E/R Models

Three Pieces of Course

- Database design
  - Modeling data
- Database programming
  - SQL (other languages)
  - Constructing applications
- Database implementation
  - Learning how the guts work

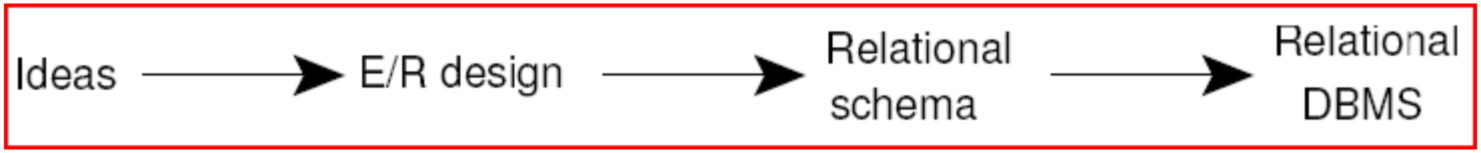
Why Learn About Database Modeling?

- The way in which data is stored is very important for subsequent access and manipulation by SQL.

- Properties of a good data model:
  - It is easy to write correct and easy to understand queries.
  - Minor changes in the problem domain do not change the schema.
  - Major changes in the problem domain can be handled without too much difficulty.
  - Can support efficient database access.

Purpose of the E/R Model

- The E/R model allows us to sketch the design of a database informally.
  - Represent different types of data and how they relate to each other
- Designs are drawings called *entity-relationship diagrams*.
- Fairly mechanical ways to convert E/R diagrams to real implementations like relational databases exist.

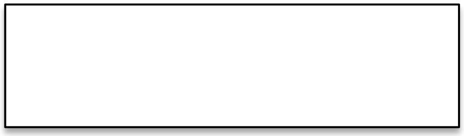


Purpose of E/R Model

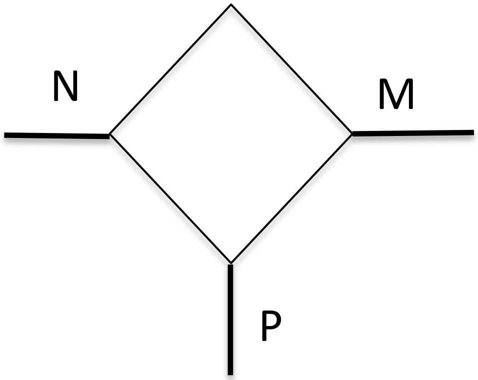
- When designing E/R diagrams,
  - – forget about relations/tables!
  - – only consider how to model the information you need to represent in your database.

Tools

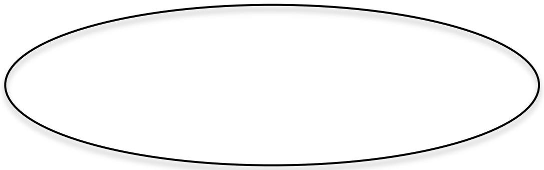
- Entities (‘entity sets’)



