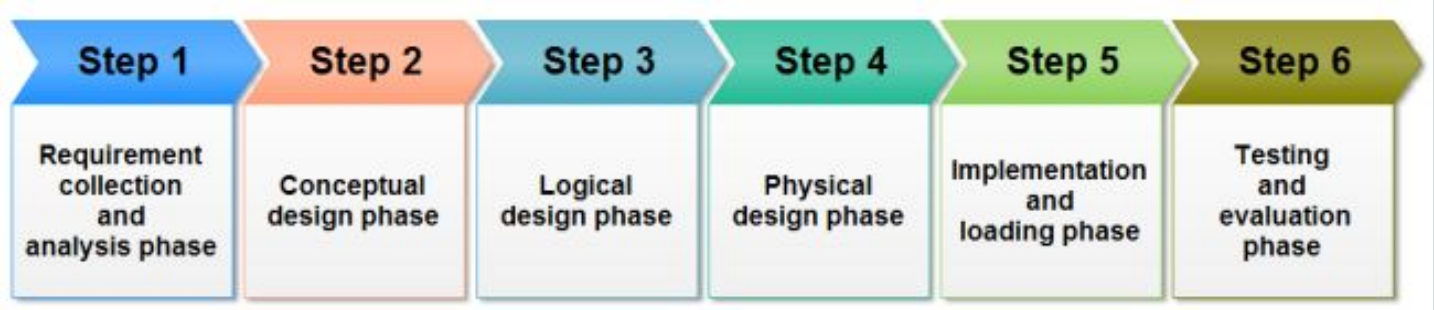
The E-R Model

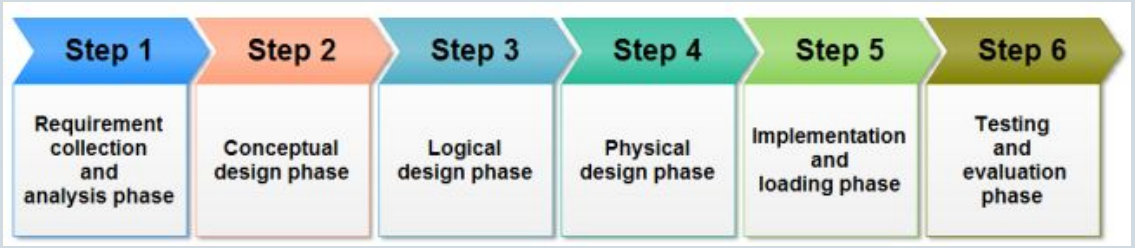
Database Design Process



Step # 1: Requirements collection and analysis

- During this step, the database designers interview prospective database users to understand and document their data requirements.
- The result of this step is a **concisely written set of users' requirements**. These requirements should be specified in as detailed and complete a form as possible

Step # 2: Conceptual Design Phase



- Once the requirements have been collected and analyzed, the next step is to create a conceptual schema of the database, using conceptual data model – **ER diagram**.
- The conceptual schema is a concise description of the data requirements of the users.
- **Does not include implementation details of the database -** enables database designers to concentrate on specifying the properties of the data, without being concerned with storage and implementation details.

Step # 3: Logical Design Phase

- The conceptual schema is transformed from the high-level data model into the implementation data model.
- Result is a database schema in the implementation data model of the DBMS.

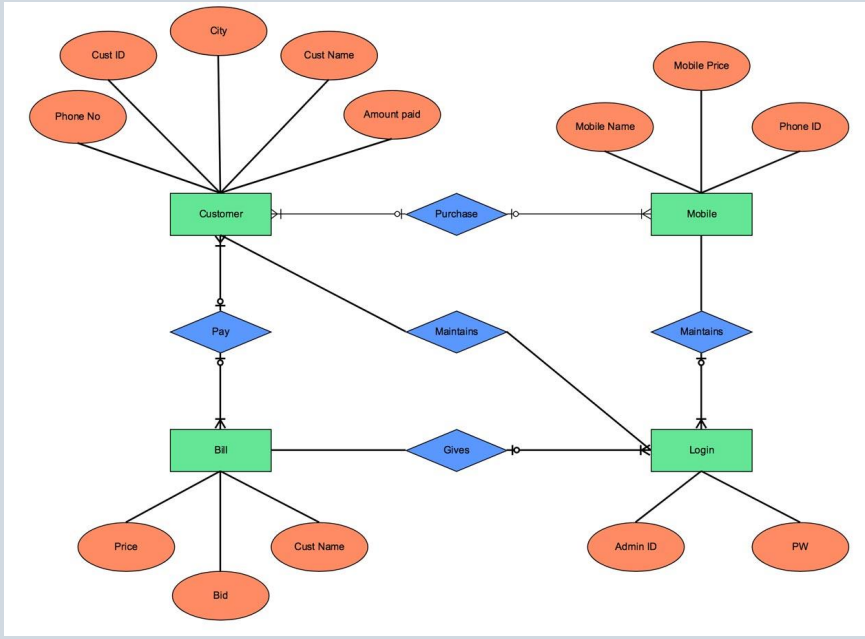
Step # 4: Physical Design Phase:

- The last step in the database design process.
- Involves the construction of a database according to the specification of a logical schema.
- In parallel with these activities, application programs are designed and implemented as database transactions corresponding to the high-level transaction specifications.

