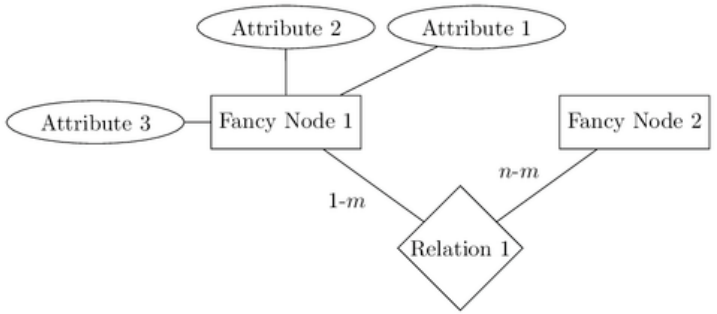
UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Information Storage and Management I

Dr. Alejandro Arbelaez



Entity Relationship Models

Entities

- Entity: an object that is involved in the enterprise
  - Ex: John, CSE305
- Entity Type: set of similar objects
  - Ex: students, courses
- Attribute: describes one aspect of an entity type
  - Ex: name, maximum enrollment

Entities are the people, places, things, or events that are of interest for a system that we are planning to build.

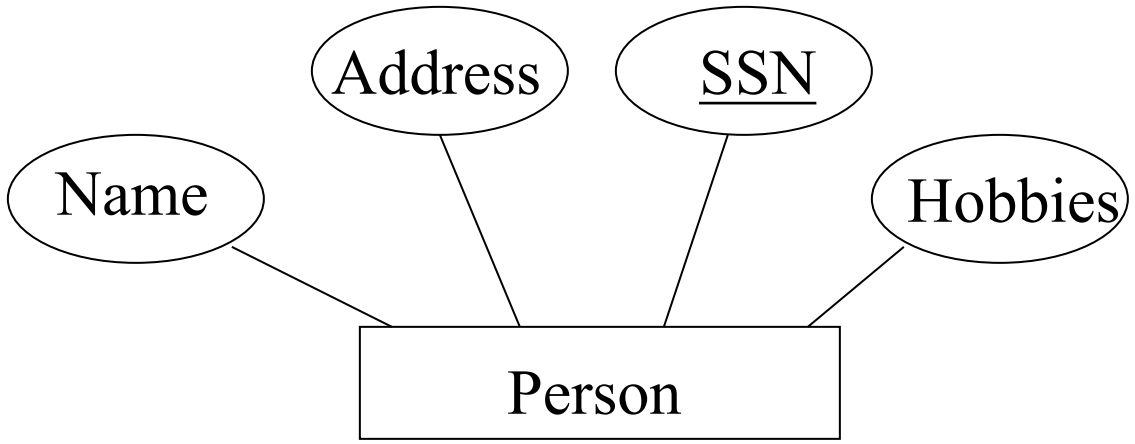
Attributes

- Attributes are the characteristics that describe entities
- A Student entity may be described by attributes including:
  - student number
  - name
  - Address
  - date of birth
  - degree major

Person Entity (Example)

We illustrate attributes using ovals.

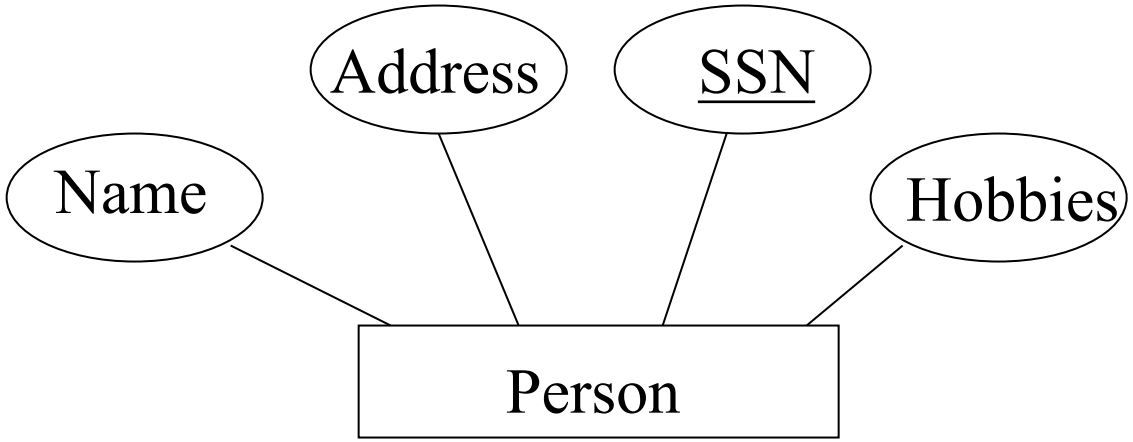
- Graphical Representation in ER diagram:

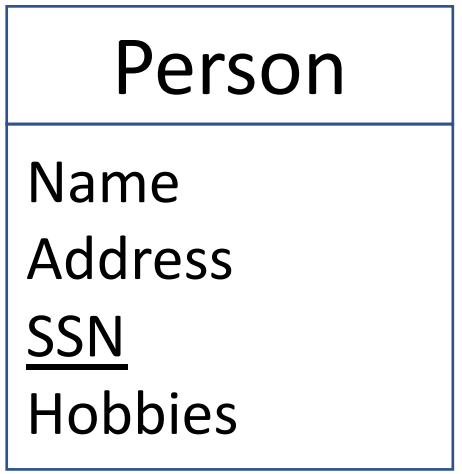


We use a rectangular symbol to represent an entity set in the Entity Relationship Diagram.

Person Entity (Example)

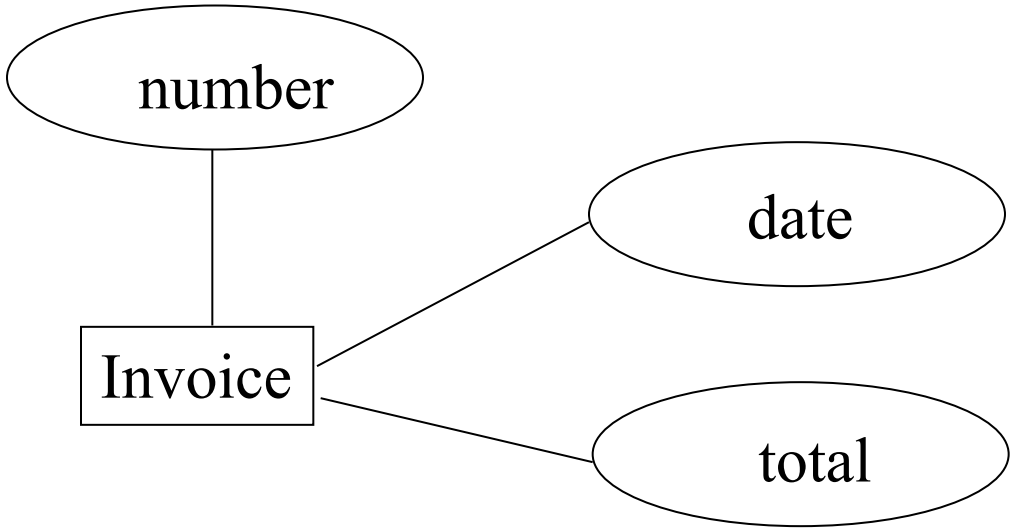
- Graphical Representation in ER diagram:





Alternative Notation

Invoice Entity (Example)



An Invoice entity may be described by attributes including:

- invoice number
- invoice date
- invoice total

Representation in Relational Model

- Entity type corresponds to a relation
- Relation’s attributes = entity type’s attributes

Attributes are the characteristics that describe entities

Representation in Relational Model

- Entity type corresponds to a relation
- Relation's attributes = entity type's attributes
  - Problem: entity type can have set valued attributes.

Two phone numbers

Attributes are the characteristics that describe entities

Attribute Classification

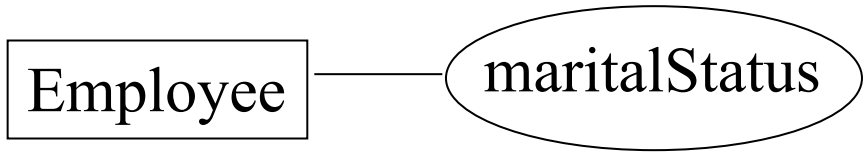
- Atomic Attributes
- Single-valued Attributes
- Multi-valued Attributes
- Key Attributes
- Partial Keys
- Surrogate Key
- Non-key Attributes
- Derived Attributes

Atomic Attributes

- A simple, or atomic, attribute is one that cannot be decomposed into meaningful components
- **Example**: an attribute for product price, prodPrice, cannot be decomposed, because you cannot subdivide prodPrice into a finer set of meaningful attributes. The value of the attribute prodPrice could be $21.03.
- Of course, one could decompose prodPrice into two attributes: the dollar component (21), and the cents component (03)
- but our assumption here is that such a decomposition is not meaningful to the intended application or system that will make use of it.

Single-valued (or atomic) Attributes

An attribute is single-valued if, for any instance of the pertinent entity set, there is only one value at a given time for the attribute.



