OVERVIEW OF DATABASE MANAGEMENT SYSTEM

- **Data :** Known facts that can be recorded and that have implicit meaning.
- For eg : Consider the names, phone numbers, and the address of the people.
- This collection of related data with an implicit meaning is a database.

Database :

- A database is a collection of related data.
- A database represents some aspects of the real world, sometimes called as mini world. Changes to the mini world are reflected in the database.
- A database is a logically coherent collection of data with some inherent meaning
- A database can be of any size and complexity
- An eg. Of a large commercial database is Amazon.com . It contains data for over millions of active users, books, electronic items etc
- A database may be generated and maintained manually or it may be computerized
- A computerized database may be created and maintained either by a group of application programs or by a database management system.

• A **database management system (DBMS)** is a computerized system that enables users to create and maintain a database.

• **DBMS** is a ***general purpose software system that facilitates the process of defining, constructing, manipulating and sharing database among various users and applications***

• **Defining** a database involves specifying the data types, structures, and constraints of the data to be stored in the database.

• Database definition or descriptive information is also stored by DBMS in the form of a database catalog or dictionary and it is called as **metadata (data about data)**

• Constructing database is the process of storing the data on some storage medium that is controlled by DBMS.

• Manipulating a database includes functions such as querying the database to retrieve specific data, updating the database

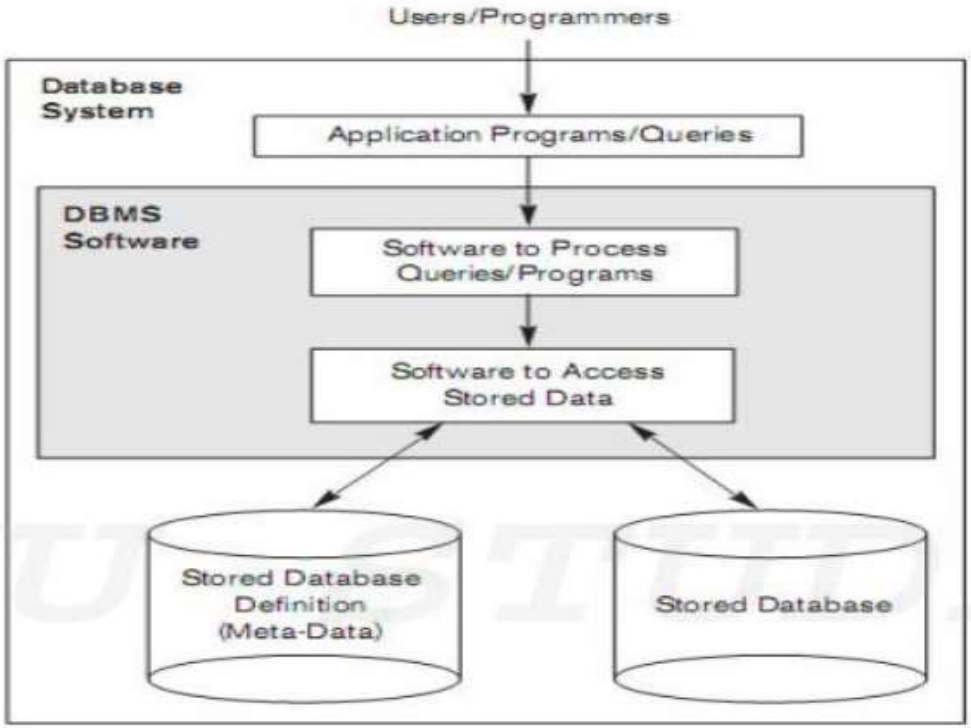
• Sharing a database allows multiple users and programs to access the database simultaneously.

• Other important functions provided by DBMS include protecting the database and maintaining it over a long period of time.

• Protection includes the system protection against hardware or software malfunction and security protection against the unauthorized access.

• The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

Simplified database system environment



- Users or programmers can access the dbms software by using application programs or queries
- In DBMS software , it contains :
    - Software to process the queries
    - Software to access the stored data
- DBMS software can access the data from the stored database.

Typical File System V/S DBMS

- In typical file processing system, permanent records stored in various files.
- Limitations of File System are :
    - Data redundancy
    - Data inconsistency
    - Unsharable data
    - Unstandardized data
    - Insecure data
    - Incorrect data

- DBMS can overcome all these limitations
- DBMS features to manage the data in robust and efficient manner
- In DBMS,
    - Avoid redundancy
    - Avoid inconsistency
    - Shared data
    - Standardized data
    - Provide security in data

CHARACHTERISTICS OF DATABASE APPROACH

• In db approach , a single repository maintains data that is defined once and then accessed by various users repeatedly through queries, transactions, and application programs.

• The main characteristics are :

- Self describing nature of database system
- Insulation between programs and data, and data abstraction
- Support of multiple views of the data
- Sharing of the data and multiuser transaction processing

1. **Self describing nature of database system**

• Database system contain not only the database itself but also a complete definition or description of database structure and constraints.

• This definition is stored in the dbms catalog which contains the information such as structure of each file, the type and the storage format of each data item

• The information stored in the catalog is called **meta data**

• Rather the data is stored as self describing data that includes the data item names and data values together in one structure

2. **Insulation between programs and data, and data abstraction**

• The structure of data files is stored in the DBMS catalog separately from the access programs.

• This property is called **program-data independence**

• An operation (also called a function or method) is specified in two parts.

- **Interface** : The interface of an operation includes the operation name and the data types of its arguments
- **Implementation** : is specified separately and can be changed without affecting the interface.

• The characteristic that allows program-data independence and program operation independence is called **data abstraction.**

• A data model is a type of data abstraction that is used to provide conceptual representation

• In database approach detailed structure and organization of each file are stored in a catalog

3. **Support of multiple views of the data**

• A database has many type of users, each user may require a different perspective or **view** of the database.

• A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored.

• For eg : In a library management system in college

  • User1→ Admin : Admin includes the details like user accounts, fine management, book acquisition records, and system logs

  •User 2→ Librarian : Librarian focuses book availability, issue records, and due dates

  • User 3 → Students : For members only shows the book issued to them and their due dates

4. **Sharing of the data and multiuser transaction processing**

• DBMS must include concurrency control software

  • to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct

• DBMS must enforce several transaction properties

  ▫ Isolation property : Ensures that each transaction appears to execute in isolation from other transactions even though hundreds of transactions may be executing concurrently.

  ▫ Atomicity property : Ensures that either all the database operations in a transaction are executed or none In this case we have to ensure that data should be restored to a consistent state

DATABASE USERS

- Several db users are :
  - Database Admin
  - Database Designers
  - System Analysts and Application Programmers
  - End Users

1. Database Administrator (DBA)

- Chief administrator to oversee and manage the resources
- Administering the resources is the responsibility of a DBA
- Responsible for authorizing access to the database, coordinating and monitoring its use.
- Also acquiring software and hardware resources
- DBA should be responsible for the problems such as security breaches and poor system response time

2. Database Designers

- DB designers are responsible for identifying the data to be stored in the database
- Choose the appropriate structures to represent and store this data
- Responsible to communicate with all perspective db users in order to understand their requirements (ie, how the data to be stored, processed and how it reused)
- Also responsible for creating a design that meets all these requirements
- They interact with each group of users and develop views.

3. End Users

- They are the people whose jobs require access to the database for querying , updating, and generating reports, the database primarily exists for their use
- Several categories of end users are :
    - Casual End Users
    - Naïve Users
    - Sophisticated End Users
    - Standalone Users

a) 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.

b) Naïve Users

- Also called parametric end users
- Their main job function is constantly querying and updating the database.
- A few examples are :
    - Bank customers check account balances and post withdrawals and deposits
    - Social media users post and read items on social media web sites

c) Sophiticated End Users

• Include engineers, scientists, business analysts and others who thoroughly familiarize themselves with the facilities of DBMS

• They are familiar with the database and they implement their own applications

d) Standalone Users

• Maintain personal database 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** determine the requirements of end users , especially the naïve users
- Develop specifications for standard canned transactions that meet their requirements.
- **Application Programmers** implement these specifications as programs
- Then they test, debug, document, and maintain these canned transactions
- Such analysts and programmers commonly referred to as software developers or software engineers

Advantages of DBMS

a) Controlling Redundancy

- In typical file system, redundancy in storing the same data multiple times may lead to several problems such as duplication of effort, storage space is wasted, files that represent the same data may become inconsistent
- In the db approach, the views of different user groups are integrated during the db design
- Ideally, we should have a database design that stores each logical data item is known as **data normalization ,** it ensures consistency and saves storage space

b) Restricting Unauthorized Access

- When multiple users share a large database , it is likely most users will not be authorized to access all information in the database
- A DBMS should provide a **security and authorization subsystem**

c) Providing persistent storage for program objects

- The persistent storage of program objects and data structures is an important function of a database systems
- Object oriented database systems offers data structure compatibility with one or more object oriented programming languages

d) Providing Storage Structures and Search Techniques for Efficient Query Processing

• Database systems must provide capabilities for efficiently executing queries and updates

•The query processing and optimization module of the DBMS is responsible for choosing an efficient query execution plan for each query based on the existing storage structures

e) Providing Backup and Recovery

• A DBMS must provide facilities for recovering from hardware or software failures.

• The backup and recovery subsystem of the DBMS is responsible for recovery

f) Providing Multiple User Interfaces

• Because many types of users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interfaces.

• These include

- query languages for casual users
- programming language interfaces for application programmers
- forms and command codes for parametric users
- menu-driven interfaces and natural language interfaces for standalone users

g) Representing Complex Relationships among Data

h) Enforcing Integrity Constraints

• New functionality being added to DBMS is :

- Scientific Applications
- Extensible Markup Language (XML)
- Image, Audio, Video Data Management
- Data Warehousing & Data Mining
- Spatial & Time Data Management

Disadvantages of DBMS

• In spite of the advantages of using a DBMS, there are a few situations in which a DBMS may involve unnecessary overhead costs

- High initial investment in hardware, software,and training
- Overhead for providing security, concurrency control, recovery, and integrity functions

STRUCTURED, SEMI STRUCTURED & UNSTRUCTURED DATA

- Datas are known facts that can be recorded and have an implicit meaning
- Eg . Are :
    - Name , Marks , Phone number , Age etc
- In DBMS , data can be categorized into 3, based on
    - how the data to be organized
    - how the data to be stored
    - how these data are to be accessed

And the categories are :

- Structured Data
- Semi Structured Data
- Un- Structured Data

Structured Data

• Structured data is data whose elements are addressable for effective analysis.

•It has been organized into a formatted repository that is typically a database.

• It concerns all data which can be stored in database SQL in a table with rows and columns.

• They have relational keys and can easily be mapped into pre-designed fields.

• Easy to query using SQL

• *Example:* Relationaldata.

Characteristics of Structured Data

- Data conforms to a data model and has easily identifiable structure
- Data is stored in the form of rows and columns **Example : Database**
- Data is well organised
- Data resides in fixed fields within a record or file
- Data elements are addressable, so efficient to analyse and process

Advantages of Structured Data:

- Easy to manage
- High level of consistency and integrity
- Enhanced data security

Disadvantages of Structured Data:

- Not Flexible for handling complex data types
- Expensive

Semi-Structured Data

- Semi-structured data is information that does not reside in a relational database but that has some organizational properties that make it easier to analyze.
- ie, it does not have strict table format , but it contains tags to separate data elements
- With some processes, you can store them in the relation database
- *Example*: XML data.

Characteristics of semi-structured Data:

- Data does not conform to a data model but has some structure.
- Data can not be stored in the form of rows and columns as in Databases
- It contains tags and elements which is used to group data and describe how the data is stored
- Similar entities are grouped together and organized in a hierarchy
- Entities in the same group may or may not have the same attributes or properties
- Does not contain sufficient metadata which makes automation and management of data difficult
- Size and type of the same attributes in a group may differ
- Due to lack of a well-defined structure, it can not used by computer programs easily

Advantages of Semi-Structured Data :

1. More flexible than structured data
2. Scalable
3. Easy to manage the nested data
4. Faster data processing

Disadvantages of Semi- Structured Data :

• Queries are less efficient as compared to structured data.

• Lack of standardization

• Semi-structured data can be more difficult to secure than structured data

Unstructured Data

Unstructured data is a data which is not organized in a predefined manner or does not have a predefined data model

•. So for Unstructured data, there are alternative platforms for storing and managing

• Ie, it requires advanced techniques (such as AI or ML) for processing

• *Example*: Word, PDF, Text, Media logs.

Characteristics of Unstructured Data

- **Lack of Format**: Unstructured data does not fit neatly into tables or databases
- **Variety**: This type of data can include a wide range of formats, such as: Text documents, Multimedia files, Social media content etc
- **Volume**: Unstructured data represents a significant portion of the data generated today. It is often larger in volume compared to structured data.

Advantages :

- It supports the data that lacks a proper format or sequence
- The data is not constrained by a fixed schema
- Very Flexible due to the absence of schema.
- Data is portable
- It is very scalable
- It can deal easily with the heterogeneity of sources.
- These types of data have a variety of business intelligence and analytics applications.

Disadvantages :

- Difficult to analyze without special tools

DATA MODELS, SCHEMAS, AND INSTANCES

• One fundamental characteristic of the database approach is that it provides some level of **data abstraction.**

• Data abstraction generally refers to the suppression of details of data organization and storage, and the highlighting of the essential features for an improved understanding of data

• **DATA MODEL**

- A collection of concepts that can be used to describe the structure of a database (how the data are to be logically structured, stored , accessed etc) that provides the means to **achieve abstraction**.
- Ie , the data abstraction which can be achieved by creating data models
- By structure of a database we mean the data types, relationships, and constraints that apply to the data.
- Most data models also include a set of basic operations for specifying retrievals and updates on the database
- In simple, data model consist of the structure as well as the basic operations

Categories of Data Models

• Data models which we can categorize according to the types of concepts they use to describe the database structure

- High-level or conceptual data models
- Low-level or physical data models
- Representational (or implementation) data models

1. **High-level or conceptual data models**

- Also called as **semantic model**
- It provide the concept that are close to the way many users perceive data.
- Ie, describes that how the users view the datas
- Conceptual data models use concepts such as entities, attributes, and relationships
- Eg : Entity Relationship Models (ER Models)

2. **Low-level or physical data models**

- It describes that how the datas are stored in the computer
- Ie, how the data is represented internally
- Eg : B Tree, Hashing techniques

3. **Representational (or implementation) data models**

- Provide the concepts that fall between above two models
- It is used by many commercial DBMS implementations
- Eg : Relational Model, Network Model etc

Database Schema

• The description of a database is called the **database schema**, which is specified during database design and is not expected to change frequently.

• 3 types of database schema are : Physical schema, logical schema, view schema

• Most data models have certain conventions for displaying schemas as diagrams.

