Information Management II

9. Weak and Enhanced Entity Modelling

Yvette Graham

CSU34041

Today’s Lecture

- Weak Entities
- Enhanced Entity Relationships (EERs)
- Constraints and Specialisations
- Mapping EERs to relations

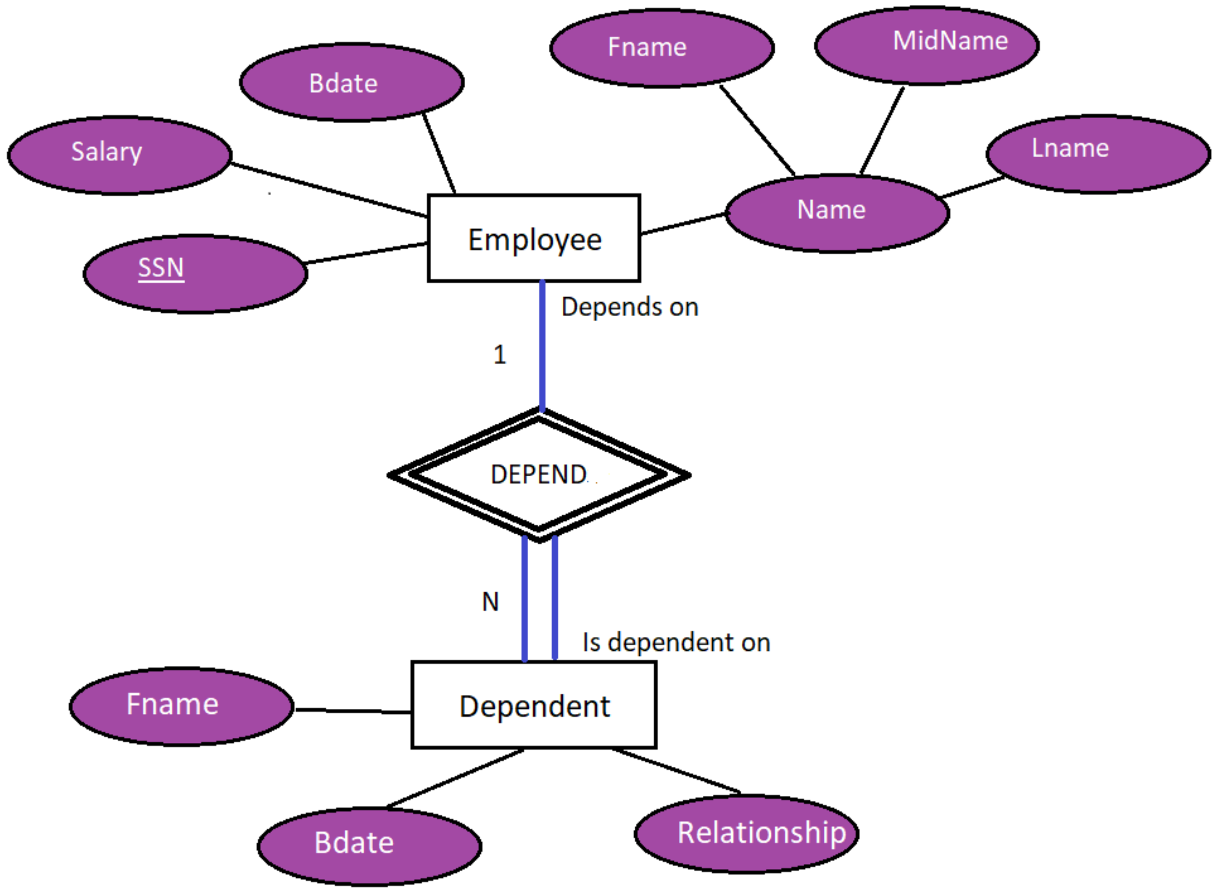
Weak Entities

Weak Entities

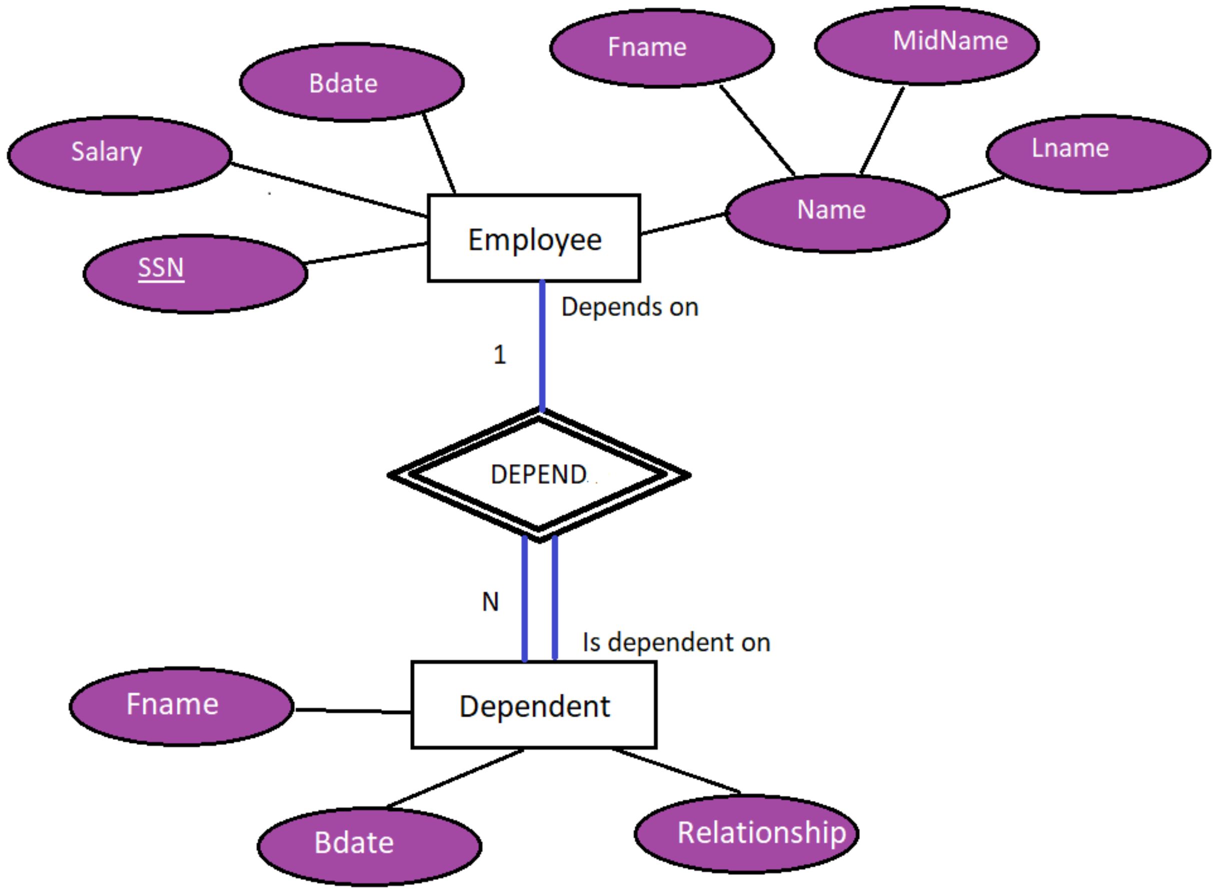
- Entities which do not have key attributes are called "**weak entity** types". It is an entity whose tuples cannot be identified by the attributes in that entity.
- Such entities are identified by being related to specific entities from another entity type in combination with one of their own attribute values
- This other entity is called the "**identifying** or **owner entity**"
- The relationship which relates the weak entity to the identifying entity is called the **identifying relationship** of the weak entity
- ***A weak entity ALWAYS has a total participation constraint*** (i.e. an existence dependency) ***with respect to its identifying relationship*** because the weak entity cannot be identified without an owner entity.

