

CH-1 DATABASE ARCHITECTURE

INTRODUCTION

Data: It is a collection of raw facts that can be recorded

Database: A database is a collection of related data.

Database has the following properties

1. It represents some aspects of real world.
2. It is a logically coherent collection of data with some inherent meaning
3. It is designed, built and populated with data for a specific purpose
4. It has an intended group of users and some preconceived applications in which these users are interested

E.g. An address book containing names, addresses and phone numbers of people in a software like MS-Excel or MS-Access

DBMS- Database Management System: It is a collection of programs that enables users to create and maintain databases. It is a general purpose software system that facilitates the processes of defining, constructing, manipulating and sharing the database amongst various users and applications.

Functions of DBMS:

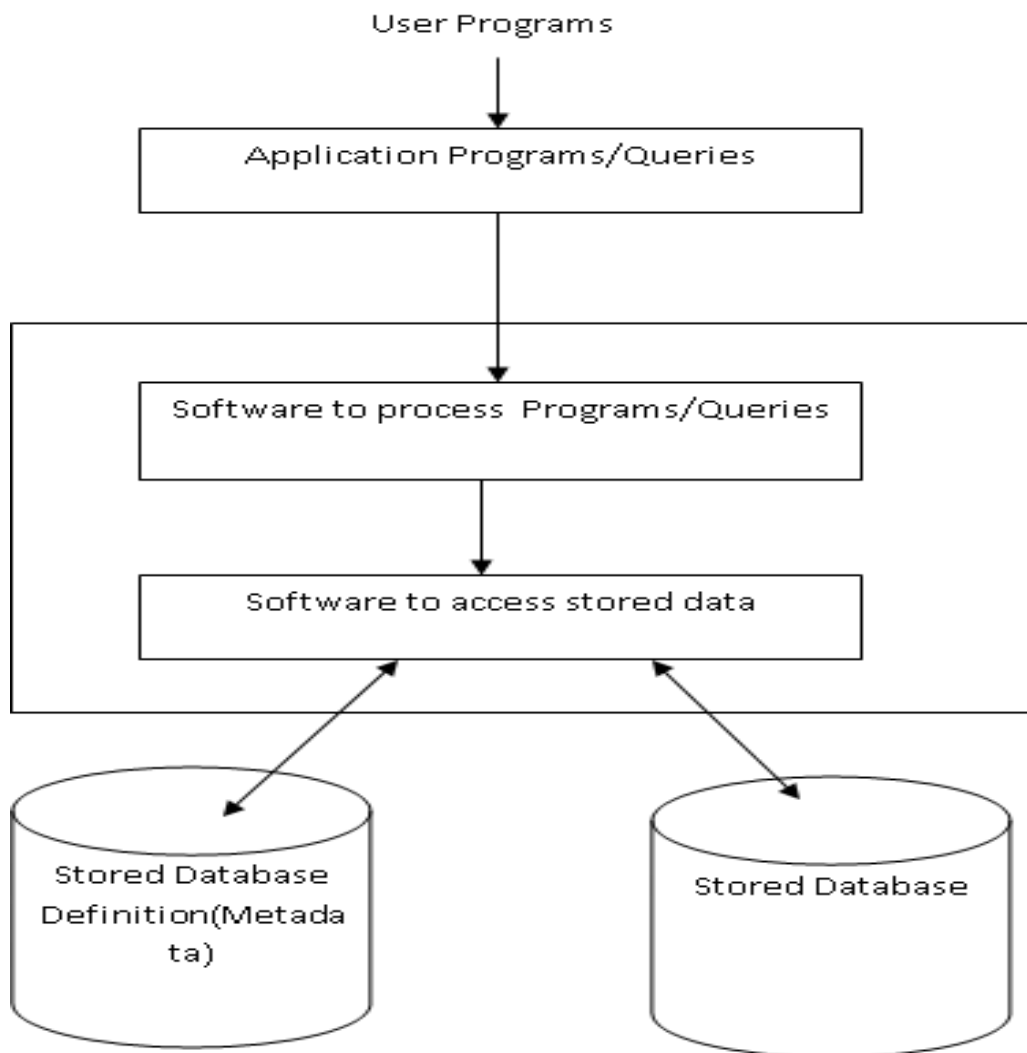
The functions performed by DBMS are as follows

- **Defining** a database involves specifying the data types, structures and constraints for the data to be stored in the database
- **Constructing** the database is the process of storing the data itself on some storage medium that is controlled by the DBMS
- **Manipulating** the database includes functions like querying the database to retrieve specific data, updating the database so as to reflect changes and generating reports from the data
- **Sharing** the database allows multiple users and programs to access the database concurrently

- DBMS protects the database. Protection includes both system protection against hardware or software malfunction and security malfunction against unauthorized access
- A large database may have a life cycle for many years. Hence the DBMS must be able to maintain the database system by allowing the system to evolve as requirements change over time.

Example:

Consider a COLLEGE database maintaining information concerning students, courses and their grades in a particular college. The database is organized into three files namely STUDENT, COURSE and GRADE REPORT.



simplified database system environment.

CHARACTERISTICS OF DATABASE APPROACH

In file processing, each user defines and implements the files needed for a specific software application as a part of programming the application.

E.g. Consider a student file. A person trying to grade the students uses the student file and calculates grade. A clerk who wants to print students' attendance will create another file with the same data.