Multi-valued Attributes

- Consider an attribute to track each employee's university degrees - empDegree.
- Since an employee could have none, one, or several degrees, we say empDegree is multi-valued.
- sample data for three employees.

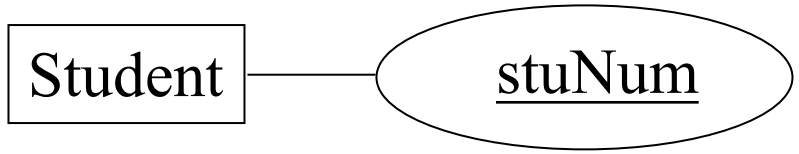
empNum	empPhone	empDegree
123	233-9876	
333	233-1231	BA, BSc, PhD
679	233-1231	BSc, MSc

One of these employees has no degrees, another has 3 degrees, and the last one has 2 degrees.

Key Attributes

- A key is an attribute, or combination of attributes, where the attribute value identifies an individual entity
- Suppose an educational institution assigns students individual student numbers in such a way that each student's number is different from that assigned to any other student
- Student numbers are *unique*, each student has a different number.

In an ER keys are shown underlined



For every entity set, we want to specify which attribute is a key attribute. In this case, Student is shown to have a key named stuNum.

Non-key Attributes

- Most attributes are simply descriptive, and fall into this category.
- Consider attributes for first name, last name, birth date; usually these attributes are non-key attributes.

Non-key Attributes

- Most attributes are simply descriptive, and fall into this category.
- Consider attributes for first name, last name, birth date; usually these attributes are non-key attributes.
- Consider a name attribute:
  - People may join your organization and arrive with a name; we expect many people in a large organization to have the same first name, same last name, and even the same combination of first and last name.
  - Names cannot usually be used as a key.
  - Names that are chosen for entities such as departments in an organisation could be keys because of the way the company would choose department names - they wouldn't give two different departments the same name.

Non-key Attributes

- Could **nationality** be a non-key attribute in a student table?
- Could **advisor's nationality** be a non-key attribute in the student table?

Developing an ER Diagram

- The process of database design is an iterative rather than a linear or sequential process.
- It usually begins with a general narrative of the organization's operations and procedures.
- The basic ER model is **graphically** depicted and presented for review.
- The process is repeated until the end users and designers agree that the ER diagram is a fair representation of the organization's activities and functions.

ER Modeling

- We create a physical model in some (relational) database management system (DBMS)
- We translate the ER Model into a relational database
- Suppose we know of four student entities and two courses entities
  - Students: John, Amelia, Lee, and April
  - Courses: Int. to Art and Int. to History

Relational Database

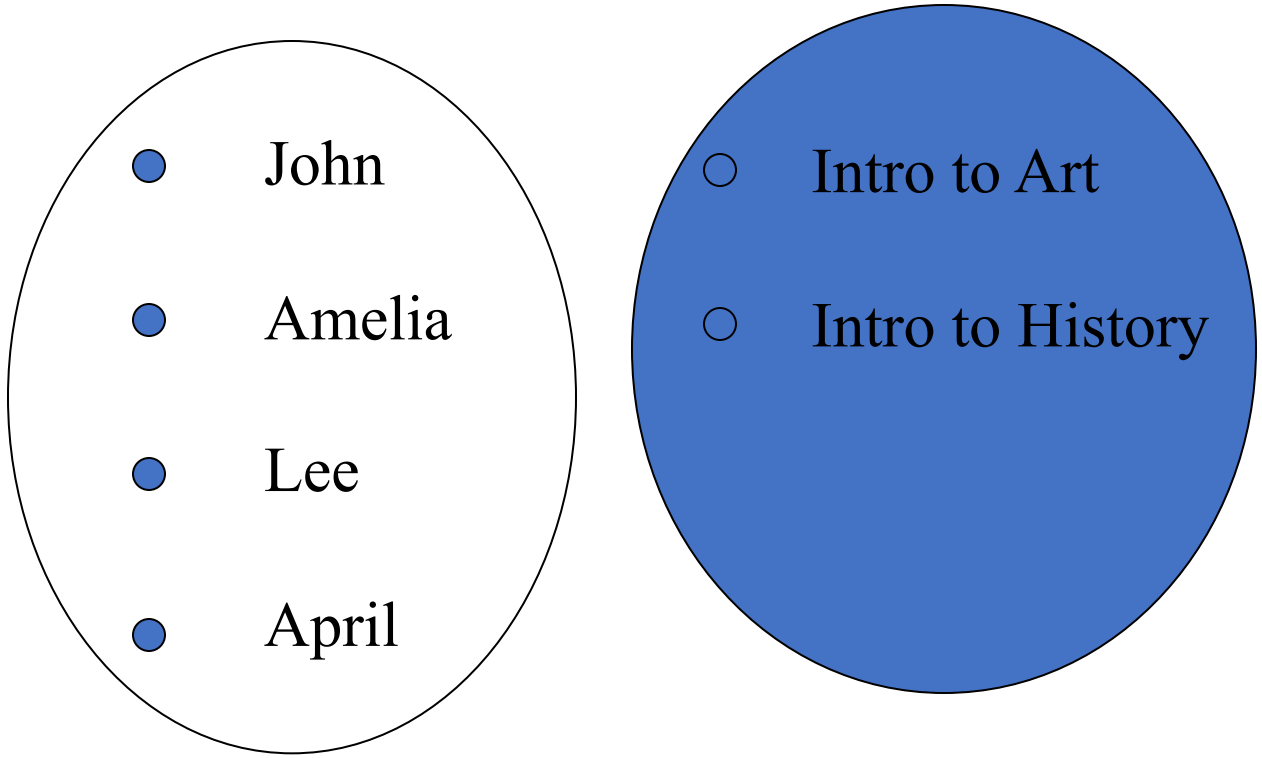
Students

Name	Id#	Phone
John	184	283-4984
Amelia	337	838-3737
Lee	876	933-2211
April	901	644-3838

Course

Name	Course#	Dept
Intro to Art	661	Art
Intro to History	765	History

We can represent these two sets of entities using set diagrams.



