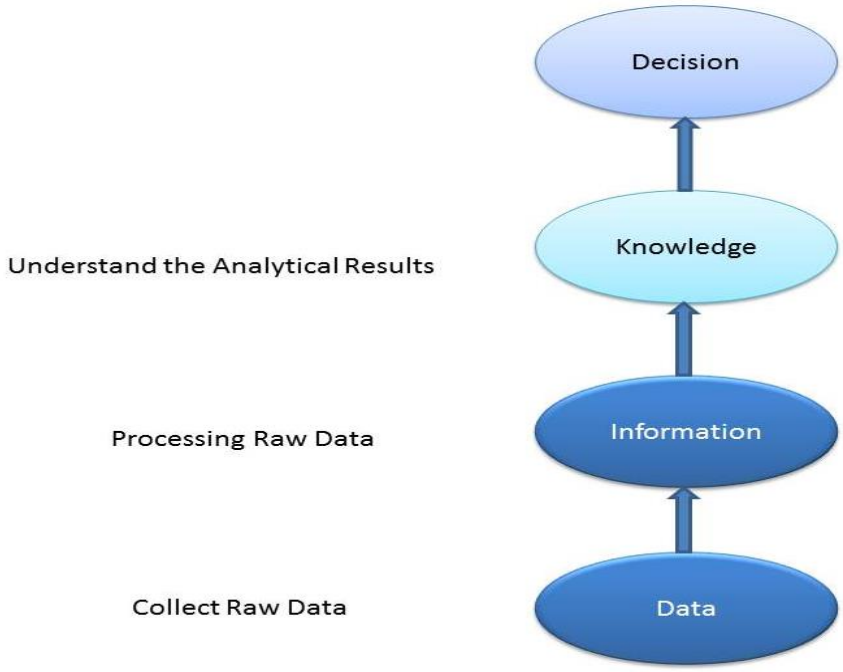
Introduction to Database Management System

Contents

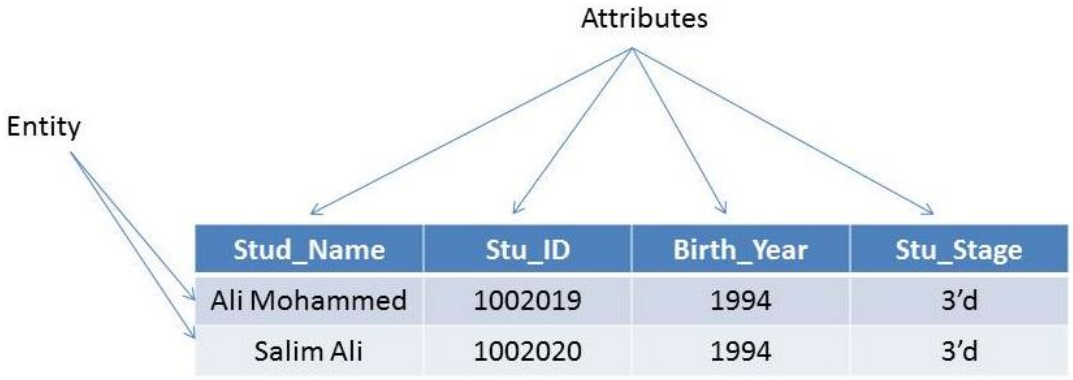
1. Introduction

- A single piece of **data** is a single **fact** about something we are interested in.
- In any environment there are things that are important to you and there are facts about those things that are worth remembering.
- A ‘‘thing’’ as a concept is broad enough to include a person, an organization like a company, or an event that took place such as a particular meeting.
- **Information** is the result of processing raw data to reveal its meaning.
- Data processing can be as simple as organizing data to reveal patterns or as complex as making forecasts or drawing inferences using statistical modeling.
- A database is a shared, integrated computer structure that stores a collection of: **End-user data**: that is, raw facts of interest to the end user, and **Metadata:** or data about data, through which the end-user data are integrated and managed.



2. Database Concepts

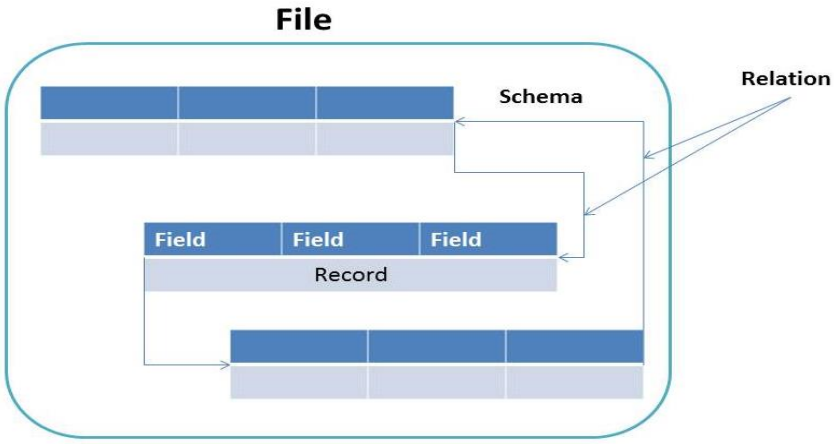
- Thing or object in our environment that we want to keep track of is called an **entity**.
- A collection of entities of the same type (e.g., all the company's employees) is called an **entity set**.
- An **attribute** is a property of, a characteristic of, or a fact that we know about an entity.



