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.
    - *Structure of database* means data types, relationships, and constraints that apply to the data.
  - Provides means to achieve data abstraction.

- **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 Models (continued)

- **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)
  - It is becoming more common to specify the *dynamic aspect* or *behavior* of database application
    - specify the user-defined operations allowed on the database objects (i.e., *stored procedures*)
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).
  - **Object data model**, e.g., ODMG object model, is a higher-level implementation data model that is closer to conceptual data model.

- **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.
    - Many types of constraints are NOT represented in schema diagrams.

- **Schema Construct:**
  - A *component* of the schema or an object within the schema, e.g., STUDENT, COURSE.
Example of a Database Schema

**STUDENT**

<table>
<thead>
<tr>
<th>Name</th>
<th>Student_number</th>
<th>Class</th>
<th>Major</th>
</tr>
</thead>
</table>

**COURSE**

<table>
<thead>
<tr>
<th>Course_name</th>
<th>Course_number</th>
<th>Credit_hours</th>
<th>Department</th>
</tr>
</thead>
</table>

**PREREQUISITE**

<table>
<thead>
<tr>
<th>Course_number</th>
<th>Prerequisite_number</th>
</tr>
</thead>
</table>

**SECTION**

<table>
<thead>
<tr>
<th>Section_identifier</th>
<th>Course_number</th>
<th>Semester</th>
<th>Year</th>
<th>Instructor</th>
</tr>
</thead>
</table>

**GRADE_REPORT**

<table>
<thead>
<tr>
<th>Student_number</th>
<th>Section_identifier</th>
<th>Grade</th>
</tr>
</thead>
</table>

Figure 2.1

Schema diagram for the database in Figure 1.2.
Schemas versus 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 *current database instance* (or occurrence or snapshot).
    - The term *instance* is also applied to individual database components, e.g. *record instance*, *table instance*, *entity instance*.
Example of a database state

<table>
<thead>
<tr>
<th>COURSE</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Course name</td>
<td>Course</td>
<td>Credit</td>
<td>Department</td>
<td></td>
</tr>
<tr>
<td>Intro to Computer Science</td>
<td>CS1310</td>
<td>4</td>
<td>CS</td>
<td></td>
</tr>
<tr>
<td>Data Structures</td>
<td>CS3320</td>
<td>4</td>
<td>CS</td>
<td></td>
</tr>
<tr>
<td>Discrete Mathematics</td>
<td>MATH2410</td>
<td>3</td>
<td>MATH</td>
<td></td>
</tr>
<tr>
<td>Database</td>
<td>CS3380</td>
<td>3</td>
<td>CS</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SECTION</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Section identifier</td>
<td>Course</td>
<td>Semester</td>
<td>Year</td>
<td>Instructor</td>
</tr>
<tr>
<td>85</td>
<td>MATH2410</td>
<td>Fall</td>
<td>04</td>
<td>King</td>
</tr>
<tr>
<td>92</td>
<td>CS1310</td>
<td>Fall</td>
<td>04</td>
<td>Anderson</td>
</tr>
<tr>
<td>102</td>
<td>CS3320</td>
<td>Spring</td>
<td>05</td>
<td>Knuth</td>
</tr>
<tr>
<td>112</td>
<td>MATH2410</td>
<td>Fall</td>
<td>05</td>
<td>Chang</td>
</tr>
<tr>
<td>119</td>
<td>CS1310</td>
<td>Fall</td>
<td>05</td>
<td>Anderson</td>
</tr>
<tr>
<td>135</td>
<td>CS3380</td>
<td>Fall</td>
<td>05</td>
<td>Stone</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GRADE REPORT</th>
<th></th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student number</td>
<td>Section</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>112</td>
<td>B</td>
</tr>
<tr>
<td>17</td>
<td>119</td>
<td>C</td>
</tr>
<tr>
<td>8</td>
<td>85</td>
<td>A</td>
</tr>
<tr>
<td>8</td>
<td>92</td>
<td>A</td>
</tr>
<tr>
<td>8</td>
<td>102</td>
<td>B</td>
</tr>
<tr>
<td>8</td>
<td>135</td>
<td>A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PREREQUISITE</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Course number</td>
<td>Prerequisite number</td>
<td></td>
</tr>
<tr>
<td>CS3380</td>
<td>CS3320</td>
<td></td>
</tr>
<tr>
<td>CS3380</td>
<td>MATH2410</td>
<td></td>
</tr>
<tr>
<td>CS3320</td>
<td>CS1310</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1.2 A database that stores student and course information.
Database Schema vs. Database State

- **Database State:**
  - Refers to the *content* of a database at a moment in time.
  - When we define a new database, we specify its database schema only to the DBMS. At this point, the database state is the *empty state* with no data.

- **Initial Database State:**
  - Refers to the database state when it is initially *loaded* or *populated* into the system.

- **Valid State:**
  - A state that satisfies the *structure and constraints* of the database specified in schema. *(DBMS is partially responsible for this)*
Database Schema
vs. Database State (continued)

- Distinction
  - The database schema changes very infrequently.
    - schema evolution
  - The database state changes every time the database is updated.

