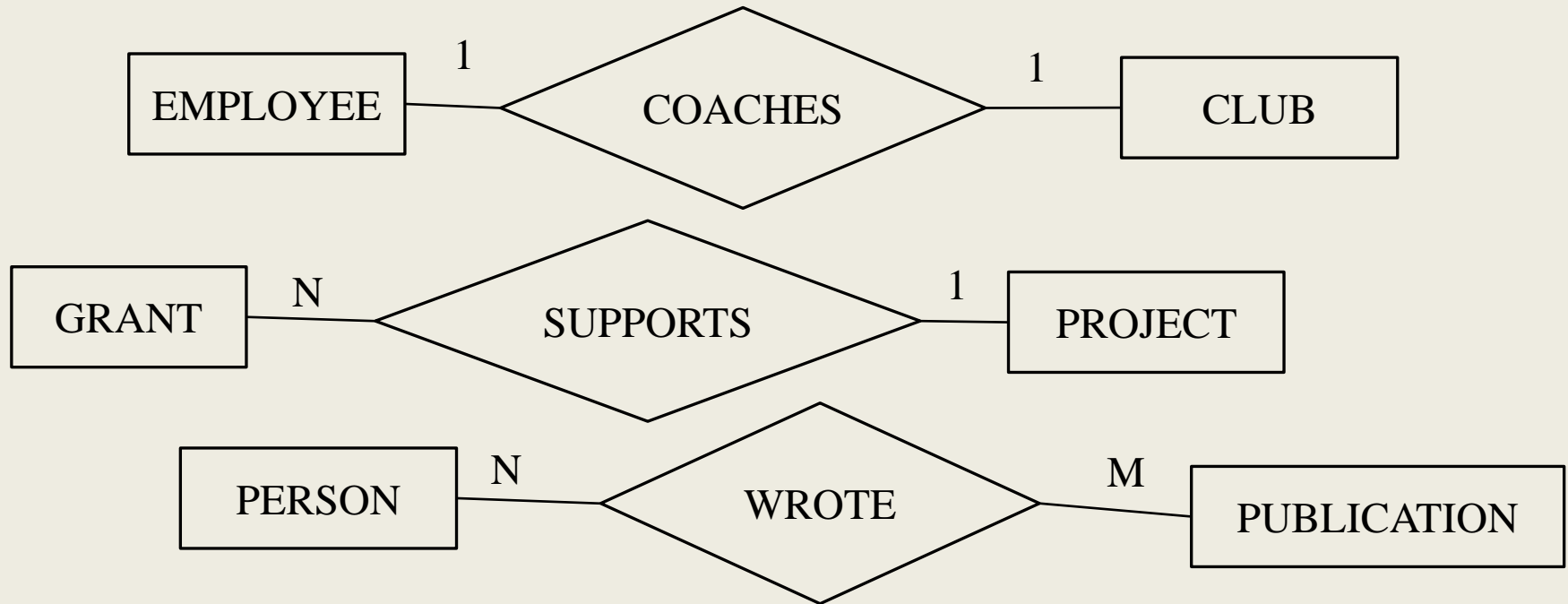


# The Relational Data Model

Textbook: chapter 3 and 9

# Wk1 Content - Relationship Cardinality

---



# Wk1 Content – Data Modelling

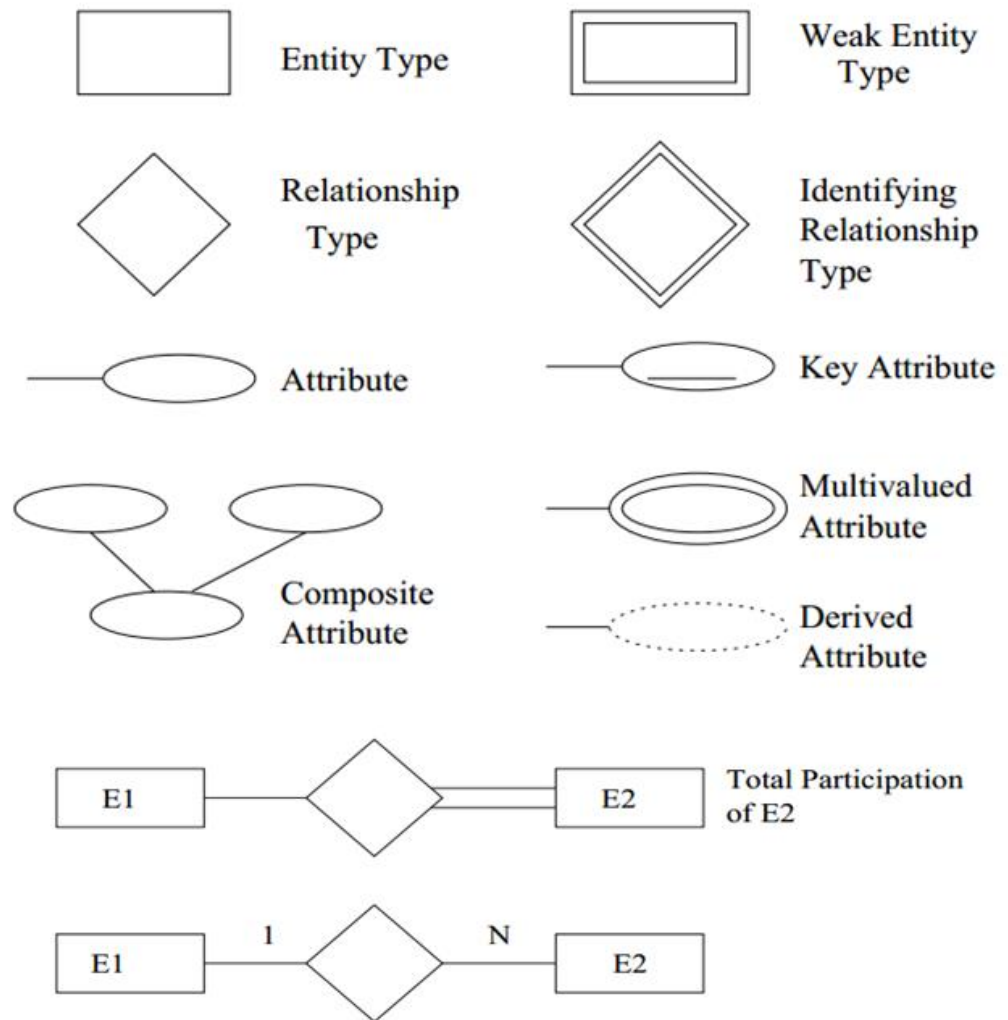
---

## Check list on ER modelling

1. Did you model every significant entity that has independent instances?
2. Did you model the entity in the correct type? Strong entity or weak entity?
3. Did you capture all the main relationships between entities?
4. Does every relationship have the correct cardinality
5. Did you correctly capture participation? is it too loose? Too strict?
6. Is each attribute modelled with the most appropriate attribute type?
7. (For comp9311) did you use the comp9311 notation?

# Wk1 Content – Standard Notation

1. For ER Diagrams
2. These should be complete



# Wk1 Content (Copied from Slides)

---

Recall the ER Model

In time, improvements were made to ER model.

## English Sentence Structures and EER Modeling

Sven Hartmann

Sebastian Link\*

Department of Information Systems, Information Science Research Centre  
Massey University, Palmerston North, New Zealand  
E-mail: [S.Hartmann,S.Link]@massey.ac.nz

We will look at one:

- *Specialisation*: the process of defining a set of subclasses of an entity type; this entity type is called the superclass of the specialization.
- *Generalisation*: a reverse process of specialisation.

# Enhanced ER (EER) model<sub>(cont)</sub>

---

A specialisation involves the following aspects:

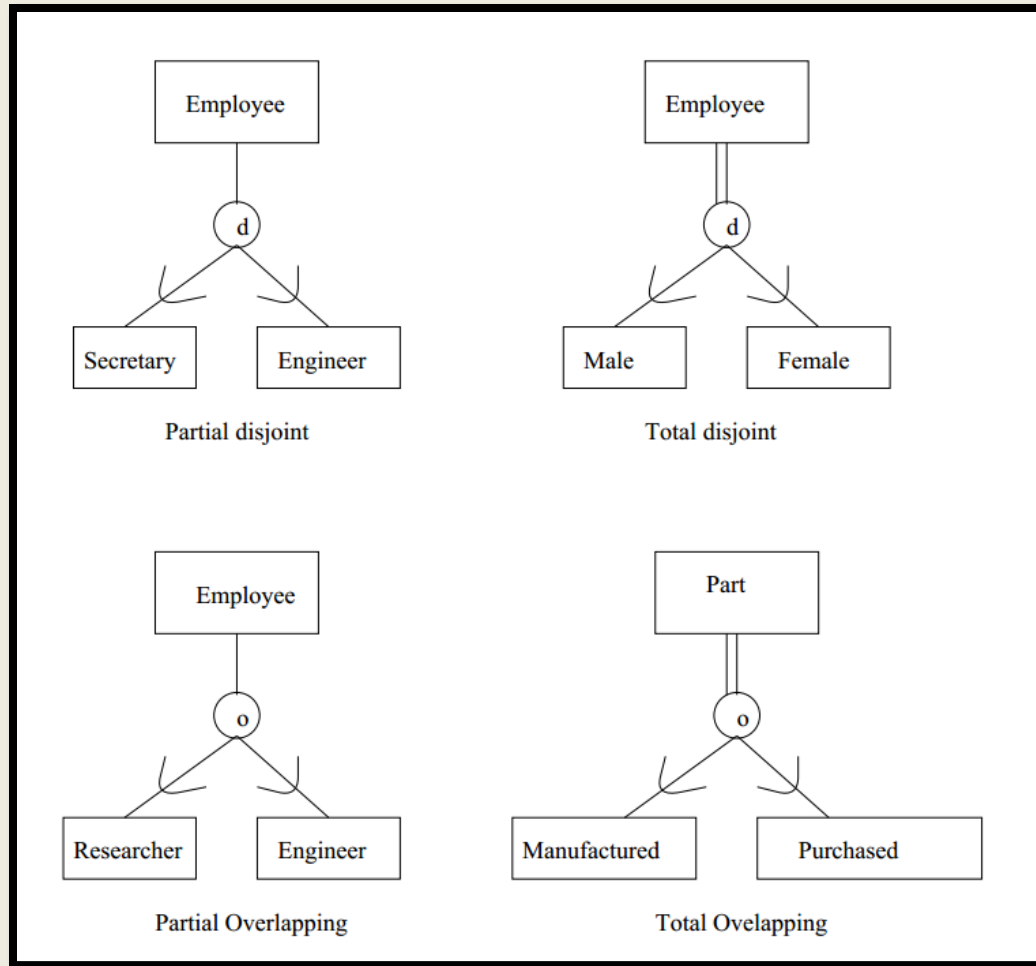
- Define a set of **subclasses** of an entity type.
- Associate additional specific attributes with each subclass.
- Establish additional specific relationship types between each subclass and other entity types, or other subclasses.

