Data Modeling

- **Data Modeling**

  Conceptual database design revolves around discovering and analyzing organizational and user data requirements.

  This requires identifying what data are important and what data should be maintained, and this is data modeling.

  Process modeling should precede data modeling as the process model helps to identify data needs of an information system problem.

  When a data model encompasses the entire organization, it is called the corporate data model or the enterprise-wide data model.

  One of the techniques for data modeling is called the E-R diagram and there are other techniques also.

  E-R stands for Entity-Relationship.

  The technique involves identifying entities, attributes, and relationships, and constructing the E-R diagram to represent them.

  The data stores identified in a process model can be used as the starting point for data modeling.

  But a data store does not translate directly into a data model.

Data Modeling (Cont.)

- **Entity Relationship Model**

  Developed by Peter Chen in 1976.

  This is a high level modeling technique, but the outcome of an ER modeling step can be directly translated into a preliminary database design.

  The ER diagram uses graphic symbols to conceptually represent organizational data.

  There are several variations of the ER diagram, and different corporations may use different variations of notations and terminology for the ER diagram

- **Symbols and Terminology of ER Diagrams:**

  Entity – "A thing which can be distinctly identified" - Peter Chen (1976)

  "Any distinguishable object that is to be represented in the database" - C. J. Date (1980)

  The word "entity" means something about which we usually want to store information.

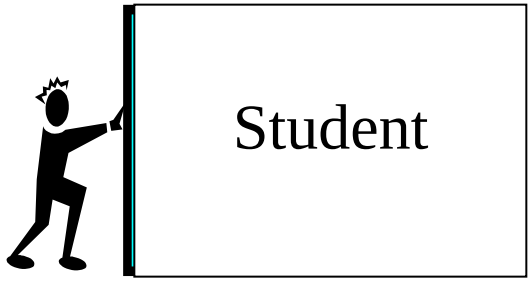
  An entity can be something physical or conceptual, e.g. students, course, company, etc.

  Entities are represented in an ER diagram as rectangles as shown below:

Data Modeling (Cont.)

Entities need not be disjoint; that is, an entity can belong in both the student entity set and also in the employee set.

- **Conventions related to entities:**



– Entity name should be meaningful and accurate, and unique.

Bad name for an entity: resource

Because a resource can be equipment, person, or facility.

– should not be an abbreviation that is not readily known by everyone. e.g. SCI

– should not be one or the other: building OR structure, i.e. should reflect only one thing but not both.

– should not contain unnecessary words and phrases: e.g. the equipment at a location

– an entity must represent a class or set of things (there are some exceptions, which we may see later)

Entity should not be used to represent single instance of a class or set of things

If you have only one instance of something, then there is no need for an entity set.

For example, if you have only one student in a school then you don't need a school database!

Data Modeling (Cont.)

- **Conventions related to entities (cont.)**

  If you are designing an information system for a restaurant about its sales, inventories, etc., then do not create an entity called "restaurant" because there will be only one instance of that restaurant.

  That is, the information needed is about one restaurant, and therefore, there is no need for an information system

  However, if you want to maintain information about different franchises of a restaurant chain, then an entity called "restaurant" may be appropriate as there will be several instances of restaurants at different locations, etc.

  If a clinic wants to maintain information about just one patient then there is no need for an entity called "patient", and there is no need for an information system

  However, if the clinic wants to maintain information about several patients, then an entity called "patient" is appropriate as the clinic wants to maintain data about several patients

  The key step in data modeling is choosing the right entities for the problem situation. Otherwise, the entire data model will be incorrect.

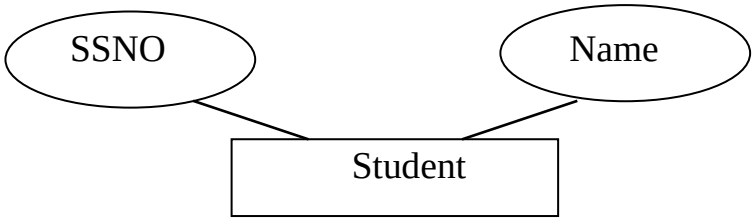
- **Attributes – describe the characteristics or properties of entities related to the problem situation, which we want to store in an information system.**

  Example: Attributes of an Entity called Student can be: StudentID, Name, Address, etc.

Data Modeling (Cont.)

Attributes are represented by circles or ellipses and connected to the entity with a straight line.

Example:



- **Conventions related to Attributes:**

  Names should be clear, concise, meaningful, and self-explanatory.

  Should not be a plural, or a verb.

  Each attribute has a set of permitted values called "domain". For example, the domain of an attribute called Salary can be 1000 – 100000.

- **Relationships**

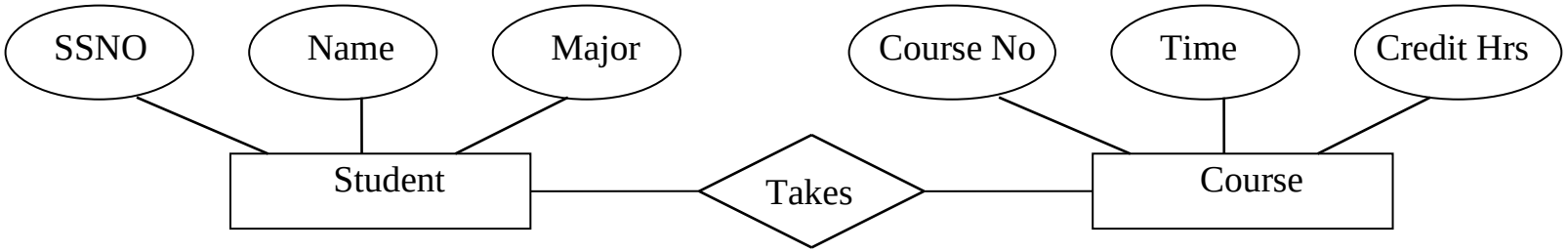
  Relationship is an association among several entities.

  Relationships are represented by "diamonds".

  Relationships are linked with entities by lines.

Data Modeling (Cont.)