Example

- Suppose we have two entities, an EMPLOYEE entity representing employees of a company and DEPENDENT Entity which represents dependents of each employee
- Suppose each Employee has attributes SSn, Bdate, Salary, …
- And each Dependent has Name, Bdate, Relationship

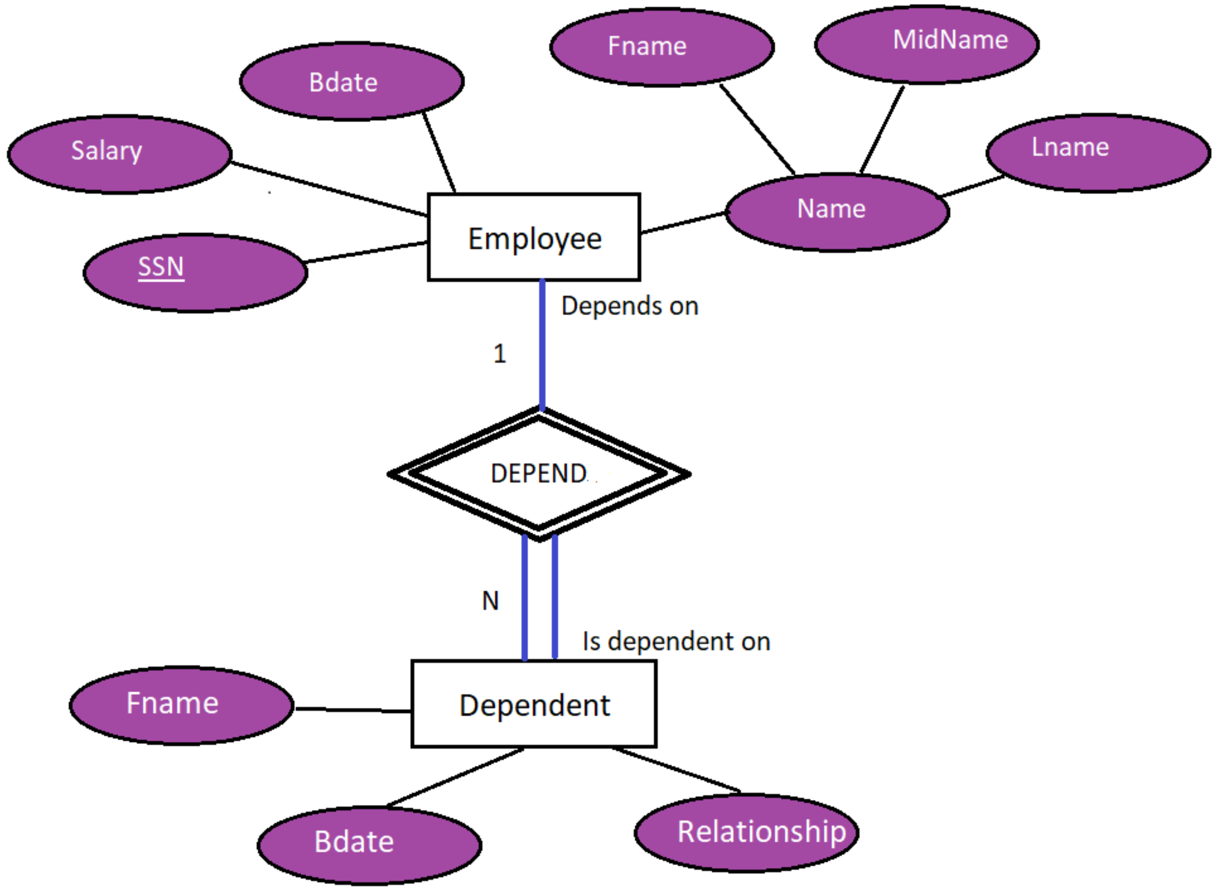


- Note: as two dependents of two different Employees can have the same Fname, Bdate and Relationship → no identifying attribute occurs in the DEPENDENT entity
- Therefore, DEPENDENTS are only identified after determining the particular employee entity to which each dependent is related!
- Each Employee entity is said to **own** the dependent entities it relates to



