Course Objectives:

To provide knowledge of Data Base Management System.

1. INTRODUCTION

Objectives of this topic:

- ♦ To briefly understand what is data, database and database management system.
- ♦ Purpose and uses of Database course in the modern organisation.
- ♦ Various components of DataBase Management System(DBMS).

Definitions:

DataBase is a collection of information (data & associated objects) which help to organise the related information in a logical fashion for easy access and retrieval.

Database, basically is nothing more than a computer based record-keeping system. The overall purpose of database is to record and maintain information required by organisation in decision making.

A *Database Management System (DBMS)* is a software package designed to store and manage databases.

Data that is stored more or less permanently in a computer, we term a database. The software that allows one or many persons to use and/or modify this data is a *database management system* (DBMS).

A *Relational Data Base Management System* (RDBMS) stores data in many related Tables. The user can ask complex questions from one or more of these related tables and answers come back to the user as ***Forms*** and **Reports**.

Manual and Computer based Database:

Manual Database consists of filing cabinet, papers and people. There are formal filing method used to in/out the information from the filing cabinet. We might use calculator or computer spreadsheet to analyse the data further or to report it.

A **Computer Database** is nothing more than an automated version of the filing-and-retrieval functions of a manual paper filing system. Computer database store the information in a structured format in the form of ***Tables*** and ***Forms***. The information from the ***Tables*** and ***Forms*** are retrieve to our required format using ***Queries*** and ***Reports***.

Data base management system involves four major components:

1. Data
2. Hardware

3. Software
4. Users

Data is a information stored in the system and is partitioned into one or more databases. A database, then, is a repository for stored data. In general, it is both **integrated** and **shared**.

Database, when there is no redundancy among fields of different records is called integrated. For example, fields of record EMPLOYEE (Name, Address, Department, Salary etc) and record ENROLLMENT (representing training courses)

By shared we mean that individual pieces of data in the database may be shared among several different users, in the sense that each of those users may have access to the same piece of data. The sharing of data may be concurrent: meaning sharing of same pieces of data at the same time.

Hardware: the hardware consists of the secondary storage volumes – disks, drums, etc on which the database resides, together with the associated devices, control units, channels and so forth.

Software: Between the physical database itself (Data actually stored) and the users of the system is a layer of software, usually called the database management system (DBMS). All request from users for access to the database is handled by the DBMS.

Users: Users are classified in three classes.

1. *Application programmer:* The user who is responsible for writing application programme that use the database using languages as Access, Foxpro, Cobol etc. These application programs operate on the data in all the usual ways: retrieving information, creating new information, deleting or changing existing information.
2. *End user:* The end-user is for accessing databse from a terminal. End user issue different command from terminal through application program to perform all the functions of retrieval, creation, deletion and modification.
3. *Database administrator:* Database administrator is one who maintain the databse description in original form. Some of the responsibilities are
   - The creation of the original description of the database structure and the way that structure is reflected by the files of the physical database.
   - Granting access the database to different users.
   - Modification of the database description or its relationship to the physical organisation of the database.
   - Making backup copies of the database and repairing damage to the database due to hardware or software failures or misuse.

1.1. Purpose and data abstraction

Objectives of this topic:

Database Management Systems

1. A **database management system** (DBMS), or simply a **database system** (DBS), consists of
   a. A collection of interrelated and persistent data (usually referred to as the **database** (DB)).

   b. A set of application programs used to access, update and manage that data (which form the data management system (MS)).
2. The goal of a DBMS is to provide an environment that is both **convenient** and **efficient** to use in
   a. Retrieving information from the database.
   b. Storing information into the database.

1. DBMS are usually designed to manage **large** bodies of information. This involves

   ♦ Definition of structures for information storage (data modeling).

   ♦ Provision of mechanisms for the manipulation of information (file and systems structure, query processing).

