DATABASE MANAGEMENT SYSTEMS
UNIT-I

Introduction:

Data - Data is meaningful known raw facts that can be processed and stored as information.

Database - Database is a collection of interrelated and organized data. In general, it is a collection of files (tables).

Why Database:

Conventionally, in an information system, the information is obtained by developing the systems and integrating them. This calls for breaking the system into various subsystems and developing the information systems independently. In this approach, each system will have its master files and transaction files. They have to be processed separately at different times, depending upon the needs and schedules. The file layouts and the access methods could be different in different systems. Therefore, the files will be updated at different times. This approach does affect the quality of the information across all the systems due to various reasons.

The data in many systems are common, and there is repetition of data storage in various systems. This called data redundancy. The redundancy of data gives rise to problems of keeping the data current and same in all the files. The data management is complex in such a situation. The reports generated out of such files show discrepancies in the information. Since the data files are different for different systems, data sharing is not possible. These files need to be created at different times. Transaction updating is also carried out at different times. It requires the increase of a magnetic media for storage because the systems are developed independently. The redundancy causes lack of integrity and inconsistency of the data in the various files.

DBMS - Database Management System (DBMS) is a collection of interrelated data [usually called database] and a set of programs to access, update and manage those data [which form part of management system]

OR

It is a software package to facilitate creation and maintenance of computerized database.

- Database management system is designed to manage large bodies of information.
- The DBMS is a general purpose software system that facilitates the process of defining constructing and manipulating database.
- Defining database: specifying the datatypes, structures
- Constructing database: process of storing the data on some storage medium.
- Manipulating database: using queries to retrieve the specific data, updating database.
- Another important functions provides by the DBMS includes protecting database.
- Protection includes system protection against hardware and software malfunction and security protection against unauthorized access

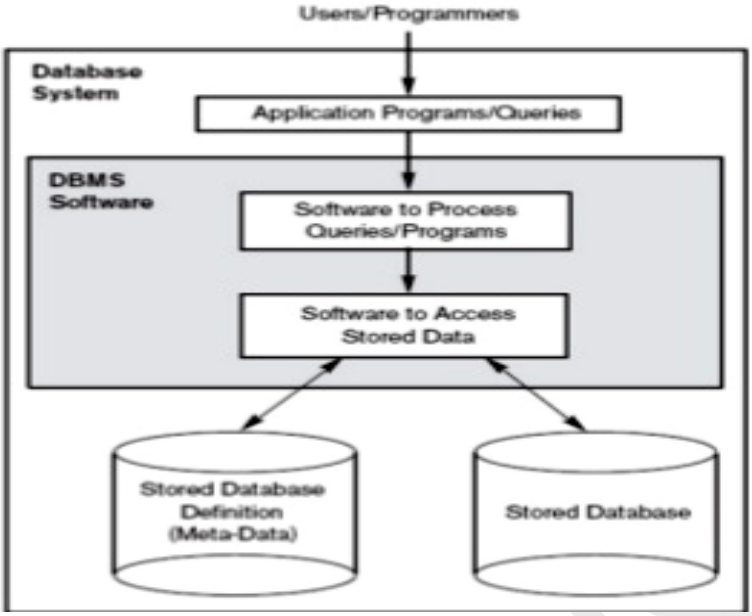
A Database Management System (DBMS) is a set of programs that manages any number of databases.

A DBMS is responsible for:

- accessing data
- inserting, updating, and deleting data
- security
- integrity, facilitated by:
  - locking
  - logging
  - application-defined rules, including triggers
- supporting batch and on-line programs
- facilitating backups and recoveries
- optimizing performance
- maximizing availability

- maintaining the catalog and directory of database objects
- managing the buffer pools
- acting as an interface to other systems programs
- supporting user interface packages, such as the popular SQL interface for relational database systems

Database System Environment is as follows:



CHARACTERSTICS OF THE DATABASE APPROACH:

In file processing, each user implements the files needed for a specific application.

Ex: A bank officer wants to collect information about customers in a particular postal code area. A new program must be implemented for this.

- But in the database approach, a single database is maintained that is defined once and then access by various users.
- Here data can be accessed by queries.

The characteristics are:

- Self describing nature of a database system.
- Insulation between programs and data.
- Easy operation implementation
- Support multiple views of data.
- Sharing of data.

Self describing nature of a database system:

- The database system contains not only the database itself but also contains a complete definition or description of database structure. This definition is stored in DBMS catalog, which contains information such as the structure of each file , the data type, storage format. The information stored in the catalog is called metadata.
- The catalog is used by the DBMS software , database users who need information about the database structure.

Insulation between programs and data:

In file processing, the structure of data files is embedded in application programs.

➔ But in database approach, the structure of data files is stored in the DBMS catalog separately from the access programs. We call this property program-data independence.

Easy Operation implementation:

All the operations like insert, delete, update, search etc. are carried out in a flexible and easy way. Database makes it very simple to implement these operations. A user with little knowledge can perform these operations. This characteristic of database makes it more powerful.

Support of multiple views of data:

Database can be accessed by many users. Different users require different view of the database. A view is a subset of the database.

Sharing of data: DBMS allow multiple users to access the database at the same time. DBMS must include the concurrency control software because several users try to update the same data at a time.

Ex: When several reservation clerks try to assign a seat on an airline flight, the DBMS should ensure that each seat can be accessed by only one clerk at a time for assignment to a passenger.

Data Administrator and Database Administrator:

DA (Data Administrator) and DBA (Database Administrator) both are responsible for managing database for an organization. They differ from each other in their required skills and responsibilities.

Data Administrator (DA) :

"Person in the organization who controls the data of the database refers data administrator."

DA determines what data to be stored in database based on requirement of the organization.

DA works on such as requirements gathering, analysis, and design phases.

DA does not to be a technical person, any kind of knowledge about database technology can be more beneficiary

DA is some senior level person in the organization. in short, DA is a business focused person but should understand about the database technology.

Database Administrator (DBA) :

"Person in the organization who controls the design and the use of the database refers database administrator."

DBA provides necessary technical support for implementing a database.

DBA works on such as design, development , testing, and operational phases.

DBA is a technical person having knowledge of database technology.

DBA does not need to be a business person. in short, DBA is a technically focused person but should understand about the business to administrator the database effectively.

ACTORS ON THE SCENE:

In large organizations many people are involves in the design, use and maintenance of a large database.

- People who are involved in the design and use of database are called actors on the scene.
- People who work to maintain database environment are called workers behind the scene.
- Both actors on the scene and workers behind the scene are collectively called DBMS users.

1) **Database administrator**:

DATABASE USERS AND ADMINISTRATOR:

A Primary goal of a database system is to retrieve information from and store new information in the database. People who work with a database can be categorized as database users or database administrators.

Database Users: The users of a database system can be classified into following groups, depending on their degree of expertise or the mode of their interactions with the DBMS. The users can be,

1. Naïve Users. 2. Online Users. 3. Application programmers.
4. Sophisticated users. 5. Specialized users.

Naïve Users: Naïve users are those users who need not be aware of the presence of the database system or any other system supporting their usage. A user of an automatic teller machine falls in this category. The user is instructed through each step of a transaction. He or she then responds by pressing a coded key or entering a numeric value.

The operations that can be performed by naïve users are very limited and affect only a precise portion of the database. For example, in the case of the user of the automatic teller machines, user's action affects only one or more of his own accounts.

Online Users: Online users are those who may communicate with the database directly via an online terminal or indirectly via a user interface and application program. These users are aware of the presence of the database system and may have acquired a certain amount of expertise within the limited interaction they are permitted with a database.

Application Programmers: Professional programmers or application programmers are those who are responsible for developing application programs or user interface. The application programs could be written in a general purpose programming language or the commands available to manipulate a database.

Sophisticate Users: Sophisticated users interact with the system without writing programs. Instead, they form their requests in a database query language. They submit each such query to a query processor, whose function is to break down DML statements into instructions that the storage manager understands. Analysis who submit queries to explore data in the database fall in this category.

Specialized Users: Specialized users are sophisticated users who write specialized databases applications that do not fit into the traditional data processing framework. Among these applications are computer-added design systems, knowledge-based and expert system, systems that store data with complex data types and environment modeling system.

Database Administrator: The people who have central control of both data and the programs that access those data in the database is called the Database Administrator (DBA). Functions of the DBA are,

1. **Schema Definition:** The DBA creates the original database schema by writing a set of definitions that are translated by a special language called a Data Definition Language (DDL). The result of compilation of DDL statements is a set of tables of databases that is stored permanently in a special file called data dictionary or dta directory which contains metadata that is, data bout data.
2. **Storage Structure and Access Methods:** The DBA create storage structure and access methods by writing a set of definitions, which are translated by the data definition language compiler.
3. **Schema and Physical Organization Modification:** The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.
4. **Granting of Authorization for Data Access:** The DBA can grants permissions for accessing different parts of the database to various users.
5. **Integrity-constraint Specification:** The data values stored in the database must satisfy a consistent constraint (condition) specified only by DBA, known as integrity constraint. So that, the data values stored in the database must be according to integrity constraints.
6. **Routine Maintenance:**
   - **i)** DBA periodically updates the database, to prevent loss of data in case of disasters.
   - **ii)** DBA ensures that there is enough free space available for doing normal operations.
   - **iii)** DBA monitors all the jobs running on the database and ensures that performance is not degraded by very expensive tasks submitted by some users.

ADVANTAGES OF DATABASE MANAGEMENT SYSTEM

1. **Reduction of Redundancies**
   - i) Redundancy means duplication (making the same copy again). Reduction of redundancy means avoiding duplication of data and reducing the total amount of storage space required.
   - ii) It also reduces the extra processing time to search the required data in a large mass of data.

iii) Another advantage of avoiding duplication is the elimination of the inconsistencies (difficult) in searching the exact data file required.

2. **Data Independence and Efficient Access**
   i) Data base programs in the data base are independent of their storage details.
   ii) The conceptual schema provides physical storage details and external schema provides logical storage details i.e. the conceptual schema provides independence from external schema. It means physical storage details are independent from logical storage details.
   iii) DBMS strongly provides the efficient access, retrieval of the stored information, including support for very large files, index structures in query optimization.
3. **Data Integrity**
   i) Data integrity means that the data values entered in the database must be checked to ensure that they fall within a specified range are of correct format.
   For example the value for the age of an employee must be in the range of 16 and 55.
   ii) Date integrity also checks that if we are referring any field, then that field must exist.
   For example a user is not allowed to transfer funds from a existing savings account to an non-existent savings account.
4. **Data Security**
   i) Data is of vital importance to an organization and must be confidential. Such confidential data must be secured strongly and may not be by any unauthorised person.
   ii) DBA (Database administrator) should ensure that proper and different access permissions are given to different types of users.
   For example a manager can access the salary details of employees in his department only.
5. **Reduced Application Development Time**
   i) Since the DBMS provides several pre-defined functions like concurrency control, crash recovery, high level query facilities etc. Only application specified code needs to be written by the users.
6. **Data Administration:**
   i) DBMS facilitates maintenance and administration of date by providing a common base for a large collection of a data that is shared by several users.
   ii)In addition, the DBA ensures the fine-tuning of data representation, periodic back-ups, ensures proper permissions of data access, monitoring all jobs etc.
7. **Concurrent Access:**
   i) Many uses access a single program concurrently (at the same time) as if their programs were running in isolation.
   ii) The DBMS executes the actions of the program in such a way that the concurrent access is permitted, but the conflicting operations are not permitted to proceed concurrently.
8. **Crash Recovery:**
   i) The DBMS maintains, continuous log (record) of the changes made to the data , so that, if there is any system crash by power failure, it can restore the database to a previously stored consistent state.
   ii) That is the actions of incomplete transactions are undone, so that the database stores only the actions of complete transactions after recovery from a crash.

DISADVANTAGES OF DATABASE MANAGEMENT SYSTEM:

There are some disadvantages in DBMS. In some situations they may create unnecessary problems. At that time, file processing is the best option. The disadvantages of DBMS are,

1) Conversion costs: The older systems in an organization are based on file processing and/or older database technology. The converting the older systems to modern database technology is measured in terms of dollars, time and organizational commitment and may often seems prohibitive to an organization.

2) Installation and Management Cost:

→ A multiuser database management system is a large and complex software that has a high initial cost, requires a staff of trained personnel to install and operate, annual maintenance etc.

→ Installing such a system may also require upgrades to the hardware and data communication system in the organization.

→ Substantial training is required at ongoing basis to keep up with new releases and upgrades.

3) New Specialized personnel:

→ Organizations that adopt the database approach need to hire or train individuals to design and implement database, provide database administration services and manage a staff or new people.

→ Further, because of the rapid changes in technology, these new people will have to be retrained or upgraded on a regular basis.

4)Need for explicit Backup and Recovery*:*

A shared database must be accurate and available at all times. This requires the procedures that should be developed and used for providing backup copies of data and for restoring a database when damage occurs.

5) Security Breaches: Centralization also means that the data is accessible from a single source, namely the database. This increases the severity of security breaches (breaking) and disrupting the operation of the organization.

