Data Model

- A database model shows the logical structure of a database, including the relationships and constraints that determine how data can be stored and accessed.
- Data models define how the logical structure of a database is modeled.
- Data Models are fundamental entities to introduce abstraction in a DBMS.
- Data models define how data is connected to each other and how they are processed and stored inside the system.

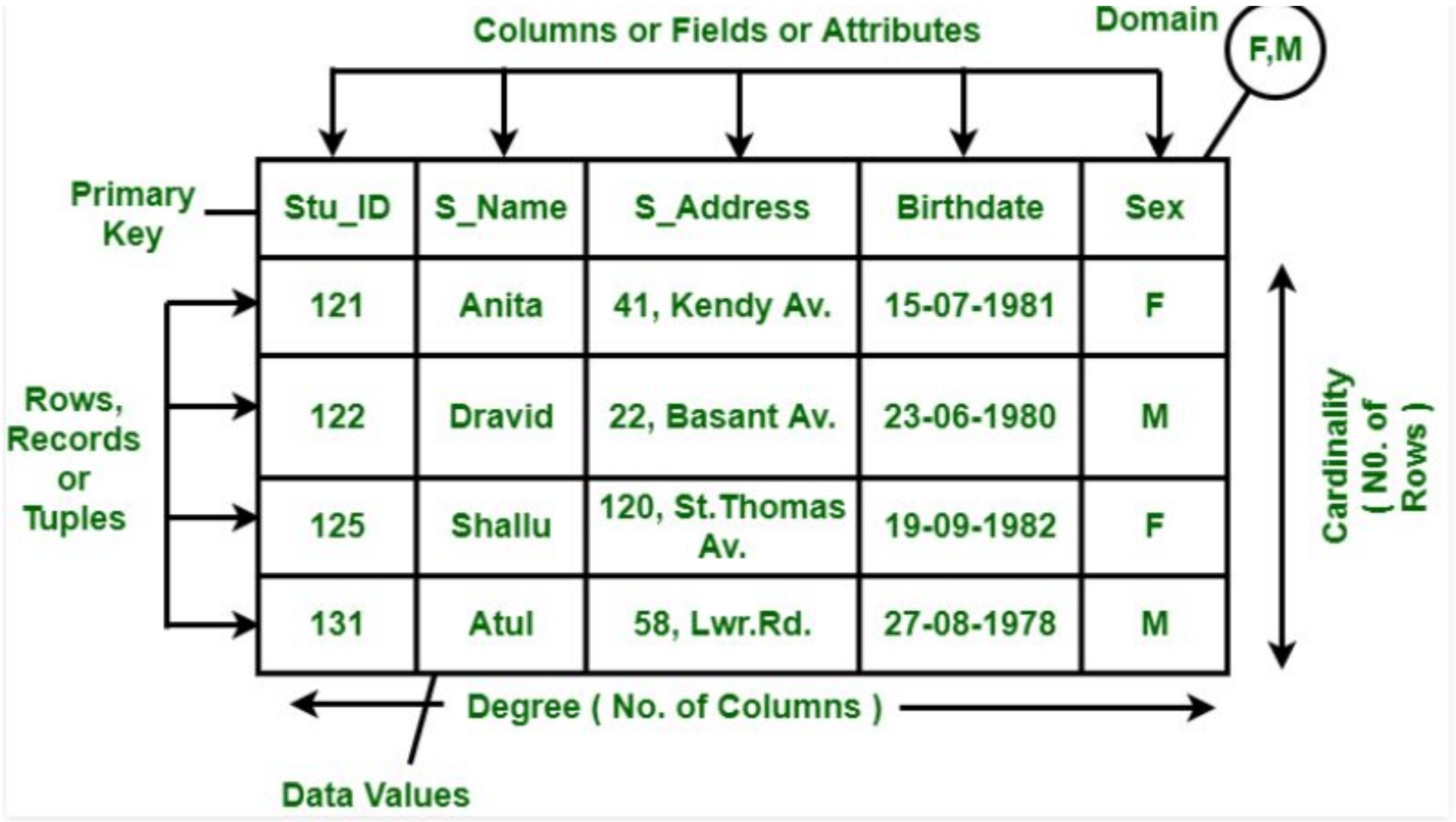
Types of database models

- There are many kinds of data models. Some of the most common ones include:
- Hierarchical database model
- Relational model
- Network model
- Object-oriented database model
- Entity-relationship model
- Document model
- Entity-attribute-value model
- Star schema
- The object-relational model, which combines the two that make up its name

Relational model

- The most common model, the relational model sorts data into tables, also known as relations, each of which consists of columns and rows. Each column lists an attribute of the entity in question, such as price, zip code, or birth date. Together, the attributes in a relation are called a domain. A particular attribute or combination of attributes is chosen as a primary key that can be referred to in other tables, when it’s called a foreign key.
- Each row, also called a tuple, includes data about a specific instance of the entity in question, such as a particular employee.
- The model also accounts for the types of relationships between those tables, including one-to-one, one-to-many, and many-to-many relationships. Here’s an example:

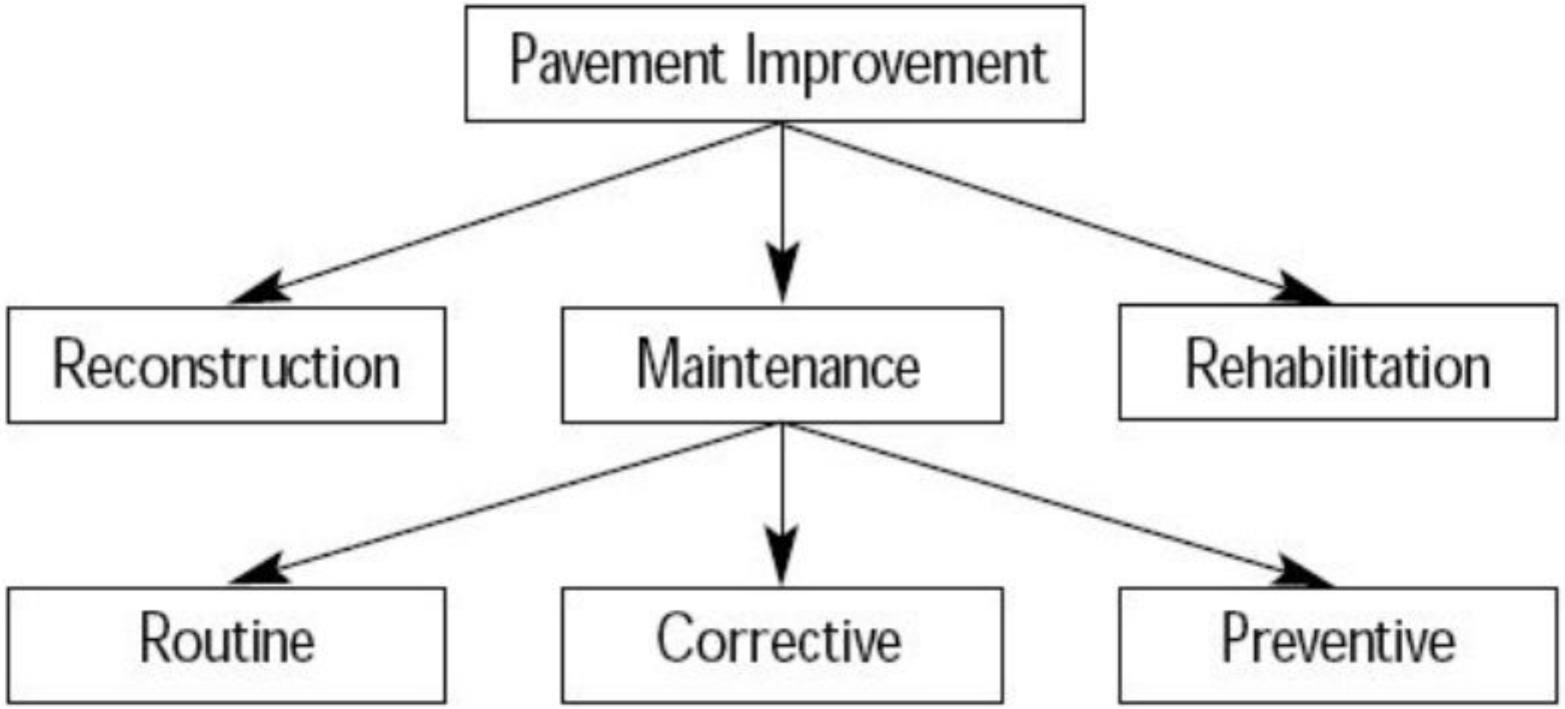
- Within the database, tables can be normalized, or brought to comply with normalization rules that make the database flexible, adaptable, and scalable. When normalized, each piece of data is atomic, or broken into the smallest useful pieces.
- Relational databases are typically written in Structured Query Language (SQL). The model was introduced by E.F. Codd in 1970.


Columns or Fields or Attributes
Domain
F,M
Primary Key
Rows, Records or Tuples
Cardinality (N0. of Rows)
Degree (No. of Columns)
Data Values

Stu_ID	S_Name	S_Address	Birthdate	Sex
121	Anita	41, Kendy Av.	15-07-1981	F
122	Dravid	22, Basant Av.	23-06-1980	M
125	Shallu	120, St.Thomas Av.	19-09-1982	F
131	Atul	58, Lwr.Rd.	27-08-1978	M

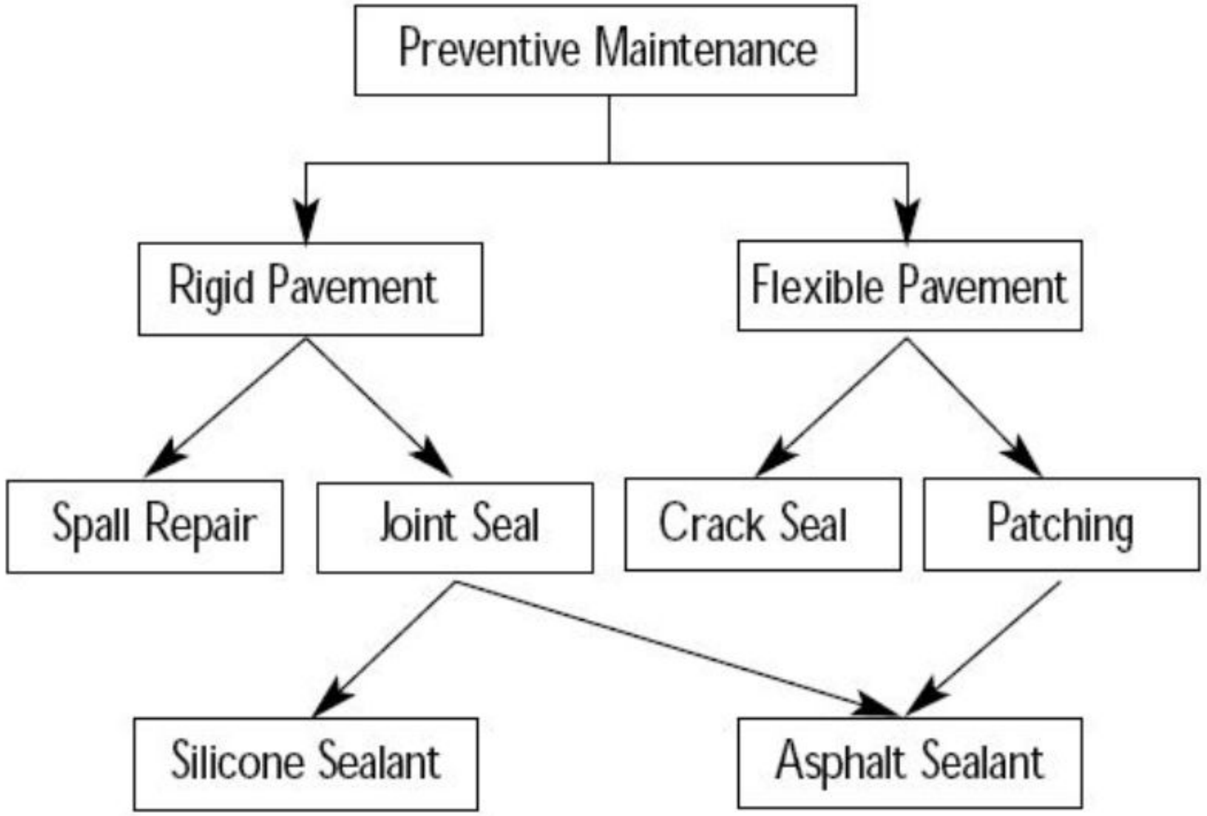
Hierarchical model

- The hierarchical model organizes data into a tree-like structure, where each record has a single parent or root.
- Sibling records are sorted in a particular order.
- That order is used as the physical order for storing the database. This model is good for describing many real-world relationships.
- This model was primarily used by IBM’s Information Management Systems in the 60s and 70s, but they are rarely seen today due to certain operational inefficiencies.


