E/R Models

(Chapter 4)

Three Pieces of Course

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

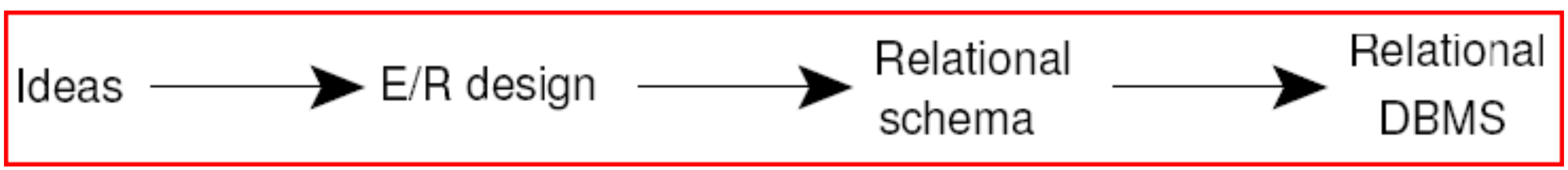
Why Learn About Database Modeling?

- 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.



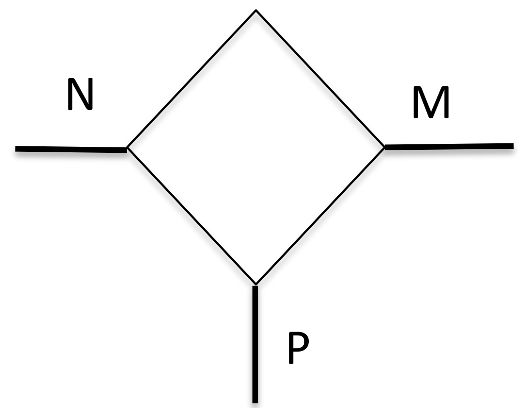
Purpose of E/R Model

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

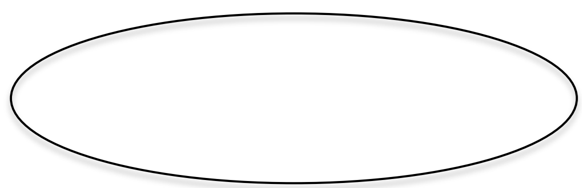
Tools

- Entities ('entity sets')

- Relationships ('rel. sets') and mapping constraints



- Attributes



Entity Sets

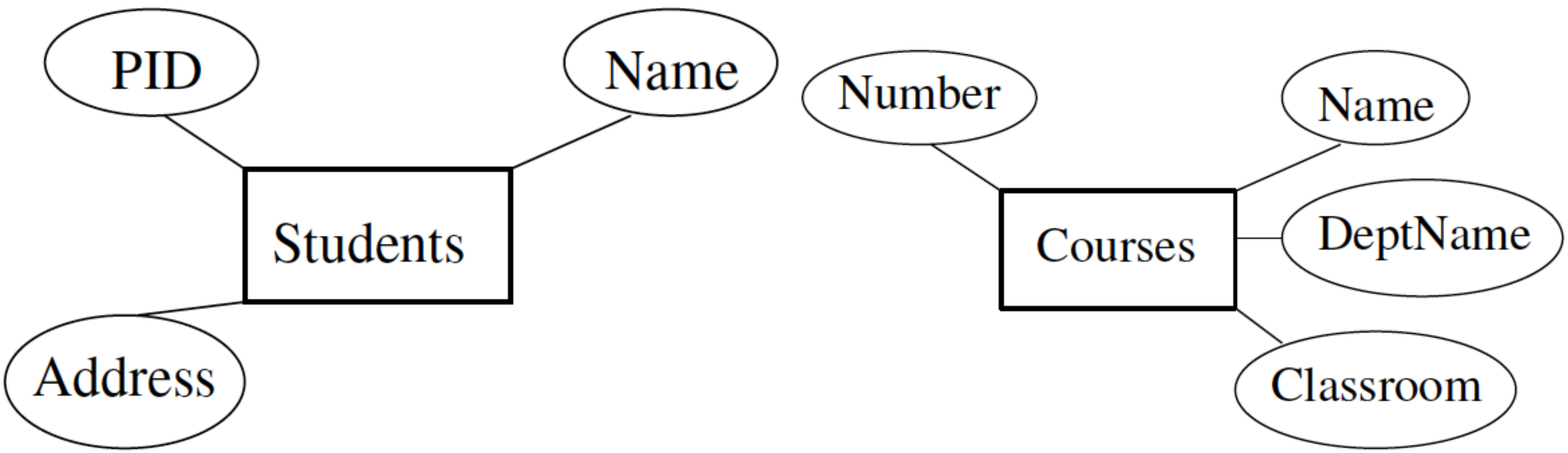
- *Entity* = "thing" or "object instance" or "noun"
- *Entity set* = collection of similar entities.
  - Similar to a class in object-oriented languages.
- Attribute = property of an entity set.
  - Generally, all entities in a set have the same set of properties.
  - Attributes can only be “primitive” types, like strings, ints, floats. No “collection” types or objects.

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
  – ***instances of*** relationship sets.

Instance of an Entity Set

- For each entity set, an instance of that entity set stores a specific set of entities.
- Each entity is a tuple containing specific values for each attribute.
- What are the examples of entity sets for our relations so far?

Instances of (binary) relationship sets

- Binary relation with entities *E* and *F*:
- Instance is a set of pairs {(*e*, *f*) : *e* is in *E* and *f* is in *F*}
  - Instance need not relate every tuple in *E* with every tuple in F. Depends on what the relationship means.
- (At the moment) Hard to visualize an instance of relationship set as a table (or relation) because *e* and *f* are entities, not simple scalar values.