A subclass may have multiple superclasses.

A specialisation:

- may be either total or partial; and
- may be either disjoint or overlapping.

# Wk1 Content - Enhanced ER (EER) model



Just be mindful that these solutions exist.

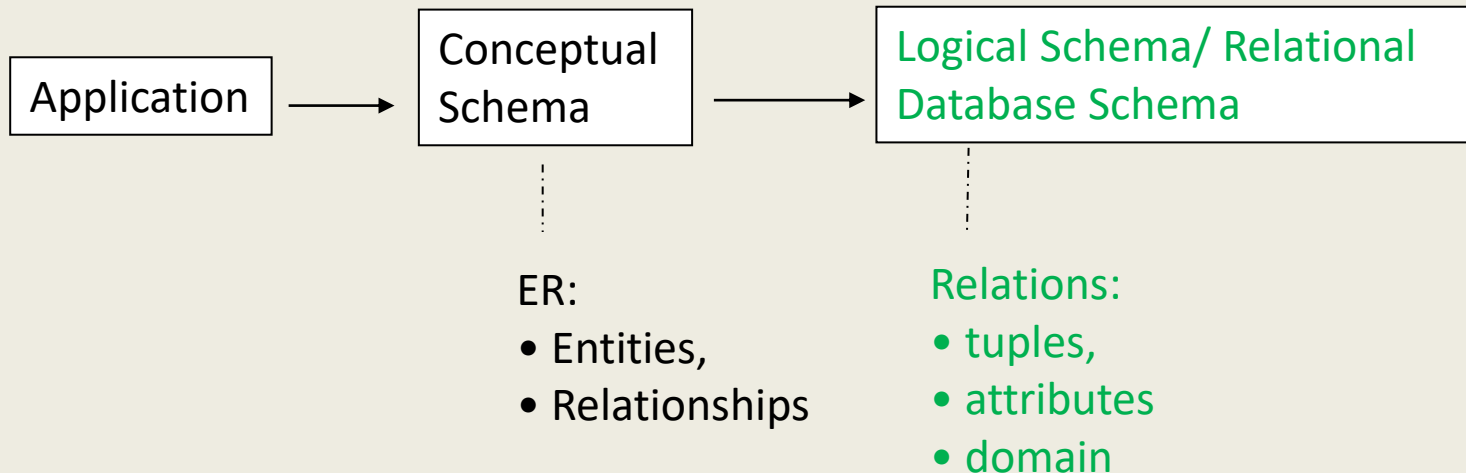
We won't ask you to model EER, not assessed in COMP9311

Notation here is also not std. notation

# Introduction

---

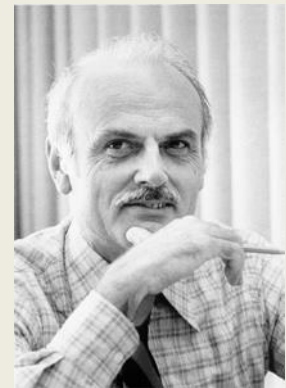
Most popular data model for database systems (see wk1 Monday)



English computer scientist Edgar F. Codd

**A Relational Model of Data for Large Shared Data Banks (1970)**

<https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf>





# The Relational Data Model

Definition:

A **relation schema**  $R$  is a finite set of attribute names,  $R = (A_1, A_2, \dots, A_n)$ .

- For each **attribute** name,  $A_i$ , there is a corresponding **domain** of  $A_i$ , denoted  $D_i$ , which is a **set of values** that  $A_i$  can take.

A **relation**  $r$  on the relation schema  $R$  is a subset of  $\mathbf{D} = D_1 \times D_2 \times \dots \times D_n$ .

From a different angle, a relation  $r$  is a finite set of mappings,  $\{t_1, t_2, \dots, t_m\}$ , from  $R$  to  $\mathbf{D}$  under the condition that each attribute  $A_i$  value of  $t_j$  must be taken from  $D_i$ .

A mapping  $t_j$  is called a **tuple**.

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

# The Relational Data Model (2)

---

Basically...

$A_1, A_2, \dots, A_n$  are attributes

$R = (A_1, A_2, \dots, A_n)$  is a **relation schema**

Example:

instructor = (ID, name, dept name, salary)

Formally, given **domains**  $D_1, D_2, \dots, D_n$ , a **relation**  $r$  is a subset of  $D_1 \times D_2 \times \dots \times D_n$

Thus, a relation is a set of  $n$ -tuples  $(a_1, a_2, \dots, a_n)$  where each  $a_i \in D_i$

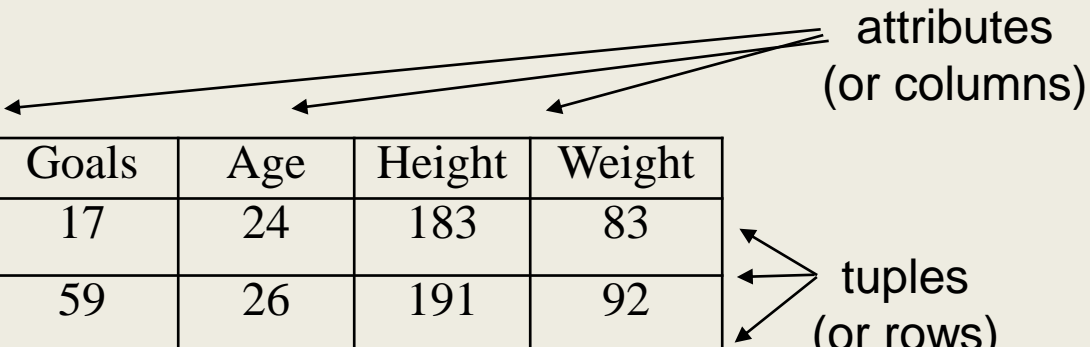
Current values (instance) of a relation are specified by a table.

An element  $t$  of  $r$  is called a **tuple**, represented by a row in a table

ID	name	dept_name	salary
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

# Example of a Relation

---



Name	Position	Goals	Age	Height	Weight
Heady	Half-forward	17	24	183	83
Sumich	Full-forward	59	26	191	92
Langdon	Utility	23	23	189	86

## Terminologies

- Type 1 : relation, tuple, attribute, ...
- Type 2 : table, record, field/column, ...

Note: If you see people talk about it differently; know they refer to the same thing

# Attributes

---

*Recall: Formally, given domains  $D1, D2, \dots, Dn$ , a relation  $r$  is a subset of  $D1 \times D2 \times \dots \times Dn$*

1. **Domain**: determines the set of values allowed in an attribute
2. Attribute values are required to be **atomic**; that is, indivisible.
3. The special value **NULL** is a member of every domain.

# Relations are Unordered

---

- Why is the order of tuples irrelevant?
- An ***unordered collection*** of elements is a ***set***:

$$\{1, 2, 3\} = \{2, 1, 3\}.$$

- An ***ordered collection*** of elements is a ***list***:

$$(1, 2, 3) \neq (2, 1, 3).$$

- A ***set*** expresses ***membership***.
  - Example: we care you are a student, but we don't care whether you're the 6th student to register (the order).

# Example of Unordered Relation

---

Both are ***the same*** relation. Ordering of column or rows are irrelevant.

PLAYER					
Name	Position	Goals	Age	Height	Weight
Heady	Half-forward	17	24	183	83
Sumich	Full-forward	59	26	191	92
Langdon	Utility	23	23	189	86

=

PLAYER					
Name	Age	Height	Weight	Goals	Position
Sumich	26	191	92	59	Full-forward
Langdon	23	189	86	23	Utility
Heady	24	183	83	17	Half-forward

# Keys

---

Assuming no two people have the same name, then {Name} is unique and therefore is a **candidate key** for PLAYER

{Goals} usually cannot be a candidate key since different players might have the same number of goals.

{Name, Goals} is a super- key but not a **minimal key** (because {Name} is a key).

Worst case, if there is no natural key, then we may need to use the whole tuple as key, or create a candidate key, say PID

PLAYER					
Name	Position	Goals	Age	Height	Weight
Heady	Half-forward	17	24	183	83
Sumich	Full-forward	59	26	191	92
Langdon	Utility	23	23	189	86

# Keys (2)

Example: Let  $\mathbf{R} = (A1, A2, A3, A4)$ .

