Module 2

DBMS ARCHITECTURE

Data Models

- A **data model** is a collection of concepts that can be used to describe the structure of a database.
- By structure of a database means the data types, relationships, and constraints that apply to the data.
- Most data models also include a set of basic operations for specifying retrievals and updates on the database.
- **Data Abstraction** is the suppression of details of data organization and storage, and the highlighting of the essential features for an improved understanding of data(simply means general features). In order to achieve the data abstraction data model is used.

Categories of data models

- Hight level or Conceptual data model
- Low level or Physical data model
- Representational or Implementational data model

High-level or conceptual data models

- In general it provide concepts that are close to the way many users perceive data. Conceptual data models use concepts such as **entities, attributes, and relationships.**
- An **entity** represents a real-world object or concept, such as an employee or a project from the miniworld that is described in the database.
- An **attribute** represents some property of interest that further describes an entity, such as the employee's name or salary.
- The **relationship** among two or more entities represents an association among the entities, for example, a works-on relationship between an employee and a project.

Low-level or physical data models

- It describe how data is stored as files in the computer by representing information such as record formats, record orderings, and access paths.
- An access path is a search structure that makes the search for particular database records efficient, such as **indexing or hashing**.
- An index is an example of an access path that allows direct access to data using an index term or a keyword. It is similar to the index at the end of this text, except that it may be organized in a linear, hierarchical (tree-structured), or some other fashion.

Representational or Implementation Data Model

- This type of data model is used to represent only the logical part of the database and does not represent the physical structure of the database. The representational data model allows us to focus primarily, on the design part of the database.
- A popular representational model is a Relational model. The relational Model consists of Relational Algebra and Relational Calculus.
- In the Relational Model, we basically use tables to represent our data and the relationships between them. It is a theoretical concept whose practical implementation is done in Physical Data Model.
- The advantage of using a Representational data model is to provide a foundation to form the base for the Physical model

- Another class of data models is known as **self-describing data models**.
- The data storage in systems based on these models combines the description of the data with the data values themselves. In traditional DBMSs, the description (schema) is separated from the data. These models include XML as well as many of the key-value stores and NOSQL systems that were recently created for managing big data.

Schemas

- In data model, it is important to distinguish between the **description** of the database and the database itself.
- The description of a database is called the **database schema**, which is specified during database design and is not expected to change frequently.
- Most data models have certain conventions for displaying schemas as diagrams.
- A displayed schema is called a **schema diagram.** Each object in the schema—such as STUDENT or COURSE—a **schema construct**

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

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

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	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	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

Instance

- The actual data in a database may change quite frequently
- The database shown in Figure changes every time we add a new student or enter a new grade.
- The data in the database at a particular moment in time is called a **database state or snapshot**. It is also called the **current set of occurrences or instances** in the database.

