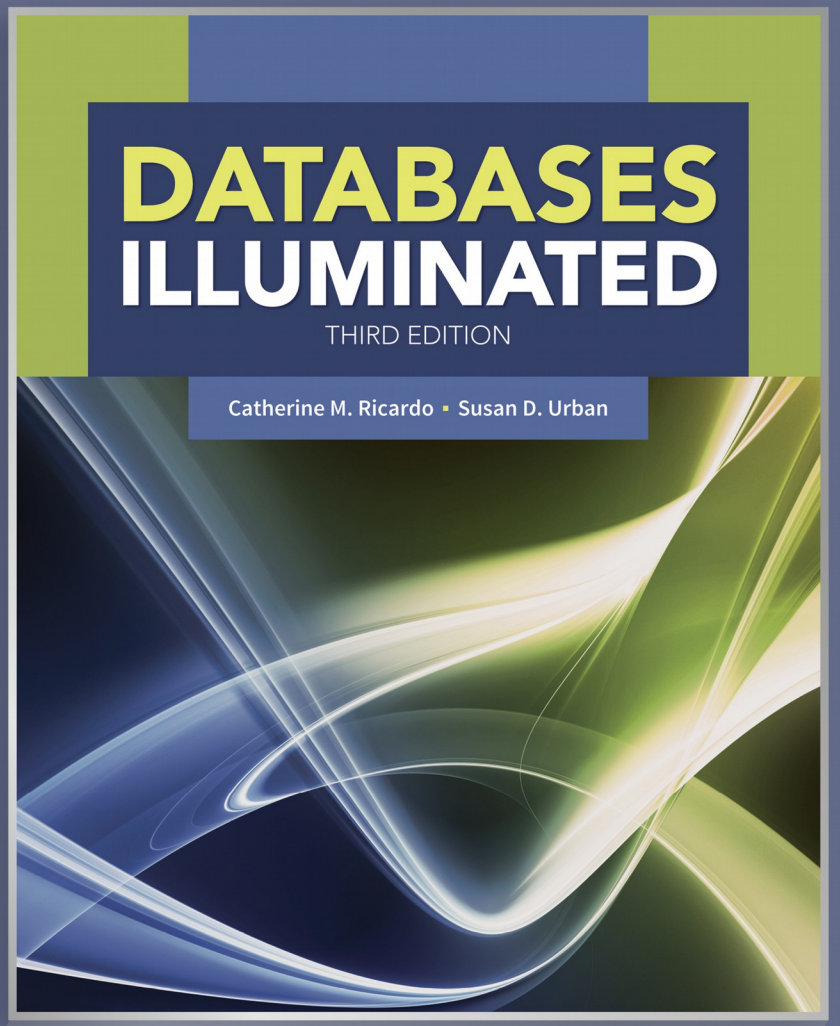
Databases Illuminated

Chapter 3
The Entity Relationship Model



Purpose of E-R Model

(ER Model)

- Facilitates database design
- Express logical properties of mini-world of interest within enterprise - Universe of Discourse
- Conceptual level model
- Not limited to any particular DBMS
- E-R diagrams used as design tools
- A semantic model – captures meanings

Basic Symbols for E-R Diagram

- Entity – rectangle
- Attribute – oval
- Relationship – diamond
- Link - line

FIGURE 2.8
Basic Symbols for E-R Diagrams

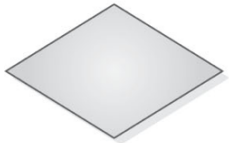
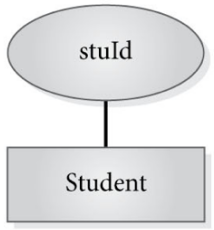
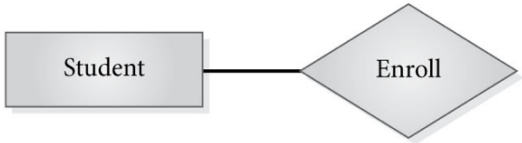
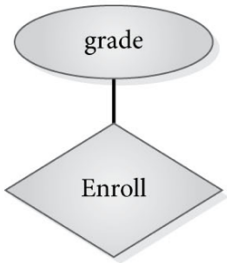
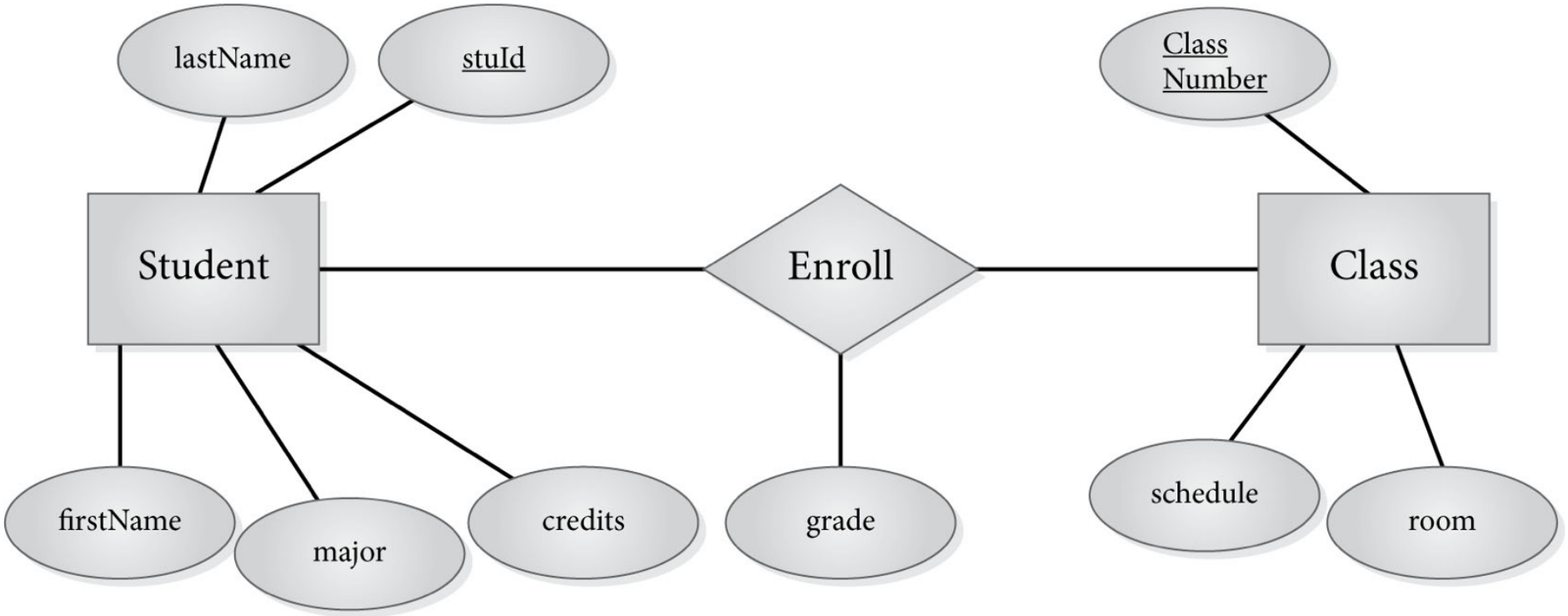
SYMBOL	NAME	MEANING	EXAMPLE
	Rectangle	Entity Set	
	Oval	Attribute	
	Diamond	Relationship	
	Line	Links: Attribute to Entity	
		Entity Set to Relationship	
		Attribute to Relationship	

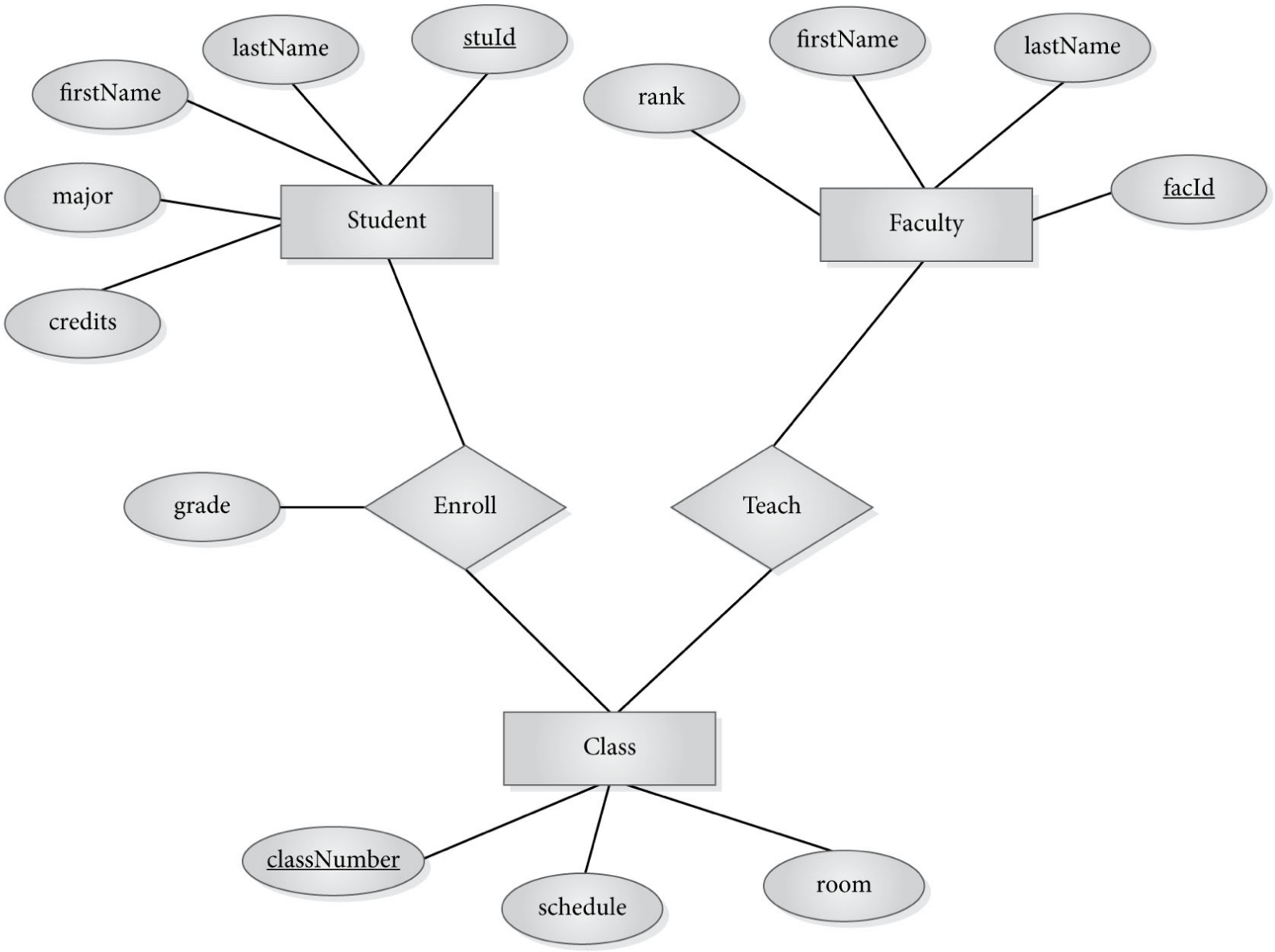
FIGURE 2.9
Simplified E-R Diagram



Entity

- Object that exists and that can be distinguished from other objects
- Can be person, place, event, object, concept in the real world
- Can be physical object or abstraction
- Entity **instance** is a particular person, place, etc.
- Entity **type** is a category of entities
- Entity **set** is a collection of entities of same type-must be well-defined
- Entity type forms **intension** of entity – permanent definition part
- Entity instances form **extension** of entity – all instances that fulfill the definition at the moment
- In E-R diagram, rectangle represents entity set, not individual entities

FIGURE 3.1
A Simplified E-R Diagram for the `University` Database



Attributes

- Defining properties or qualities of entity type
- Represented by oval on E-R diagram
- **Domain** – set of allowable values for attribute
- Attribute maps entity set to domain
- May have **null values** for some entity instances – no mapping to domain for those instances
- May be **multi-valued** – use double oval on E-R diagram
- May be **composite** – use oval for composite attribute, with ovals for components
- May be **derived** – use dashed oval

FIGURE 3.2
Credits as Mapping of the `Student` Entity Set to the `credits` Domain

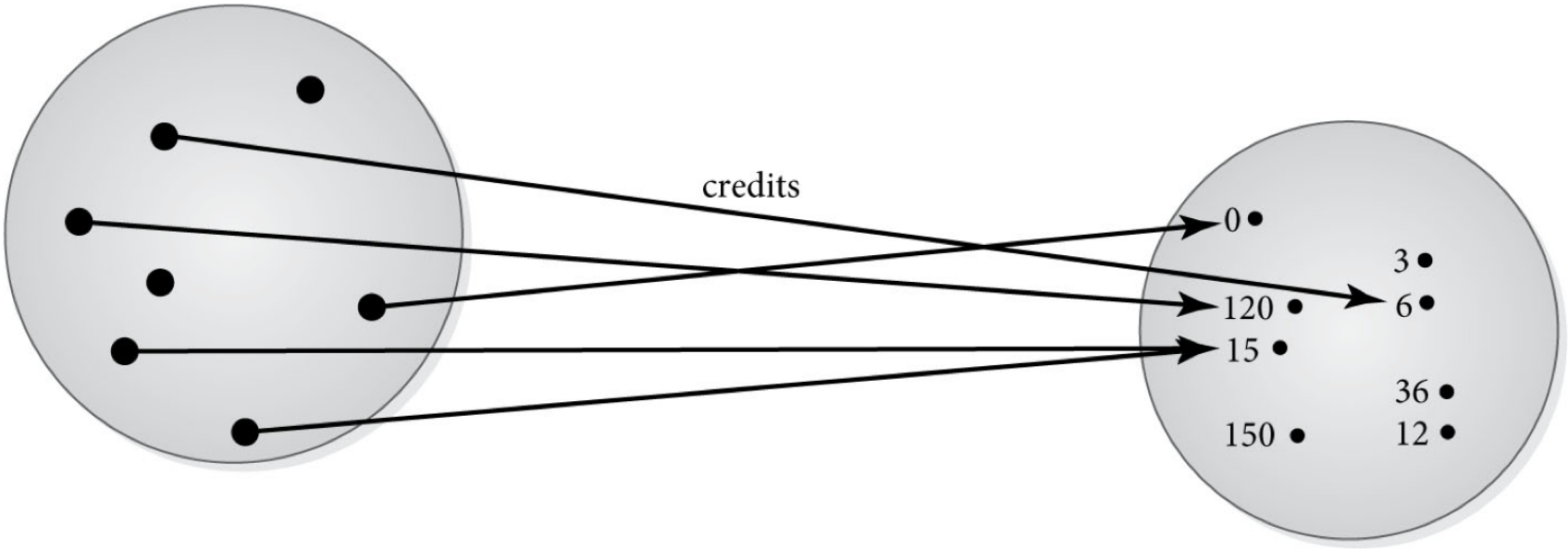


FIGURE 3.3

`Student` Entity Set with Multivalued Attribute `emailAddress`

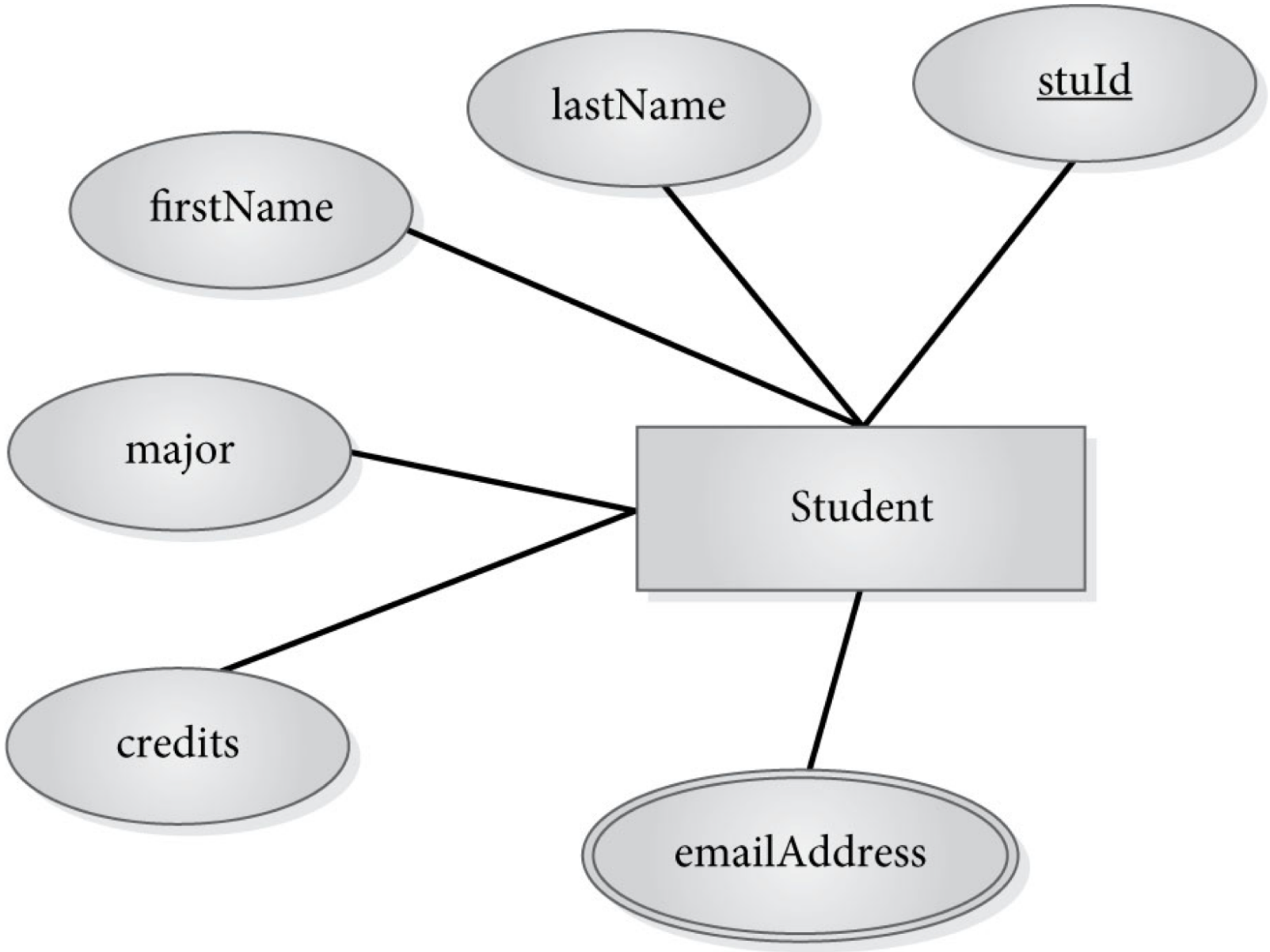
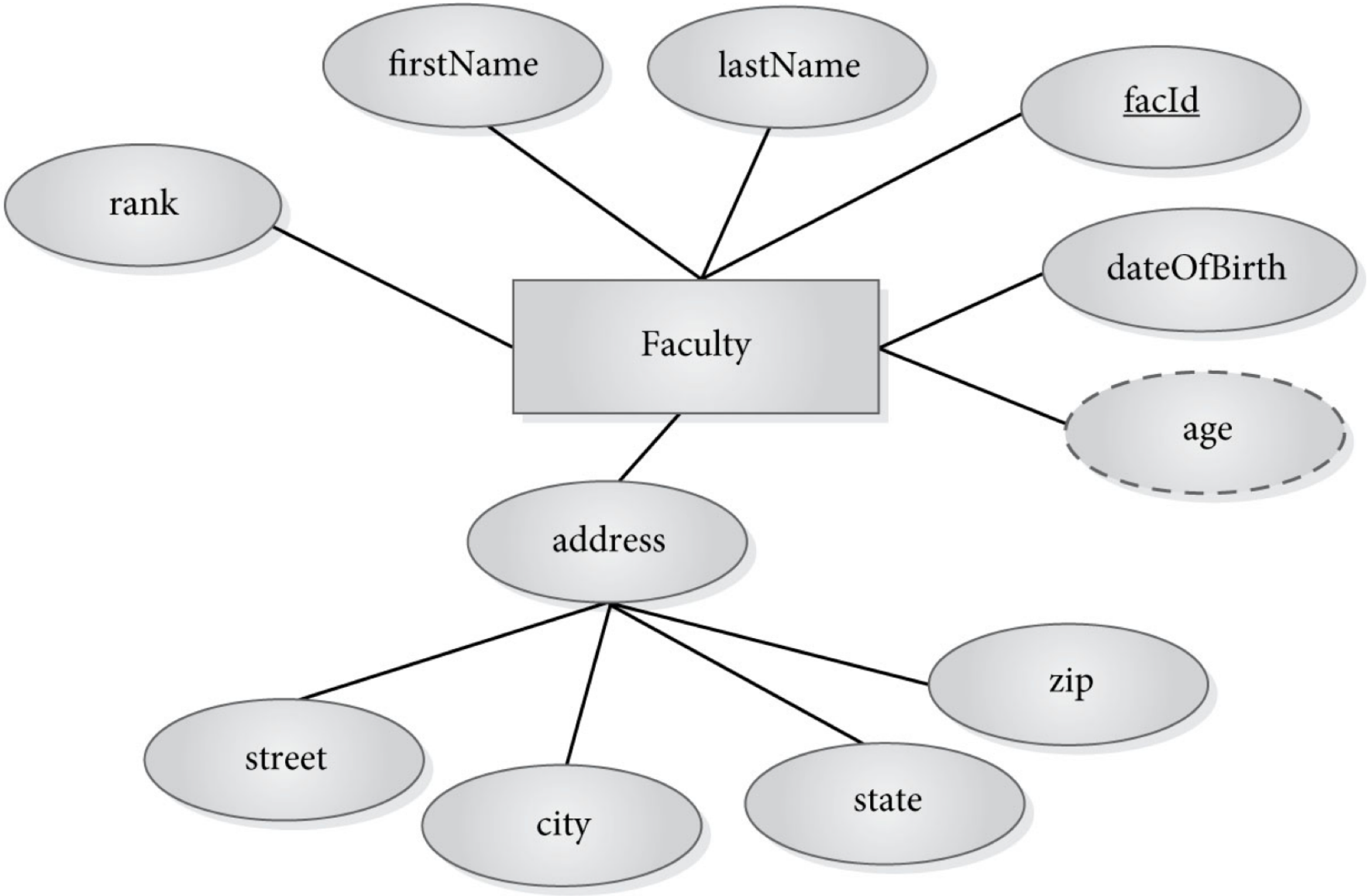


FIGURE 3.4
Faculty Entity Set with Composite Attribute `address` and Derived Attribute `age`