Pavement Improvement
Reconstruction
Maintenance
Rehabilitation
Routine
Corrective
Preventive

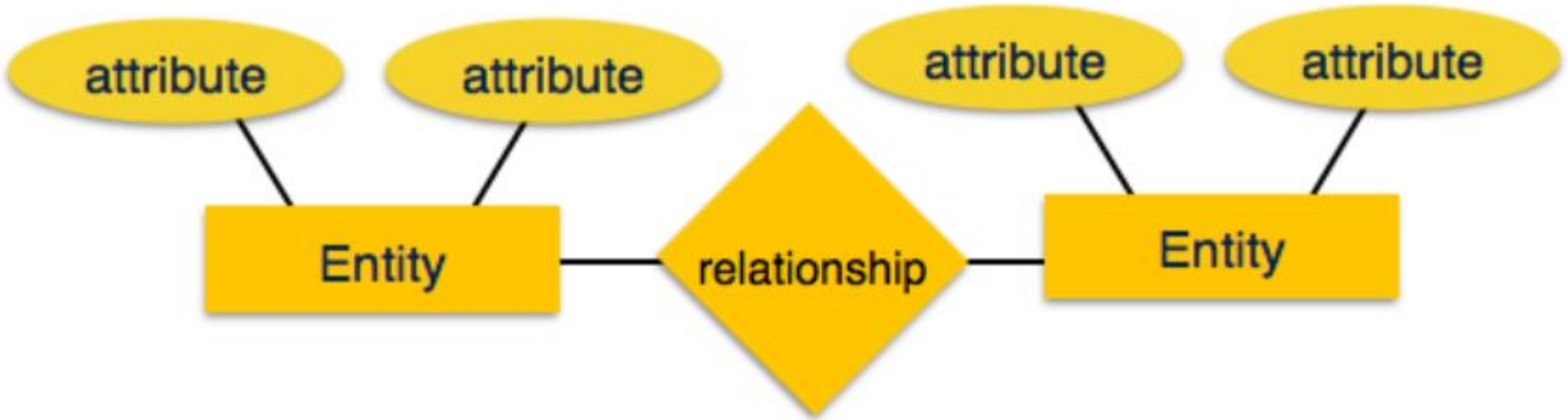
Network Model

- A network model is an extension of the hierarchical model, which arranges data in a structure similar to a graph.
- The network database model was created to solve the shortcomings of the hierarchical database model.
- In this type of model, a child can be linked to multiple parents, a feature that was not supported by the hierarchical data model.
- The parent nodes are known as owners and the child nodes are called members.
- The network data model can be represented as −


Preventive Maintenance
Rigid Pavement
Flexible Pavement
Spall Repair
Joint Seal
Crack Seal
Patching
Silicone Sealant
Asphalt Sealant

Entity-Relationship Model

- Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them. While formulating real-world scenario into the database model, the ER Model creates entity set, relationship set, general attributes and constraints.
- ER Model is best used for the conceptual design of a database.
- ER Model is based on −
- **Entities** and their *attributes.*
- **Relationships** among entities.
- These concepts are explained below.


attribute
attribute
attribute
attribute
Entity
relationship
Entity

- **Entity** − An entity in an ER Model is a real-world entity having properties called **attributes**. Every **attribute** is defined by its set of values called **domain**. For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class, etc.
- **Relationship** − The logical association among entities is called ***relationship***. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of association between two entities.
- Mapping cardinalities −
    - one to one
    - one to many
    - many to one
    - many to many

Types of DBMS Entities

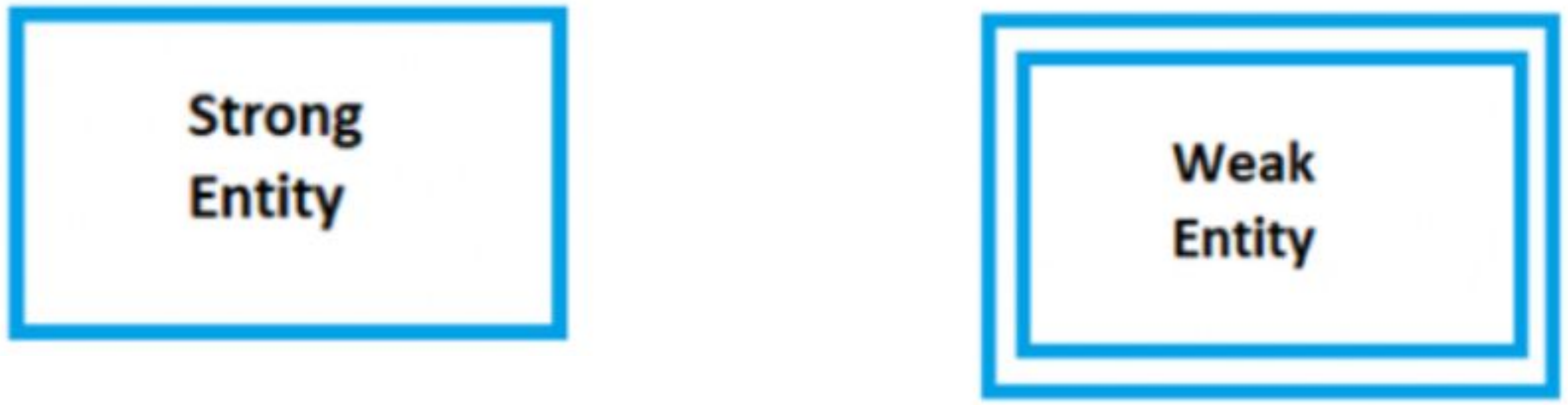
- The following are the types of entities in DBMS −

a. Strong Entity

- The strong entity has a primary key. Weak entities are dependent on strong entity. Its existence is not dependent on any other entity.
- Strong Entity is represented by a single rectangle −

b. Weak Entity

- The weak entity in DBMS do not have a primary key and are dependent on the parent entity. It mainly depends on other entities.
- Weak Entity is represented by double rectangle −


