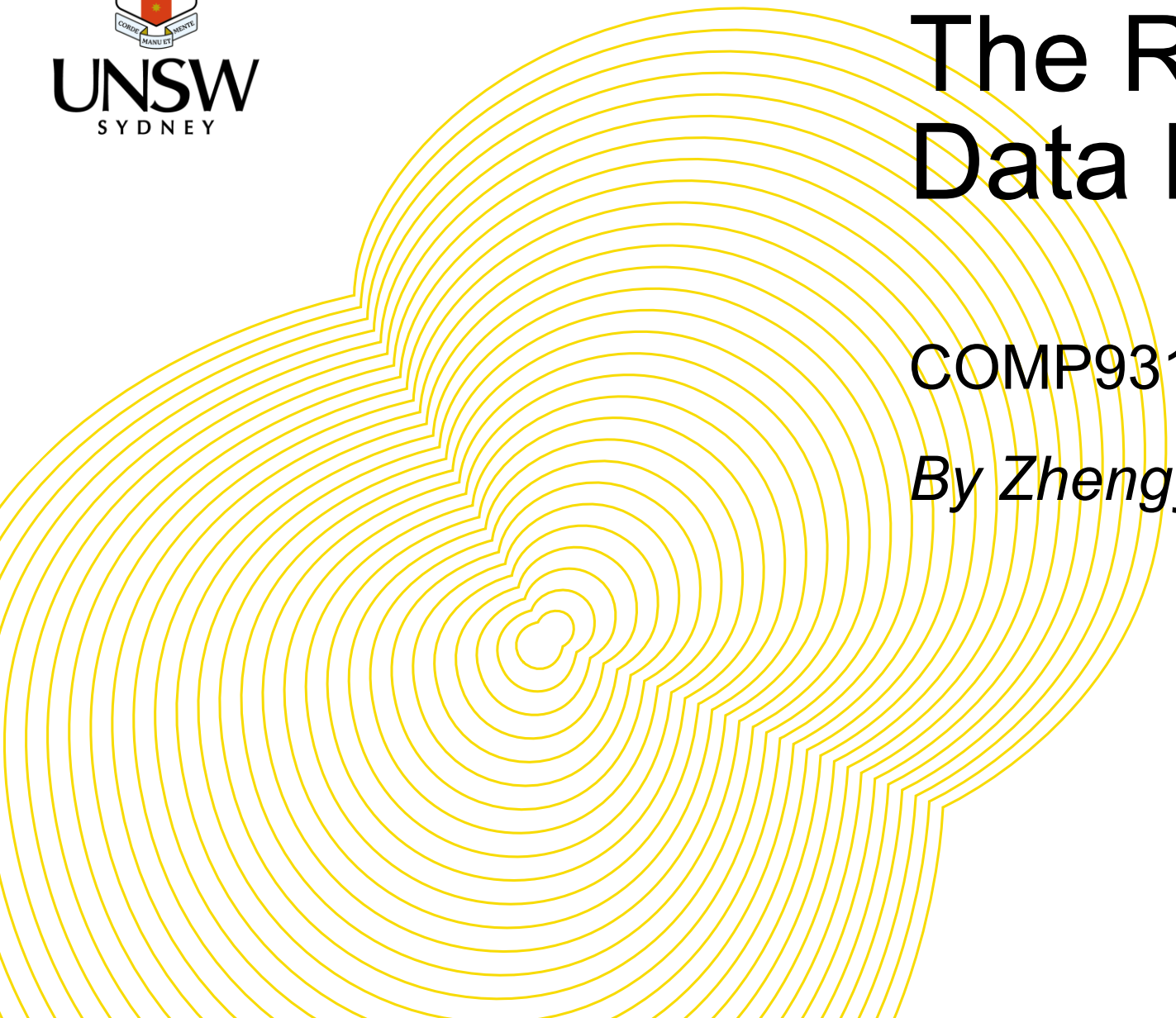


The Relational Data Model

COMP9311 24T2; Week 2.1

By Zhengyi Yang, UNSW

Notice

Lab 01 this week:

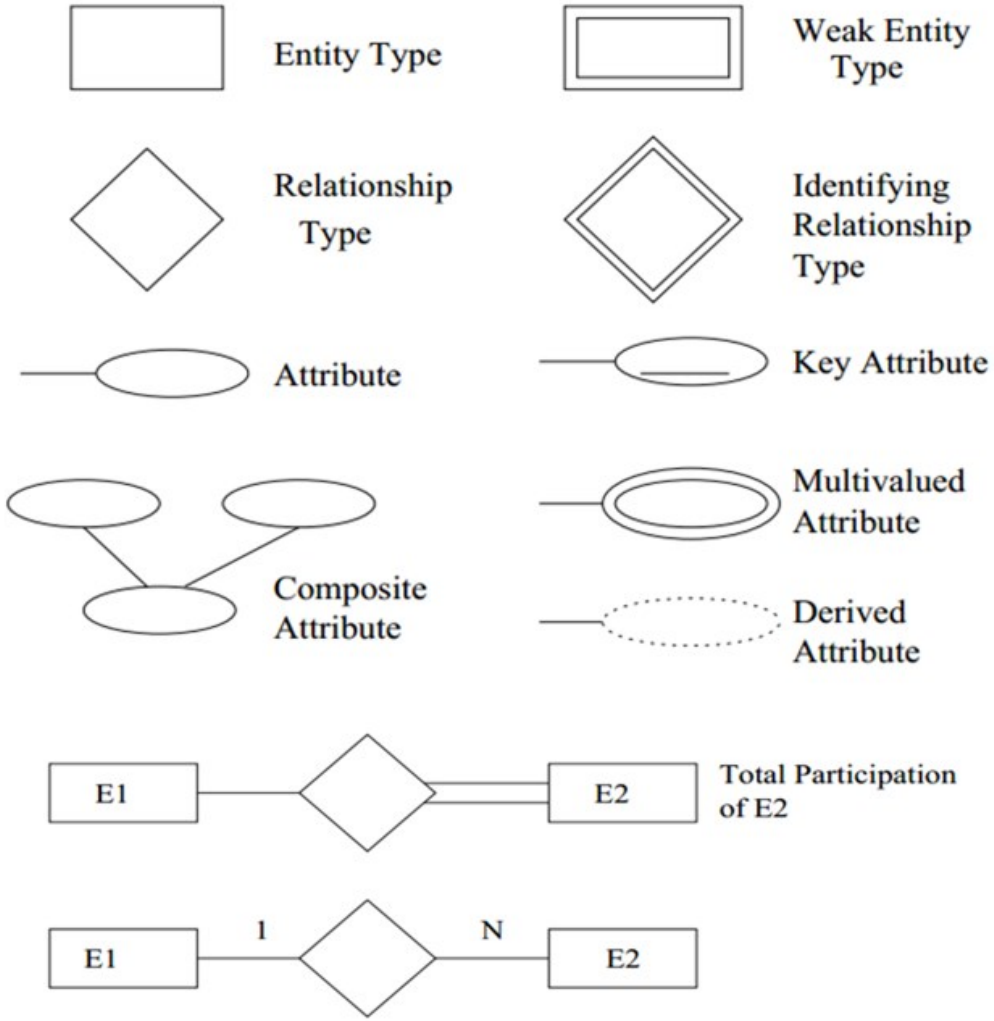
- **Bring Your Own Device**
- Setting up your PostgreSQL server

