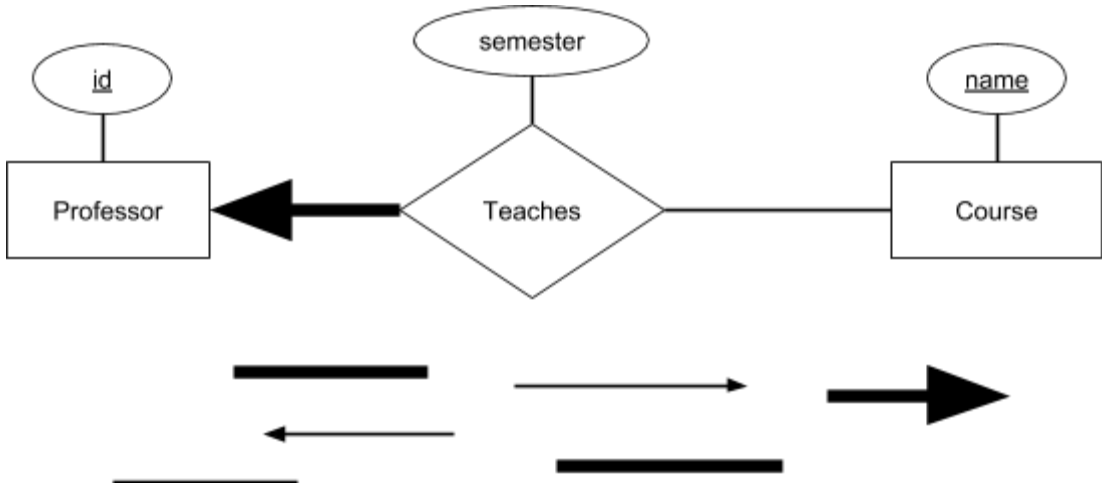
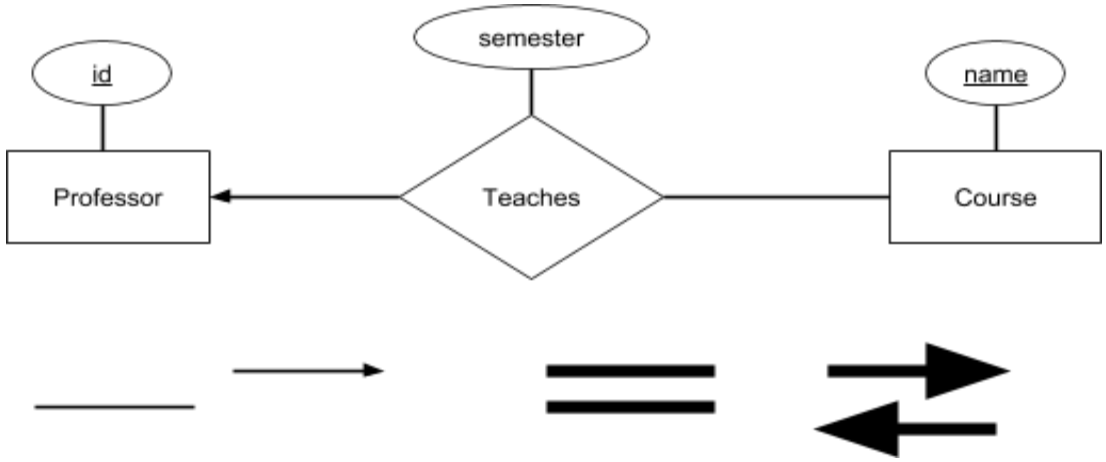
Problem Set 1, Part I