Weak Entity Type

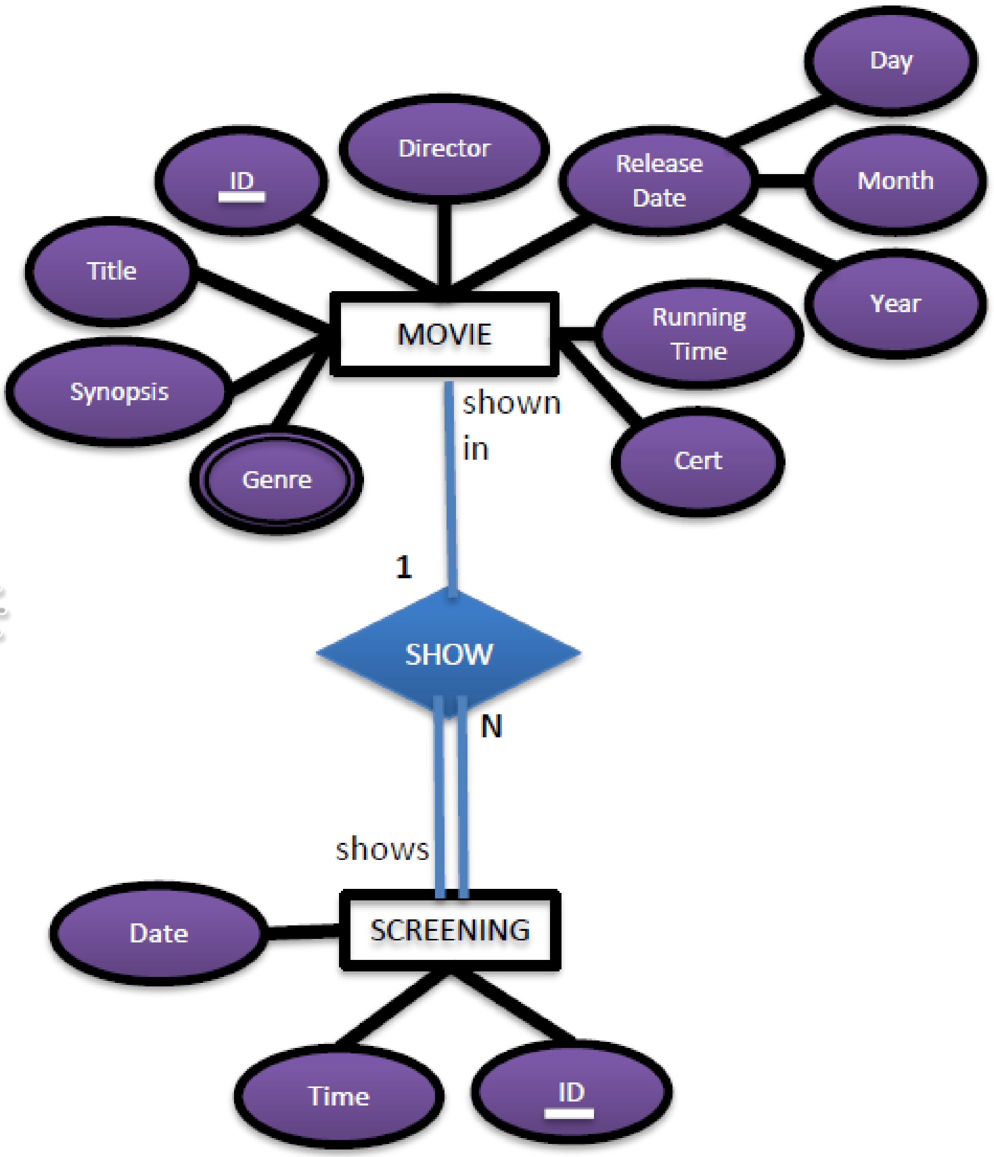
- A Weak Entity normally has a ***partial key –*** which ***is an attribute (or set of attributes) which uniquely identify weak entities that are related to the same owner entity***
- In the example we make the assumption that each dependent of a particular employee has a unique FName. i.e within the family of a employee, no two dependent family members have the same FName