DATABASE SYSTEM APPLICATIONS:

Database management system are used by many individuals either directly or indirectly. Some of the applications of DBMS are listed below.

1) **Transactions in Bank**: The user accesses the bank database for crediting or debiting the account. The bank database stores the details of individuals customers, their account, loans etc.
2) **Ticket Reservation*:*** For schedule information and for reserving the ticket, we access the database of airlines. The database should be reliable.
3) **Students of Universities*:*** Universities have database of various courses they offer and a database for the faculty and students. Each student records contains the name of the student, marks scored etc. Each faculty record contains the name of the faculty, his employee id, salary, subjects dealing with etc.
4) **Internet Interactions*:*** Internet applications are mainly database driven. Suppose you want to send an e-card to your friend. You first interact with the database based on purpose of the e-card. If you want to send a birthday card, you select from categories and retrieve various cards from the database.

   Online shopping is another example. You select an item from the database and then send your credit card number with some additional information to buy the product.
5) **Computer Horoscope*:*** We enter our name, date birth, our likes and dislikes in a form presented to us. All these details are compared against a database to search for a perfect match and our future is predicted.
6) **Data base in a library:** This database stores details of the books available. The user access this database to find a book quickly. The database allows easy management by allowing a user to reserve a book and intimate him through a mail when the book is available. The system also sends a remainder to customers who did not return the book by due-date. This system uses a bar code reader to provide access to database.
7) **Shopping of Supermarket*:*** The cashier places the bar code of a product against a bar code reader. The application program uses the bar code to identify the price and reduce the number of items on the database.

   The user can also call the super market to know the availability of the product. The telecaller accesses the database to know if the product is available or not.
8) **Telecommunications:** Database is used to store the details of number of calls made to generate the bill. For prepaid customers, it stores the available credit.

VIEW OF DATA:

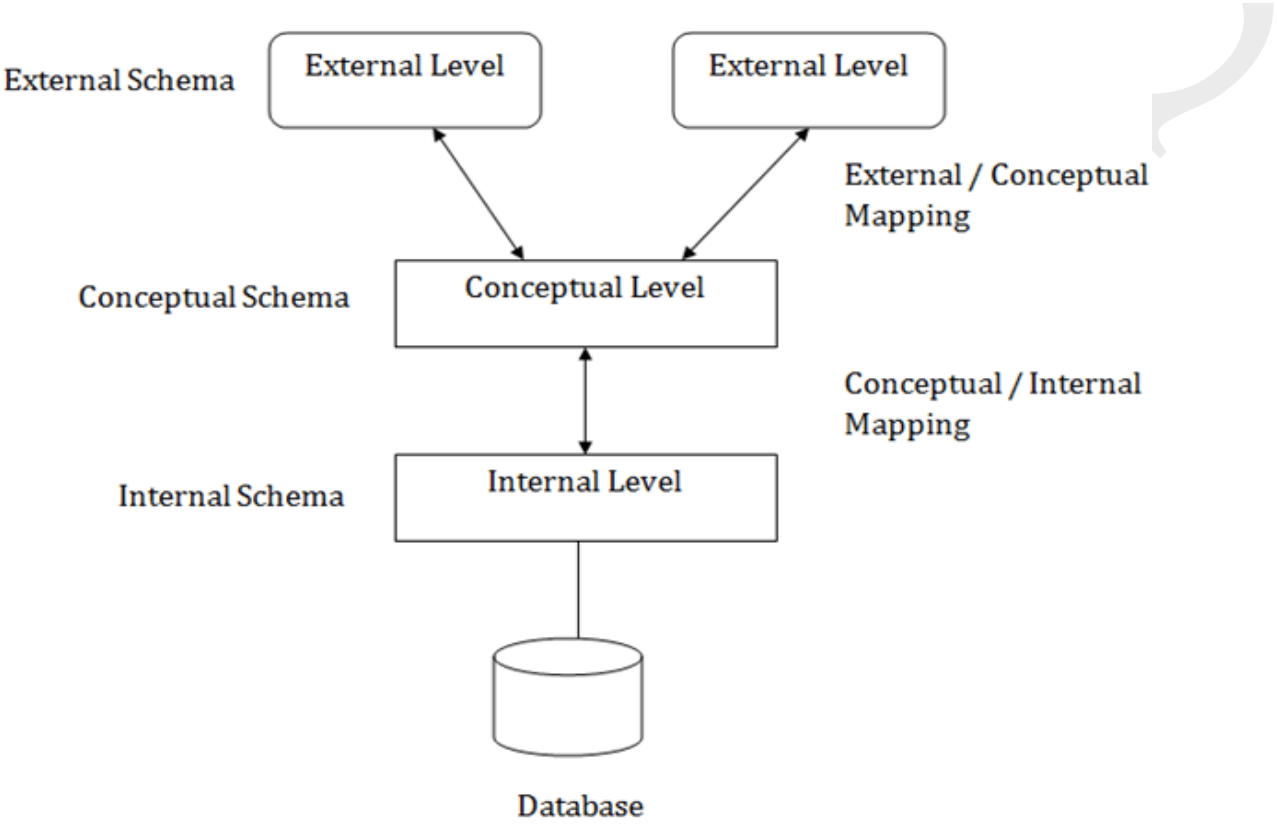
Database is a collection of large volumes of data. A user will not always require the complete data. The responsibility of the database system is to provide only the data that is required to the user.

Similarly, a programmer who wishes to enhance the features of database system is not concerned about the data, but need to know how the data is stored. Hence different users need different view of data.

THREE SCHEMA ARCHITECTURE

- The three schema architecture is also called ANSI/SPARC architecture or three-level architecture.
- This framework is used to describe the structure of a specific database system.
- The three schema architecture is also used to separate the user applications and physical database.
- The three schema architecture contains three-levels. It breaks the database down into three different categories.

The three-schema architecture is as follows:



In the above diagram:

- It shows the DBMS architecture.
- Mapping is used to transform the request and response between various database levels of architecture.
- Mapping is not good for small DBMS because it takes more time.
- In External / Conceptual mapping, it is necessary to transform the request from external level to conceptual schema.
- In Conceptual / Internal mapping, DBMS transform the request from the conceptual to internal level.

1. Internal Level

- The internal level has an internal schema which describes the physical storage structure of the database.
- The internal schema is also known as a physical schema.

- It uses the physical data model. It is used to define that how the data will be stored in a block.
- The physical level is used to describe complex low-level data structures in detail.

2. Conceptual Level

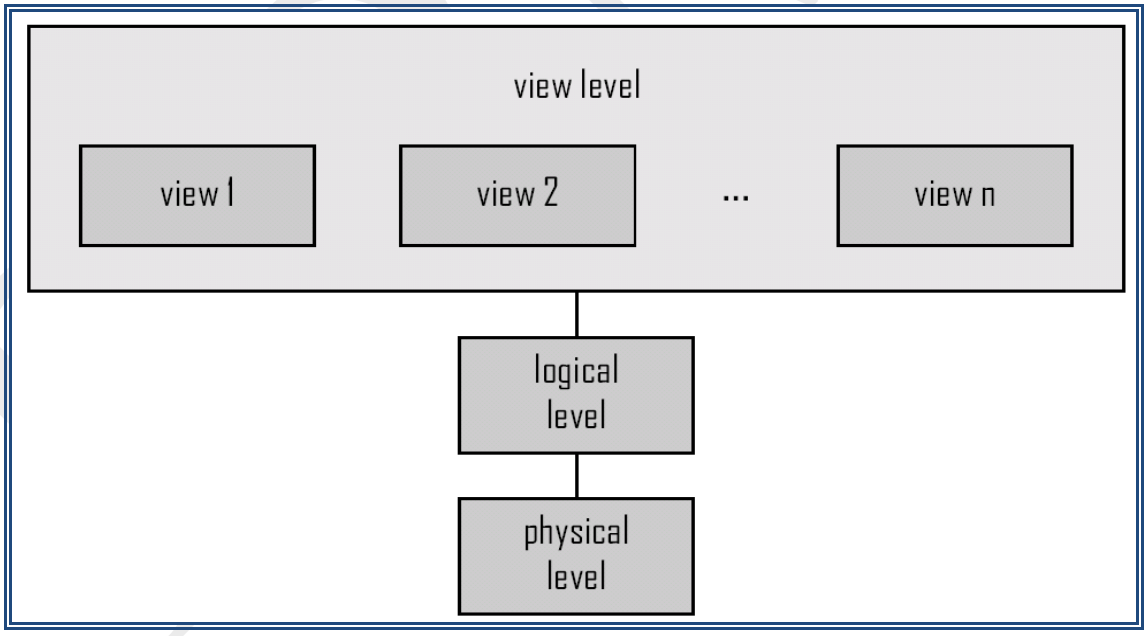
- The conceptual schema describes the design of a database at the conceptual level. Conceptual level is also known as logical level.
- The conceptual schema describes the structure of the whole database.
- The conceptual level describes what data are to be stored in the database and also describes what relationship exists among those data.
- In the conceptual level, internal details such as an implementation of the data structure are hidden.
- Programmers and database administrators work at this level.

3. External Level

- At the external level, a database contains several schemas that sometimes called as subschema. The subschema is used to describe the different view of the database.
- An external schema is also known as view schema.
- Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group.
- The view schema describes the end user interaction with database systems.

LEVELS OF ABSTRACTION IN A DBMS:

In DBMS, the data can be abstracted in three levels. The goal of the three level abstraction in a DBMS is to separate the user's request and the physical storage of data in database. The data abstracted at each of these levels is described by a "Schema". A schema is a systematic plan for attaining some goal. For example, the storage of data in a database is the concept of data storage. This concept is called schema.



1.Physical level: The design of a database at physical level is called **physical schema**.

Features:

a) Lowest level of abstraction.

b) It describes how data are actually stored.

c) It describes low-level complex data structures in detail.

d) At this level, efficient algorithms to access data are defined.

2. **Logical level**: Design of database at logical level is called **logical schema**

It describes what data stored in database, and the relationships among the data

Features:
a) It is next-higher level of abstraction. Here whole Database is divided into small simple structures.
b) Users at this level need not be aware of the physical-level complexity used to implement the simple structures.
c) Here the aim is ease of use.
d) Generally, database administrators (DBAs) work at logical level of abstraction.

3**. View level**: Design of database at view level is called **view schema**. This generally describes end user interaction with database systems
Application programs hide details of data types. Views can also hide information (e.g., salary) for security purposes.
Features:
a) It is the highest level of abstraction.
b) It describes only a part of the whole Database for particular group of users.
c) This view hides all complexity.
d) It exists only to simplify user interaction with system.
e) The system may provide many views for the whole system.

Instances and Schemas: Databases change from time to time as information is inserted and deleted. The view of collection information stored in the database at that particular moment is called an "instance" of the database.

Design of a database is called the schema or The overall design (plan) of the database to store the information is called as database "**Schema**".

For example, declaration of variable is a plan to store the data is known as "**Schema**". Assign the value into a variable is nothing but "instance".

DATA MODELS:
Data model means to model the data i.e., to give a shape to the data and to give a figure to the stored data. A data model makes it easier to understand the meaning of the data by its figure.

The data model can be used to convey the designers understanding of the information requirement of the organization. Increasingly, organizations are standardizing the way that they model data by selecting a particular approach to data modeling and using it throughout their database development projects.

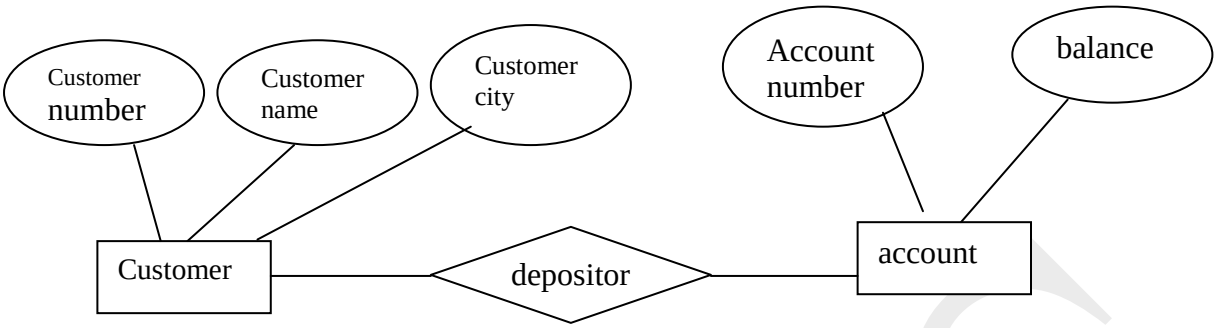
A data model has a collection of tools for describing Data, Data relationships, Data semantics, and Data constraints.
→ The data models are divided into three different groups.

1. Object-based logical models.
2. Record based logical models.
3. Physical data model.

Object-Based Logical Models:
Object-based logical models are used in describing data at logical level and view level. Logical level and view level are used to retrieve the data. Logical level describes what data are stored in the database and what relationships exist among those data. Logical level is used by DBA, who must decide what information is to be kept in the database. View level describes only part of the entire database to be viewed by the user of the database hiding the details of ht information stored.
Object-based logical models are described in the different following models.