Stud_Name	Stu_ID	Birth_Year	Stu_Stage
Ali Mohammed	1002019	1994	3'd
Salim Ali	1002020	1994	3'd

- **Field** is a character or group of characters (alphabetic or numeric) that has a specific meaning. A field is used to define and store data.
- **Record** is a logically connected set of one or more fields that describes a person, place, or thing. For example, the fields that constitute a record for a customer named Samir Ali might consist of Samir Ali's name, address, phone number, date of birth, credit limit, and unpaid balance.
- **File** is a collection of related records.
- The **schema**, which is the conceptual organization of the entire database as viewed by the database administrator. The schema includes a definition of the database name, the record type for each record, and the components that make up those records.

- The **subschema**, which defines the portion of the database "seen" by the application programs that actually produce the desired information from the data contained within the database. The existence of subschema definitions allows all application programs to simply invoke the subschema required to access the appropriate database file(s).

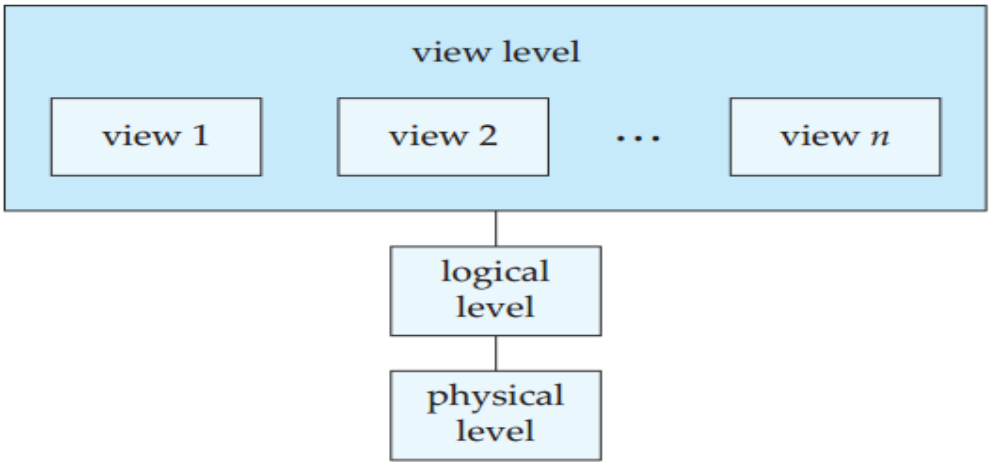


3. View of Data

- A major purpose of a database system is to provide users with an *abstract* view of the data.
- The system hides certain details of how the data are stored and maintained

3.1. Data Abstraction

- For the system to be usable, it must retrieve data efficiently. The need for efficiency has led designers to use complex data structures to represent data in the database.



- Since many database system users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users' interactions with the system:

  • **Physical level**. The lowest level of abstraction describes *how* the data are actually stored. The physical level describes complex low-level data structures in detail.

```
type instructor = record
                    ID : char (5);
                    name : char (20);
                    dept_name : char (20);
                    salary : numeric (8,2);
                  end;
```

  • **Logical level**. The next-higher level of abstraction describes *what* data are stored in the database, and what relationships exist among those data. The logical level thus describes the entire database in terms of a small number of relatively simple structures. Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity.

This is referred to as **physical data independence**. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.

• **View level**. The highest level of abstraction describes only part of the entire database. Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database. Many users of the database system do not need all this information; instead, they need to access only a part of the database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.

3. 2 Instances and Schemas

- The collection of information stored in the database at a particular moment is called an **instance** of the database.
- The overall design of the database is called the database **schema**. Schemas are changed infrequently, if at all. The concept of database schemas and instances can be understood by analogy to a program written in a programming language. A database schema corresponds to the variable declarations (along with associated type definitions) in a program. Each variable has a particular value at a given instant.
- Database systems have several schemas, partitioned according to the levels of abstraction. The **physical schema** describes the database design at the physical level, while the **logical schema** describes the database design at the logical level.
- A database may also have several schemas at the view level, sometimes called **subschemas**, that describe different views of the database.

