Administrative Notes- Jan 26, 2023

- Jan 27: Assignment 1 due
- Assignment 2 available on Canvas
- Assignment 3 released on Friday night
  - You are allowed to complete this in pairs (more instructions on how to sign up as a pair in the assignment description released on Friday night)
- You will need a physical student card for the class so please get one

Review

Student

sid	name	address	phone	major
99111120	K. Jones	1234 W. 12th Ave., Van	889-4444	CPSC
92001200	S. Selvarajah	2020 E. 18th St., Van	409-2222	MATH
94001020	A. Alberty	2020 E. 18th St., Van	222-2222	FREN
94001150	J. Wang	null	null	null

Student(sid: integer, name: string, address: string, phone: string, major: string)

Or without the domains:

Student (sid, name, address, phone, major)

Review

- The statement on the right creates the Student relation
  - the type (domain) of each attribute is specified and enforced when tuples are added or modified

```
CREATE TABLE Student
        (sid      INTEGER,
         name     CHAR(20),
         address  CHAR(30),
         phone    CHAR(13),
         major    CHAR(4))
```

- The statement on right creates Grade information about courses that a student takes

```
CREATE TABLE Grade
(sid        INTEGER,
 dept       CHAR(4),
 course#    CHAR(3),
 mark       INTEGER)
```

Keys Constraints (for Relations)

- Similar to those for entity sets in the ER model
- One or more attributes in a relation form a ***key*** (or ***candidate key***) for a relation, where S is the set of all attributes in the key, if:
  1. No distinct tuples can have the same values for all attributes in the key, and
  2. No subset of S is itself a key (according to (1)).

  (If such a subset exists, then S is a *superkey* and not a key.)
- One of the possible keys is chosen (by the DBA) to be the ***primary key*** (PK).

```
CREATE TABLE Student
        (sid      INTEGER  PRIMARY KEY,
         name     CHAR(20),
         address  CHAR(30),
         phone    CHAR(13),
         major    CHAR(4))
```

Keys Constraints in SQL

- A **PRIMARY KEY** constraint specifies a table's primary key
  - values for primary key must be unique
  - a primary key attribute cannot be *null*
- Other keys are specified using the **UNIQUE** constraint
  - values for a group of attributes must be unique (if they are not null)
  - these attributes can be *null*
- Key constraints are checked when
  - new values are inserted
  - values are modified

Clicker Question

Does the constraint PRIMARY KEY(Dept, Course#) hold for this instance?

Course#	Dept	Title
100	CPSC	Computational Thinking
100	MATH	Differential Calculus with Applications to Physical Sciences and Engineering

[✓ A. Yes]

B. No

C. It depends

Clicker Question

Does the constraint PRIMARY KEY(Dept, Course#) hold for this instance?

Dept	Course #	Term	Section
CPSC	368	2021W2	201
CPSC	368	2021W2	202

A. Yes

[✓ B. No]

C. It depends

Keys Constraints in SQL (cont’)

(Ex.1- Normal) “For a given student and course, there is a single grade.”

```
CREATE TABLE Grade
  (sid        INTEGER,
   dept       CHAR(4),
   course#  CHAR(3),
   mark       INTEGER,
   PRIMARY KEY  (sid,dept,course#) )
```

vs.

(Ex.2 - Silly) “Students can take a course once, and receive a single grade for that course; further, no two students in a course receive the same grade.”

```
CREATE TABLE Grade2
  (sid        INTEGER,
   dept       CHAR(4),
   course#  CHAR(3),
   mark       CHAR(2),
   PRIMARY KEY (sid,dept,course#),
   UNIQUE (dept,course#,mark) )
```

Keys Constraints in SQL (cont')

For single attribute keys, can also be declared on the same line as the attribute.

```
CREATE TABLE Student
        (sid        INTEGER  PRIMARY KEY,
         name       CHAR(20),
         address    CHAR(30),
         phone      CHAR(13),
         major      CHAR(4))
```

Foreign Keys Constraints

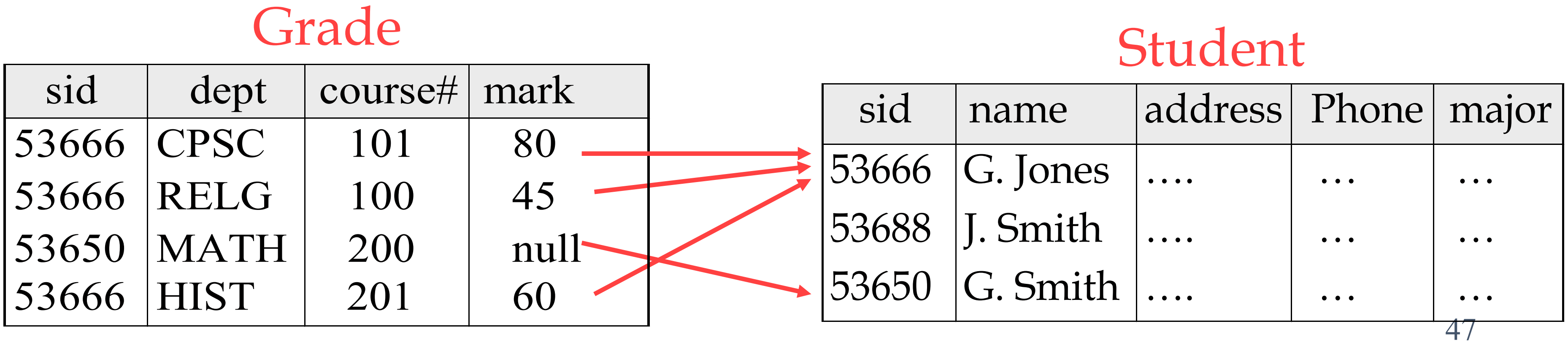
- ***Foreign key*** : Set of attributes in one relation used to 'reference' a tuple in another relation.
  - **Must correspond to the primary key of the other relation.**
  - Like a 'logical pointer'.
- E.g.: Grade(*sid, dept, course#, grade*)
  - *sid* is a foreign key referring to Student:
  - (*dept, course#*) is a foreign key referring to Course
- *Referential integrity*: All foreign keys reference existing entities.
  - i.e. there are no dangling references
  - all foreign key constraints are enforced

Foreign Keys in SQL

Only students listed in the Student relation should be allowed to have grades for courses that are listed in the Course relation.

```
CREATE TABLE Grade
   (sid INTEGER,  dept  CHAR(4), course# CHAR(3), mark INTEGER,
    PRIMARY KEY  (sid,dept,course#),
    FOREIGN KEY (sid) REFERENCES Student,
    FOREIGN KEY (dept, course#) REFERENCES Course(dept, cnum))
```

Sometimes you can not specify which attributes are referenced, but in this case they are needed. Never hurts to include them!



Grade

sid	dept	course#	mark
53666	CPSC	101	80
53666	RELG	100	45
53650	MATH	200	null
53666	HIST	201	60

Student

sid	name	address	Phone	major
53666	G. Jones	….	…	…
53688	J. Smith	….	…	…
53650	G. Smith	….	…	…

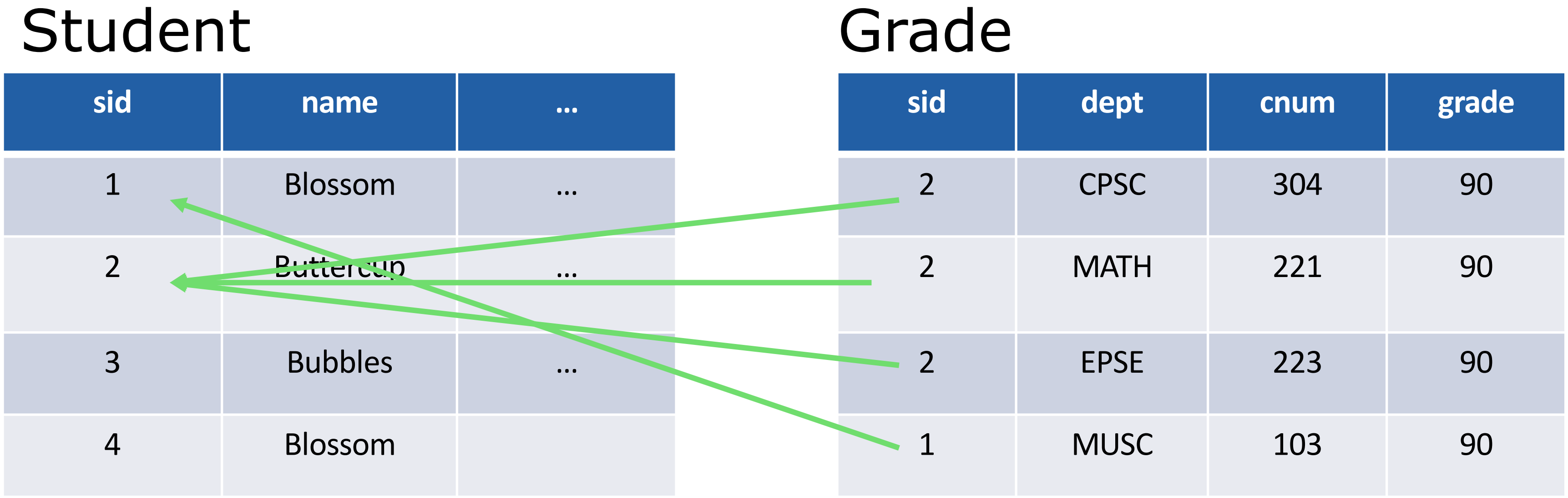
Enforcing Referential Integrity

Grade

sid	dept	cnum	grade
2	CPSC	304	90
2	MATH	221	90
2	EPSE	223	90
1	MUSC	103	90

```
CREATE TABLE Grade (
        sid INTEGER,
        dept  CHAR(4),
        course# CHAR(3),
        mark INTEGER,
        PRIMARY KEY  (sid,dept,course#),
        FOREIGN KEY (sid) REFERENCES
                Student,
        FOREIGN KEY (dept, course#)
                REFERENCES Course(dept, cnum)
    )
```

Enforcing Referential Integrity



Student

sid	name	…
1	Blossom	…
2	Buttercup	…
3	Bubbles	…
4	Blossom	

Grade

sid	dept	cnum	grade
2	CPSC	304	90
2	MATH	221	90
2	EPSE	223	90
1	MUSC	103	90

A foreign key is a set of attributes in one relation (e.g., Grades.sid) used to ‘reference’ a tuple in another relation (e.g., Students.sid).

Enforcing Referential Integrity

- *sid* in Grade is a foreign key that references Student.
- What should be done if a Grade tuple with a non-existent student id is inserted? (*Reject it!*)
- What should be done if a **Student tuple** is deleted?
    - Also delete all Grade tuples that refer to it?
    - Disallow deletion of this particular Student tuple?
    - Set sid in Grade tuples that refer to it, to *null*, (the special value denoting *`unknown’* or *`inapplicable’*.)
        - Problem if sid is part of the primary key
    - Set sid in Grade tuples that refer to it, to a *default sid*.
- Similar if primary key of a Student tuple is updated

Referential Integrity in SQL/92

- SQL/92 supports all 4 options on deletes and updates.
  - Default is **NO ACTION** (*delete/update is rejected*)
  - **CASCADE** (also updates/deletes all tuples that refer to the updated/deleted tuple)
  - **SET NULL / SET DEFAULT** (referencing tuple value is set to the default foreign key value)

```
CREATE TABLE Grade
  (sid CHAR(8),  dept  CHAR(4),
    course# CHAR(3),  mark INTEGER,
  PRIMARY KEY  (sid,dept,course#),
  FOREIGN KEY (sid)
      REFERENCES Student(sid)
       ON DELETE CASCADE
       ON UPDATE CASCADE
  FOREIGN KEY (dept, course#)
      REFERENCES
       Course(dept,course#)
       ON DELETE SET DEFAULT
       ON UPDATE CASCADE );
```

Clicker Question

Consider the following table definition.

```
CREATE TABLE  BMW  ( bid  INTEGER, sid INTEGER,  …
           PRIMARY KEY (bid),
           FOREIGN KEY (sid) REFERENCES STUDENTS
                ON DELETE CASCADE);
```

If bid = 1000 and sid = 5678 for a row in Table BMW, choose the best answer.

A. If the row for sid value 5678 in STUDENTS is deleted, then the row with bid = 1000 in BMW is automatically deleted.

B. If a row with sid value 5678 in BMW is deleted, then the row with sid=5678 in STUDENTS is automatically deleted.

C. Both of the above.

Clicker Question

Consider the following table definition.

```
CREATE TABLE  BMW  ( bid  INTEGER, sid INTEGER,  …
           PRIMARY KEY (bid),
           FOREIGN KEY (sid) REFERENCES STUDENTS
                ON DELETE CASCADE);
```

A. [✓ If the row for sid value 5678 in STUDENTS is deleted, then the row with bid = 1000 in BMW is automatically deleted.]

B. If a row with sid value 5678 in BMW is deleted, then the row with sid=5678 in STUDENTS is automatically deleted.

C. Both of the above.

BMW

bid	Sid
1000	5678

Student

sid	name	Address
5678	James	Null

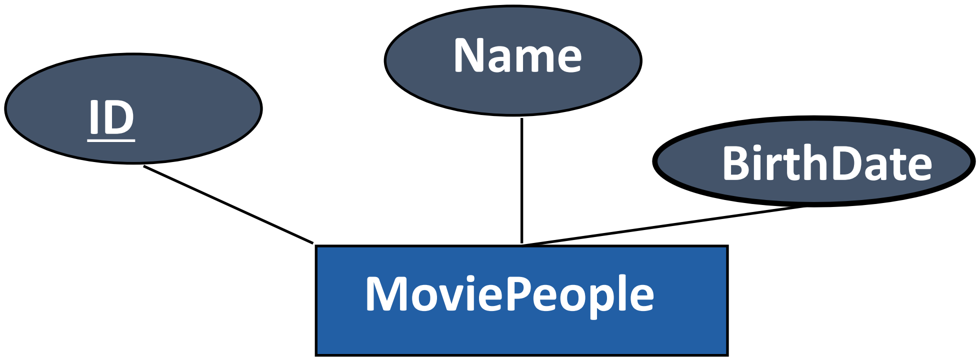
Where do ICs Come From?

- ICs are based upon the real-world semantics being described (in the database relations).
- We *can* check a database instance to verify an IC, but we *cannot* tell the ICs by looking at the instance.
  - For example, even if all student names differ, we cannot assume that name is a key.
  - An IC is a statement about *all possible* instances.
- All constraints must be identified during the conceptual design.
- Some constraints can be explicitly specified in the conceptual model
  - Key and foreign key ICs are shown on ER diagrams.
- Others are written in a more general language.

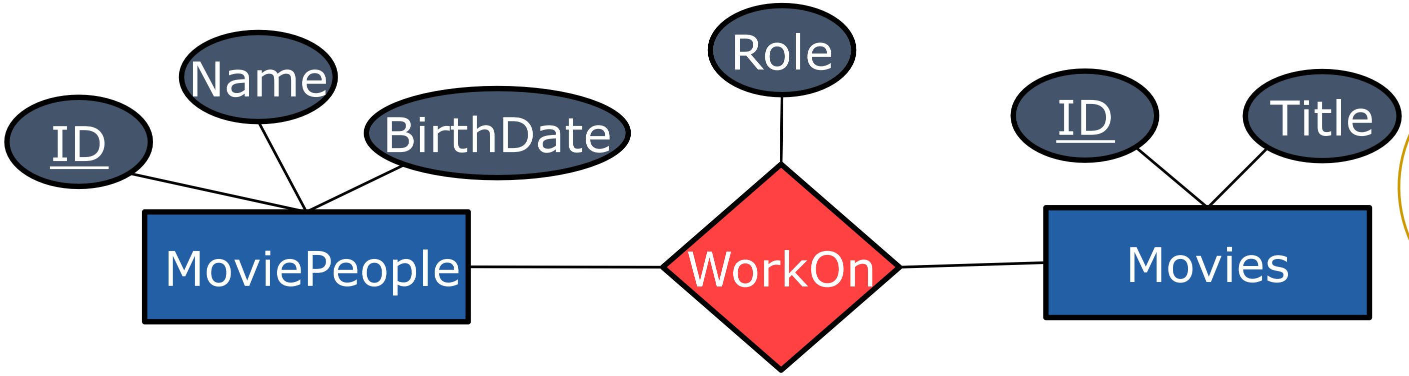
Logical DB Design: ER to Relational

- Each entity set is mapped to a table.
  - Entity attributes become table attributes
  - Entity keys become table keys

```
CREATE TABLE MoviePeople
      (ID     CHAR(11),
       Name CHAR(20),
       BirthDate DATE,
       PRIMARY KEY  (ID))
```



Relationship Sets to Tables

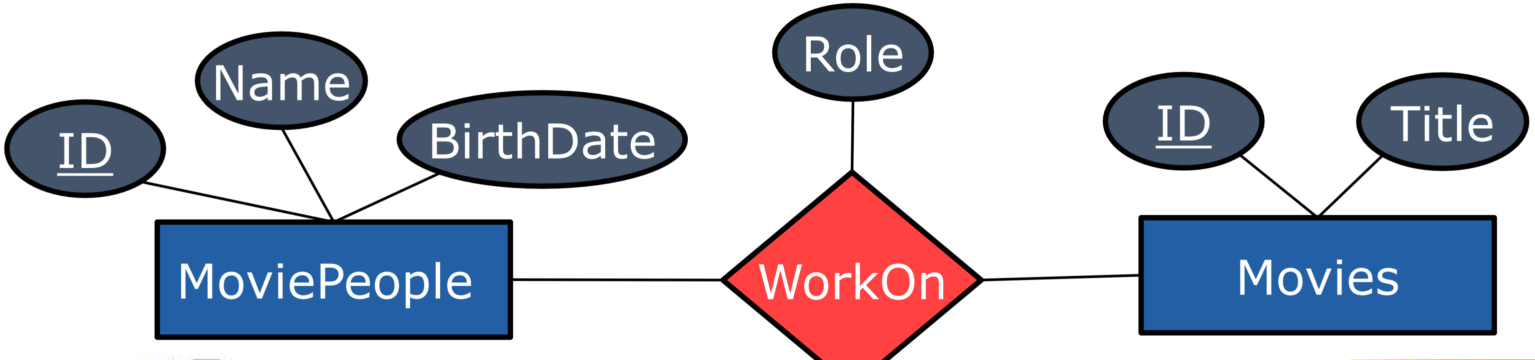


MPID and MID CANNOT be null

- A relationship set id is mapped to a single relation (table).
- Simple case: relationship has no constraints (i.e. many-to-many)
- In this case, attributes of the table must include:
  - Keys for each participating entity set as foreign keys.
    - This is a *key* for the relation.
  - All descriptive attributes.

```
CREATE TABLE WorkOn(
 MPID      CHAR(11),
 MID     INTEGER,
 Role     CHAR(20),
 PRIMARY KEY (MPID, MID),
 FOREIGN KEY (MPID)
     REFERENCES MoviePeople,
 FOREIGN KEY (MID)
     REFERENCES Movies)
```

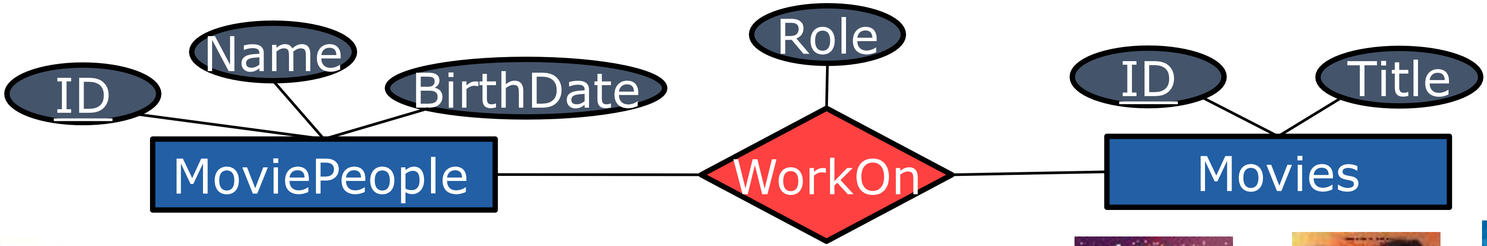
Example: Many to Many Relationships



Scrooge McDuck has worked on two movies

Pua has worked on one movie (Moana)

Example: Many to Many Relationships



Can we reduce redundancy by combining these tables in any way?

MoviePeople

ID	Name	Birthdate
1	Scrooge	…
2	Pua	…

WorkOn

MPID	**MID**	Role
1	1	…
1	2	…
2	3	…

Movies

ID	Title
1	Ducktales
2	A Christmas Carol
3	Moana

Example: Many to Many Relationships

MoviePeople

ID	Name	Birthdate
1	Scrooge	…
2	Pua	…

Movies

ID	Title
1	Ducktales
2	A Christmas Carol
3	Moana

WorkOn

MPID	**MID**	Role
1	1	…
1	2	…
2	3	…

Can we integrate the information in WorkOn into MoviePeople?

Example: Many to Many Relationships

MoviePeople

ID	Name	Birthdate	**MID**
1	Scrooge	…	
2	Pua	…	

Movies

ID	Title
1	Ducktales
2	A Christmas Carol
3	Moana

WorkOn

MPID	**MID**	Role
1	1	…
1	2	…
2	3	…

Do we put MID 1 or MID 2? We can’t have both in the column (i.e., we can’t store a list there)

Not much we can do in terms of reducing tables

Example: Many to Many Relationships

MoviePeople

ID	Name	Birthdate
1	Scrooge	…
2	Pua	…

Movies

ID	Title	MPID
1	Ducktales	
2	A Christmas Carol	
3	Moana	

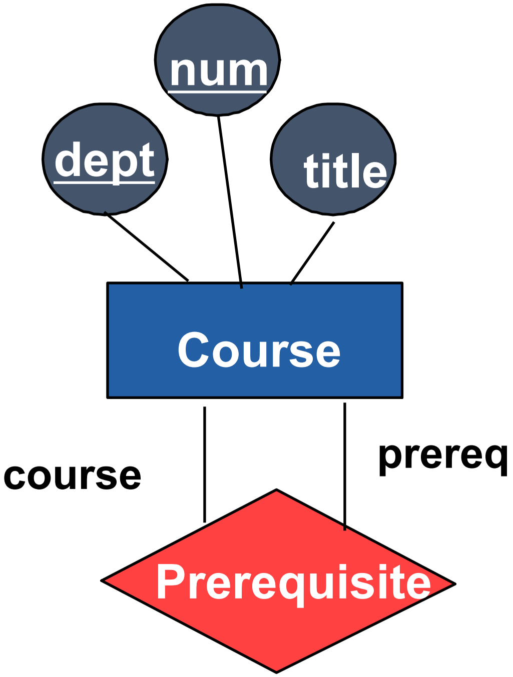
WorkOn

MPID	MID	Role
1	1	…
1	2	…
2	3	…

What if we have more than one person work on this movie? Same problem as before!

Not much we can do in terms of reducing tables

Relationship Sets to Tables (cont’)



- In some cases, we need to use the roles:

```
CREATE TABLE Prerequisite(
  course_dept     CHAR(4),
  course_num      CHAR(3),
  prereq_dept     CHAR(4),
  prereq_num     CHAR(3),
  PRIMARY KEY (course_dept, course_num,
                         prereq_dept, prereq_num),
  FOREIGN KEY (course_dept, course_num)
      REFERENCES Course(dept, num),
  FOREIGN KEY (prereq_dept, prereq_num)
      REFERENCES Course(dept, num))
```

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 Beeri
- 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
- Desiderious Erasmus
- Jan Standonck

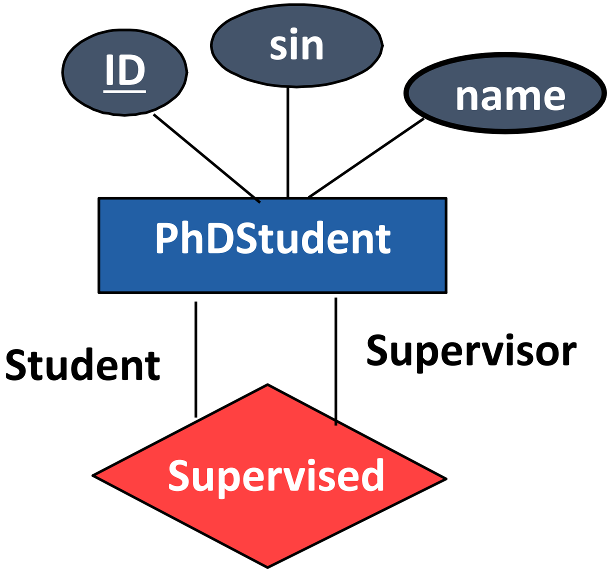
One possible partial representation of this data is

```
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

One possible partial 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

Self Referencing Relations

Goal: have Advisor be foreign key reference for same table PhDstudent

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

Could a foreign key be null?

For referential integrity to hold in a relational database, any field in a table that is declared a foreign key should contain either a null value, or only values from a parent table's primary key.

Clicker Question

Clicker Question

Consider the table definition: CREATE TABLE PhDStudent(

Numbers denote lines only ->

(1) id INT,
(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

- 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,
Numbers denote lines only -> (2) sin INT,
(3) name CHAR(20),
(4) advisorID INT);

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).

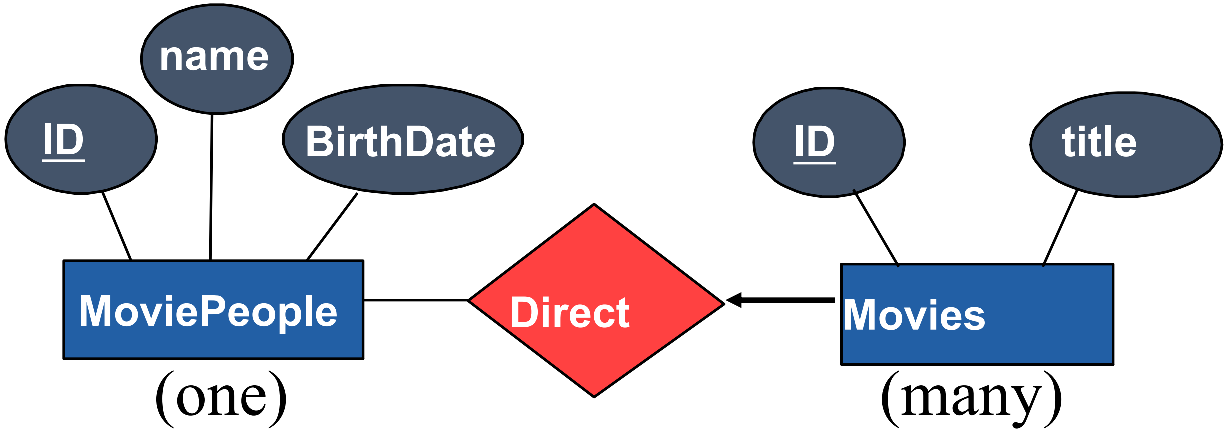
B. Add PRIMARY KEY just before the comma on lines (1) and (2), and add REFERENCES PhDStudent(id) before the) on line (4).

C. Add PRIMARY KEY just before the comma on line (1), add UNIQUE just before the comma on line (2), and add FOREIGN KEY REFERENCES PhDStudent(sin) before the) on line (4).

D. All are legal

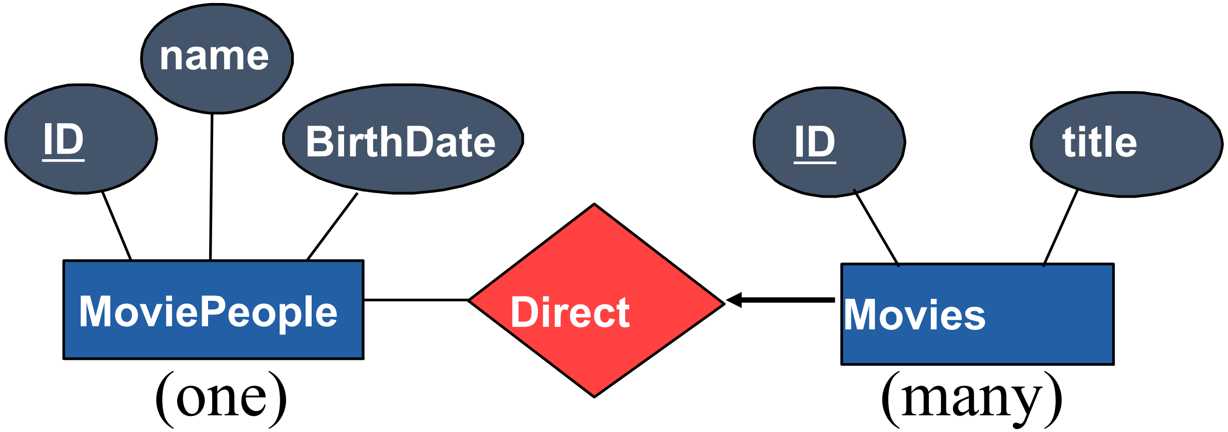
[✓ E. None are legal]

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?

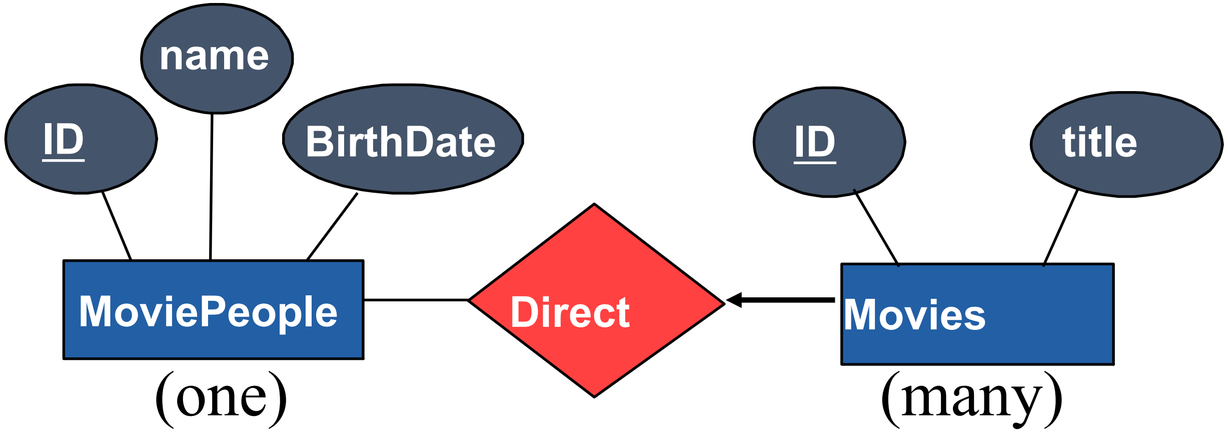
For the one to many case, combine the many side with the relationship



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



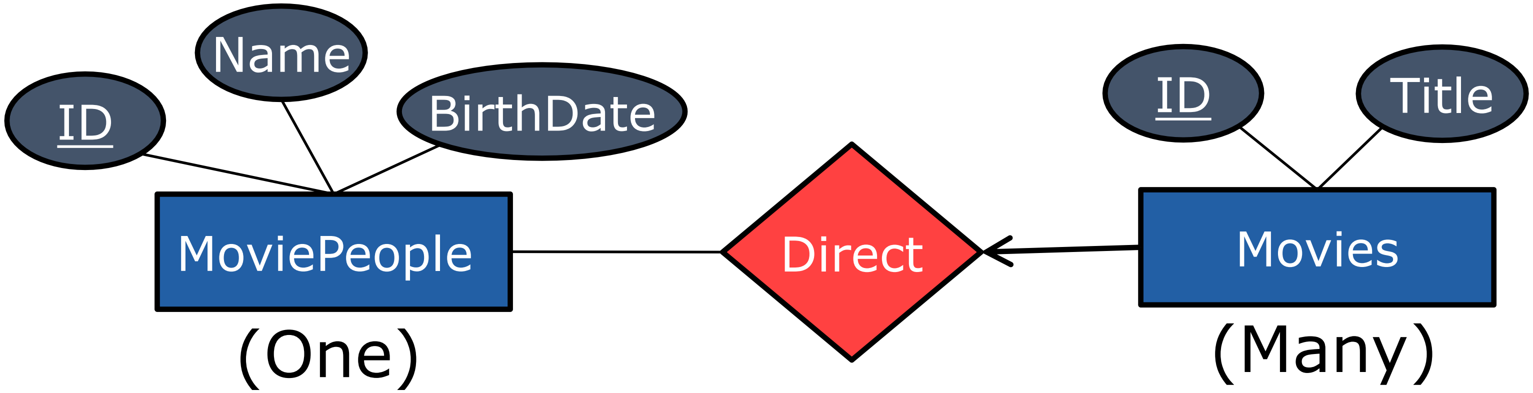
MoviePeople

ID	Name	Birth Date
1		
2		

Movies

ID	MoviePerson ID	Title
1	1	
2	1	
3	2	

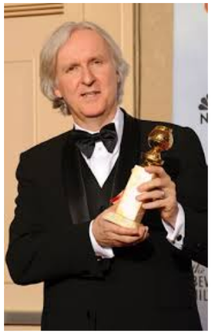
For the one to many case, combine the many side with the relationship

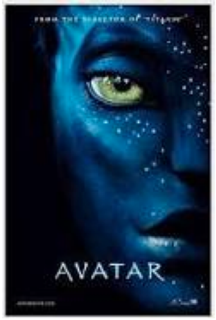

ID
Name
BirthDate
MoviePeople
Direct
ID
Title
Movies

(One)

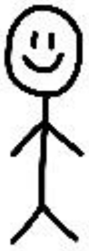
(Many)

James Cameron





You



