Entity-Relationship Model

E/R Diagrams

Converting E/R Diagrams to Relations

Jeff Ullman

(edited Dirk Van Gucht)

Purpose of E/R Model

- The E/R model allows us to sketch database schema designs.
  - Includes some constraints, but not operations.
- Designs are pictures called *entity-relationship diagrams*.
- Later: convert E/R designs to relational DB designs.

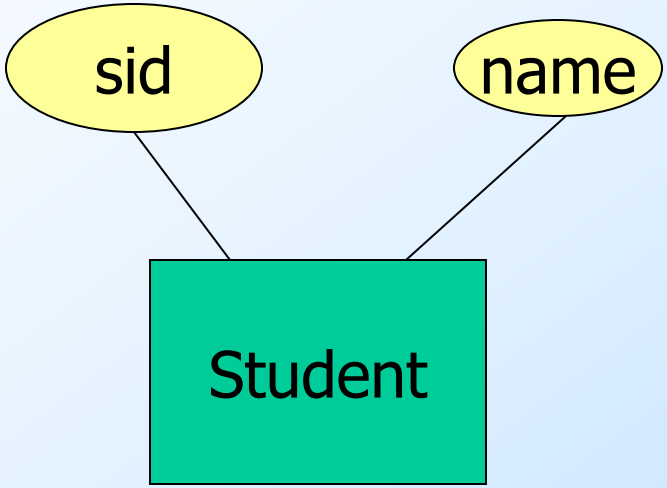
Entities and Entity Sets

- *Entity* = "thing" or object.
- *Entity set* = collection of similar entities.
  - Similar to a class in object-oriented languages.
- *Attribute* = property of an entity
  - Attributes are simple values, e.g. integers or character strings, not structs, sets, etc.

E/R Diagrams

- In an entity-relationship diagram:
  - Entity = rectangle.
  - Attribute = oval, with a line to the rectangle representing the entity

Example:

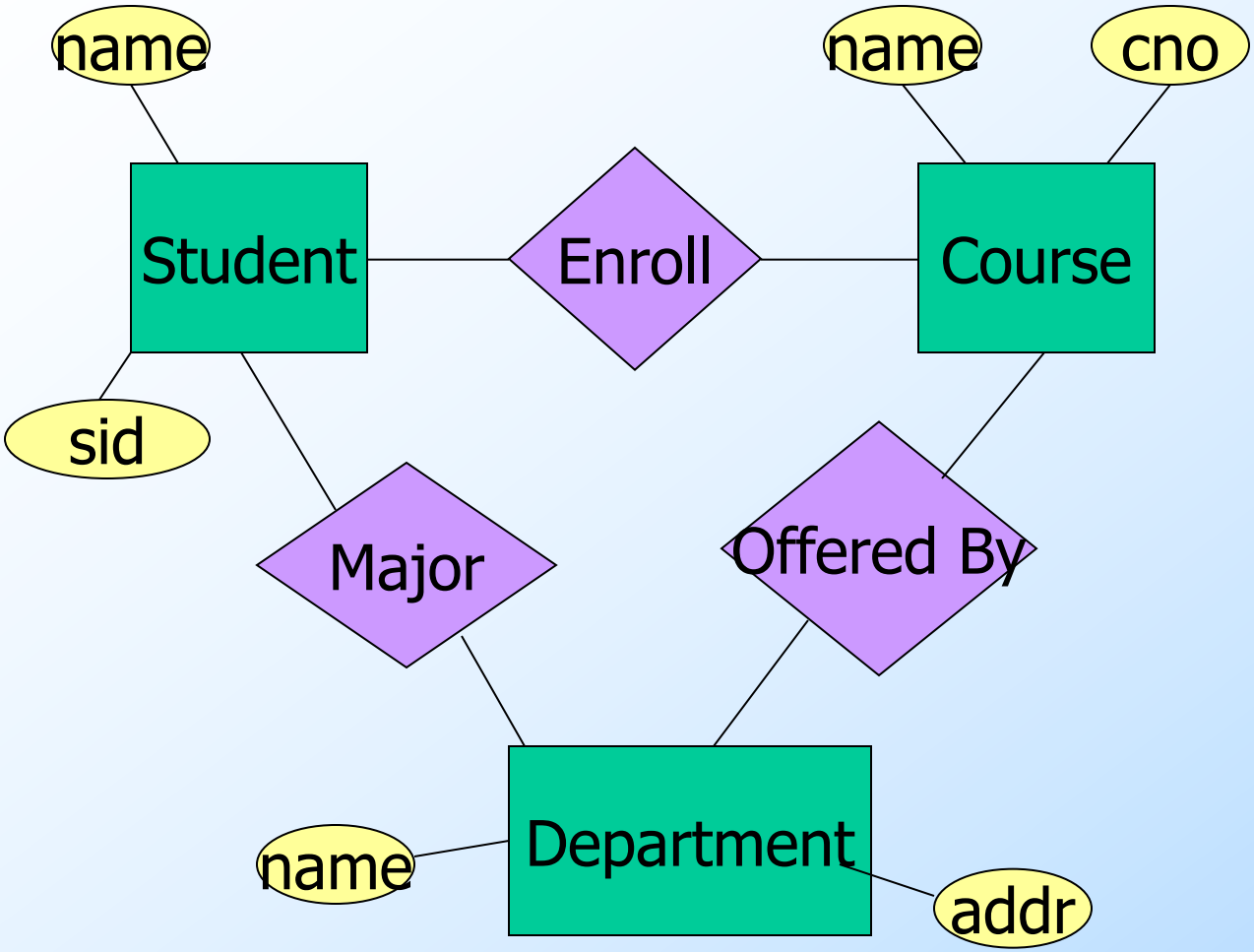


- Entity Student has two attributes, sid and name
- Each Student entity has values for these two attributes, e.g. (s1, Anna)

Relationships

- A relationship connects two or more entities
- It is represented by a diamond, with lines to each of the entities involved

Example: Relationships



Relationship Set

- The current "value" of an entity set is the set of entities that belong to it
  - Example: the set of all students in our database
- The "value" of a relationship is a *relationship set*, a set of tuples with one component for each related entity set