3. 3 Data Models

- **Data model**: a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints. A data model provides a way to describe the design of a database at the physical, logical, and view levels. The data models can be classified into four different categories :

  • **Relational Model**. The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. Tables are also known as **relations**. The relational model is an example of a record-based model. Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes.

  • **Entity-Relationship Model**. The entity-relationship (E-R) data model uses a collection of basic objects, called *entities*, and *relationships* among these objects.

  • **Object-Based Data Model**. Object-oriented programming (especially in Java, C++, or C#) has become the dominant software-development methodology. This led to the development of an object-oriented data model that can be seen as extending the E-R model with notions of encapsulation, methods (functions), and object identity.

• **Semi-structured Data Model**. The semi-structured data model permits the specification of data where individual data items of the same type may have different sets of attributes. This is in contrast to the data models mentioned earlier, where every data item of a particular type must have the same set of attributes. The **Extensible Markup Language (XML)** is widely used to represent semi-structured data.

4. File System

- In the recent past, a manager of almost any small organization was able to keep track of necessary data by using a manual file system.
- Such a file system was traditionally composed of a collection of file folders, each properly tagged and kept in a filing cabinet.
- Ideally, the contents of each file folder were logically related. For example, a file folder in a doctor's office might contain patient data, one file folder for each patient. All of the data in that file folder would describe only that particular patient's medical history.
- As long as a data collection was relatively small and an organization's managers had few reporting requirements, the manual system served its role well as a data repository.

C_NAME	C_PHONE	C_ADDRESS	C_ZIP	A_NAME	A_PHONE	TP	AMT	REN
Alfred A. Ramas	615-844-2573	218 Fork Rd., Babs, TN	36123	Leah F. Hahn	615-882-1244	T1	100.00	05-Apr-2008
Leona K. Dunne	713-894-1238	Box 12A, Fox, KY	25246	Alex B. Alby	713-228-1249	T1	250.00	16-Jun-2008
Kathy W. Smith	615-894-2285	125 Oak Ln, Babs, TN	36123	Leah F. Hahn	615-882-2144	S2	150.00	29-Jan-2009
Paul F. Olowski	615-894-2180	217 Lee Ln., Babs, TN	36123	Leah F. Hahn	615-882-1244	S1	300.00	14-Oct-2008
Myron Orlando	615-222-1672	Box 111, New, TN	36155	Alex B. Alby	713-228-1249	T1	100.00	28-Dec-2008
Amy B. O'Brian	713-442-3381	387 Troll Dr., Fox, KY	25246	John T. Okon	615-123-5589	T2	850.00	22-Sep-2008
James G. Brown	615-297-1228	21 Tye Rd., Nash, TN	37118	Leah F. Hahn	615-882-1244	S1	120.00	25-Mar-2009
George Williams	615-290-2556	155 Maple, Nash, TN	37119	John T. Okon	615-123-5589	S1	250.00	17-Jul-2008
Anne G. Farriss	713-382-7185	2119 Elm, Crew, KY	25432	Alex B. Alby	713-228-1249	T2	100.00	03-Dec-2008
Olette K. Smith	615-297-3809	2782 Main, Nash, TN	37118	John T. Okon	615-123-5589	S2	500.00	14-Mar-2009

C_NAME = Customer name
C_PHONE = Customer phone
C_ADDRESS = Customer address
C_ZIP = Customer zip code

A_NAME = Agent name
A_PHONE = Agent phone
TP = Insurance type
AMT = Insurance policy amount, in thousands of $
REN = Insurance renewal date

- Using the CUSTOMER file's contents, the Data Processing specialist wrote programs that produced very useful reports for the insurance company's sales department:
  - Monthly summaries that showed the types and amounts of insurance sold by each agent.
  - Monthly checks to determine which customers must be contacted for renewal.
  - Reports that analyzed the ratios of insurance types sold by each agent.
  - Periodic customer contact letters designed to summarize coverage and to provide various customer relations bonuses.