Recap – Data Modelling

Checklist on ER modeling

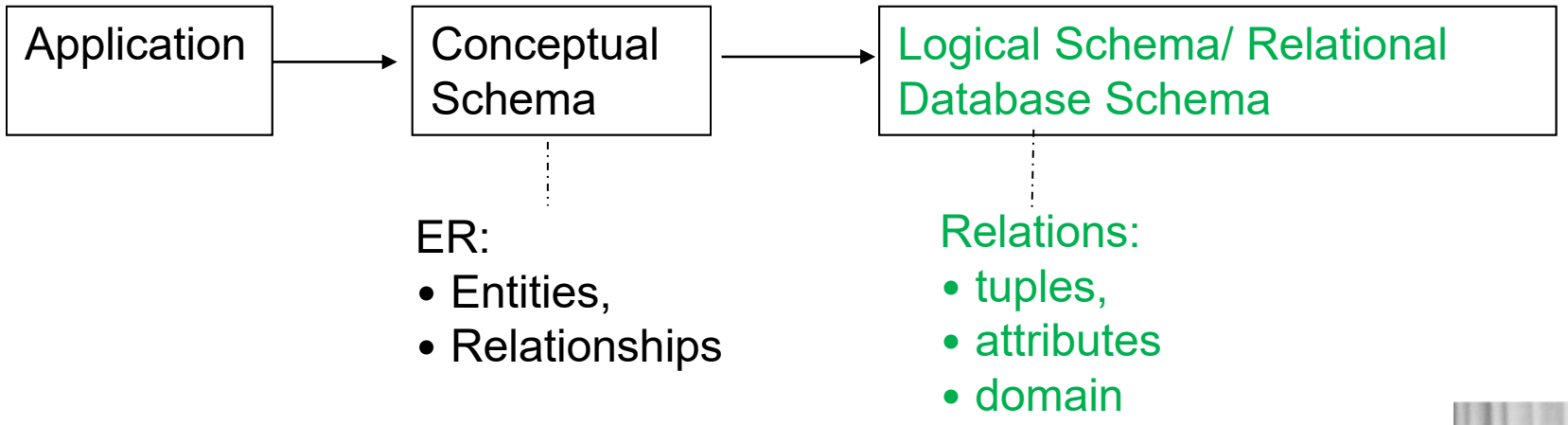
1. Did you model every significant **entity** that has independent instances?
2. Did you model the entity in the correct type? **Strong entity** or **weak entity**?
3. Did you capture all the main **relationships** between entities?
4. Does every relationship have the correct **cardinality**
5. Did you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

Recap – Standard Notation



Introduction

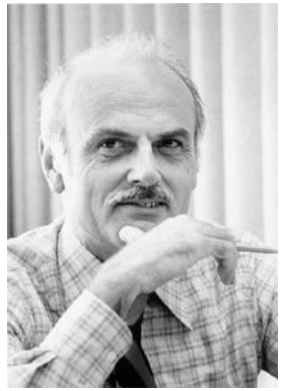
The most popular data model for database systems (see Week1 Monday)



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



Relational Data Model Concepts

The relational data model is the most widely used data model for database systems.

The *relational data model* describes the world as

- a **collection** of inter-connected ***relations***

The goal of the relational model:

- a simple, general data modelling formalism
- which maps easily to file structures (i.e. implementable)

The relational model has **two styles** of terminology:

- mathematical: relation, tuple, attribute, ...
- data-oriented: table, record, field/column, ...

Structures

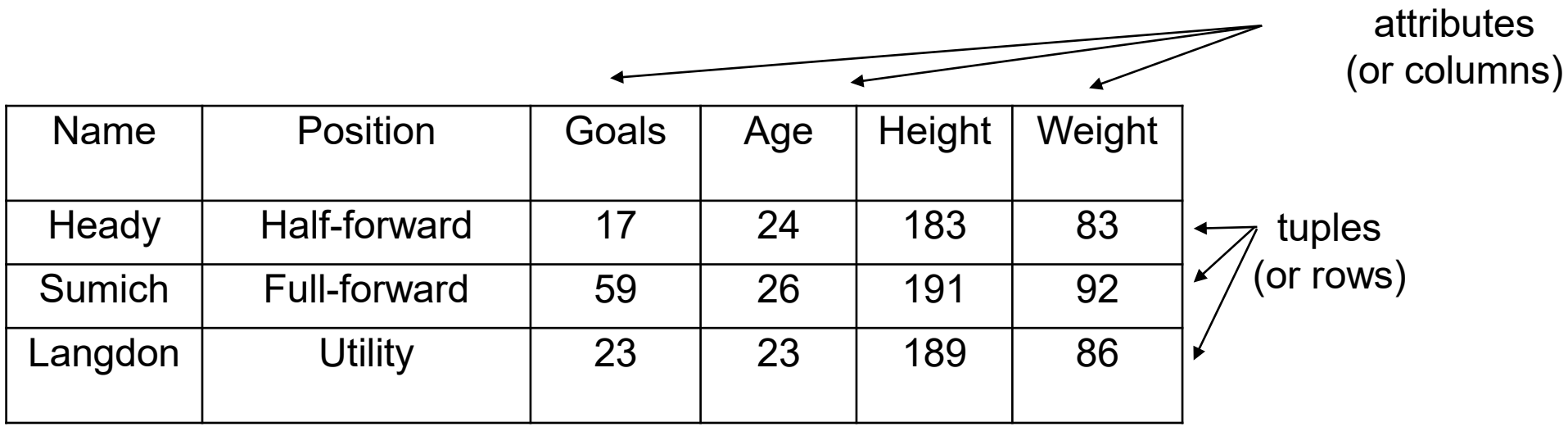
In the relational model, everything is described using **relations**.

A relation can be thought of as a **named table**.

- Each column of the table corresponds to a *named attribute*.
- Each row of the table is called a ***tuple*** of the relation.

The set of allowed values for an attribute is called its *domain*.

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

Relational Data Model

Mathematically,

- a *domain D* is a set of atomic values (having some fixed data type) representing some semantic meaning.
- an *attribute A* is the name of a role played by a *domain*, *dom*(*A*).
- a *relation schema R*, denoted by $R(A_1,A_2, ...,A_n)$, is a set of attributes $R = \{A_1,A_2, ...,A_n\}$.

Composite and multivalued attributes are not allowed!

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) ≠ (2, 1, 3).
- A ***set*** expresses ***membership***.
- Example: we care that 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. The ordering of columns or rows is 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

Question: Ordering within a tuple?

Yes, *but an alternative definition exists that contains no ordering.*

Why Relational Model?

- Very simple model
- Often a good match for the way we think about our data
- Foundations in logic and set theory (will be introduced in later parts of the course)

Keys

Keys are used to identify tuples in a relation.

A ***superkey*** is a set of attributes that uniquely determines a tuple.

A ***candidate key*** is a *minimal* superkey, i.e., none of whose subsets is a superkey.

Example

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 ***candidate key*** (because {Name} is a key).

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

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

A ***primary key*** is a designated candidate key.

In many applications, it is necessary to invent a primary key if there is no natural one - often this would be a non-negative integer.

e.g. Person_ID.

When a relation schema has several candidate keys, choosing a primary key with a single attribute or a small number of attributes is usually better.

Number of Superkeys

{A}: {A} 1

{A,B}: {A},{B},{A,B} 3

{A,B,C}: {A},{B},{C},{A,B},{B,C},{A,C},{A,B,C} 7

(2^n)-1

Relation Referring to Another Relation

How do we store relationships? For example, ENROLLMENT in this case?

STUDENT:

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

RESEARCHER:

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

COURSE:

Department	Degree
Psychology	Ph.D.
Comp.Sci.	Ph.D.
Comp.Sci.	M.Sc.
Psychology	M.Sc.

Store the values of the Primary Key?

STUDENT:

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

RESEARCHER:

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

COURSE:

Department	Degree
Psychology	Ph.D.
Comp.Sci.	Ph.D.
Comp.Sci.	M.Sc.
Psychology	M.Sc.

ENROLMENT:

Enrolment#	Student	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.

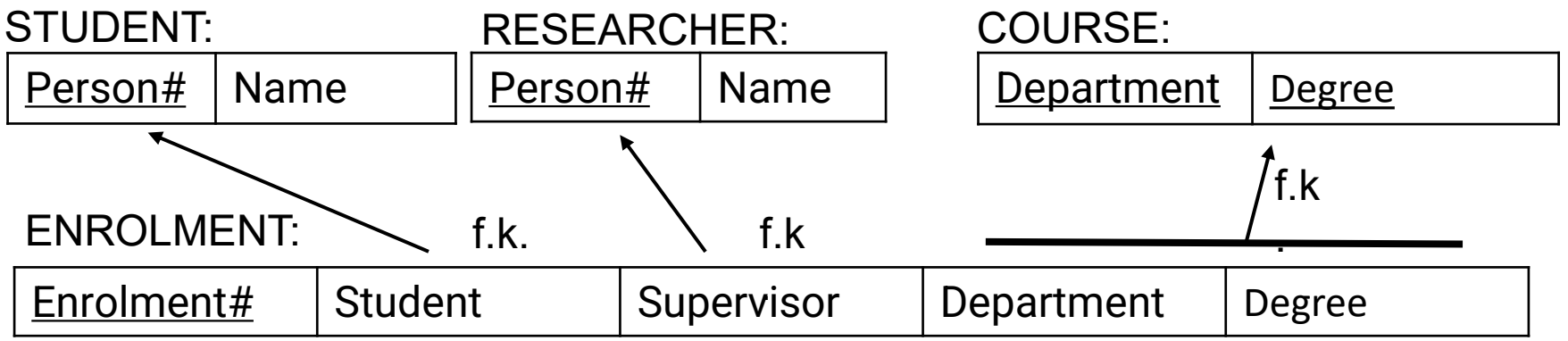
A set of attributes from a relation schema R1 may be a foreign *key*, *FK*, if

- 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.

