UNIT 1 NOTES

Database

A database is a collection of data, typically describing the activities of one or more related organizations.

Example: a university database might contain information about the following:
• *Entities* such as students, faculty, courses, and classrooms.
• *Relationships* between entities, such as students' enrollment in courses, faculty teaching courses, and the use of rooms for courses.

Database Management System (DBMS)

- A database management system, or DBMS, is software designed to assist in maintaining and utilizing large collections of data.
- The alternative to using a DBMS is to store the data in files and write application-specific code to manage it.

A Historical Perspective or History of Data base Systems

- Finding information from a huge volume of papers or deleting/modifying an entry is a difficult task in pen and paper based approach.
- To overcome the hassles faced in manual record keeping, it is desirable to computerize storage of data.
- From the earliest days of computers, storing and manipulating data have been a major application focus.
- In the late 1960s, IBM developed the Information Management System (IMS) DBMS, used even today in many major installations.
- IMS formed the basis for an alternative data representation framework called the *hierarchical data model.*
- In 1970, Edgar Codd, at IBM's San Jose Research Laboratory, proposed a new data representation framework called the *relational data model.*
- In the 1980s, the relational model consolidated its position as the dominant DBMS paradigm, and database systems continued to gain widespread use.
- SQL was standardized in the late 1980s, and the current standard, SQL: 1999, was adopted by the American National Standards Institute (ANSI) and International Organization for Standardization (ISO).
- Specialized systems have been developed by numerous vendors for creating *data warehouses,* consolidating data from several databases, and for carrying out specialized analysis.
- Commercially, database management systems represent one of the largest and most vigorous market segments.

FILE SYSTEMS VERSUS DBMS

File System	Database Management System (DBMS)
1. It is a software system that manages and controls the data files in a computer system.	1. It is a software system used for creating and managing the databases. DBMS provides a systematic way to access, update, and delete data.
2. File system does not support multi-user access.	2. Database Management System supports multi-user access.
3. Data consistency is less in the file system.	3. Data consistency is more due to the use of normalization.
4. File system is not secured.	4. Database Management System is highly secured.
5. File system is used for storing the unstructured data.	5. Database management system is used for storing the structured data.
6. In the file system, data redundancy is high.	6. In DBMS, Data redundancy is low.
7. No data backup and recovery process is present in a file system.	7. There is a backup recovery for data in DBMS.
8. Handling of a file system is easy.	8. Handling a DBMS is complex.
9. Cost of a file system is less than the DBMS.	9. Cost of database management system is more than the file system.
10. If one application fails, it does not affect other application in a system.	10. If the database fails, it affects all application which depends on it.
11. In the file system, data cannot be shared because it is distributed in different files.	11. In DBMS, data can be shared as it is stored at one place in a database.
12. These system does not provide concurrency facility.	12. This system provides concurrency facility.
13. **Example:** NTFS (New technology file system), EXT (Extended file system), etc.	13. **Example:** Oracle, MySQL, MS SQL Server, DB2, Microsoft Access, etc.

ADVANTAGES OF A DBMS

Data Independence: The DBMS provides an abstract view of the data that hides data representation and storage details.
Efficient Data Access: A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently.
Data Integrity and Security: If data is always accessed through the DBMS, the DBMS can enforce integrity constraints. For example, before inserting salary information for an employee, the DBMS can check that the department budget is not exceeded. Also, it can enforce *access controls* that govern what data is visible to different classes of users.
Data Administration: When several users share the data, centralizing the administration of data can offer significant improvements.
Concurrent Access and Crash Recovery: A DBMS schedules concurrent accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time. Further, the DBMS protects users from the effects of system failures.
Reduced Application Development Time: The high-level interface to the data, facilitates quick application development.

Database System Applications

Applications where we use Database Management Systems are:

- **Telecom**: There is a database to keep track of the information regarding calls made, network usage, customer details etc. Without the database systems it is hard to maintain that huge amount of data that keeps updating every millisecond.
- **Industry**: Where it is a manufacturing unit, warehouse or distribution centre, each one needs a database to keep the records of ins and outs. For example distribution centre should keep a track of the product units that supplied into the centre as well as the products that got delivered out from the distribution centre on each day; this is where DBMS comes into picture.
- **Banking System**: For storing customer info, tracking day to day credit and debit transactions, generating bank statements etc. All this work has been done with the help of Database management systems.
- **Education sector**: Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance details, fees details etc. There is a hell lot amount of inter-related data that needs to be stored and retrieved in an efficient manner.
- **Online shopping**: You must be aware of the online shopping websites such as Amazon, Flipkart etc. These sites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query. All this involves a Database management system.

DATA MODEL

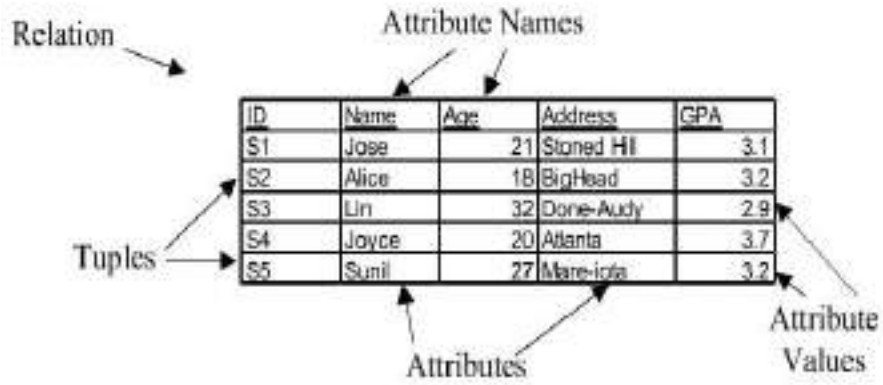
- A **data model** is a collection of high-level data description constructs that hide many low-level storage details.
- A DBMS allows a user to define the data to be stored in terms of a data model.
- Most database management systems today are based on the **relational data model.**
- A **semantic data model** is a more abstract, high-level data model that makes it easier for a user to come up with a good initial description of the data in an enterprise.
- A widely used semantic data model called the entity-relationship (ER) model allows us to pictorially denote entities and the relationships among them.

Data model types

- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-Oriented data models
- Other older models:
  - Network model
  - Hierarchical model

Relational model

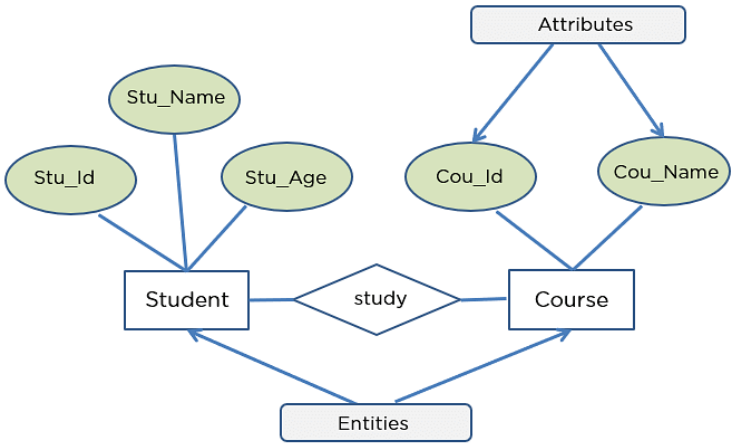
- The central data description construct in this model is a relation, which can be thought of as a set of records.
- A description of data in terms of a data model is called a schema.
- In the relational model, the schema for a relation specifies its name, the name of each field (or attribute or column), and the type of each field.
- Students (*sid:* string, *name:* string, *login:* string, *age:* integer, *gpa:* real)