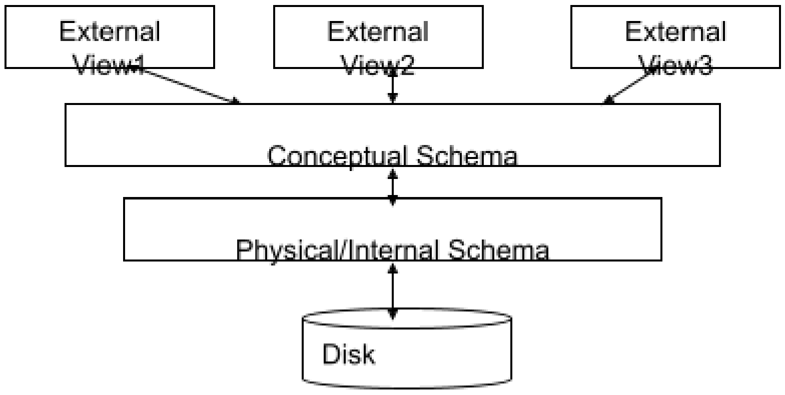
   ♦ Providing for the safety of information in the database (crash recovery and security).

   ♦ Concurrency control if the system is shared by users.

1.1.1. LEVELS OF ABSTRACTION

- Many views, single conceptual (logical) schema and physical schema.
- Views describe how users see the data.
- Conceptual schema defines logical structure
- Physical schema describes the files and indexes used.

Note: Schemas are defined using DDL; data is modified/queried using DML.



Example: University Database

Conceptual schema:
Students(sid, name, Level, age)

Courses(cid, cname, credits)

Enrolled(sid, cid, grade)

Physical schema:
Relations stored as unordered files.

Index on first column of Students.

External Schema (View):
Course_info (cid:string, no_of_student_enroll:integer)

1.2. DMS Model, instances and schemes

Objectives of this topic:

The objectives of this topics are to learn about

1. What is DMS model ?
2. What are the types of DMS Model?
3. What is instances and schemes in Database?

1.2.1. DATA MODELS:

Data models are a collection of conceptual tools for describing *data*, *relationships*, *data semantics* and *data constraints*.

Data model conceptual tools can be group in three and they are:

1. Object-based Logical Models.

a. The E-R Model

b. The Object-Oriented Model

c. Binary model

d. Semantic data model

e. Infological model

f. Functional data model

2. Record-based Logical Models.

g. The Relational Model

h. The Network Model

i. The Hierarchical Model

3. Physical Data Models.

j. Unifying model

k. Frame Memory

1. Object-based Logical Models

♦ Describe data at the conceptual and view levels.

- Provide fairly flexible structuring capabilities.
- Allow one to specify data constraints explicitly.
- Over 30 such models, including
    - Entity-relationship model.
    - Object-oriented model.
    - Binary model.
    - Semantic data model.
    - Infological model.
    - Functional data model.

1.1 The E-R Model

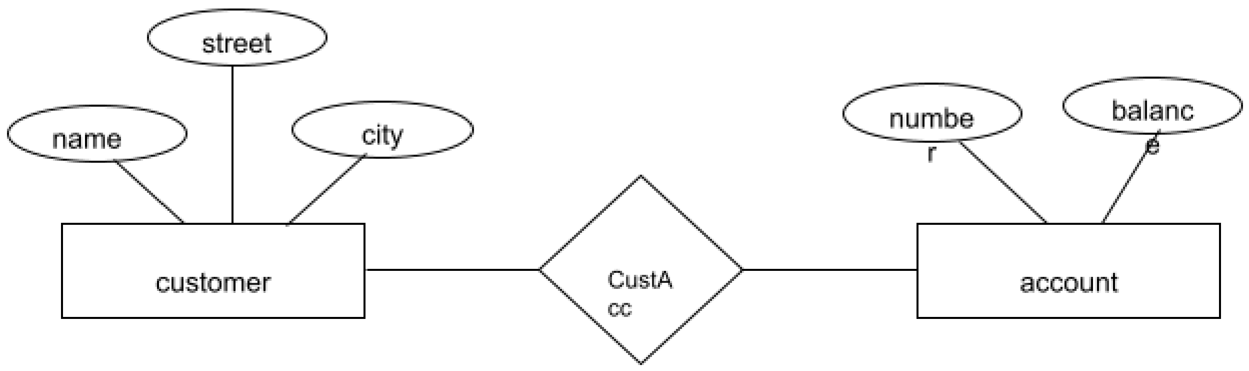
The entity-relationship model is based on a perception of the world as consisting of a collection of basic **objects** (entities) and **relationships** among these objects.