Multiplicity of binary relationships

- Many-one from A to B: when each entity in A is connected to ***at most one*** entity in B.
  – If I give you a particular instance of entity A, you can give me back at most one entity in B.
  – But, each instance of B may have multiple As.
- 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.

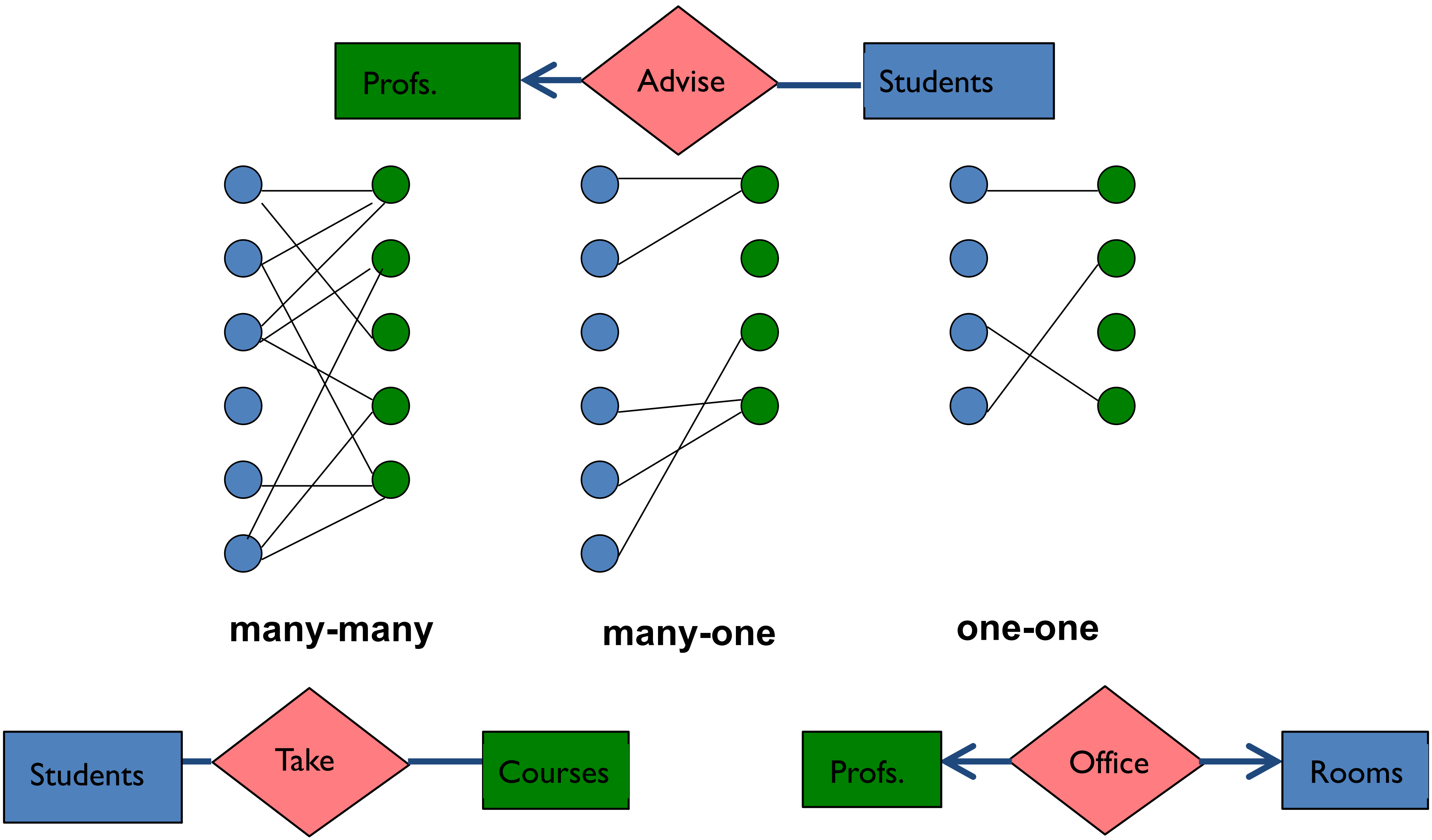
One-One Relationships

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

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

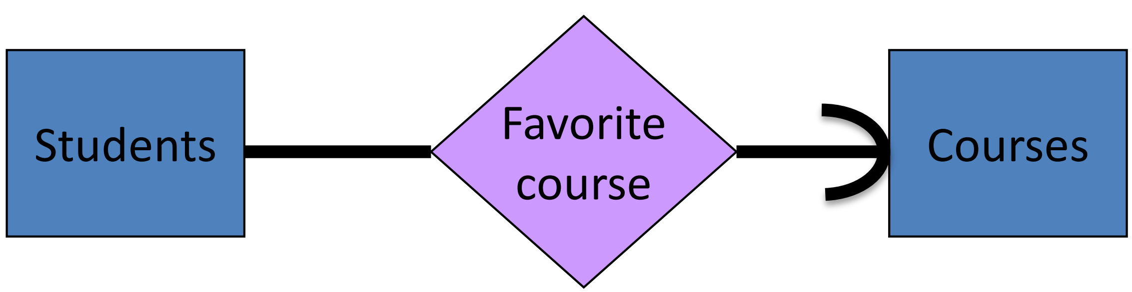


Exactly one

- 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 an arrow pointing into *Students* would be inappropriate.
- But a student has to have a favorite-course.