Consider  $K \subseteq R$ , for example  
 $\{A1, A2\} \subseteq \{A1, A2, A3, A4\}$ .

Possible **super keys**

- $\{A1, A2\}$  and anything including  $\{A1, A2\}$ .
- $\{A3\}$  and anything including  $\{A3\}$
- ...

**Candidate keys:**  $\{A1, A2\}$ ,  $\{A3\}$ , ...

One of many candidate keys can be chosen as a **primary key**.

A1	A2	A3	A4
1	a	alpha	x
2	a	beta	x
2	b	gamma	y
3	c	delta	z

*In this example, we chose a key based on instances to explain key concepts. BUT always remember in practice we define the key first, and do not allow data to be inserted if it violates the key definition*



# Relation Referring to Another Relation

---

STUDENT:

<u>Person#</u>	Name
1	Dr C.C.Chen
3	Ms K.Juliff
4	Ms J.Gledill
5	Ms B.K.Lee

RESEARCHER:

<u>Person#</u>	Name
1	Dr C.C.Chen
2	Dr R.G.Wilkinson

COURSE:

<u>Department</u>	<u>Degree</u>
Psychology	Ph.D.
Comp.Sci.	Ph.D.
Comp.Sci.	M.Sc.
Psychology	M.Sc.

How do we store relationships? We can create an attribute to refer to the primary key of relation we want to refer to.

# Store the values of the Primary Key?

STUDENT:

<u>Person#</u>	Name
1	Dr C.C.Chen
3	Ms K.Juliff
4	Ms J.Gledill
5	Ms B.K.Lee

RESEARCHER:

<u>Person#</u>	Name
1	Dr C.C.Chen
2	Dr R.G.Wilkinson

COURSE:

<u>Department</u>	<u>Degree</u>
Psychology	Ph.D.
Comp.Sci.	Ph.D.
Comp.Sci.	M.Sc.
Psychology	M.Sc.

ENROLMENT:

<u>Enrolment#</u>	Supervisee	Supervisor	Department	Degree
1	1	2	Psychology	Ph.D.
2	3	1	Comp.Sci.	Ph.D.
3	4	1	Comp.Sci.	M.Sc.
4	5	1	Comp.Sci.	M.Sc.

# Relation Referring to Another Relation

---

**Foreign key:** *an attribute that keeps the value of a primary key of another relation.*

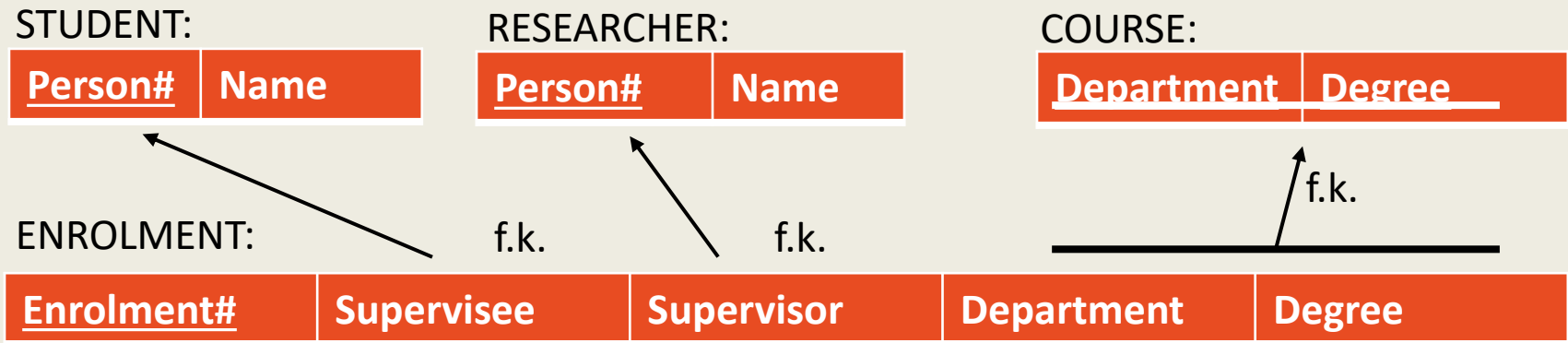
A set of attributes from a relation schema  $R_1$  may be a foreign key,  $FK$ , if

- the attributes have ***the same domains*** as the attributes in the primary key of another relation schema  $R_2$ , and
- a value of FK in a tuple  $t_1$  of  $R_1$  either occurs as a value of PK for some tuple  $t_2$  in  $R_2$  or is null.

*Closely related to the referential integrity:* The value of  $FK$  must occur in the other relation or be entirely NULL. (more on this later)

# Example of Foreign keys

This is what we mean



# Relational Integrity Constraints

---

We need to keep the relational database in a ***valid state***:

Three integrity constraints are important

1. **Key constraint**: candidate key values must be unique for every relation instance.
2. **Entity integrity**: an attribute that is part of a primary key cannot be NULL.
3. **Referential integrity**

*Valid state: a relation does not violate any integrity constraints.*

*Invalid state: a relation violates at least one integrity constraint*

# Relational Integrity Constraints

---

How can a valid relation can it ever become invalid?

A: Operations on the database can result in an invalid state.

Before proceeding with an ***update***, we need to...

- check that the result of the update will not be violate any integrity constraints.

# Insertion: Key constraint violation

STUDENT:

<u>Person#</u>	Name
1	Dr C.C.Chen
3	Ms K.Juliff
4	Ms J.Gledill
5	Ms B.K.Lee

RESEARCHER:

<u>Person#</u>	Name
1	Dr C.C.Chen
2	Dr R.G.Wilkinson

COURSE:

<u>Department</u>	<u>Degree</u>
Psychology	Ph.D.
Comp.Sci.	Ph.D.
Comp.Sci.	M.Sc.
Psychology	M.Sc.

ENROLMENT:

<u>Enrolment#</u>	Supervisee	Supervisor	Department	Degree
1	1	2	Psychology	Ph.D.
2	3	1	Comp.Sci.	Ph.D.
3	4	1	Comp.Sci.	M.Sc.
4	5	1	Comp.Sci.	M.Sc.

1. Insert < 2, *Dr.V.Ciesielski* > into RESEARCHER Allowed?

# Insertion: Entity integrity violation

STUDENT:

<u>Person#</u>	Name
1	Dr C.C.Chen
3	Ms K.Juliff
4	Ms J.Gledill
5	Ms B.K.Lee

RESEARCHER:

<u>Person#</u>	Name
1	Dr C.C.Chen
2	Dr R.G.Wilkinson

COURSE:

<u>Department</u>	<u>Degree</u>
Psychology	Ph.D.
Comp.Sci.	Ph.D.
Comp.Sci.	M.Sc.
Psychology	M.Sc.

ENROLMENT:

<u>Enrolment#</u>	Supervisee	Supervisor	Department	Degree
1	1	2	Psychology	Ph.D.
2	3	1	Comp.Sci.	Ph.D.
3	4	1	Comp.Sci.	M.Sc.
4	5	1	Comp.Sci.	M.Sc.

2. Insert < *Comp.Sci.*, *NULL* > into COURSE Allowed?



# Insertion: Referential integrity violation

STUDENT:

<u>Person#</u>	Name
1	Dr C.C.Chen
3	Ms K.Juliff
4	Ms J.Gledill
5	Ms B.K.Lee

RESEARCHER:

<u>Person#</u>	Name
1	Dr C.C.Chen
2	Dr R.G.Wilkinson

COURSE:

<u>Department</u>	<u>Degree</u>
Psychology	Ph.D.
Comp.Sci.	Ph.D.
Comp.Sci.	M.Sc.
Psychology	M.Sc.

ENROLMENT:

<u>Enrolment#</u>	Supervisee	Supervisor	Department	Degree
1	1	2	Psychology	Ph.D.
2	3	1	Comp.Sci.	Ph.D.
3	4	1	Comp.Sci.	M.Sc.
4	5	1	Comp.Sci.	M.Sc.

3. Insert < 5, 6, 2, *Psychology*, *Ph.D.* > into ENROLMENT Allowed?

# Summary on Insertion

---

How can insertions cause an invalid state?

To avoid an invalid state...

- Check that the candidate keys are not already present,
- Check that the value of each foreign key either
  - all null, or
  - all non-NULL and occurs in the referenced relation.

# Deletion: Constraint Checks

---

How can deletions cause an invalid state?

The record you're trying to delete could be referenced to as a foreign key in another relation.

Example: Delete ***tuple*** with Person# = 2 from RESEARCHER

RESEARCHER:

Person#	Name
1	Dr C.C.Chen
2	Dr R.G.Wilkinson

ENROLMENT:

Enrolment#	Supervisee	Supervisor	Department	Degree
1	1	2	Psychology	Ph.D.
2	3	1	Comp.Sci.	Ph.D.
3	4	1	Comp.Sci.	M.Sc.
4	5	1	Comp.Sci.	M.Sc.

What do we do?

# Deletion: Constraint Checks

---

We do need to delete tuples from relations sometimes, and it's possible for the record to be referred in other relations.

What can we do?

1. Delete it (*this requires another integrity check, possibly causing a cascade of deletions*), or
2. Set foreign key value to NULL (*note this can't be done if it is part of a primary key*)

# Updating Values: Constraint Checks

---

Can changing a value lead to an invalid state? Not unless you're modifying the value of a key.