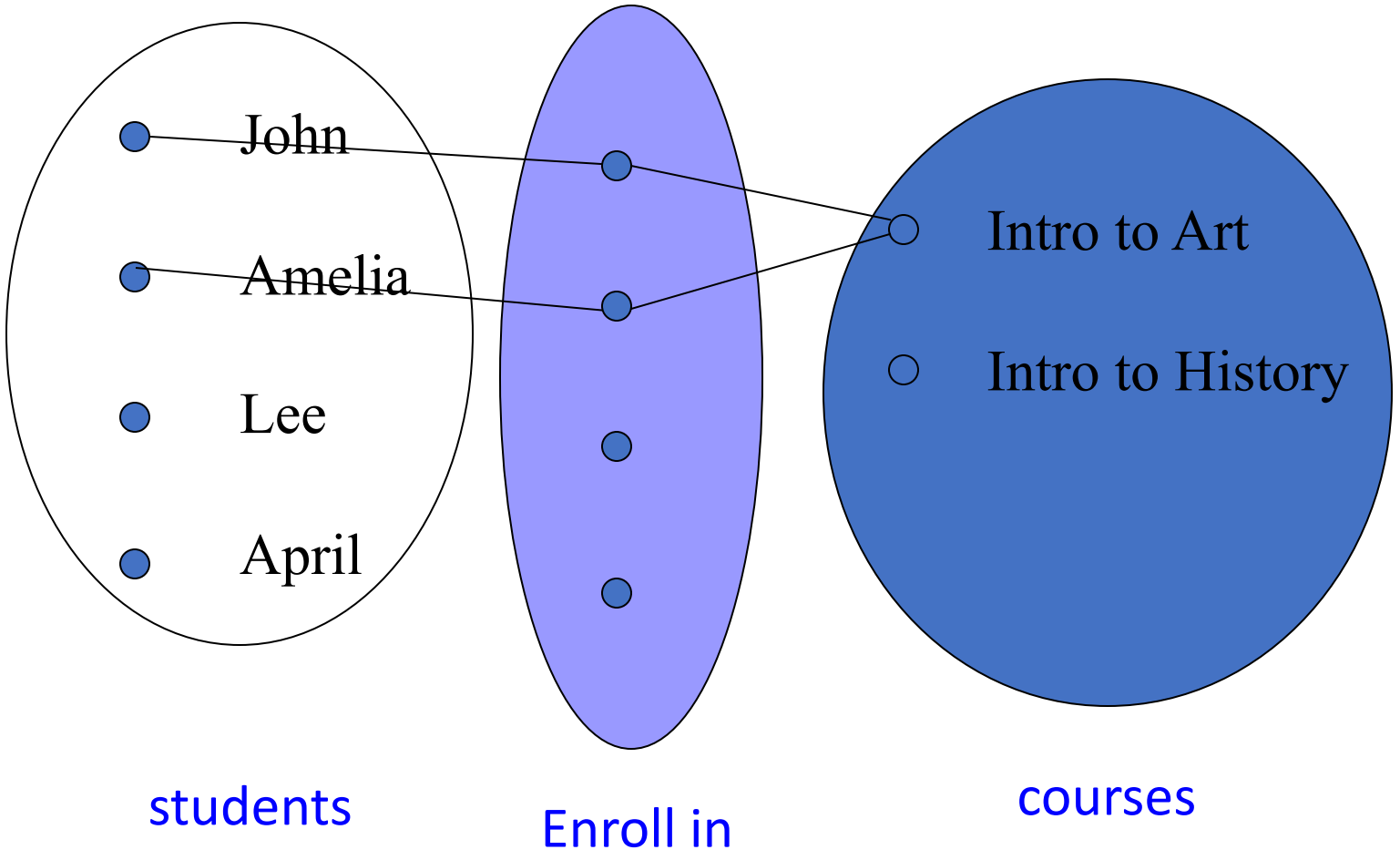
set of students

set of courses

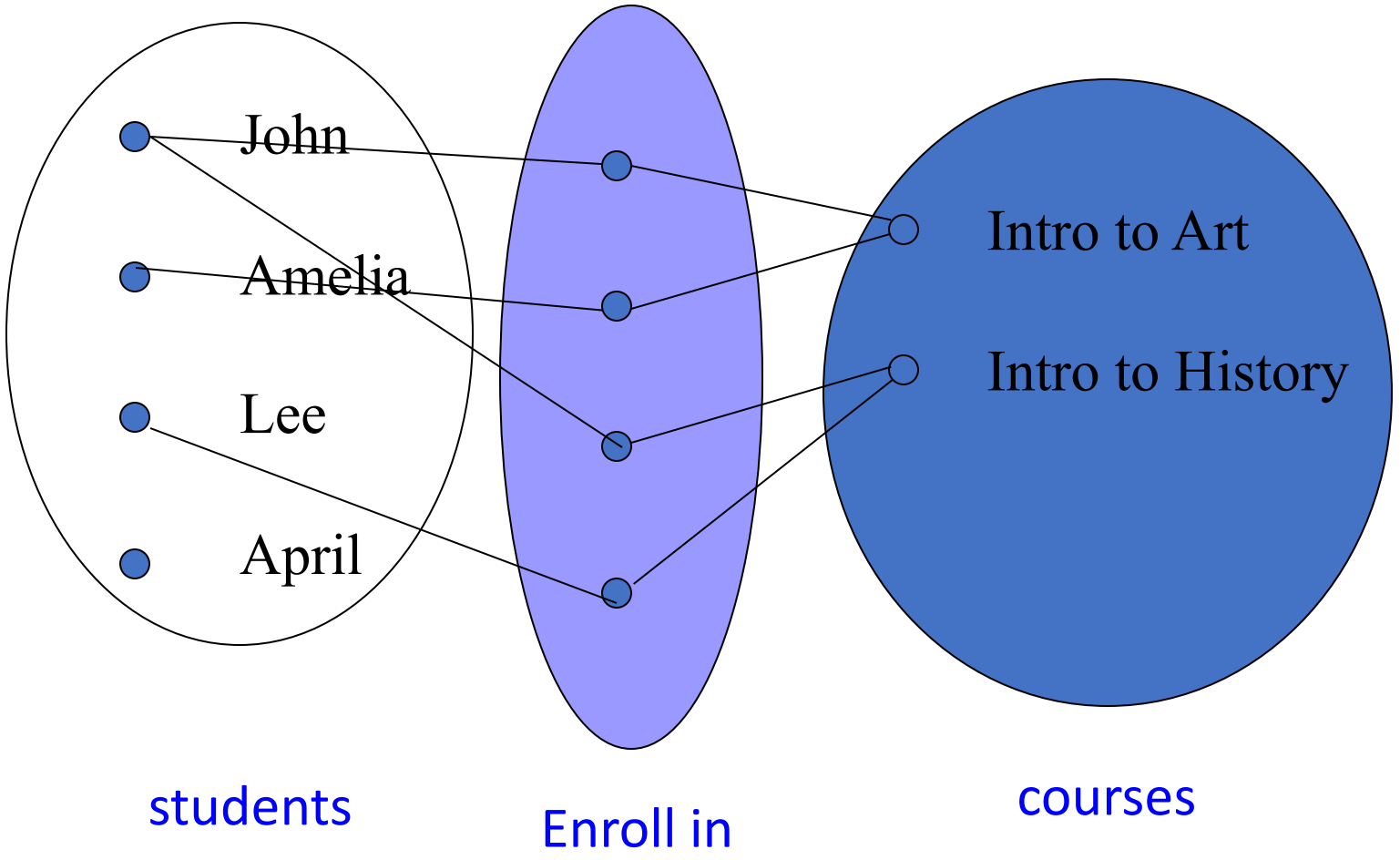
Relationships describe how entities relate to one another

Example

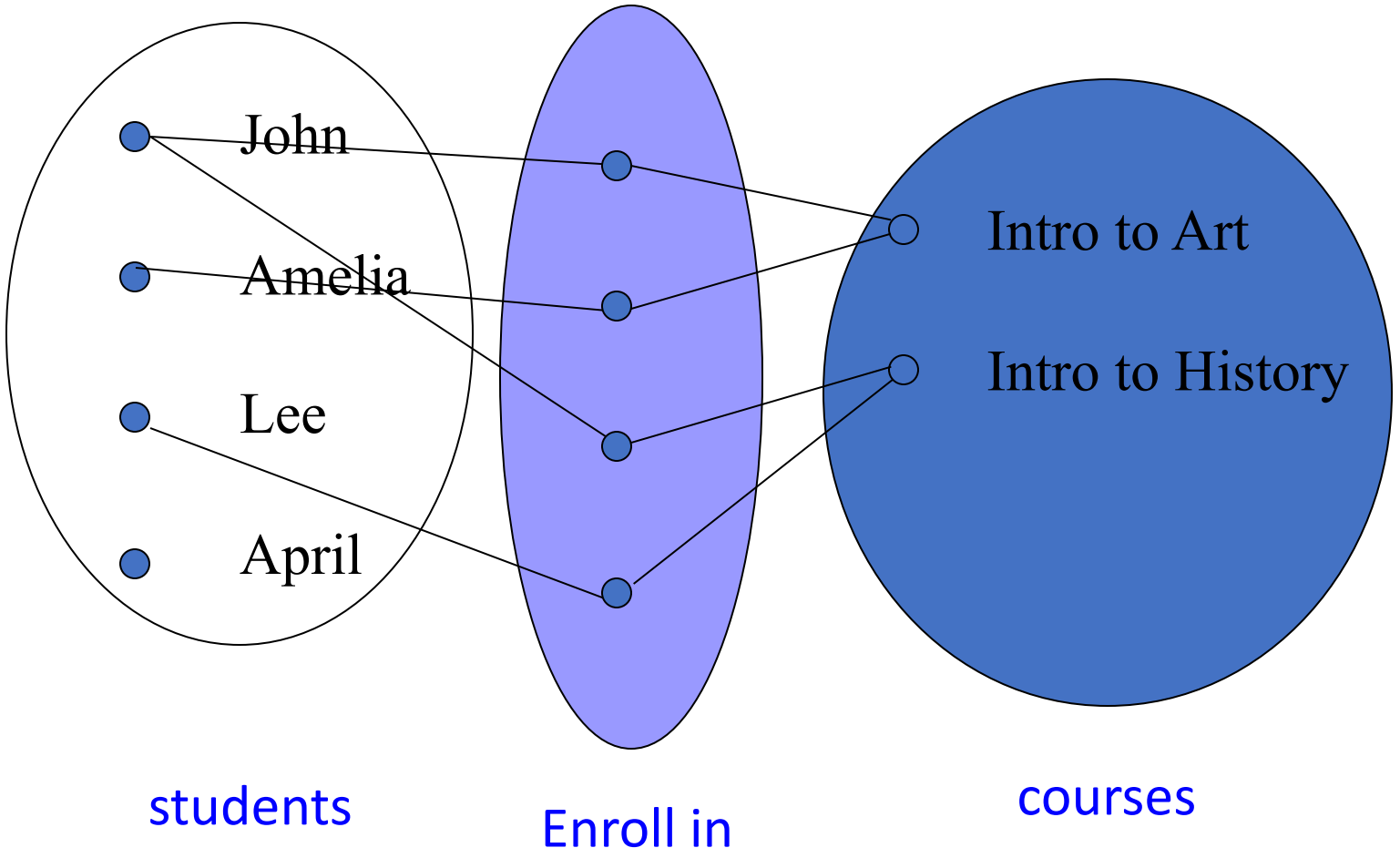
- Suppose we have the two courses and four students listed
- previously. Suppose also that
  - John and Amelia are enrolled in Intro to Art
  - John and Lee are enrolled in Intro to History
  - April is not enrolled in any course.
- Next, we depict a relationship set and show the four
- instances of an *enroll in* relationship.



