Chapter 2: Database Systems Architecture
Outline of Chapter 2

1 Data Models
2 Schemas and Instances
3 The Three Level Architecture
4 Data Independence
5 DBMS Languages
6 DBMS Interfaces
7 DB System Utilities
8 DBMS Classification
9 Overview of the data models
1. Data Models

• **A data model**: A set of concepts to describe the *structure* of a database.

• The **structure** of a Database refers to the *data types*, *relationships*, and *constraints* that the database should obey.

• A Data Model can also contain **user-defined operations** that specify database retrievals and updates by referring to the concepts of the data model.
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.

• **Implementation (record-oriented)** data models: Provide concepts that fall between the above two, balancing user views with some computer storage details.

*Object data models* can be seen as **higher-level implementation data models** that are closer to conceptual data models.
2. Schemas and Instances

- **Database Schema**: The *description* of a database. Includes descriptions of the *database structure* and the *constraints* that should hold on the database.

- **Schema Diagram**: A diagrammatic display of (some aspects of) a database schema.

Figure 2.1  Schema diagram for the database of Figure 1.2.

<table>
<thead>
<tr>
<th>STUDENT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>StudentNumber</td>
</tr>
<tr>
<td>COURSE</td>
<td></td>
</tr>
<tr>
<td>CourseName</td>
<td>CourseNumber</td>
</tr>
<tr>
<td>PREREQUISITE</td>
<td></td>
</tr>
<tr>
<td>CourseNumber</td>
<td>PrerequisiteNumber</td>
</tr>
<tr>
<td>SECTION</td>
<td></td>
</tr>
<tr>
<td>SectionIdentifier</td>
<td>CourseNumber</td>
</tr>
<tr>
<td>GRADE_REPORT</td>
<td></td>
</tr>
<tr>
<td>StudentNumber</td>
<td>SectionIdentifier</td>
</tr>
</tbody>
</table>
2. Schemas and Instances (Cont.)

- **Database Instance**: The actual data stored in a database at a *particular moment in time*. Also called *database state* (or *occurrence*).

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

  **Schema** is also called *intension*, whereas *state* is called *extension*. 
3. The three level Architecture

The three level architecture is proposed to support the DBMS characteristics of:

- **Program-data independence.**
- **Multiple views** of the data.

It defines DBMS schemas at **three levels**:

- **Internal schema** at the internal level to describe data storage structures and access paths. Typically uses a *physical data model*.

- **Conceptual schema** at the conceptual level to describe the structure and constraints for the *whole* database. 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 level*. 
3. The three level Architecture (Cont)

**Mappings** among schema levels are also needed. Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.

**Figure 2.2** Illustrating the three-schema architecture.
4. Data Independence

**Logical Data Independence**: The capacity to change the conceptual schema without having to change the external schemas and their application programs.

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.

**Physical Data Independence**: The capacity to change the internal schema without having to change the conceptual schema.
5. 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 respectively.

**Data Manipulation Language (DML):** Used to specify database retrievals and updates.

In current DBMS a *comprehensive integrated language* is used that includes constructs for *conceptual schema definition, view definition and schema and data manipulation.*
5. DBMS Languages (Cont.)

DML commands (data sublanguage) can be embedded in a general-purpose programming language (host language), such as COBOL, PL/1 or PASCAL. Alternatively, stand-alone DML commands can be applied directly (query language).

A DML that can be used on its own to specify complex DB operations in a concise manner is called high-level DML (as opposed to a low-level DML.)

A high-level DML can specify and retrieve many records in a single DML statement (set-at-a-time DML). A low-level DML retrieves and process each time one record from a set of records (record-at-a-time DML).

A query in a high-level DML often specifies which data to retrieve (declarative language) rather than how to retrieve it (procedural language).
6. DBMS Interfaces

- Stand-alone query language interfaces.
- Programmer interfaces for embedding DML in programming languages:
  - Pre-compiler Approach
  - Procedure (Subroutine) Call Approach
- User-friendly interfaces:
  - Menu-based
  - Graphics-based (Point and Click, Drag and Drop etc.)
  - Forms-based
  - Natural language
  - Combinations of the above
  - Speech as Input (?) and Output
  - Web Browser as an interface
- Parametric interfaces using function keys.
- Report generation languages.
- Interfaces for the DBA:
  - Creating accounts, granting authorizations
  - Setting system parameters
  - Changing schemas or access paths
Most DBMS have utilities that help the DBA to perform certain functions such as:

- **Loading** data stored in files into a database.
- **Backing up** the database periodically on tape.
- **Reorganizing** database file structures.
- **Generating reports**.
- **Monitoring performance**.
- Other functions, such as **sorting, user monitoring, data compression**, etc.