Referential integrity: The value of *FK* must occur in the other relation or be entirely NULL.

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 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 violate any integrity constraints.

Insertions

Insertions: When inserting, we need to check

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

Insertion: Key constraint violation

STUDENT:

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

RESEARCHER:

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

COURSE:

Department	Degree
Psychology	Ph.D.
Comp.Sci.	Ph.D.
Comp.Sci.	M.Sc.
Psychology	M.Sc.

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

ENROLMENT:

Enrolment#	Student	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.

Insertion: Key constraint violation

STUDENT:

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

RESEARCHER:

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

COURSE:

Department	Degree
Psychology	Ph.D.
Comp.Sci.	Ph.D.
Comp.Sci.	M.Sc.
Psychology	M.Sc.

ENROLMENT:

Enrolment#	Student	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? No. Violates a key constraint.

Action? Reject or allow the user to correct.

Insertion: Key constraint violation

STUDENT:

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

RESEARCHER:

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

COURSE:

Department	Degree
Psychology	Ph.D.
Comp.Sci.	Ph.D.
Comp.Sci.	M.Sc.
Psychology	M.Sc.

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

ENROLMENT:

Enrolment#	Student	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.

Insertion: Key constraint violation

STUDENT:

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

RESEARCHER:

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

COURSE:

Department	Degree
Psychology	Ph.D.
Comp.Sci.	Ph.D.
Comp.Sci.	M.Sc.
Psychology	M.Sc.

ENROLMENT:

Enrolment#	Student	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? No. "Degree" is a part of the primary key, it cannot be NULL.

Action: Reject or allow the user to correct.

Insertion: Referential integrity violation

STUDENT:

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

RESEARCHER:

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

COURSE:

Department	Degree
Psychology	Ph.D.
Comp.Sci.	Ph.D.
Comp.Sci.	M.Sc.
Psychology	M.Sc.

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

ENROLMENT:

Enrolment#	Student	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.

Insertion: Referential integrity violation

STUDENT:

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

RESEARCHER:

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

COURSE:

Department	Degree
Psychology	Ph.D.
Comp.Sci.	Ph.D.
Comp.Sci.	M.Sc.
Psychology	M.Sc.

ENROLMENT:

Enrolment#	Student	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? No. Violates a referential integrity constraint (There is no person 6).

Action: Reject, correct or accept after insertion of person number

Deletion

Deletions: When deleting, we need to check **referential integrity** – check whether the primary key occurs 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.

Allowed? No. Violates the referential integrity.

Action: Reject, correct or modify the ENROLMENT tuple by the actions on the next slides.

Deletion: Constraint Checks

We sometimes need to delete tuples from relations, and the record may be referenced in other relations.

What can we do?

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

Modifications

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

If the modified attribute is the 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 a foreign key

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

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

Relational database definition

- A *relational database schema*, is a set of relation schema $\{R_1, \ldots, R_m\}$ and a set of integrity constraints.

- A relational database instance is a set of relation instances $\{r_1, \ldots, r_m\}$ such that each r_i is an instance of R_i, and the integrity constraints are satisfied.

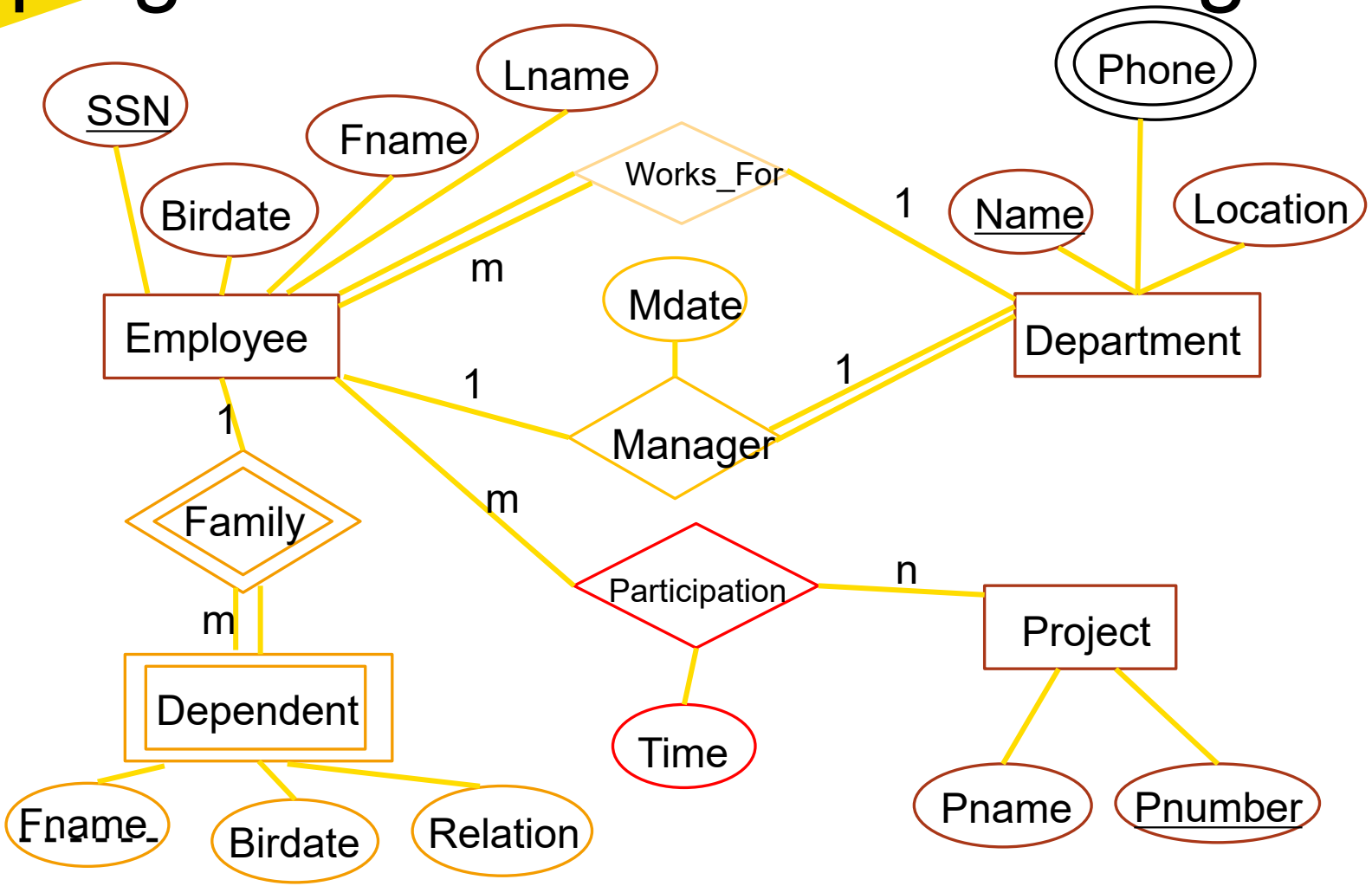
ER to Relational Data Model Mapping

One technique for database design is to first design a conceptual schema using a high-level data model, and then map it to a conceptual schema in the DBMS data model for the chosen DBMS.

Here, we look at a way to do this mapping from the ER to the relational data model.

It involves the following **7 steps**.