- An **entity** is a distinguishable object that exists.
- Each entity has associated with it a set of **attributes** describing it.
- E.g. *number* and *balance* for an account entity.
- A **relationship** is an association among several entities. e.g. A *cust_acct* relationship associates a customer with each account he or she has.
- The set of all entities or relationships of the same type is called the **entity set** or **relationship set**.
- Another essential element of the E-R diagram is the **mapping cardinalities**, which express the number of entities to which another entity can be associated via a relationship set.

We'll see later how well this model works to describe real world situations.

The overall logical structure of a database can be expressed graphically by an **E-R diagram**:

- **rectangles**: represent entity sets.
- **ellipses**: represent attributes.
- **diamonds**: represent relationships among entity sets.
- **lines**: link attributes to entity sets and entity sets to relationships.



A sample E-R diagram.

1.2 The Object-Oriented Model

The object-oriented model is based on a collection of objects, like the E-R model.

- An object contains values stored in **instance variables** within the object.
- Unlike the record-oriented models, these values are themselves objects.
- Thus objects contain objects to an arbitrarily deep level of nesting.
- An object also contains bodies of code that operate on the object.
- These bodies of code are called **methods**.
- Objects that contain the same types of values and the same methods are grouped into **classes**.
- A class may be viewed as a type definition for objects.
- Analogy: the programming language concept of an abstract data type.
- The only way in which one object can access the data of another object is by invoking the method of that other object.
- This is called **sending a message** to the object.
- Internal parts of the object, the instance variables and method code, are not visible externally.
- Result is two levels of data abstraction.

For example, consider an object representing a bank account.

- The object contains instance variables *number* and *balance*.
- The object contains a method *pay-interest* which adds interest to the balance.

♦ Under most data models, changing the interest rate entails changing code in application programs.

♦ In the object-oriented model, this only entails a change within the *pay-interest* method.

Unlike entities in the E-R model, each object has its own unique identity, independent of the values it contains:

♦ Two objects containing the same values are distinct.

♦ Distinction is created and maintained in physical level by assigning distinct object identifiers.

2. Record-based Logical Models

Record-based logical models:

♦ Also describe data at the conceptual and view levels.

♦ Unlike object-oriented models, are used to

- Specify overall logical structure of the database, **and**
- Provide a higher-level description of the implementation.

♦ Named so because the database is structured in fixed-format records of several types.

♦ Each record type defines a fixed number of fields, or attributes.

♦ Each field is usually of a fixed length (this simplifies the implementation).

♦ Record-based models do not include a mechanism for direct representation of code in the database.

♦ Separate languages associated with the model are used to express database queries and updates.

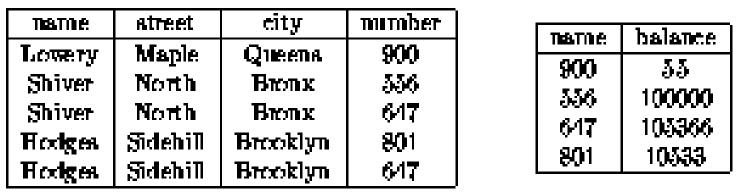
♦ The three most widely-accepted models are the **relational, network**, and **hierarchical**.

♦ This course will concentrate on the **relational** model.

♦ The **network** and **hierarchical** models will not covered in details.

2.1 The Relational Model

- Data and relationships are represented by a collection of **tables**.
- Each **table** has a number of columns with unique names, e.g. *customer, account*.
- Figure a sample relational database.



name	street	city	number
Lowery	Maple	Queens	900
Shiver	North	Bronx	556
Shiver	North	Bronx	647
Hodges	Sidehill	Brooklyn	801
Hodges	Sidehill	Brooklyn	647

name	balance
900	55
556	100000
647	105366
801	10533

Figure: A sample relational database.

2.2 The Network Model

- Data are represented by collections of records.
- Relationships among data are represented by links.
- Organization is that of an **arbitrary graph**.
- Figure shows a sample network database that is the equivalent of the relational database.

