The Entity-Relationship (ER) Model

The E-R Model

The **E-R** (entity-relationship) data model views the real world as a set of basic **objects** (entities) and **relationships** among these objects.

It is intended primarily for the DB design process by allowing the specification of an **enterprise scheme**. This represents the overall logical structure of the DB.

Entities and Entity Sets

•An **entity** is an object that exists and is distinguishable from other objects. For instance, Michelle Lee with S.S.N.890-12-3456 is an entity, as she can be uniquely identified as one particular person in the universe.

•An entity may be **concrete** (a person or a book, for example) or **abstract** (like a holiday or a disease).

•An **entity set** is a set of entities of the same type (e.g., all persons having an account at a bank).

•Entity sets **need not be disjoint**. For example, the entity set *Student* (all students in a university) and the entity set *professor* (all professors in a university) may have members in common. (i.e a computer science professor might take a class in anthropology). 4

Entities and Entity Sets Continued

•An entity is represented by a set of **attributes**. (e.g. *name*, *SSN*, *Phone-Num* for "customer" entity.)

•The **domain** of the attribute is the set of permitted values (e.g. the telephone number must be ten positive integers).

•Formally, an attribute is a **function** which maps an entity set into a domain.

•Every entity is described by a set of (attribute, data value) pairs. There is one pair for each attribute of the entity set.

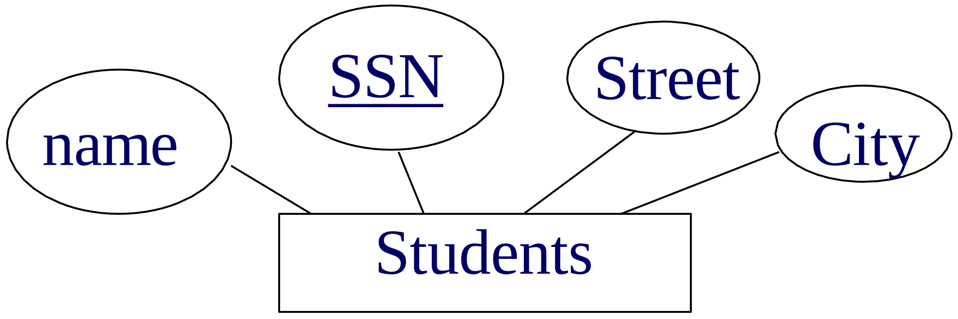
•e.g. a particular *student* entity is described by the set {(name, Lee), (SSN, 890-123-456), (street, Blaine), (city, Riverside)}.

E-R Diagrams

We can express the overall logical structure of a database **graphically** with an E-R diagram.

Its components are:

- **rectangles** representing entity sets.
- **ellipses** representing attributes.
- **diamonds** representing relationship sets.
- **lines** linking attributes to entity sets and entity sets to relationship sets.



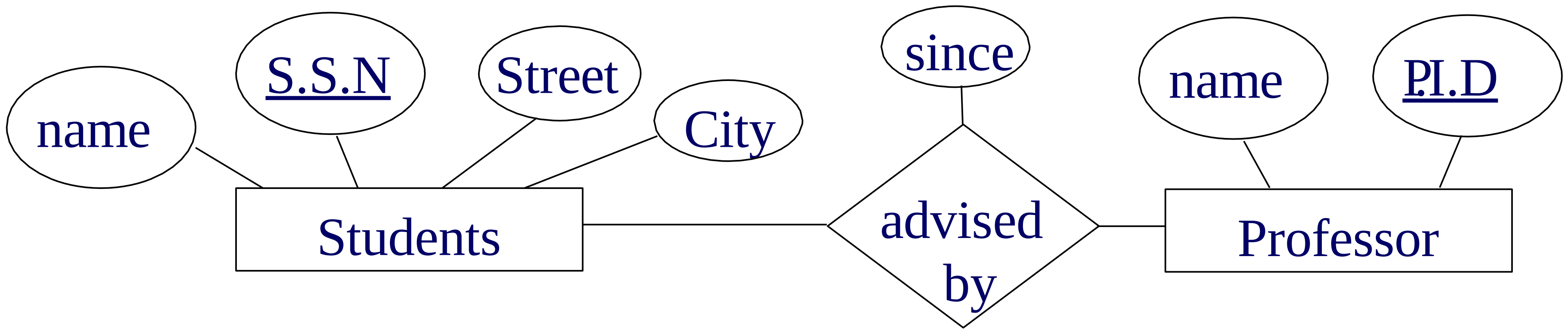
E-R Diagrams Cont.

We can express the overall logical structure of a database **graphically** with an E-R diagram.

Its components are:

- **rectangles** representing entity sets.
- **ellipses** representing attributes.
- **diamonds** representing **relationship** sets.
- **lines** linking attributes to entity sets and entity sets to relationship sets.

The "since" attribute in this example is called a **descriptive attribute**, since it describes the mapping from A to B

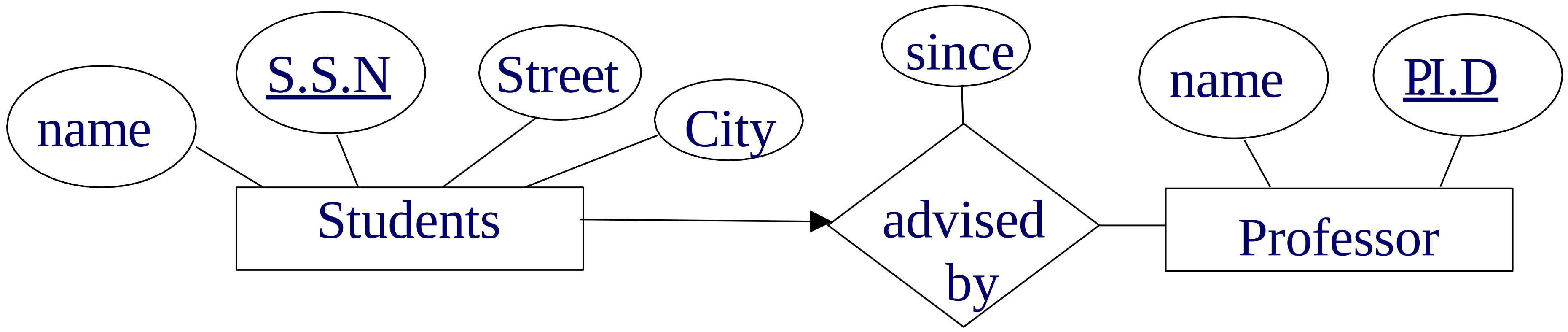


Key Constraints

We can also use arrows to indicate **key constraints** (often simply referred to as **constraints**)

Suppose the university has the following rule: A student is allowed to be advised by at most one professor. However, a professor is allowed to advise more than one student.

This is an example of a **many-to-one constraints**, that is, many students can be advised by one professor, but each student can only have (at most) one advisor. We can represent this with an arrow as shown below.



Key Constraints Continued

There are four possible **key constraints,** they express the number of entities to which another entity can be associated via a relationship.
For binary relationship sets between entity setsA and B,the mapping cardinality must be one of:

1.**One-to-one**:An entity in A is associated with at most one entity in B,and an entity in B is associated with at most one entity inA.

2.**One-to-many**:An entity in A is associated with any number in B. An entity in B is associated with at most one entity inA.

3.**Many-to-one**:An entity in A is associated with at most one entity in B.An entity in B is associated. with any number inA.

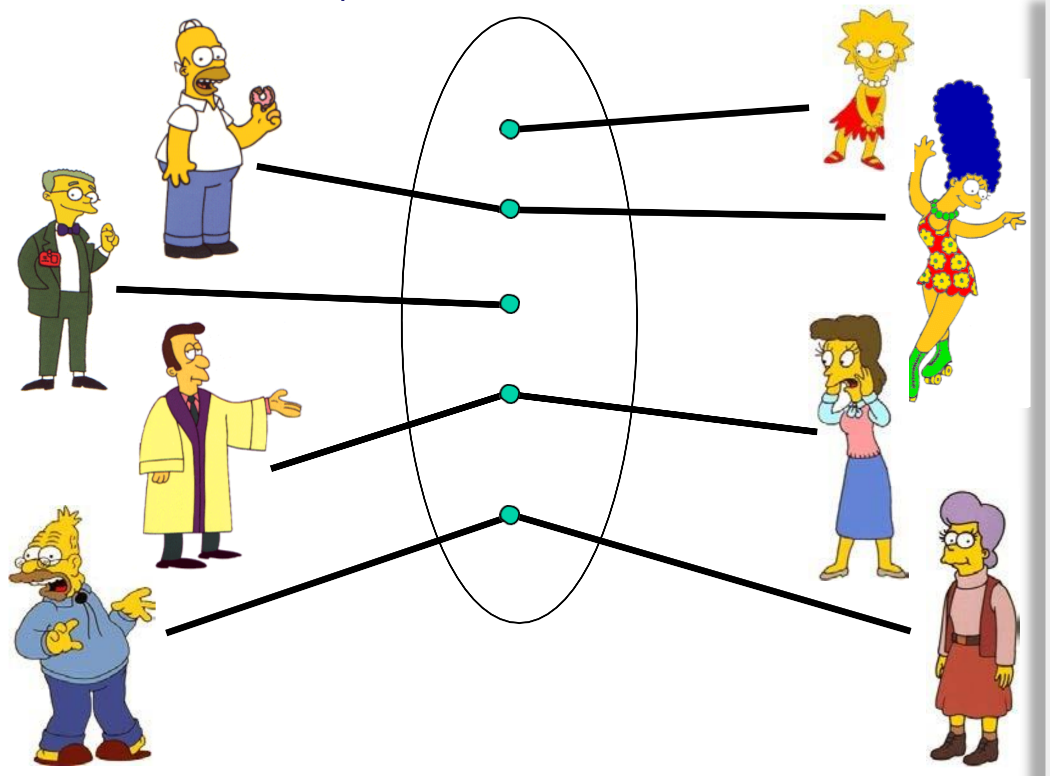
4.**Many-to-many**:Entities inA and B are associated with any number from each other.

