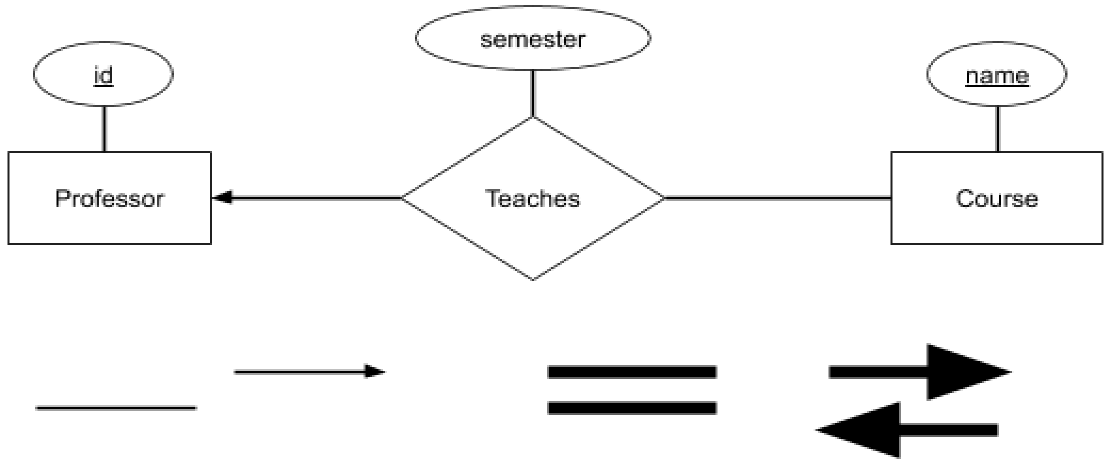
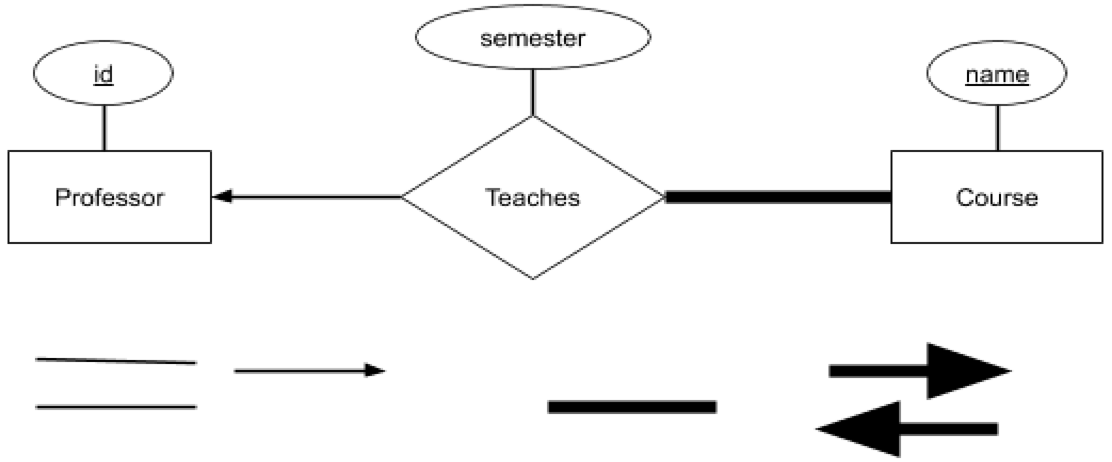
Problem Set 1, Part I

Problem 1: ER diagram basics

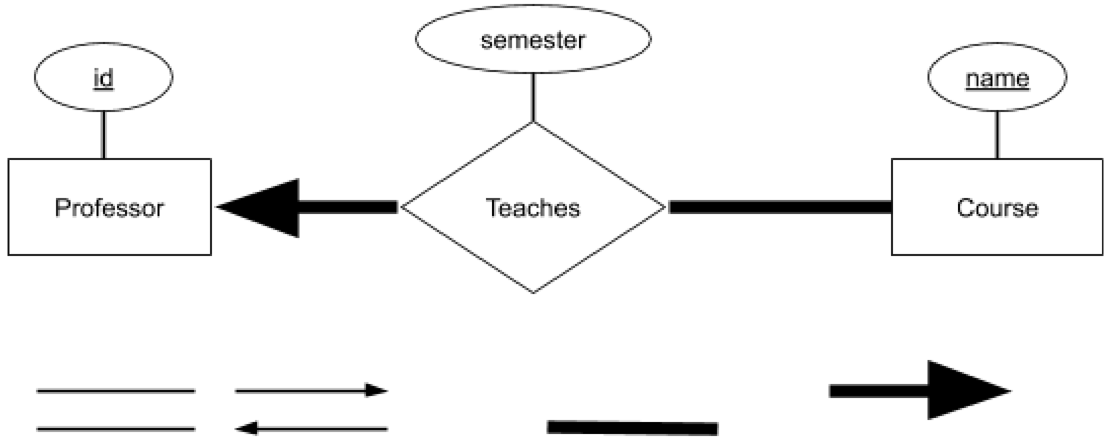
1.