1) The Entity-relationship model.
2) The object-oriented logical model
3) The semantic data model.
4) The functional data model,

Entity-relationship model: An entity is a "thing" or "object" in the real world that is distinguishable from other objects. The Entity-Relationship model is based on a collection of basic objects called entities and the relationship among these objects. Consider the E-R diagram as shown in fig.



i) Rectangle represents entities.
ii) Diamonds represent relationship among entities.
iii) Ellipse represents attributes.
iv) Lines represents link of attributes to entities and entities to relationship.

Here customer and account are two different entities such "things" or "objects" and there is one relationship between these two entities i.e. depositor. It means customer deposits the account. When customer deposits account, the customer should specify the customer_name, customer_number customer_city known as attributes. Likewise, account should specify the attributes account_number and balance.

The Object-oriented Model: Like the ER model, the object-oriented model is based on a collection of objects. An object contains values stored in instance variables, within the object also contains bodies of code that operates on the objects. These bodies of code are called as methods.

Objects that contain the same types of values and the same methods are grouped together into classes. A class may be viewed as a definition for objects. This combination data and methods comprising a definition is similar to a programming language abstract data type.

The only way in which one object can access the data of another object is by invoking a method of that other object. This action is called sending a message to the object.

The semantic Data Model: A semantic data model is a more high level data model that makes it easier for a user to give starting description of data in an enterprise. These models contain a wide starting description of data in an enterprise. These models contain a wide variety of relations that helps to describe a real application scenario. A DBMS cannot support all these relations directly, so it is built only with few relations known as relational model in DBMS. A widely used semantic data model is the Entity-Relationship (ER) data model which allows us to graphically denote entities and relationship between them.

The Functional Data Model: In conventional database systems, procedures, data structures and actual content are usually separated. Thus, a conventional Database Management systems (DBMS) provides users with a possibility to store, modify or retrieve data that structured in accordance with a current databases schema.

Record-Based Logical Models:

Record-Based logical models describes data at logical and view levels. When compared with object-based data models, the record-based logical model specifies the overall logical structure of the database and provides higher-level implementation.

Record-Based models are so named because the data is kept in the form of records of several types. Each record has fixed number of attributes and each field is of fixed length. The record-based models are of three types. They are,

1. Relational model 2. Network model. 3. Hierarchical model.

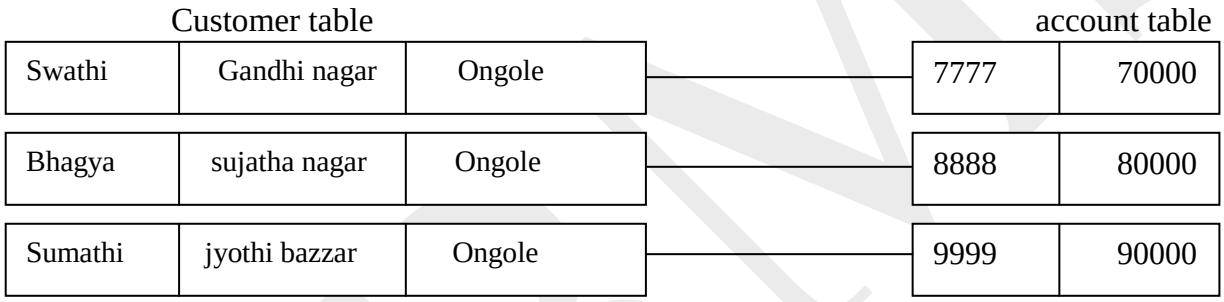
Relational model: The relational model represents both data and relationship among data in the form of tables. Each table has multiple columns and each column has a unique name. Consider the following relational model.

Customer Relation			
Customer_name	customer_street	Customer_city	Account_number
Chaitanya	Gandhi nagar	Ongole	7777
Bhagya	sujatha nagar	Ongole	8888
Sumathi	jyothi bazzar	Ongole	9999

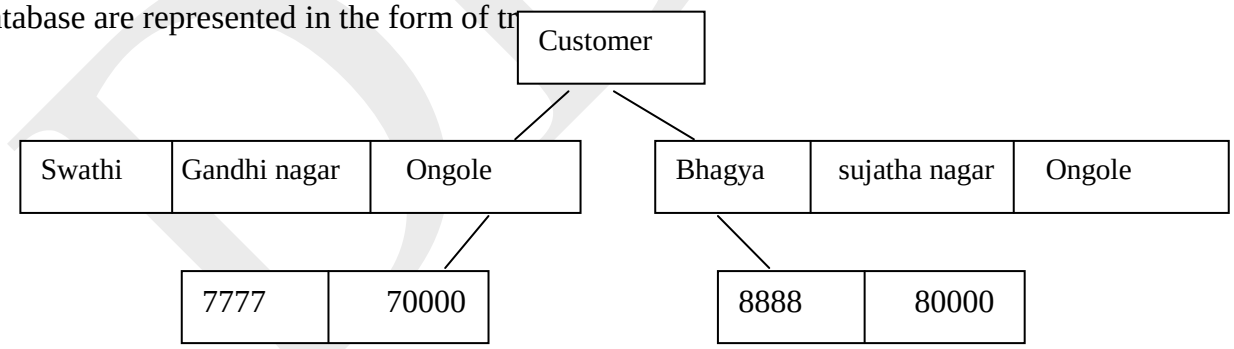
Account Relation	
Account_number	balance
7777	70000
8888	80000
9999	90000

The description of data in terms of tables are called as relations. From the above customer and account relations, we can make a condition that customer details are maintained in customer relation database and their deposit details are maintained in account relation database.

Network Model: Data in the network model are represented by collection of records and relationships among data are connected by links. The links can be viewed as pointers. The records in the database are represented in the form of graphs.



Hierarchical Model: Hierarchical model is same as the network model i.e. data in the hierarchical model are also represented by collection of records and relationships among data are connected by links. The links can be viewed as pointers. But, the difference from network model is that the records in the database are represented in the form of tr



The relational model differs from the network and hierarchical models. It means, the relational model does not use pointers or links.

Physical Data Models:

Physical data models are used to describe data at the lowest level, which explains how the data is actually stored using complex low-level data structures. Actually the physical data models are rarely used.

Two types of physical data models are,

1. Unifying model. 2. Frame-memory model.

DATA INDEPENDENCE:

The ability to modify a scheme definition in one level without affecting a scheme definition in a higher level is called data independence. 1. There are two kinds:

Logical data independence

- The ability to modify the conceptual scheme without causing application programs to be rewritten.
- Immunity of external schemas to changes in the conceptual schema.
- Usually done when logical structure of database is altered

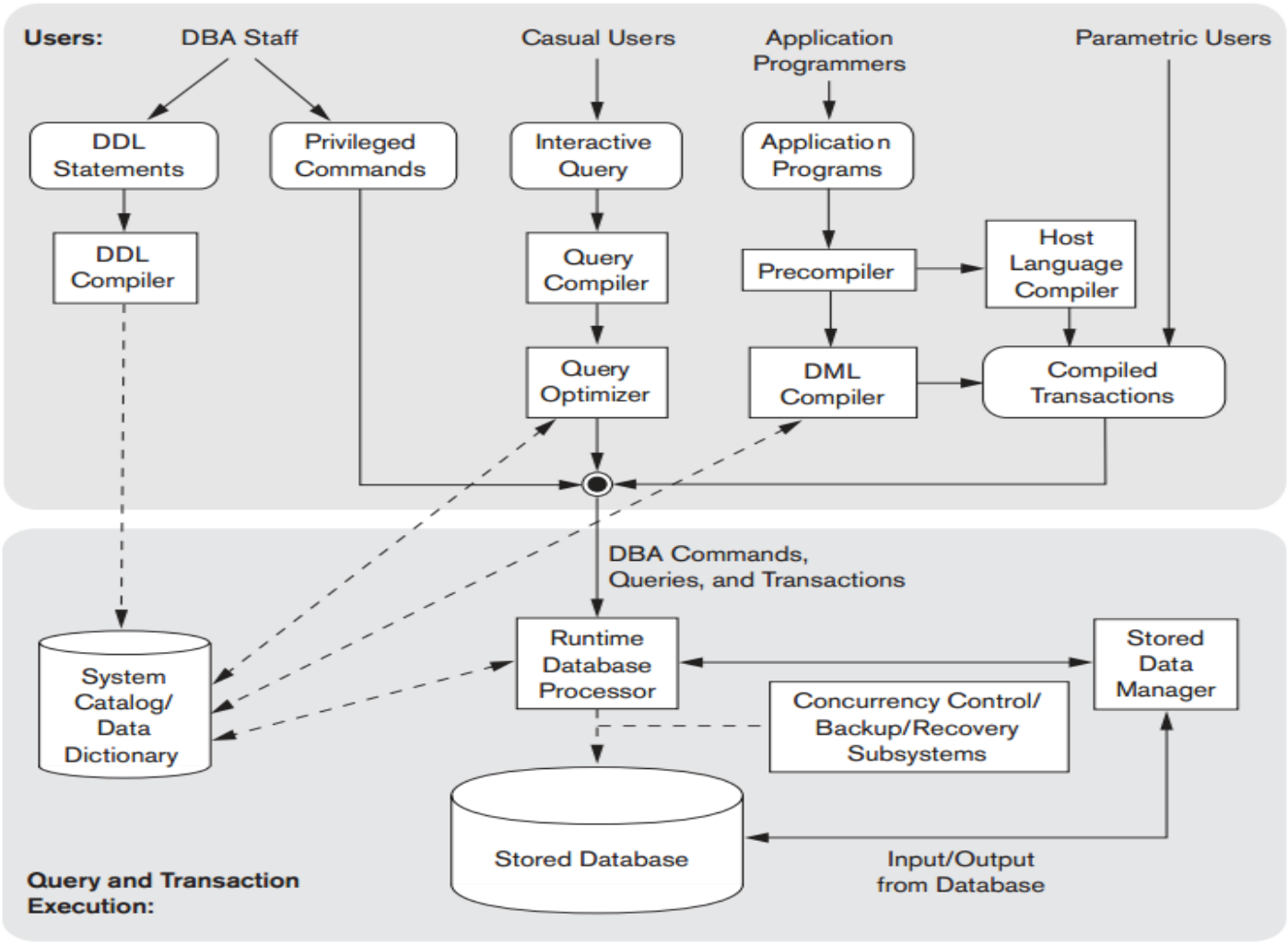
Physical data independence

- The ability to modify the internal scheme without having to change the conceptual or external schemas.
- Modifications at this level are usually to improve performance.

DATABASE SYSTEM ENVIRONMENT :

DBMS System Structure and its Components:

We can explain the overall structure of DBMS/System structure and its components by the diagram



Here we have diagrammatic representation of database system environment.

- The figure is divvied into two levels. The top half of the figure refers to the various users of the database environment and their interfaces.
- The lower half shows the storage of data and system catalogs.
- The database and the dbms catalogs are stored on disk.
- Access to the disk is controlled by the operating system (OS).
- Stored data managers controls access to DBMS information stored on disk.
- Consider the top half of the figures:

- It shows the interfaces from the DBA staff, casual users programmers who write programs using host languages.
- DBA staff workers on defining the database and making changes to it by using DDl and other privileged commands.
- Casual users work with interactive quires when they need information from database.
- These queries are analyzed by the query compiler and these compiled queries are given to the query optimizer, which rearrange the operations, eliminate redundancies.
- Application programmers write programs in host language such as C, JAVA and COBOL. These programs are submitted to precompile.
- The pre compiler get DML commands from an application programs and these commands are sent to the DML compiler for compilation.
- The remaining programs are sent to the host language compiler.
- In the lower half of figure, the runtime database processor is shown to execute.
- Privileged commands ii) executable queries
- It works with data dictionary.
- It works with stored managers for carrying I/O operations between disk and main memory.

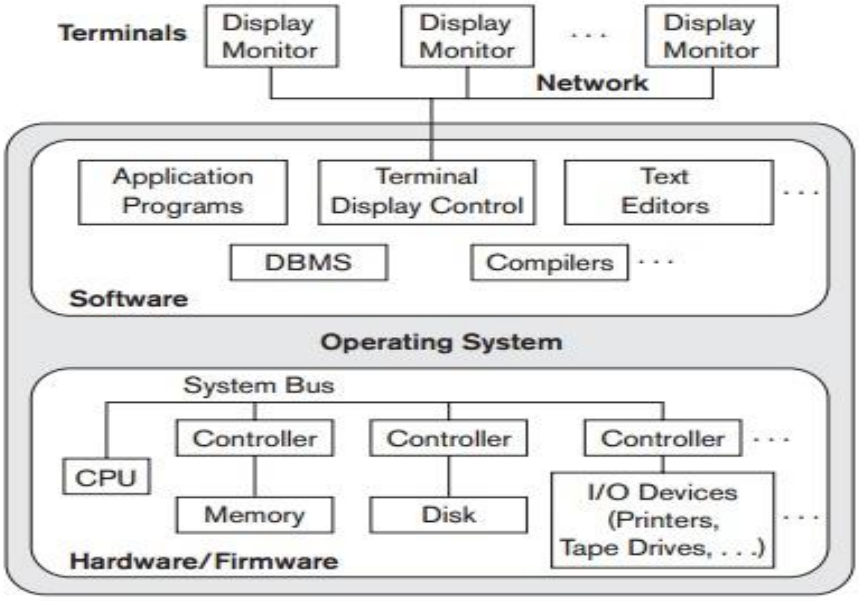
CENTRALIZED AND CLIENT/SERVER ARCHITECTURES FOR DBMS

Centralized DBMSs Architecture

. Earlier architectures used mainframe computers to provide the main processing for all system functions, including user application programs and user interface programs, as well as all the DBMS functionality.