Problem 1: ER diagram basics

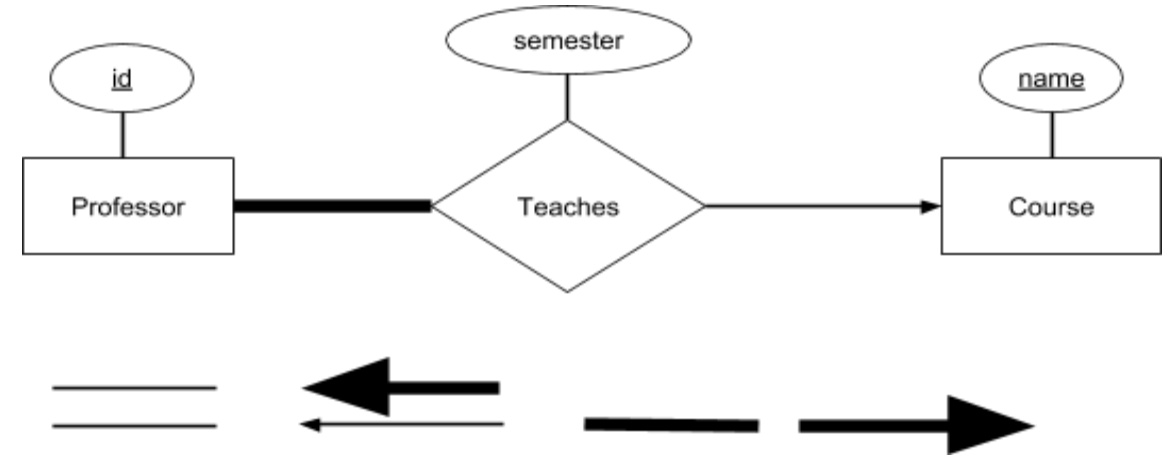
1.



2.



3.



Problem 1 (cont)

4. *Teaches(professor, course, semester)* is not a valid schema for *at most one professor* relationship illustrated in *Problem 1 Part 2* , because there are courses having multiple professor foreign keys. There must be a constraint on course to force it to be unique, e.g. *Teaches(course, professor, semester)*. If not, an example like so would happen:
Professor Sullivan teaches CSCI-E66
Professor Molina teaches CSCI-E66
which violates the constraint of each course having *at most one* professor. Also, having another table to illustrate a many-to-one relationship seems extraneous.

Another schema design would be to not combine the two relations. For example, *Course(course, professor, semester)* and *Professor(professor);* here, course would be the primary key for course and professor would be the foreign key to the professor relation in the *Course* relation. This configuration enforces courses to have at most one professor.

Problem 2: Database design

1. HasType is a many to one relationship from Account to AccountType
   Borrows is a many to one relationship from Customer to Loan

2. Each Customer borrows at most one Loan
   Each Loan has at least one Customers
   Each Account has at least one Customers
   Each Account must have exactly one AccountType

3. *Customer(Customer id , Loan id)*: is the combined relation between Customer and Loan where Customer is a primary key and Loan is a foreign key since the relationship represents Each Customer borrows at least one Loan. The two relations are not combined because this represents a many-to-one relationship.

Owns(Customer id , Account id): is the combined relation between Customer and Account where Customer is the foreign key and Account is the non-nullable foreign key since the relationship represents Each Account has total participation in Customers. The two relations are combined into a separate relation because this represents a many-to-many relationship.

Account(Account id , AccountType_id , *balance*): represents the relation for Account where Account is the primary key and the AccountType id is the foreign key to the AccountType relation.

Problem 3: Combining relations

Use the Insert->Table *menu option to insert an appropriate table for each answer.*

1. Cartesian

a	b	c	b	a
1	2	3	2	1
1	2	5	4	3
1	2	9	8	7
3	4	3	2	1
3	4	5	4	3
3	4	9	8	7
5	6	3	2	1
5	6	5	4	3
5	6	9	8	7

2. Natural Join

a	b	c
1	2	3
3	4	5

3. Left Outer Join

a	b	c
1	2	3
3	4	5
5	6	null

4. Right Outer Join

c	b	a
3	2	1
5	4	3
9	8	7

5. Full Outer Join

a	b	c
1	2	3
3	4	5
5	6	null
7	8	9

Problem 4: Relational algebra queries

1.
PROJECT{name, pob, dob}SELECT{ name='Emma Stone' or name='Rachel Weisz'}(Person)

2.
OscarMovies <-- PROJECT {Oscar.person_id, Movie.name, Oscar.type, Oscar.year} (Oscar) CONDITIONAL_JOIN {movie_id = Movie.id} (Movie)

PROJECT{OscarMovies.name, OscarMovies.type, OscarMovies.year} (SELECT{name = 'Christian Bale'} (Person) CONDITIONAL_JOIN {id=person_id} OscarMovies)

3.
Top25Movies <-- SELECT{ earning_rank <= 25 }(Movies)

PROJECT{earning_rank, name, type}(Top25Movies LEFT_OUTER_JOIN {id = movie_id} Oscar)