

# CPSC 304 – September 20/24, 2024

## Administrative Notes

---

- Project
  - Groups are all formed
  - Milestone 1 due on October 1
- Tutorial
  - week of the 23<sup>rd</sup>: project work time (get feedback!)
  - Week of September 30: Relational model
    - If you have tutorial on Monday, go to another section
- New “In Class” exercise today
  - All in-class exercises must be submitted individually!!!

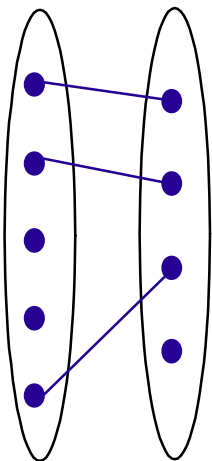
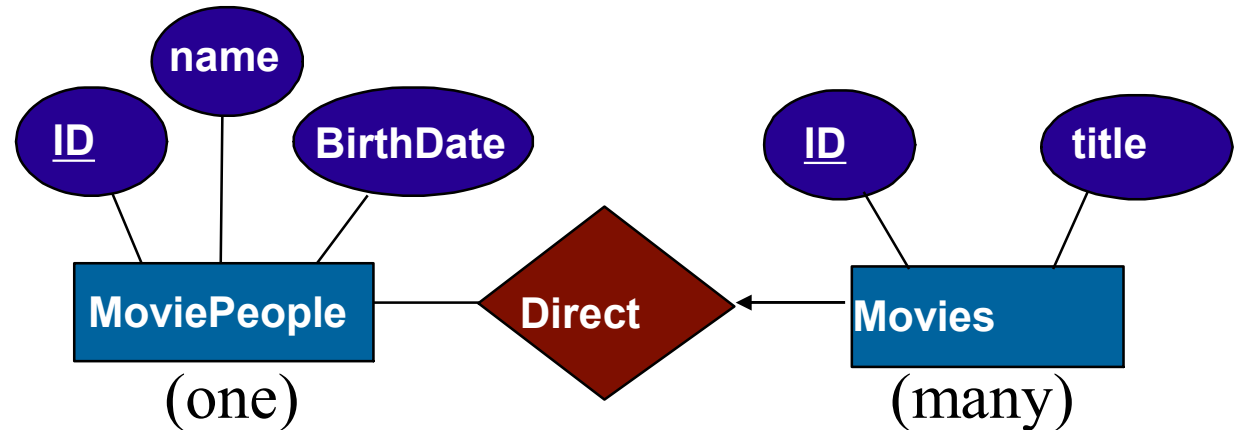
## Now where were we...

---

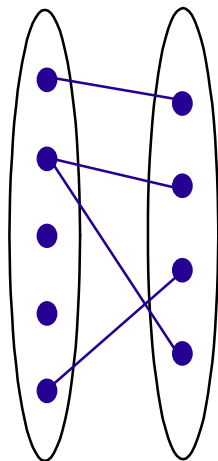
- We'd been covering how to translate an ER diagram to a relational model
- We'd discussed that for many-to-many relationships, you create separate tables for:
  - Each entity
  - The relationship
- What if there are constraints?

# Review: Key Constraints

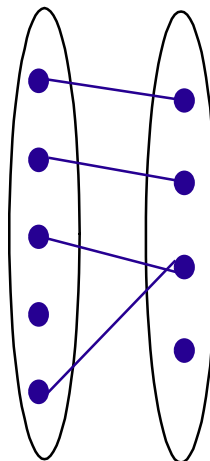
- Each movie has at most one director, according to the key constraint on Direct.