Strong Entity
Weak Entity

Types Of attributes

- There are five such types of attributes: Simple, Composite, Single-valued, Multi-valued, and Derived attribute. One more attribute is their, i.e. Complex Attribute, this is the rarely used attribute.
- **Simple attribute :**
- An attribute that cannot be further subdivided into components is a simple attribute.
  Example: The roll number of a student, the id number of an employee.

b. Composite attribute :

- An attribute that can be split into components is a composite attribute.
- **Example:** The address can be further split into house number, street number, city, state, country, and pin code, the name can also be split into first name middle name, and last name.

c. Single-valued attribute :

- The attribute which takes up only a single value for each entity instance is a single-valued attribute.
- **Example:** The age of a student.

d. **Multi-valued attribute :**

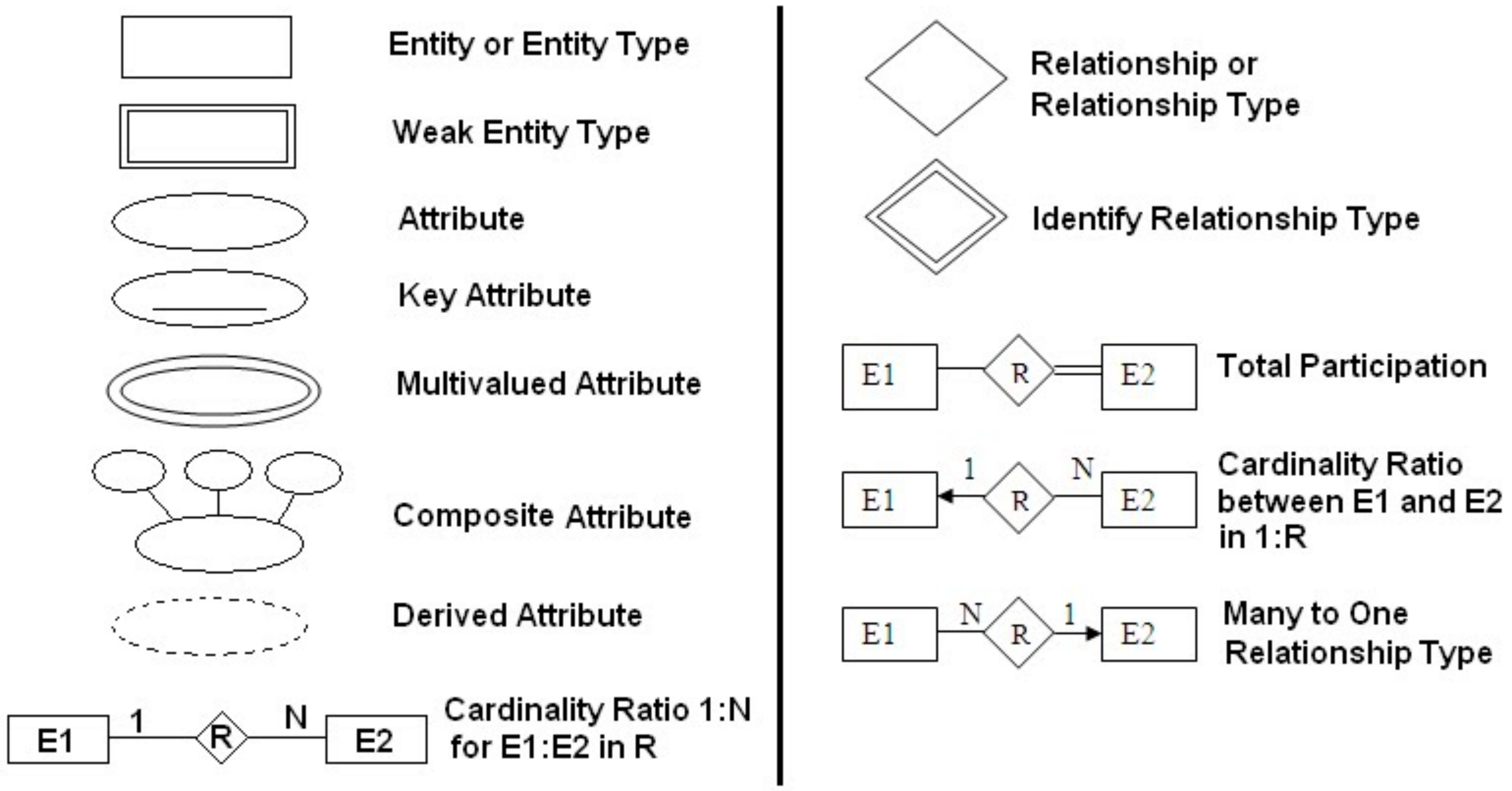
- The attribute which takes up more than a single value for each entity instance is a multi-valued attribute.
- **Example:** Phone number of a student: Landline and mobile.

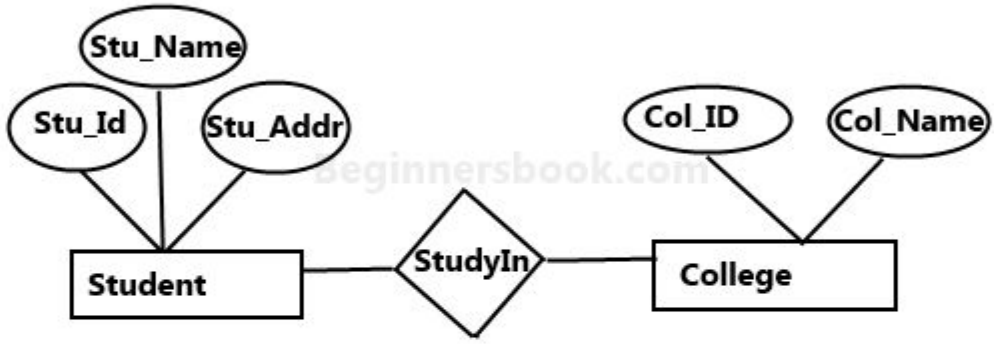
e. Derived attribute :

- An attribute that can be derived from other attributes is derived attributes.
- **Example:** Total and average marks of a student.