- Relationships (‘rel. sets’) and mapping constraints



- Attributes



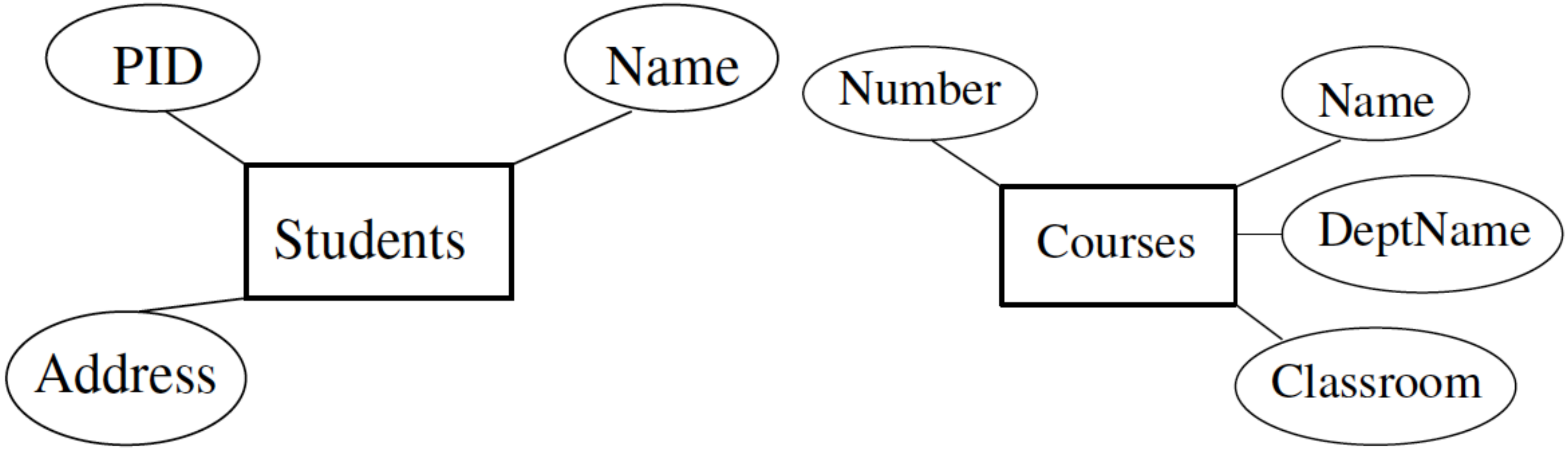
Entity Sets

- *Entity* = "thing" or "object instance" or "noun"
- *Entity set* = collection of similar entities.
  - Similar to a class in object-oriented languages.
- Attribute = property of an entity set.
  - Generally, all entities in a set have the same properties.
  - Our convention is to use ‘atomic attributes’ e.g. integers, character strings etc.

E/R Diagrams

- In an entity-relationship diagram, each entity set is represented by a rectangle.
- Each attribute of an entity set is represented by an oval, with a line to the rectangle representing its entity set.

Example: Entity Sets



Relationships

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

- Don’t confuse ‘Relationships’ with ‘Relations’!

Instance of an E/R Diagram

- E/R diagram describes a schema, not the DB content itself.
- However, we can visualize what the DB tuples might look like by thinking of an ***instance of the E/R diagram***:
  - – contains ***instances of*** entity sets and
  - – relationship sets.

Instance of an Entity Set

- For each entity set, an instance stores a specific set of entities
- Each entity is a tuple containing specific values for each attribute
- Example: Instance of an entity set for students.

(Binary) relationship sets

- Binary relation with entities E and F:
- Instance is a set of pairs of (e, f) where e is in E and f is in F
  - Instance need not relate every tuple in E with every tuple in F
  - Relationship set for R: the pairs of tuples (e, f) related by R
- Relationships sets are not tables or relations.
- (Conceptually) An instance of R is simply the ‘concatentation’ of the attribute lists for all pairs of tuples (e, f) in the relationship set for R

Attributes for a Relationship

- Question: What is Grade an attribute of?

Multiplicity of binary relationships

- Many-one from A to B: when each entity in A is connected to ***at most one*** entity in B.
- One-one: when a relationship is many-one from A to B and from B to A.
- Many-many: everything else.

Many-Many Relationships

- In a *many-many* relationship, an entity of either set can be connected to many entities of the other set.

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.