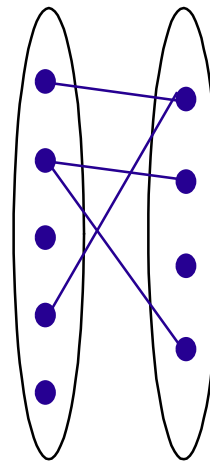
1-to-1



1-to Many



Many-to-1

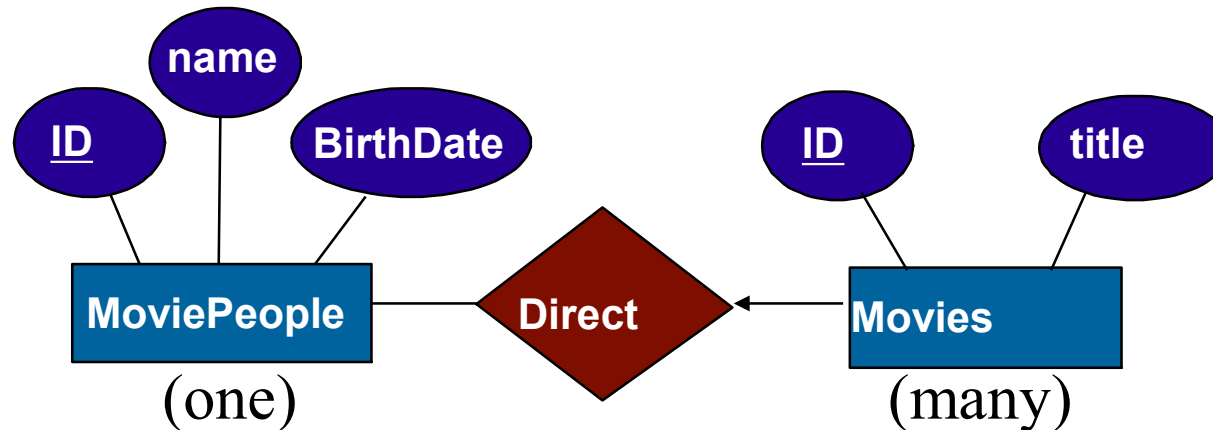


Many-to-Many

*Translation to relational model?*

# Relationship Sets with Key Constraints

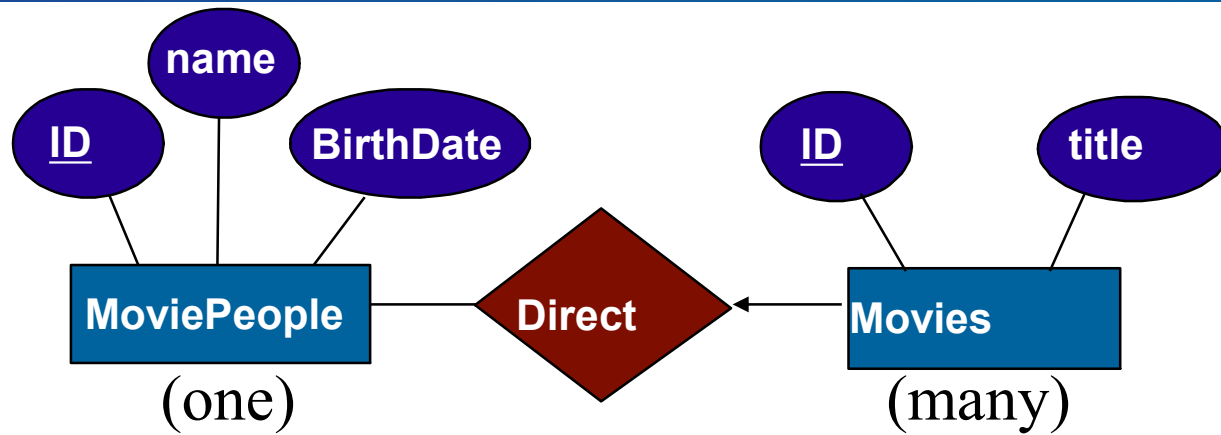
---



- Each movie has at most one director, according to the key constraint on Direct.
- How can we take advantage of this?

# For the one to many case, combine the many side with the relationship

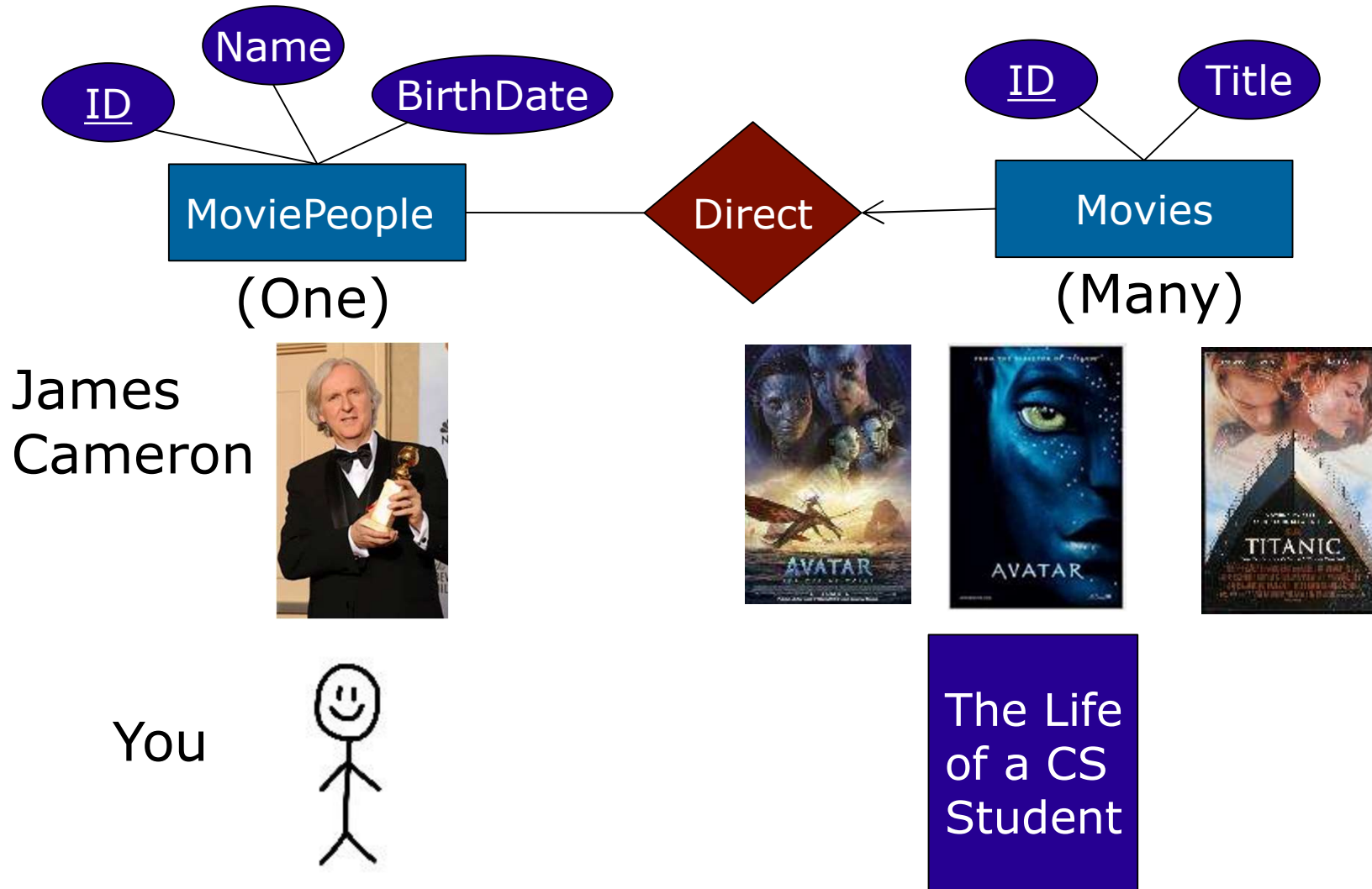
---



I know who the MoviePerson is if I know the movie!