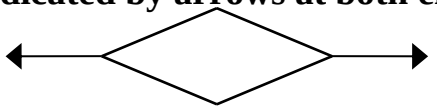
Example:



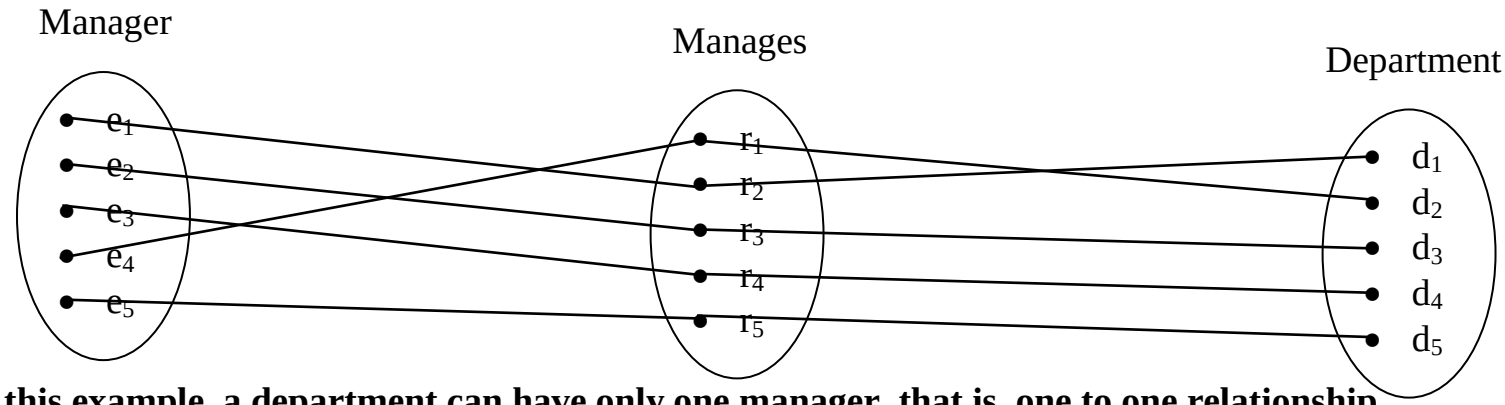
- **Cardinalities**

  Indicate how many entities from one set can be associated with how many entities from another set?

- **One – to – One (indicated by arrows at both ends)**



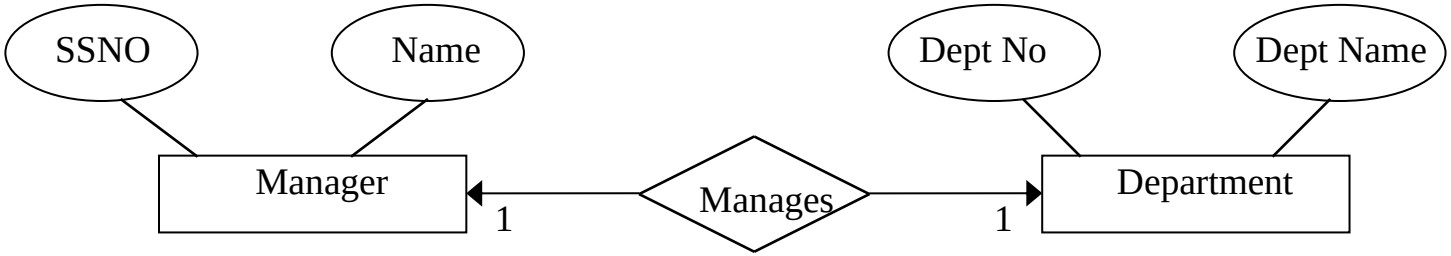
An entity in A is associated with at most one entity in B, and vice versa.



In this example, a department can have only one manager, that is, one to one relationship

Data Modeling (Cont.)

Example:

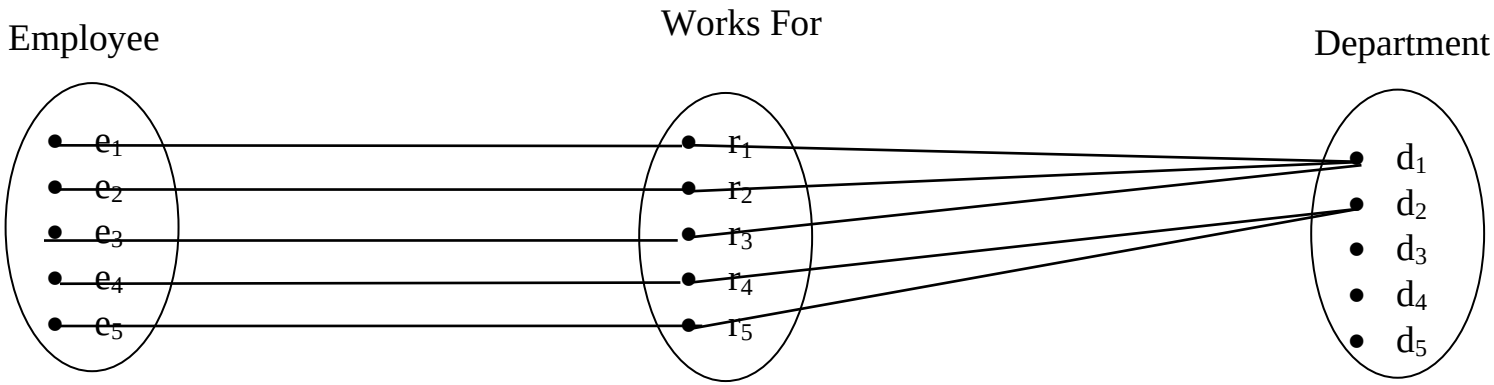


Each manager manages one department.

Each department is managed by only one manager.

- **One-to-Many: an entity in A is associated with any number of entities in B, that is, Many entities in B.**

  An entity in B is associated with at most one entity in A, that is, only One entity in A.



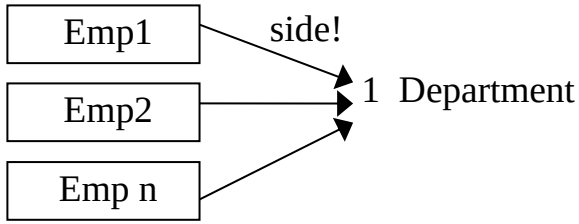
Data Modeling (Cont.)

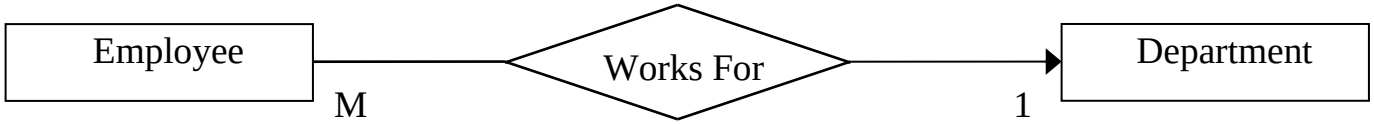
Example:

Each employee works for one department.