- all processing was performed remotely on the computer system, and only display information and controls were sent from the computer to the display terminals, which were connected to the central computer via various types of communications networks.
- DBMS itself was still a centralized DBMS in which all the DBMS functionality, application program execution, and user inter-face processing were carried out on one machine

Basic Client/Server Architectures



The client/server architecture was developed to deal with computing environments in which a large number of PCs, workstations, file servers, printers, database servers, Web servers, e-mail servers, and other software and equipment are connected via a network

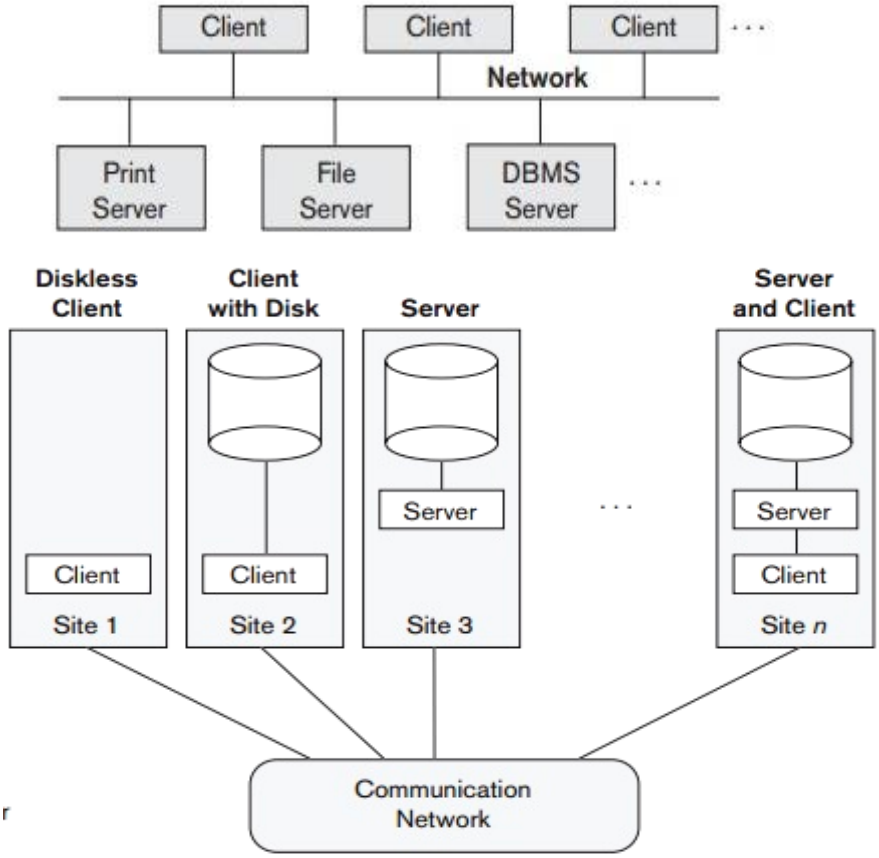
- File server that maintains the files of the client machines

- Printer server by being connected to various printers; all print requests by the clients are forwarded to this machine
- Web servers or e-mail servers also fall into the specialized server category. The resources provided by specialized servers can be accessed by many client machines
- A client makes a request to access database
- Server provides services to the client by receiving the request
- Server provides services such as file access, printing database access.
- Two main types of dbms architectures were
  i) Two-tier ii) three-tire

Two-Tier Client/Server Architectures for DBMS:

In this architecture the application programs were moved to the client side, query and transaction functionally remained on the server side.

- In this architecture the server is often called a query server or transaction server.
- The user interface programs and application programs can run on the client side.
- When DBMS access is required, the program establishes a connection to the DBMS (which is on the server side); once the connection is created, the client program can communicate with the DBMS
- A standard called Open Database Connectivity. allows client-side programs to call the DBMS

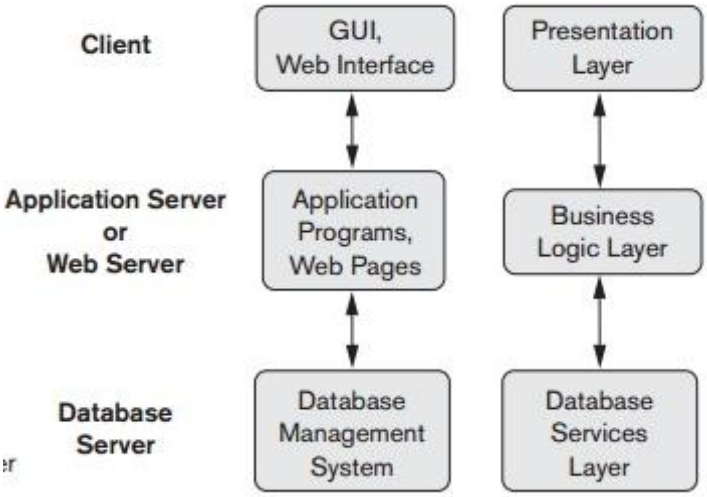


Three-Tier and n-Tier Architectures for Web Applications:

Many Web applications use an architecture called the three-tier architecture, which adds an intermediate layer between the client and the database server.

- This intermediate layer or middle tier is called the application server or the Web servers.
- This server plays an intermediary role by stroing rules that are used to access data from the database server

- The intermediate server accepts request from the client process the request and sends database commands to the database server.



ENTITY RELATIONSHIP MODEL, RELATIONAL MODEL

Database Design and ER Diagrams:

→ **Database Design**:

The database design process can be divided into six steps.

Requirements Analysis: The very first step in designing a database applications is to understand what data is to be stored in the database, what applications must be built on the database and what operations must be performed on the database.

Conceptual Database Design: The information which is gathered from requirement analysis step is used to develop a high-level description of the data to be stored in the database along with the conditions. The goal is to create a description of the data that matches to how both users and developers think of the data. This facilitates all the people involved in the design process ie. Developers and as well as users who have no technical background. Thus, the conceptual database design phase is used in drawing ER model.

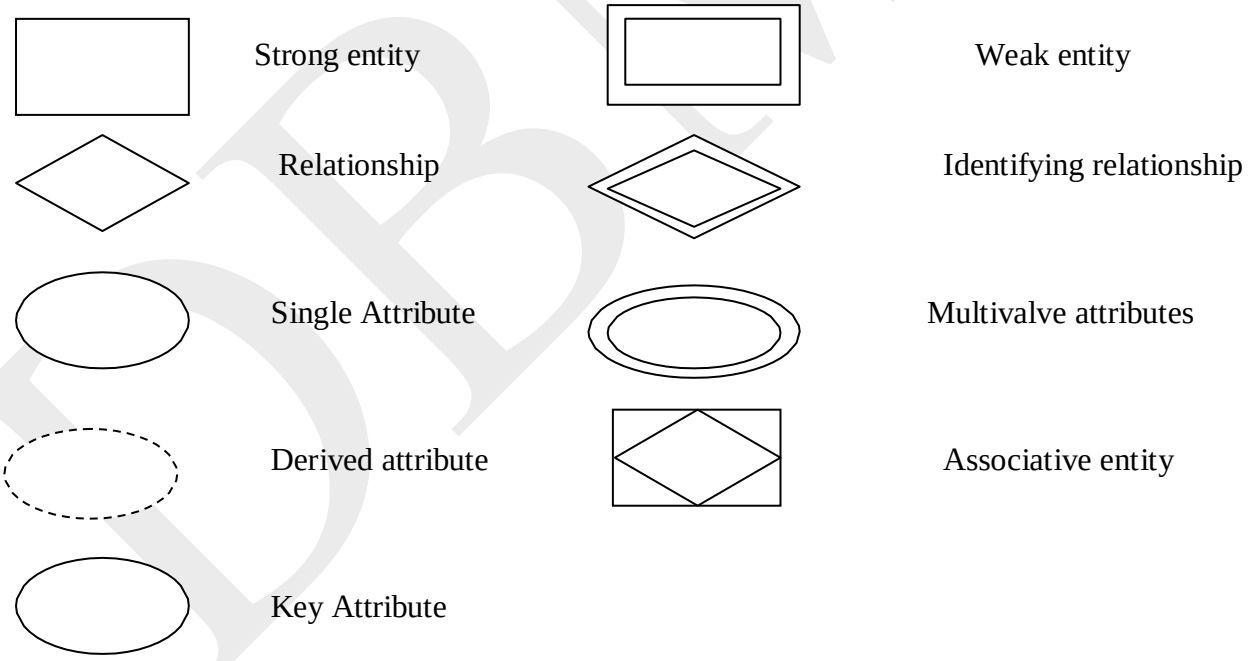
Logical Database Design: This step converts the conceptual database design into a database schema in the data model of the DBMS. It means, convert the ER model diagrams into a relational database schema.

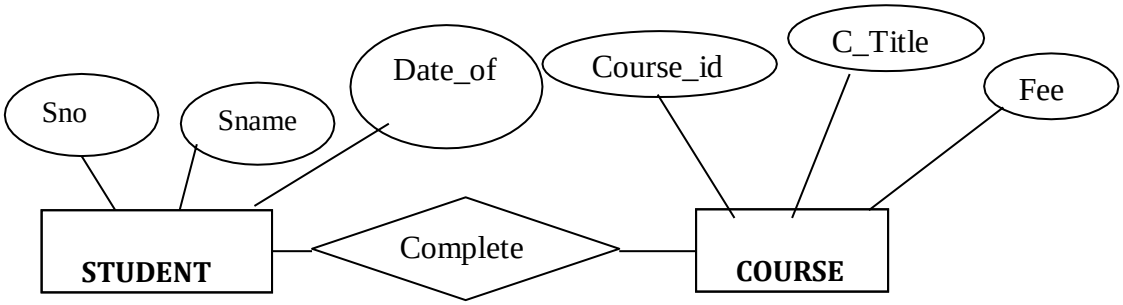
Schema Refinement: This step is used to analyze the collection of relations (tables) in our relational database schema to identify the future problems and to refine (clear) it.

Physical Database Design: This step involve for building indexes on some tables and clustering some tables or it may involve redesign of parts of the database schema obtained from the earlier design steps.

Application and Security Design: Any software project that involves a DBMS must consider applications that involve processes and identify the entities. **Examples:** users, user groups, departments etc.

→**E-R diagrams:** The following diagrams are using to construct or build the ER models.

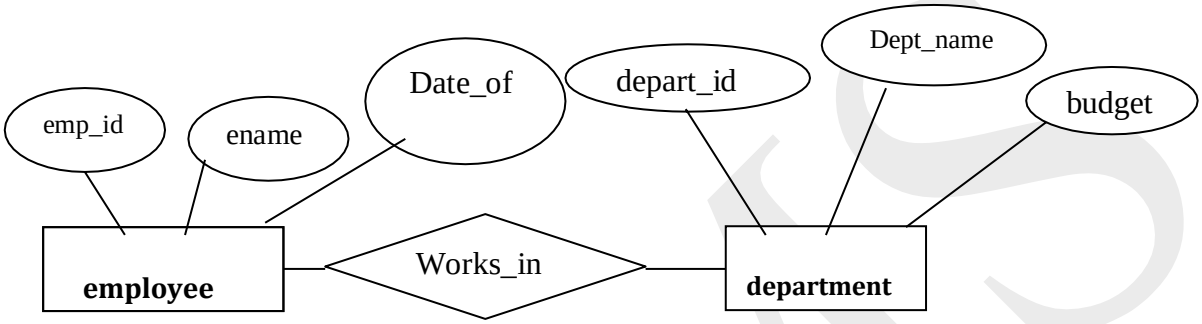




Beyond ER (Entity-Relationship) Design: The Entity-Relationship (ER) data model allows us to describe the data involved in real-world enterprises in terms of objects (entities) and their relationships and is widely used to develop an initial database design.

The important role of ER model in database design is to provide a useful concept that allows changing the detailed and informal description of what users want to implement in DBMS. The E-R diagram is built up from the following components,

1. **Rectangles:** Which represent entity sets.
2. **Diamonds:** Which represent relationship among entity sets that are connected to the rectangles by lines.
3. **Ellipses :** Which represent attributes and are connected to the entities or relationship by lines.
4. **Lines :** Which link attributes to entity sets and entity sets to relationships.



E-R MODEL[Entity Relationship Model]:

A logical representation of the data for an organization (or) for a business area is called E-R Model.