Hence, when files are used there is redundancy in storing the same data and redundant effort to maintain the common data. In database approach, a single repository of data is maintained that is defined and then accessed by various users.

The characteristics of database approach vs. file processing are as follows

1. Self-describing nature of database system
2. Insulation between data and programs and data abstraction
3. Support of multiple views of data
4. Sharing of data and multi-user transaction processing

1. Self-describing nature of database system:

A fundamental characteristic of database approach is that the database system contains not only the database but also a complete definition or description of the database structure and constraints. The definition is stored in a DBMS catalog. The catalog contains information such as structure of each file, the type and storage format for each data item and various constraints on data. The information stored in the catalog is called the **meta-data** and it describes the structure of primary database. This is used only by the DBMS users and DBMS software who need information about the database structure

2. Insulation between data and programs and data abstraction:

The structure of data files is stored in DBMS catalog(a complete list of items) separately from access programs. This property is called **program-data independence**. The structure of the data file can be modified to reflect any changes in the data description.

Program-operation independence is a term where users define operations on data as a part of database operation. An operation is specified in 2 parts- the interface of an operation includes

the name of the operation and data types of the parameters. The implementation of the operations is specified separately without affecting the interface. The characteristic that allows both program-data independence and program-operation independence is called **data abstraction**.

DBMS provides user with conceptual representation of data that does not include how data is stored or how methods are implemented. In database approach, the detailed structure and organization of the file are stored in a catalog. Database users and applications only refer to the conceptual (**representation** of a system) representation.

For example, a file access program may be written in such a way that it can access only STUDENT records of the structure. If we want to add another piece of data to each STUDENT record, say the Birth_date, such a program will no longer work and must be changed. By contrast, in a DBMS environment, we only need to change the description of STUDENT records in the catalog.

3. Support of multiple views of data:

A database typically has many users, each of whom may require a different view or perspective of the database. A view may be a subset of the database or may contain virtual data that is derived from database files. A multi-user DBMS whose users have a variety of distinct applications must provide facility for defining multiple views.

4. Sharing of data and multi-user transaction processing

A multi user DBMS must allow multiple users to access the database at the same time. DBMS must include concurrency control software to ensure that several users trying to update the same data in a controlled manner. DBMS must support multiple users to make concurrent transactions.

For example, when several reservation agents try to assign a seat on an airline flight, the DBMS should ensure that each seat can be accessed by only one agent at a time for assignment to a passenger. These types of applications are generally called **online transaction processing** (OLTP) applications.

A transaction is an executing program or a process that includes one or more database accesses, such as reading or updating a database records. Each transaction is supposed to execute a logically correct database. The two most important properties of transactions are

1. **Isolation(separation):** A transaction appears to execute in isolation from other transaction even though hundreds of transactions execute concurrently.
2. **Atomicity:** This ensures that either all the database operations in a transaction are executed or none

ACTORS ON THE SCENE

1. Database Administrators

DBA stands for Data Base Administrator. In a database environment, the primary resource is the database itself and the secondary resource is the database itself. Administering these resources is the responsibility of the DBA. The DBA is responsible for authorizing access to the database, coordinating and monitoring its use and acquiring the software and hardware resources as needed. DBA is also accountable for system security

2. DB Designers: These are people who are responsible for identifying data to be stored in the database and for choosing appropriate structures to represent and store data. It is the responsibility of database designers to communicate with all prospective database users in order to understand their requirements and to create a design that meets these requirement

3. End Users: End users are people whose jobs require access to the database for querying, updating and generating reports. These people make use of the existing database

There are several categories of end users:

- i. **Casual end users** occasionally access the database, but they may need different information each time. They use a sophisticated database query interface to specify their requests and are typically middle- or high-level managers or other occasional browsers.
- ii. **Naive or parametric end users** make up a sizable portion of database end users. Their main job function revolves around constantly querying and updating the database, using standard types of queries and updates—called canned transactions—that have been carefully programmed and tested. Many of these tasks are now available as mobile apps for use with mobile devices. The tasks that such users perform are varied. Examples are: Bank customers , Reservation agents or customers for airlines,

iii. Sophisticated end users include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS in order to implement their own applications to meet their complex requirements.

iv. Standalone users maintain personal databases by using ready-made program packages that provide easy-to-use menu-based or graphics-based interfaces. An example is the user of a financial software package that stores a variety of personal financial data.

4. System Analysts and Application Programmers (Software Engineers)

- System analyst: People who determine the requirements of the end users
- Application programmers: These people implement the above specification as programs, then test and debug and maintain the software for which the database was designed

WORKERS BEHIND THE SCENE

1. DBMS system designers and implementers:

Design and implement the DBMS modules and interfaces as a software package. A DBMS is a very complex software system that consists of many components, or modules, including modules for implementing the catalog, query language processing, interface processing, accessing and buffering data, controlling concurrency, and handling data recovery and security. The DBMS must interface with other system software, such as the operating system and compilers for various programming languages.

2. Tool developers design and implement tools:

The software packages that facilitate database modeling and design, database system design, and improved performance. Tools are optional packages that are often purchased separately. They include packages for database design, performance monitoring, natural language or graphical interfaces, prototyping, simulation, and test data generation. In many cases, independent software vendors develop and market these tools.

3. Operators and maintenance personnel (system administration personnel)

These are responsible for the actual running and maintenance of the hardware and software environment for the database system.