One-One Relationships

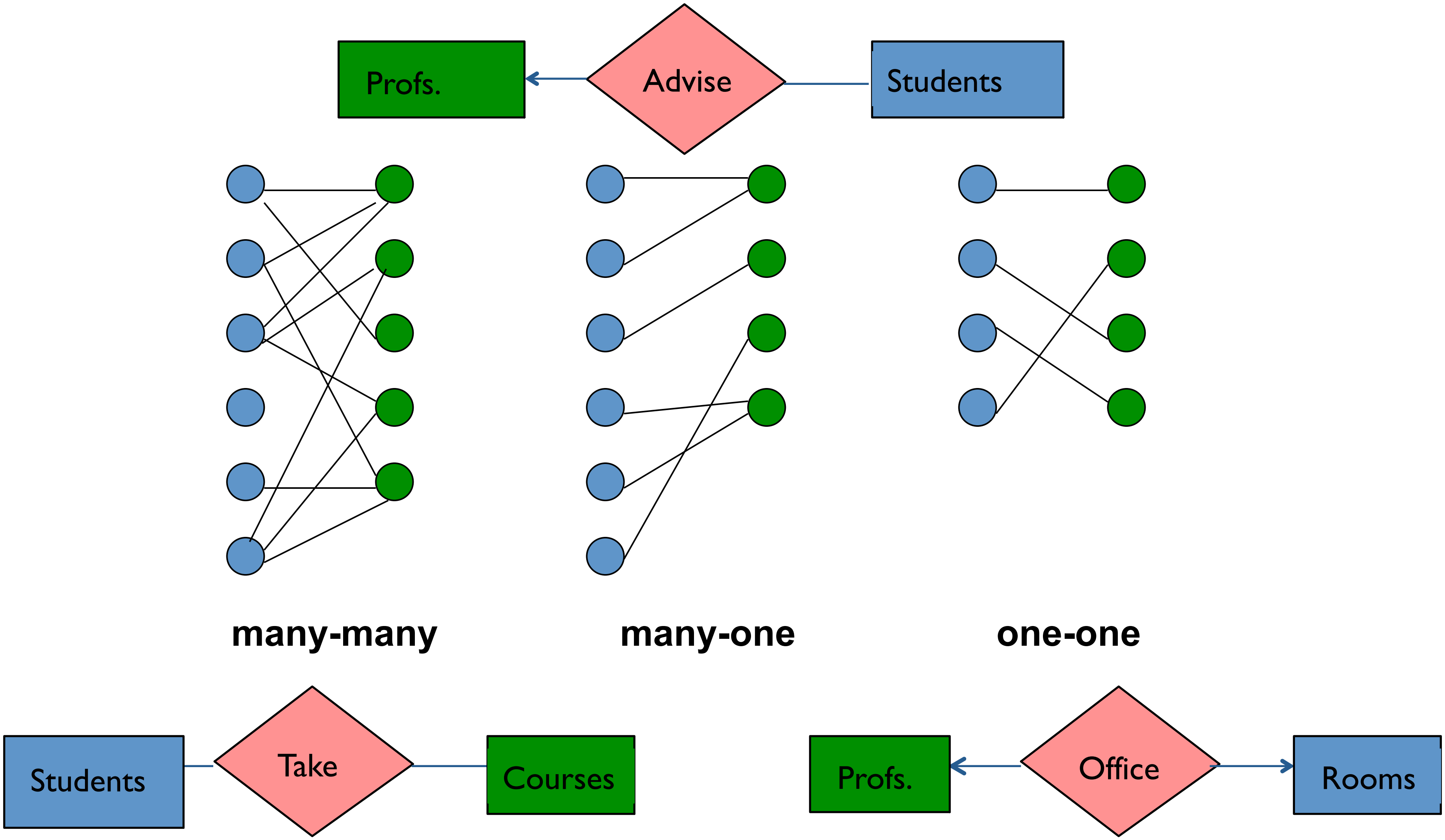
- In a one-one relationship, each entity of either entity set is related to at most one entity of the other set.

- The schema defines the multiplicity of relationships. Don't use the instances of the schema to determine multiplicity.

Representing Multiplicity

- Show a many-one relationship by an arrow entering the "one" side.
- Show a one-one relationship by arrows entering both entity sets.

Different kinds of relationships



Profs.

Advise

Students

many-many

many-one

one-one

Students

Take

Courses

Profs.

Office

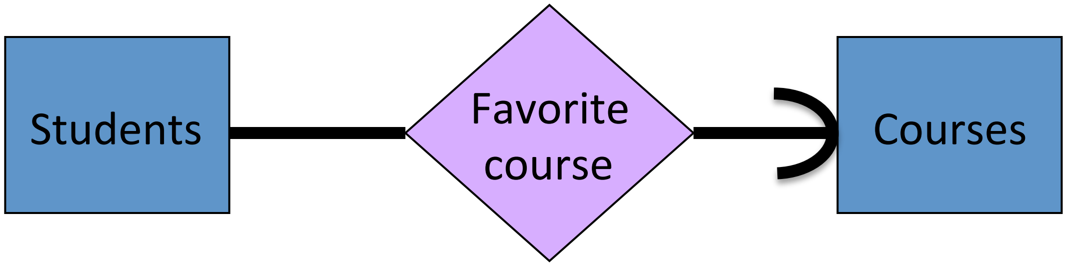
Rooms

Exactly one

- In some situations, we can also assert "exactly one," i.e., each entity of one set must be related to exactly one entity of the other set. To do so, we use a rounded arrow.

