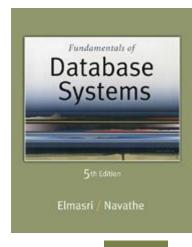


5th Edition

Elmasri / Navathe

Chapter 2

Database System Concepts and Architecture





Outline

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

Data Models

Data Model:

 A set of concepts to describe the structure of a database, the operations for manipulating these structures, and certain constraints that the database should obey.

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. These are usually specified in an ad-hoc manner through DBMS design and administration manuals
- Implementation (representational) data models:
 - Provide concepts that fall between the above two, 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 (most aspects of) a database schema.
- Schema Construct:
 - A component of the schema or an object within the schema, e.g., STUDENT, COURSE.

Database Instances

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).
 - The term *instance* is also applied to individual database components, e.g. *record instance, table instance, entity instance*

Database Schema vs. Database State (continued)

- Distinction
 - The database schema changes very infrequently.
 - The database state changes every time the database is updated.
- Schema is also called intension.
- State is also called extension.

Example of a Database Schema

STUDENT

Name Student_number Class Major

Figure 2.1

Schema diagram for the database in Figure 1.2.

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
----------------	--------------------	-------

Example of a database state

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	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

GRADE REPORT

Student_number	Section_identifier	Grade
17	112	В
17	119	С
8	85	Α
8	92	Α
8	102	В
8	135	Α

PREREQUISITE

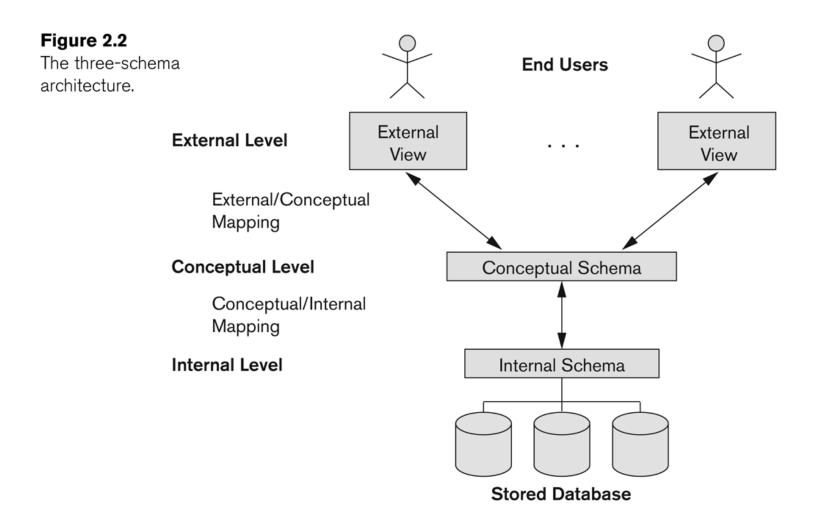
Figure 1.2A database that stores student and course information.

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Three-Schema Architecture

- 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 and constraints for the whole database for a community of users.
 - Uses a conceptual or an implementation data model.
 - External schemas at the external level to describe the various user views.
 - Usually uses the same data model as the conceptual schema.

The three-schema architecture



Three-Schema Architecture

- Mappings among schema levels are needed to transform requests and data.
 - Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.
 - Data extracted from the internal DBMS level is reformatted to match the user's external view (e.g. formatting the results of an SQL query for display in a Web page)

Data Independence

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.
- For example, the internal schema may be changed when certain file structures are reorganized or new indexes are created to improve database performance

DBMS Languages

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
 - High-Level or Non-procedural Languages: These include the relational language SQL
 - May be used in a standalone way or may be embedded in a programming language
 - Low Level or Procedural Languages:
 - These must be embedded in a programming language

DBMS Languages

Data Definition Language (DDL):

- Used by the DBA and database designers to specify the conceptual schema of a database.
- In many DBMSs, the DDL is also used to define internal and external schemas (views).
- In some DBMSs, separate storage definition language (SDL) and view definition language (VDL) are used to define internal and external schemas.
 - SDL is typically realized via DBMS commands provided to the DBA and database designers

DBMS Languages

- Data Manipulation Language (DML):
 - Used to specify database retrievals and updates
 - DML commands (data sublanguage) can be embedded in a general-purpose programming language (host language), such as COBOL, C, C++, or Java.
 - A library of functions can also be provided to access the DBMS from a programming language
 - Alternatively, stand-alone DML commands can be applied directly (called a *query language*).

DBMS Interfaces

- Stand-alone query language interfaces
 - Example: Entering SQL queries at the DBMS interactive SQL interface (e.g. SQL*Plus in ORACLE)
- Programmer interfaces for embedding DML in programming languages
- User-friendly interfaces
 - Menu-based, forms-based, graphics-based, etc.