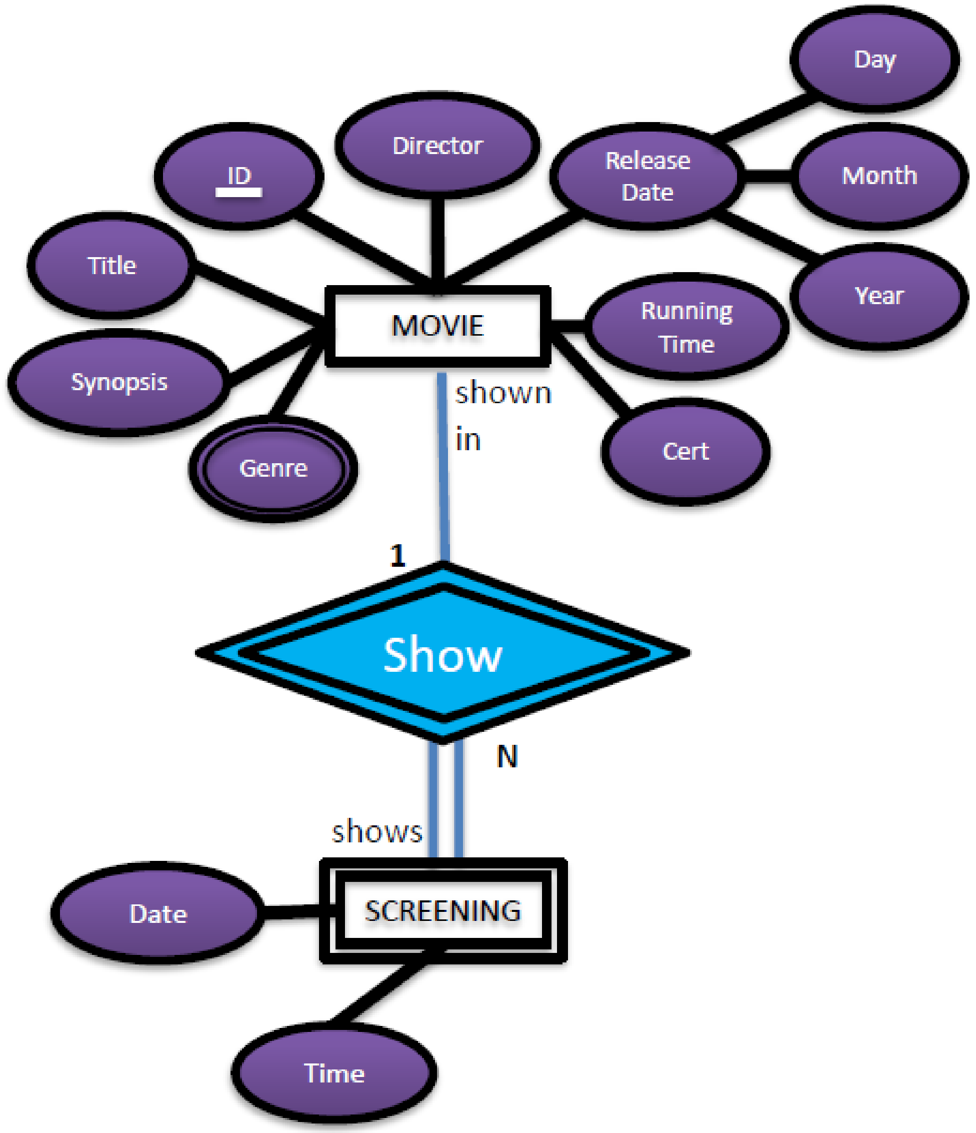
Movie ER Model

- In the Movie ER model (given in a previous lecture on ER Modeling) we could have represented the entity SCREENING as a Weak Entity, with Movie being its identifying or owner entity

ORIGINAL Move EI Model



Alternative Movie ER Model



Enhanced Entity Relationships

Enhanced Entity Relationship

- An enhanced ER or EER, contains all the modelling concepts of ER with the addition of representation of:

– **Subclass**, **Superclass** (and the related concepts of specialization and generalization)

– **Union** (or category) type

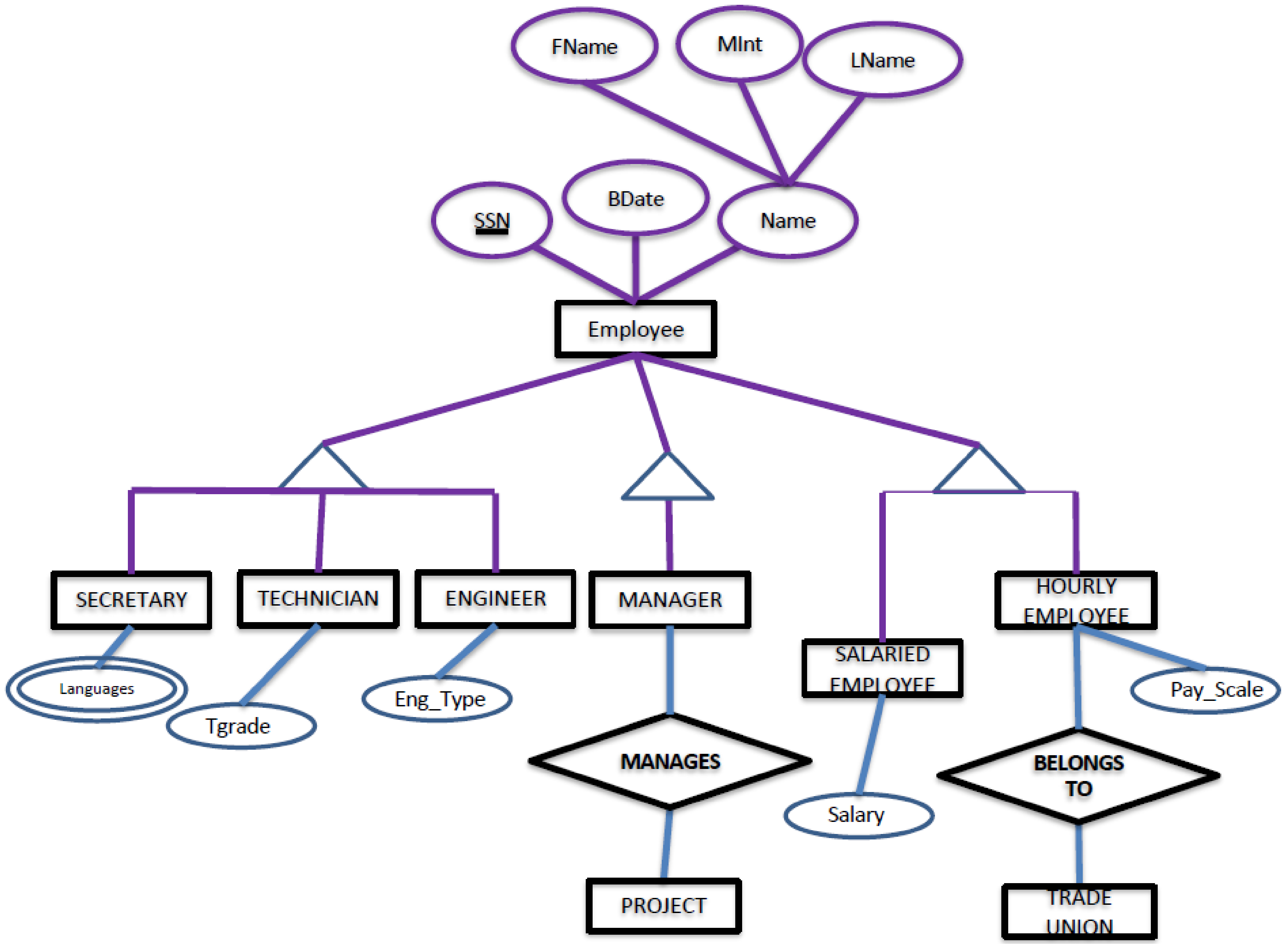
- The above concepts are important mechanisms for representing *attribute* and *relationship inheritance*

Subclass or Subtype

- An entity type is used to represent both a *type of entity* and the *entity set or collection of entities* of that type that exists in the database.
- For example, the entity type EMPLOYEE describes the types (that is the attributes and relationships) of each employee entity, and also refers to the current set of EMPLOYEE entities instances in the database.