Example: Relationship Set

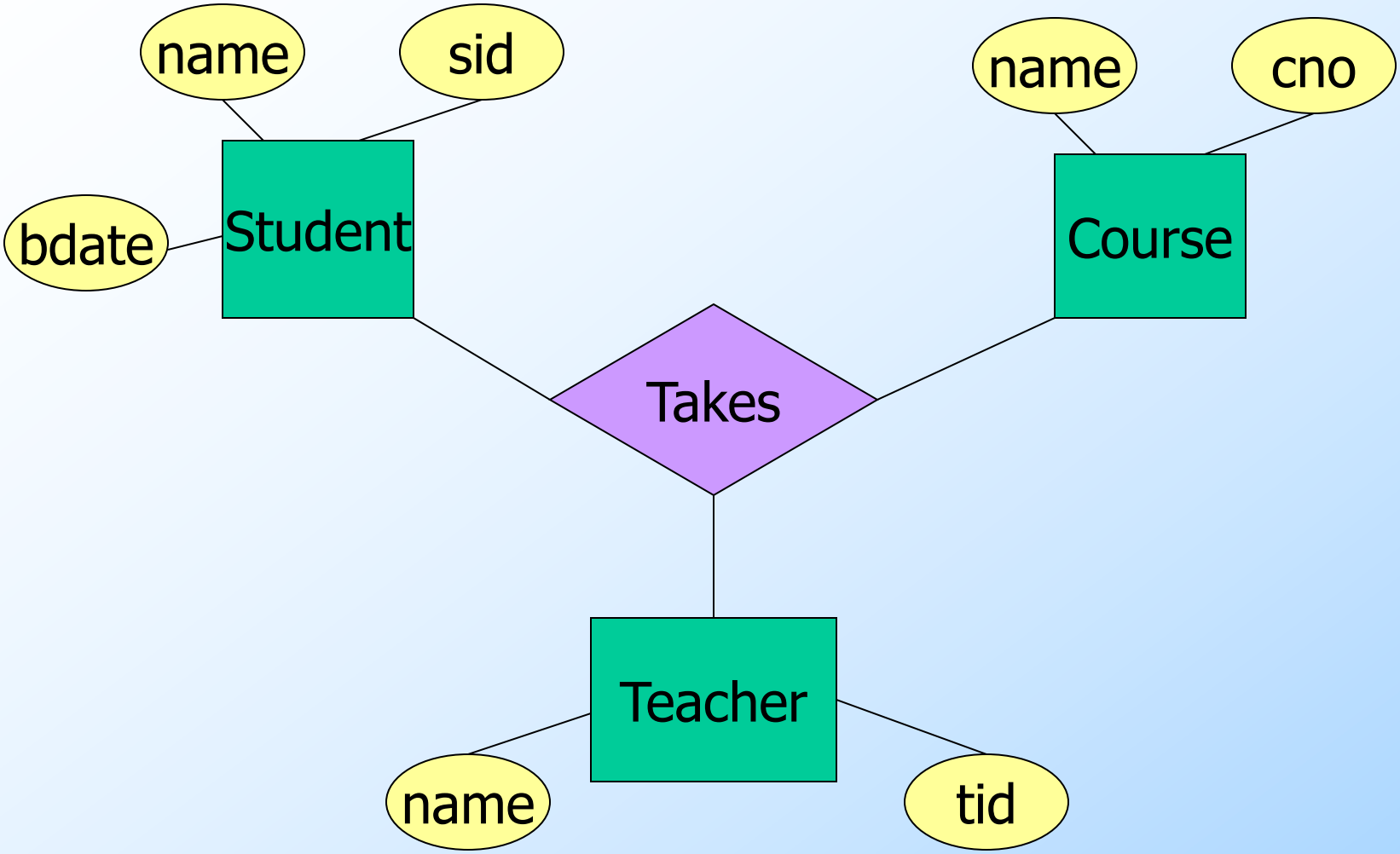
- For the relationship Major, we might have a relationship set like:

Student	Department
s1	CS
s1	Math
s2	Chemistry
s3	Math
s3	Physics

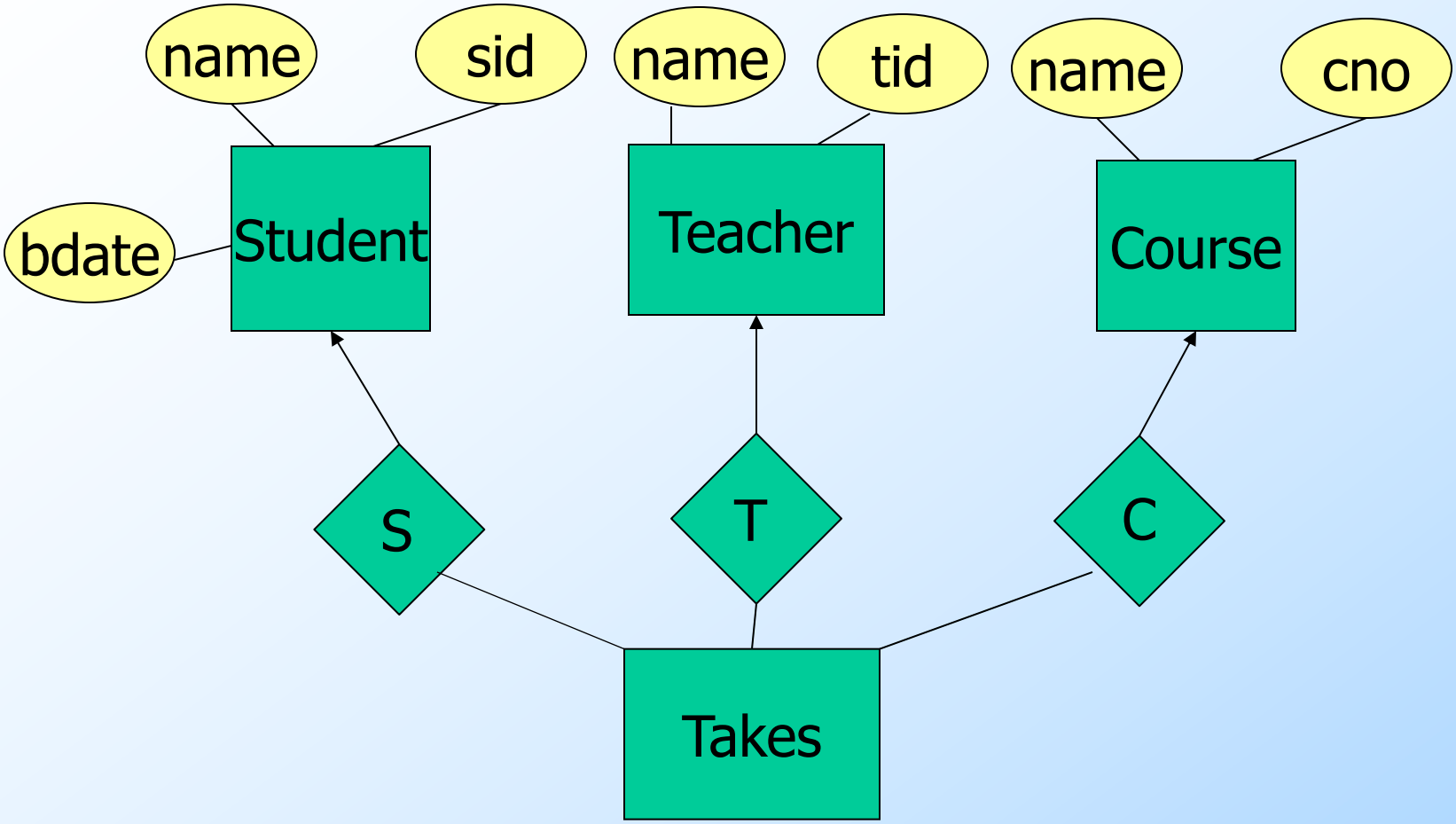
Multiway Relationships

- Sometimes, we need a relationship that connects more than two entities
- Suppose that student takes a course taught by a teacher (s,c,t)
- Three binary relationships do not allow us to precisely capture this relationship between a student, course, and teacher.
  - But a 3-way relationship would.

Example: 3-Way Relationship



Example: Representation with 3 many-to-one relationships



A Typical Relationship Set

Student	Course	Teacher
s1	c1	t1
s1	c2	t2
s2	c2	t1

Can not be captured with the three binary relationship sets

Student	Course
s1	c1
s1	c2
s2	c2

Student	Teacher
s1	t1
s1	t2
s2	t1

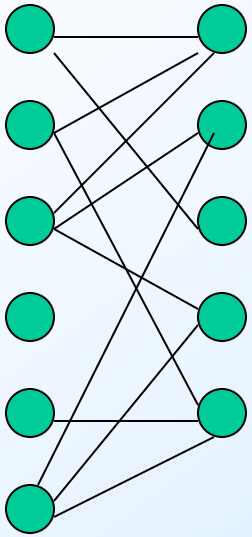
Course	Teacher
c1	t1
c2	t2
c2	t1

Notice how the triple (s1,c2,t1) is possible according to the 3 binary relations, but this tuple is not in the ternary relationship

Many-Many Relationships

- Focus: binary relationships, such as Enroll between Students and Courses
- In a *many-many* relationship, an entity of either set can be connected to many entities of the other set
  - E.g., a student may enroll in many courses; a course may enroll many students.