I can remove some redundancy in my design.

# For the one to many case, combine the many side with the relationship



# For the one to many case, combine the many side with the relationship

---

MoviePeople

<u>ID</u>	Name	Birthdate
1	James Cameron	...
2	You	...

Movies

<u>ID</u>	Title
1	Avatar
2	Titanic
3	The Life of a CS Student

Direct

<u>MPID</u>	<u>MID</u>
1	1
1	2
2	3

Can we reduce redundancy by integrating Direct into MoviePeople?

# For the one to many case, combine the many side with the relationship

MoviePeople

<u>ID</u>	Name	Birthdate	Directed-MID
1	James Cameron	...	
2	You	...	

Movies

<u>ID</u>	Title
1	Avatar
2	Titanic
3	The Life of a CS Student

Direct

<u>MPID</u>	<u>MID</u>
1	1
1	2
2	3

Same issue as before.  
We can't have  
multiple values here.



# For the one to many case, combine the many side with the relationship

---

MoviePeople

<u>ID</u>	Name	Birthdate
1	James Cameron	...
2	You	...

Movies

<u>ID</u>	Title
1	Avatar
2	Titanic
3	The Life of a CS Student

Direct

<u>MPID</u>	<u>MID</u>
1	1
1	2
2	3

Can we integrate  
Direct into Movies?

# For the one to many case, combine the many side with the relationship

MoviePeople

<u>ID</u>	Name	Birthdate
1	James Cameron	...
2	You	...

Movies

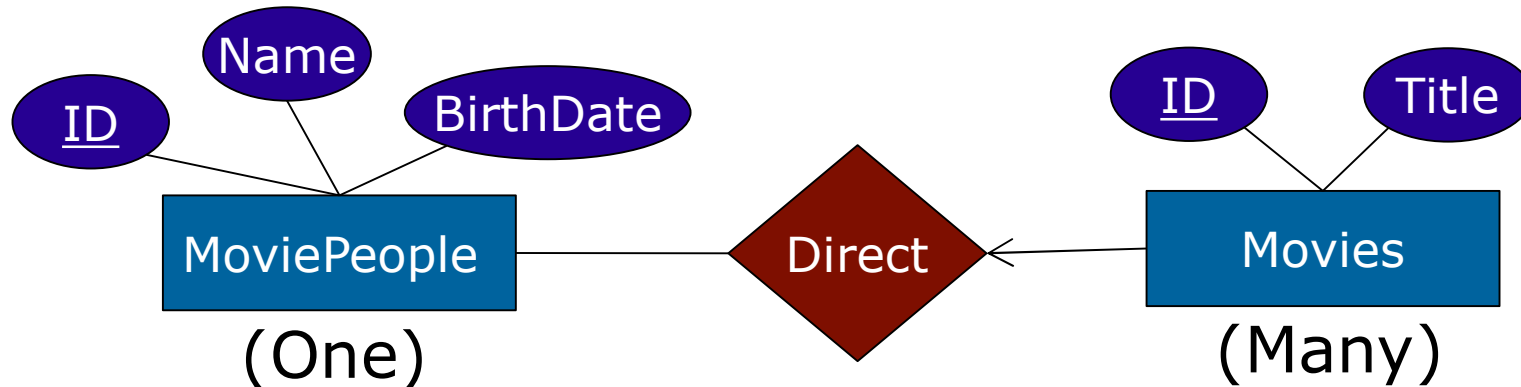
<u>ID</u>	Title	Director-MPID
1	Avatar	1
2	Titanic	1
3	The Life of a CS Student	2



Will there ever be two different directors for the same movie?

No! (because of our one to many relationship)

# For the one to many case, combine the many side with the relationship



MoviePeople

<u>ID</u>	Name	Birthdate
1	James Cameron	...
2	You	...

Movies

<u>ID</u>	Title	Director-MPID
1	Avatar	1
2	Titanic	1
3	The Life of a CS Student	2

# Translating ER Diagrams with Key Constraints

- Method 1 (unsatisfactory):

- Create a separate table for Direct:
- Note that **MID** is the key now!
- Create separate tables for MoviePeople and Movies.

```
CREATE TABLE Direct(  
  MPID CHAR(11),  
  MID INTEGER,  
  PRIMARY KEY (MID),  
  FOREIGN KEY (MPID) REFERENCES  
    MoviePeople,  
  FOREIGN KEY (MID) REFERENCES Movies)
```



- Method 2 (better)