f. Complex attribute :

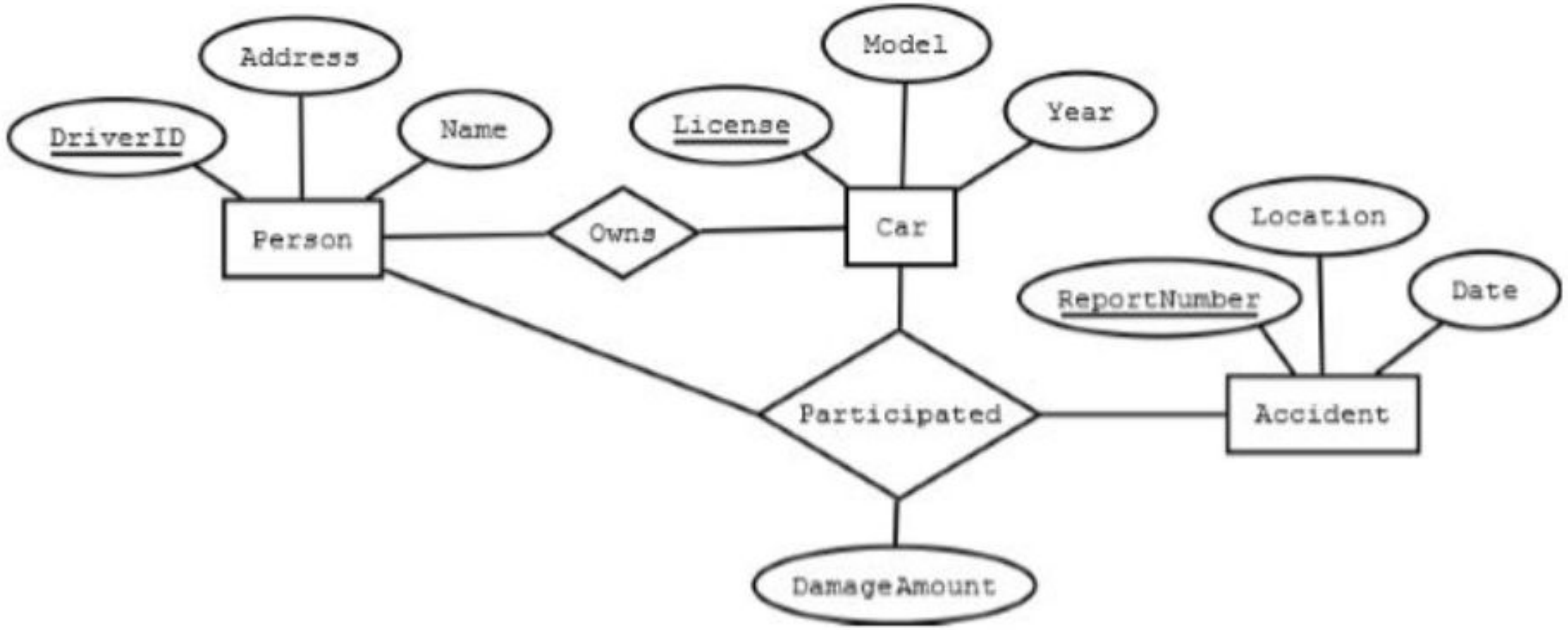
- Those attributes, which can be formed by the nesting of composite and multi-valued attributes, are called "*Complex Attributes*". These attributes are rarely used in DBMS(Database Management System). That's why they are not so popular.

