

NMK20303 - Database System Concepts and Architecture

Chapter 2

Mohamed Elshaikh

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.

- **Data Model Structure and Constraints:**

- Constructs are used to define the database structure
- Constructs typically include **elements** (and their **data types**) as well as groups of elements (e.g. **entity, record, table**), and **relationships** among such groups
- Constraints specify some restrictions on valid data; these constraints must be enforced at all times

- **Data Model Operations:**

- These operations are used for specifying database *retrievals* and *updates* by referring to the constructs of the data model.
- Operations on the data model may include **basic model operations** (e.g. generic insert, delete, update) and **user-defined operations** (e.g. compute_student_gpa, update_inventory)

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).
- **Self-Describing Data Models:**
 - Combine the description of data with the data values. Examples include XML, key-value stores and some NOSQL 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 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*

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 |
|----------------|--------------------|-------|

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.
- 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 Database Schema - database state

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 |
|----------------|--------------------|-------|

Figure 2.1

Schema diagram for the database in Figure 1.2.

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 | 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 |

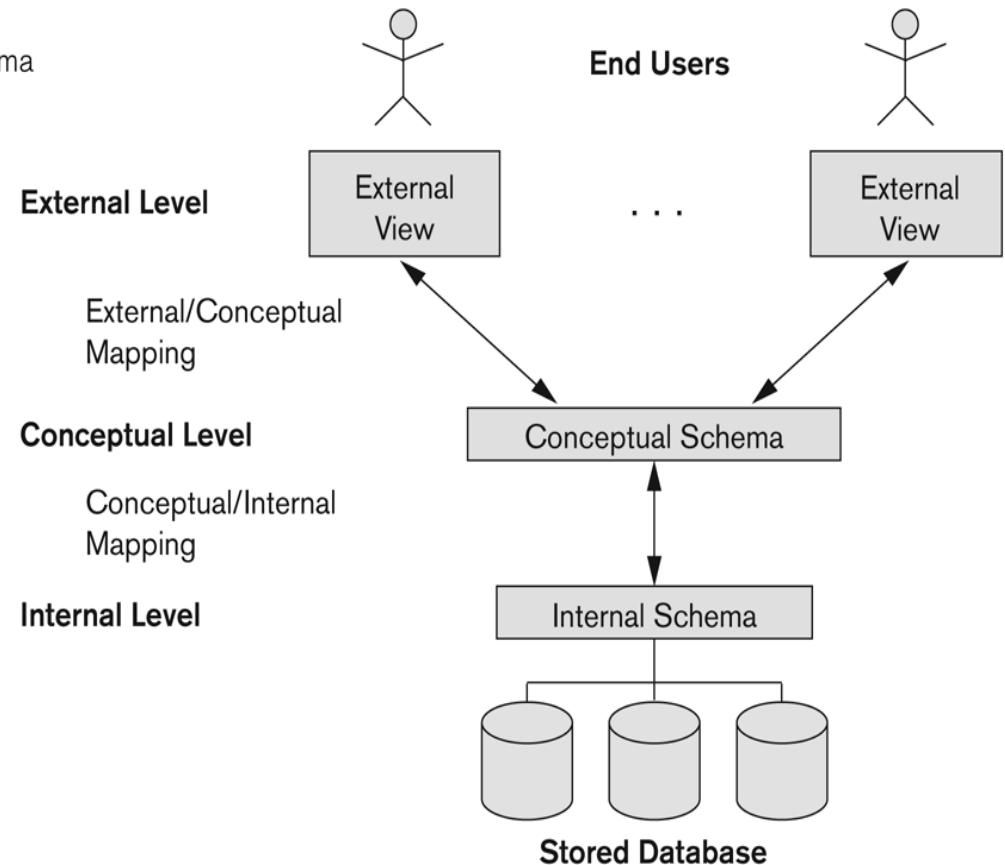
Figure 1.2

A database that stores student and course information.