The **Data dictionary / repository** is used to store schema descriptions and other information such as design decisions, application program descriptions, user information, usage standards, etc.
8. DBMS Classification

DBMS can be classified according to various criteria.

• Based on the data model used:
  - *Traditional*: Relational, Network, Hierarchical.
  - *Emerging*: Object-oriented, Object-relational.

• Based on the number of users:
  - *Single-user* (typically used with micro-computers)
  - *multi-user* (most DBMSs).

• Based on the cost:
  DBMSs usually range from $100 to $3,000.

• Based on the generality:
  - *Special purpose DBMS*
  - *General purpose DBMS*. 
8. DBMS Classification (Cont.)

• **Based on the number of sites:**
  - *Centralized* (uses a single computer with one database)
  - *Distributed* (uses multiple computers, multiple databases)

Distributed Database Systems 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.

The DBMS of the different DBs in a Distributed DBMS can be the same (**Homogeneous DBMSs**) or different (**Heterogeneous DBMSs**).
9. Overview of the data models (1)

The **Relational data model** is based on the notion of **Relation**.

It represents a database as a **collection of tables**.

Most relational databases use a **high-level query language called SQL** and support user views.

**Example:**

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>Name</th>
<th>StudentNumber</th>
<th>Class</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>17</td>
<td>1</td>
<td>CS</td>
<td></td>
</tr>
<tr>
<td>Brown</td>
<td>8</td>
<td>2</td>
<td>CS</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COURSE</th>
<th>Course Name</th>
<th>Course Number</th>
<th>Credit Hours</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intro to Computer Science</td>
<td>CS3110</td>
<td>4</td>
<td>CS</td>
<td></td>
</tr>
<tr>
<td>Data Structure</td>
<td>CS3300</td>
<td>4</td>
<td>CS</td>
<td></td>
</tr>
<tr>
<td>Discrete Mathematics</td>
<td>MATH3110</td>
<td>3</td>
<td>MATH</td>
<td></td>
</tr>
<tr>
<td>Database</td>
<td>CS3300</td>
<td>3</td>
<td>CS</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SECTION</th>
<th>Section Identifier</th>
<th>Course Number</th>
<th>Semester</th>
<th>Year</th>
<th>Instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>85</td>
<td>MATH210</td>
<td>Fall</td>
<td>95</td>
<td>King</td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>CS1310</td>
<td>Fall</td>
<td>95</td>
<td>Anderson</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>CS3320</td>
<td>Spring</td>
<td>95</td>
<td>Knuth</td>
<td></td>
</tr>
<tr>
<td>112</td>
<td>MATH210</td>
<td>Fall</td>
<td>95</td>
<td>Chang</td>
<td></td>
</tr>
<tr>
<td>119</td>
<td>CS1310</td>
<td>Fall</td>
<td>95</td>
<td>Anderson</td>
<td></td>
</tr>
<tr>
<td>135</td>
<td>CS3300</td>
<td>Fall</td>
<td>95</td>
<td>Stone</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GRADE_REPORT</th>
<th>StudentNumber</th>
<th>Section Identifier</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>112</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>119</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>85</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>92</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>102</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>135</td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PREREQUISITE</th>
<th>Course Number</th>
<th>PrerequisiteNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS3350</td>
<td>CS3300</td>
<td></td>
</tr>
<tr>
<td>CS3320</td>
<td>MATH3110</td>
<td></td>
</tr>
<tr>
<td>CS3320</td>
<td>CS1310</td>
<td></td>
</tr>
</tbody>
</table>
9. Overview of the data models (2)

The **Object data model** defines a database in terms of **objects**, their **properties**, and their **operations**.

Objects belong to **classes**; classes are organized into **hierarchies**; the operations of each class are specified in terms of predefined procedures called **methods**.

Many Object DBMS use a **high-level query language** called **OQL**.

The model of relational DBMS has been extended to incorporate object database concepts and other capabilities (**Object-Relational systems**).
The **Network data model** represents data as record types and 1:N relationships (called set types).

It has an associated record-at-a-time language that must be embedded in a host programming language.

**Example:**

![Figure 2.4](image) The schema of Figure 2.1 in the notation of the network data model.
The **Hierarchical data model** represents data as hierarchical tree structures (parent-child relationship types)

There is no standard language. Most hierarchical DBMS use a record-at-a-time language.

**Example:**

*Figure D.2*  Occurrences of Parent-Child Relationships.  
(a) Two occurrences of the PCR type (DEPARTMENT, EMPLOYEE).  
(b) Two occurrences of the PCR type (DEPARTMENT, PROJECT).