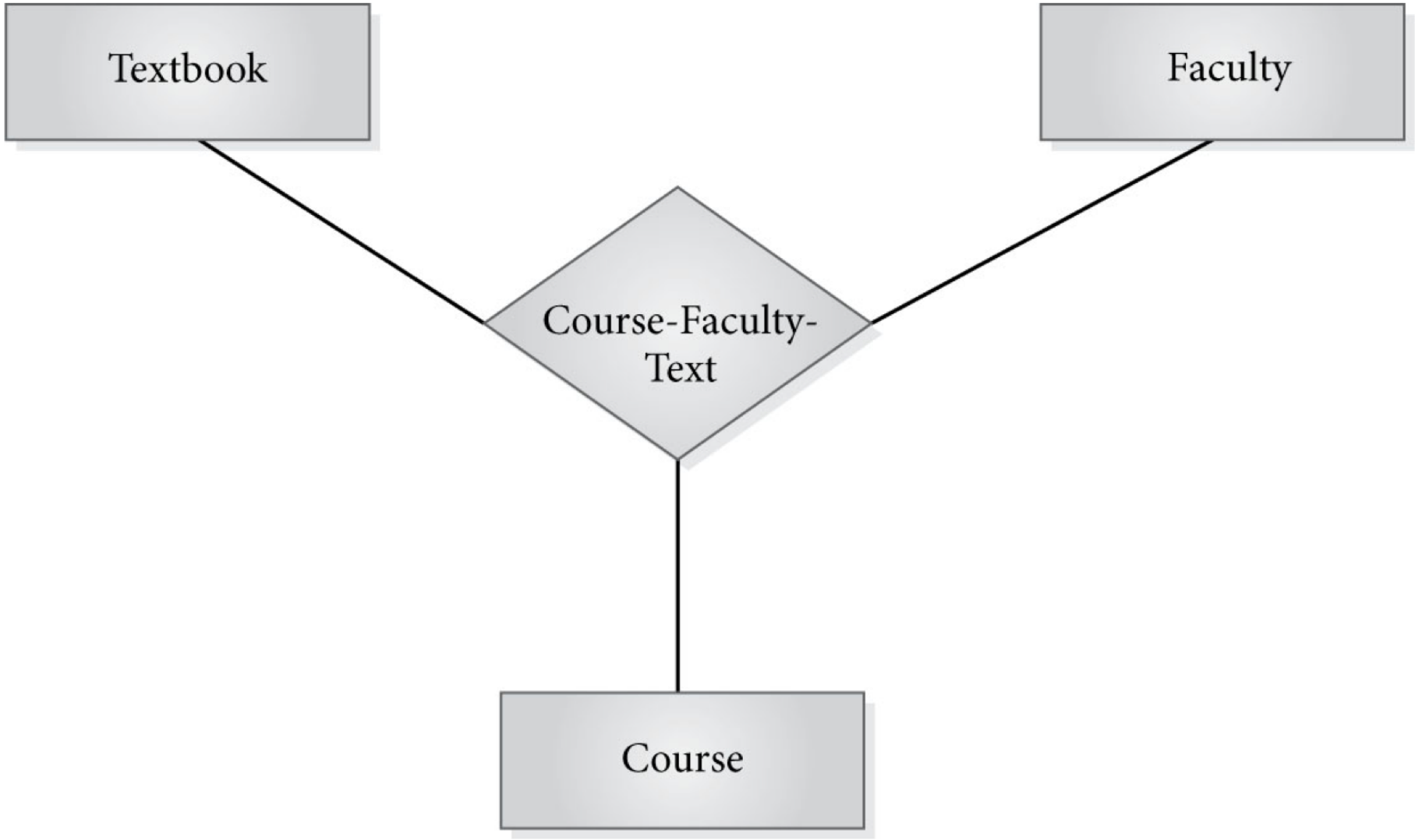
Keys

- **Superkey**: attribute or set of attributes that uniquely identifies an entity
- **Composite key**: key with more than one attribute
- **Candidate key**: superkey such that no proper subset of its attributes is also a superkey (minimal superkey –no unnecessary attributes)
- **Primary key**: candidate key actually used for identifying entities and accessing records
- **Alternate key**: candidate key not used for primary key
- **Secondary key**: attribute or set of attributes used for accessing records, but not necessarily unique
- **Foreign key**: term used in relational model (but not in the E-R model) for an attribute that is primary key of a table and is used to establish a relationship, usually with another table, where it appears as an attribute also

Relationships

- Connections or interactions between entity sets
- Represented by diamond on E-R diagram
- Relationship **type** – category of relationships
- Relationship **set** – collection of relationships of same type, consists of relationship instances – relationships that exist at a given moment
- Type forms intension; set forms extension of relationship
- Relationship can have descriptive attributes
- **Degree** of relationship
  - **Binary** – links two entity sets; set of ordered pairs
  - **Ternary** – links three entity sets; ordered triples
  - **N-ary** – links n entity sets; ordered n-tuples
  - **Unary**(**recursive**)–links entity set to itself; ordered tuples from same set

FIGURE 3.6
A Ternary Relationship



Cardinality of Binary Relationships

- Number of entity instances to which another entity instance can map under the relationship
- **One-to-one**: X:Y is 1:1 is each entity in X is associated with at most one entity in Y and each entity in Y with at most one entity in X.
- **One-to-many**: X:Y is 1:M is each entity in X can be associated with many entities in Y, but each entity in Y with at most one entity in X. (many=more than one)
- **Many-to-many**: X:Y is M:M if each entity in X can be associated with many entities in Y, and each entity in Y with many entities in X
- **Figure 3.7** shows several representation methods

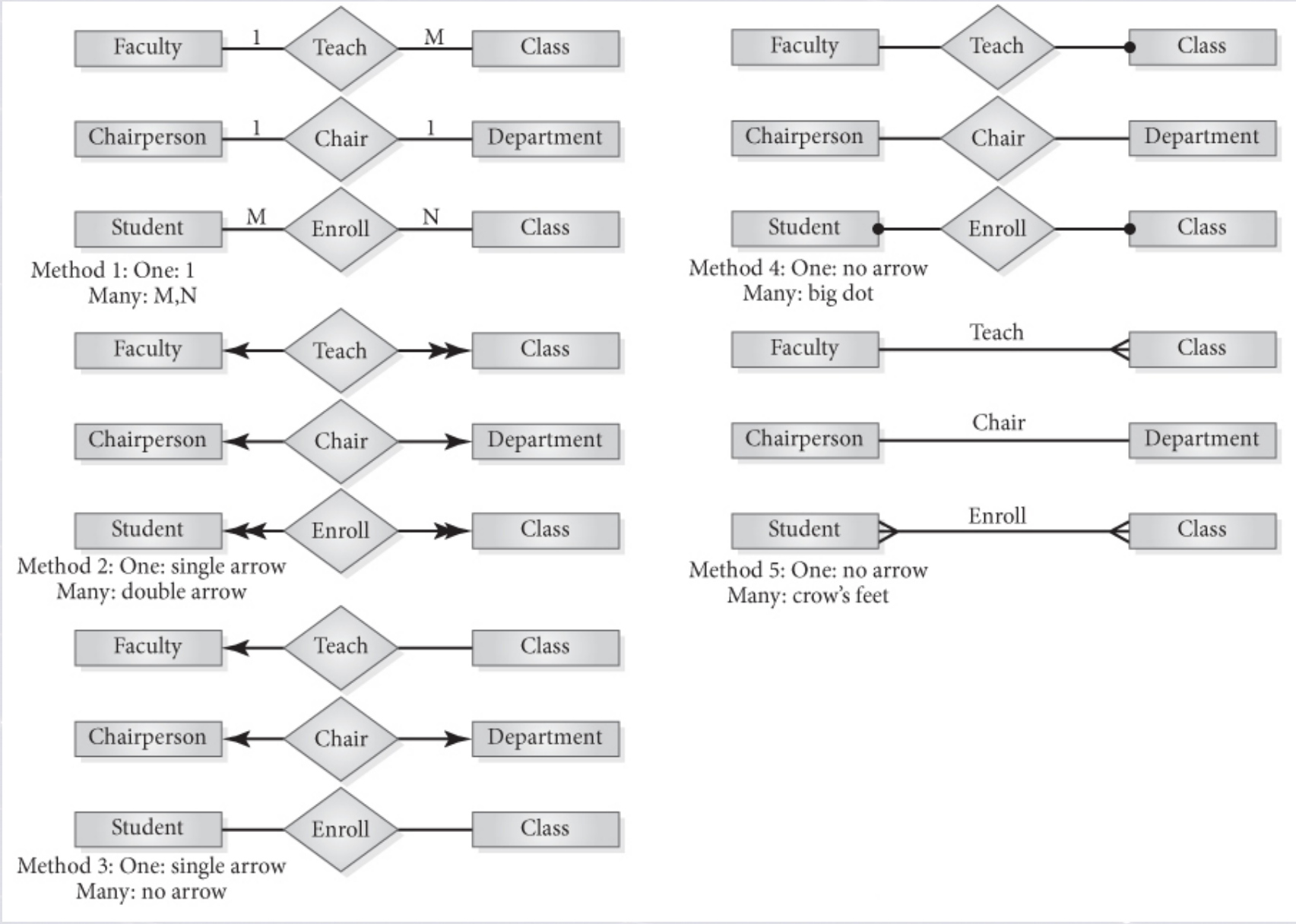
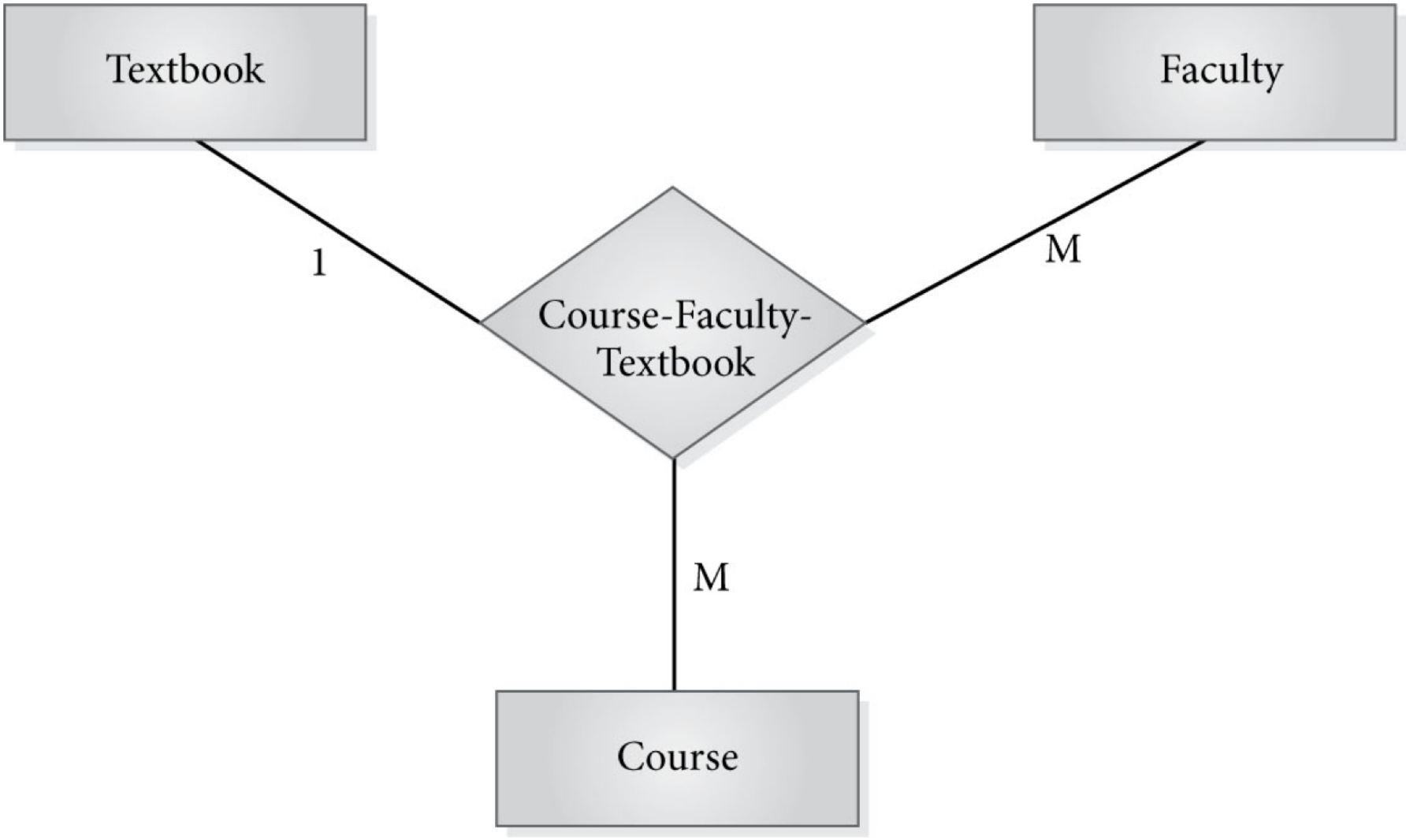


Figure 3.7 Showing Cardinalities on an E-R Diagram

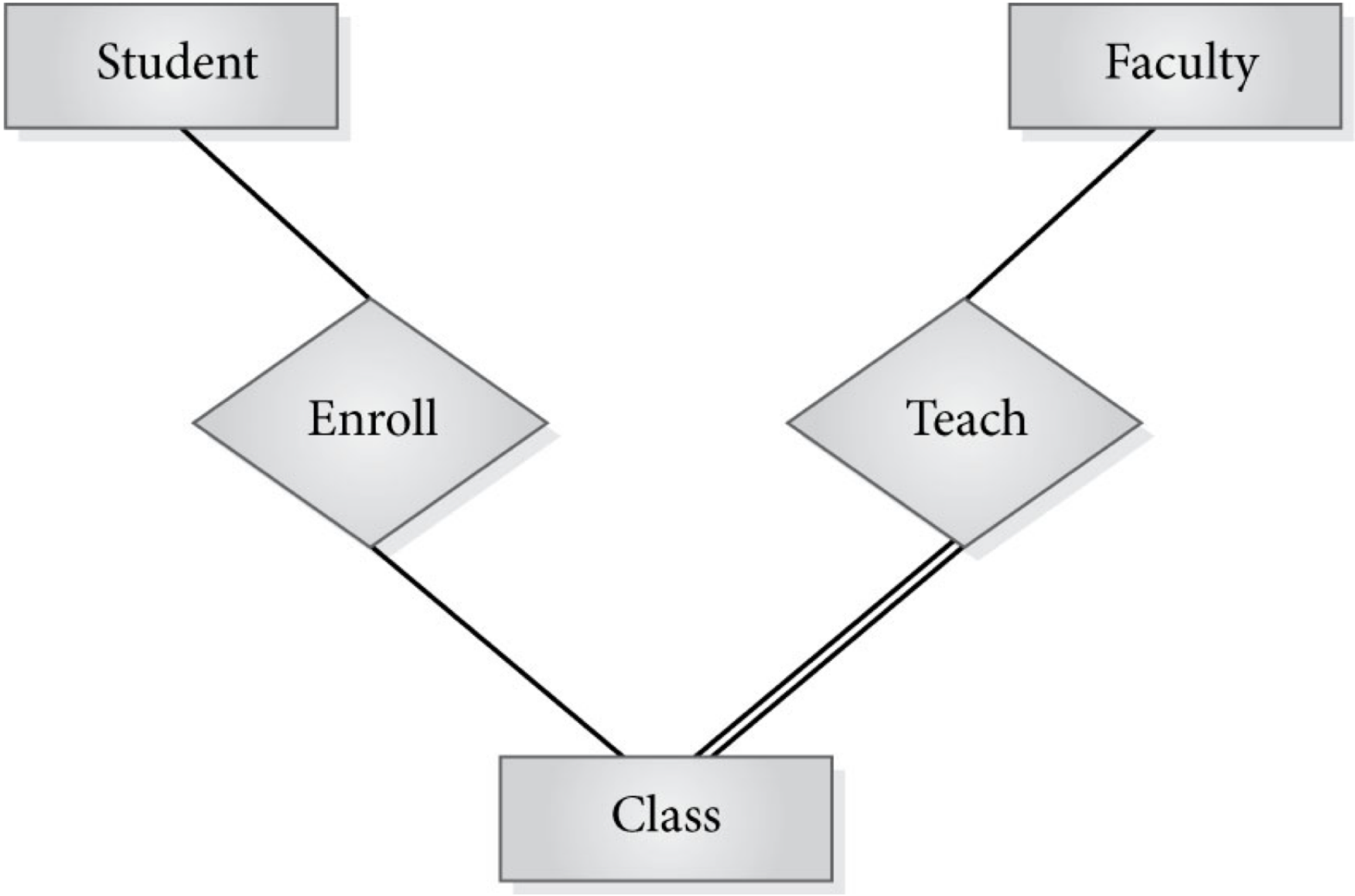
FIGURE 3.8
Cardinalities for a Ternary Relationship



Relationship Participation Constraints

- **Total participation**
  - Every member of entity set must participate in the relationship
  - Represented by double line from entity rectangle to relationship diamond
- **Partial participation**
  - Not every entity instance must participate
  - Represented by single line from entity rectangle to relationship diamond

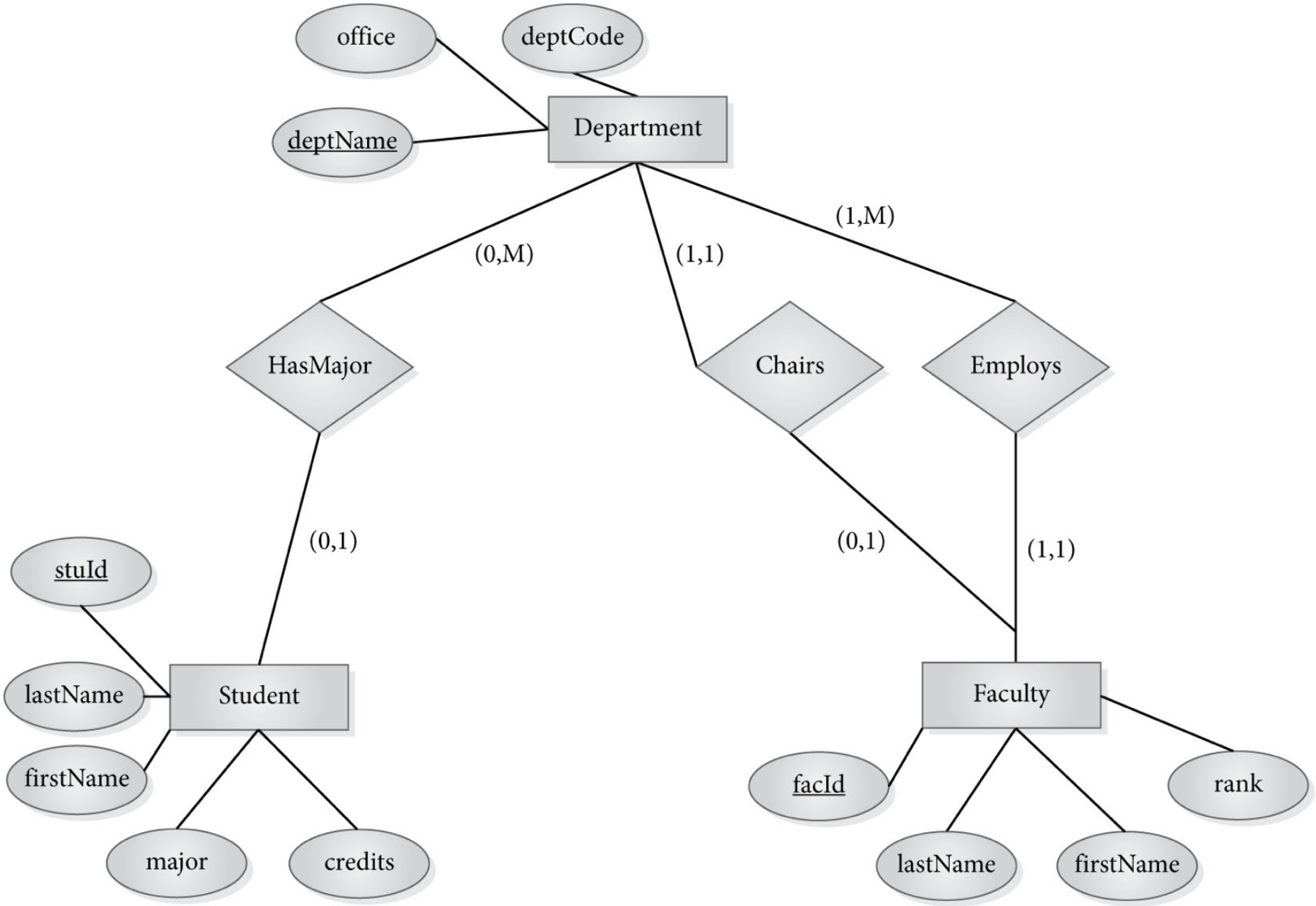
FIGURE 3.9
Total vs. Partial Participation in a Relationship



(min, max) Notation for Cardinality and Participation

- **Min**
  - least number of relationship instances an entity instance **must** participate in
  - can be 0 (partial participation)
  - or 1 or more (total participation)
- **Max**
  - greatest number of relationship instances the entity **can** participate in
  - can be 1
  - or many (written M,N, or *)
  - or some constant integer
- Written on line connecting entity rectangle to relationship diamond

FIGURE 3.10
Part of the `University` E-R Diagram with (min, max) Notation



Roles

- **Role**: function that an entity plays in a relationship
- Optional to name role of each entity, but helpful in cases of
  - Recursive relationship – entity set relates to itself
  - Multiple relationships between same entity sets