Three-Schema Architecture

- Proposed to support DBMS characteristics of:
 - **Program-data independence.**
 - Support of **multiple views** of the data.
- Not explicitly used in commercial DBMS products
- Useful in explaining database system organization
- Defines DBMS schemas at **three** levels:
 - **Internal schema**
 - Describe physical storage structures and access paths (e.g indexes).
 - Typically uses a **physical** data model.
 - **Conceptual schema** [Conceptual level]
 - Describe the structure and constraints for the whole database for a community of users.
 - Uses a **conceptual** or an **implementation** data model.
 - **External schemas** [external level]
 - Describe various user views.
 - Usually uses the same data model as the conceptual schema.

Figure 2.2
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:**

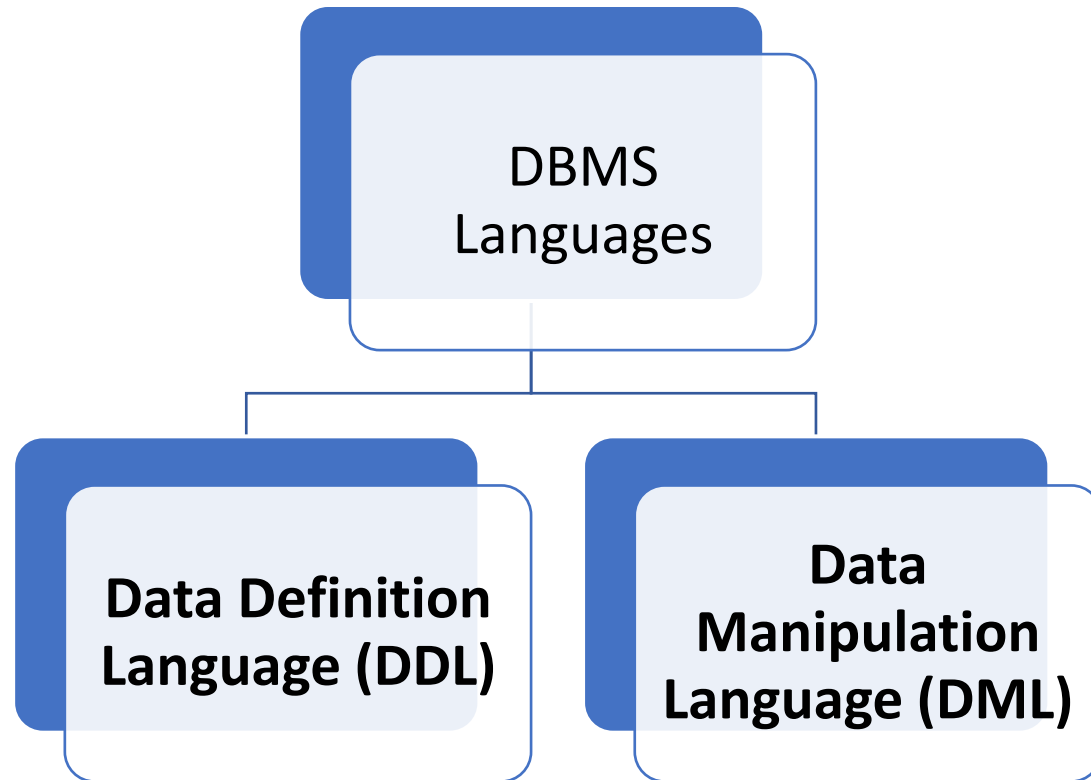
- The capacity to change the conceptual schema without having to change the external schemas and their associated application programs.