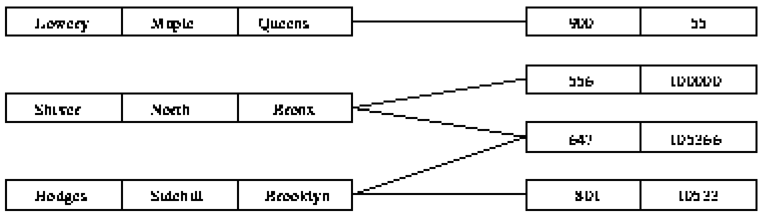


Figure : A sample network database

2.3 The Hierarchical Model:

- Similar to the network model.
- Organization of the records is as a collection of **trees**, rather than arbitrary graphs.

♦ Figure shows a sample hierarchical database that is the equivalent of the relational database.

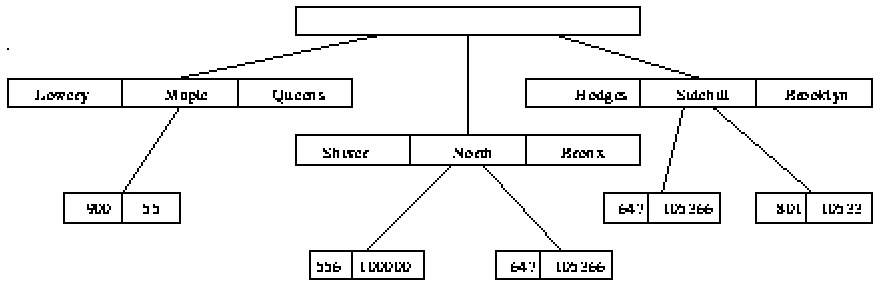


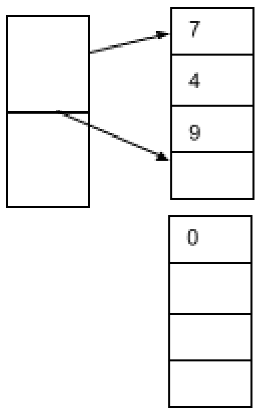
Figure : A sample hierarchical database

The relational model does not use pointers or links, but relates records by the values they contain. This allows a formal mathematical foundation to be defined.

3. Physical Data Models

1. Are used to describe data at the lowest level.
2. Very few models, e.g.
   - Unifying model.
   - Frame memory.

1.2.2. SCHEMES AND INSTANCES:

	Physical	**Conceptual**	**View**
Scheme	ZAP= Directory block plus pointers to blocks holding files.	ZAP= Files of integer	
Instances		ZAP= 7,4,9,0,….	

When the database is designed, we are interested in plans for the database. But when it is used, we are concerned with the actual data present in the database. The data in a database changes frequently, while the plans remain the same over long period of time.

The current contents of a database, we term an instance of the database. The scheme refers to plan and based on different levels of database, scheme (or plan) is different for different level which we called Conceptual scheme, Physical scheme and view scheme.

Plans consist to describe the types of entities that the database deals with, the relationships among these types of entities, and the ways in which the entities and relationship at one level of abstraction are expressed at the next lower level.

The term scheme is used to refer to plans. So we call conceptual scheme as the plan for the conceptual database and physical scheme as the plan for the physical database. The plan for a view is called sub scheme.

Points to remember on Instances and Schemes:

1. Databases change over time.
2. The information in a database at a particular point in time is called an **instance** of the database.
3. The overall design of the database is called the database **scheme**.
4. Analogy with programming languages:
   - ♦ Data type definition - scheme
   - ♦ Value of a variable - instance
5. There are several schemes, corresponding to levels of abstraction:
   - ♦ Physical scheme
   - ♦ Conceptual scheme
   - ♦ Subscheme (can be many)

1.3. Data independence, DDL, DML

1.3.1. DATA INDEPENDENCE:

Applications insulated from how data is structured and stored. The ability to modify a scheme definition in one level without affecting a schema definition in the next higher level is called data independence.

From the three level of data abstraction, view to conceptual to physical database, provides two levels of data independence.

Physical data independence:

Protection from changes in *physical* structure of data. It defines the corresponding between the conceptual view and the physical view (stored database). The physical scheme can be changed by the database administrator, without altering the conceptual scheme. This independence is referred to as physical database independence. In other words, the effects of changes at physical database must be isolated from the conceptual level, in order that data independence might be preserved.