Mapping ER to Relational: Guiding Example

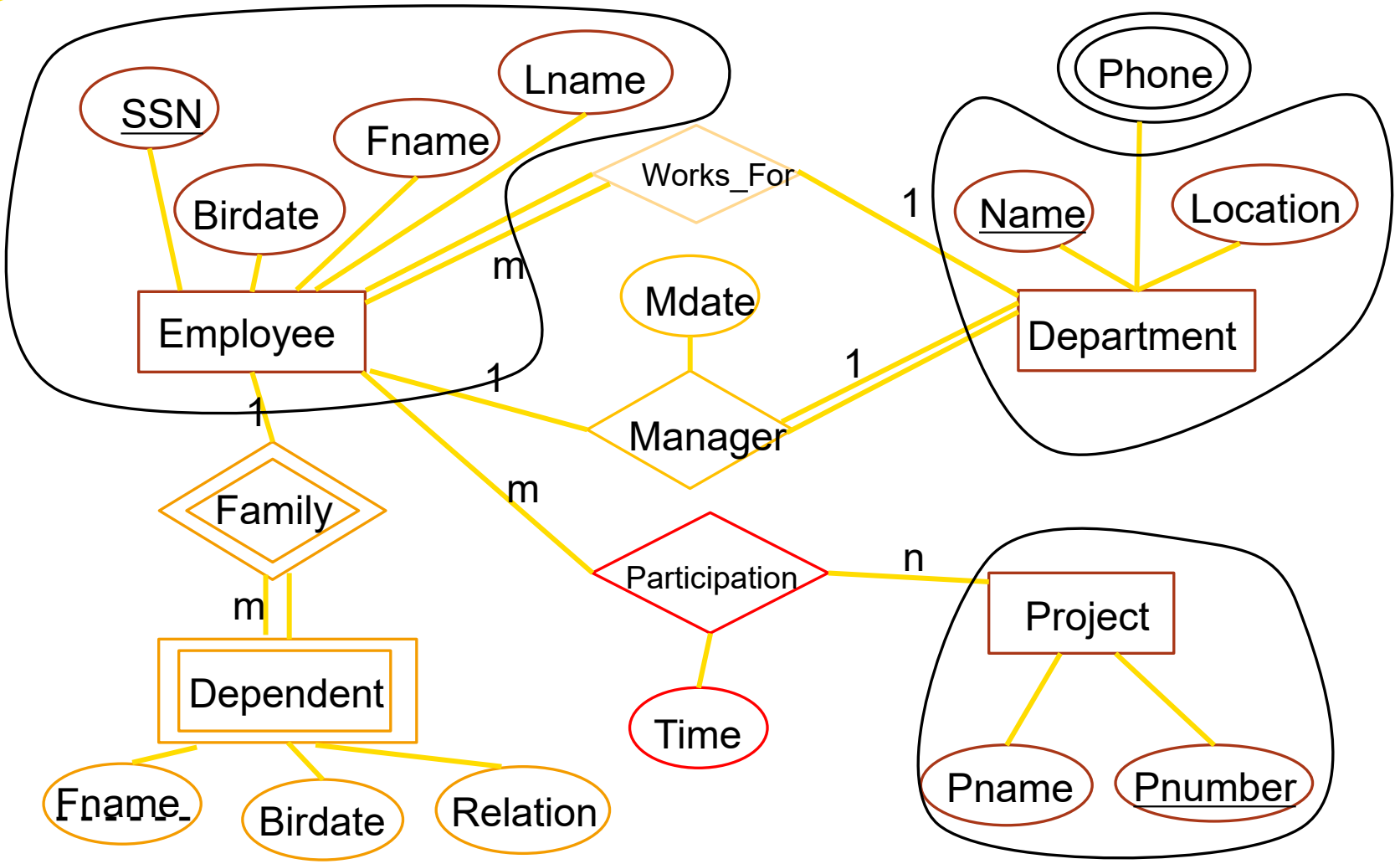


Mapping Strong Entity Types

Step 1: For each ***strong 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



Mapping Strong Entity Types

Employee

SSN	Fname	Lname	Birdate

Department

Name	Location

Project

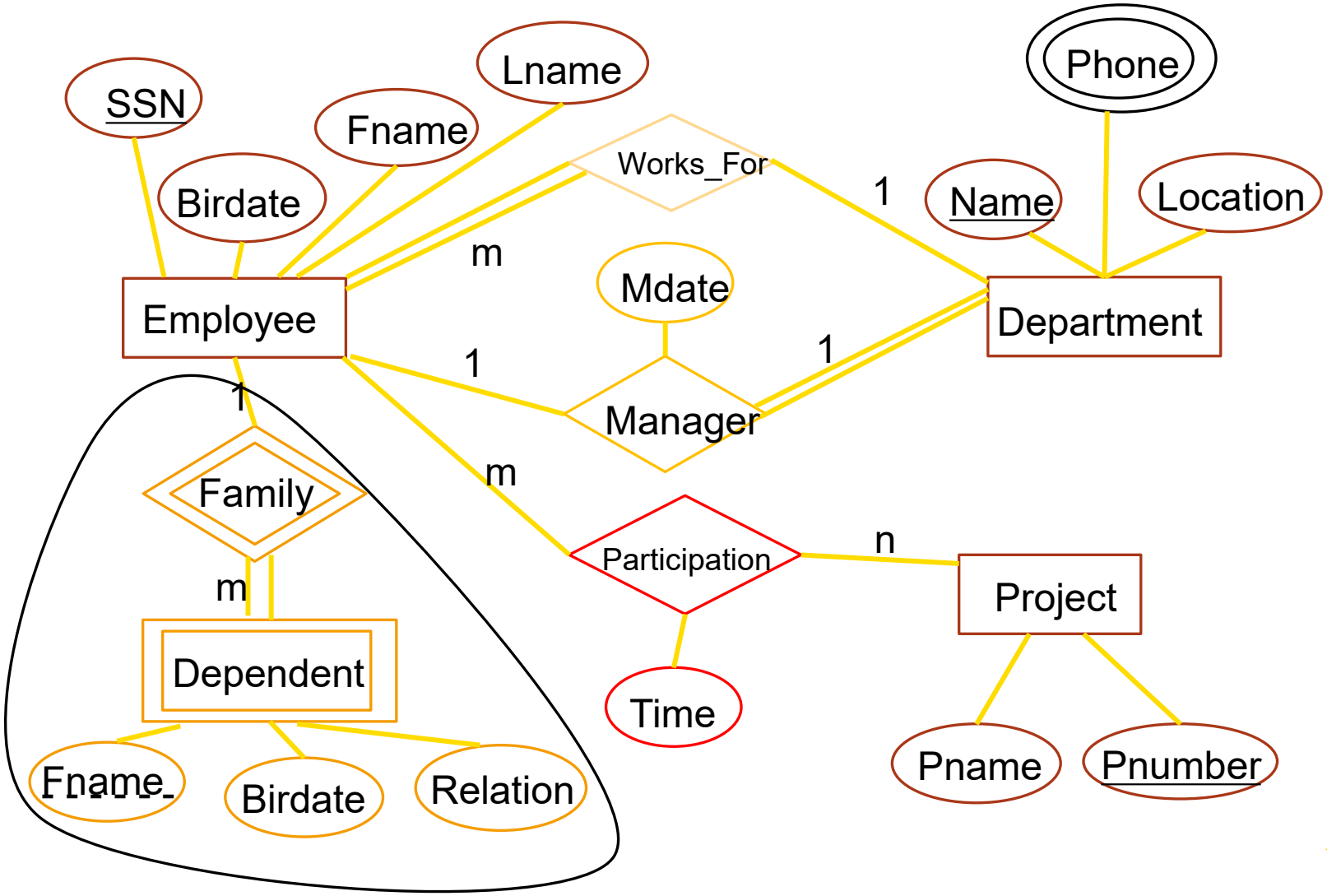
Pnumber	Pname

Mapping Weak Entity Types

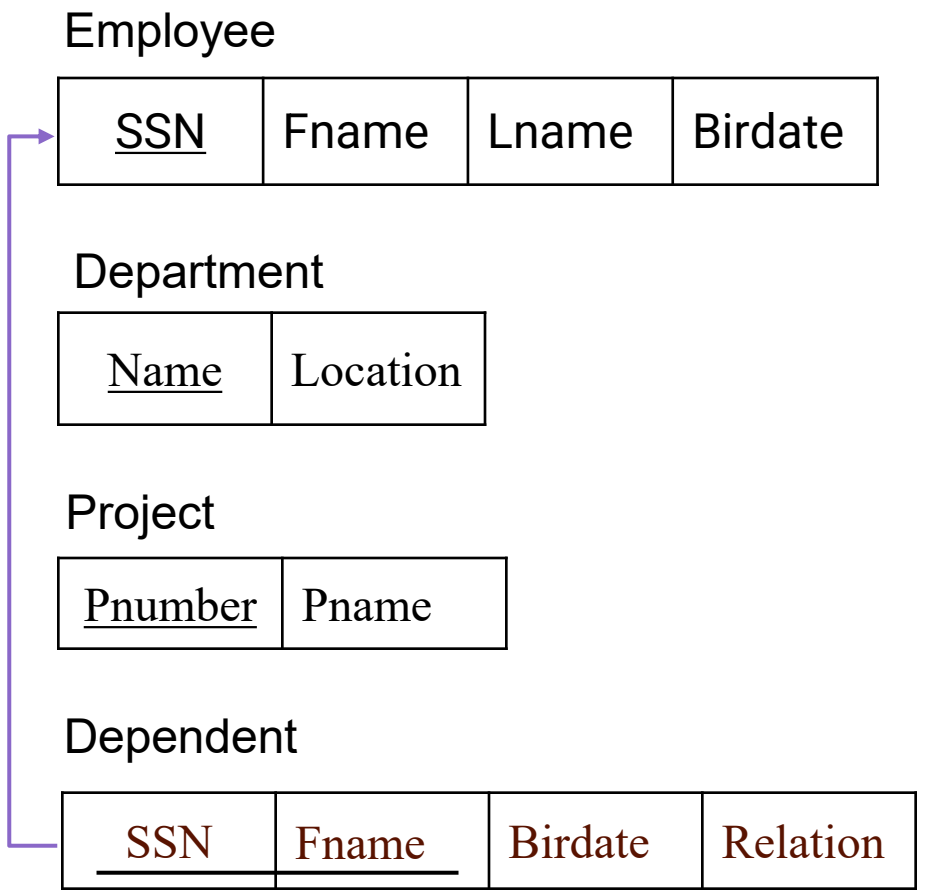
Step 2 : For each ***weak entity type*** W with the owner entity type E, create a new relation R with