- **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

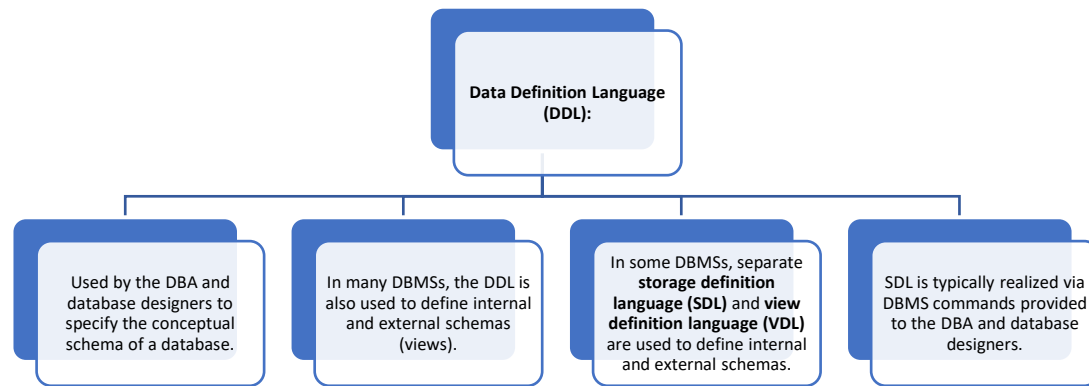
- When a schema at a lower level is changed, only the **mappings** between this schema and higher-level schemas need to be changed in a DBMS that fully supports data independence.
- The higher-level schemas themselves are **unchanged**.
 - Hence, the application programs need not be changed since they refer to the external schemas.

DBMS Languages



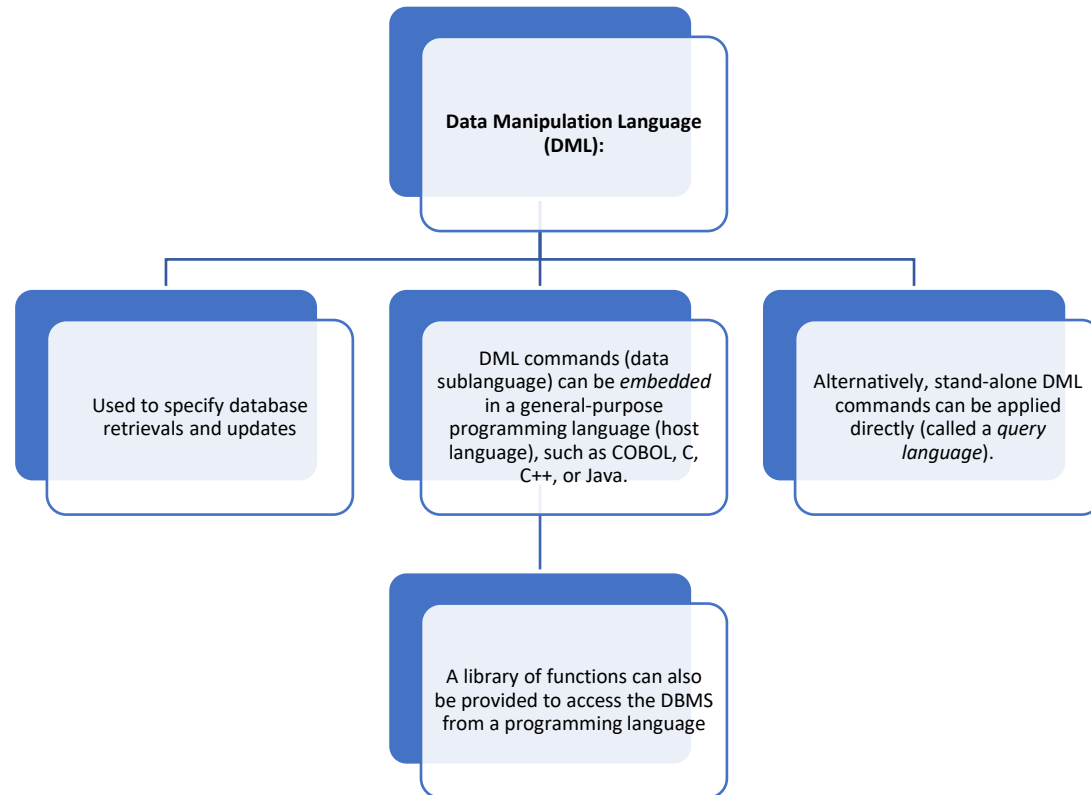
- **Data Manipulation Language (DML)**
 - **High-Level or Non-procedural Languages:** These include the relational language SQL
 - May be used in a standalone way or may be embedded in a programming language
 - **Low Level or Procedural Languages:**
 - These must be embedded in a programming language