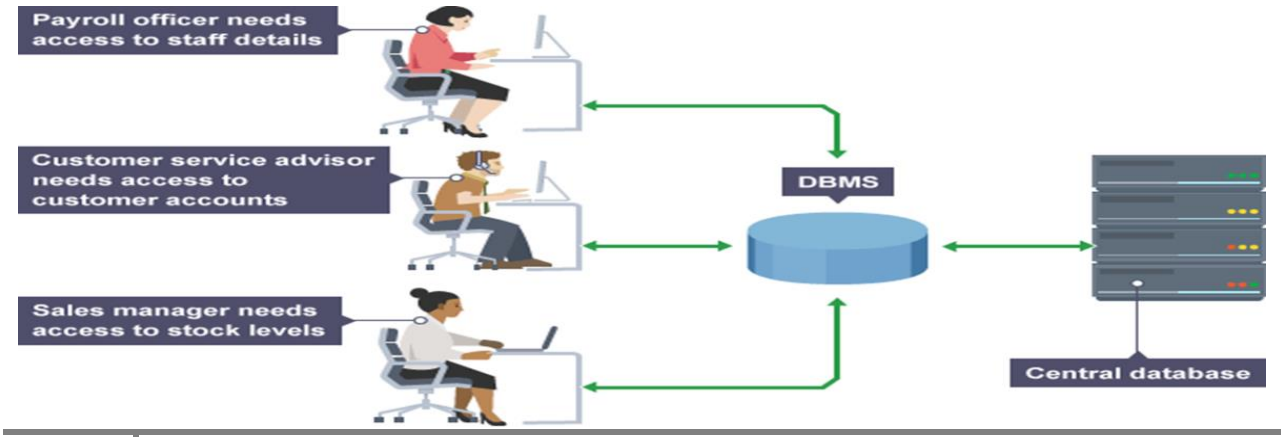
4.1 File System Problems

- The simplest **data retrieval** task requires extensive programming.
- The need to write programs to produce even the simplest reports makes **ad hoc queries impossible**.
- Making changes in an **existing structure** can be difficult in a file system environment.
- As the number of files in the system **expands**, system administration becomes more difficult.

- The **security features** are difficult to program and are, therefore, often omitted in a file system environment.

5. Database Management System (DBMS)

- Database Management System (**DBMS**) is a collection of inter-related data and a set of programs to access those data that provide a way to **store** and **retrieve** database information that is both convenient and efficient.
- Database systems are designed to **manage** large bodies of information. Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information.
- Database system must ensure the **safety of the information** stored, despite system crashes or attempts at unauthorized access.
- The DBMS serves as the **intermediary between the user and the database**. The database structure itself is stored as a collection of files, and the only way to access the data in those files is through the DBMS.
- The DBMS **receives all application requests** and translates them into the complex operations required to fulfill those requests.
- The DBMS hides much of the database's internal complexity from the application programs and users.

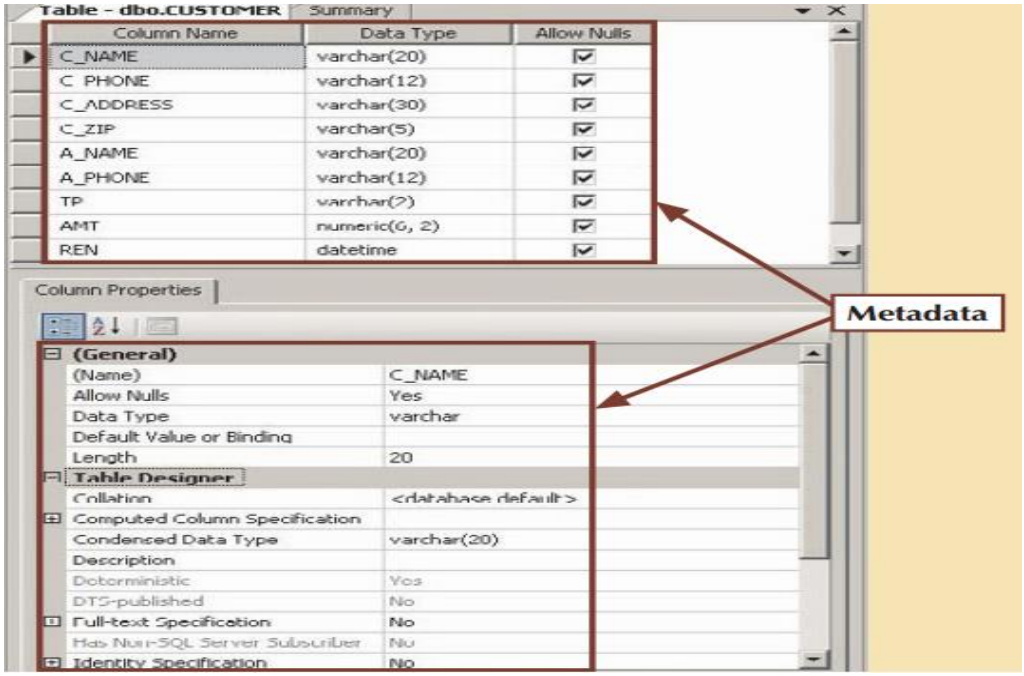


6. Functions of DBMS

- A DBMS performs several important functions that guarantee the integrity and consistency of the data in the database. Most of those functions are transparent to end users, and most can be achieved only through the use of a DBMS.