If the modified attribute is primary key

- the same issues as deleting PK1 and then immediately inserting PK2.
- make sure deletion and insertion don't violate any steps.

if the modified attribute is foreign key

- check that the new value refers to an existing tuple.

Note: all relational integrity constraints are to do with the key values.

# Reducing to Relational Schema

---

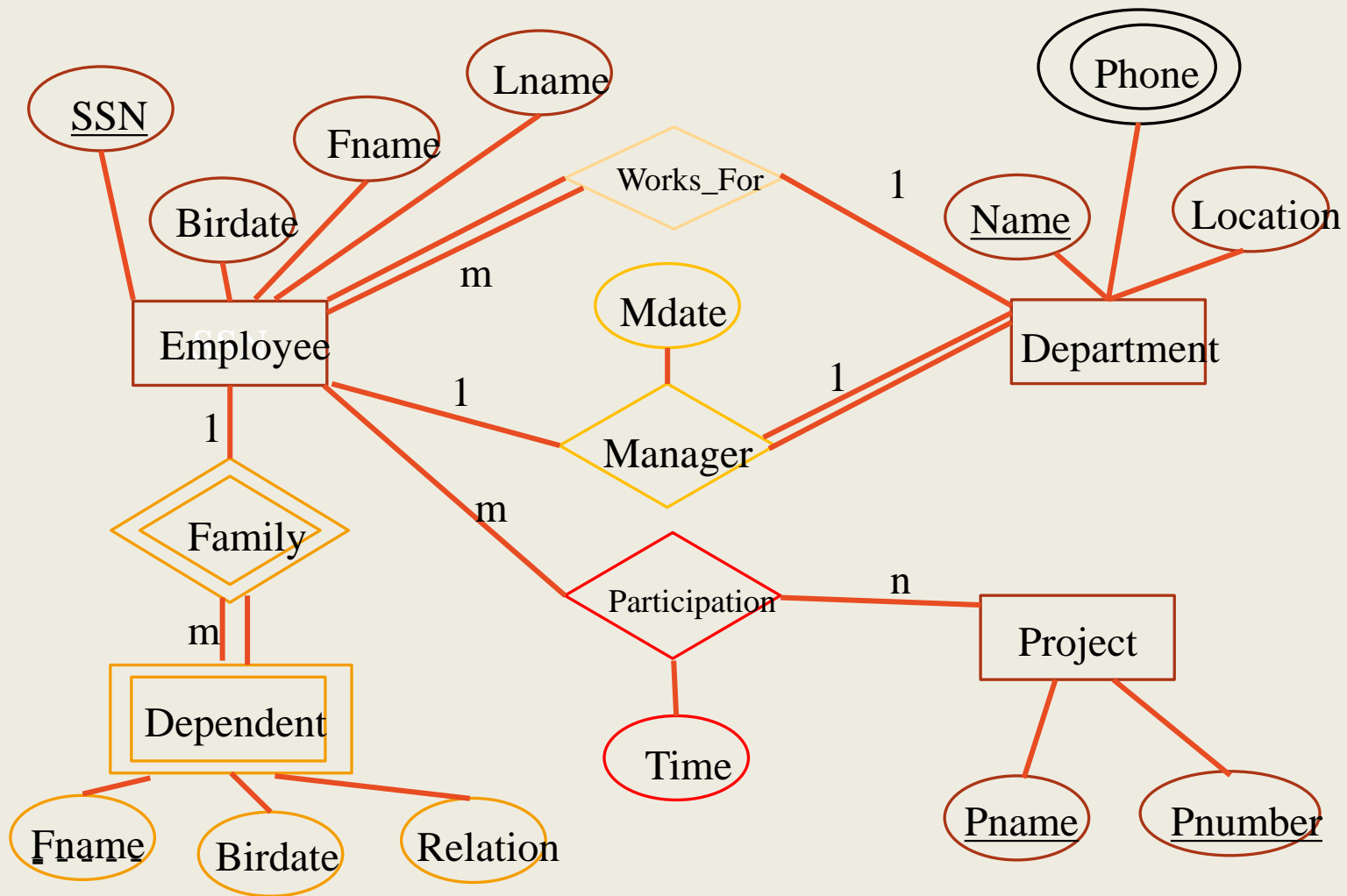
Entity sets and relationship sets can be expressed uniformly as relation schemas that represent the contents of the database.

**A database which conforms to an E-R diagram can be represented by a collection of relation schemas.**

For each entity set and relationship set, there is a unique schema with the same name of the corresponding entity set or relationship set.

Each schema has several columns which have unique names.

# Mapping ER to Relational: Guiding Example



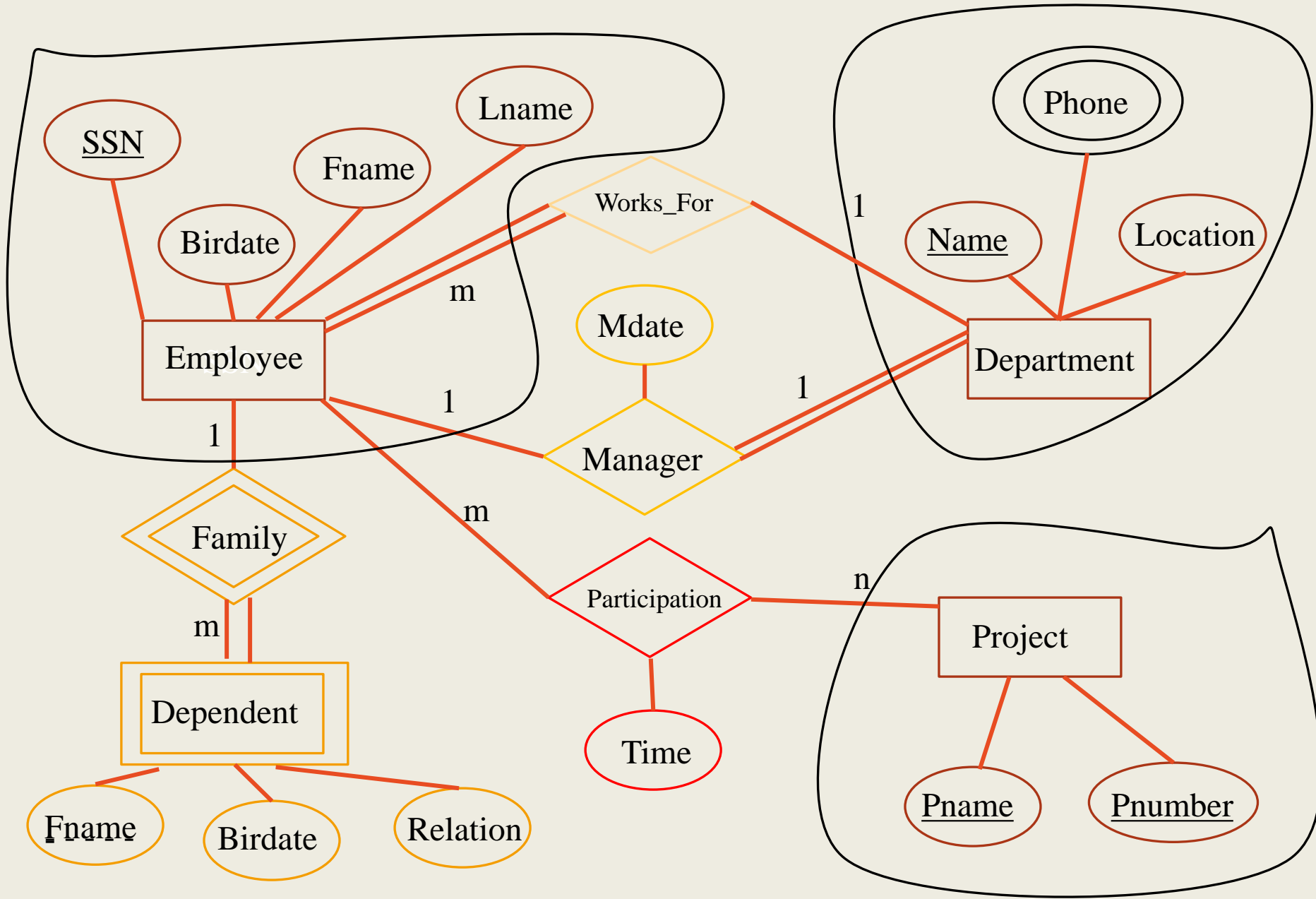
# Mapping Strong Entity Types

---

Step 1a : For each ***regular entity*** (not weak entity) type E, create a new relation R with

- Attributes : all simple attributes (and simple components of composite attributes) of E.
- Key : key of E as the primary key for the relation.