ID	Name	Age	Address	GPA
S1	Jose	21	Stoned Hil	3.1
S2	Alice	18	BigHead	3.2
S3	Lin	32	Donie-Audy	2.9
S4	Joyce	20	Atlanta	3.7
S5	Sunil	27	Mare-iota	3.2

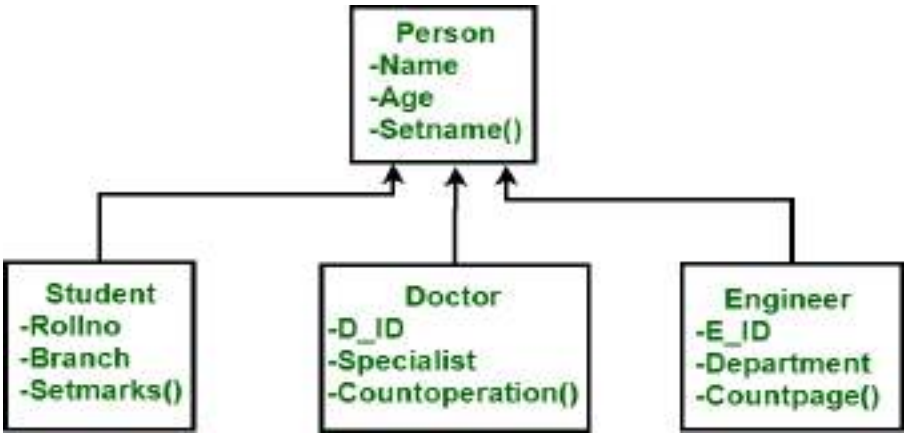
Entity-Relationship Model

- Entity-Relationship Model or simply ER Model is a high-level data model diagram.
- In this model, we represent the real-world problem in the pictorial form to make it easy for the stakeholders to understand.
- It is also very easy for the developers to understand the system by just looking at the ER diagram.
- We use the ER diagram as a visual tool to represent an ER Model.



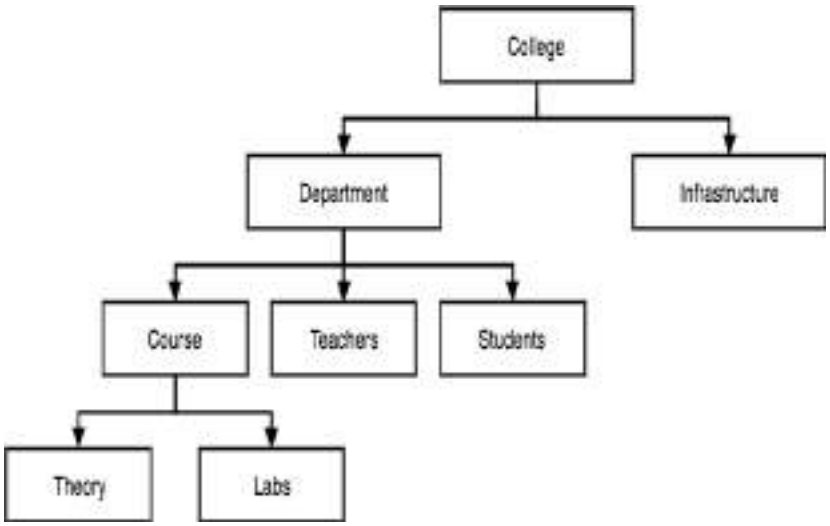
Object-Oriented Data Model

- The real-world problems are more closely represented through the object-oriented data model.
- In this model, both the data and relationship are present in a single structure known as an object.
- We can store audio, video, images, etc in the database which was not possible in the relational model (although you can store audio and video in relational database, it is advised not to store in the relational database).
- In this model, two are more objects are connected through links. We use this link to relate one object to other object.



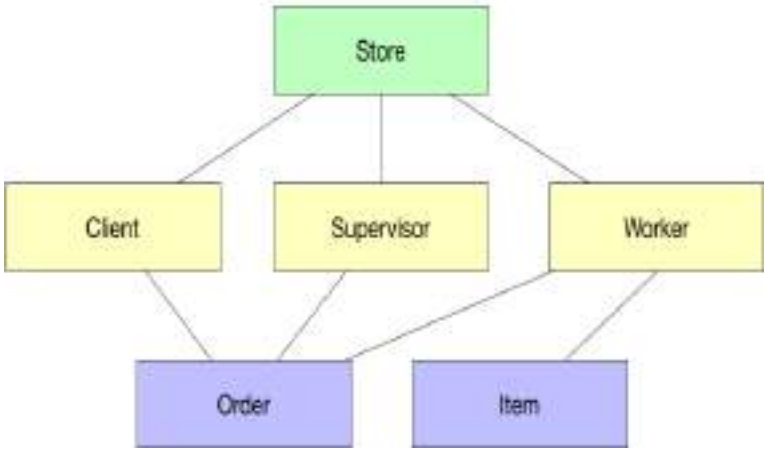
Hierarchical Model

- Hierarchical Model was the first DBMS model.
- This model organizes the data in the hierarchical tree structure.
- The hierarchy starts from the root which has root data and then it expands in the form of a tree adding child node to the parent node.
- This model easily represents some of the real-world relationships like food recipes, sitemap of a website etc.



Network Model

- This model is an extension of the hierarchical model.
- It was the most popular model before the relational model.
- This model is the same as the hierarchical model; the only difference is that a record can have more than one parent.
- It replaces the hierarchical tree with a graph.



LEVELS OF DATA ABSTRACTION

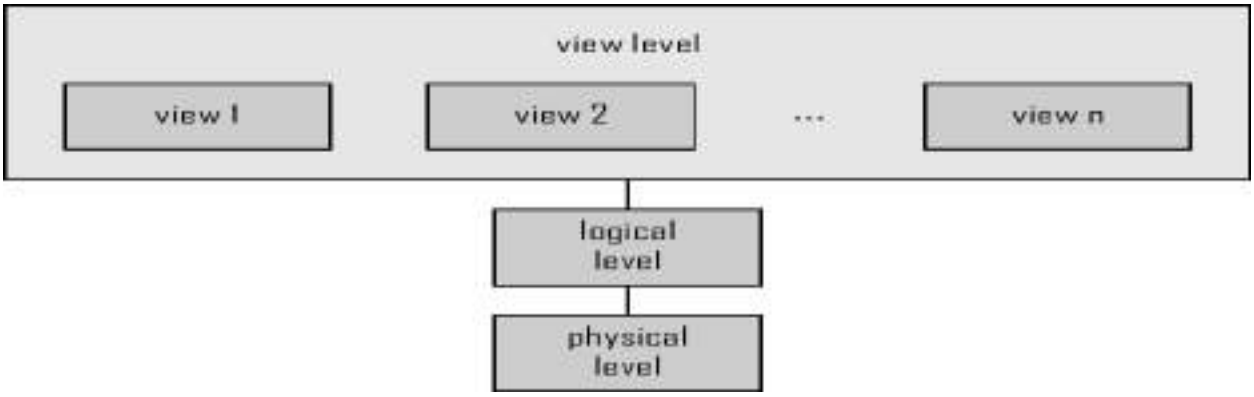
There are 3 level architecture of database design.