The Life of a CS Student

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

MoviePeople

ID	Name	Birthdate
1	James Cameron	…
2	You	…

Movies

ID	Title
1	Avatar
2	Titanic
3	The Life of a CS Student

Direct

MPID	MID
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

ID	Name	Birthdate	Directed MID
1	James Cameron	…	
2	You	…	

Movies

ID	Title
1	Avatar
2	Titanic
3	The Life of a CS Student

Direct

MPID	MID
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

ID	Name	Birthdate
1	James Cameron	…
2	You	…

Movies

ID	Title
1	Avatar
2	Titanic
3	The Life of a CS Student

Direct

MPID	MID
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

ID	Name	Birthdate
1	James Cameron	…
2	You	…

Movies

ID	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?

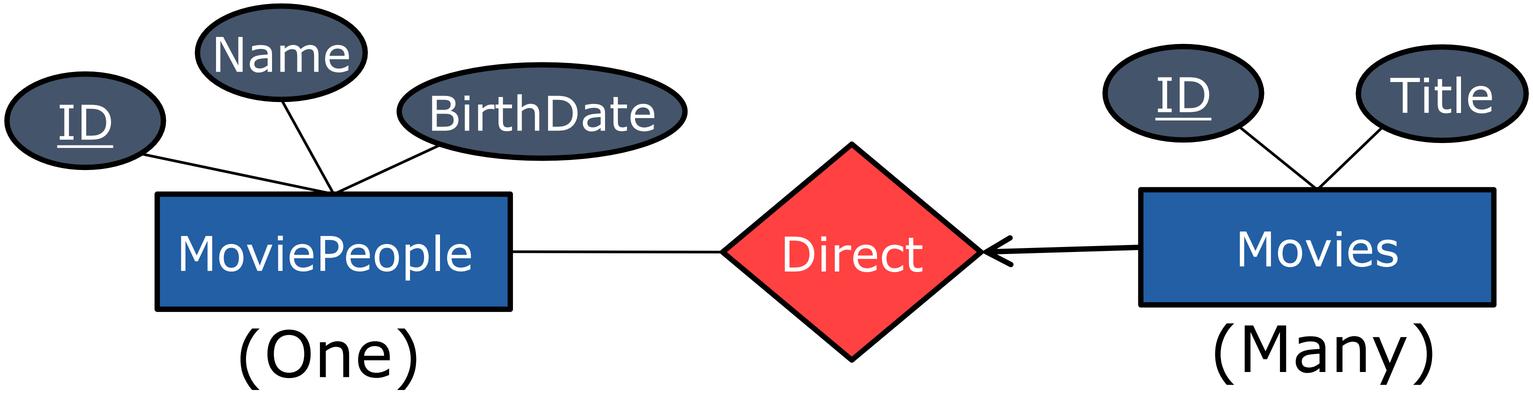
No! (because of our one to many relationship)

We’d covered basic translating of ER to relational

- Short version: everything’s a table

- Slightly longer version: in many to many relationships, create one table per entity and one table per relationship. Link the two by foreign keys

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



MoviePeople

ID	Name	Birthdate
1	James Cameron	…
2	You	…

Movies

ID	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.

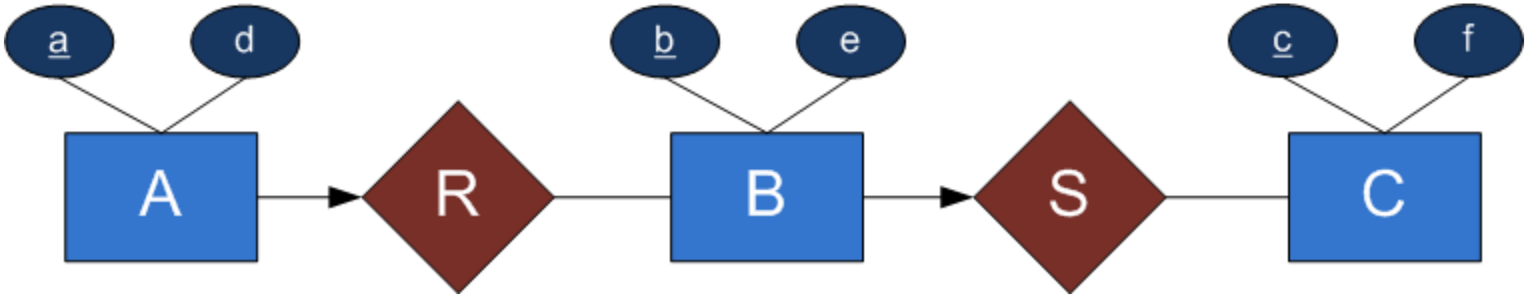
```
CREATE TABLE  Direct(
  MPID     CHAR(11),
  MID    INTEGER,