John and Amelia are enrolled in Intro to Art



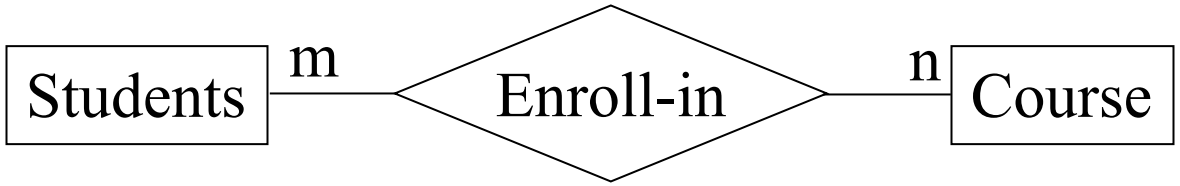
John and Amelia are enrolled in Intro to Art
John and Lee are enrolled in Intro to History



Note there are four instances of *enroll in*. Each Instance involves exactly two entities: a student entity and a course entity. Because each instance of *enroll in* involves two entities, it is referred to as a binary relationship. Binary relationships will be the focus of most of our discussions.

An example of an Entity Relationship Model:

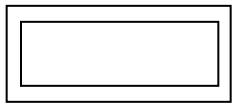
Entities are indicated using rectangular shapes, and relationships between entities are shown with a diamond shape on a line connecting the entities.



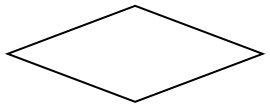
- Entity sets are shown with a rectangular shape.
- Relationship sets are shown with a diamond shape.
- The "m" indicates that one course may have many students enrolled in it.
- The "n" indicates that one student may have enrolled in many courses.

ER symbols

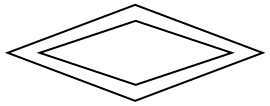
Entity



Weak entity



Relationship

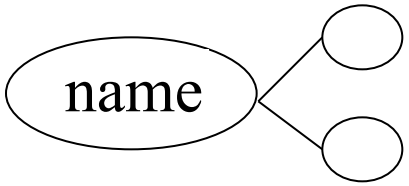


Identifying relationship

Attribute

Key attribute

ER symbols

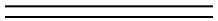


Composite attribute

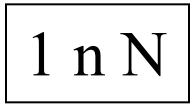
Derived attribute

Mutlti-valued attribute

Partial participation



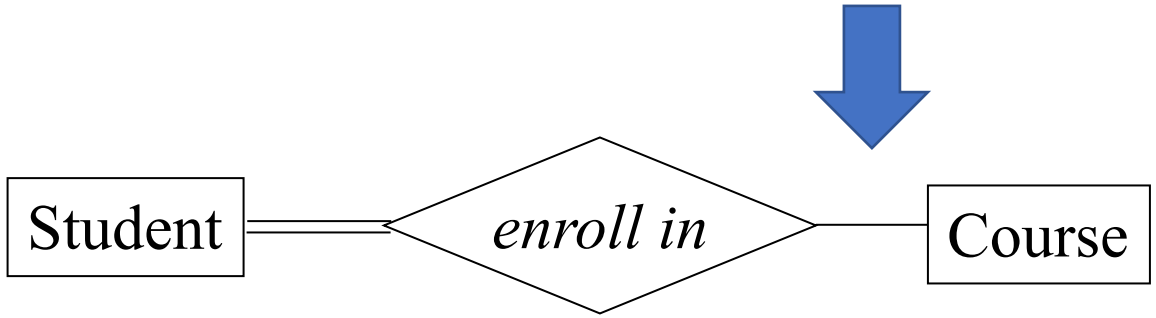
Total participation



Cardinality

Relationships (Partial Participation)

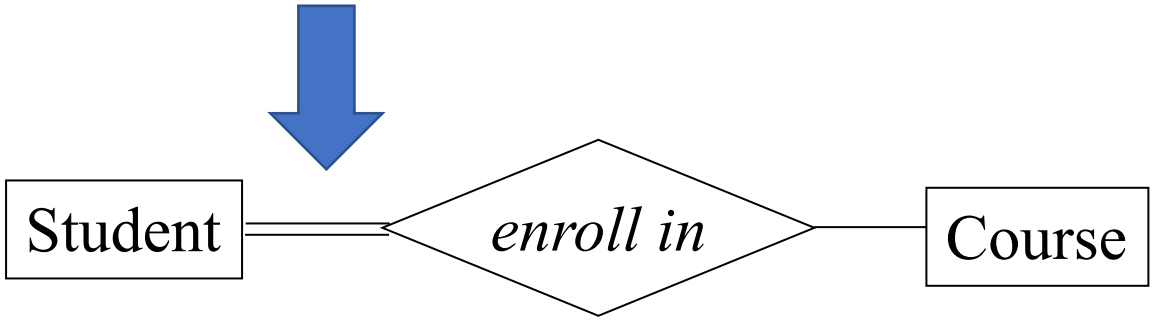
- **The single relationship line** means that ‘enroll in’ is optional to a course
- A course can exist and have zero students enrolled



Partial Participation: some entities may not participate in any relationship in the relationship set

Relationships

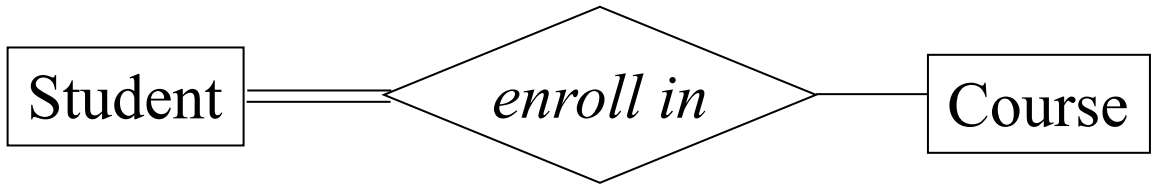
- **The double line** means that ‘enroll in’ is mandatory for student
- Each Student must be enrolled in at least one course.



Total Participation: Every entity in the entity set participates in at least one relationship in the relationship set

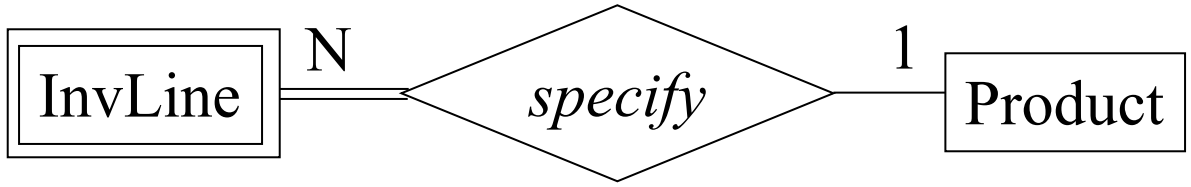
Relationships

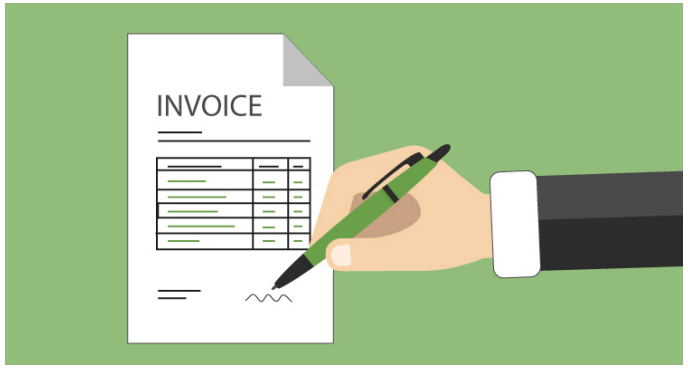
- **The single relationship line** means that ‘enroll in’ is optional to a course
- A course can exist and have zero students enrolled
- **The double line** means that ‘enroll in’ is mandatory for student
- Each Student must be enrolled in at least one course.



Relationships

Cardinality is a constraint on a relationship specifying the number of entity instances that a specific entity may be related to via the relationship.

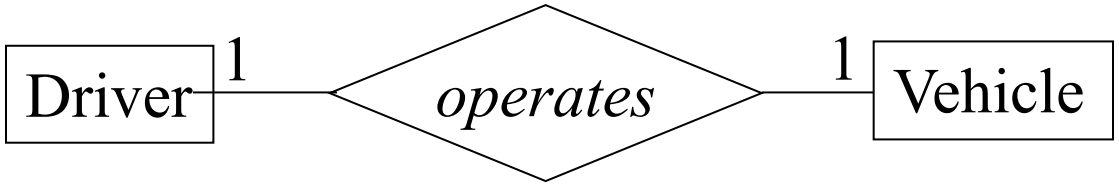




An Invoice Line will specify exactly one Product. A Product may appear on any number, zero or more, Invoice Lines.

Relationships

- **One-to-One Relationships** have “1” specified for both cardinalities, and do not arise very often.