ER MODEL - Introduction

- The **Entity-Relationship Data Model**, also called **ER**, **is one of the *conceptual data model*** (not linked to any particular implementation).
- Is a graphical representation of the database structure.
- Is a blueprint that can be used to create a description of the database.
- At first look, an ER diagram looks very similar to the flowchart.
- However, ER Diagram includes many specialized symbols,

  and its meanings make this model unique.



ER MODEL

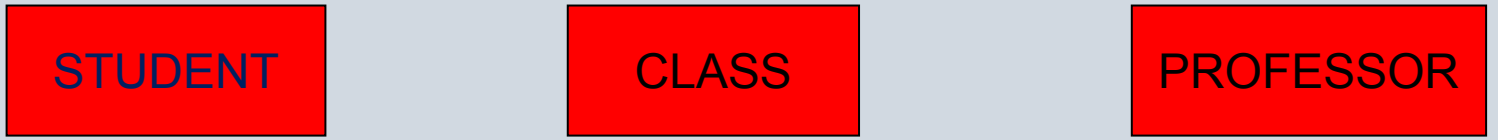
- The **E-R** (entity-relationship) data model views the real world as a set of basic **objects** (entities) and **relationships** among these objects.
- It is intended primarily for the DB design process this represents the overall **logical structure of the DB**.
- Is a detailed, logical representation of the data for an organization or for a business area.
- Expressed in terms of:
  - **Entities**
  - **Attributes**
  - **Relationships**

Entities and Entity Sets

- Entities are specific objects or things in the mini-world that are represented in the database
- An **entity** is an object that exists and is distinguishable from other objects - has its own identity that distinguishes it from other entities.

Examples: PROFESSOR, STUDENT, STORE, UNIVERSITY, MACHINE, BUILDING, SALE, REGISTRATION, ACCOUNT, COURSE etc

- An entity may be **concrete** (a person or a book, for example) or **abstract** (like a holiday or a disease or a concept).
- **Entities should always be placed in a rectangle.**



Entities and Entity Sets

- Entities with the same basic attributes are grouped or typed into an **entity type.**

  For example, the EMPLOYEE entity type or the PROJECT entity type.

- An entity set is a set of entities of the same type.

  Example: set of all persons, companies, trees, holidays, books, all persons having an account at a bank etc.

- An entity type is a collection of entities that share common properties or characteristics.

- **Entity sets need not be disjoint**. For example, the entity set Student (all students in a university) and the entity set professor (all professors in a university) may have members in common. (i.e., a computer science professor might take a class in anthropology).

ATTRIBUTES

- Each entity has a set of attributes ; i.e., descriptive properties possessed by all members of an entity set.

  Example: instructor = (ID, name, street, city, salary)

  course= (course_id, title, credits)

- **Attributes are properties used to describe an entity**.

  For example an **EMPLOYEE entity** may have a **Name, SSN, Address, Sex, BirthDate**

- **A specific entity will have a value for each of its attributes**.

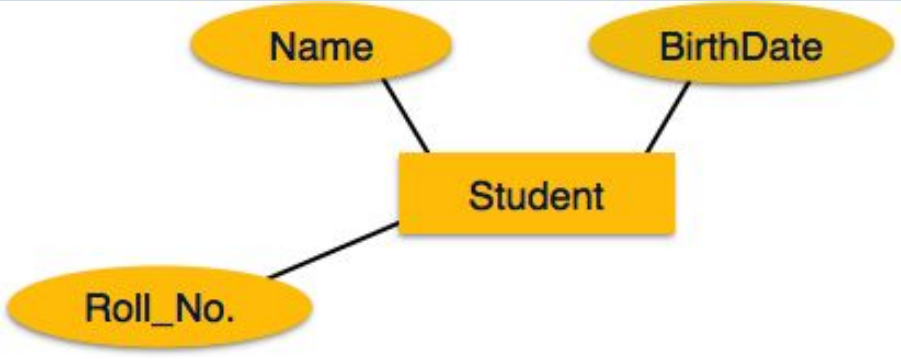
  For example a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Sex='M', BirthDate='09-JAN-55‘

- Each attribute has a *value set* (or data type) associated with it – e.g. integer, string, subrange, enumerated type, …

ATTRIBUTES

- In ER diagram, attribute is represented by an oval.