E-R MODEL CONSTRUCTS: The basic constructs of the entity-relationship model are **entities**, **relationships**, and **attributes**.

Entities : An entity is a thing, person, place, object, event, or concept in the user environment about which the organization wishes to maintain data.

Person : EMPLOYEE, STUDENT, PATIENT
Place **:** STORE, WAREHOUSE, STATE
Object **:** MACHINE, BUILDING, AUTOMOBILE
Event **:** SALE, REGISTRATION, RENEWAL
Concept**:** ACCOUNT, COURSE, WORK CENTER

Entity type: A collection of entities that share common properties or characteristics(attributes). Each entity type in an E-R model is given a name. The name represents a collection of items and is always singular.

Entity Instance : A single occurrence of an entity type .

Relationship type: The Relationship type is a meaningful association between entity types. An association between entity instances where each relationship instance includes exactly one entity from each participating entity type is called relationship instance.

Attribute: An attribute is a property or characteristic of an entity type that is of interest to the organization.

ENTITIES, ATTRIBUTES AND ENTITY SETS:

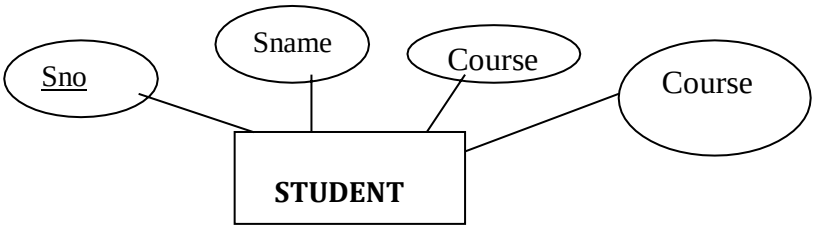
Entity: An entity is an object in the real world that is distinguishable from other objects". For examples Building, room, chair, transaction, course, machine, department, employee etc.

Entities are the basic units used in modeling classes. Entities can have constitute ideas or concepts.

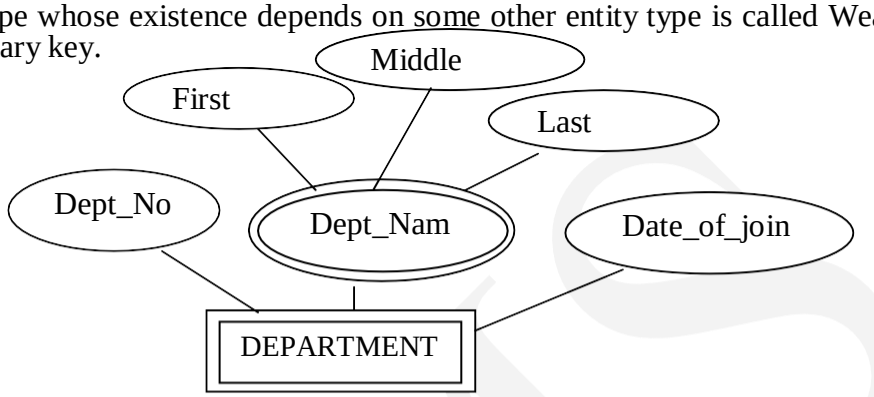
Entity Set: An entity set is a collection of similar entities or objects which maintains the data. Example, job details, course details, staff details manager details, flight details etc.

Entity Types: Entities are two types. They are 1. Strong entity 2. Weak entity.

Strong entity : An entity that exists independently of other entity types is called Strong entity. Eg: STUDENT, EMPLOYEE, AUTOMOBILE, and COURSE.



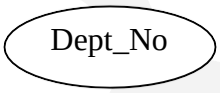
Weak entity : An entity type whose existence depends on some other entity type is called Weak entity and which does not have a primary key.



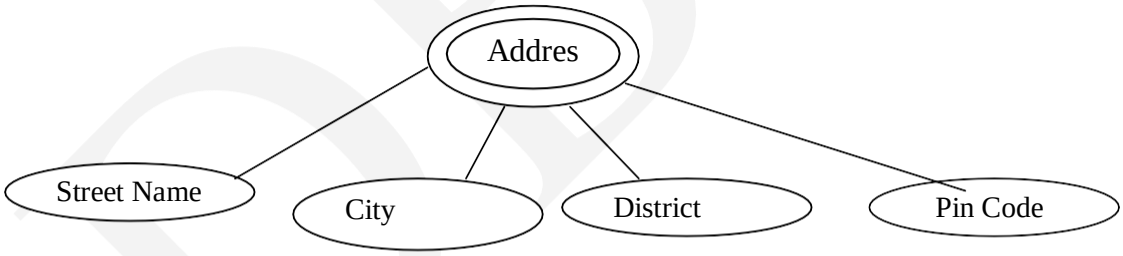
Attribute: An entity is represented by an ellipse symbol. An attribute is a property or characteristic of an "entity type" of an organization. For example, employee entity has its own attributes such emp_number, ename, salary etc.

Types:

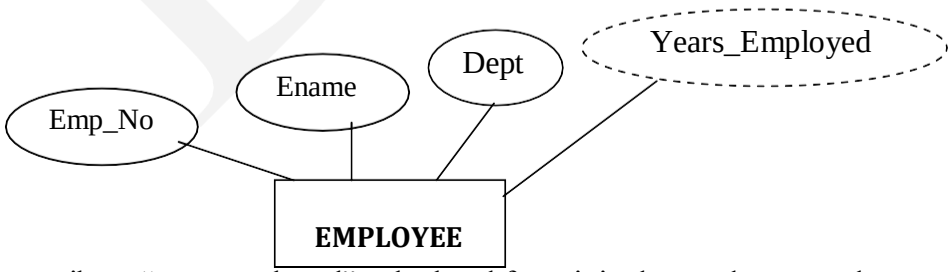
Single valued attribute: The attributes that have a single value for a particular entity is known as single valued attribute. For example, emp_number can be only a single value for a particular employee.



Multivalued valued Attribute: An attribute that may take on more than one value for a given entity instance. Eg:



Derived attribute: An attribute whose values can be calculated from related attribute values.

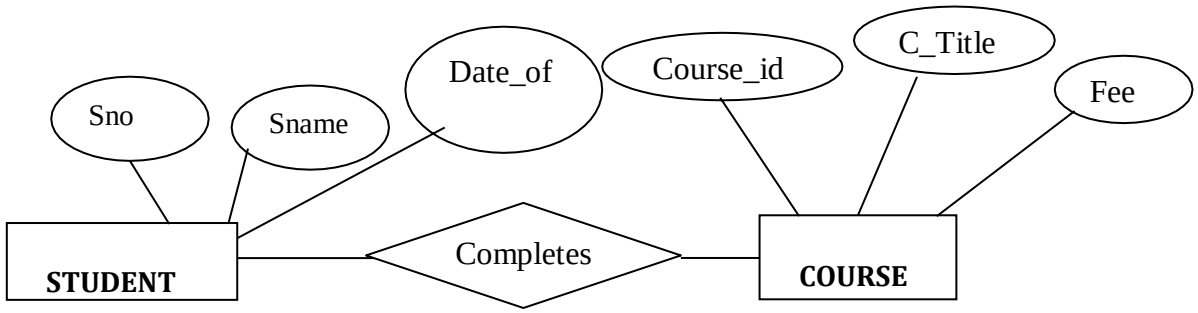


In this, the attribute "year employed" calculated from join date and present date.

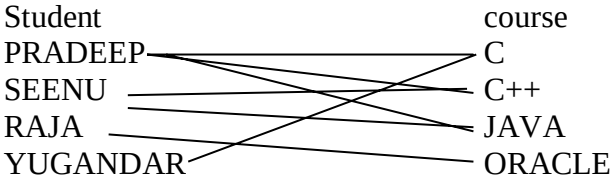
Composite attribute: An attribute that can be broken down into component parts.

Null Attributes: A null attribute is an attribute that uses a null value when a entity does not have a value for an attribute.

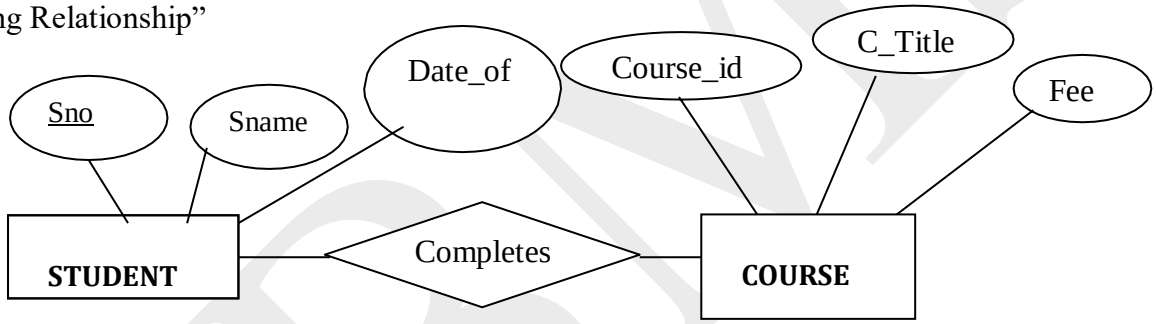
Relationship and Relationship Sets: The Relationship is an association between two or more entities. This relation is denoted by a diamond symbol that containing the name of the relationship.



Relationship instances: It is an association between among entity instances which establish the relationship with more than one instance. For example, A student may taken more than one course at the same time. I.e.

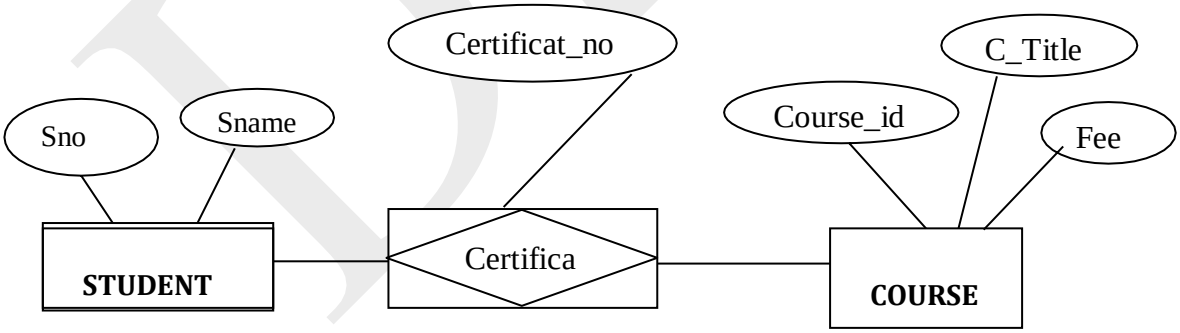


Identifying Relationship : The relationship between strong entity type and weak entity type is called "Identifying Relationship"



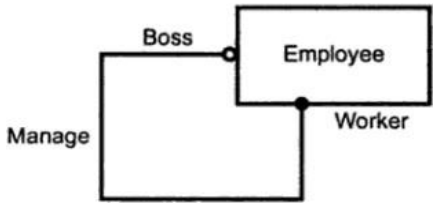
In the above example in student entity property „sno" is primary key and course entity property „course_id" is discriminator key

Associative entity: An entity type that associates the instances of one or more entity types and contains attributes that are peculiar to the relationship between those entity instances.



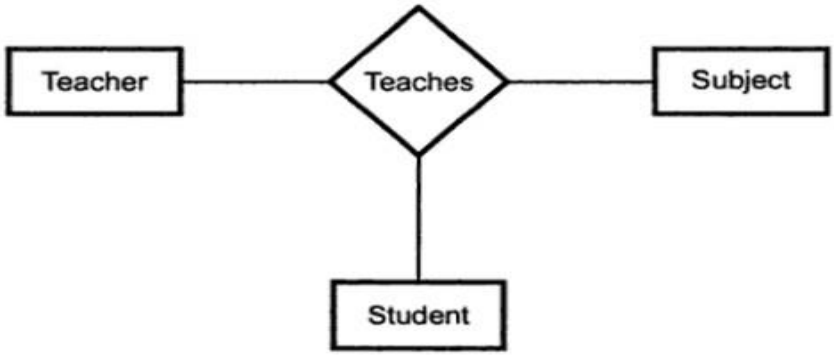
TYPES OF RELATIONSHIPS

Unary relationship: A unary relationship exists when an association is maintained within a single entity. For example boss and worker distinguish the two employees participating in the manage association

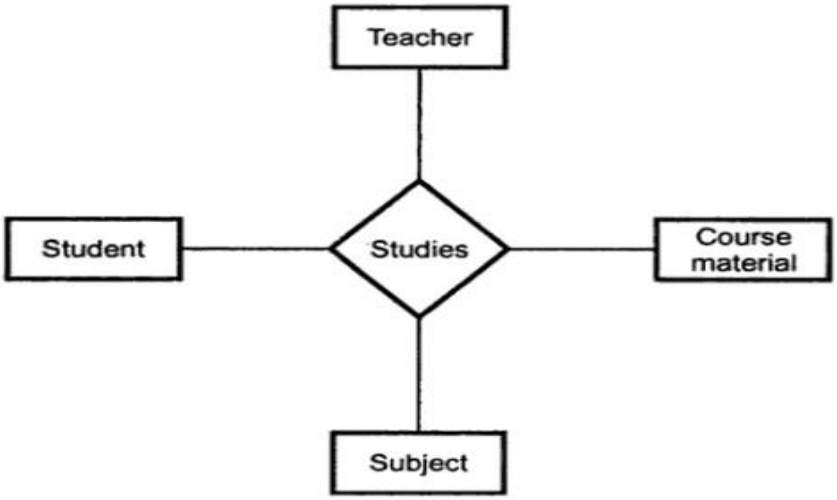


Binary relationship: A binary relationship exists when two entities are associated. For example the book, publisher relationship shown below.