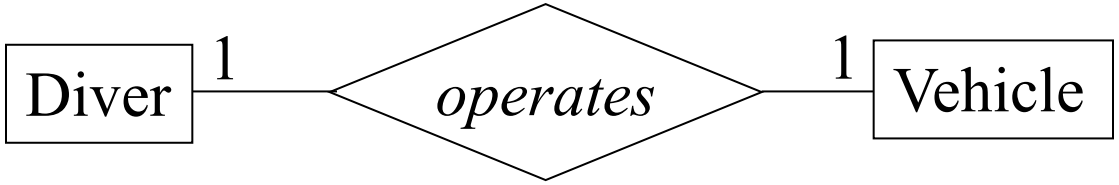
  - A driver is associated with at most one vehicle via the relationship operates
  - An vehicle is associated with at most one driver via the relationship advisor

Arrow notation

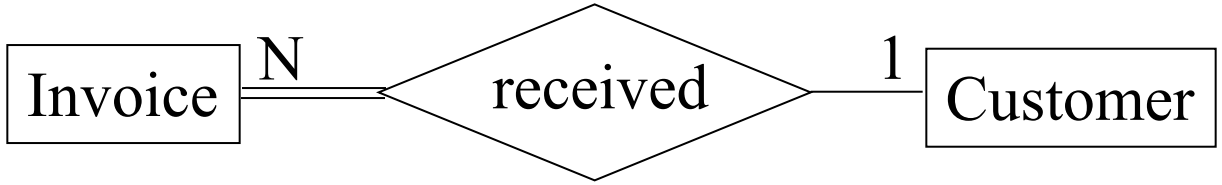
- We express constraints by drawing either a direct line (→), signifying “one”, or an undirected line (—) signifying “many”, between the relationship set and the entity set
- One-to-one relations between
  - A student is associated with at most one instructor via the relationship advisor
  - An instructor is associated with at most one student via the relationship advisor

Relationships

- **One-to-One Relationships** have “1” specified for both cardinalities, and do not arise very often.



- **One-to-Many Relationships**



Many-to-Many Relationships

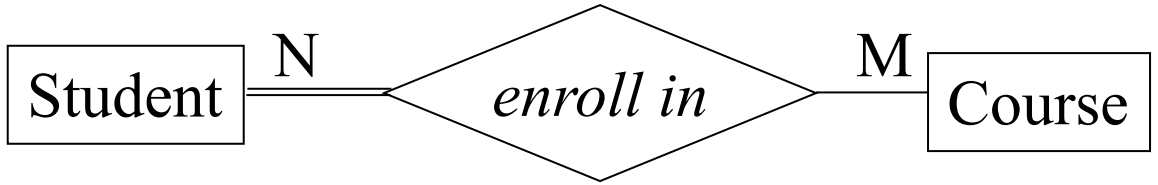
- Many-to-many relationships have “N” and “M” indicating cardinalities, and are also very common.

- However, should you examine a data model in some business, there is a good chance you will not see any many-to-many relationships on the diagram. In those cases, the data modeler has *resolved* the many-to-many relationships into two ***one-to-many*** relationships.

Many-to-Many Relationships

- Many-to-many relationships have “N” and “M” indicating cardinalities, and are also very common.

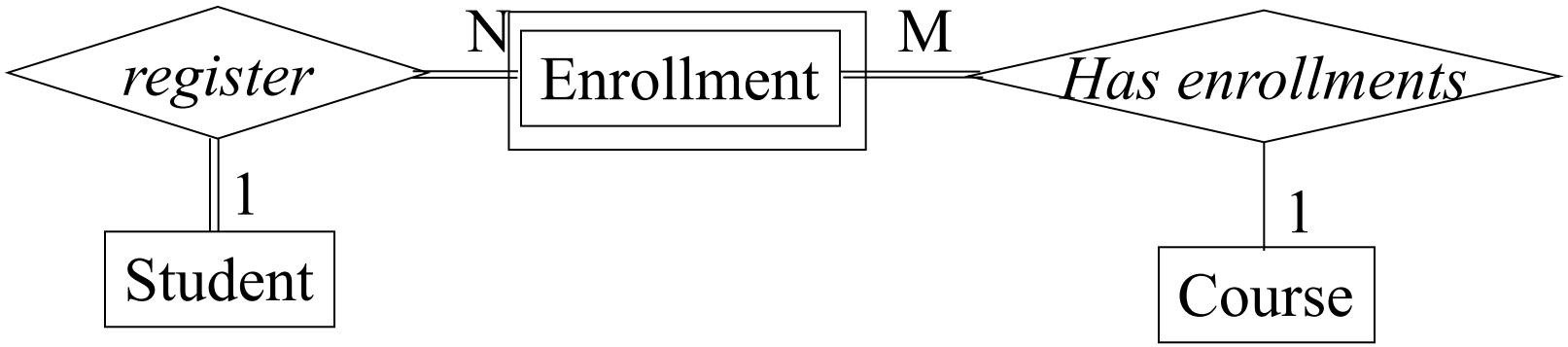
- However, should you examine a data model in some business, there is a good chance you will not see any many-to-many relationships on the diagram. In those cases, the data modeler has *resolved* the many-to-many relationships into two one-to-many relationships.



This many-to-many relationship implies there may be more than one Student entity for each Course entity, and that there may be more than one Course entity for each Student entity.

Many-to-Many Relationships

- A many-to-many relationship is implemented in a relational database using its own relation.
- Because of this, many modelers will resolve a many-to-many into two one-to-many relationships in their diagrams.
- So, for this situation we could redraw the above many-to-many as two one-to-many relationships, as shown in the figure below.



Recursive Relationships

- If an entity has a relationship with another entity of the same entity set, then we have a **recursive** relationship. Of the relationship types, these are the most difficult to master.

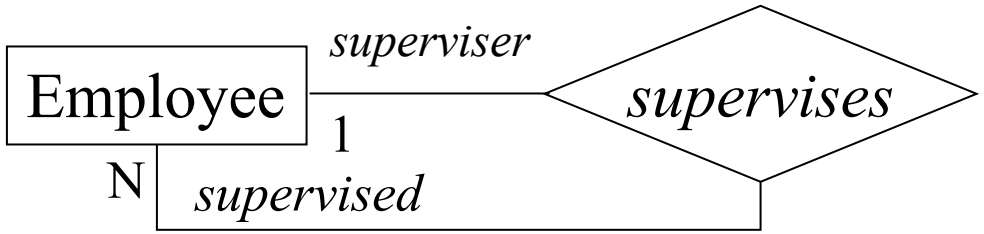
Recursive Relationships

- If an entity has a relationship with another entity of the same entity set, then we have a **recursive** relationship. Of the relationship types, these are the most difficult to master.

- Some situations where recursive relationships can be used:
  - An employee supervises other employees
  - A person marries another person
  - A person is a child of a person
  - A course is a prerequisite for another course
  - A team plays against another team
  - Organizational units report to other organizational units
  - A bill-of-materials system, where a part is composed of other parts.

Recursion

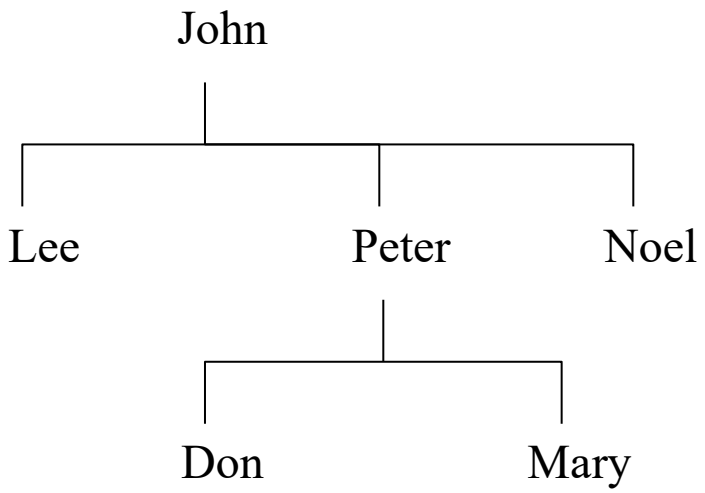
- **Recursion** in a data model is an especially difficult topic. We have a recursive relationship if the same entity set appears more than once in a relationship.



- Any instance of this relationship involves two employees, and so it is a recursive relationship. In this relationship, one employee is designated the ***supervisor*** and the other as the 'supervised'. Employee fills two roles in this relationship.

Hierarchies

- An instance of the *supervises* relationship involves two entities from the same entity set. See the following figure, where five examples of the supervise relationship are depicted in a reporting hierarchy.