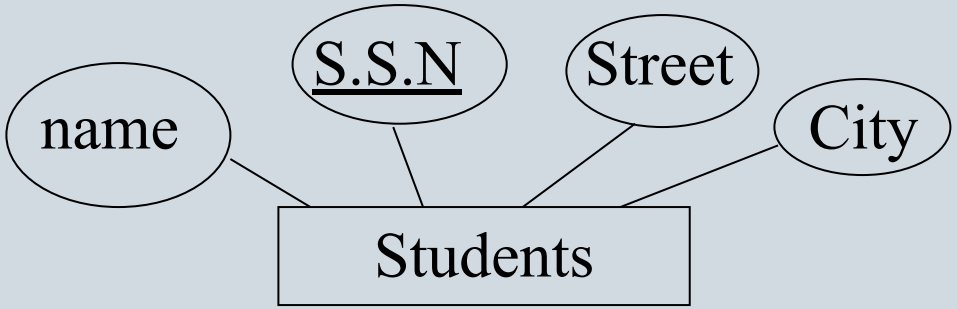
- Every ellipse represents one attribute and is directly connected to its entity (rectangle).

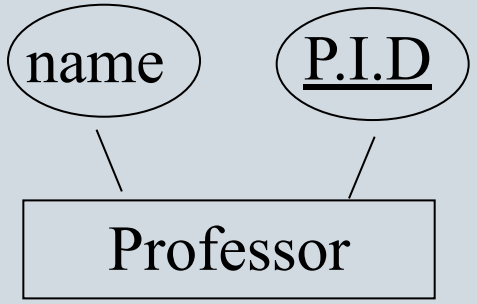


TYPES OF ATTRIBUTES

1) Simple V/s Composite

- Attributes that are **not divisible** are called simple or atomic attributes.
- Each entity has a single atomic value for the attribute. For example, SSN or Sex.



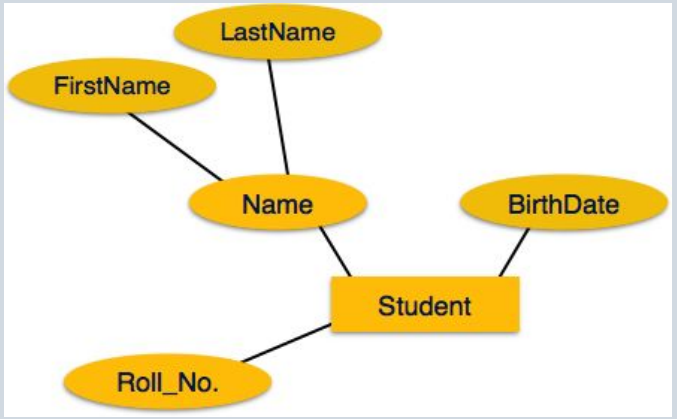


TYPES OF ATTRIBUTES

1) Simple V/s Composite

Composite attributes can be divided into smaller subparts, which represent more basic attributes with independent meanings.

- For example, the Name attribute of the STUDENT entity can be subdivided into FirstName and LastName.

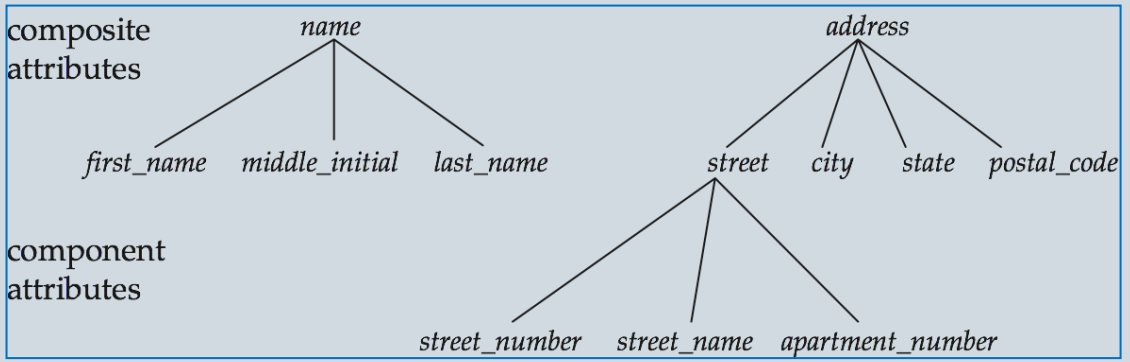


TYPES OF ATTRIBUTES

1) Simple V/s Composite

- Composite attributes can form a hierarchy (where some components are themselves composite);

  For example, Street_address can be further subdivided into three simple component attributes: Number, Street, and Apartment_number, as shown below. The value of a composite attribute is the concatenation of the values of its component simple attributes.



TYPES OF ATTRIBUTES

2) Single-Valued V/s Multivalued Attributes

- Most attributes have a single value for a particular entity; such attributes are called **single-valued.**

  For example, Age is a single-valued attribute of a person.

- In some cases an attribute can have a set of values for the same entity—such attributes are called **multivalued**.

