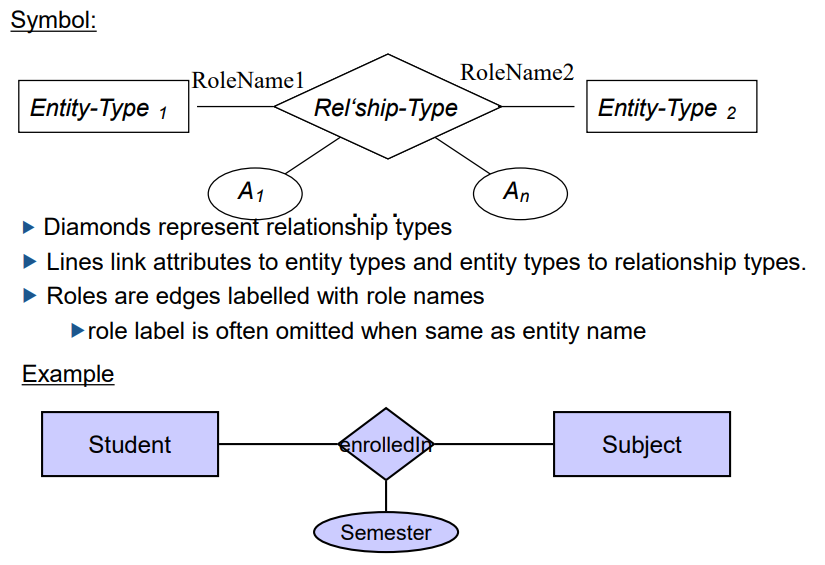
# Conceptual Data Model

Aim: specification of database schema

## Conceptual design

* Technique for understanding and capturing business **information requirements graphically**
* Depiction of association among **different entity** within business or information system

It does not define **how it is** **implemented** but it focuses on **inter-relational links**

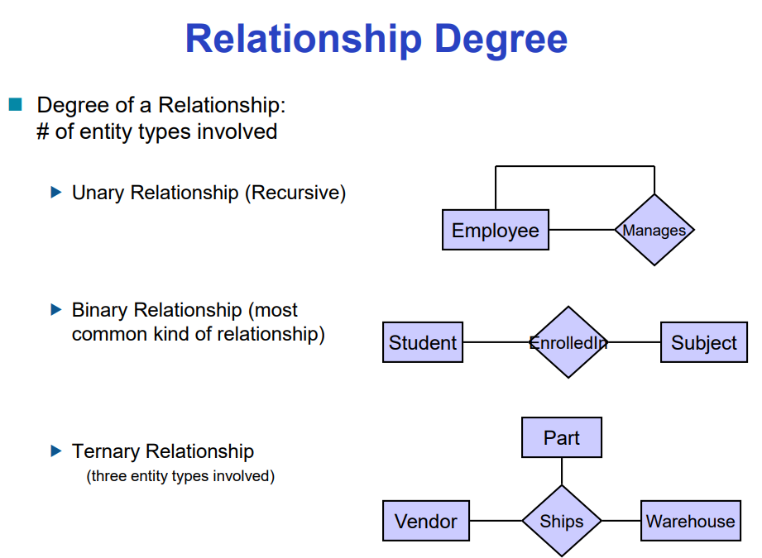
**Model & database independent**

# Entity Relationship Model

* Graphical representation of
  + what data needs to be contained in the system
  + association among different categories/entity

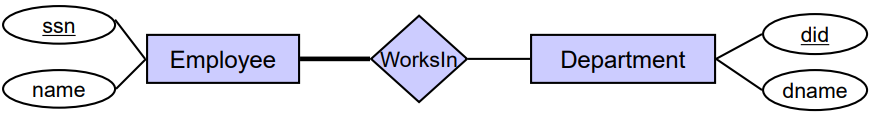
## ERD

There are three objects that govern ERD

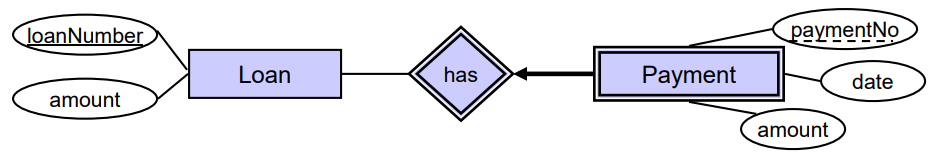
* Entity type : Rectangle
* Attribute : Ellipses
* Relationship : Diamond