In Pictures:

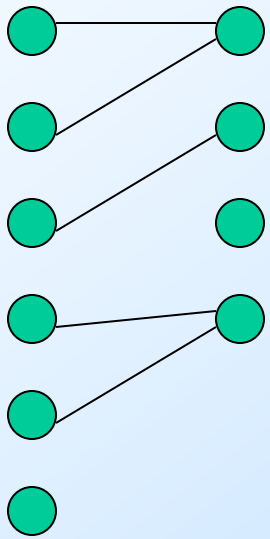


many-many

Many-One Relationships

- Some binary relationships are *many - one* from one entity set to another
- Each entity of the first set is connected to at most one entity of the second set
- But an entity of the second set can be connected to zero, one, or many entities of the first set
- A many-one relationship is a function

In Pictures:



many-one

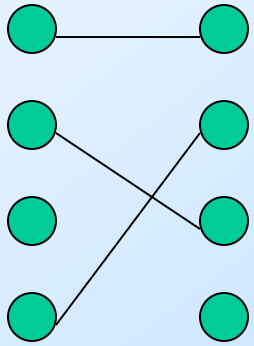
Example: Many-One Relationship

- Offered By, from Courses to Departments is many-one
- A course is offered by at most one department
- But a department can offer any number of students, including zero

One-One Relationships

- In a *one-one* relationship, each entity of either entity set is related to at most one entity of the other set
- Example: Relationship Chair Person between entity sets Teachers and Department
  - A department cannot be chaired by more than one teacher, and no teacher can chair more than one department

In Pictures:

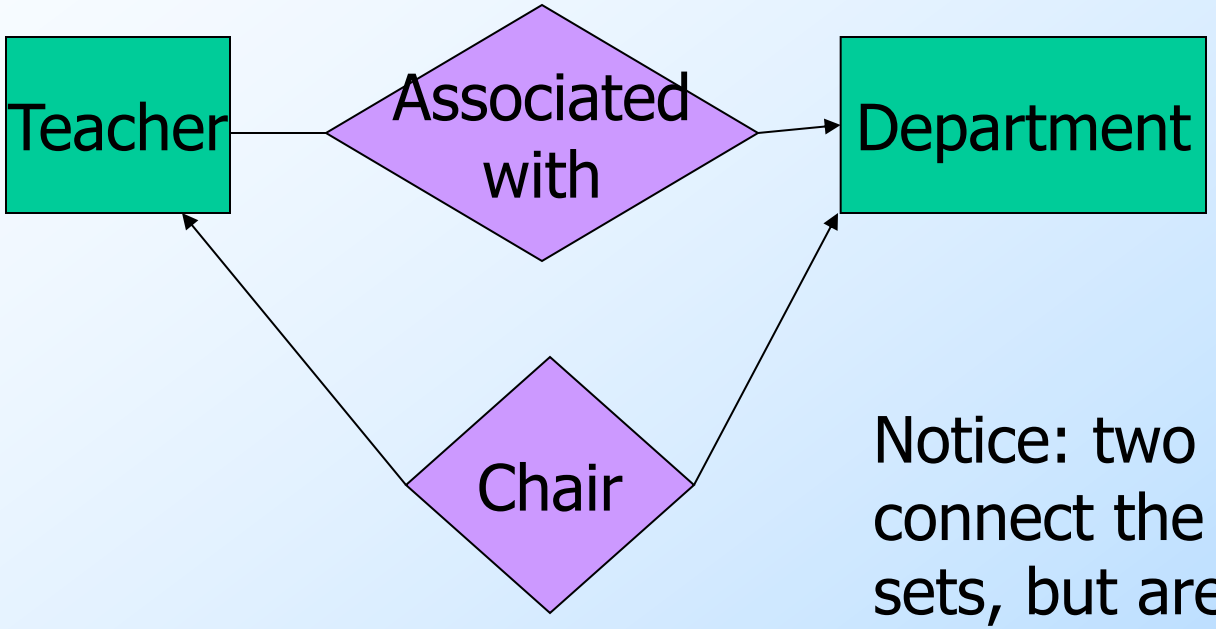


one-one

Representing "Multiplicity"

- Show a many-one relationship by an arrow entering the "one" side
  - Remember: Like a function
- Show a one-one relationship by arrows entering both entity sets
- Rounded arrow = "exactly one," i.e., each entity of the first set is related to exactly one entity of the target set

Example: Many-One Relationship

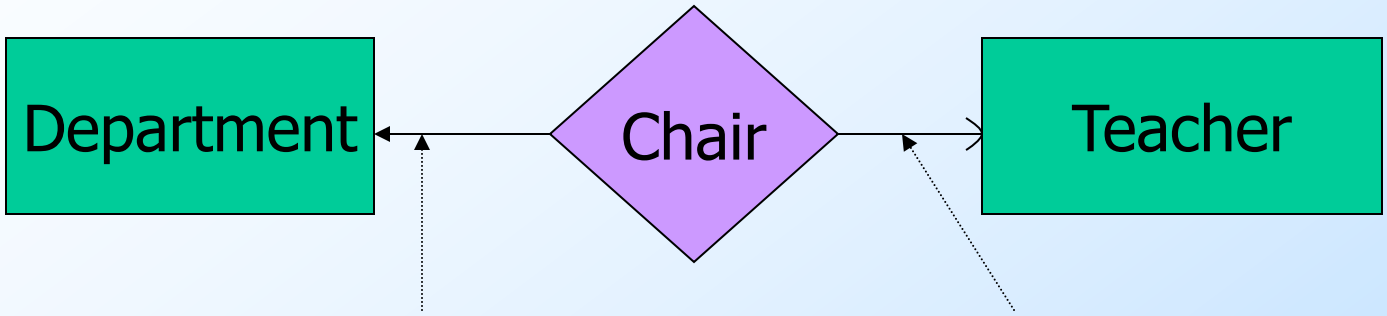


Notice: two relationships connect the same entity sets, but are different.

Example: One-One Relationship

- Consider Chair between Teachers and Departments.
- Some teachers are not the chair of any department, so a rounded arrow to department would be inappropriate
- But a department must have chair person.

In the E/R Diagram



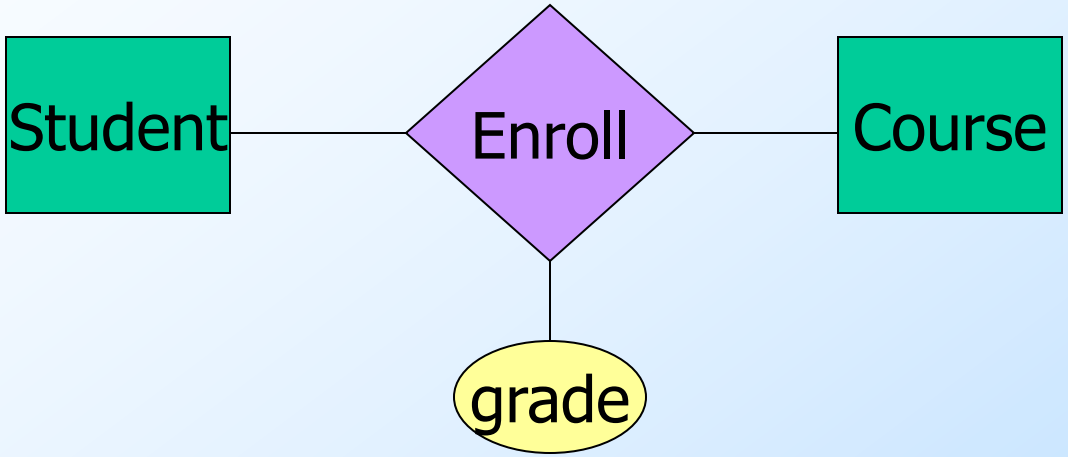
A department has exactly one chair.

A teacher is the chair of at most one (zero or one) department.

