College of Computer Science and Mathematics
Computer Science Department
Third Class

DATABASE

Part (1)

References :

1. Modern Database Management ,Hoffer, Jeff, Ramesh Venkataraman, and Heikki Topi, 13th Edition, Prentice Hall Press, 2019.
2. Database system concepts, Silberschatz, Korth and Sudarshan, 7th ed, McGraw-Hill, 2019.

1. Introduction:

1.1. The Concept of a Database

A database is a collection of data and a set of rules that organize the data by specifying certain relationships among the data.

Through these rules, the user describes a logical format for the data. The data items are stored in a file, but the precise physical format of the file is of no concern to the user. While **database administrator** is a person who defines the rules that organize the data and also controls who should have access to what parts of the data.

The user interacts with the database through a program called a **database manager or a database management system** (DBMS), informally known as a front end.

1.2. Database Management System (DBMS):

The DBMS is the application that manages the data included within the database, it contains information about a particular enterprise, it provides an environment that is both convenient and efficient to use.

DBMS consists of:

1. Collection of interrelated data
2. Set of programs to access the data.

1.3. File System :

In the early days, database applications were built on top of file systems which has many drawbacks to store and manipulate data such as:

1. Data redundancy and inconsistency

-Multiple file formats, duplication of information in different files

2. Difficulty in accessing data.

-Need to write a new program to carry out each new task

3. Data isolation — multiple files and formats

4. Integrity problems

- ***Integrity constraints (e.g. account balance > 0) become part of program code***
- ***Hard to add new constraints or change existing ones***

5. Atomicity of updates

- ***Failures may leave database in an inconsistent state with partial updates carried out.***

E.g. transfer of funds from one account to another should either complete or not happen at all

6. Concurrent access by multiple users

- ***Concurrent accessed needed for performance***
- ***Uncontrolled concurrent accesses can lead to inconsistencies.***

E.g. two people reading a balance and updating it at the same time

7. Security problems

1.4. Advantages of Using Databases

The logical idea behind a database is this: A **database** is a single collection of data, stored and maintained at one central location, to which many people have access as needed. The essence of a good database is that the users are unaware of the physical arrangements; the unified logical arrangement is all they see.

The database management system (DBMS)offers many advantages over a simple file system such as:

1. **Shared access**, so that many users can use one common, centralized set of data.
2. **Minimal redundancy**, so that individual users do not have to collect and maintain their own sets of data.
3. **Data consistency**, so that a change to a data value affects all users of the data value.
4. **Data integrity**, so that data values are protected against accidental or malicious undesirable changes.
5. **Controlled access**, so that only authorized users are allowed to view or to modify data values.

1.5. Database Applications:

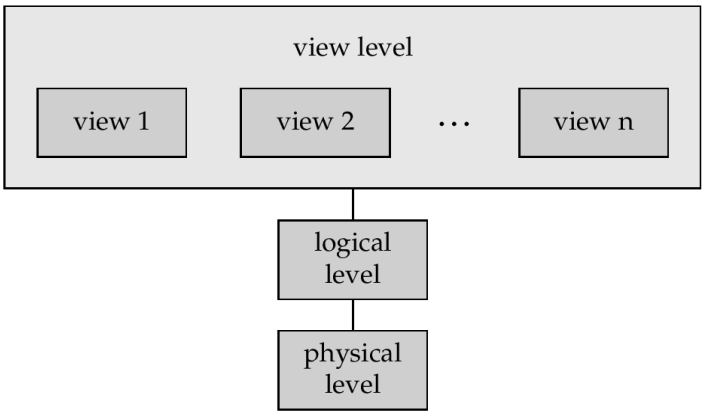
1) Banking: all transactions
2) Airlines: reservations, schedules
3) Universities: registration, grades
4) Sales: customers, products, purchases
5) Manufacturing: production, inventory, orders, supply chain
6) Human resources: employee records, salaries, tax deductions

1.6. Data Levels of Abstraction:

1. Physical level: describes how a record (e.g., customer) is stored in storage media.
2. Logical level: describes data stored in database, and the relationships among the data.

```
type customer = record
                    name : string;
                    street : string;
                    city : integer;
                end;
```

3. View level: application programs hide details of data types. Views can also hide information (e.g., salary) for security purposes.



1.7. Schemas and Instances:

In terms of databases schemas and instances are like types and variables in programming languages.

***1. Schema*: is the logical structure of the database ,** it is analogous to type information of a variable in a program, **it has two types:**
Physical schema: database design at the physical level
Logical schema: database design at the logical level
e.g., the database consists of information about a set of customers and accounts and the relationship between them)

***2. Instance*: is the actual content of the database at a particular point in time , it is** analogous to the value of a variable

Physical Data Independence: the ability to modify the physical schema without changing the logical schema.

2. Data Models

Data Models : A collection of tools for describing

1. **Data**
2. **Data Relationships**
3. **Data Semantics**
4. **Data Constraints**

2.1 Types of Data Models:

1. **Entity-Relationship Model**
2. **Relational Model**
3. **Object-Oriented Model**
4. **Semi-Structured Data Models**

Older models:

5. **Network model**
6. **Hierarchical model**

3. Database Manipulation Languages :

3.1 Data Definition Language (DDL):

It is a specification notation for defining the database schema.
E.g.

create table ***account*** **(**
account-number char**(10),**
balance integer**)**

DDL compiler generates a set of tables stored in a *data dictionary.*

Data dictionary contains metadata (i.e., data about data)

3.2 Data Storage and Definition Language (DSDL):

It is a language in which the storage structure and access methods used by the database system are specified. Usually an extension of the data definition language.

3.3 Data Manipulation Language (DML)

It is a query language for accessing and manipulating the data organized by the appropriate data model.

4. Information and knowledge :

Information is the Data that have been processed in such way of the person who use the Data. Since there are three types of data:

1. **Input Data**
2. **Output Data**
3. **Operational Data**

Knowledge : is the gained facts and predictions after processing the information.

5. Database Users:

Database users are differentiated by the way they expect to interact with the system.

1. **Application programmers – interact with system through DML calls.**
2. **Sophisticated users – form requests in a database query language**
3. **Specialized users – write specialized database applications that do not fit into the traditional data processing framework.**
4. **Naïve users – invoke one of the permanent application programs that have been written previously.**

5.1 Database Administrator

Database Administrator is the person who coordinates all the activities of the database system; the database administrator has a good understanding of the enterprise's information resources and needs.

Database administrator's duties include:

1. **Schema definition**
2. **Storage structure and access method definition**
3. **Schema and physical organization modification**
4. **Granting user authority to access the database.**
5. **Specifying integrity constraints**
6. **Acting as liaison with users**
7. **Monitoring performance and responding to changes in requirements**

6. Transaction Management:

Transaction: is a collection of operations that performs a single logical function in a database application

- **Transaction-management component ensures that the database remains in a consistent (correct) state despite system failures.**

e.g.: power failures and operating system crashes, and transaction failures.

7. Storage Management:

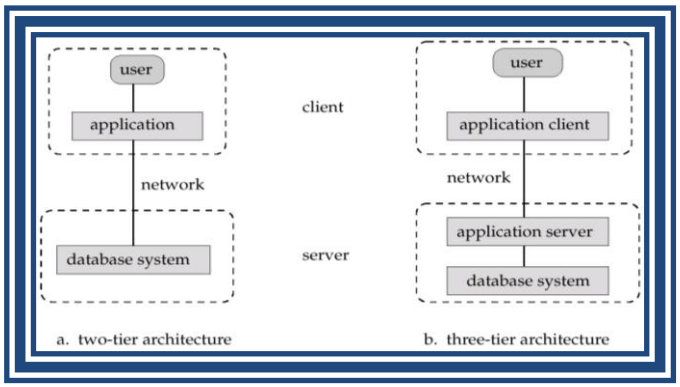
Storage manager : is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.

The storage manager is responsible to the following tasks:

1. **Interaction with the file manager**
2. **Efficient storing, retrieving, and updating of data .**

8. Database Application Architectures :

1. **Two-tier architecture:**
2. **Three-tier architecture:**



9. General Aspects of Database:

1. Entity : is the object can be recognized from other objects depending on specific set of attributes. Such as Person, Student, Event, Plant.

2. Relationship: it is the union between entities

3. Group Items:the set of facts in the data base , it includes Entities + Relationships.

4. Data Attributes: a general and special properties or characteristic of an entity or relationship that is of interest to the organization.

5. Data Value: it is the information included by Data Attributes

6. Data Domain: the allowable domain used by the attributes to represent its data.

7. Data Structure :specification of the relationships among entities.

8. Records: (Instances): the set of values for an entity.

- **Record 1**
- **Record 2**

Name	Street	City
Mazen Sabah	Alnaser	19
Azhar Khaled	Alhadbaa	21

9. Primary Key : It is one or more than attributes included within an entity , it has the a unique value for each instance of an entity , used for distinguishing between records .

e.g. : each bank customer has a fixed account_no could not be changed , and can used to determine the customer name and his account .

10. Meta Data: The data that describes the properties of another data.
e.g.: declaring data structures.

```
type customer = record
        name : string;
        street : string;
        city : integer;
end;
```

10.Entity Relationship (ER) Model:

ER Model is a logical representation of the data for an organization or for a business area. Within this model the database can be modeled as:

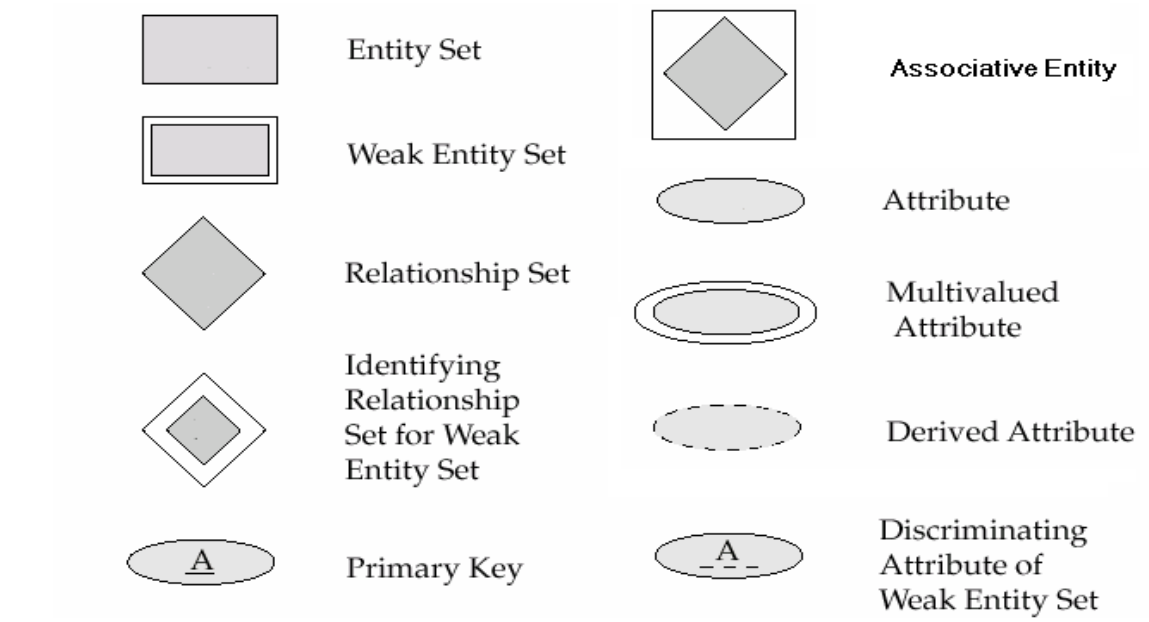
1. Collection of Entities.

2. Relationship between Entities.

10.1. Entity Relationship (ER) Diagram

ER- Diagram is the graphical representation of an Entity-Relationship Model.

The symbols used to draw ER diagram are:



Notably that:

- Each entity type has a set of attributes associated with it.
- In naming attributes, we use an initial capital letter followed by lower case letters, and if attributes consist of two words ,we use underscore character '_' to connect the words and we start each word with capital letter.
- An attribute is associated with exactly one entity or relationship.

As an example, an Entity is STUDENT with attributes: Student_Id ,Student_Name , Address and Phone_Number

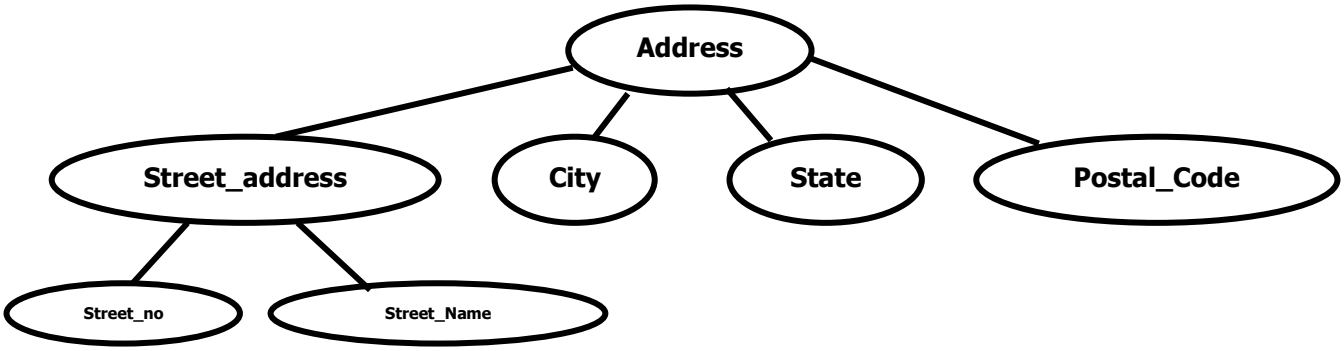
Below are two entity instances with attribute values:

Student_Id = **1209** Student_Name = **Sufyan Salim Mahmood** Address = **Mosul – Alkafaat** Phone_Number = **+96460817543**	Student_Id = **1190** Student_Name = **Mohammad Zaki Hasan** Address = **Mosul – AlnabiSheet** Phone_Number = **+96460813241**

10.2. Attribute Types :

1- Simple Attribute : An attribute that cannot be broken down into smaller components like all attributes associated with STUDENT.