  For example, a Colors attribute for a car, or a College_degrees attribute for a person, a phone_number attribute for a household

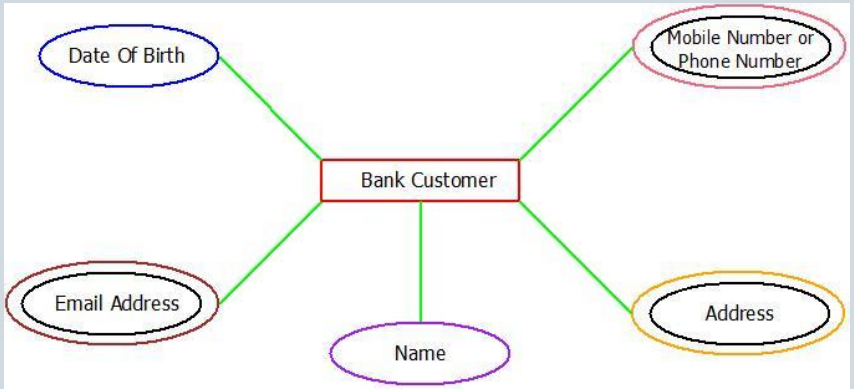
  - A person may have several college degrees.
  - A household may have several phones with different numbers.
  - A car color.

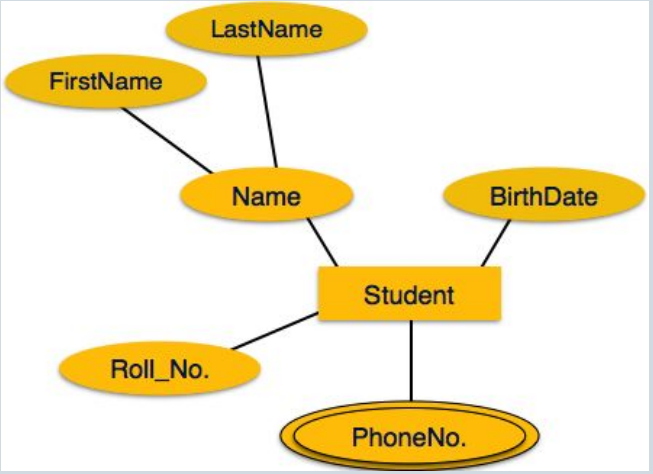
TYPES OF ATTRIBUTES

- A multivalued attribute may have lower and upper bounds to constrain the number of values allowed for each individual entity.

For example, the Colors attribute of a car may be restricted to have between one and three values, if we assume that a car can have three colors at most

- **<u>Multivalued attributes are depicted by double ellipse.</u>**





TYPES OF ATTRIBUTES

3) Stored versus Derived Attributes

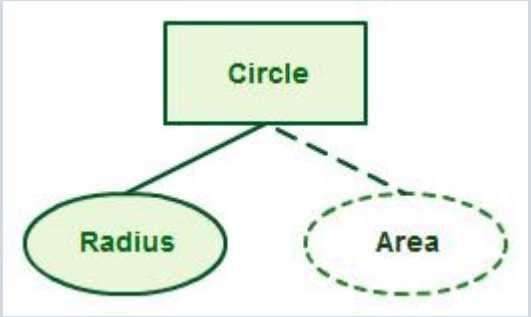
- In some cases, two (or more) attribute values are related and some attribute values can be derived from related entities.

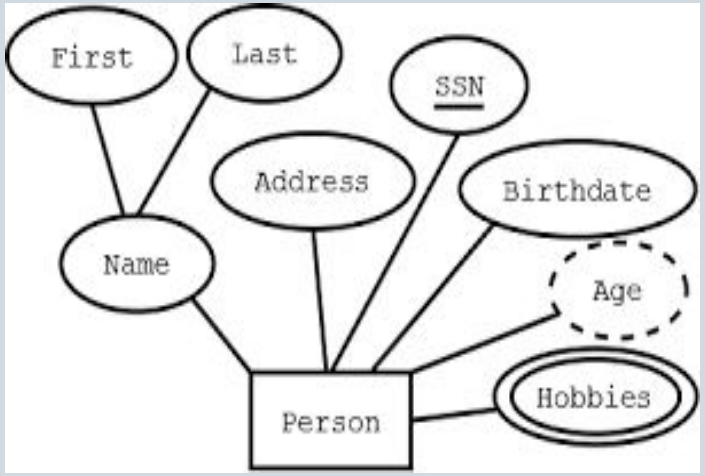
  For example, the Age and Birth_date attributes of a person.

