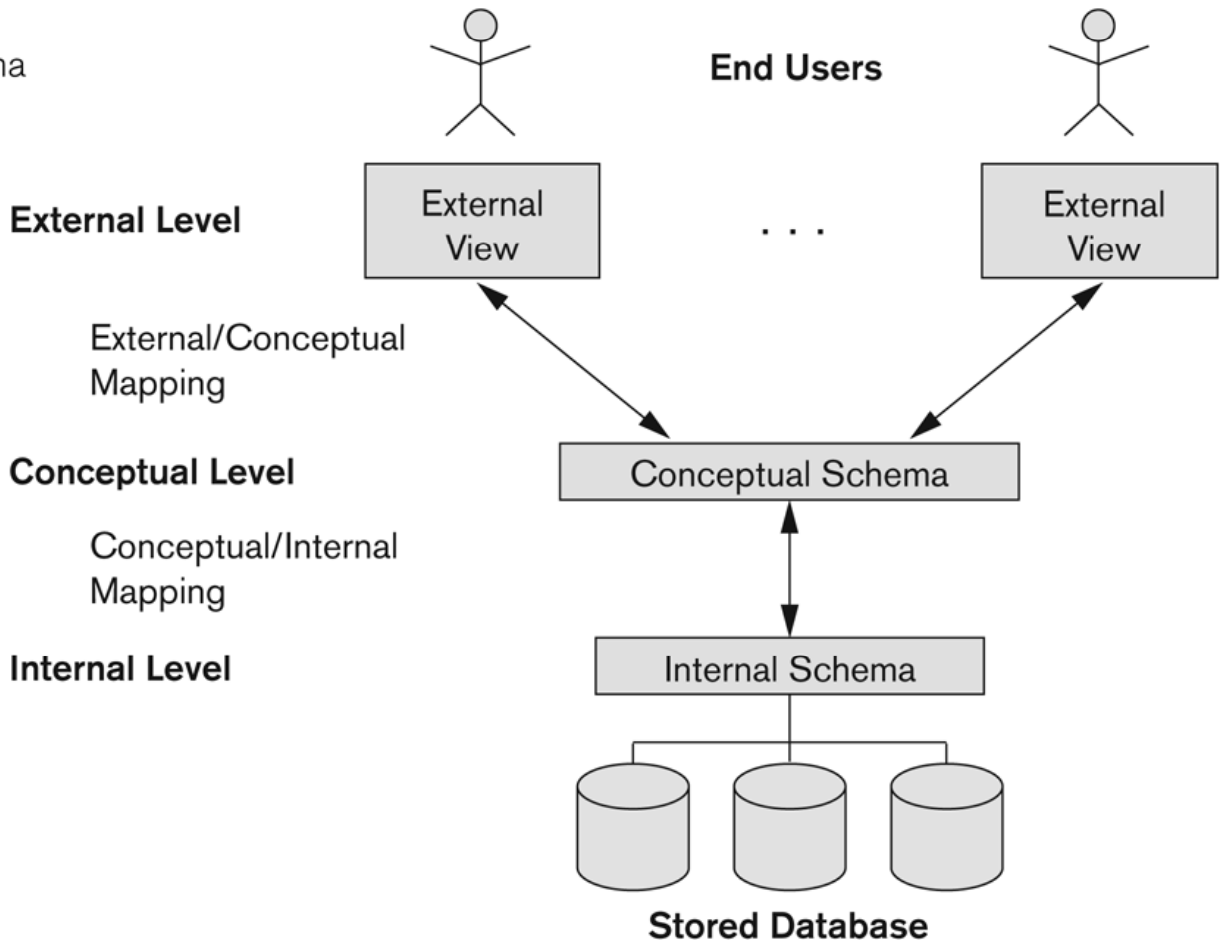
Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Three-Schema Architecture

- Proposed to help achieve the DB characteristics
  - Specially to separate user applications and the physical database
- It defines DBMS schemas at ***three*** levels:
  - **Internal schema** at the internal level to describe physical storage structures and access paths (e.g indexes).
    - Typically uses a **physical** data model.
  - **Conceptual schema** at the conceptual level to describe the structure for the whole database for a community of users. It hides the details of the physical storage structures and concentrates on describing entities, data types, relationships, and constraints.
    - Uses a **conceptual** or an **implementation** data model.
  - **External schemas** at the external level to describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.
    - Usually uses the same data model as the conceptual schema.