E/R Diagrams Day 2: Review

- Entity sets (rectangles)
- Attributes (ovals)
- Relationships (diamonds connecting entity sets)
- Multiplicity of relationships (arrows)
- Running examples: BannerWeb-style DB, bookstore DB

Attributes on relationships

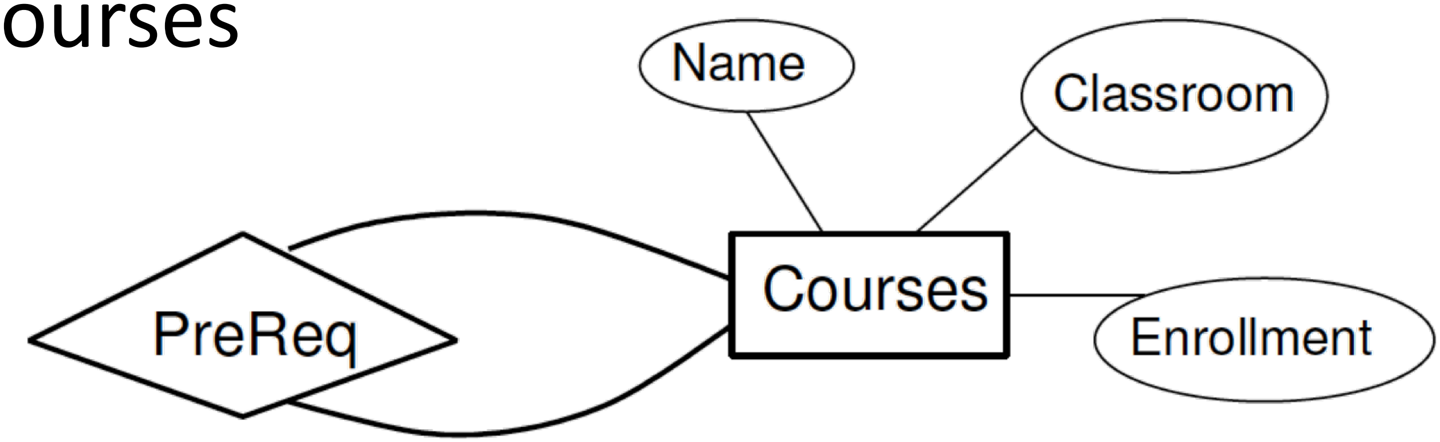
- Attributes can also be placed on a relationship, as well as on an entity set.
- Only necessary if the attribute cannot be determined from a single entity instance.
- Example:
  - Students and Courses: where do we store grades?

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/exactly) one entity in E.
- *Multiway relationships can often be converted into multiple binary relationships. (later)*

Roles in Relationships

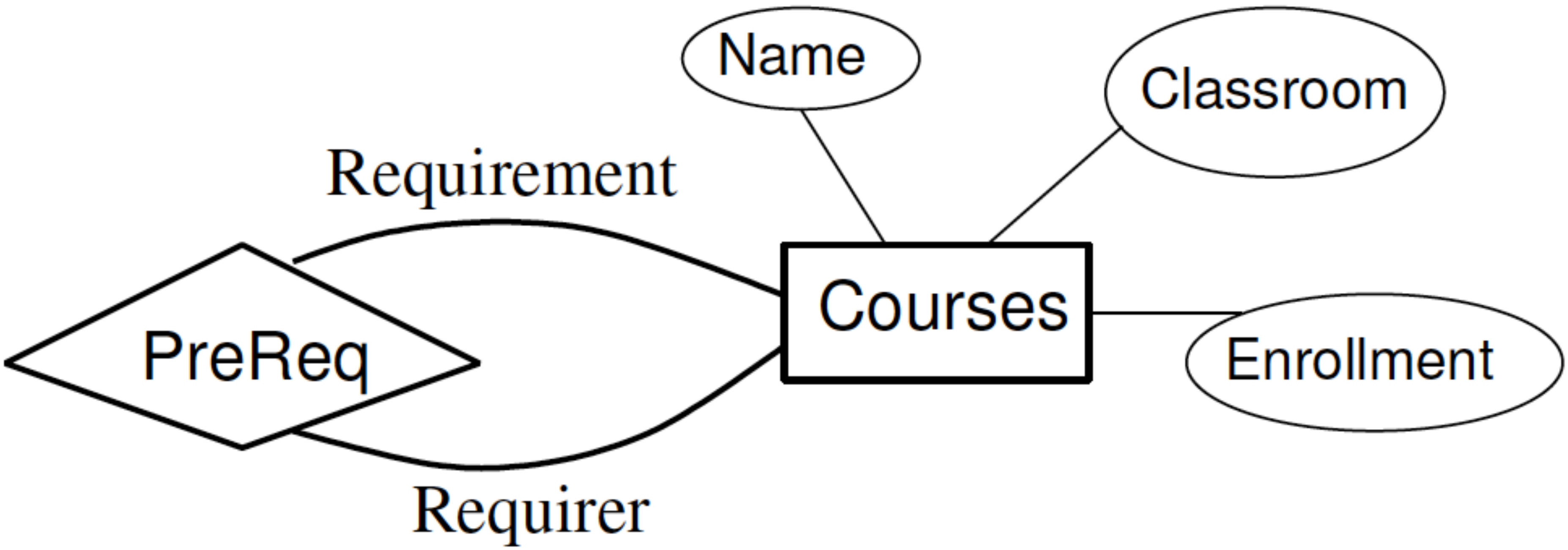
- Can the same entity set appear more than once in the same relationship?
- Prerequisite relationship between two Courses



- But which course is the pre-req?