2- Composed Attributes: an attribute that can be broken down into components parts like address.

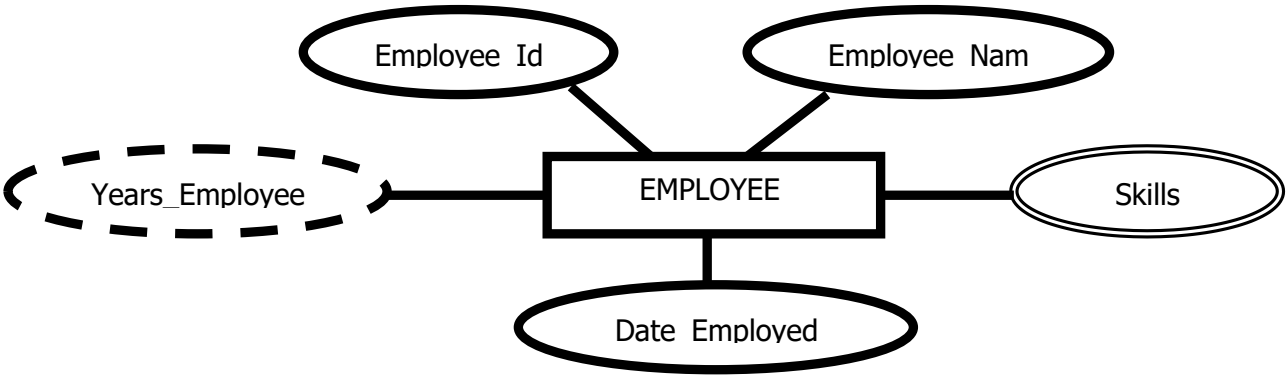


3-**Single Valued Attribute :** An attribute that take single one value for a given entity instance, like Student_Id

4- Multi Valued attribute : An attribute that may take more than one value for a given entity instance like : EMPLOYEE may have more than one Skill.

5- Stored Attribute : An attribute whose value is fixed and stored .

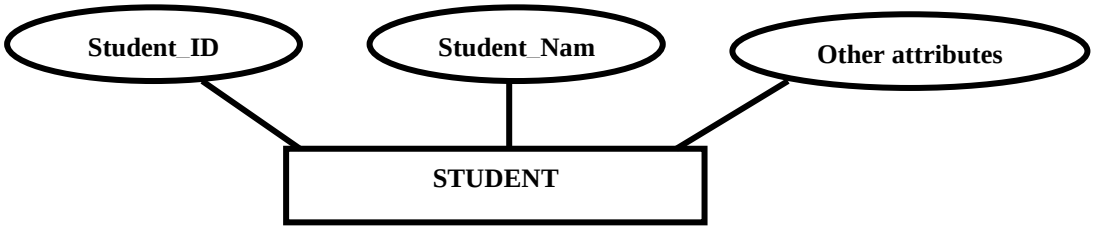
6-Derived Attribute : An attribute whose values can be calculated from related stored attribute values .



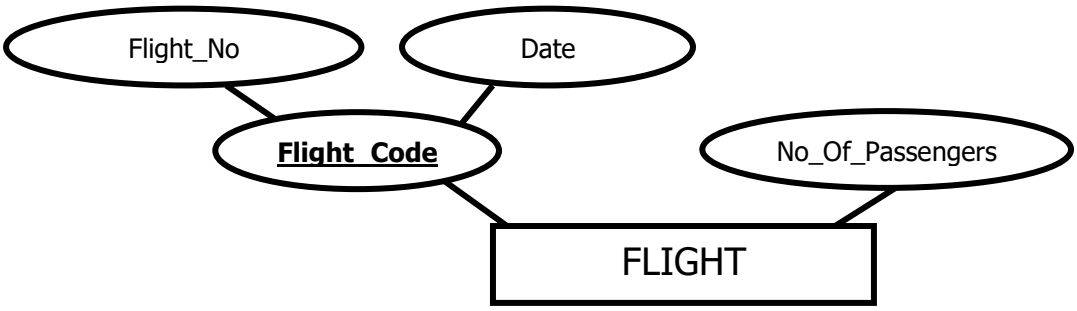
*** Identifier:** an attribute or combination of attributes that uniquely identifies individual instances of an entity type, we express it by underline its name.

Note: If the entity type has one attribute, and it is an identifier, this entity considered "illegal".******
Composed Identifier : An identifier that consists of composite attribute

a- Simple Key attribute :



b- Composite Key Attribute :



10.3. Entity Types:

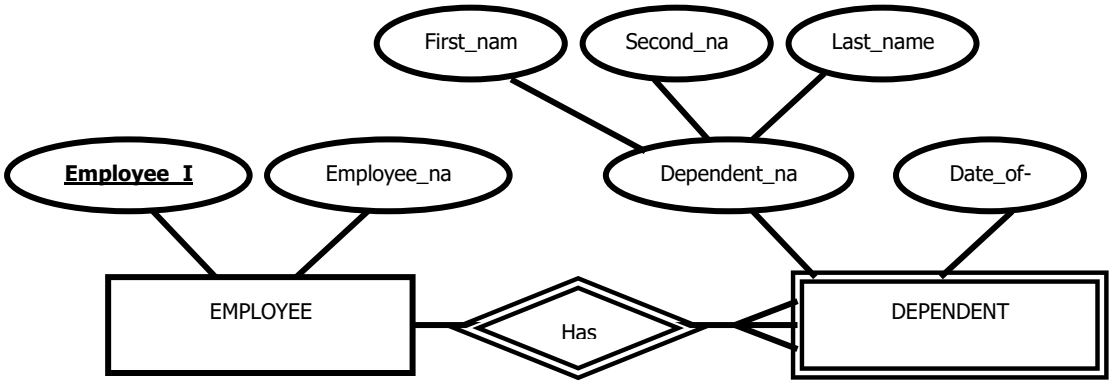
There are three types of entities:

1. **Strong Entity** : an entity that exists independently of other entity types. It has identifier that is an attribute or combination of attributes that uniquely distinguish each occurrence of that entity.

2. **Weak Entity** : an entity whose existence depend on some other entity type . It does not have its own identifier. But have an attribute that serves as partial identifier.

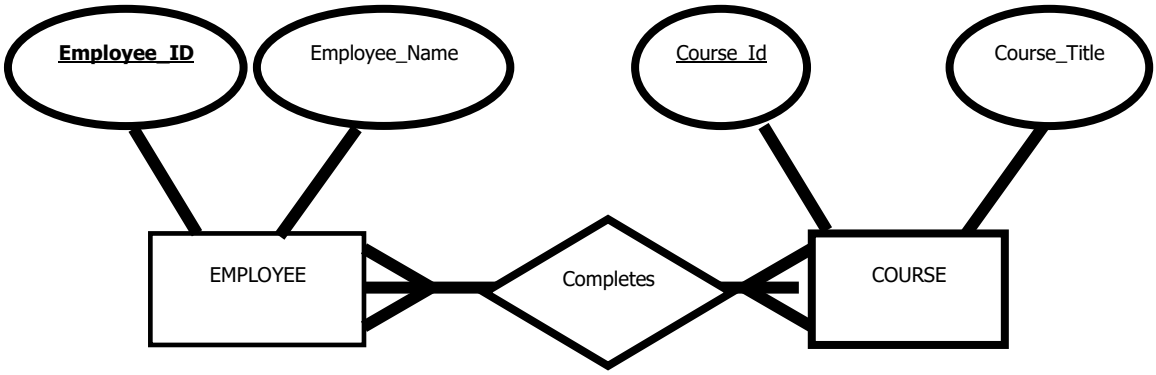
- **Full identifier** for weak entity = ***partial identifier + identifier of its owner***
- **Identifying Owner (**owner entity**) the entity on which the weak entity type depends.**
- **Identifying relationship:** the relationship between a weak entity and its owner.

3. **Associative Entity**.



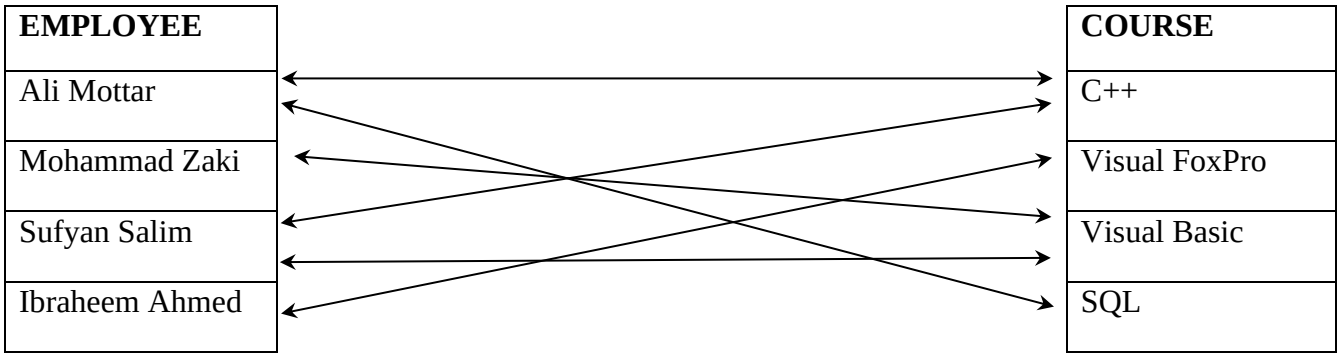
10.4. Relationship: A meaningful association between entities.

The relationship name is created using a short descriptive verb phrase and should be in present tense, which is meaningful to the user in naming the relationship.



((Relationship type))

Relationship Instance : An association between (or among) entity instances ,where each relationship instance includes exactly one entity from each participating entity type,

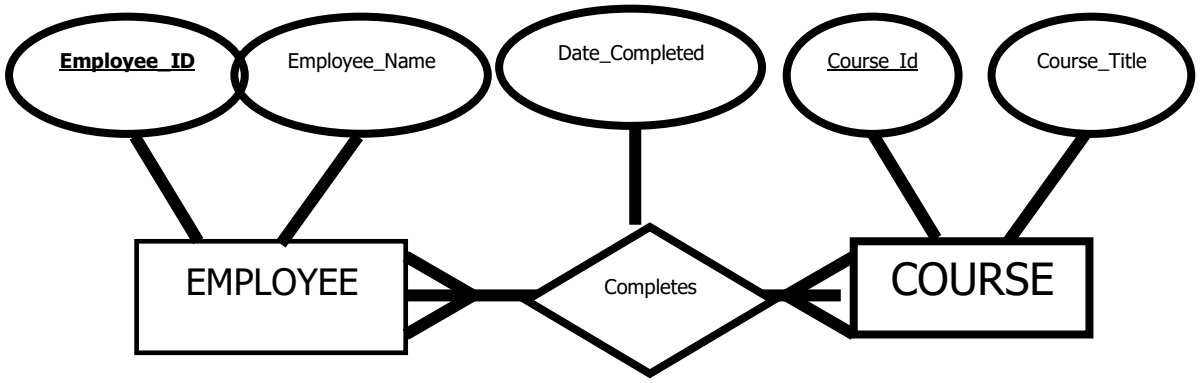


((Relationship Instance))

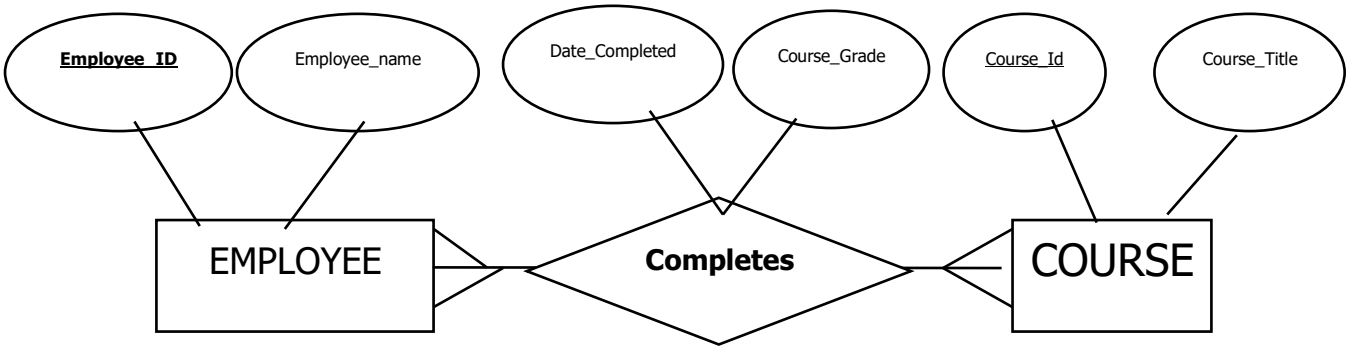
Attributes on Relationships:
From the previous figure we can consider a table , if we want to add an attribute (Date_Completed), it will be:

EMPLOYEE	COURSE	Date_Completed
Ali Mottar	C++	12-12-2021
Ali Mottar	SQL	12-7-2022
Mohammad Zaki	Visual Basic	10-5-2022
Sufyan Salim	C++	21-7-2022
Sufyan Salim	Visual Basic	2-8-2022
Ibraheem Ahmed	Visual FoxPro	22-8-2022

As shown in the following E-R diagram :



Also, we can add another attribute, like the grade of the course (Course_Grade) then the diagram will be:



10.5. Associative Entities:

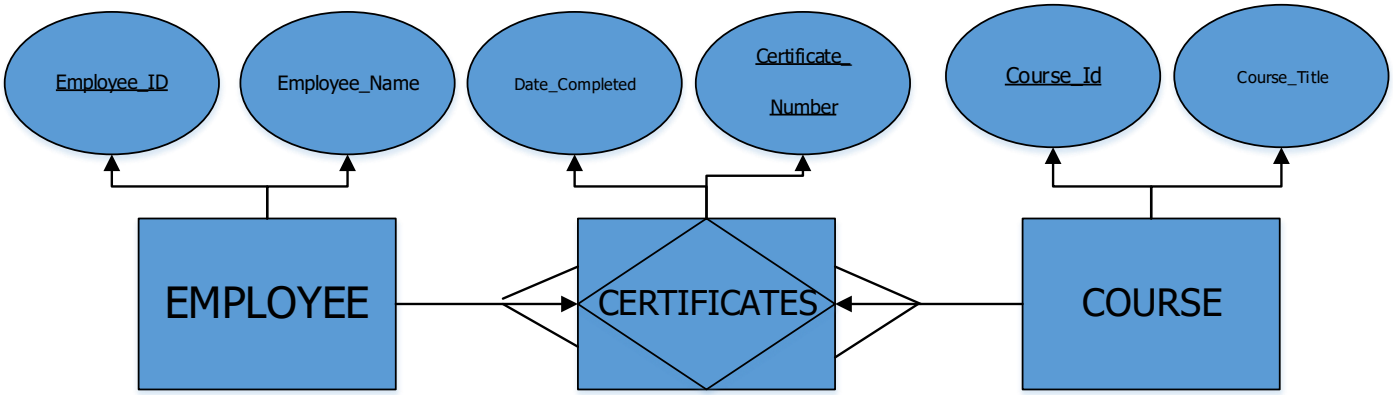
Associative Entity is an entity that associates the instances of one or more entity types and contains attributes that are peculiar to the relationship between those entity instances.