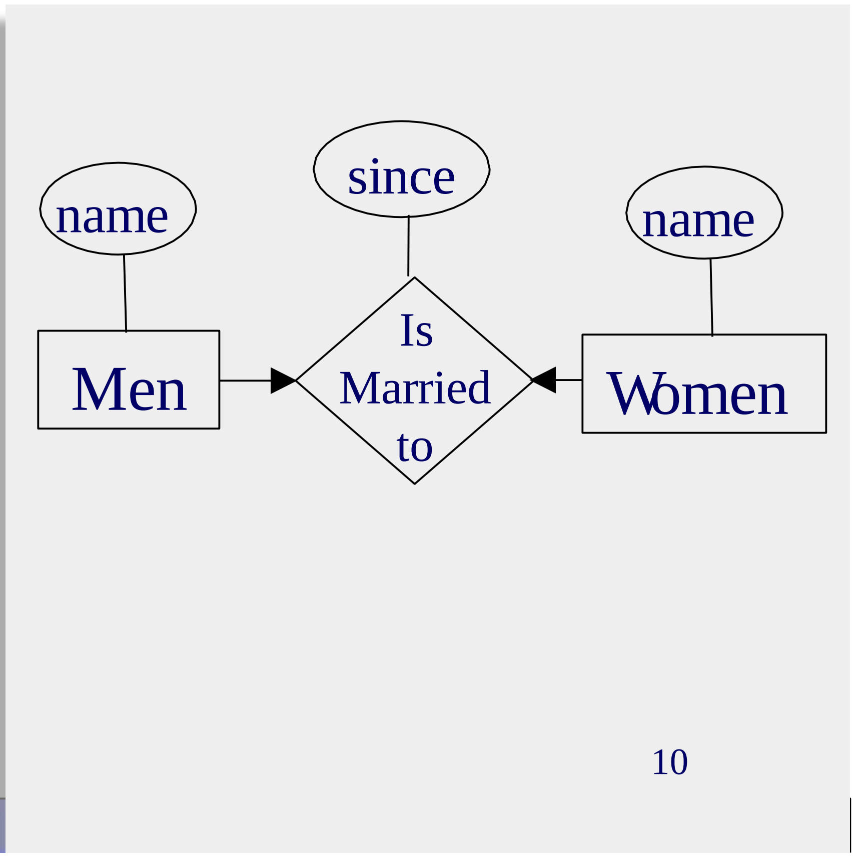
The appropriate **key constraint** for a particular relationship set 9 depends on the real world being modeled

Key Constraints: Examples

One-to-one: An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.

A man may be married to at most one woman, and a woman may be married to at most one man (both men and women can be unmarried)

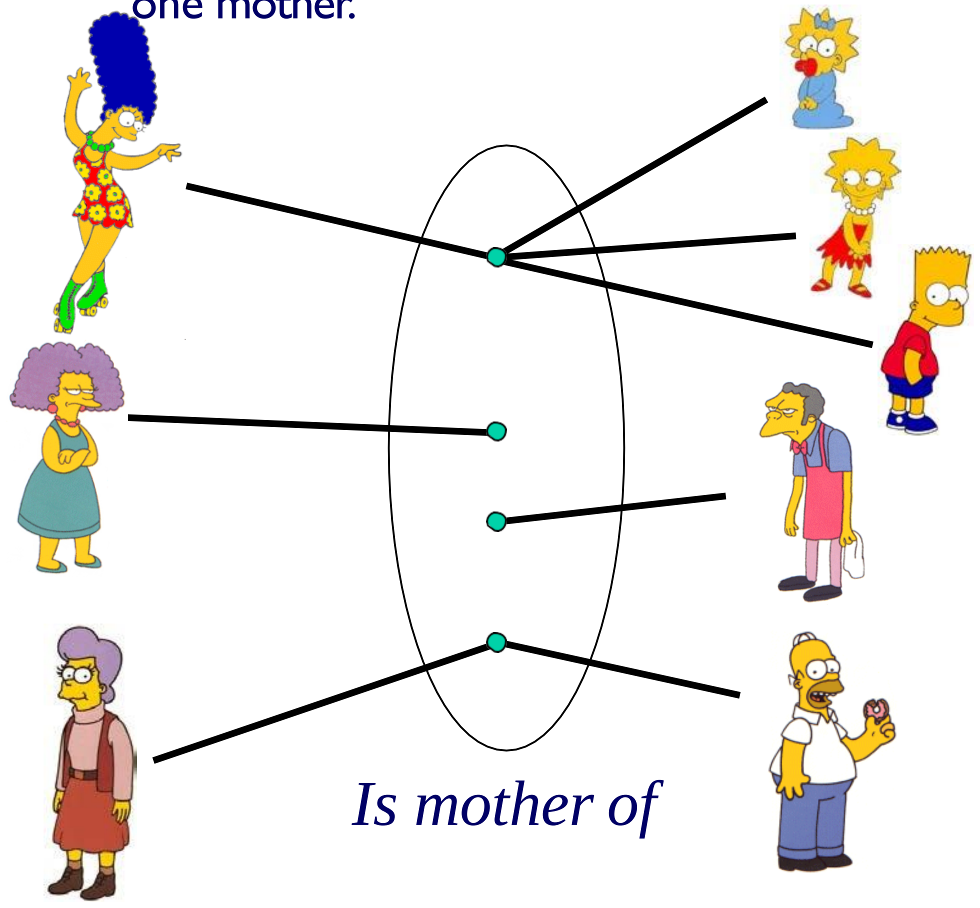


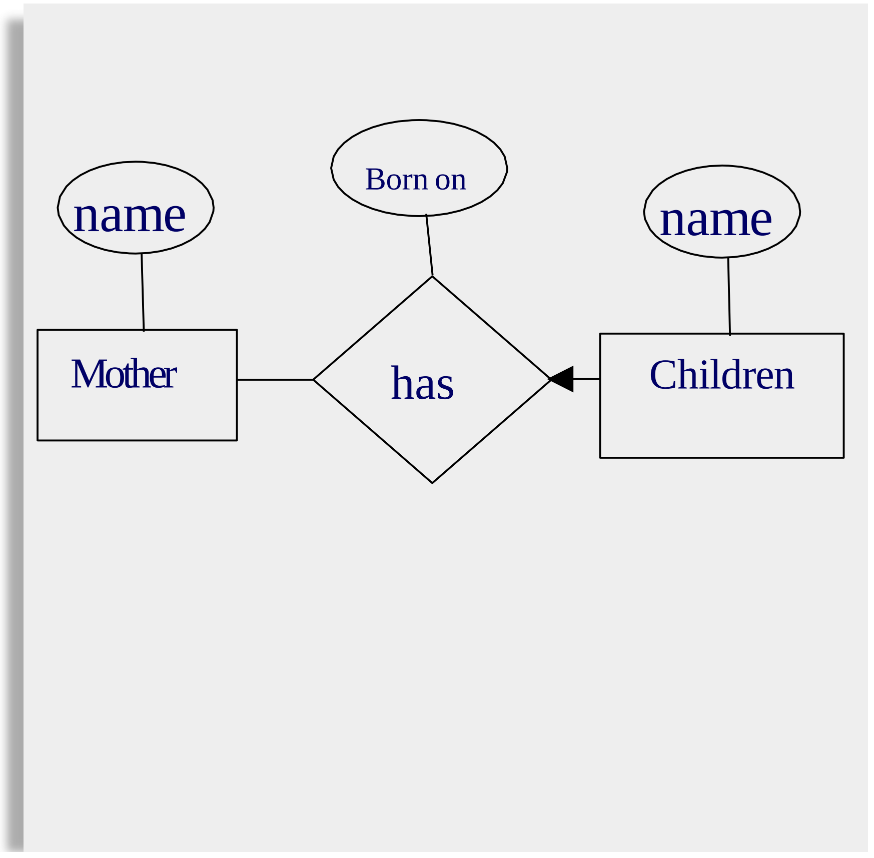


Key Constraints: Examples

One-to-many: An entity in A is associated with any number in B. An entity in B is associated with at most one entity in A.