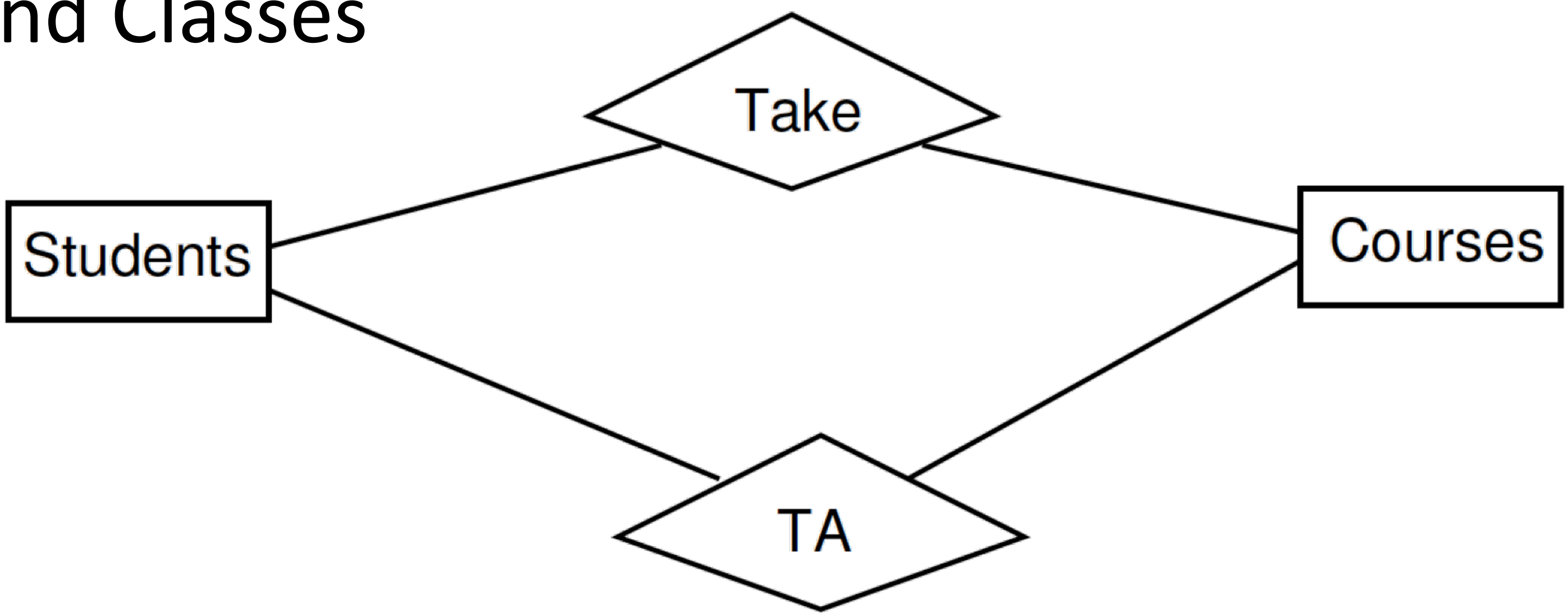
Roles in Relationships

- 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



Converting Multiway to Binary

- It is is easy to convert a multiway relationship to multiple binary relationships
  - Create a new ***connecting entity set***. Think of its entities as the tuples in the relationship set for the multiway relationship
  - Introduce many-one relationships from the connecting entity set to each of the entities in the original relationship
  - If an entity set plays > 1 role, create a relationship for each role

Try this

- Partners or triples.
- Design an E/R diagram for a bank, including info about customers and accounts.
- Customer info: name, addr, phone, SSN.
- Account info: type (checking/savings), balance.
- Accounts may have multiple customers; customers may have multiple accounts.

Try this

- What if an account can have only one customer?
- What if a customer can have only one account?
- What if a customer can have multiple addresses and multiple phones?
- (Think pre-cell-phones) What if we want to associate phones with addresses?

Is-A Hierarchies (Subclasses)

- Certain entities might need to store special properties that not all entities possess.
- Create two entity sets: a “super-entity” and a “sub-entity” and connect them with a Is-A relationship (triangle instead of diamond).

Good design principles (4.2)

- Faithfulness
  - Entity sets & attributes should reflect reality in choice of attributes and multiplicity of relationships.
  - The real-world situation can dictate what faithfulness means.
  - E/R diagram cannot convey all the information.
  - Consider Students/Courses/Profs & multiplicity – can be different ways to do this diagram.

Good design principles

- Avoid redundancy
  - Watch out for an attribute duplicating a relationship.
- Choosing the right relationships
  - Does every relationship express all the information you need it to express?

Good design principles

- Picking an attribute or entity set
- Replace E by an attribute when
  - All relationships involving E must have arrows entering E.
  - If E has >1 attribute, then no attribute depends on any other attribute.
  - No relationship involves E more than once.

Keys in E/R diagrams (4.3)

- Entity sets will have one or more ***keys***.
  - Customary to choose a ***primary key*** and underline the attributes.
- Possible for an entity set's key attributes to belong to another entity set in certain situations.
  - Is-a hierarchies
  - weak entity sets (later)

One perspective on real-world keys

- Multi-attribute and/or string keys…
- …can be time consuming and sometimes may not guarantee a lack of duplicates.
  - movie(title, year, date-released, etc)
  - title + year = lots to type to identify a movie in SQL.
  - integer key movieID saves typing!
- …break encapsulation
  - patient(first, last, DOB, etc)
  - Are these keys being transmitted in an insecure manner? Is this a security/privacy risk?
  - integer key patientID fixes this.
- …are brittle
  - Name change? Two movies with the same name/year?
  - Unique integer ID always exists, never changes.

Referential integrity in E/R

- ***Referential integrity***: requires every value of an attribute in one relation to appear as the value of an attribute in another (or the same) relation.
- Enforced through multiplicity arrows
- Degree constraints can be added to further restrict multiplicity.