Associative entity sometimes referred to as [gerunds] since the relationship name [verb] is usually converted to an entity name that is noun (an –ing form of the verb).

How do we know whether to covert a relationship to an associative entity type?

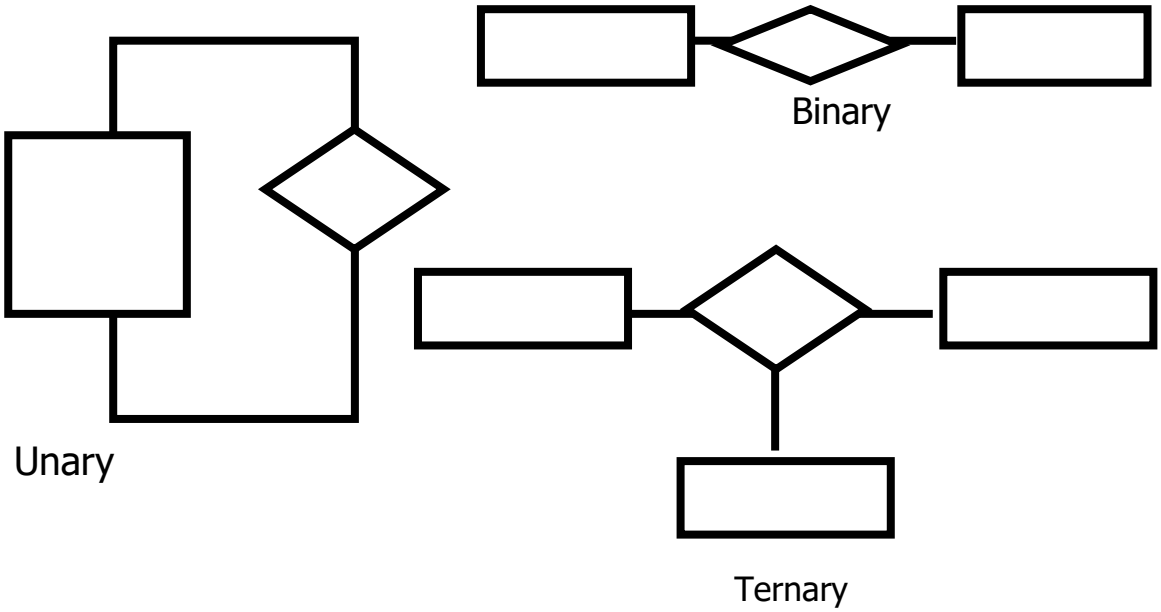
❖ Following are four conditions that should exists:

1. All the relationships for the participating entity type are (Many) relationship.
2. The resulting associative entity type has independent meaning to end user, and preferably can be identified with a single attribute identifier .
3. the associative entity has one or more attributes in addition to the identifier .
4. the associative entity participates in one or more relationships independent of the entities related in the associated relationship.

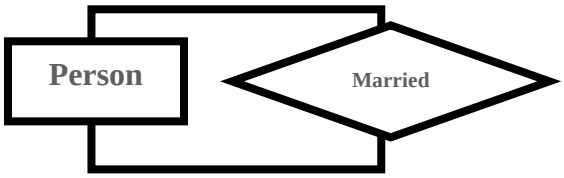


10.6. Relationship Degree:

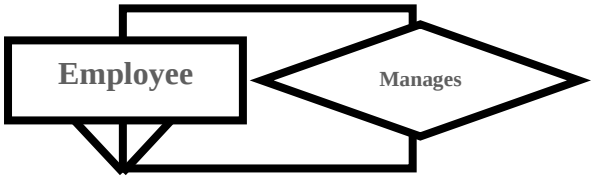
Refers to number of entities that participate in a relationship set.



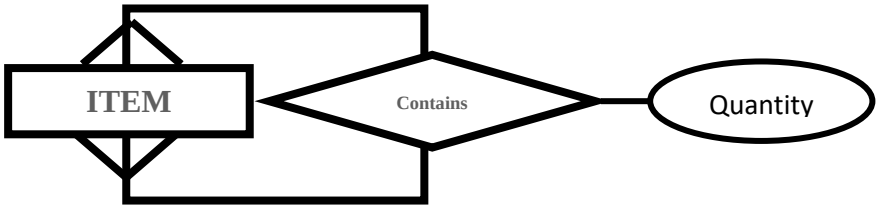
Unary relationship : the relationship between the instances of a single entity type (recursive relationships)



One – to -One

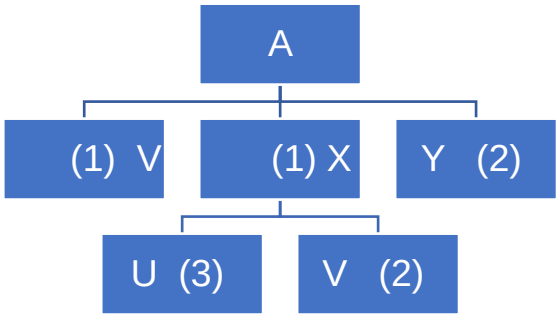


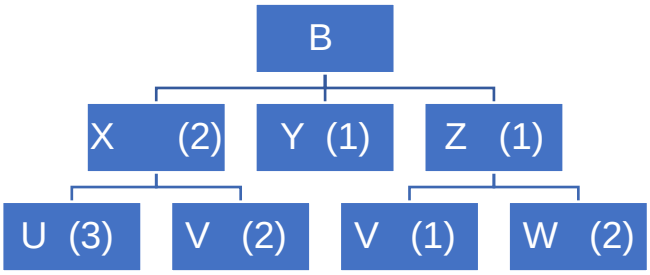
One – to -Many



Many – to – Many

Two Instances for Unary Many- to -Many



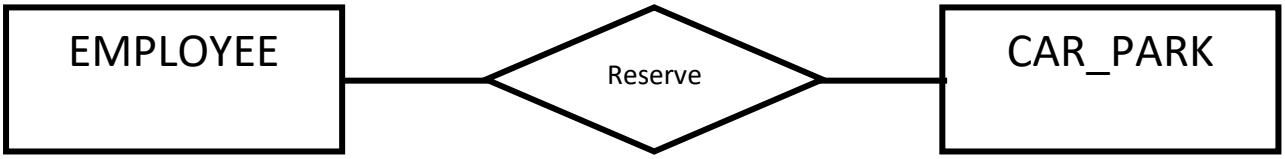


2. Binary Relationship: the relation involves two entities, i.e. the relation between the instances of two entity type

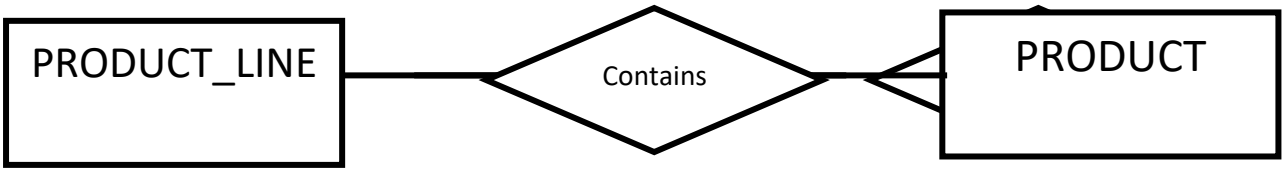
Generally, most relationships in a database system are binary.

Examples of Binary relationships:

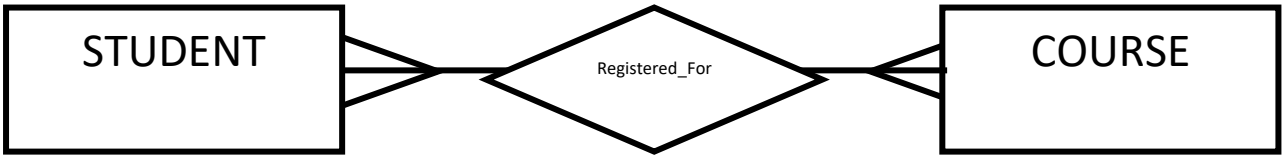
One – to – One



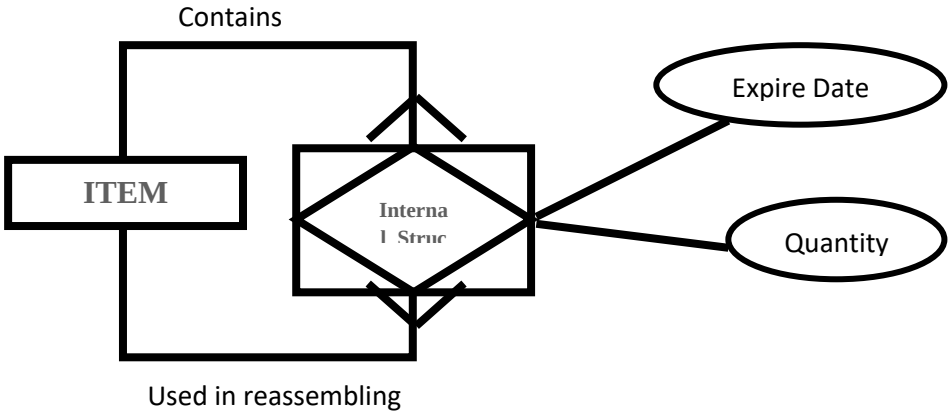
One – to - Many



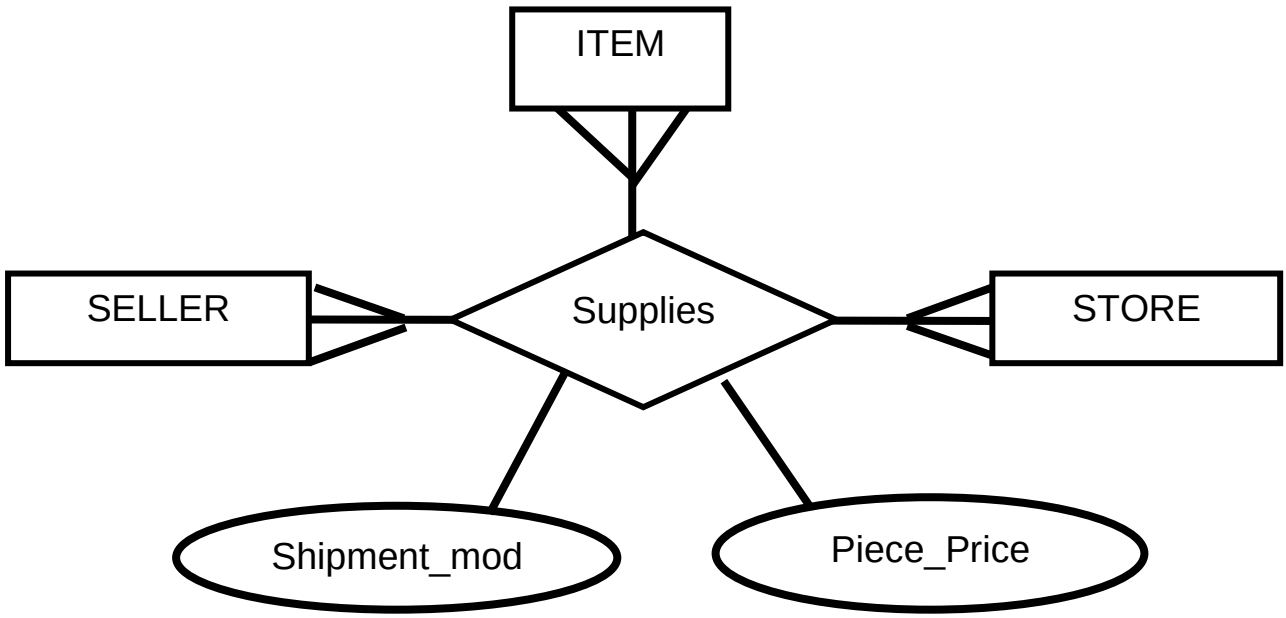
Many – to Many

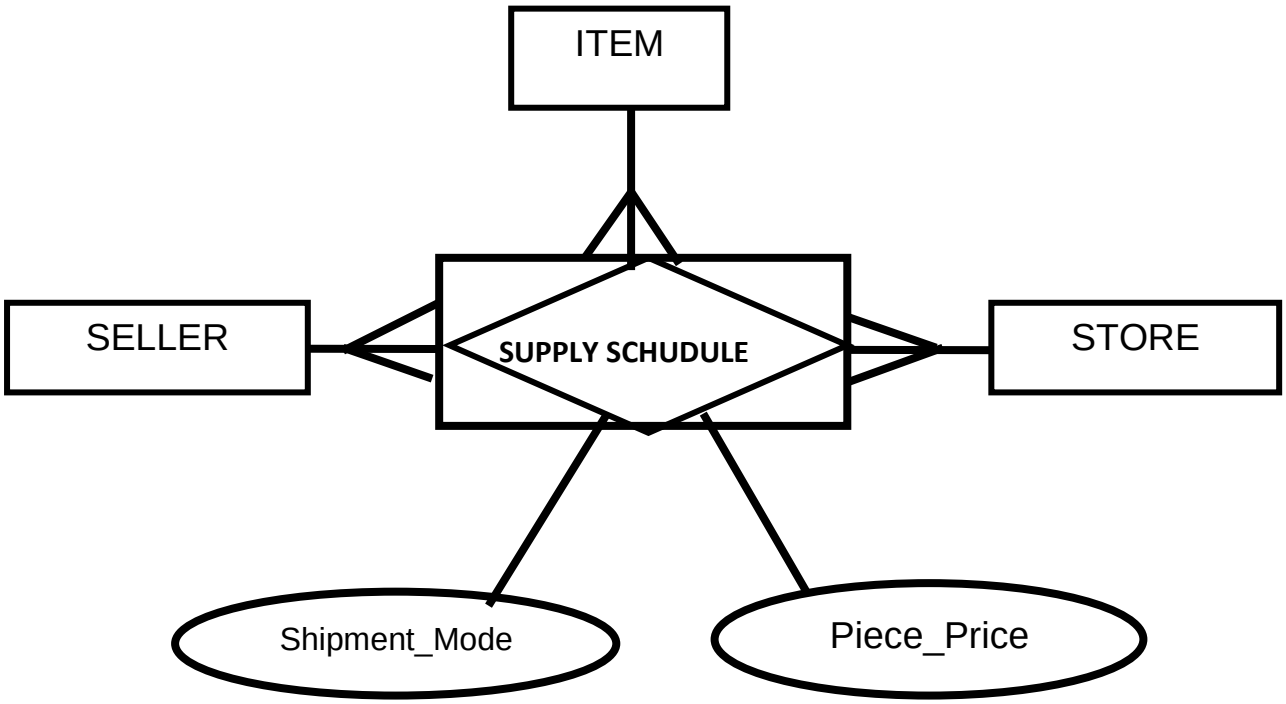


Associative Entity



3.Ternary relationship: involve more than two entity sets. I.e.: simultaneous relationship among the instances of three entity types at least





Associative Entity

E.g. :

Suppose employees of a bank may have jobs (responsibilities) at multiple branches, with different jobs at different branches. Then there is a ternary relationship set between entity sets *employee, job and branch.*

Relationships between more than two entity sets are rare. Most relationships are binary.

10.7. Entity Type versus System Input, Output, or Users

Example of inappropriate entities:

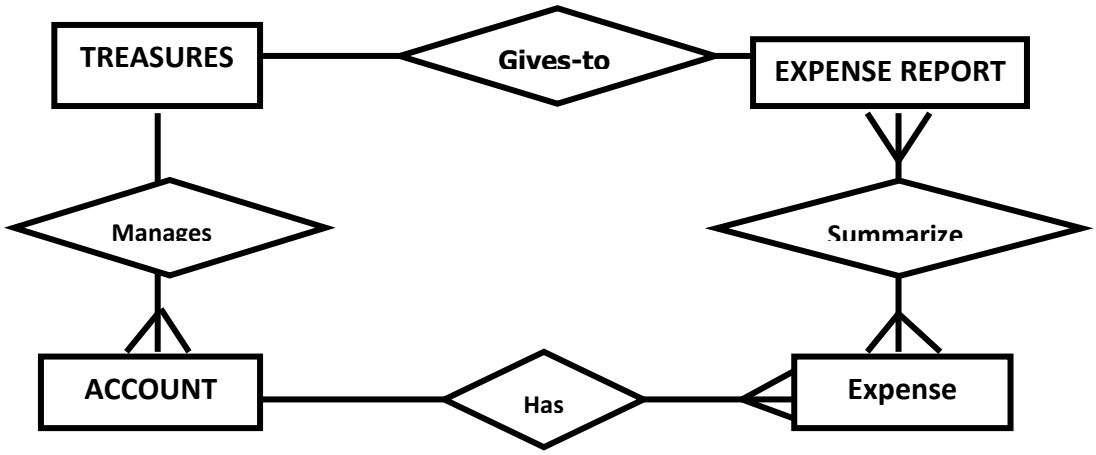


Figure (a)

a- System user (Treasurer) and output (expense report) shown as entities.

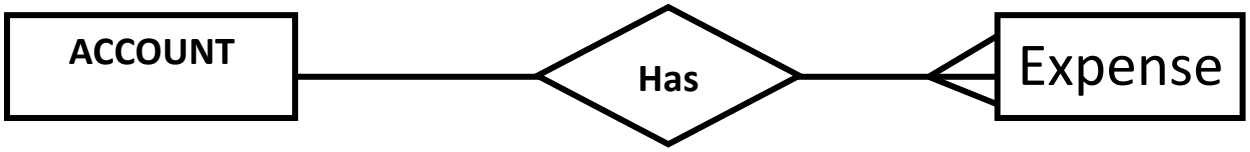


Figure (b)

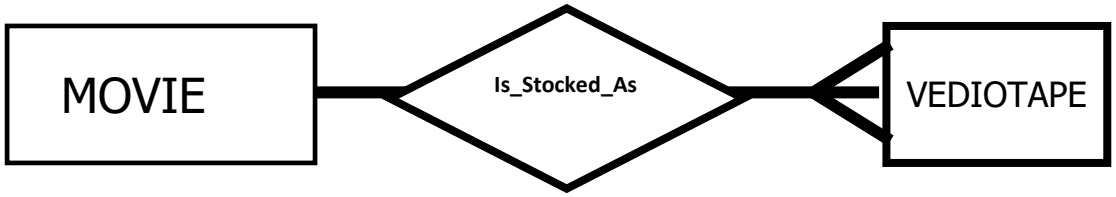
b- E-R Model with only necessary entities.

10.8. Cardinality Constraints :

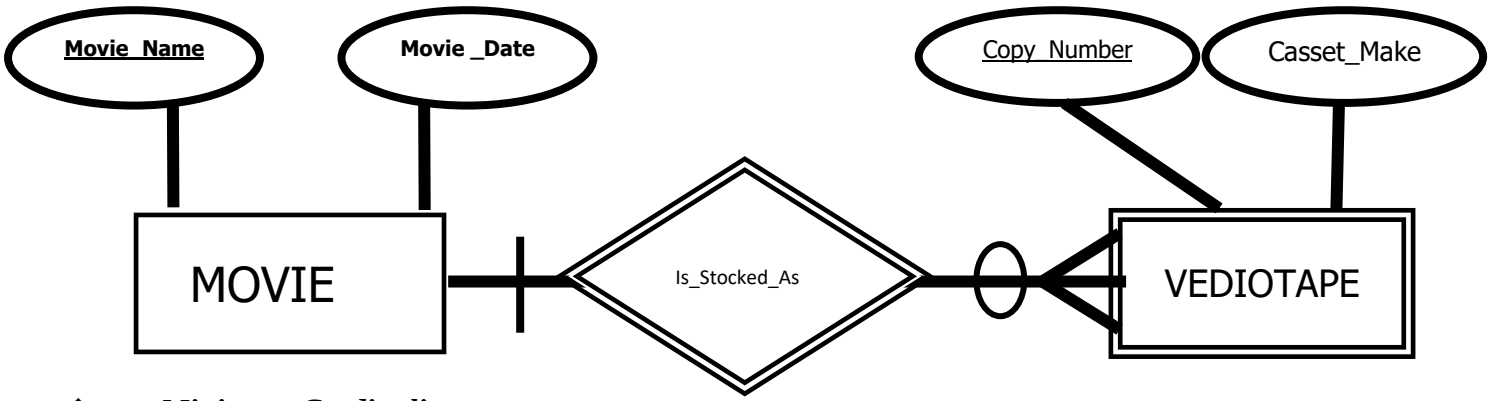
Specifies the number of instances of one entity that can (or must) be associated with each instance of another entity :

Introducing cardinality Constrains :

A. Basic relationship:



B. Relationship with cardinality constrains:



❖ ***Minimum Cardinality:***

The minimum number of instances of one entity that may be associated with each instance of another entity. When the minimum number of participants is zero, we say that the entity is an *optional participant* in the relationship.

❖ ***Maximum Cardinality :***

The maximum number of instances of one entity that may be associated with a single occurrence of another entity.

**Note :

➢ If the maximum cardinality is zero , participation is *optional* ,if the minimum cardinality is one , participation is *mandatory* .

- If the minimum and maximum are both one , this is called *mandatory one* cardinality

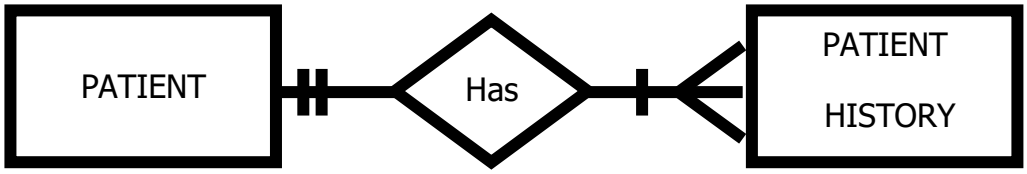
Some Examples :

1. PATIENT has PATIENT HISTORY , each patient has one or more patient history , each instance of PATIENT HISTORY "belongs to" exactly one PATIENT.

2. EMPLOYEE is assigned to PROJECT each PROJECT has at least one EMPLOYEE assigned to it (some projects have more than one). Each EMPLOYEE may or may not be assigned to any existing PROJECT, or may assigned to one or more PROJECT.

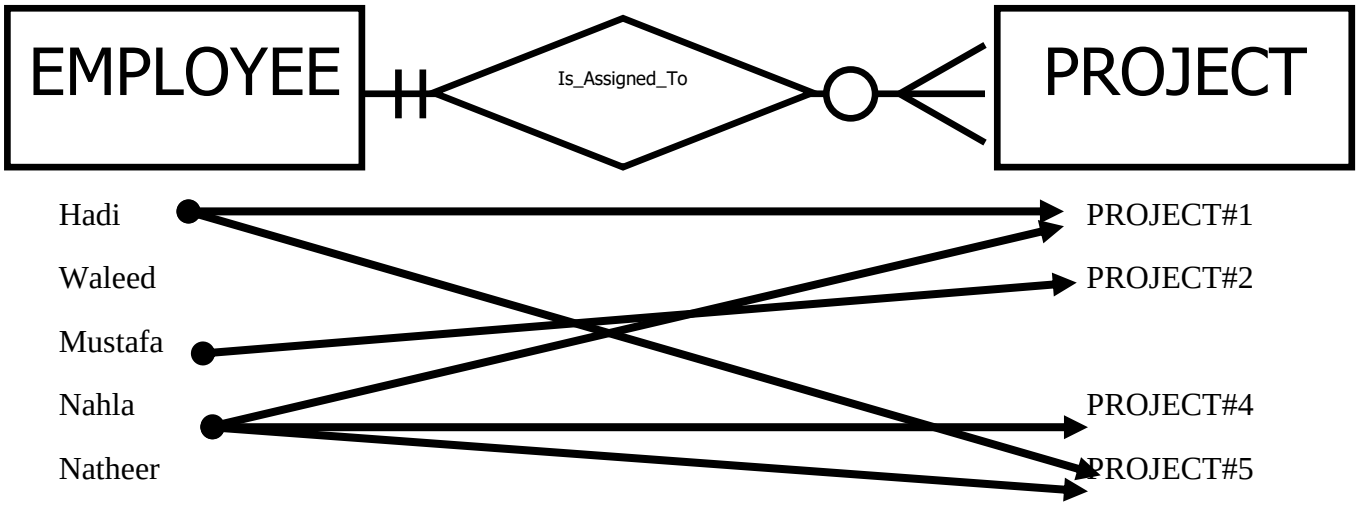
3. PERSON Is_Married _to PERSON. This is an optional zero or one cardinality in both directions. Since a person may or may not be married.