FIGURE 3.11(A)
Recursive Relationship Showing Roles

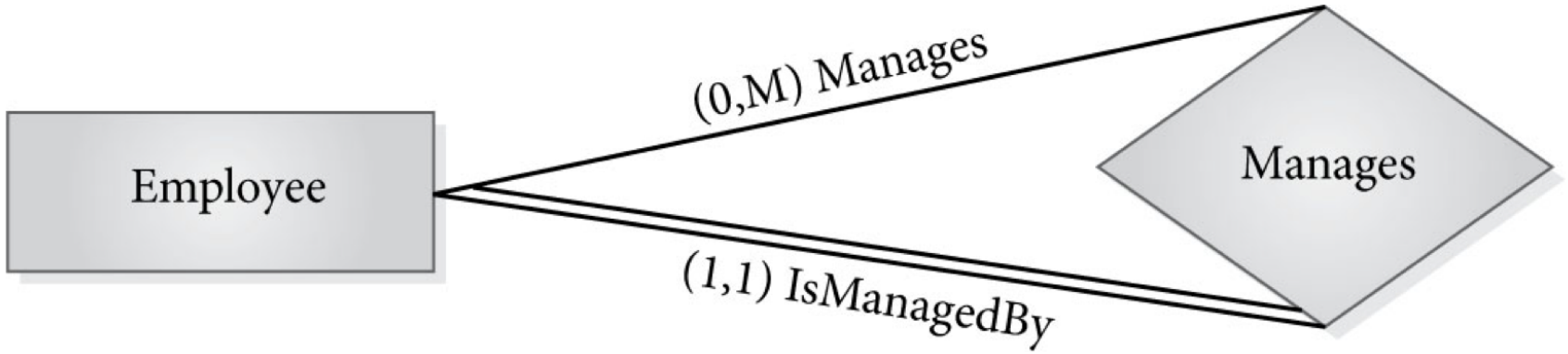
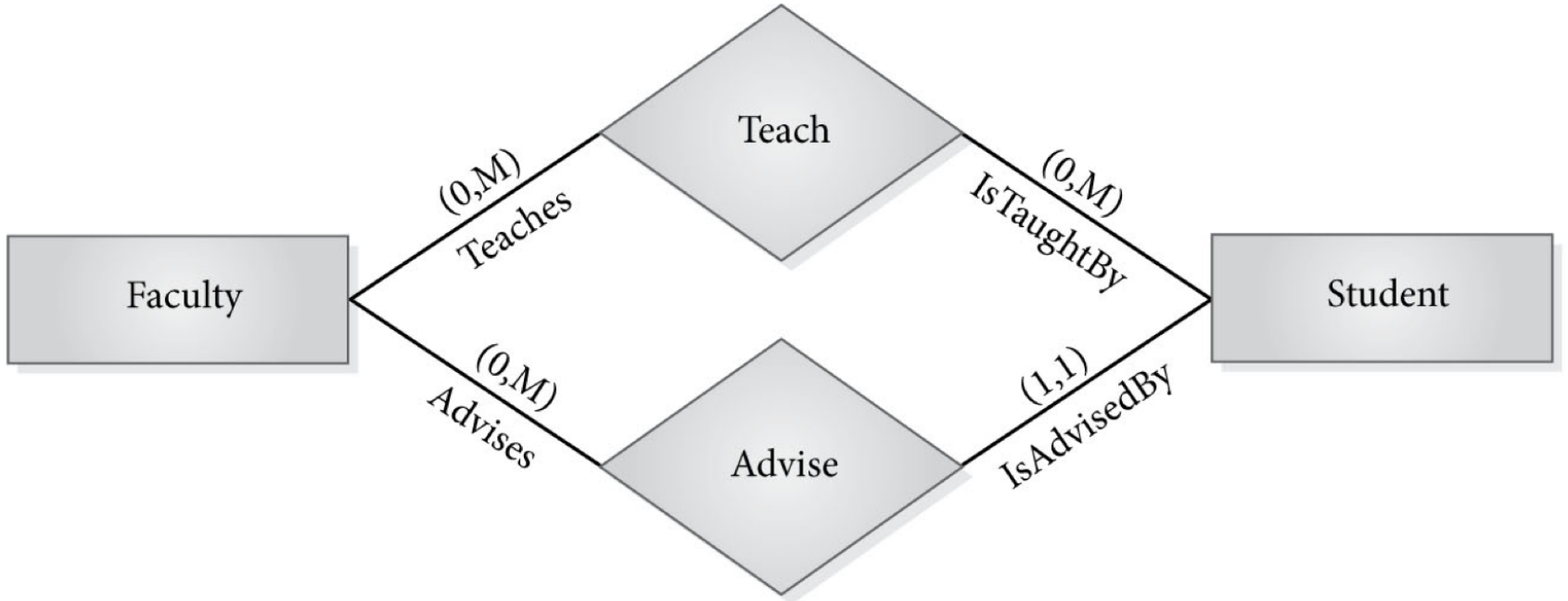


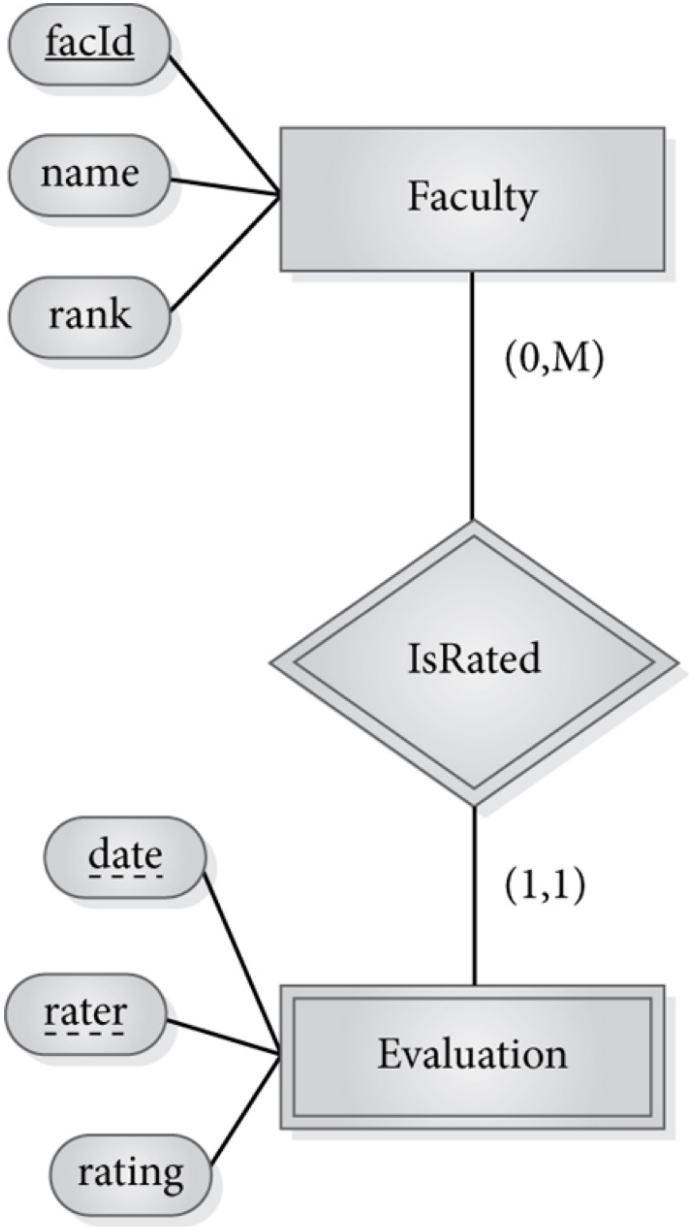
FIGURE 3.11(B)
Entity Sets with Two Relationships Showing Roles



Existence Dependency and Weak Entities

- Entity Y is **existence dependent** on entity X if each instance of Y must have a corresponding instance of X
- Y must have **total participation** in its relationship with X
- **If Y does not have its own candidate key**, Y is called a **weak entity**, and X is **strong entity**
- Weak entity may have a partial key, a **discriminator**, that distinguishes instances of the weak entity that are related to the same strong entity
  - The discriminator is indicated by a dashed underline on the E-R diagram.
- Use double rectangle for weak entity, with double diamond for relationship connecting it to its strong entity
- **Note: *Not all existence dependent entities are weak – the lack of a key is essential to definition of weakness***

FIGURE 3.12
Weak Entity Set

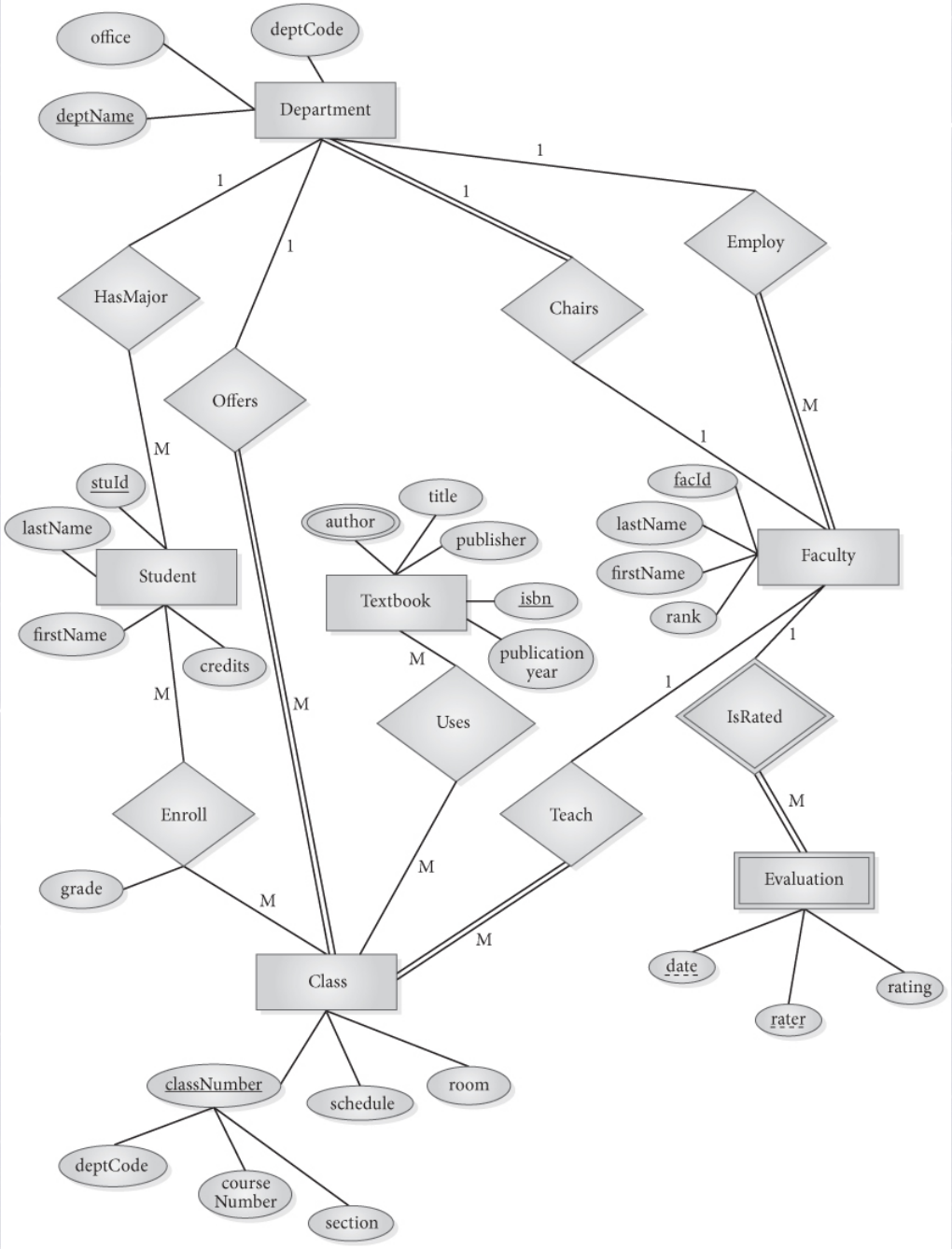


E-R Diagram Examples

Figure 3.13

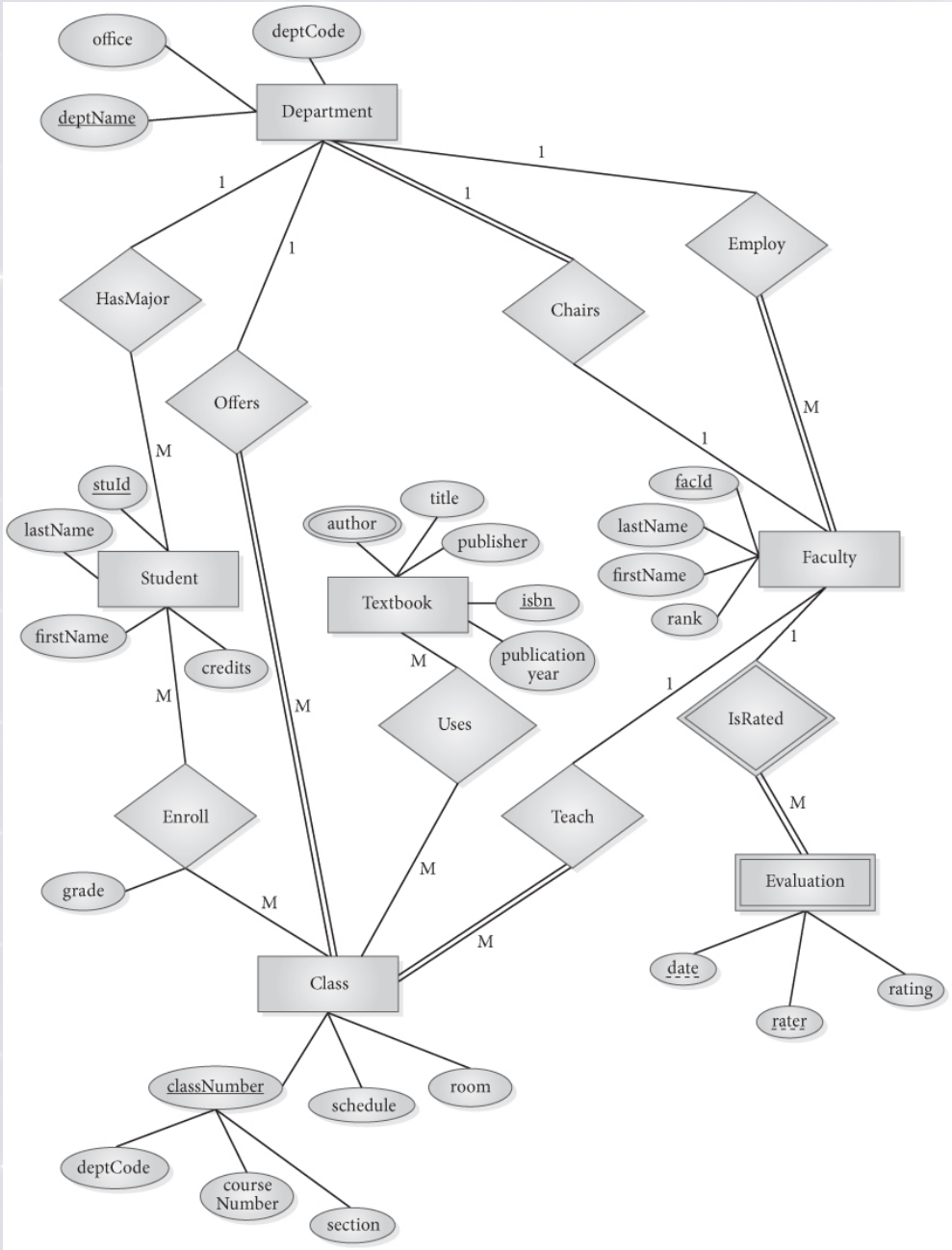
An E-R Diagram:

For this example, we will use the traditional method of showing cardinality and participation constraints.



STEPS:

- Our first task is to identify entities.
- Note that we do not make the enterprise one of the entities, so we will have no entity set called University, since this is the entire enterprise.
- Everything in the diagram represents some facet of the university.
- A good way to identify entities is to look for nouns in the scenario that may be entities.
- The nouns in the scenario include *students, classes, faculty, textbooks, departments*, and *evaluations*.
- For each of these potential entities, we choose attributes that describe them, making assumptions as we record them. In reality, we would consult with users before making these assumptions.



- The initial entities and attributes are:

Student: stuId, lastName, firstName, major, credits

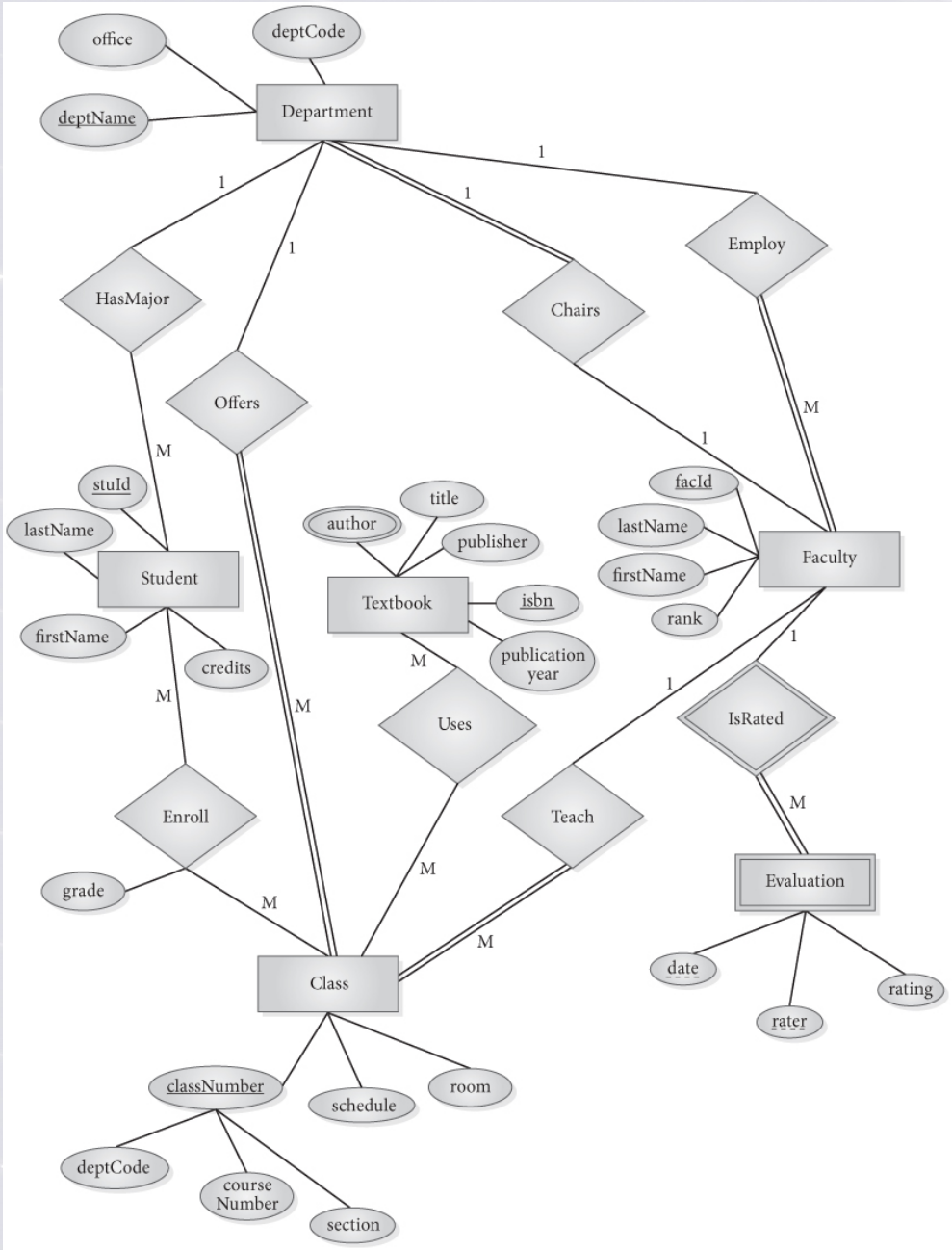
- We are assuming each student has a unique ID and has at most one major.

Department: deptCode, deptName, office

- We are assuming that each department has a unique code and a unique name, and that each department has one office designated as the departmental office.

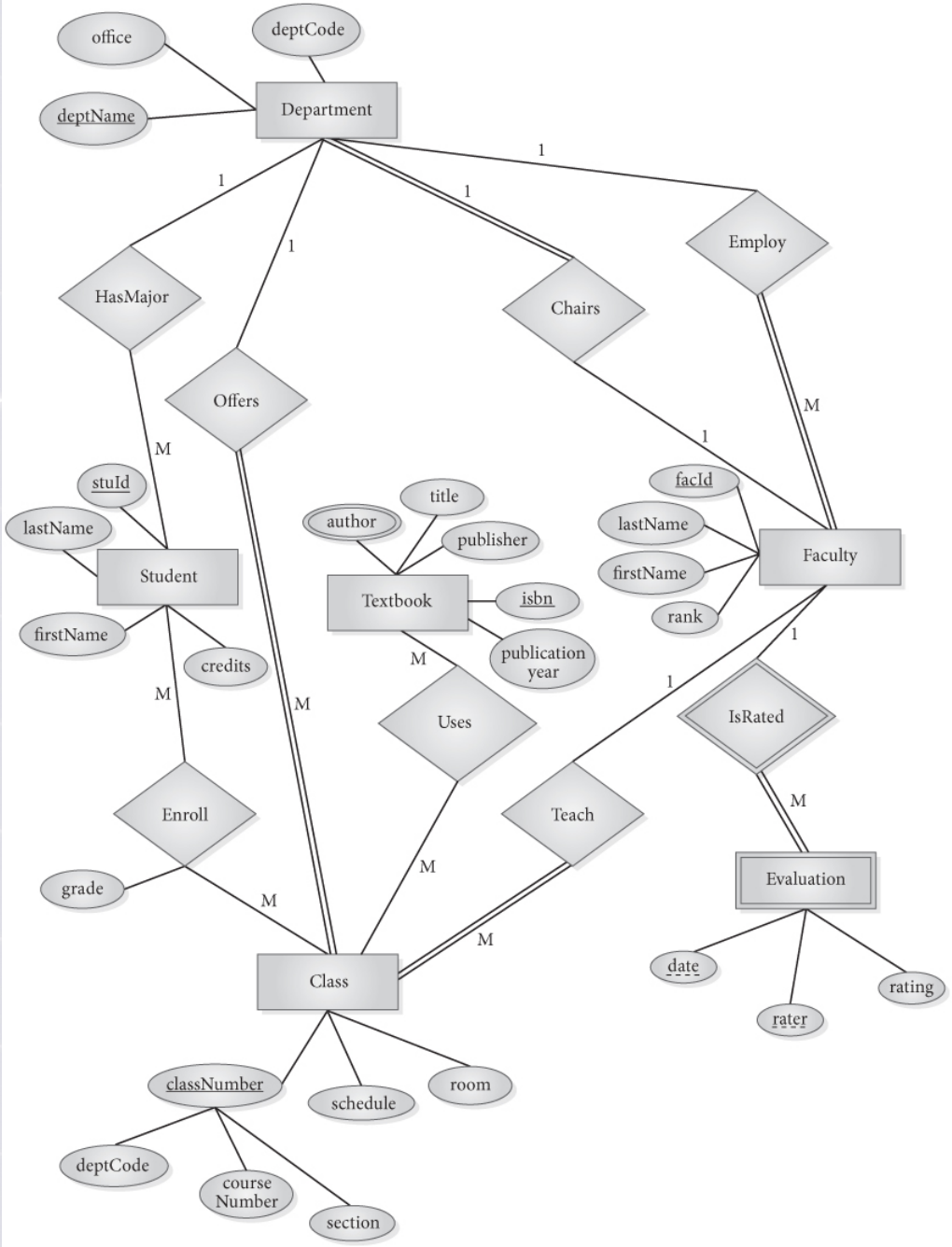
Faculty: facId, lastName, firstName, rank , department

- We are assuming that facId is unique and that every faculty member must belong to a department. One faculty member in each department is the chairperson.