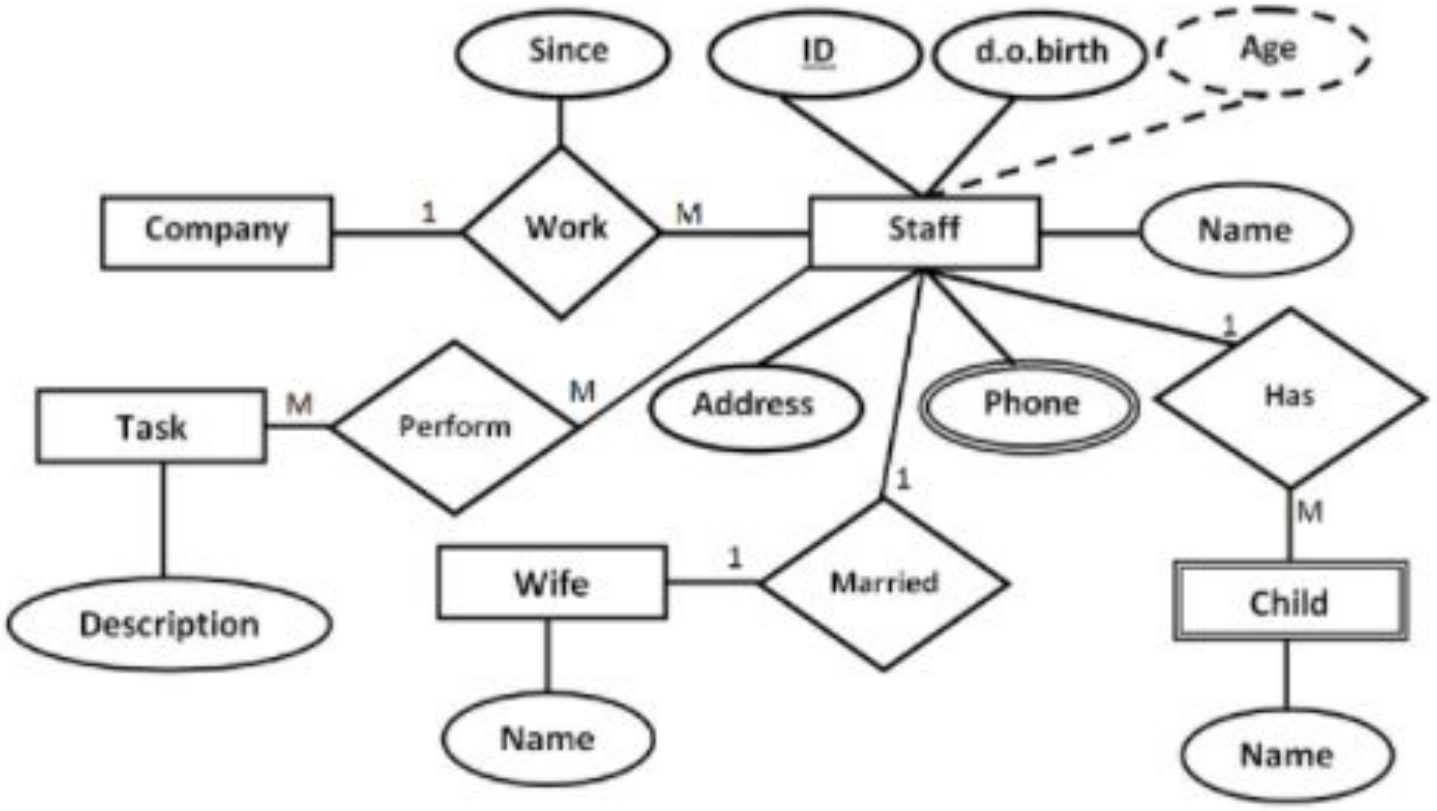

Entity or Entity Type
Weak Entity Type
Attribute
Key Attribute
Multivalued Attribute
Composite Attribute
Derived Attribute
E1
1
R
N
E2
Cardinality Ratio 1:N for E1:E2 in R
Relationship or Relationship Type
Identify Relationship Type
E1
R
E2
Total Participation
E1
1
R
N
E2
Cardinality Ratio between E1 and E2 in 1:R
E1
N
R
1
E2
Many to One Relationship Type

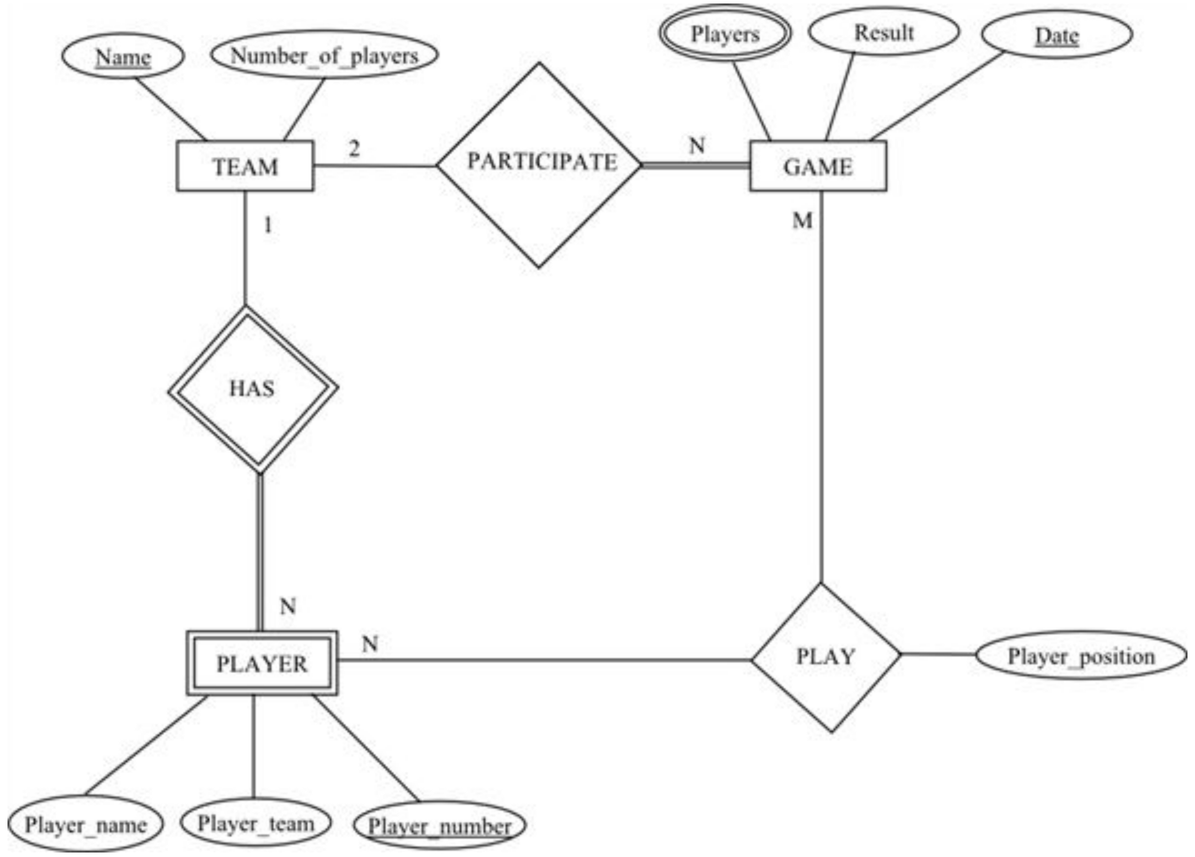


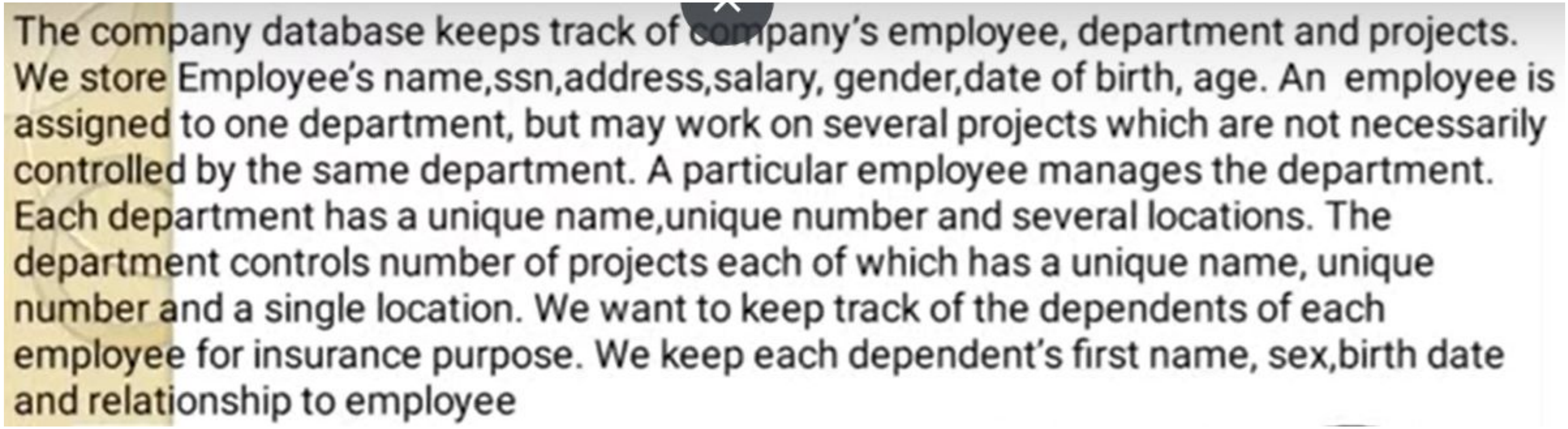
Sample E-R Diagram

Construct an ER diagram for car insurance company that has a set of customers each of whom owns one or more cars. Each car has associated with it zero to any number of recorded accidents