# Mapping Strong Entity Types

---

Employee

<u>SSN</u>	Fname	Lname	Birdate
------------	-------	-------	---------

Department

<u>Name</u>	Location
-------------	----------

Project

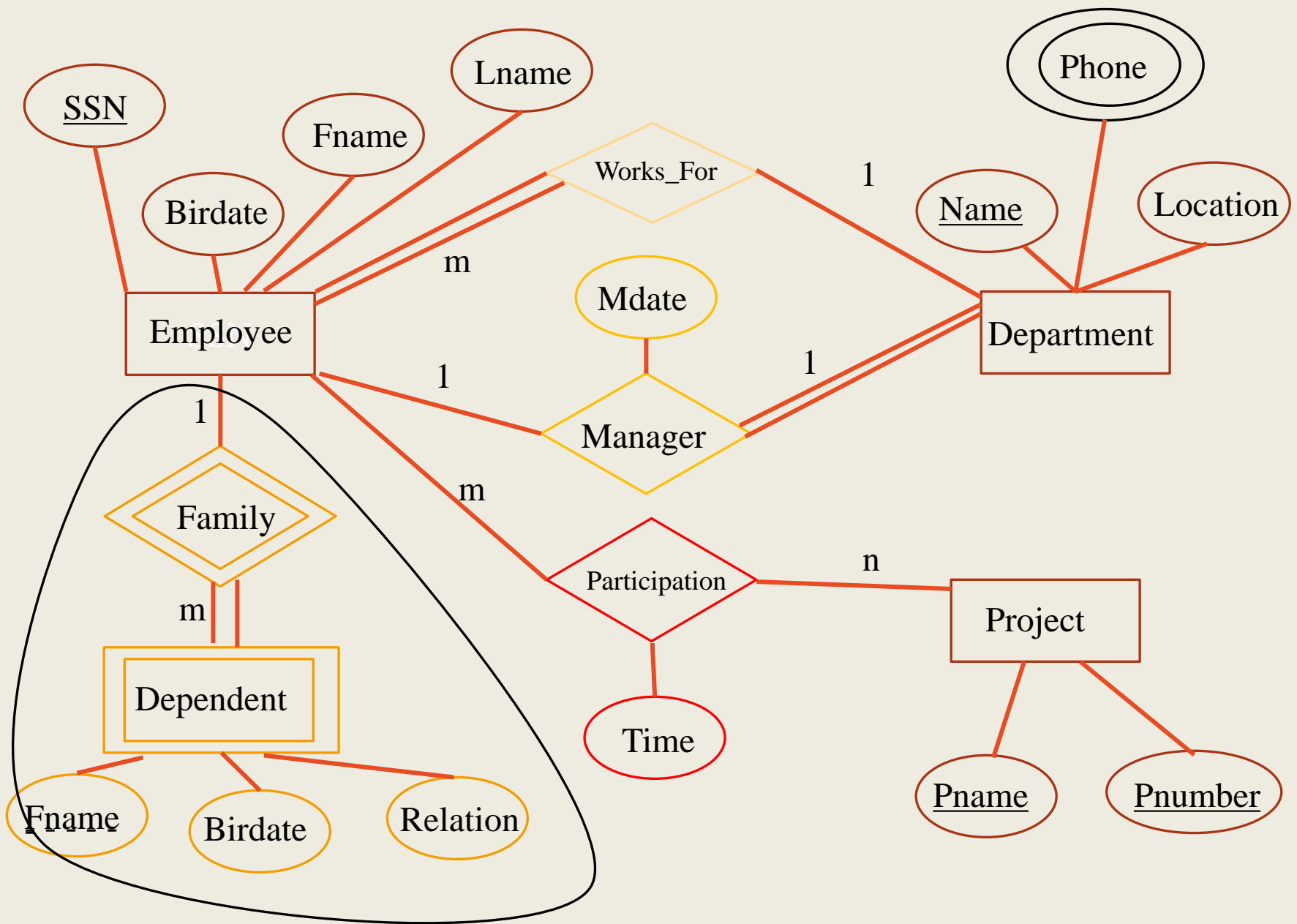
<u>Pname</u>	Pnumber
--------------	---------

# Mapping Weak Entity Types

---

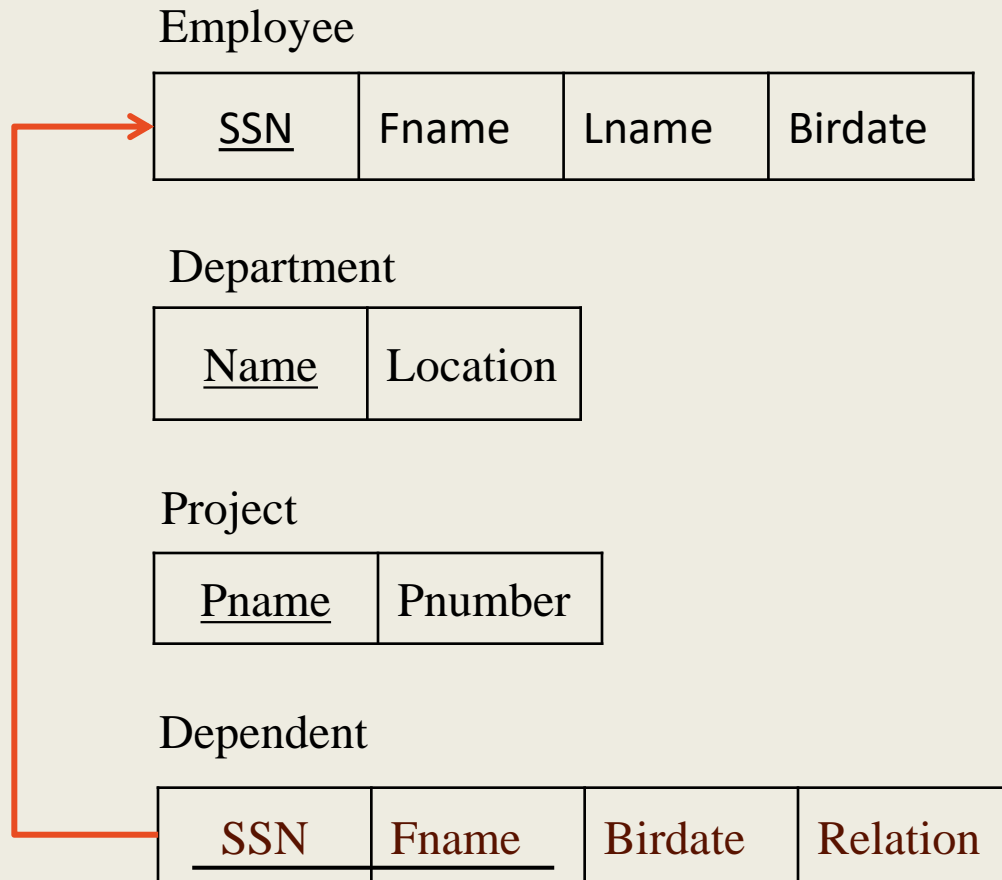
Step 2 : For each ***weak entity type*** W with the owner entity type E, create a new relation R with

- Attributes :
  - all simple attributes (and simple components of composite attributes) of W,
  - and include the primary key attributes of the relation derived from E as the foreign key.
- Key of R: foreign key to E and partial key of W.



# Mapping Weak Entity Types

---



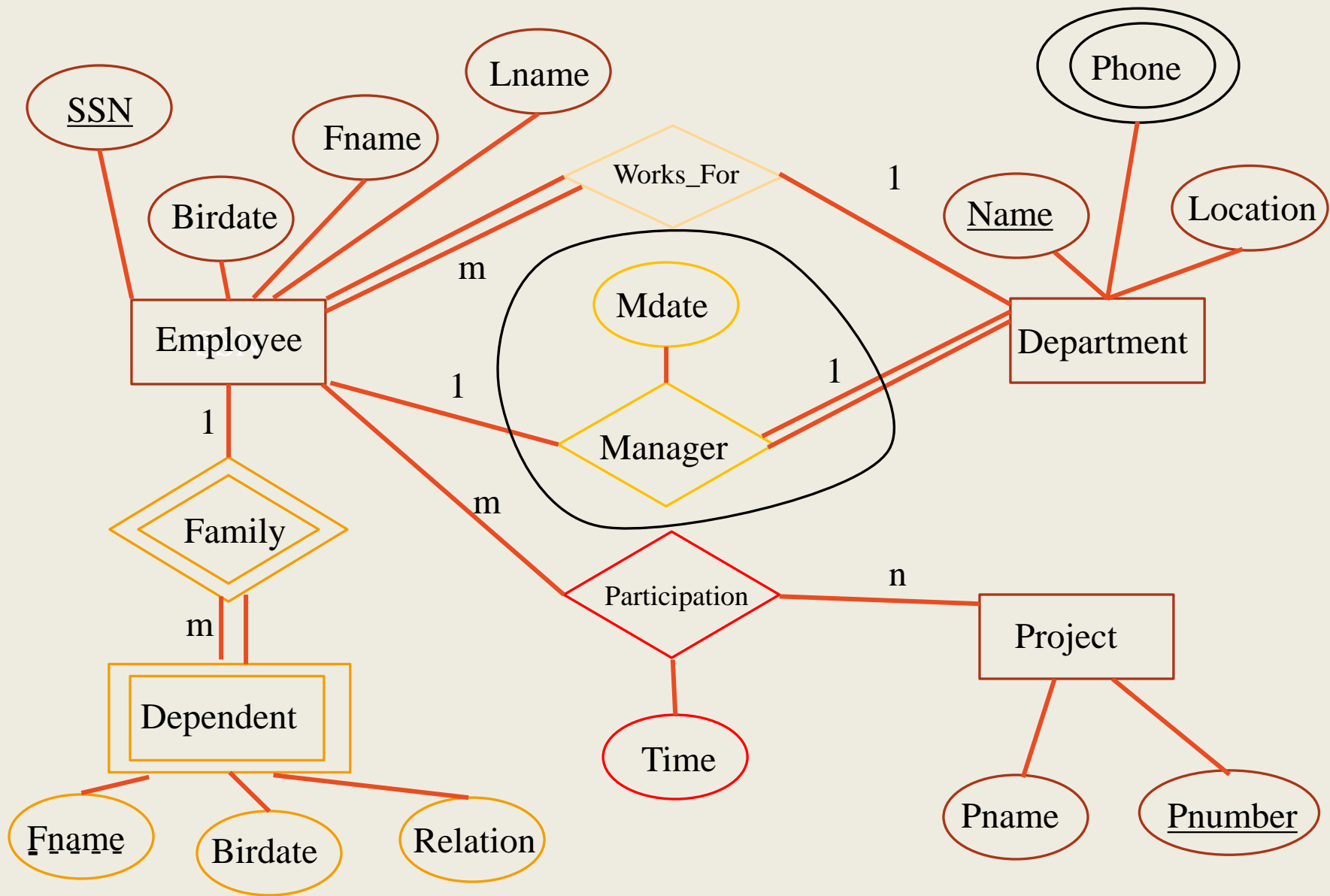
# Mapping 1:1 Relationship Types

---

Step 3 : For each **1:1 relationship type** B. Let E and F be the participating entity types. Let S and T be the corresponding relations.