Ternary Relationship: A ternary relationship exists when there are three entities associated for example, the entities teacher, subject and student are related using ternary relationship called „teaches".



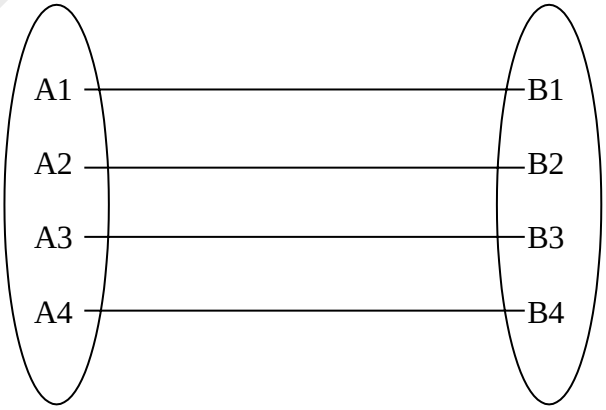
Quaternary relationship: A quaternary relationship exists when there are four entities associated. An example of quaternary relationship is „studies" where four entities are involved student, teacher, subject and course-material.



Mappin Cardinalities: mapping cardinalities the number of entities to which another entity can be associated via a relationship set.

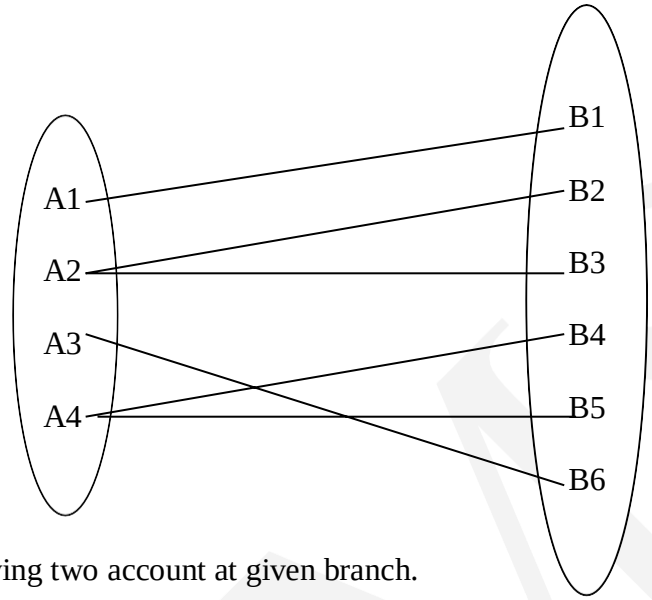
For a binary relationship set R between entity sets A and B, the mapping cardinalites must be one of the falowing:

One to one: An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.



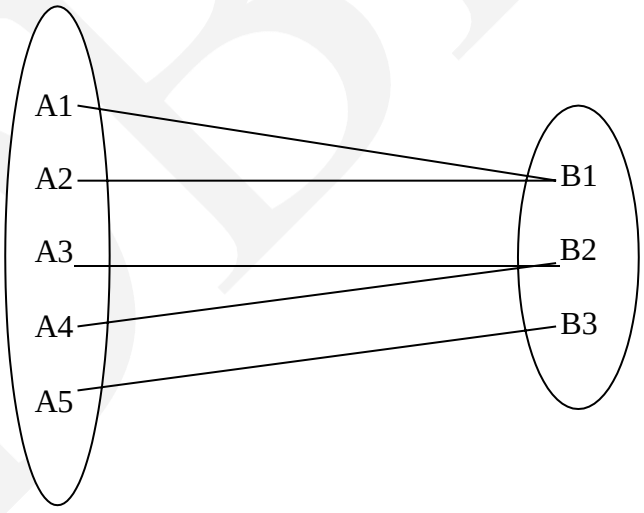
Example: A customer with single account at given branch.

One-to-many: An entity in A is associated with any number in B. An entity in B is associated with at most one entity in A

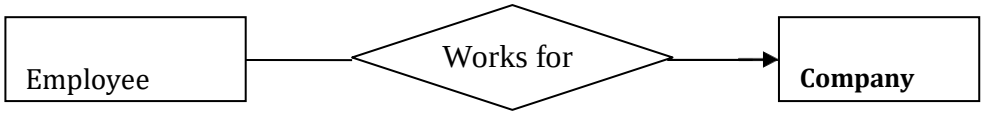


Example: A customer having two account at given branch.

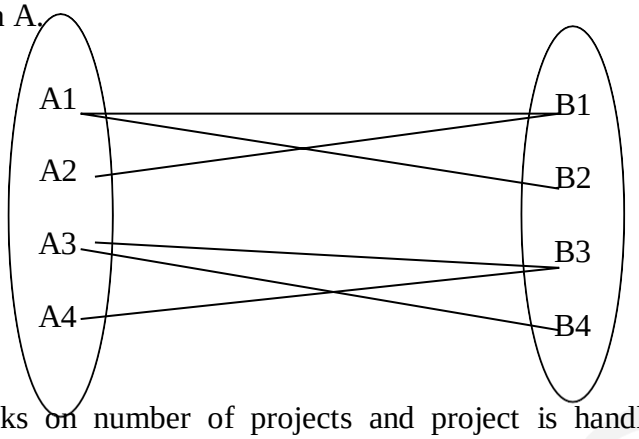
Many-to-one: An entity in A is associated with at most one entity in B. An entity in B is associated with any number of entities in A.



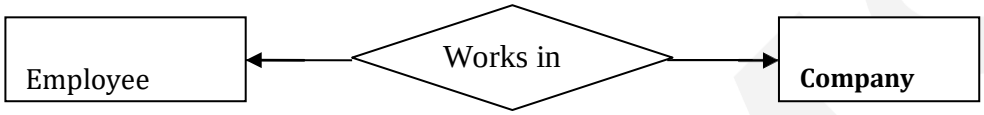
Example: Many employees works for a company. This relationship is shown by mant-to-one



Many-to-Many: An entity in A is associated with any number of entities in B. An entity in B is associated with any number entities in A.



Example: Employee works on number of projects and project is handled by number of employees. Therefore the relationship between employee and project is manu-to-many



Cardinality constraints: It specifies the number of instances of one entity that can be associated with each instance of another entity.

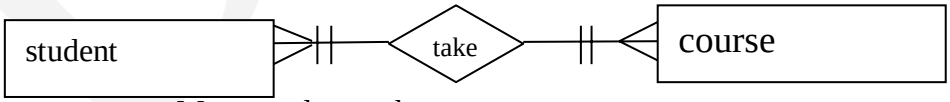
Mandatory one: It means an instance of entity must be associated with another instances of entity. For example, examine the following relationship "PROFESSOR GUIDE THE STUDENTS" . In this, professor must guide the students.

A professor guide only one student

Optional one : It means an instance of entity optionally associated with another instance of entity. For example, examine the following relationship "PROFESSOR GUIDE THE STUDENTS" . In this, professor may or may not guide the students.

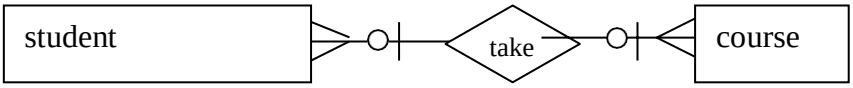
A professor may guide only one student

Mandatory many: It means the number of instances of an entity must be associated with another instance of entity.



Many students take many courses

Optional many : It means, the number of instances of an entity optionally associated with another instance of entity.



Many students optianally take many courses

Minimum cardinality: A cardinality constraint specifies the minimum number of instances of one entity that can be associated with each instance of another entity.

Maximum cardinality : A cardinality constraint that specifies the maximum number of instances one entity that can be associated with each instance of another entity.

Rules for naming relationships:

1) A relationship name is verb phrase.
2) A relationship name is in lower case letter.
3) Relationship names represents action being taken , i.e., usually in present tense.

Rules for defining relationships:

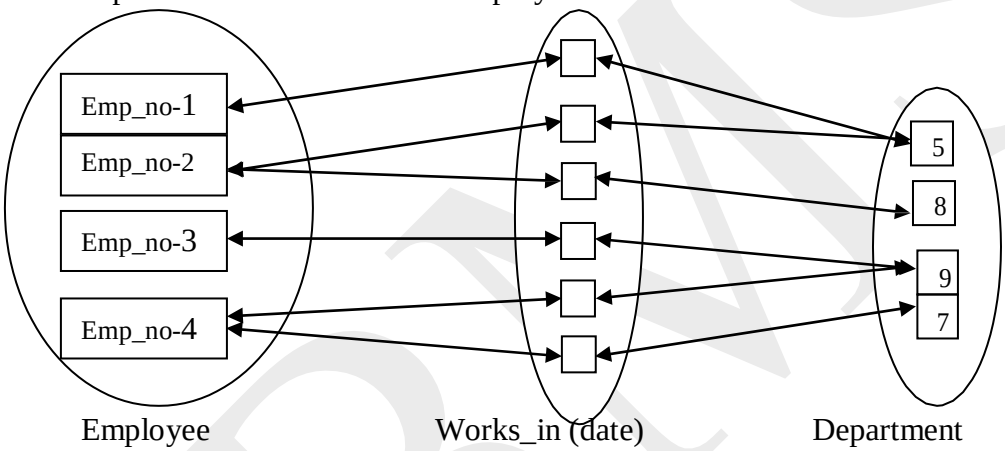
A relationship definition explains what relationship is and what is important.

1) It may be important to state that who does the action.
2) It may be important to give examples to clarify the action.
3) The definition should explain any optional participation.
4) A relationship definition should explain any restrictions on participating in relationship.

Additional Features of ER model:

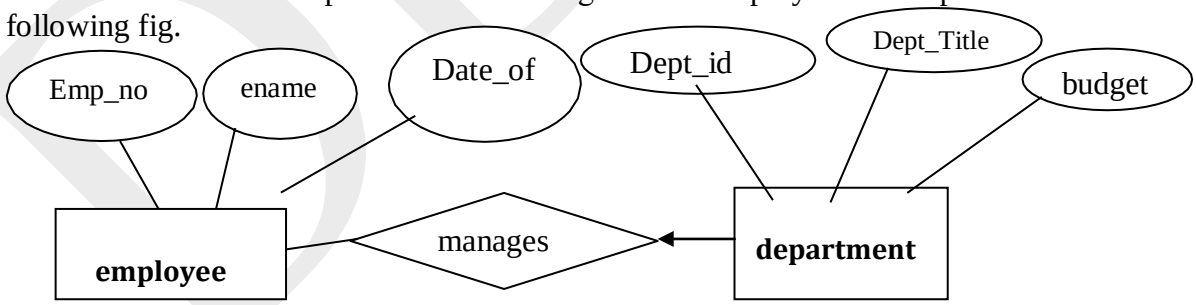
The additional features of ER model allow us to describe some common properties (attributes) of the data.

Key constraints: The Key constraints are used to provide the restrictions to an entity. So that allows unique instance. Consider the "Works_In relationship" example shown in fig that an employee can work in several departments and a department can have several employees.



In the above fig. each employee is denoted by its "emp_no" and each department is denoted by number such as 5,8,9,7. Here "works_in" is a relationship that establish the relationship b/w employee and department. Here the emp_no 2 works in 5 & 8 departments and emp_no4 works in 9 & 7 departments.

Now consider another relationship set called "manages" b/w employee and department entities. This is shown in following fig.



In this example, we applied "key constraint" by indicating the arrow mark from "department" entity to "manages" relationship. So that each department entity appears in at most one manages relationship. Thus, the arrow mark said that given a department entity is uniquely determine the "manages" relationship in which it appears.

Participation Constraints: The participation of 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.

If only some entities in E participate in relationships in R, the participation of entity set E in relationship R is said to be **partial.**

For example, we expect every loan entity to be related to at least one customer through the borrower relationship. Therefore, the participation of loan in the relationship set borrower is total. While, a bank

customer may or may not have a loan. Therefore, the participation of customer in the borrower relationship is partial.

Class Hierarchies: To classify the entities in an entity set into subclass entity is known as class hierarchy. This hierarchy contains the model that has resulted from extending the original E-R model with new modeling constructs. Such type of new modeling constructs are Sub type/Super type relationship. This facility allows us to model a general entity type, called super type and then subdivide it into several specialized entity types, called subtype.