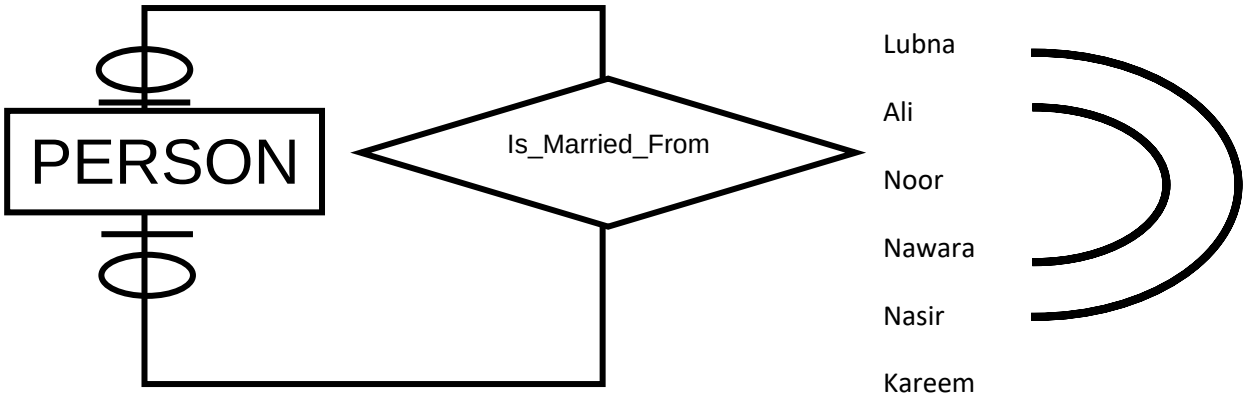
Examples of Cardinality Constrains



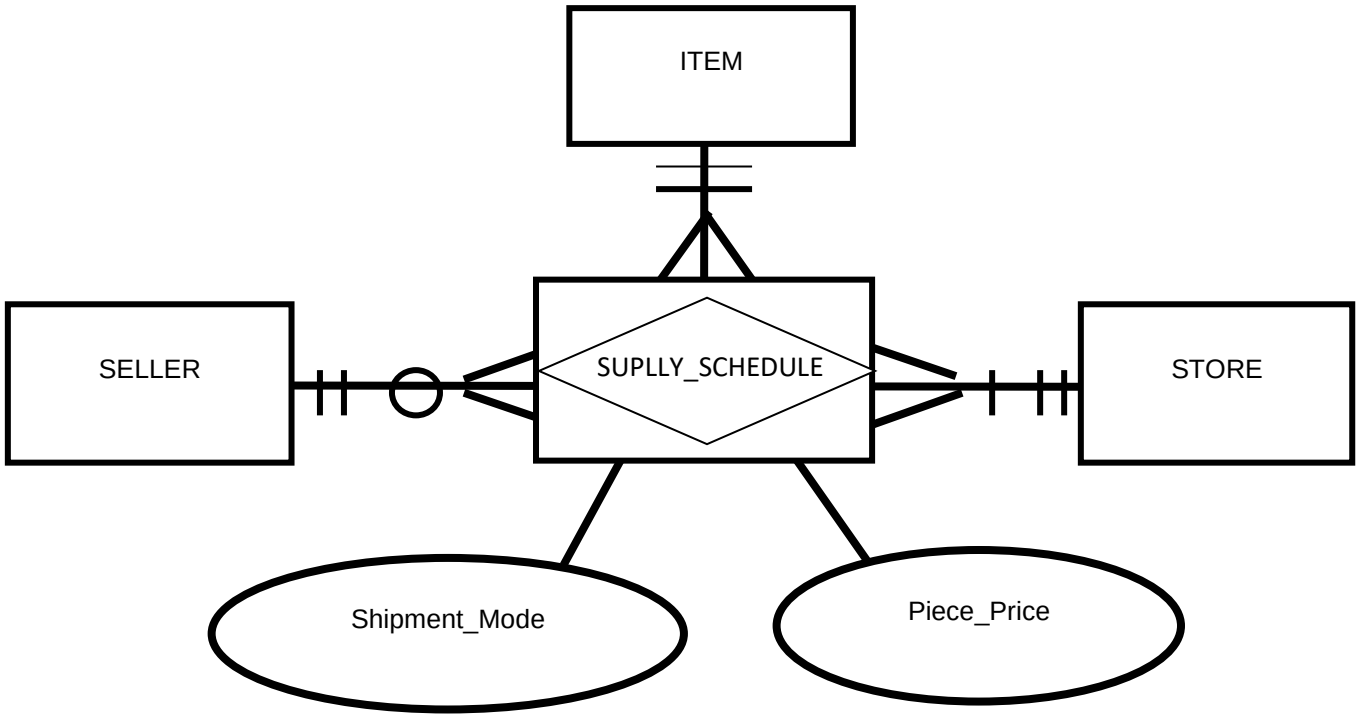
Mandatory Cardinalities.

One optional , one mandatory Cardinalities.





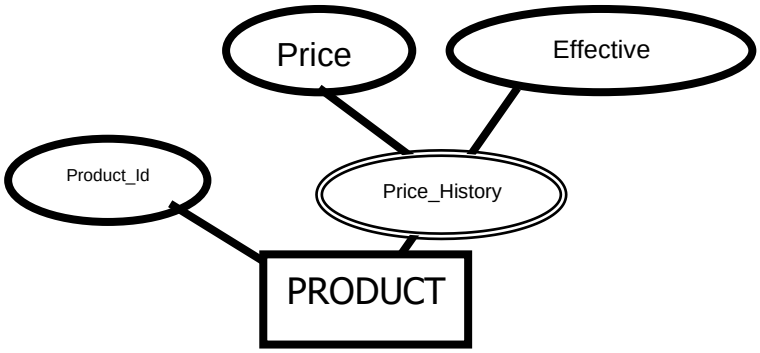
A ternary relationship (Add cardinality constraints to this diagram):



Time Stamp :

A time value that associates with a data value .

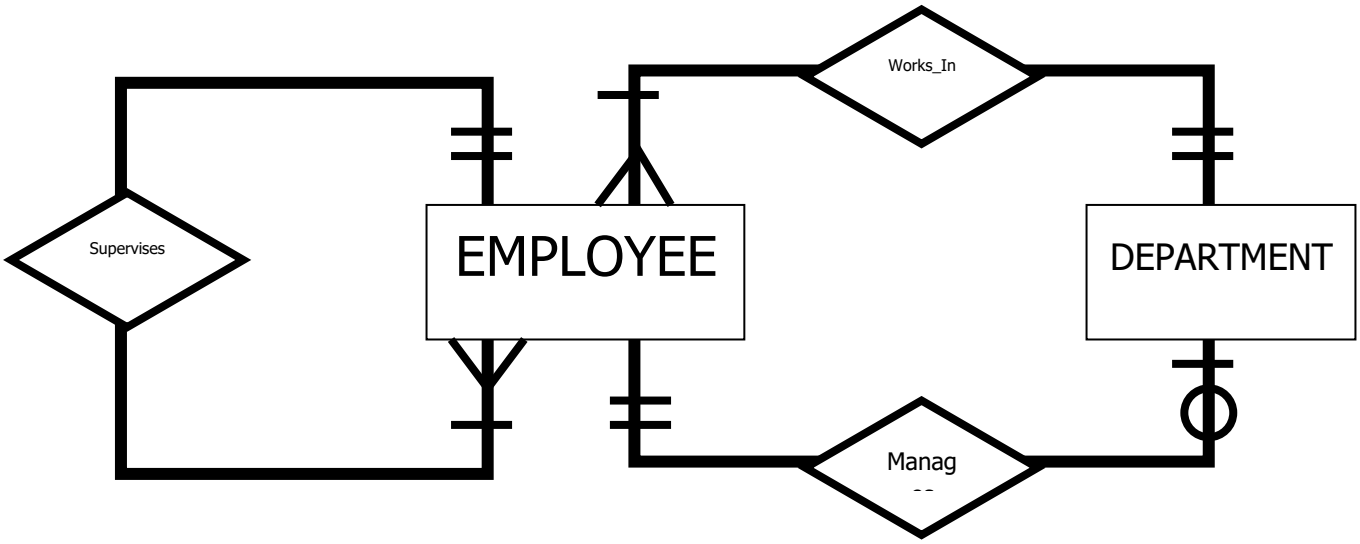
A time stamp may be associated with any data value that changes over time when ew need to maintain a history of those data values.



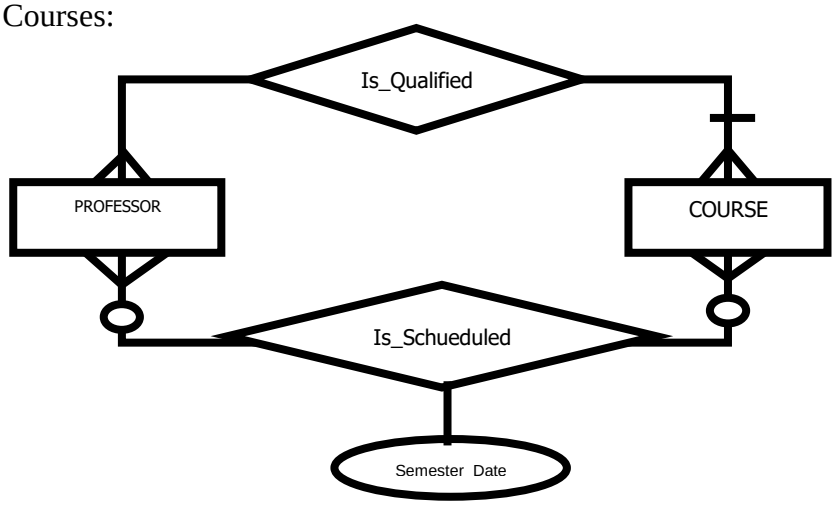
Multiple Relationships: An organization of more than one relationship between the same entity type .

Example:

a. Employees and departments:



b. Professors and Courses:



11. The Enhanced E-R Model and Business Rules :

Enhanced Entity – Relationship Model : (EER):

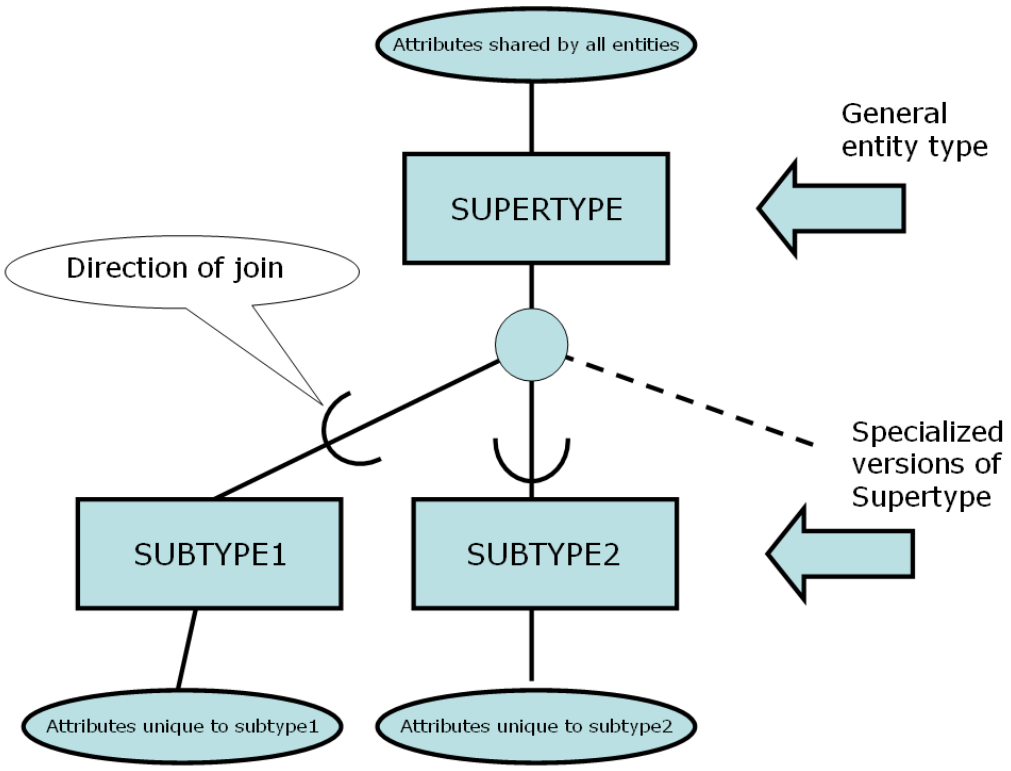
Is used to identify the model that has resulted from extending the original E-R Model with new modeling constraints .

The most important new modeling constraint incorporated in the EER Model is Supertype / Subtype relationship .

This facility allows us to model a general entity type (called the Supertype) and then subdivide it into several specialized entity type called subtype.

Representing Supertype and Subtypes:

- **Subtype:** a subgrouping of the entities in an entity type that is meaningful to the organization and that share common attributes or relationship distinct from other sub groupings.
- **Supertype**: is a generic entity type that has a relationship with one more subtype.

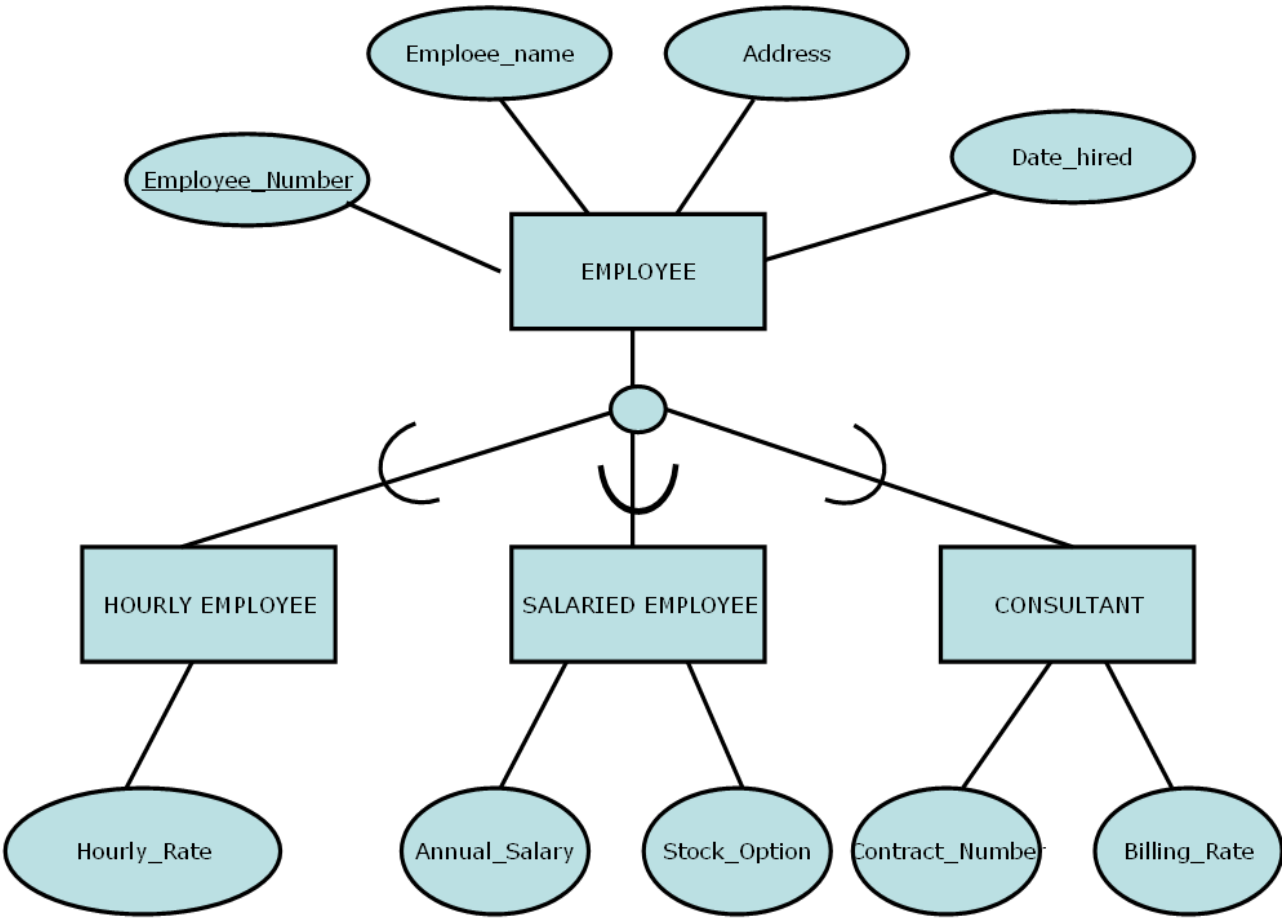


In order to develop the Data Model Concept for the EMPLOYEE example, where we have three different kinds of employees, we have three choices:

1. Define a single entity type called EMPLOYEE, this approach has the disadvantage that EMPLOYEE would have to contain all of the attributes for the three types of employees . for an instance of an hourly – employee attributes such as annual - salary and contract - no would be null or not used when taken to development environment , programs that use this entity type would necessarily be quite complex to deal with the many variations.