Attributes on Relationships

- Sometimes it is useful to attach an attribute to a relationship
- Think of this attribute as a property of each tuple (relationship) in the relationship set

Example: Attribute on Relationship



Grade is a function of both the student and the course, not of one alone

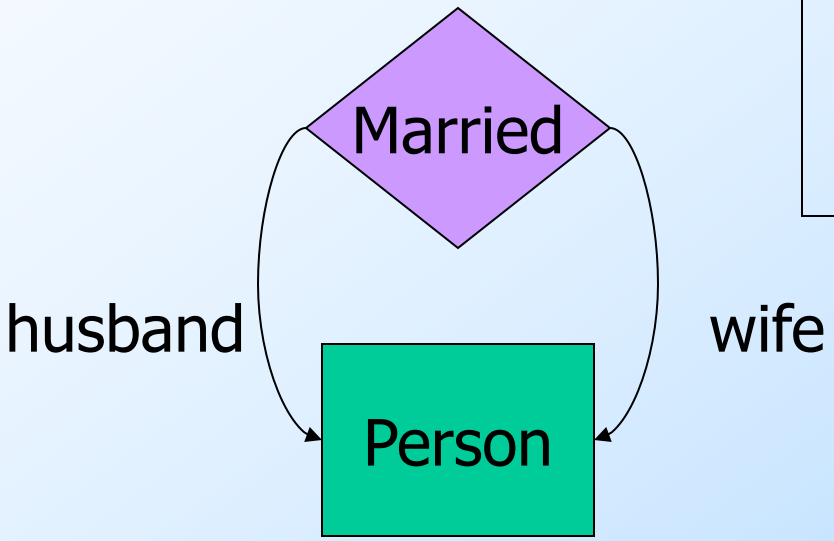
Roles

- Sometimes an entity set appears more than once in a relationship
- Label the edges between the relationship and the entity set with names called *roles*

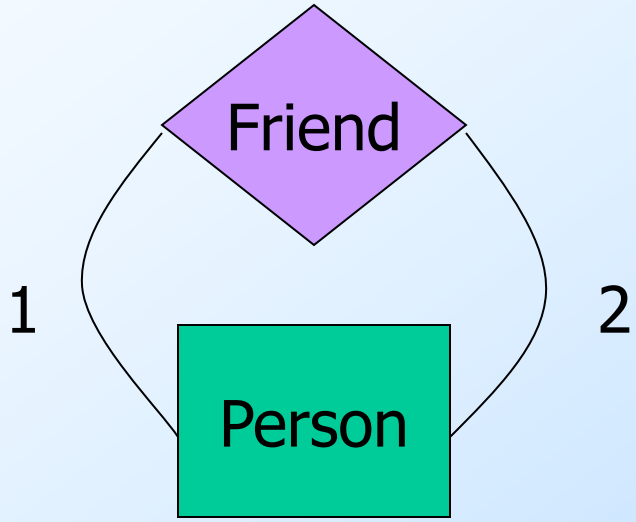
Example: Roles

Relationship Set

Husband	Wife
Bob	Ann
Joe	Sue
…	…



Example: Roles



Relationship Set

Friend1	Friend2
Bob	Ann
Joe	Sue
Ann	Bob
Joe	Moe
…	…

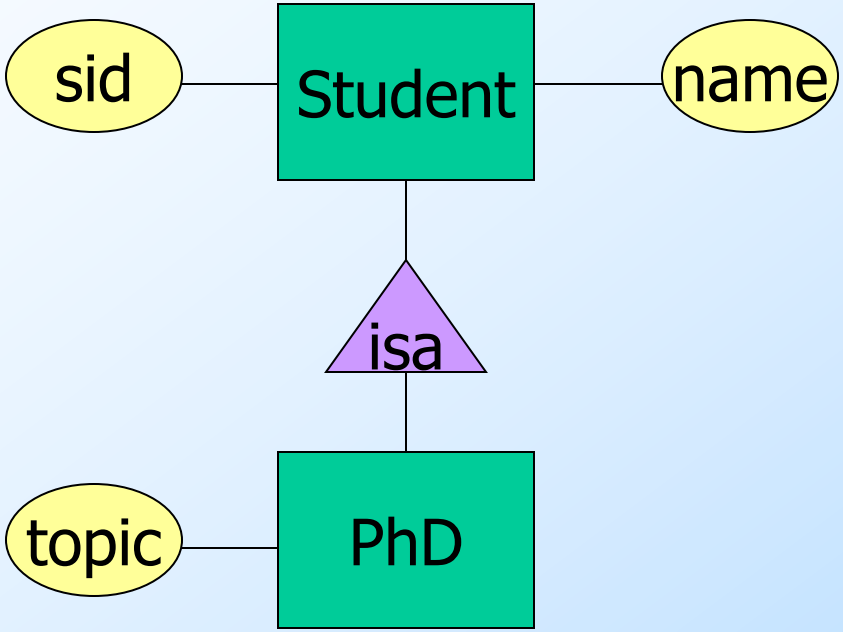
Subclasses

- *Subclass* = special case = fewer entities = more properties = more relationships
- Example: PhD students are a kind of student.
  - Not every student is a PhD student, but some are.
  - Let us suppose that in addition to all these *properties* (attributes and relationships), a PhD student also has the attribute topic

Subclasses in E/R Diagrams

- Assume subclasses form a tree.
  - I.e., no multiple inheritance.
- Isa triangles indicate the subclass relationship.
  - Point to the superclass.

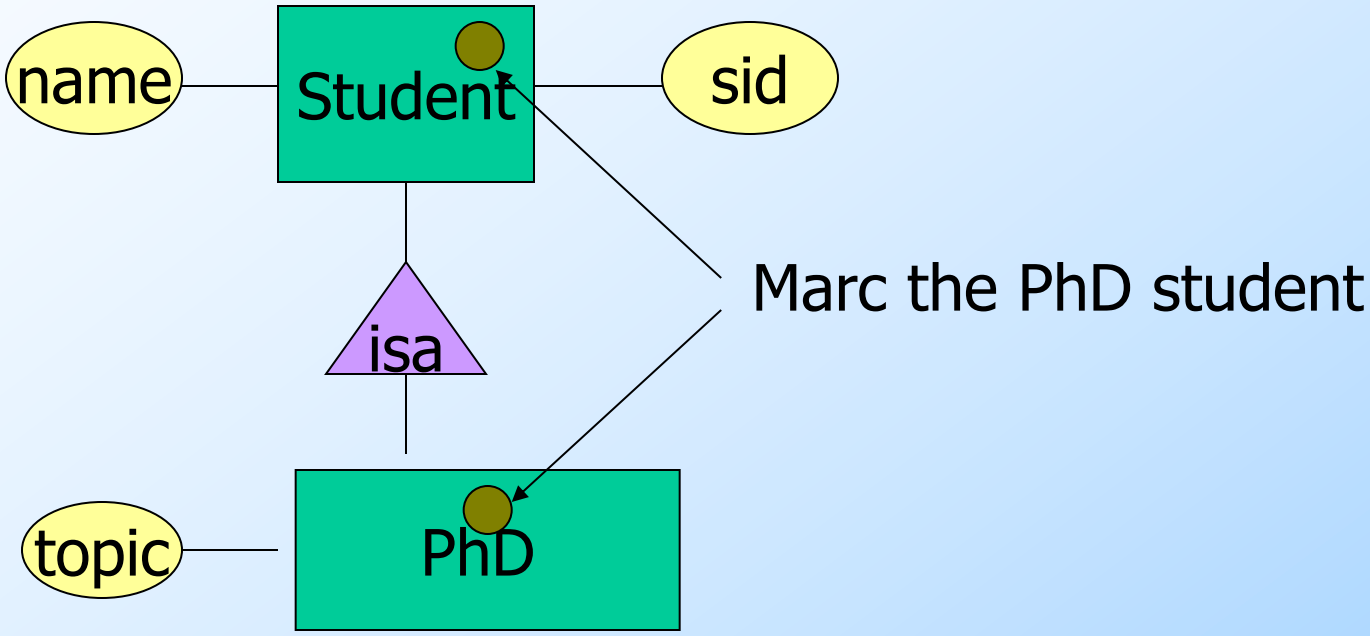
Example: Subclasses



E/R Vs. Object-Oriented Subclasses

- In OO, objects are in one class only.
  - Subclasses inherit from superclasses.
- In contrast, E/R entities have *representatives* in all subclasses to which they belong.
  - Rule: if entity *e* is represented in a subclass, then *e* is represented in the superclass (and recursively up the tree).