Each department has many employees.

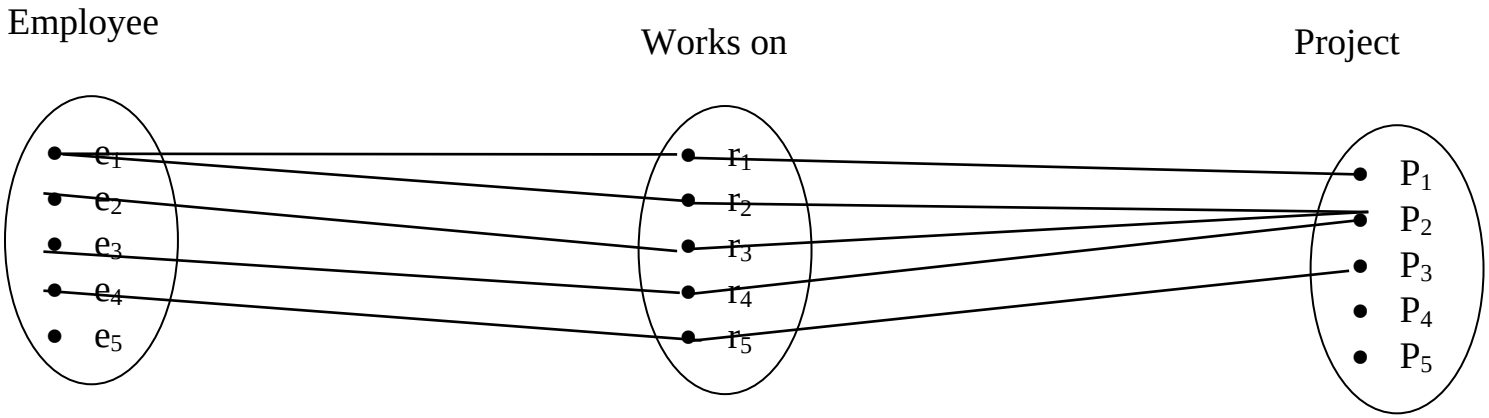
Think of One-to-Many this way to remember the arrow goes at the one side!





Similar to this is Many – to – One

- **Many-to-many: an entity in A is associated with any number of entities in entity set B**



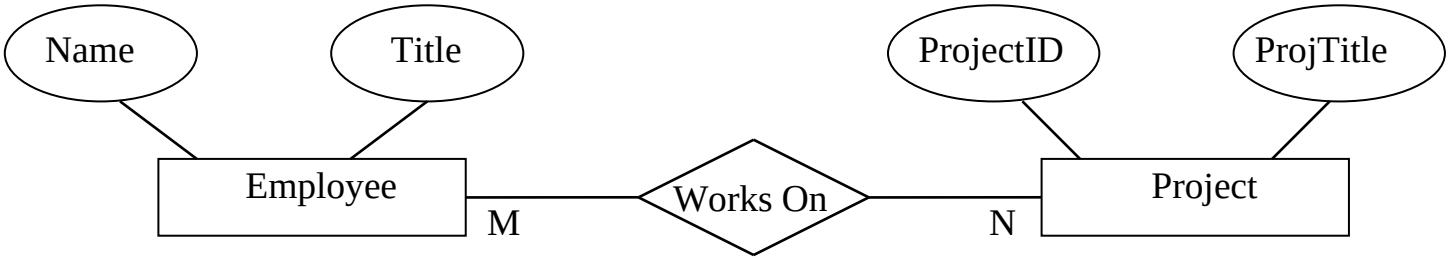
Data Modeling (Cont.)

- **Many-to-Many: (Cont.)**

Each employee works on many projects.

Each project can have many employees working for it.

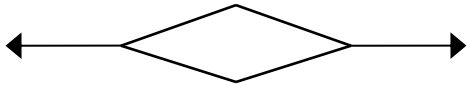
Example:



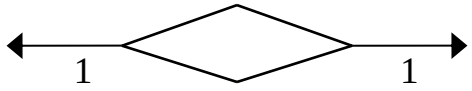
Note it is M:N and not M:M because M and N can be different numbers and need not be the same

- **Cardinality Conventions**

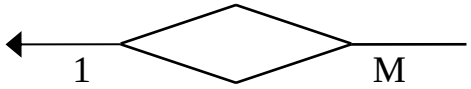
One to One



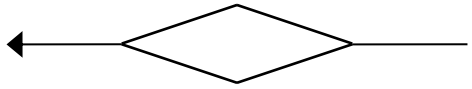
or



One to Many

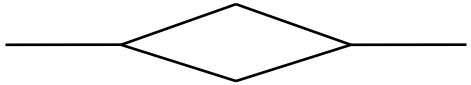


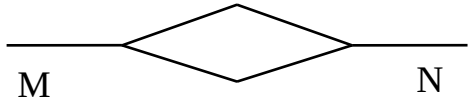
or



Many to Many

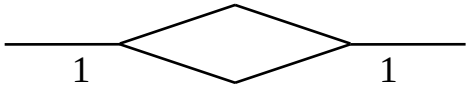
or

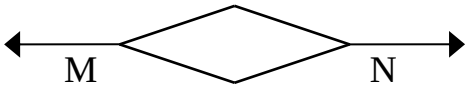


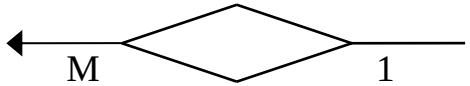


Data Modeling (Cont.)

- **Don't do the following because these diagrams are contradictory in terms of cardinality definitions:**







- **Example:**

  Let us assume we want to design an information system for maintaining student grades. For this purpose, first we want to develop a data model of the problem.

  The data mode will depend on the description of the problem and the assumptions we make about the problem.

  How can we go about designing a data model for this problem?

  Let us ask ourselves the question – what is the thing or object of interest?

  Is it grade? If it is, can it exist independently on its own?

  A grade cannot exist by itself in a database system, i.e. we cannot just have a table of grades.

