

# **The Relational Model**

# History

- The relational model was proposed by **Codd** in 1970. (At that time most databases were based on the hierarchical model or the network model)
- Prototype relational DBMSs were developed in IBM and UC-Berkeley by the mid-1970s.
- The relational model attracted a lot of attention at that time because it is **simple and elegant**.
- Today, the relational model is by far the dominant data model and the foundation for the leading DBMS.

# Introduction

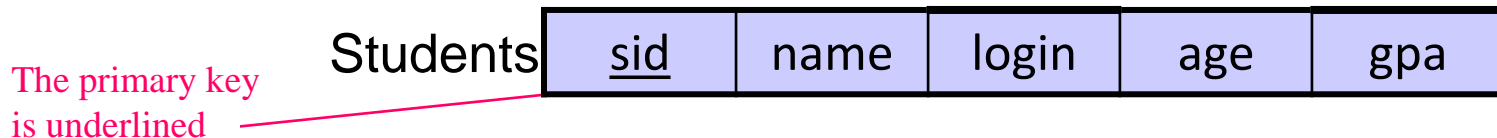
- The main construct for representing data in the relational model is a set of **relations**.
- A relation consists of **relation schema** and **relation instance**.
- **Relation instance**
  - **i.e.** simply a table with rows and columns

- Relation schema

- Describes the column heads for the table.
- Specifies the relation's name, the name of each field, and the **domain** of each field.
  - A domain is referred to by the **domain name** and has a set of associated **values**.

Example:

Students(sid: string, *name*: string, *login*: string, *age*: integer, *gpa*: real)



Students	<u>sid</u>	name	login	age	gpa
----------	------------	------	-------	-----	-----

- An example of the corresponding relational instance is as follows:

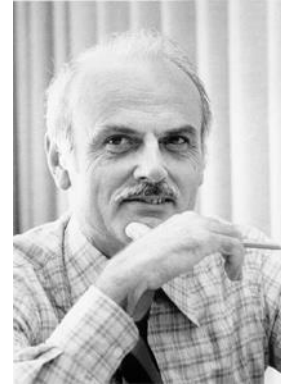
sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith#@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

- The relation model requires that no two rows are identical.  
(In practice, some commercial systems may allow tables to have duplicate rows)
- Each relation is defined to be a set of unique tuples or rows.
- The order in which the rows are listed is not important.
- The order of the fields is not important too.

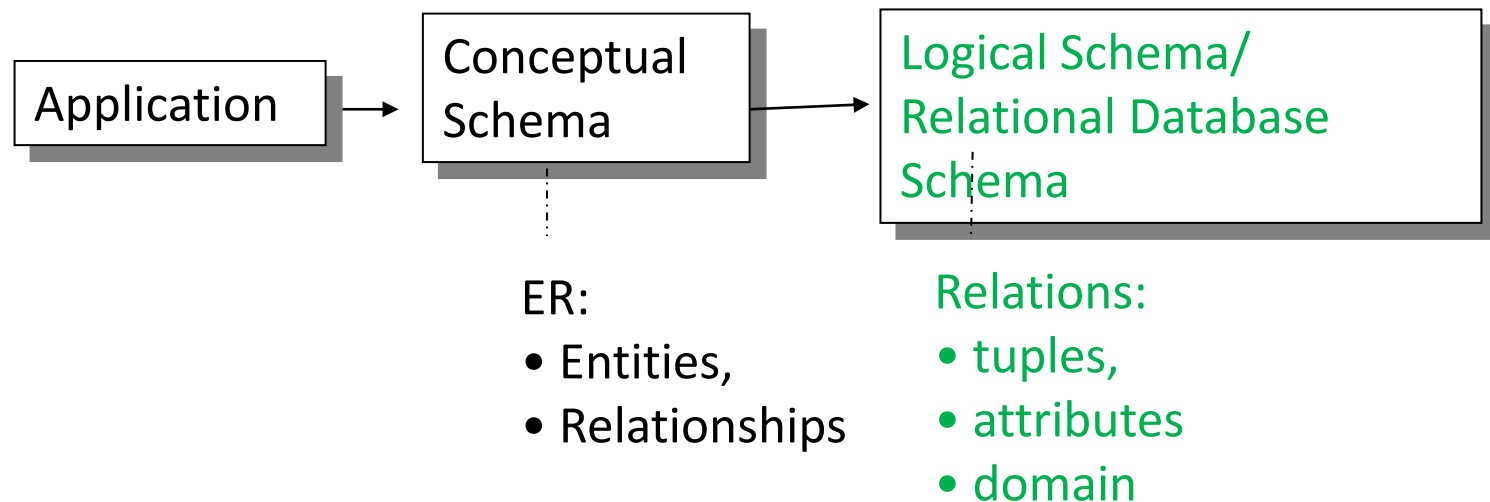
# Wk1 Content – Data Modelling

- 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 CSCI3170) did you use the our notation?

# Introduction



- Most popular data model for database systems



- 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

- 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**.

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

# 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**
- E.g., 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

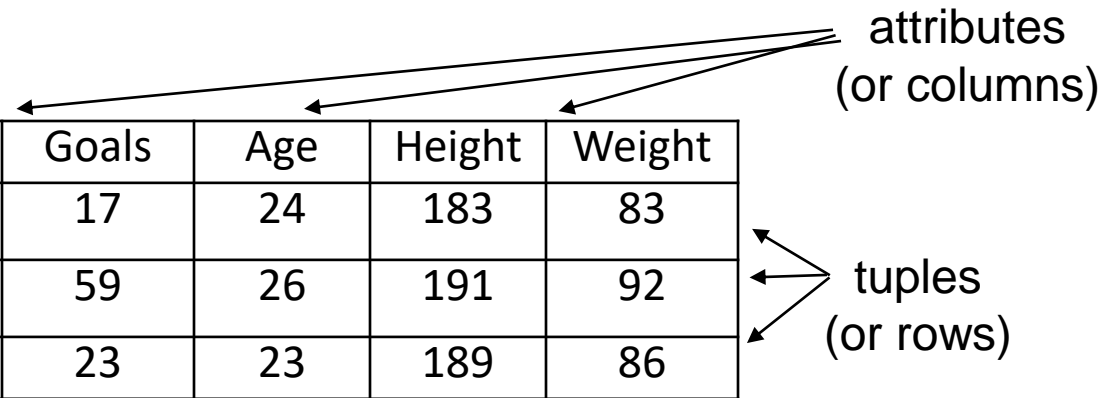
$(a_1, a_2, \dots, a_n)$  where each  $a_i \in D_i$