Physical level: describes how a record (e.g., customer) is stored.

Logical level: describes data stored in database, and the relationships among the data**.**

```
type customer = record
customer_id : string;
customer_name : string;
customer_street : string;
customer_city : string;
end;
```

View level: application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.



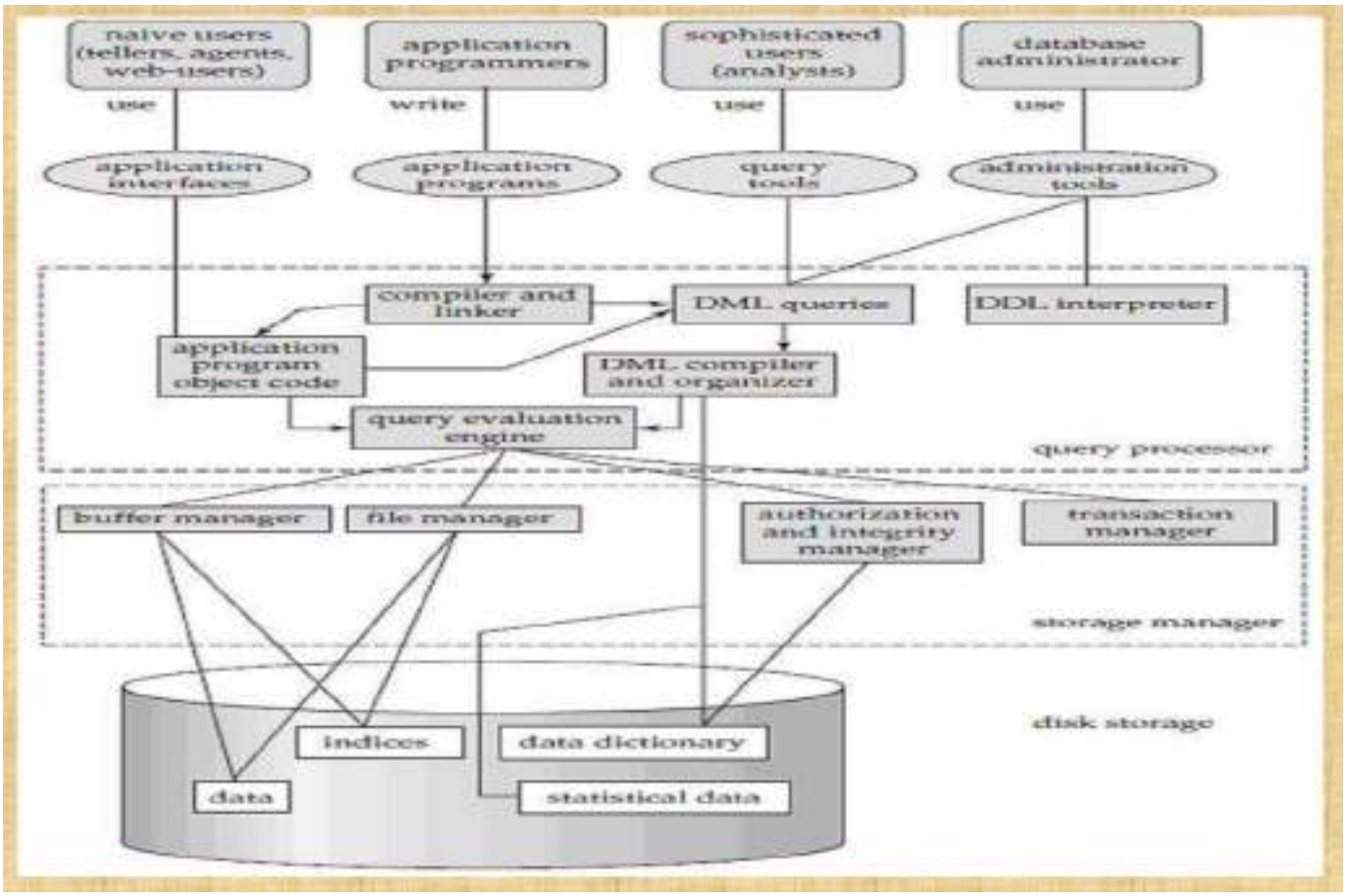
DATA INDEPENDENCE

The ability to modify the schema in one level without affecting the schema in next higher level is called data independence.

- **Logical data independence**: The ability to modify the logical schema without affecting the schema in next higher level (external schema.)
- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema

STRUCTURE OF DBMS

- A database system is partitioned into modules that deal with each of the responsibilities of the overall system.
- The functional components of a database system can be broadly divided into the storage manager and the query processor components.
- The storage manager is important because databases typically require a large amount of storage space.
- The query processor is important because it helps the database system to simplify and facilitate access to data.
- It is the job of the database system to translate updates and queries written in a nonprocedural language, at the logical level, into an efficient sequence of operations at the physical level.



1. **Query Processor:**
   It interprets the requests (queries) received from end user via an application program into instructions. It also executes the user request which is received from the DML compiler.

   Query Processor contains the following components

   - **DML Compiler –**
     It processes the DML statements into low level instruction (machine language), so that they can be executed.
   - **DDL Interpreter –**
     It processes the DDL statements into a set of table containing meta data (data about data).
   - **Embedded DML Pre-compiler –**
     It processes DML statements embedded in an application program into procedural calls.
   - **Query Optimizer –**
     It executes the instruction generated by DML Compiler.

2. **Storage Manager :**
   - Storage Manager is a program that provides an interface between the data stored in the database and the queries received.
   - It is also known as Database Control System.

- It maintains the consistency and integrity of the database by applying the constraints and executes the DCL statements.
- It is responsible for updating, storing, deleting, and retrieving data in the database.

It contains the following components –

- **Authorization Manager –**
  It ensures role-based access control, i.e. it checks whether the particular person is privileged to perform the requested operation or not.

- **Integrity Manager –**
  It checks the integrity constraints when the database is modified.

- **Transaction Manager –**
  It controls concurrent access by performing the operations in a scheduled way that it receives the transaction. Thus, it ensures that the database remains in the consistent state before and after the execution of a transaction.

- **File Manager –**
  It manages the file space and the data structure used to represent information in the database.

- **Buffer Manager –**
  It is responsible for cache memory and the transfer of data between the secondary storage and main memory.

3. Disk Storage:
It contains the following components –
Data Files –
It stores the data.

- **Data Dictionary –**
  It contains the information about the structure of any database object. It is the repository of information that governs the metadata.

- **Indices –**
  It provides faster retrieval of data item.

Types of Database Users

1. Database Administrator (DBA):
DBA Stands for Database Administrator.

- It is a person or a team, who is responsible for managing the overall database management system.

- It is the leader of the database. It is like a super-user of the system.
- It is responsible for the administration of all the three levels of the database.

DBA is responsible for:

- Deciding the instances for the database.
- Defining the Schema
- Liaising with Users
- Define Security
- Back-up and Recovery
- Monitoring the performance

2. Database Designers:

- Database designers design the appropriate structure for the database, where we share data.