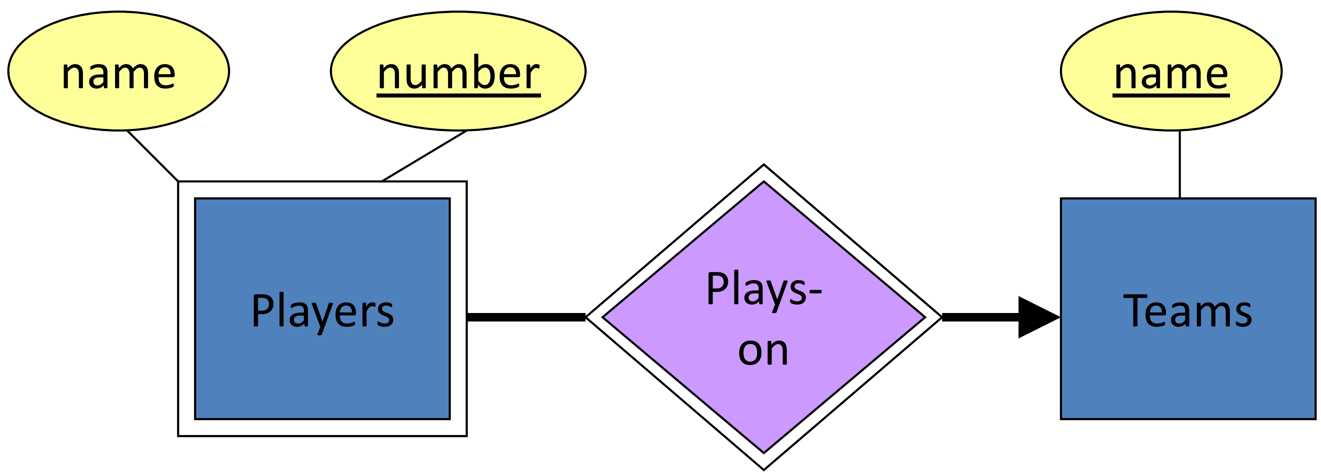
Try US Congress handout

Weak entity sets

- A weak entity set is an entity set whose (primary) key contains attributes from one or more other entity sets.
- In other words, an entity set E is 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 entity sets in E's key.
- Possible that all attributes in a weak entity set's key come from other entity sets.

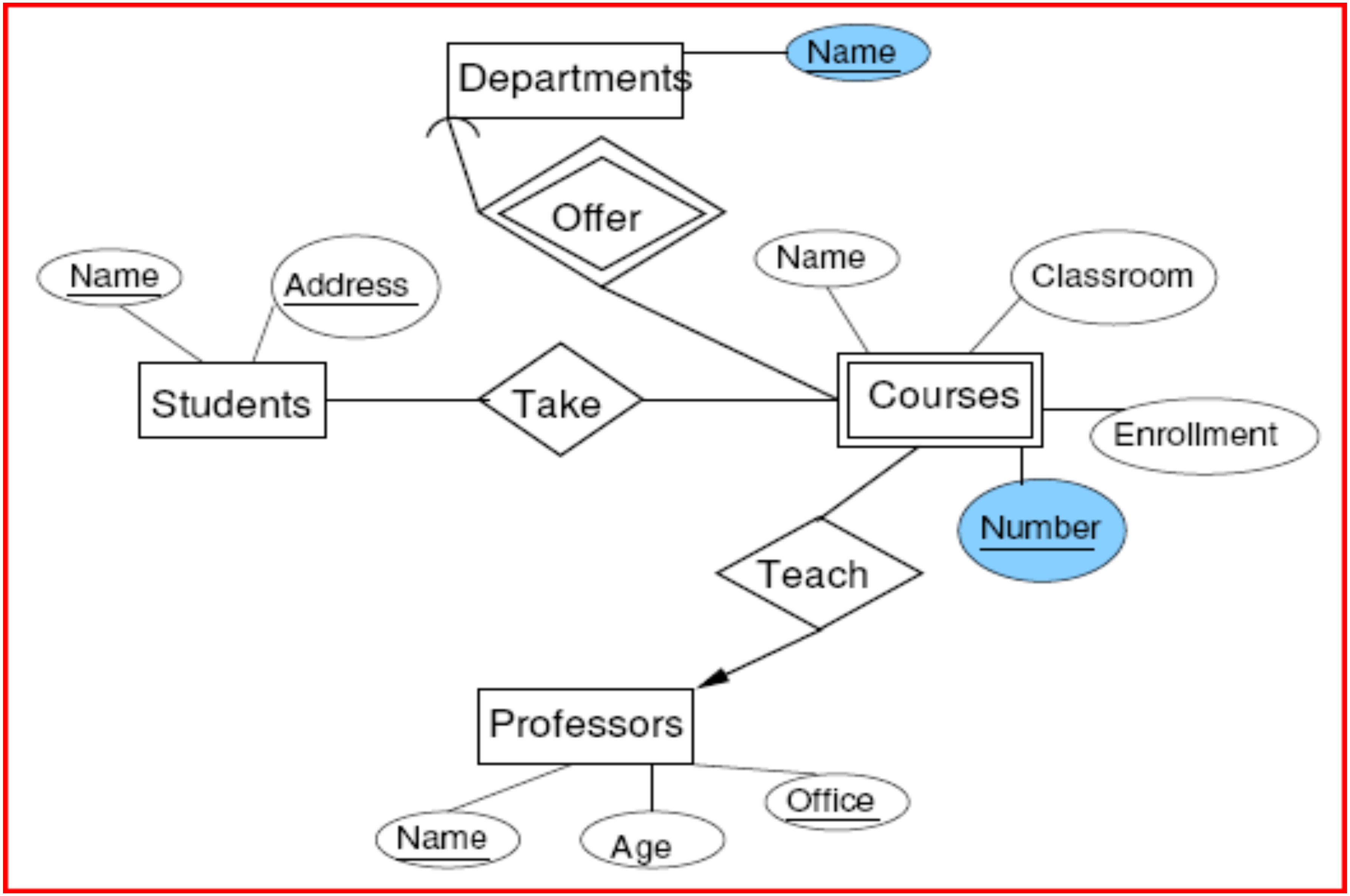
Example

- Consider players in a sports league:
  - Name is not a key (might be duplicate names)
  - Number is certainly not a key (numbers will be duplicated across teams)
  - But number + team should be a key



- Use double border for weak entity sets and their supporting many-one relationships.