ADVANTAGES OF DBMS:

The advantages of DBMS approach are as follows

1. Controlled Redundancy: In traditional file processing each user group maintain their own files and handle data processing application. This leads to a stage where the same data is stored in multiple files. The redundancy of storing data in more than one file creates a lot of problems

- Entering a new record should be done on multiple files leads to duplication of effort
- Storage space is wasted since the same details appear in all the files
- Data inconsistency- if a record is modified in one file this updation does not reflect in other files

In order to overcome these limitations, the database approach integrates all the views of the different user group during database design. All the required details are stored in the database in one place. This ensures consistency and saves storage space. But, in practice, only some data items may repeat themselves at the required places. This phenomenon is called controlled redundancy.

2. Restricting unauthorized access: When multiple users share a large database, it is likely that all users will not want all the information in the database. Moreover, all the users will not be authorized to use all the data. Some users may be only allowed to retrieve while others may be allowed to retrieve as well as to update data. In order to provide authorization, users are given user names and passwords. A DBMS provides security and authorization subsystem, which the DBA uses to create accounts and passwords to different users

3. Providing Storage structures for efficient query processing: Database systems must provide capabilities for efficiently executing a query. DBMS must provide specialized data structures to speed up disk access. The query processing and optimization module of the DBMS is responsible for choosing efficient query execution plan for each query based on the existing storage structure

4. Providing Backup and recovery: DBMS must provide facilities for recovering from hardware or software failures. The backup and recovery systems are responsible for recovery of data in such situations

5. Providing multiple user interfaces: DBMS has a wide variety of users. Hence, it should provide multiple user interfaces for each class of user.

For example: Query language for casual users, programming language interface for application programmers.

6. Represent complex relationships among data: A database may contain a variety of data that are interrelated. DBMS must provide a capability to represent complex relationship among data as well as o retrieve and update related data only.

7. Enforcing integrity constraints: DBMS should provide capability for defining and enforcing integrity constraints. For example specifying data types for each data item is done to ensure that erroneous data is not entered into the data base

8. Permitting inferencing and actions using rules: Some database provide capabilities for defining deduction rules for inferencing new information from the stored database facts. Such systems are called deductive database systems. Some database provide active rules that can automatically initiate actions when certain events and conditions occur

Additional Implications of using database approach:

Some of the added benefits if database approach are

- **Potential for enforcing standards-** Database approach permits DBA to define and enforce standards among database users in large organizations. These rules include format of data elements, display formats and report structures.
- **Reduced Application Development time-** Developing an application using DBMS requires very little time.
- **Availability of up-to-date information-** DBMS makes the database available to all the users. Once any change is made to the database, these changes can be viewed by all the users.
- **Flexibility-** DBMS facilitates changes in structure of database as the requirements change.
- **Economies of scale:** DBMS reduces the overall cost of operation and management.

DATA MODELS

A data model is a collection of concepts that can be used to describe the structure of database and provides necessary means to achieve this abstraction. The structure of database means the data types, relationships, and constraints that should hold good for the data. Data

models include a set of basic operations that are used for specifying updates and retrieval on the data base.

Categories of data models:

1. **High-Level or conceptual data models**- provide concepts (Such as entities, attributes, and relationships) that are close to the way how many users perceive data
2. **Low-Level or physical data models** – provide concepts that describe details of how the data is stored in computers.

Between these two extremes is a class of **representational data models**, which provide concepts that may be understood by the end users and also specifies details about data storage. Representational data models represent data using their record structures & can be categorized as

- a. Relational model
- b. Hierarchical model
- c. Network model

a. Relational Data Model: In this model, there are no physical links between records. All the data is maintained in the form of tables, comprising of rows and columns. Data in two tables is related through common columns. Querying is much easier in this model. It is very much programmer friendly and is the most widely used model.

b. Hierarchical Data Model: Here different records are interrelated through a hierarchical or a tree-like structure. A parent record can have several children but a child can have only one parent.

c. Network Data Model: Here, a parent record can have several children and a child record can also have many parent records.

Conceptual data models use concepts such as entities, attributes, and relationships

An **entity** represents a real-world object.

An **attribute** represents the property that further describes an entity

A **relationship** among two or more entities represents an association among the entities

Eg: Entity-Relationship model

Schemas and Instances

Schema: The description of database is called a database schema which is specified during database design and is not expected to change.

A diagram used to display schema is called a **schema diagram**.

Each object in the schema is called a **schema construct**.

The data in the database at a particular instant of time is called a **database state** or a **snapshot**. It is also called the current set of occurrences or **instances** in the database.

STUDENT

Roll No	Student Name	Class	Department
---------	--------------	-------	------------