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

# 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:

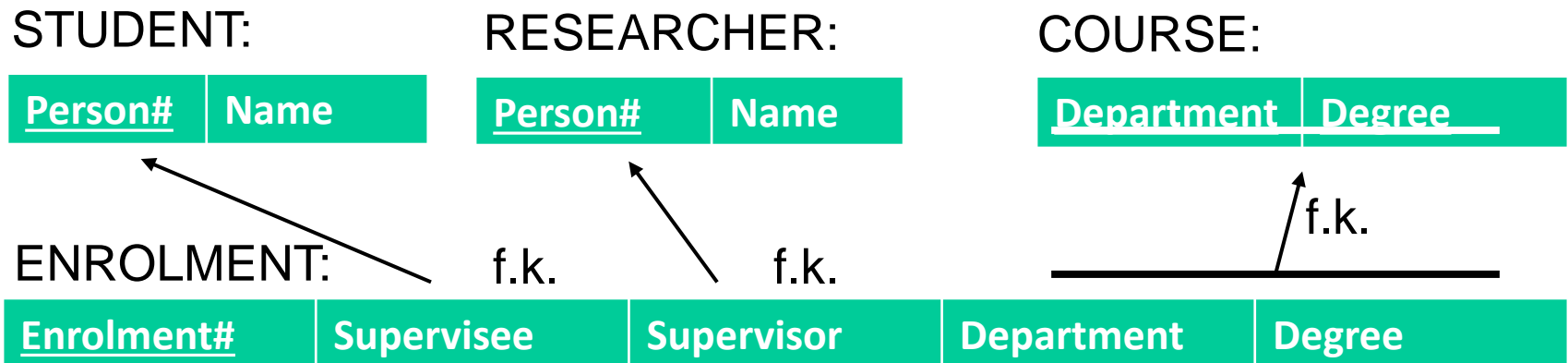
<u>Enrolment#</u>	Supervisee	Supervisor	Department	Degree
1	1	2	Psychology	Ph.D.
2	3	1	Comp.Sci.	Ph.D.
3	4	1	Comp.Sci.	M.Sc.
4	5	1	Comp.Sci.	M.Sc.

# 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.
- *If you're ahead, think of how this is 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