How about courses and departments?



Keys for a weak entity set

- A relationship R from a weak entity set E to F is ***supporting*** if
  - R is a binary, many-one relationship from E to F.
  - R has referential integrity from E to F.
- F supplies its key attributes to define E's key.
- If F itself is a weak entity set, then we must find F's supporting relationships and also use the keys from those supporting entity sets.

Where do weak entity sets come from?

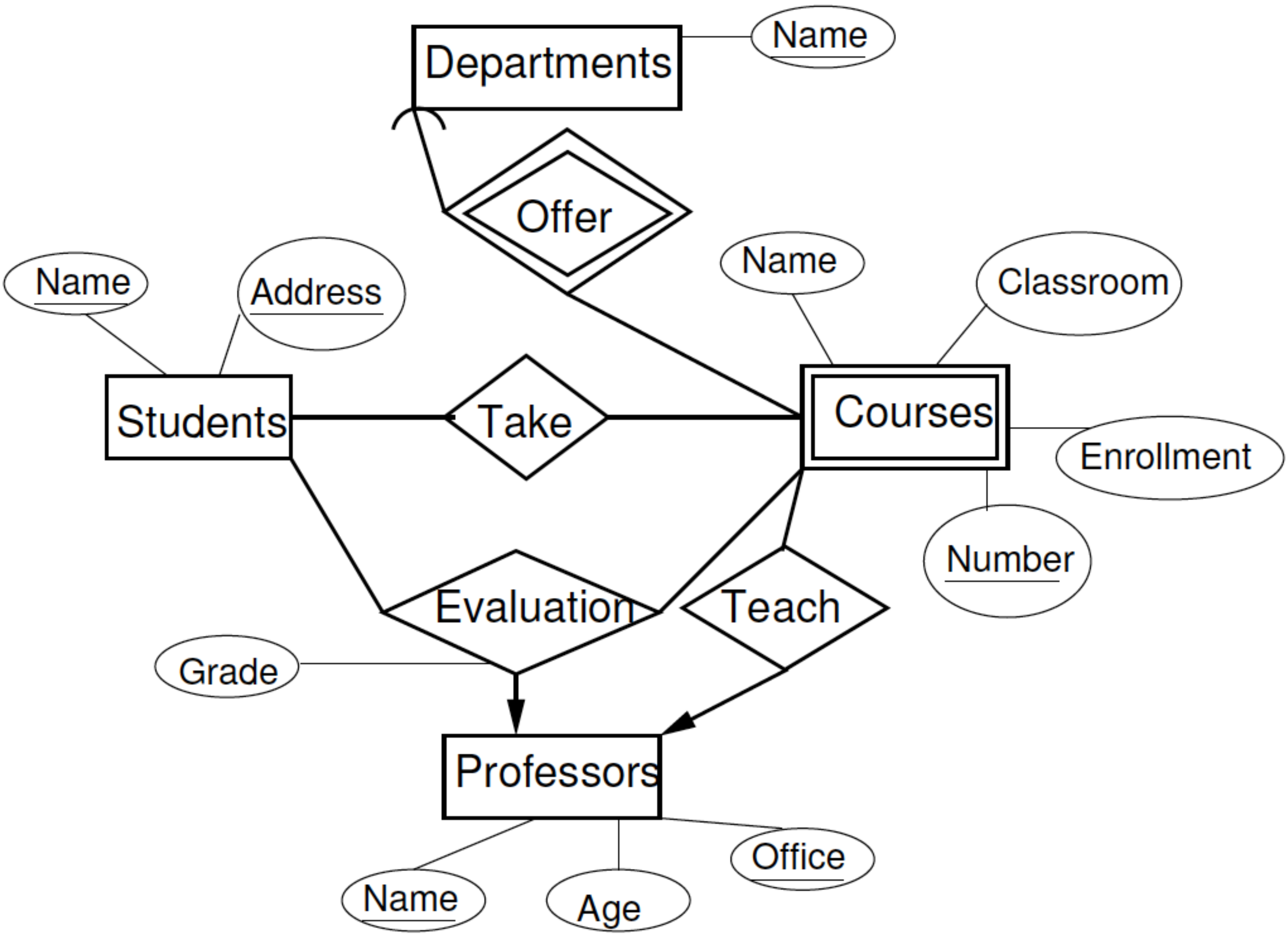
- Cause 1: Implicit hierarchies not from an "is-a" relationship.
  - A player “belongs to” a team, or a flight “is flown by” an airline.
  - Happens when a piece of a key is represented as an entity set rather than an attribute.
    - Can (technically) be solved by putting a unique ID on an entity set, but sometimes this causes more trouble than it’s worth.
  - "is-a" hierarchies seem to lead to weak entity sets (subclasses), but we don't notate them with double borders because their hierarchical relationships are always one-one.

Where do weak entity sets come from?