COURSE

Course Name	Course Number	Department
-------------	---------------	------------

GRADE REPORT

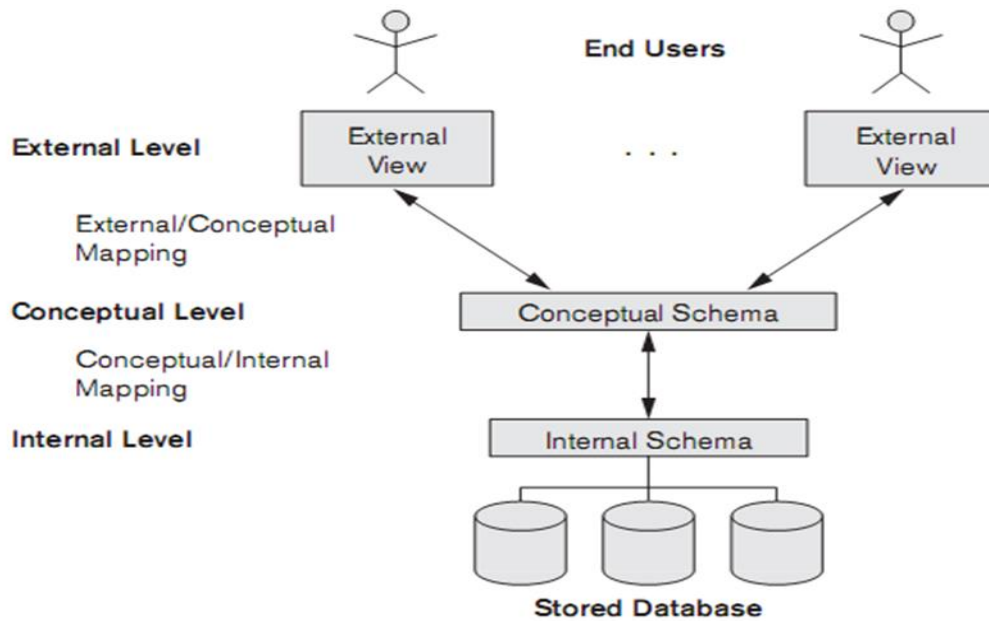
Roll No	Class	Grade
---------	-------	-------

Meta data: The description of the schema constructs and constraints is called meta data. The meta data is stored in the DBMS catalog so that DBMS software can refer to it whenever necessary.

Three Schema Architecture

The goal of three schema architecture is to separate user applications and the physical database. The schemas are 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 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. It hides details of physical storage and concentrates on



describing entities, data types, relationships, operations and constraints. The representational model is used to describe the conceptual schema.

3. The external or view level: includes a number of external schemas or user views. Each external schema describes a part of the database that a particular user group is interested and hides the rest of the database from that user group.

The process of transforming requests and results between levels is called **mappings**.

Data Independence: Data independence can be defined as the capacity to change the schema at one level of the database system without having to change the schema at the next higher level.

There are two types of data independence

1. Logical data independence
2. Physical data independence.

1. Logical Data Independence: It is the capacity to change the conceptual schema without having to change external schemas or application programs. We may need to change the constraints or reduce the database.

Eg: In Student database, we can add a column 'grades' to a existing schema without changing the external view which displays only roll_no & names

2. Physical Data Independence: It is the capacity to change the internal schema without having to change conceptual schemas.

Eg: If size of physical storage of a particular data item is increased in internal level it does not effect in conceptual schema/external schema

DBMS Languages

The languages used to specify the description of the data to manipulate the data & retrieve the data are called database languages

Classification of DBMS languages

- 1) Data Definition Language (DDL)
 - i. Storage Definition Language (SDL)
 - ii. View Definition Language (VDL)
- 2) Data Manipulation Language(DML)
 - i. High level or non-procedural DML
 - ii. Low level or procedural DML

1) Data Definition Language (DDL)

The DBMS will have a DDL compiler whose function is to process the DDL statements in order to identify the descriptions of the schema constructs and to store the schema description in the DBMS catalog. DBA and designers uses DDL to define both the schema

- i. **Storage Definition Language** is used to describe the internal schema to define storage structure of data(either sequential or indexed)
- ii. **View Definition Language** is used to specify user views and their mappings to the conceptual schema.

2) Data Manipulation Language(DML)

Is used to perform manipulations on the data in the database. Manipulations in the database include retrieval, insertion, deletions and modification on data. . There are two main types of DML. They are

- i. **High level or non-procedural DML:** It can be used on its own to specify complex database operations in a concise manner. They specify only what to do rather than how to do i.e. the procedural details are not mentioned here. E.g. SQL.

High level DML such as SQL can specify and retrieve many records in a single DML statement. Hence they are called **set-at-time** or **set oriented DML**. They are called declarative since high level DML specify which data to retrieve rather than how to retrieve it. High level DML used in stand-alone interactive manner is called a **query language**

- ii. **Low level or procedural DML:** They are embedded in a general purpose programming language. This type of DML retrieves individual records or objects from the database and processes them separately. They are also called record-at-time DML. When DML commands are embedded in a general purpose programming language the language is called **host language** and DML is called data sub language

DBMS Interfaces:

Interfaces in DBMS are user-friendly and consists of the following.