Subclass or Subtype

- Suppose we wish to represent subgroups or subtypes of EMPLOYEEs e.g. SECRETARY, ENGINEER, MANAGER, TECHNICIAN, SALARIED_EMPLOYEE, HOURLY_EMPLOYEE
- We call each of these subgroupings a **Subclass** or a Subtype of the EMPLOYEE entity type
- The EMPLOYEE entity type is called a **Superclass** or Supertype of each of these subclasses
- We call the relationship between a superclass and any one of the subclasses, a **Superclass-subclass relationship**
- A superclass/subclass relationship **can also be called an 'Is a' relationship** as every instance of the subclass is also an instance of the superclass


FName
MInt
LName
SSN
BDate
Name
Employee
SECRETARY
TECHNICIAN
ENGINEER
MANAGER
SALARIED EMPLOYEE
HOURLY EMPLOYEE
Languages
Tgrade
Eng_Type
MANAGES
Salary
Pay_Scale
BELONGS TO
PROJECT
TRADE UNION

Subclasses and SuperClasses – some things to note

- All entities that are subclasses of a (superclass) entity are also instances of that superclass and therefore inherit he attribute values of an instance of that Superclass entity – e.g. a SECRETARY is also an EMPLOYEE, a MANAGER is also an EMPLOYEE. In other words, an instance of a subclass (e.g. SECRETARY) represents the same real-world instance of a Superclass entity
- An instance of a Subclass cannot only exist as a subclass instance, it MUST also be an instance of its superclass
- That instance may also (optionally) be an instance of any other subclass of that superclass – e.g. an instance of SECRETARY can also be a SALARIED EMPLOYEE. However, its is not necessary that every instance of a superclass is also an instance of a subclass

Subclass/Superclass represent inheritance relationship

- We can say that an ***instance of a subclass inherits all the attributes of the instance of a superclass.***
- That ***subclass also inherits all the relationships in which the superclass participates***
- Going from a more general superclass to subclass can be thought of as Specialisation
- Likewise going from subclass(es) entities to a superclass entity can be thought of as ***Generalisation***

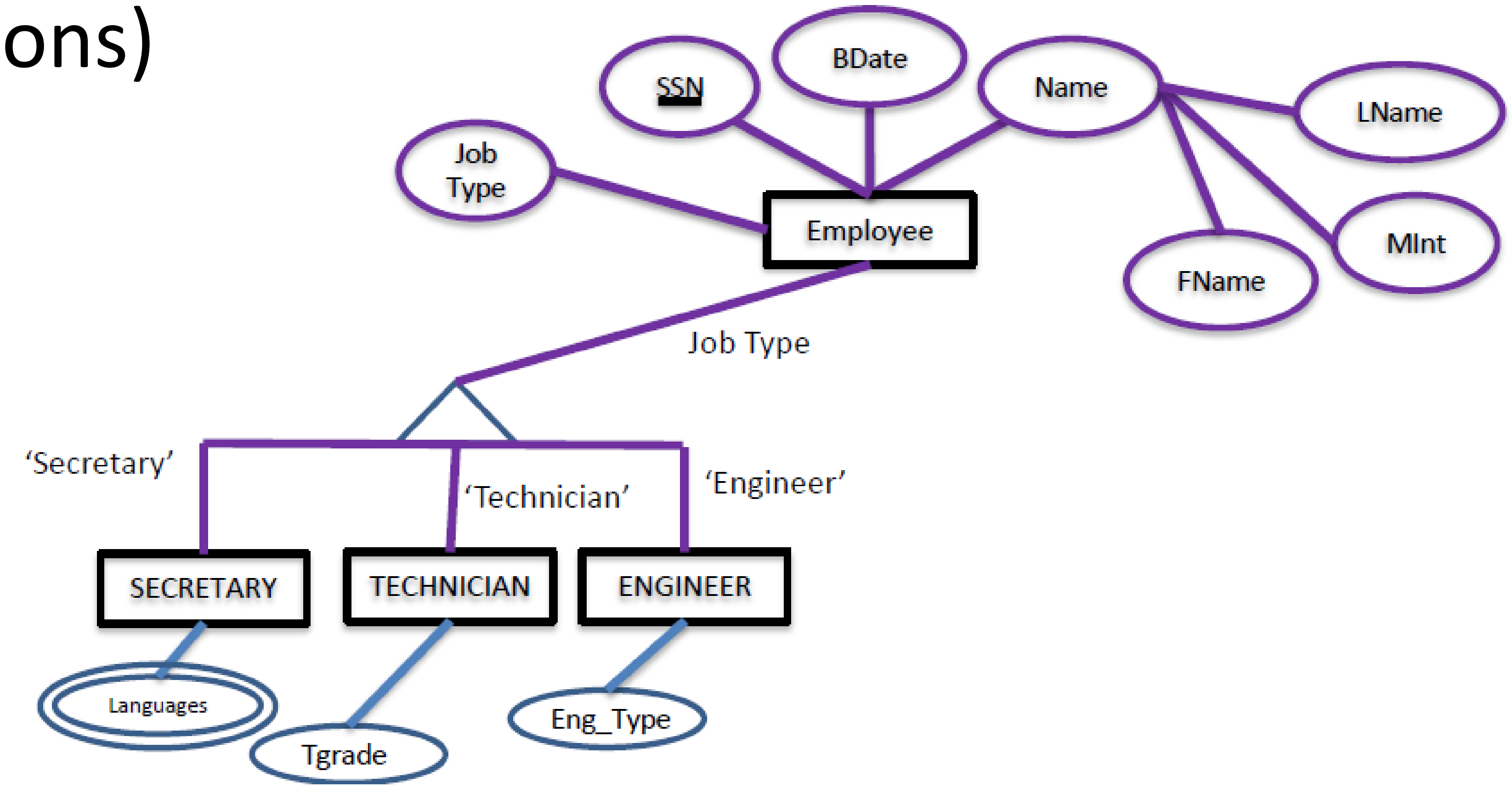
Why represent Superclass entities and subclass entities in the Information model ?