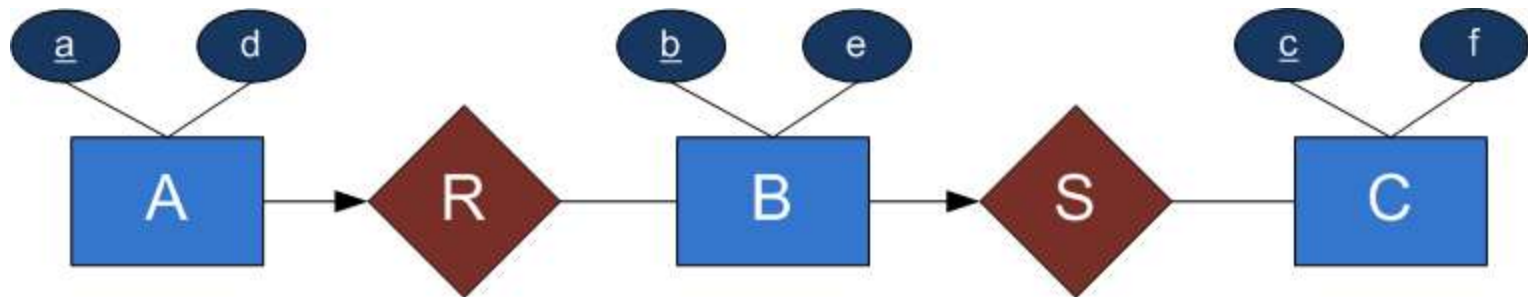
- Since each movie has a unique director, we can **combine Direct and Movies into one table.**
- Create another table for MoviePeople
- Must have on delete and on update in this case!

```
CREATE TABLE Directed_Movie(  
  MID INTEGER,  
  title CHAR(20),  
  MPID CHAR(11),  
  PRIMARY KEY (MID),  
  FOREIGN KEY (MPID) REFERENCES  
    MoviePeople  
    ON DELETE SET NULL  
    ON UPDATE CASCADE)
```

Oracle does not support "on update" – still use method 2

# Clicker Question

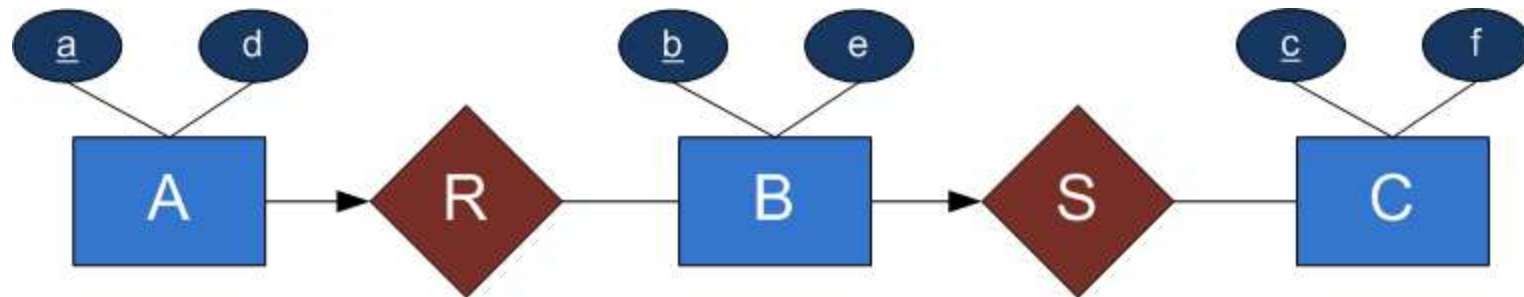
---



Translate the ER diagram to relational schema.  
Underline key attributes, and make FKs bold.

- Schema example: BMW (bid, **sid**)

# Clicker Question



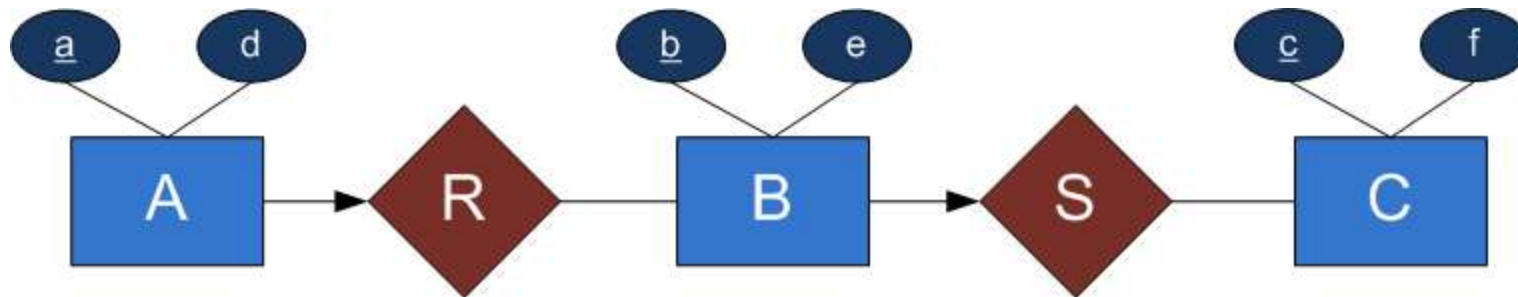
Translate the ER diagram to relational.

Which of the following appears in your relational schema:

- A.  $AR(\underline{a}, \underline{b}, d)$
- B.  $BS(\underline{b}, \mathbf{c}, e)$
- C.  $S(\underline{b}, \underline{c})$
- D. All of these
- E. None of these

Primary keys are underlined. Foreign keys are bolded.

# Clicker Question



Translate the ER diagram to relational.