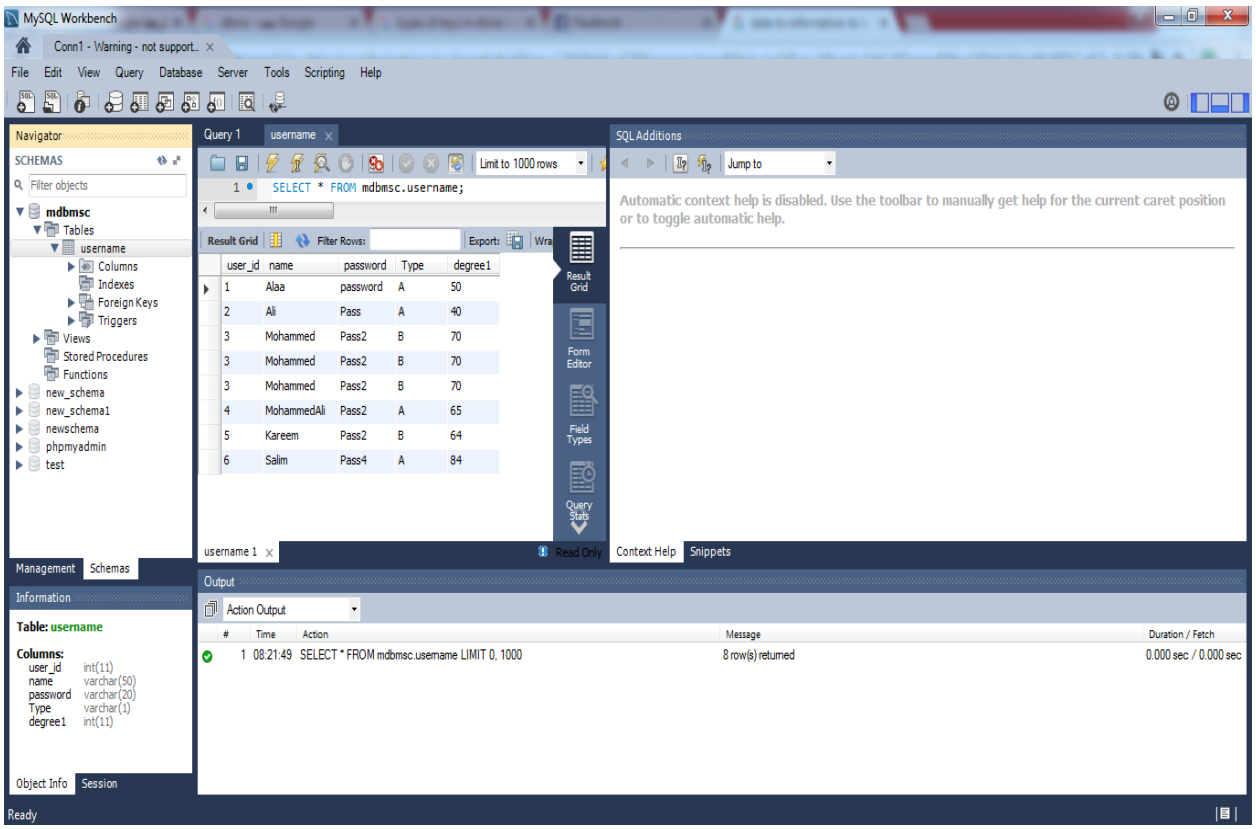
  1. ***Data dictionary management***.

  - The DBMS stores definitions of the data elements and their relationships (metadata) in a data dictionary.
  - The DBMS uses the data dictionary to look up the required data component structures and relationships, thus relieving you from having to code such complex relationships in each program.
  - Additionally, any changes made in a database structure are automatically recorded in the data dictionary, thereby freeing you from having to modify all of the programs that access the changed structure.



2. ***Data storage management***.

- The DBMS creates and manages the complex structures required for data storage, thus relieving you from the difficult task of defining and programming the physical data characteristics.
- A modern DBMS provides storage for the data, data entry forms or screen definitions, report definitions, data validation rules, procedural code, structures to handle video and picture formats, and so on.
- Data storage management is also important for database performance tuning. Performance tuning relates to the activities that make the database perform more efficiently in terms of storage and access speed.
- Although the user sees the database as a single data storage unit, the DBMS actually stores the database in multiple physical data files.



3. ***Data transformation and presentation*.**

- The DBMS transforms entered data to conform to required data structures.
- The DBMS relieves you of the chore of making a distinction between the logical data format and the physical data format.
- That is, the DBMS formats the **physically** retrieved data to make it conform to the user's **logical** expectations. For example, imagine an enterprise database used by a multinational company.
- An end user in England would expect to enter data such as July 11, 2008 as "11/07/2008." In contrast, the same date would be entered in the United States as "07/11/2008." Regardless of the data presentation format, the DBMS must manage the date in the proper format for each country.