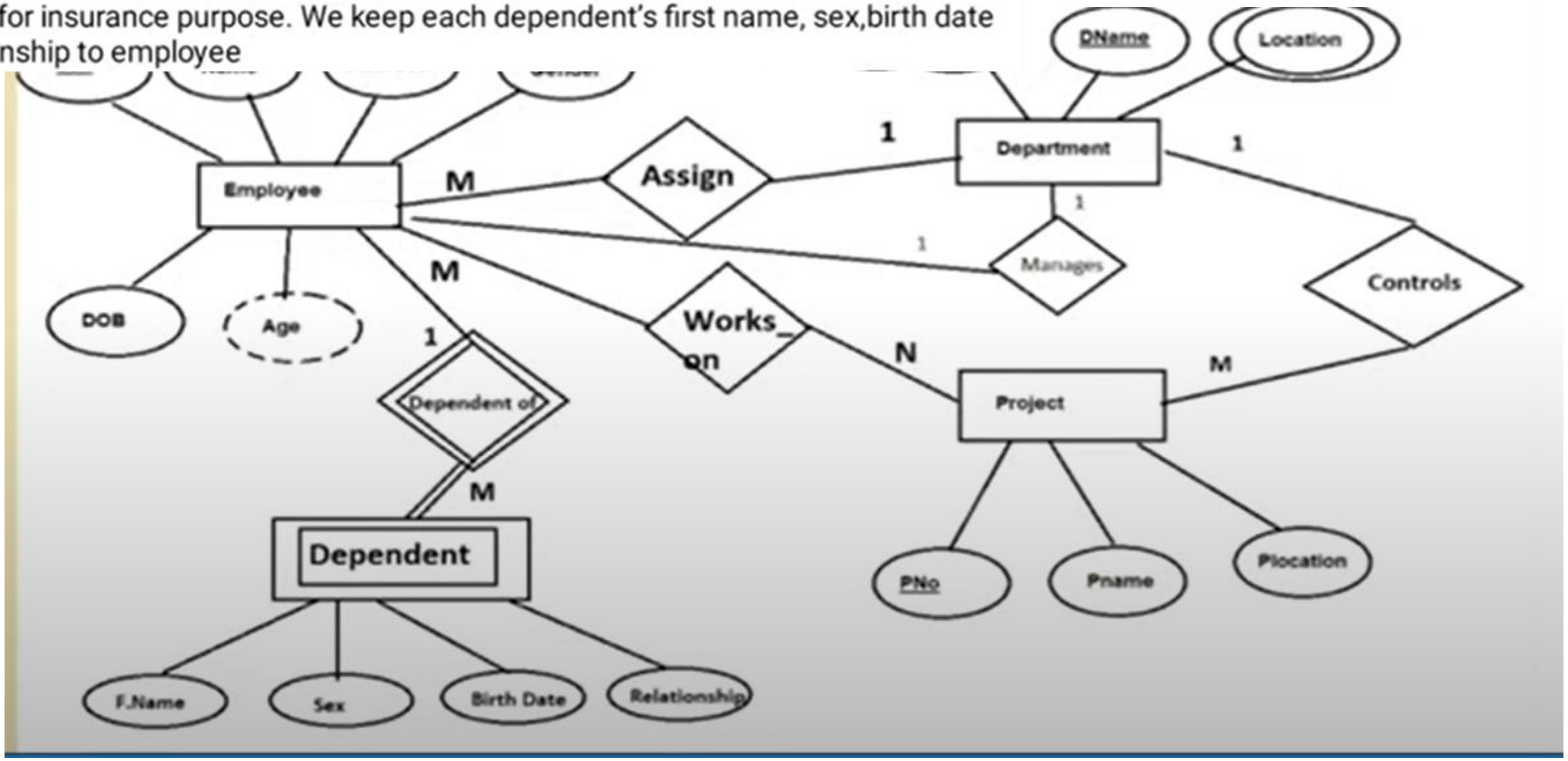

Since
ID
d.o.birth
Age
Company
1
Work
M
Staff
Name
Task
M
Perform
M
Address
Phone
1
Has
M
1
Wife
1
Married
Child
Description
Name
Name

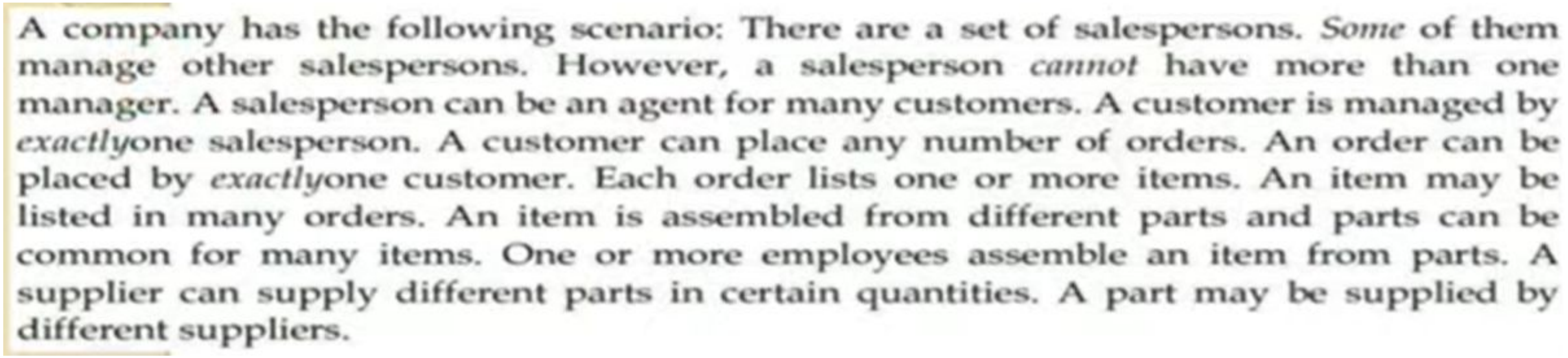

Name
Number_of_players
Players
Result
Date
TEAM
2
PARTICIPATE
N
GAME
1
M
HAS
N
PLAYER
N
PLAY
Player_position
Player_name
Player_team
Player_number