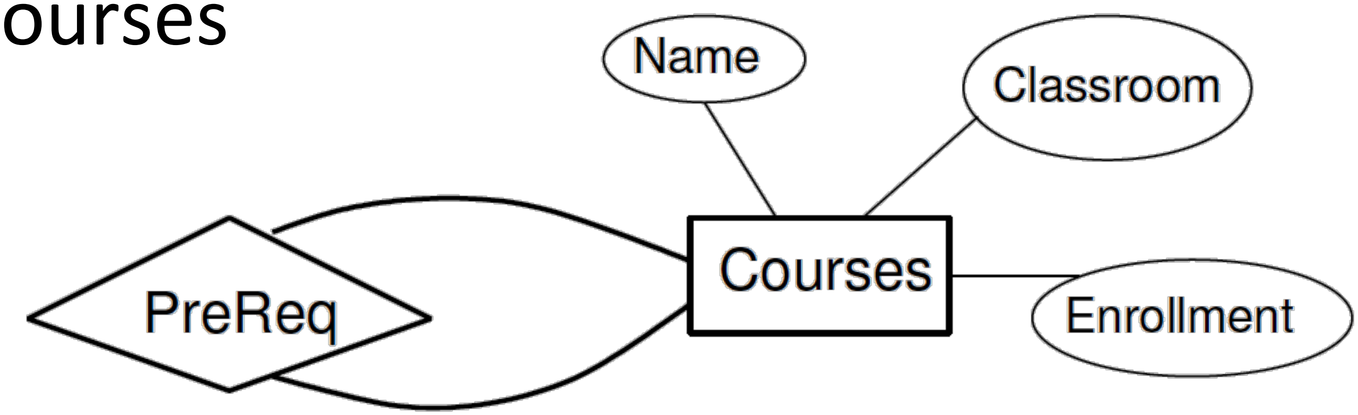
Example: Exactly One

- Consider *favorite-course* between *Students* and *Courses*.
- Some courses are not the favorite-course of any student, so a rounded arrow to *Students* would be inappropriate.
- But a student has to have a favorite-course



Roles in Relationships

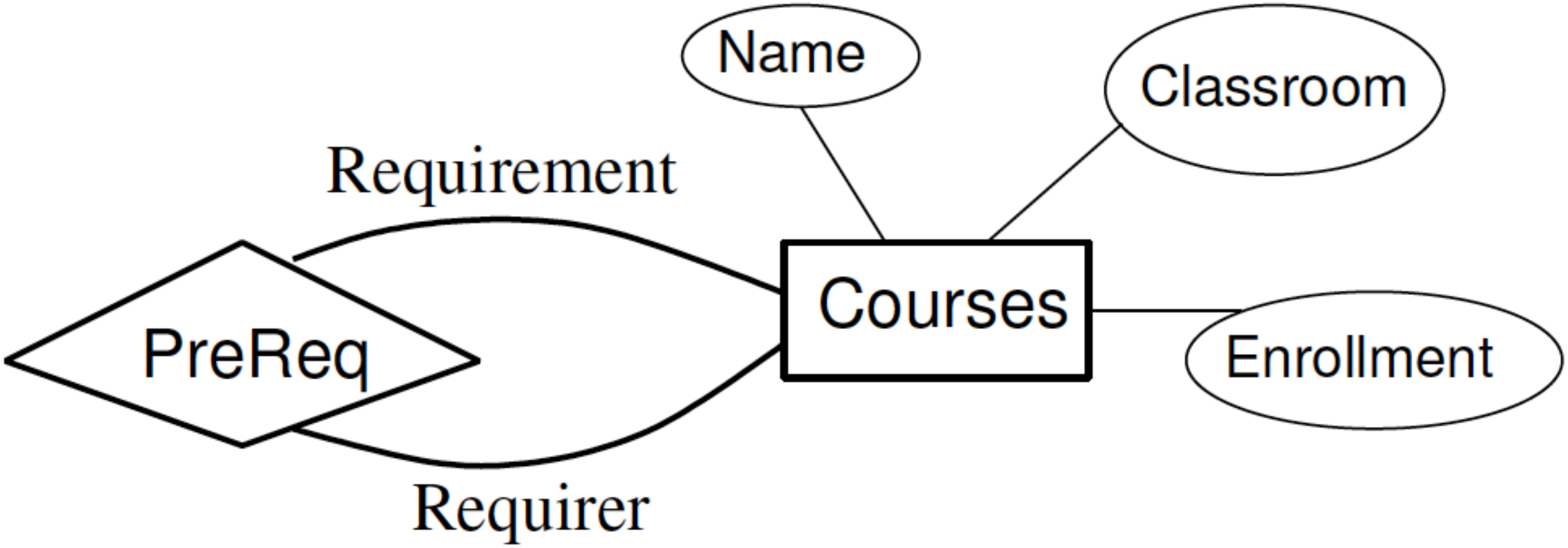
- Can the same entity set appear more than once in the same relationship?
- Prerequisite relationship between two Courses