3. System Analyst:

- System analyst analyses the requirements of end users, especially naïve and parametric end users.

4. Application Programmers:

- Application programmers are computer professionals, who write application programs.

5. Naïve Users / Parametric Users:

- Naïve Users are Un-sophisticated users, which has no knowledge of the database. These users are like a layman, which has a little bit of knowledge of the database.
- Naive Users are just to work on developed applications and get the desired result.
- For Example: Railway's ticket booking users are naive users. Or Clerical staff in any bank is a naïve user because they don't have any DBMS knowledge but they still use the database and perform their given task.

6. Sophisticated Users:

- Sophisticated users can be engineers, scientists, business analyst, who are familiar with the database. These users interact with the database but they do not write programs

7. Casual Users / Temporary Users:

- These types of users communicate with the database for a little period of time.

Introduction to Database Design

The **entity-relationship (ER) data model** allows us to describe the data involved in a real-world enterprise in terms of objects and their relationships and is widely used to develop an initial database design.

It provides useful concepts that allow us to move front an informal description of what users want from their database to a more detailed, precise description that can be implemented in a DBMS.

DATABASE DESIGN AND ER DIAGRAMS

The database design process can be divided into six steps. The ER model is most relevant to the first three steps.

1. **Requirements Analysis**: The very first step in designing a database application is to understand what data is to be stored in the database, what applications must be built on top of it, and what operations are most frequent and subject to performance requirements.
2. **Conceptual Database Design**: The information gathered in the requirements analysis step is used to develop a high-level description of the data to be stored in the database, along with the constraints known to hold over this data.
3. **Logical Database Design**: We must choose a DBMS to implement our databctse design, and convert the conceptual database design into a database schema in the data model of the chosen DBMS.

Beyond ER Design

4. **Schema Refinement**: The fourth step ill database design is to analyze the collection of relations in our relational database schema to identify potential problems, and to refine it.
5. **Physical Database Design**: In this step, we consider typical expected workloads that our database must support and further refine the database design to ensure that it meets desired performance criteria.
6. **Application and Security Design**: Any software project that involves a DBMS must consider aspects of the application that go beyond the database itself.

ENTITIES, ATTRIBUTES AND ENTITY SETS

In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes.

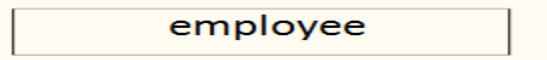
Basic Concepts of ER Model in DBMS

As we described in the tutorial Database models, Entity-relationship model is a model used for design and representation of relationships between data.

The main data objects are termed as Entities, with their details defined as attributes, some of these attributes are important and are used to identify the entity, and different entities are related using relationships.

- An **entity** is an object that exists and is distinguishable from other objects.
  - Examples:
    - Person: PROFESSOR, STUDENT
    - Place: STORE, UNIVERSITY
    - Object: MACHINE, BUILDING
    - Event: SALE, REGISTRATION

- An **entity** is represented with a RECTANGLE



- Entities have **attributes**

  Example: people have *names* and *addresses*

  - An **entity set** is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, trees, holidays

ATTRIBUTES

- An entity is represented by a set of attributes that is descriptive properties possessed by all members of an entity set.

  Example:

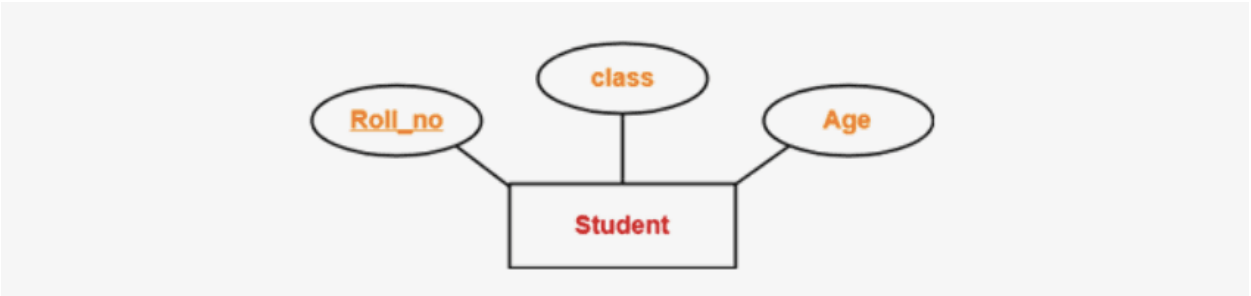
  - *customer* = (*Customer_id, name, street, city, salary*)
  - *movie*= (*title, director, written by, duration, release date*)

Attributes are represented with **ELLIPSE**

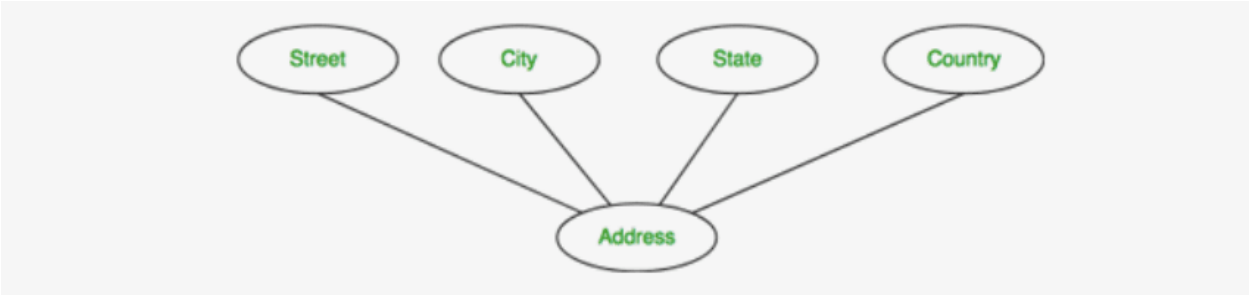
Use **LINES** to link attributes to entities

An attribute can be of many types, here are different types of attributes defined in ER database model:

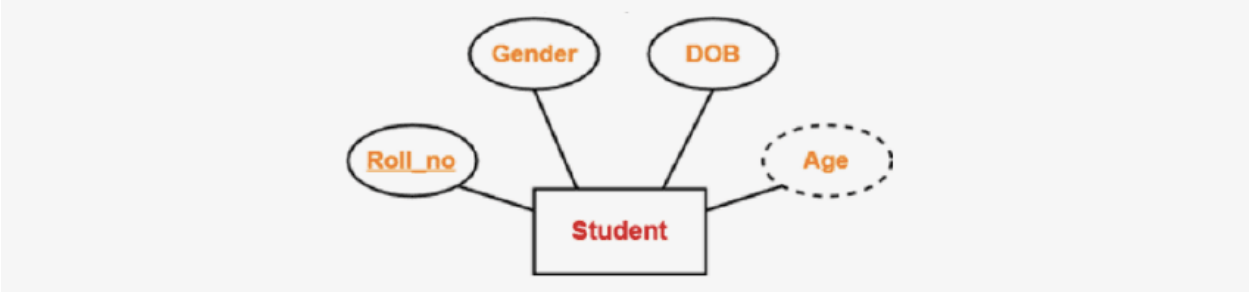
1. **Simple attribute**: The attributes with values that are atomic and cannot be broken down further are simple attributes. For example, student's age.