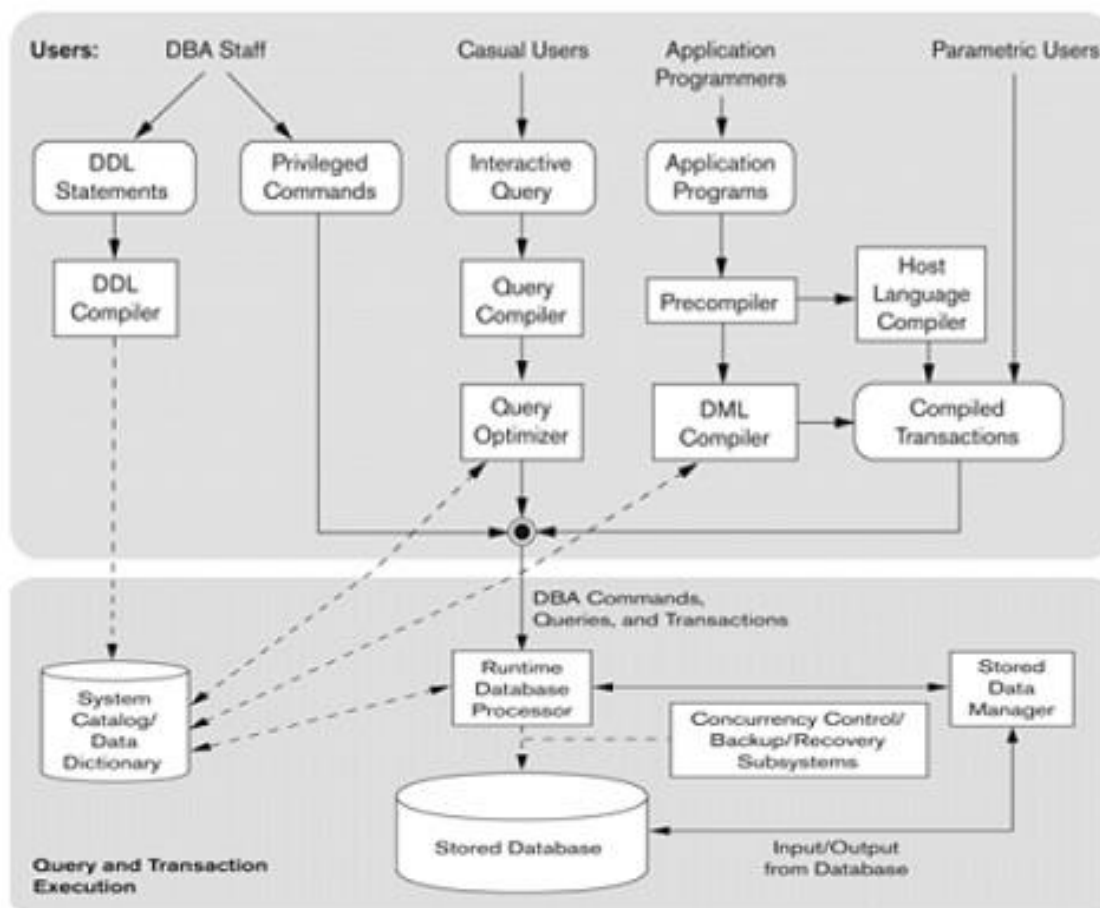
- a. **Menu based interfaces** for web clients or browsers: These help the users with a list of options called menus that lead the user through formulation of requests. The user need not remember any command or syntax of the query language. The query is composed step by step by picking options from the menu.
- b. **Form based Interfaces:** It displays a form to each user. Users can fill data in the form to insert new data. The user can also fill specific entries in which case the DBMS retrieves data from the database
- c. **Graphical user interfaces:** A GUI displays a schema in a diagrammatic form to the user. The user specifies a query by manipulating the diagram. A pointing device such as mouse is used.
- d. **Natural Language Interfaces:** These interfaces accept request written in English and attempts to understand them. It has its own schema and dictionary of important words. If the interpretation is successful, the interface generates a high level query corresponding to the natural language request and submits it to the DBMS. Otherwise a dialog is started with the user for clarification of request.

- e. **Interface for parametric users:** Parametric users such as bank tellers perform small set of operations on the database repeatedly. A small set of abbreviated commands are included to minimize the number of keystrokes for each request.
- f. **Interface for DBA:** Privileged commands can be used only by the DBA. These include commands for creating user accounts, granting account authorization and reorganizing storage structure of data.

Database System Environment

DBMS Component Modules:

The database and DBMS catalog is stored on disk. Access to disk is controlled primarily by the operating system which schedules disk I/O. The **stored data manager** module controls



access to the DBMS information stored on the disk whether it is a part of catalog or database. Some DBMS use a **buffer manager module** that transfers the data from the disk to the main

memory buffer so that it can be processed by other DBMS modules as well as application programs. The **DDL compiler** process the schema definition specified in DDL and stores the description of the schema in the catalog. The **run-time database processor** handles database access at run time. It receives retrieval and updates operation and carries them on the database. A **query compiler** handles high level queries that are entered interactively. The **pre-compiler** extracts the DML commands from an application program and sends these commands to DML compiler for compilation into object code. The DBMS runs on a computer which can be accessed by the end users. This is known as **client program/ computer**. The database resides on a computer called the **database server**.

Database System Utilities

DBMSs have database utilities that help the DBA manage the database system. Common utilities have the following types of functions:

- i. **Loading:** A loading utility is used to load existing data files—such as text files or sequential file into the database.
- ii. **Backup:** A backup utility creates a backup copy of the database, usually by dumping the entire database onto tape or other mass storage medium.
- iii. **Database storage reorganization:** This utility can be used to reorganize a set of database files into different file organizations, and create new access paths to improve performance.
- iv. **Performance monitoring:** Such a utility monitors database usage and provides statistics to the DBA. The DBA uses the statistics in making decisions such as whether or not to reorganize files or whether to add or drop indexes to improve performance.