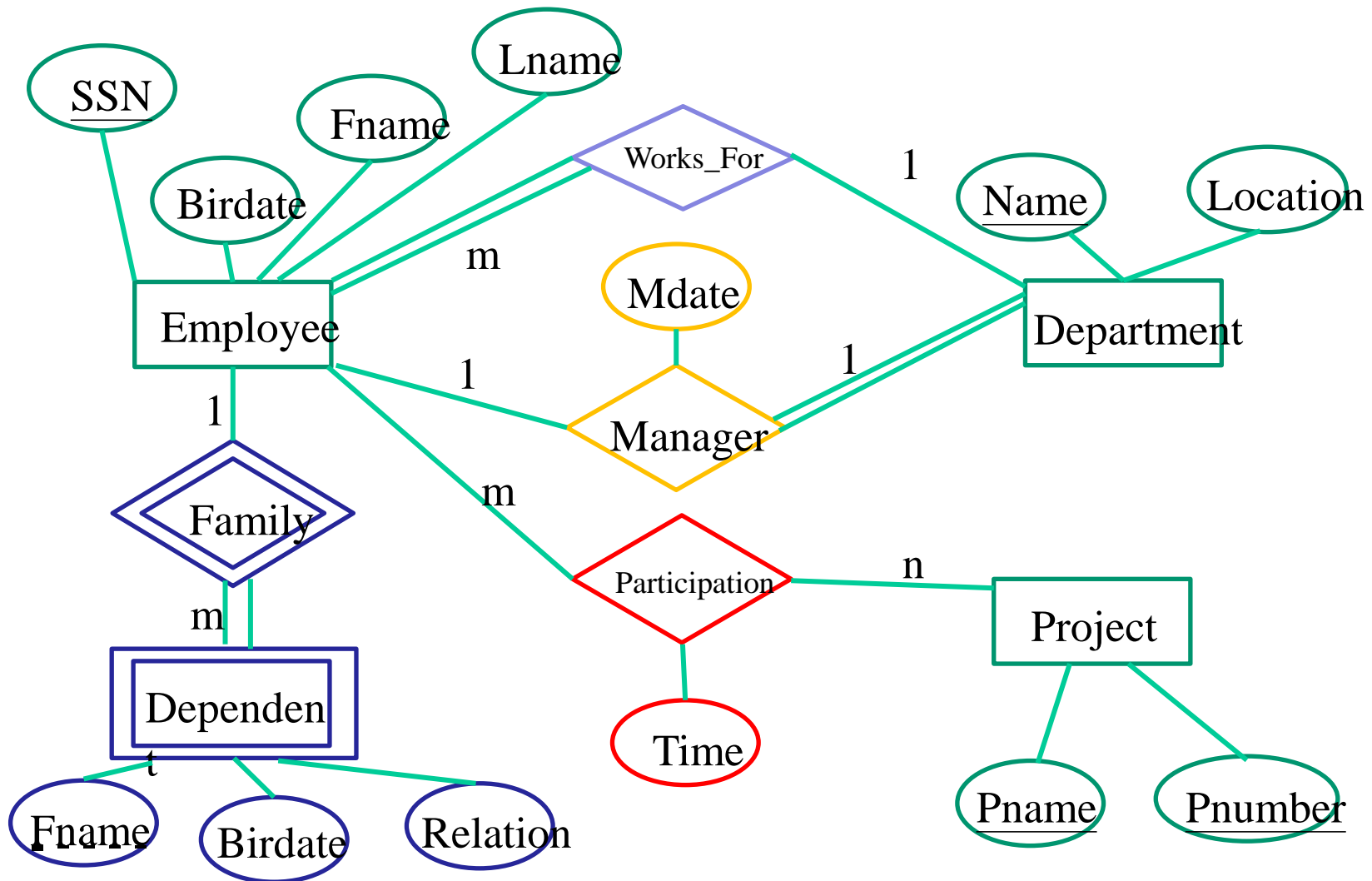
- This is what we mean

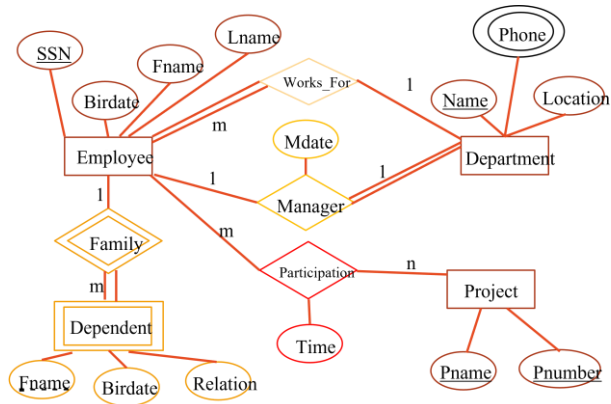


# Reducing to Relational Schema

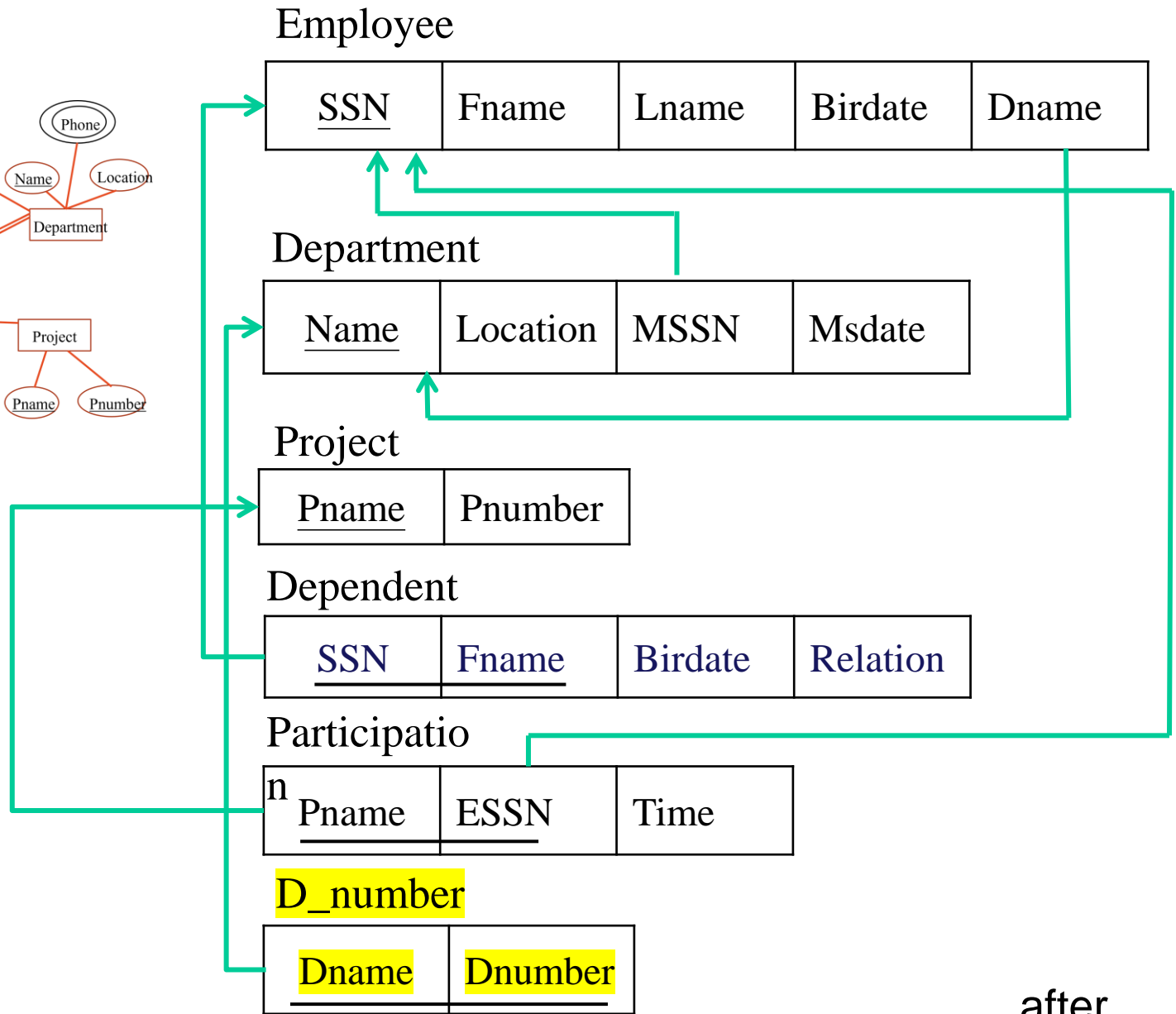
- Entity sets and relationship sets can be expressed uniformly as relation schemas that represent the contents of the database.
- **A database which conforms to an E-R diagram can be represented by a collection of relation schemas.**
- For each entity set and relationship set, there is a unique schema 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