For example, the entity type EMPLOYEE can be modeled as a super type with subtypes such as "hourly employees", "salaried employees", and "contract consultants".
super type entity and sub type entity.

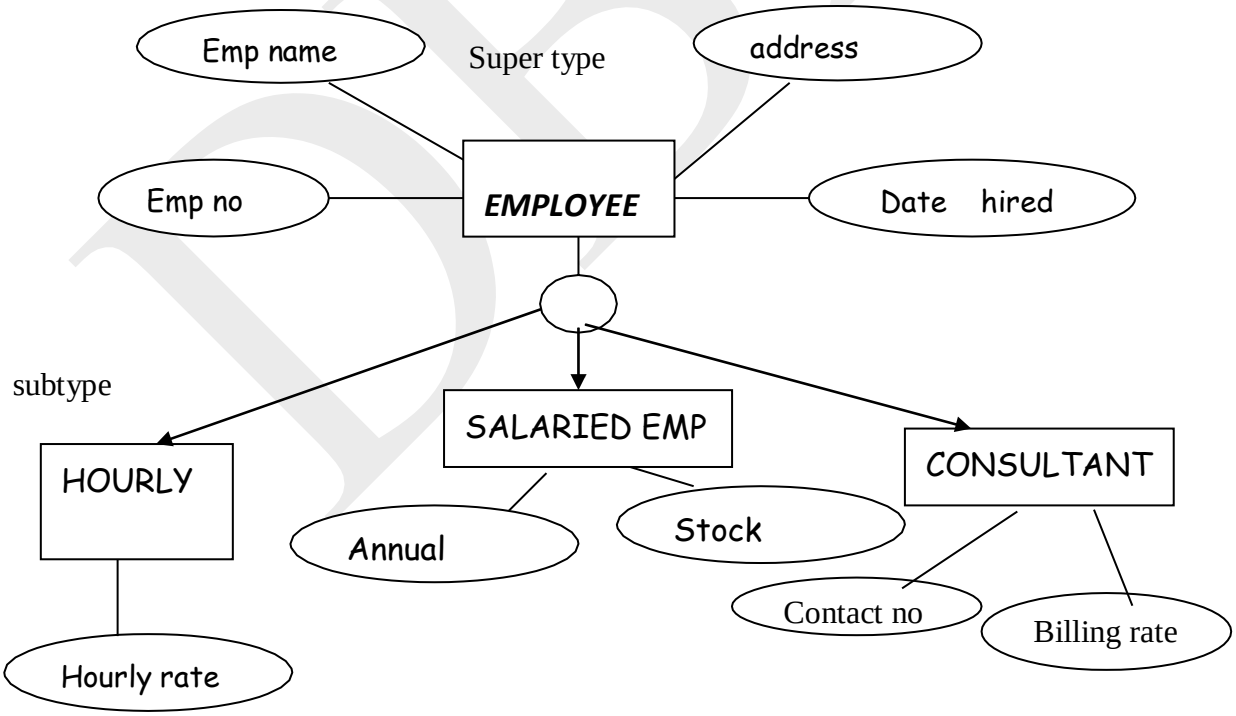
Sub type: A sub grouping of entities in an entity type that is meaningful to organization and that share the common attributes that are distinct from other attribute of other type.

Super type : It is a generic entity type that has a relationship with one or more subtypes. The relationship of Super type and sub type is shown in simple example. Suppose that an organization has three basic types of **employee**: i.e. hourly employees, salaried employees, and contract consultants.

Ex: Hourly employees contain Employee no, emp name ,address, date hired, hourly rate.

Salaried employees contain emp no,emp name ,address, date hired, annual salary, stock option.

Contract consultants contain emp no ,emp name, address, date hired, contract no, billing rate .



In the above example, the **subtype** entities inherit the values of all attributes of the super type. This type of process is called Attribute inheritance. i.e. emp no, empname, address etc are also assigned to subtypes.

Extended E-R model:

The E-R model that is supported with the additional semantic concepts is called the **extended entity relationship model or EER model.** The EER model includes all the concepts of the original E-R together with the following additional concepts:

- Specialization
- Generalization
- Aggregation

Specialization:

"Specialization is the process of designating sub grouping within an entity set."

Specialization is a top-down process. Consider an entity set person, with attributes name, street, and city. A person may be further classified as one of the following

- Customer
- Employee

Each of these person types is described by a set of attributes that includes all the attributes of entity set person plus additional attributes. For example, customer entities may be further described by customer_id and employee entities by employee_code and salary.

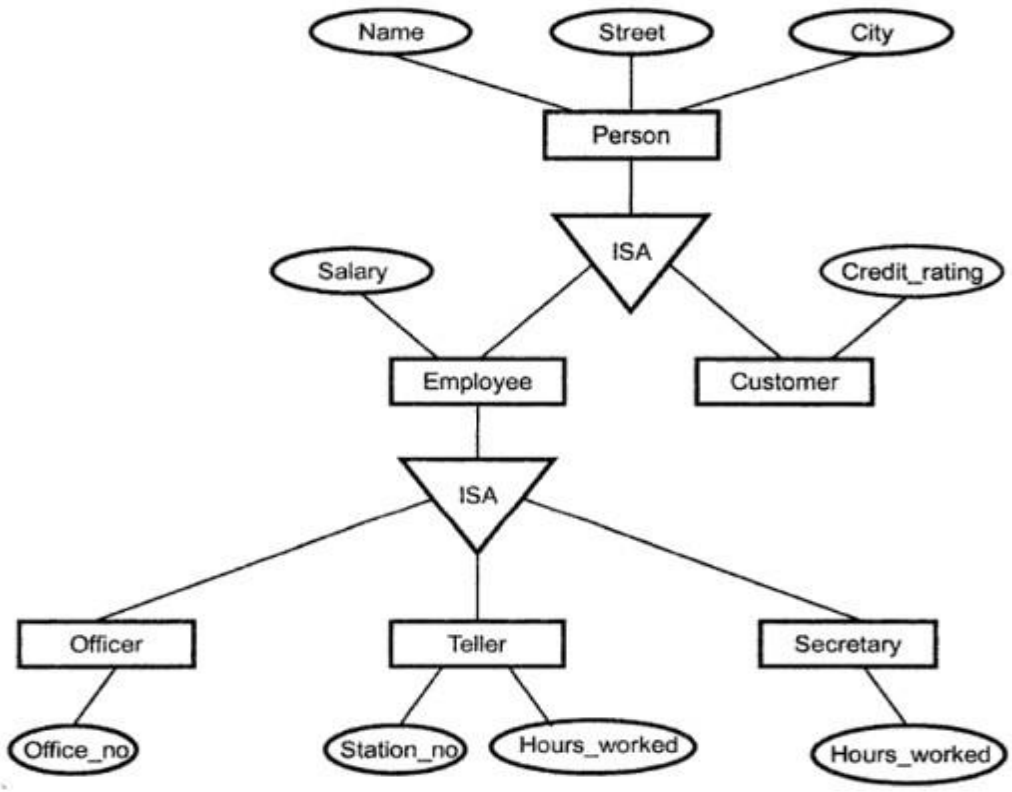
Specialization which is represented by triangle. The lable ISA stands for "is a" and represents for example that customer "is a" person. The ISA relationship may also be referred to as a super class-subclass relationship.

Generalization:

"Generalization is the process of defining a more general entity type from a set of more specialized entity types"

Generalization is a bottom-up approach. This approach results in the identification of a generalized super class from the original subclasses.

Consider that the attributes of „customer" entity are customer_id, name, street, city and an „employeee" entity attributes are employee_code, street, city and salary. Thus, the entity sets employee and customer have several attributes in common. This commonality can be expressed by generalization which is a containment relationship that exists between a higher-level entity set and one or more lower-level entity sets. Person is higher-level entity set and customer and employee can lower-level entity sets. In other words, person is a super class if customer and employee are subclasses.



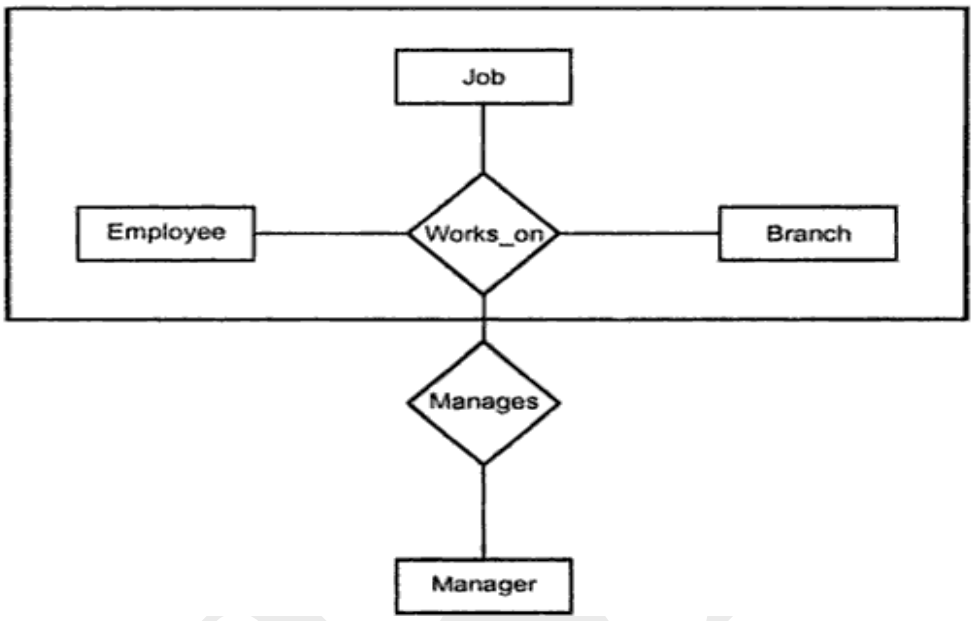
Aggregation

One limitation on E-R model is that cannot express relationships among relationships.

Consider a quaternary relationship manages between employee, branch, job and manager, Using the basic E-R modeling constructs, we obtain the E-R diagrams as shown below,

There is redundant information in the resultant figure, since every employee, branch, job combination in manages is also in Works_on.

The best way to model above situation is to use aggregation. Aggregation is an abstraction through hich relationships are treated as higher-level entities. Thus, the relationship set Works_on relating the entity sets employee, branch and job is considered as a higher_level entity set called Works_on. We can then create a binary relationship manages between Works_on and manager to represent who manages what tasks.



CONCEPTUAL DESIGN WITH THE ER MODEL:

Developing an ER diagram presents several design issues including the following:

1. Entity versus Attribute.
2. Entity versus Relationship
3. Binary versus Ternary Relationships
4. Aggregation versus Ternary Relationships.

Entity Versus Attribute: While identifying the attributes of an entity set, it is sometimes not clear whether a property should be modeled as an attribute or as an entity set.

For example Should address be an attribute of Employees or an entity (connected to Employees by a relationship)?

Depends upon the use we want to make of address information, and the semantics of the data:

If we have several addresses per employee, address must be an entity (since attributes cannot be set-valued).

If the structure (city, street, etc.) is important, e.g., we want to retrieve employees in a given city, address must be modeled as an entity (since attribute values are atomic).

Entity Vs Relationship: It is not always clear whether an object is best expressed by an entity set or a relationship set.

For example, A bank loan is modeled as an entity. An alternative is to model a loan not as an entity, but rather as a relationship b/w customers and branches, with loan number and amount as descriptive attributes. Each loan is represented by a relationship b/w a customer and a branch.