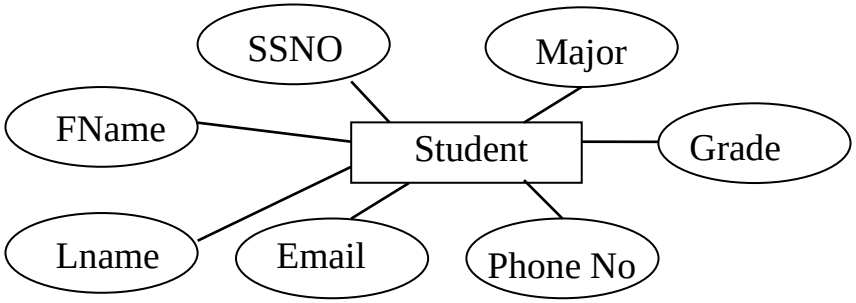
  A grade is associated with something else.

  So, what is the entity?

Data Modeling (Cont.)

A grade is assigned to a student and so it is a property of a student. Therefore, student is the actual object of interest and so the entity.

So we can begin by describing the entity student.



Can we include grade as an attribute of the student?

If we do that what does that imply?

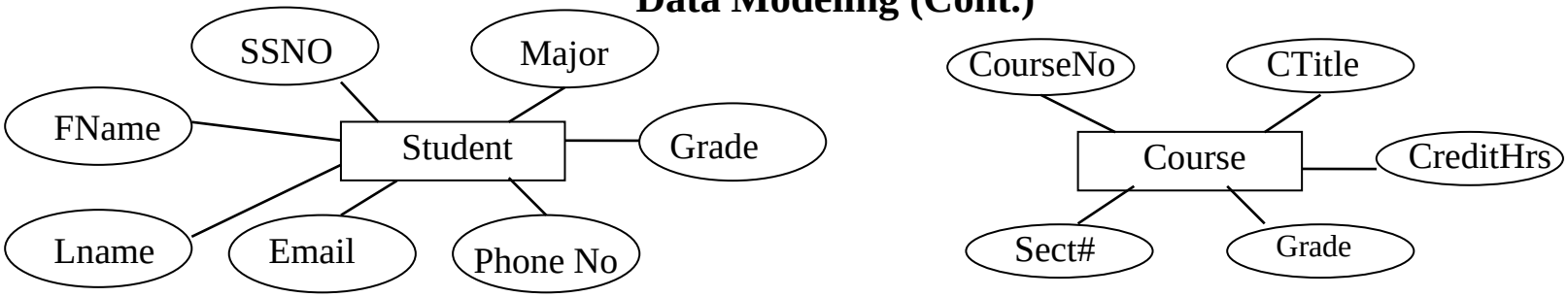
That implies a student can take only one course for which a grade can be recorded as shown below:

SSNO	Fname	Lname	Major	Email	Phone No	Grade
123	Billy	Bob	OMIS	…	…	B

This is obviously wrong since a grade is associated with a particular course for a student.

Therefore, we need to define an entity called a course.

Data Modeling (Cont.)



Where should the attribute "grade" go?

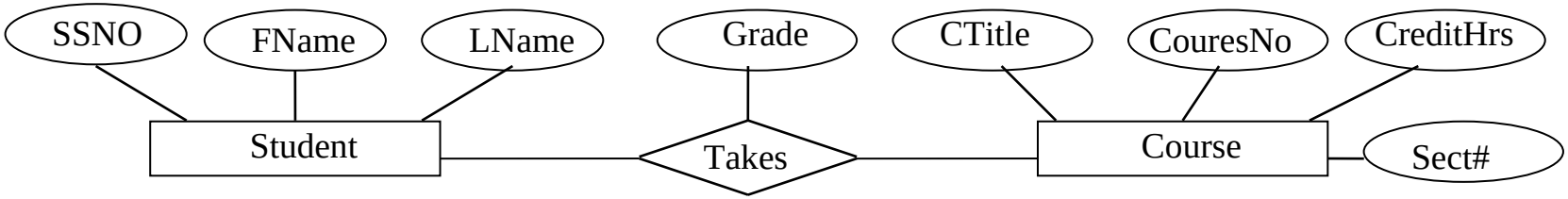
If it is added as an attribute of the entity course, then we will have:

CourseNo	CTitle	CreditHrs	Grade
ISYE 451	Expert Systems in Engineering	3	A
ISYE 482	Engineering Information Systems	3	A

So, this does not really describe a particular student's grade.

Secondly, this is repetitive. Storing redundant information is not a good idea.

So we want to relate student and course and relate them together and assign the attribute to the relation.



Data Modeling (Cont.)

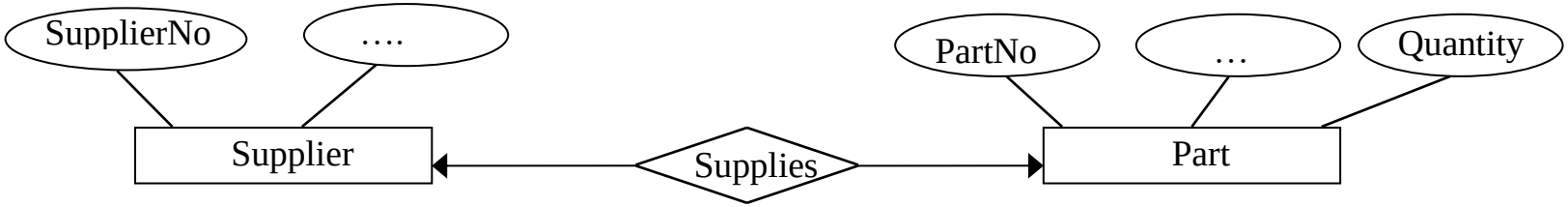
Each student can take many courses.

Each course can be taken by many students.

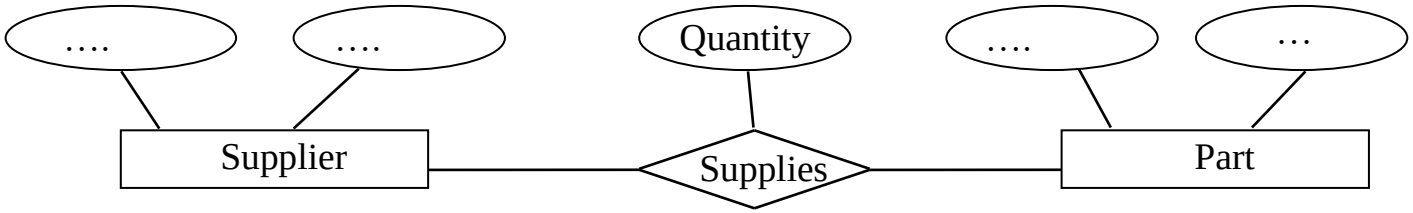
But for each "student – course" relationship there is an attribute called "grade".

So where an attribute goes depends on the problem situation and the assumptions about that problem.

