INFO 210: Database Management Systems

Homework 3 solutions

- This assignment covers
  - The entity-relationship (ER) model
  - Translating ER models to relational schemas

Review

Recall and Re-Summarization

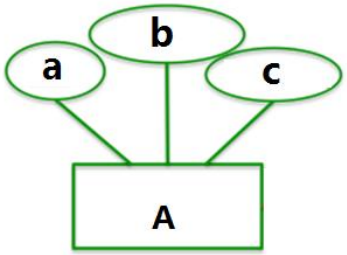
some common descriptions (included below but not limited to below) for some situations referring to topic4 & topic5

- Assume an entity set A with attributes **a**, **b**, **c** etc.
  Assume there exists a relationship ***X*** between A and another entity set B
  - Candidate/Primary key:
    - **No two** entities in A have **same** a
    - **No two** entities in A have (**both**) **same** a **and same** b
  - Relationship:
    - A …***X***… share ***one/same*** B
    - A …***X***… contain ***different/many/several*** B
  - Relationship with constraint:
    - A …***X***… ***at most one*** B
    - A …***X***… ***at least one*** B
    - A …***X***… ***exactly one*** B
  - Translating from ER to relational:
    - Without constraint
    - With only key constraint(s)
    - With participation constraint
- Refer to more example questions with solutions in topic4 and topic5 lecture notes

Recall and Re-Summarization

some common descriptions (included below but not limited to below) for some situations referring to topic4 & topic5

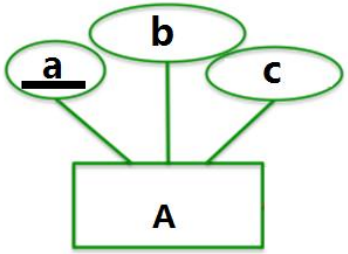
- Assume an entity set A with attributes **a**, **b**, **c** etc. (In the "Chen Notation" for ER diagram) :



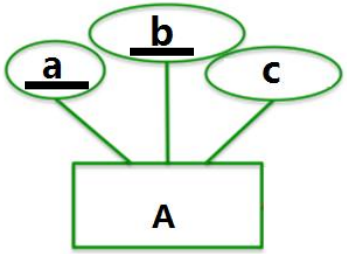
Candidate/Primary key:

No two entities in A have **same** a

No two entities in A have (**both**) **same** a **and same** b



Single (a) as the primary key in A

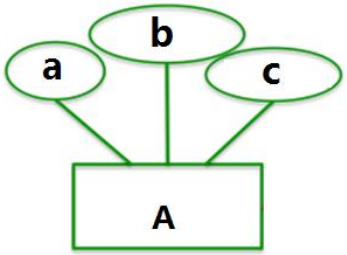


Combination of (a,b) as the primary key in A

Recall and Re-Summarization

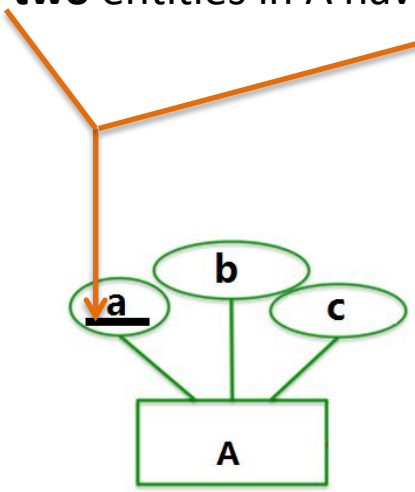
some common descriptions (included below but not limited to below) for some situations referring to topic4 & topic5

- Assume an entity set A with attributes **a**, **b**, **c** etc. (In the "Chen Notation" for ER diagram) :



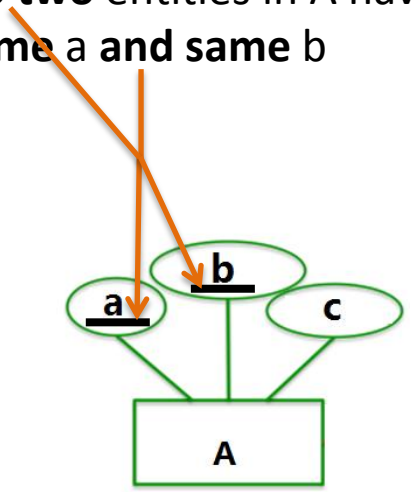
Candidate/Primary key:

No two entities in A have **same** a



Single (a) as the primary key in A

No two entities in A have (**both**) **same** a **and same** b



Combination of (a,b) as the primary key in A

Recall and Re-Summarization

some common descriptions (included below but not limited to below) for some situations referring to topic4 & topic5

- Assume there exists a relationship ***X*** between A and another entity set B. (In the "Chen Notation" for ER diagram) :

Relationship:

A …***X***… share ***one/same*** B

xx-to-one

A …***X***… contain ***different/many/several*** B

xx-to-many

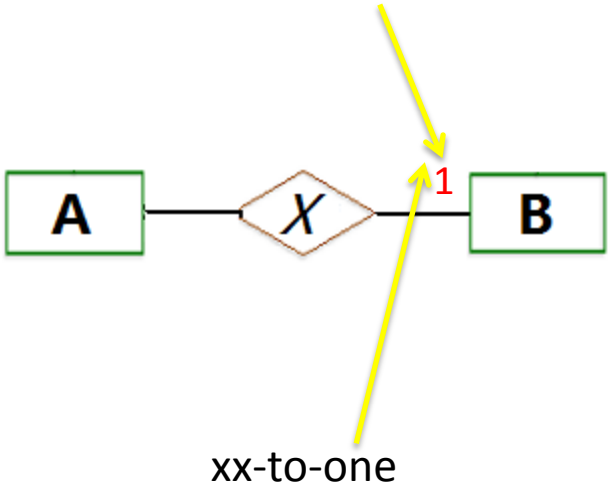
Recall and Re-Summarization

some common descriptions (included below but not limited to below) for some situations referring to topic4 & topic5

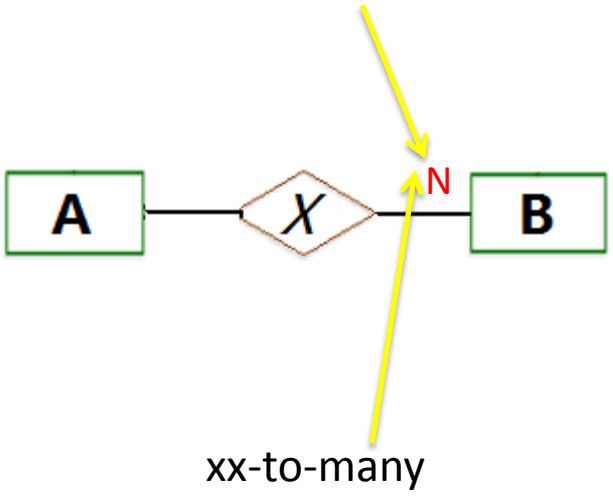
- Assume there exists a relationship ***X*** between A and another entity set B. (In the "Chen Notation" for ER diagram) :

Relationship:

A …***X***… share ***one/same*** B



A …***X***… contain ***different/many/several*** B



Recall and Re-Summarization

some common descriptions (included below but not limited to below) for some situations referring to topic4 & topic5

- Assume there exists a relationship ***X*** between A and another entity set B. (In the "Chen Notation" for ER diagram) :

Relationship with constraint:

A ...***X***... ***at most one*** B

xx-to-one

key constraint **on** A

constraint "at most one" means maybe 0 maybe 1 but definitely not more than 1

A ...***X***... ***at least one*** B

xx-to-many

participation constraint **on** A

constraint "at least one" means maybe 1 maybe more than 1 but definitely not 0 (every one has to participate).

A ...***X***... ***exactly one*** B

xx-to-one

key and **participation** constraint **on** A

constraint "exactly one" means 1, not any other (not more than 1 **and also** every one has to participate).

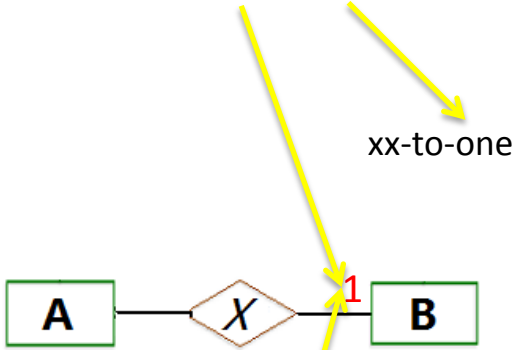
Recall and Re-Summarization

some common descriptions (included below but not limited to below) for some situations referring to topic4 & topic5

- Assume there exists a relationship ***X*** between A and another entity set B. (In the "Chen Notation" for ER diagram) :

Relationship with constraint:

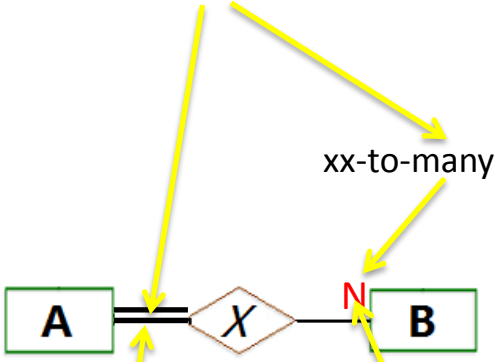
A ...***X***... ***at most one*** B



key constraint **on** A

constraint "at most one" means maybe 0 maybe 1 but definitely not more than 1

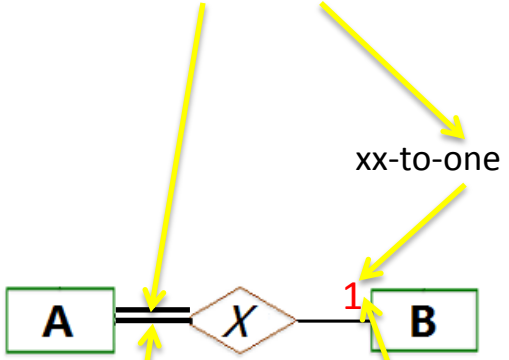
A ...***X***... ***at least one*** B



participation constraint **on** A

constraint "at least one" means maybe 1 maybe more than 1 but definitely not 0 (every one has to participate).

A ...***X***... ***exactly one*** B



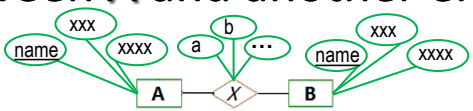
key and **participation** constraint **on** A

constraint "exactly one" means 1, not any other (not more than 1 **and also** every one has to participate).

Recall and Re-Summarization

some common descriptions (included below but not limited to below) for some situations referring to topic4 & topic5

- Assume there exists a relationship ***X*** between **A** and another entity set **B**. (In the "Chen Notation" for ER diagram) :



Translating from ER to relational:

Without constraint


```
create table A(
  name  varchar(128)  primary key,
  …   …(…)
);

create table B(
  name  varchar(128) primary key ,
  …   …(…)
);

create table X (
  a   char(12),
  …   …(…),
  A_name varchar(128),
  B_name varchar(128),
  primary key (A_name, B_name),
  foreign key (A_name ) references A(name),
  foreign key (B_name ) references B(name)
);
```

Include: 1. descriptive attributes **a**,… of relationship *X*

2. primary key attributes of each entity set A,B that involved in the relationship *X*

Make: **Combination** of the key attributes **from both A and B** as the primary key (here is the (A_name, B_name)) in table *X*, which can represent **Many-to-Many** relationship

With only key constraint(s)


```
create table B(
  name  varchar(128)  primary key,
  …   …(…)
);
create table A_X (
  a   char(12),
  name varchar(128) primary key,
  B_name varchar(128) ,
  foreign key (B_name ) references B(name),
);
```

One key constraint **on A** (xx-to-one)

Option 1:

Use: **three relations/tables** similar to the left

Make: **only make the key from A** (not combination) as the **primary key** in the *A_X*.

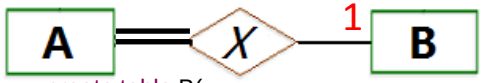
Option 2:

Use: **one relation/table** (here is the *A_X*), to represent **both the *X* and the entity set that has the key constraint on** (here is the A), and use **the other table for B**.

Make: **only the key attributes of A** as **primary key** in *A_X*, which can represent the key constraint **on A**, and xx-to-one relationship.

For **several key constraints** in *X*, choose one to treat as the entity set A here, and then make keys from other entity sets as **unique** in *A_X*.

With participation constraint




```
create table B(
  name  varchar(128)  primary key,
  …   …(…)
);
create table A_X (
  a   char(12),
  name varchar(128) primary key,
  B_name varchar(128) not null,
  foreign key (B_name ) references B(name),
);
```

One participation constraint **on A**

Must use only one relation/table (here is the *A_X*) to represent **both the *X* and the entity set that has the participation constraint on** (here is the A), use the other table for B. Make the **key from B with "not null" constraint** to guarantee **participation** constraint **on A**.

If there is **no key constraint** on the same entity which has the participation constraint (here is the A), then **make the combination of the key attributes from both A and B** as the **primary** key in the *A_X*.

If there is a **key constraint on the same entity** which also has the participation constraint (here is the A), then **only make the key from A** (not combination) as the **primary** key in the *A_X*.

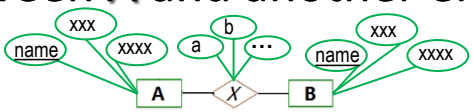
If key and participation constraints are not on same one entity set, then we are not able to exactly translate.

One special case: **BOTH** entity sets have **Both** key and participation constraint on, then **map everything into a single table** *A_X_B*, make key from an entity, e.g. A, the **primary** key, and make the key from B as "**unique not null**" in *A_X_B*.

Recall and Re-Summarization

some common descriptions (included below but not limited to below) for some situations referring to topic4 & topic5

- Assume there exists a relationship ***X*** between **A** and another entity set **B**. (In the "Chen Notation" for ER diagram) :



Translating from ER to relational:

Without constraint


```
create table A(
  name  varchar(128)  primary key,
  …   …(…)
);

create table B(
  name  varchar(128) primary key ,
  …   …(…)
);

create table X (
  a   char(12),
  …   …(…),
  A_name varchar(128),
  B_name varchar(128),
  primary key (A_name, B_name),
  foreign key (A_name ) references A(name),
  foreign key (B_name ) references B(name)
);
```

Include: 1. descriptive attributes **a**,… of relationship *X*

2. primary key attributes of each entity set A,B that involved in the relationship *X*

Make: **Combination** of the key attributes **from both A and B** as the primary key (here is the (A_name, B_name)) in table *X*, which can represent **Many-to-Many** relationship

With only key constraint(s)


```
create table B(
  name  varchar(128)  primary key,
  …   …(…)
);
create table A_X (
  a   char(12),
  name varchar(128) primary key,
  B_name varchar(128) ,
  foreign key (B_name ) references B(name),
);
```

One key constraint **on A** (xx-to-one)

Option 1:

Use: **three relations/tables** similar to the left

Make: **only make the key from A** (not combination) as the **primary key** in the *A_X*.

Option 2:

Use: **one relation/table** (here is the *A_X*), to represent **both the *X* and the entity set that has the key constraint on** (here is the A), and use **the other table for B**.

Make: **only the key attributes of A** as **primary key** in *A_X*, which can represent the key constraint **on A**, and xx-to-one relationship.

For **several key constraints** in *X*, choose one to treat as the entity set A here, and then make keys from other entity sets as **unique** in *A_X*.

With participation constraint


```
create table B(
  name  varchar(128)  primary key,
  …   …(…)
);
create table A_X (
  a   char(12),
  name varchar(128) primary key,
  B_name varchar(128) not null,
  foreign key (B_name ) references B(name),
);
```

One participation constraint **on A**

Must use only one relation/table (here is the *A_X*) to represent **both the *X* and the entity set that has the participation constraint on** (here is the A), use the other table for B. Make the **key from B with "not null" constraint** to guarantee **participation** constraint **on A**.

If there is **no key constraint** on the same entity which has the participation constraint (here is the A), then **make the combination of the key attributes from both A and B** as the **primary** key in the *A_X*.

If there is a **key constraint on the same entity** which also has the participation constraint (here is the A), then **only make the key from A** (not combination) as the **primary** key in the *A_X*.

If key and participation constraints are not on same one entity set, then we are not able to exactly translate.

One special case: **BOTH** entity sets have **Both** key and participation constraint on, then **map everything into a single table** *A_X_B*, make key from an entity, e.g. A, the **primary** key, and make the key from B as "**unique not null**" in *A_X_B*.

Part 1 (a)

Part 1 (40pts):
Entity-Relationship modeling

- A recording studio needs help designing its database. The studio stores information about musicians and albums**. Draw an ER diagram** describing the studio's database for each of the two scenarios described below. Assume that the only business rules that hold are those stated below, and that no additional business rules hold. **Clearly mark all key and participation constraints.**

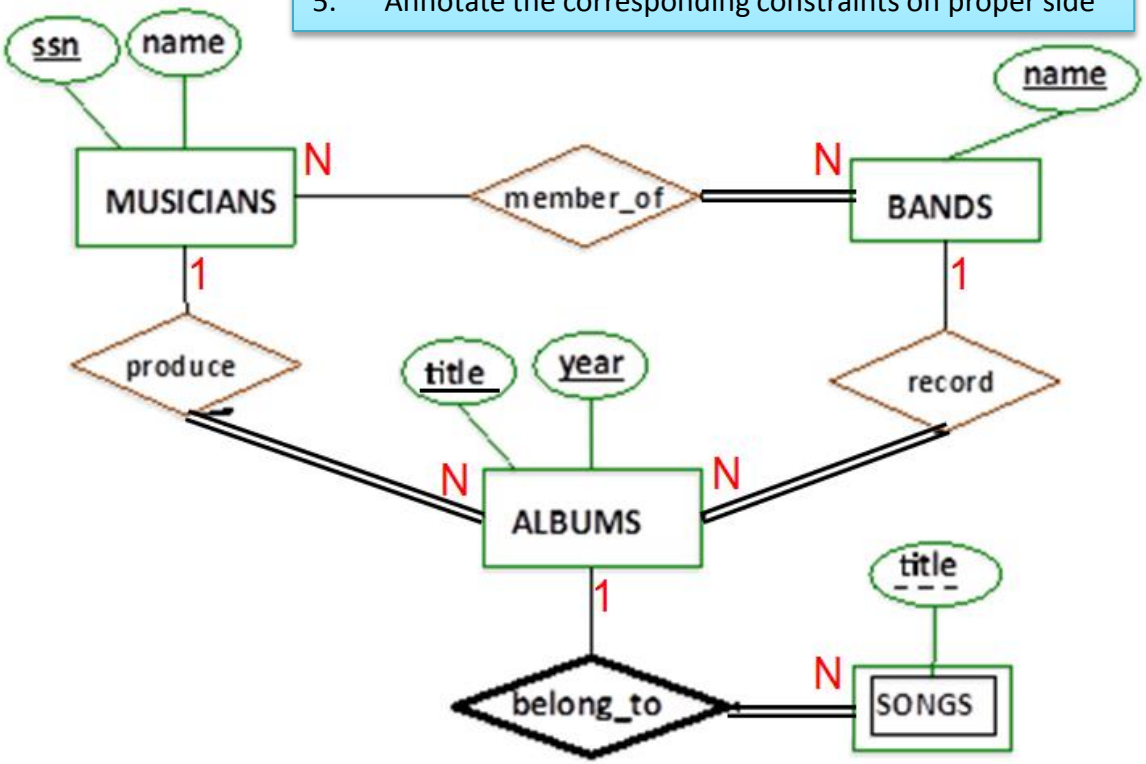
Entity , **attribute** , primary key , *relationship*

1. Find entity sets
2. Find their attributes correspondingly
3. Decide primary key among attributes for each entity set
4. Identify the relationships involving which entity sets
5. Annotate the corresponding constraints on proper side

(a) Each **musician** who records at the studio has a social security number (**ssn**) and a **name**, and no two musicians have the same ssn. Musicians form **bands**. A band is described by a unique name and has *at least one* musician as a member.

Bands record **albums**, which have a **title** and a **year** of production. Each album is recorded by *exactly one* band, and no two albums have the same title and the same production year. Each album is produced by *exactly one* musician. (Don't worry about whether that musician is a member of the recording band.)

Albums are made up of **songs**, described by their **titles**. You may assume that, if the studio no longer wants to store information about an album, then it also does not store the songs belonging to that album. Naturally, each song belongs to *exactly one* album, and all songs on the same album *have different* titles.



Part 1 (40pts):
Entity-Relationship modeling

- A recording studio needs help designing its database. The studio stores information about musicians and albums**. Draw an ER diagram** describing the studio's database for each of the two scenarios described below. Assume that the only business rules that hold are those stated below, and that no additional business rules hold. **Clearly mark all key and participation constraints.**

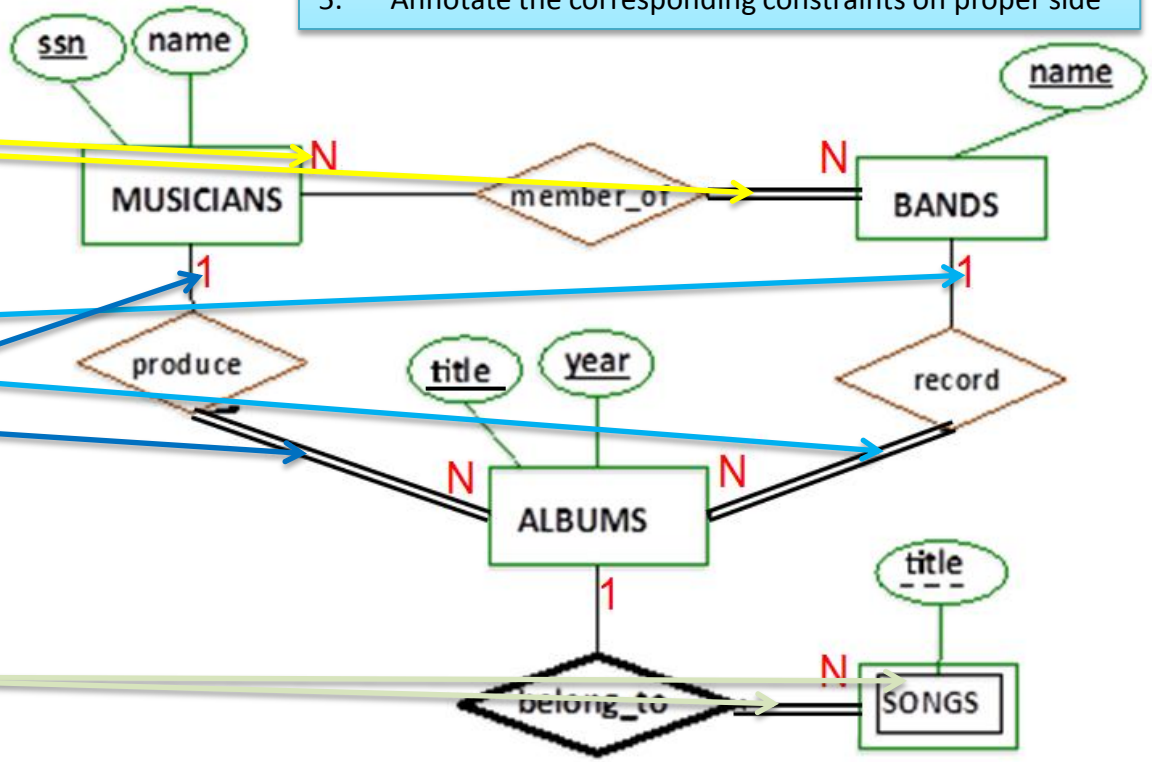
Entity , **attribute** , primary key , *relationship*

1. Find entity sets
2. Find their attributes correspondingly
3. Decide primary key among attributes for each entity set
4. Identify the relationships involving which entity sets
5. Annotate the corresponding constraints on proper side

(a) Each **musician** who records at the studio has a social security number (**ssn**) and a **name**, and no two musicians have the same ssn. Musicians form **bands**. A band is described by a unique name and has *at least one* musician as a member.

Bands record **albums**, which have a **title** and a **year** of production. Each album is recorded by *exactly one* band, and no two albums have the same title and the same production year. Each album is produced by *exactly one* musician. (Don't worry about whether that musician is a member of the recording band.)

Albums are made up of **songs**, described by their **titles**. You may assume that, if the studio no longer wants to store information about an album, then it also does not store the songs belonging to that album. Naturally, each song belongs to *exactly one* album, and all songs on the same album *have different* titles.



Part 1 (b)

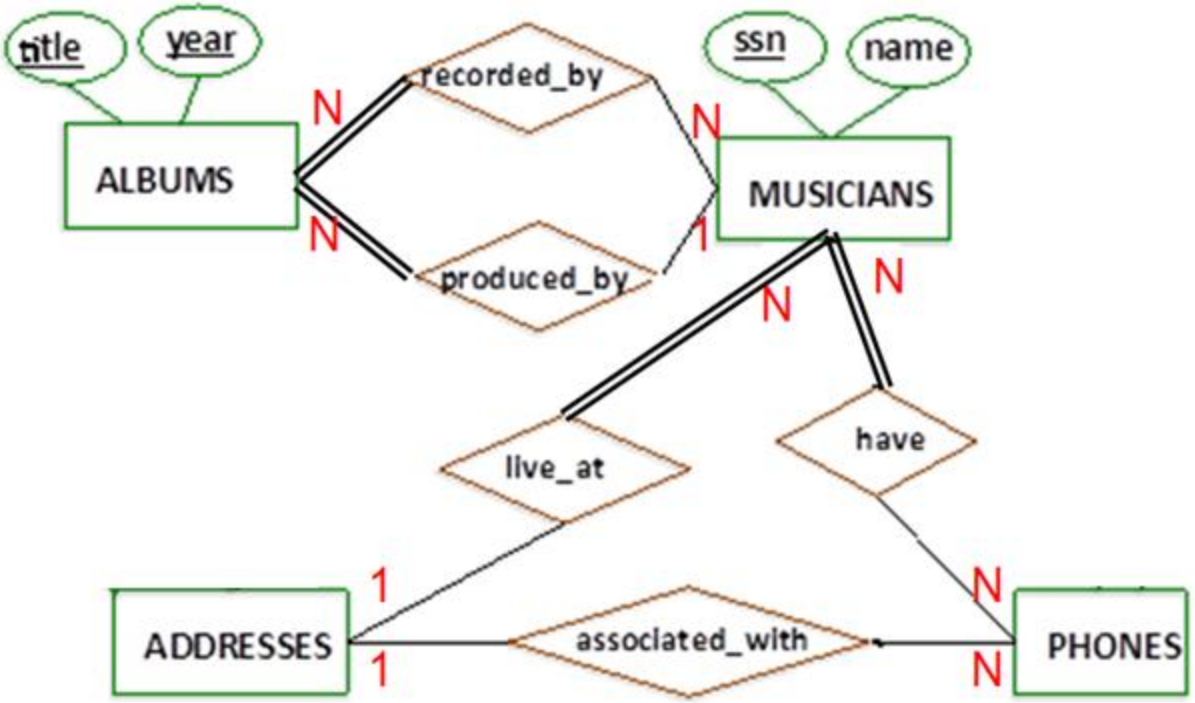
Part 1 (40pts):
Entity-Relationship modeling

- A recording studio needs help designing its database. The studio stores information about musicians and albums**. Draw an ER diagram** describing the studio's database for each of the two scenarios described below. Assume that the only business rules that hold are those stated below, and that no additional business rules hold. **Clearly mark all key and participation constraints.**

1. Find entity sets
2. Find their attributes correspondingly
3. Decide primary key among attributes for each entity set
4. Identify the relationships involving which entity sets
5. Annotate the corresponding constraints on proper side

(b) Each musician who records at the studio has a social security number (ssn), a name, an address, and a phone number. Poorly paid musicians often share the same address. A phone number may be associated with at most one address (if it's a landline). Each musician has exactly one address and at least one phone number.

Albums are described by a title and a recording year, and no two albums have both the same title and the same year. Albums are recorded by one or several musicians. Each album has exactly one musician who acts as its producer. A particular musician may produce zero, one or several albums.



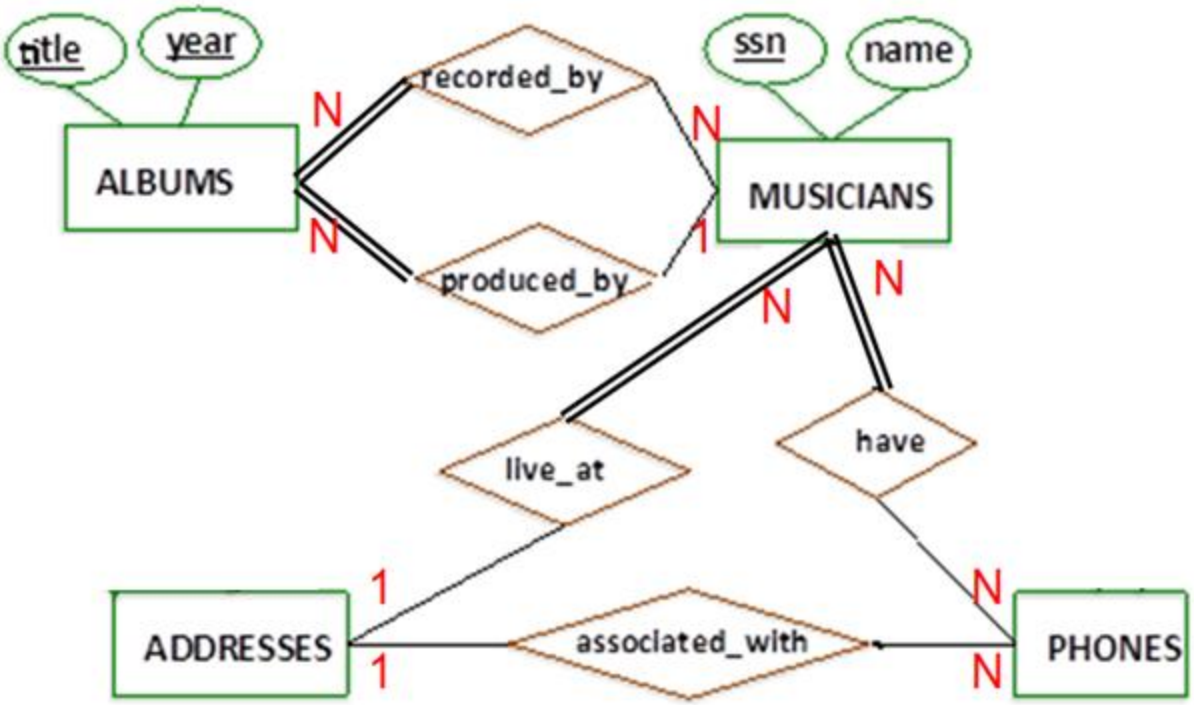
Part 1 (40pts):
Entity-Relationship modeling

- A recording studio needs help designing its database. The studio stores information about musicians and albums**. Draw an ER diagram** describing the studio's database for each of the two scenarios described below. Assume that the only business rules that hold are those stated below, and that no additional business rules hold. **Clearly mark all key and participation constraints.**

Entity , **attribute** , primary key , *relationship*

(b) Each **musician** who records at the studio has a social security number (**ssn**), a **name**, an **address**, and a **phone** number. Poorly paid musicians often *share the same* address. A phone number may be associated with *at most one* address (if it's a landline). Each musician has *exactly one* address and *at least one* phone number.

Albums are described by a **title** and a recording **year**, and no two albums have both the same title and the same year. Albums *are* recorded by *one or several* musicians. Each album has *exactly one* musician who acts as its producer. A particular musician may produce *zero, one or several* albums.



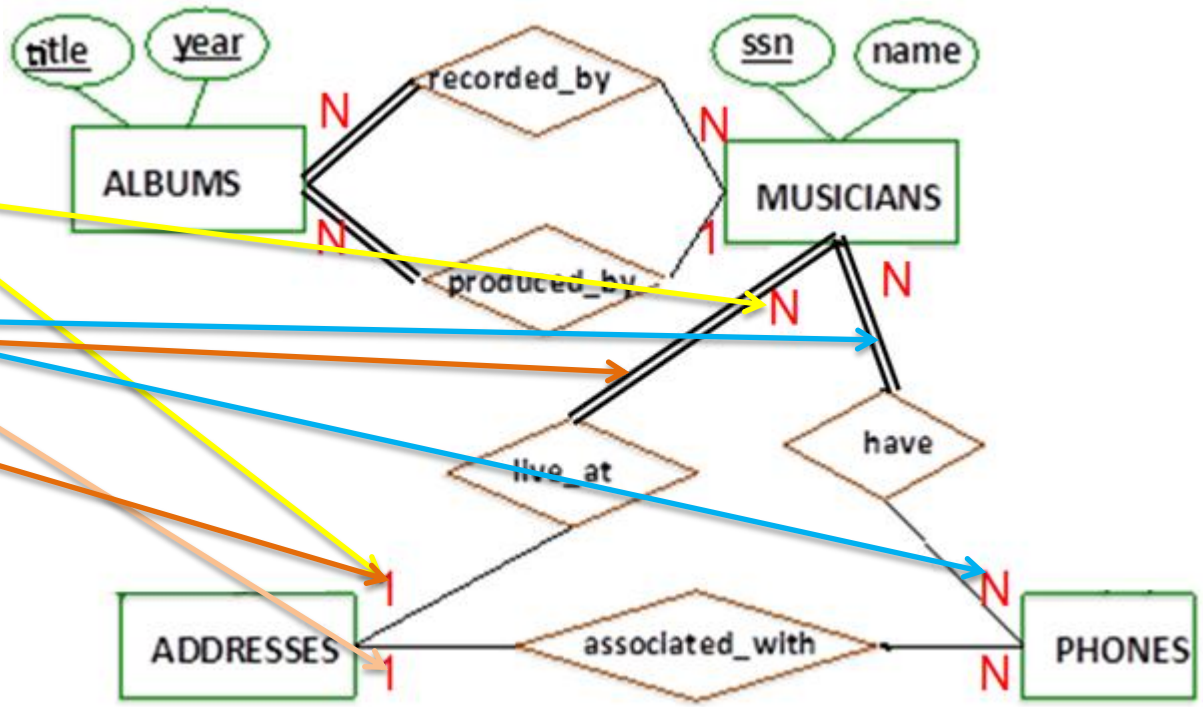
Part 1 (40pts):
Entity-Relationship modeling

- A recording studio needs help designing its database. The studio stores information about musicians and albums**. Draw an ER diagram** describing the studio's database for each of the two scenarios described below. Assume that the only business rules that hold are those stated below, and that no additional business rules hold. **Clearly mark all key and participation constraints.**

Entity , **attribute** , primary key , *relationship*

(b) Each **musician** who records at the studio has a social security number (**ssn**), a **name**, an **address**, and a **phone** number. Poorly paid musicians often *share the same* address. A phone number may be associated with *at most one* address (if it's a landline). Each musician has *exactly one* address and *at least one* phone number.

Albums are described by a **title** and a recording **year**, and no two albums have both the same title and the same year. Albums *are* recorded by *one or several* musicians. Each album has *exactly one* musician who acts as its producer. A particular musician may produce *zero, one or several* albums.



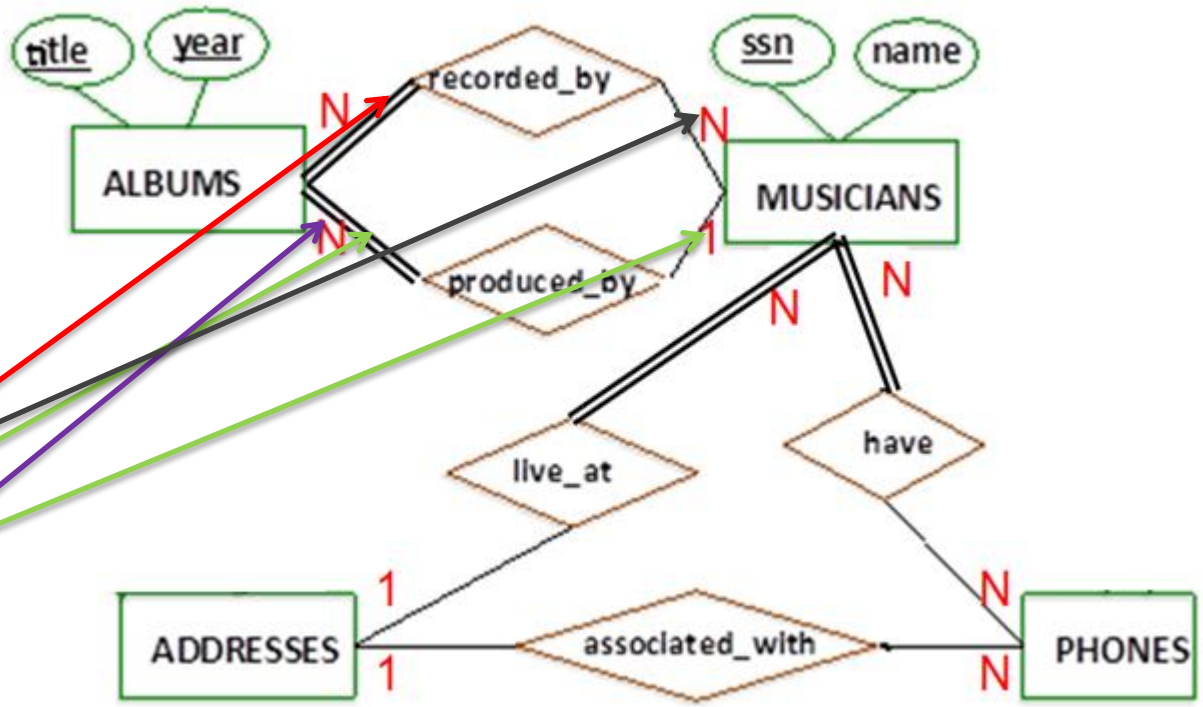
Part 1 (40pts):
Entity-Relationship modeling

- A recording studio needs help designing its database. The studio stores information about musicians and albums**. Draw an ER diagram** describing the studio's database for each of the two scenarios described below. Assume that the only business rules that hold are those stated below, and that no additional business rules hold. **Clearly mark all key and participation constraints.**

Entity , **attribute** , primary key , *relationship*

(b) Each **musician** who records at the studio has a social security number (**ssn**), a **name**, an **address**, and a **phone** number. Poorly paid musicians often *share the same* address. A phone number may be associated with *at most one* address (if it's a landline). Each musician has *exactly one* address and *at least one* phone number.

Albums are described by a **title** and a recording **year**, and no two albums have both the same title and the same year. Albums *are* recorded by *one or several* musicians. Each album has *exactly one* musician who acts as its producer. A particular musician may produce *zero, one or several* albums.



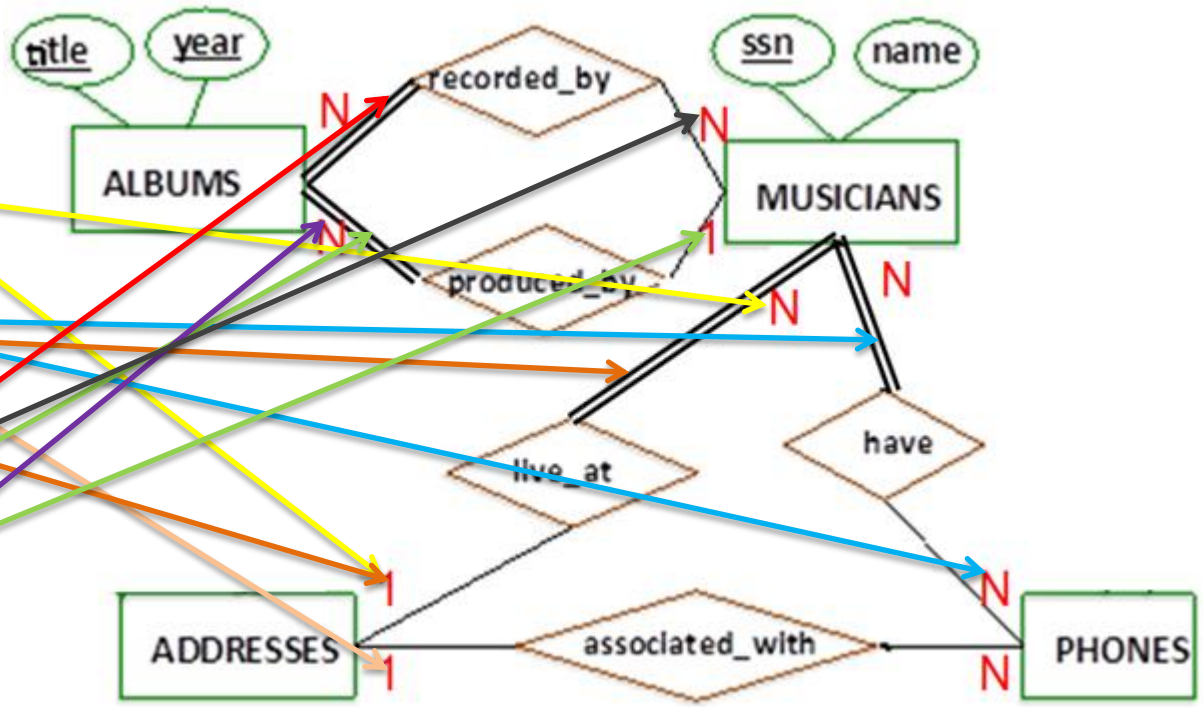
Part 1 (40pts):
Entity-Relationship modeling

- A recording studio needs help designing its database. The studio stores information about musicians and albums**. Draw an ER diagram** describing the studio's database for each of the two scenarios described below. Assume that the only business rules that hold are those stated below, and that no additional business rules hold. **Clearly mark all key and participation constraints.**

Entity , **attribute** , primary key , *relationship*

(b) Each **musician** who records at the studio has a social security number (**ssn**), a **name**, an **address**, and a **phone** number. Poorly paid musicians often *share the same* address. A phone number may be associated with *at most one* address (if it's a landline). Each musician has *exactly one* address and *at least one* phone number.

Albums are described by a **title** and a recording **year**, and no two albums have both the same title and the same year. Albums *are* recorded by *one or several* musicians. Each album has *exactly one* musician who acts as its producer. A particular musician may produce *zero, one or several* albums.

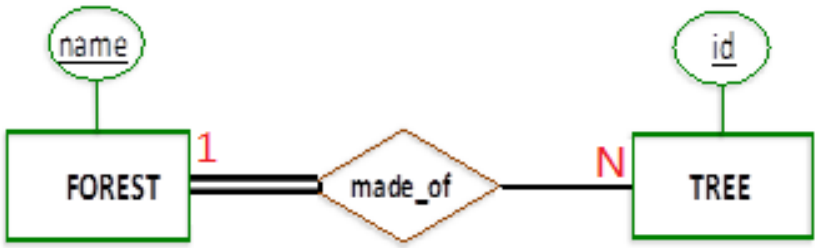


Part 2 (a)

Part 2 (60pts):
Translating ER models to relational schemas

- Consider ER diagrams below. **Write a SQL statement** (*create table*) that implements the constraints specified by the ER diagram below. Create as many tables as required. **Briefly explain** which constraints are captured in your relational implementation, and in what way. If a constraint cannot be implemented, state that explicitly in your explanation. You will not receive full credit without an explanation**.**

(a)



(the line connecting FOREST and made_of is double lined)

Approach 1:

```
create table Forest (
   name     varchar(128) primary key
);

create table Made_Of_Trees (
   id              number primary key,
   forest_name     varchar(128),
   foreign key (forest_name) references Forest(name)
);
```

For Approach 1:

We **cannot** implement the **participation constraint** on FOREST in this relational schema. Because key constraint (on TREE) and participation constraints (on FOREST) are not on the same entity.

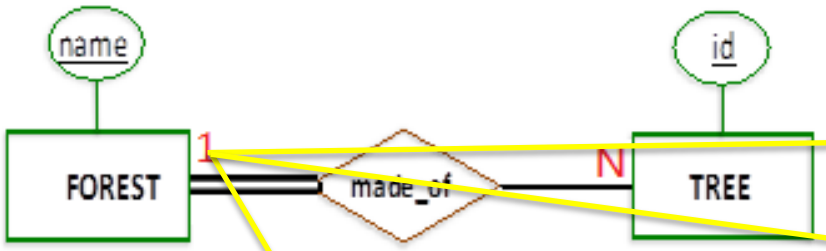
This SQL statement only models the **key constraint** on TREE, which states that a tree belongs to **at most one** forest. This constraint is implemented by making id in Made_Of_Trees a primary key in that table.

Trees that do not belong to any forest will still appear in this relation, but where the value of forest_name is the null (forest_name is not constrainted as “not null”).