before

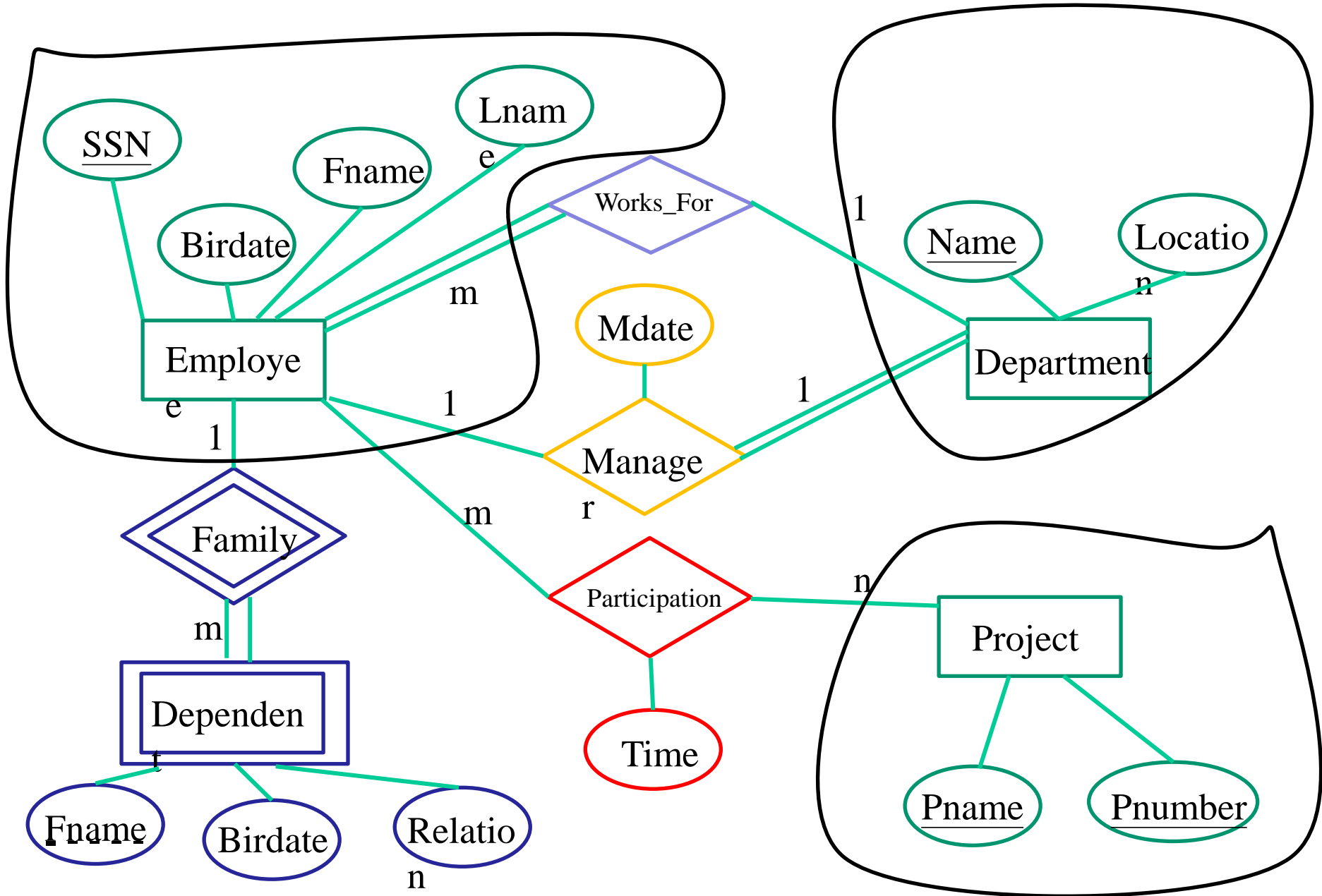


after

# Mapping Strong Entity Types

Step: 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
-------------	----------

