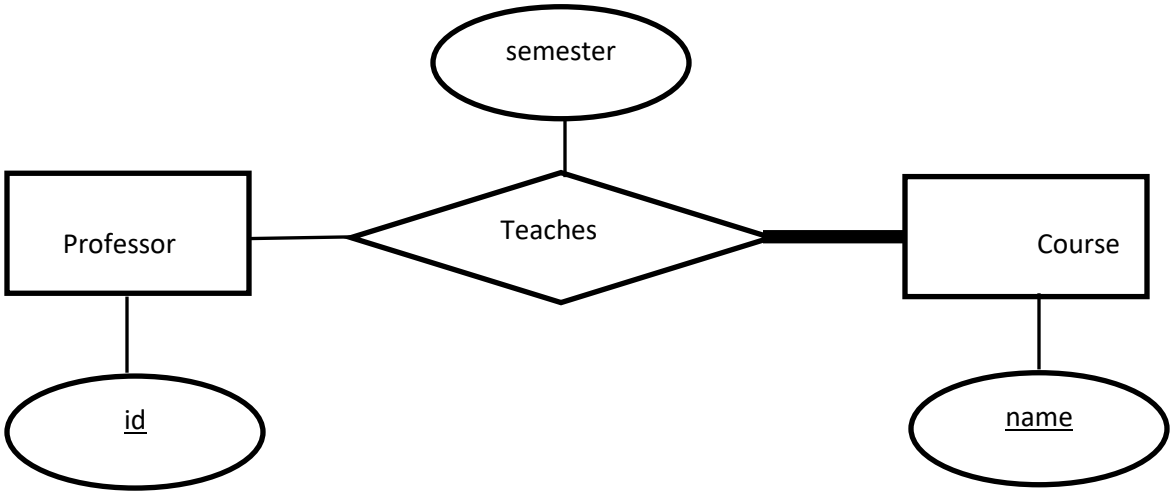
Brendan Murphy
Database Systems E-66

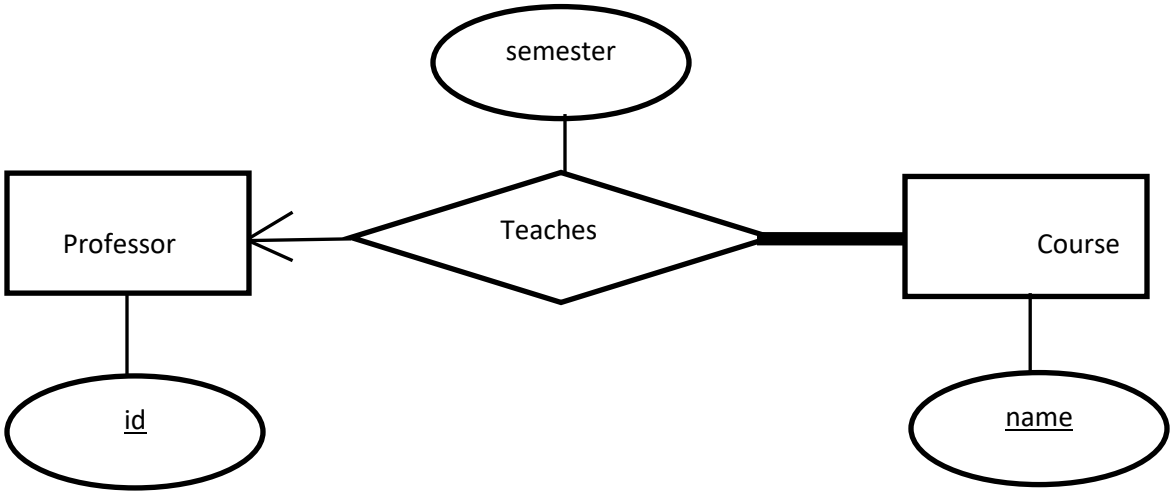
Part 1
Problem 1, ER Diagram Basics

1. Describe how you would modify Figure 1-1 to reflect the fact that every professor teaches *at least one* course. (Describing the change in words is fine, although you may use a diagram if you prefer.)



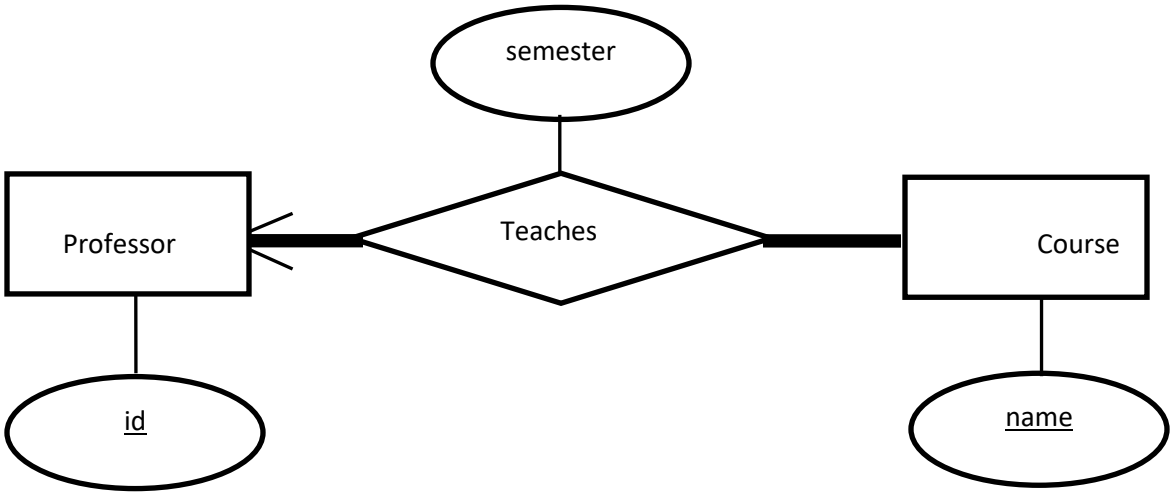
Thick line from Course to Teaches specifies every professor teaches *at least one* course.