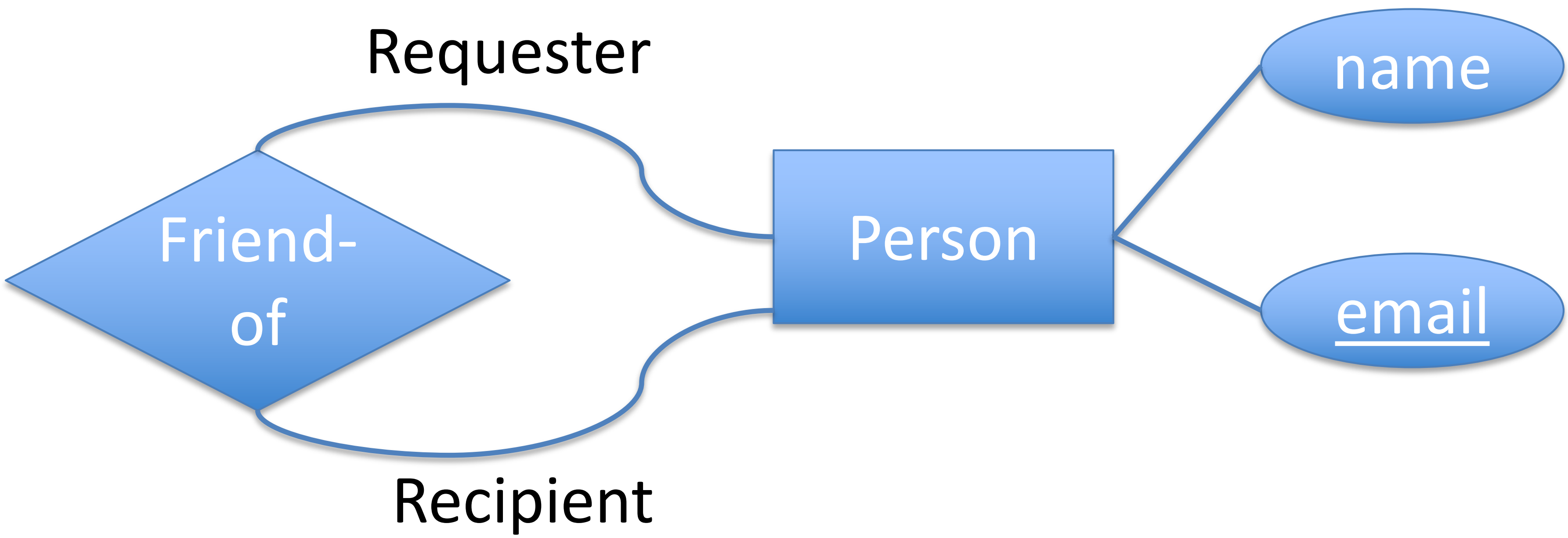
- Cause 2: Connecting entity sets created by eliminating a multi-way relationship.
  - Often, connecting entity sets have no attributes of their own; they must pick up their key attributes from the entity sets they connect.
  - Example: A CUSTOMER rents a CAR from a SALESPERSON.

Converting E/R diagrams to relational designs

- Entity set -> Relation
  - Attribute of entity set -> attribute of relation
  - Key of entity set -> primary key of relation
- Relationship -> Relation
  - Attribute of relationship -> attribute of relation
  - Key attribute of connecting entity set -> key attribute of relation
- Special cases: weak entity sets, "is-a" hierarchies, combining relations.


Departments
Name
Offer
Name
Address
Name
Classroom
Students
Take
Courses
Enrollment
Number
Evaluation
Teach
Grade
Professors
Office
Name
Age

Handling multiple roles

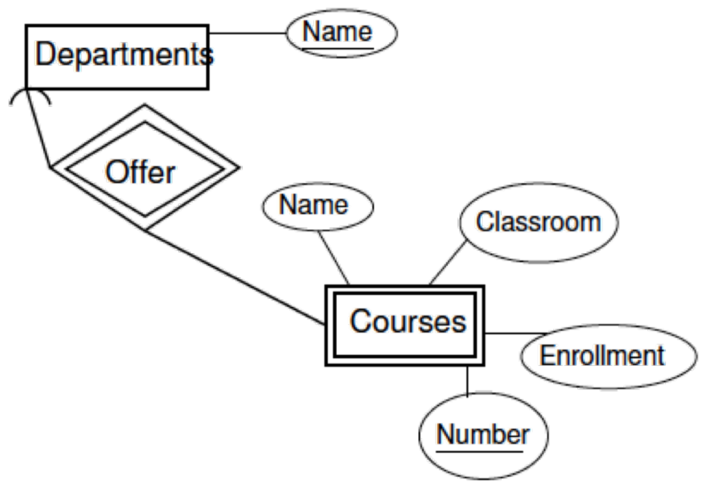


If an entity set E appears k > 1 times in a relationship R, then the key attributes for E appear k times in the relation for R, appropriately renamed.

Handling weak entity sets

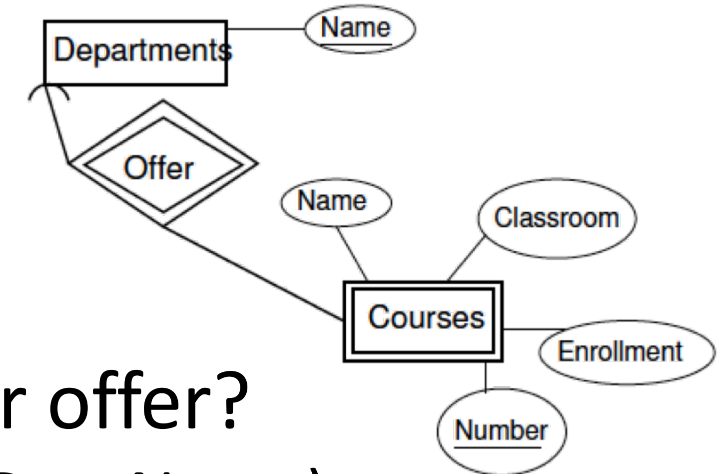
- For each weak entity set W, create a relation with attributes:
  - attributes of W
  - attributes of supporting relationships for W
  - *key* attributes of supporting entity sets for W