- This representation is useful where certain attributes may apply to some but not all instances of an entity. A subclass can be defined in order to group the instances of the entities to which these attributes apply
- e.g. SECRETARY has attribute Languages but ENGINEER has attribute Eng_Type

Constraints and Specialisations

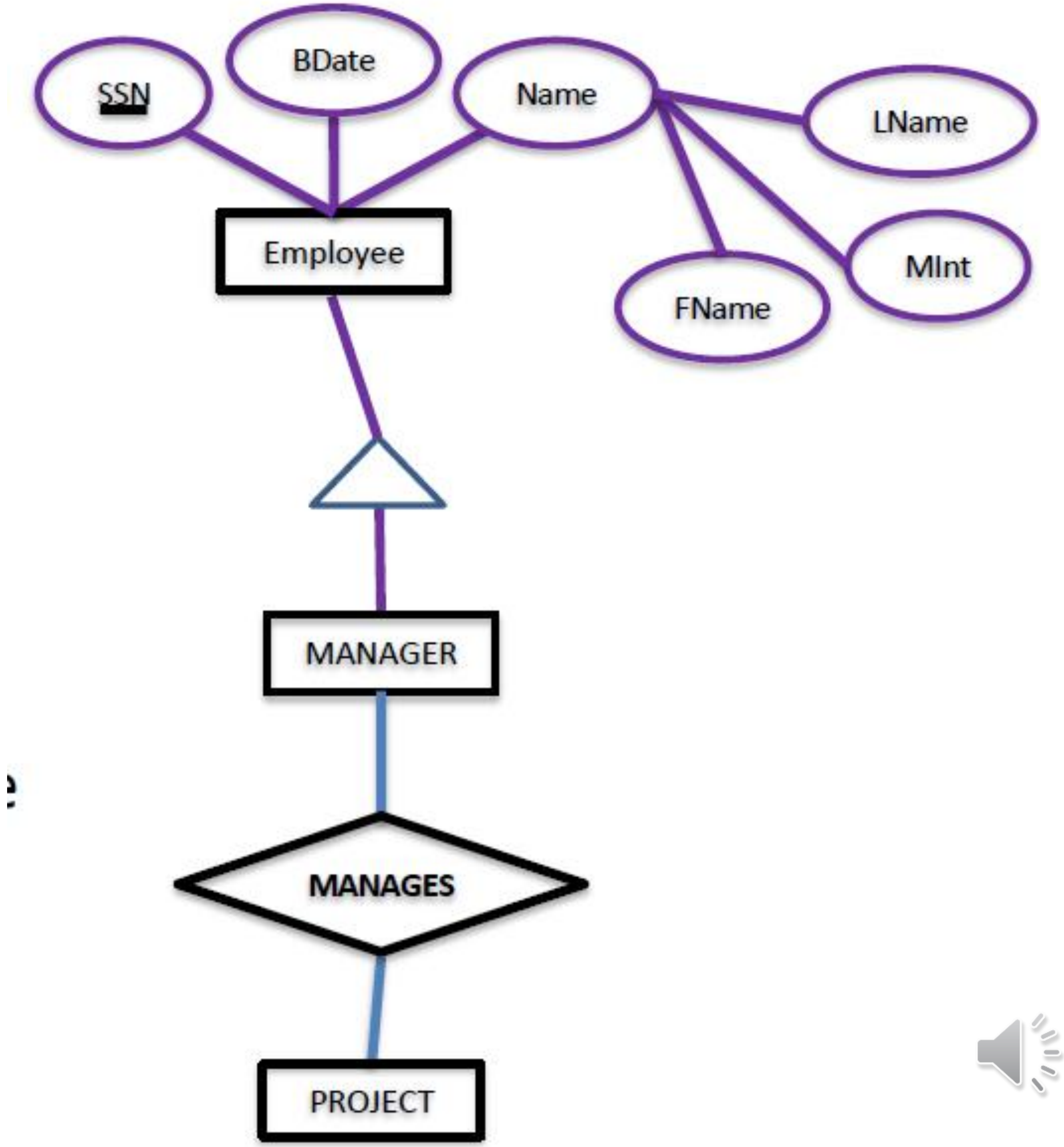
Constraints on Specialisations (and Generalisations)

- In some situations, we can determine exactly the entities that will become instances of each subclass by placing a condition on the value of some attribute of the superclass. Such subclasses are called ***predicate-defined*** or condition-defined subclasses.