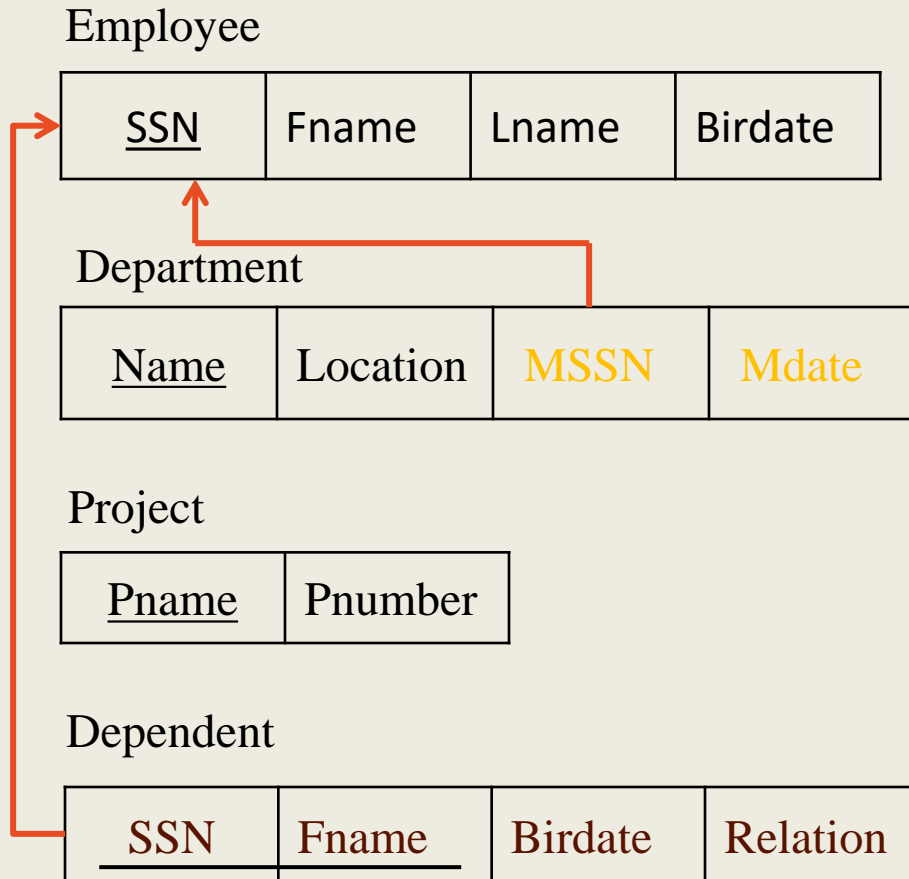
- Choose one of S and T (let S be one that participates totally if there is one).
- Add attributes from the primary key of T to S as a foreign key.
- Add all simple attributes (and simple components of composite attributes) of B as attributes of S.

*(Alternatively: merge the two entity types and the relationship into a single relation, especially if both participate totally and do not participate in other relationships).*



# Mapping 1:1 Relationship Types

---





# Mapping 1:N Relationship Types

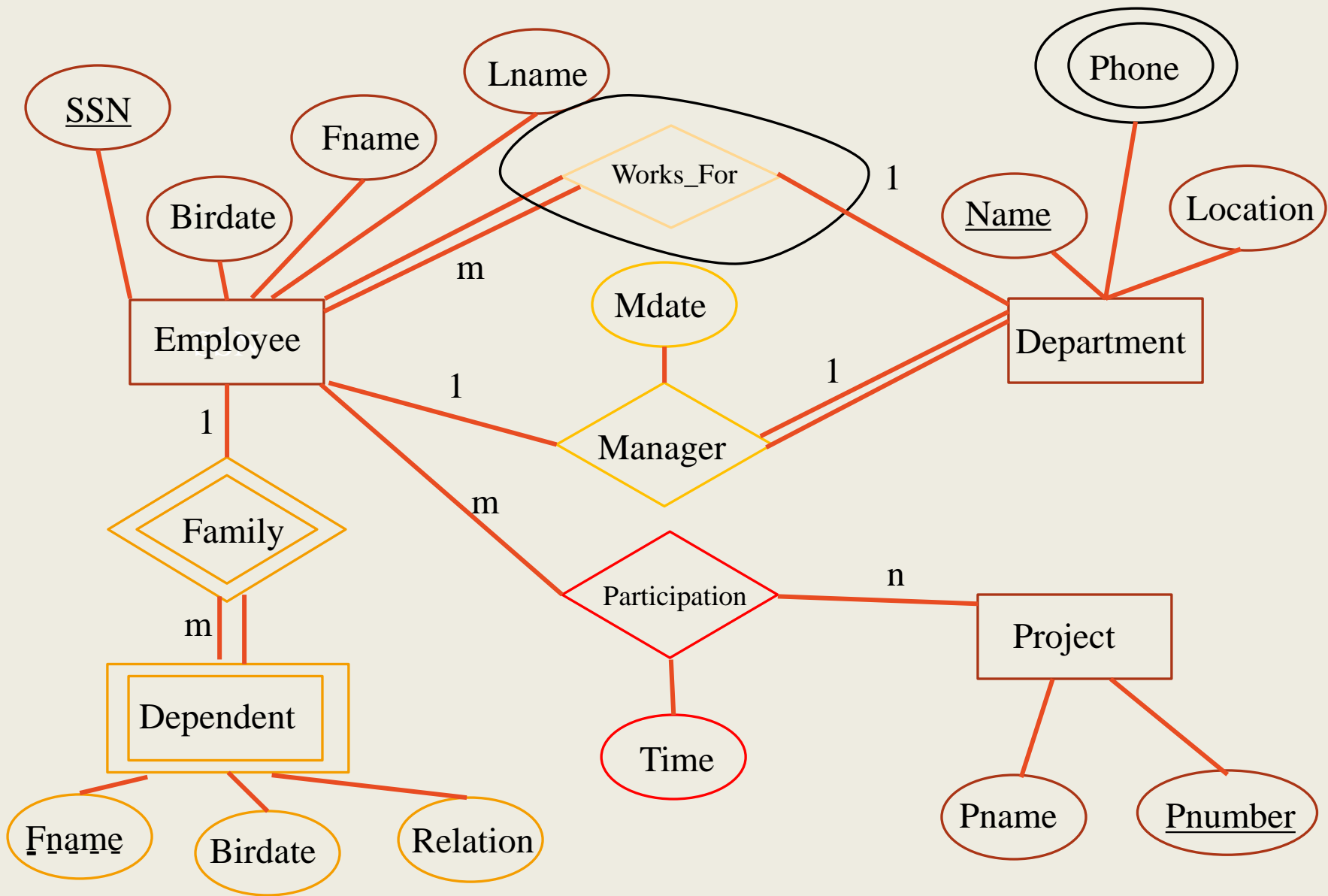
---

Step 4 : For each **1:N relationship type** B. Let S and T be the participating entity types, where S is on the 1 side and T on the N side.

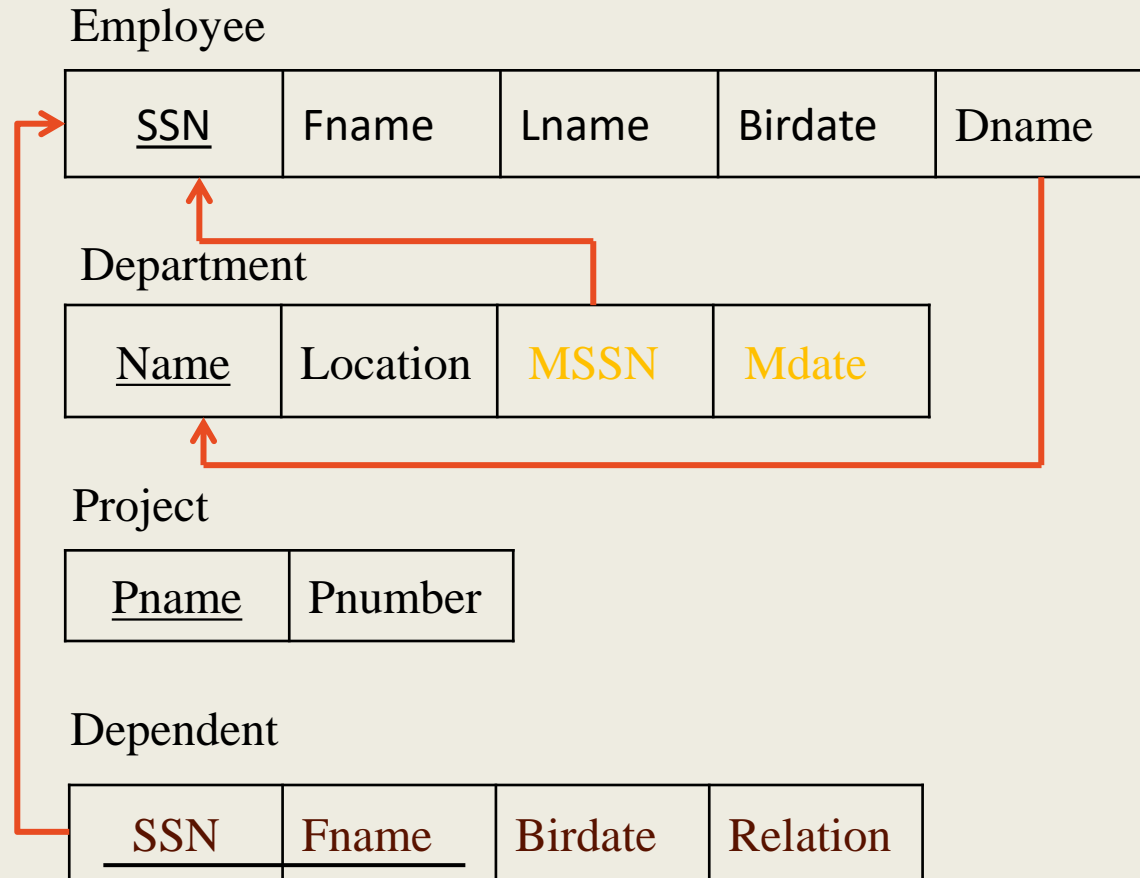
Add to the relation belonging to entity T,