For example, if a situation is such that parts are supplied by suppliers, and if only one supplier can supply a particular part then the ER diagram will be:



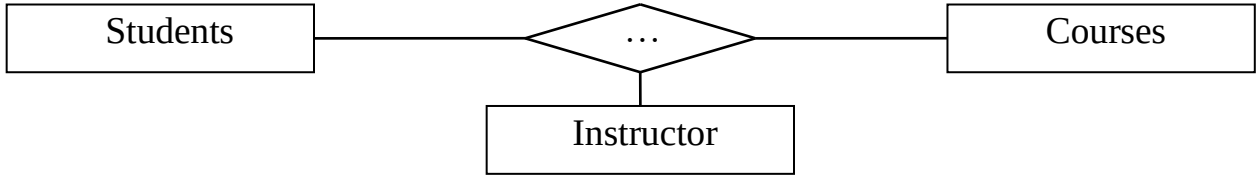
If many suppliers can supply the same part in different quantities, then the ER diagram will be:

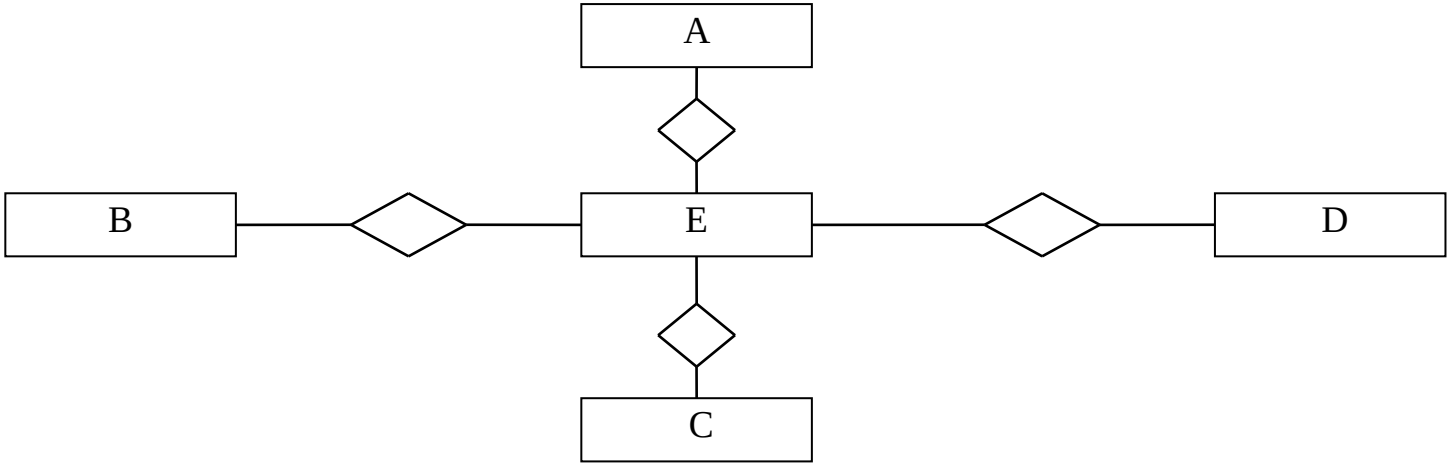


Data Modeling (Cont.)

For example, Supplier A can supply 10 units of part1. Supplier B can supply 20 units of part1 and Supplier B can also supply 10 units of part2, and so on.

There can be many entity sets participating in a relationship and the same entity can participate in several relationships.





Data Modeling (Cont.)

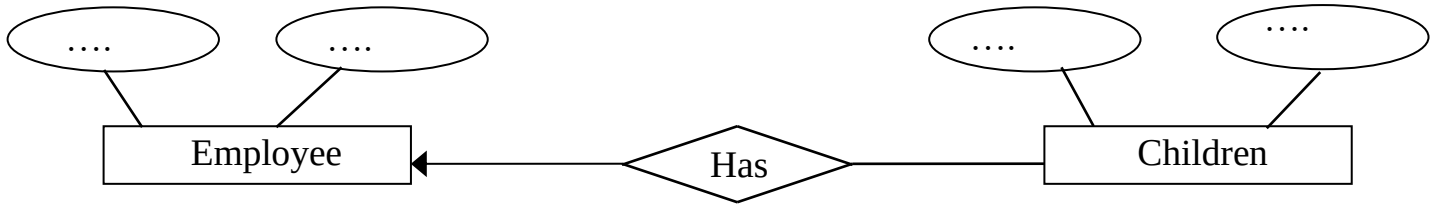
- **Dependent and Independent Entity Sets OR Weak Entity and Strong Entity Sets**

  Entity sets can be classified as Weak or Strong entity sets depending on the nature of their existence

- **A weak entity set depends on the strong entity set for its existence and a weak entity cannot exist by itself in the system**

  Entities in a weak entity set must have a strong entity to depend on in a system.

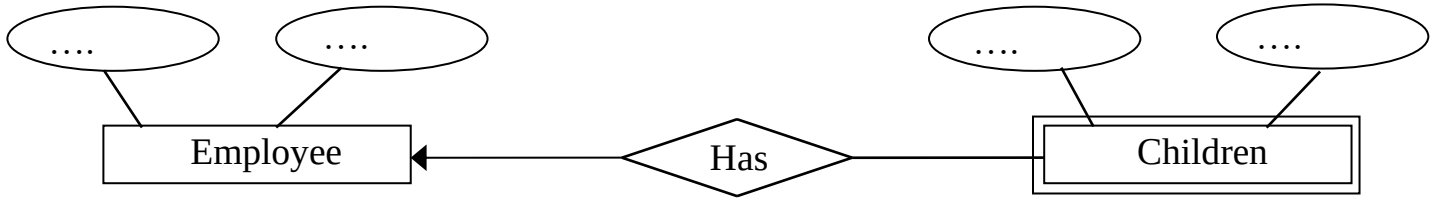
- **Example: Employee benefits information system**



Each employee can have many children. Each child must be a dependent of an employee in the system.

Information on a child entity cannot exist in this system without an employee.