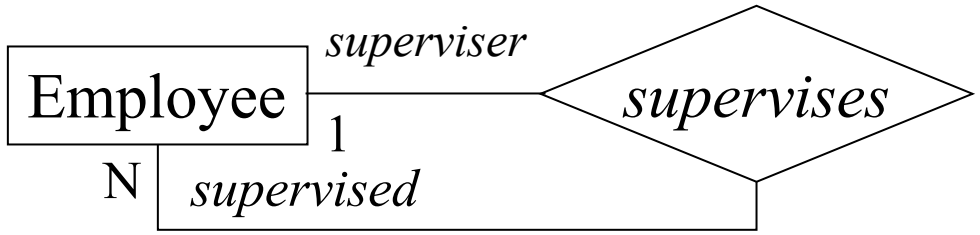


John does not have supervisor
John supervises Lee, Peter, and Noel
Lee and Noel don't supervise anyone
Peter supervises Don and Mary
Don and Mary don't supervise anyone

Hierarchies

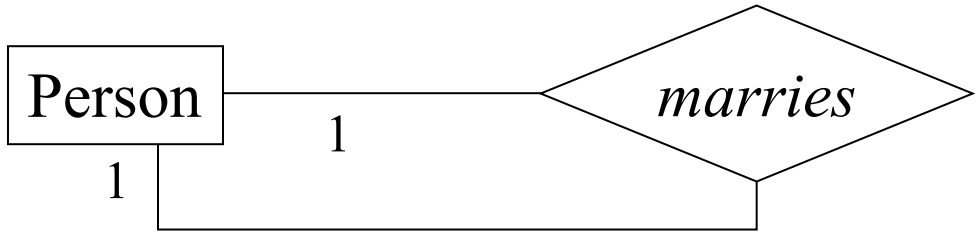
ER for the supervises relationship. Optionality specifications allow for:

- at least one employee does not have a supervisor
- some employees do not supervise others



Hierarchies

- Now consider the second example listed above: “A person marries another person”
- You must be certain of the business rules that are valid for the miniworld you are concerned with
- Depending on that miniworld, you model *marries* as 1:1, 1:m, or m:n. Is the relationship “A person marries another person”, 1:1, 1:m, or m:n?



What is the cardinality of this relationship?

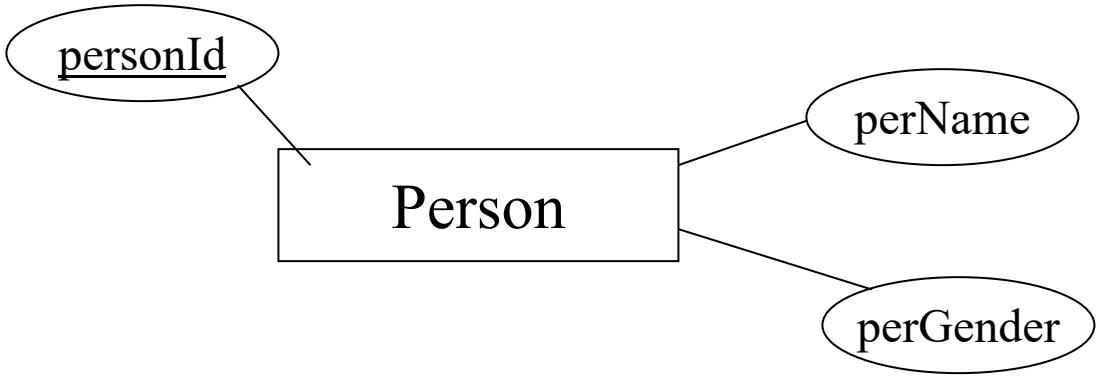
- *1:1 ?*
- *1:m ?*
- *m:n ?*

Hierarchies – 1:1 relation

- If you say 1:1, then only the current marriage for people is of interest in your miniworld.
- A recursive 1:1 relationship allows two entities of the same entity set to be related, but a given entity can only be related to one other entity.

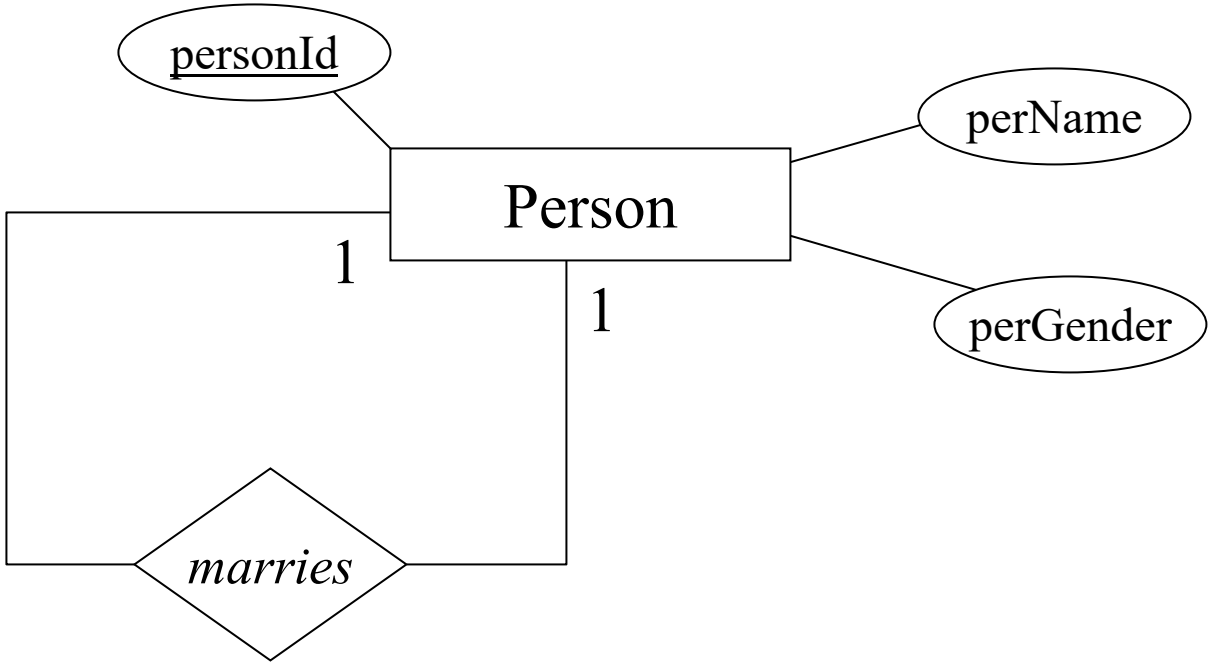
Hierarchies

A person can be represented using the following diagram.

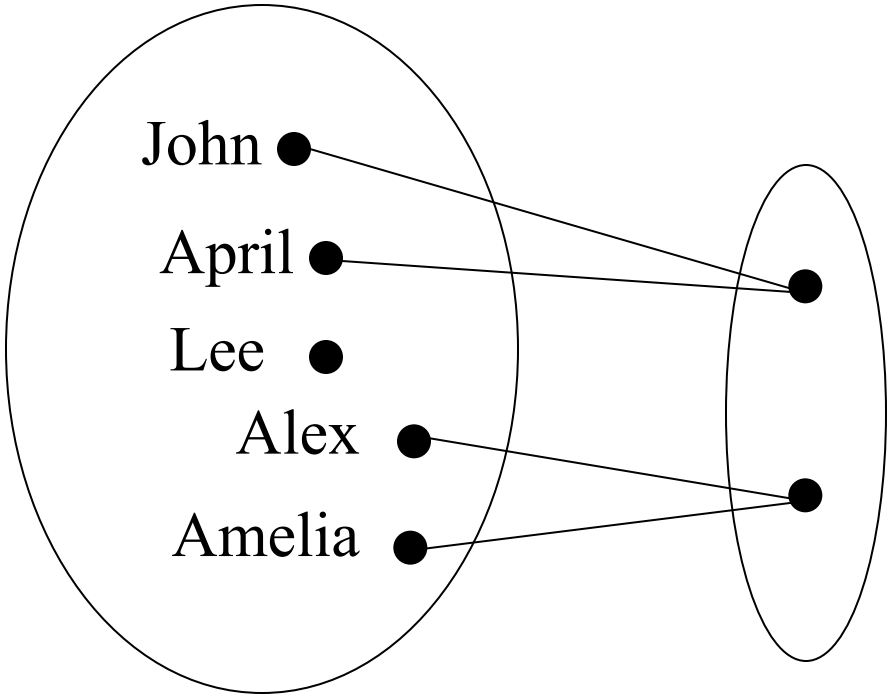


Recursion

Since *marries* is an association between two people (two instances of Person) we understand *marries* is a recursive relationship



Since we are only interested in someone's current marriage partner, we understand that a person is associated with at most one other person via *marries*