Example: Representatives of Entities



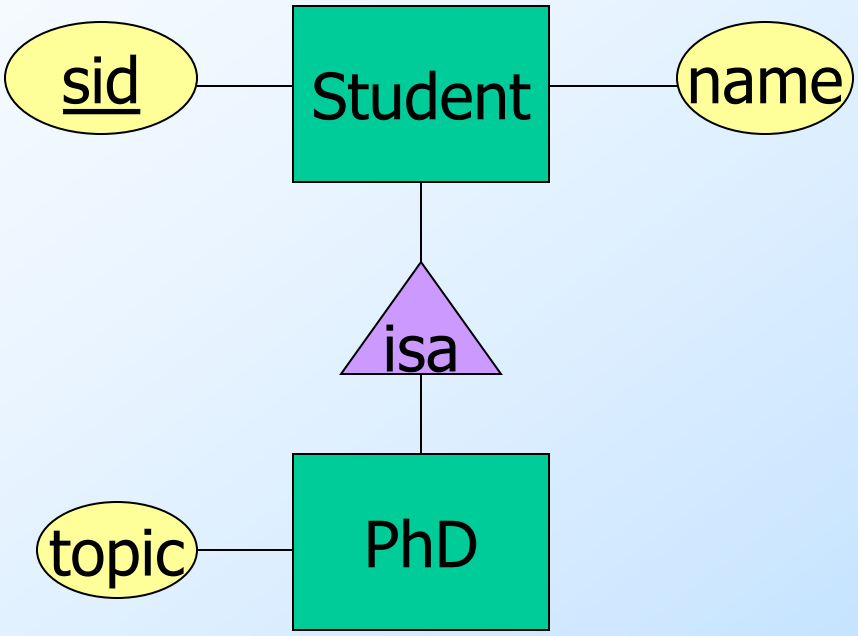
Keys

- A *key* is a set of attributes for one entity set such that no two entities in this set agree on all the attributes of the key
  - It is allowed for two entities to agree on some, but not all, of the key attributes.
- We must designate a key for every entity set

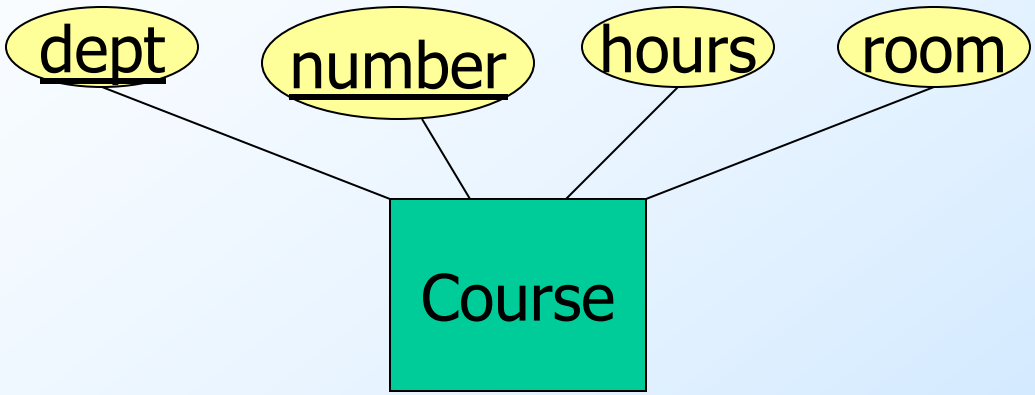
Keys in E/R Diagrams

- Underline the key attribute(s)
- In an Isa hierarchy, only the root entity set has a key, and it must serve as the key for all entities in the hierarchy

Example: name is Key for Student



Example: a Multi-attribute Key



- Note that hours and room could also serve as a key, but we must select only one key.

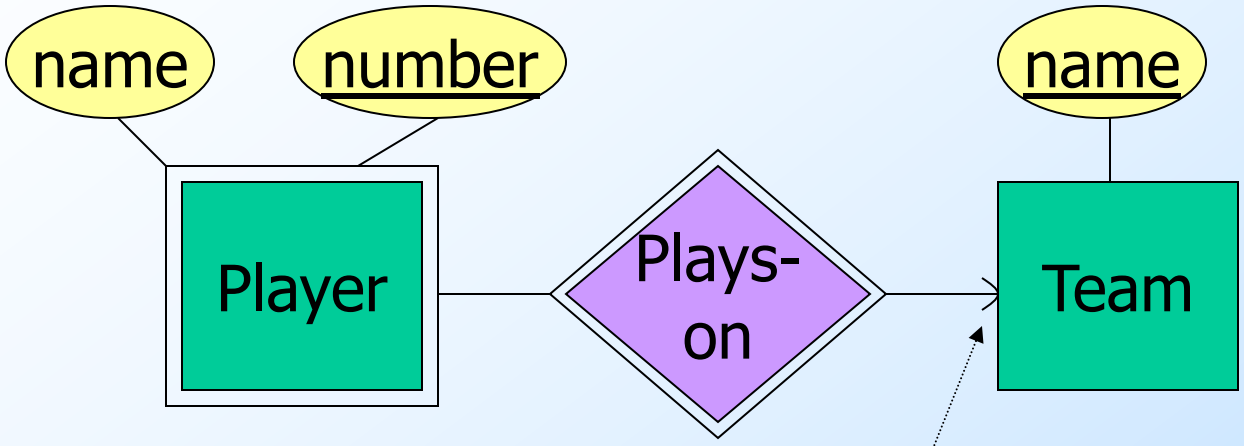
Weak Entity Sets

- Occasionally, entities of an entity set need "help" to identify them uniquely.
- Entity set *E* is said to be *weak* if in order to identify entities of *E* uniquely, we need to follow one or more many-one relationships from *E* and include the key of the related entities from the connected entity sets.

Example: Weak Entity Set

- name is almost a key for football players, but there might be two with the same name
- number is certainly not a key, since players on two teams could have the same number.
- But number, together with the team name related to the player by Plays-on should be unique.

In E/R Diagrams



Note: must be rounded because each player needs a team to help with the key.

- Double diamond for *supporting* many-one relationship.
- Double rectangle for the weak entity set.

Weak Entity-Set Rules

- A weak entity set has one or more many-one relationships to other (supporting) entity sets
  - Not every many-one relationship from a weak entity set need be supporting
  - But supporting relationships must have a rounded arrow (entity at the "one" end is guaranteed).

Weak Entity-Set Rules – (2)

- The key for a weak entity set is its own underlined attributes and the keys for the supporting entity sets.
  - E.g., (player) number and (team) name is a key for Players in the previous example.