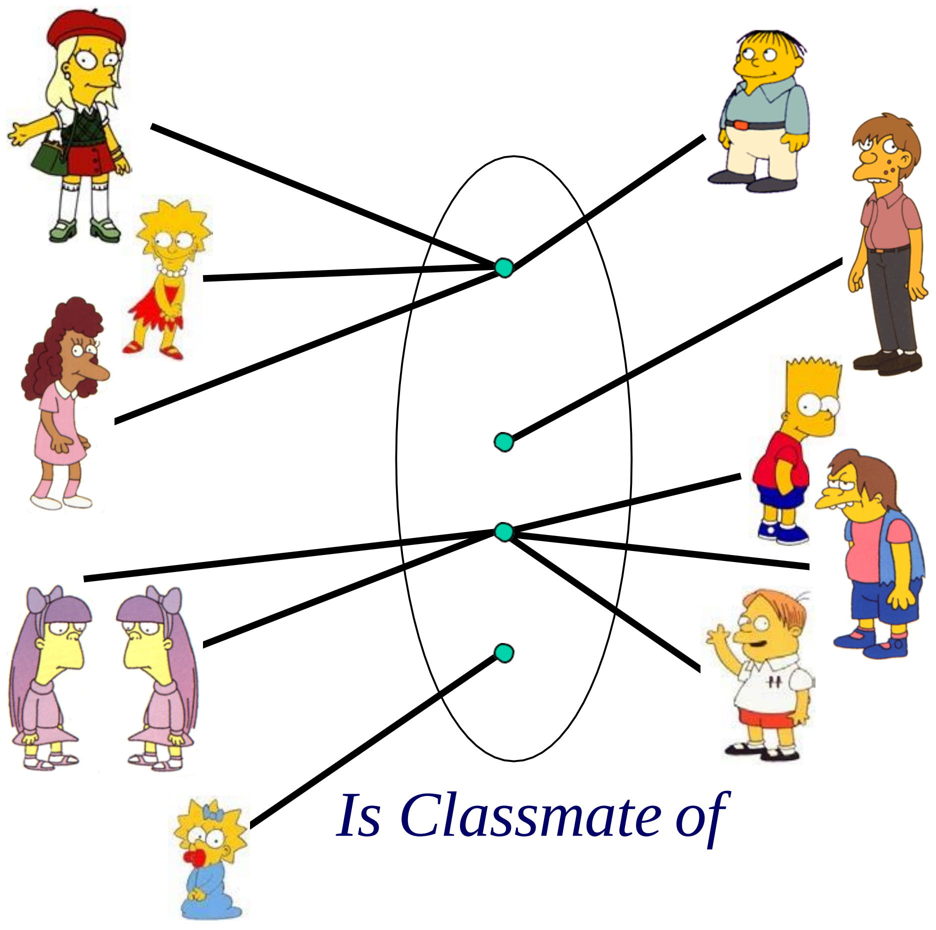
A woman may be the mother of many (or no) children. A person may have at most one mother.

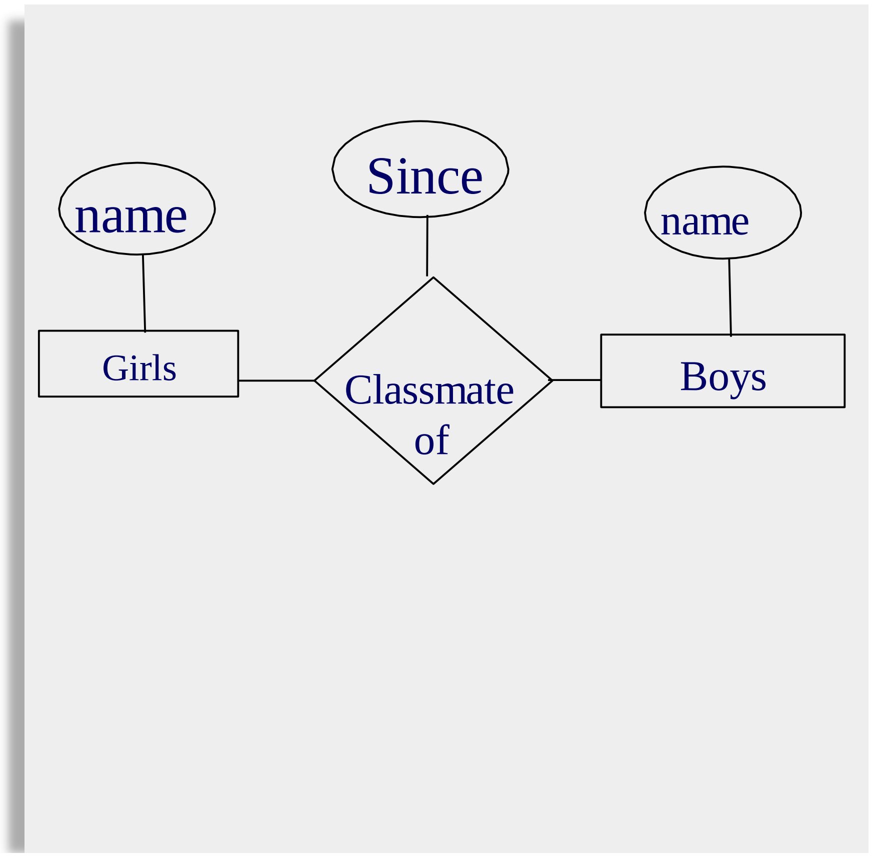




Key Constraints: Examples

Many-to-many: Entities in A and B are associated with any number from each other.





Key Constraints

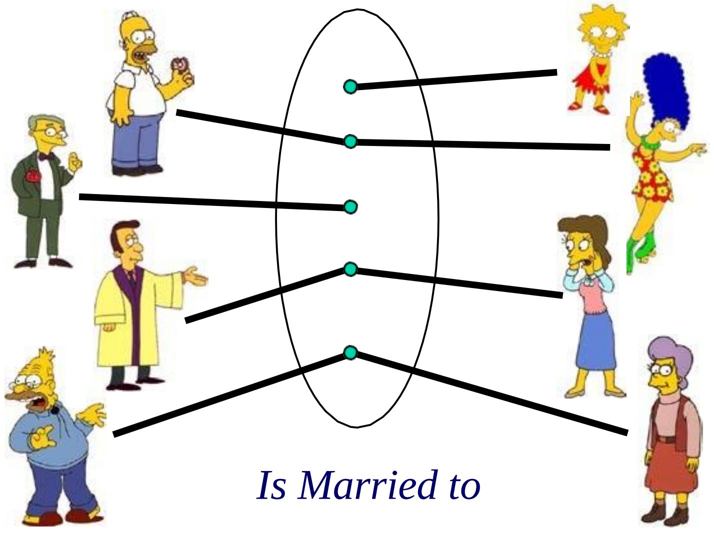
- The arrow positioning is simple, think of the arrow head as pointing to the entity that "one" refers to.
- Key constraints is also being referred to as "**Mapping Cardinalities**" or“**Multiplicity**”

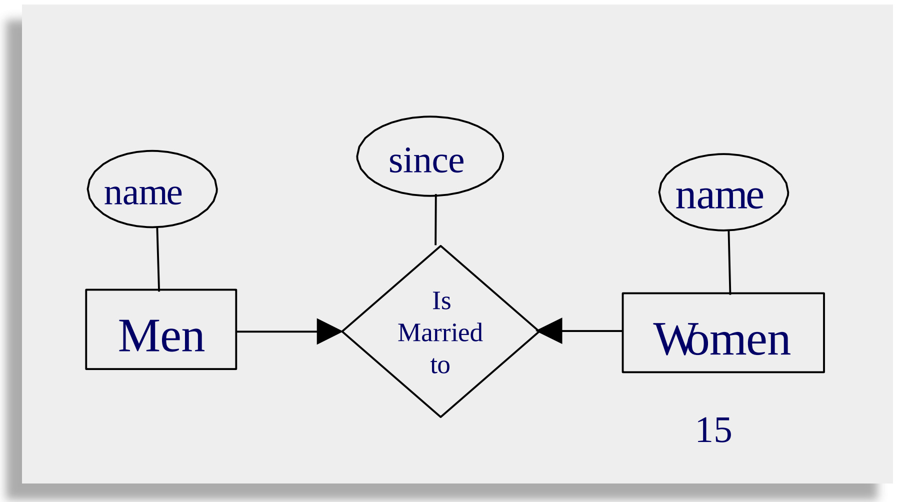
Participation Constraints

Earlier, we saw an example of a one-to-one key constraint, noting that a man may be married to at most one woman, and a woman may be married to at most one man (both men and women can be unmarried).

Suppose we want to build a database for the "Santa Clara Married Persons Association". In this case *everyone* must be married! In database terms their participation must be **total**. (the previous case that allows unmarried people is said to have **partial** participation.)

How do we represent this with ER diagrams? (answer on next slide)