The three-schema architecture

Figure 2.2
The three-schema architecture.



Data Independence

- **The ability to change the schema at one level of the database system without having to change the schema at the next higher level.**
  - **Logical Data Independence:**
    - The capacity to change the conceptual schema without having to change the external schemas and their associated application programs.
  - **Physical Data Independence:**
    - The capacity to change the internal schema without having to change the conceptual schema.

DBMS Languages

- **Data Definition Language (DDL):** a language that is used to define database schemas. The DDL statement is used to identify description of the schema construct and store the schema description in the DBMS catalog.

- **Data Manipulation Language (DML):** a language that is used to manipulate (retrieve, insert, delete, and modify) data.
  - A **high-level or non-procedural DML** can be used on its own to specify complex database operations in a concise manner, such as SQL.
  - A **low-level or procedural DML** typically retrieves individual records or objects from the database and processes each separately, such as PL/SQL.
  - Embedded Languages**:** DML with Java, VB, C++, etc.

DBMS Interfaces

- Stand-alone query language interfaces
  - Example: Entering SQL queries at the DBMS interactive SQL interface (such as, SQL*Plus in ORACLE)
- Programmer interfaces for embedding DML in programming languages
- User-friendly interfaces
  - Menu-based, popular for browsing on the web
  - Forms-based, designed for naïve users
  - Graphics-based
  - Natural language: requests in written English
  - Parametric interfaces, e.g., bank tellers using function keys.
  - Interfaces for the DBA

DBMS Component Modules

- A typical DBMS consists of the following components:
  - DDL compiler: process schema definitions, specified in the DDL statements, and stores descriptions of the schemas in the system catalog.
  - DML compiler: compiles the DML commands into object code for database access.
  - Run-time database processor: handles database access at run time. It receives retrieval and update operations and carries them out on the database.
  - Query compiler: handles high-level queries that are entered interactively.
  - Data manager: controls access to DBMS information that is stored on disk through interaction with operating system.