Classification of DBMS

There are several criteria on which DBMS can be classified. They are

1. Based on data models
2. Based on number of users
3. Based on number of sites
4. Based on types of application.

1. Data Models: Based on data models, DBMS can be classified as

- a. **Relational Data Model**- data is organized as tables. where each table can be stored as a separate file & uses SQL
- b. **Object Data Model**-defines in terms of objects and their properties
- c. **Hierarchical Data model**- data is stored and related through tree like structures.
- d. **Network Data Model**- data records have a 1:N relation between them
- e. **Object – relational Data Model**- Relational data model extending their models to incorporate object database concepts and other capabilities.
- f. **XML Model**: The XML model has emerged as a standard for exchanging data over the Web, and has been used as a basis for implementing several prototype native XML systems.XML uses hierarchical tree structures

2. Number of users: Depending on the number of users supported by the system, DBMS can be categorized into

- a. Single user systems- that support only one user at a time
- b. Multi-user system – that support multiple users concurrently

3. Number of sites: Based on the number of sites over which the database is distributed , DBMS can be classified as

- a. **Centralized DBMS**- Here the data is stored at a single computer site. It supports multiple users but the DBMS and database reside at a single computer site.
- b. **Distributed DBMS**- The database and DBMS are distributed over many sites connected through network. This can again be divided as
 - i. **Homogeneous DDBMS**: They use same DBMS software at multiple sites
 - ii. **Heterogeneous DDBMS**: They use different DBMS software at different sites

4 .Types of application: Based on the types of application DBMS can be classified as follows

- a. General Purpose DBMS: DBMS that can be used for all types of application
- b. Special Purpose DBMS: DBMS used for a specific application for which they are designed. E.g. airline reservation – it cannot be used for other applications.

CH-2 :E-R Model

E-R Model Concepts

Entity: The basic object of ER model is an entity

An entity may be an object with physical existence, like person, employee, student or it could be an object with conceptual existence like company, job, and course. Each entity has attributes (particular properties that describe it).

A particular entity will have a value for each of its attributes. The attribute values that describe the entity become major part of the data stored in database. E.g. Employee is an entity.

Types of Attributes

The different types of attributes that occur in ER model are:

- a. Simple Vs. composite attributes
- b. Single valued Vs. Multi valued attributes
- c. Stored Vs. Derived Attributes

a. Simple Vs. Composite attributes

Attributes that are not divisible are called simple attributes or atomic attributes.

A composite attribute can be divided into smaller subparts which represent more basic attributes with independent meanings.

b. Single valued Vs. Multi valued attributes:

Attributes that can have a single value for a particular entity are called single valued attributes.

E.g. Age of an employee can have only single value.

An attribute that can have a set of values for the same entity is called multi valued attribute.

E.g. Qualification of an employee can have multiple values.

d. Null Values:

An entity may not have an applicable value for an attribute or some times the value may be unknown. In such cases, a special value called null is assigned to the attributes.

e. Complex Attributes: We can represent arbitrary nesting by grouping components of a composite attribute between parentheses () and separating the components with commas, and by displaying multivalued attributes between braces { }. Such attributes are called complex attributes

Entity Types

An entity type defines a collection (or set) of entities that have the same attributes

Each entity type in the database is described by its name and attributes. An entity type describes the schema or intension for a set of entities that share the same structure. The collection of entities of a particular entity type is grouped into an entity set, which is also called the **extension** of the entity type.

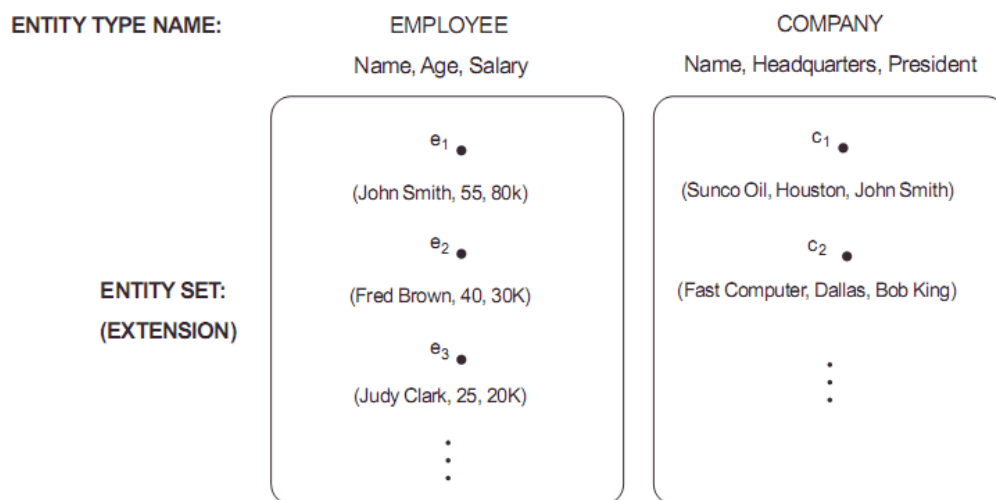
An entity type is represented in ER diagrams as a rectangular box enclosing the entity type name.

Attribute names are enclosed in ovals and are attached to their entity type by straight lines.

Composite attributes are attached to component attributes by straight lines. Multivalued attributes are displayed in double ovals.

Entity Sets

The collection of all entities of a particular entity type in a database at a point of time is called entity set. This is also called the extension of the entity type.



Key Attributes of entity types

An entity type has an attribute whose values are distinct for each individual entity in the entity set. Such an attribute is called a key attribute and its value can be used to identify each entity uniquely. An entity with no key attribute is called a **weak entity**.

Value sets(Domains) of attributes: Each simple attribute is associated with a value set or domain of values. It specifies the set of values that may be assigned to that attribute for each individual entity.

E.g. The value set for attribute age of employee to be the set of integers between 16 and 70.