2. Define a separate entity type for each of the three entities, this approach would fail to exploit the common properties of employees, and users would have to be careful to detect the correct entity when using the system.

3. Define a Supertype called EMPLOYEE with Subtypes for: HOURLY- EMPLOYEE, SALARIED – EMPLOYEE and CONSULTANT, this approach exploits the common properties of all employees, yet recognized the distinct properties of each type.



Attributes Inheritance:

A property that subtypes entity inherit values of all attributes of the Supertype .

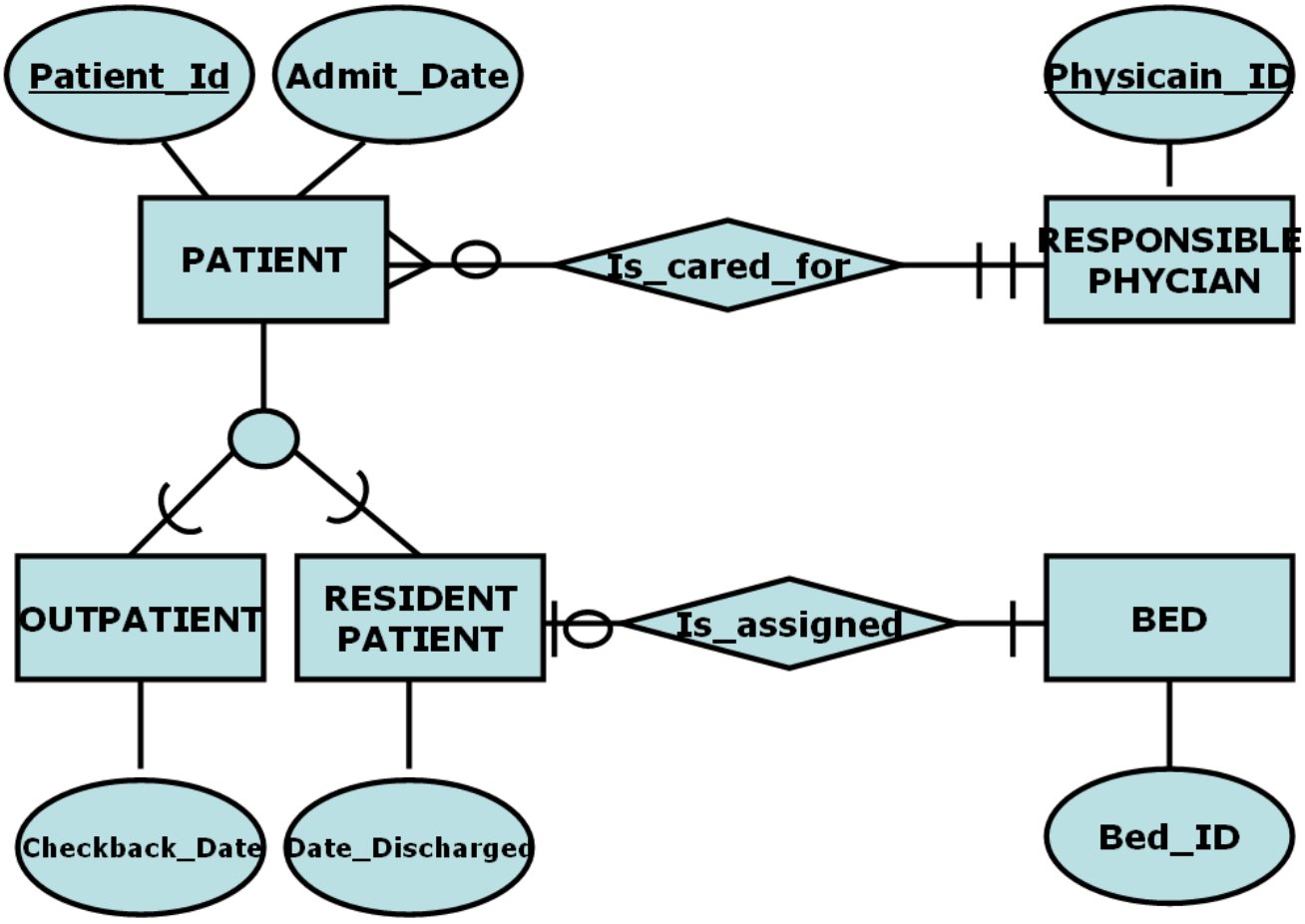
Any attribute in the Supertype does not need to re-declare it in the subtype , its value came from the Supertype .

Q: when to use the Supertype / Subtype Relationship?

You should consider using subtypes when either (or both) of the following conditions are present:

1. There are attributes that apply to some (but not all) of the instances of any entity type. ex : EMPLOYEE entity type

2. The instances of subtype participate in a relationship unique to that subtype .



ex : the hospital entity type PATIENT

11.1. Representing Specialization and Generalization:

There are two processes that serves as mental models in developing

Supertype/ Subtype relationships.

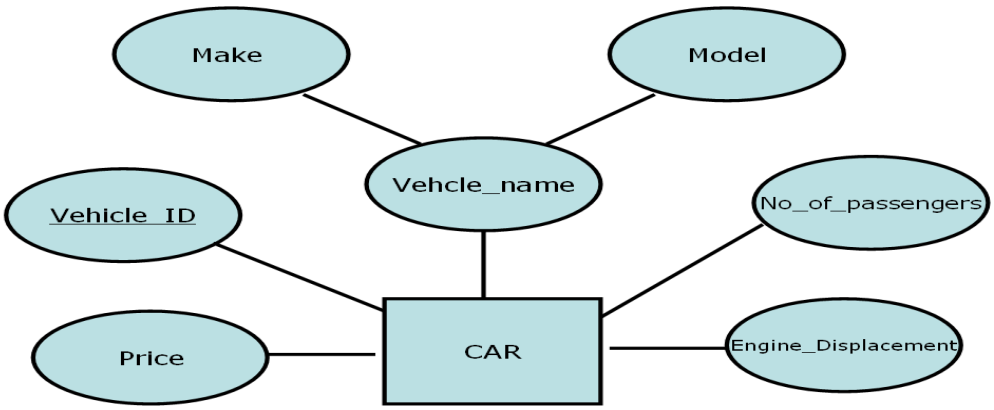
1. **Generalization**: it is a bottom – up process of defining a more general entity type from a set of more specialized entity type.

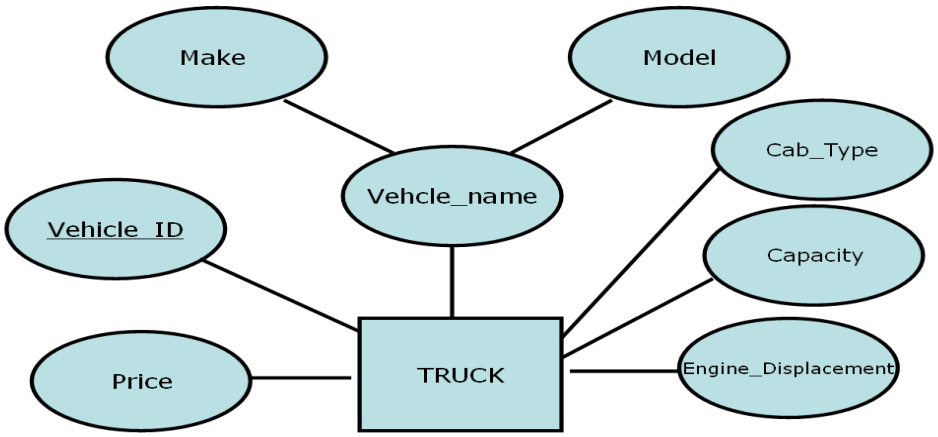
Examples for it: CAR, TRUCK, MOTORCYCLE.

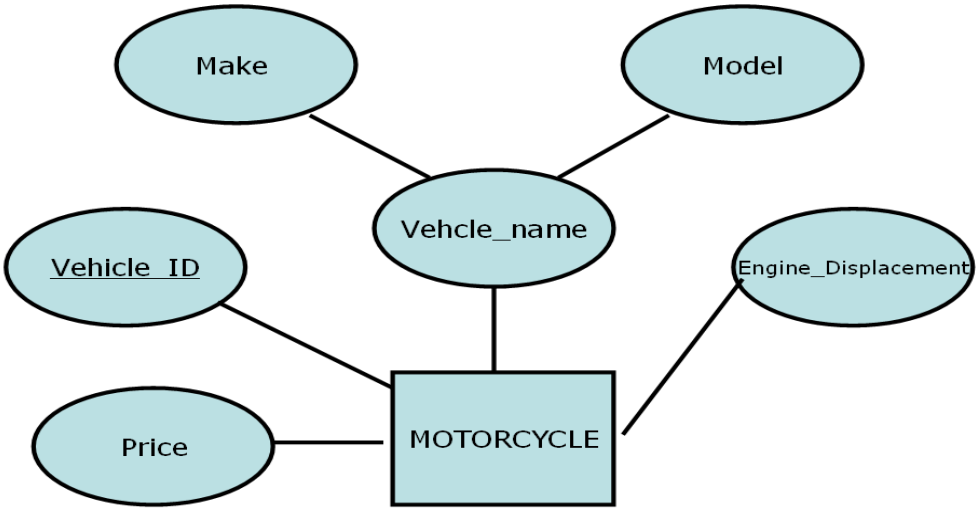
2. **Specialization:** it is atop – down process f defining one or more subtype relationship.

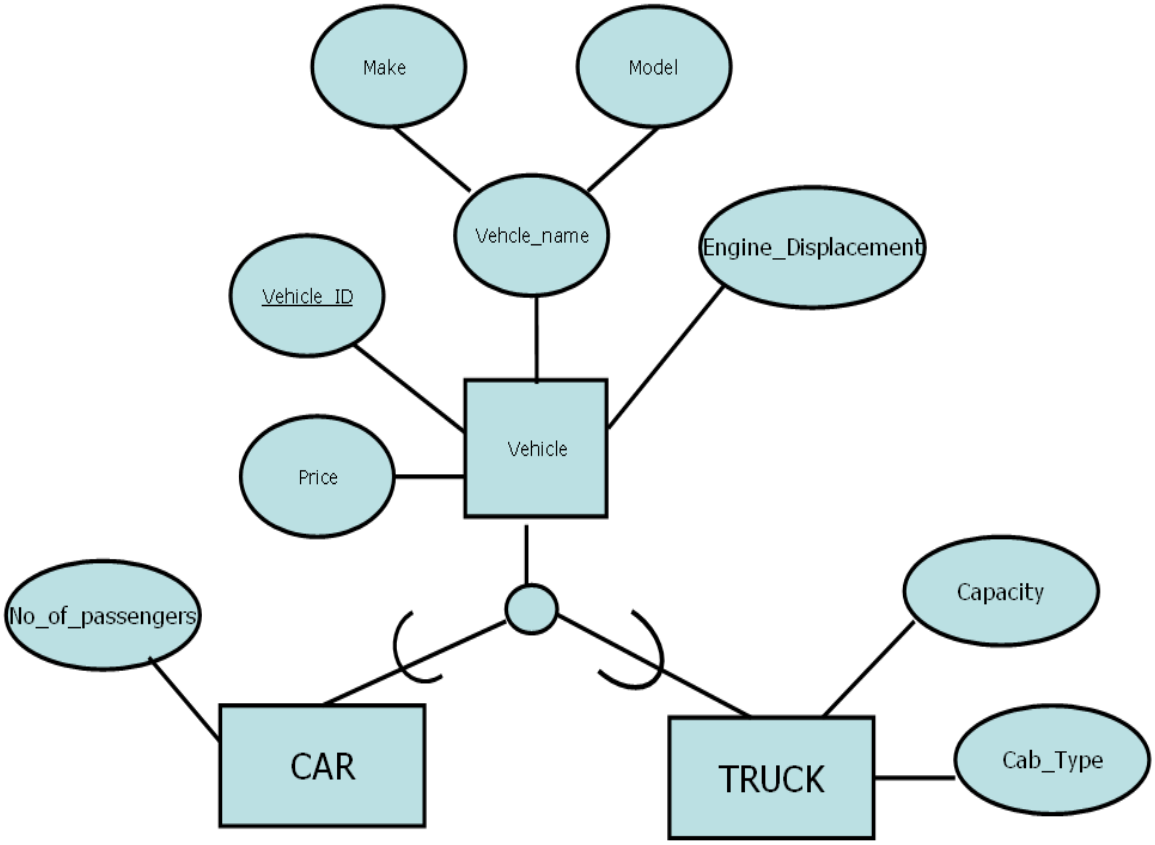
Examples for it: PART

A → Example for generalization:

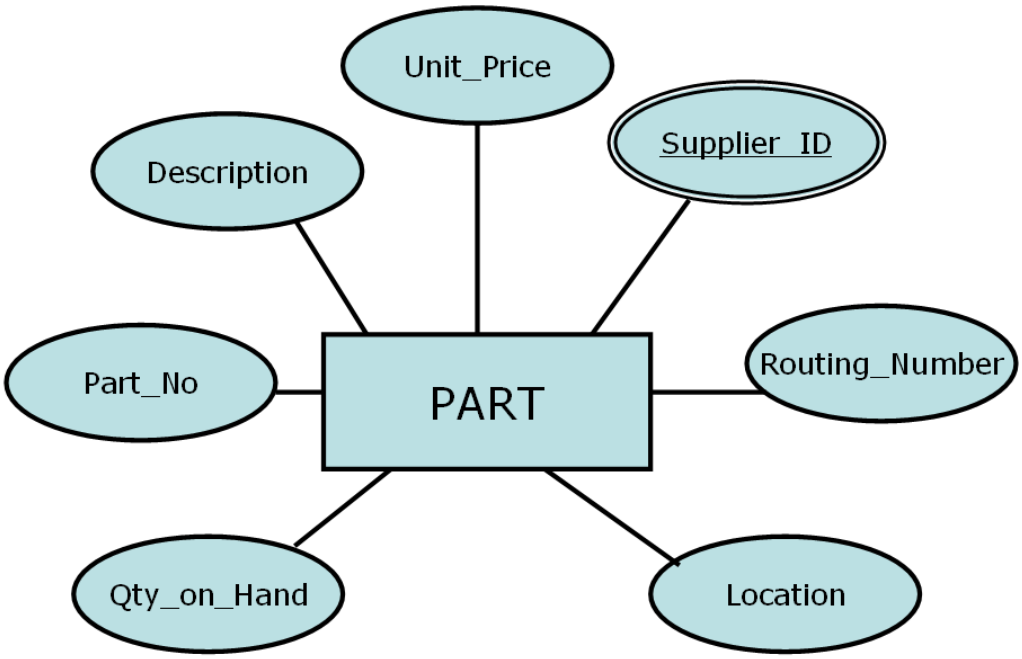


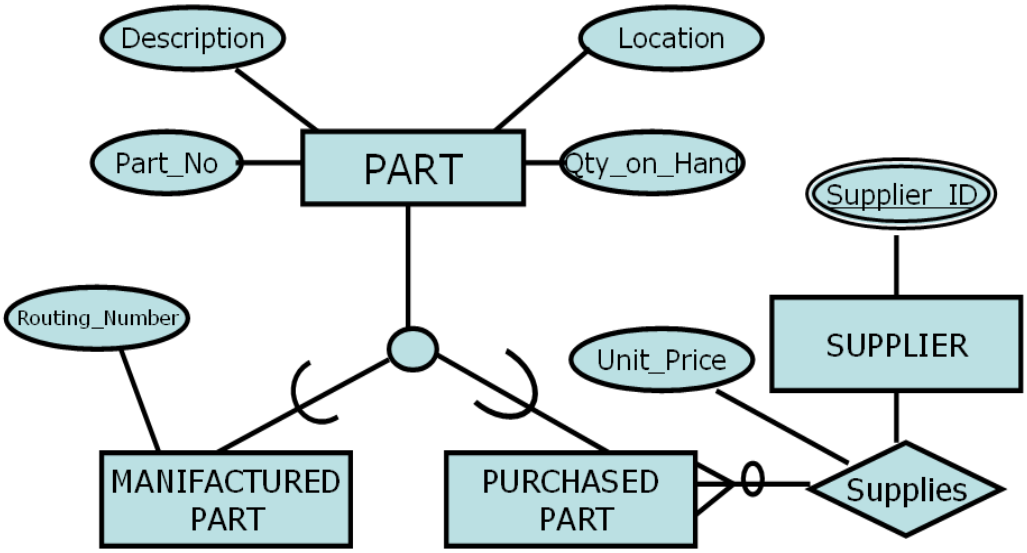



Make
Model
Vehcle_name
Vehicle_ID
Engine_Displacement
Price
MOTORCYCLE


Make
Model
Vehcle_name
Engine_Displacement
Vehicle_ID
Vehicle
Price
No_of_passengers
Capacity
CAR
TRUCK
Cab_Type

Example for Specialization :





11.2. Specifying Constraints in Supertype/Subtype Relationships:

We introduce notation to represent constraints on Supertype / subtype relationship . These constraints allow us to capture some of the important business rules that apply to these relationships.

The two most important types of constraints are :

1. Completeness Constraints.

2. Disjointness Constraints.

1. Specifying Completeness Constraints:

■ **Completeness Constraints** : a type of constraints that address the question whether an instance of Supertype a member of at least one subtype must be also.

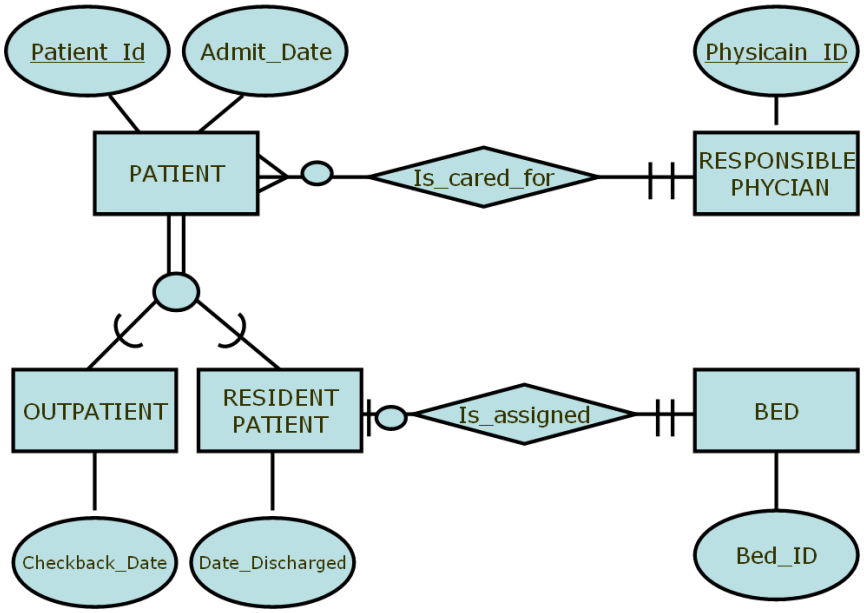
It has two possible rules :

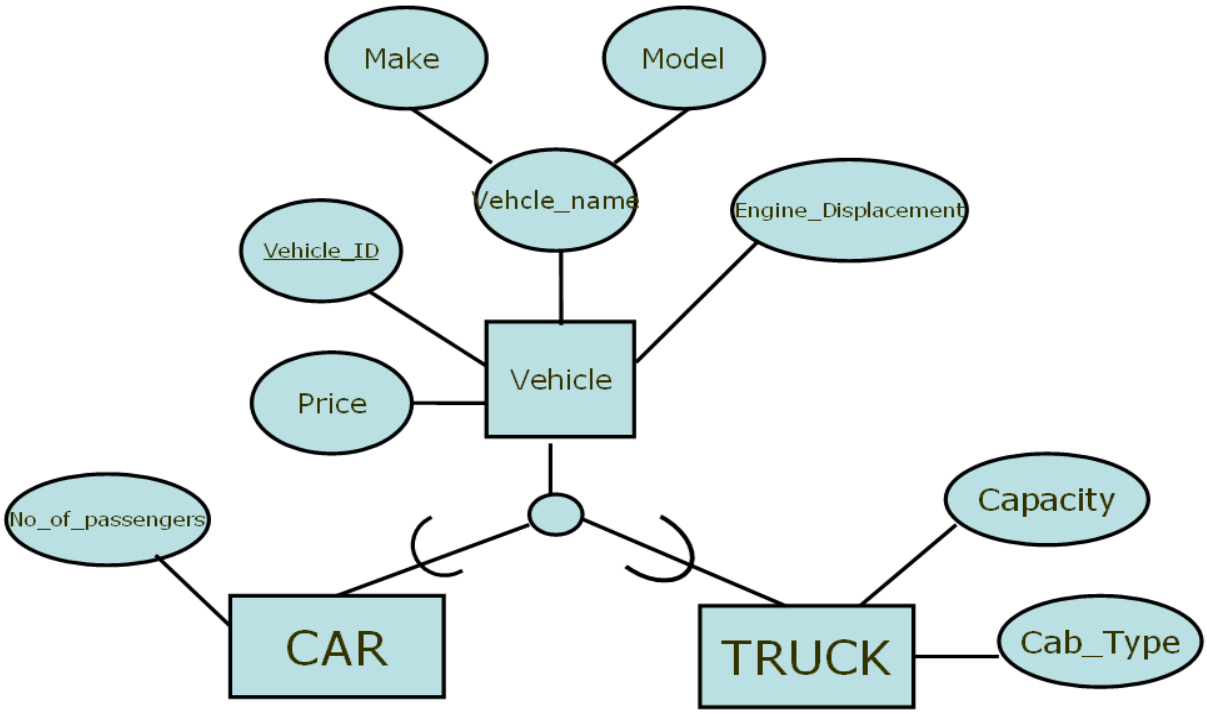
A. Total specialization rule: specifies that each entity instance of super type must be member of some subtype in the relationship, Ex: EMPLOYEE,

Represented by Double line.

B. Partial specialization rule: specifies that an entity instance of super type is allowed not to belong to any subtype. Ex: VEHICLE

Represented by Single line.





11.3. Specifying Disjointness Constraints

Disjointness Constraints: a constraint that addresses the question whether an instance of Supertype member of two (or more)subtypes may simultaneously be. It has two possible rules :

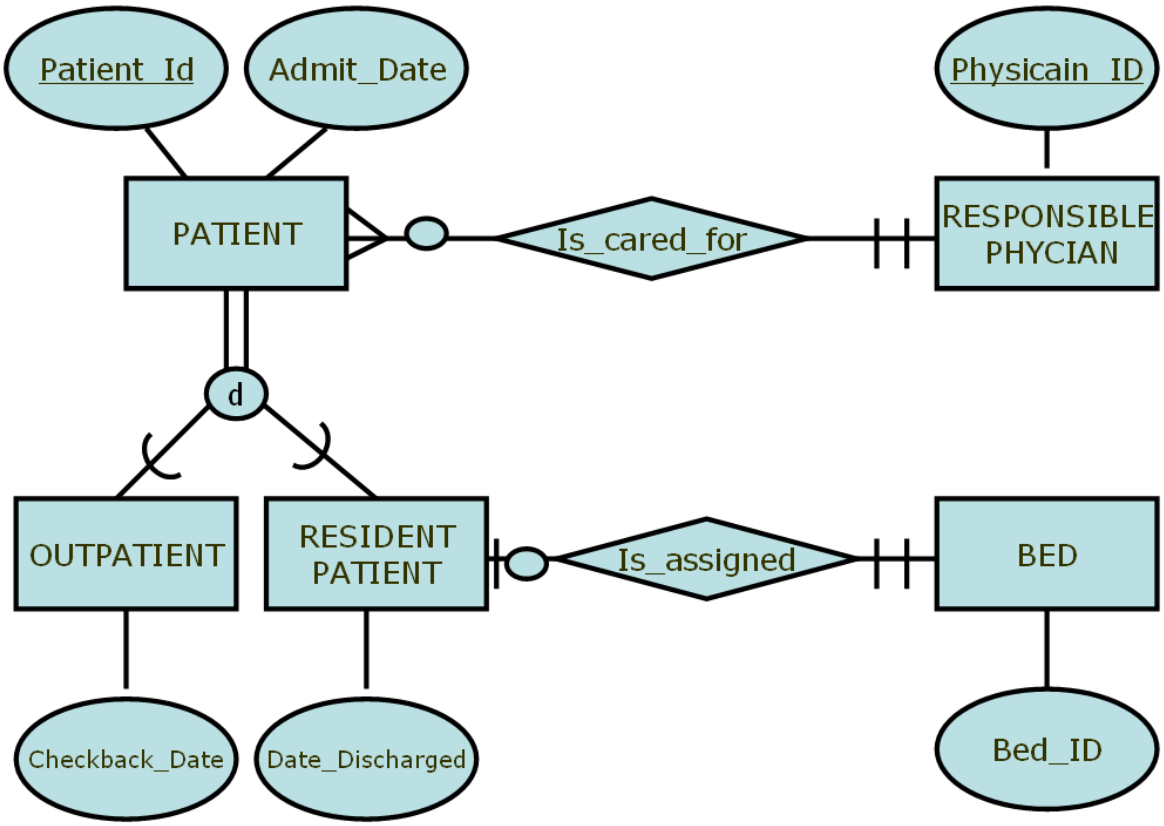
A. The disjoint rule specifies that if an entity instance (of the same Supertype) ,is a member of one subtype , it ***cannot simultaneously*** be a member of any other subtype.