2.



3.



Problem 1 (cont)

4. It is possible that a professor teaches different sections of a given course in a given semester. There must be another primary key like the time when the lectures are held to enforce uniqueness of the combination of the primary keys. Therefore, it may violate the primary key constraint as it stands.

Problem 2: Database design

1) There is a many to one relationship from Author to Agent

2)

- Every author has at most one agent.
- Every agent has at least one author that it manages.
- Every appearance has more or authors that show up
- An author may have multiple appearances
- Many authors can participate in writing a single book
- An author may write many books

3)

- **Agent(_agentid_, name, officenumber, Author._authorid_)** -> _authorid_ references _authorid_ in Author. Since an Author may have at most one Agent and an Author can be maybe managed by at least one Agent, the result is an Author can be managed by exactly one agent and thus the one to one relationship indicated by the foreign key. The trade off of having the foreign key here instead of in Author table is to reduce the amount of null values since some Authors may not have agents.

- **Author(_authorid_, name, address)** -> authors are identified by id

- **Appearance(_date_, _location_, Author._authorid_)** -> _authorid_ is a foreign key that references _authorid_ in Author. Each appearance has one or more authors that show up.

- **Book(_isbn_, title)** -> books are identified by isbn number

- **Wrote(_authorid_, _isbn_, publisher)** -> Each value of _authorid_ references _authorid_ from Author table. Each value of _isbn_ references _isbn_ from Book table. The combination of author id and isbn is unique. There is also a refertial intergity constraint that _isbn_ and _authorid_ must be valid values defined in Book and Author tables respectively. A book maybe written by many authors and an Author may write many books.

Problem 3: Combining relations

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

1)

R.a	***R.b***	***c***	***S.b***	***S.a***

1	*2*	*1*	*2*	*3*
1	*2*	*3*	*4*	*3*
1	*2*	*7*	*6*	*5*
3	*4*	*1*	*2*	*3*
3	*4*	*3*	*4*	*3*
3	*4*	*7*	*6*	*5*
5	*6*	*1*	*2*	*3*
5	*6*	*3*	*4*	*3*
5	*6*	*7*	*6*	*5*

2)

a	***b***	***c***
3	*4*	*3*
5	*6*	*7*

3)

a	***b***	***c***
1	*2*	*null*
3	*4*	*3*
5	*6*	*7*

4)

a	***b***	***c***
3	*2*	*1*
3	*4*	*3*
5	*6*	*7*

5)

a	***b***	***c***
1	*2*	*null*
3	*4*	*3*
5	*6*	*7*

3	*2*	*1*

Problem 4: Relational algebra queries

1) $\Pi_{\text{name, Oscar.year}} (\sigma >=_{\text{Oscar.year <= 2019}} (\sigma (_{\text{Oscar.year >= 2010}} (\text{Movie} \bowtie \text{Oscar}))))$

2) $\Pi_{\text{Movie.earnings\_rank, Movie.name, Oscar.type}} (\sigma (_{\text{Movie.earnings\_rank <= 25}} (\text{Movie} \rtimes \text{Oscar})))$

3)

BestActor <-- $\Pi_{\text{person\_id}}\ \sigma(_{\text{Oscar.type = 'BEST-ACTOR'}} (\text{Oscar}))$
BestActress <-- $\Pi_{\text{person\_id}}\ \sigma(_{\text{Oscar.type = 'BEST-ACTRESS'}} (\text{Oscar}))$

BestSupportingActor <-- $\Pi_{\text{person\_id}}\ \sigma(_{\text{Oscar.type = 'BEST-SUPPORTING-ACTOR'}} (\text{Oscar}))$
BestSupportingActress<-- $\Pi_{\text{person\_id}}\ \sigma(_{\text{Oscar.type = 'BEST-SUPPORTING-ACTRESS'}} (\text{Oscar}))$

(BestSupportingActor - BestActor) U (BestSupportingActress - BestActor)