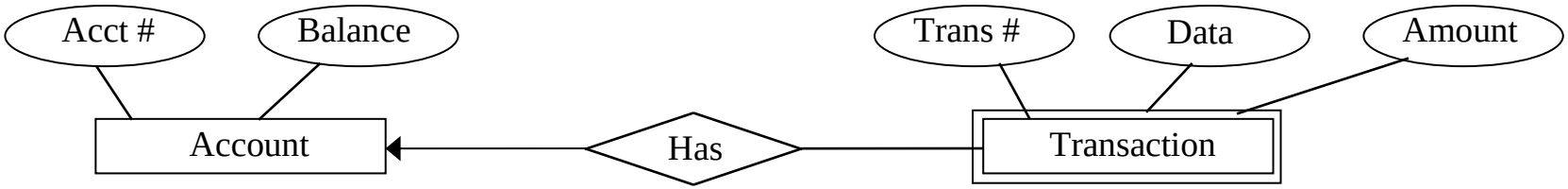
- **Weak entity sets are represented by double rectangles in an E-R diagram.**



Data Modeling (Cont.)

- **Another Example:**

Bank account transaction system



Each account can have many transactions.

Each transaction is for a particular account.

Without an account there can be no transactions.

Transactions can be withdraws or deposits.

An example of Transaction No. is the Check No. for a bank account.

But different accounts can have the same Check No. transacted.

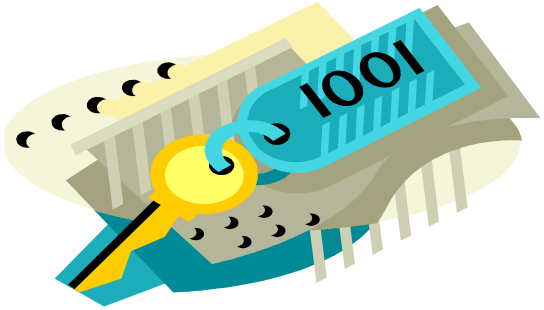
So, we need the Account # along with the Transaction No. to distinguish between transactions.

- **There are other ER situations and we will see them later**

Data Modeling (Cont.)

- **But first we need to talk about distinguishing entities from one another in an entity set.**
- **One or more attributes of an entity set should be chosen as keys to distinguish entities from one another.**

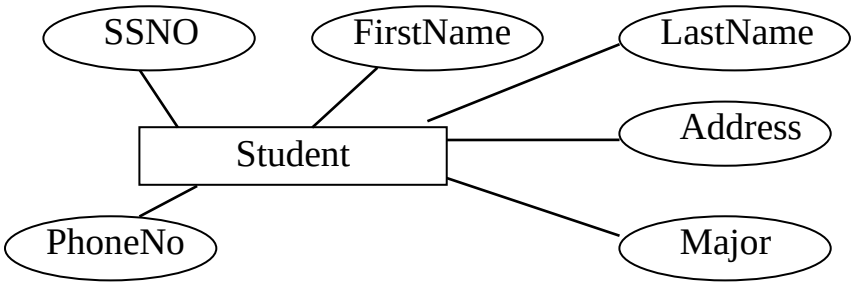
  Keys allow the unique identification of an entity in an entity set.



- **There is a simple procedure for determining the keys for an entity set.**
- **The selection of keys can have a tremendous impact on the performance of an information system.**
- **Since the attributes chosen as keys will be defined as indexes, we must select a minimum possible number of attributes as keys. Otherwise, you will unnecessarily end up with a large database system.**
- **Now you know why it helps to know more about how data are stored, accessed, updated, and how indexes are created.**
- **Database software products in general do not help you in selecting keys and selecting keys is part of the design process.**
- **Suppose, we have an entity set called "student" which has the following attributes:**

SSNO	**Major**
FirstName	**PhoneNo.**
LastName	**Address**

Systems Analysis and Design (Cont.)

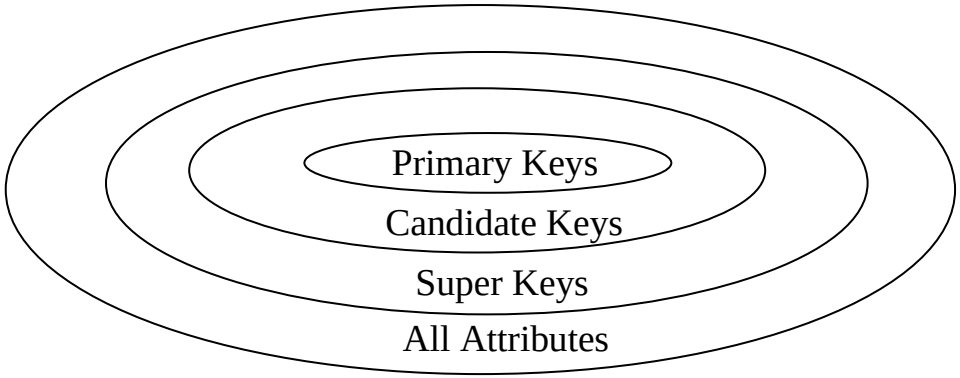


- **Can we distinguish one student entity from another by defining all attributes as keys?**

  Yes. But it is not necessary and it is not efficient since some of the attributes by themselves can uniquely identify a student.

- **The procedure for selecting keys involves first selecting the Super Keys for an entity set.**

- **Then from the set of Super Keys select the Candidate Keys, and finally select the Primary Keys from the set of Candidate Keys.**



Data Modeling (Cont.)

- **Super keys – all possible sets of attributes that can be chosen as keys to distinguish entities from one another in an entity set.**

- **What are the possible combinations of attributes that can uniquely identify each student from other students in the entity set? With certain assumptions, we can select the following super keys:**

  {SSNO}

  {FirstName, LastName, PhoneNo}

  {FirstName, LastName, Address}

  {FirstName, LastName, Major}

  Note: The attributes within { } form a set and all the attributes within { } together form a possible key

- **Why can't we select {SSNO, LastName } as super keys?**

  Because LastName is redundant since SSNO by itself uniquely identifies each student.

  Of course, the database software is not going to prevent you from making such design mistakes.

- **Candidate Keys – these are the smallest possible sets of super keys that are candidates for becoming primary keys. Therefore, the Candidate Keys must be selected from Super Keys.**

- **In the example, the smallest possible set is {SSNO}, and so this is the candidate key.**

Data Modeling (Cont.)

- **In some situations, we may have several candidate keys.**

  For example, assume that Social Security Numbers do not exist then the possible candidate keys could be

  {FirstName, LastName, Address } **{FirstName, LastName, Major }**

  {FirstName, LastName, PhoneNo}

  With the assumption that each set of attributes uniquely identifies a student.

- **Primary Keys – the best Candidate Key chosen based on storage space and efficiency and other considerations. The Primary Key must be selected from the set of Candidate Keys.**

  In the current example, SSNO is the only choice and so it will be chosen as the Primary Key.