Which of the following appears in your relational schema:

A.  $AR(\underline{a}, \underline{b}, d)$

B.  $BS(\underline{b}, \underline{c}, e)$

C.  $S(\underline{b}, \underline{c})$

D. All of these

E. None of these

<del>A</del>		
<del>B</del>	AR	$AR(\underline{a}, \underline{b}, d)$
<del>C</del>	BS	$BS(\underline{b}, \underline{c}, e)$
<del>R</del>	C	$C(\underline{c}, f)$
<del>S</del>		

→

# To motivate examples on upcoming slides, let's talk about getting a PhD

---



- PhD students all have to have advisors, all of whom also have had advisors (etc.)
- There exist databases where you can go back hundreds of years to see people's academic lineage
- Out of curiosity, and to make this more interesting for you (you're welcome), you can use <https://www.genealogy.math.ndsu.nodak.edu/> to look this information up





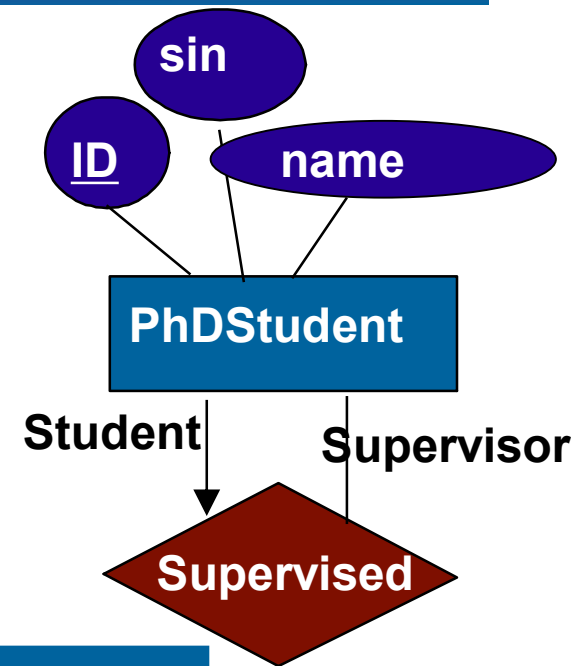
# Rachel's academic genealogy:

---

- Rachel Pottinger
- Phil Bernstein
- Catriel Beerli
- Eli Shamir
- Shmuel Agmon
- Szolem Mandelbroit
- Jacques Salomon Hadamard
- C Emile (Charles) Picard
- Gaston Darboux
- Michel Chasles
- Simeon Denis Poisson
- Joseph Louis Lagrange
- Leonhard Euler
- Johann Bernoulli
- Jacob Bernoulli
- Peter Werenfels
- Theodor Zwinger, Jr.
- Sebastian Beck
- Johann Jacob Grynaeus
- Simon Sulzer
- Wolfgang Fabricius Capito
- Desiderius Erasmus
- Jan Standonck

# One possible representation of this data is

```
CREATE TABLE PhDStudent(  
  id INT,  
  sin INT,  
  name CHAR(20),  
  advisorID INT);
```

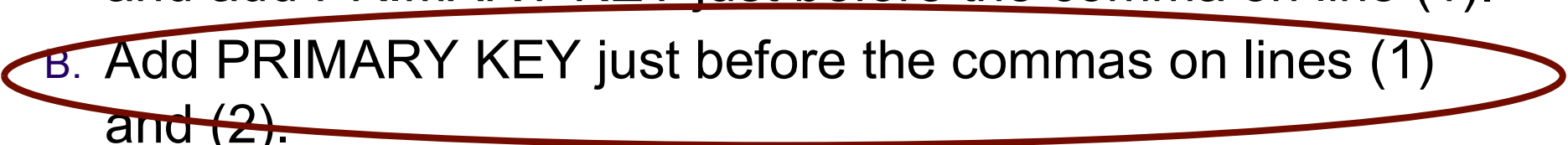


id	sin	name	AdvisorID
1	Null	Jan Standonck	Null
2	Null	Desiderious Erasmus	1

# Clicker question

Consider the table definition: CREATE TABLE PhDStudent(  
(1) id INT,  
Numbers denote lines only → (2) sin INT,  
(3) name CHAR(20),  
(4) advisorID INT);

Which of the following is **not** a legal addition?

- A. Add UNIQUE just before the commas on lines (2) and (3) and add PRIMARY KEY just before the comma on line (1).
- B. Add PRIMARY KEY just before the commas on lines (1) and (2).
- C. Add UNIQUE just before the comma on line (1), and add PRIMARY KEY just before the comma on line (2).
- D. All are legal
- E. None are legal

# Reasoning for the preceding clicker question

---

- B is not legal because it attempts to create two primary keys:

```
CREATE TABLE PhDStudent(  
  id INT PRIMARY KEY,  
  sin INT PRIMARY KEY,  
  name CHAR(20),  
  advisorID INT);
```

- Creating a complex primary key that consisting of the combination of id and sin, would be done as follows:

```
CREATE TABLE PhDStudent(  
  id INT,  
  sin INT,  
  name CHAR(20),  
  advisorID INT,  
  PRIMARY KEY (id, sin));
```

- Note, that this is a terrible idea because each of sin and id are keys by themselves

# Clicker question

- Consider the table definition: CREATE TABLE PhDStudent (
- (1) id INT,
  - (2) sin INT,
  - (3) name CHAR(20),
  - (4) advisorID INT );
- Numbers denote lines only →

Goal: have advisorID be foreign key reference for **same table** PhDStudent.