The advantage to physical data independence is that it allows tuning of the physical database for efficiency while permitting application programs to run as if no change had occurred.

Logical data independence:

Protection from changes in *logical* structure of data. It defines the correspondence between a particular external view and the conceptual view. The type of independence between external (view) and conceptual level database is called Logical Data independence. Some of changes at conceptual level are like data types, fields and record name and so on. Many modifications to the conceptual scheme can be made if we redefine the mapping from the view schema to the conceptual schema.

Points to remember on Data Independence:

1. The ability to modify a scheme definition in one level without affecting a scheme definition in a higher level is called **data independence**.
2. There are two kinds:
    - **Physical data independence**
        - The ability to modify the physical scheme without causing application programs to be rewritten
        - Modifications at this level are usually to improve performance
    - **Logical data independence**
        - The ability to modify the conceptual scheme without causing application programs to be rewritten
        - Usually done when logical structure of database is altered
3. Logical data independence is harder to achieve as the application programs are usually heavily dependent on the logical structure of the data. An analogy is made to abstract data types in programming languages.

1.3.2. DATA DEFINITION LANGUAGE (DDL)

1. Used to specify a database scheme as a set of definitions expressed in a DDL
2. DDL statements are compiled, resulting in a set of tables stored in a special file called a **data dictionary** or **data directory**.
3. The data directory contains **metadata** (data about data)
4. The storage structure and access methods used by the database system are specified by a set of definitions in a special type of DDL called a **data storage and definition** language

5. **basic idea:** hide implementation details of the database schemes from the users

1.3.3. DATA MANIPULATION LANGUAGE (DML)

1. **Data Manipulation** is for:

    - **retrieval** of information from the database
    - **insertion** of new information into the database
    - **deletion** of information in the database
    - **modification** of information in the database

2. A DML is a language which enables users to access and manipulate data.
3. The goal is to provide efficient human interaction with the system.
4. There are two types of DML:

    - **procedural**: the user specifies *what* data is needed and *how* to get it. E.g. SQL
    - **nonprocedural**: the user only specifies *what* data is needed (E.g. QBE)

        - Easier for user
        - May not generate code as efficient as that produced by procedural languages

A **query language** is a portion of a DML involving information retrieval only. The terms DML and query language are often used synonymously.

1.4. DATABASE MANAGER, ADMINISTRATOR, USERS

1.4.1. DATABASE MANAGER

1. The **database manager** is a **program module** which provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
2. Databases typically require lots of storage space (gigabytes). This must be stored on disks. Data is moved between disk and main memory (MM) as needed.

3. The goal of the database system is to **simplify** and **facilitate** access to data. Performance is important. Views provide simplification.
4. **So** the database manager module is responsible for

- ♦ **Interaction with the file manager:** Storing raw data on disk using the file system usually provided by a conventional operating system. The database manager must translate DML statements into low-level file system commands (for storing, retrieving and updating data in the database).
- ♦ **Integrity enforcement:** Checking that updates in the database do not violate consistency constraints (e.g. no bank account balance below $25)
- ♦ **Security enforcement:** Ensuring that users only have access to information they are permitted to see
- ♦ **Backup and recovery:** Detecting failures due to power failure, disk crash, software errors, etc., and restoring the database to its state before the failure
- ♦ **Concurrency control:** Preserving data consistency when there are concurrent users.

Some small database systems may miss some of these features, resulting in simpler database managers. (For example, no concurrency is required on a PC running MS-DOS.) These features are necessary on larger systems.