- But which course is the pre-req?

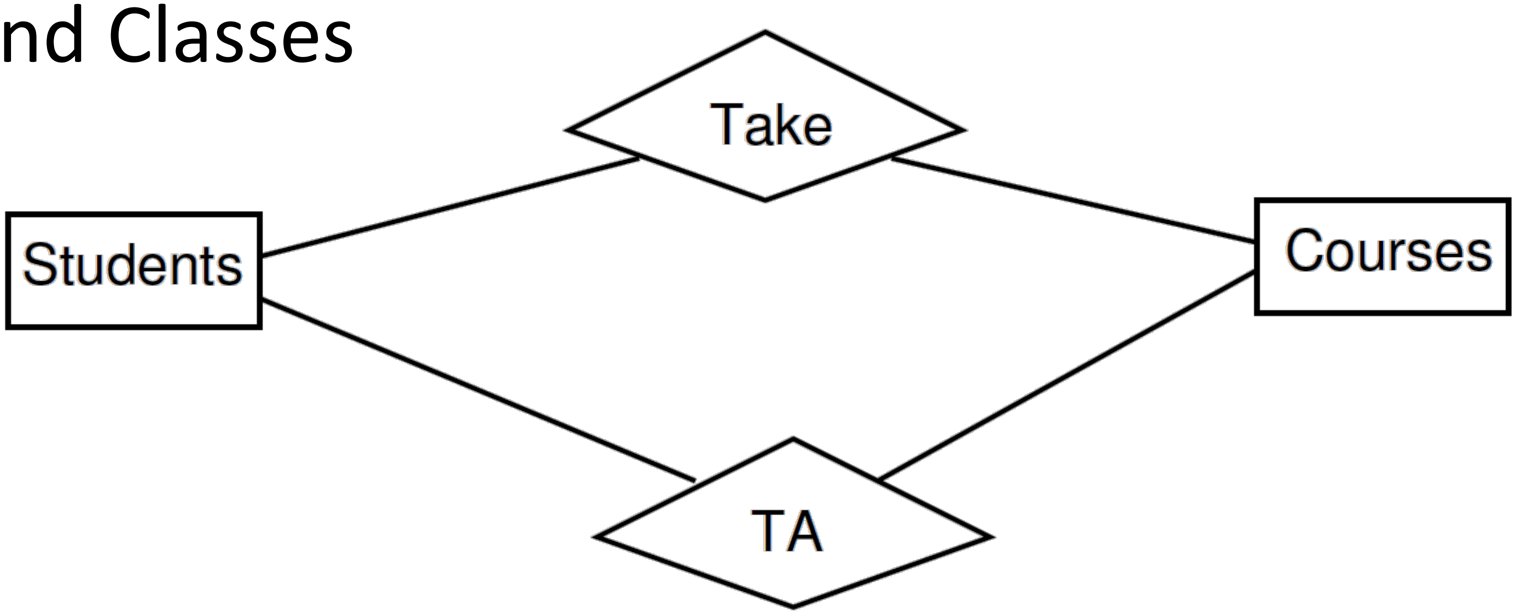
Roles in Relationships

- Label the connecting lines with the *role* of the entity

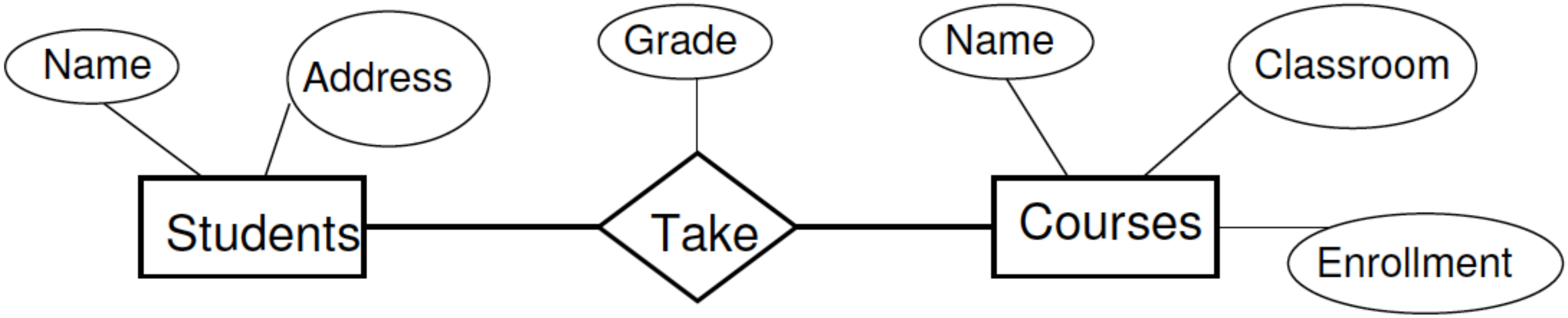


Parallel Relationships

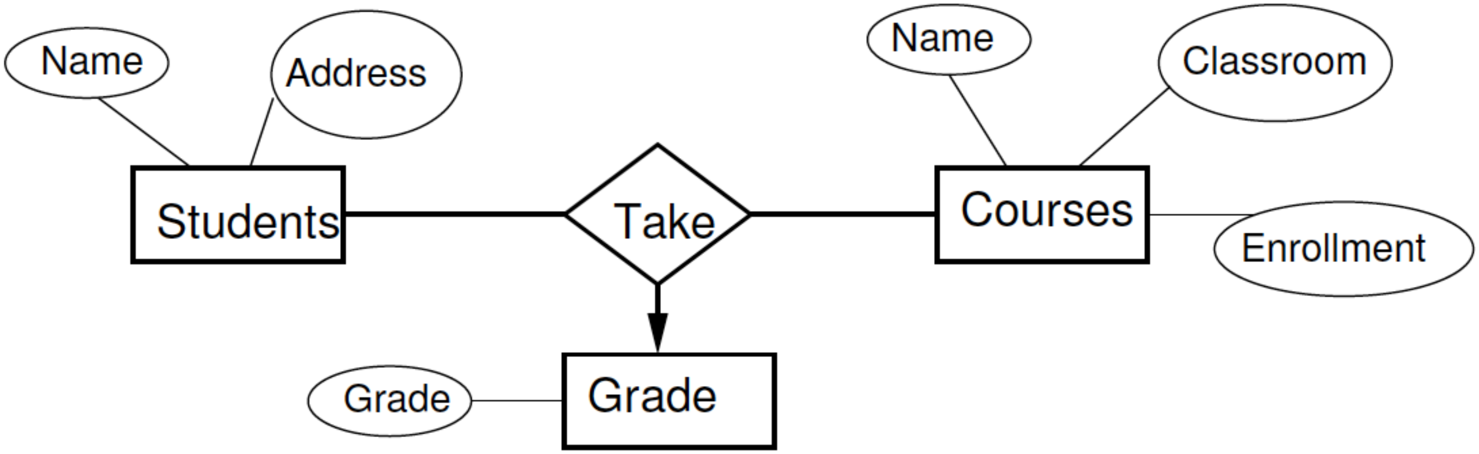
- Can there be more than one relationship between the same pair of entities?
- TA and Take relationship between Students and Classes



Are Attributes on Relationships Needed?



- Attribute on relationship → Attribute to an entity and make relationship multi-way



Multiway relationships

- Rare
- An arrow pointing to entity set E means if we select one entity from each of the other entity sets in the relationship, those entities are related to at most one entity in E.