2. Describe how you would modify the original version of Figure 1-1 to reflect the fact that every professor must teach *exactly one* course. (Describing the change in words is fine, although you may use a diagram if you prefer.)



Arrow into Professor specifies every professor teaches at most one course.
Thick line from Course to Teaches specifies every professor teaches at least one course.
At most one + at least one = <u>exactly one</u>

3. Describe how you would modify the original version of Figure 1-1 to reflect the fact that every professor must teach *exactly one* course, and every course must be taught by *at least one* professor. (Describing the change in words is fine, although you may use a diagram if you prefer.)



Arrow into Professor specifies every professor teaches at most one course.
Thick line from Course to Teaches specifies every professor teaches at least one course.
At most one + at least one = exactly one
Thick line from Professor to Teaches specifies every course is taught by *at least one* professor.

4. Consider your answer to part 1—the ER diagram for the situation in which every professor teaches *at least one* course. If we converted that ER model to a relational schema, would the following be an acceptable schema for a relation used to capture the Teaches relationship set?

   Teaches(professor, course, semester)

   where *professor* is a foreign key referring to *Professor(id)*, *course* is a foreign key referring to *Course(name)*, and the primary-key attributes of *Teaches* are underlined. Explain your answer briefly.

   The schema used to capture the Teaches relationship is acceptable given that every professor teaches *at least one* course. If the ER diagram indicated that every professor teaches at *most one course* the set would need to be constrained to ensure reflect this. To accomplish this, we could change the schema to only have one primary key; course. This could be reflected in the schema by removing the underline from professor.

   Teaches(professor, course, semester)

Problem 2: Database Design

1: BelongsTo is a many-to-one relationship from Artist to Label.

2: Describe all constraints on relationships that are specified by the diagram.

- Each Album must have at least one Song appear on it
- Each Album must be produced by one Label
- Each Artist must sing at least one Song
- Each Artist belongs to at most one Label
- Each Artist belongs to at least one label

3: Relational Schema of Figure 2-1:

Artist(name, dob, _id_)
Sings(_artist_id_, _song_id_)
Sings has two foreign keys: _artist_id_, _song_id_
Referential-integrity Constraints:
Each value of _artist_id_ attribute must match a value of _id_ attribute from Artist relation.
Each value of _song_id_ attribute must match a value of _id_ attribute from the Song relation.

Song(name, duration, _id_)
AppearsOn(_song_id_, _album_id_)
AppearsOn has two foreign keys: _song_id_, _album_id_
Referential-integrity Constraints:
Each value of _album_id_ attribute must match a value of _id_ attribute from the Artist relation.
Each value of _song_id_ attribute must match a value of _id_ attribute from the Song relation.

Album(name, _id_, _label_name_)
Special Case. Because Artist is in a many-to-one relationship with Label, I was
able to capture the BelongsTo relationship set by adding a _label_name_ attribute to the
Album relation because it is the entity being restrained.

Label(_name_, address)
Produces(_id_, _name_)
Produces has two foreign keys: _id_, _name_
Referential-integrity Constraints:
Each value of _id_ attribute must match a value of _id_ attribute from the Label relation.
Each value of _name_ attribute must match a value of name attribute from the Artist relation.

Problem 3: Combining relations

What is the Cartesian product of R and S?

a b c a b
(1, 2, 3, 2, 3)
(1, 2, 3, 3, 4)
(1, 2, 3, 7, 6)
(3, 4, 3, 2, 3)
(3, 4, 3, 3, 4)
(3, 4, 3, 7, 6)
(7, 6, 5, 2, 3)
(7, 6, 5, 3, 4)
(7, 6, 5, 7, 6)

What is the natural join of R and S?

a b c
(3, 4, 3)
(7, 6, 5)

What is the left outer join of R and S?

a b c
(3, 4, 3)
(7, 6, 5)
(1, null, null)

What is the right outer join of R and S?

a b c
(3, 4, 3)
(7, 6, 5)
(2, null, null)

What is the full outer join of R and S?

a b c
(3, 4, 3)
(7, 6, 5)
(1, null, null)
(2, null, null)

Problem 4: Relational algebra queries

1. (3 points) problem 7 (Oscars won by Meryl Streep)

π Oscar.year, Oscar.type, Movie.name (σ Oscar.person_id = 000658 (Movie ⋈ Movie.id = Oscar.movie_id Oscar))

2. (3 points) problem 12 (actors who have not appeared in any movies from the current decade), but instead of producing a count, your query should produce the ids of the actors:

(π Actor.actor_id (Actor)) - (π Actor.actor_id (σ Movie.year≥2010 (Movie ⋈ Movie.id = Actor.movie_id Actor)))

3. (4 points) problem 13 (Oscars won by people born in Mexico), but instead of producing counts, your query should produce tuples of the form (name, award type, year). If a person has won multiple awards, he or she should have multiple tuples in the results, one for each award. If a person has won no awards, he or she should have a single tuple with NULL values for the award type and year.

π Person.name, Oscar.type, Oscar.year (σ Person.pob LIKE '%Mexico' (Oscar ⋈ Oscar.person_id = Person.id Person))