- 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 Weak Entity Types



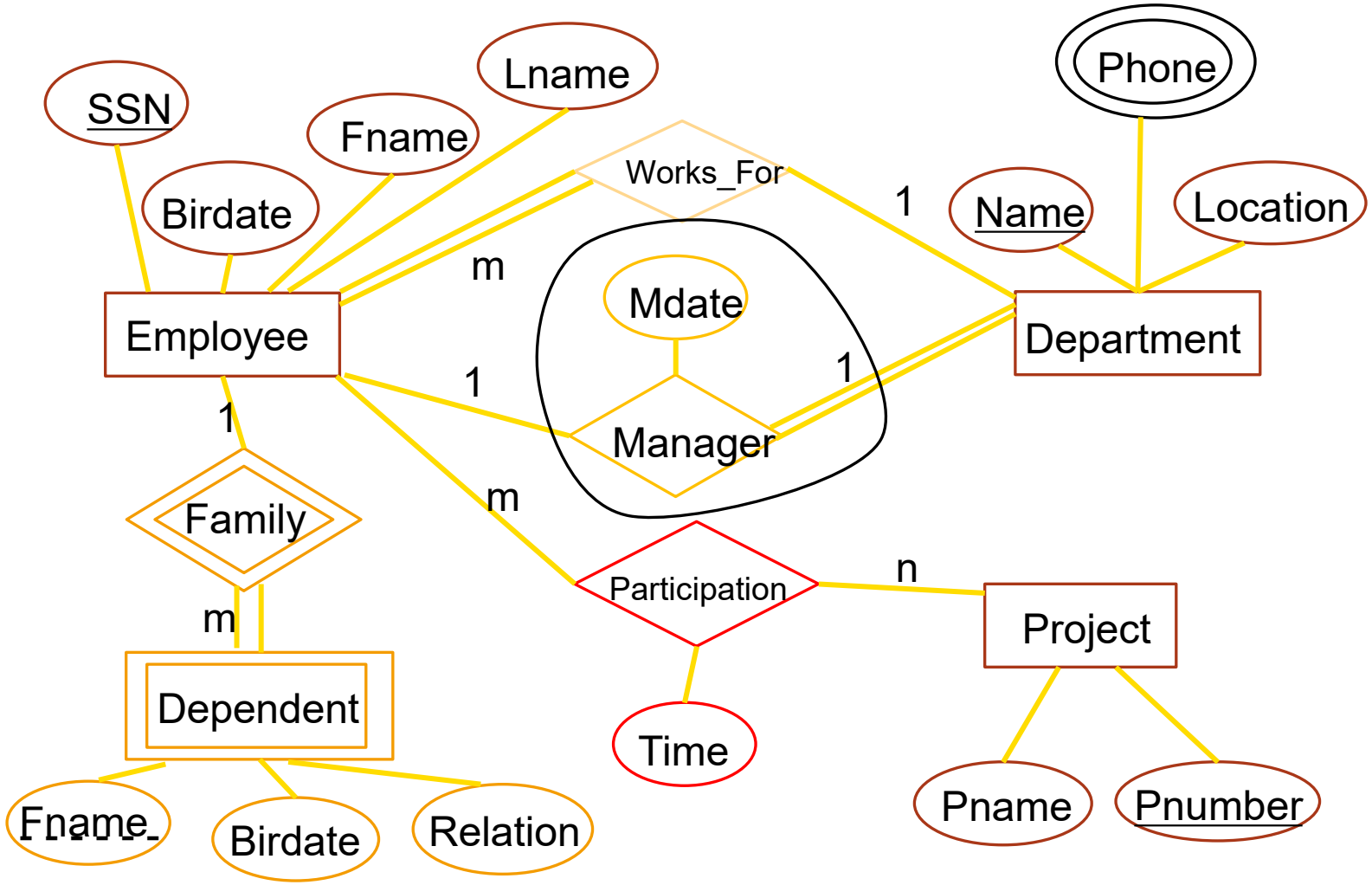
Mapping 1:1 Relationship Types

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.

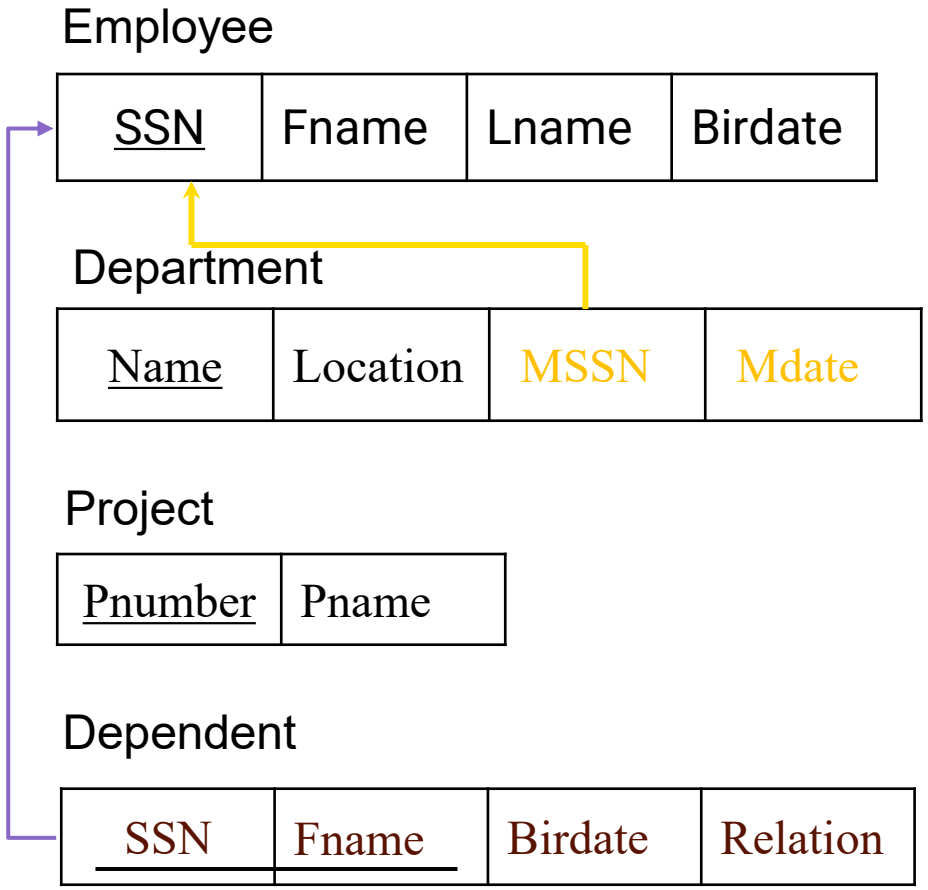
- Choose one of S and T (let S be the one that participates totally if there is one).
- Add attributes from the primary key of T to S as a foreign key.
- Add all simple attributes (and simple components of composite attributes) of B as attributes of S.