Part 2 (60pts):
Translating ER models to relational schemas

- Consider ER diagrams below. **Write a SQL statement** (*create table*) that implements the constraints specified by the ER diagram below. Create as many tables as required. **Briefly explain** which constraints are captured in your relational implementation, and in what way. If a constraint cannot be implemented, state that explicitly in your explanation. You will not receive full credit without an explanation**.**

(a)



(the line connecting FOREST and made_of is double lined)

Approach 1:

```
create table Forest (
   name     varchar(128) primary key
);

create table Made_Of_Trees (
   id              number primary key,
   forest_name     varchar(128),
   foreign key (forest_name) references Forest(name)
);
```

For Approach 1:

We **cannot** implement the **participation constraint** on FOREST in this relational schema. Because key constraint (on TREE) and participation constraints (on FOREST) are not on the same entity.

This SQL statement only models the **key constraint** on TREE, which states that a tree belongs to **at most one** forest. This constraint is implemented by making id in Made_Of_Trees a primary key in that table.

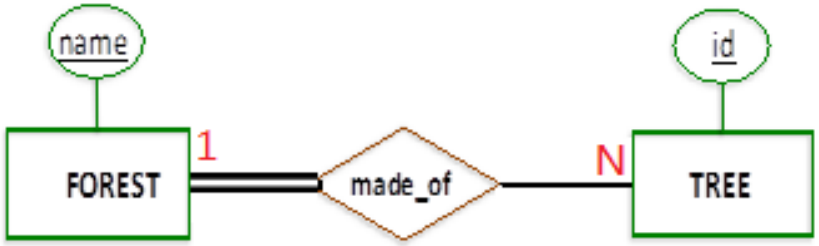
Trees that do not belong to any forest will still appear in this relation, but where the value of forest_name is the null (forest_name is not constrainted as “not null”).

Part 2 (60pts):

Translating ER models to relational schemas

- Consider ER diagrams below. **Write a SQL statement** (*create table*) that implements the constraints specified by the ER diagram below. Create as many tables as required. **Briefly explain** which constraints are captured in your relational implementation, and in what way. If a constraint cannot be implemented, state that explicitly in your explanation. You will not receive full credit without an explanation**.**

(a)



(the line connecting FOREST and made_of is double lined)

Approach 2:

```
create table Tree (
   name     varchar(128) primary key
);

create table compose_Forest (
   name     varchar(128)  primary key,
   tree_id    number not null,
   foreign key (tree_id) references Tree(id)
);
```

For Approach 2:

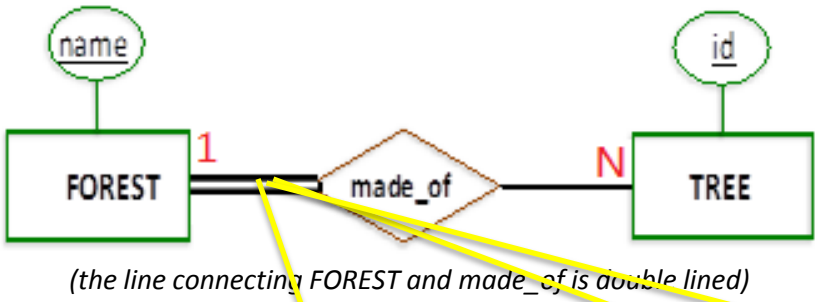
We **cannot** implement the **key constraint** on TREE in this relational schema. Because key constraint (on TREE) and participation constraints (on FOREST) are not on the same entity.

This SQL statement only models the **participation constraint** on Forest, which states that every forest has at least one tree. This participation constraint is implemented by making tree_id in the table compose_ Forest with a “not null” constraint.

Part 2 (60pts):
Translating ER models to relational schemas

- Consider ER diagrams below. **Write a SQL statement** (*create table*) that implements the constraints specified by the ER diagram below. Create as many tables as required. **Briefly explain** which constraints are captured in your relational implementation, and in what way. If a constraint cannot be implemented, state that explicitly in your explanation. You will not receive full credit without an explanation**.**

(a)



(the line connecting FOREST and made_of is double lined)

Approach 2:

```
create table Tree (
   name     varchar(128) primary key
);

create table compose_Forest (
   name     varchar(128)  primary key,
   tree_id    number not null,
   foreign key (tree_id) references Tree(id)
);
```

For Approach 2:

We **cannot** implement the **key constraint** on TREE in this relational schema. Because key constraint (on TREE) and participation constraints (on FOREST) are not on the same entity.

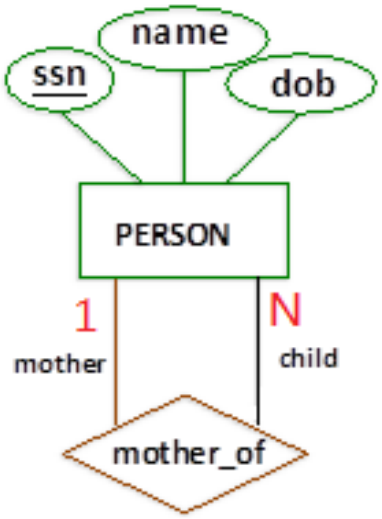
This SQL statement only models the **participation constraint** on Forest, which states that every forest has at least one tree. This participation constraint is implemented by making tree_id in the table compose_ Forest with a “not null” constraint.

Part 2 (b)

Part 2 (60pts):
Translating ER models to relational schemas

- Consider ER diagrams below. **Write a SQL statement** (*create table*) that implements the constraints specified by the ER diagram below. Create as many tables as required. **Briefly explain** which constraints are captured in your relational implementation, and in what way. If a constraint cannot be implemented, state that explicitly in your explanation. You will not receive full credit without an explanation**.**

(b)




```
create table Person (
   ssn     char(11) primary key,
   name    varchar(128),
   dob     date
);

create table Mother_Of (
   child_ssn     char(11) primary key,
   mother_ssn    char(11),
   foreign key (child_ssn) references Person(ssn),
   foreign key (mother_ssn) references Person(ssn)
);
```

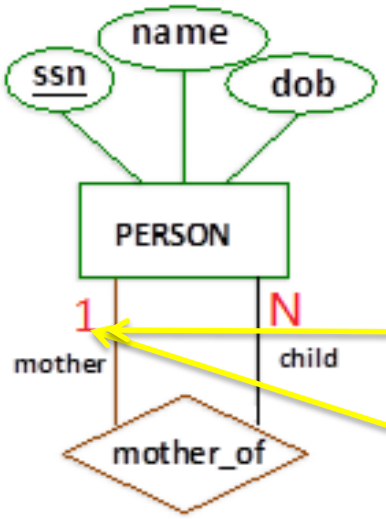
The ER diagram specifies that a person has ***at most* one** mother (key constraint on child).

The **key constraint** is implemented by designating child_ssn as primary key in the relation Mother_Of

Part 2 (60pts):
Translating ER models to relational schemas

- Consider ER diagrams below. **Write a SQL statement** (*create table*) that implements the constraints specified by the ER diagram below. Create as many tables as required. **Briefly explain** which constraints are captured in your relational implementation, and in what way. If a constraint cannot be implemented, state that explicitly in your explanation. You will not receive full credit without an explanation**.**

(b)




```
create table Person (
    ssn     char(11) primary key,
    name    varchar(128),
    dob     date
);

create table Mother_Of (
    child_ssn     char(11) primary key,
    mother_ssn    char(11),
    foreign key (child_ssn) references Person(ssn),
    foreign key (mother_ssn) references Person(ssn)
);
```

The ER diagram specifies that a person has ***at most* one** mother (key constraint on child).