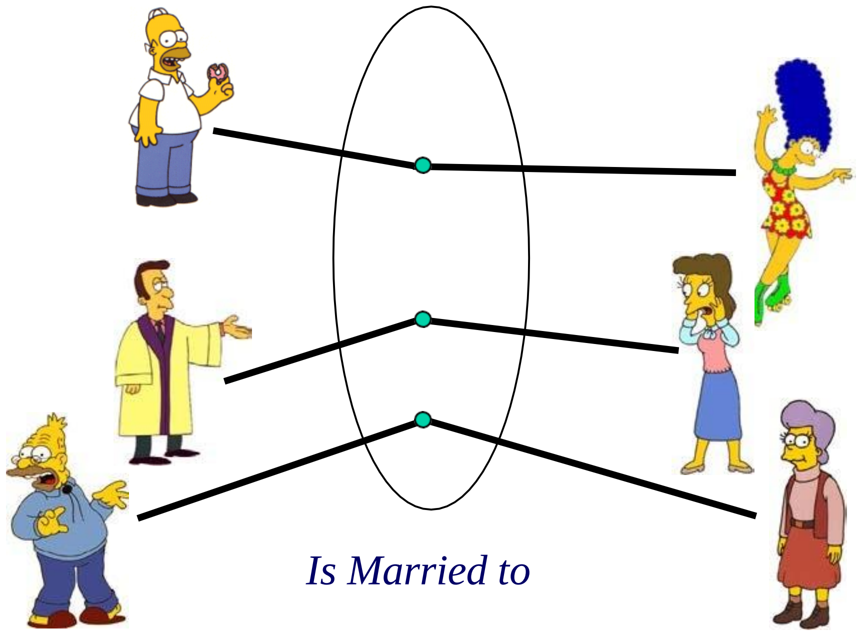


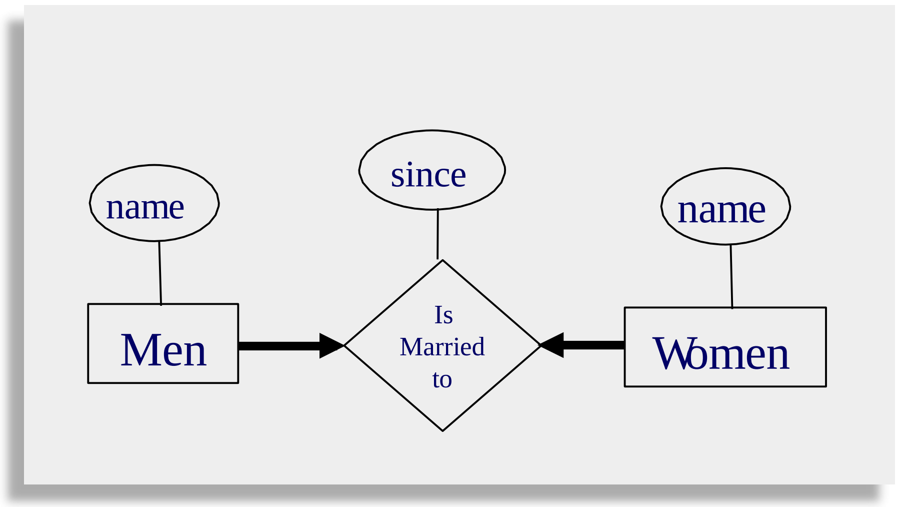


Participation Constraints

Participation Constraints are indicated by bold lines in ER diagrams.

We can use bold lines (to indicate participation constraints), and arrow lines (to indicate key constraints) independently of each other to create an expressive language of possibilities.

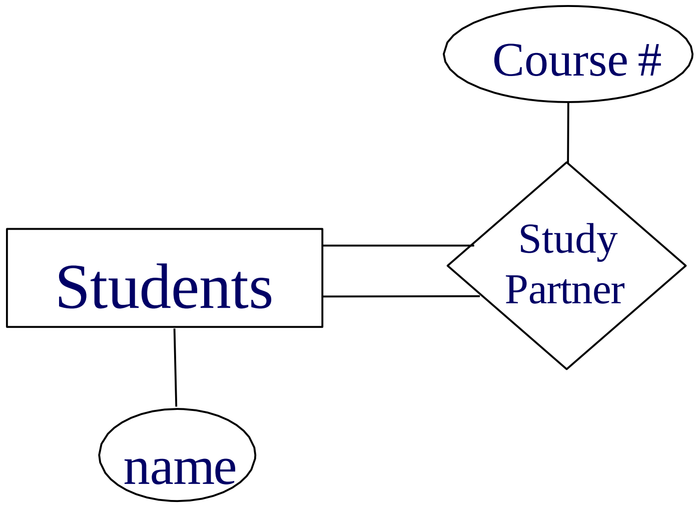


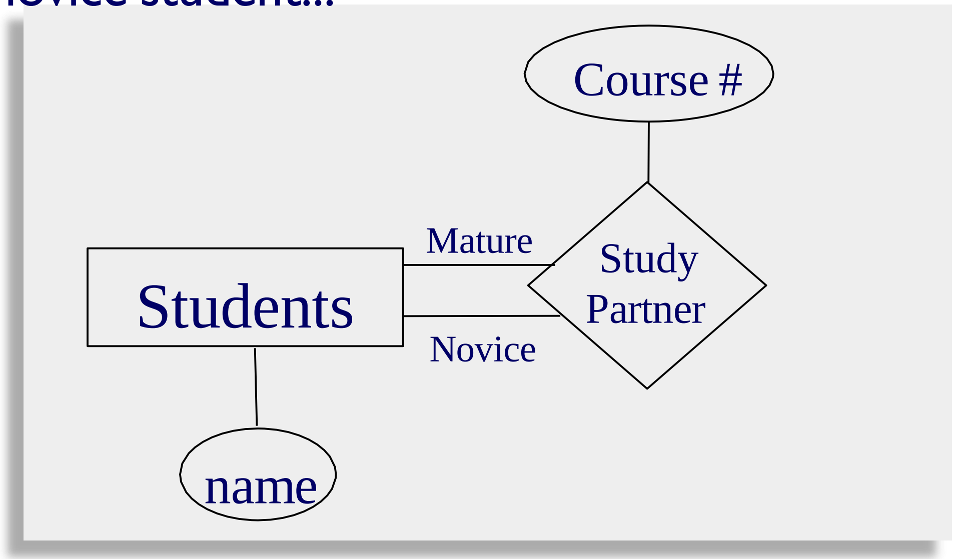


More on Relations

- Entities sets can be related to themselves.

We can annotate the **roles** played by the entities in this case. Suppose that we want to pair a mature student with a novice student...



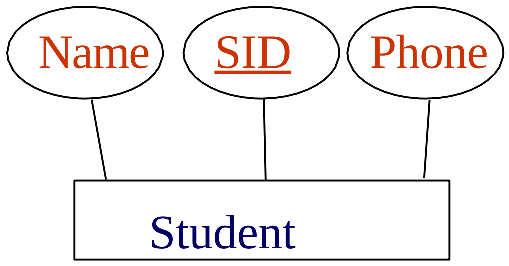


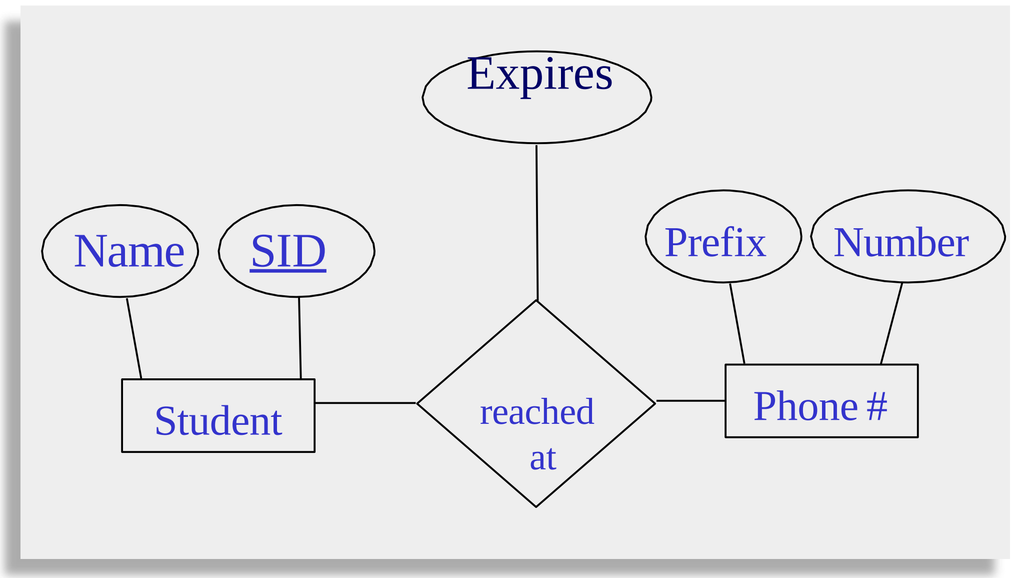
When entities are related to themselves, it is almost always a good idea to indicate their roles.

Entity vs.Attribute

Sometimes we have to decide whether a property of the world we want to model should be an attribute of an entity, or an entity set which is related to the attribute by a relationship set.

A major advantage of the latter approach is that we can easily model the fact that a person can have multiple phones, or that a phone might be shared by several students. (attributes can not be set-valued)

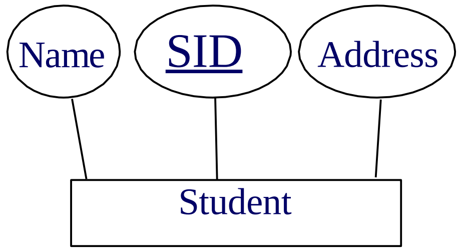




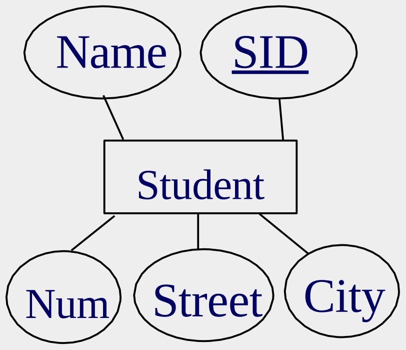
Entity vs.Attribute Cont.

A classic example of a feature that is best modeled as a an entity set which is related to the attribute by a relationship set is an *address*.

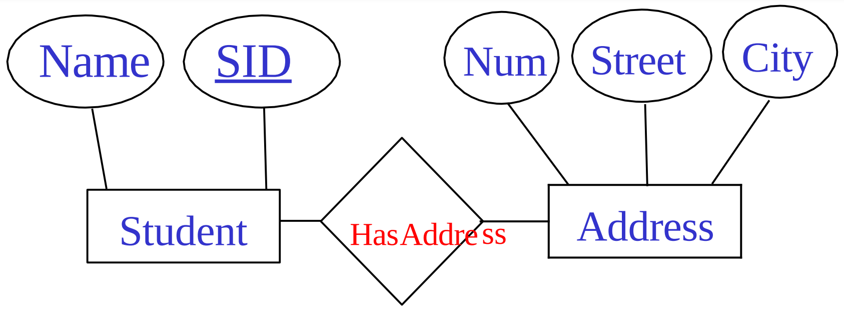
Bad choice for most applications. It would make it difficult to print mailing labels, it would make it difficult/impossible to do queries such as "how many students live in Santa Clara?"



