1. Data Models, Schemas, and Instances

- Data Model:
  - A set of concepts to describe the structure of a database, the operations for manipulating the data, and the constraints that the data should follow.

- Data Model Structure and Constraints:
  - Data Model constructs define the database structure.
  - Data model constructs often include: data elements and their data types (often called attributes); grouping of related elements into entities (also called objects or records or tuples); and relationships among entities.
  - Constraints specify restrictions on the stored data; the data that satisfies the constraints is called valid data.

- 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).

1.1 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).

1.2 Schemas, Instances, and Database State

- Database Schema:
  - The description of a database.
  - Includes descriptions of the database structure, relationships, data types, and constraints.

- Schema Diagram:
  - An illustrative display of (some aspects of) a database schema.
• Schema Construct:
  o A **component** of the schema or an object within the schema, e.g., STUDENT, COURSE, Name

• Database State:
  o The actual data stored in a database at a **particular moment in time**. This includes the collection of all the data in the database.
  o Also called a database instance (or occurrence or snapshot).
    ▪ **NOTE**: The term *instance* is also used to refer to individual database components, e.g. a record instance, table instance, or entity instance

• Database State:
  o Refers to the **content** of a database at a particular moment in time.

• Initial Database State:
  o Refers to the database state when it is initially loaded into the system.

• Valid State:
  o A state that satisfies the structure and constraints of the database.

• Distinction
  o The **database schema** changes very infrequently.
  o The **database state** changes every time the database is updated.

• **Schema** is also called **intension**.
• **State** is also called **extension**.

• Example of a DB schema

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>Figure 2.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Student_number</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COURSE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Course_name</td>
<td>Course_number</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PREREQUISITE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Course_number</td>
<td>Prerequisite_number</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SECTION</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Section_identifier</td>
<td>Course_number</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GRADE REPORT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Student_number</td>
<td>Section_identifier</td>
</tr>
</tbody>
</table>

---

2. **Three-Schema Architecture and Data Independence**

• Proposed to support DBMS characteristics of:
  o **Program-data independence**.
  o Support of **multiple views** of the data.

• Not explicitly used in commercial DBMS products, but has been useful in explaining database system organization

• Defines DBMS schemas at **three** levels:
  o **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 an implementation (or a conceptual) data model.
- **External schemas** at the external level to describe the various user views.
  - Usually uses the same data model as the conceptual schema.

**Figure 2.2**
The three-schema architecture.

- Mappings among schema levels are needed to transform requests and data.
  - Users and 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 as a Web page)
- **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.

3. DBMS Languages and Interfaces

- **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).
  - Theoretically, separate **storage definition language (SDL)** and **view definition language (VDL)** can be used to define internal and external schemas. In practice:
    - SDL is typically realized via DBMS commands provided to the DBA and database designers
    - VDL is typically part of the same language as DDL

- **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 (see Chapter 13)
    - 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*).

  **Types of DML:**
  - High-Level Declarative (Set-oriented, Non-procedural) Languages, such as the relational language SQL
    - Specify “what” data to retrieve rather than “how” to retrieve it
    - May be used in a standalone way or may be embedded in a programming language
  - Low Level or Procedural (Record-at-a-time) Languages:
    - Must be embedded in a programming language
    - Need programming language constructs such as looping

- **Stand-alone query language interfaces**
  - Example: Typing SQL queries directly through the DBMS interactive SQL interface (e.g., MySQL command interface, MySQL Workbench)

- **Programmer interfaces for embedding DML in programming languages**

- **User-friendly interfaces (often Web-based)**
  - Menu-based, forms-based, graphics-based, etc.

- **Programmer interfaces for embedding DML in a programming language:**
  - **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, JPA, JPQL
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 and Web-based DBMS Interfaces**
  - Menu-based, popular for browsing on the web
  - Forms-based, designed for naive 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

- **Other DBMS Interfaces**
  - Speech as Input and Output
  - Web Browser as an interface
  - Parametric interfaces, e.g., bank tellers using function keys.
  - Interfaces for the DBA:
    - Creating user accounts, granting authorizations
    - Setting system parameters
    - Changing schemas or storage structures/access paths (physical database)

4. **Database System Environment**
4.2. 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.
  - Report generation utilities.
  - Performance monitoring utilities.
  - Other functions, such as sorting, user monitoring, data compression, etc.

4.3. 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.
Application Development Environments and CASE (computer-aided software engineering) tools often have a database design component

Examples:
- PowerBuilder (Sybase/SAP)
- JBuilder (Embarcadero Technologies)
- JDeveloper (Oracle)
- Websphere development tools (IBM)
- Open Source IDEs: NetBeans, Eclipse

5. **Centralized and Client/Server Architectures for DBMS**

5.1 Centralized DBMS Architecture:
- Combines everything into single computer system, including: DBMS software, hardware, application programs, and user interface processing software.
- User can still connect through a remote terminal – however, all processing is done at centralized site (computer).

5.2 & 5.3 Two-tier Client-Server Architecture
- Client nodes can access the specialized servers as needed
- Provide appropriate interfaces through a client software module to access and utilize the various server resources.
- Clients may be PCs or Workstations (or even diskless machines) with the client software installed.
- Connected to the servers via a network.
  - LAN: local area network
  - wireless network
  - etc.
- DBMS servers provide 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
- A program running at a client may connect to several DBMSs (also called data sources).
- In general, data sources can be files or other non-DBMS software that manages data.
- Client focuses on user interface interactions and only accesses database when needed.
- In some cases (e.g. some object DBMSs), more functionality is transferred to clients (e.g. data dictionary functions, optimization and recovery across multiple servers, etc.)

5.4 Three-Tier Architecture

- Common for Web applications
- Third intermediate layer (middle tier) called Application Server or Web Server:
  - Stores the web connectivity software and the business logic part of the application
  - Accesses and updates data on 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
6. Classification of DBMSs

- Based on the data model used
  - Traditional: Relational, Network, Hierarchical.
  - Emerging: Object-oriented, Object-relational.
- Other classifications
  - Single-user (typically used with personal computers) vs. multi-user (most DBMSs).
  - Centralized (uses a single computer with one database) vs. distributed (uses multiple computers, multiple databases, see Chapter 25)
- Cost Range: from free open-source systems to configurations costing millions of dollars
- Examples of open source relational DBMSs: MySQL, PostgreSQL, others
- Commercial DBMSs offer additional specialized modules, e.g. time-series module, spatial data module, document module, XML module
  - These offer additional specialized functionality when purchased separately
  - Sometimes called cartridges (e.g., in Oracle) or blades
- Different licensing options: site license, maximum number of concurrent users (seat license), single user, etc.