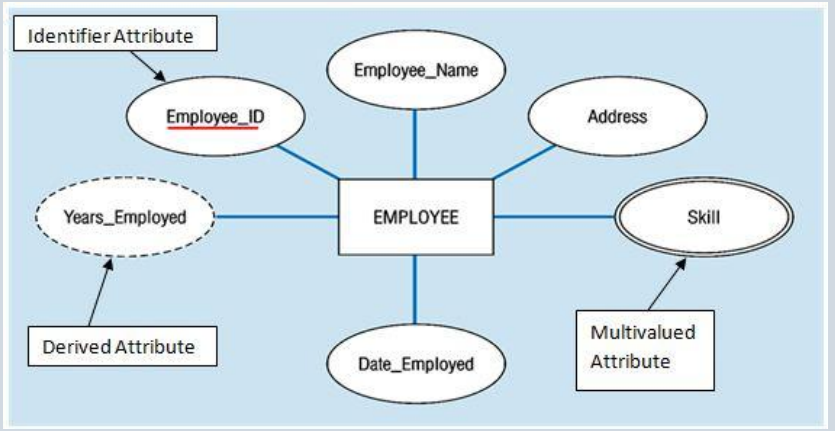
- A derived attribute is not physically stored within the database.
- **In ER diagram, derived attribute is represented by dashed oval.**

TYPES OF ATTRIBUTES

3) Stored versus Derived Attributes







Types of Attributes	Description
Simple attribute	Simple attributes can't be divided any further. For example, a student's contact number. It is also called an atomic value.
Composite attribute	It is possible to break down composite attribute. For example, a student's full name may be further divided into first name, second name, and last name.
Derived attribute	This type of attribute does not include in the physical database. However, their values are derived from other attributes present in the database. For example, age should not be stored directly. Instead, it should be derived from the DOB of that employee.
Multivalued attribute	Multivalued attributes can have more than one values. For example, a student can have more than one mobile number, email address, etc.

ATTRIBUTES

4) Key Attributes of an Entity Type

- A key attribute can uniquely identify an entity from an entity set.

  For example, student roll number can uniquely identify a student from a set of students.

Key attribute is represented by oval same as other attributes however the text of key attribute is underlined.

- An important constraint on the entities of an entity type is the key or uniqueness constraint on attributes.
- An entity type usually has one or more attributes whose values are distinct for each individual entity in the entity set. Such an attribute is called a key attribute, and its values can be used to identify each entity uniquely.

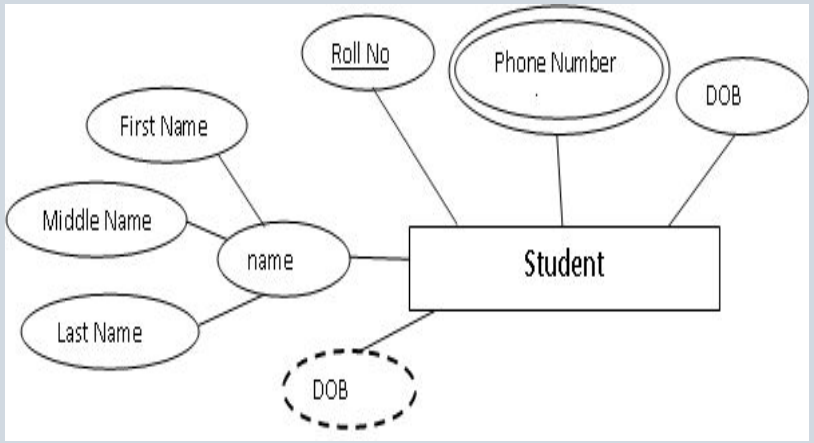
TYPES OF ATTRIBUTES

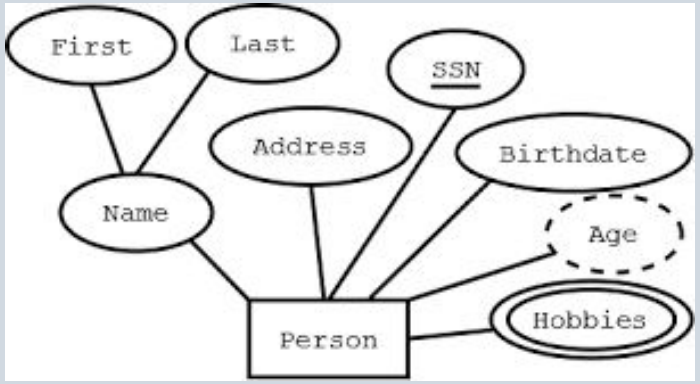
4) Key Attributes of an Entity Type

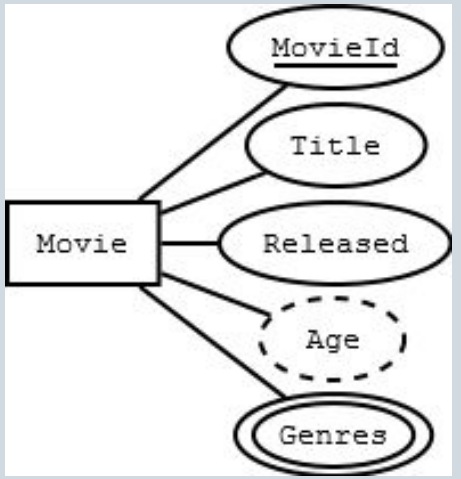
- **A key is an attribute in ER diagrams whose values are distinct for each individual entity in an entity set.**
- An entity type usually has one or more attributes whose values are distinct for each individual entity in the entity set. Such an attribute is called a key attribute, and its values can be used to identify each entity uniquely.
- A key attribute can uniquely identify an entity from an entity set.
  - For example for a **student entity set** Roll Number or Registration Number can be a key because it can uniquely identify a student from a set of students.
- An important constraint on the entities of an entity type is the key or uniqueness constraint on attributes.
- **Key attribute is represented by oval same as other attributes however the text of key attribute is underlined**.