• A displayed schema is called a **schema diagram (** illustrative display of db schema)

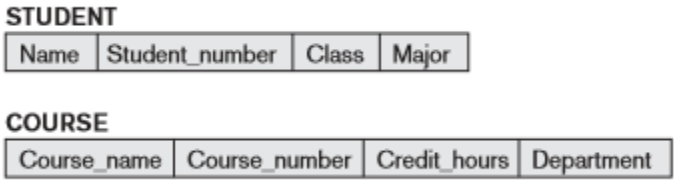


Fig : Schema Diagram for a database

• The diagram displays the structure of each record type but not the actual instances of records.

• And call each object in the schema—such as STUDENT or COURSE—a **schema construct**.

• Ie , component of the schema or it is an object within the schema

• The actual data in a database may change quite frequently

• The data in the database at a particular moment in time is called a **database state or snapshot.**

• It is also called the **current set of occurrences or instances** in the database.

THREE-SCHEMA ARCHITECTURE AND DATA INDEPENDENCE

• **Levels of Abstraction / 3 Schema Architecture /ANSI Architecture**

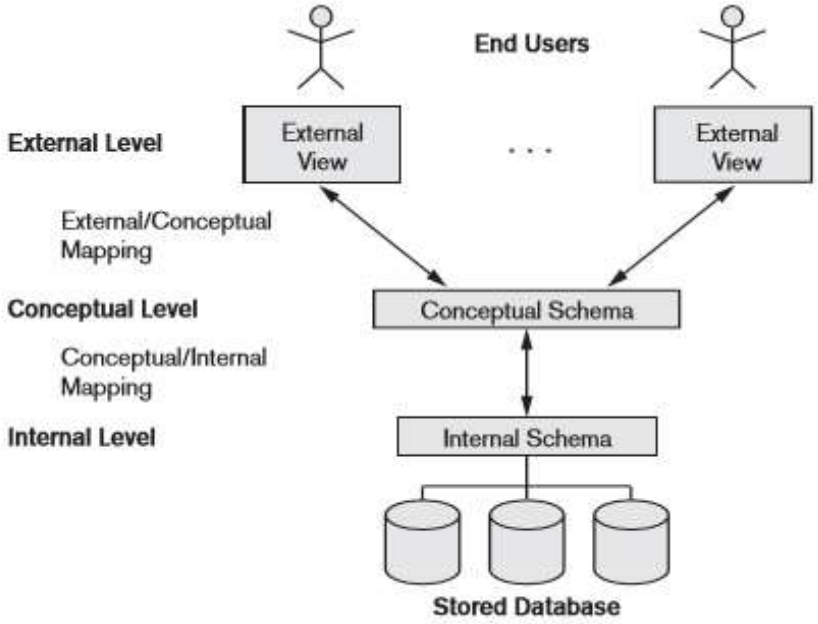
• It is a framework which is used to describe the structure of a specific database system

• The goal of the three-schema architecture is to separate the user applications from the physical database

• It seperates the database system into 3 levels to provide the data abstraction and data independence for the users and applications.

• It ensures data storage and user interaction remain distinct

• One of the main objective of thid architecture is to enable the multiple users to access the same data with a personalized view (and this can be achieved through the data abstraction)



Schemas can be defined at the following three levels:

1. **Internal Schema (Physical Schema)**

- The internal level has an internal schema ,which describes the physical storage structure of the database.
- The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.
- ie, it describes **how the data is physically stored in database** , including the data structures, file organizations
- There is no user interaction in this level
- Key Characteristics :
    - Handles efficiency , performance and storage management
    - Hidden from end users and application developers

2. **Conceptual Schema (Logical Schema)**

- The conceptual level has a conceptual schema, which **describes the structure of the whole database** for a community of users (ie, what data are to be used and what relationship exists among the datas)
- The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.
- Usually, a representational data model is used to describe the conceptual schema when a database system is implemented.
- This implementation conceptual schema is often based on a conceptual schema design in a high-level data model.
- Key characteristics of this level is it ensures consistency and integrity across applications

3. External Schema (View Schema)

- The external or view level includes a number of external schemas or user views.
- Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.
- Ie, **it defines how specific users or applications view the data**
- As in the previous level, each external schema is typically implemented using a representational data model, possibly based on an external schema design in a high-level data model.
- Key Characteristics are :
    - Provides security by restricting access to sensitive data
    - Offers user friendly views of complex data

Mappings

• The DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database.

• If the request is a database retrieval, the data extracted from the stored database must be reformatted to match the user's external view.

• The processes of transforming requests and results between levels are called **mappings**.

• ie, **it describes how the 3levels actually correspond (relate) to each other**

• These mappings may be time-consuming

• Two types of mapping :

- **Conceptual / Internal Mapping**
  - It lies in between conceptual level and internal level
  - It is to define the correspondence between records , fields of conceptual level and files, data structures of internal level
- **External / Conceptual Mapping**
  - It lies in between external level and conceptual level
  - It is to define the correspondence between particular external level and conceptual view

DATA INDEPENDENCE

- The three-schema architecture can be used to further explain the concept of data independence, which can be defined as the **capacity to change the schema at one level of a database system without having to change the schema at the next higher level**
- Two types of data independence:
    - Logical data independence
    - Physical data independence

Logical Data Independence

- It is the capacity to change the conceptual schema without having to change external schemas or application programs.
- We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item)

Physical data independence

- It is the capacity to change the internal schema without having to change the conceptual schema.
- Hence, the external schemas need not be changed as well

Database Languages

- DBMS has appropriate languages to express the database queries and updates
- DB languages can be used to read, store and update the data in the database

1. Data Definition Languages (DDL)

- It is used to define database structure
- Ie, to create schema, tables , indexes and constraints in the db
- It is used by the DBA and by database designers to define both schemas.
- The DBMS will have a DDL compiler whose function is to process DDL statements in order to identify descriptions of the schema constructs and to store the schema description in the DBMS catalog.
- In DBMSs where a clear separation is maintained between the conceptual and internal levels ,the DDL is used to specify the conceptual schema only

- DDL Commands :
  - CREATE - Creates objects in database
  - ALTER - Alters the structure of database
  - TRUNCATE - Remove all records from table
  - DROP - To delete the objects from database
  - RENAME - Rename an object
  - COMMENT - Comment on data dictionary

2. Data Manipulation Languages

- Once the database schemas are compiled and the database is populated with data, users must have some means to manipulate the database.
- Typical manipulations include retrieval, insertion,deletion,and modification of the data.
- The DBMS provides a set of operations or a language called the data manipulation language (DML) for these purposes.
- ie, It is used for accessing and manipulating the data in a database
- It handles the user requests

- DML Commands :
    - SELECT - Retrieve the data from database
    - INSERT - Insert data into a table
    - UPDATE - Update the existing data within a table
    - DELETE - To delete all records from a table

3 . Data Control Languages

- It is used to retrieve the stored data
- DCL Commands are : GRANT - Give user access privilages to database
  REVOKE - To take back permissions from the user

DBMS Interfaces

- It is the mechanism that enable communication and interaction between database and its users , applications etc
- In other words, this will act as a bridge that facilitates the data retrieval, manipulation and management of data
- It allows for ability to input queries to a database without using query languages itself

- **User-friendly interfaces provided by a DBMS may include the following:**

1. Menu-Based Interfaces

- ***It is designed for Web Clients or Browsing*** .
- These interfaces present the user with lists of options (called menus) that lead the user through the formulation of a request.
- Menus do away with the need to memorize the specific commands and syntax of a query language; rather, the query is composed step-by step by picking options from a menu that is displayed by the system.
- ***Pull-down menus*** are a very popular technique in Web-based user interfaces. They are also often used in browsing interfaces, which allow a user to look through the contents of a database in an exploratory and unstructured manner.
- Key Features are :
    - Predefined options
    - Error reduction
    - Consistency
- One of the drawback is , limited flexibility and scalability (ie, users are restricted to predefined options

2. Forms-Based Interfaces.

- A forms-based interface displays a form to each user.
- Users can fill out all of the form entries to insert new data, or they can fill out only certain entries, in which case the DBMS will retrieve matching data for the remaining entries.
- Forms are ***usually designed and programmed for naive users*** as interfaces to canned transactions.
- Key Features are :
    - It makes user interactions straight forward
    - Validates the input data to reduce mistakes

3. Graphical User Interfaces.

- A GUI typically displays a schema to the user in diagrammatic form.
- The user then can specify a query by manipulating the diagram.
- In many cases, GUIs utilize both menus and forms.
- Most GUIs ***use a pointing device***, such as a mouse, to select certain parts of the displayed schema diagram.
- Key features are :
    - Visual representation of data
    - Simplified Query Execution
    - Error handling and validation

4. Natural Language Interfaces.

- These interfaces accept requests written in English or some other language and attempt to understand them.
- A natural language interface usually has its own
- The natural language interface refers to the words in its schema, as well as to the set of standard words in its dictionary, to interpret the request.
- If the interpretation is successful, the interface generates a high-level query corresponding to the natural language request and submits it to the DBMS
- Key features are :
    - Language Parsing (ie, system translates the natural language to structured database queries)
    - It provide flexibility and efficiency

One of the drawback in this interface is ambiguity

5. Speech Input and Output.

- Allows the users to interact with database system through spoken commands and responses
- Limited use of speech as an input query and speech as an answer to a question or result of a request is becoming common
- The speech input is detected using a library of predefined words and used to set up the parameters that are supplied to the queries.
- For output, a similar conversion from text or numbers into speech takes place.

THE DATABASE SYSTEM ENVIRONMENT

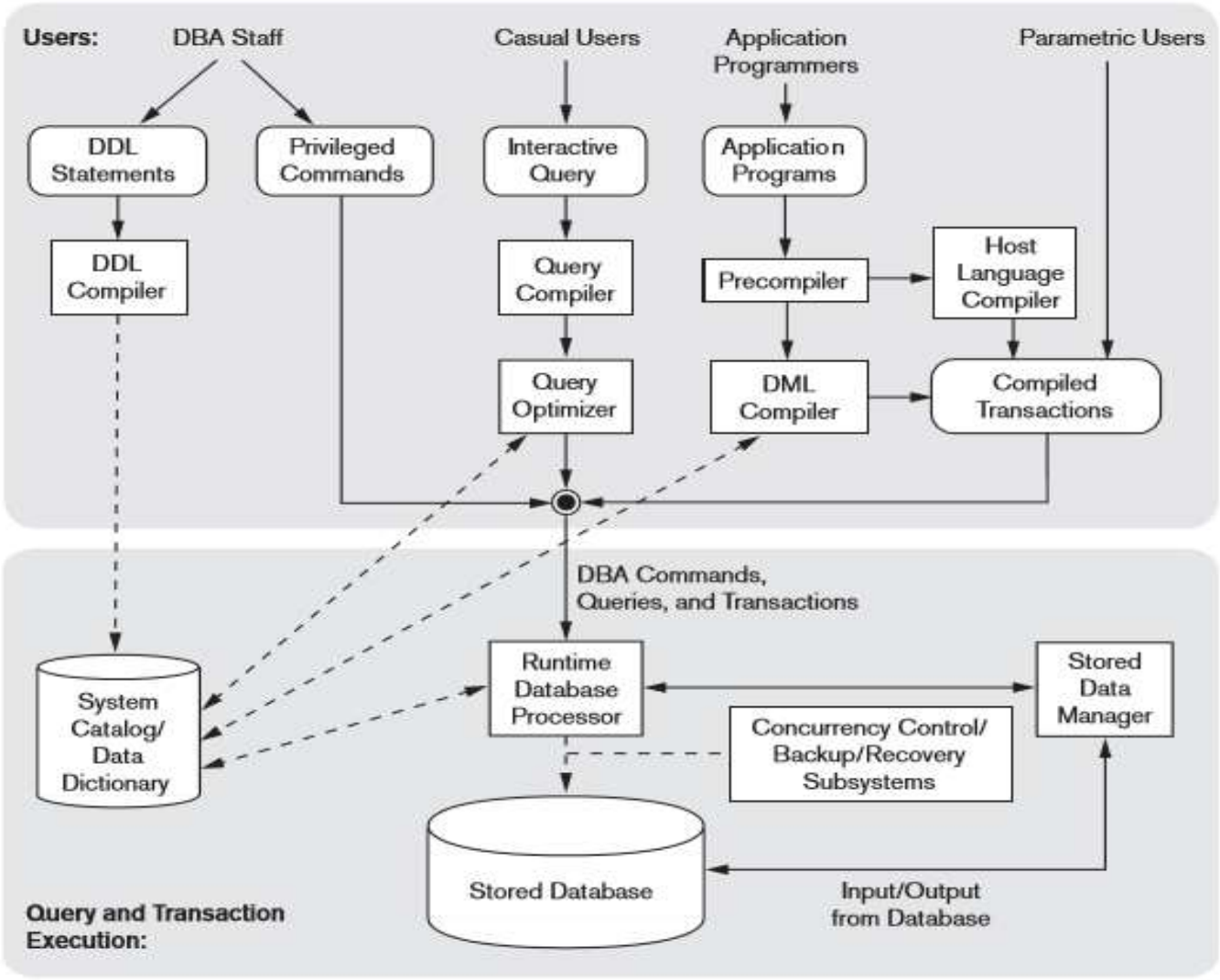


Figure 2.3
Component modules of a DBMS and their interactions.

DBMS Component Modules

- The figure is divided into two parts.
- The top part of the figure refers to the various users of the database environment and their interfaces.
- The lower part shows the internals of the DBMS responsible for storage of data and processing of transactions.

Consider the top part of Figure :

- It shows interfaces for the DBA staff, casual users who work with interactive interfaces to formulate queries, application programmers who create programs using some host programming languages
- . The DDL compiler processes schema definitions, specified in the DDL, and stores descriptions of the schemas (meta-data) in the DBMS catalog.
- The catalog includes information such as the names and sizes of files, names and data types of data items, storage details of each file, mapping information among schemas, and constraints
- Casual users and persons with occasional need for information from the database interact using some form of interface, which we call the interactive query interface
- These queries are parsed and validated for correctness of the query syntax by a query compiler that compiles them into an internal form.
- This internal query is subjected to query optimization
- The query optimizer is concerned with the rearrangement and possible reordering of operations, elimination of redundancies, and use of correct algorithms and indexes during execution

- The precompiler extracts DML commands from an application program written in a host programming language.
- These commands are sent to the DML compiler for compilation into object code for database access.
- The rest of the program is sent to the host language compiler.
- The object codes for the DML commands and the rest of the program are linked, forming a canned transaction whose executable code includes calls to the runtime database processor.

In the lower part of Figure

- The runtime database processor executes
  (1) the privileged commands
  (2) the executable query plans
  (3) the canned transactions with runtime parameters.
- It works with the system catalog and may update it with statistics.
- It also works with the stored data manager, which in turn uses basic operating system services for carrying out low-level input/output (read/write) operations between the disk and main memory.
- The runtime database processor handles other aspects of data transfer, such as management of buffers in the main memory
- Concurrency control and backup and recovery systems separately as a module and they are integrated into the working of the runtime database processor for purposes of transaction management.
- On the other hand, if the computer system is mainly dedicated to running the database server, the DBMS will control main memory buffering of disk pages.
- The DBMS also interfaces with compilers for general purpose host programming languages, and with application servers and client programs running on separate machines through the system network interface.

DATABASE ARCHITECTURES

- A Database stores a lot of critical information to access data quickly and securely. Hence it is important to select the correct architecture for efficient data management.

- DBMS Architecture helps users to get their requests done while connecting to the database.

- Choose database architecture depending on several factors like the size of the database, number of users, and relationships between the users.

- DBMS architecture depends upon how users are connected to the database to get their request done.

- Database architectures refers to the structural design of a database system, which dictates how the data stored, accessed, managed and manipulated.

- DBMS architecture are of different types

    (1) Centralized database

    (2) Basic client/server architecture

    (3) Two tire client server architecture

    (4) Three and n-tire architecture for web applications

1. **Centralized Database**

• A Centralized Database is a type of database that is stored, located as well as maintained at a single location only.

• This type of database is modified and managed from that location itself.

• The centralized location is accessed via an internet connection (LAN, WAN, etc).

• This centralized database is mainly used by institutions or organizations.

• Centralized database architecture can be beneficial for businesses and institutions that need to manage large amounts of data across multiple locations.

• It can help with:

    - **Data consistency**
    - **Data governance**

Data Consistency :

• In Centralized Database, all data are stored in a single central repository

• This reduces the duplication of data, and it ensures that all users access the same, updated and accurate information

• Eg : In a banking system, it ensures that customer's account balance remains consistent across all branches

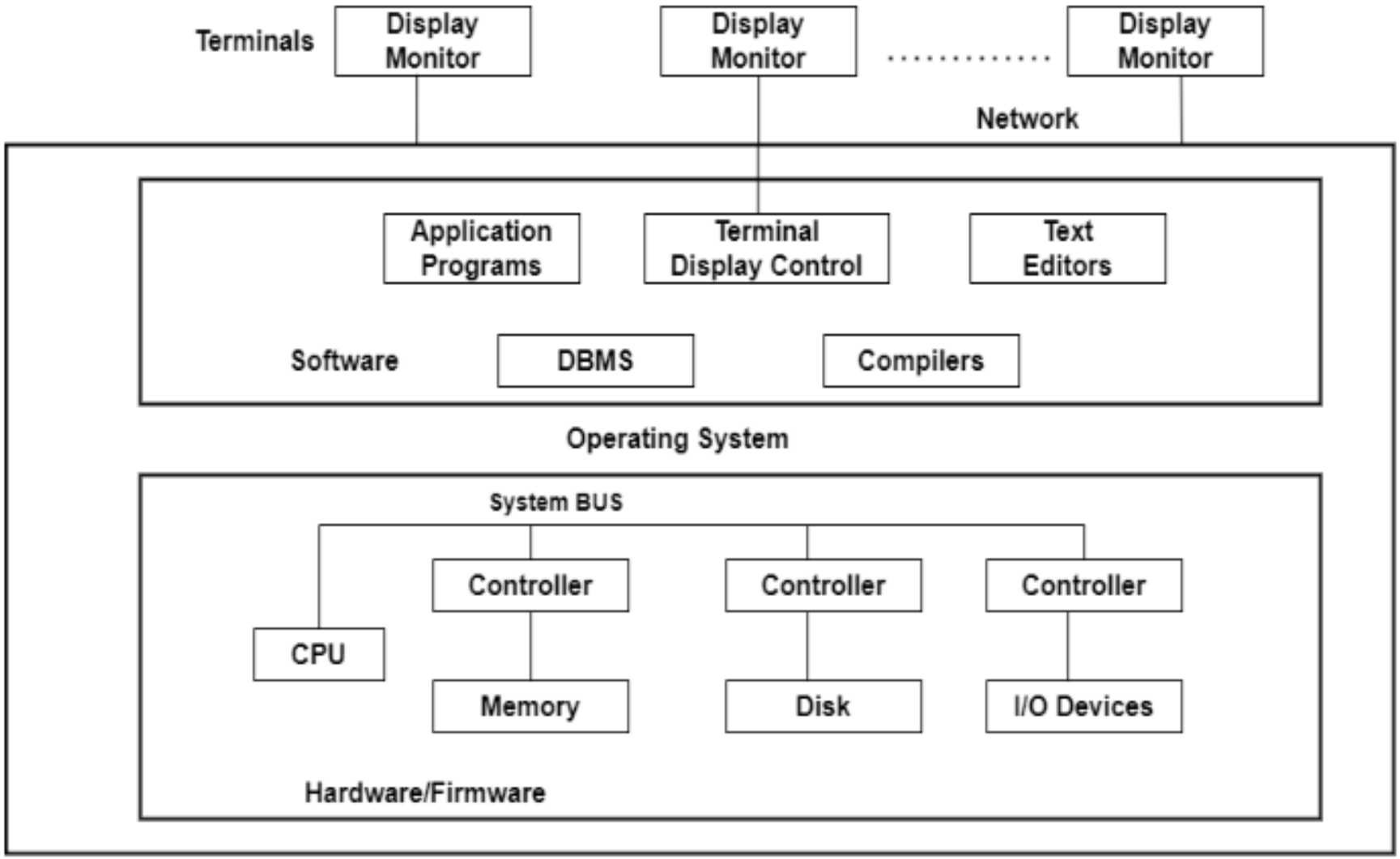
Data Governance:

• It helps to ensure control and effective management of data

• Eg : If government uses a centralized database, it can ensure lawful usage of data and prevent the unauthorized access of data

So, Key features are :

- Easy to manage
- Accuracy and reliability
- Security



Centralized architecture of DBMS

Advantages

• Since all data is stored at a single location only thus it is easier to access and coordinate data.

• The centralized database has very minimal data redundancy since all data is stored in a single place.

• It is cheaper in comparison to all other databases available.

• Easy to manage and backup

Disadvantages

• The data traffic in the case of a centralized database is more.

• If any kind of system failure occurs in the centralized system then the entire data will be destroyed.

• Limited Scalability

2. Basic Client / Server Architecture

- In the client-server architecture, when the client computer sends a request for data to the server through the internet, the server accepts the requested process and delivers the data packets requested back to the client.
- **Client** is a user machine that provide user interface capabilities and local processing (Sends requests to the server, typically SQL queries, and presents the results to the user)
- **Server** system provide service to client such as file (Processes the queries, retrieves the data from the database, and sends it back to the client)

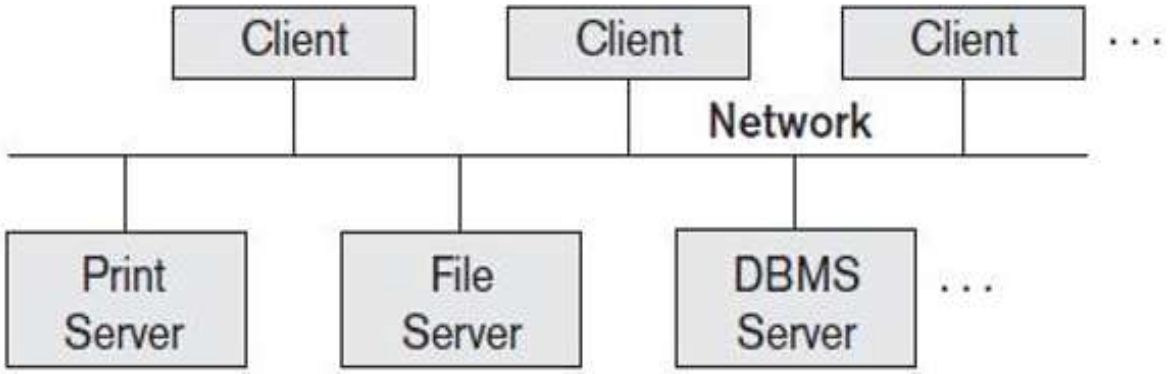


Fig : Logical Two Tier Architecture

a) Two-Tier Client Server Architecture

- Here, the term "two-tier" refers to our architecture's two layers-the Client layer and the Data layer.
- There are a number of client computers in the client layer that can contact the database server.

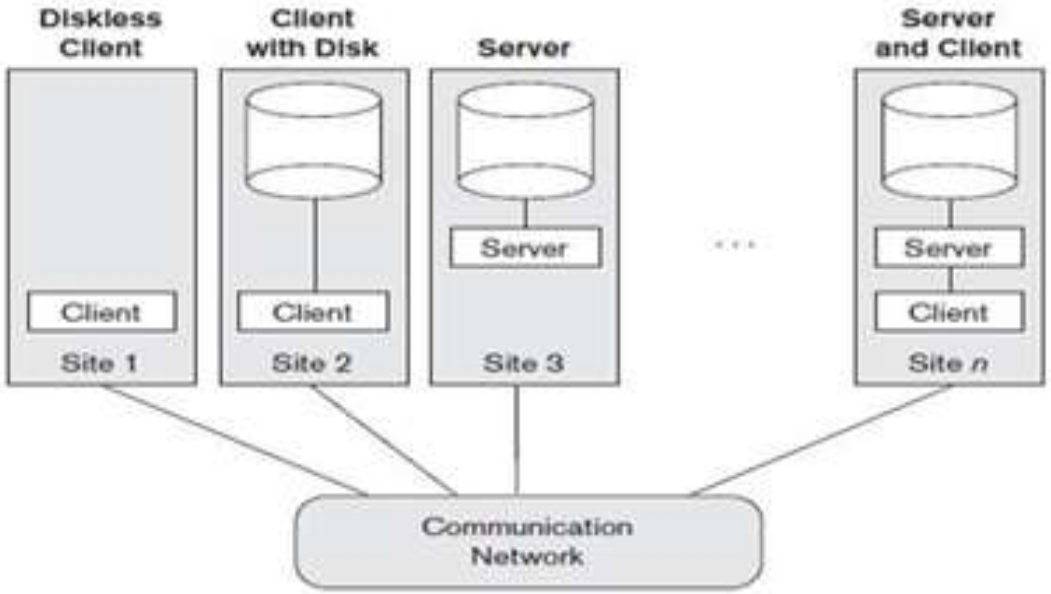


Fig : Physical Representation of two tier architecture

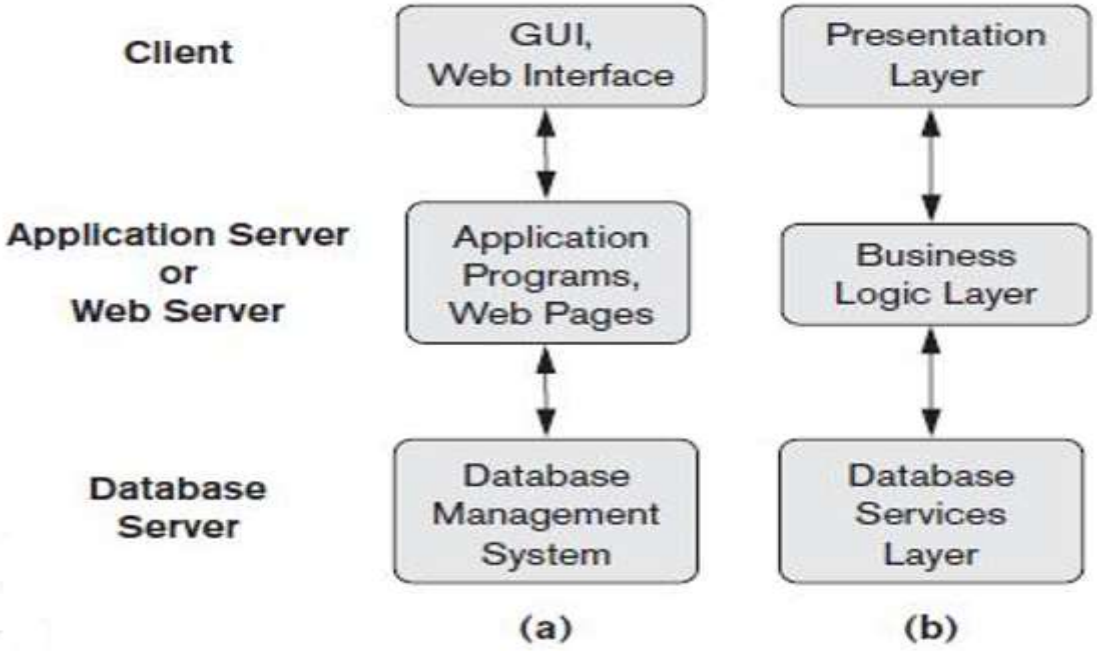
- Created a logical division between client and server
- Client Side :
    - User interface and application programs
- Server Side :
    - Query server and transaction server
- Application Program Interface (API) to access server databases
    - ODBC (Open Database Connectivity Standard)
    - JDBC (for Java Programming Interface)

b) Three-Tier Client-Server Architecture

• The Application Server / Business Logic Layer is an additional layer that serves as a link between the Client layer and the Data layer in this instance.

• The layer where the application programs are processed is the business logic layer, unlike a Two-tier architecture, where queries are performed in the database server.

• Here, the application programs are processed in the application server itself.



In Fig (a) ,

- This intermediate layer or middle tier is called the application server or the Web server, depending on the application.
- This server plays an intermediary role by running application programs and storing business rules (procedures or constraints) that are used to access data from the database server.
- It can also improve database security by checking a client's credentials before forwarding a request to the database server.
- The intermediate server accepts requests from the client, processes the request and sends database queries and commands to the database server, and then acts as a conduit for passing (partially) processed data from the database server to the clients
- Thus, the user interface, application rules, and data access act as the three tiers

In Fig (b) ,

- The presentation layer displays information to the user and allows data entry.
- The business logic layer handles intermediate rules and constraints before data is passed up to the user or down to the DBMS.
- The bottom layer includes all data management services.
- The middle layer can also act as a Web server, which retrieves query results from the database server and formats them into dynamic Web pages that are viewed by the Web browser at the client side.

Advantages

- ***Centralized Management***
- ***Scalability*** : Can handle multiple clients simultaneously
- ***Flexibility*** : Clients can run on different platforms
- ***Data Integrity*** : Server ensures that all data manipulations are consistent and follow the db schema
- ***Enhances Security***

Disadvantages

- ***Network dependency*** : A reliable network is required for communication between clients and servers
- ***High traffic overhead***
- ***Complexity in multi tier systems***

Features	Two Tier Architecture	N- Tier Architecture
Definition	Direct interaction between Client and DB Server	Intermediate layers exist between client and DB Server
Scalability	Limited	Highly Scalable
Security	Database directly exposed	Database secured by applications server
Complexity	Simple	Complex due to multiple layers
Performance	Fast for small applications	Better suited for large scale applications with high concurrency
Structure	Two Layers: Client and Server	Three or more Layers : Presentation, Application and Database Layers
Maintenance	Simple to maintain	More Complex
Flexibility	Less Flexible	Highly Flexible
Cost	Cost effective for small systems	High cost
Examples	Desktop Database Applications	Web based applications(e- commerce or online banking)

CLASSIFICATION OF DBMS

- Database management systems can be classified based on several criteria, such as

1. **Classification Based on Data Model**
2. **Classification Based on Users**
3. **Classification Based on Database Distribution**
4. **Classification Based on Costs**

1. **Classification Based on Data Model**

- Data models in DBMS are classified into several types, including:

a) Relational Data Model (SQL Systems)

b) Object Data Model

c) Object Relational Data Model

d) Hierarchical Data Model

e) Network Data Model

f) Flat Data Model

g) Big Data Systems (NoSQL Systems)

a) Relational Data Model (SQL Systems)

- The most popular data model today
- A relational database is defined as a group of independent tables which are linked to each other using some common fields of each related table.
- This model can be represented as a table with columns and rows.
- ***Tables*** are also known as ***relations***
- Each ***row*** is known as a ***tuple***.
- Each table of the ***column*** has a name or ***attribute***.
- It is well knows in database technology because it is usually used to represent real-world objects and the relationships between them.
- Relational model makes the query much easier than in hierarchical or network database systems
- Some popular relational databases are used nowadays like Oracle, Sybase, DB2, MySQL Server etc

Relational Model Terminologies:

Relation	Table
Tuple	Row / Record
Attribute	Column / Feild
Domain	It consist of set of legal values
Cardinality	It consists of number of rows
Degree	It consists of number of columns

- The relational model represents how data is stored in Relational Databases.
- A relational database consists of a collection of tables, each of which is assigned a unique name.
- Consider a relation ***STUDENT*** with attributes ***ROLL_NO, NAME, ADDRESS, PHONE, and AGE*** shown in the table.

Table Student

ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI		18

- **Attributes** : ROLL_NO, NAME, ADDRESS, AGE, PHONE

- **Data Item** **:** The smallest unit of data in the relation is the individual data item. It is stored at the intersection of rows and columns are also known as cells.

- **Tuple** **:** The above relation contains 4 tuples, one of which is shown as:

1	RAM	DELHI	9455123451	18

- **Column** **:** The column ROLL_NO is extracted from the relation STUDENT.

ROLL_NO
1
2
3
4

- **Degree :**

  - It consists of no. of attributes or columns
  - The STUDENT relation defined above has degree 5.
  - The degree of a relation doesn't change with time as tuples get added or deleted.
  - Ie, degree of a relation is fixed and based only on no.of attributes.
    - 1 attribute → Unary relation (degree=1)
    - 2 attributes → Binary relation (degree=2)
    - 3 attributes → Ternary relation (degree=3)
    - n attributes → N-Ary relation (degree= n)

- **Cardinality :**
  - It consists of number of tuples or rows
  - The STUDENT relation defined above has cardinality 4.
  - Adding or deleting tuples will change the cardinality .
- **NULL Values :**
  - The value which is not known or unavailable is called a NULL value.
  - It is represented by blank space.
  - e.g.; PHONE of STUDENT having ROLL_NO 4 is NULL.
- **Domain :**
  - It contains a set of atomic values that an attribute can take.
  - It could be accomplish explicitly by listing all possible values or specifying conditions that all values in that domain must be confirmed.
  - **For example :** the domain of gender attributes is a set of data values "M" for male and "F" for female.

- **Relation Schema :**
    - A relation schema defines the structure of the relation and represents the name of the relation with its attributes.
    - e.g.; STUDENT (ROLL_NO, NAME, ADDRESS, PHONE, and AGE) is the relation schema for STUDENT.
    - If a schema has more than 1 relation, it is called Relational Schema.

- **Relational instance :**
    - In the relational database system, the relational instance is represented by a finite set of tuples.
    - Relation instances do not have duplicate tuples.

- **Relational key :**
    - In the relational key, each row has one or more attributes.
    - It can identify the row in the relation uniquely.

PROPERTIES OF RELATIONS

- Each attribute in a relation has only one data value corresponding to it i.e. they do not contain two or more values.
- Name of the relation is distinct from all other relations.
- Each relation cell contains exactly one atomic (single) value
- Each attribute contains a distinct name
- Attribute domain has no significance
- Tuple has no duplicate value
- Order of tuple can have a different sequence
- It also provides information about metadata.

ADVANTAGES OF THE RELATIONAL MODEL

- **Simple model :** Relational Model is simple and easy to use.
- **Flexible :** Relational Model is more flexible than any other relational model present.
- **Secure:** Relational Model is more secure than any other relational model.
- **Data Accuracy:** Data is more accurate in the relational data model.
- **Data Integrity:** The integrity of the data is maintained in the relational model.
- **Operations can be Applied Easily**

DISADVANTAGES OF THE RELATIONAL MODEL

- Relational Database Model is not very good for large databases.
- Sometimes, it becomes difficult to find the relation between tables.
- Because of the complex structure, the response time for queries is high.

b) Object Data Model

- In Object Oriented Data Model, data and their relationships are contained in a single structure which is referred as object in this data model.
- In this, real world problems are represented as objects with different attributes.
- All objects have multiple relationships between them.

Advantages of Object Oriented Data Model :

- Codes can be reused due to inheritance.
- Easily understandable.
- Cost of maintenance can reduced due to reusability of attributes and functions because of inheritance.

Disadvantages of Object Oriented Data Model :

It is not properly developed so not accepted by users easily.

c) Object Relational Data Model (ORDBMS)

- It is a database system that combines the advantages of an object-oriented programming model with the efficiency of a relational database system.
- It is useful for managing and organizing complex data, especially when the data relationships are complicated

Advantages of ORDBMS

- **SQL Support:** It supports the structures of the SQL query language, are quite flexible and make it easier to handle complex queries.
- **Scalability:** It is easier and can accommodate large data sets.
- **Interoperability**

Disadvantages of ORDBMS

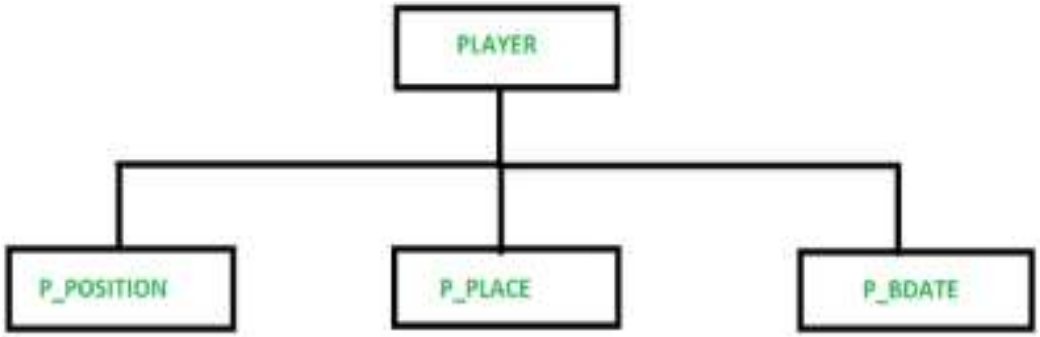
- **Complexity in Implementation**

d) Hierarchical Data Model

- A hierarchical model represents the data in a tree-like structure
- In this model , data is stored in the form of records (or nodes) with parent – child relationships
- In a hierarchical model, data are viewed as a collection of tables

- Key Features :
    - Data is represented as a tree
    - `Each record has one parent and can have multiple children
    - Root node represents the top of the hierarchy, and all other nodes descend from it
    - Relationships between data elements are defined at the schema level
    - Accessing data requires navigating the hierarchy from the root node to the desired child node.

Example 1: Consider the below cricket database system hierarchical model scheme.



Structure :

Nodes : Represent the individual records or data items

Edges : Represent relationships between parent and child nodes

Advantages of the hierarchical model :

- It has a very simple hierarchical database structure.
- It has data sharing as all data are held in a common database data and therefore sharing of data becomes practical.
- It offers data security and this model was the first database model that offered data security.
- Also offers data integrity as it is based on the parent-child relationship and also there's always a link between the parents and the child segments.
- Data is easily accessed due to its predefined structure

Disadvantages of the hierarchical model :

- Lack of flexibility
- Data Redundancy , due to the strict parent - child relationships
- Cannot handle many- to- many relationships efficiently

e) Network Data Model

- It is a data model that allows the representation of ***many-to-many relationships***
- Like a hierarchical model, this model also does not have any database standard
- And this model allows to represent multi parent relationships.
- It is not commonly used due to its complexity.

f) Flat Data Model

- It is a simple data model that stores all data in a single table with columns and rows.

g) NoSQL DBMS

- It is designed for handling large volumes of unstructured or semi-structured data.
- Includes document databases, key-value stores, column-family stores, and graph databases.
- Data access varies depending on the specific NoSQL type.

2. Classification Based on Users

a) Single-User Database Systems

- A single-user database system is designed to be used by only one user at a time.
- Simple, but they are not suitable for environments where multiple users need to access the same data at the same time.
- **Example:** Personal Computers

Advantages :

- Simplicity
- Lower Costs
- Less Complexity

Disadvantages :

- Limited Scalability
- Not Ideal for Large Data Handling

b **) Multi-user Database Systems**

- It can be accessed by multiple users simultaneously..
- They are more complex and require more resources than single-user systems, but they are essential for organizations where multiple users need to access the same data at the same time.
- **Example:** Databases of Banks, insurance agencies, stock exchanges, supermarkets, etc.

Advantages :

- Concurrency
- Scalability
- Data Consistency and Integrity

Disadvantages :

- Higher Complexity
- Cost
- Performance Overhead

3. Classification Based on Database Distribution

Databases can be classified based on distribution as either homogeneous or heterogeneous distributed databases

a) Homogeneous Database:

- In a homogeneous database, all different sites store database identically. ie, use the same DBMS software from multiple sites
- Data exchange between these various sites can be handled easily.
- Hence, they're easy to manage.

b) Heterogeneous Database:

- Different sites might use different DBMS software, but there is additional common software to support data exchange between these sites.

Using High-Level Conceptual Data Models for Database Design

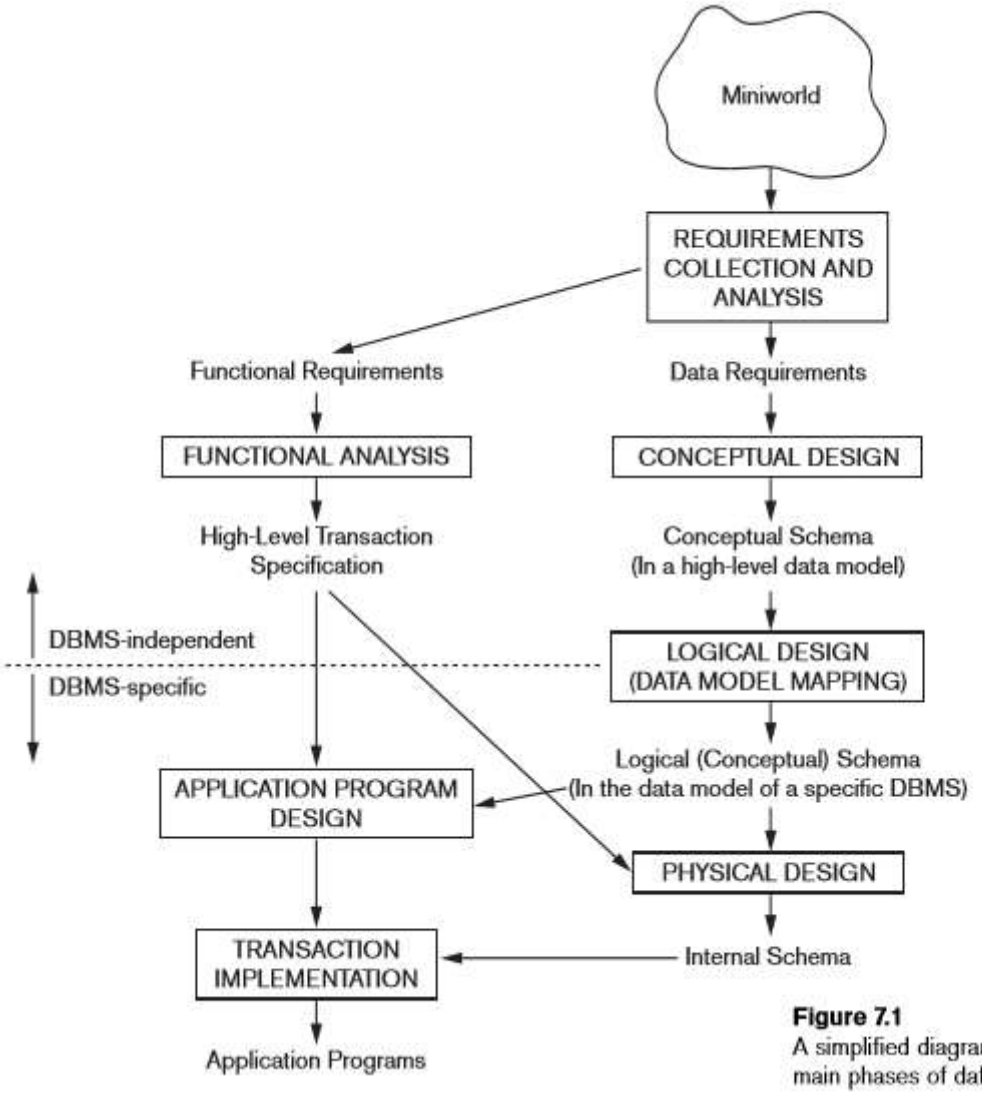


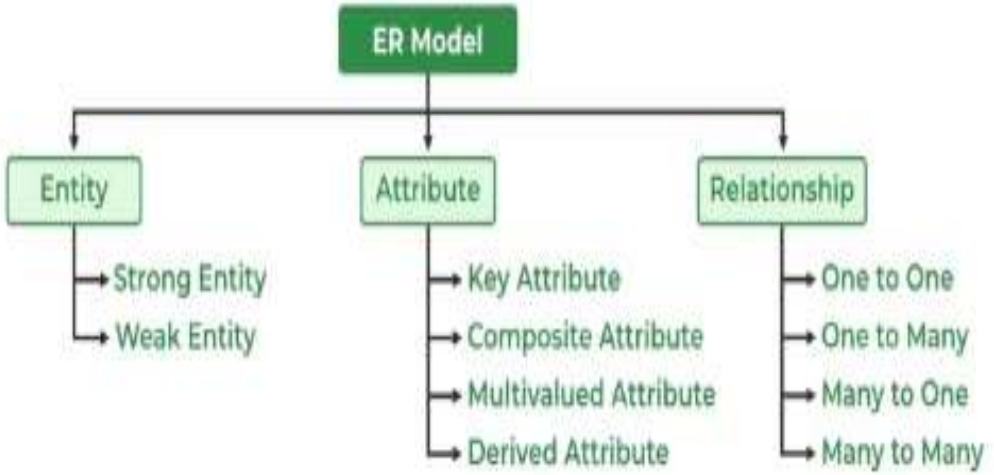
Figure 7.1
A simplified diagram to illustrate the main phases of database design.

- The first step shown is **requirements collection and analysis**.
- During this step, the database designers interview prospective database users to understand and document their data requirements
- Once the requirements have been collected and analyzed, the **next step is to create a conceptual schema** for the database , using a high-level conceptual data model.
- The conceptual schema is a concise description of the data requirements of the users these concepts do not include implementation details
- The next step in database design is the **actual implementation of the database**, using a commercial DBMS.
- Most current commercial DBMSs use an implementation data model—such as the relational or the object-relational database model
- So the conceptual schema is transformed from the high-level data model into the implementation data model. This step is called **logical design or data model mapping**
- The **last step is the physical design phase**, during which the internal storage structures, file organizations, indexes, access paths, and physical design parameters for the database files are specified

ER MODEL

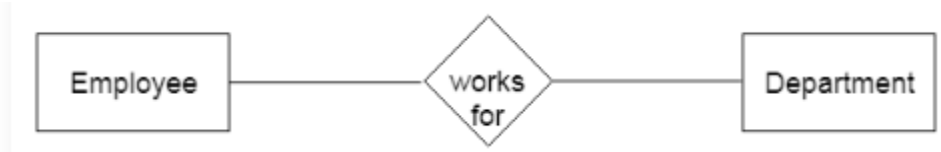
- ER model stands for an Entity- Relationship Model
- It is **a high level conceptual data model**
- This model is used to define the data elements and relationship for a specified system.
- It is used for conceptual data design of database applications
- It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an **entity-relationship diagram.**
- The Entity Relationship Diagram explains the relationship among the entities present in the database.
- ER models are used to model real-world objects like a person, a car, or a company and the relation between these real-world objects.
- In short, the ER Diagram is the structural format of the database and for non technical users
- These diagrams are very easy to understand and easy to create even for a naive user. It gives a standard solution for visualizing the data logically.

COMPONENT OF ER DIAGRAM



ENTITY

- An Entity may be **an object with a physical existence** – a particular person, car, house, or employee
- Or it may be **an object with a conceptual existence** – a company, a job, or a university course.
- In the ER diagram, an entity can be represented as rectangles.
- Entity is distinguished from other objects on basis of attributes
- Entities can be tangible and intangible
- Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



Types of Entity

There are two types of entity:

1. Strong Entity

.

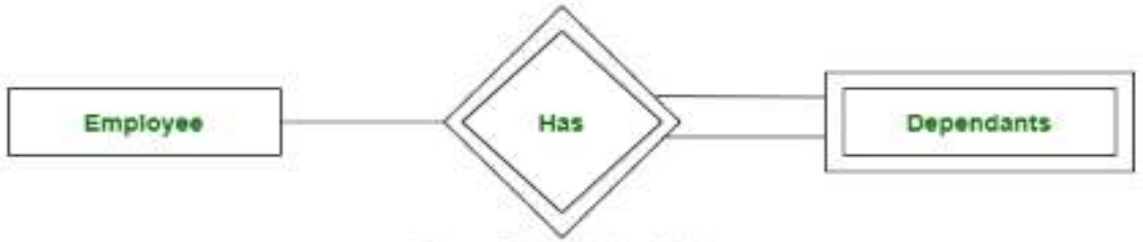
2. Weak Entity

1. STRONG ENTITY

- A type of entity that has a key Attribute.
- Strong Entity does not depend on other Entity in the Schema.
- It has a primary key, that helps in identifying it uniquely
- It is represented by a rectangle.

2. WEAK ENTITY

- An Entity type has a key attribute that uniquely identifies each entity in the entity set.
- A weak entity type is represented by a Double Rectangle.
- The relationship between the weak entity type and its identifying strong entity type is called **identifying relationship** and it is represented by a double diamond.



Strong Entity and Weak Entity

Examples for Strong and weak entity

1. In a banking system, have two entities : Transaction and Account

   Transaction has stheir own properties such as Trans_id,Trans_date,Amount

   Similiarly, Account entity has their own properties : Acc_id, Acc_type,Balance

   Here, Account is a strong entity and Transaction is a weak entity

   ie, Transaction entities and their properties all are depends on other entity Account

Another eg :

2. In educational system , have two entities : Course and Module

   Course have properties : Courseid, Coursename, Duration

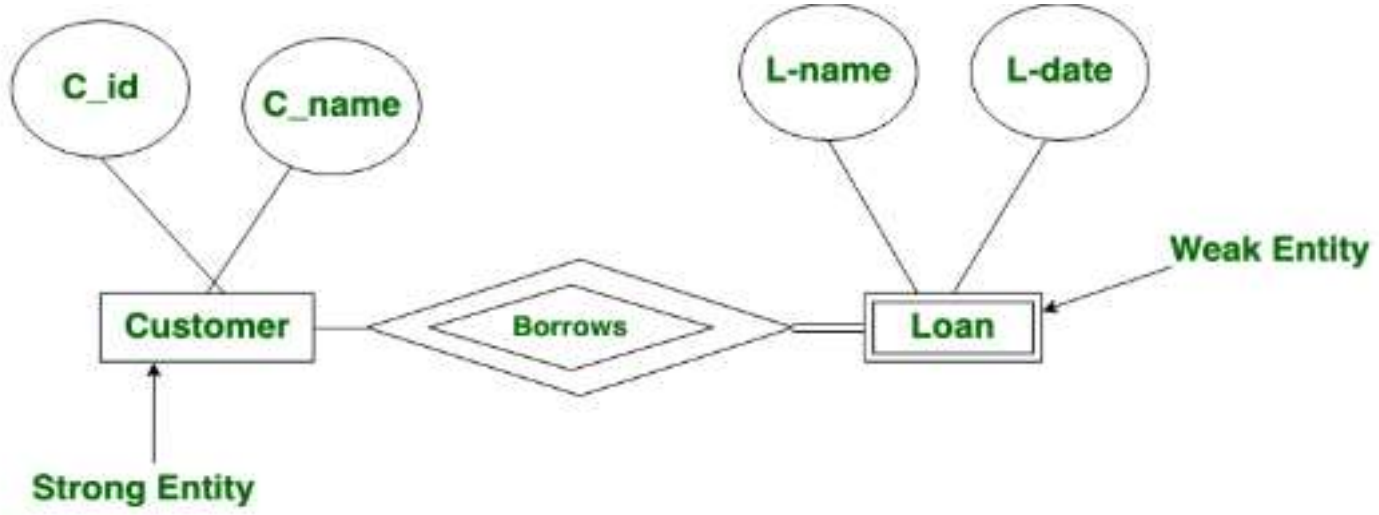
   Module have properties : Moduleid, Title

   Here, Course is a strong entity and Module is a weak entity

   ie, Module entity that will depends on the other entity Course

Difference between Strong Entity& Weak Entity

Strong Entity	Weak Entity
Strong entity always has a primary key.	While a weak entity has a partial discriminator key.
Strong entity is not dependent on any other entity.	Weak entity depends on strong entity.
Strong entity is represented by a single rectangle.	Weak entity is represented by a double rectangle.
Two strong entity's relationship is represented by a single diamond.	While the relation between one strong and one weak entity is represented by a double diamond.
Strong entities have either total participation or partial participation.	A weak entity has a total participation constraint.

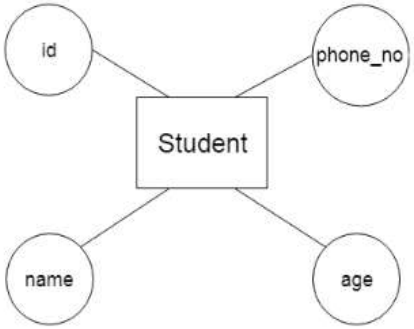


In ER models,

- **Strong entities can exist independently**
- **Weak entities depend on strong entities.**

ATTRIBUTE

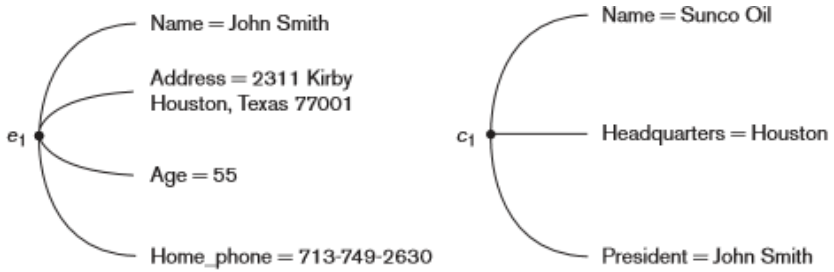
- Each entity has attributes
- The attribute is used to **describe the property of an entity**.
- Eclipse or Oval is used to represent an attribute.
- An entity may contain any number of attributes.
- Attributes can also be subdivided into another set of attributes.
- Attributes help define and organize the data, making it easier to retrieve and manipulate information within the database.
- For eg, id, age, contact number, name, etc. can be attributes of a student.
- Attributes define the properties of entities in an ER model
- A particular entity will have a value for each of its attributes.
- The attribute values that describe each entity become a major part of the data stored in the database.



For eg :

• The EMPLOYEE entity e1 has four attributes: Name, Address, Age, and Home_phone; their values are ‘John Smith,’ ‘2311 Kirby, Houston, Texas 77001’, ‘55’, and ‘713-749-2630’, respectively.

• The COMPANY entity c1 has three attributes: Name, Headquarters, and President; their values are ‘Sunco Oil’,‘Houston’,and ‘John Smith’, respectively.



Several types of attributes occur in the ER model

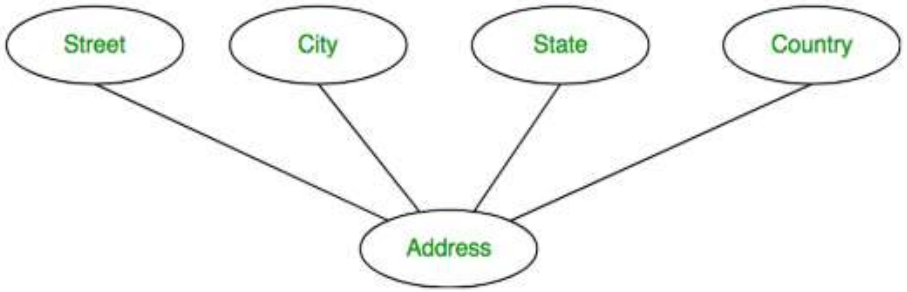
- Simple Attribute
- Composite Attribute
- Single valued Attribute
- Multi valued Attribute
- Stored Attribute
- Derived Attribute
- Key Attribute
- Null Attribute
- Descriptive Attribute

1. Simple Attribute

- Simple attributes are atomic values
- ie, an attribute that cannot be further subdivided into components is a simple attribute.
- **Eg:** The roll number of a student, the ID number of an employee, gender, and many more.

2. Composite Attribute

- An attribute composed of many other attributes is called a composite attribute.
- An attribute that can be split into components is a composite attribute.
- In ER diagram, the composite attribute is represented by an oval comprising of ovals
- **For eg**, the Address attribute of the student Entity type consists of Street, City, State, and Country.
- Composite attributes can form a hierarchy
- If the composite attribute is referenced only as a whole, there is no need to subdivide it into component attributes.

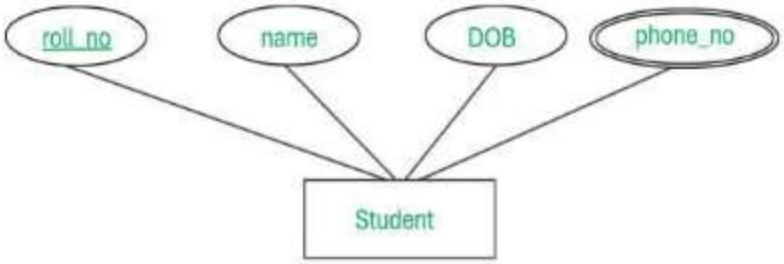


3. Single-Valued Attribute

- The attribute which takes up only a single value for each entity instance
- **Eg:** The age of a student, Aadhar card number , DOB of of a person , Vehicle registration number

4. Multi-valued Attribute

- An attribute consisting of more than one value for a given entity.
- They represent a one to many relationship with an entity
- In ER diagram, a multi valued attribute is represented by a double oval.
- **For eg,** a student can have more than one phone number or email address



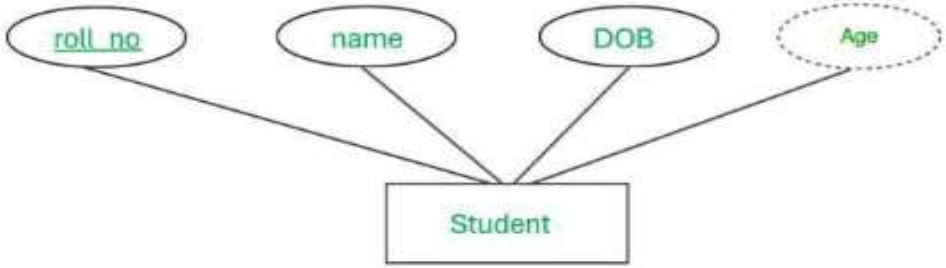
5. Stored Attribute

- The stored attribute are those attribute which doesn't require any type of further update since they are stored in the database.
- ie, whose values are explicitly stored in the database

- **Eg:** DOB(Date of birth) is the stored attribute.

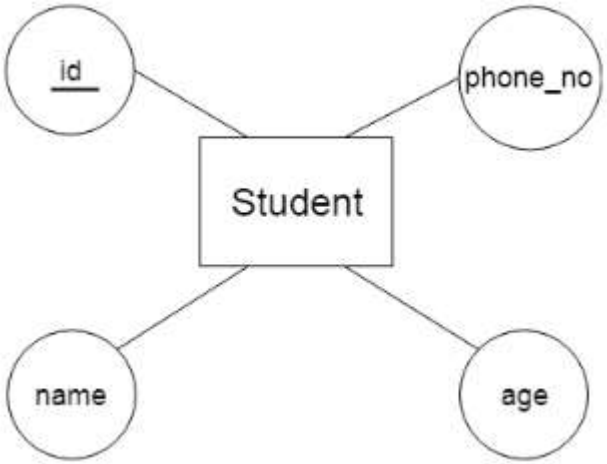
6. Derived Attribute

- An attribute that can be derived from other attributes of the entity type is known as a derived attribute.
- ie, they are not exist in the physical database, but their values are derived from other attributes
- So this avoids the data redundancy and save storage spaces
- **e.g**.; Age (can be derived from DOB). [Age = Current Date – DOB]
- In ER diagram, the derived attribute is represented by a dashed or dotted oval.



7. Key attribute

- Key attributes are those attributes that can uniquely identify the entity in the entity set.
- The key attribute is used to represent the main characteristics of an entity.
- The key attribute is represented by an ellipse with the text underlined.
- It ensures the data integrity and avoid the duplication of data
- **Eg:** Roll-No is the key attribute because it can uniquely identify the student.



8. Complex Attribute

- In general, composite and multi-valued attributes can be nested arbitrarily.
- We can represent arbitrary nesting by grouping components of a composite attribute between parentheses () and separating the components with commas, and by displaying multi-valued attributes between braces { }. Such attributes are called complex attributes.
- These attributes are rarely used in DBMS.
- **Eg:** Address because address contain composite value like street, city, state, PIN code and also multi valued because one people has more that one house address.

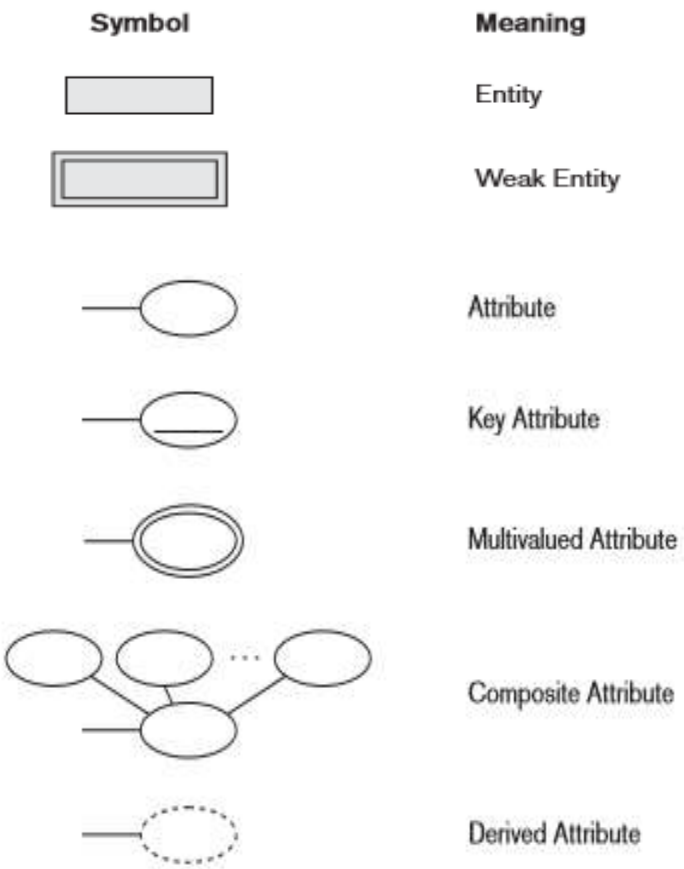
{Address_phone({Phone(Area_code,Phone_number)},Address(Street_address (Number,Street,Apartment_number),City,State,Zip))}

Figure
A complex attribute: Address_phone.

9. Null Attribute / Null Values

- In some cases, a particular entity may not have an applicable value for an attribute.
- For eg, in a Person entity, attribute Spouse_Name may not have a value if a person is not married
- Then , we can give a special value called ‘NULL’ is created
- Null value is a value which is not inserted but it does not hold zero value.
- The attributes which can have a null value called null valued attributes.
- NULL can also be used if we do not know the value of an attribute for a particular entity
- **For eg**, if we do not know the home phone number of a person

Symbols Used in Entity Model



ENTITY TYPE

- A database usually contains groups of entities that are similar.
- For eg.,a company employing hundreds of employees may want to store similar information concerning each of the employees. These employee entities share the same attributes, but each entity has its own values for each attribute.
- **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.

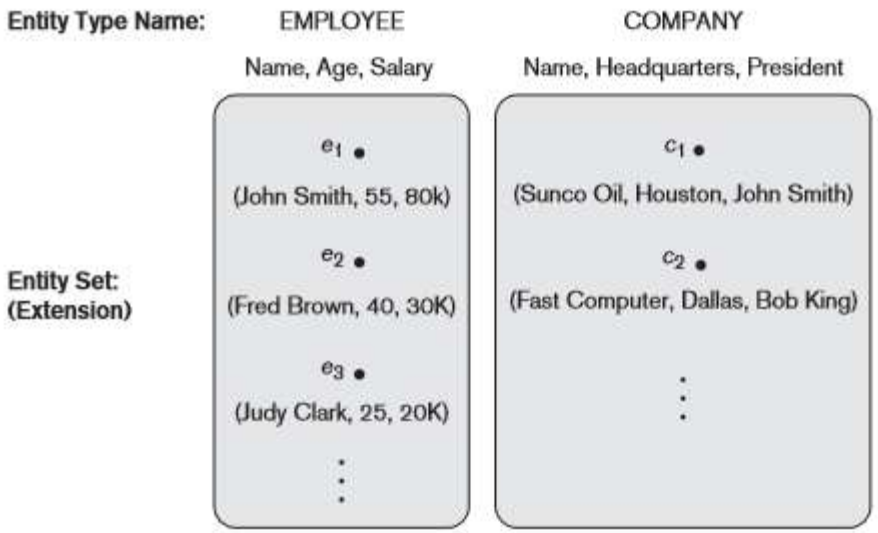


Figure
Two entity types, EMPLOYEE and COMPANY, and some member entities of each.

Entity set

- The collection of all entities of a particular entity type in the database at any point in time is called an entity set
- In other words, the collection of same type of entities that is their attributes are same is called entity set
- The entity set is usually referred to using the same name as the entity type
- We can say that entity type is a superset of the entity set as all the entities are included in the entity type
- **An entity type is represented in ER diagrams as a rectangular box enclosing the entity type name.**
- 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.

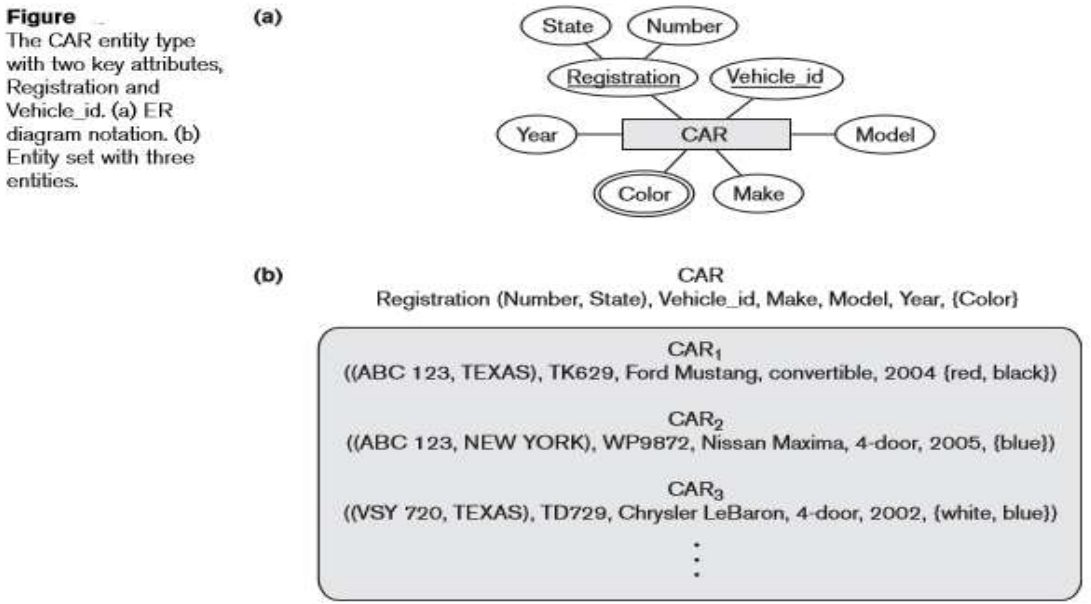


Figure
The CAR entity type with two key attributes, Registration and Vehicle_id. (a) ER diagram notation. (b) Entity set with three entities.

Domain or Value Sets of an attribute

- Each simple attribute of an entity type is associated with a value set (or domain of values), which specifies the set of values that may be assigned to that attribute for each individual entity
- Domain of an attribute is the allowed set of values of that attribute.
- **For eg.,** if attribute is 'grade', then its allowed values are A,B,C,F.

  Ie, Grade ={A, B,C,F}

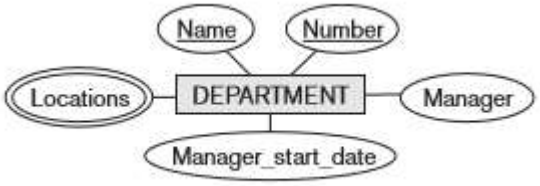
- Value sets are not displayed in ER diagrams
- And are typically specified using the basic data types available in most programming languages, such as integer, string, Boolean, float, enumerated type, subrange, and so on.
- Additional data types to represent common database types such as date, time, and other concepts are also employed.
- Mathematically, an attribute A of entity set E whose value set is V can be defined as a function from E to the power set P(V) of V:

  $A : E \rightarrow P(V)$

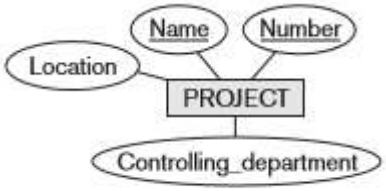
- We refer to the value of attribute A for entity 'e' as A(e).

Define the entity types for the COMPANY database, based on the requirements :

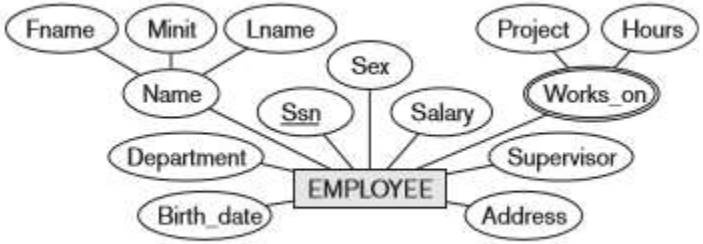
1. An entity type DEPARTMENT with attributes Name, Number, Locations, Manager, and Manager_start_date. Locations is the only multi-valued attribute. We can specify that both Name and Number are (separate) key attributes because each was specified to be unique.
2. An entity type PROJECT with attributes Name, Number, Location, and Controlling_department.Both Name and Number are (separate) key attributes.
3. An entity type EMPLOYEE with attributes Name, Ssn, Sex, Address, Salary, Birth_date, Department, and Supervisor. Both Name and Address may be composite attributes; however, this was not specified in the requirements. We must go back to the users to see if any of them will refer to the individual components of Name—First_name, Middle_initial,Last_name—or of Address.
4. An entity type DEPENDENT with attributes Employee, Dependent_name, Sex, Birth_date,and Relationship (to the employee).



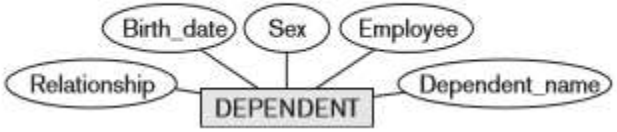
(1)



(2)



(3)



(4)

Relationship

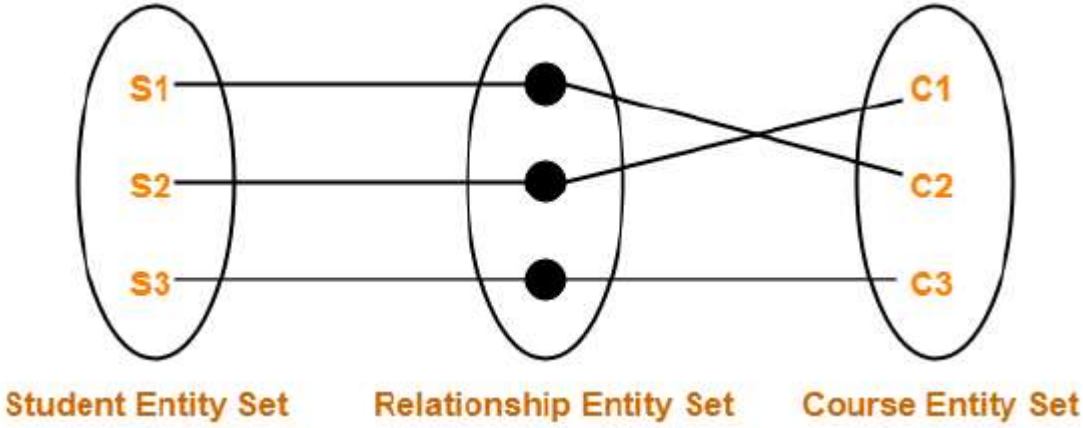
- An attribute of one entity type refers to another entity type, some relationship exists.
- Relates two or more distinct entities with a specific meaning.
- It is an association between two or more entities of same or different entity set
  - For example, EMPLOYEE John works on the PROJECT

Relationship Type

- A Relationship Type represents the association between entity types.
- ie, A relationship is used to describe the relation between entities
- **In ER diagram, the relationship type is represented by a diamond or rhombus and connecting the entities** with lines.
- For example, 'Teaches' is a relationship type that exists between entity type Student and Teacher.

Relationship Set

- A relationship set is a set of relationships of the same type.
- Characteristics of Relationship Set :
  - **Degree :** Indicates to the number of properties related with the relationship set.
  - **Arity :** Arity of a relationship set indicates the number of taking part relations.
  - **Cardinality :** The number of occurrences or records that can be related with each substance on both sides of the relationship.



- A relationship type R among n entity types E1, E2, ..., En defines a set of associations or a relationship
- The relationship set R is a set of relationship instances ri, where each ri associates n individual entities (e1,e2,...,en),and each entity ej in ri is a member of entity set Ej
- Each of the entity types E1, E 2,...,En is said to participate in the relationship type R; similarly, each of the individual entities e1, e2, ..., en is said to participate in the relationship instance ri = (e1,e2,...,en).
- **For eg** : Consider a relationship type WORKS_FOR between the two entity types EMPLOYEE and DEPARTMENT, which associates each employee with the department for which the employee works in the corresponding entity set. Each relationship instance in the relationship set WORKS_FOR associates one EMPLOYEE entity and one DEPARTMENT entity.