INSTANCE

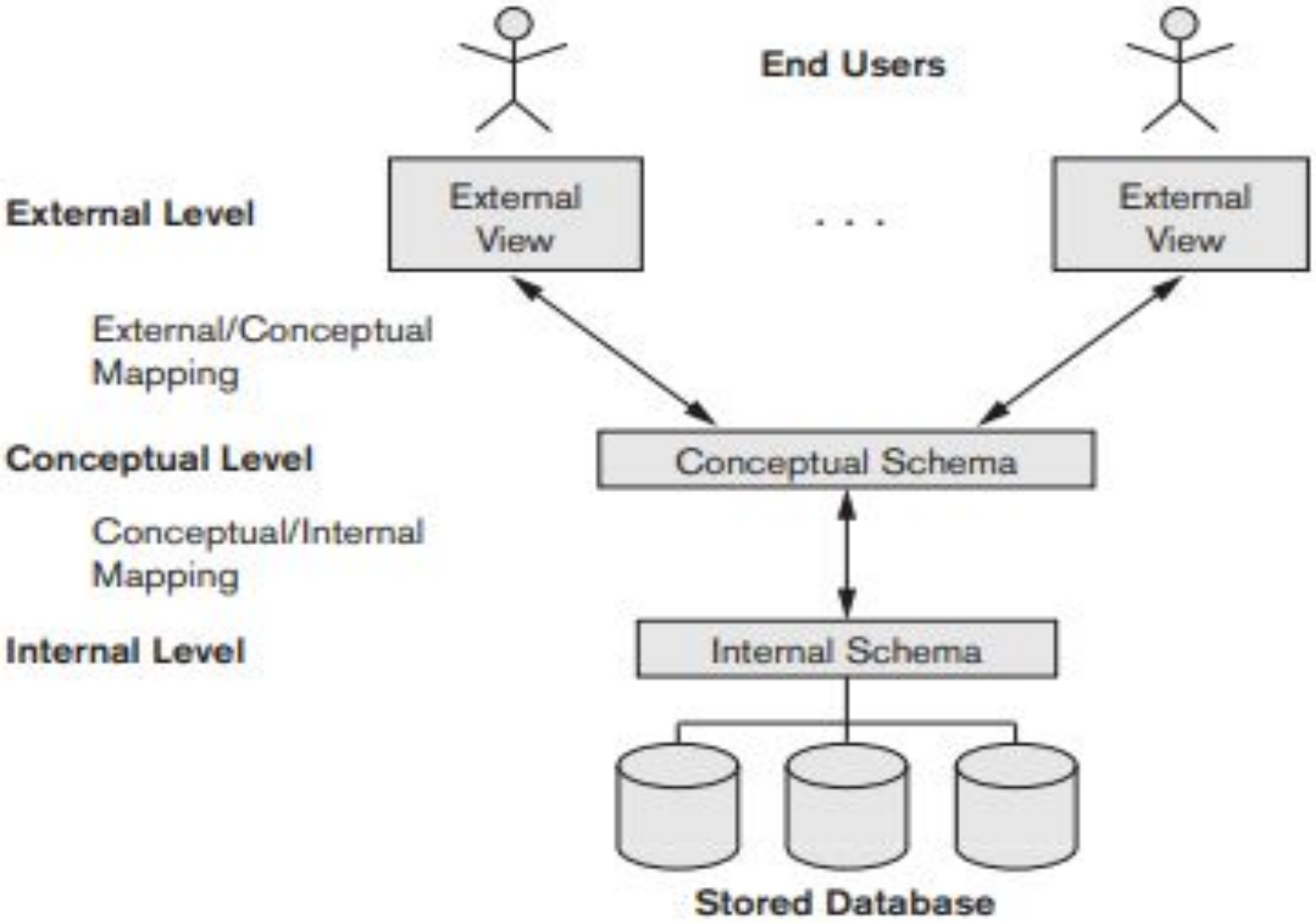
- In a given database state, each schema construct has its own current set of instances; for example, the STUDENT construct will contain the set of individual student entities (records) as its instances
- Many database states can be constructed to correspond to a particular database schema.
- Every time we insert or delete a record or change the value of a data item in a record, we change one state of the database into another state.
- The distinction between **database schema and database state** is very important. When we define a new database, we specify its **database schema** only to the DBMS. At this point, the corresponding database state is the **empty state with no data**.

- The initial state of the database when the database i**s first populated or loaded with the initial data**. From then on, every time an update operation is applied to the database, we get another database state.
- At any point in time, the database has a current state. The DBMS is partly responsible for ensuring that every state of the database is a valid state—that is, a state that satisfies the structure and constraints specified in the schema.
- Hence, specifying a correct schema to the DBMS is extremely important and the schema must be designed with utmost care. The DBMS stores the descriptions of the schema constructs and constraints called the **meta-data**—in the DBMS catalog so that DBMS software can refer to the schema whenever it needs to.
- The schema is sometimes called the **intension**, and a database state is called an **extension** of the schema.

SCHEMA EVOLUTION

- The schema is not supposed to change frequently, it is not uncommon that changes occasionally need to be applied to the schema as the application requirements change.
- For example, we may decide that another data item needs to be stored for each record in a file, such as adding the Date_of_birth to the STUDENT schema . This is known as **schema evolution.**

DBMS Architecture



- The goal of the three-schema architecture is to separate the user applications from the physical database. In this architecture, schemas can be defined at the following three levels:

1. The internal level has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database

2. The conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. Usually, a representational data model is used to describe the conceptual schema when a database system is implemented. This implementation conceptual schema is often based on a conceptual schema design in a high-level data model.

3. The external or view level includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. As in the previous level, each external schema is typically implemented using a representational data model, possibly based on an external schema design in a high-level conceptual data model.

- The three schemas are only descriptions of data; the actual data is stored at the physical level only.
- In the three-schema architecture, each user group refers to its own external schema.

- Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database.
- If the request is a database retrieval, the data extracted from the stored database must be reformatted to match the user's external view.
- The processes of transforming requests and results between levels are called **mappings**.
- These mappings may be time-consuming, so some DBMSs—especially those that are meant to support small databases—do not support external views. Even in such systems, however, it is necessary to transform requests between the conceptual and internal levels.

DATA INDEPENDENCE

- Logical data independence
- Physical data independence

Logical data independence:

- Logical data independence is the capacity to change the conceptual schema without having to change external schemas or application programs.
- It allows us to make changes in a conceptual structure like adding, modifying, or deleting an attribute in the database.