(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****).*

Relationship Types



Mapping 1:1 Relationship Types



Mapping 1:N Relationship Types

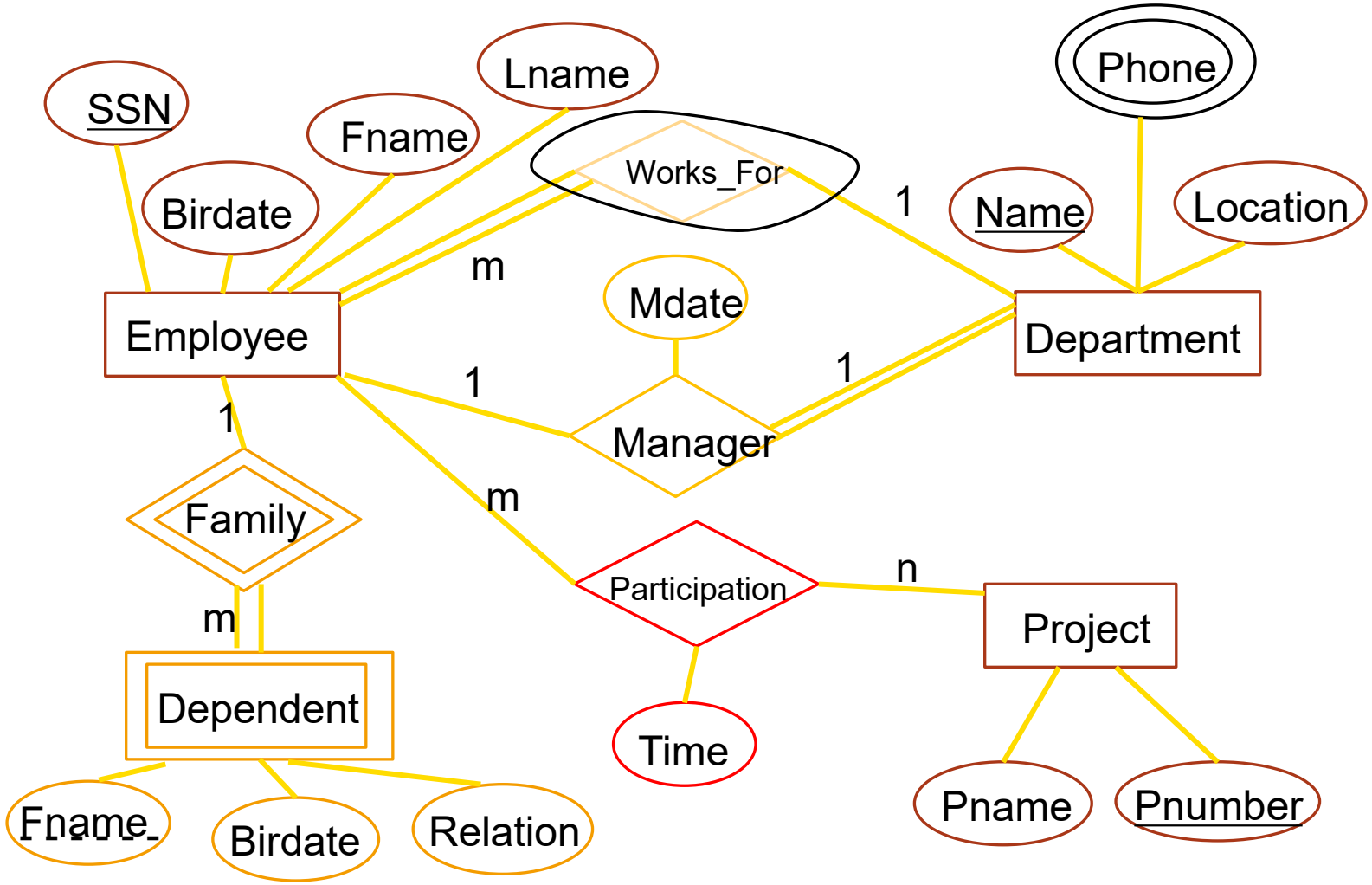
Step 4 : For each ***1:N relationship type*** B. Let E and F be the participating entity types. Let S and T be the corresponding relations. Let E be the entity on the 1 side and F on the N side.

Add to the relation belonging to entity T,

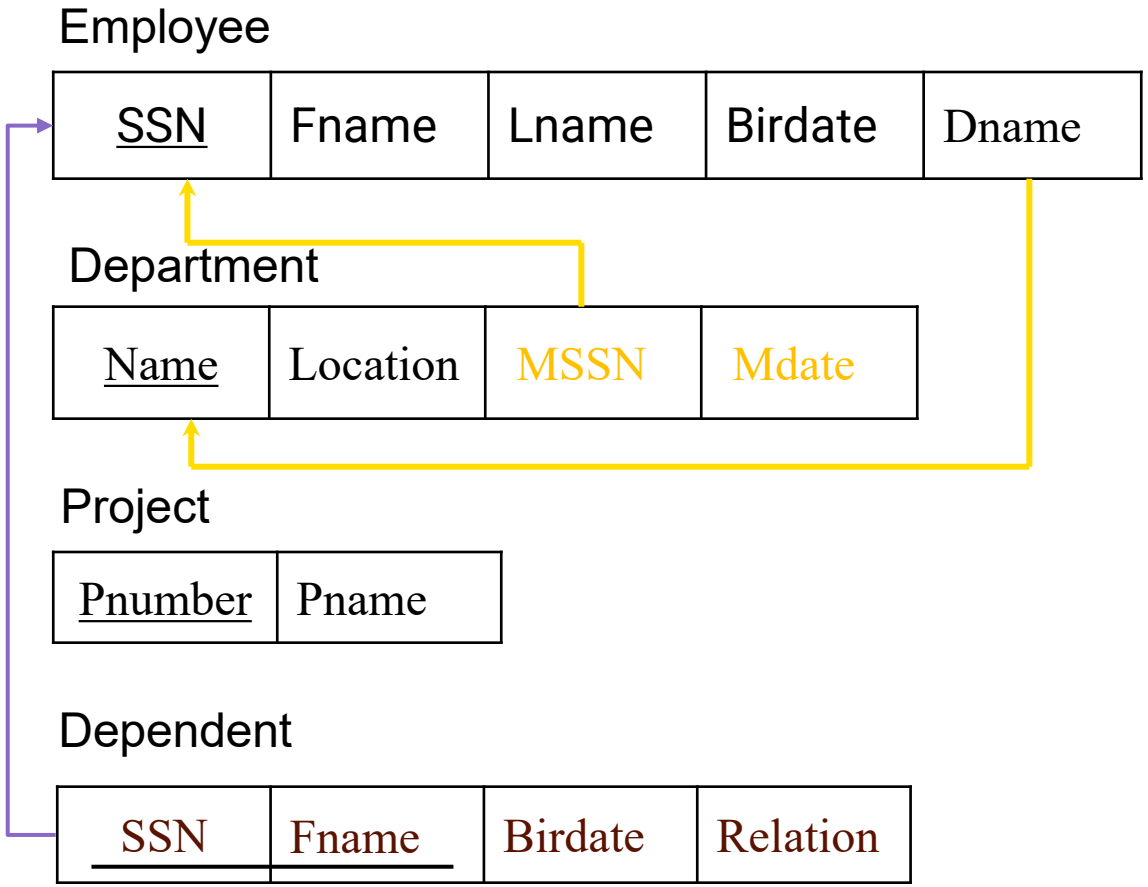
- the attributes from the primary key of S as a foreign key.
- any simple attributes (or simple components of composite attributes) from relationship B.