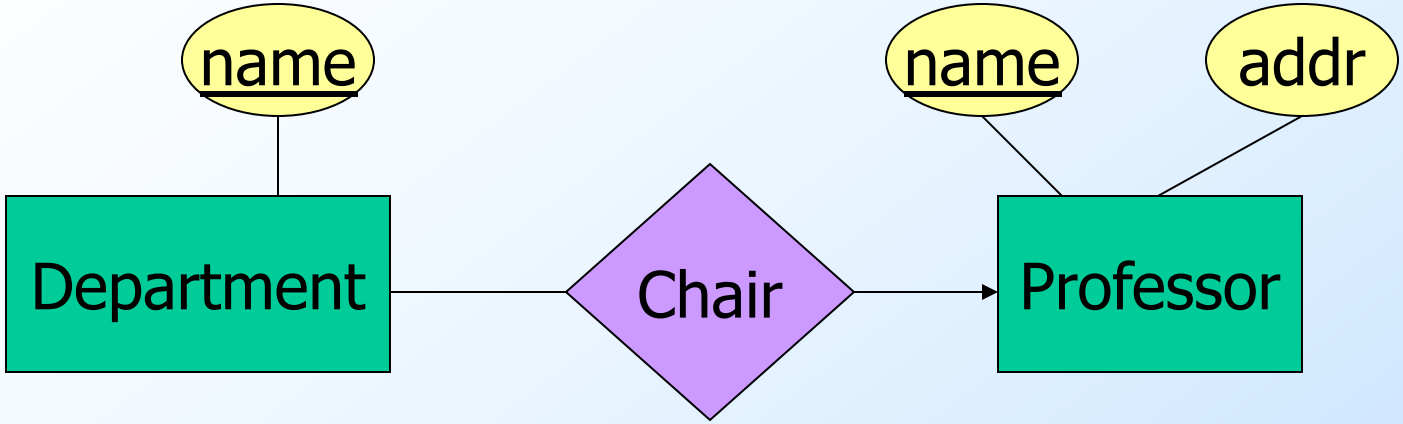
Design Techniques

1. Avoid redundancy
2. Limit the use of weak entity sets
3. Don’t use an entity set when an attribute will do

Avoiding Redundancy

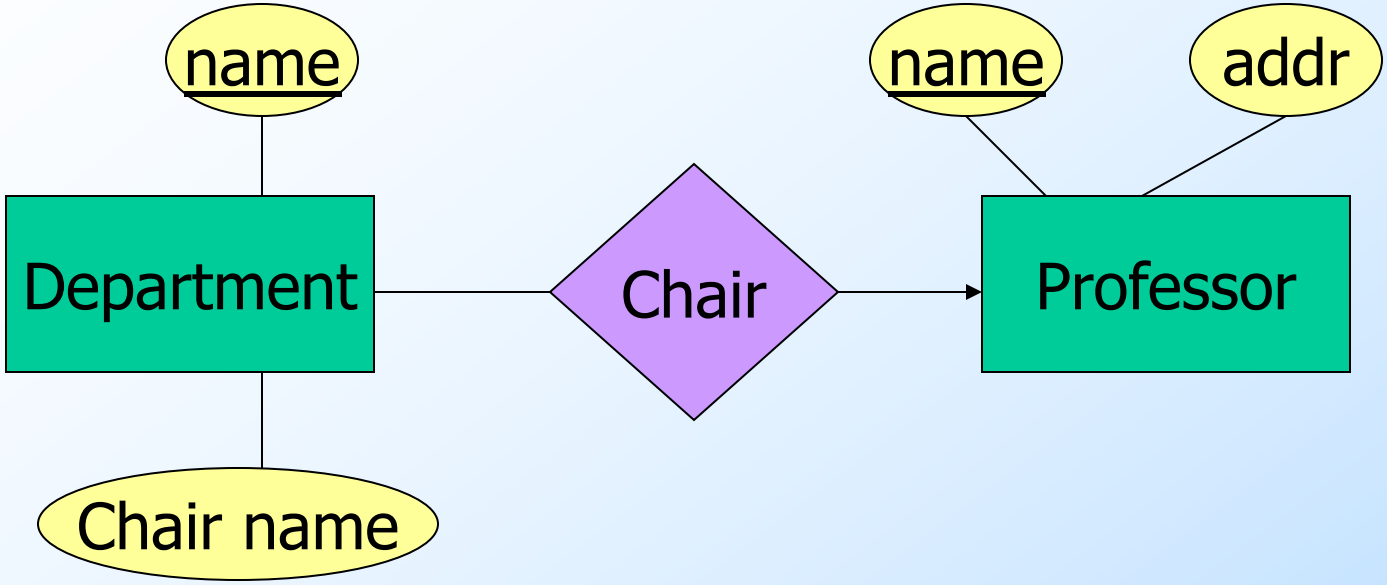
- *Redundancy* = saying the same thing in two (or more) different ways
- Wastes space and (more importantly) encourages inconsistency
  - Two representations of the same fact become inconsistent if we change one and forget to change the other.

Example: Good



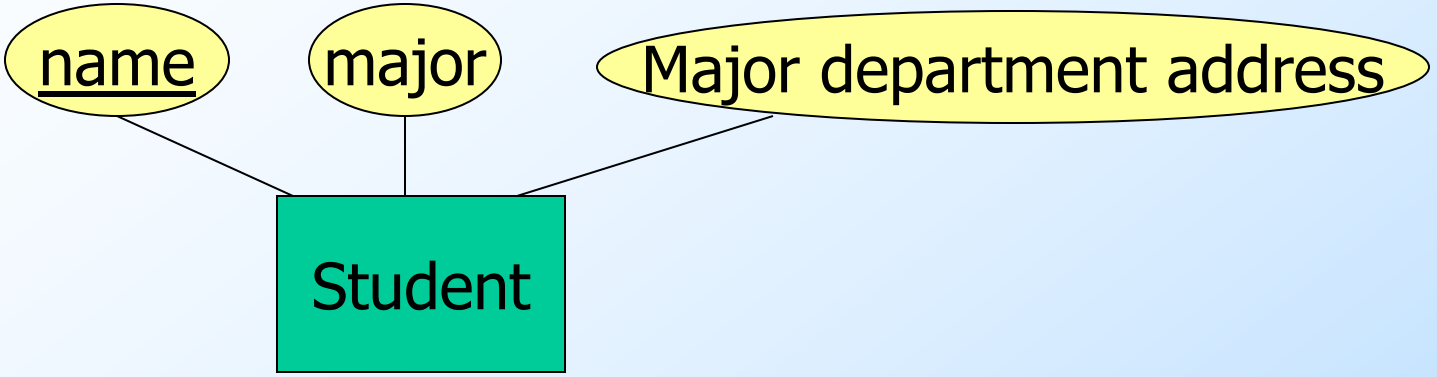
This design gives the address of each professor exactly once

Example: Bad



This design states the chair of a department twice: as an attribute and as a related entity.

Example: Bad



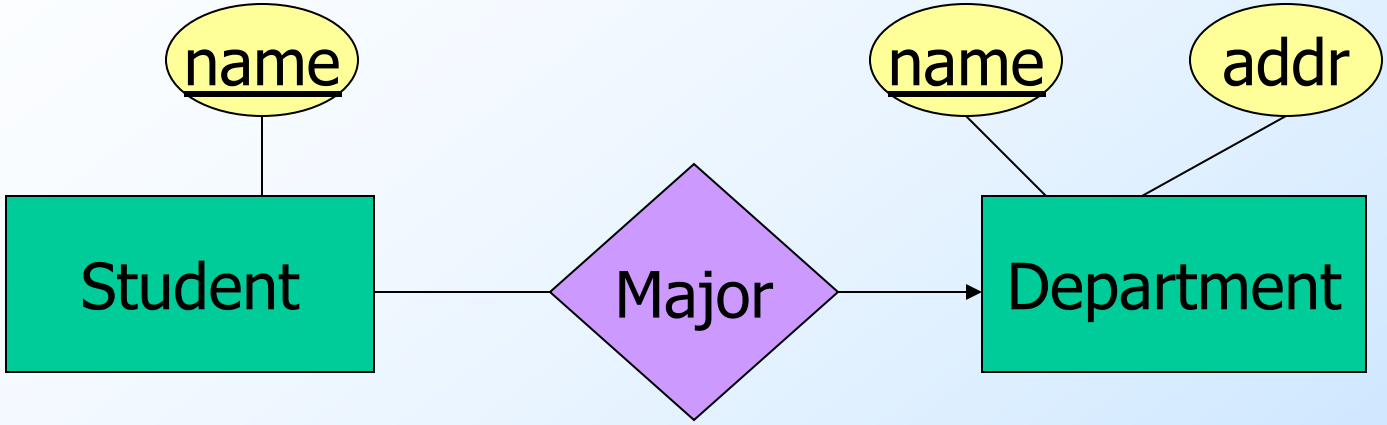
This design repeats the major's department address once for each student and loses the major department address if there are temporarily no students for a major

Entity Sets Versus Attributes

- An entity set should satisfy at least one of the following conditions:
  - It is more than the name of something; it has at least one nonkey attribute.

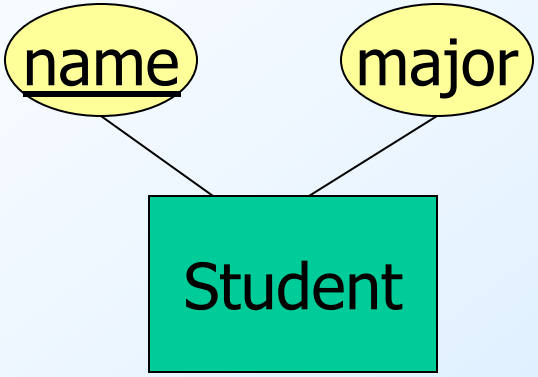
    or
  - It is the "many" in a many-one or many-many relationship.