4. ***Security management*.**

- The DBMS creates a security system that enforces user security and data privacy.
- Security rules determine which users can access the database, which data items each user can access, and which data operations (**read, add, delete, or modify**) the user can perform. This is especially important in **multiuser database systems**.

5. ***Multiuser access control*.**

- To provide **data integrity and data consistency**, the DBMS uses sophisticated algorithms to ensure that multiple users can access the

database concurrently without compromising the integrity of the database.

- **Data Consistency** refers to the usability of data; we want data to be constant in time and for us to be capable of using and showing them in different ways without changing their structure.

6. ***Backup and recovery management.***

- The DBMS provides backup and data recovery to ensure data safety and integrity. Current DBMS systems provide special utilities that allow the DBA to perform routine and special backup and restore procedures.
- Recovery management deals with the recovery of the database after a failure, such as a bad sector in the disk or a power failure. Such capability is critical to preserving the database's integrity.

7. ***Data integrity management.***

- The DBMS promotes and enforces integrity rules, thus **minimizing data redundancy and maximizing data consistency**.
- The data relationships stored in the data dictionary are used to enforce data integrity. Ensuring data integrity is especially important in **transaction-oriented database systems**.

8. ***Database access languages and application programming interfaces.***

- The DBMS provides data access through a query language. A query language is a nonprocedural language one that lets the user specify what must be done without having to specify how it is to be done.

- Structured Query Language (SQL) is the de facto query language and data access standard supported by the majority of DBMS vendors.

9. ***Database communication interfaces*.**
   - Current-generation DBMSs accept end-user requests via multiple, different network environments. For example, the DBMS might provide access to the database via the Internet through the use of Web browsers such as Mozilla Firefox or Microsoft Internet Explorer.
   - In this environment, communications can be accomplished in several ways:
     - End users can generate answers to queries by filling in screen forms through their preferred Web browser.
     - The DBMS can automatically publish predefined reports on a Web site.
     - The DBMS can connect to third-party systems to distribute information via e-mail or other productivity applications.

7. Types of DBMS

The most popular way of classifying databases today, however, is based on how they will be used and on the time sensitivity of the information gathered from them.

- *Online retailers*: For sales data noted above plus online order tracking, generation of recommendation lists, and maintenance of online product evaluations.
- *Banking*: For customer information, accounts, loans, and banking transactions.

- *Credit card transactions*: For purchases on credit cards and generation of monthly statements.
- *Finance*: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds; also for storing real-time market data to enable online trading by customers and automated trading by the firm.
- *Universities*: For student information, course registrations, and grades (in addition to standard enterprise information such as human resources and accounting).
- *Airlines*: For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner.
- *Telecommunication*: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks

8. History of DBMS

- **1950s and early 1960s**:
  - Data processing using magnetic tapes for storage.
    - Tapes provided only sequential access.
  - Punched cards for input.
- **Late 1960s and 1970s**:
  - Widespread use of **hard disks** in the late 1960s changed the scenario for data processing greatly.
    - Hard disks allowed direct access to data.
  - The position of data on disk was immaterial, since any location on disk could be accessed in just tens of milliseconds.

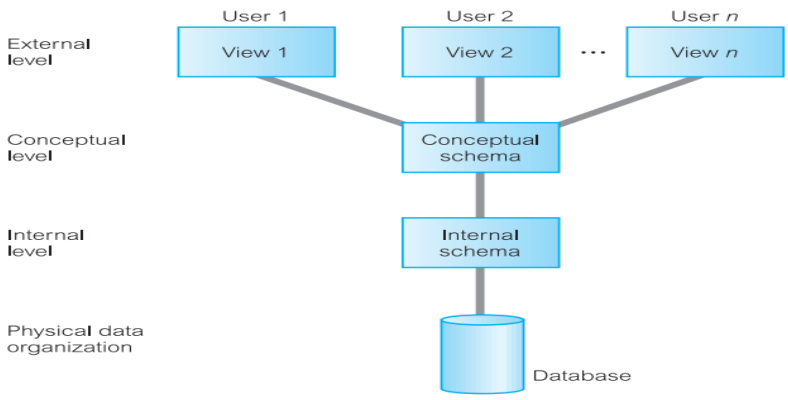
- **1980s**:
  - **IBM Research** developed techniques for the construction of an efficient **relational database system**.
  - The fully functional **System R** prototype led to IBM's first relational database product, SQL/DS.
  - At the same time, the **Ingres system** was being developed at the University of California at Berkeley. It led to a commercial product of the same name.
  - Initial commercial relational database systems, such as IBM DB2, Oracle, Ingres, and DEC Rdb, played a major role in advancing techniques for efficient processing of declarative queries.
  - The 1980s also saw much research on **parallel and distributed databases**, as well as initial work on **object-oriented databases**.
- **Early 1990s**:
  - The **SQL language** was designed primarily for decision support applications.
  - Decision support and querying reemerged as a major application area for databases.
  - Database vendors also began to add object-relational support to their databases.
- **1990s**:
  - The major event of the 1990s was the explosive growth of the **World Wide Web**.
  - Database systems now had to support very high transaction-processing rates, as well as very high reliability and 24 × 7 availability (availability 24 hours a day, 7 days a week, meaning no downtime for scheduled maintenance activities).