- These changes can be done at a logical level without affecting the application program or external layer.
  - Adding, deleting, or modifying the entity or relationship.
  - Merging or breaking the record present in the database.
- The view definition and the mappings need to be changed in a DBMS that supports logical data independence

Example:

Banking application

Physical data independence:

- Physical data independence is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas need not be changed as well.
- Changes to the internal schema may be needed because some physical files were reorganized—for example, by creating additional access structures—to improve the performance of retrieval or update.
- Physical data independence gives us the freedom to modify the - Storage device, File structure, location of the database, etc. without changing the definition of conceptual or view level.

- Below changes can be done at the physical layer without affecting the conceptual layer –
  - Changing the storage devices like SSD, hard disk and magnetic tapes, etc.
  - Changing the access technique and modifying indexes.
  - Changing the compression techniques or hashing algorithms.

For example: if we take the database of the banking system and we want to scale up the database by changing the storage size and also want to change the file structure, we can do it without affecting any functionality of logical schema.

Database Languages:

- DDL: Data Definition Language(Create, Alter, Drop, Truncate)
- DML: Data Manipulation Language(Insert, Update, Delete)
- DQL: Data Query Language(Select)
- DCL: Data Control Language(Grant, Revoke)
- TCL: Transaction Control Language(Commit, Rollback, Savepoint)

DBMS Interfaces

Menu-based Interfaces for Web Clients or Browsing.

- These interfaces present the user with lists of options (called menus) that lead the user through the formulation of a request.
- Menus do away with the need to memorize the specific commands and syntax of a query language; rather, the query is composed step-by step by picking options from a menu that is displayed by the system.
- Pull-down menus are a very popular technique in Web-based user interfaces. They are also often used in browsing interfaces, which allow a user to look through the contents of a database in an exploratory and unstructured manner.

Apps for Mobile Devices. These interfaces present mobile users with access to their data.

- For example, banking, reservations, and insurance companies, among many others, provide apps that allow users to access their data through a mobile phone or mobile device.
- The apps have built-in programmed interfaces that typically allow users to login using their account name and password; the apps then provide a limited menu of options for mobile access to the user data, as well as options such as paying bills (for banks) or making reservations (for reservation Web sites).

Forms-based Interfaces. A forms-based interface displays a form to each user.

- Users can fill out all of the form entries to insert new data, or they can fill out only certain entries, in which case the DBMS will retrieve matching data for the remaining entries.
- Forms are usually designed and programmed for naive users as interfaces to scanned transactions. Many DBMSs have forms specification languages, which are special languages that help programmers specify such forms.
- SQL*Forms is a form-based language that specifies queries using a form designed in conjunction with the relational database schema.
- Oracle Forms is a component of the Oracle product suite that provides an extensive set of features to design and build applications using forms.
- Some systems have utilities that define a form by letting the end user interactively construct a sample form on the screen

Graphical User Interfaces. A GUI typically displays a schema to the user in diagrammatic form.

- The user then can specify a query by manipulating the diagram.
- In many cases, GUIs utilize both menus and forms.

Keyword-based Database Search. These are somewhat similar to Web search engines, which accept strings of natural language (like English or Spanish) words and match them with documents at specific sites (for local search engines) or Web pages on the Web at large (for engines like Google or Ask).

- They use predefined indexes on words and use ranking functions to retrieve and present resulting documents in a decreasing degree of match.
- Such “free form” textual query interfaces are not yet common in structured relational databases, although a research area called keyword-based querying has emerged recently for relational databases.

- **Speech Input and Output**. Limited use of speech as an input query and speech as an answer to a question or result of a request is becoming commonplace.
- Applications with limited vocabularies, such as inquiries for telephone directory, flight arrival/departure, and credit card account information, are allowing speech for input and output to enable customers to access this information.
- The speech input is detected using a library of predefined words and used to set up the parameters that are supplied to the queries.
- For output, a similar conversion from text or numbers into speech takes place.

Interfaces for Parametric Users. Parametric users, such as bank tellers, often have a small set of operations that they must perform repeatedly.

- For example, a teller is able to use single function keys to invoke routine and repetitive transactions such as account deposits or withdrawals, or balance inquiries.
- Systems analysts and programmers design and implement a special interface for each known class of naive users.
- Usually a small set of abbreviated commands is included, with the goal of minimizing the number of keystrokes required for each request.

Interfaces for the DBA. Most database systems contain privileged commands that can be used only by the DBA staff.