  For example, if SSNO do not exist, then we will have to choose from:

  { FirstName, LastName, Address } **{FirstName, LastName, Major}**

  { FirstName, LastName, PhoneNo }

  Which of these is better?

  Consider storage space and efficiency. The second one may be better since storing Phone No. may take less space than Address or Major, and Phone Numbers (integers) are much easier to sort and search.

Data Modeling (Cont.)

- **Common mistakes and considerations in deriving keys**

For the sake of illustration, let us assume all the attributes of an entity set Part are:
{PartName, BrandName, Description, Dimensions, Color, and Weight}

Based on some assumptions, let the superkeys be: {PartName, BrandName}
{BrandName, Description}
{Description, Dimensions}

This means that we need attributes {PartName, BrandName} or {BrandName, Description} or {Description, Dimensions} together to uniquely identify each part in the entity set Part.

If attributes {PartName, BrandName} alone can uniquely identify each Part in the entity set then it is a mistake to include any other attribute in that set of attributes as it will be redundant to do so and inefficient.

Similarly, if the assumption is that attributes {BrandName, Description} or {Description, Dimensions} can uniquely identify each part then it is a mistake to include the other attributes PartName, Color, and Weight.

Candidate keys must be the superkey(s) with the fewest number of attributes in the superkeys, and so all three sets of attributes will be Candidate Keys as they all have 3 attributes to uniquely identify each part.

The candidate key(s) that require least storage of the three possible candidate keys is {PartName, BrandName} and so that will be the Primary Key for the entity set Part.

Note: Curly brackets { } are used just to group attributes into a compound key
If there has been an attribute PartNo with numerical values then it would have been the PrimaryKey

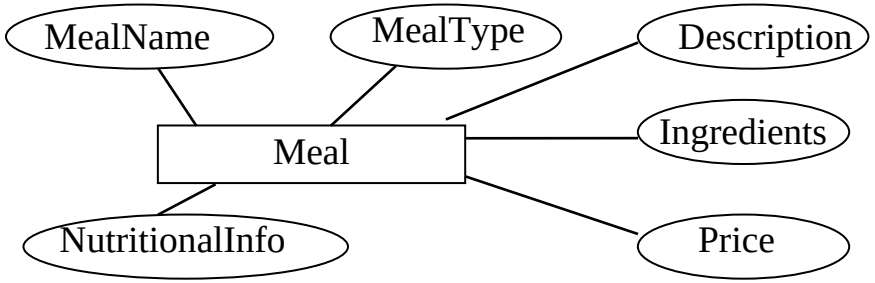
Data Modeling (Cont.)

- **Another example on deriving keys:**

  A diet organization supplies its clients with meals. Each meal is characterized by a meal name (e.g. FitnessOne, HealthEats, DietSimple, etc.)**, meal type** (e.g. breakfast, lunch, dinner, snack, etc.), **description** (details of the meal), **ingredients, price, and nutritional info** (e.g. calories, fat, etc.).

  By making reasonable assumptions and not adding any other attributes, derive the super keys, candidate keys, and primary key(s) just for the entity "Meal" and explain briefly the reasons for your choice.

  We can draw the E-R diagram as follows for the entity "Meal" in problem situation and make up some data for the attributes as in the table below to understand the entity and its attributes better:



MealName	MealType	Description	Price$	Ingredients	NutritionalInfo
FitnessOne	Breakfast	Omlettee and yogurt	4.00	Eggs, milk, corn oil, sodium, ..	Calories 400, Fat 10g,
HealthEats	Lunch	Soup and salad	5.00	Tomato paste, chicken broth, sodium, vegetables, ..	Calories 300, Fat 5g, ….
DietSimple	Lunch	Cheese sandwich	4.00	Whole wheat bread, cheese, sodium, …	Calories 500, Fat 4g, ….
CarbControl	Dinner	Fish and vegetables	8.00	Fish, vegetables, cooking oil, sodium ..	Calories 400, Fat 10g,
MuscleBuild	Dinner	Pasta and vegetables	8.00	Pasta, vegetables, sodium, …	Calories 500, Fat 5g, ….

Data Modeling (Cont.)

- **Another example on deriving keys (Cont.):**

  All attributes of the entity Meal are:

  {MealName, MealType, Description, Price, Ingredients, NutritionalInfo}

  From all attributes the Superkeys that can uniquely identify each entity in the set can be derived as follows with the assumption that all the chosen attribute(s) will have unique values:

  {MealName}
  {Description}
  {MealType, Ingredients, NutritionalInfo)

  As there are three choices for Superkeys, we can select the ones with the fewest number of attributes as Candidate Keys (that is, candidates to be chosen as the Primary Key) as follows:

  {MealName}
  {Description}

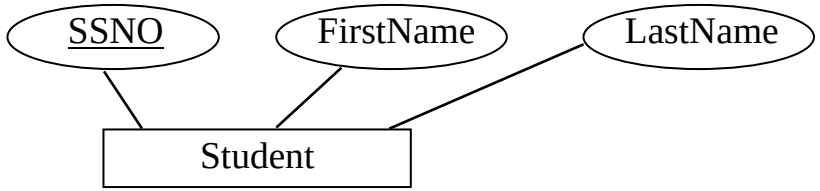
  From the set of Candidate Keys, Primary Key(s) can be chosen based on size of the data value for the attribute, stability of the values, etc., as follows:

  {MealName}

  The assumption is that the candidate keys, MealName and Description, are unique but MealName is chosen as the primary key because its possible values are stable and require less storage than the possible values of the attribute Description (which may be all long string of text).

Data Modeling (Cont.)

- **The attributes chosen as primary keys will be defined as indexes in the database system.**
- **Primary keys are indicated in the ER diagram by underlining the attribute names.**
- **Example:**



- **Characteristics of primary keys: a primary key should be:**

1. **Stable – must not change or become null (i.e. without a value) during the life of an entity.**

   e.g. a stock symbol may not be a stable primary key as companies can merge.

2. **Minimal – must have fewer attributes as possible; a compound key (that is, several attributes together forming the primary key) is not preferable but in some cases a compound key it is unavoidable.**

3. **Factless – the attribute value should not hold additional information about the entity.**

   This is very important. But it is often violated. If the meaning is stored in a key, the meaning may be lost when the designer leaves or the key can become obsolete when the meaning has to be modified.

Data Modeling (Cont.)