- **2000s**:
  - **XML** and the associated query language **XQuery** as a new database technology.
  - Open-source database systems are emerged, particularly **PostgreSQL and MySQL**. Several novel **distributed data-storage** systems have been built to handle the data management requirements of very large Web sites such as **Amazon, Facebook, Google, Microsoft and Yahoo**.
  - Data mining algorithms and **streaming data**, such as **stock-market ticker data or computer network monitoring data**.

9. Database Architecture

- An early proposal for a standard terminology and general architecture for database systems was produced in 1971 by the **DBTG** (Data Base Task Group).
- The DBTG recognized the need for a two-level approach with a system view called the **schema** and user views called **subschemas**.
- The American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARC), **ANSI/X3/SPARC**, produced a similar terminology and architecture in 1975, ANSI-SPARC recognized the need for a three-level approach with a system catalog.
- The levels form a three-level architecture comprising *an **external, a conceptual, and an internal level.***
- The way users perceive the data is called the **external level**.
- The way the DBMS and the operating system perceive the data is the **internal level**, where the data is actually stored using the data structures and files organizations.

- The **conceptual level** provides both the mapping and the desired independence between the external and internal levels.
- ***The objective of the three-level architecture*** is to separate each user's view of the database from the way the database is physically represented.
- There are several reasons why this separation is desirable:
    - Each user should be able to access the same data, but have a different customized view of the data. Each user should be able to change the way he or she views the data, and this change should not affect other users.
    - Users should not have to deal directly with physical database storage details, such as indexing or hashing. In other words, a user's interaction with the database should be independent of storage considerations.
    - The Database Administrator (DBA) should be able to change the database storage structures without affecting the users' views.
    - The internal structure of the database should be unaffected by changes to the physical aspects of storage, such as the changeover to a new storage device.
    - The DBA should be able to change the conceptual structure of the database without affecting all users.



9.1 External Level

- The external level consists of a number of different **external views** of the database. Each user has a view of the 'real world' represented in a form that is familiar for that user. The external view includes only those entities, attributes, and relationships in the 'real world' that the user is interested in.
- Other **entities, attributes, or relationships** that are not of interest may be represented in the database, but the **user will be unaware of them**.
- Different views may have different representations of the same data. For example, one user may view dates in the form (day, month, year), while another may view dates as (year, month, day).
- Views may include data **combined or derived** from several entities (Age from Birthdate).

9.2 Conceptual Level

- The community view of the database. This level describes what data level is stored in the database and the relationships among the data.
- This level contains the logical structure of the entire database as seen by the DBA.
- It is a complete view of the data requirements of the organization that is independent of any storage considerations.

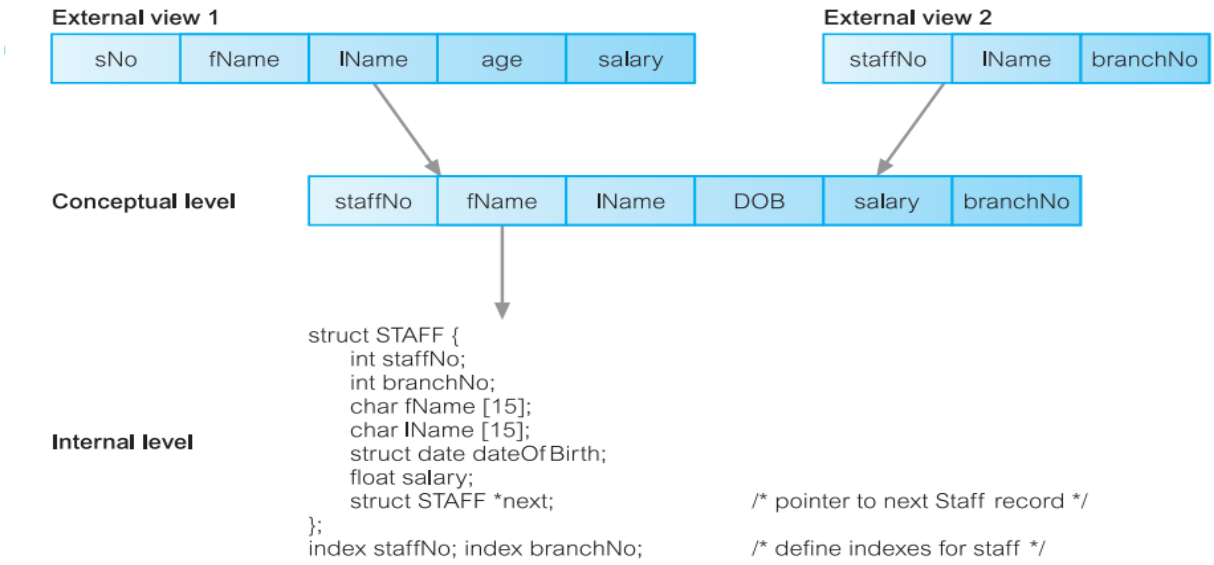
- The conceptual level represents:
    - All entities, their attributes, and their relationships;
    - The constraints on the data; semantic information about the data; security and integrity information.
- The conceptual level supports each external view.
- This level **must not contain any storage-dependent details**. For instance, the description of an entity should contain only data types of attributes (integer, real, character) and their length (maximum number of digits or characters), but not any storage considerations, such as the number of bytes occupied.

9.3 Internal Level

- This level describes how the data is stored in the database.
- The internal level covers the physical implementation of the database to achieve **optimal runtime performance** and **storage space utilization**.
- It covers the **data structures and file organizations** used to store data on storage devices. It interfaces with the operating system access methods (file management techniques for storing and retrieving data records) to place the data on the storage devices, build the indexes, retrieve the data, and so on.
- The internal level is concerned with such things as:
    - Storage space allocation for data and indexes;
    - Record descriptions for storage (with stored sizes for data items);
    - Record placement;
    - Data compression and data encryption techniques.

9.4 Physical Organization

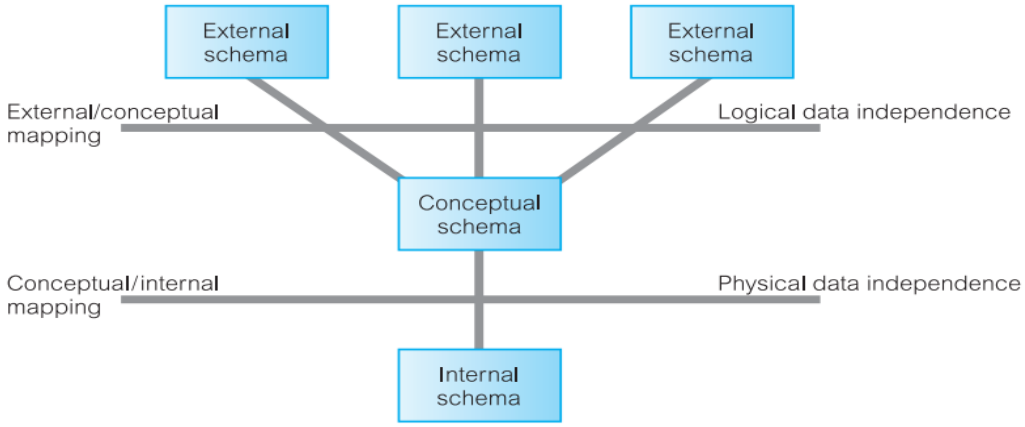
- Below the internal level there is a physical level that may be managed by the operating system under the direction of the DBMS. However, the functions of the DBMS and the operating system at the physical level are not clear-cut and vary from system to system.
- Some DBMSs take advantage of many of the operating system access methods, while others use only the most basic ones and create their own file organizations.



- The physical level below the DBMS consists of items only the operating system knows, such as exactly how the sequencing is implemented and whether the fields of internal records are stored as contiguous bytes on the disk.

10. Data Independency

- A major objective for the three-level architecture is to provide data independence, which means that upper levels are unaffected by changes to lower levels. There are two kinds of data independence: logical and physical.



- Logical data independence refers to the **immunity** of the external independence schemas to changes in the conceptual schema.
- **Changes to the conceptual schema**, such as the addition or removal of new entities, attributes, or relationships, should be possible without having to change existing external schemas or having to rewrite application programs.
- **Physical data independence** refers to the immunity of the conceptual independence schema to changes in the internal schema changes to the internal schema, such as using different file organizations or storage structures, using different storage devices, modifying indexes, or hashing algorithms, should be possible without having to change the conceptual or external schemas.
- From the users' point of view, the only effect that may be noticed is a change in **performance** (important for all DBAs).
- The two-stage mapping in the ANSI-SPARC architecture may be **inefficient**, but provides **greater data independence**. However, for more efficient mapping, the ANSI-SPARC model allows the direct mapping of external schemas on to the internal schema, thus by passing the conceptual schema. This, of course, reduces data independence, so that every time the internal schema changes, the external schema, and any dependent application programs may also have to change.