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

Mapping 1:N Relationship Types



Mapping 1:N Relationship Types



Mapping M:N Relationship Types

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

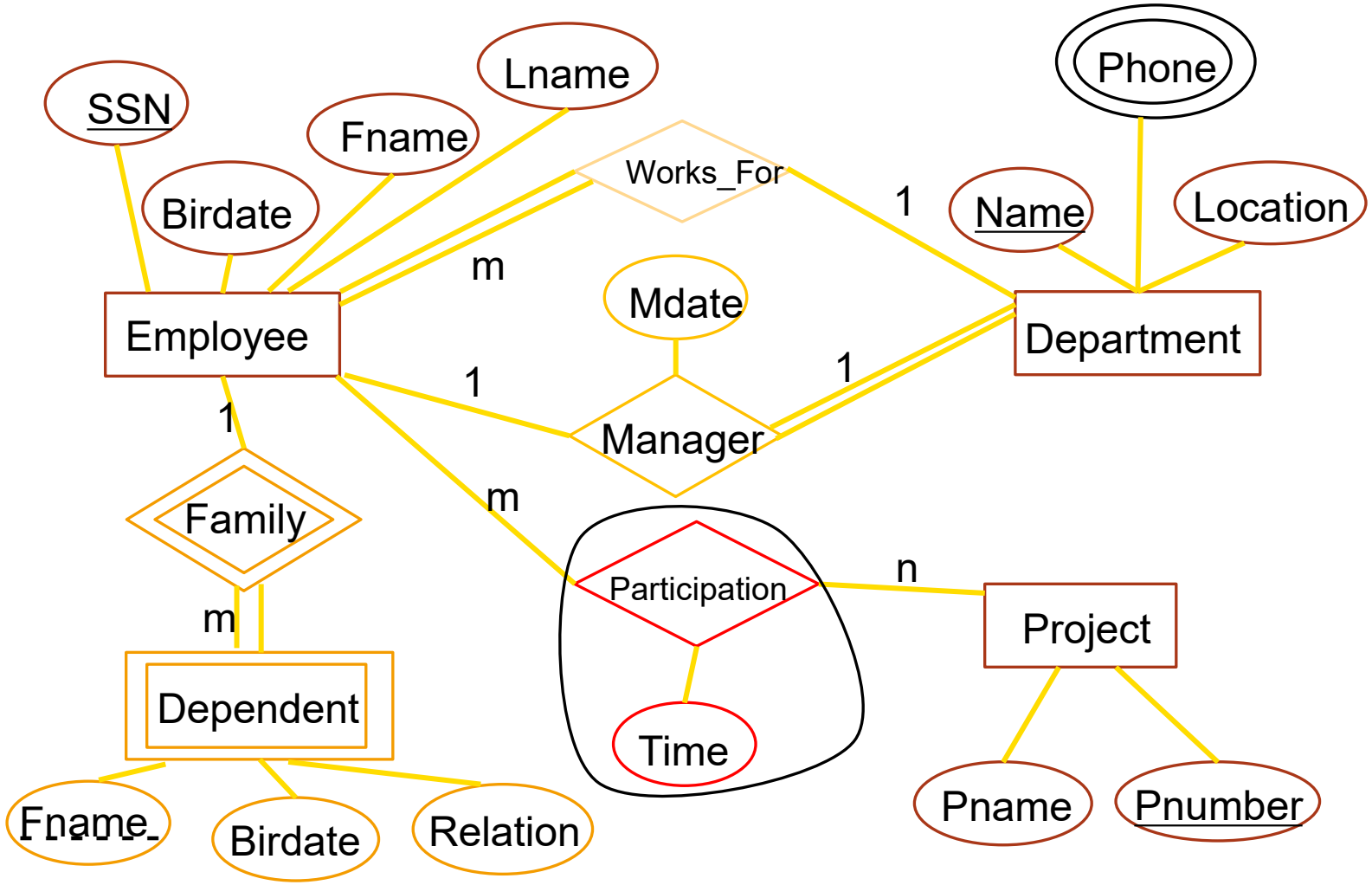
Create a new relation R (*cross-reference*) with

Attributes :

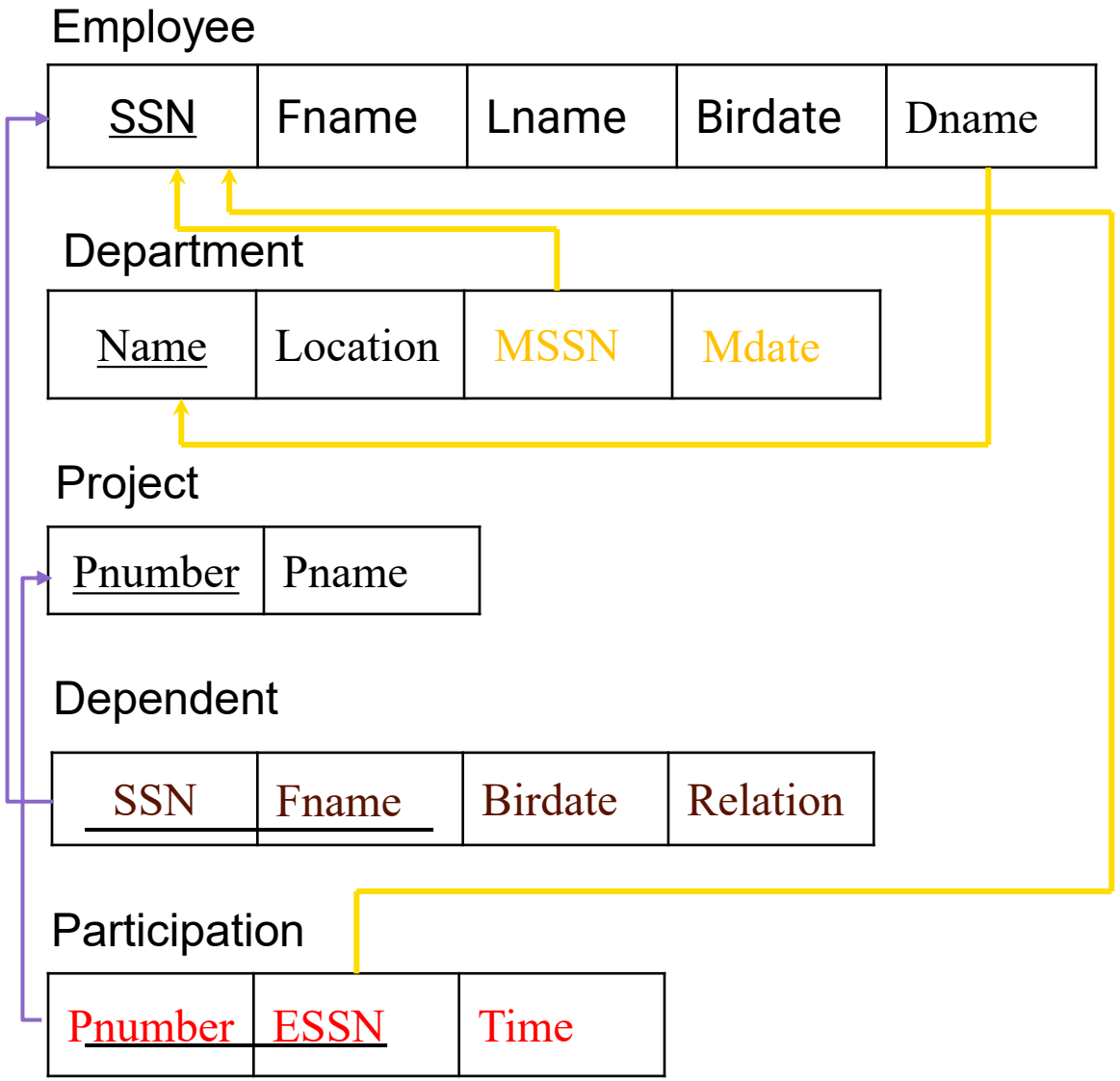
- Attributes from the key of S as a foreign key,
- Attributes from the key of T as a foreign key,
- Simple attributes and simple components of composite attributes of relation B.