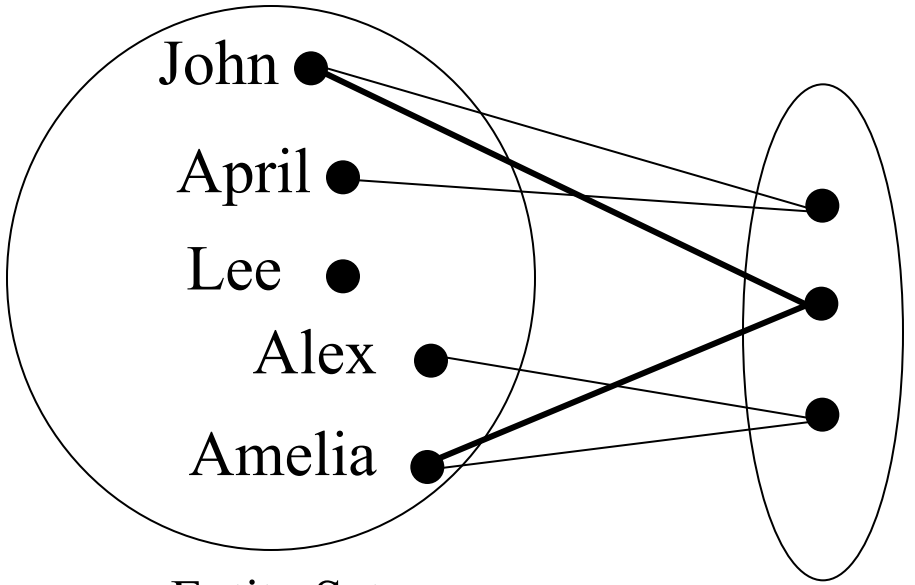


Entity Set
Person

Relationship Set
marries

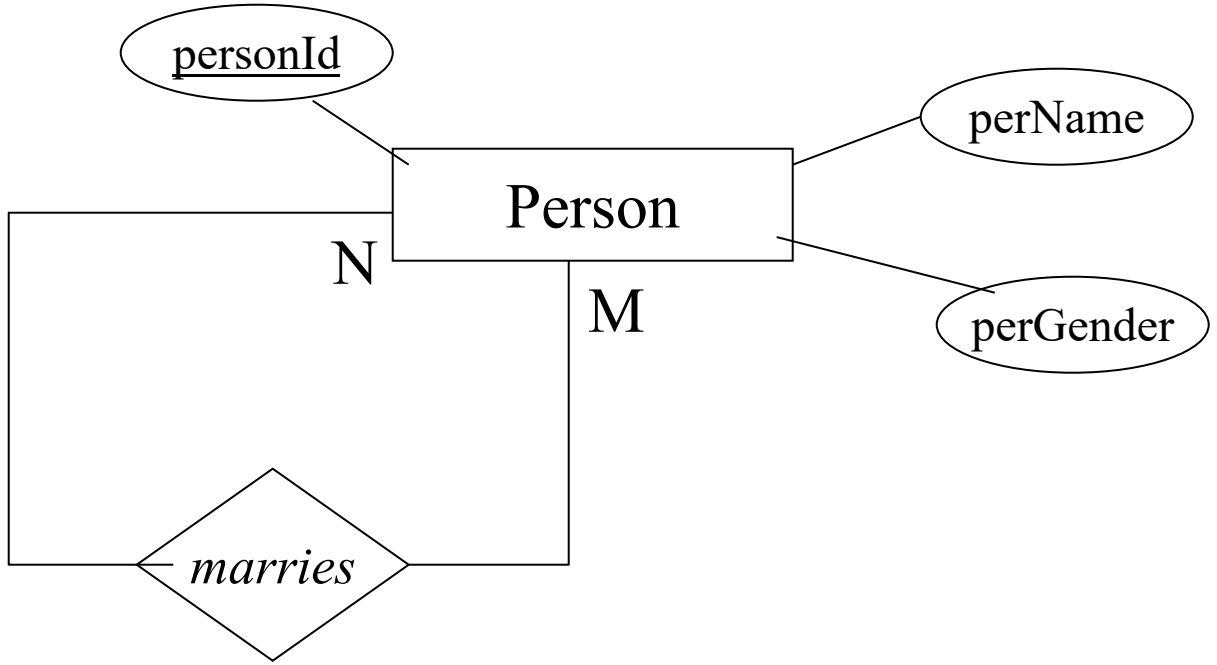
Each person is involved in no more than one 'marries' relationship

- If you said m:n, then you are tracking of all marriages, past and present, for people in your miniworld
- You are allowing for people to remarry and you are keeping track of all their marriages.
- The following figure shows John and Amelia being involved in two *marries* relationships.
- A datum that is recorded for marriages, is the date the ceremony occurred. Where would that be kept in the model? We will address that soon, in the section on attributes of relationships.



Two of these entities are involved in more than one 'marries' relationship.

Since *marries* is an association between two people (two instances of Person) we understand *marries* is a recursive relationship



Weak entities

If an entity is existence-dependent on another entity, then it is a weak entity. Examples where this may occur are:

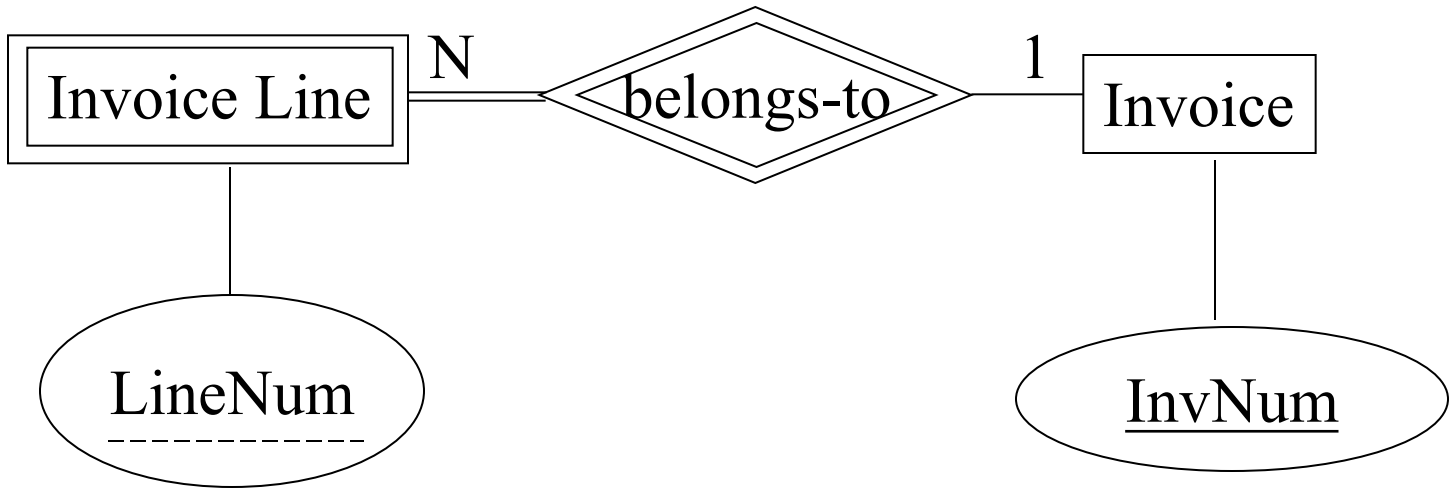
- An invoice line is existence-dependent on an invoice
- A dependent is existence-dependent on an employee
- A section is existence-dependent on a course.

Weak entities

- In these cases: we have a discriminator that differentiates the lines on the same invoice from one another, the dependents of the same employee from one another, and the courses within the same department.

- In these cases, the primary key of the weak entity needs two components: the primary key of the strong entity and the discriminator of the weak entity.

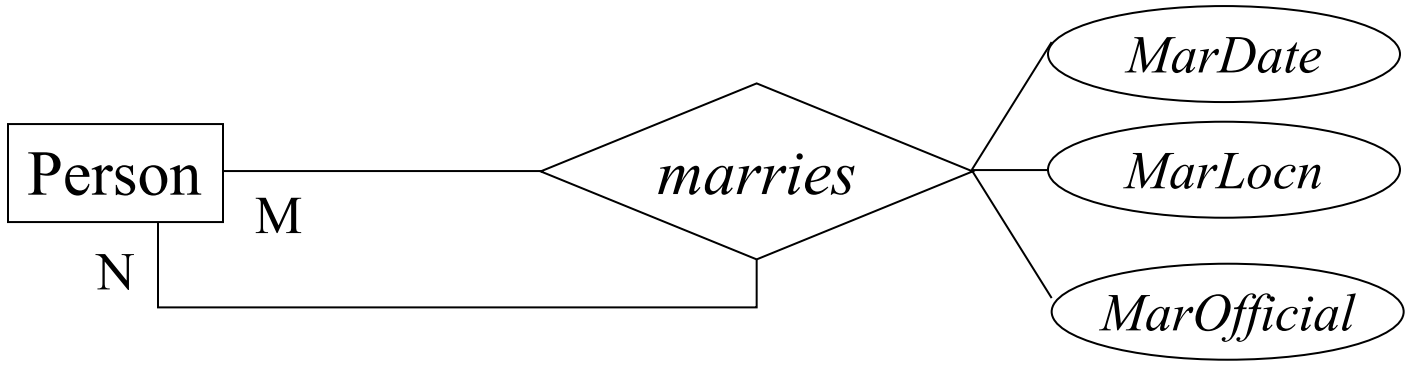
- Because the primary key of the strong entity is required to uniquely identify a weak entity, we say the relationship is an **identifying** relationship, and we use a double lined relationship symbol. Note that one entity set is a weak entity set, and its participation in the relationship is mandatory



An invoice line is partly identified by the Invoice and its invoice number).
An invoice line must participate in the relationship.
The invoice line is a weak entity.
Consider the relationship between Course and Section