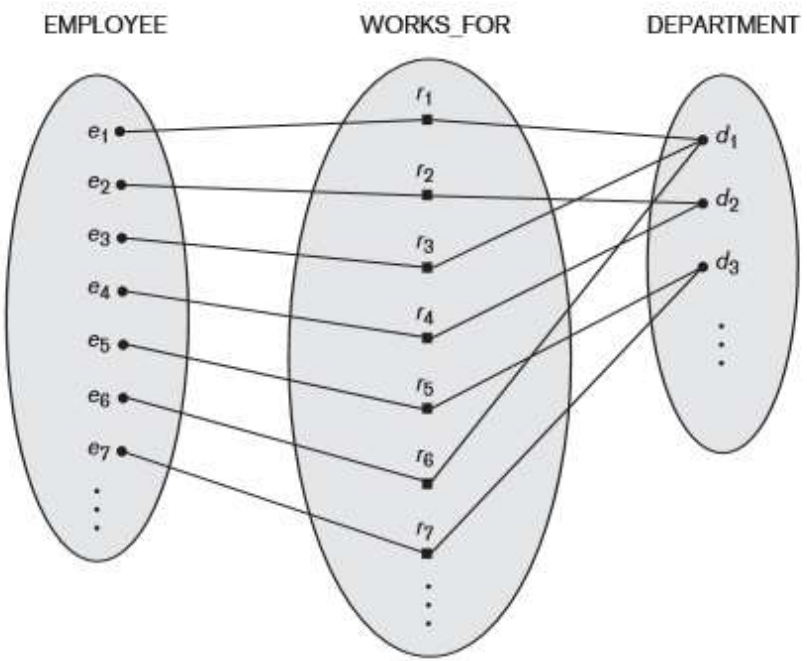
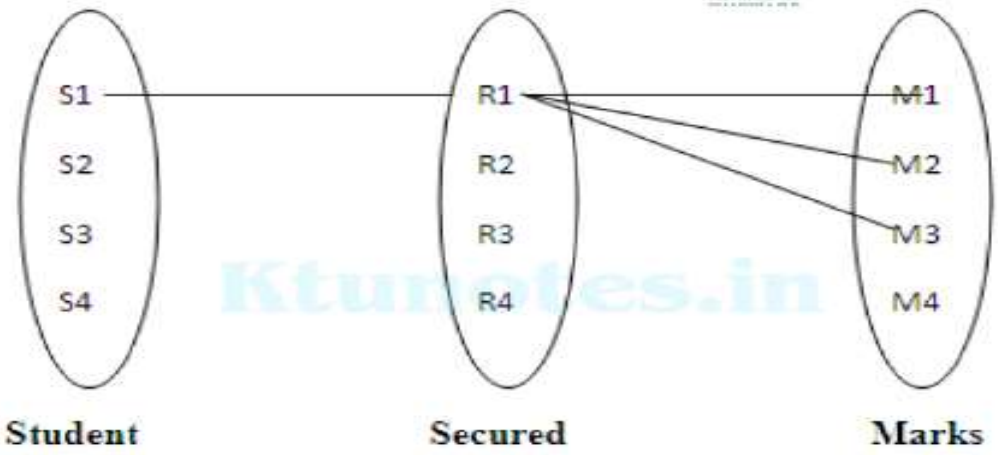


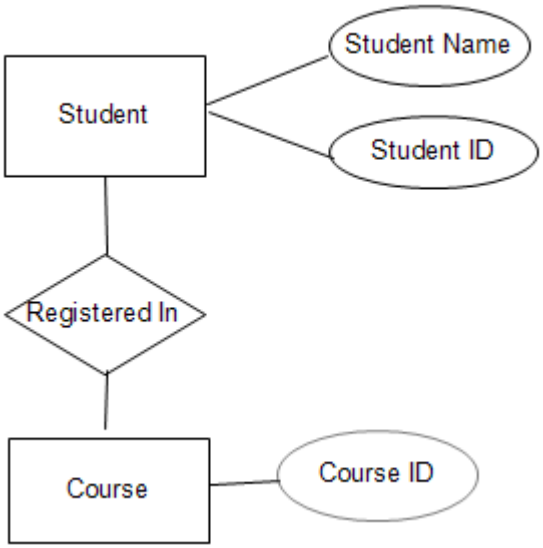
Figure
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Eg : 2



Relationship type: secured
Relationship set: {R1, R2, R3, R4}
Relationship instances: R1

GRAPHICAL REPRESENTATION OF RELATION SETS

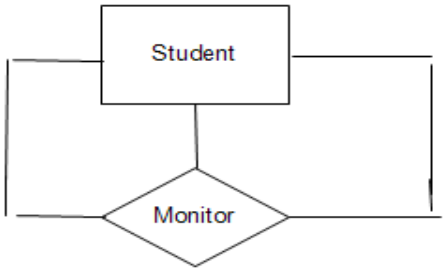


Degree of a Relationship Set

- The number of different entity sets participating in a relationship set.
- ie, the number of participating entity type is called as a degree of relationship type
- Relationships can generally be of any degree, but the ones most common are binary relationships.
- Higher degree relationships are generally more complex than binary relationships

1. Unary Relationship:

- When there is only ***ONE*** entity set participating in a relation
- For example, one person is married to only one person.
- Also called as ***recursive relationship***



2. Binary Relationship:

- When there are ***TWO*** entities set participating in a relation
- For example, a Student is enrolled in a Course.

3. N-ary Relationship:

- When there are n entities set participating in a relationship
- When there are ***THREE*** entities set participating in a relationship is called as **Ternary Relationship**

What is Cardinality?

- The number of times an entity of an entity set participates in a relationship set
- . Cardinality can be of different types:

1. **One-to-One:**

- When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.
- Let us assume that a male can marry one female and a female can marry one male. So the relationship will be one-to-one.

Another example of a 1:1 binary relationship is MANAGES which relates a department entity to the employee who manages that department. This represents the mini world constraints - an employee can manage one department only and a department can have one manager only.

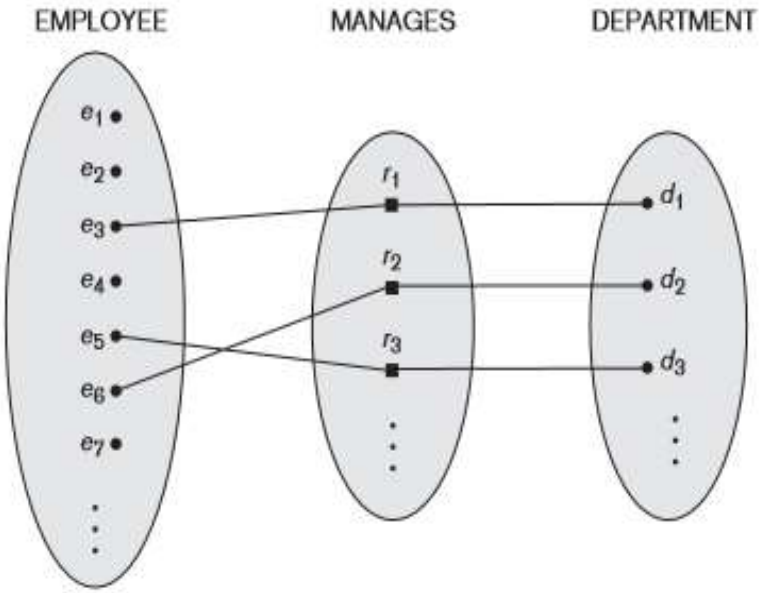
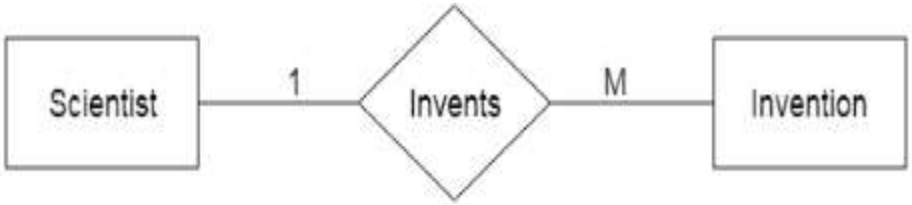


Figure
A 1:1 relationship, MANAGES.

2. One-to-Many:

- When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.
- **For eg,** Scientist can invent many inventions, but the invention is done by the only specific scientist.



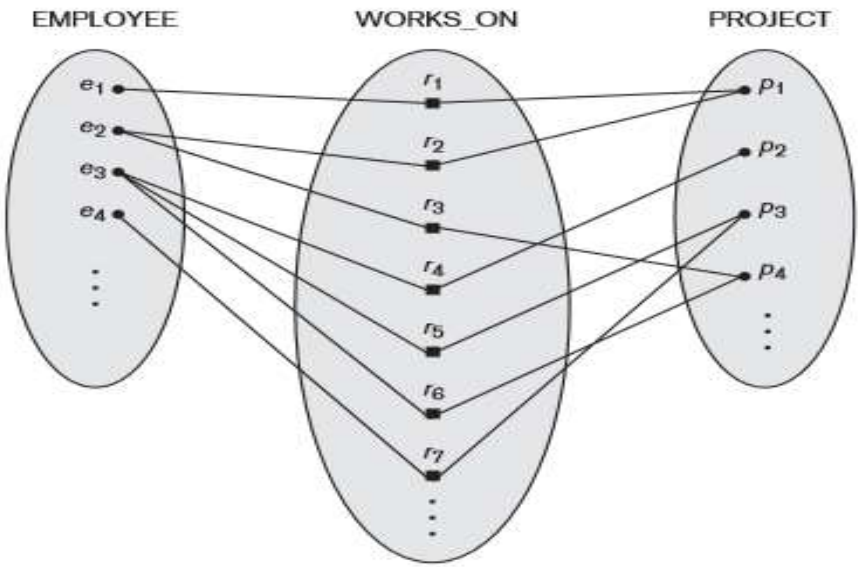
3. Many-to-One:

- When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.
- **For eg,** Student enrolls for only one course, but a course can have many students.

4. Many-to-Many:

- When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.
- **For eg,** Employee can assign by many projects and project can have many employees.

Eg 2: The relationship type WORKS_ON is of cardinality ratio M:N, because the mini world rule is that an employee can work on several projects and a project can have several employees.



Constraints on ER Model

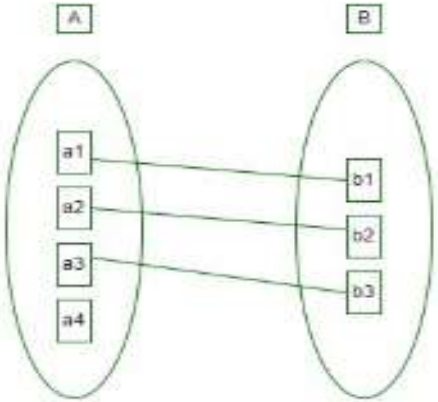
- Constraints are used for modeling limitations on the relations between entities.
- There are two types of constraints on the Entity Relationship (ER) model −
    - Mapping Cardinality or Cardinality ratio
    - Participation Constraints

1. Mapping Cardinality

- It is expressed as the number of entities to which another entity can be associated via a relationship set.
- For the binary relationship set there are entity set A and B then the mapping cardinality can be one of the following :
    - One – to - One
    - One – to - Many
    - Many – to - One
    - Many – to - Many

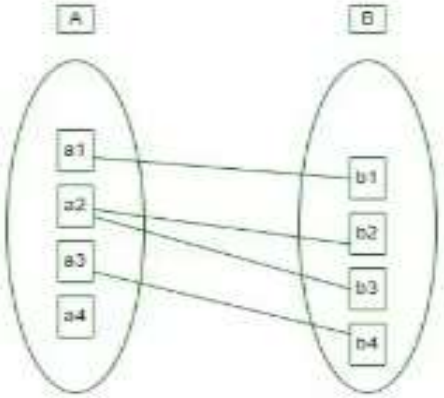
One-to-one relationship

An entity set A is associated with at most one entity in B and an entity in B is associated with at most one entity in A.



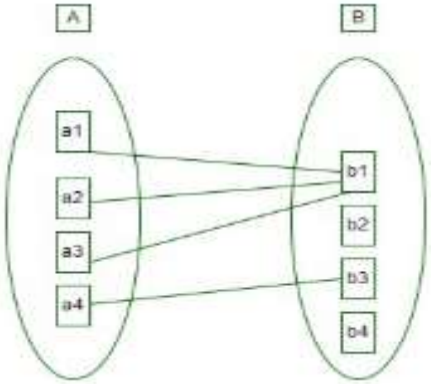
One-to-many relationship

An entity set A is associated with any number of entities in B with a possibility of zero and an entity in B is associated with at most one entity in A.



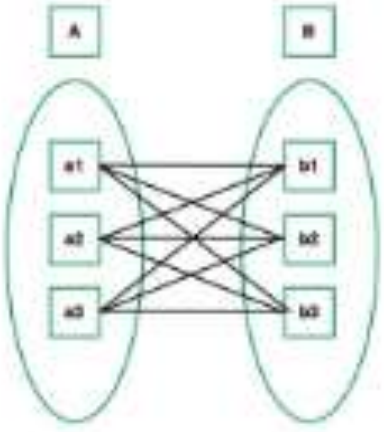
Many-to-one relationship

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.



Many-to-many relationship

An entity set A is associated with any number of entities in B with a possibility of zero and an entity in B is associated with any number of entities in A with a possibility of zero.



2. Participation Constraints

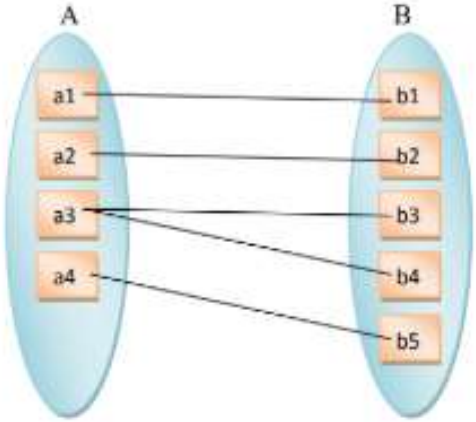
- It refer to rules that ***determine the minimum and maximum participation*** of entities or relationships in a given relationship set.
- It specifies whether the existence of an entity depends on being related to another entity through relationship types
- These constraints defines max and min number of relationship instance that each entity can participate in
    - **Maximum cardinality** : Maximum no. of times an entity can participate in a relationship
    - **Minimum Cardinality :** Minimum no. of times an entity can participate in a relationship

- For example, in a College Database, partial participation would permit courses with no enrolled students, while entire participation might require all students to be enrolled in at least one course.