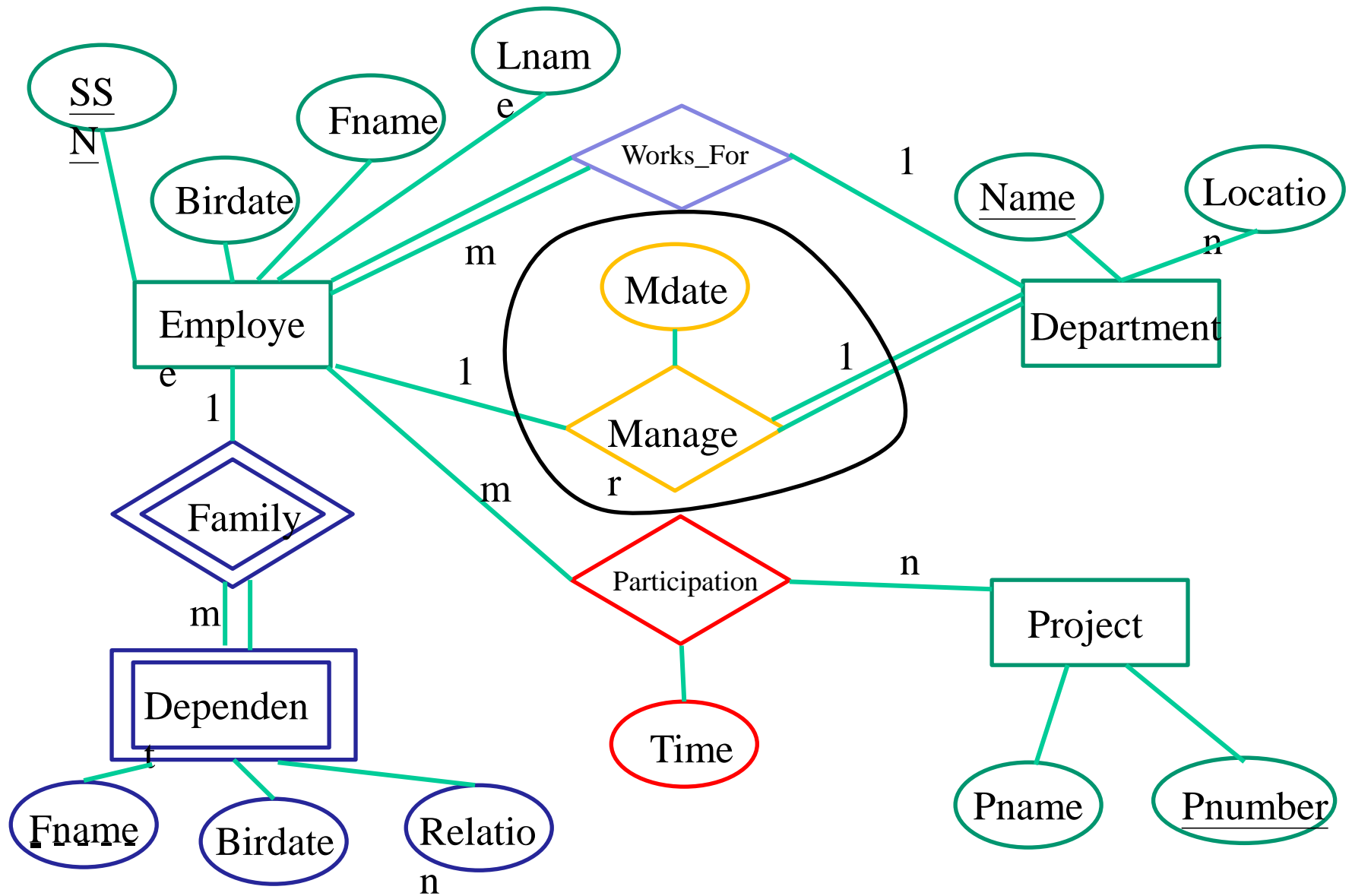
## Project

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

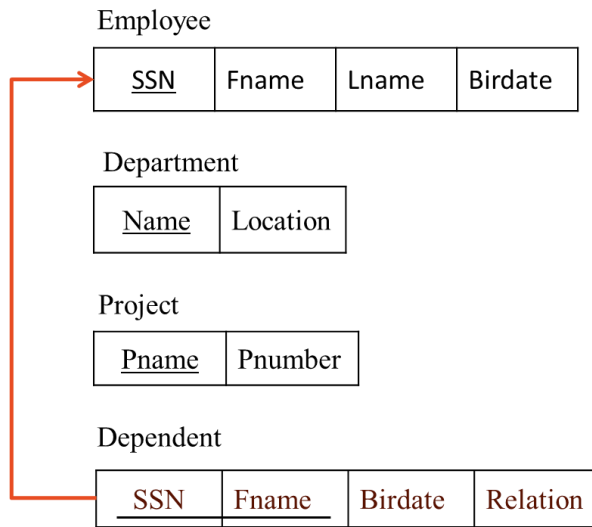
# Mapping 1:1 Relationship Types

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

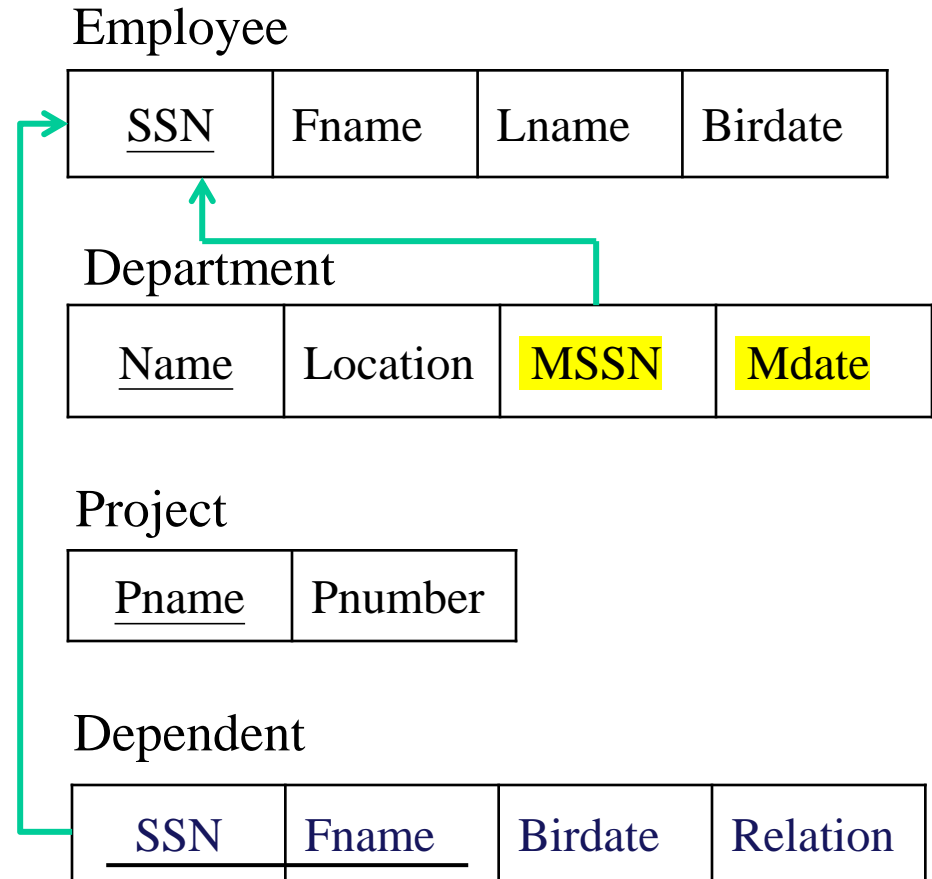
- Choose one of S and T (let S be one that participates totally if there is one).
- Add attributes from the primary key of T to S as a foreign key.
- Add all simple attributes (and simple components of composite attributes) of B as attributes of S.