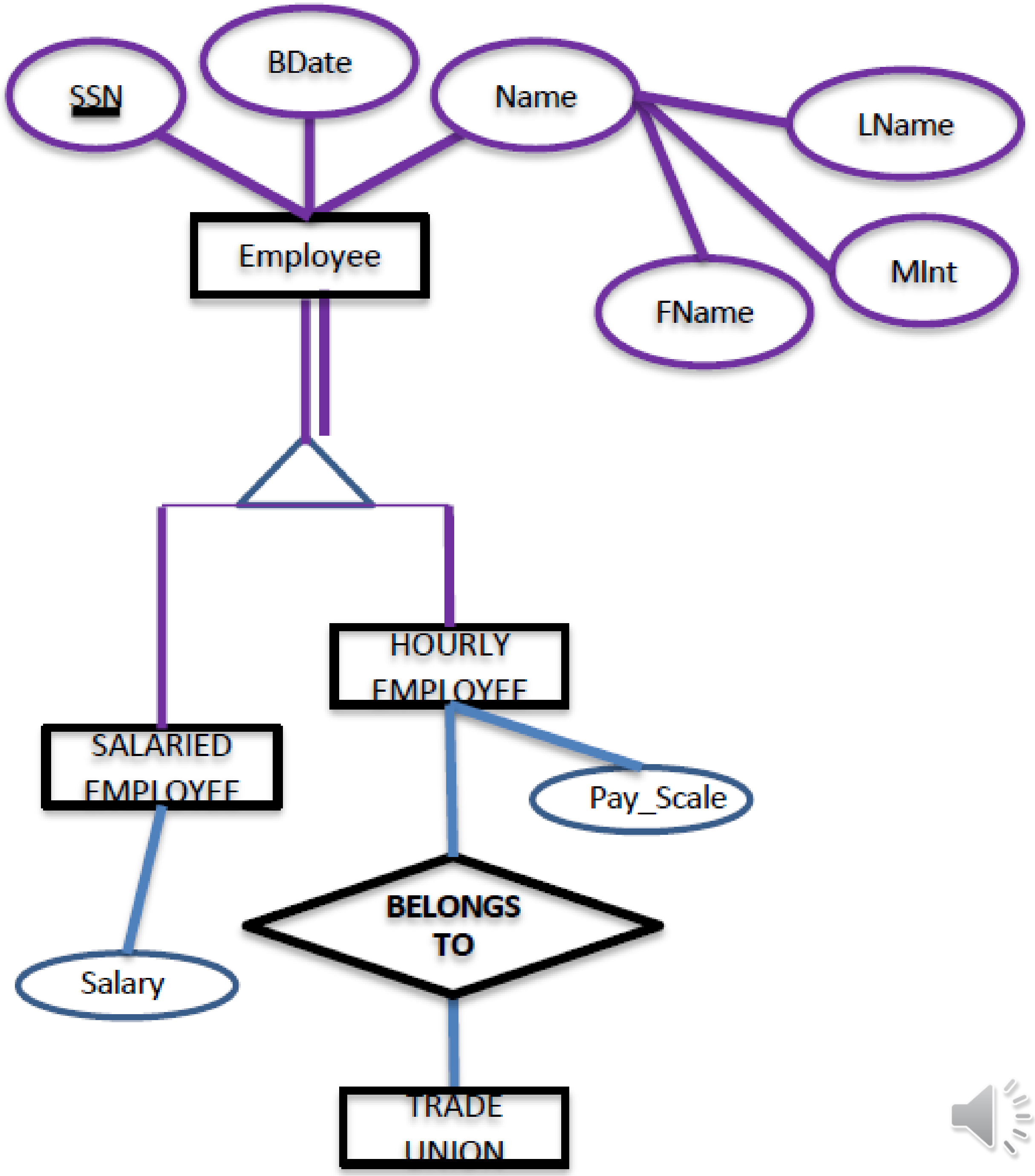
Constraints on Specialisations (and Generalisations)

- However, not all subclass instances can be determined by such an explicit condition. Instances of subclasses can be defined as the entities are populated into the database. These are called ***user-defined***
- For example, the EMPLOYEES who are MANAGERS are created when the database is being populated



Total or Completeness in subclasses

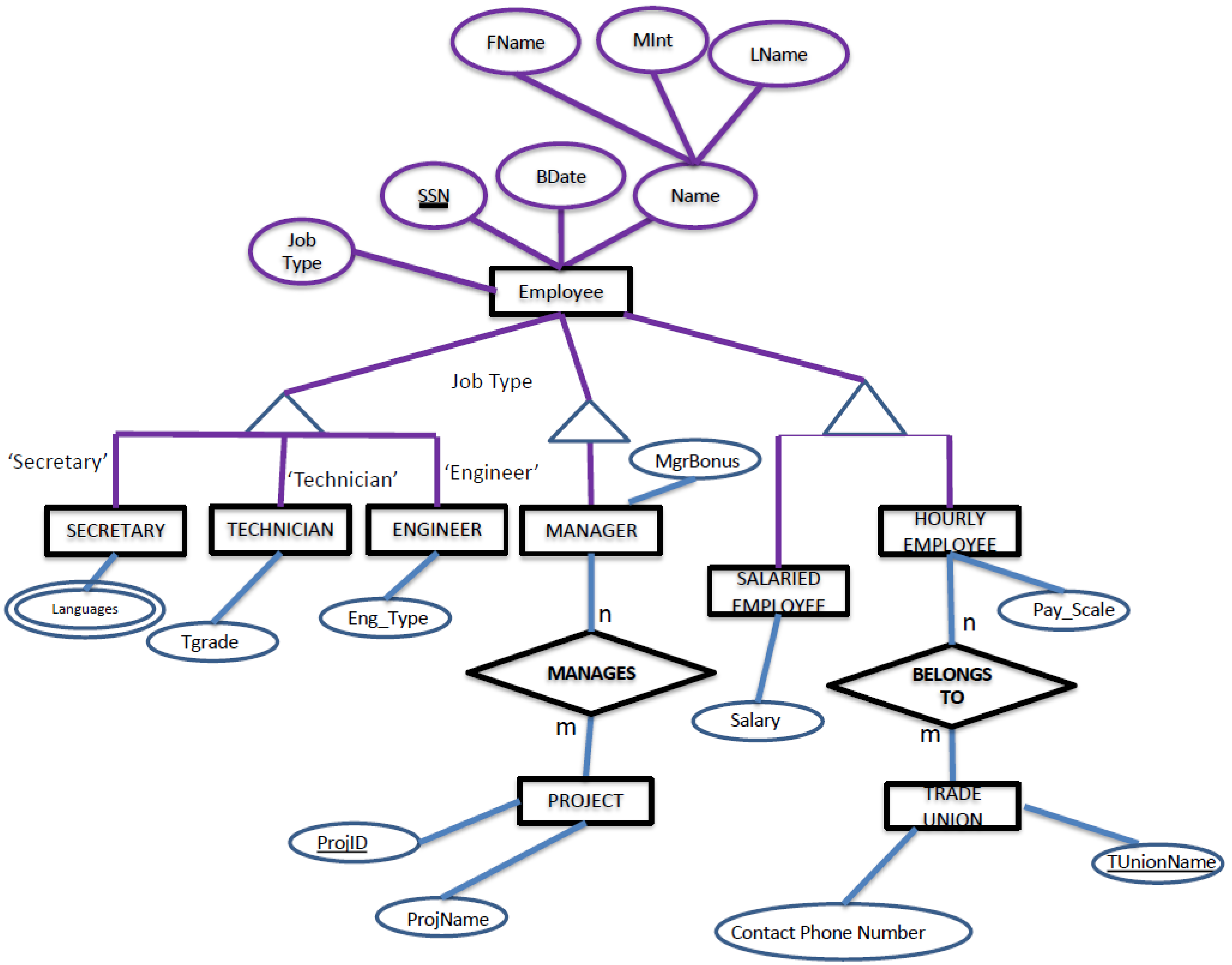
- Sometimes you may wish to ensure an instance of a superclass is one of a set of subclasses
- We can represent this by using the ‘double line’ (which we use for total participation in a relationship).
- In the diagram, EVERY Employee MUST ALSO BE either a SALARIED EMPLOYEE or an HOURLY EMPLOYEE