- the attributes from the primary key of S as a foreign key.
- any simple attributes (or simple components of composite attributes) from relationship B.

(Notice that this doesn't add any new tuples, just attributes.)



# Mapping 1:N Relationship Types



# Mapping M:N Relationship Types

---

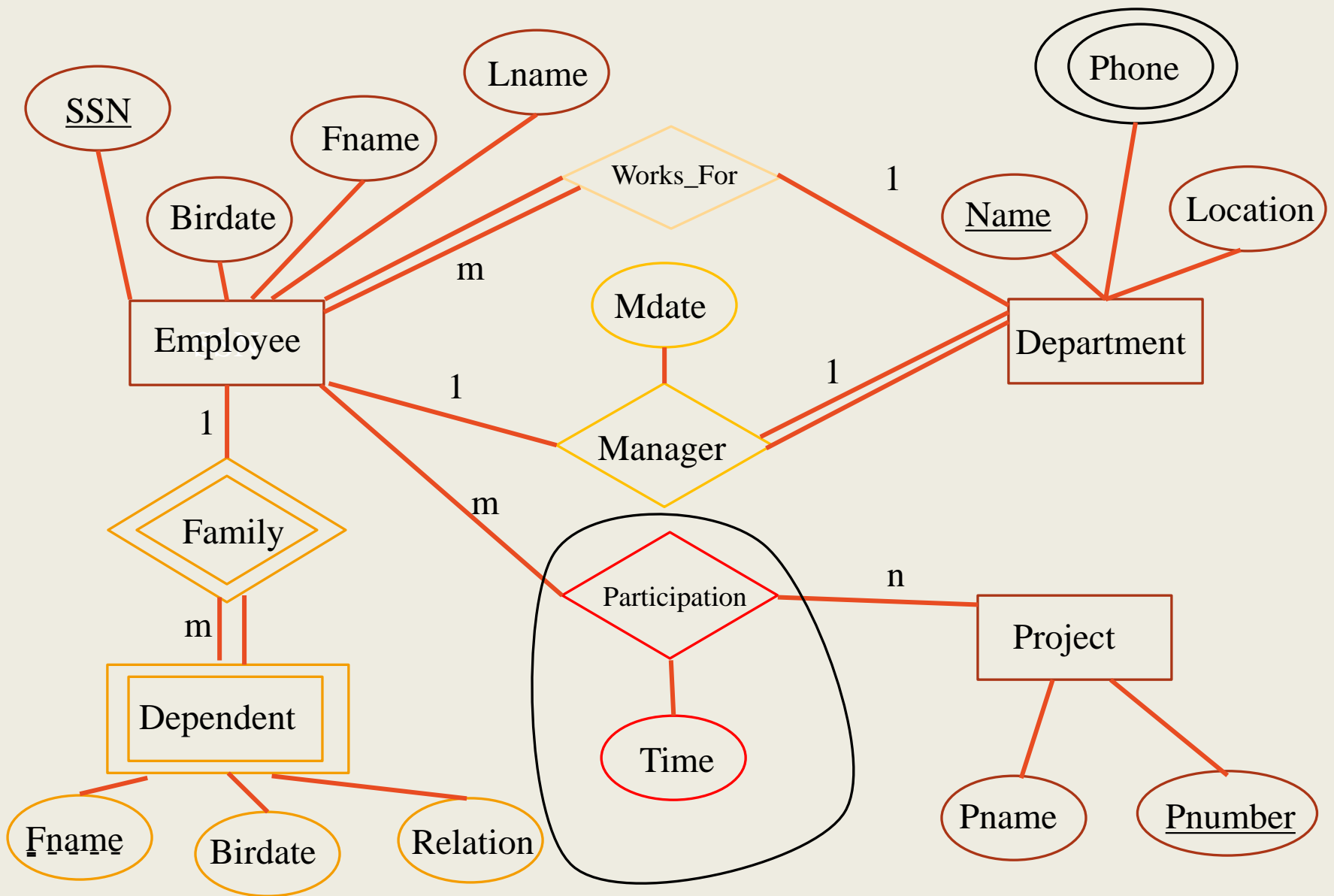
Step 5 : For each ***N:M relationship type*** B. Let S and T be the participating entity types.

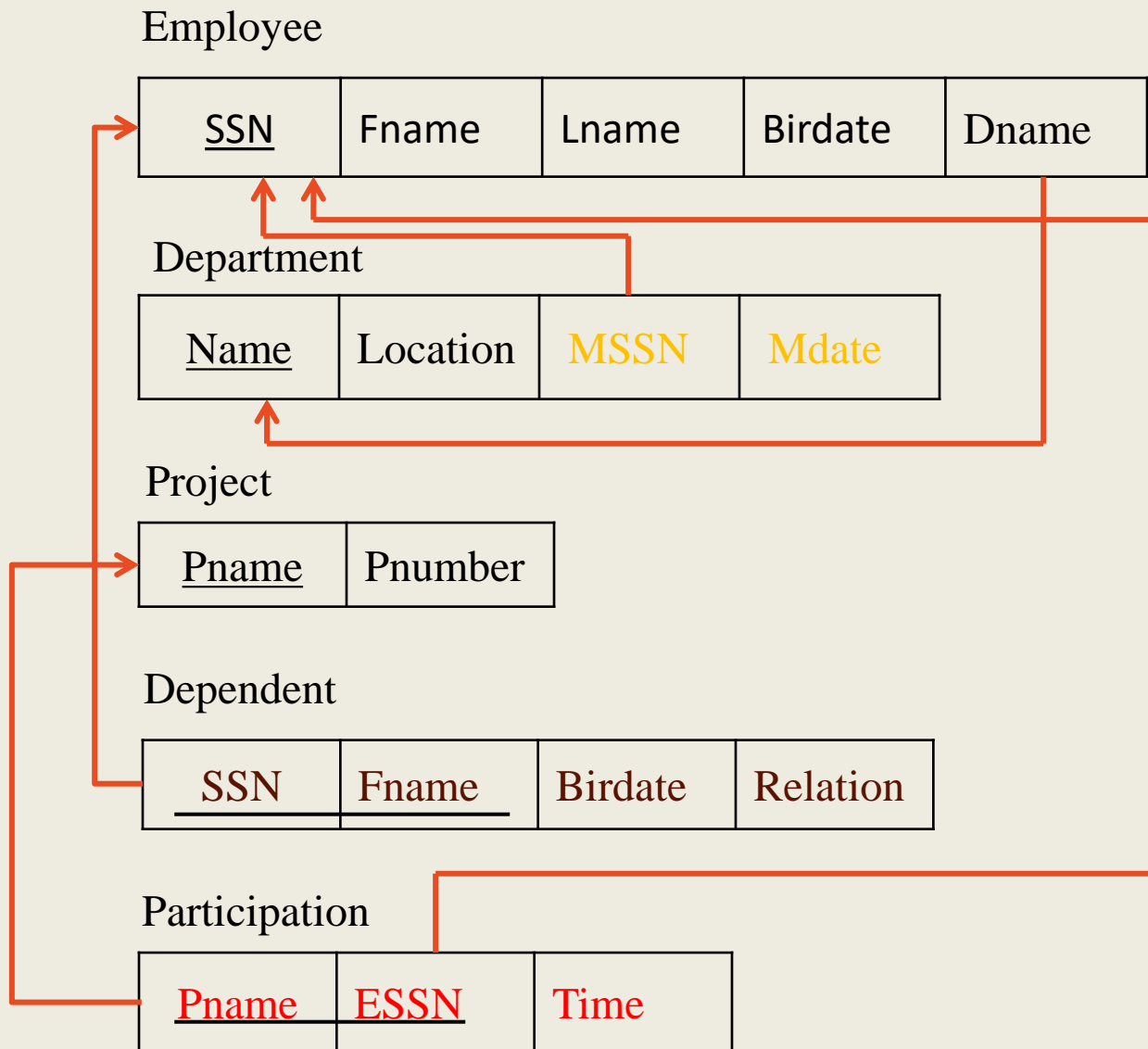
Create a new relation R with

–Attributes :

- Attributes from the key of S as foreign key,
- And attributes from the key of T as foreign key,
- And simple attributes, and simple components of composite attributes of relation B.