Supporting Relationships



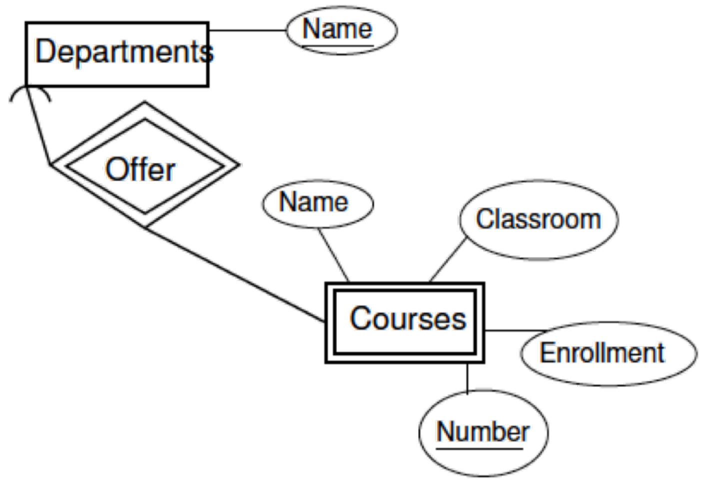
- Schema for Departments is Departments(Name)
- Schema for Courses is Courses(Number, DeptName, CourseName, Classroom, Enrollment)
- What is the schema for Offer?

Supporting Relationships



- What is the schema for offer?
  - Offer(Name, Number, DeptName)
  - But Name and DeptName are identical, so the schema for Offer is Offer(Number, DeptName)
  - The schema for Offer is a subset of the schema for the weak entity set, so ***we can dispense with the relation for Offer.***
  - ***Key point: Don't make a relation for supporting relationships.***

Summary of Weak Entity Sets



- If W is a weak entity set, the relation for W has a schema whose attributes are
  - all attributes of W
  - all attributes of supporting relationships for W
  - for each supporting relationship for W to an entity set E
    - the key attributes of E
- There is no relation for any supporting relationship for W

Combining Relations

- Consider many-one Teach relationship from Courses to Professors
- Schemas are:

  Courses(<u>Number</u>, <u>DepartmentName</u>, CourseName, Classroom, Enrollment)

  Professors(<u>Name</u>, <u>Office</u>, Age)

  Teach(<u>Number</u>, <u>DepartmentName</u>, <u>ProfessorName</u>, <u>Office</u>)

Combining Relations

Courses(Number, DepartmentName, CourseName, Classroom, Enrollment)

Professors(Name, Office, Age)

Teach(Number, DepartmentName, ProfessorName, Office)

- The key for Courses uniquely determines all attributes of Teach
- We can combine the relations for Courses and Teach into a single relation whose attributes are
  - All the attributes for Courses,
  - Any attributes of Teach, and
  - The key attributes of Professors

Rules for Combining Relations

- We can combine into one relation Q
  - The relation for an entity set E
  - all many-to-one relationships R1, R2, …, Rk from E to other entity sets E1, E2, …, Ek respectively
- The attributes of Q are
  - All the attributes of E
  - Any attributes of R1, R2, …, Rk
  - The key attributes of E1, E2, …, Ek
- Combining a ***many-many*** relationship with one of its entity sets often leads to redundancy. You probably never want to do this!

Is-a to Relational

- Three approaches:
  - E/R viewpoint
  - Object-oriented viewpoint
  - “Flatten” viewpoint

Rules Satisfied by an Is-a Hierarchy

- The hierarchy has a root entity set.
- The root entity set has a key that identifies every entity represented by the hierarchy.
- A particular entity can have components that belong to entity sets of any subtree of the hierarchy, as long as that subtree includes the root.

Example ISA hierarchy

Is-a to Relational Method I: E/R Approach

- Create a relation for each entity set
- The attributes of the relation for a non-root entity set E are
  - the attributes forming the key (obtained from the root) and
  - any attributes of E itself
- An entity with components in multiple entity sets has tuples in all the relations corresponding to these entity sets
- Do not create a relation for any is-a relationship
- Create a relation for every other relationship

Is-a to Relational Method II: Object Oriented Approach

- Treat entities as objects that are members of ***a*** particular subtree in the tree.
  - Subtrees must contain the root.
  - Subtrees may contain more than one entity set.
- What are all the logically-possible classes for books in our hierarchy?

Is-a to Relational Method II: Object Oriented Approach

- Enumerate all subtrees of the hierarchy that contain the root.
- For each such subtree,
  - Create a relation that represents entities that have components in exactly that subtree.
  - The schema for this relation has all the attributes of all the entity sets in that subtree.

Is-a to Relational Method III: “Flatten” Approach (or "NULLs")

- Make one relation for the whole hierarchical structure.
- Use NULL for any attribute that is not defined for a particular entity.

Comparison of the Three Approaches

- Trade-offs
  - In general, we want to minimize joins (takes time) and also minimize duplicated or redundant information (takes space [memory]).
  - It is expensive to answer queries involving several relations (advantage: flatten)
  - E/R approach works well for some queries where info is duplicated among relations.
  - E/R approach is hard for other queries because we may need joins.

Comparison of the Three Approaches

- Number of relations for n relations in the hierarchy
  - We like to have a small number of relations
  - Flatten
    - 1
  - E/R
    - n
  - OO
    - Can be 2^n

Comparison of the Three Approaches

- Redundancy and space usage
  - Flatten
    - May have a large number of NULLs
    - (also prevents you from using NULL to denote something besides class membership)
  - E/R
    - Several tuples per entity, but only key attributes are repeated
  - OO
    - Only one tuple per entity