Mapping EERs to Relations

Mapping EER Model Constructs to Relations

- There are several options for mapping subclasses – I will give just one approach, but others are possible *(Elmasri & Navathe Chpt 8.2)*
- *Mapping Technique :*
  - *– Suppose C is the superclass with attributes (k,a1,…,an) with m subclasses {S1,S2, … , Sm). Also suppose k is the Primary Key of the superclass C.*
  - *– Create a relation L to represent the attributes that are only in the superclass C. The attributes of L are {k, a1,…,an } . The Primary Key of L = k*
  - *– Create a relation Li for each subclass Si 1≤i≤m with the attributes {k} u {attributes of Si } and Primary Key of Li = k*


FName
MInt
LName
SSN
BDate
Name
Job Type
Employee
Job Type
'Secretary'
'Technician'
'Engineer'
MgrBonus
SECRETARY
TECHNICIAN
ENGINEER
MANAGER
HOURLY EMPLOYEE
SALARIED EMPLOYEE
Languages
Tgrade
Eng_Type
n
Pay_Scale
n
MANAGES
BELONGS TO
m
m
Salary
PROJECT
TRADE UNION
ProjID
TUnionName
ProjName
Contact Phone Number

Relations

- EMPLOYEE (<u>SSn</u>, Job Title, Bdate, Fname, Minit, LName)

Note 1: Created a relation for the superclass EMPLOYEE from all of the attributes of that Superclass and took the primary key of the superclass to be the Primary key of the new relation.

Note 2: if any of the attributes of the superclass had been multivalued, then we would have created separate relation(s) for each multivalued attribute, with the primary key of these relations being the primary key of the superclass and the multivalued attribute. This is exactly as we'd map to a relation of any multivalued attribute of an Entity.

Relations

- EMPLOYEE (SSn, Job Title, Bdate, Fname, Minit, LName)
- SECRETARY (SSn, Languages)
- TECHNICIAN (SSn, Tgrade)
- ENGINEER (SSn, EngType)
- MANAGER (SSn, Bonus)
- SALARIED_EMPLOYEE (SSn, PayScale)
- HOURLY_EMPLOYEE (SSn, Salary)

Note: We take each subclass and create a relation for it. Each (subclass) relation contains the Primary key of the superclass and whatever attributes were In that subclass
Note2: if a subclass has any attributes that are multivalued, then each multivalued attribute would be placed in a separate (subclass related) relation with the Primary key of the superclass

Relations

- EMPLOYEE (SSn, Job Title, Bdate, Fname, Minit, LName)
- SECRETARY (SSn, Languages)
- TECHNICIAN (SSn, Tgrade)
- ENGINEER (SSn, EngType)
- MANAGER (SSn, Bonus)
- SALARIED_EMPLOYEE (SSn, PayScale)
- HOURLY_EMPLOYEE (SSn, Salary)
- ManagesProject (SSn, ProjId)
- Project (ProjId, ProjName)

Note: A relationship between a Subclass and another entity is Represented by a relation where the Primary key of the relation is a Composite primary key of the Primary Key of the Superclass and the primary key of the other entity

Note2: The entity to which the subclass has a relationship is just represented as any other entity would be represented, i.e. as a separate relation.

Relations

- EMPLOYEE (SSn, Job Title, Bdate, Fname, Minit, LName)
- SECRETARY (SSn, Languages)
- TECHNICIAN (SSn, Tgrade)
- ENGINEER (SSn, EngType)
- MANAGER (SSn, Bonus)
- SALARIED_EMPLOYEE (SSn, PayScale)
- HOURLY_EMPLOYEE (SSn, Salary)
- ManagesProject (SSn, ProjId)
- Project (ProjId, ProjName)
- BelongsToTUnion (SSn, TUnionName)
- TUnion(TUnionName, ContactPhoneNumber)

Note: Same as before, a relationship between a subclass and another entity is represented by a relation with Primary being a Composite key made up of the Primary Key of the Superclass and the primary key of the other entity

Today’s Lecture

- Weak Entities
- Enhanced Entity Relationships (EERs)
- Constraints and Specialisations
- Mapping EERs to relations