1.4.2. DATABASE ADMINISTRATOR:

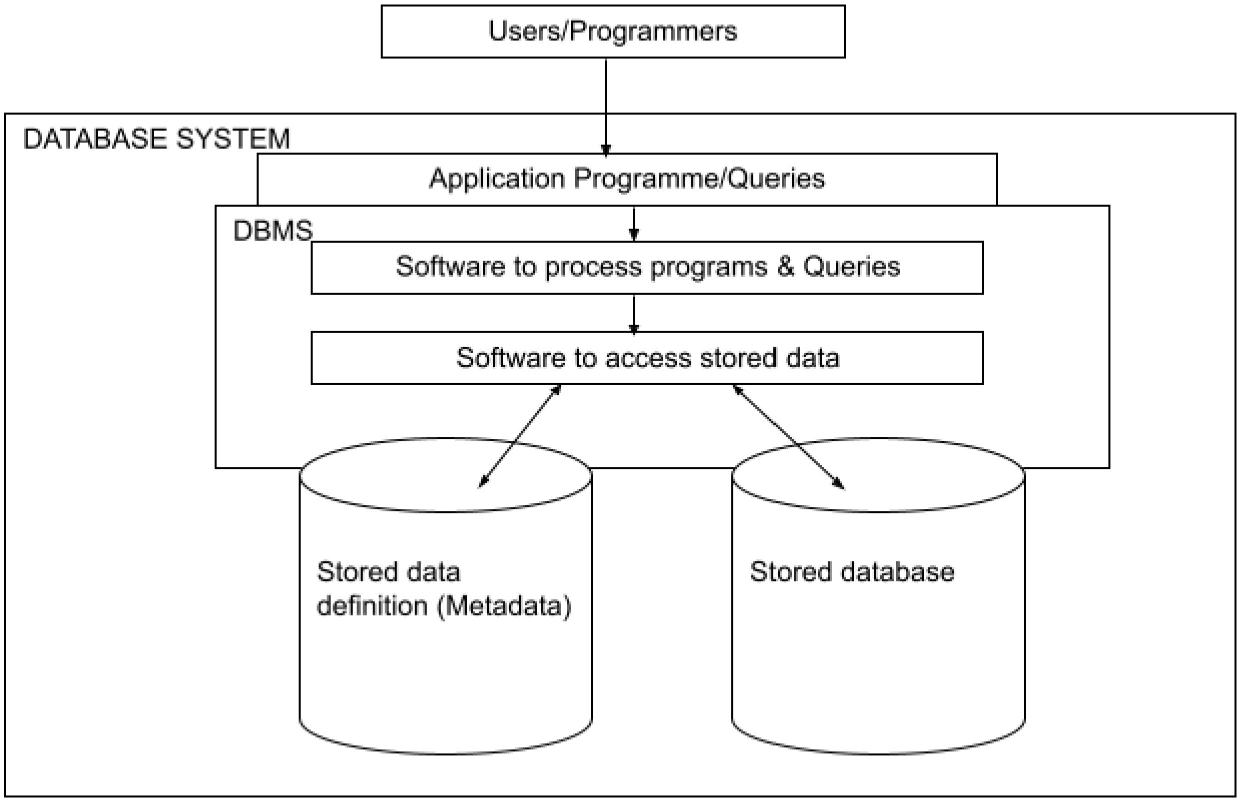
1. The **database administrator** is a **person** having central control over data and programs accessing that data. Duties of the database administrator include:

- ♦ **Scheme definition:** the creation of the original database scheme. This involves writing a set of definitions in a DDL (data storage and definition language), compiled by the DDL compiler into a set of tables stored in the data dictionary.
- ♦ **Storage structure and access method definition:** writing a set of definitions translated by the data storage and definition language compiler
- ♦ **Scheme and physical organization modification:** writing a set of definitions used by the DDL compiler to generate modifications to appropriate internal system tables (e.g. data dictionary). This is done rarely, but sometimes the database scheme or physical organization must be modified.
- ♦ **Granting of authorization for data access:** granting different types of authorization for data access to various users
- ♦ **Integrity constraint specification:** generating integrity constraints. These are consulted by the database manager module whenever updates occur.

1.5. OVERALL SYSTEM STRUCTURE

1.5.1. STRUCTURE OF A DBMS

A typical DBMS has a layered architecture. The figure does not show the concurrency control and recovery components. This is one of several possible architectures; each system has its own variations.


Users/Programmers
DATABASE SYSTEM
Application Programme/Queries
DBMS
Software to process programs & Queries
Software to access stored data
Stored data definition (Metadata)
Stored database

1.5.2. SUMMARY

DBMS used to maintain, query large datasets. Benefits include recovery from system crashes, concurrent access, quick application development, data integrity and security. Levels of abstraction give data independence. A DBMS typically has a layered architecture. DBAs hold responsible jobs and are well-paid! DBMS R&D is one of the broadest, most exciting areas in CS.