The **key constraint** is implemented by designating child_ssn as primary key in the relation Mother_Of

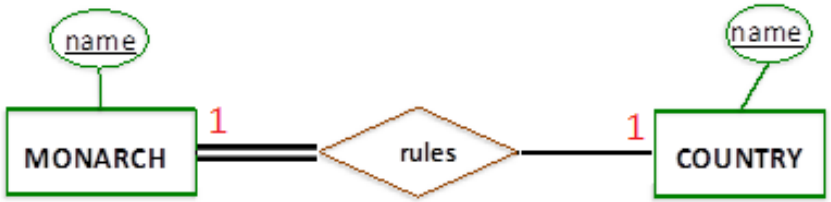
Part 2 (c)

Part 2 (60pts):

Translating ER models to relational schemas

- Consider ER diagrams below. **Write a SQL statement** (*create table*) that implements the constraints specified by the ER diagram below. Create as many tables as required. **Briefly explain** which constraints are captured in your relational implementation, and in what way. If a constraint cannot be implemented, state that explicitly in your explanation. You will not receive full credit without an explanation**.**

(c)



Approach 1:

```
create table Monarch (
   name    varchar(128) primary key
);

create table Country_Ruled_By (
   name         varchar(128) primary key,
   monarch_name  char(11) unique,
   foreign key (monarch_name) references Monarch(name)
);
```

For Approach 1:

This ER diagram specifies a **one-to-one** relationship. Further, it specifies that each monarch rules **exactly one** country (key and participation constraint on monarch). It also specify that each country is ruled by **at most one (=1 or =0)** monarch (key constraint on country).

In this SQL statement, the **key** constraint on monarch is implemented by making name in Country_Rule_By a **primary key**, and the **key** constraint on country is implemented by constraint monarch_name with "unique".

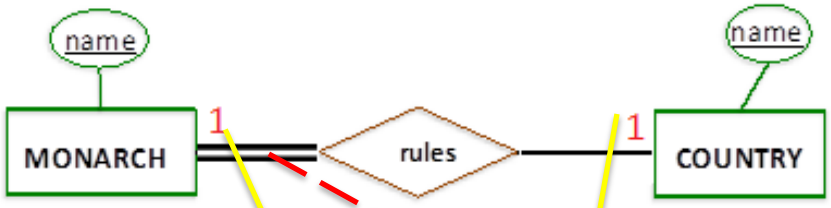
the **participation** constraint on monarch, however, is **not able to** be totally implemented in this relational schema represented by this SQL statement, since when key and participation constraints are not on same one entity set, we are not able to exactly translate it.

Part 2 (60pts):

Translating ER models to relational schemas

- Consider ER diagrams below. **Write a SQL statement** (*create table*) that implements the constraints specified by the ER diagram below. Create as many tables as required. **Briefly explain** which constraints are captured in your relational implementation, and in what way. If a constraint cannot be implemented, state that explicitly in your explanation. You will not receive full credit without an explanation**.**

(c)



Approach 1:

```
create table Monarch (
   name    varchar(128)  primary key
);

create table Country_Ruled_By (
   name          varchar(128)  primary key,
   monarch_name   char(11)  unique,
   foreign key (monarch_name) references Monarch(name)
);
```

For Approach 1:

This ER diagram specifies a **one-to-one** relationship. Further, it specifies that each monarch rules **exactly one** country (key and participation constraint on monarch). It also specify that each country is ruled by **at most one (=1 or =0)** monarch (key constraint on country).

In this SQL statement, the **key** constraint on monarch is implemented by making name in Country_Rule_By a **primary key**, and the **key** constraint on country is implemented by constraint monarch_name with "unique".

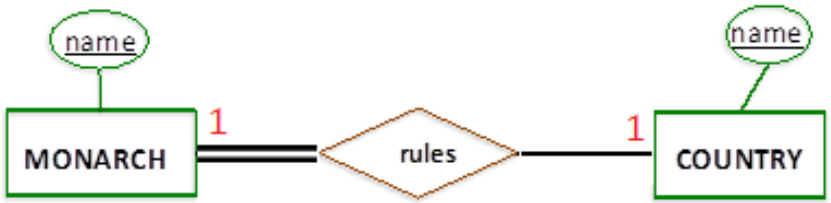
the **participation** constraint on monarch, however, is **not able to** be totally implemented in this relational schema represented by this SQL statement, since when key and participation constraints are not on same one entity set, we are not able to exactly translate it.

Part 2 (60pts):

Translating ER models to relational schemas

- Consider ER diagrams below. **Write a SQL statement** (*create table*) that implements the constraints specified by the ER diagram below. Create as many tables as required. **Briefly explain** which constraints are captured in your relational implementation, and in what way. If a constraint cannot be implemented, state that explicitly in your explanation. You will not receive full credit without an explanation**.**

(c)



Approach 2:

```
create table Country(
   name    varchar(128) primary key
);

create table Monarch_Rule (
   name          varchar(128) primary key,
   country_name  varchar (128) not null unique,
   foreign key (country_name) references Country(name)
);
```

For Approach 2:

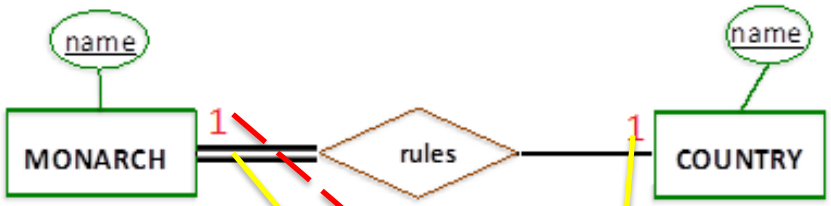
This ER diagram specifies a **one-to-one** relationship. Further, it specifies that each monarch rules **exactly one** country (key and participation constraint). The **key** constraint is implemented by making name in Monarch_Rule a **primary key**, and **participation** constraint by the **not null** constraint on country_name.

It also specify that each country is ruled by **at most one (=1 or =0)** monarch (key constraint). This key constraint could be partially implemented by the **unique** constraint on country_name (for the situation of a country ruled by 1 monarch).
However since we **cannot** totally **guarantee** the number of entity in the entity set Country is larger than the number of tuples in the relation/table Monarch_Rule (for the situation of a country ruled by 0 monarch) , so strictly speaking, We cannot totally implement this **key constraint** on COUNTRY in this relational schema.

Part 2 (60pts):
Translating ER models to relational schemas

- Consider ER diagrams below. **Write a SQL statement** (*create table*) that implements the constraints specified by the ER diagram below. Create as many tables as required. **Briefly explain** which constraints are captured in your relational implementation, and in what way. If a constraint cannot be implemented, state that explicitly in your explanation. You will not receive full credit without an explanation**.**

(c)



Approach 2:

```
create table Country(
   name    varchar(128) primary key
);

create table Monarch_Rule (
   name          varchar(128) primary key,
   country_name  varchar (128) not null unique,
   foreign key (country_name) references Country(name)
);
```

For Approach 2:

This ER diagram specifies a **one-to-one** relationship. Further, it specifies that each monarch rules **exactly one** country (key and participation constraint). The **key** constraint is implemented by making name in Monarch_Rule a **primary key**, and **participation** constraint by the **not null** constraint on country_name.

It also specify that each country is ruled by **at most one (=1 or =0)** monarch (key constraint). This key constraint could be partially implemented by the **unique** constraint on country_name (for the situation of a country ruled by 1 monarch).
However since we **cannot** totally **guarantee** the number of entity in the entity set Country is larger than the number of tuples in the relation/table Monarch_Rule (for the situation of a country ruled by 0 monarch) , so strictly speaking, We cannot totally implement this **key constraint** on COUNTRY in this relational schema.