ATTRIBUTES

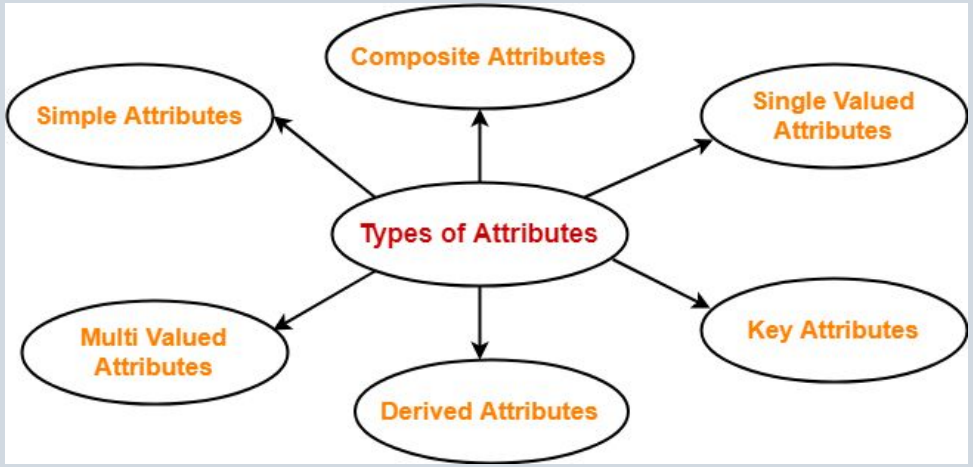
4) Key Attributes of an Entity Type

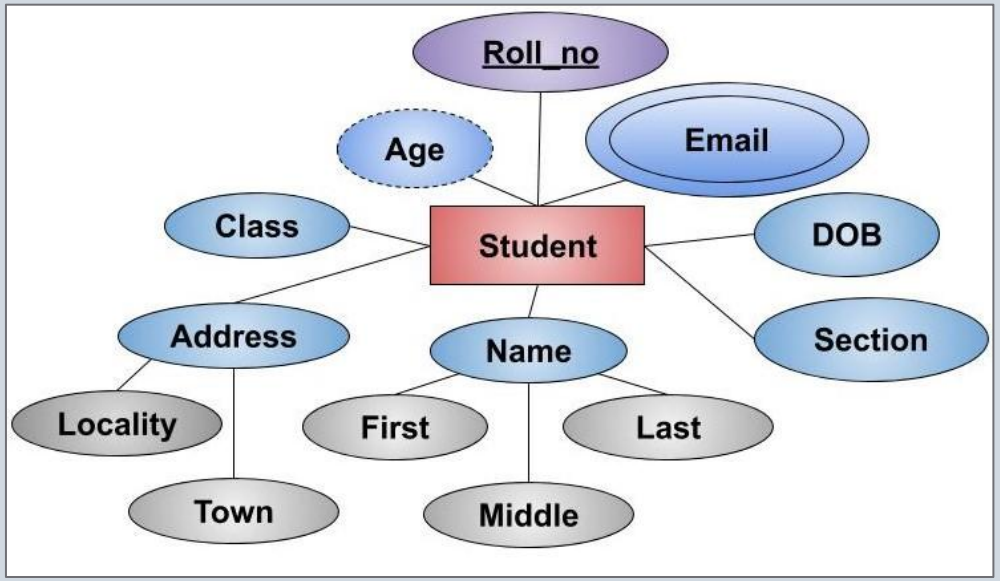






TYPES OF ATTRIBUTES - EXAMPLE

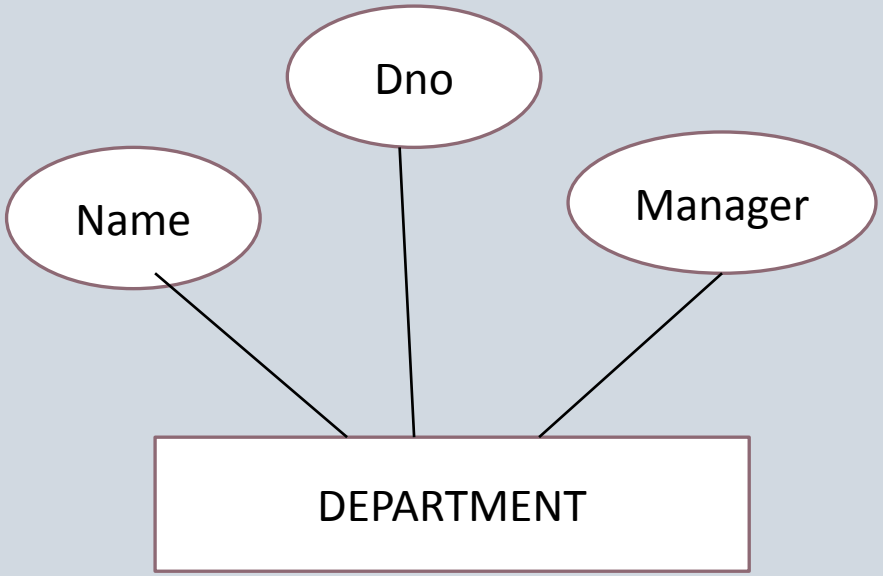


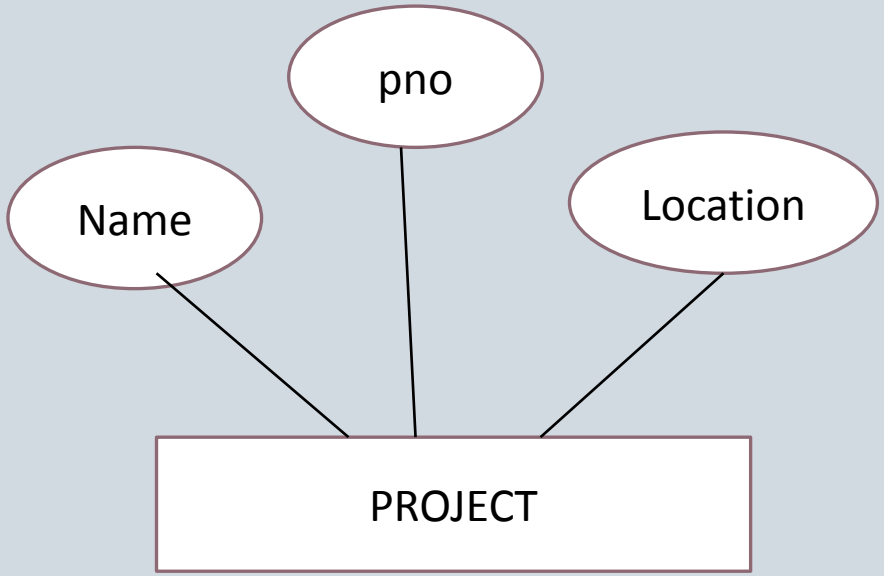


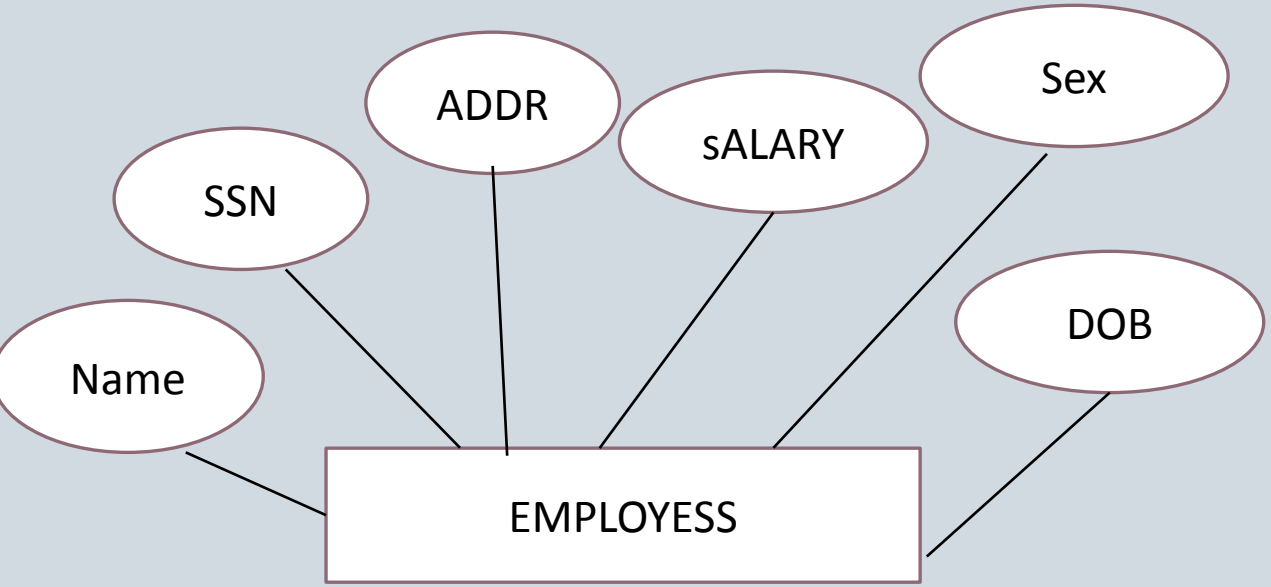
EXAMPLE

Requirements of the Company (oversimplified for illustrative purposes)

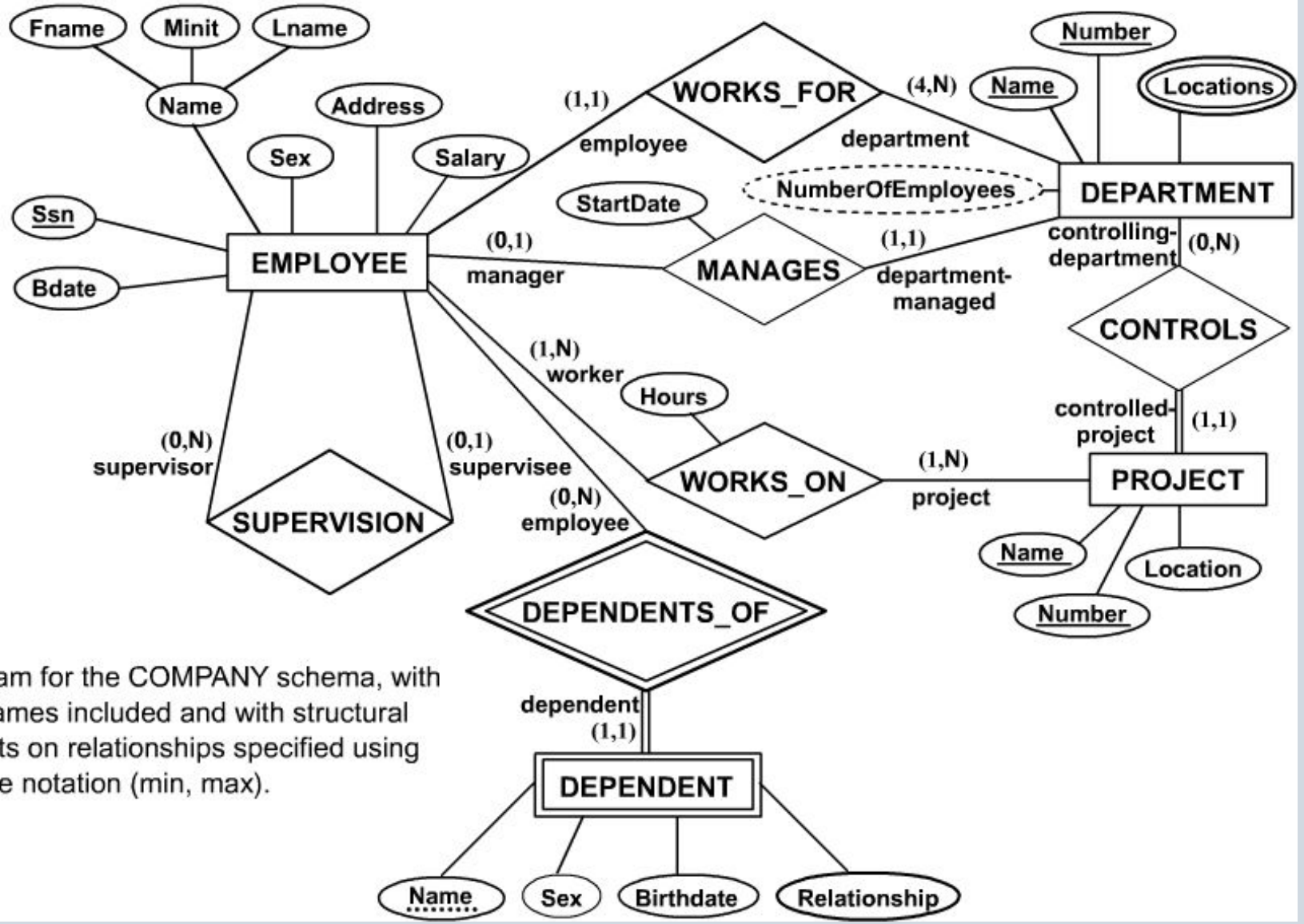
- The company is organized into DEPARTMENTs. Each department has a name, number and an employee who *manages* the department. We keep track of the start date of the department manager.
- Each department *controls* a number of PROJECTs. Each project has a name, number and is located at a single location.
- We store each EMPLOYEE's social security number, address, salary, sex, and birthdate. Each employee *works for* one department but may *work on* several projects. We keep track of the number of hours per week that an employee currently works on each project. We also keep track of the *direct supervisor* of each employee.
- Each employee may *have* a number of DEPENDENTs. For each dependent, we keep track of their name, sex, birthdate, and relationship to employee.


Dno
Name
Manager
DEPARTMENT


pno
Name
Location
PROJECT


ADDR
Sex
sALARY
SSN
DOB
Name
EMPLOYESS

Alternative ER Notations



ER diagram for the COMPANY schema, with all role names included and with structural constraints on relationships specified using alternative notation (min, max).