  PRIMARY KEY  (MID),
  FOREIGN KEY (MPID) REFERENCES
MoviePeople,
  FOREIGN KEY (MID) REFERENCES Movies)
```

- 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”

In-Class Exercise (no need to hand it in)

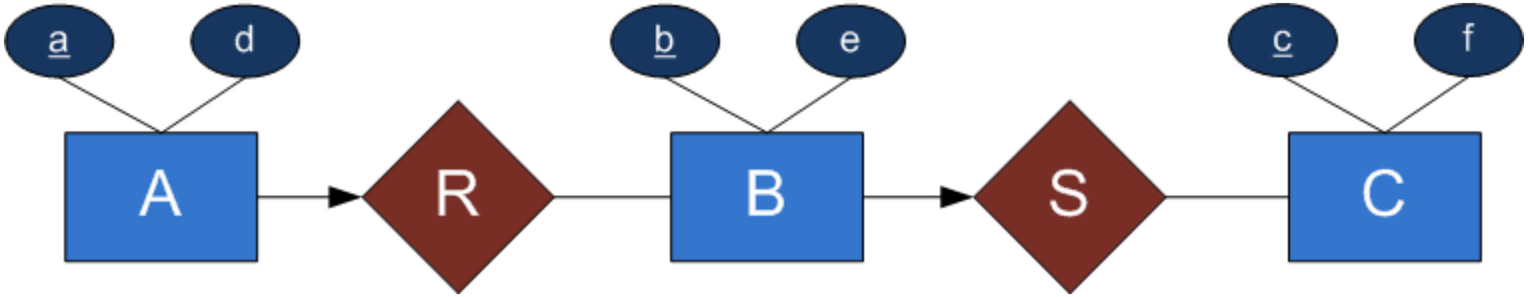


Translate the ER diagram to relational. Underline key attributes, and make FKs bold (or circle the FK if you are doing this by hand).

Back at 3:01

Work with the people around you!

Clicker Question



Translate the ER diagram to relational.

Which of the following appears in your relational schema:

A. AR(a,**b**,d)

B. BS(b,**c**,e)

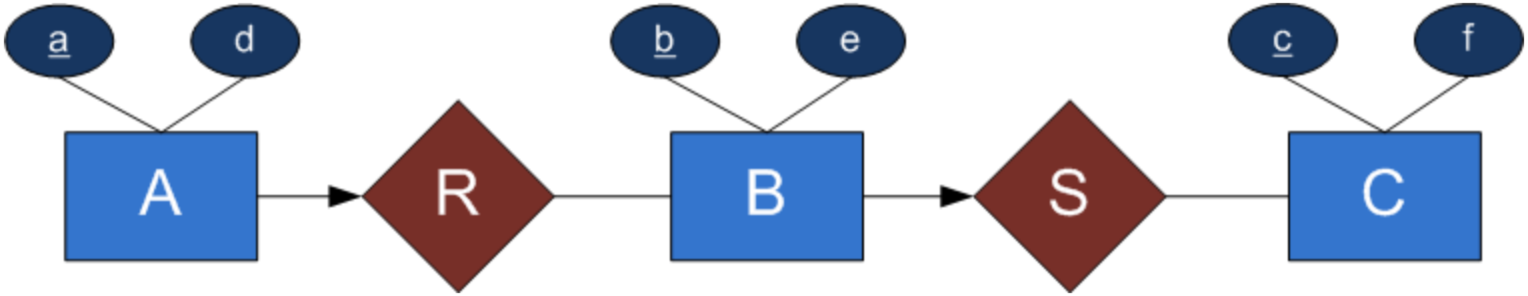
C. S(b,c)

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(a,**b**,d)

[✓ B. BS(b,**c**,e)]

C. S(b,c)

D. All of these

E. None of these

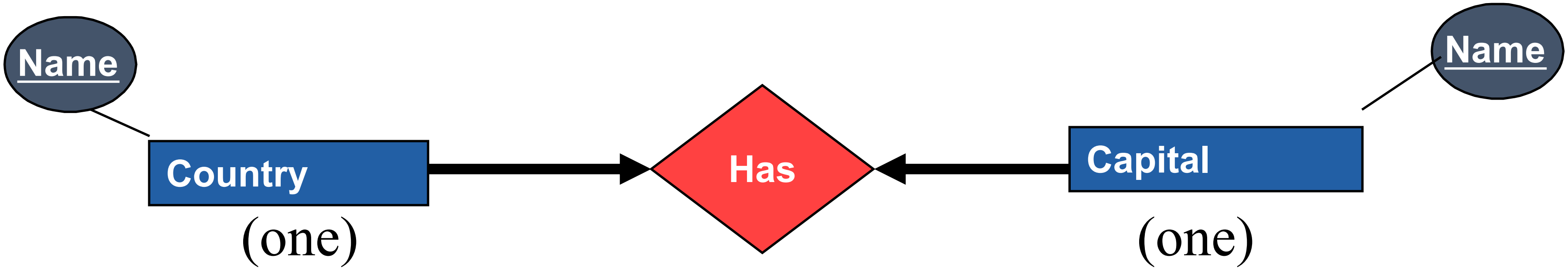
~~A~~
~~B~~
~~C~~
~~R~~
~~S~~

AR
BS
C

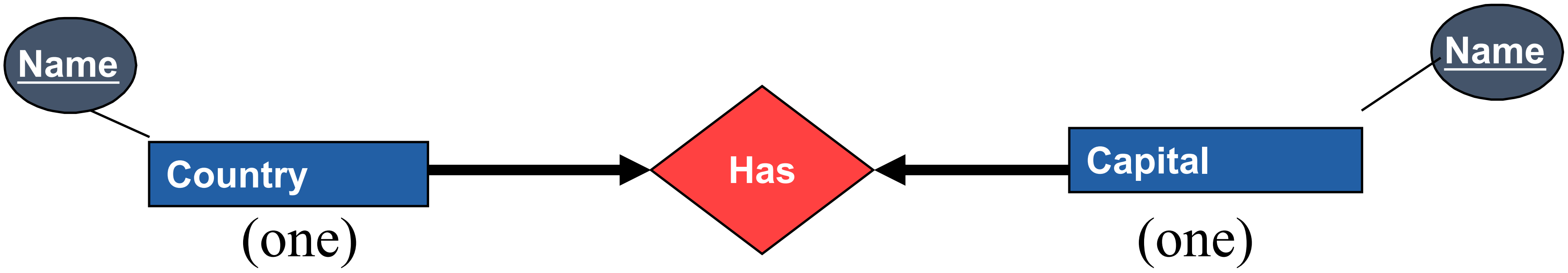
→

AR(a,**b**,d)
BS(b,**c**,e)
C(c,f)

Relationship Sets with Key Constraints (one to one case)



Relationship Sets with Key Constraints (one to one case)



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

A. Country(coName, **caName**)

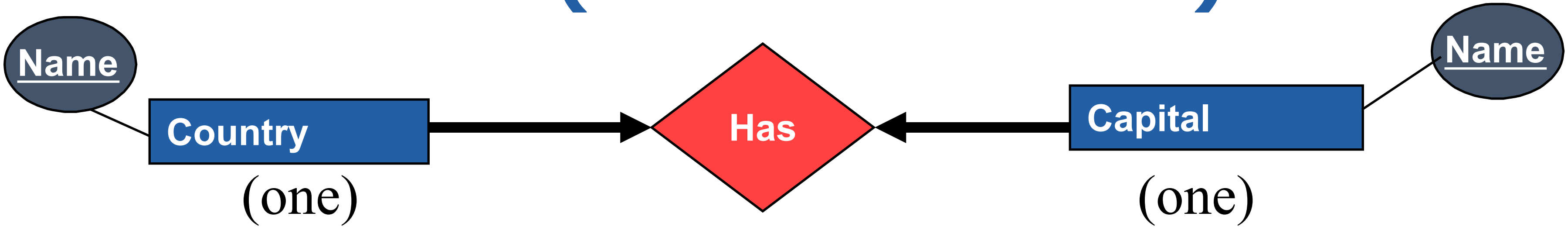
B. Country(name), Capital(name)

C. Capital (caName, **coName**)

[✓ 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. 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!