2. **Composite attribute**: A composite attribute is made up of more than one simple attribute. For example, student's address will contain, house no., street name, pin code etc.

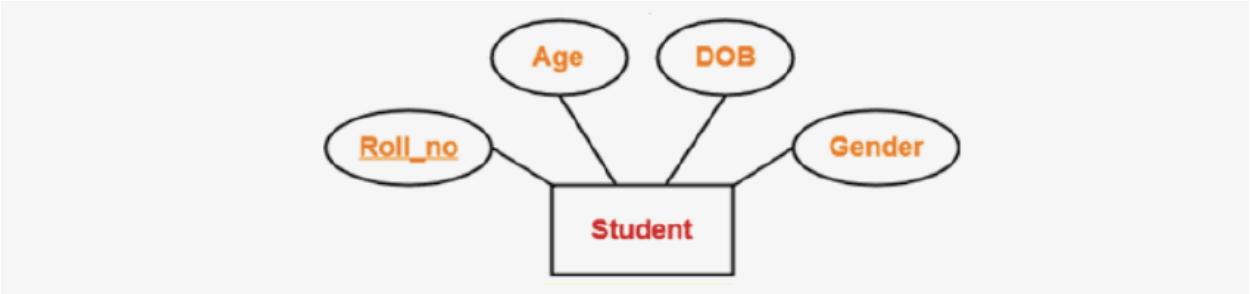


3. **Derived attribute**: These are the attributes which are not present in the whole database management system, but are derived using other attributes. For example, *average age of students in a class*.

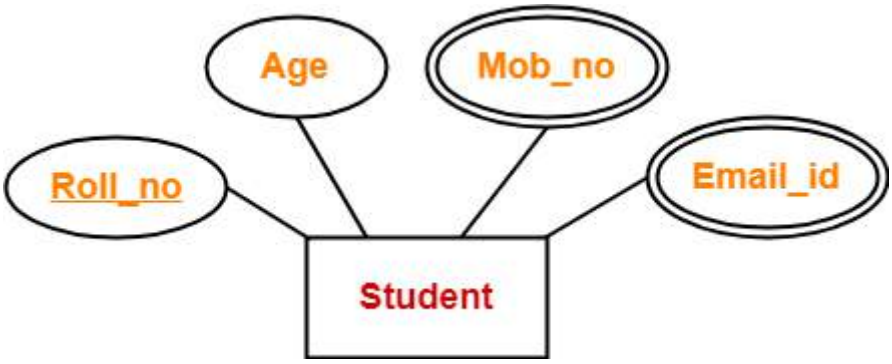


Example: age, and its value is derived from the stored attribute Date of Birth.

4. **Single-valued attribute**: As the name suggests, they have a single value.

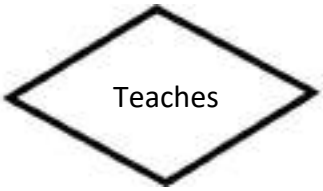


5. **Multi-valued attribute**: They can have multiple values.

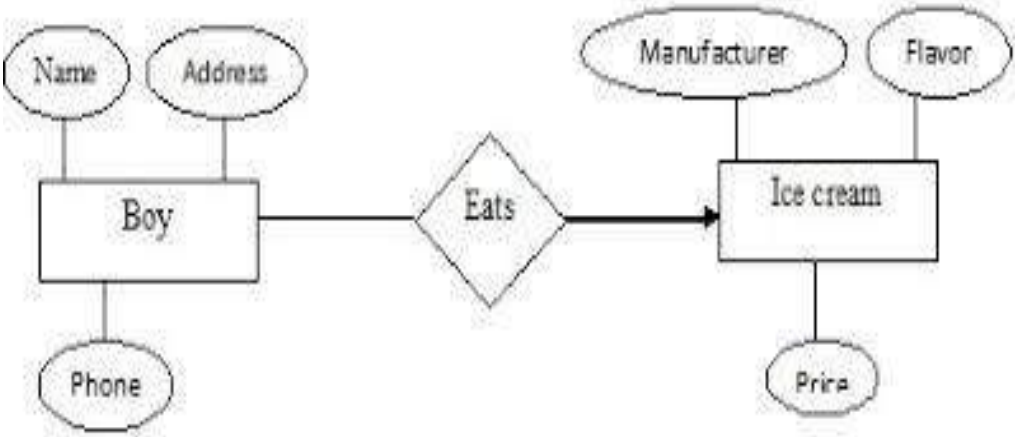


Relationship

- Named set of all similar relationships with the same attributes and relating to the same entity types
- Relationship is represented with DIAMOND



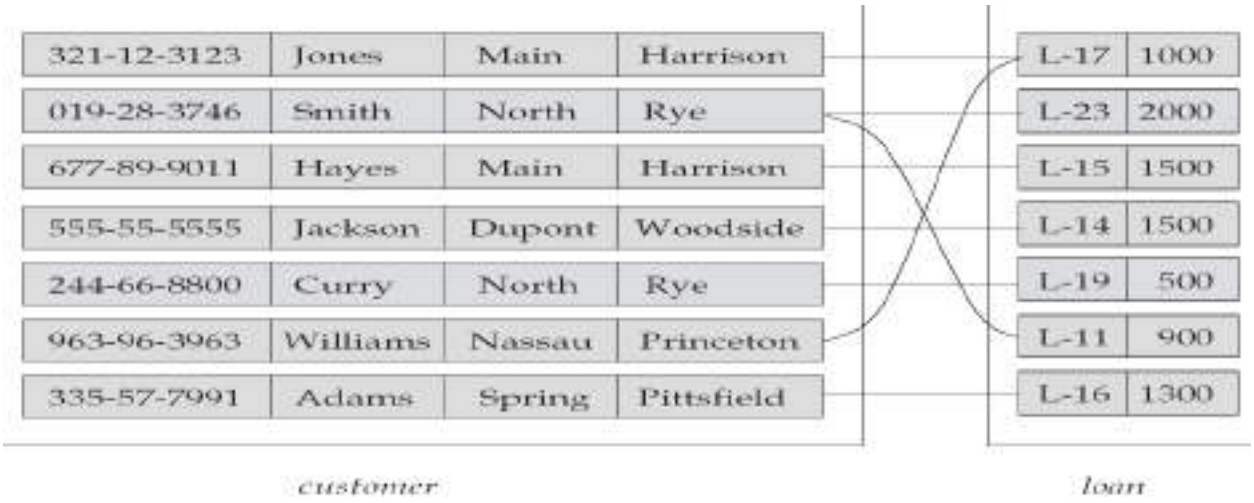
- Relationships relate entities within the entity sets involved in the relationship type to each other.