A better choice, but it only allows a student to have one address. Many students have a two or more address (i.e. a different address during the summer months) This method cannot handle this.



The best choice for this problem



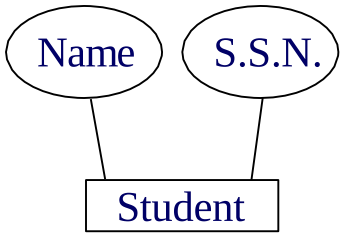
Keys

For each entity set, we choose a key.

•A **superkey** is a set of one or more attributes which, taken collectively, allow us to identify uniquely an entity in the entity set.

•For example, in the entity set *student*, {*name*, *S.S.N.*} is a superkey.

•A superkey may contain extraneous attributes, and we are often interested in the smallest superkey. A superkey for which no subset is a superkey is called a **candidate key** (minimal superkey)



Name	**S.S.N.**
Lisa	111-11-1111
Bart	222-22-2222
Lisa	333-33-3333
Sue	444-44-4444

We can see that {*Name*, *S.S.N.*} is a **superkey**

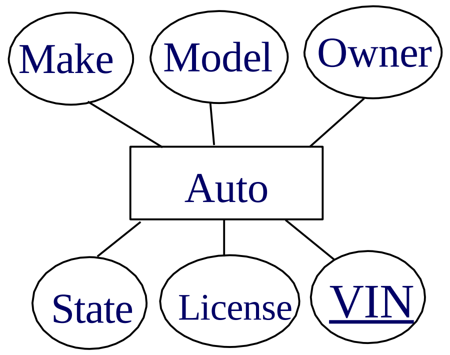
In this example, *S.S.N.* is a **candidate** key, as it is minimal, and uniquely identifies a students entity.

Keys cont.

•The **primary key** is denoted in an ER diagram by underlining.

•A **primary key** is a candidate key (there may be more than one) chosen by the DB designer to identify entities in an entity set.

•An entity that has a primary key is called a **strong entity**.



Note that a good choice of primary key is very important!

For example, it is usually much faster to search a database by the primary key, than by any other key (we will see why later).

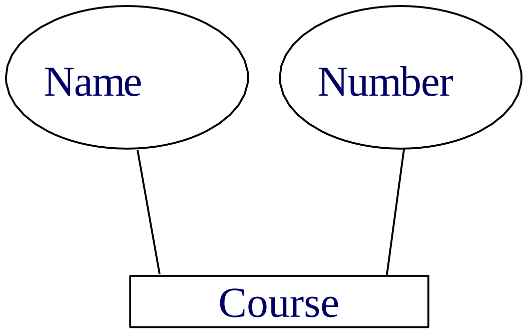
Keys cont.

An entity set that does not possess sufficient attributes to form a primary key is called a **weak entity set**.

In the example below, there are two different sections of Java being offered (let's say, for example, one by Dr. Smith, one by Dr. Lee).

{Name,Number} is not a superkey, and therefore *course* is a **weak entity**.

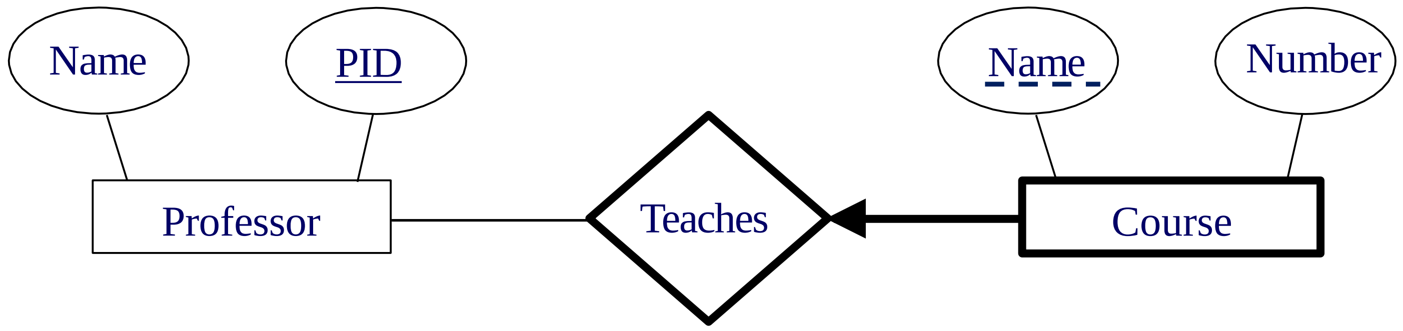
This is clearly a problem, we need someway to distinguish between different courses….



Name	Number
Java	CS211
AI	CS480
Java	CS211
DB	CS450

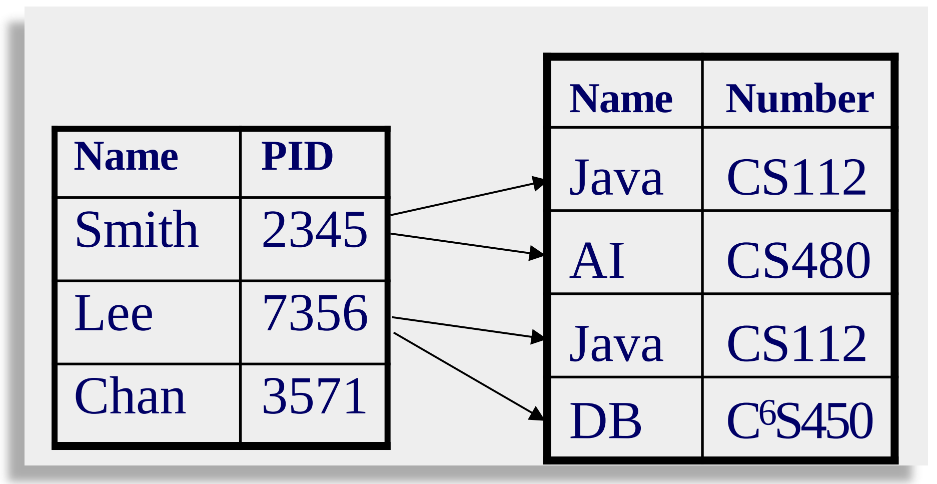
Keys cont.

In order to be able to uniquely refer to an item in a weak entity set we must consider some (or all) of its attributes in conjunction with some strong entities primary key.The entity whose primary key is being used is called the **identifying owner**.



For this to work, two conditions must be met.

- The weak entity set must have total participation in the relationship
- The identifying owner and the weak entity must participate is a one-to-many relationship.

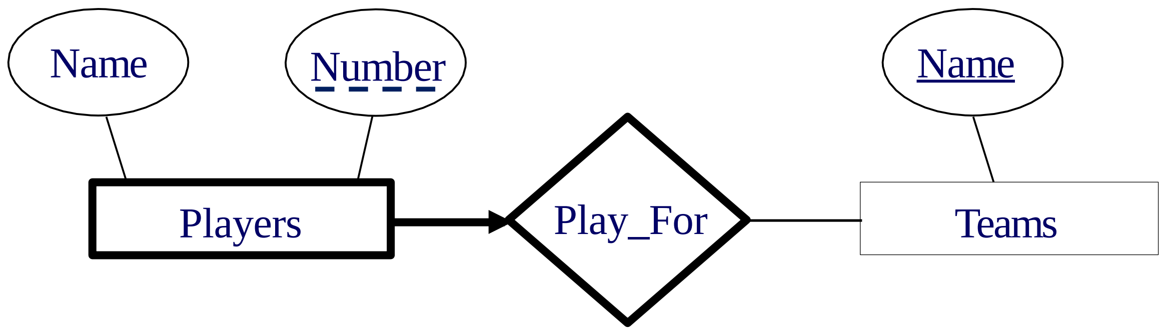


Name	**PID**
Smith	2345
Lee	7356
Chan	3571

Name	**Number**
Java	CS112
AI	CS480
Java	CS112
DB	CS450

Weak Entity Sets Example

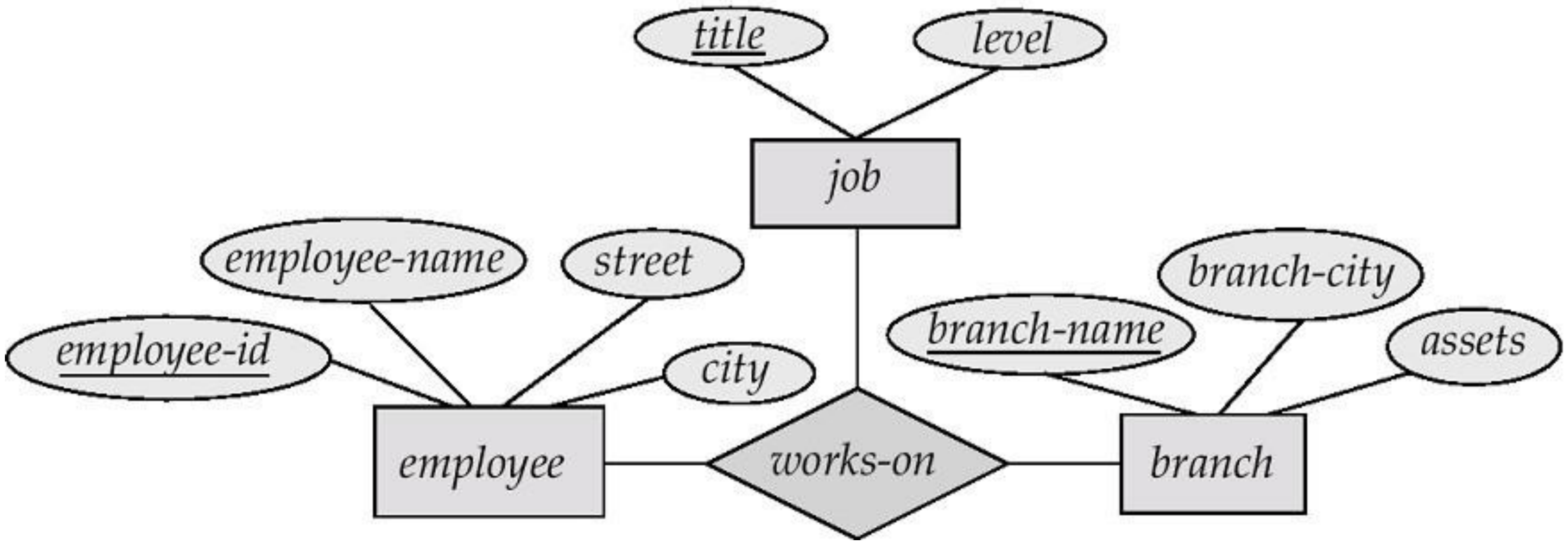
- Entity sets Teams, and Players.
  - No team has two players with the same number.
  - However, there can be players with the same number on different teams