Relationships types and relationship roles and Structural constraints

A relationship type R among n entity types E_1, E_2, \dots, E_n defines a set of associations or relationship set among entities from these entity types. For example, a relation Employee is associated with Department. The relationship used here is “works for”. In ER diagrams, relationship types are displayed in diamond boxes, which are connected using straight lines to rectangles representing entity types. The name of the relationship type is written inside the diamond.

Degree of a relationship

The degree of relationship is the number of participating entity types. The works-for relationship is of degree 2 because the two entity types participating in the relation are emp and dept. A relationship of degree two is called a binary relation and relation of degree three is called ternary relation. Binary relations are the most common relation.

Relationships as Attributes

Consider the works for relation. Department of employee or employees of department represent works for relation. Hence relation can be represented as attributes.

Role Names

Each entity type that participates in a relation type plays a particular role in the relation. The role name signifies the role that the participating entity from the entity type plays in each relation instance and helps to explain helps to explain what the relation means.

Example: In ‘works for’ relation EMP plays employee or worker role and DEPT plays the employer role.

Recursive Relationships:

Sometimes the same entity type participates more than once in a relation type in different roles. Such a relation type is called a recursive relation.

Example: The supervise relation relates an employee to a supervisor ; where emp and supervisor are members of same EMP entity type. Here EMP entity type participates twice in ‘supervision’ – once in the role of a boss and once in the role of a subordinate

Constraints on relationship types:

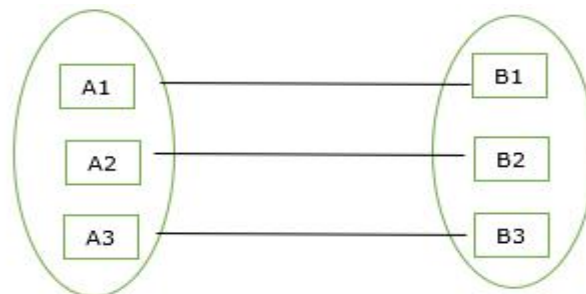
The two types of relation constraints are

- Cardinality Ratio
- Participation Constraints

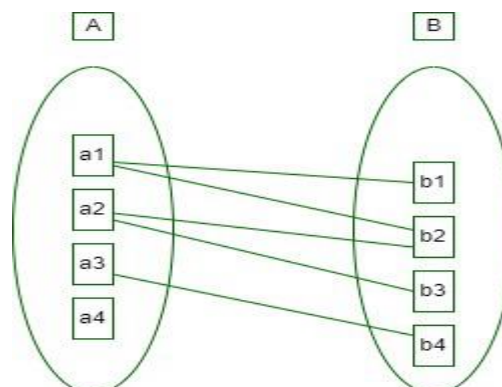
Cardinality Ratio for a Binary Relation:

The cardinality ratio for a binary relation specifies the maximum number of instances that an entity participates in. The possible cardinality ratios for a binary relationship type are 1:1, 1:N, N:1, M:N

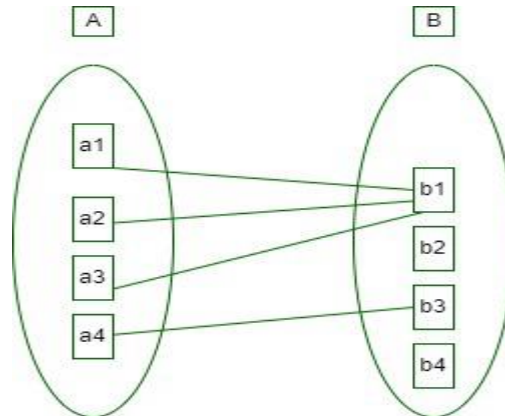
- 1) **One-to-one relationship (1:1):** One entity of A is associated with one entity of B.



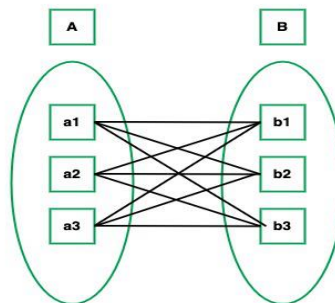
- 2) **One-to-many relationship (1:N):** An entity set A is associated with any number of entities in B with a possibility of zero and entity in B is associated with at most one entity in A.



- 3) **Many-to-one relationship (M:N):** An entity set A is associated with at most one entity in B and an entity set in B can be associated with any number of entities in A with a possibility of zero.



4) **Many-to-many relationship (M:N)** Many entities of A are associated with many entities of B.



Participation Constraints

The participation constraint specifies whether the existence of an entity depends on its being related to another entity via a relation type. It specifies the minimum number of relation instance that each entity can participate in and is also called the minimum cardinality relation. The two types of participation constraints are

1. Total participation.
2. Partial participation

1. **Total participation:** Example: If a company policy states that every employee must work for a department. An employee entity can exist only if it participates in at least one 'works for' relation instance. The participation of EMPLOYEE in 'works for' is called total participation. i.e.

every entity in a set of employee must be related to department entity via works for. Total participation is also called existency dependency.

Partial participation: For example: Not every employee can manage a department. Hence participation of employee in 'manages' relation is partial i.e. a part of the set of employee entities are related to some department entity via 'manages' relation.

Cardinality ratio and participation constraint together are known as **structural constraints**.

Attributes of Relationship Types

Relationship types can also have attributes, similar to those of entity types.

For example, to record the number of hours per week that a particular employee works on a particular project, we can include an attribute Hours for the WORKS_ON relationship type.