- These include commands for creating accounts, setting system parameters, granting account authorization, changing a schema, and reorganizing the storage structures of a database

Classifications of Database Management Systems

1. Based on Data Model
2. Based on Number of Users
3. Based on Database Distribution
4. Based on Cost of Database
5. Based on Usage
6. Based on Flow Control

1. Based on Data Model:

- The data model defines the physical and logical structure of a database which involves the data types, the relationship among the data, constraints applied on the data and even the basic operations specifying retrieval and updating of data in the database. Depending upon how the data is structured, data models are further classified into:

a. Relational Data Model In the relational data model, we use tables to represent data and the relationship among that data. Each of the tables in the relational data model has a unique name. A table has multiple columns where each column name is unique. A table holds records which has value for each column of the table. We also refer to the relational database model as a record-based data model as it holds records of fixed-format. The relational database model is the most currently used data model.

b. **Entity-Relationship Model** The Entity-Relationship model (E-R data model) represents data using objects and the relationship among these objects. These objects are referred to as entities that represent the real ‘thing’ or ‘object’ in the real world. Each entity in the E-R model is distinguishable from other entities in the model. Like, relational model, the E-R model is also used widely used to design the database.

c. **Object-Based Data Model** Nowadays, object-oriented programming such as Java, C++, etc. is widely used to develop most of the software. This motivated the development of an object-based data model. The object-based data model is an extension of the E-R model which also include notion for encapsulation, methods. There is also an object-relational data model which is a combination of the object-oriented data model and relational data model.

d. **Semi structured Data Model** The semi structured data model is different from what we have studied above. In the semi structured data model, the data items or objects of the same kind might have a different set of attributes. The Extensible Markup Language (XML) represents the semi structured data.

2. **Classification Based on Number of Users** The database management system can also be classified on the basis of its user. So, a DBMS can either be used by a single user or it can be used by multiple users. The database system that can be used by a single user at a time is referred to as a single-user system and the database system that can be used by multiple users at a time is referred to as a multiple user system.

3. **Based on Database Distribution** Depending on the distribution of the database over numerous sites we can classify the database as:

a. **Centralized DBMS** In the centralized DBMS, the entire database is stored in a single computer site. Though the centralized database support multiple users still the DBMS software and the data both are stores on a single computer site.

b. **Distributed DBMS** In the distributed DBMS (DDBMS) the database and the DBMS software are distributed over many computer sites. These computer sites are connected via a computer network. The DDBMS is further classified as homogeneous DDBMS and heterogeneous DDBMS.

- Homogeneous DDBMS: The homogeneous DDBMS has the same DBMS software at all the distributed sites.
- Heterogeneous DDBMS: The heterogeneous DDBMS has different DBMS software for different sites

4. **Based on Cost of Database** Well, it is quite difficult to classify the database on the basis of its cost as nowadays you can have free open source DBMS products such as MySQL and PostgreSQL. Although the personal version of RDBMS can cost up to $100. The large systems along with the components that can handle distribution of database, replication of database, parallel processing, mobile capability and so on can be sold in the form of licenses. The site license allows unlimited use while another kind of license limits the number of concurrent licenses. Some systems with single-user versions such as Microsoft Access are sold per copy or included in the configuration of your desktop or laptop. You may also have to pay millions of dollars for the installation and maintenance of a large database system.

5**. Classification Based on Usage** On the basis of the access path that is used to store the files, the database can be classified as general-purpose DBMS and special-purpose DBMS.

- The special-purpose DBMS is the one that is designed for a specific application and it can not be used for another application without performing any major changes we refer to this as online transaction processing (OLTP).
- The OLTP system supports a large number of transactions concurrently without any delay.
- The general-purpose DBMS is the one that is designed to meet the need of as many applications as possible.

6. Based on Flow Control

- With the **passive database management** system, the user needs to specify the query to the current state of the database system to retrieve the desired information. It is similar to traditional DBMS where the user or the application program is responsible to initiate the operation. As the application forwards the request to DBMS and waits for DBMS to process the query or requested operation and provide a possible answer. The requested operation may be used to define or update the schema of the database or retrieve or update the data within the database. The passive database management systems are also referred to as program driven systems.
- The **active database management** system on other hand are referred to as data-driven systems or event-driven systems where the control flow between the application and DBMS is based on the occurrence of an event. Inactive DBMS the control flow is two ways i.e. application can call DBMS and even DBMS can call application.

- **Inactive DBMS** the users need to specify what data they require. The DBMS processes the request and if the requested information is currently available then it provides it to the user. If the requested data is not available currently then inactive DBMS the scope of query also includes the future data i.e., the DBMS will monitor the arrival of desired information in future data and will provide it to the relevant user