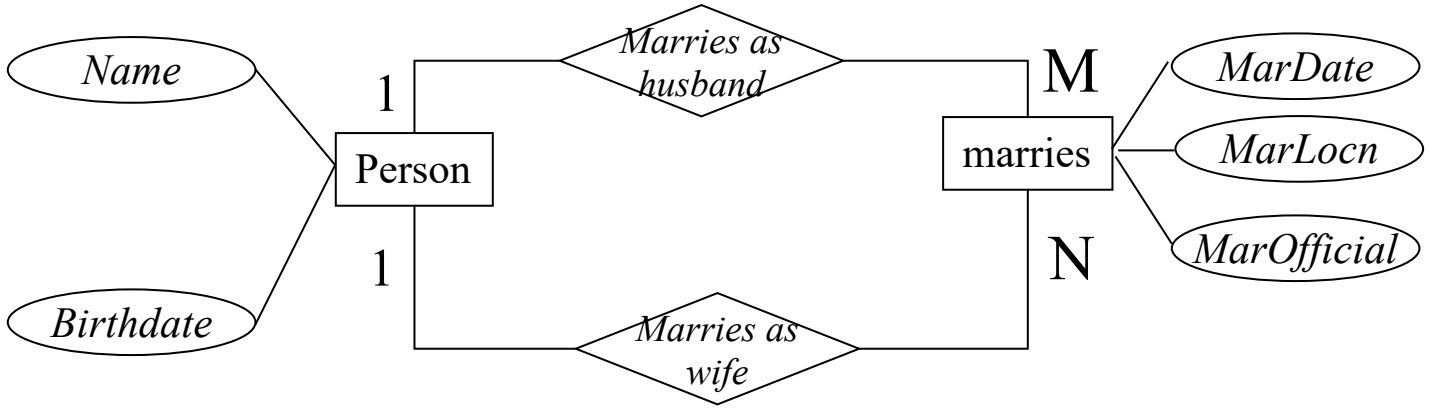
Attributes associated with relationships

- Attributes describe entities and sometimes attributes describe aspects of relationships too.
- Consider the m:n relationship *marries*.
- Useful attributes for *marries* are the date the ceremony occurs, the location of the ceremony, and the name of the person officiating the marriage.



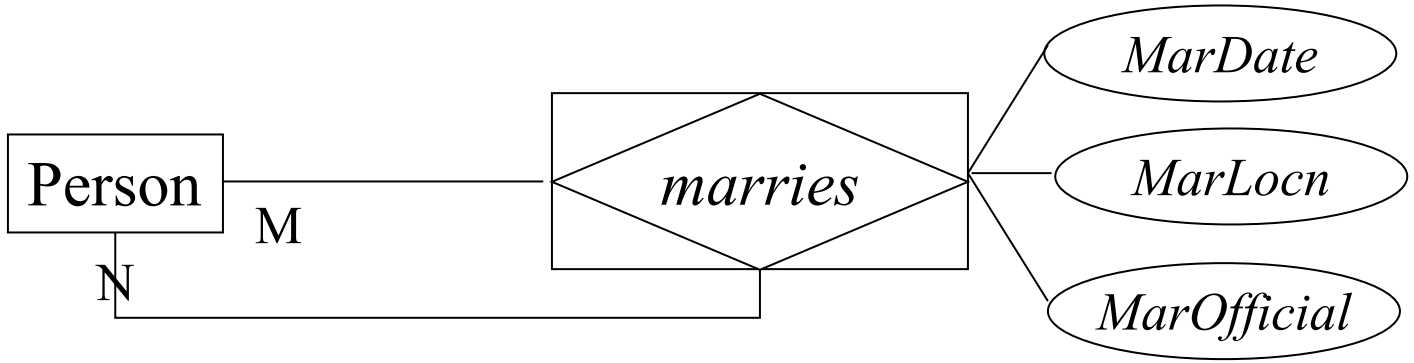
Entity or Relationship

When your model includes a many-to-many relationship, you have a choice to show it as a relationship or to modify the diagram to have another entity set and a pair of relationships. For instance, the marries relationship above could be redrawn with a marriage entity set; onsider the following:



Entity or Relationship

A convention some modelling tools and designers use is shown below. In this case, they are using a composite entity set (the relationship symbol is enclosed in the entity set symbol). Their intention is to show that *marries* is both a relationship and an entity set. We will not be using this notation.



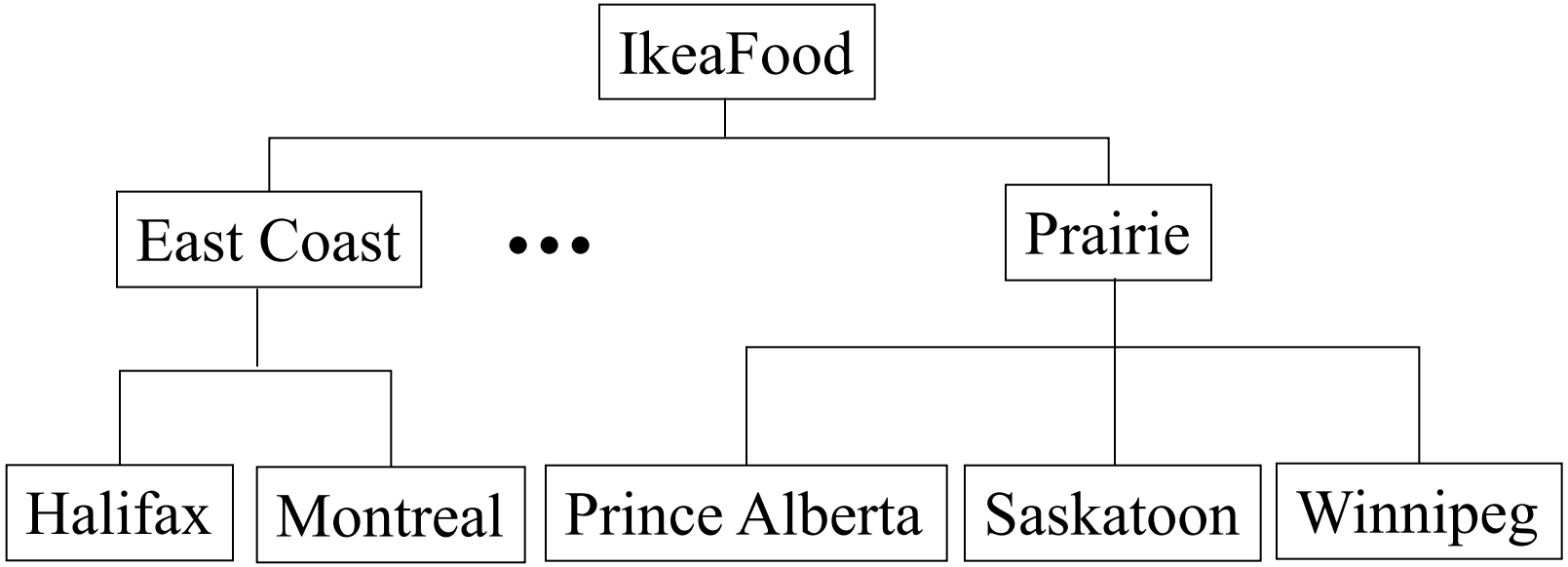
We will avoid using this convention.

Organizational hierarchies

- Organizational hierarchies are very common.
- Most businesses have an inherent hierarchical structure: one component of the business is made from other components.

Example

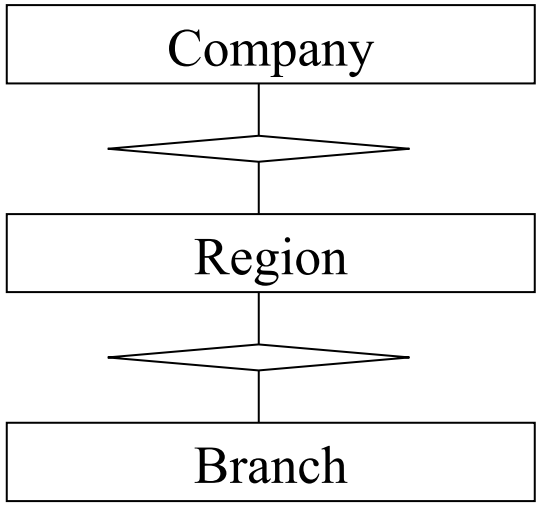
A national company may have regional offices and each region comprises several branch offices. Consider the following that shows part of the hierarchy for a hypothetical video rental company.



Example

The diagram shows two of possibly many regions the company is divided into

- East cost region has two branches Halifax and Montreal
- The Prairie region has three branches
- Knowing that we have a company divided into regions that are divided into branches, could lead us to the following ER model:



A company comprises regions, which comprises branches

Example

The above illustrates the business rules very clearly, but at the same time it can be considered a very rigid structure.

Consider:

- If the structure of the company changes (perhaps we now have Divisions between regions and branches), we cannot accommodate such a change in the model as it stands – we need to change the model to allow for another entity set.
- These kinds of changes are serious in the sense that significant work may be required in the database and/or applications, or to other documentation related to the system in question.
- Another approach is to use a more generic and flexible model.

Example

Suppose we consider the company to comprise organizational units, and so we can consider a new model of our requirements: