CS2102: Database Systems

ER Model to Schema Translation

Introduction

Introduction

Intro

Why

We have seen how some ER diagrams are translated into schema in lecture. In this *extended* notes, we will look at

- why those translation techniques are correct
- why other translation techniques are not *fully* correct
- what translation techniques you can use if your ER diagram does not match the technique mentioned
- what happen if some constraints cannot be enforced

Indeed, there are some ER diagram that does not match the known translation technique but can actually be translated with all constraints satisfied.

Introduction

General Technique

Three Main Constraints

There are three main constraints that need to be satisfied on a general ER diagram *(excluding ISA hierarchy for a moment)*. These are

1. Are the key attributes uniquely identifying the rest of the attributes?
   - Should not uniquely identifying additional attributes not part of the entity set unless bound constraints enforces it.
2. Are the lower bound constraints satisfied
   - Can you have 0 participation?
   - Must you have 1 participation?
3. Are the upper bound constraints satisfied
   - Can you have ∞ participation?
   - Must you have 1 participation?

Assume that the data type are the *most reasonable* type.

Introduction

1. Uniquely Identifying

Preliminary

> This constraint is related to the concept of "entity" in an entity set. A student is uniquely identified by their "matric number". But this is only true for attributes directly related to the entity set student.
>
> For instance, a matric number may not uniquely identify the courses the student is taking since now we are talking about the relationship between student and course.
>
> This is typically enforced by `PRIMARY KEY` and/or `UNIQUE NOT NULL` constraint. You still have to be careful not to add attributes from other constructs *(e.g., the course as above)* as it means that now student can uniquely identify those too.
>
> This discussion will side-step the issue with *multi-valued* attributes since by definition, those cannot be *uniquely identified* by the key attribute.

2. Lower Bound Constraints

Preliminary

> This constraint is related to the participation constraints. Total participation constraint requires the entity set to participate at least once in the relationship.
>
> If there is no total participation constraint *(typically called partial participation constraint)*, then the entity set may not participate in the relationship.

3. Upper Bound Constraints

Preliminary

> This constraint is related to the cardinality constraint. Cardinality constraint requires the entity set to participate at most once in the relationship. This is typically called key constraint because it can be enforced by making the key attribute of the entity set to be the key attribute of the relationship set.
>
> If there is no restriction on cardinality constraint, then the entity set may participate in the relationship as many times as it needs while still respecting the set property of the relationship set.

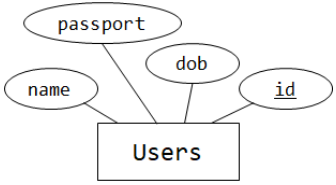
Simple Translation

Simple Translation

Entity Set

Preliminary

> **ER Diagram**
>
> Consider the following ER diagram on the right. Assume that the ER diagram consists *only* the given entity set *(never really happen in practice but we have to start from a simple example)*.



Schema

```
CREATE TABLE Users (
  id       INT PRIMARY KEY,
  dob      DATE,
  passport VARCHAR(20),
  name     VARCHAR(200)
);
```

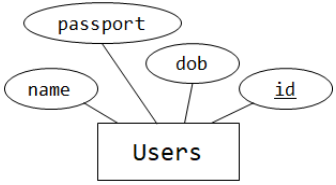
> Consider the three questions again:
>
> 1. Uniquely identify
> 2. ~~Lower bound~~
> 3. ~~Upper bound~~
>
> The last two are irrelevant for our current discussion

Simple Translation

Entity Set

Uniquely Identify

> **ER Diagram**
>
> Consider the following ER diagram on the right. Assume that the ER diagram consists *only* the given entity set *(never really happen in practice but we have to start from a simple example)*.



Schema

```
CREATE TABLE Users (
  id       INT PRIMARY KEY,
  dob      DATE,
  passport VARCHAR(20),
  name     VARCHAR(200)
);
```

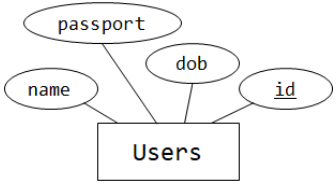
> **Question**
>
> Can you uniquely identify **dob**, ... from only **id**?

Simple Translation

Entity Set

Uniquely Identify

> **ER Diagram**
>
> Consider the following ER diagram on the right. Assume that the ER diagram consists *only* the given entity set *(never really happen in practice but we have to start from a simple example)*.



Schema

```
CREATE TABLE Users (
  id       INT PRIMARY KEY,
  dob      DATE,
  passport VARCHAR(20),
  name     VARCHAR(200)
);
```

> Is the following table allowed?

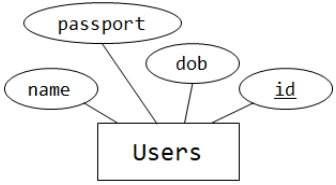
id	dob	passport	name
1	27-01-1990	P0	John
1	25-01-1990	P0	John

Simple Translation

Entity Set

Uniquely Identify

> **ER Diagram**
>
> Consider the following ER diagram on the right. Assume that the ER diagram consists *only* the given entity set *(never really happen in practice but we have to start from a simple example)*.



Schema

```
CREATE TABLE Users (
  id       INT PRIMARY KEY,
  dob      DATE,
  passport VARCHAR(20),
  name     VARCHAR(200)
);
```

Is the following table allowed?

id	dob	passport	name
1	27-01-1990	P0	John
1	25-01-1990	P0	John

NO *(because `id` is the primary key and no duplicates allowed)*

Simple Translation

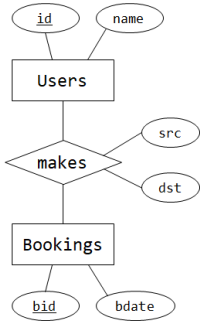
Entity Set
Relationship Set
- Correct
- Incorrect
Cardinality
One-to-One

Relationship Set

Correct Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Bookings (
  bid   INT PRIMARY KEY,
  bdate DATE
);
CREATE TABLE Makes (
  id    INT REFERENCES Users,
  bid   INT REFERENCES Bookings,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  PRIMARY KEY (id, bid)
);
```



Main Constraints

1. Uniquely Identifying
   - Can the key of Users uniquely identify the rest of attributes?
   - Can the key of Bookings uniquely identify the rest of attributes?
2. Lower Bound
   - Can Users not make Bookings?
   - Can Bookings not be made by any Users?
3. Upper Bound
   - Can Users *(with same `id`)* make *n* number of Bookings *(with different `bid`)*?
   - Can Bookings *(with same `bid`)* made by *n* Users *(with different `id`)*?

Simple Translation

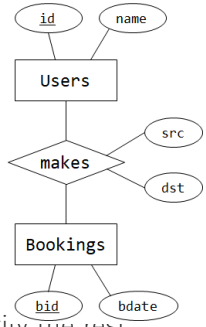
Entity Set
Relationship Set
- Correct
- Incorrect
Cardinality
One-to-One

Relationship Set

Correct Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Bookings (
  bid   INT PRIMARY KEY,
  bdate DATE
);
CREATE TABLE Makes (
  id    INT REFERENCES Users,
  bid   INT REFERENCES Bookings,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  PRIMARY KEY (id, bid)
);
```



1. Uniquely Identifying

- Can the key of Users uniquely identify the rest of attributes?
  - Based on previous discussions, `id` in Users uniquely identify `name`
- Can the key of Bookings uniquely identify the rest of attributes?
  - Based on previous discussions, `bid` in Bookings uniquely identify `bdate`
- Additionally, `bdate` need not be uniquely identified by `id` and `name` need not be uniquely identified by `bid`

Simple Translation

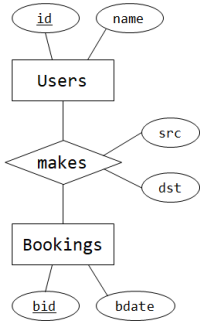
Entity Set
Relationship Set
- Correct
- Incorrect
Cardinality
One-to-One

Relationship Set

Correct Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Bookings (
  bid   INT PRIMARY KEY,
  bdate DATE
);
CREATE TABLE Makes (
  id    INT REFERENCES Users,
  bid   INT REFERENCES Bookings,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  PRIMARY KEY (id, bid)
);
```



2. Lower Bound

- Can Users not make Bookings?
  - **YES:** insert into Users but not insert into Makes

```
INSERT INTO Users ...;
~~INSERT INTO Makes ...;~~
```

- Can Bookings not be made by any Users?
  - **YES:** insert into Bookings but not insert into Makes

```
INSERT INTO Bookings ...;
~~INSERT INTO Makes ...;~~
```

Simple Translation

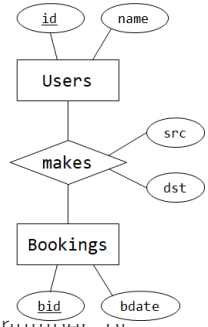
Entity Set
Relationship Set
- Correct
- Incorrect
Cardinality
One-to-One

Relationship Set

Correct Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Bookings (
  bid   INT PRIMARY KEY,
  bdate DATE
);
CREATE TABLE Makes (
  id    INT REFERENCES Users,
  bid   INT REFERENCES Bookings,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  PRIMARY KEY (id, bid)
);
```



3. Upper Bound

- Can Users *(with same **id**)* make *n* number of Bookings *(with different **bid**)*?
  - **YES:** in table Makes *(corresponding to relationship set "makes")*, the primary key is (**id**, **bid**) so the same **id** can be inserted with different **bid**
- Can Bookings *(with same **bid**)* made by *n* Users *(with different **id**)*?
  - **YES:** in table Makes *(corresponding to relationship set "makes")*, the primary key is (**id**, **bid**) so same **bid** can be inserted with the different **id**

Simple Translation

Entity Set
Relationship Set
- Correct
- Incorrect
Cardinality
One-to-One

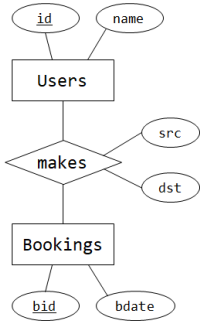
Relationship Set

Incorrect Translation

Schema

> Say we combine the three constructs into one schema. Why can't we do this?

```
CREATE TABLE Makes (
  id    INT -- no Users table,
  bid   INT -- no Bookings table,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  bdate DATE,
  name  VARCHAR(200)
  PRIMARY KEY (id, bid)
);
```



Main Constraints

1. Uniquely Identifying
   - Can the key of Users uniquely identify the rest of attributes?
   - Can the key of Bookings uniquely identify the rest of attributes?
2. Lower Bound
   - Can Users not make Bookings?
   - Can Bookings not be made by any Users?
3. Upper Bound
   - Can Users *(with same `id`)* make *n* number of Bookings *(with different `bid`)*?
   - Can Bookings *(with same `bid`)* made by *n* Users *(with different `id`)*?

Simple Translation

Entity Set
Relationship Set
- Correct
- Incorrect
Cardinality
One-to-One

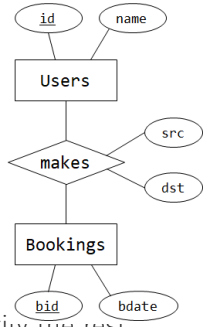
Relationship Set

Incorrect Translation

Schema

> Say we combine the three constructs into one schema. Why can't we do this?

```
CREATE TABLE Makes (
  id    INT -- no Users table,
  bid   INT -- no Bookings table,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  bdate DATE,
  name  VARCHAR(200)
  PRIMARY KEY (id, bid)
);
```



1. Uniquely Identifying

- Can the key of Users uniquely identify the rest of attributes?
  - **NO:** because the primary key is (`id`,`bid`) so we can have the following rows inserted

id	bid	name	...
1	10	John	...
1	12	Jane	...

  ⇒ So `id` does not uniquely identify `name`

- Can you work out the similar problem with Bookings?

Simple Translation

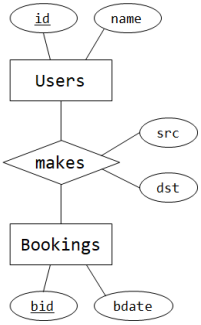
Relationship Set

Incorrect Translation

Schema

> Say we combine the three constructs into one schema. Why can't we do this?

```
CREATE TABLE Makes (
  id    INT -- no Users table,
  bid   INT -- no Bookings table,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  bdate DATE,
  name  VARCHAR(200)
  PRIMARY KEY (id, bid)
);
```



2. Lower Bound

- Can Users not make Bookings?
  - **NO:** because of the primary key, every `id` must have a `bid` *(i.e., `bid` cannot be `NULL`)*
- Can Bookings not be made by any Users?
  - **NO:** because of the primary key, every `bid` must have a `id` *(i.e., `id` cannot be `NULL`)*

Simple Translation

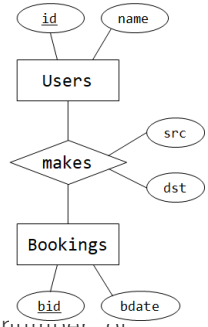
Relationship Set

Incorrect Translation

Schema

> Say we combine the three constructs into one schema. Why can't we do this?

```
CREATE TABLE Makes (
  id    INT -- no Users table,
  bid   INT -- no Bookings table,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  bdate DATE,
  name  VARCHAR(200)
  PRIMARY KEY (id, bid)
);
```



3. Upper Bound

- Can Users *(with same `id`)* make *n* number of Bookings *(with different `bid`)*?
  - **YES:** in table Makes *(corresponding to relationship set "makes")*, the primary key is (`id`, `bid`) so the same `id` can be inserted with different `bid`
- Can Bookings *(with same `bid`)* made by *n* Users *(with different `id`)*?
  - **YES:** in table Makes *(corresponding to relationship set "makes")*, the primary key is (`id`, `bid`) so same `bid` can be inserted with the different `id`

Simple Translation

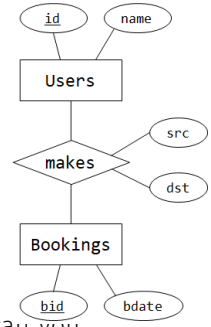
Relationship Set

Incorrect Translation

Schema

> Say we combine the three constructs into one schema. Why can't we do this?

```
CREATE TABLE Makes (
  id    INT -- no Users table,
  bid   INT -- no Bookings table,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  bdate DATE,
  name  VARCHAR(200)
  PRIMARY KEY (id)
);
```



Check

> Now that the primary key is `id`, can you check the following?
>
> 1. Uniquely Identifying
> 2. Lower Bound
> 3. Upper Bound

[-]Answer

Simple Translation

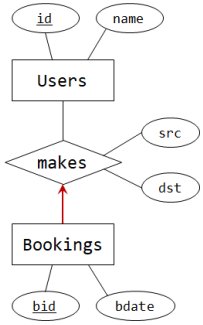
Entity Set
Relationship Set
Cardinality
- Correct
- Incorrect
One-to-One

Cardinality Constraints

Correct Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Makes (
  id    INT REFERENCES Users,
  bid   INT,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  bdate DATE,
  PRIMARY KEY (bid)
);
```



Main Constraints

1. Uniquely Identifying
   - Can the key of Users uniquely identify the rest of attributes?
   - Can the key of Bookings uniquely identify the rest of attributes?
2. Lower Bound
   - Can Users not make Bookings?
   - Can Bookings not be made by any Users?
3. Upper Bound
   - Can Users *(with same* ***id****)* make *n* number of Bookings *(with different* ***bid****)*?
   - Can Bookings *(with same* ***bid****)* made by *n* Users *(with different* ***id****)*?

Simple Translation

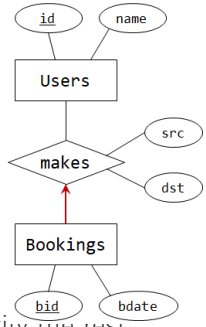
Entity Set
Relationship Set
Cardinality
- Correct
- Incorrect
One-to-One

Cardinality Constraints

Correct Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Makes (
  id    INT REFERENCES Users,
  bid   INT PRIMARY KEY,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  bdate DATE
);
```



1. Uniquely Identifying

- Can the key of Users uniquely identify the rest of attributes?
  - **YES:** `id` can uniquely identify `name` because `id` is the primary key of Users
- Can the key of Bookings uniquely identify the rest of attributes?
  - **YES:** `bid` can uniquely identify `bdate` because `bid` is the primary key of Makes and Bookings have been *merged* with Makes

Exercise

Can you generalize to multiple attributes and/or multiple key attributes.

Simple Translation

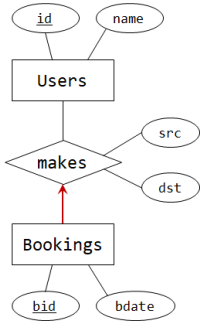
Entity Set
Relationship Set
Cardinality
- Correct
- Incorrect
One-to-One

Cardinality Constraints

Correct Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Makes (
  id    INT REFERENCES Users,
  bid   INT PRIMARY KEY,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  bdate DATE
);
```



2. Lower Bound

- Can Users not make Bookings?
  - **YES:** insert into Users but not insert into Makes

```
INSERT INTO Users ...;
INSERT INTO Makes ...;
```

- Can Bookings not be made by any Users?
  - **YES:** insert into Bookings but with `id` being `NULL` *(notice the lack of `NOT NULL` on `id` in Makes)*

```
INSERT INTO Bookings VALUES
  ( NULL , bid, ...);
```

Simple Translation

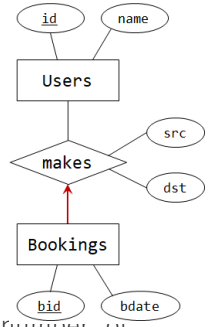
Entity Set
Relationship Set
Cardinality
- Correct
- Incorrect
One-to-One

Cardinality Constraints

Correct Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Makes (
  id    INT REFERENCES Users,
  bid   INT PRIMARY KEY,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  bdate DATE
);
```



3. Upper Bound

- Can Users *(with same **id**)* make *n* number of Bookings *(with different **bid**)*?
  - **YES:** in table Makes *(as we merged Bookings and "makes")*, the primary key is (**bid**) so the same **id** can be inserted with different **bid**
- Can Bookings *(with same **bid**)* made by *n* Users *(with different **id**)*?
  - **NO*:** in table Makes *(as we merged Bookings and "makes")*, the primary key is (**bid**) so same **bid** cannot be inserted with the different **id**

*This is **NO** in a good way because the ER diagram enforces such constraints so it is in green.

Simple Translation

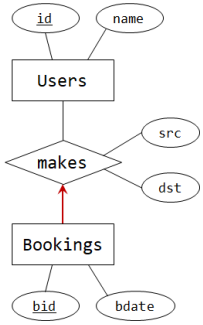
Entity Set
Relationship Set
Cardinality
- Correct
- Incorrect
One-to-One

Cardinality Constraints

Correct Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Bookings (
  bid   INT PRIMARY KEY,
  bdate DATE
);
CREATE TABLE Makes (
  id    INT NOT NULL REFERENCES Users,
  bid   INT PRIMARY KEY REFERENCES Bookings,
  src   VARCHAR(20),
  dst   VARCHAR(20)
);
```



Main Constraints

1. Uniquely Identifying
   - Can the key of Users uniquely identify the rest of attributes?
   - Can the key of Bookings uniquely identify the rest of attributes?
2. Lower Bound
   - Can Users not make Bookings?
   - Can Bookings not be made by any Users?
3. Upper Bound
   - Can Users *(with same `id`)* make *n* number of Bookings *(with different `bid`)*?
   - Can Bookings *(with same `bid`)* made by *n* Users *(with different `id`)*?

[#]Slightly different from lecture note by making `id` to be `NOT NULL`.

Simple Translation

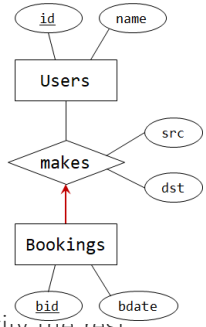
Entity Set
Relationship Set
Cardinality
- Correct
- Incorrect
One-to-One

Cardinality Constraints

Correct Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Bookings (
  bid   INT PRIMARY KEY,
  bdate DATE
);
CREATE TABLE Makes (
  id    INT NOT NULL REFERENCES Users,
  bid   INT PRIMARY KEY REFERENCES Bookings,
  src   VARCHAR(20),
  dst   VARCHAR(20)
);
```



1. Uniquely Identifying

- Can the key of Users uniquely identify the rest of attributes?
  - **YES:** `id` can uniquely identify `name` because `id` is the primary key of Users
- Can the key of Bookings uniquely identify the rest of attributes?
  - **YES:** `bid` can uniquely identify `bdate` because `bid` is the primary key of Bookings

Simple Translation

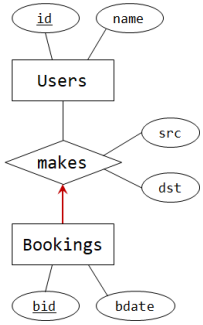
Entity Set
Relationship Set
Cardinality
- Correct
- Incorrect
One-to-One

Cardinality Constraints

Correct Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Bookings (
  bid   INT PRIMARY KEY,
  bdate DATE
);
CREATE TABLE Makes (
  id    INT NOT NULL REFERENCES Users,
  bid   INT PRIMARY KEY REFERENCES Bookings,
  src   VARCHAR(20),
  dst   VARCHAR(20)
);
```



2. Lower Bound

- Can Users not make Bookings?
  - **YES:** insert into Users but not insert into Makes

```
INSERT INTO Users ...;
~~INSERT INTO Makes ...;~~
```

- Can Bookings not be made by any Users?
  - **YES:** insert into Bookings but not insert into Makes

```
INSERT INTO Bookings ...;
~~INSERT INTO Makes ...;~~
```

Simple Translation

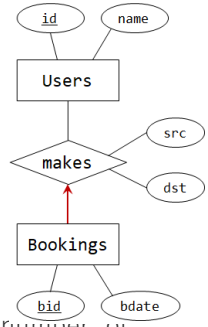
Entity Set
Relationship Set
Cardinality
- Correct
- Incorrect
One-to-One

Cardinality Constraints

Correct Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Bookings (
  bid   INT PRIMARY KEY,
  bdate DATE
);
CREATE TABLE Makes (
  id    INT NOT NULL REFERENCES Users,
  bid   INT PRIMARY KEY REFERENCES Bookings,
  src   VARCHAR(20),
  dst   VARCHAR(20)
);
```



3. Upper Bound

- Can Users *(with same `id`)* make *n* number of Bookings *(with different `bid`)*?
  - **YES:** in table Makes *(corresponding to relationship set "makes")*, the primary key is (`bid`) so the same `id` can be inserted with different `bid`
- Can Bookings *(with same `bid`)* made by *n* Users *(with different `id`)*?
  - **NO:** in table Makes *(corresponding to relationship set "makes")*, the primary key is (`bid`) so same `bid` cannot be inserted with the different `id`

Simple Translation

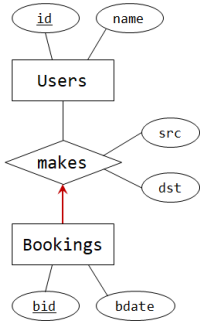
Entity Set
Relationship Set
Cardinality
- Correct
- Incorrect
One-to-One

Cardinality Constraints

Incorrect Translation

Schema

```
CREATE TABLE Makes (
  id    INT -- no Users table,
  bid   INT -- no Bookings table,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  name  VARCHAR(200),
  bdate DATE,
  name  VARCHAR(200),
  PRIMARY KEY (bid)
);
```



Main Constraints

1. Uniquely Identifying
   - Can the key of Users uniquely identify the rest of attributes?
   - Can the key of Bookings uniquely identify the rest of attributes?
2. Lower Bound
   - Can Users not make Bookings?
   - Can Bookings not be made by any Users?
3. Upper Bound
   - Can Users *(with same **id**)* make *n* number of Bookings *(with different **bid**)*?
   - Can Bookings *(with same **bid**)* made by *n* Users *(with different **id**)*?

Simple Translation

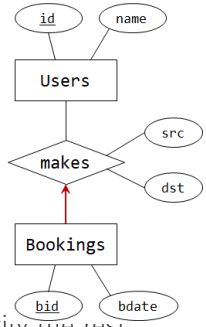
Entity Set
Relationship Set
Cardinality
- Correct
- Incorrect
One-to-One

Cardinality Constraints

Incorrect Translation

Schema

```
CREATE TABLE Makes (
  id    INT -- no Users table,
  bid   INT -- no Bookings table,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  name  VARCHAR(200),
  bdate DATE,
  name  VARCHAR(200),
  PRIMARY KEY (bid)
);
```



1. Uniquely Identifying

- Can the key of Users uniquely identify the rest of attributes?
  - **NO:** because the primary key is `id`
  - The following can be inserted
    - (`id1`, ..., `name1`, ...)
    - (`id1`, ..., `name2`, ...)
- Can the key of Bookings uniquely identify the rest of attributes?
  - **YES:** because the primary key is `id`
  - The following cannot be inserted
    - (..., `bid1`, ..., `bdate1`, ...)
    - (..., `bid2`, ..., `bdate2`, ...)

Simple Translation

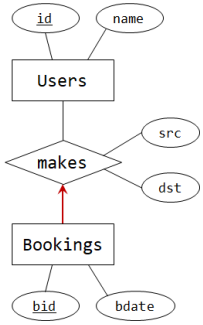
Entity Set
Relationship Set
Cardinality
- Correct
- Incorrect
One-to-One

Cardinality Constraints

Incorrect Translation

Schema

```
CREATE TABLE Makes (
  id    INT -- no Users table,
  bid   INT -- no Bookings table,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  name  VARCHAR(200),
  bdate DATE,
  name  VARCHAR(200),
  PRIMARY KEY (bid)
);
```



2. Lower Bound

- Can Users not make Bookings?
  - **NO:** `bid` cannot be `NULL`
- Can Bookings not be made by any Users?
  - **YES:** `id` can be `NULL`

Simple Translation

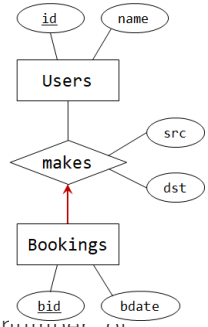
Entity Set
Relationship Set
Cardinality
- Correct
- Incorrect
One-to-One

Cardinality Constraints

Incorrect Translation

Schema

```
CREATE TABLE Makes (
  id    INT -- no Users table,
  bid   INT -- no Bookings table,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  name  VARCHAR(200),
  bdate DATE,
  name  VARCHAR(200),
  PRIMARY KEY (bid)
);
```



3. Upper Bound

- Can Users *(with same `id`)* make *n* number of Bookings *(with different `bid`)*?
  - **YES:** `bid` is the primary key
  - The following can be inserted
    - (`id1`, `bid1`, ...)
    - (`id1`, `bid2`, ...)
- Can Bookings *(with same `bid`)* made by *n* Users *(with different `id`)*?
  - **NO:** `bid` is the primary key
  - The following can be inserted
    - (`id1`, `bid1`, ...)
    - (`id2`, `bid1`, ...)

Simple Translation

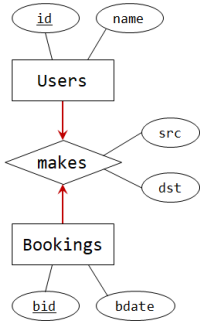
Entity Set
Relationship Set
Cardinality
One-to-One
- Correct
- Incorrect
- Reverse

One-to-One Relationship

Correct Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Bookings (
  bid   INT PRIMARY KEY,
  bdate DATE
);
CREATE TABLE Makes (
  id    INT UNIQUE NOT NULL REFERENCES Users,
  bid   INT PRIMARY KEY REFERENCES Bookings,
  src   VARCHAR(20),
  dst   VARCHAR(20)
);
```



Main Constraints

1. Uniquely Identifying
   - Can the key of Users uniquely identify the rest of attributes?
   - Can the key of Bookings uniquely identify the rest of attributes?
2. Lower Bound
   - Can Users not make Bookings?
   - Can Bookings not be made by any Users?
3. Upper Bound
   - Can Users *(with same **id**)* make *n* number of Bookings *(with different **bid**)*?
   - Can Bookings *(with same **bid**)* made by *n* Users *(with different **id**)*?

[#]Yet another alternative from lecture notes! There can be many other different mappings!

Simple Translation

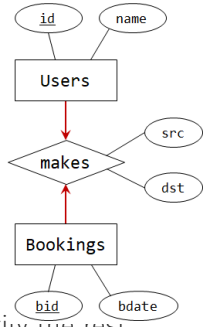
Entity Set
Relationship Set
Cardinality
One-to-One
- Correct
- Incorrect
- Reverse

One-to-One Relationship

Correct Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Bookings (
  bid   INT PRIMARY KEY,
  bdate DATE
);
CREATE TABLE Makes (
  id    INT UNIQUE NOT NULL REFERENCES Users,
  bid   INT PRIMARY KEY REFERENCES Bookings,
  src   VARCHAR(20),
  dst   VARCHAR(20)
);
```



1. Uniquely Identifying

- Can the key of Users uniquely identify the rest of attributes?
  - **YES:** `id` can uniquely identify `name` because `id` is the primary key of Users
- Can the key of Bookings uniquely identify the rest of attributes?
  - **YES:** `bid` can uniquely identify `bdate` because `bid` is the primary key of Bookings

Simple Translation

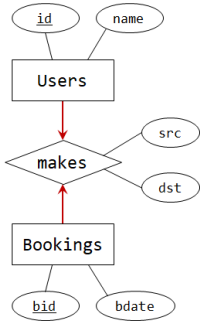
Entity Set
Relationship Set
Cardinality
One-to-One
- Correct
- Incorrect
- Reverse

One-to-One Relationship

Correct Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Bookings (
  bid   INT PRIMARY KEY,
  bdate DATE
);
CREATE TABLE Makes (
  id    INT UNIQUE NOT NULL REFERENCES Users,
  bid   INT PRIMARY KEY REFERENCES Bookings,
  src   VARCHAR(20),
  dst   VARCHAR(20)
);
```



2. Lower Bound

- Can Users not make Bookings?
  - **YES:** insert into Users but not insert into Makes

```
INSERT INTO Users ...;
~~INSERT INTO Makes ...;~~
```

- Can Bookings not be made by any Users?
  - **YES:** insert into Bookings but not insert into Makes

```
INSERT INTO Bookings ...;
~~INSERT INTO Makes ...;~~
```

Simple Translation

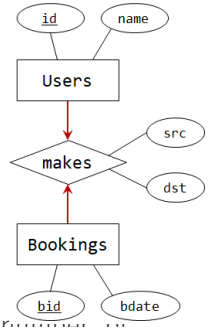
Entity Set
Relationship Set
Cardinality
One-to-One
- Correct
- Incorrect
- Reverse

One-to-One Relationship

Correct Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Bookings (
  bid   INT PRIMARY KEY,
  bdate DATE
);
CREATE TABLE Makes (
  id    INT UNIQUE NOT NULL REFERENCES Users,
  bid   INT PRIMARY KEY REFERENCES Bookings,
  src   VARCHAR(20),
  dst   VARCHAR(20)
);
```



3. Upper Bound

- Can Users *(with same `id`)* make *n* number of Bookings *(with different `bid`)*?
  - **NO:** in table Makes *(corresponding to relationship set "makes")*, (`id`) is unique so the same `id` cannot be inserted with different `bid`
- Can Bookings *(with same `bid`)* made by *n* Users *(with different `id`)*?
  - **NO:** in table Makes *(corresponding to relationship set "makes")*, the primary key is (`bid`) so same `bid` cannot be inserted with the different `id`

Can we make `id` to be the primary key and `bid` the candidate key?

Simple Translation

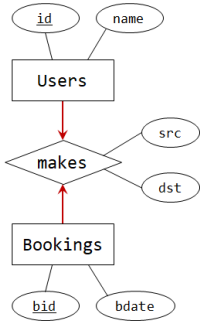
Entity Set
Relationship Set
Cardinality
One-to-One
- Correct
- Incorrect
- Reverse

One-to-One Relationship

(Slightly) Incorrect Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Makes (
  id    INT NOT NULL REFERENCES Users,
  bid   INT PRIMARY KEY,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  bdate DATE
);
```



Main Constraints

1. Uniquely Identifying
   - Can the key of Users uniquely identify the rest of attributes?
   - Can the key of Bookings uniquely identify the rest of attributes?
2. Lower Bound
   - Can Users not make Bookings?
   - Can Bookings not be made by any Users?
3. Upper Bound
   - Can Users *(with same `id`)* make *n* number of Bookings *(with different `bid`)*?
   - Can Bookings *(with same `bid`)* made by *n* Users *(with different `id`)*?

[#]One of the alternative in lecture notes. Please attempt the second alternative.

Simple Translation

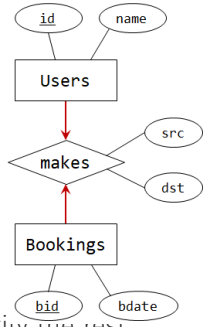
Entity Set
Relationship Set
Cardinality
One-to-One
- Correct
- Incorrect
- Reverse

One-to-One Relationship

(Slightly) Incorrect Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Makes (
  id    INT NOT NULL REFERENCES Users,
  bid   INT PRIMARY KEY,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  bdate DATE
);
```



1. Uniquely Identifying

- Can the key of Users uniquely identify the rest of attributes?
  - **YES:** `id` can uniquely identify `name` because `id` is the primary key of Users
- Can the key of Bookings uniquely identify the rest of attributes?
  - **YES:** `bid` can uniquely identify `bdate` because `bid` is the primary key of Makes *(after merging)*

Simple Translation

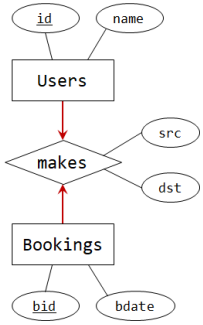
Entity Set
Relationship Set
Cardinality
One-to-One
- Correct
- Incorrect
- Reverse

One-to-One Relationship

(Slightly) Incorrect Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Makes (
  id    INT NOT NULL REFERENCES Users,
  bid   INT PRIMARY KEY,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  bdate DATE
);
```



2. Lower Bound

- Can Users not make Bookings?
  - **YES:** insert into Users but not insert into Makes

    ```
    INSERT INTO Users ...;
    INSERT INTO Makes ...;
    ```

- Can Bookings not be made by any Users?
  - **NO:** the following cannot be inserted
    - (`NULL`, `bid`, ...)

Simple Translation

Entity Set
Relationship Set
Cardinality
One-to-One
- Correct
- Incorrect
- Reverse

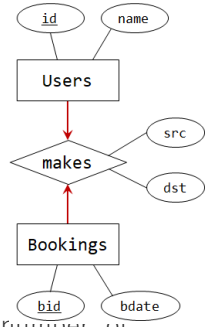
One-to-One Relationship

(Slightly) Incorrect Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Makes (
  id    INT NOT NULL REFERENCES Users,
  bid   INT PRIMARY KEY,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  bdate DATE
);
```

id bid
(1, 1)
(1, 2)
(1, 3).



3. Upper Bound

- Can Users *(with same **id**)* make *n* number of Bookings *(with different **bid**)*?
  - **NO:** in table Makes *(since we merge)*, (**id**) is unique so the same **id** cannot be inserted with different **bid**
- Can Bookings *(with same **bid**)* made by *n* Users *(with different **id**)*?
  - **NO:** in table Makes *(since we merge)*, the primary key is (**bid**) so same **bid** cannot be inserted with the different **id**

Simple Translation

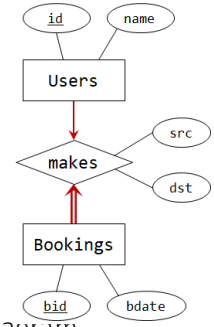
Entity Set
Relationship Set
Cardinality
One-to-One
- Correct
- Incorrect
- Reverse

One-to-One Relationship

Reverse Translation

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Makes (
  id    INT NOT NULL REFERENCES Users,
  bid   INT PRIMARY KEY,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  bdate DATE
);
```



Correct ER Diagram

> Can you check that the ER diagram above is the *more correct* ER diagram that produces this translation?
>
> **Hint:** all Bookings must be made by a Users, and just nice that `id` is `NOT NULL`.

More Translation

More Translation

Preliminary

> Certain cases where we do not have a general translation technique can actually be translated under special case. We will discuss one case below.

Total Participation without Cardinality Constraint

> Using the current translation technique, we cannot enforce total participation constraints *without* cardinality constraint. However, on certain cases, this can be done. We will discuss first why the current translation is inadequate in general then look at some specific case where it can be done.

Task

- You should try thinking about this when translating some ISA hierarchy.

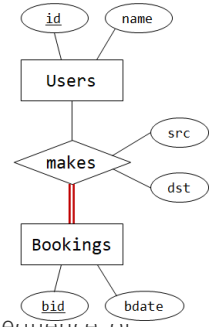
More Translation

Total Participation

Incorrect Translation #1

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Bookings (
  bid   INT PRIMARY KEY,
  bdate DATE
);
CREATE TABLE Makes (
  id    INT REFERENCES Users,
  bid   INT REFERENCES Bookings,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  PRIMARY KEY (id, bid)
);
```



A Potential Problem

- We cannot prevent the following sequence of operations:

  ```
  INSERT INTO Bookings ...
  INSERT INTO Makes ...
  ```

  - In other words, we have a Bookings that is not made by any Users
  - Because we do not have to insert into Makes!

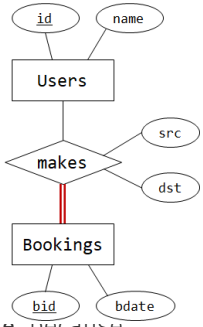
More Translation

Total Participation

Incorrect Translation #2

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Makes (
  id    INT REFERENCES Users,
  bid   INT,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  bdate DATE,
  PRIMARY KEY (id, bid)
);
```



A Potential Problem

- `bid` cannot uniquely identify `bdate` because the primary key is (`id`, `bid`)
  - We can insert the following
    - (..., `bid1`, ..., `bdate1`)
    - (..., `bid1`, ..., `bdate2`)

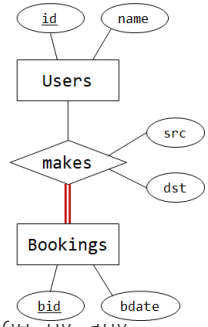
More Translation

Total Participation

Incorrect Translation #3

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Makes (
  id    INT REFERENCES Users,
  bid   INT,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  bdate DATE,
  PRIMARY KEY (bid)
);
```



A Potential Problem

- We can have a Bookings not made by any Users
  - Because `id` can be `NULL`
  - We can insert the following:
    - (`NULL`, `bid1`, ...)
- We cannot have the same `bid` booked by different `id`
  - Because `bid` is the primary key

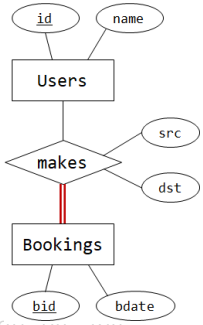
More Translation

Total Participation

Incorrect Translation #4

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Makes (
  id    INT NOT NULL REFERENCES Users,
  bid   INT,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  bdate DATE,
  PRIMARY KEY (bid)
);
```



A Potential Problem

- We can have a Bookings not made by any Users
  - Because `id` can be `NULL`
  - We can insert the following:
    - (`NULL`, `bid1`, ...)
- We cannot have the same `bid` booked by different `id`
  - Because `bid` is the primary key

Notes

So the second problem still persists even when we solve the first problem. If nothing can be done, choose the translation with most requirements enforced.

More Translation

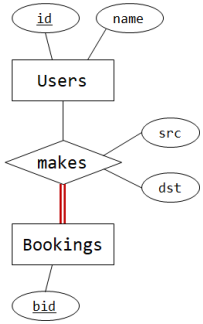
Total Participation

Correct Translation #1

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Makes (
  id    INT REFERENCES Users,
  bid   INT,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  -- no more bdate
  PRIMARY KEY (id, bid)
);
```

> Notice the lack of other attributes in Bookings



Main Constraints

1. Uniquely Identifying
   - Can the key of Users uniquely identify the rest of attributes?
   - Can the key of Bookings uniquely identify the rest of attributes?
2. Lower Bound
   - Can Users not make Bookings?
   - Can Bookings not be made by any Users?
3. Upper Bound
   - Can Users *(with same `id`)* make *n* number of Bookings *(with different `bid`)*?
   - Can Bookings *(with same `bid`)* made by *n* Users *(with different `id`)*?

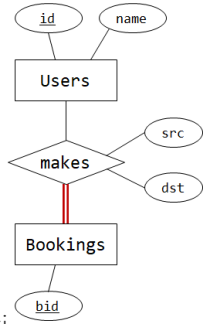
More Translation

Total Participation

Correct Translation #1

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Makes (
  id    INT REFERENCES Users,
  bid   INT,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  -- no more bdate
  PRIMARY KEY (id, bid)
);
```



1. Uniquely Identifying

- Can the key of Users uniquely identify the rest of attributes?
  - **YES:** `id` can uniquely identify `name` because `id` is the primary key of Users
- Can the key of Bookings uniquely identify the rest of attributes?
  - **YES:** there is ***NO*** other attributes to identify!

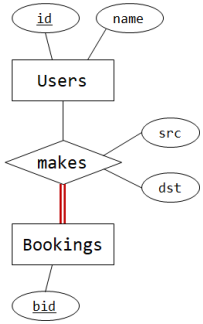
More Translation

Total Participation

Correct Translation #1

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Makes (
  id    INT REFERENCES Users,
  bid   INT,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  -- no more bdate
  PRIMARY KEY (id, bid)
);
```



2. Lower Bound

- Can Users not make Bookings?
  - **YES:** insert into Users but not insert into Makes

    ```
    INSERT INTO Users ...;
    INSERT INTO Makes ...;
    ```

- Can Bookings not be made by any Users?
  - **NO:** `id` in Makes cannot be `NULL` because `id` is part of the primary key

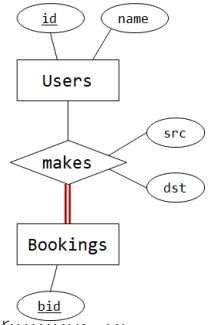
More Translation

Total Participation

Correct Translation #1

Schema

```
CREATE TABLE Users (
  id    INT PRIMARY KEY,
  name  VARCHAR(200)
);
CREATE TABLE Makes (
  id    INT REFERENCES Users,
  bid   INT,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  -- no more bdate
  PRIMARY KEY (id, bid)
);
```



3. Upper Bound

- Can Users *(with same `id`)* make *n* number of Bookings *(with different `bid`)*?
  - **YES:** in table Makes *(as we merge)*, the primary key is (`id`, `bid`) so the same `id` can be inserted with the different `bid`
- Can Bookings *(with same `bid`)* made by *n* Users *(with different `id`)*?
  - **YES:** in table Makes *(as we merge)*, the primary key is (`id`, `bid`) so the same `bid` can be inserted with the different `id`

More Translation

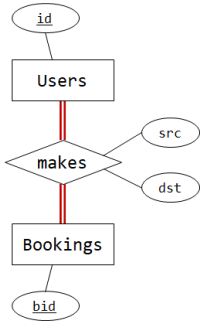
Total Participation

Correct Translation #2

Schema

```
CREATE TABLE Makes (
  id    INT -- no Users table,
  bid   INT -- no Bookings table,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  -- no more name
  -- no more bdate
  PRIMARY KEY (id, bid)
);
```

> Notice the lack of other attributes in Bookings and Users



Main Constraints

1. Uniquely Identifying
   - Can the key of Users uniquely identify the rest of attributes?
   - Can the key of Bookings uniquely identify the rest of attributes?
2. Lower Bound
   - Can Users not make Bookings?
   - Can Bookings not be made by any Users?
3. Upper Bound
   - Can Users *(with same `id`)* make *n* number of Bookings *(with different `bid`)*?
   - Can Bookings *(with same `bid`)* made by *n* Users *(with different `id`)*?

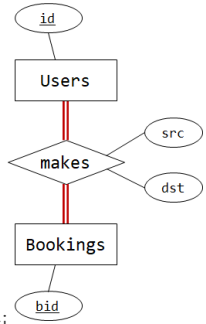
More Translation

Total Participation

Correct Translation #2

Schema

```
CREATE TABLE Makes (
  id    INT -- no Users table,
  bid   INT -- no Bookings table,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  -- no more name
  -- no more bdate
  PRIMARY KEY (id, bid)
);
```



1. Uniquely Identifying

- Can the key of Users uniquely identify the rest of attributes?
  - **YES:** there is ***NO*** other attributes to identify!
- Can the key of Bookings uniquely identify the rest of attributes?
  - **YES:** there is ***NO*** other attributes to identify!

Question

What about `src` and `dst`?

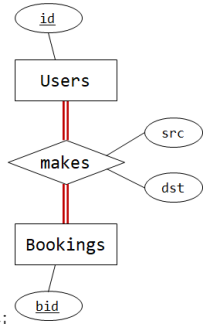
More Translation

Total Participation

Correct Translation #2

Schema

```
CREATE TABLE Makes (
  id    INT -- no Users table,
  bid   INT -- no Bookings table,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  -- no more name
  -- no more bdate
  PRIMARY KEY (id, bid)
);
```



1. Uniquely Identifying

- Can the key of Users uniquely identify the rest of attributes?
  - **YES:** there is ***NO*** other attributes to identify!
- Can the key of Bookings uniquely identify the rest of attributes?
  - **YES:** there is ***NO*** other attributes to identify!

[-]Answer

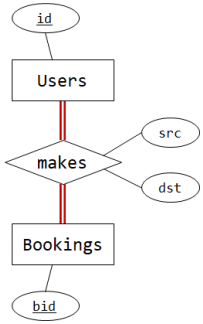
More Translation

Total Participation

Correct Translation #2

Schema

```
CREATE TABLE Makes (
  id    INT -- no Users table,
  bid   INT -- no Bookings table,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  -- no more name
  -- no more bdate
  PRIMARY KEY (id, bid)
);
```



2. Lower Bound

- Can Users not make Bookings?
  - **NO:** `bid` in Makes cannot be `NULL` because `bid` is part of the primary key
- Can Bookings not be made by any Users?
  - **NO:** `id` in Makes cannot be `NULL` because `id` is part of the primary key

Notes

We require that both cases cannot be made. So we are really enforcing what the ER diagram says.

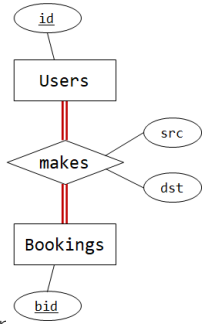
More Translation

Total Participation

Correct Translation #2

Schema

```
CREATE TABLE Makes (
  id    INT -- no Users table,
  bid   INT -- no Bookings table,
  src   VARCHAR(20),
  dst   VARCHAR(20),
  -- no more name
  -- no more bdate
  PRIMARY KEY (id, bid)
);
```



3. Upper Bound

- Can Users *(with same `id`)* make *n* number of Bookings *(with different `bid`)*?
  - **YES:** in table Makes *(as we merge)*, the primary key is (`id`, `bid`) so the same `id` can be inserted with different `bid`
- Can Bookings *(with same `bid`)* made by *n* Users *(with different `id`)*?
  - **YES:** in table Makes *(as we merge)*, the primary key is (`id`, `bid`) so the same `bid` can be inserted with different `id`

```
postgres=# exit
Press any key to continue . . .
```