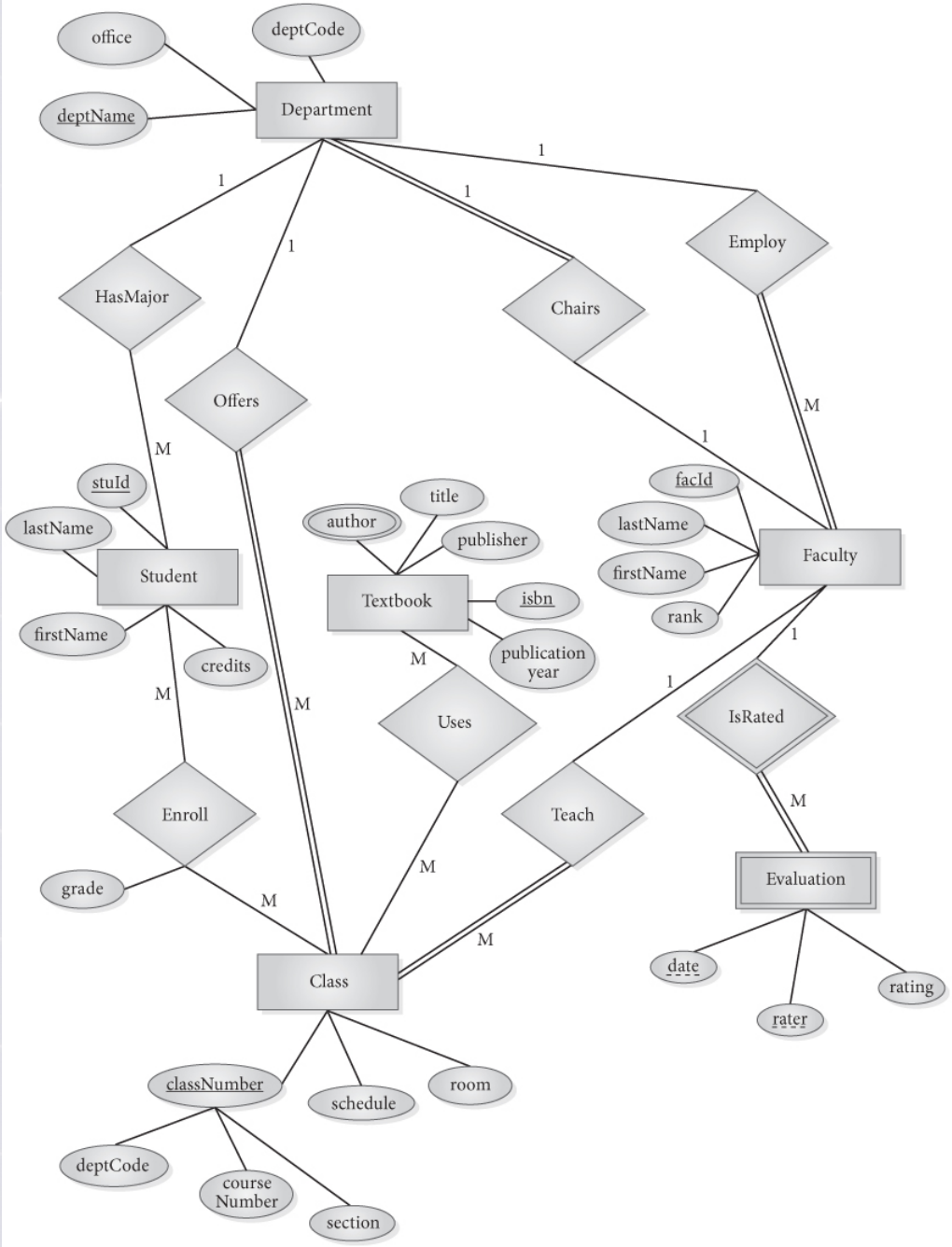
STEPS:

- First, we draw rectangles for each of the entities and connecting ovals for their attributes.
- Next, we identify relationships. These are usually indicated in the scenario by verbs that describe some type of interaction between the entities.
- Verbs in the scenario include *enroll, teach, employ, uses*, and *chairs*, so we try out these relationships, drawing diamonds and connecting the entities to them. Then, we decide on cardinality and participation constraints for each relationship. The relationship sets so far are:



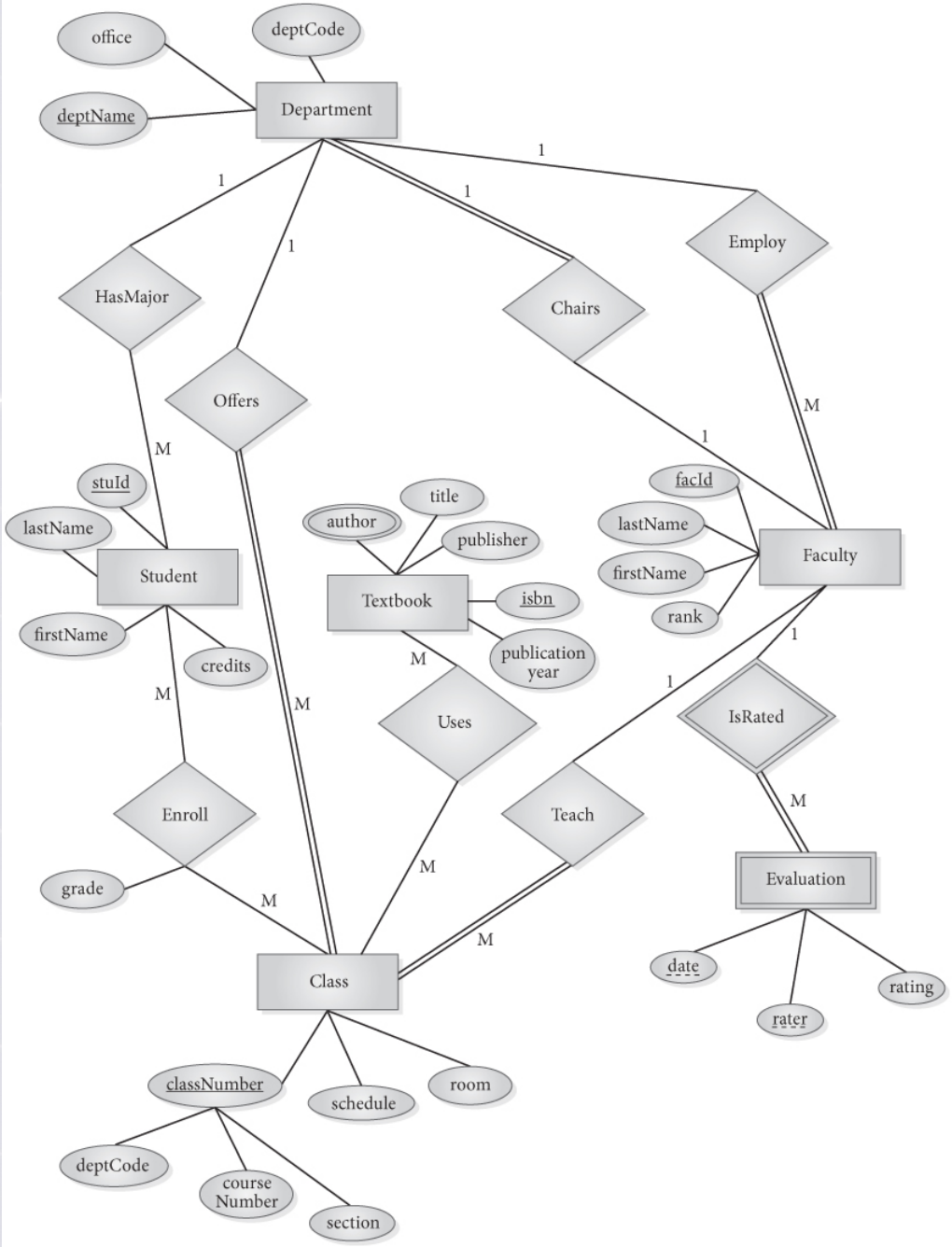
Enroll

- A many-to-many relationship that connects students to the classes in which they are enrolled.
- We are assuming only current enrollments are kept.
- We show grade as a descriptive attribute for this relationship set.
- Since the students are still enrolled in the classes, this attribute represents a midterm grade.
- Some students are not enrolled in any class, and some class offerings may have no students enrolled.



Employ

- A one-to-many relationship that connects departments to faculty members assigned to them.
- Faculty members must belong to exactly one department.
- A department might or might not have faculty assigned to it, but it can also have many faculty.
- Because we are representing Department as an entity in the diagram, we do not need the department attribute for faculty members.
- The Employ relationship already describes their department.

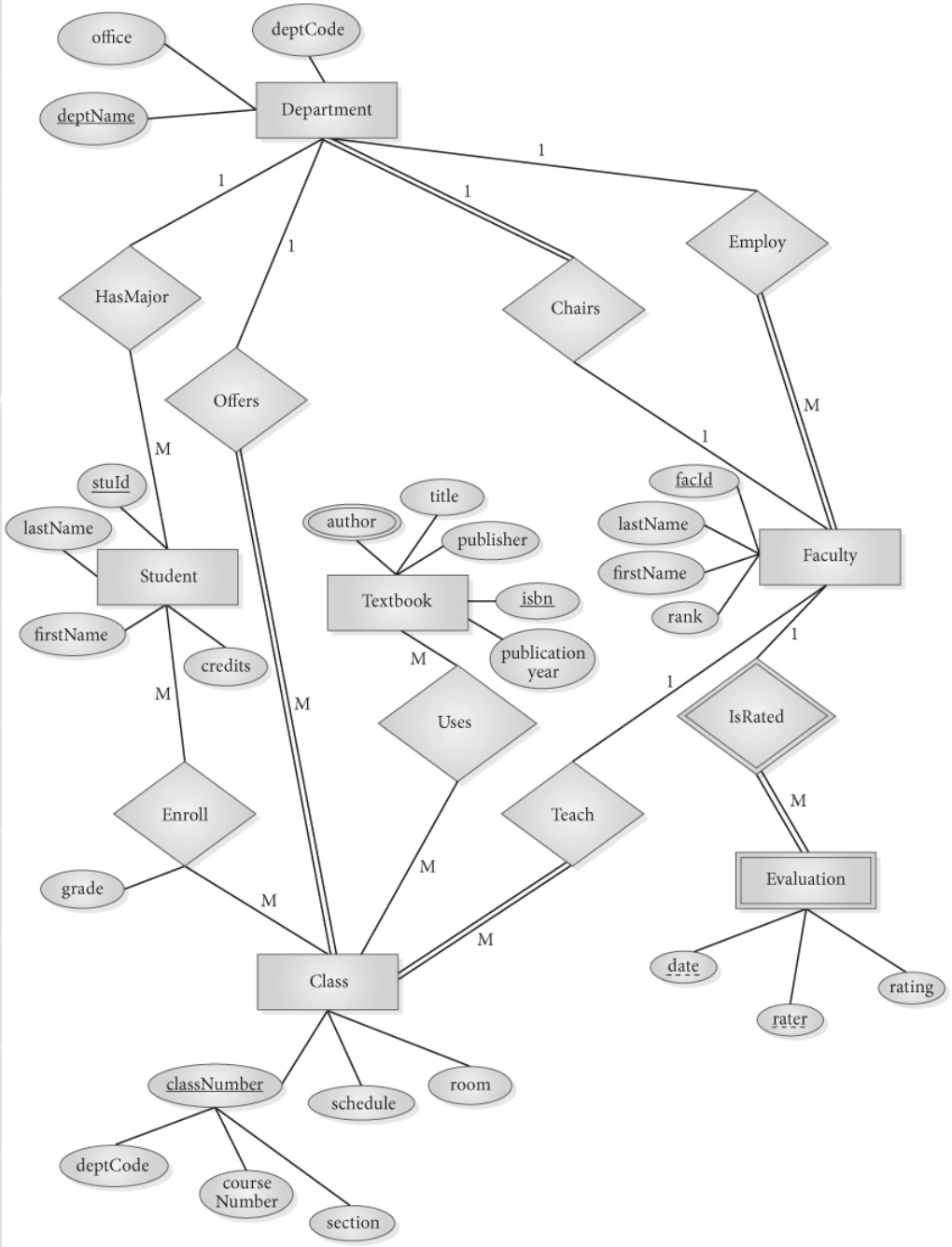


Teach

- A one-to-many relationship that connects faculty members to the classes they teach.
- We assume a faculty member may teach zero or many classes, and each class has exactly one faculty member assigned to it.

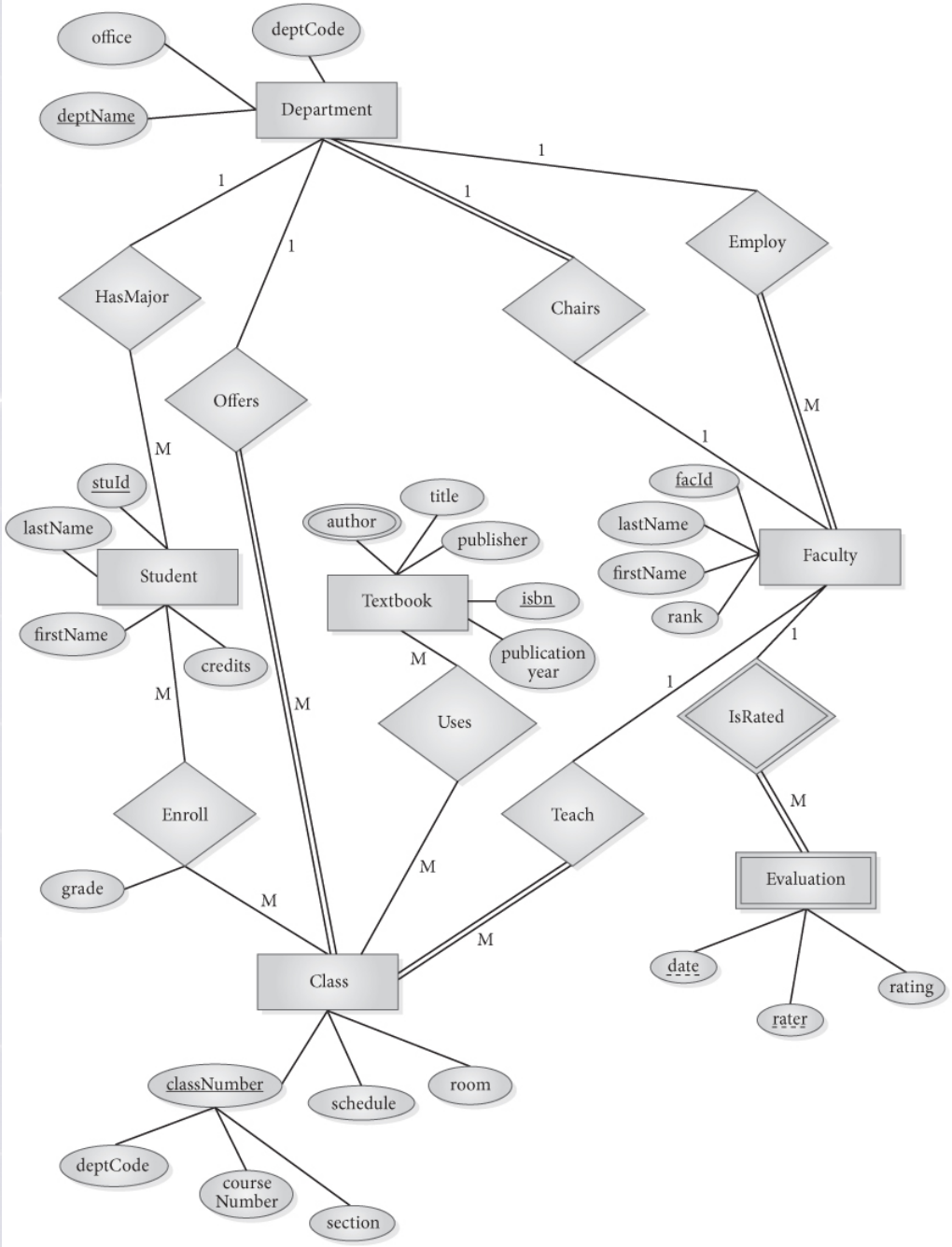
IsRated

- A one-to-many relationship between Faculty and the weak entity, Evaluation, which has no primary key of its own.
- Not every faculty member has a rating, but every rating must belong to a faculty member.



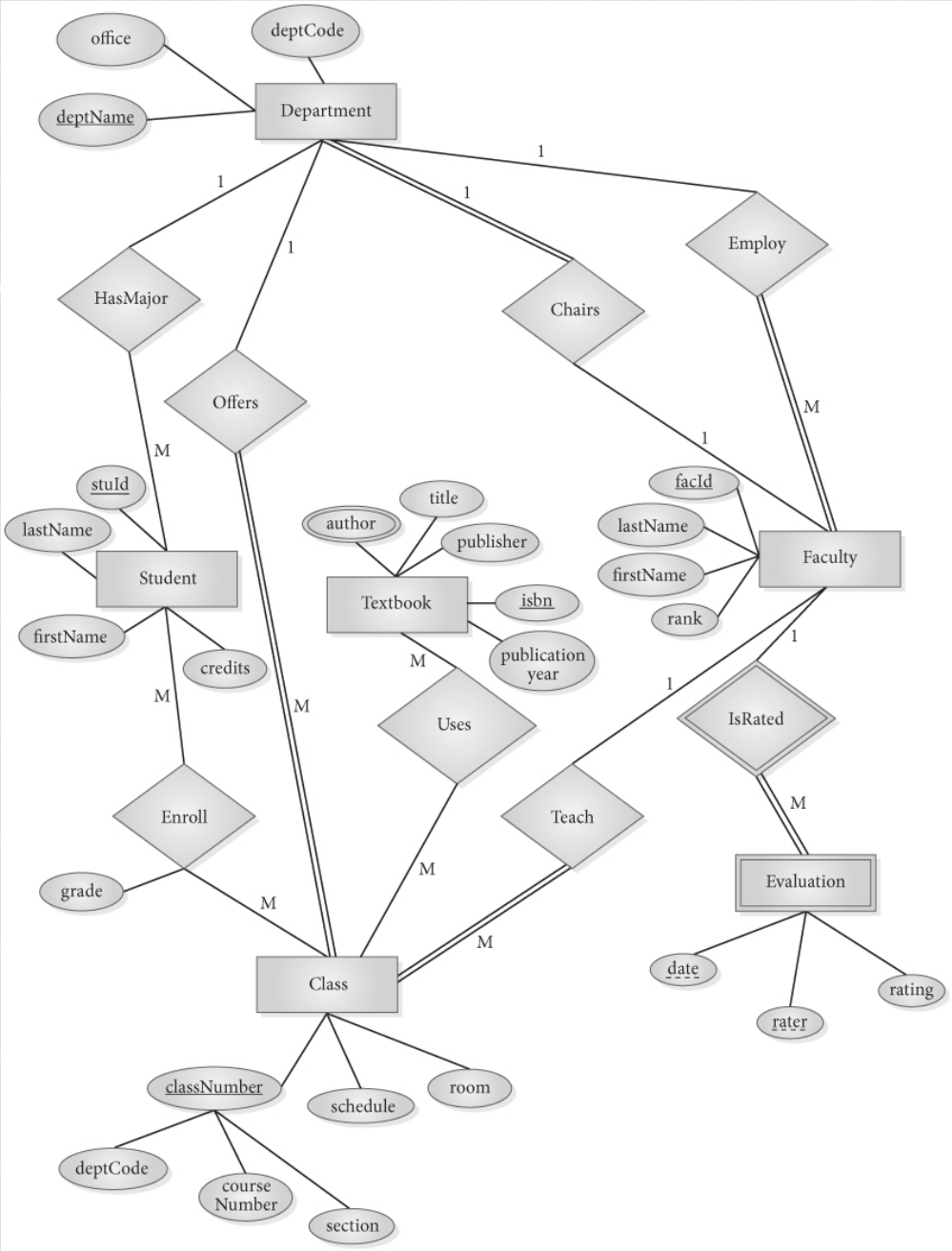
Uses

- A binary relationship between Class and Textbook.
- Note that we have changed our assumptions from our earlier example called Course-Faculty-Text.
- Here we are showing Class, not Course.
- Class includes the section, and since there is exactly one faculty member for the section, we already have the connection to Faculty.
- Note also that there may be several textbooks for the class, and the same text may be used for different classes.