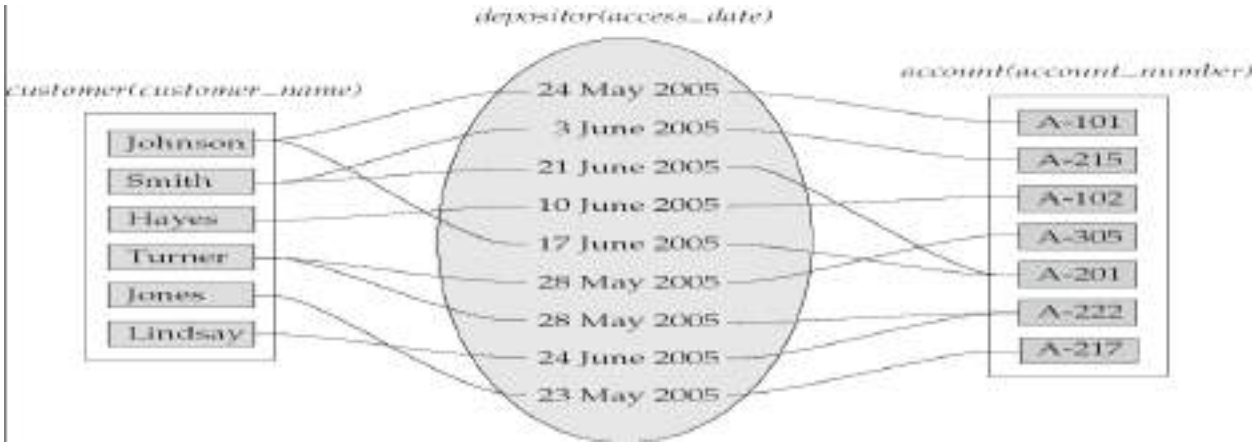


ER – Relationships

Relationship Set: Collection of similar relationships.

- An attribute can also be property of a relationship set.For instance, the depositor relationship set between entity sets customer and account may have the attribute access-date



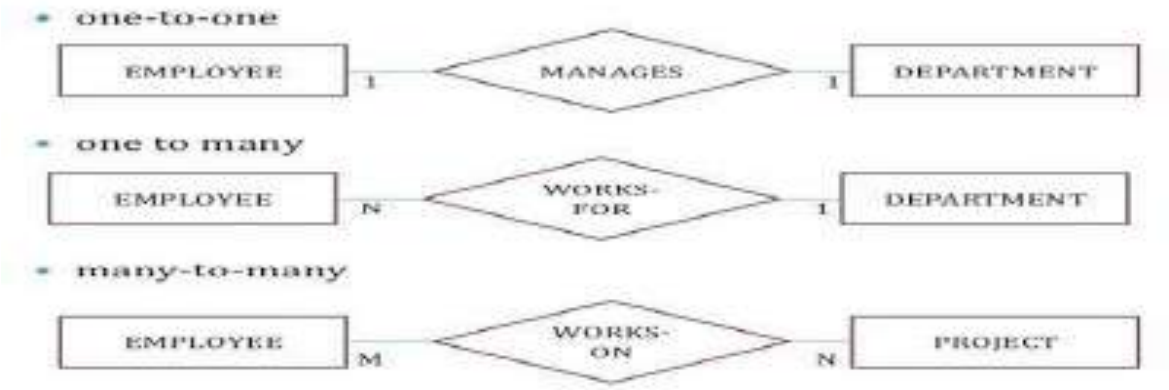


Mapping Cardinality

Types of Relationships

- Three types of relationships can exist between entities
- One-to-one relationship (1:1): One instance in an entity (parent) refers to one and only one instance in the related entity (child).
- One-to-many relationship (1:M): One instance in an entity (parent) refers to one or more instances in the related entity (child)

- Many-to-many relationship (M:N): exists when one instance of the first entity (parent) can relate to many instances of the second entity (child), and one instance of the second entity can relate to many instances of the first entity.



ER Diagram: Relationship

A Relationship describes relation between entities. Relationship is represented using diamonds or rhombus.

There are three types of relationship that exist between Entities.

1. Binary Relationship
2. Recursive Relationship
3. Ternary Relationship

1.**Binary Relationship**

Binary Relationship means relation between two Entities. This is further divided into three types.

One to One Relationship

This type of relationship is rarely seen in real world

The above example describes that one student can enroll only for one course and a course will also have only one Student. This is not what you will usually see in real-world relationships.

One to Many Relationships

The below example showcases this relationship, which means that 1 student can opt for many courses, but a course can only have 1 student. Sounds weird! This is how it is.

Many to One Relationship

It reflects business rule that many entities can be associated with just one entity. For example, Student enrolls for only one Course but a Course can have many Students.

Many to Many Relationships

A many-to-many relationship occurs when multiple records in a table are associated with multiple records in another table.

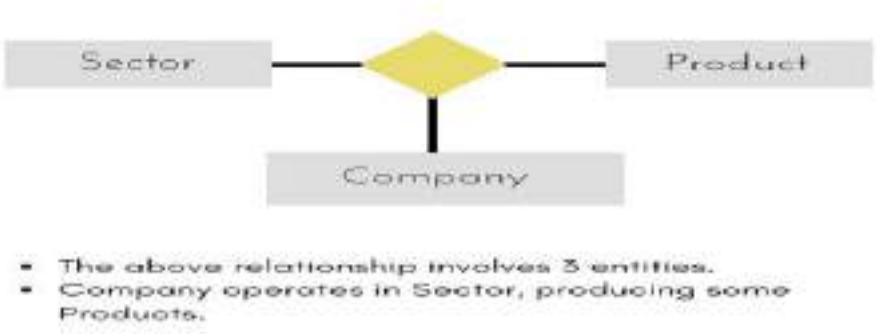
2.**Recursive Relationship**

When an Entity is related with itself it is known as Recursive Relationship.

3. Ternary Relationship

Relationship of degree three is called Ternary relationship.

A Ternary relationship involves three entities. In such relationships we always consider two entities together and then look upon the third.



Strong Entity

The Strong Entity is the one whose existence does not depend on the existence of any other entity in a schema. It is denoted by a single rectangle. A strong entity always has the primary key in the set of attributes that describes the strong entity. It indicates that each entity in a strong entity set can be uniquely identified.

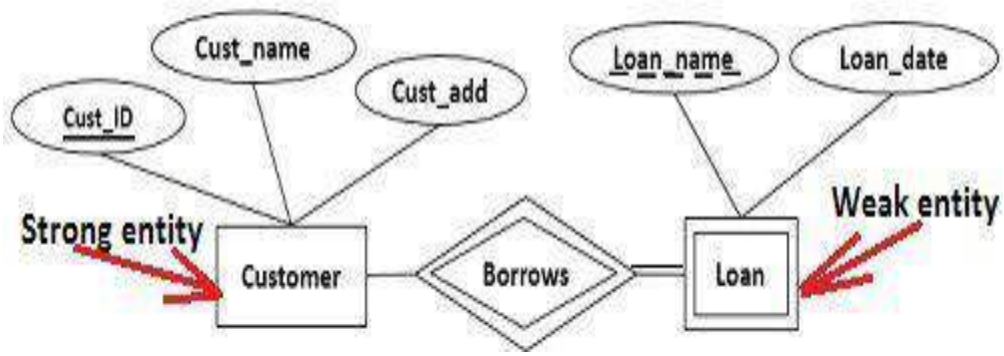
Set of similar types of strong entities together forms the Strong Entity Set. A strong entity holds the relationship with the weak entity via an Identifying Relationship, which is denoted by double diamond in the ER diagram. On the other hands, the relationship between two strong entities is denoted by a single diamond and it is simply called as a relationship.