- **Characteristics of primary keys: a primary key should be (Cont.):**

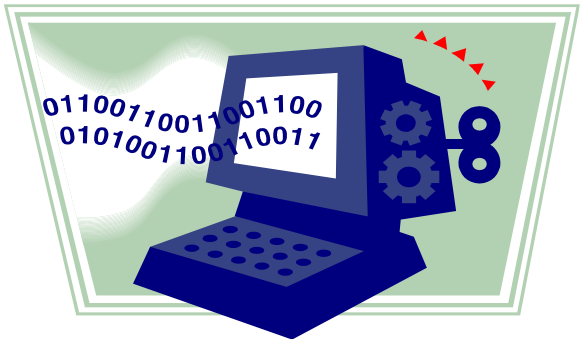
  Example: if the primary key is EmployeeId and if you implicitly give the following meanings to ID's:

  000 – 100 → Managers

  101 – 800 → Employees

  801 – 900 → etc.

  Then you are not using the key to distinguish entities.



If the meaning is not documented or if the situation changes, this will affect the use or the future modifications of the system drastically.

4. **Definitive – that is, a value must exist for the primary key at the time of creation of entity.**

   Do not assume that a key value can be null or it can be entered later.

   Some database software won't let you define a primary key attribute and not enter a value.

5. **Accessible – must be accessible to all users who have the right to access that entity set. Otherwise someone could store a duplicate row of data.**

Remember: You have to define a primary key for each entity set (that is, table) in a database system

Data Modeling (Cont.)

- **Artificial Keys**

  Artificial key is an attribute added artificially to the set of attributes of the entity and is defined to be the primary key.

  Artificial keys should be the last resort and should be assigned under one of the following prescribed situations:

  1. **No attribute possesses all the characteristics of a primary key or**

  2. **Candidate keys are complex and large or**

  3. **Other valid reason such as privacy, i.e. use employee ID instead of SSNO.**

- **Foreign Keys**

  Every dependent entity should inherit the primary key of the independent entity.

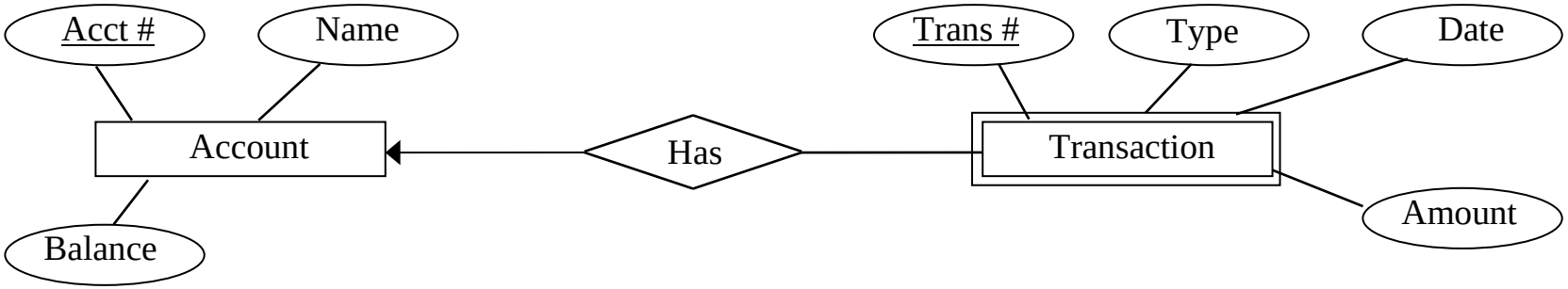
  The resulting composite key should be the primary key of the dependent entity.

  The inherited keys are called foreign keys.

  Foreign keys are needed in the case of dependent entity sets.

Data Modeling (Cont.)

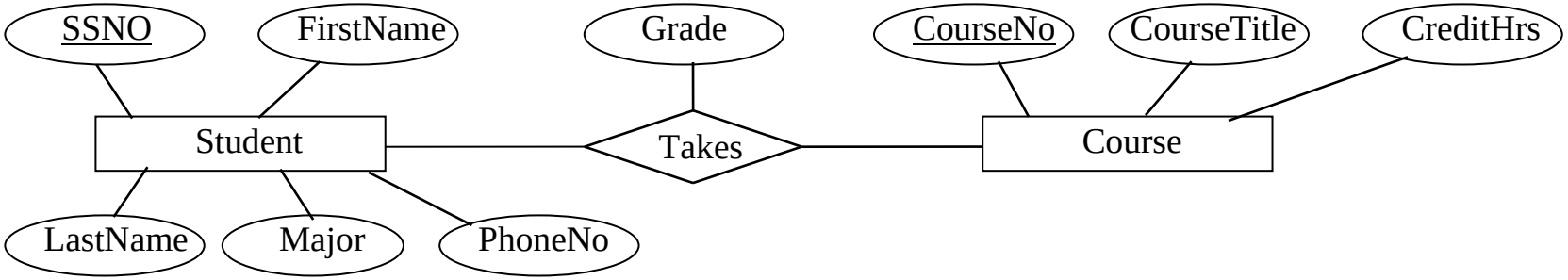
- **Example**



In this example, the weak entity set Transaction will inherit the primary key of strong entity set Account.

The primary keys of Transaction will be {Acct#, Trans#}. Without the key Acct # we cannot tell to which Acct# a transaction belongs to since many accounts can have the same transaction number.

- **Relationships also inherit the primary keys of related entities, but the inherited keys must NOT be shown in the E-R Diagram. Without inheriting the keys, a relationship cannot support or relate entities.**
  - **In the example below, the relationship "Takes" will inherit the primary keys SSNO and CourseNo to relate a Grade obtained by a Student for a particular Course.**



Data Modeling (Cont.)

- **Summary**

Data modeling of a problem involves capturing data and their relationships

ER diagram is a common data modeling technique, which uses the terminologies:

1. **Entity**
2. **attributes,**
3. **relationships,**
4. **cardinalities,**
5. **primary keys,**
6. **weak and strong entity sets,**
7. **inherited keys**

Identifying a suitable primary key for an entity set involves:

1. **deriving the superkeys from all attributes of the entity set,**
2. **then selecting the smallest of attributes from the superkeys to be the candidate keys, and finally,**
3. **selecting the candidate keys that require least storage to be the primary key for the entity set**

Factless is an important requirement for a primary key

Artificial attributes are added to an entity when there are no natural attributes available to become the primary key

Foreign keys are inherited primary keys in the cases of:

1. **a weak entity inheriting the primary key of the strong entity as foreign keys**
2. **relationships inheriting the primary keys of relating entity sets as foreign keys**