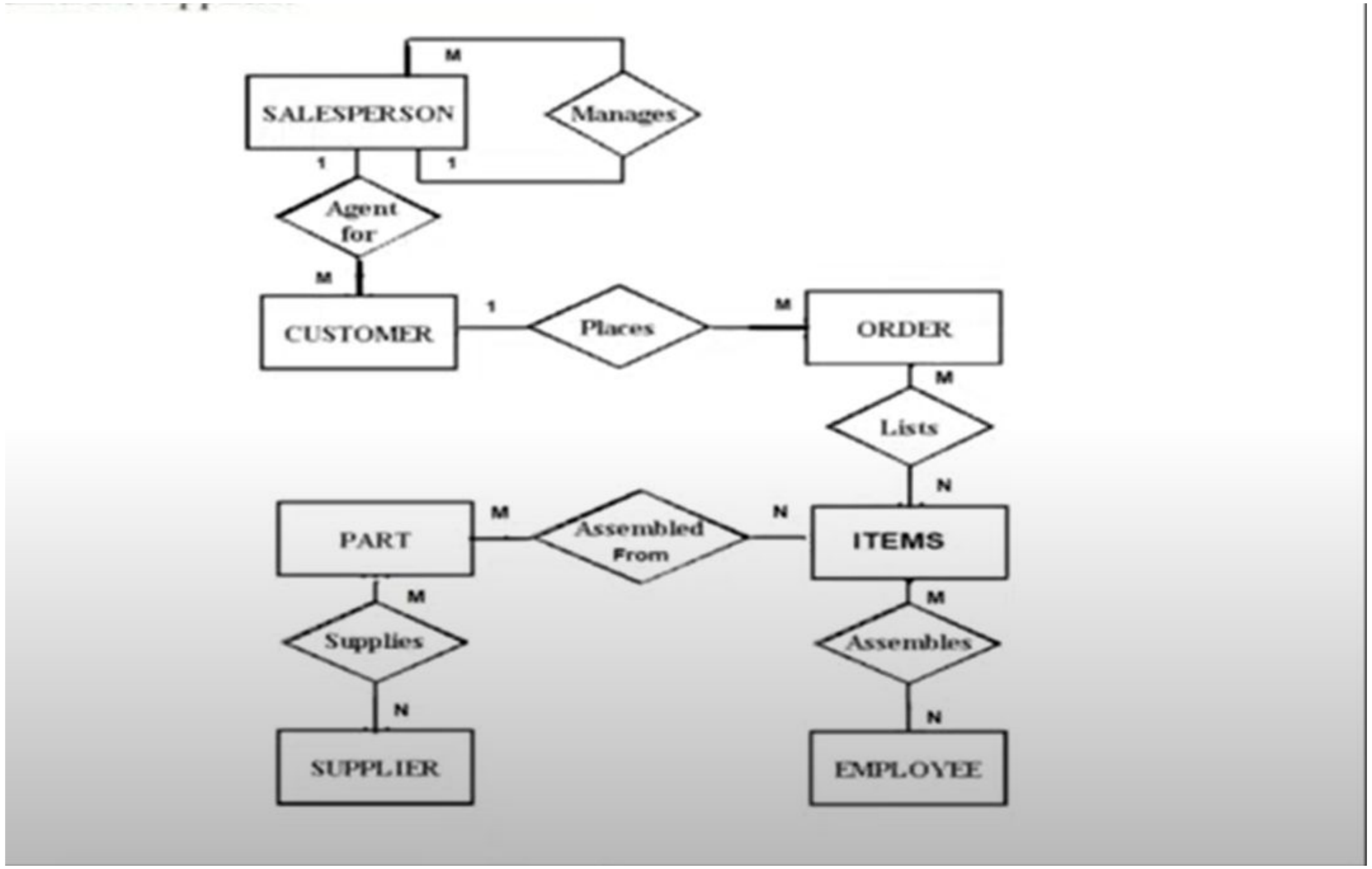


The company database keeps track of company's employee, department and projects. We store Employee's name,ssn,address,salary, gender,date of birth, age. An employee is assigned to one department, but may work on several projects which are not necessarily controlled by the same department. A particular employee manages the department. Each department has a unique name,unique number and several locations. The department controls number of projects each of which has a unique name, unique number and a single location. We want to keep track of the dependents of each employee for insurance purpose. We keep each dependent's first name, sex,birth date and relationship to employee

The company database keeps track of company's employee, department and projects. We store Employee's name,ssn,address,salary, gender,date of birth, age. An employee is assigned to one department, but may work on several projects which are not necessarily controlled by the same department. A particular employee manages the department. Each department has a unique name,unique number and several locations. The department controls number of projects each of which has a unique name, unique number and a single location. We want to keep track of the dependents of each employee for insurance purpose. We keep each dependent's first name, sex,birth date and relationship to employee




A company has the following scenario: There are a set of salespersons. *Some* of them manage other salespersons. However, a salesperson *cannot* have more than one manager. A salesperson can be an agent for many customers. A customer is managed by *exactly*one salesperson. A customer can place any number of orders. An order can be placed by *exactly*one customer. Each order lists one or more items. An item may be listed in many orders. An item is assembled from different parts and parts can be common for many items. One or more employees assemble an item from parts. A supplier can supply different parts in certain quantities. A part may be supplied by different suppliers.


M
SALESPERSON
Manages
1
1
Agent for
M
CUSTOMER
1
Places
M
ORDER
M
Lists
N
PART
M
Assembled From
N
ITEMS
M
Supplies
M
Assembles
N
N
SUPPLIER
EMPLOYEE