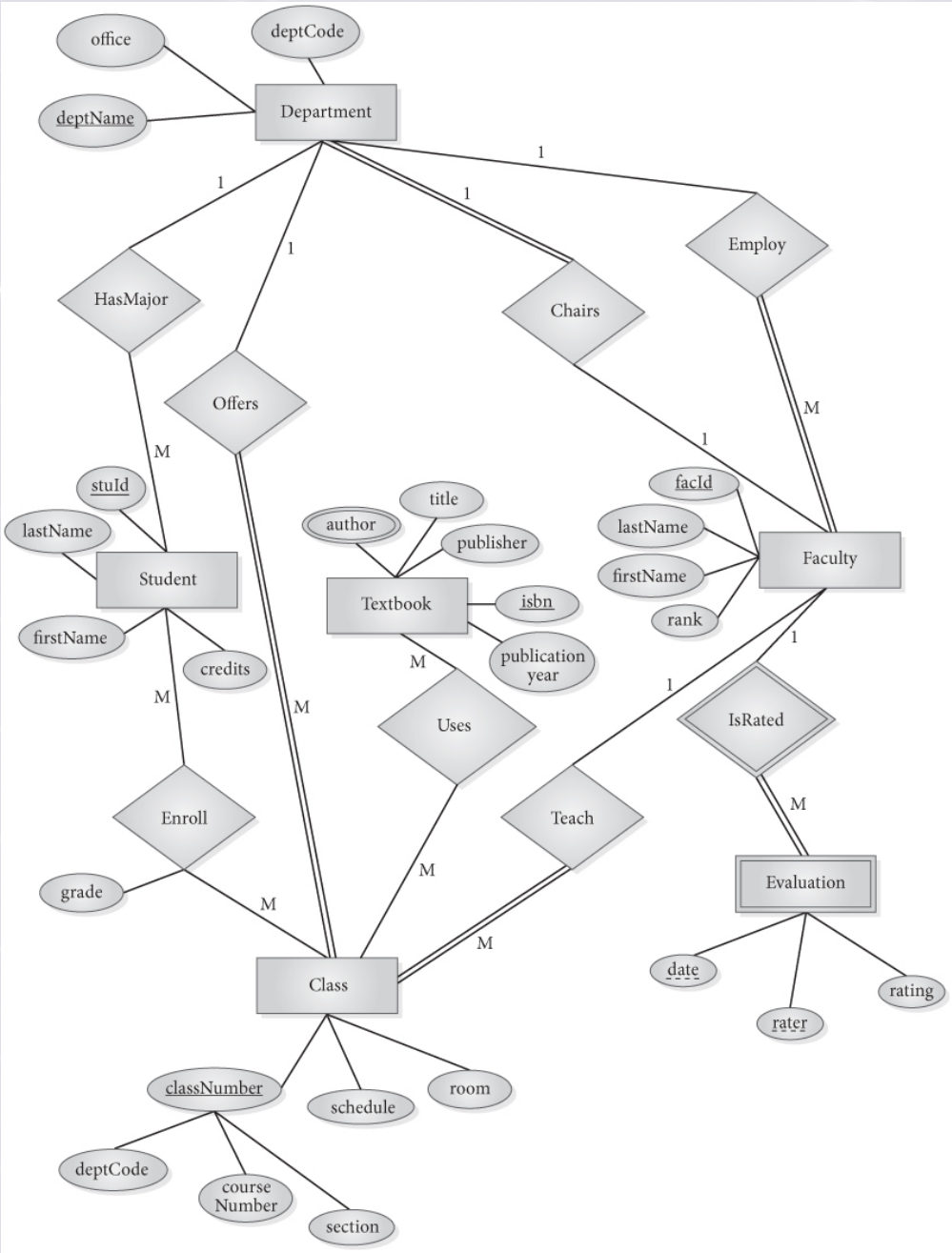


HasMajor

- A one-to-many relationship that connects students to their major departments.
- We are assuming students have at most one major.
- Not every department has majors, and not every student has declared a major.
- On closer examination, we also see that we do not need the major attribute for students, because their connection to Department also tells us the major.
- Note that we are assuming that the values of major are identical to the department name. We drop the major attribute and create a relationship.



- We also note that the Class entity is related to the Department entity because each class is offered by a department.
- We add the relationship Offers, which is a one-to-many relationship that connects departments to the classes they offer.
- Only offerings for the current semester are kept in this database.
- A department may have no class offerings this semester, but every class has a department that offers it.



- If we want to show which faculty member in each department is the chairperson, one way to do so is to create a special relationship between departments and faculty to represent the chair.
- We are assuming a chairperson is a regular faculty member who has the additional responsibility of chairing the department.
- We add the relationship Chairs, which is a one-to-one relationship that connects departments to faculty.
- One faculty member in each department is the chairperson of the department.
- Every department must have a chairperson, but not every faculty member must be a chairperson.

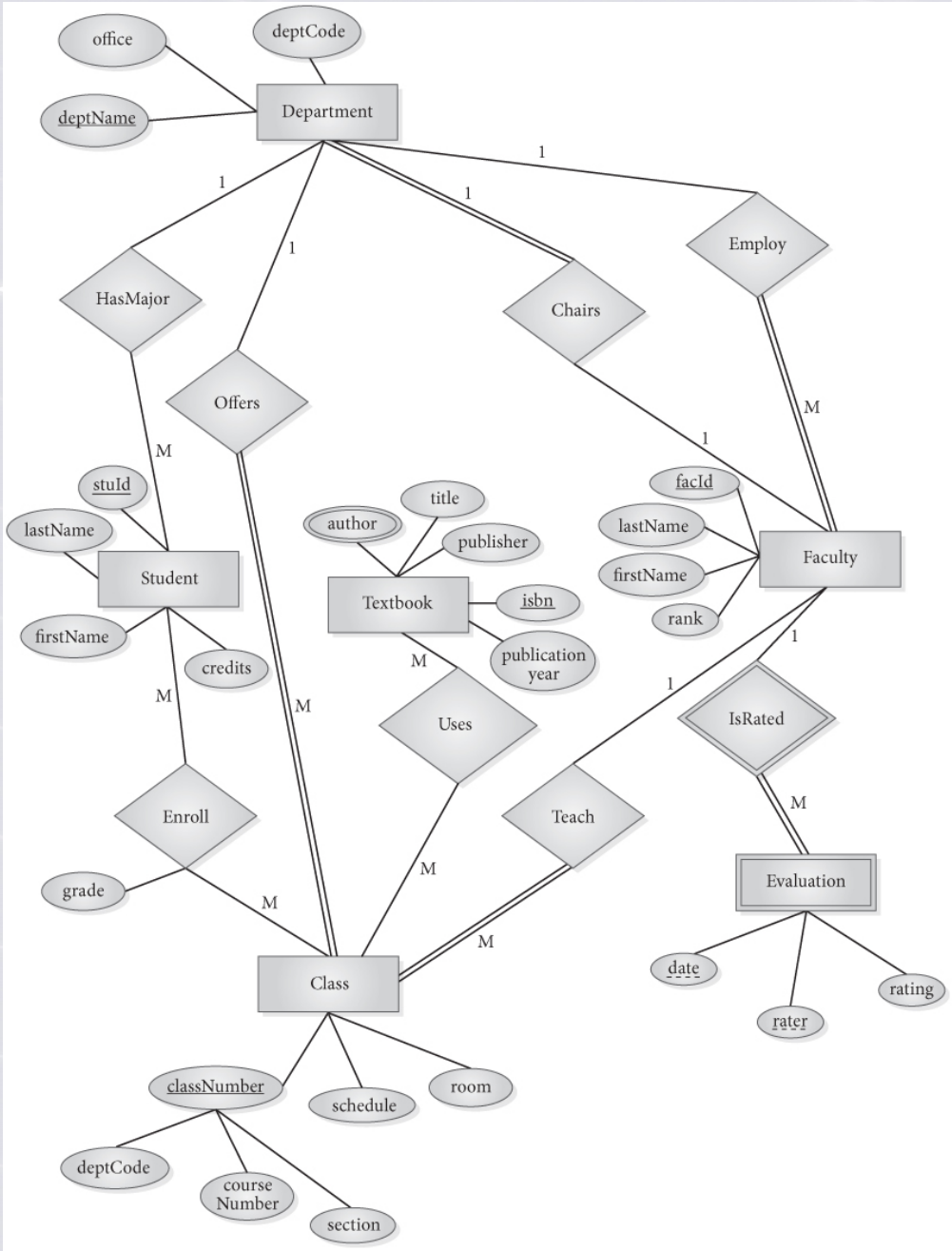
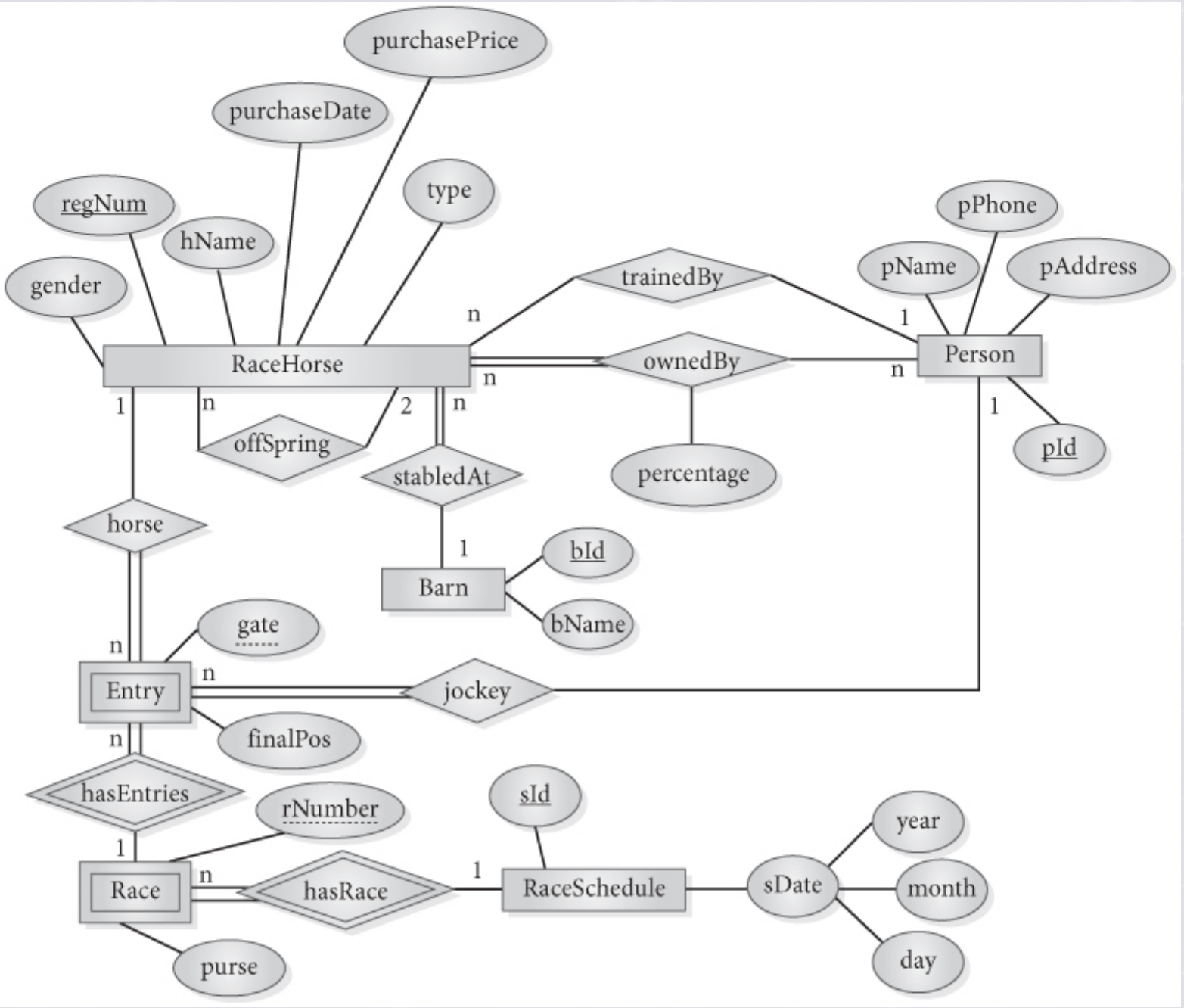
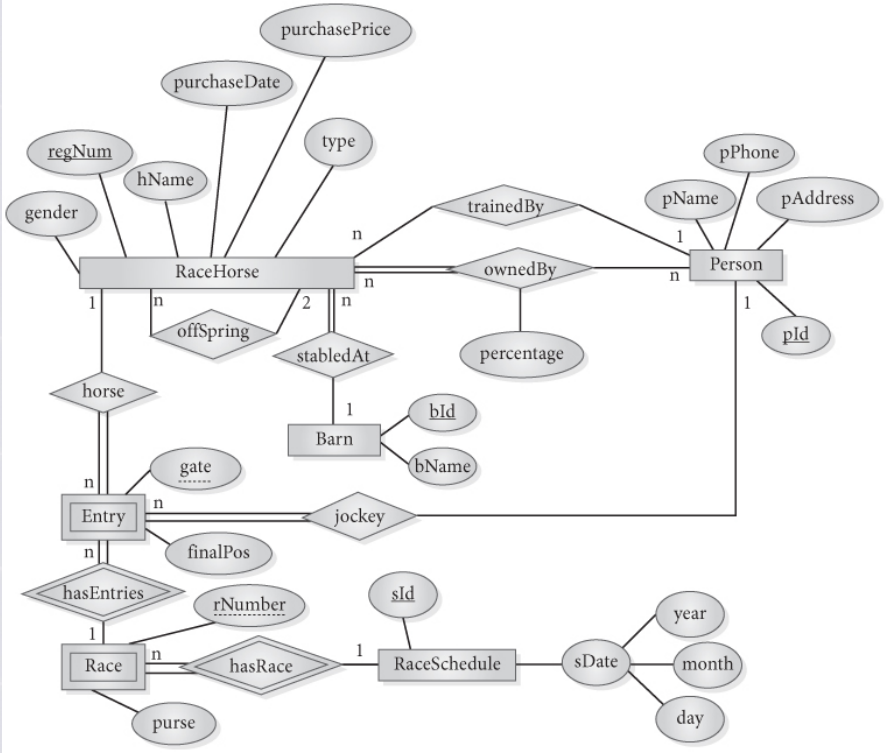


Figure 3.14 E-R Diagram for `Horse Racing` Database



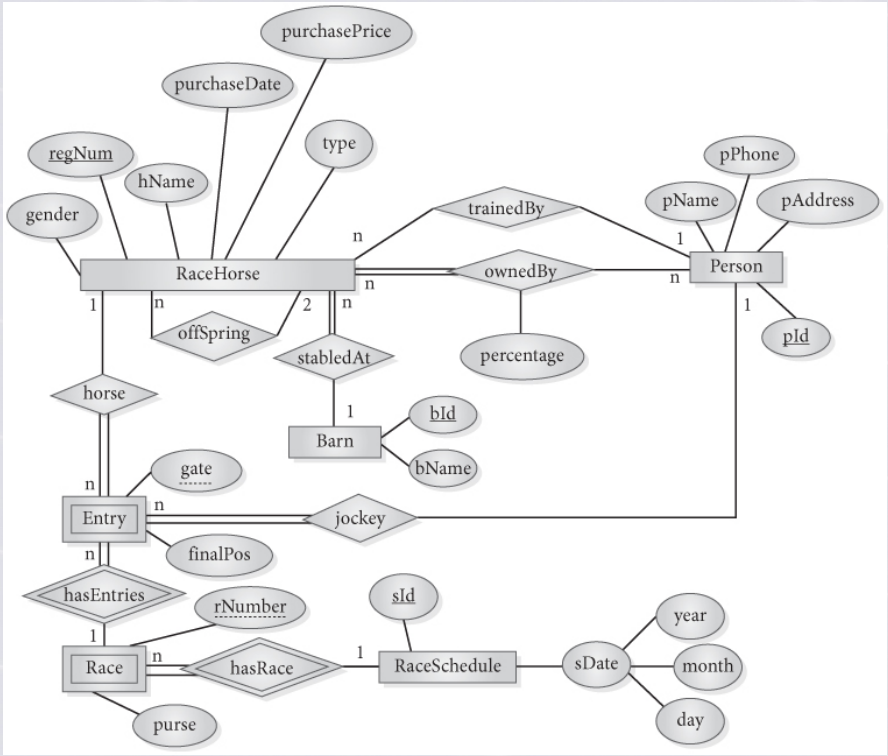
STEPS:

- We begin by trying to identify the entities for this scenario.
- We look for nouns, asking ourselves whether these are people, places, events, or concepts that are important to store information about.
- We identify *race horses, barns, people, owners, trainers, race tracks, race schedules, races*, and *race entries* as potential entities.
- We try to choose attributes for each of these as follows:



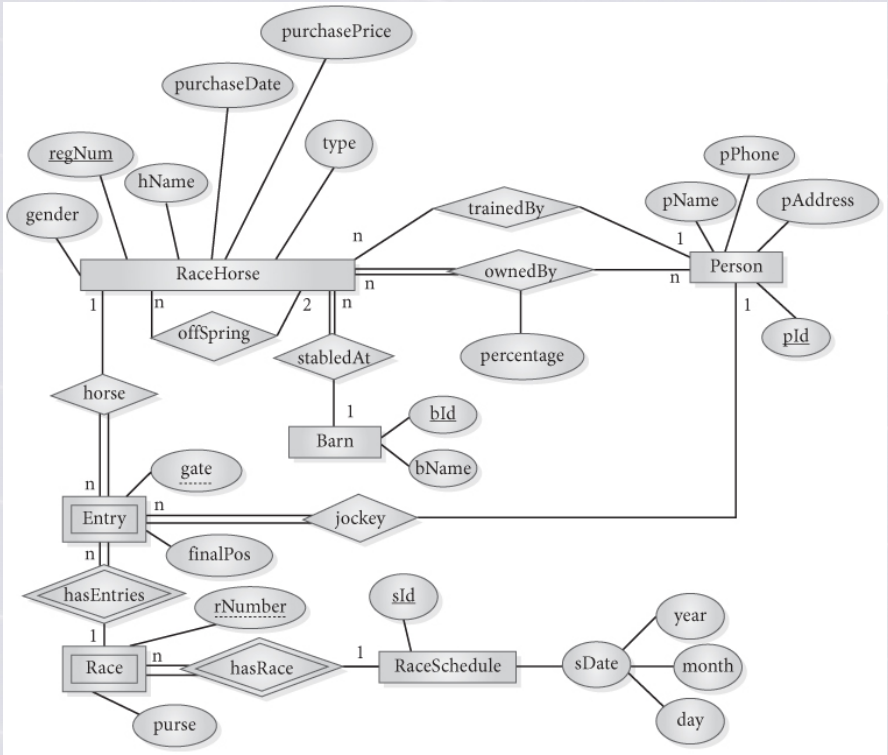
- **RaceHorse**: registrationNumber, name, type, gender, trainer, owner, purchaseDate, purchasePrice, sire, dam, offspring (multivalued)
- **Barn**: barnId, barnName
- **Person**: personId, name, address, phoneNumber
- **Trainer**: personId, name, address, phoneNumber, horsesTrained (multivalued)
- **Owner**: personId, name, address, phoneNumber, horsesOwned (multivalued), percentage (multivalued)
- **Jockey**: personId, name, address, phoneNumber, horsesRidden (multivalued)
- **RaceSchedule**: sId, rNumber (multivalued), date
- **Race**: rNumber, purse
- **Entry**: horse, jockey, gate, finalPosition

- Examining this initial list of entities, we see that the Trainer, Owner, Jockey, and Person entities are very similar, so we cancategorize all of them as Person and make them one entity set.
- Instead of keeping horses as a multivalued attribute of Person, we will use a relationship between the Person entity set and the RaceHorse entity set.
- For RaceHorse, the attribute owner can be replaced by a relationship with the person who is the owner, and the attribute trainer by a relationship with the person who is the trainer.

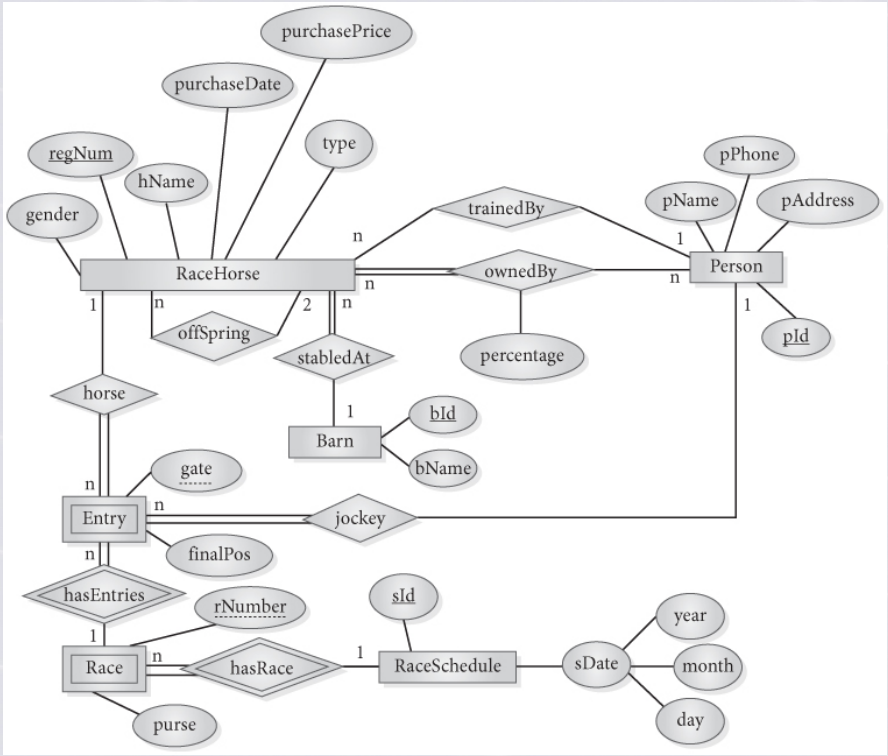


- We notice that sire, dam, and offspring are all other horses, so we will replace these attributes by a recursive relationship.
- For RaceSchedule, instead of the multivalued attribute rNumber, we use a relationship with Race.
- For Entry, the attribute horse should be replaced by a relationship with RaceHorse, and the attribute jockey replaced by a relationship with Person.
- This relationship also allows us to drop the multivalued attribute horsesRidden that we had planned for Jockey.

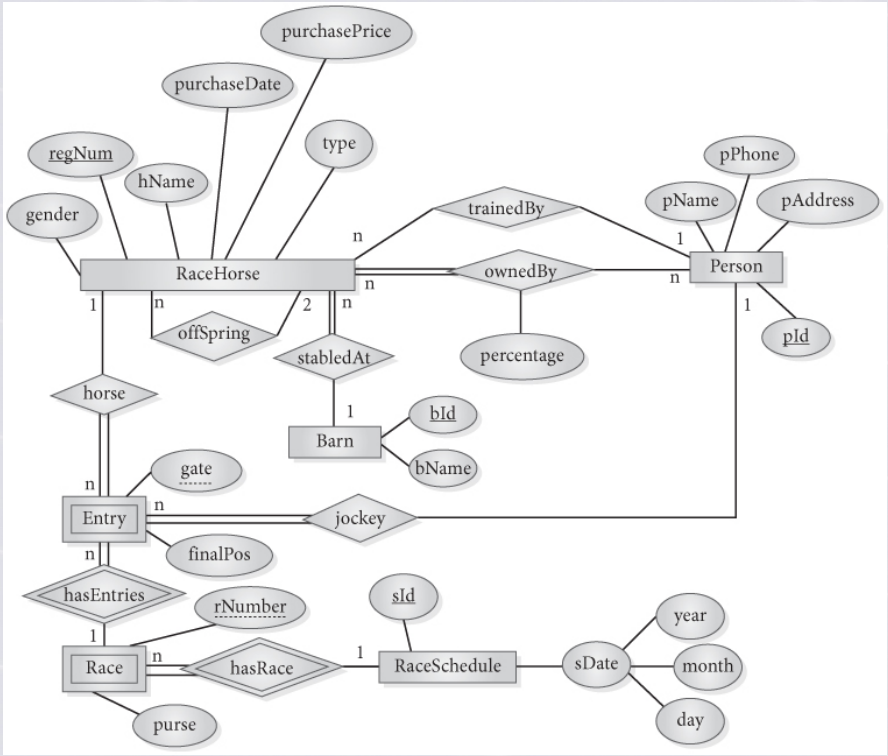
- We revise our entities to the new set:
- **RaceHorse**: regNum, hName, type, gender, purchaseDate, purchasePrice
- **Barn**: bId, bName
- **Person**: pId, pName, pAddress, pPhone
- **RaceSchedule**: sId, date, which has subattributes year, month, and day
- **Race**: rNumber, purse
- **Entry**: gate, finalPos



- We add relationships:
- **offSpring**, a recursive relationship of the entity set RaceHorse to itself.
  - Since a horse is the offspring of two horses, and may be the parent of many horses, this relationship has cardinality two-to-many.
- **trainedBy**, connecting the Person entity set to the RaceHorse entity set.
  - A person can train many horses, but each horse has exactly one trainer, so this is a one-to-many relationship.
- **ownedBy**, connecting the Person entity set to the RaceHorse entity set.
  - Since a person can own many horses and a horse can belong to more than one person, this is a many-to-many relationship.
  - percentage is a descriptive attribute of this relationship.
  - Since every horse has an owner, RaceHorse has total participation in this relationship.
- **stabledAt** is a one-to-many relationship connecting barn to RaceHorse. Since every horse has to have a stable, RaceHorse has



- **stabledAt** is a one-to-many relationship connecting barn to RaceHorse.
  - Since every horse has to have a stable, RaceHorse has total participation in this relationship.
- **hasRace** is a one-to-many relationship connecting RaceSchedule to Race.
  - We note that an individual race could not exist in the database without a corresponding RaceSchedule, so Race is a weak entity, dependent on RaceSchedule, and it has total participation in this relationship.
- **hasEntries** is a one-to-many relationship from Race to Entry.
  - Since an Entry could not exist without a Race, Entry is a weak entity, dependent on Race, and has total participation in the relationship.
- **jockey** is a one-to-many relationship connecting a Person entity to an Entry entity.
  - Entry has total participation in this relationship.
- **horse** is a one-to-many relationship connecting a RaceHorse entity to an Entry entity.
  - Entry has total participation in this relationship.