# Mapping 1:1 Relationship Types



before



after

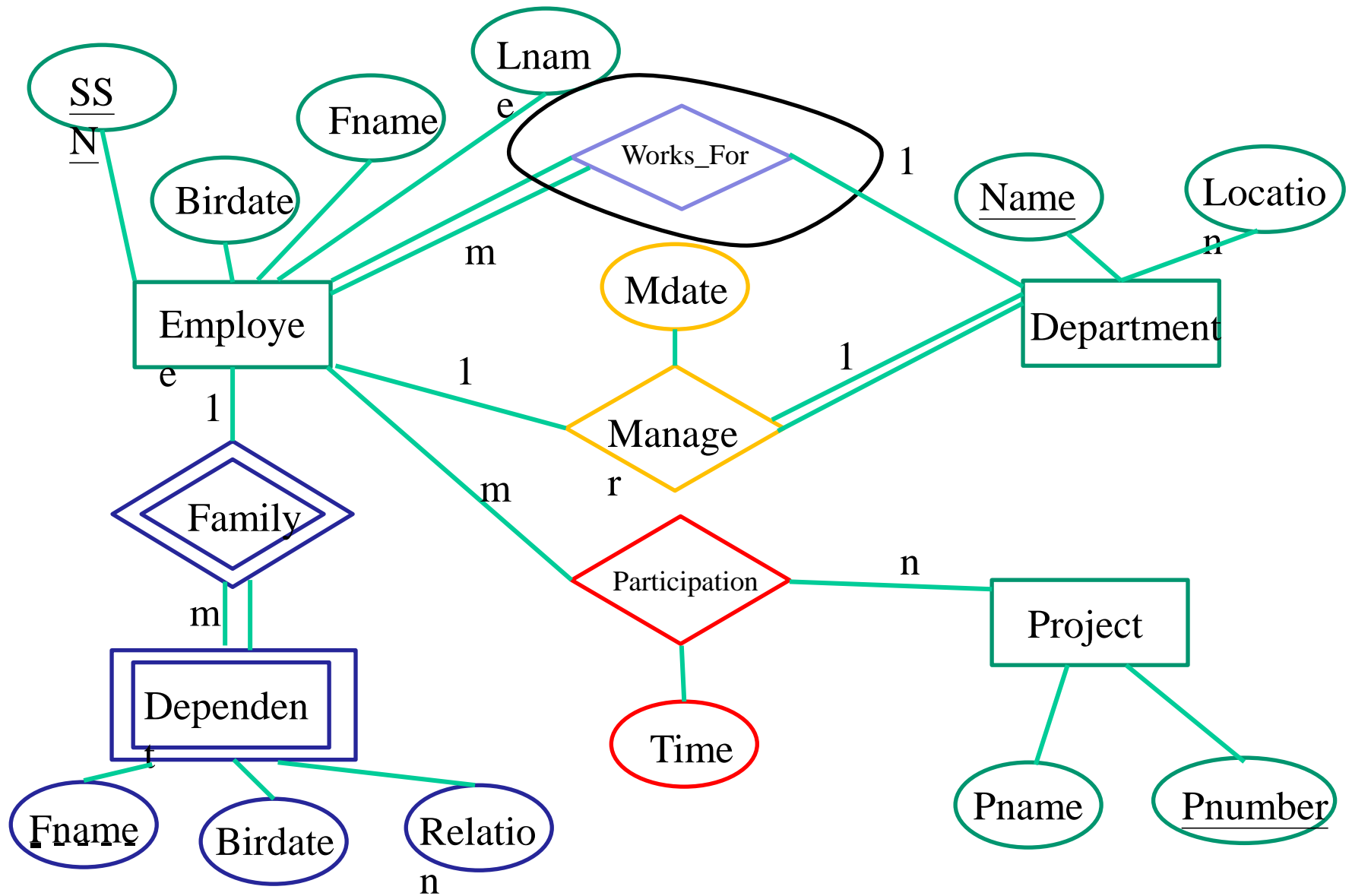
# Mapping 1:N Relationship Types

Step: 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.

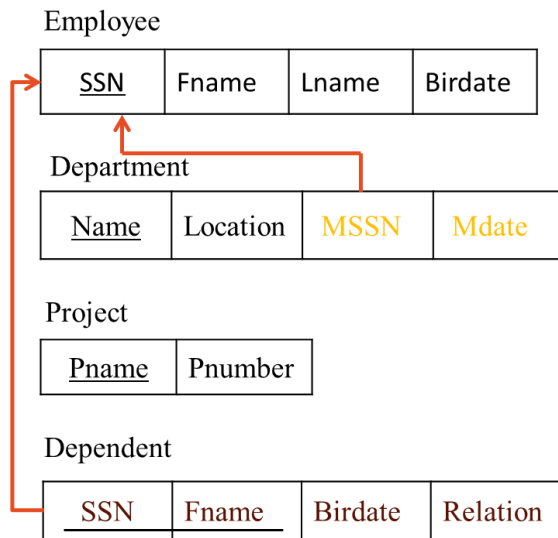
Add to the relation belonging to entity T,