Example: Good



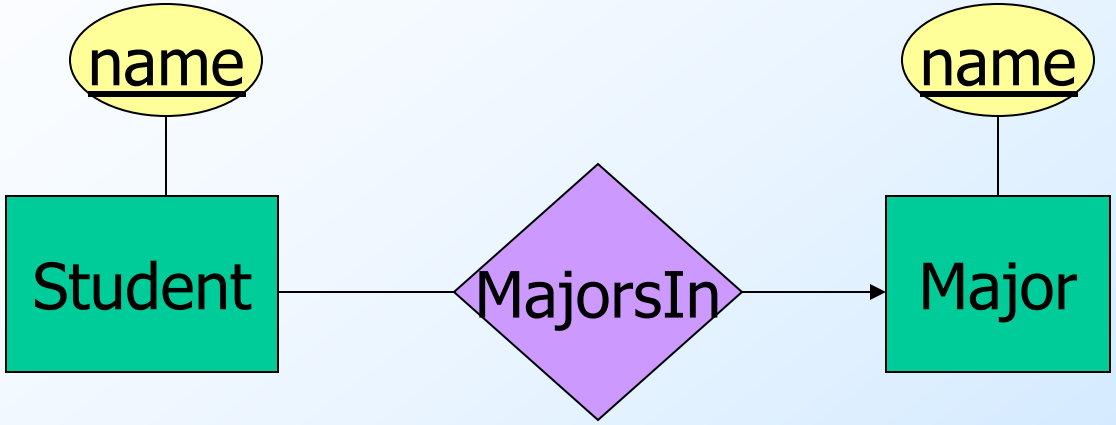
- Department deserves to be an entity set because of the nonkey attribute addr.
- Student deserves to be an entity set because it is the "many" of the many-one relationship Major.

Example: Good



There is no need to make the major an entity set, because we record nothing about majors besides their name

Example: Bad



Since the major is nothing but a name, and is not at the "many" end of any relationship, it should not be an entity set.

Don’t Overuse Weak Entity Sets

- Beginning database designers often doubt that anything could be a key by itself
  - They make all entity sets weak, supported by all other entity sets to which they are linked.
- In reality, we usually create unique ID’s for entity sets
  - Examples include social-security numbers, automobile VIN’s etc.

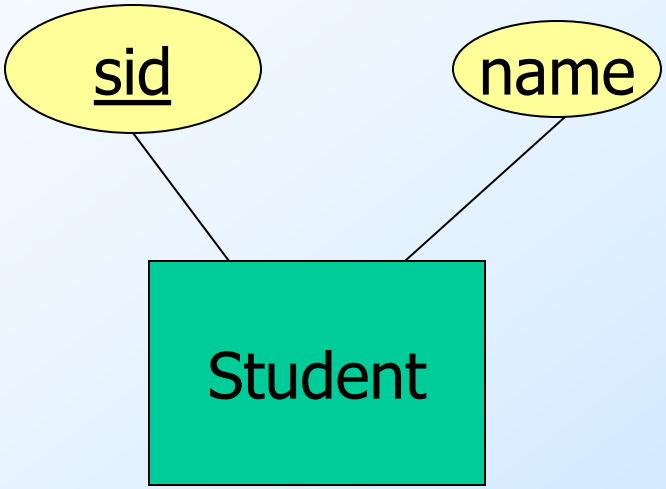
When Do We Need Weak Entity Sets?

- The usual reason is that there is no global authority capable of creating unique ID's
- Example: it is unlikely that there could be an agreement to assign unique player numbers across all football teams in the world

From E/R Diagrams to Relations

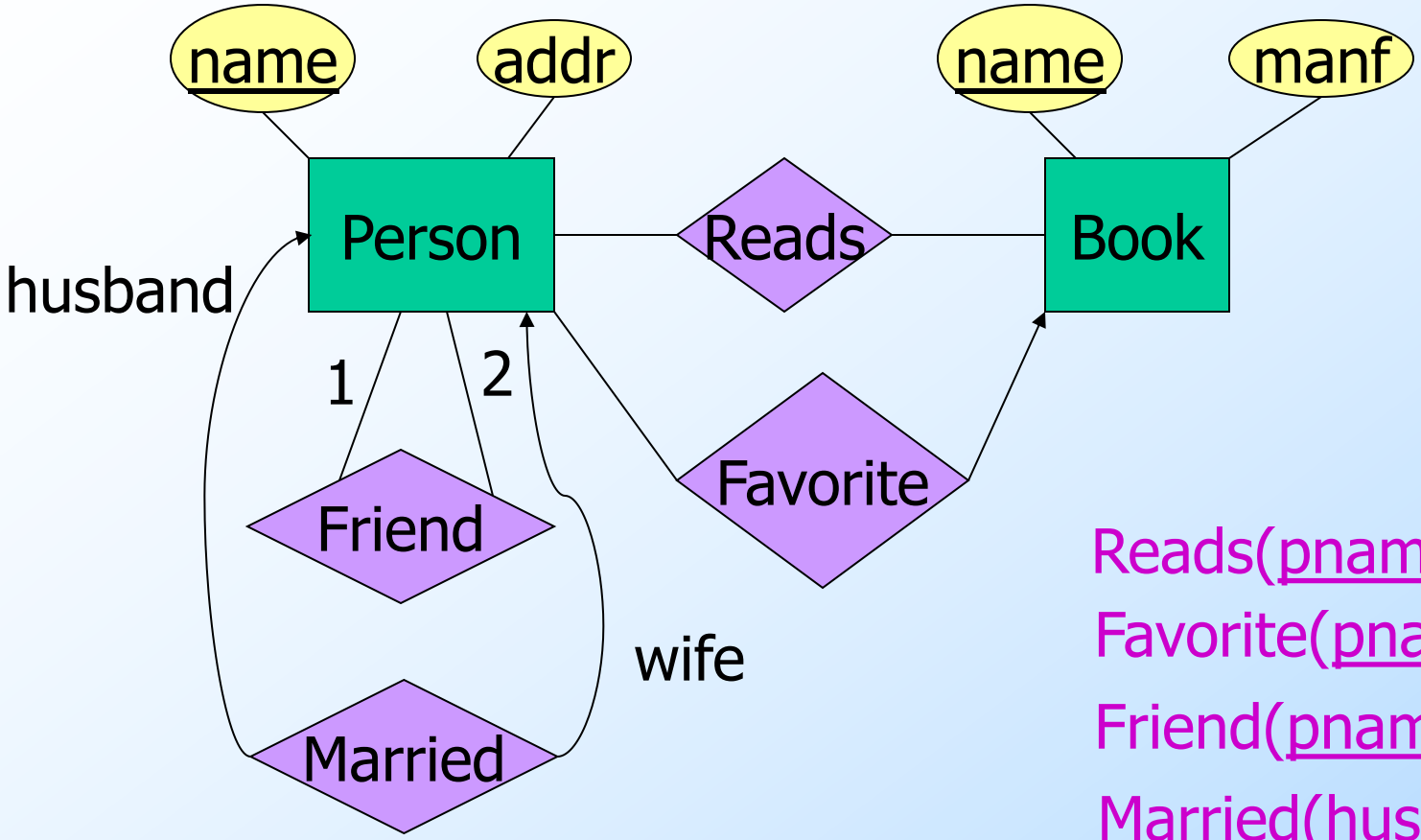
- Entity set -> relation.
  - Attributes -> attributes.
- Relationships -> relations whose attributes are only:
  - The keys of the connected entity sets.
  - Attributes of the relationship itself.