Weak Entity

A Weak entity is the one that depends on its owner entity i.e. a strong entity for its existence. A weak entity is denoted by the double rectangle. Weak entities do not have the primary key instead it has a partial key that uniquely discriminates the weak entities. The primary key of a

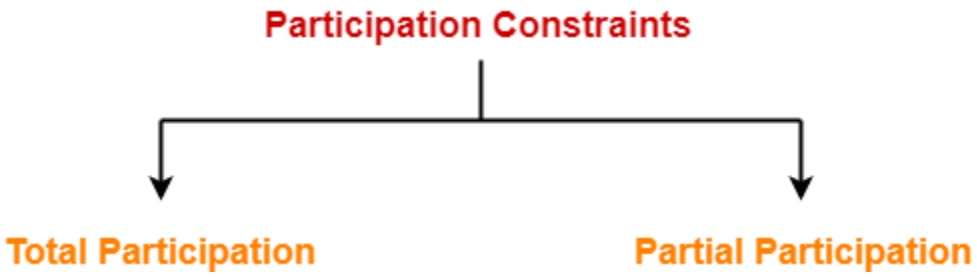
weak entity is a composite key formed from the primary key of the strong entity and partial key of the weak entity.

The collection of similar weak entities is called Weak Entity Set. The relationship between a weak entity and a strong entity is always denoted with an Identifying Relationship i.e. double diamond.



Types of Participation Constraints

There are two types of participation constraints-

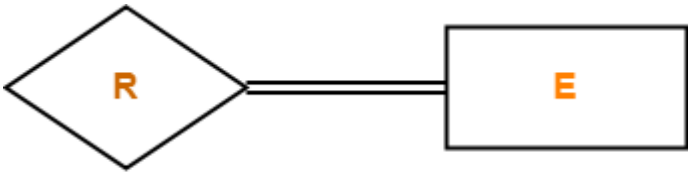


Total participation

Partial participation

1. Total Participation

- It specifies that each entity in the entity set must compulsorily participate in at least one relationship instance in that relationship set.
- That is why, it is also called as mandatory participation.
- Total participation is represented using a double line between the entity set and relationship set.

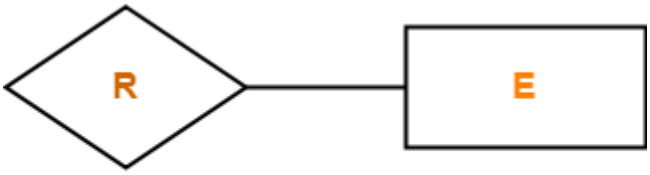


Total Participation

- Double line between the entity set “Student” and relationship set “Enrolled in” signifies total participation.
- It specifies that each student must be enrolled in at least one course.

2. Partial Participation

- It specifies that each entity in the entity set may or may not participate in the relationship instance in that relationship set.
- That is why, it is also called as **optional participation.**
- Partial participation is represented using a single line between the entity set and relationship set.



Partial Participation

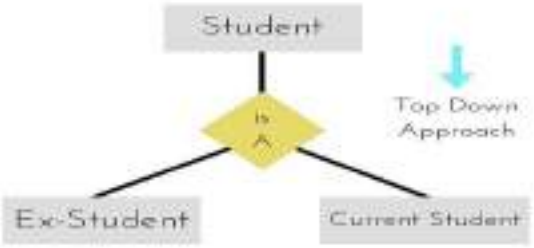
- Single line between the entity set “Course” and relationship set “Enrolled in” signifies partial participation.
- It specifies that there might exist some courses for which no enrollments are made.

ADDITIONAL FEATURES OF THE ER MODEL

1. Generalization – generalization is relationship that exist between higher level entity set and one or more lower level entity sets. Generalization synthesizes these entity sets into single entity set.

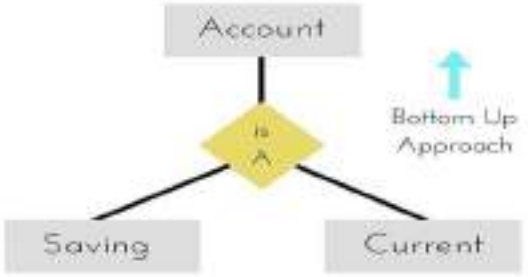
ISA (`is a') Hierarchies is used to represent Generalization

ISA (is a) referred as a superclass-subclass relationship.



2.Specialization – The process of designating to sub grouping within an entity set is called specialization.

ISA (`is a') Hierarchies is used to represent Specialization
ISA (is a) referred as a superclass-subclass relationship.



3.Aggregation: there is a one limitation with E-R model that it cannot express relationships among relationships. So aggregation is an abstraction through which relationship is treated as higher level entities.

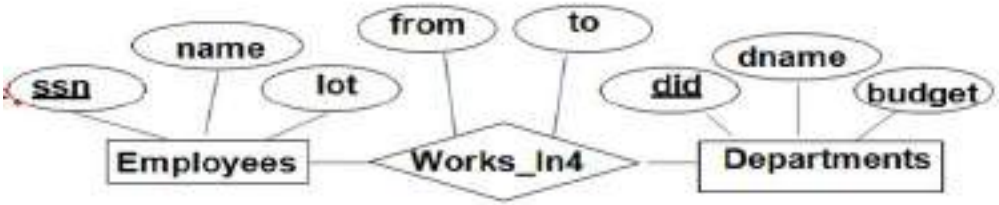
Conceptual Design Using the ER Model

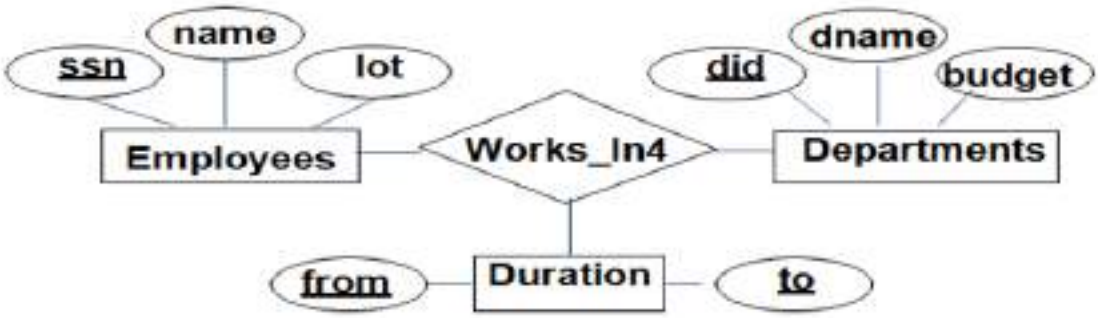
a. Entity vs. Attribute

- 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).

- Works_In4 does not allow an employee to work in a department for two or more periods.
- Similar to the problem of wanting to record several addresses for an employee:

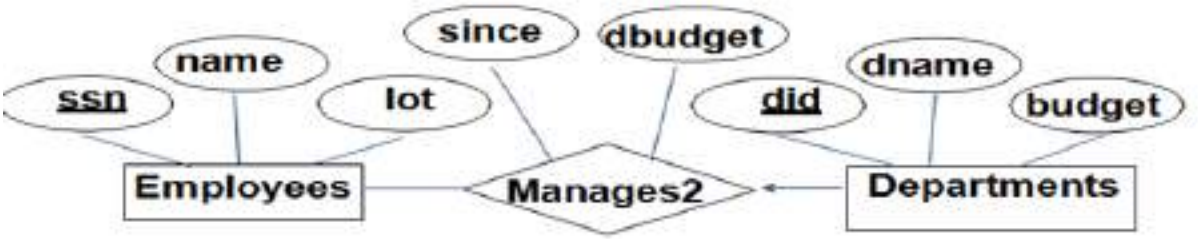
  We want to record *several values of the descriptive attributes for each instance of this relationship.* Accomplished by introducing new entity set, Duration.