Which of the following is **not** legal? (does not have to achieve all goals)

- A. Add FOREIGN KEY (advisorID) REFERENCES PhDStudent(id) before the ) on line (4).
- B. Add PRIMARY KEY just before the comma on lines (1) and (2), and add REFERENCES PhDStudent(id) before the ) on line (4).
- C. Add PRIMARY KEY just before the comma on line (1), add UNIQUE just before the comma on line (2), and add FOREIGN KEY(advisorID) REFERENCES PhDStudent(sin) before the ) on line (4).
- D. All are legal
- E. None are legal

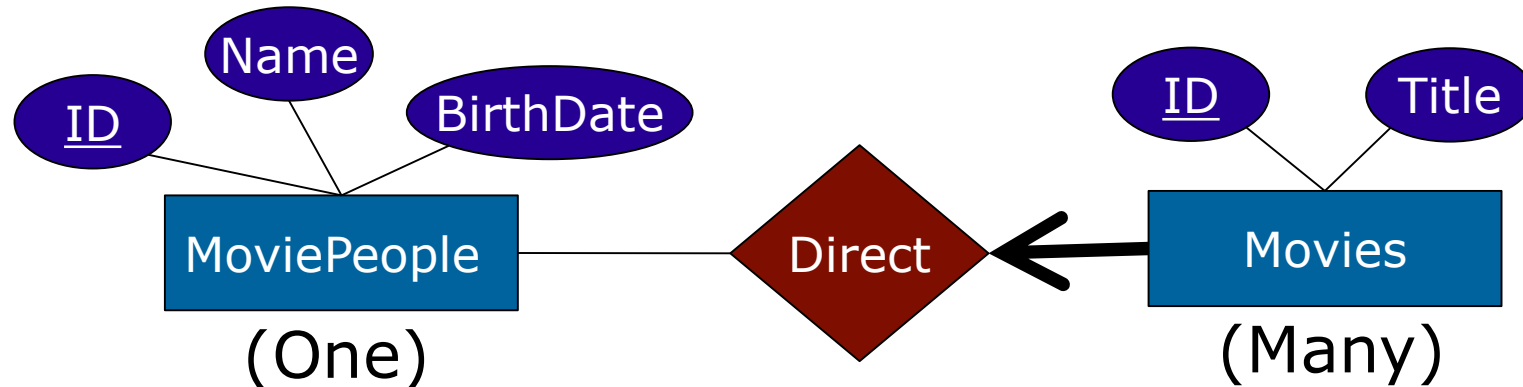
## Clicker question explanation

---

- A is incorrect because a foreign key needs to reference a primary key, and no primary key has been declared
- B is incorrect because it declares two primary keys and there can be only one primary key
- C is incorrect because the primary key is id, but it references sin, and a foreign key must reference the primary key

# Translating Participation Constraints

---



- Every movie must have a director.
  - Every tuple in the Movie table must appear with a non-null *MoviePeople ID* value
- How can we express that in SQL?

# Participation Constraints in SQL

- Using method 2 (add Directs relation in the Movie table), we can capture participation constraints by
  - ensuring that each **MID** is associated with a **MPID** that is not null
  - not allowing deletion of a director before the director is replaced

```
CREATE TABLE Directed_Movie(  
  MID      INTEGER,  
  title    CHAR(20),  
  MPID     CHAR(11) NOT NULL,  
  PRIMARY KEY (MID),  
  FOREIGN KEY (MPID) REFERENCES  
  MoviePeople  
  ON DELETE NO ACTION  
  ON UPDATE CASCADE)
```

- **Note: We cannot express this constraint if method 1 is used for Direct.**



# Participation Constraints in SQL

MoviePeople

<u>ID</u>	Name	Birthdate
1	Robert Pattinson	1986/05/13
2	James Cameron	1954/08/16

DirectedMovie

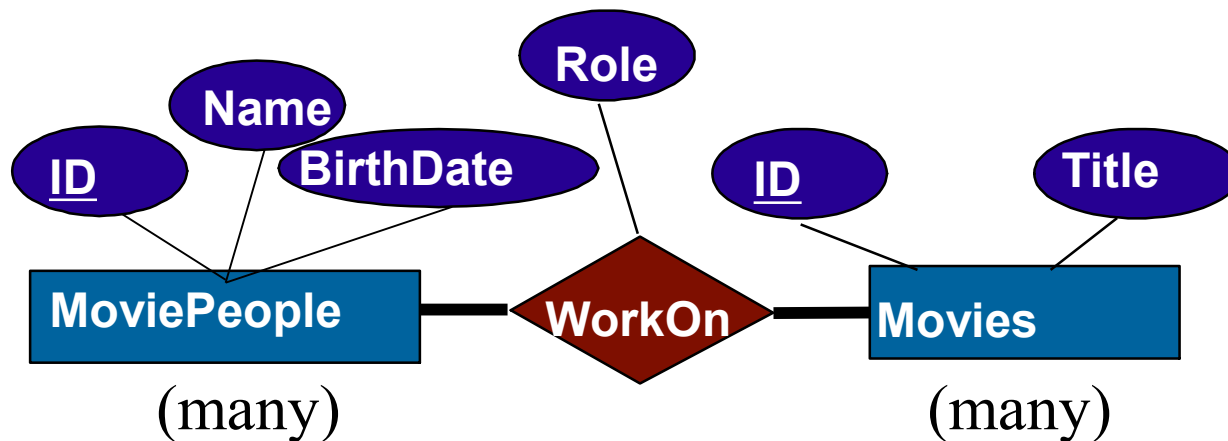
MID	Title	MPID is not null
1	Avatar	2
2	The Dark Knight	null

**Not legal!→**

Because we can't have a tuple in the DirectedMovie table with a null MPID, we can't insert any movies without directors. Therefore, all movies must participate in the Direct relationship

# Participation Constraints in SQL (cont')

---



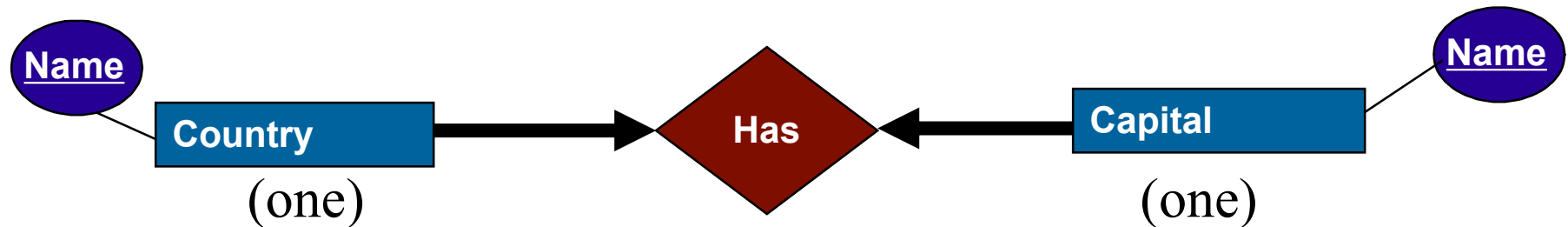
- How can we express that “every movie person works on a movie and every movie has some movie person in it”?
- Neither foreign-key nor not-null constraints in WorkOn can do that.
- We need assertions (covered later in the SQL module)

## Let's see why we can't model this participation constraint using null restrictions

MoviePeople	<u>ID</u>	Name	Birthdate
	1	Robert Pattinson	1986/05/13
	2	James Cameron	1954/08/16
Movie	<u>ID</u>	Title	
	1	Oppenheimer	
	2	Avatar	
Workon	MPID	MID	Role
	2	2	Director

No nulls, but Robert Pattinson does not work on a movie and Oppenheimer has no one working on it

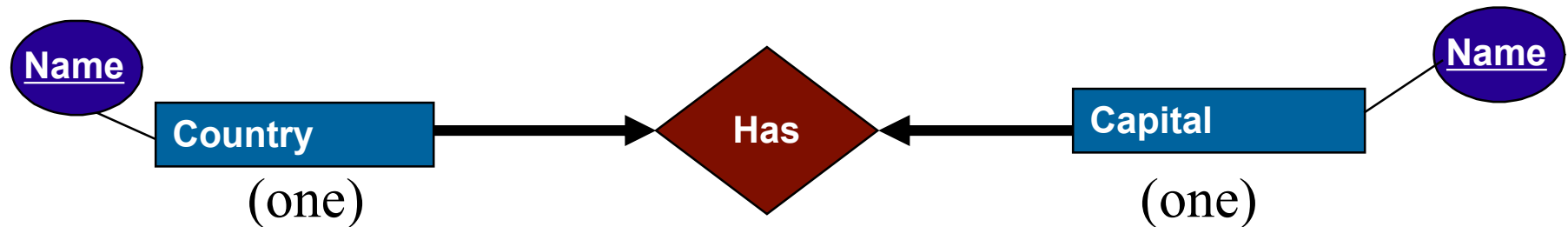
# Relationship Sets with Key Constraints (one to one case)



Which schema below is a reasonable translation Of “Has” from ER to relations? (bolded attributes are foreign keys)

- A. Country(coName, **caName**) – caName not null
- B. Country(name), Capital(name)
- C. Capital (caName, **coName**) – coName not null
- D. Both A and C
- E. All of A, B, and C

# Relationship Sets with Key Constraints (one to one case)



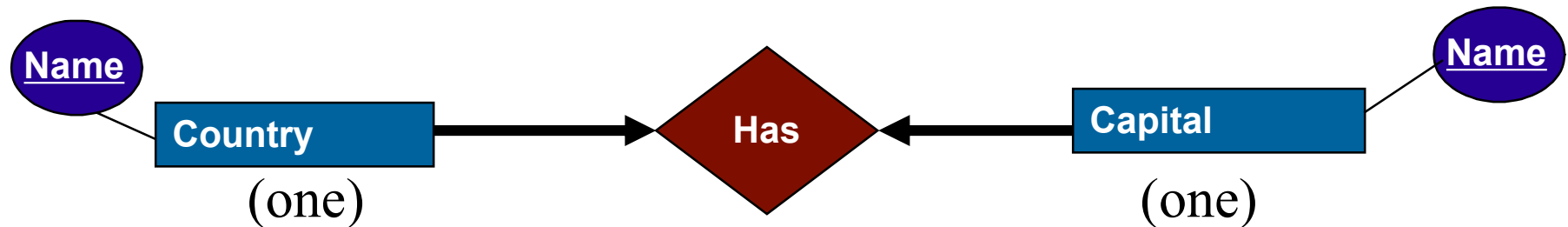
Let's assume we went with Country(coName, caName). Do we need a separate relation for Capital?

- A. Yes
- B. No**
- c. It depends

In this case there are no additional attributes in Capital, and both have total participation so no is also an acceptable answer. But if there were extra Capital attributes or there was not total participation, maybe yes

The goal on the exam is to ask you questions where the answer is clear!

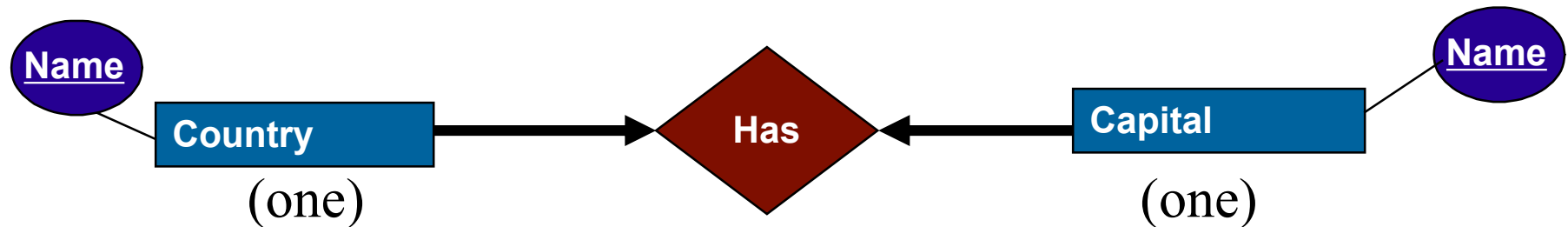
# Details, details



Assume you went with Country(coName, caName) and all attributes have type Char(20) and we're not creating a separate relation for Capital. Given what we've discussed so far, write the SQL DDL that you would need for this relation.

```
CREATE TABLE Country(  
    country-name CHAR(20) PRIMARY KEY,  
    capital-name CHAR(20) NOT NULL,  
    UNIQUE capital-name ← needed for one-to-one constraint  
)
```

# Details, details



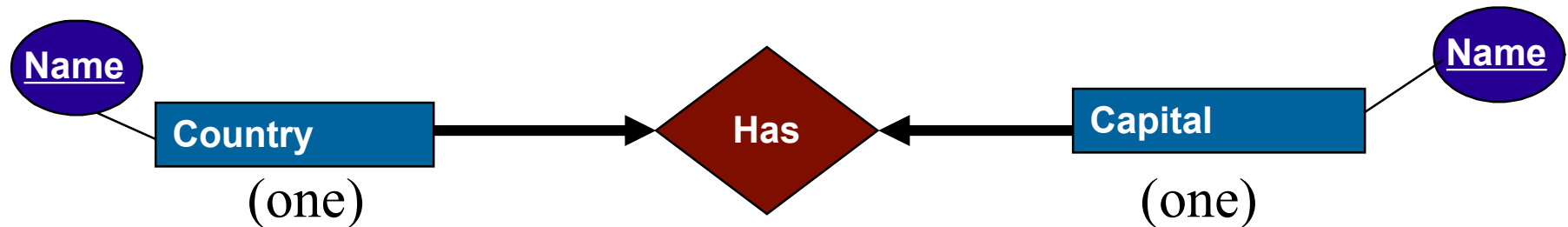
Assume Country(coName, caName) and all attributes of type Char(20) and we're not creating a separate relation for Capital.

```
CREATE TABLE Country(  
    country-name CHAR(20) PRIMARY KEY,  
    capital-name CHAR(20) NOT NULL,  
    UNIQUE capital-name); ← needed for one-to-one constraint)
```

If we did have a separate table for Capital, would we still need the unique constraint?

A. Yes. B. No

# Details, details



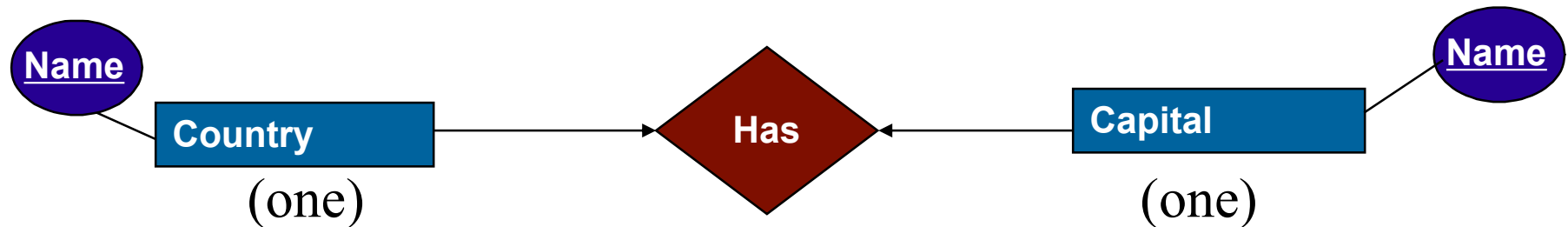
Assume Country(coName, caName) and all attributes of type Char(20) and we're not creating a separate relation for Capital.

```
CREATE TABLE Country(  
    country-name CHAR(20) PRIMARY KEY,  
    capital-name CHAR(20) NOT NULL,  
    UNIQUE capital-name); ← needed for one-to-one constraint)
```

<u>coName</u>	caName
Canada	Ottawa
United States	Ottawa
Mexico	Ottawa



# Clicker Exercise Explained



What happens when we create a separate table for capital?  
Do we still need UNIQUE?

Country(name, **capital**)

<u>Name</u>	Capital
Canada	
USA	
Mexico	

Capital(name)

<u>Name</u>
Ottawa
Washington, D.C.
Mexico City

What values are allowed in the Capital column?

# Clicker Exercise Explained

---

Country(name, **capital**)

<u>Name</u>	Capital
Canada	Ottawa
USA	Ottawa
Mexico	Ottawa

Capital(name)

<u>Name</u>
Ottawa
Washington, D.C.
Mexico City

What's to stop Ottawa from appearing multiple times in the Country table? Nothing! The data instance doesn't violate the foreign key constraint so it's a legal data instance.

However, it doesn't satisfy our one to one constraint! We need the UNIQUE constraint to ensure that Ottawa only appears once in the Country table.

# In Class Exercise: Relational Model 1

---