- Schema is also called intension.
  - DBMS stores the descriptions of schema constructs and constraints, also called the metadata, in the DBMS catalog.

- State is also called extension.
Three-Schema Architecture

- Proposed to support DBMS characteristics of:
  - Self-describing (use of a catalog to store schema)
  - Program-data independence
  - Support of multiple views of the data

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

- Most DBMS do not separate the three levels completely and explicitly, but they support the three-schema architecture to some extent
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 **representational** (or **implementation**) data model which is often based on a conceptual schema design in a high-level data model
  - **External schemas** at the **external level** to describe the various user views.
    - Each external schema describes the part of database that a particular user group is interested in and hide the rest of database.
    - Usually uses the same data model as the conceptual schema. (i.e., representational data model)
The three-schema architecture

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

- **External Level**
  - External/Conceptual Mapping

- **Conceptual Level**
  - Conceptual/Internal Mapping

- **Internal Level**

**End Users**

**External View**

**Conceptual Schema**

**Internal Schema**

**Stored Database**
Three-Schema Architecture

- The processes of transforming requests and results between levels are called mapping.
- 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.
  - For example, change GRADE_REPORT in Figure 1.2 to Figure 1.6(a)

- **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.
Data Independence (continued)

- 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 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 and internal schemas of a database.
  - In some DBMSs, separate storage definition language (SDL) and view definition language (VDL) are used to define internal and external schemas.
    - In most relational DBMSs, there is no specific language that performs the role of SDL. Instead, the internal schema is specified by a combination of functions, parameters, and specifications related to storage of files.
  - In most DBMSs, the DDL is used to define both the conceptual and external schemas (views).
  - In relational DBMSs, SQL is used in the role of VDL to define user or application views (actually SQL is a combination of DDL, VDL, and DML and other features).
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*).
Types of DML

- **High Level or Non-procedural Language:**
  - For example, the SQL relational language
  - Are “set”-oriented and specify *which data to retrieve* rather than *how to retrieve it*.
  - Also called *declarative languages*.
  - Can be used interactively or embedded in a host programming language

- **Low Level or Procedural Language:**
  - Retrieve data *one record-at-a-time*;
  - Constructs such as looping are needed to retrieve multiple records, along with positioning pointers.
  - Must be embedded in a host 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 (called **data sublanguage**) in programming languages (called **host language**)

- **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
The Database System Environment

- DBMS component modules
  - DDL compiler
  - Interactive query interface
    - Query compiler
    - Query optimizer
  - Precompiler
  - DML compiler
The Database System Environment (cont'd.)

- DBMS component modules
  - Buffer management
  - Stored data manager
  - Runtime database processor
  - System catalog
  - Concurrency control system
  - Backup and recovery system
Typical DBMS Component Modules

Figure 2.3
Component modules of a DBMS and their interactions.
Typical DBMS Component Modules

Figure 2.3
Component modules of a DBMS and their interactions.
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.
Other Tools

- **Application Development Environments and CASE (computer-aided software engineering) tools:**

- **Examples:**
  - PowerBuilder (Sybase)
  - JBuilder (Borland)
  - JDeveloper 10G (Oracle)
Centralized and Client-Server DBMS Architectures

- Centralized DBMS:
  - Combines everything into single 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.*
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.

Figure 2.6 Physical 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
Two Tier Client-Server Architecture

- Client and server must install appropriate client module and server module software for ODBC or JDBC.
- 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.
- See Chapter 10 for details on Database Programming.
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 (procedures or constraints) that are 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
  - Clients contain user interfaces and Web browsers
  - The client is typically a PC or a mobile device connected to the Web
Figure 2.7
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.
Classification of DBMSs

- Based on the data model used
  - Currently Used: Relational, Object-oriented, Object-relational
  - Recent Technologies: Key-value storage systems, NOSQL systems: document based, column-based, graph-based and key-value based.
  - Native XML DBMSs.

- Other classifications
  - Single-user (typically used with personal computers) vs. multi-user (most DBMSs).
  - Centralized (uses a single computer with one database) vs. distributed (multiple computers, multiple DBs)
Variations of Distributed DBMSs (DDBMSs)

- Homogeneous DDBMS
- Heterogeneous DDBMS
- Federated (or Multidatabase) Systems
  - Participating heterogeneous databases are loosely coupled with high degree of local autonomy.
- Distributed Database Systems (DDBMS) have now come to be known as client-server based database systems because:
  - They do not support a totally distributed environment, but rather a set of database servers supporting a set of clients.
Cost considerations for DBMSs

- **Cost Range**: from free open-source systems to configurations costing millions of dollars
- Examples of **free** relational DBMSs: MySQL, PostgreSQL, others
- **Commercial DBMS** 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.
Other Considerations

- **Type of access paths** within database system
  - E.g.- inverted indexing based (ADABAS is one such system). Fully indexed databases provide access by any keyword (used in search engines)

- **General Purpose vs. Special Purpose**
  - E.g.- Airline Reservation systems or many others- reservation systems for hotel/car etc. are special purpose OLTP (Online Transaction Processing Systems)
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