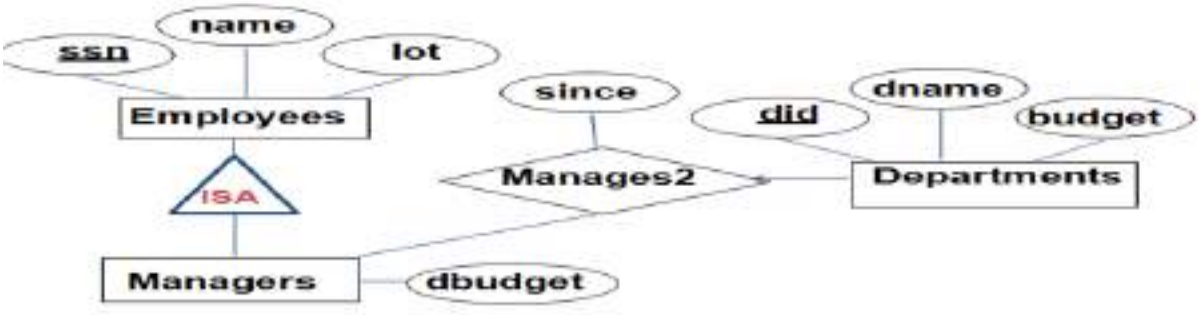




b. Entity vs. Relationship

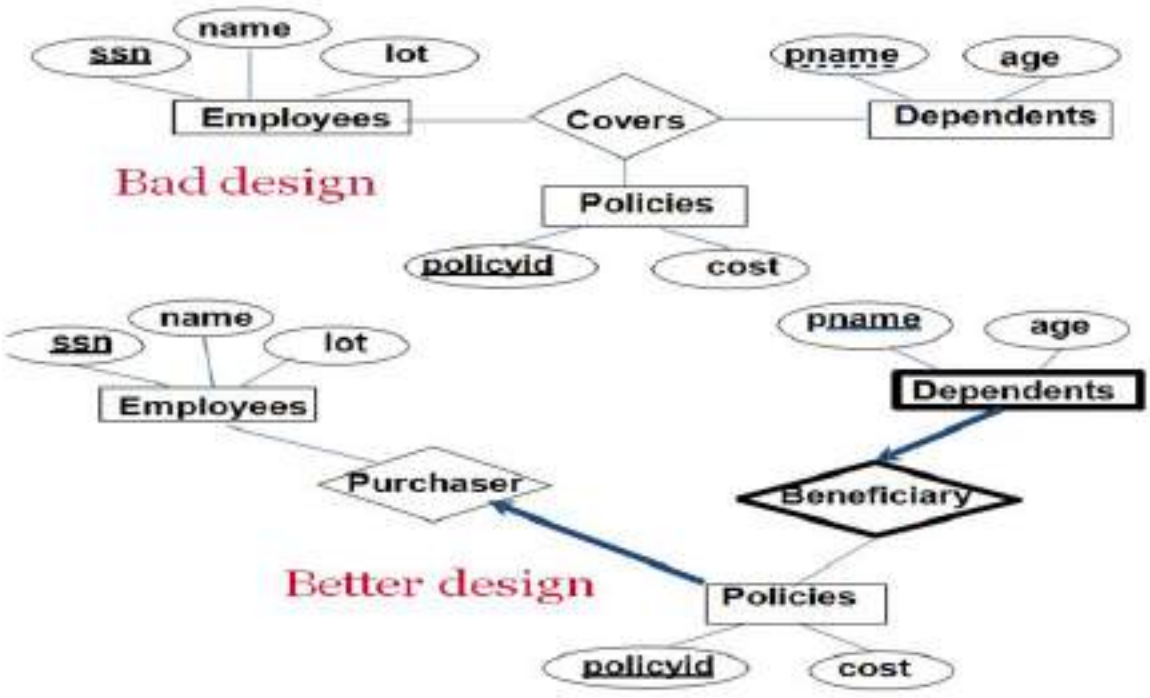
- First ER diagram OK if a manager gets a separate discretionary budget for each dept.
- What if a manager gets a discretionary budget that covers *all* managed depts?
  - **Redundancy:** *dbudget* stored for each dept managed by manager
  - **Misleading:** Suggests dbudget associated with department- mgr combination**.**





c. Binary vs. Ternary Relationships

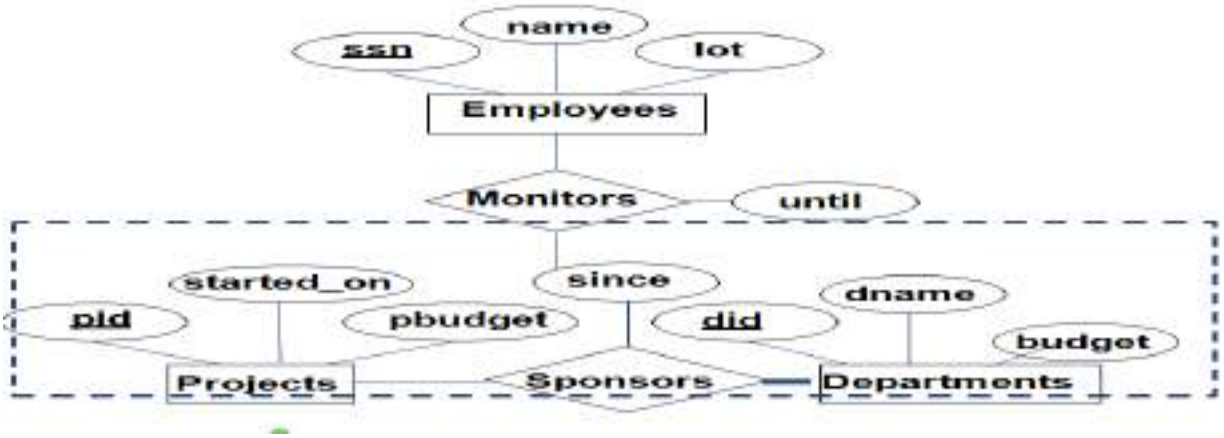
- If each policy is owned by just 1 employee, and each dependent is tied to the covering policy, first diagram is inaccurate.
- What are the additional constraints in the2nd diagram?



- Previous example illustrated a case when two binary relationships were better than one ternary relationship.
- An example in the other direction: a ternary relation Contracts relates entity sets Parts, Departments and Suppliers, and has descriptive attribute *qty*.
  - S “can-supply” P, D“needs” P, and D“deals- with” S does not imply that D has agreed to buy P from S.

d. Aggregation v/s ternary relationship

- The choice between using aggregation or ternary relationship is mainly determined by existence of a relationship that relates relationship set to entity set.
- The choice may also be guided by certain integrity constraints that we want to express.



Aggregation v/s ternary relationship

Using ternary relationship instead of aggregation