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

Mapping M:N Relationship Types



Mapping M:N Relationship Types



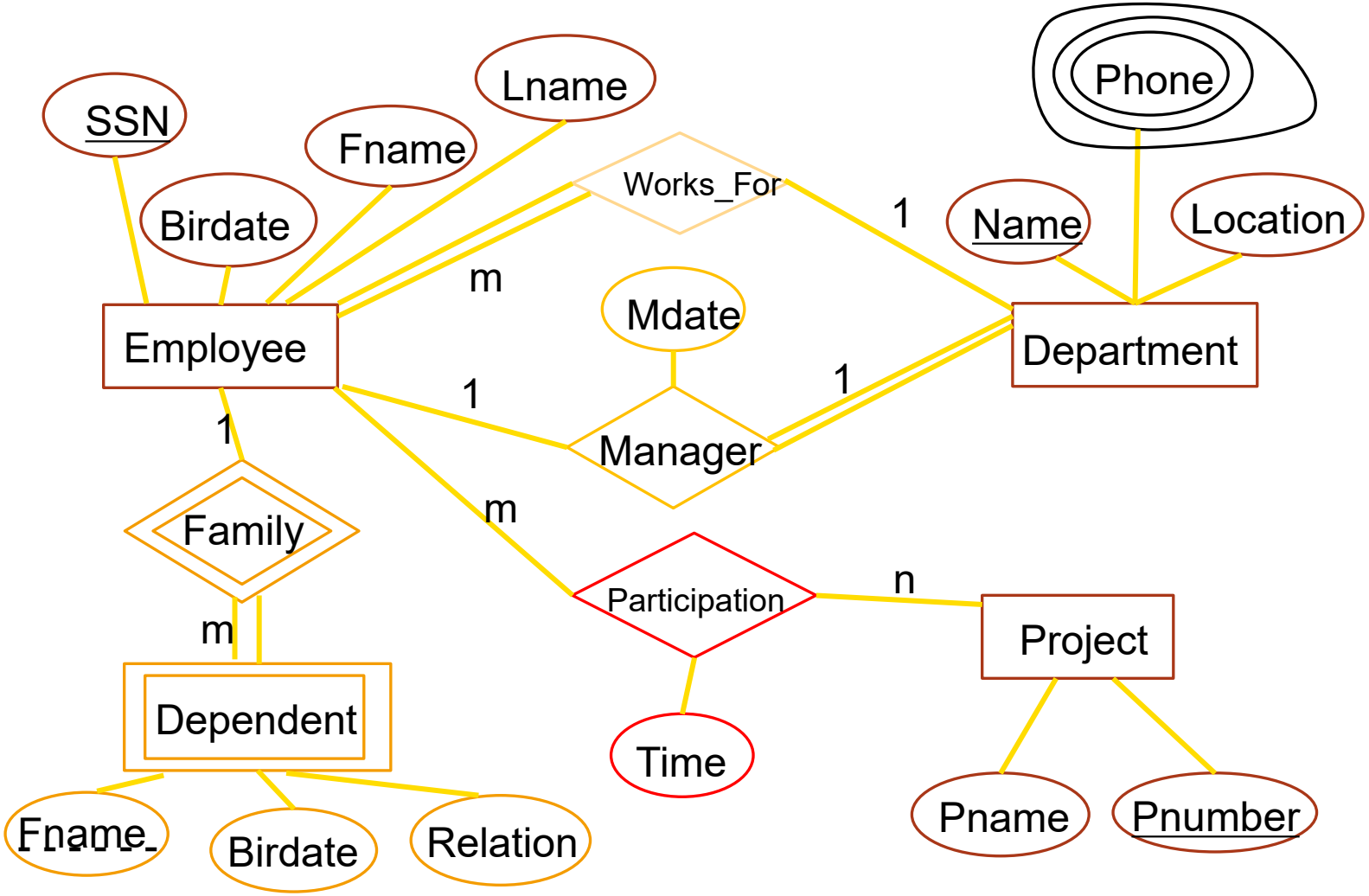
Mapping Multivalued Attributes

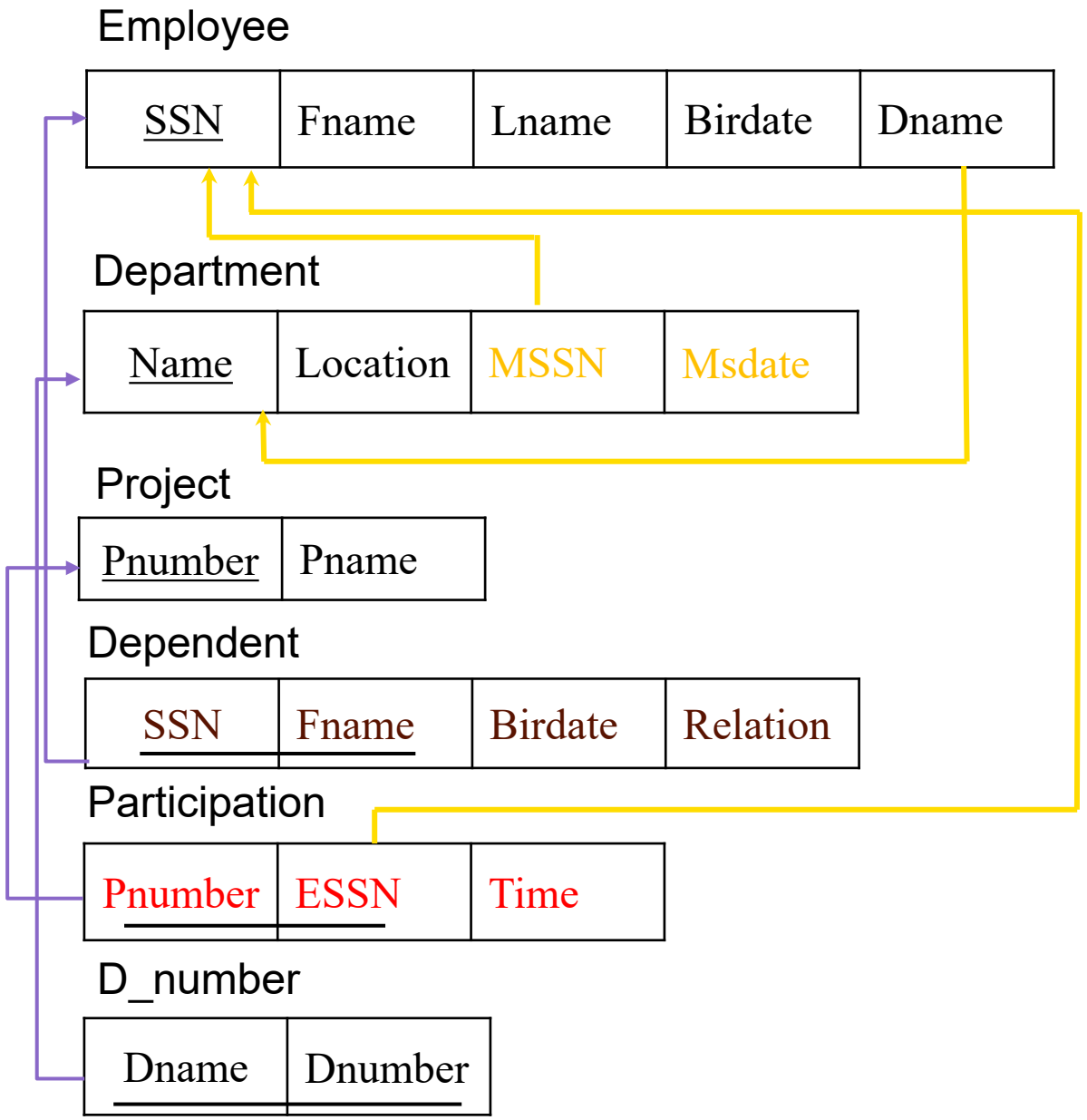
Step 6: For each ***multivalued attribute*** A, where A is an attribute of E, create a new relation R.

- *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 Multivalued Attributes




Employee
SSN
Fname
Lname
Birdate
Dname
Department
Name
Location
MSSN
Msdate
Project
Pnumber
Pname
Dependent
SSN
Fname
Birdate
Relation
Participation
Pnumber
ESSN
Time
D_number
Dname
Dnumber

Mapping N-ary Relationship Types

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

- Attributes: same as Step 5.
- Key: same as Step 5

(Advice: binary relationships are simpler to model.)

Summary of Mapping

- Map Entities first
  - Strong Entity Types (Step 1)
  - Weak Entity Types (Step 2)

- Map Relationship
  - 1:1 Relationship Types (Step 3)
  - 1:N Relationship Types (Step 4)
  - M:N Relationship Types (Step 5)
  - N-ary Relationship Types (Step 7)

- Mapping
  - Multivalued Attributes (Step 6)

ER vs Relational Model

ER MODEL	RELATIONAL MODEL
Entity Type	*Entity* relation
1:1 or 1:N relationship type	Foreign key (or *relationship* relation)
M:N relationship type	*Relationship* relation and *two* foreign key
n-ary relationship type	*Relationship* relation and *n* foreign key
Simple Attribute	Attribute
Composite Attribute	Set of simple component attributes
Multivalued Attribute	Relation and foreign key

Takeaway

Learning Outcomes

1. An understanding of relational model.

2. Knowing how to convert an ERD to relational model.