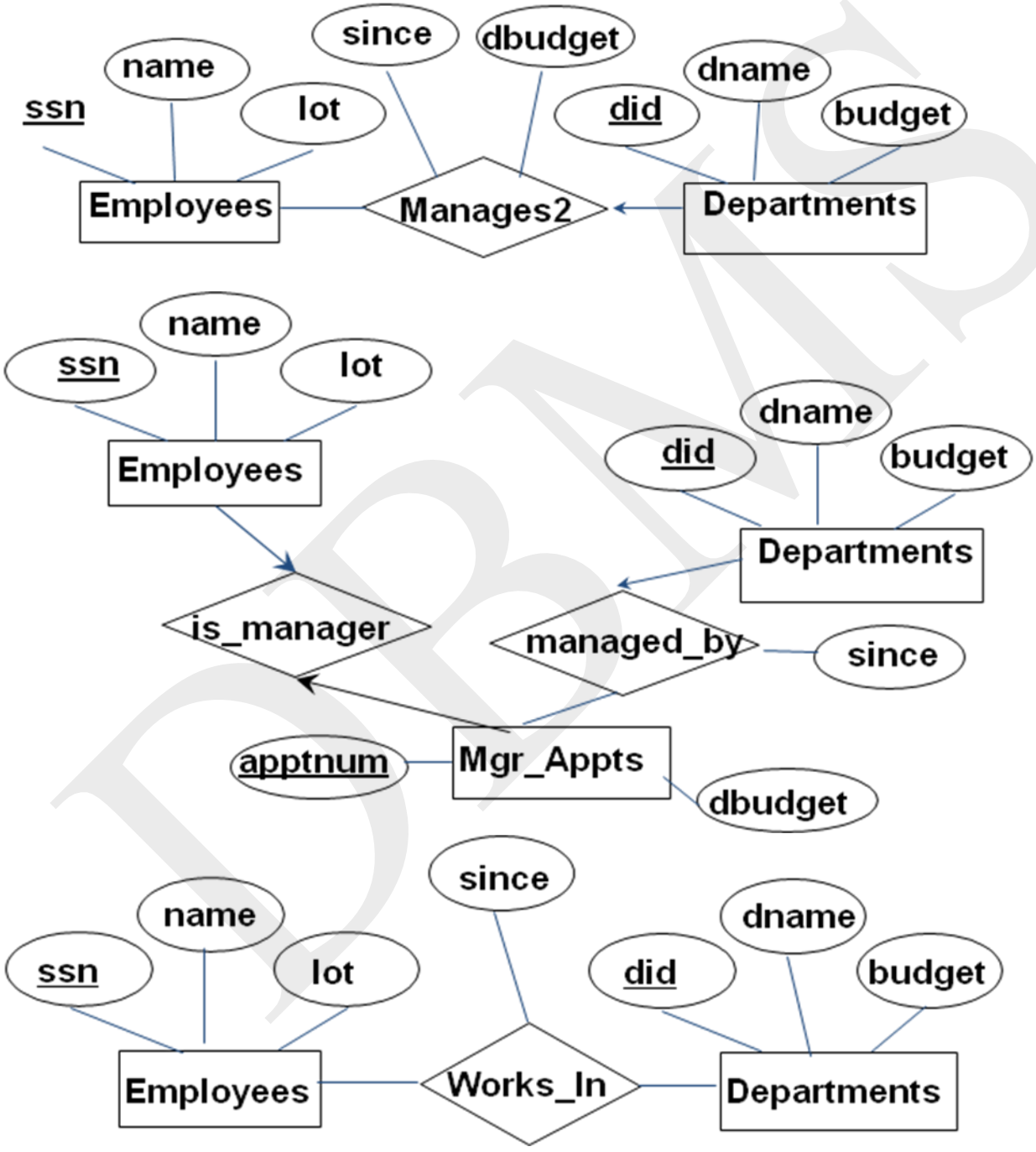
If every loan is held by exactly one customer and customer is associated with exactly one branch, we may find satisfactory the design, where a loan is represented as a relationship.

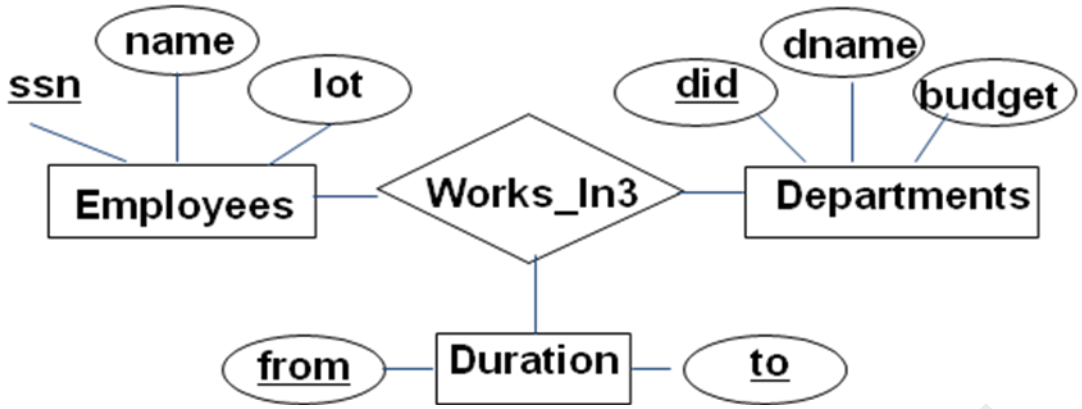
But, with this design, we cannot represent conveniently a situation in which several customers hold a loan jointly. We must define a separate relationship for each holder of the joint loan. Then we must replicate the values for the descriptive attribute loan_number and amount in each such relationship. Each such relationship must of course, have the same value for the descriptive attributes loan_number and amount.

Two problems arise as a result of the replication.

1. The data is stored multiple times, wasting storage space.
2. Updates, leave the data in an inconsistent state.

One possible guideline is determining whether to use an entity set or a relationship set to designate a relationship set, an action that occurs b/w entities. This approach can also be useful in deciding whether certain attributes may be more appropriately expressed as relationships.





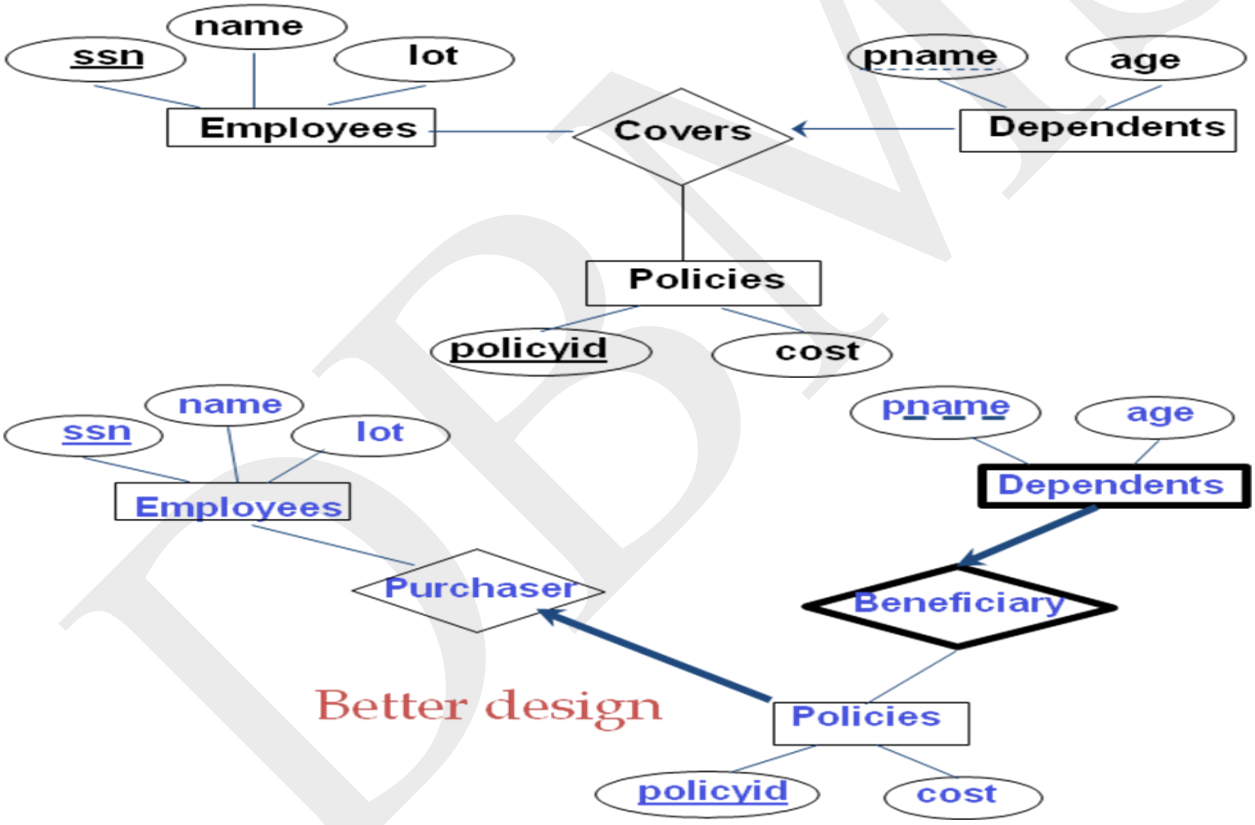
Binary Vs Ternary Relationships: It is always possible to replace a non-binary relationship set by a number of distinct binary relationship sets.

This example an employee can own several policies each policy can be owned by several employees and each dependent can be covered by several policies.

suppose that we have the following additional requirements.

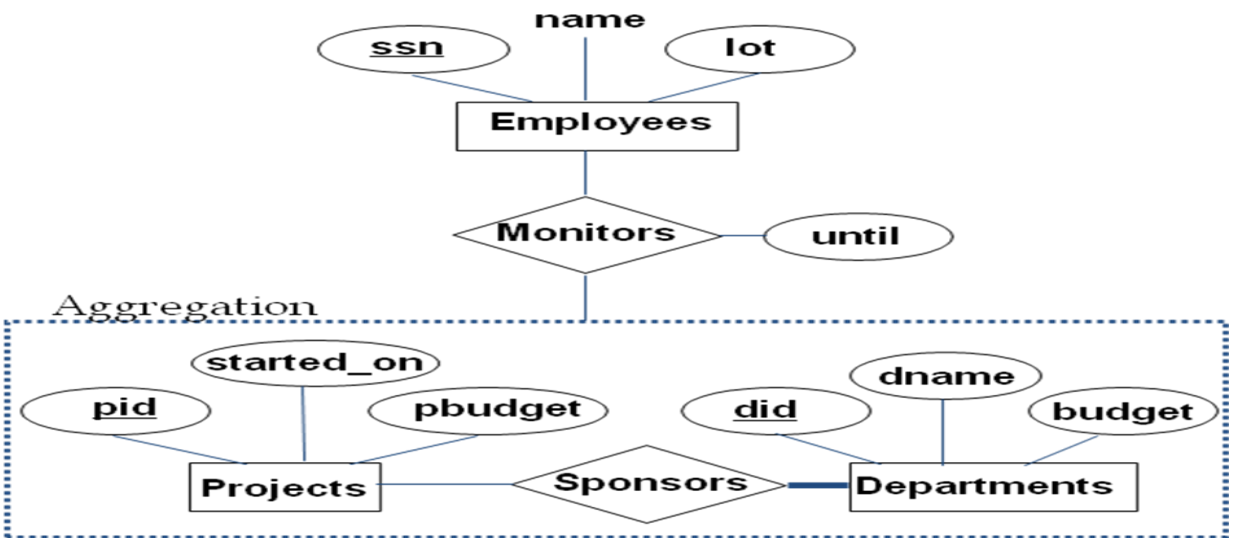
A policy cannot be owned jointly by two or more employees.

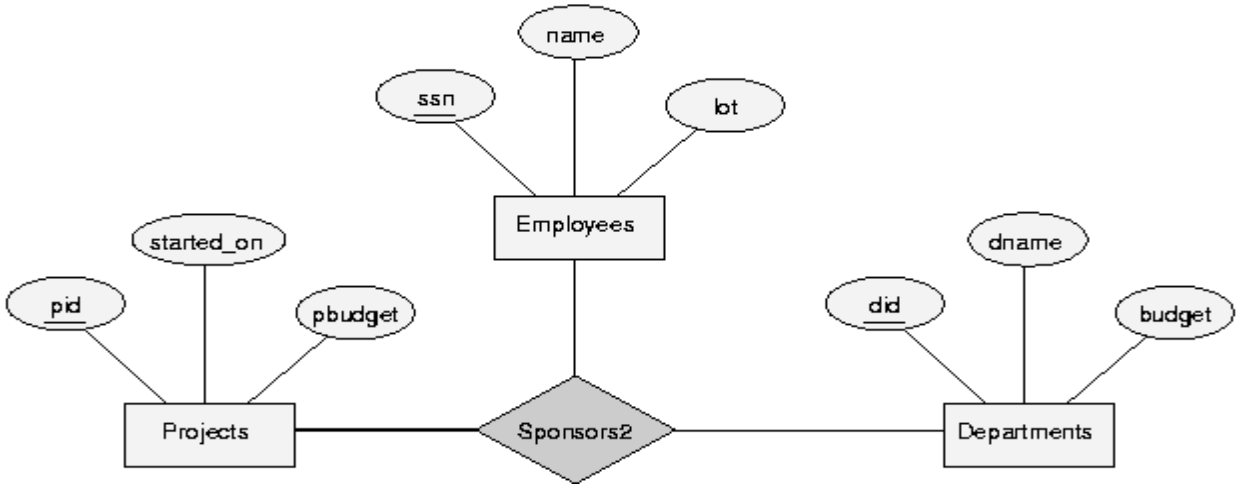
Every policy must be owned by some employee.



Aggregation Vs Ternary Relationship: The choice b/w using aggregation or a ternary relationship is mainly determined by the existence of a relationship that relates a relationship set to an entity set. The choice may also be guided by certain integrity constraints that we want to express.

Example, Consider the constraint that each sponsorship be monitored by at most one employee. We can express this constraint in terms of the sponsors relationship set. On the other hand, we can relationship sponsors to the relationship monitors. Thus, the presence of such a constraint serves a another reason for using aggregation rather than a ternary relationship set.


name
ssn
lot
Employees
Monitors
until
Aggregation
started_on
pid
pbudget
dname
did
budget
Projects
Sponsors
Departments


name
ssn
lot
Employees
started_on
pid
pbudget
dname
did
budget
Projects
Sponsors2
Departments