Data Definition Language (DDL):



- **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

Data Manipulation Language (DML):

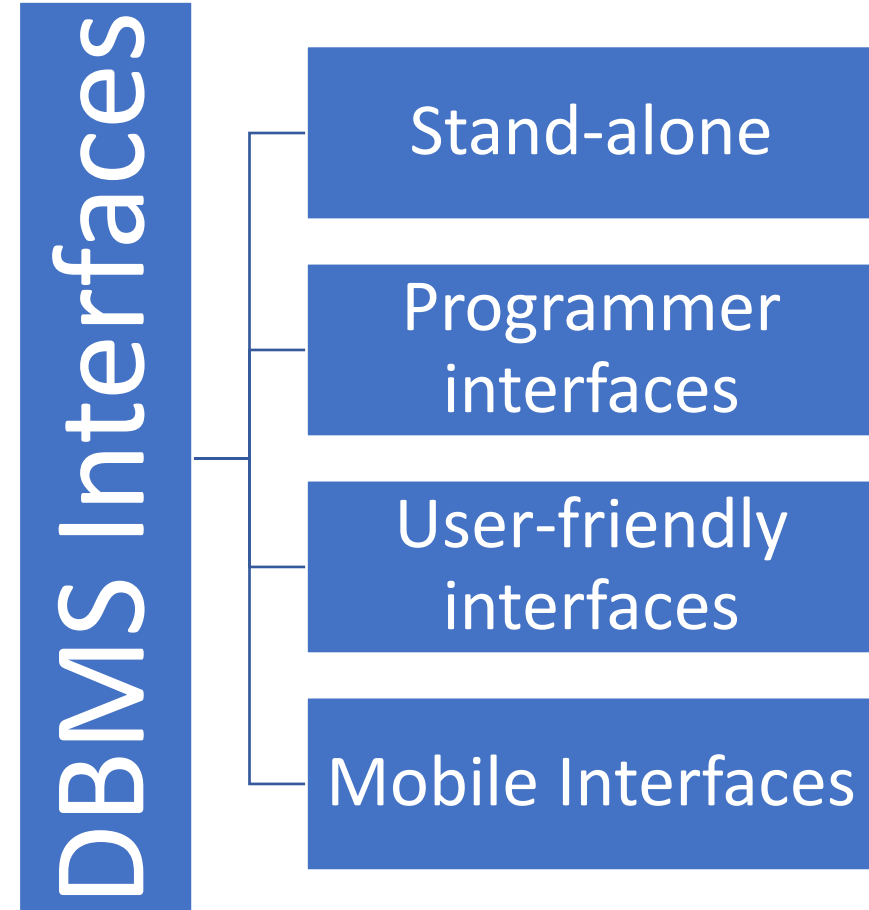


- **Data Manipulation Language (DML):**

- 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.
- Mobile Interfaces: interfaces allowing users to perform transactions using mobile apps



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 (Open Database Connectivity) for other programming languages as API's (application programming interfaces)
 - **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
 - **Scripting Languages:** PHP (client-side scripting) and Python (server-side scripting) are used to write database programs.

User-Friendly DBMS Interfaces

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

Other DBMS Interfaces

- Natural language: free text as a query
- Speech : Input query and Output response
- Web Browser with keyword search
- Parametric interfaces, e.g., bank tellers using function keys.
- Interfaces for the DBA:
 - Creating user accounts, granting authorizations
 - Setting system parameters
 - Changing schemas or access paths

Database System Utilities

- To perform certain functions such as:
 - Loading data stored in files into a database. Includes data conversion tools.
 - Backing up the database periodically on tape.
 - Reorganizing database file structures.
 - Performance monitoring utilities.
 - Report generation utilities.
 - Other functions, such as sorting, user monitoring, data compression, etc.

Other Tools

- Data dictionary / repository:
 - Used to store schema descriptions and other information such as design decisions, application program descriptions, user information, usage standards, etc.
 - **Active data dictionary** is accessed by DBMS software and users/DBA.
 - **Passive data dictionary** is accessed by users/DBA only.

Chapter Summary

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