Entity Set -> Relation



Relation: Student(sid, name)

Relationship -> Relation



Reads(pname, bname)

Favorite(pname, bname)

Friend(pname1, pname2)

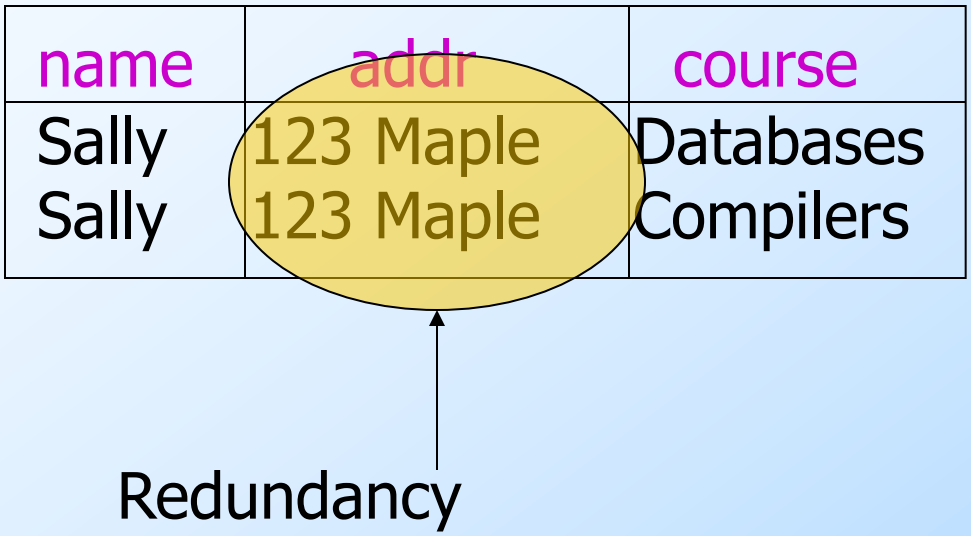
Married(husband, wife)

Combining Relations

- OK to combine into one relation:
  1. The relation for an entity-set *E*
  2. The relations for many-one relationships of which *E* is the "many."
- Example: Person(pname, addr) and Favorite(pname, bname) combine to make Person1(pname, addr, favBook).

Risk with Many-Many Relationships

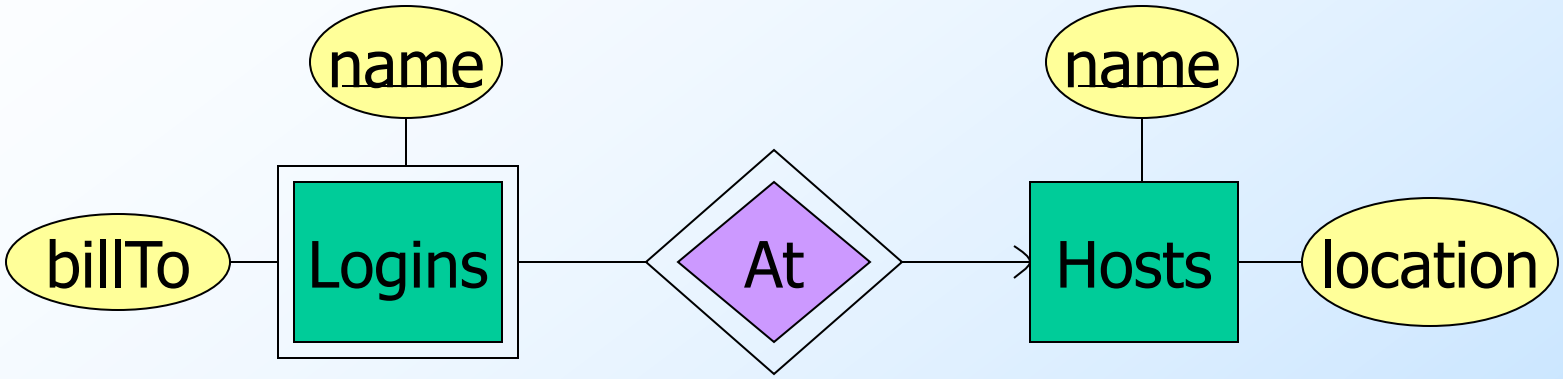
- Combining Student with Course would be a mistake. It leads to redundancy, as:



Handling Weak Entity Sets

- Relation for a weak entity set must include attributes for its complete key (including those belonging to other entity sets), as well as its own, nonkey attributes.

Example: Weak Entity Set -> Relation

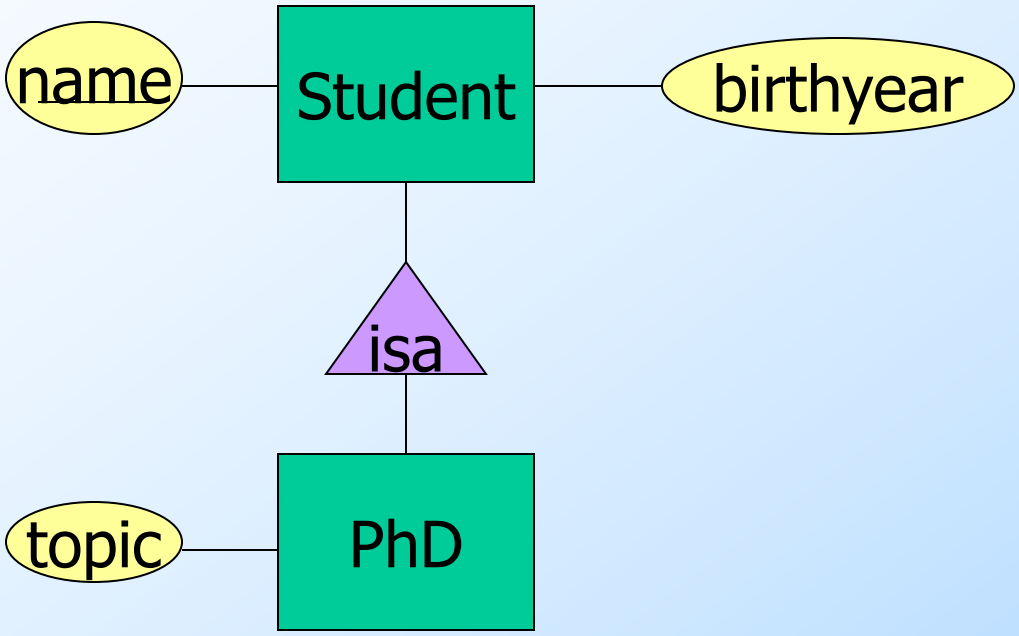


Hosts(hostName, location)
Logins(loginName, hostName, billTo)

Subclasses: Three Approaches

1. *Object-oriented* : One relation per subset of subclasses, with all relevant attributes.
2. *Use nulls* : One relation; entities have NULL in attributes that don't belong to them.
3. *E/R style* : One relation for each subclass:
   - Key attribute(s).
   - Attributes of that subclass.

Example: Subclass -> Relations



Object-Oriented

name	birthyear
Ann	1999

Student

name	birthyear	topic
Ellen	1995	PL

PhD

Good for queries like "find the topics of PhD's born after 1990"

E/R Style

name	birthyear
Ann	1999
Ellen	1995

Student

name	topic
Ellen	PL

PhD

Good for queries like "find all student (including PhDs) born in 1995."

Using Nulls

name	birthyear	topic
Ann	1999	NULL
Ellen	1995	PL

Student

Saves space unless there are *lots* of attributes that are usually NULL.

In SQL (relationships)

- Consider student, course, enroll
- Maintain referential integrity

```
create table student (sid integer PRIMARY KEY,
                      name text);
create table course (cno integer PRIMARY KEY,
                     name text);
create table enroll (sid integer  REFERENCES student(sid),
                     cno integer REFERENCES course(cno),
                     grade VARCHAR(2),
                     primary key(sid,cno))
```

In SQL (subclasses) Object-Oriented style

- Consider PhD example

```
CREATE TABLE Student(sid PRIMARY KEY,
                     name text,
                     birthyear integer);

CREATE TABLE PhD(sid PRIMARY KEY,
                 name text,
                 birthyear integer,
                 topic integer);
```

- Notice that there is no foreign key in the PhD table.

In SQL (subclasses) E/R style

- Consider student, PhD

```
create table Student (sid integer PRIMARY KEY,
                      name text,
                      birthyear integer);

create table PhD (sid integer REFERENCES Student(sid),
                  topic text);
```

In SQL (subclasses) With NULLS

- Consider PhD example

```
CREATE TABLE Student (sid integer PRIMARY KEY,
                      name text,
                      birthyear integer,
                      phd_topic text);
```