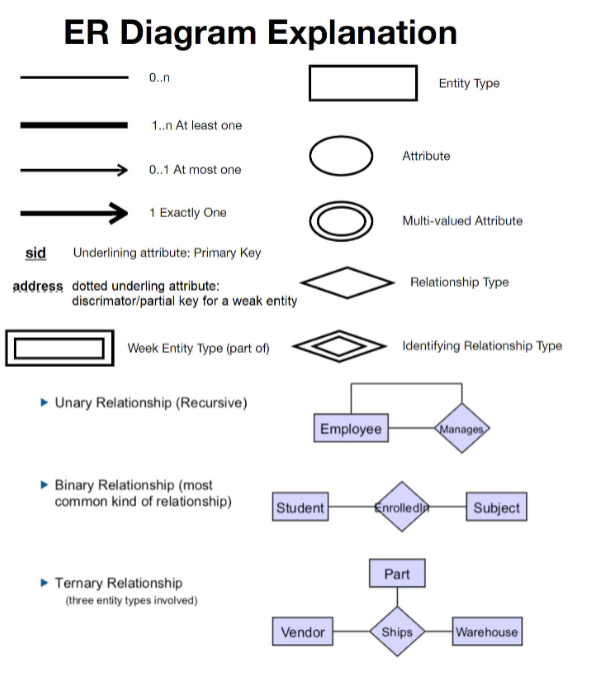
## Multiplicity / Constraints

**Key Constraint**

* AT MOST ONE
* Many-to-one
* ****Thin arrow

|  |  |
| --- | --- |
| **Participation constraint**   * AT LEAST ONE   + **Total participation**   + **Partial participation** * Depicted by a **thick line**   **Participation and key constraint**   * EXACTLY ONE * Depicted by **thick arrow** | **Cardinality Constraints**   * Restrictions on participation domain * **Number of participation** restriction   Employee works at **1~3** department  Department has **0~** employees  **Weak Entity**   * When existence of an entity **depends** on existence of another * Entity that doesn’t have **primary key** * Showed by double rectangle * **Discriminator** is a attribute that distinguishes among weak entities |



****

#Rows = Cardinality

#Columns = Degree of relation

# Conversion from ERD to SQL

Types of Attributes

|  |  |  |
| --- | --- | --- |
| Simple Attribute | Composite Attribute | Multi-value attribute |
| Simple Attribute for an entity | When attribute has attributes on themselves | When attribute holds more than one values |
| Directly onto the relation | Flattened out and added as a separate attribute columns | Separated into a table with a foreign key taken from superior entity |

## Weak entity

* Becomes a separate relation with foreign key taken from superior entity **NOT NULL**

## Multiplicity

* Many to many
  + Make a relation as a separate table
* One to many
  + Put foreign key with **NOT NULL** constraint
* One to One
  + Put it as an attribute value, or put both **foreign and primary** as **UNIQUE**

# OOP implementations

## Specialization

When you want to refer to the Sub-class

## Generalization

When you want to refer to the Superclass

## ISA Constraints

Overlap Constraints

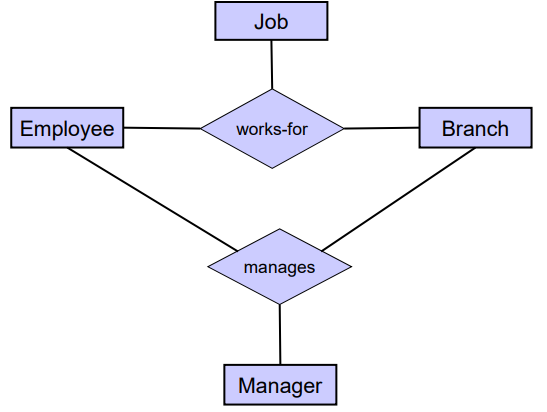
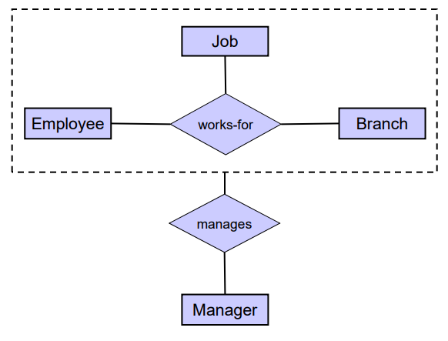
* **Disjoint**
  + Entity can belong to **only one** lower level entity set
* **Overlapping** default
  + entity can belong to **more than one** lower level entity set

Covering Constraints

* **Total** 
  + Entity must belong to one of the lower level entity set
* **Partial** 
  + Entity does not need to belong to the lower level entity set

## Aggregation

Aggregation allows any part of a relationship to become an **abstract entity** for the purpose of participating in another relationship



## Conversion from ISA Hierarchy to SQL

Super class’s primary key also becomes subclass’s primary key, (foreign key)