Why Extend the E-R Model?

(EER Model)

- E-R suitable for traditional business applications
- E-R not semantically rich enough for advanced applications
- Applications where E-R is inadequate
  - Geographical information systems
  - Search engines
  - Data mining
  - Multimedia
  - CAD/CAM
  - Software development
  - Engineering design
  - Financial services
  - Digital publishing
  - Telecommunications
  - ...and others

Specialization Abstraction

- Specialization-needed when an entity set has subsets that have special attributes or that participate in special relationships
- Process of breaking up a set into subsets

  Ex: `Faculty` contains `AdjunctFac` and `FullTimeFac`
  - All `Faculty` have attributes `facid`, `lastName`, `firstName`, `rank`.
  - `AdjunctFac` also have `coursePayRate`
  - `FullTimeFac` have `annualSalary`

Representing Specialization

- EE-R diagram shows specialization circle (*isa* relationship), and inheritance symbol (subset symbol)

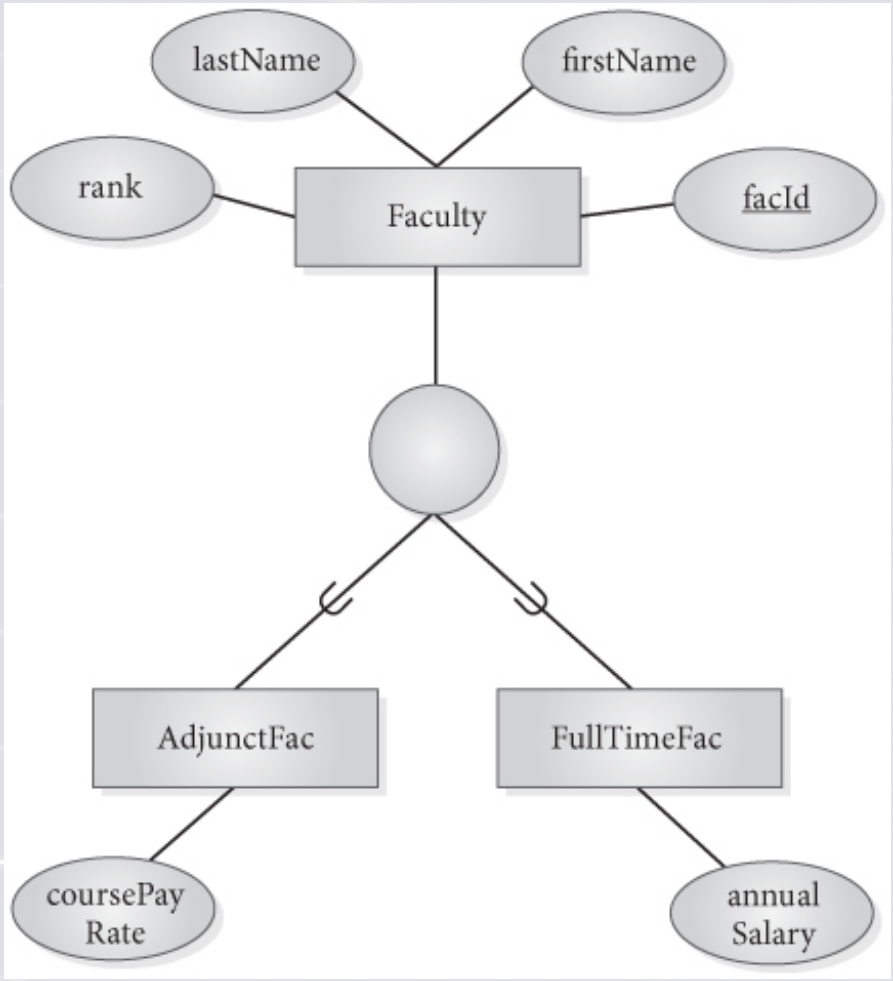


Figure 3.15A Specialization with Two Subsets

- Specialization can also involve just one subset – no need for circle, but show inheritance symbol

Ex. `Class` has `LabClass` specialization

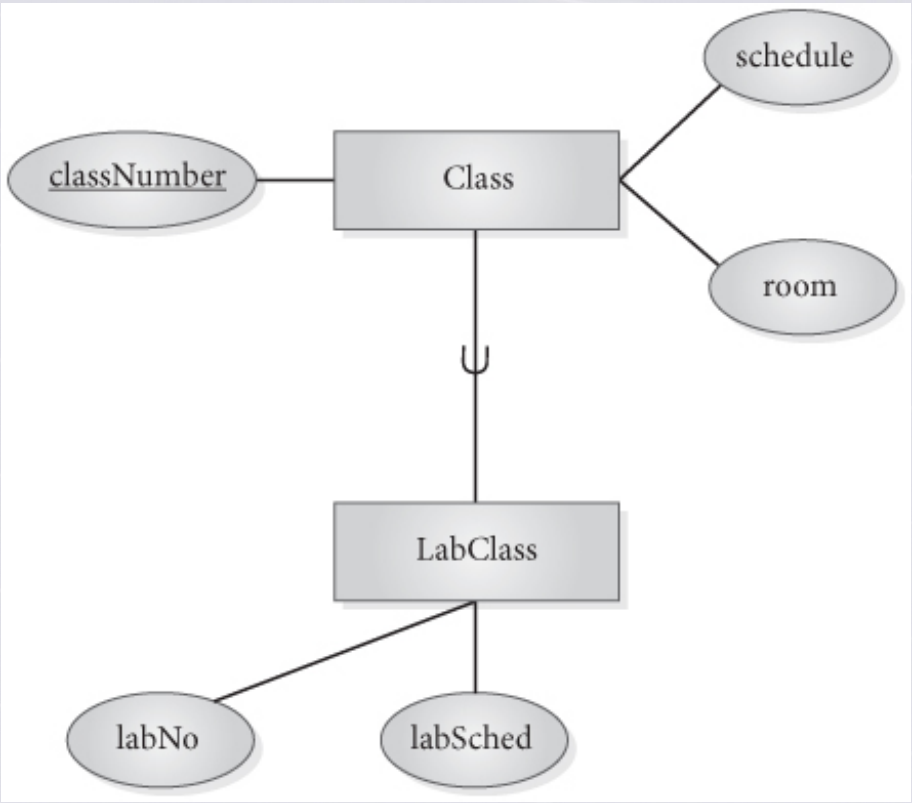


Figure 3.15B Specialization with One Subset

- Subsets can participate in their own relationships

Ex. `FullTimeFac` *`subscribes to`* `Pension`

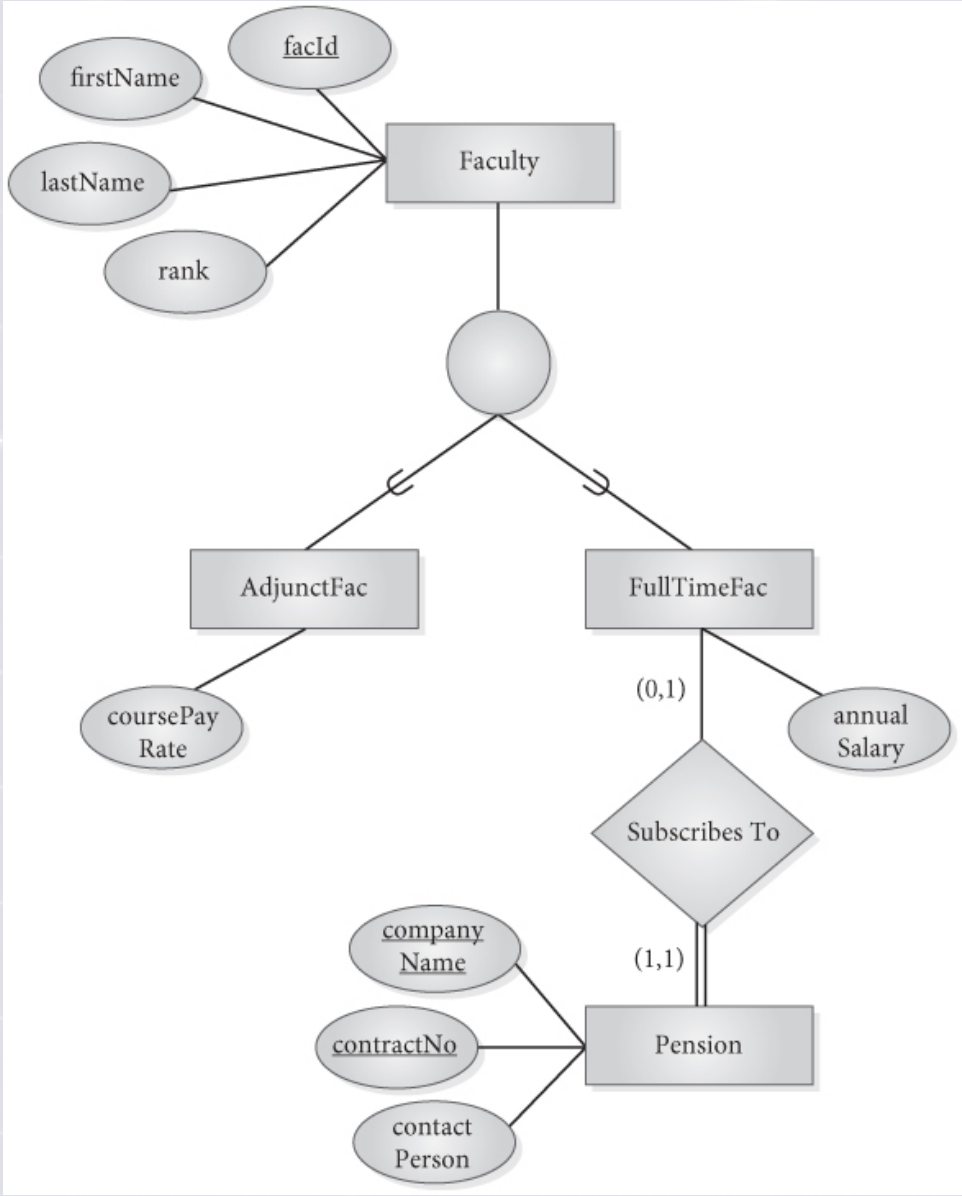


Figure 3.15C Subset with Relationship

Generalization Abstraction

- Inverse of specialization
- Recognizing that sets have common properties and identifying a superset for them
  Ex. `Student` and `Faculty` are both people
- Bottom-up process, as opposed to top-down process of specialization
- EER diagram is the same as for specialization – See Figure 3.15(d)

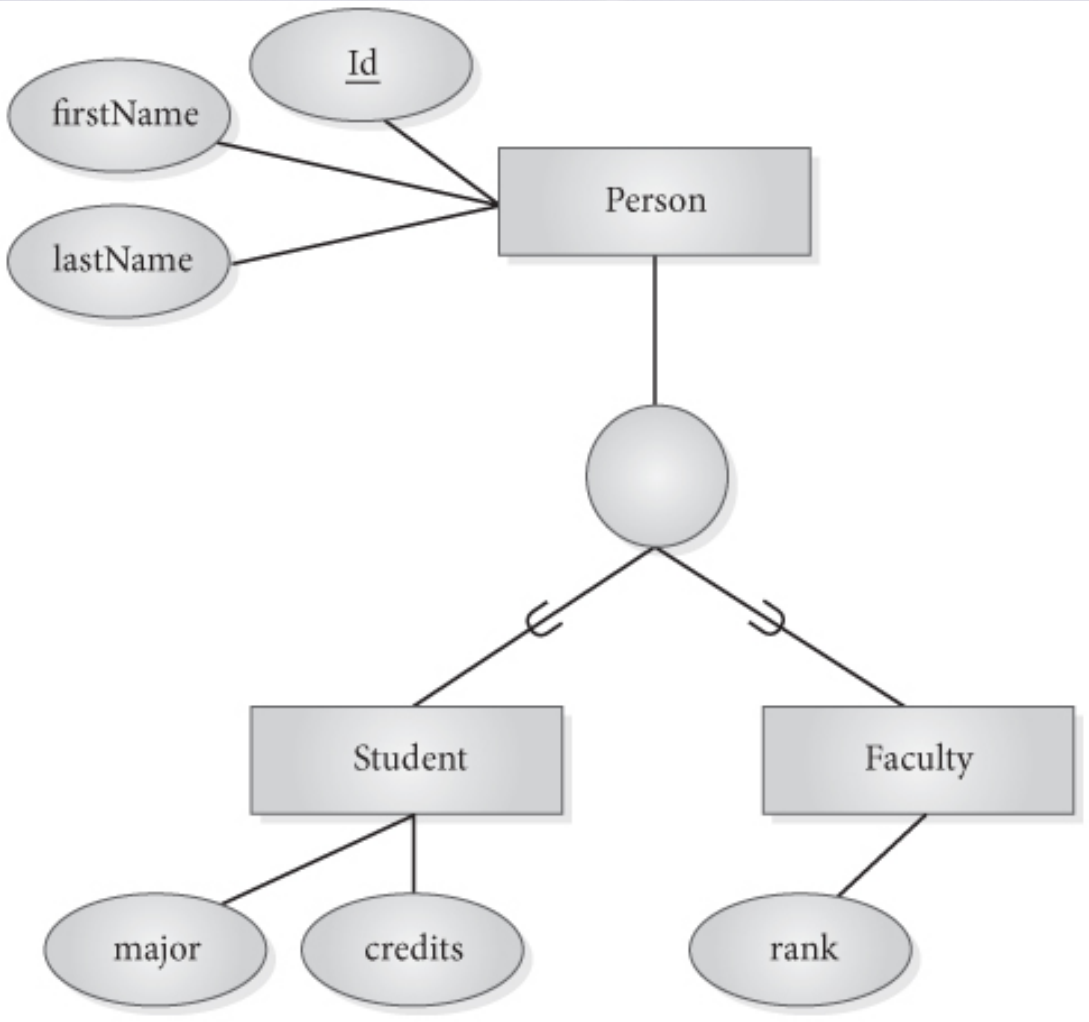


Figure 3.15D Generalization with `Person` Superset

Generalization / Specialization Constraints

- Subsets can be **overlapping** or **disjoint**
- Place **o** or **d** in specialization circle to indicate constraint
- Specialization can be **total** (every member of superset must be in some subset) or **partial**
  - **Total** - double line connecting superset to specialization circle
  - **Partial** - single line

  See Figure 3.16

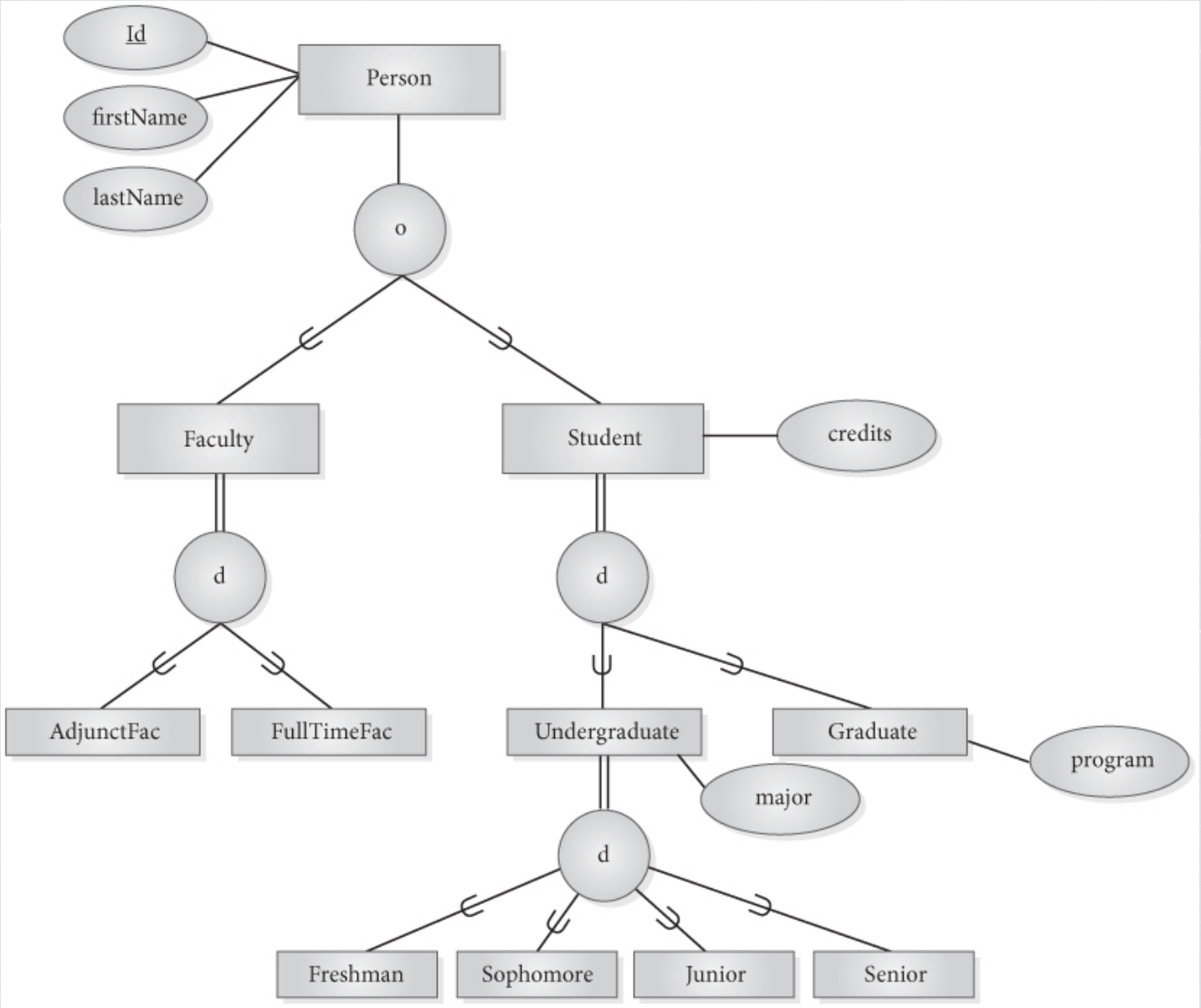


Figure 3.16 Disjointness and Participation Constraints

- Specialization definition can be
  - **predicate-defined** - each subset has a defining predicate
  - **Attribute defined** – the value of the **same attribute** is used in defining predicate for all subsets
  - **User-defined** – user responsible for identifying proper subset
    See Figure 3.17(a) and Figure 3.17(b)

Figure 3.17A Predicate-Defined Specialization

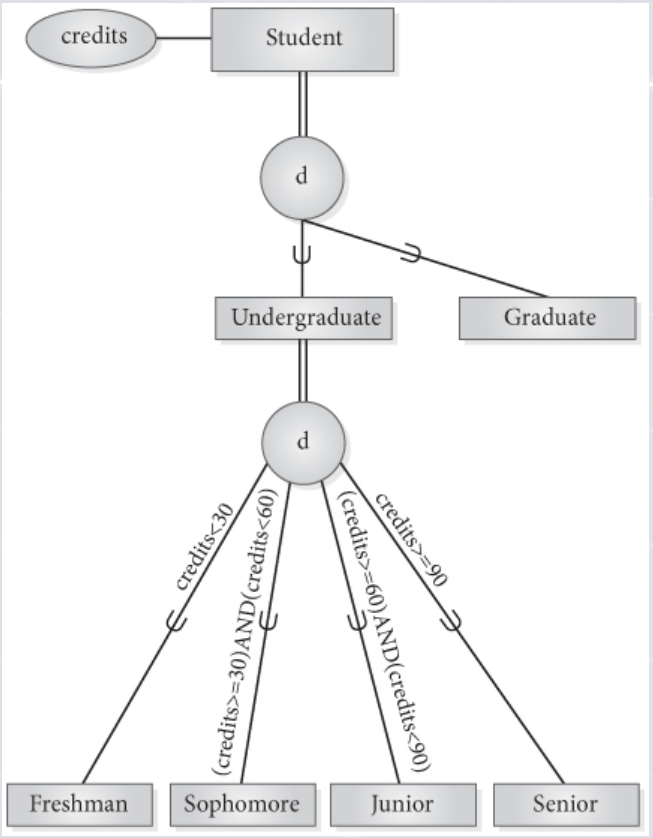
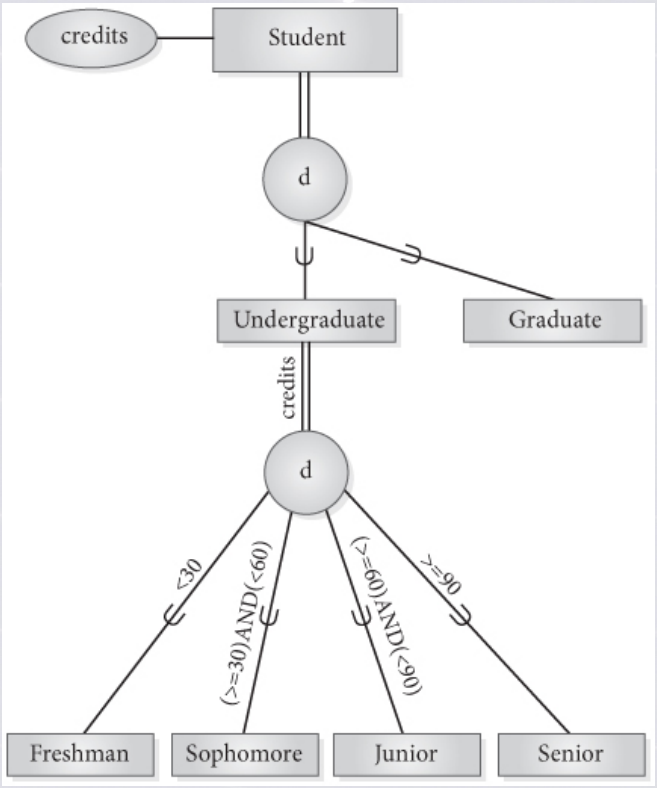