- There are two types of participation constraints :
    - Total Participation
    - Partial Participation

Total Participation

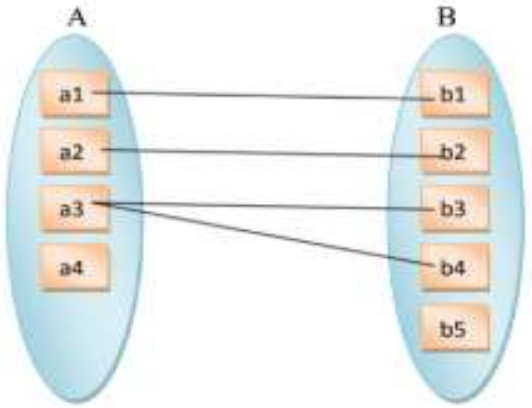
- The participation of an entity set E in a relationship set R is said to be total , ***if every entity in E participates in at least one relationship in R***
- Total participation is also called ***existence dependency***
- In ER diagrams, total participation (or existence dependency) is displayed as a double line connecting the participating entity type to the relationship
- Eg : In a university database, for instance, total participation between courses and students indicates that each student is required to be registered in a minimum of one course. It follows that no student can be excluded from a course.
- In below digram, the participation of entity set A in the relationship set is total because every entity and the participation of entity set B in the relationship set is also total because every entity of B also participates in the relationship set.



Partial Participation

- In database design, partial participation also known as ***optional participation***
- ***If only some of the entities in E participate in relationship R***, then the participation of E in R is said to be partial participation
- In ER diagrams, partial participation is represented by a single line

- Consider a database at a university, for instance, Partial participation can mean that some students are enrolled in classes but not all students are registered in them. While some objects are connected to one another, others may stand alone.
- In below diagram, The participation of entity set A in the relationship set is partial because only some entities of A participate in the relationship set, while the participation of entity set B in the relationship set is total because every entity of B participates in the relationship set.



- ***Total Participation which is represented by double lines***
- ***Partial Participation which is represented by single lines***

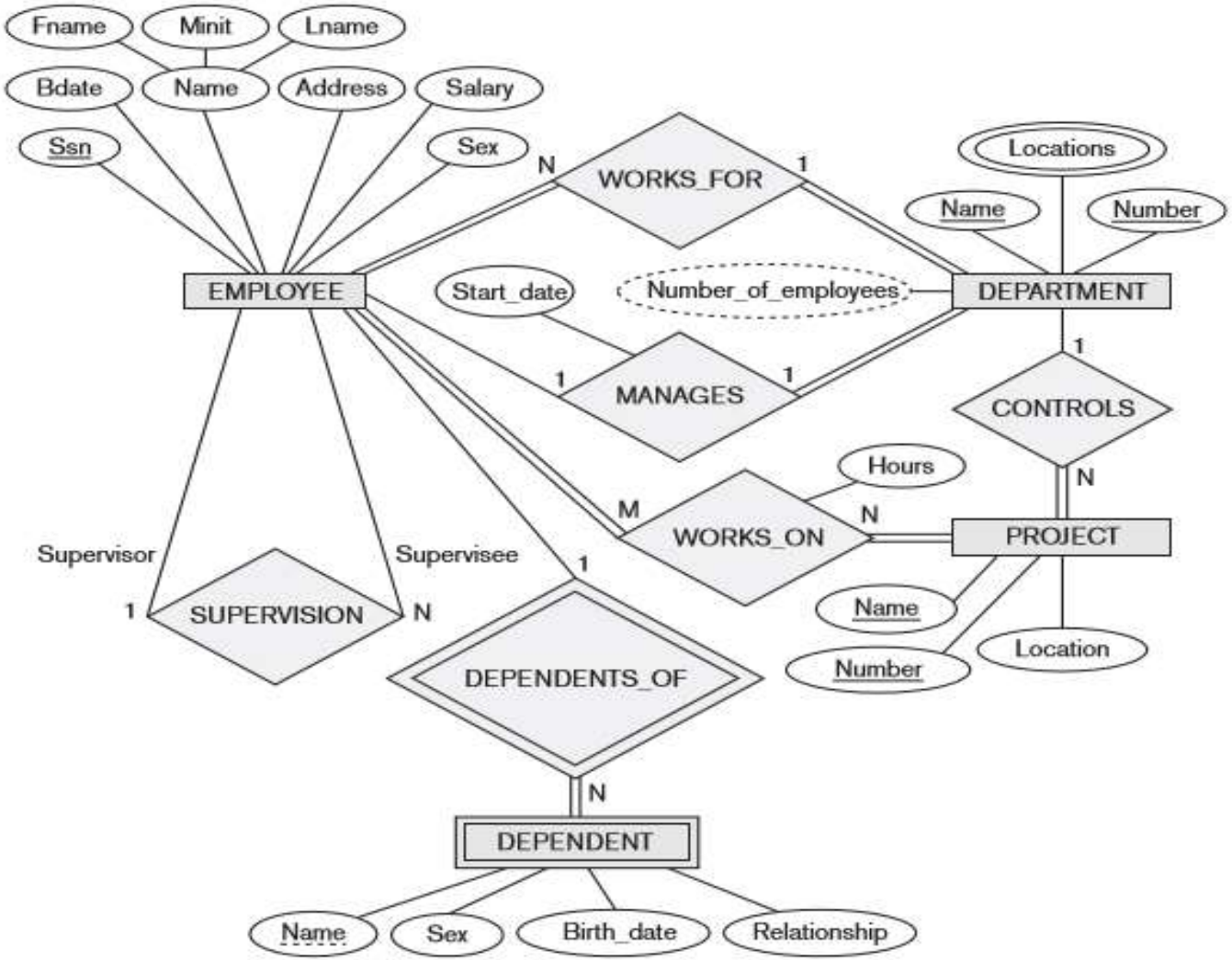
Suppose an entity set Student related to an entity set Course through Enrolled relationship set.

- The participation of entity set course in enrolled relationship set is **partial** because a course may or may not have students enrolled in. It is possible that only some of the course entities are related to the student entity set through the enrolled relationship set.

- The participation of entity set student in enrolled relationship set is **total** because every student is expect to relate at least one course through the enrolled relationship set.

Participation in Enrolled relationship set: **Partial** Course
Total Student

Sample ER Schema Diagram for COMPANY database



In our example, we specify the following relationship types:

1. MANAGES, a 1:1 relationship type between EMPLOYEE and DEPARTMENT.
   EMPLOYEE participation is partial

2. WORKS_FOR, a 1:N relationship type between DEPARTMENT and EMPLOYEE.
   Both participations are total.

3. CONTROLS, a 1:N relationship type between DEPARTMENT and PROJECT.
   The participation of PROJECT is total, whereas that of DEPARTMENT is determined to be partial

4. SUPERVISION, a 1:N relationship type between EMPLOYEE (in the supervisor role) and EMPLOYEE (in the supervisee role).
   Both participations are determined to be partial

5. WORKS_ON, determined to be an M:N relationship type with attribute Hours
   Both participations are determined to be total

6. DEPENDENTS_OF, a 1:N relationship type between EMPLOYEE and 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.

RELATIONSHIP OF DEGREE 3

• A relationship of degree 3 is also known as a **ternary relationship**, which is when **three entities are associated with each other.**

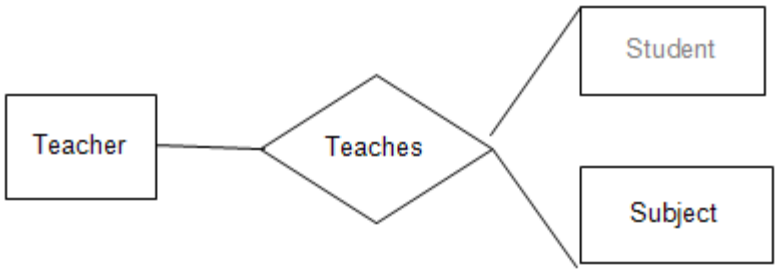
• We define a ternary relationship among three entities only when the concept cannot be represented by several binary relationships among those entities.

• The **degree of a relationship** refers to the number of entity types involved in it.

ie, Relationship Degree = 3

• **For Example,**

In a data model for an educational institution, we could define a single relationship that associates the entities Subject , Teacher , and Student .

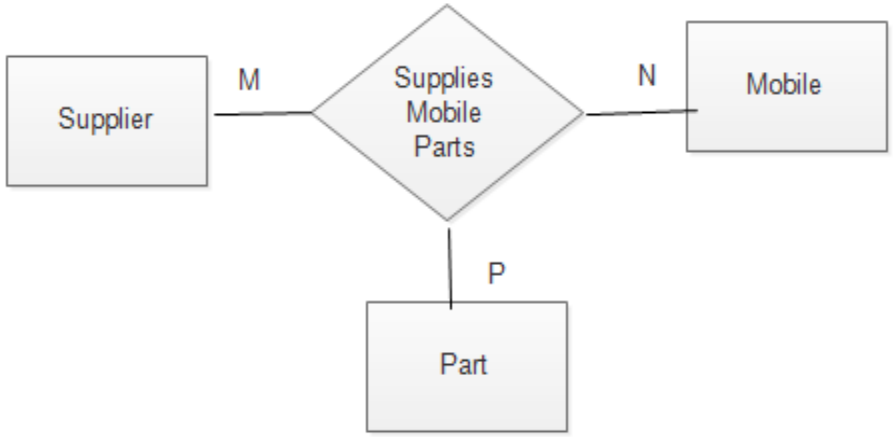


- For Example : Consider a Mobile manufacture company.

  Three different entities involved:

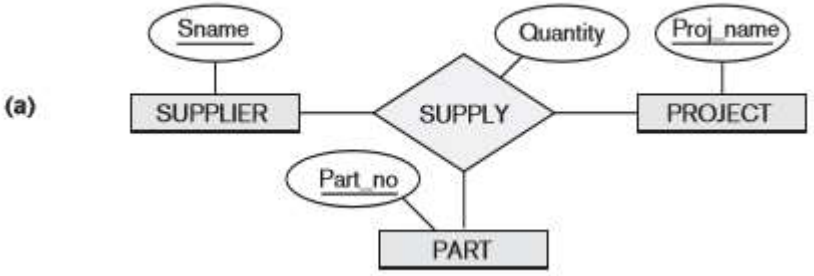
  - Mobile - Manufactured by company.
  - Part - Mobile Part which company get from Supplier.
  - Supplier - Supplier supplies Mobile parts to Company.

- Mobile, Part and Supplier will participate simultaneously in a relationship, because of this fact when we consider cardinality we need to consider it in the context of two entities simultaneously relative to third entity.



Choosing between Binary and Ternary (or Higher-Degree) Relationships

- Figure(a), which displays the schema for the SUPPLY relationship type that was displayed at the entity setor relationship set
- The relationship set of ***SUPPLY is a set of relationship instances (s, j, p),*** where ***s is a SUPPLIER*** who is currently supplying a ***PART p*** to a ***PROJECT j***.



- Figure(b) shows an ER diagram for three binary relationship types CAN_SUPPLY, USES, and SUPPLIES.

- Suppose that CAN_SUPPLY, between SUPPLIER and PART, includes an instance (s, p) whenever supplier s can supply part p

- USES, between PROJECT and PART, includes an instance (j, p) whenever project j uses part p, and SUPPLIES, between SUPPLIER and PROJECT, includes an instance (s, j) whenever supplier s supplies some part to project j.

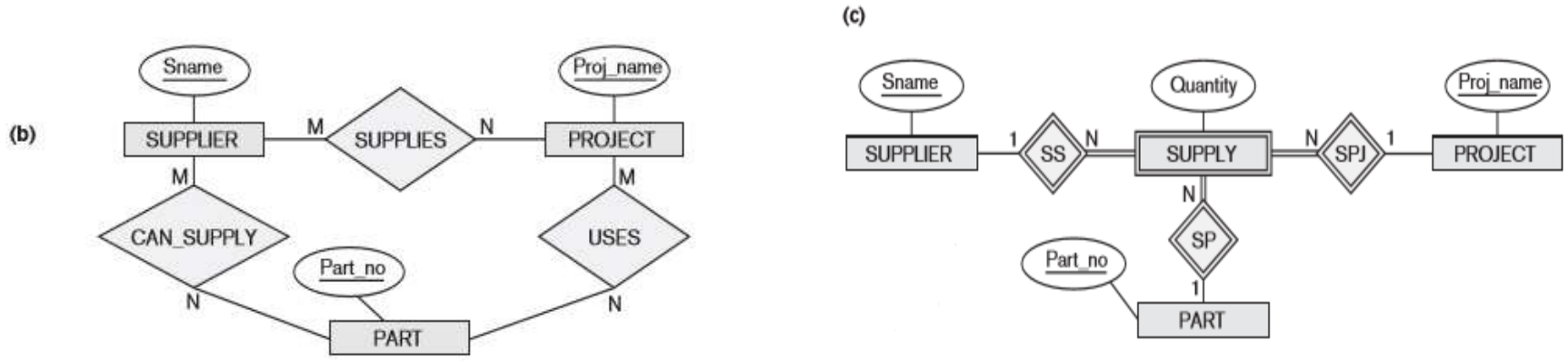
- The existence of three relationship instances (s,p),(j,p),and (s,j) in CAN_SUPPLY, USES, and SUPPLIES, respectively, does not necessarily imply that an instance (s, j, p) exists in the ternary relationship SUPPLY, because the meaning is different.

- Some database design tools are based on variations of the ER model that permit only binary relationships.

- In this case, a ternary relationship such as SUPPLY must be represented as a weak entity type, with no partial key and with three identifying relationships.

- The three participating entity types SUPPLIER, PART, and PROJECT are together the owner entity types in Figure(c).

- Hence, an entity in the weak entity type SUPPLY in Figure 7.17(c) is identified by the combination of its three owner entities from SUPPLIER, PART, and PROJECT.



Cardinality in Ternary Relationship

- In a ternary relationship, "**cardinality**" **refers to the specific number of instances of one entity that can be associated with a given combination of instances from the other two entities involved in the relationship**
- ie, essentially defining how many "links" can exist between each pair of entities within the three-way connection
- It can be expressed as a combination of
    - one-to-one
    - one-to-many
    - many-to-many

- Often represented as a notation like "**M:N:P**" where M, N, and P represent the potential multiplicity of each entity involved.
- The cardinality constraint of an entity in a ternary relationship is defined by a pair of two entity instances associated with the other single entity instance.

Say for a given instance of Supplier and an Instance of Part, can that supplier supply that particular part for multiple Mobile models.

Example − Consider a Supplier S1 that supplies a Processor P1 to the company and the uses the Processor P1 supplied by Supplier S1 in its multiple Models in that case the cardinality of Mobile relative to Supplier and Part is N (many).

In case of Supplier's cardinality we can say for a given instance of Mobile one of its Part can be supplied by multiple Suppliers.

Example − Consider a Mobile M1 that has a Part P1 and it is being supplied by multiple Suppliers in that case the cardinality of Supplier relative to Mobile and Part is M (many).

Similarly, for a given instance of Supplier and an instance for Mobile does the Supplier supply multiple Parts.

Example − Consider a Supplier S1 supplying parts for Mobile M1 like screen, Processor etc. in that case the cardinality of Part relative to Supplier and Mobile is P (many).