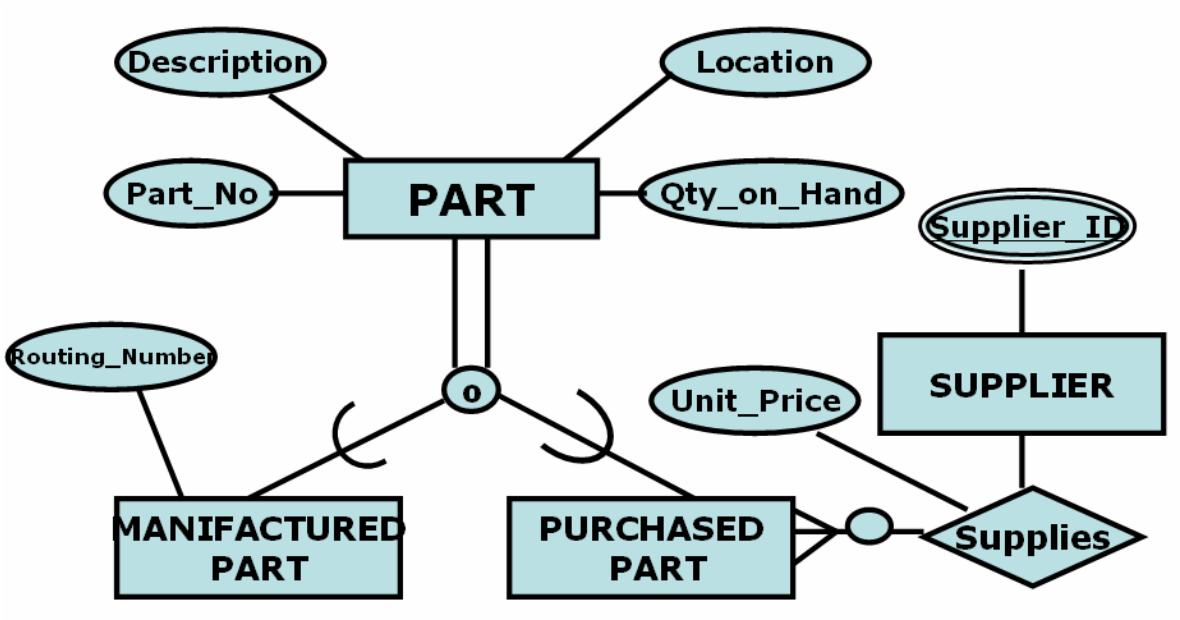
Ex: PATIENT

Represented by letter (d) inside the circle that joint Supertype with subtype.

B. The overlap rule: specifies that any entity instance ***can simultaneously*** **be a member of two or more subtypes.**
Ex: PERT .

Represented by letter (o) inside the circle that joint Supertype with subtype.


Patient_Id
Admit_Date
Physicain_ID
PATIENT
Is_cared_for
RESPONSIBLE PHYCIAN
d
OUTPATIENT
RESIDENT PATIENT
Is_assigned
BED
Checkback_Date
Date_Discharged
Bed_ID


Description
Location
Part_No
PART
Qty_on_Hand
Supplier_ID
Routing_Number
o
Unit_Price
SUPPLIER
MANIFACTURED PART
PURCHASED PART
Supplies

11.4. Defining Subtype Discriminators

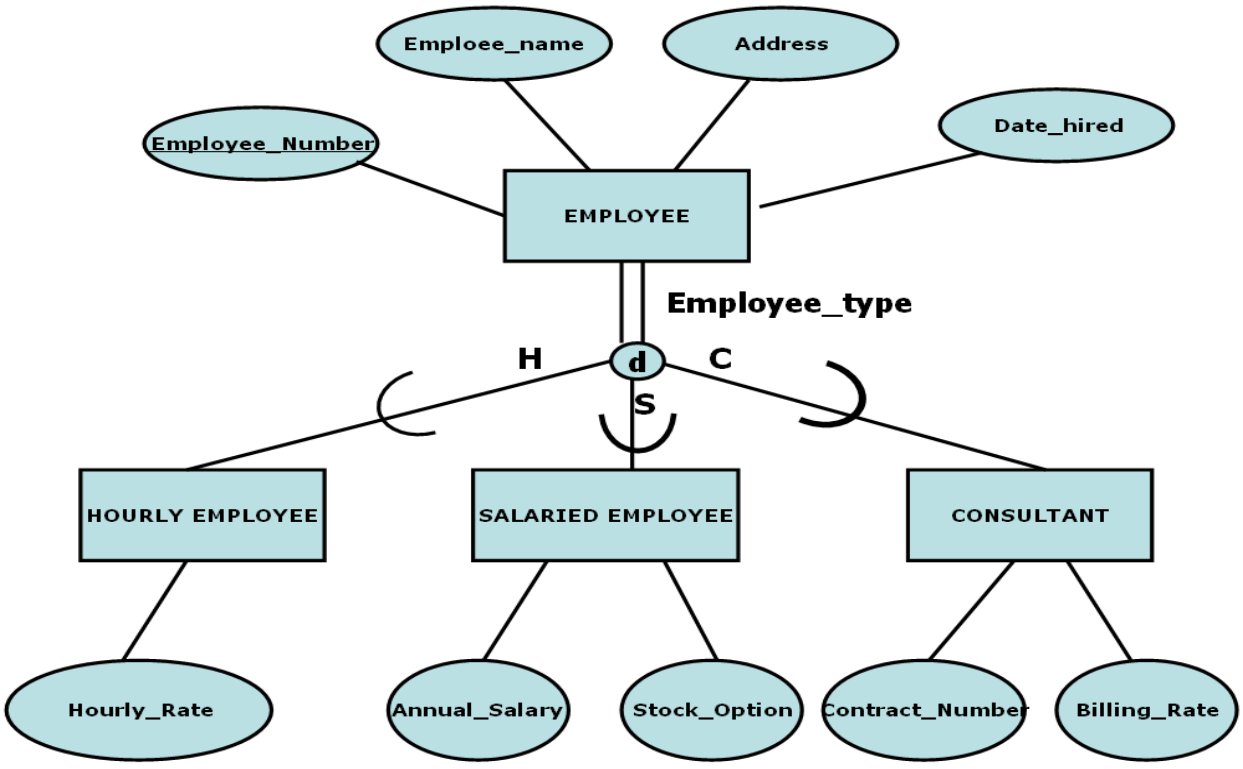
A problem occurred when new instance added to the Supertype , so we'll discuss some rules , one of them are the subtype discriminators .

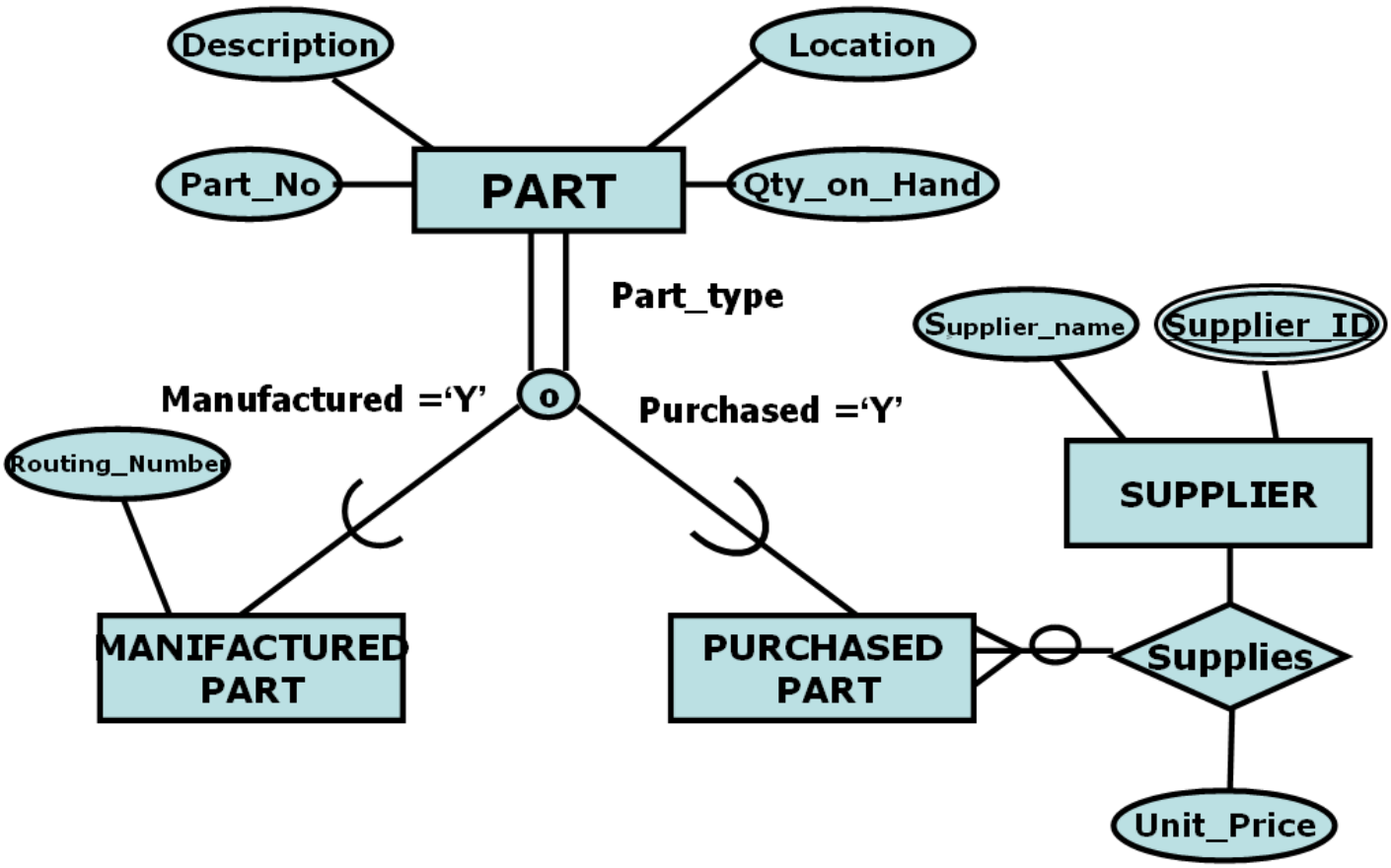
Subtype Discriminator: is an attribute of Supertype whose values determine the target subtype or subtypes. It has two types:

1. Disjoint Subtypes:
Ex: EMPLOYEE , here a new attribute Employee_Type added to the Supertype , the condition put to the left side of the line .

2. Overlap Subtype:
Ex: **PART** , here a new attribute Part_Type added , which is a composite attribute , represented by logical expression :

Type of part	Manufactured?	Purchased?
Manufactured only	'Y'	'N'
Purchased only	'N'	'Y'
Manufactured& Purchased	'Y'	'Y'

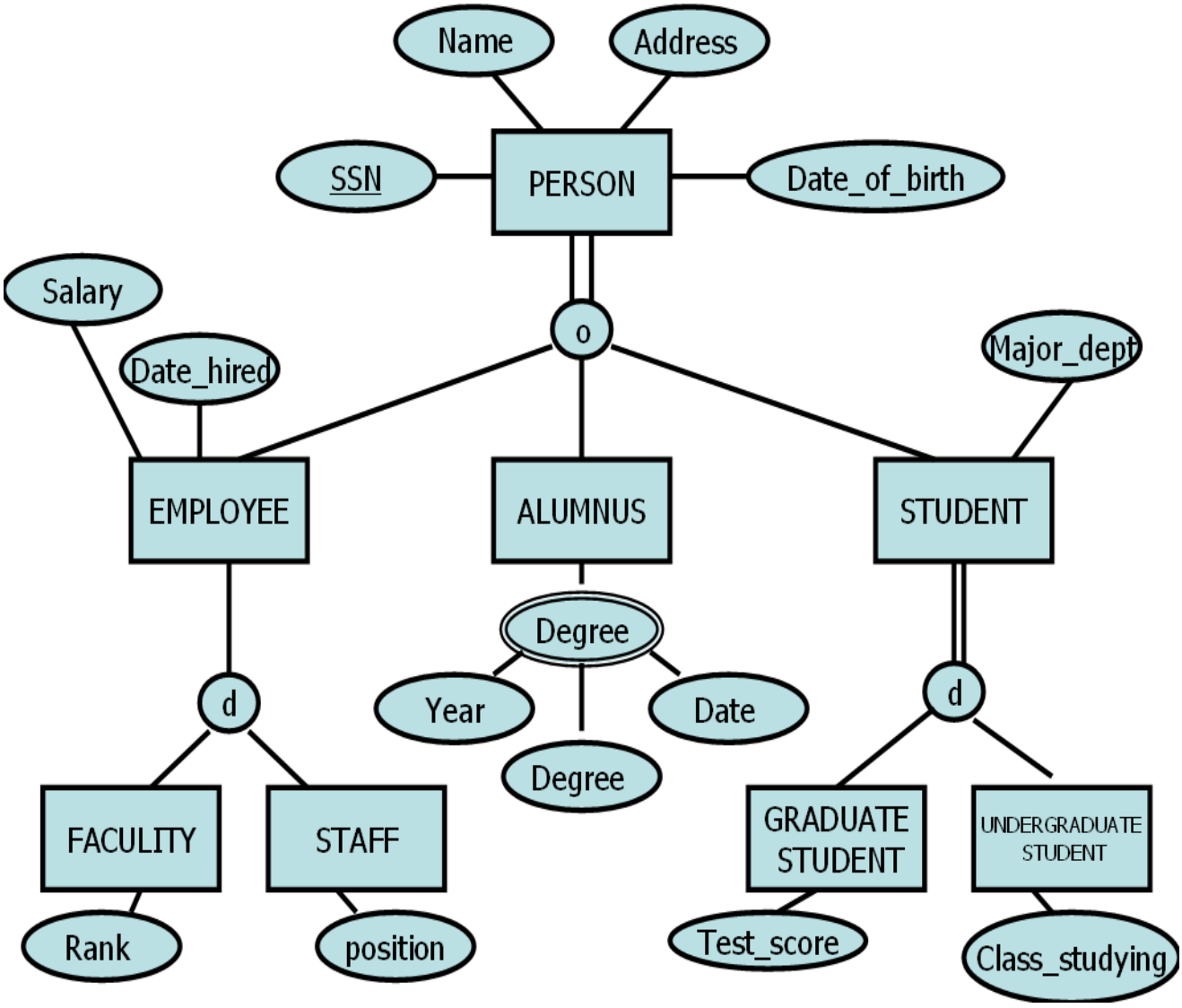




11.5. Supertype/Subtype Hierarchy :

The hierarchal arrangement of Supertypes and subtype where each subtype has only one Supertype .

- Attributes in the root are for all its subtypes, also the subtype if it has Supertype attributes. They will be to all the subtypes.
- Subtypes inherit all the attributes in the Supertype connected directly (or indirectly) with it .


Name
Address
SSN
PERSON
Date_of_birth
Salary
Date_hired
o
Major_dept
EMPLOYEE
ALUMNUS
STUDENT
Degree
d
Year
Date
d
Degree
FACULITY
STAFF
GRADUATE STUDENT
UNDERGRADUATE STUDENT
Rank
position
Test_score
Class_studying