Figure 3.17B Attribute-Defined Specialization



Multiple Specializations & Multiple Inheritance

- Can have different specializations for the same set

  Ex. **Figure 3.17(c)** shows `Undergraduate` specialized by `year` and by `residence`. These are independent of each other

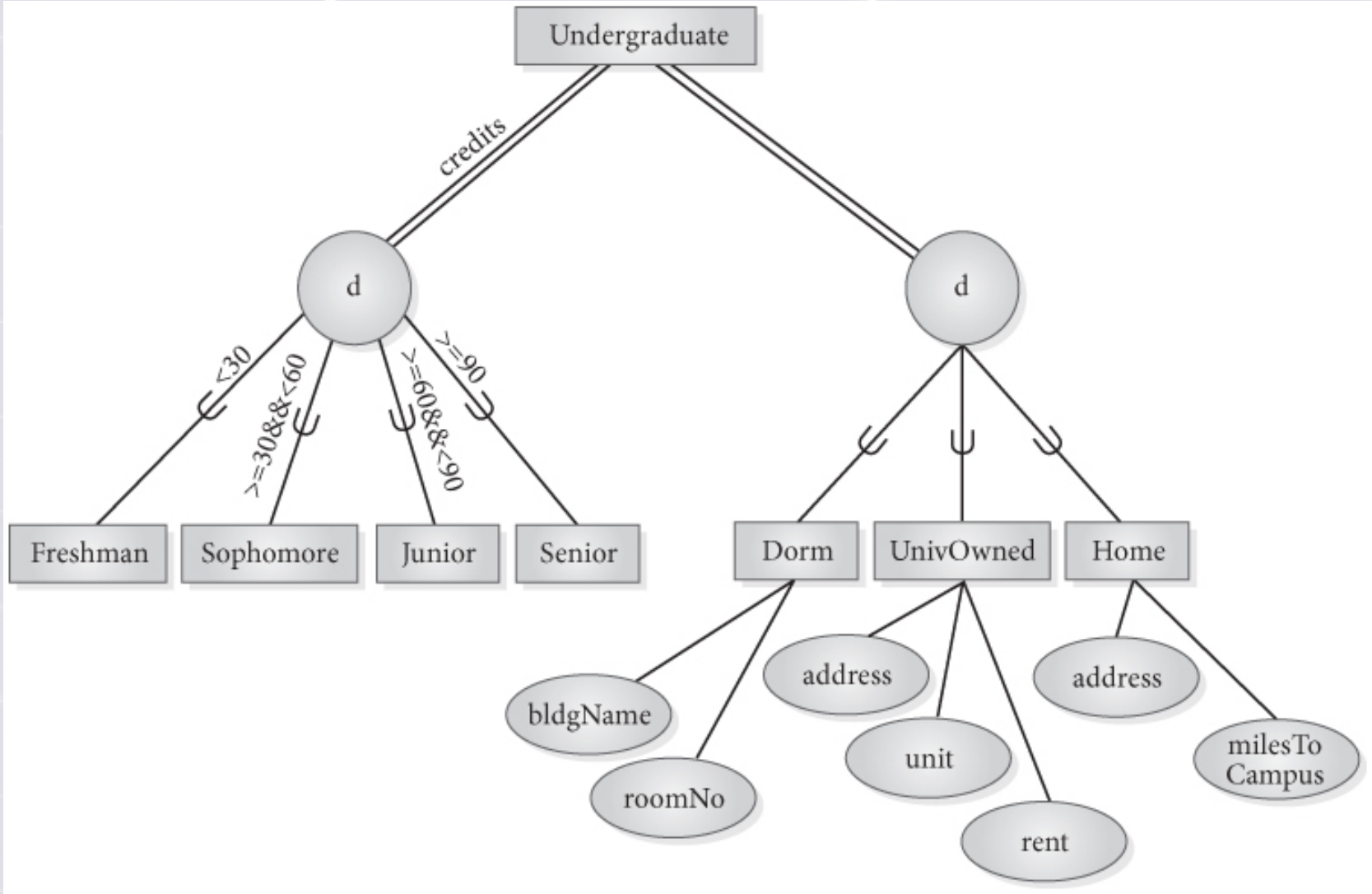


Figure 3.17C Multiple Specializations

- **Can have multiple inheritance -** two or more supersets share same subset

Ex. **Figure 3.18** shows `Teaching Assistant` as subset of both `Faculty` and `Student`

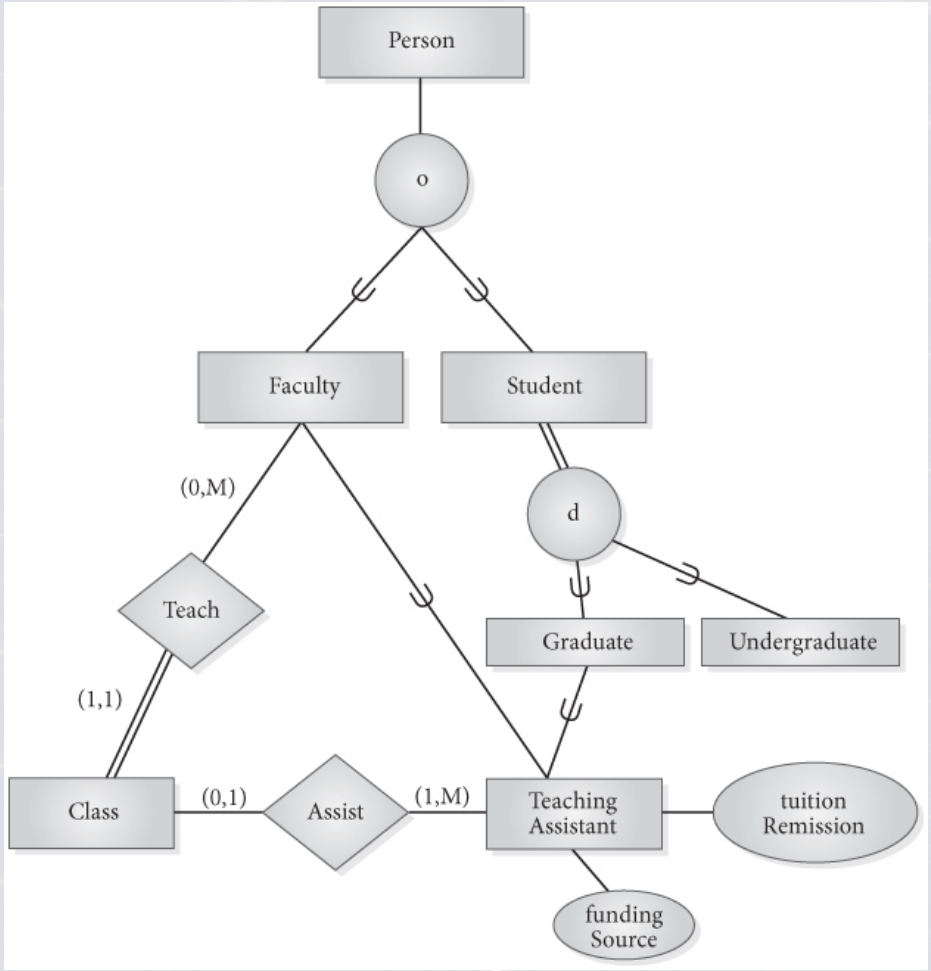
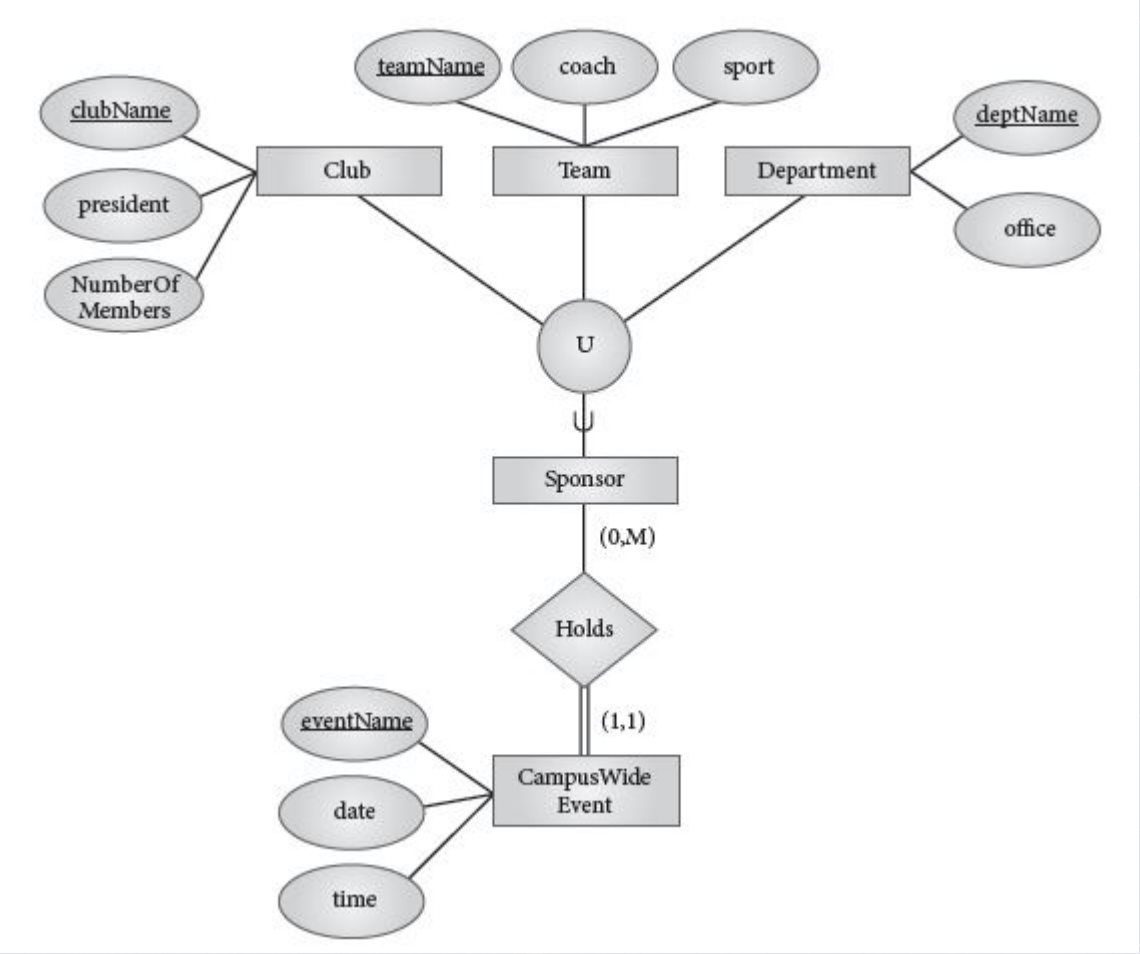


Figure 3.18 Multiple Inheritance

Union or Category

- Subset related to a **collection** of supersets
- Each instance of subset belongs to one, not all, of the supersets
- Supersets form a union or category
- Ex. A `Sponsor` may be a `Team`, a `Department`, or a `Club`. See Figure 3.19A.
- Each `Sponsor` entity instance is a member of one of these supersets, so `Sponsor` is a subset of the union of `Team`, `Department`, `Club`
- EE-R diagram - connect each superset to union circle, connect circle to subset, with subset symbol on line between circle and subset

Figure 3.19A Example of Union



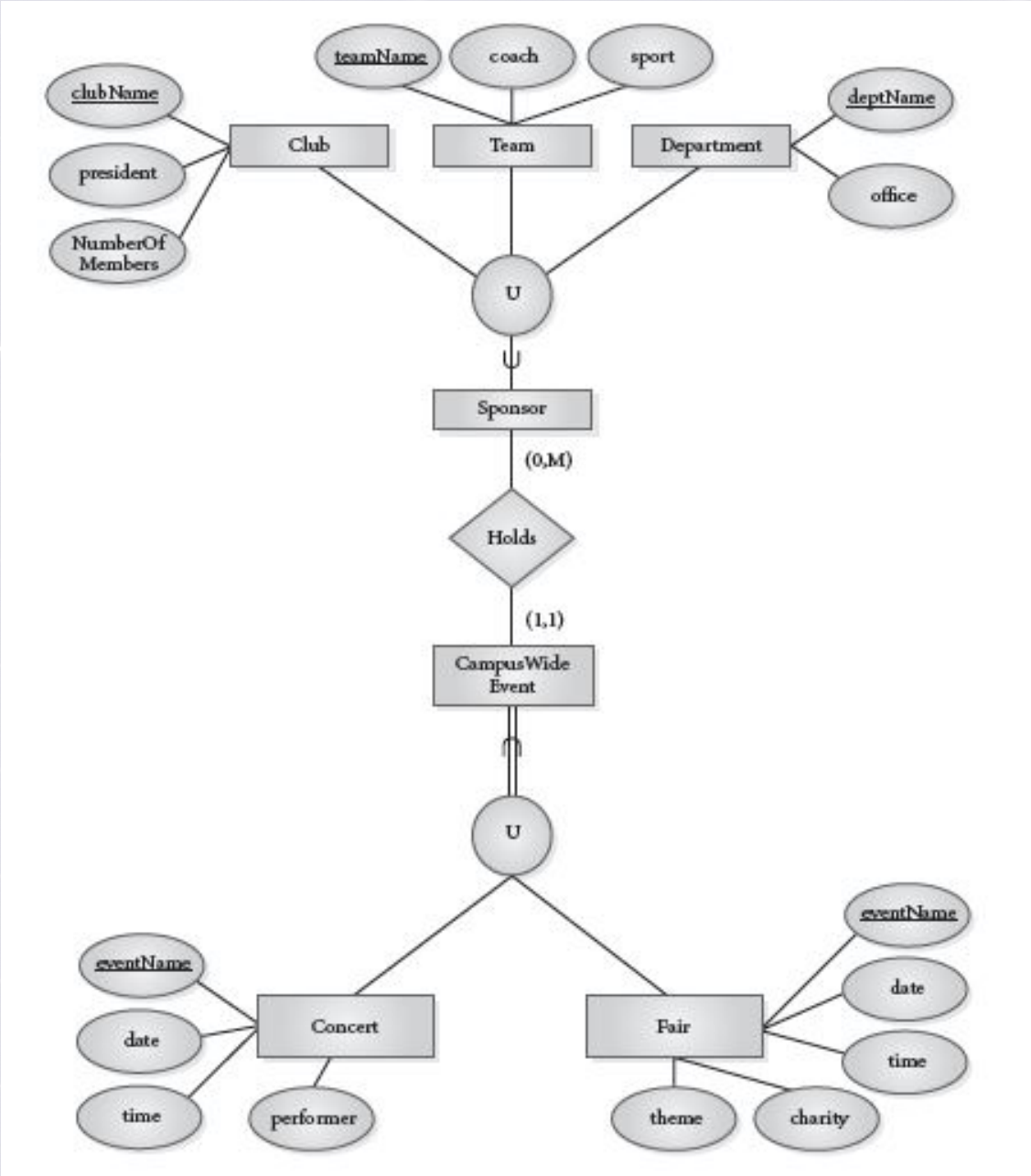
Total and Partial Unions

- **Total category** – every member of the sets that make up the union must participate
  - Shown on EE-R by double line from union circle to subset

  In **Figure 3.19B** every `Concert` or `Fair` must be a `Campus-Wide Event`
- **Partial category** – not every member of the sets must participate
  - Shown by single line

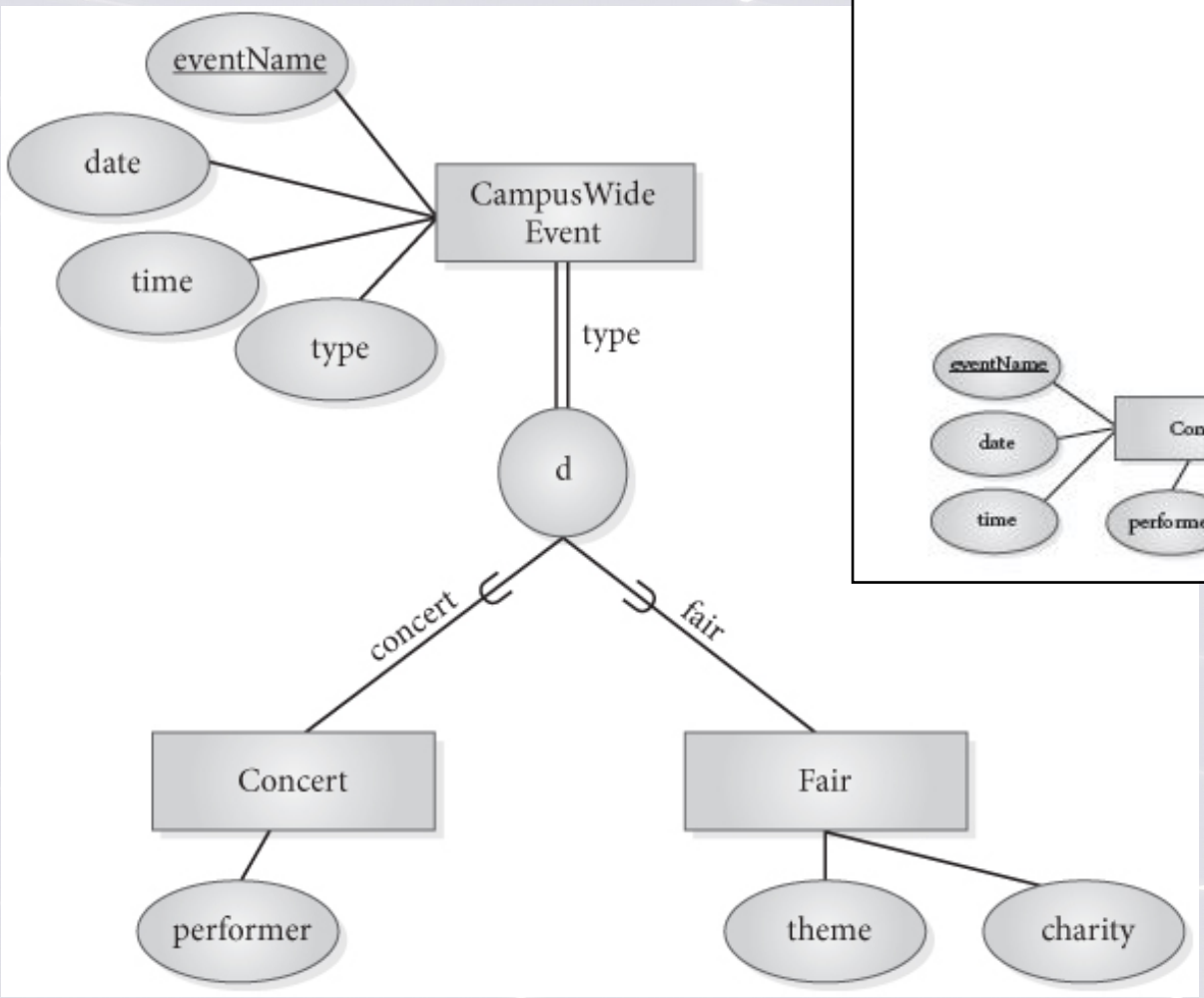
  Not every `Club`, `Team`, or `Department` must be a `Sponsor`

Figure 3.19B
Partial vs. Total
Union



Total Union vs. Specialization

- Total union can often be replaced by hierarchy
- Choose hierarchy representation if supersets have many common attributes
- See **Figure 3.20**



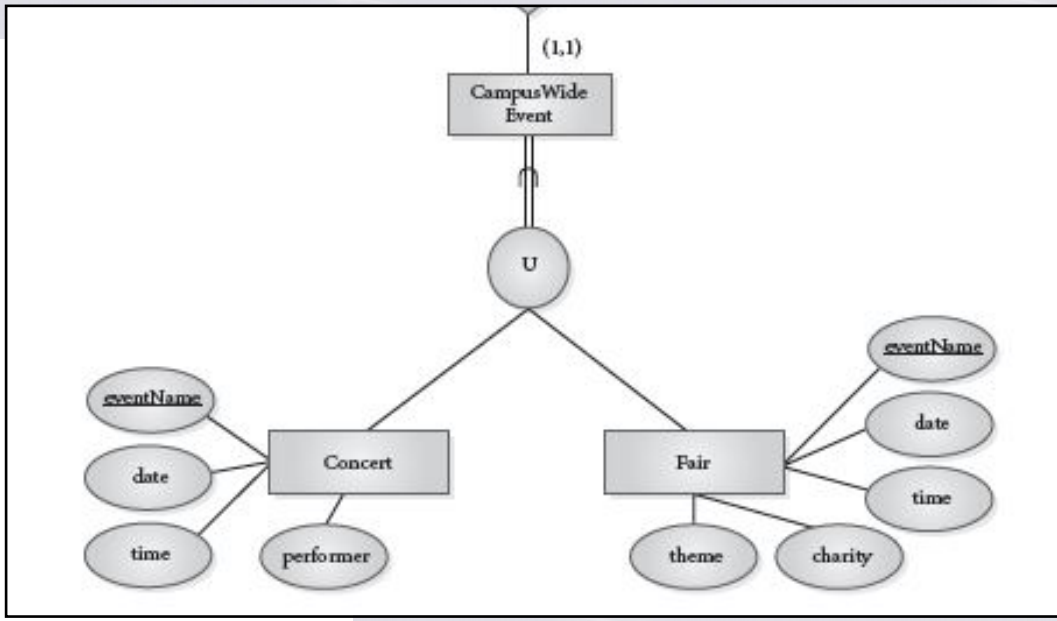


Figure 3.20 Total Union Replaced by Specialization/Generalization

Sample EE-R Diagrams

- Extended `University` Example
  see Figure 3.21
- Extended `Horse Racing` Example
  see Figure 3.22

E-R Diagram for University

Figure 3.13

An E-R Diagram:

For this example, we will use the traditional method of showing cardinality and participation constraints.