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

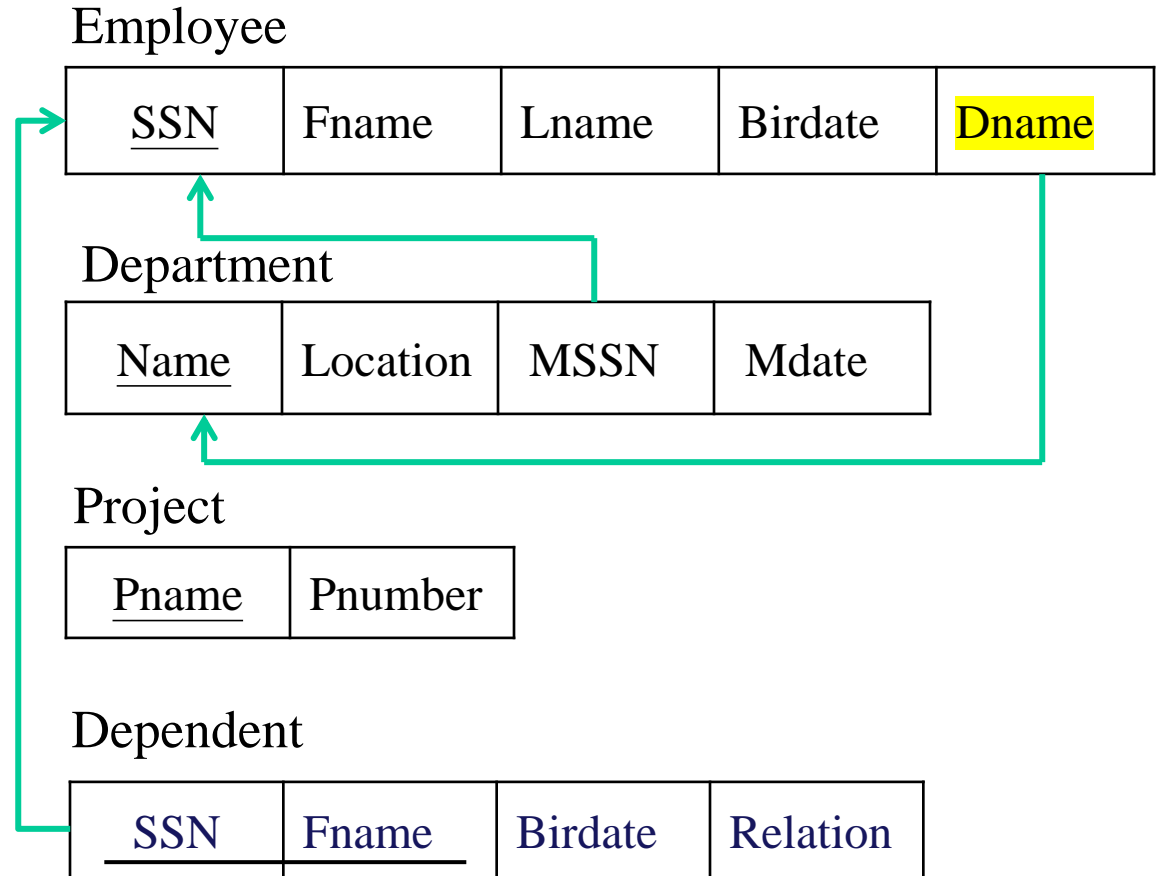
(doesn't add any new tuples, just attributes.)