Ternary Relationships

So far, we have only considered binary relationships, however it is possible to have **higher order** relationships, including **ternary** relationships.

Consider the following example that describes the fact that employees at a bank work in one or more bank branches, and have one or more job descriptions.



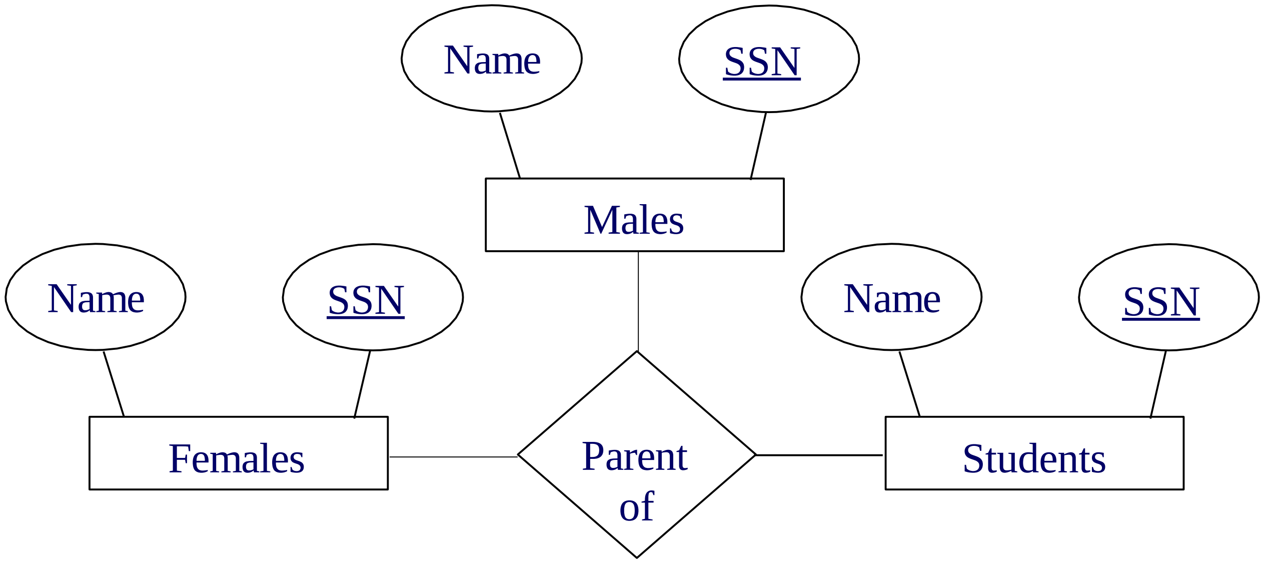
Why not remove the *job* entity by placing the *title* and *level* attributes as part of employee?

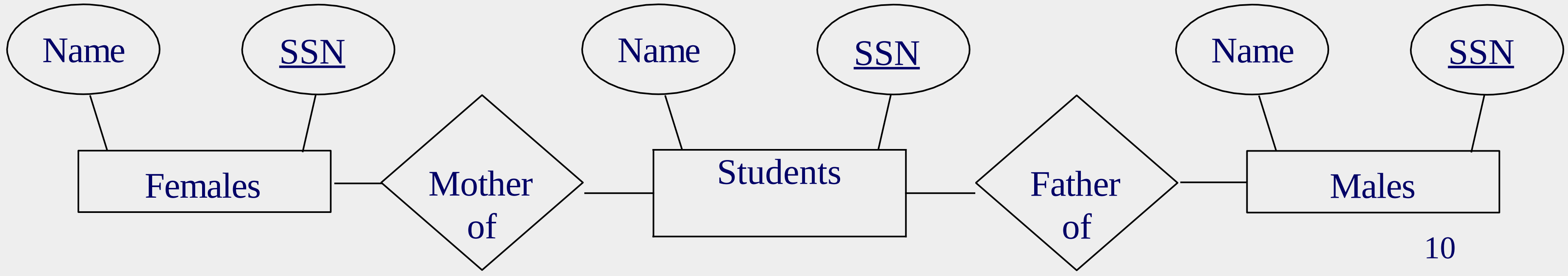
Ternary Relationships

Sometimes you have a choice of a single ternary relationship or two binary relationships…

In general, unless you really need a ternary relationship, use binary relationships.

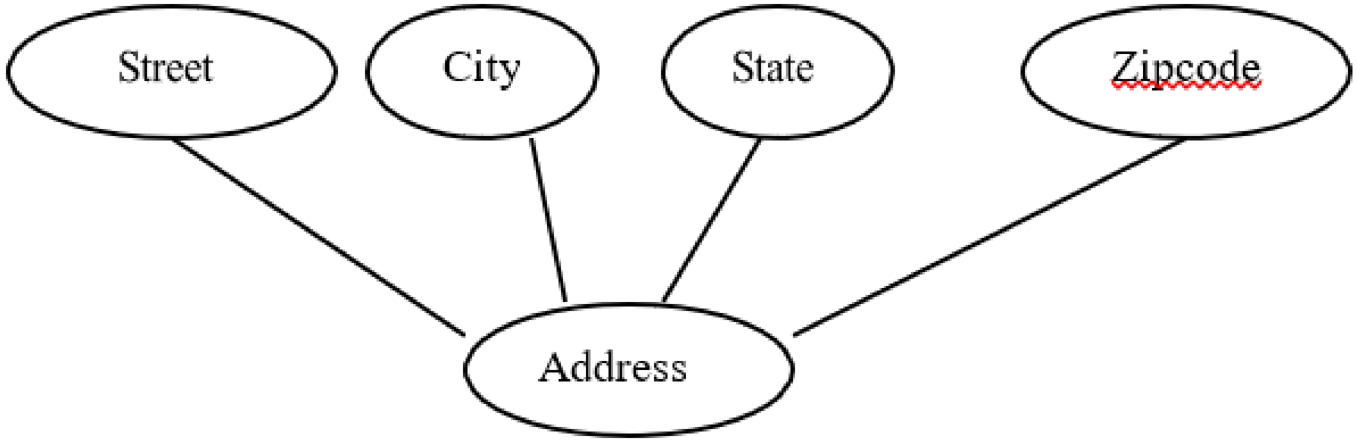
FACT: Every ternary (and higher order) relationship can be converted into a set of binary relationships.





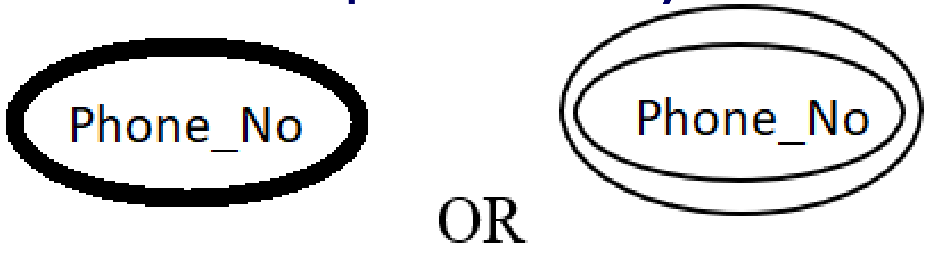
Types of Attributes - 1

- *Composite vs Simple Attributes*: An attribute composed of many other attribute is called as composite attribute.
  - Address attribute of student Entity type consists of Street, City, State, and Zip.
  - In ER diagram, composite attribute is represented by an oval comprising of ovals.
- Useful when we sometimes need to refer to the entire attributes as a unit and sometimes to each of components:

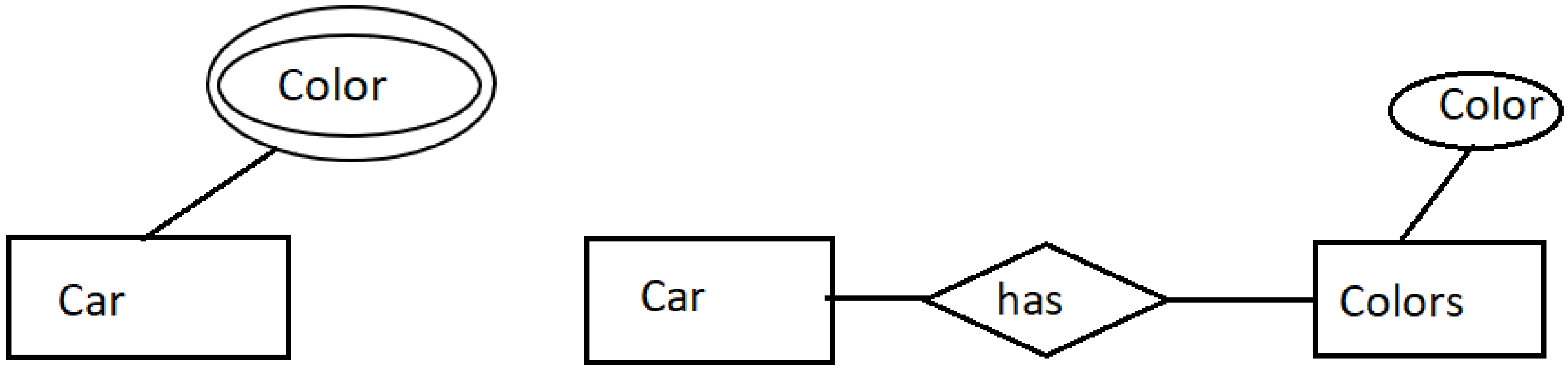


Types of Attributes - 2

- *Multivalued vs Single Attributes*: An attribute consisting of more than one value for a given entity.
  - Phone_No (can be more than one for a given student).
  - In ER diagram, multivalued attribute is represented by double or bold oval.

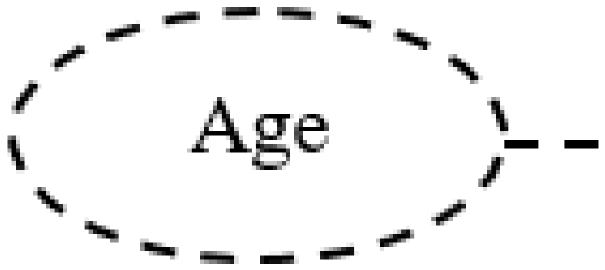


- An attribute with a set of values for the same entity.



Types of Attributes - 3

- *Derived vs Stored Attributes*: An attribute which can be derived from other attributes of the entity. e.g.; Age (can be derived from DOB).
  - In ER diagram, derived attribute is represented by dashed oval.



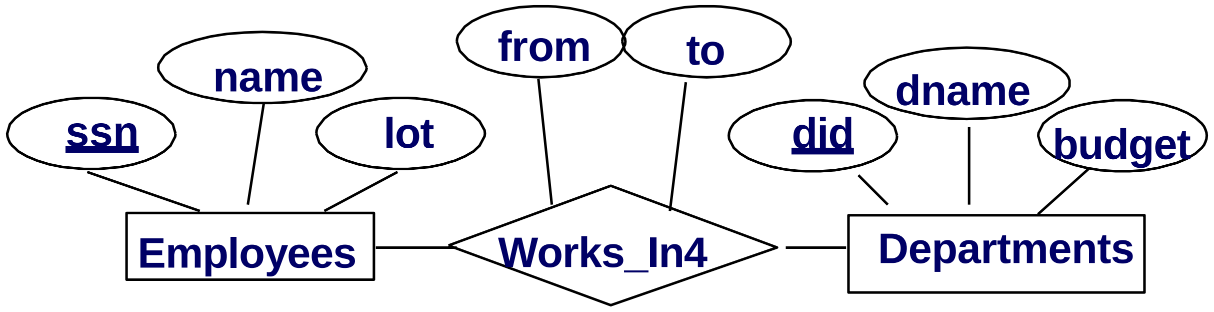
ER Design Decisions

- The use of an attribute or entity set to represent an object.
- Whether a real-world concept is best expressed by an entity set or a relationship set.
- The use of a ternary relationship versus a pair of binary relationships.
- The use of a strong or weak entity set.

Entity vs. Attribute

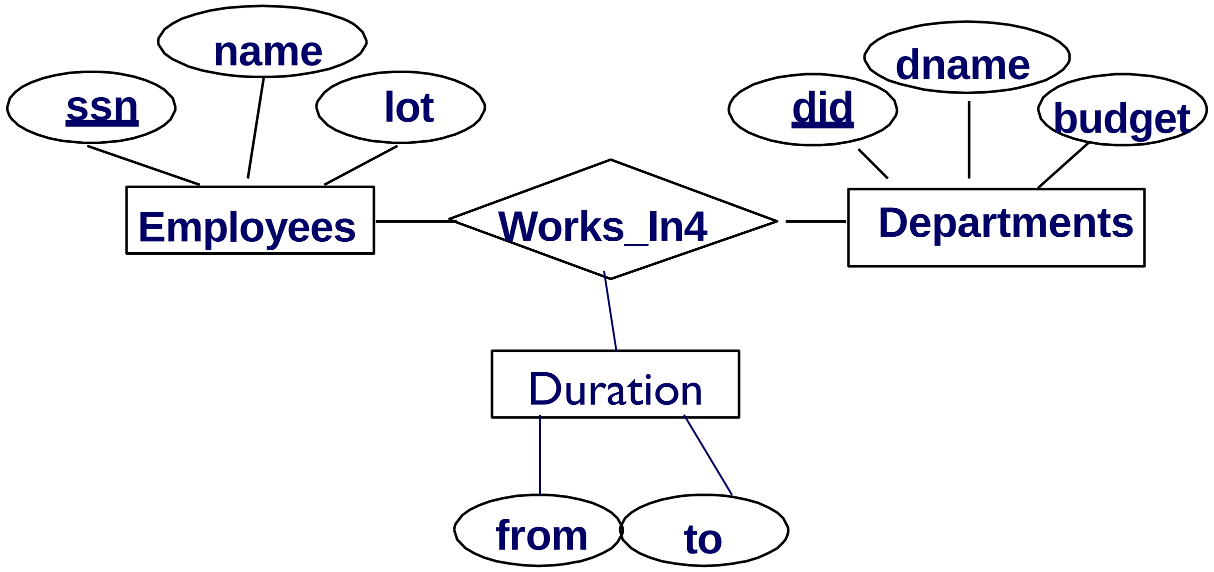
- Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?
- Depends upon the use we want to make of address information, and the semantics of the data:
  - If we have several addresses per employee, *address* must be an entity (since attributes cannot be set-valued).
  - If the structure (city, street, etc.) is important, e.g., we want to retrieve employees in a given city, *address* must be modeled as an entity (since attribute values are atomic).

Entity vs. Attribute (Cont.)

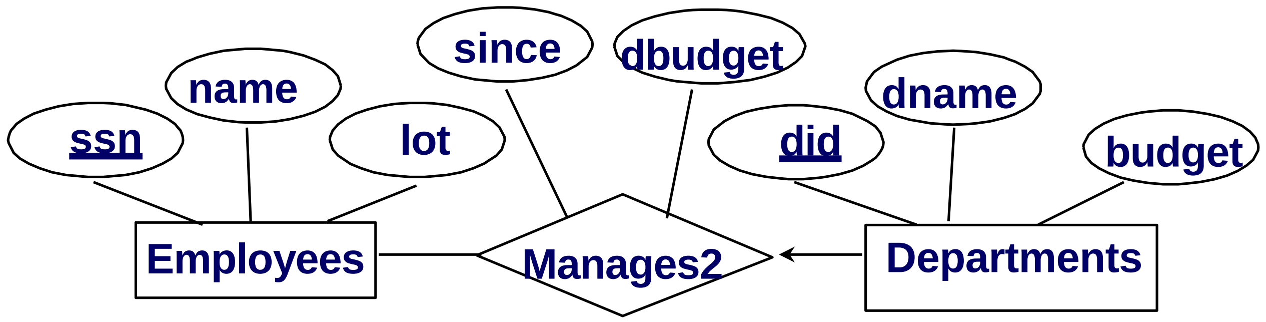


- Works_In4 does not allow an employee to work in a department for two or more periods.
- Similar to the problem of wanting to record several addresses for an employee: We want to record *several values of the descriptive attributes for each instance of this relationship.*
- *Solution?*

Entity vs.Attribute (Cont.)