DBMS Component modules

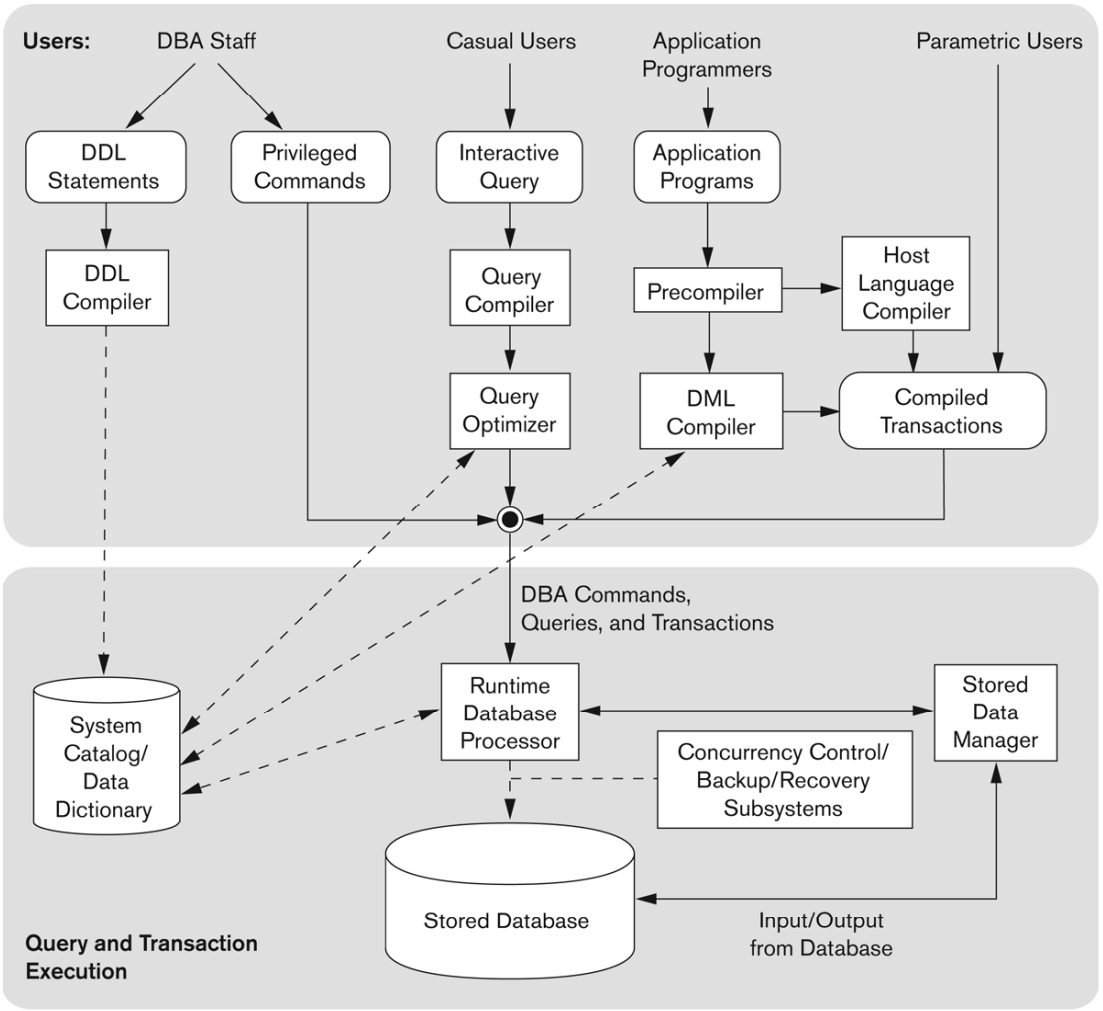


Figure 2.3
Component modules of a DBMS and their interactions.

Database System Utilities

- There are some functions that are not provided through the normal DBMS components rather they are provided through additional programs called utilities. Some of these are:
  - Loading or import utility: used to load or import existing data files into the database.
  - Backup utility: used to create backup copies of the database, usually by dumping the entire database onto tape.
  - File reorganization utility: is used to reorganize a database file into a different file organization to improve performance.
  - Performance monitoring utility: is used to monitor database usage and provides statistics to the DBA.

Other Tools

- Data dictionary / repository:
  - Used to store schema descriptions and other information such as design decisions, application program descriptions, user information, etc.
- CASE tools:
  - Used in the design phase of database
    - Examples: Rational Rose, TOAD
- Application Development Environments:
  - Provides facilities for developing database applications including database design, graphical user interface development, querying and updating, and application program development.
    - PowerBuilder (Sybase), JBuilder (Borland), JDeveloper 10G (Oracle)

Classification of DBMSs

- Several criteria are normally used to classify DBMSs
  - Data Model:
    - Relational, Object, Object-Relational, Hierarchical, Network
  - Number of Users:
    - Single-user system
    - Multi-user system
  - Number of Sites
    - Distributed
    - Centralized
  - Cost:
  - Purpose:
    - General
    - special

History of Data Models

- **Network Model:**
  - The first network DBMS was implemented by Honeywell in 1964-65 (IDS System). Later implemented in a large variety of systems
- **Hierarchical Data Model:**
  - Used in IBM's IMS (and some other products) which still have a large customer base worldwide.
  - Dominated the DBMS market during 1965-1985
- **Relational Model:**
  - Proposed in 1970 by E.F. Codd (IBM), first commercial system in 1981-82. Currently most dominant.
  - In several commercial and free open source products:
    - DB2, ORACLE, MS SQL Server, SYBASE, INFORMIX.
    - MySQL, PostgreSQL

History of Data Models

- **Object-oriented Data Models:**
  - Comprises models of persistent O-O Programming Languages such as C++ (e.g., in OBJECTSTORE), Java and Smalltalk (e.g., in GEMSTONE).
- **Object-Relational Models:**
  - Most Recent Trend. Started with Informix Universal Server.
  - Relational systems incorporate concepts from object databases leading to object-relational.
  - Exemplified in the latest versions of Oracle-10i, DB2, and SQL Server and other DBMSs.