Key : All attributes from the key of S and the key of T.





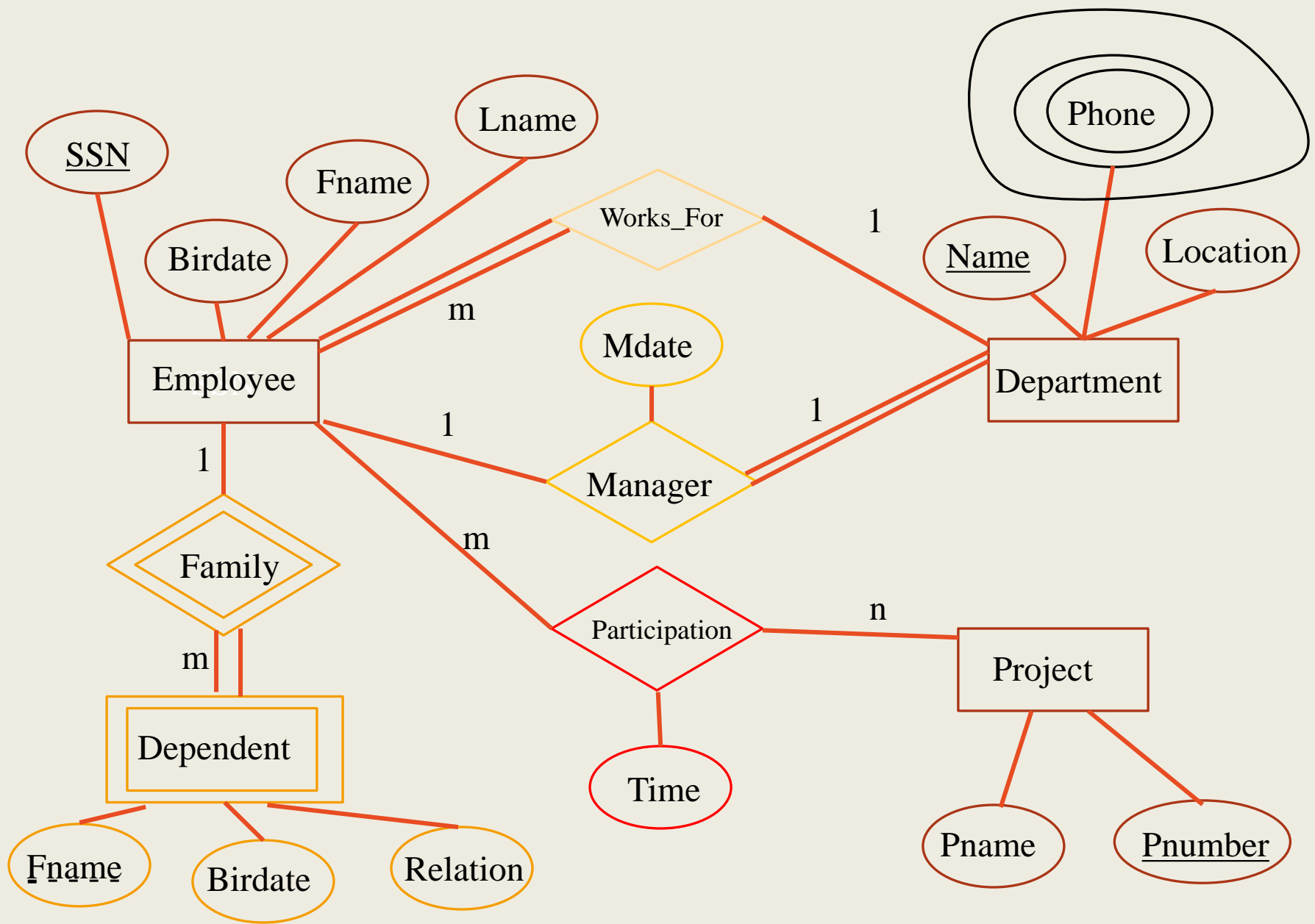
# Mapping Multivalued Attributes

---

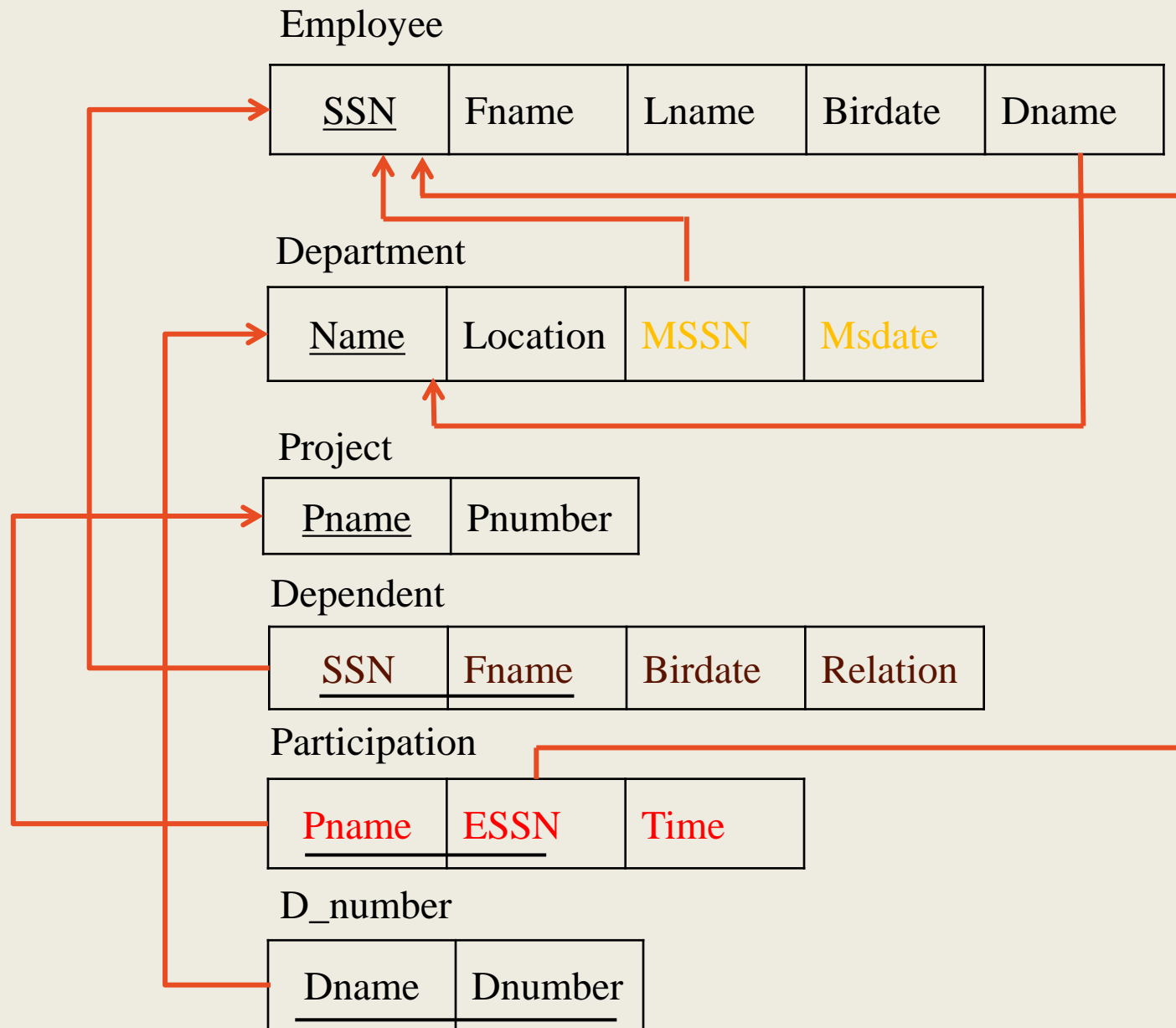
Step 6 : For each ***multivalued attribute*** A, where A is an attribute of E, create a new relation R.

- *If A is a multivalued simple attribute,*
  - Attributes of R = Simple attribute A, and key of E as a foreign key.
- *If A is a multivalued composite attribute,*
  - Attributes of R = All simple components of A, and key of E as a foreign key.

In both cases, the primary key of R is the set of all attributes in R.







# Mapping N-ary Relationship Types

---

Step 7 : For each ***n-ary relationship type*** ( $n > 2$ ), create a new relation with

- Attributes : same for Step 5.
- Key :
  - same for Step 5, see exception below
  - The exception is that if one of the participating entity types has participation ratio 1, its key can be used as a key for the new relation.

*(Advice: binary relationships simpler to model)*

# Summary of Mapping

---

- Map Entities first
  - Strong Entity Types
  - Weak Entity Types
- Map Relationship
  - 1:1 Relationship Types
  - 1:N Relationship Types
  - M:N Relationship Types
  - N-ary Relationship Types
- Mapping
  - Multivalued Attributes
  - Composite Attributes

# Modeling

---

ER Model: To capture semantics

Relational Model: To manipulate data

Mapping from ER-diagram to  
Relational Database Schema  
(a collection of relation schemas)

Several Main Points

- We do things step-by-step.
- When considering how to capture semantics, we do not consider how to map it onto a relational database schema, and how to query data on the database schema.
- When considering mapping from an ER-diagram onto a relational database schema, there exists a systematic way to do it.

# Takeaway

---

## Learning Outcomes

1. An understanding of relational model.
2. Knowing how to convert an ERD to relational model.

Lab 1 (week2) is released

Assignment 1 to be released tonight : Due by Friday March 11 14:59:59

Practice questions to be released tonight