Another example is to include the date on which a manager started managing a department via an attribute Start_date for the MANAGES relationship type.

Weak Entity Types

Entity types that do not have any key attributes of their own are known as **weak entity** types. Entity types that have key attributes are called **strong entity** types. Entities belonging to a weak entity are identified by being related to specific entities from another entity type in combination with one of their attribute values. We call this other entity type as **identifying or owner entity type**. The relation type that relates the weak entity type to its owner is called the identifying relation of the weak entity.

Example: Consider the entity type Dependent related to Employee, used to keep track of the dependents of each employee. Dependent – Name, DOB, Sex, Relation.

Two dependents of 2 distinct employees may by chance be the same value. They can be identified as distinct only after determining the particular employee entity to which each dependent is related. In this example, EMP is parent entity type or dominant entity type. Weak entity is the child/subordinate entity type. A weak entity has a partial key which is a set of attributes that can uniquely identify weak entities that are related to the same owner entity. In an ER diagram, weak entity and its attributes are surrounded by double lines. Partial key attributes is underlined with dashed lines.

Refining the ER Design for the COMPANY Database.

- i. MANAGES, which is 1:1(one-to-one) relationship type between EMPLOYEE and DEPARTMENT. EMPLOYEE participation is partial. DEPARTMENT participation is total participation.
- ii. WORKS_FOR, 1:N (one-to-many) relationship type between DEPARTMENT and EMPLOYEE. Both participations are total.
- iii. CONTROLS, 1:N relationship type between DEPARTMENT and PROJECT.
The participation of PROJECT is total, whereas that of DEPARTMENT is determined to be partial, after consultation with the users indicates that some departments may control no projects.
- iv. SUPERVISION, 1:N relationship type between EMPLOYEE (in the supervisor role) and EMPLOYEE (in the supervisee role). Both participations are determined to be partial, after the users indicate that not every employee is a supervisor and not every employee has a supervisor.
- v. WORKS_ON, determined to be an M:N (many-to-many) relationship type with attribute Hours, after the users indicate that a project can have several employees working on it. Both participations are determined to be total.
- vi. DEPENDENTS_OF, 1:N relationship type between EMPLOYEE and
- vii. DEPENDENT, which is also the identifying relationship for the weak entity type DEPENDENT. The participation of EMPLOYEE is partial, whereas that of DEPENDENT is total.

ER Diagrams, Naming Conventions, and Design Issues

In ER diagrams the emphasis is on representing the schemas rather than the instances.

This is more useful in database design because a database schema changes rarely, whereas the contents of the entity sets may change frequently.

- Regular (strong) entity types such as EMPLOYEE, DEPARTMENT, and PROJECT are shown in rectangular boxes.


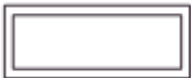







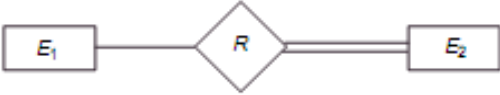
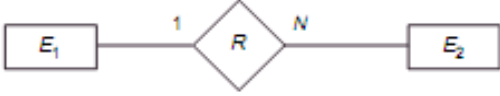
- Relationship types such as WORKS_FOR, MANAGES, CONTROLS, and WORKS_ON are shown in diamond-shaped boxes attached to the participating entity types with straight lines.
- Attributes are shown in ovals, and each attribute is attached by a straight line to its entity type or relationship type.
- Component attributes of a composite attribute are attached to the oval representing the composite attribute, as illustrated by the Name attribute of EMPLOYEE.
- Multivalued attributes are shown in double ovals, as illustrated by the Locations attribute of DEPARTMENT.
- Key attributes have their names underlined.
- Derived attributes are shown in dotted ovals, as illustrated by the Number_of_employees attribute of DEPARTMENT.
- Weak entity types are distinguished by being placed in double rectangles and by having their identifying relationship placed in double diamonds, as illustrated by the DEPENDENT entity type and the DEPENDENTS_OF identifying relationship type.
- The partial key of the weak entity type is underlined with a dotted line.
- the cardinality ratio of each binary relationship type is specified by attaching a 1, M, or N on each participating edge.
- The cardinality ratio of DEPARTMENT:EMPLOYEE in MANAGES is 1:1,
- whereas it is 1:N for DEPARTMENT: EMPLOYEE in WORKS_FOR, and M:N for WORKS_ON.
- The participation constraint is specified by a single line for partial participation and by double lines for total participation (existence dependency).

Proper Naming of Schema Constructs

- singular names is used for entity types, rather than plural ones, because the entity type name applies to each individual entity belonging to that entity type.
- In our ER diagrams, entity type and relationship type names are in uppercase letters,
- attribute names have their initial letter capitalized, and role names are in lowercase letters.

- Another naming consideration involves choosing binary relationship names to make the ER diagram of the schema readable from left to right and from top to bottom.

Symbols used in ER diagram

Symbol	Meaning
	ENTITY TYPE
	WEAK ENTITY TYPE
	RELATIONSHIP TYPE
	IDENTIFYING RELATIONSHIP TYPE
	ATTRIBUTE
	KEY ATTRIBUTE
	MULTIVALUED ATTRIBUTE
	COMPOSITE ATTRIBUTE
	DERIVED ATTRIBUTE
	TOTAL PARTICIPATION OF E_2 IN R
	CARDINALITY RATIO 1: N FOR $E_1:E_2$ IN R