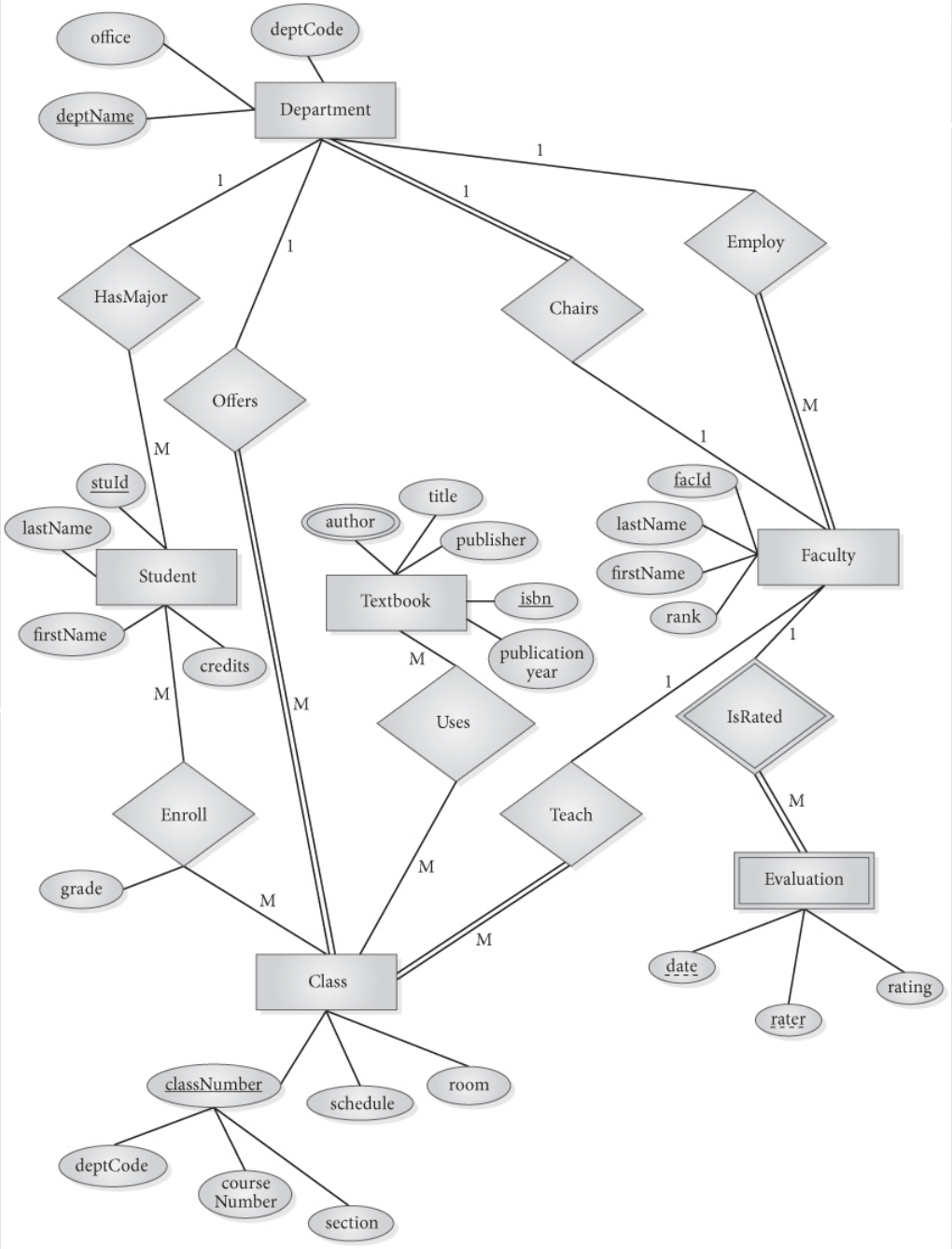


Figure 3.21 EE-R for University

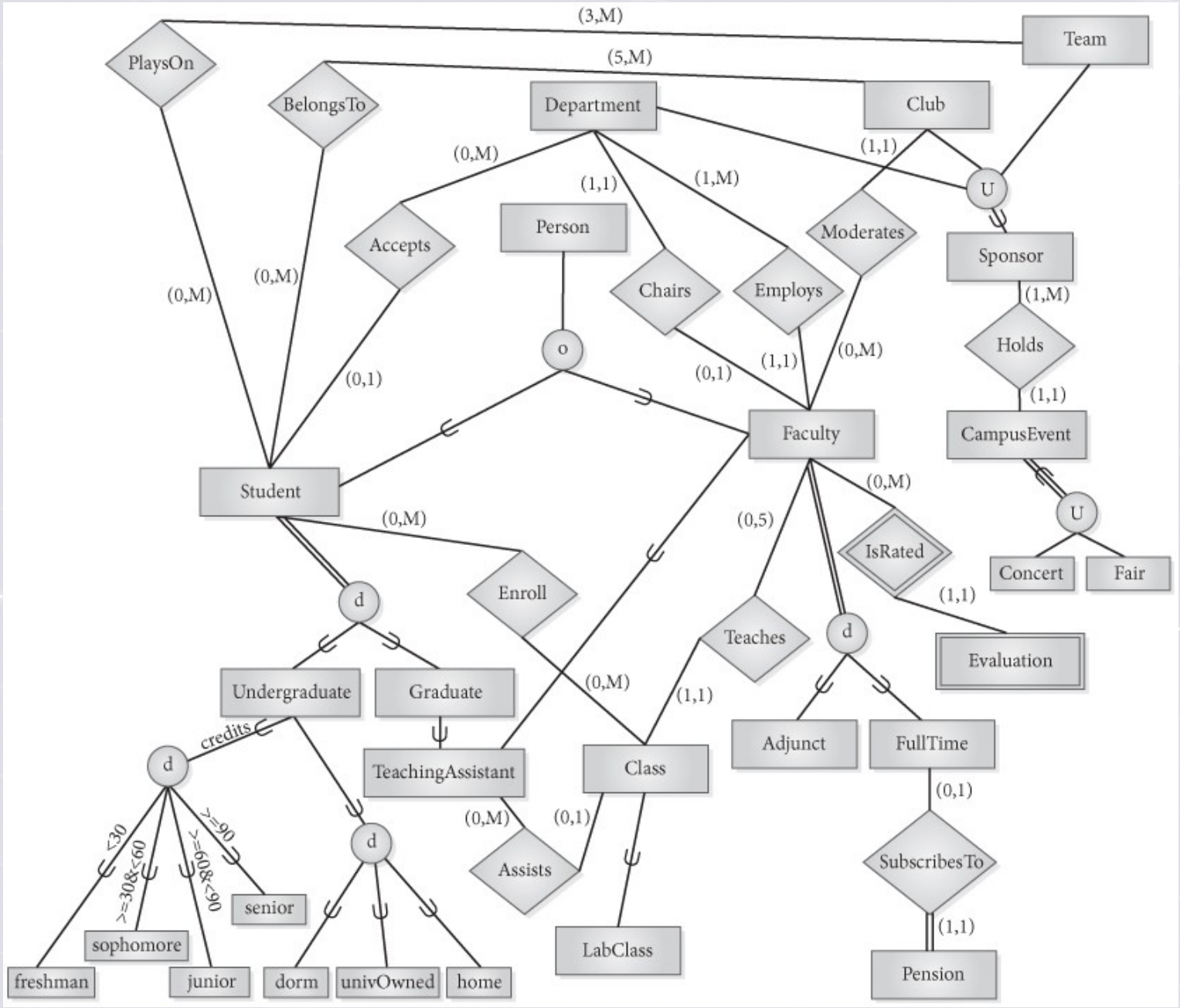
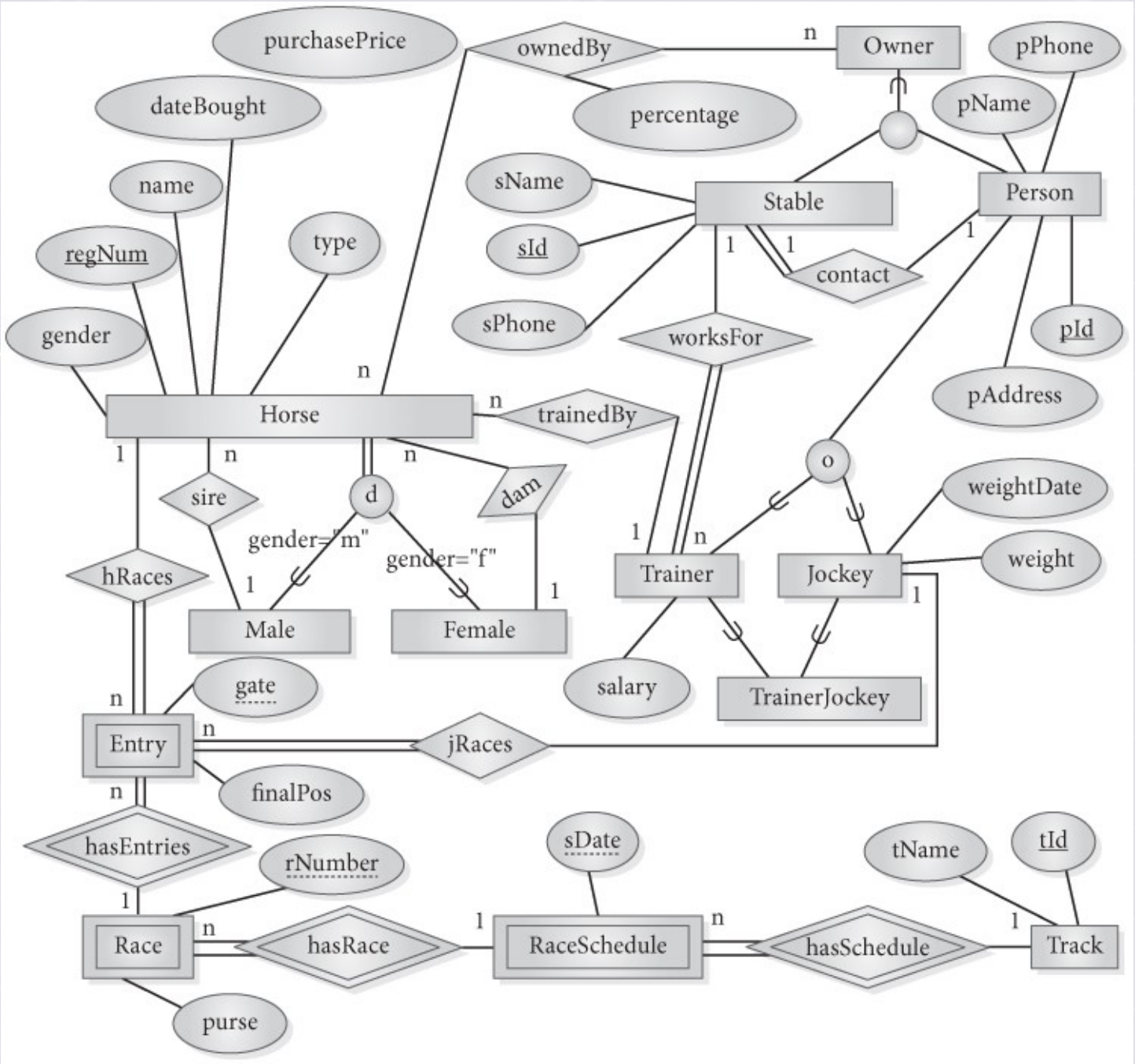
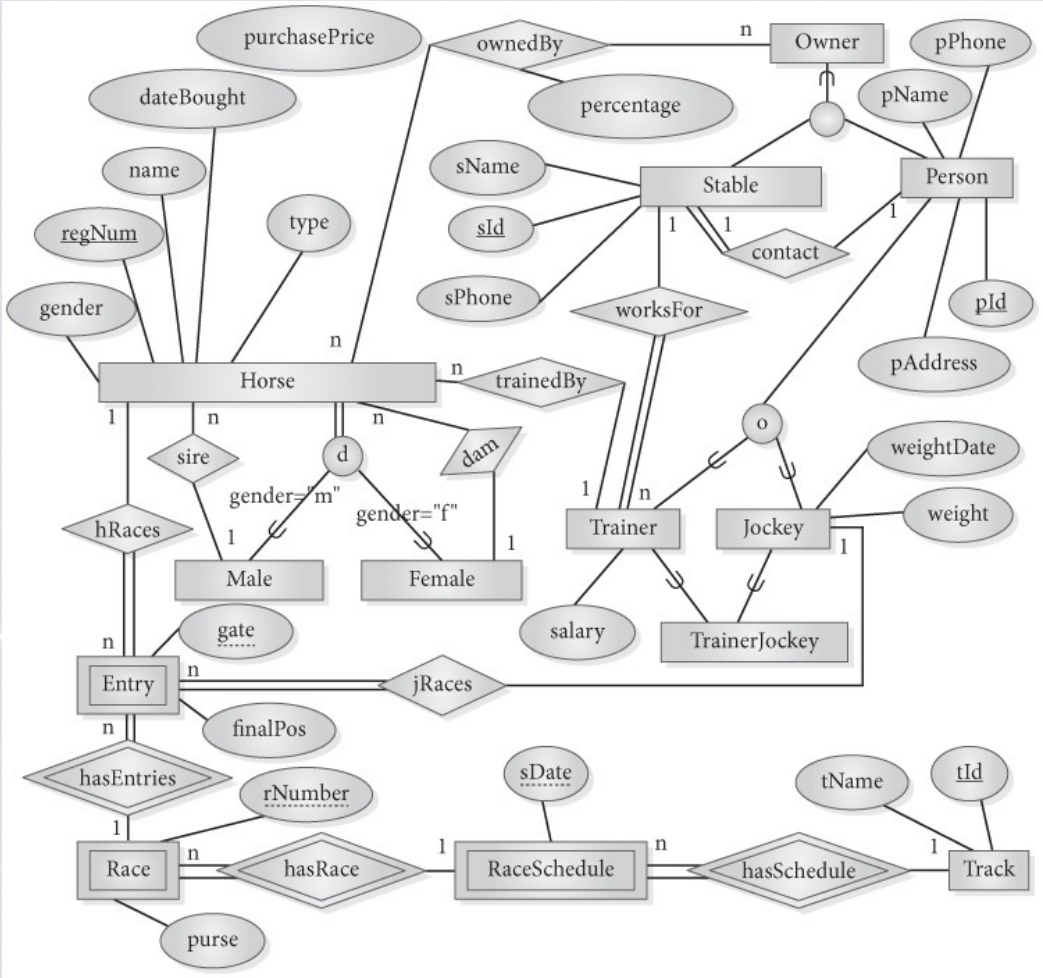


Figure 3.22
EE-R for `Horse Racing` Example



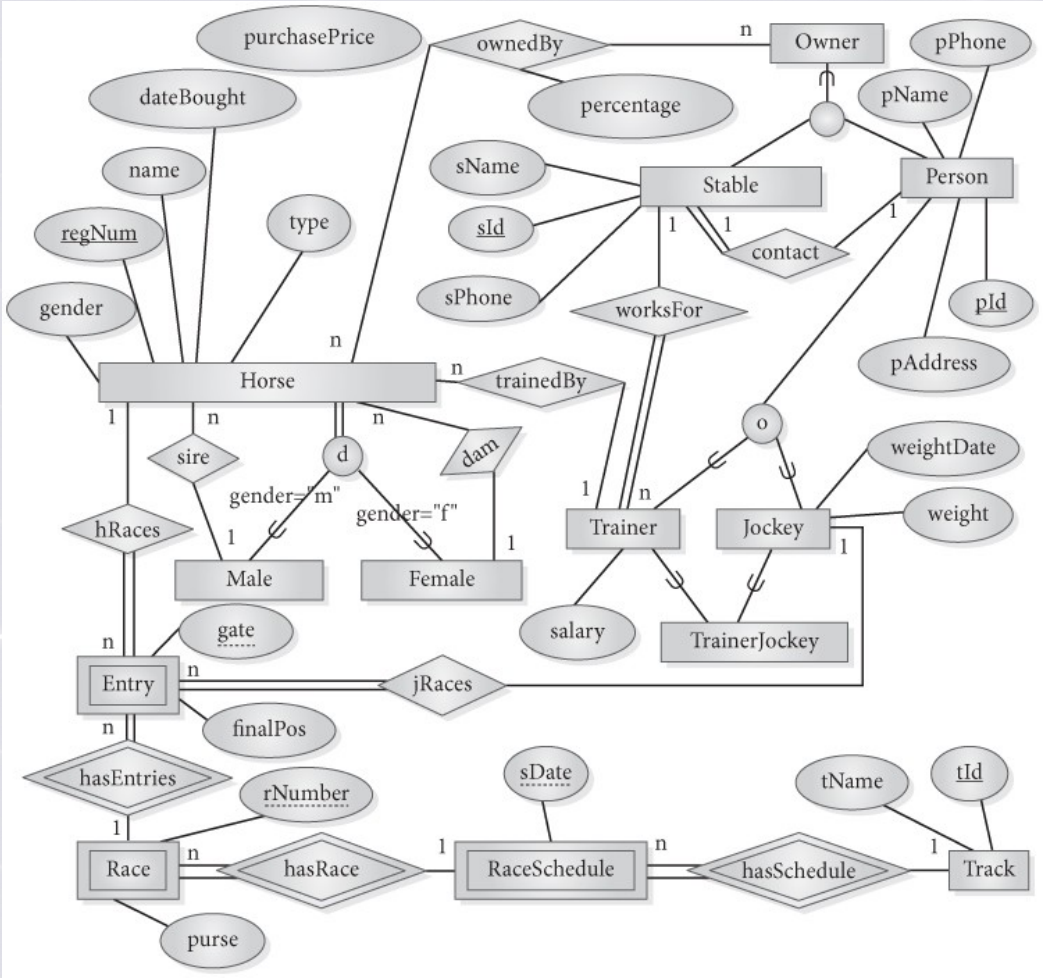
We add the following assumptions:

- *If a person is a horse trainer, the salary of the trainer should be indicated, along with the horses trained by the trainer.*
- *If a person is a jockey, the weight of the jockey must be recorded, along with the date of the last recorded weight.*
- *It is sometimes necessary to know whether a person is a trainer and a jockey.*
- *Name and contact information can also be maintained about people other than trainers and jockeys*.

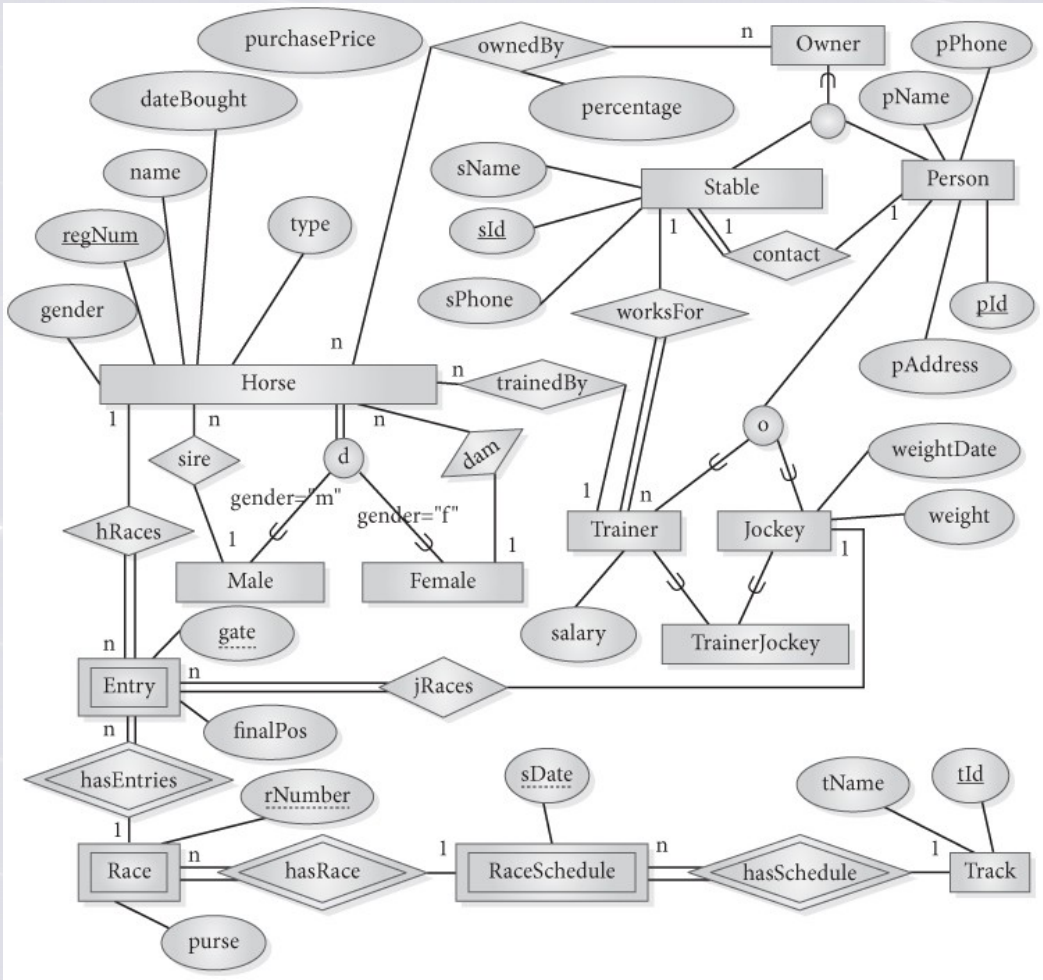


- *A stable is a company that breeds, trains, and sells/buys horses. A stable has an identifier, a name, phone, and contact person. A horse trainer works for a specific stable*.
- *An owner can either be a stable or a person*.

- To modify the E-R diagram for these additional assumptions, we need to create a specialization of the Person entity set into Owner, Trainer, and Jockey.
- Trainer and Jockey have a common subset, consisting of people who fit into both specializations, which we call TrainerJockey.
- The Trainer entity set is related to the Horse entity set by the TrainedBy relationship, and the Jockey is related to the Entryentity set by a relationship which we will call jRaces.



- We also add a new entity set, Stable, with attributes sId, sName, and sPhone.
- For the contact person, we add a relationship to the Personentity set
- Since either a stable or a person can be an owner of a horse, we create a union called Owner of the Stable and Person entity sets.
- Horse has an ownedBy relationship to this union.
- We also specialize Horse by gender into Male and Female subsets and create one-to-many relationships sire and dam for these subsets with the Horse entity set.



- **FIGURE 3.22** shows the EE-R diagram for the extended Horse Racing example.
- To simplify the diagram and focus on the specialization and union features, the Barn and StabledAt relationships from Figure 3.14 have been removed.