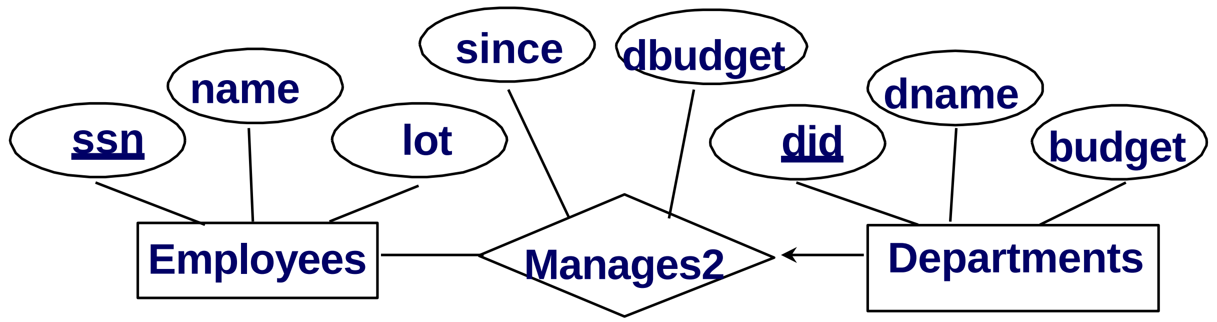


Entity vs. Relationship

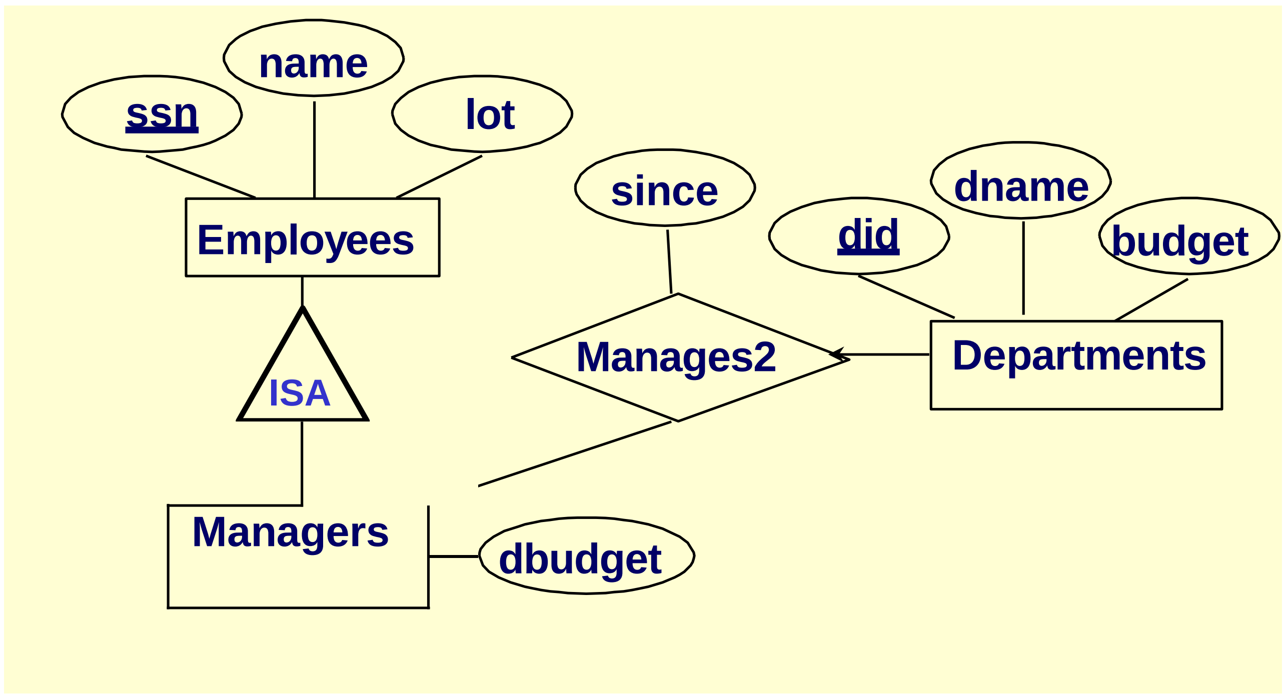


- This ER diagram OK if a manager gets a separate discretionary budget for each dept.
- What if a manager gets a discretionary budget that covers *all* managed depts?
  - Redundancy: *dbudget* stored for each dept managed by manager.
  - Misleading: Suggests *dbudget* associated with department-mgr combination.

Entity vs. Relationship

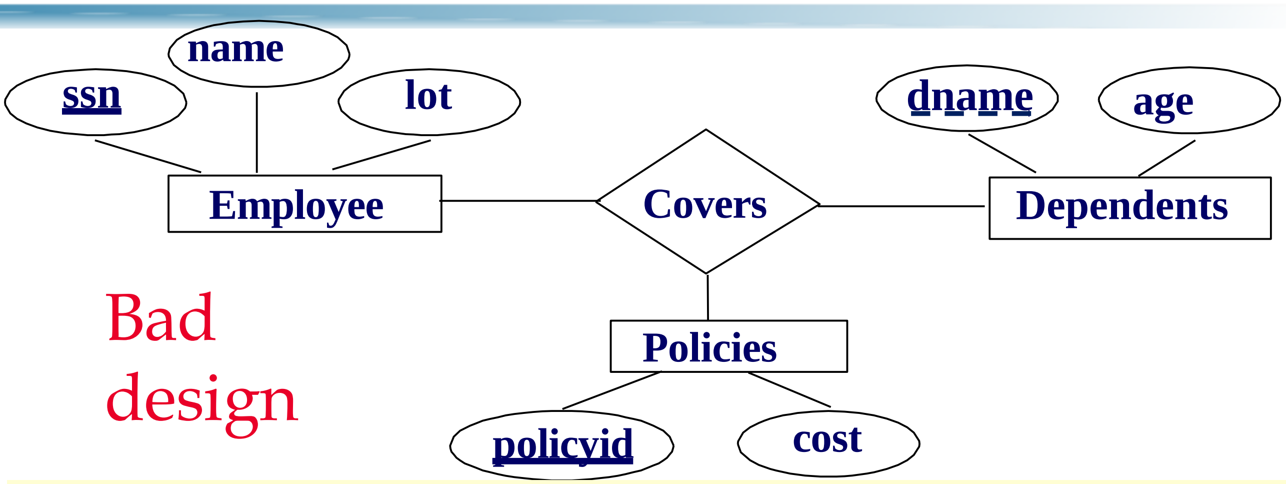


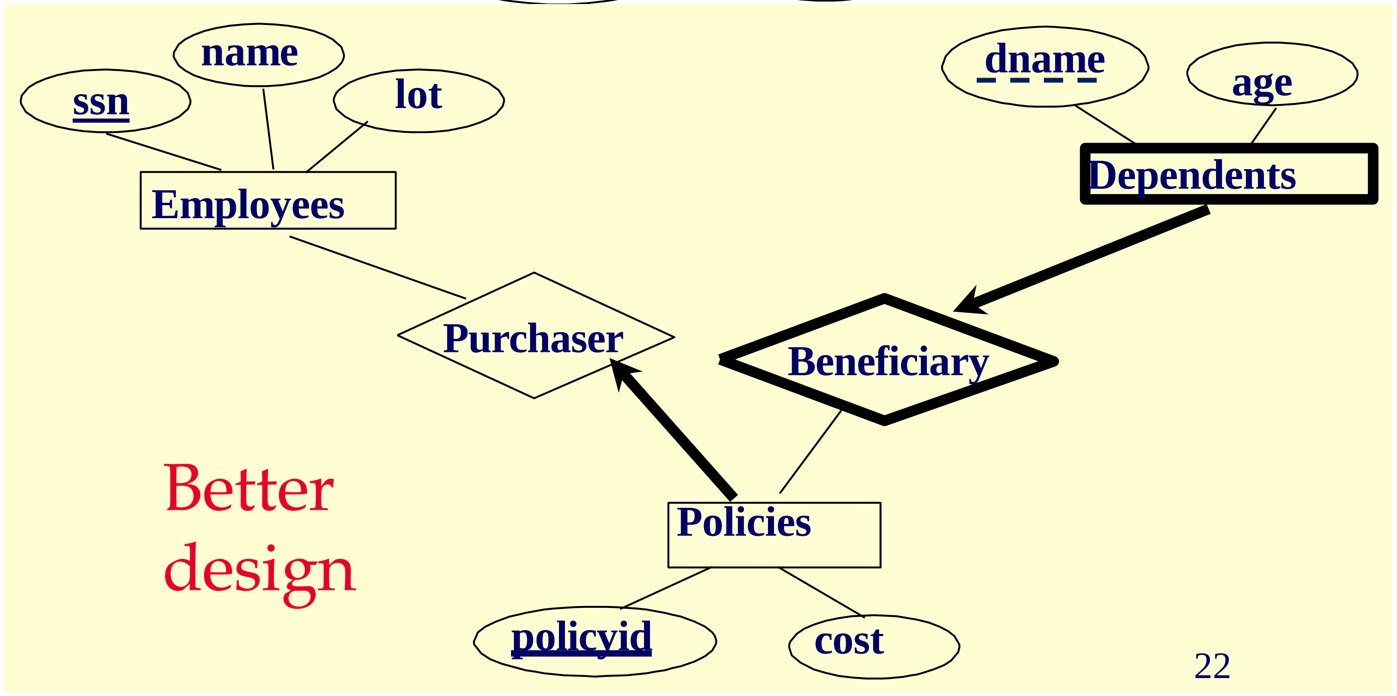
This fixes the Problem!



Binary vs. Ternary Relationships

- If each policy is owned by just 1 employee, and each dependent is tied to the covering policy, first diagram is inaccurate.
- What are the additional constraints in the 2nd diagram





Binary vs. Ternary Relationships (Cont.)

- Previous example illustrated a case when two binary relationships were better than one ternary relationship.
- An example in the other direction: a ternary relation Contracts relates entity sets Parts, Departments and Suppliers, and has descriptive attribute *qty*. No combination of binary relationships is an adequate substitute:
  - S "can-supply" P, D "needs" P, and D "deals-with" S does not imply that D has agreed to buy P from S.
  - How do we record *qty*?

Summary of Conceptual Design

- *Conceptual design* follows *requirements analysis*
  - Yields a high-level description of data to be stored
- ER model is popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications
- Basic constructs: *entities*, *relationships*, and *attributes* (of entities and relationships).
- Constraints can be expressed

Summary of ER (Cont.)

- Several kinds of integrity constraints can be expressed in the ER model: *key constraints*, *participation constraints*, and *overlap/covering constraints* for ISA hierarchies.
- Some constraints (notably, *functional dependencies*) cannot be expressed in the ER model.
  - Constraints play an important role in determining the best database design for an enterprise.
- ER design is *subjective*. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, whether or not to use ISA hierarchies, etc.
- To ensure good database design, resulting relational schema should be analyzed and refined further.