DBMS Programming Language Interfaces

- Programmer interfaces for embedding DML in a programming languages:
 - Embedded Approach: e.g embedded SQL (for C, C++, etc.), SQLJ (for Java)
 - Procedure Call Approach: e.g. JDBC for Java,
 ODBC for other programming languages
 - Database Programming Language Approach:
 e.g. ORACLE has PL/SQL, a programming language based on SQL; language incorporates
 SQL and its data types as integral components

User-Friendly DBMS Interfaces

- Menu-based, popular for browsing on the web
- Forms-based, designed for naïve users
- Graphics-based
 - (Point and Click, Drag and Drop, etc.)
- Natural language: requests in written English
- Combinations of the above:
 - For example, both menus and forms used extensively in Web database interfaces

A Physical Centralized Architecture

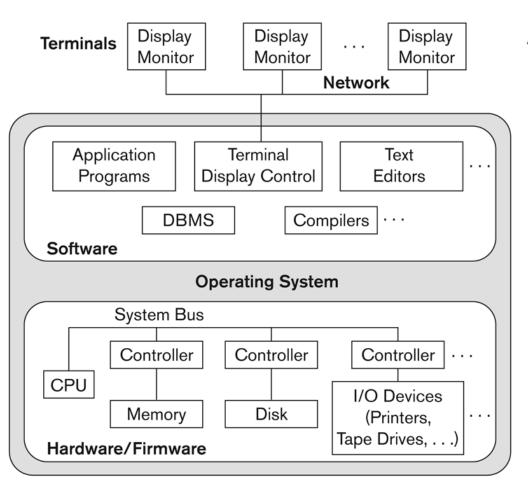


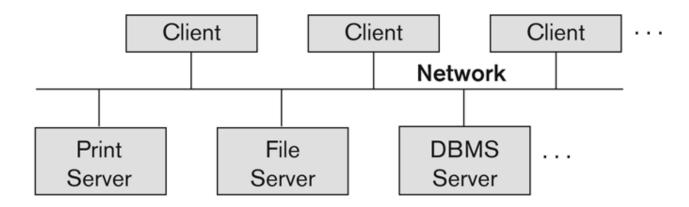
Figure 2.4
A physical centralized architecture.

Basic 2-tier Client-Server Architectures

- Specialized Servers with Specialized functions
 - Print server
 - File server
 - DBMS server
 - Web server
 - Email server
- Clients can access the specialized servers as needed

Logical two-tier client server architecture

Figure 2.5
Logical two-tier client/server architecture.



Clients

- Provide appropriate interfaces through a client software module to access and utilize the various server resources.
- Clients may be diskless machines or PCs or Workstations with disks with only the client software installed.
- Connected to the servers via some form of a network.
 - (LAN: local area network, wireless network, etc.)

DBMS Server

- Provides database query and transaction services to the clients
- Relational DBMS servers are often called SQL servers, query servers, or transaction servers
- Applications running on clients utilize an Application Program Interface (API) to access server databases via standard interface such as:
 - ODBC: Open Database Connectivity standard
 - JDBC: for Java programming access
- Client and server must install appropriate client module and server module software for ODBC or JDBC
- See Chapter 9

Two Tier Client-Server Architecture

- A client program may connect to several DBMSs, sometimes called the data sources.
- In general, data sources can be files or other non-DBMS software that manages data.
- Other variations of clients are possible: e.g., in some object DBMSs, more functionality is transferred to clients including data dictionary functions, optimization and recovery across multiple servers, etc.

Three Tier Client-Server Architecture

- Common for Web applications
- Intermediate Layer called Application Server or Web Server:
 - Stores the web connectivity software and the business logic part of the application used to access the corresponding data from the database server
 - Acts like a conduit for sending partially processed data between the database server and the client.
- Three-tier Architecture Can Enhance Security:
 - Database server only accessible via middle tier
 - Clients cannot directly access database server

Three-tier client-server architecture

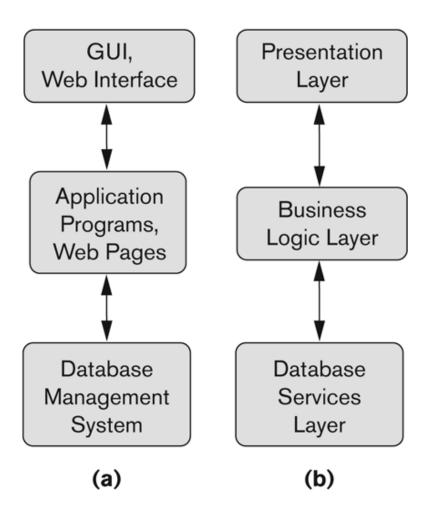
Figure 2.7

Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.

Client

Application Server or Web Server

> Database Server



Summary

- Data Models and Their Categories
- History of Data Models
- Schemas, Instances, and States
- Three-Schema Architecture
- Data Independence
- DBMS Languages and Interfaces
- Centralized and Client-Server Architectures
- Classification of DBMSs