# Mapping 1:N Relationship Types



before



after

# Mapping M:N Relationship Types

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

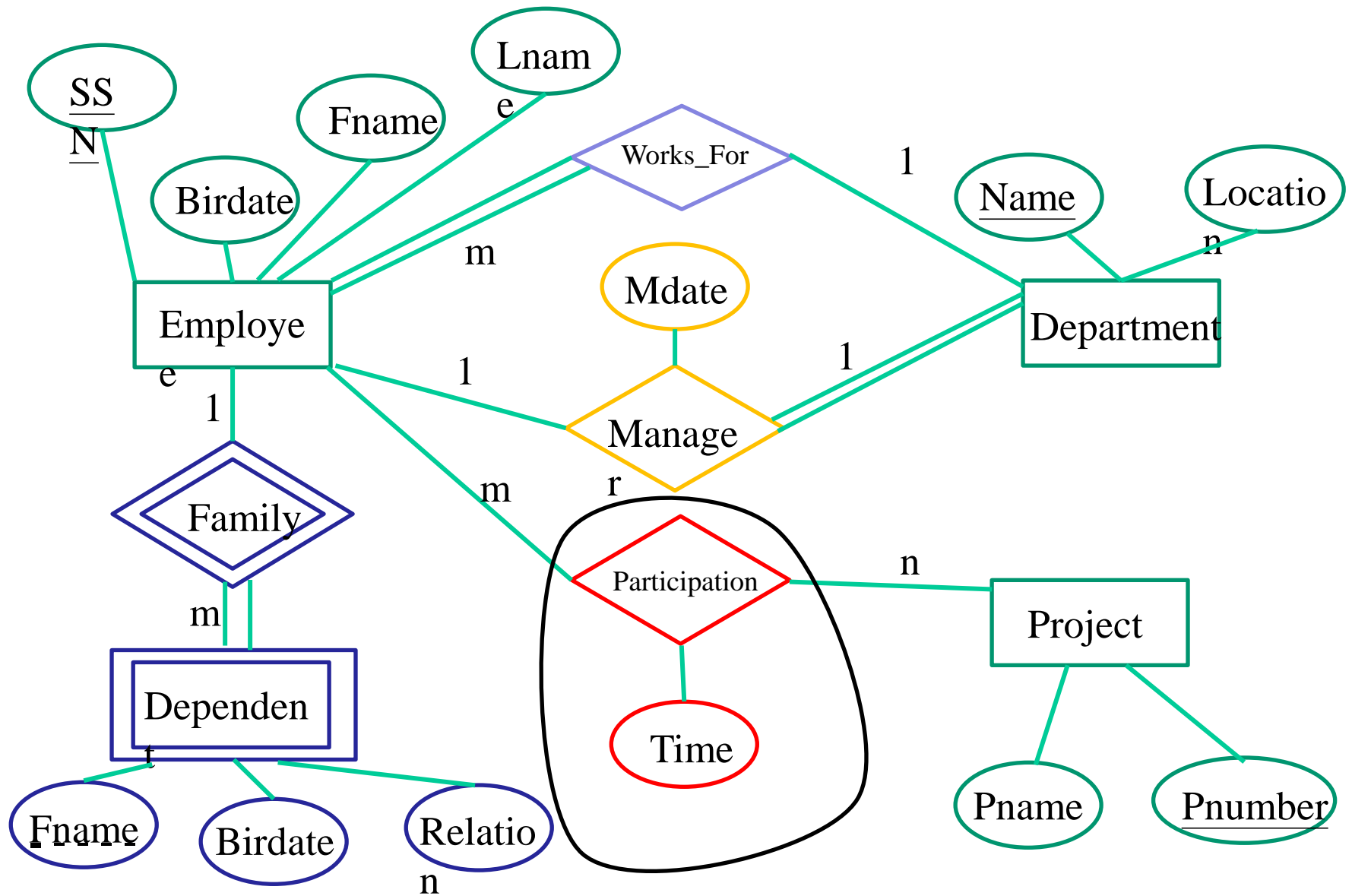
Create a new relation R with

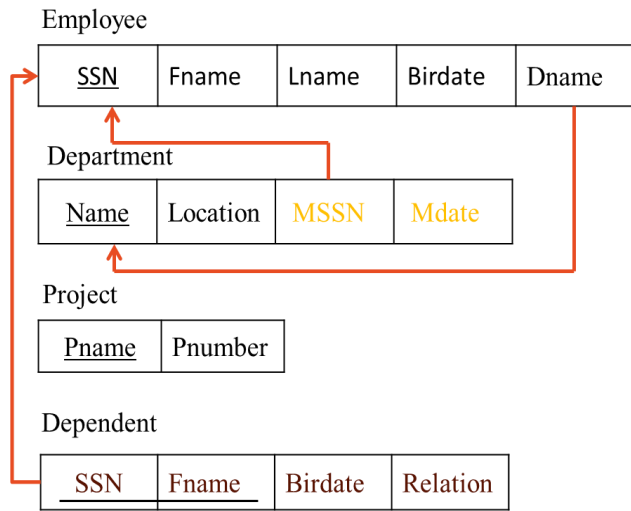
–Attributes :

- 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.

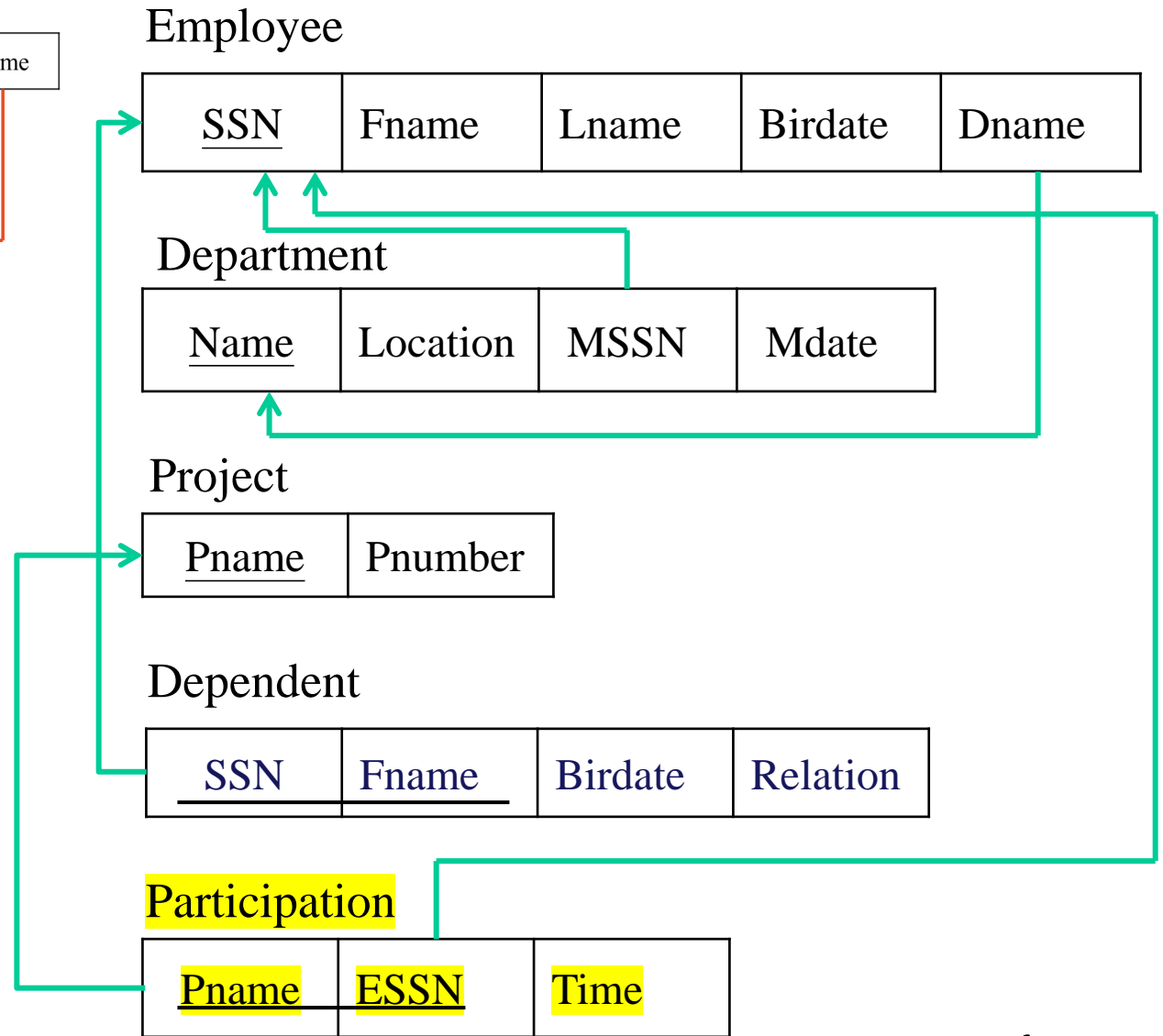
Key : All attributes from the key of S and the key of T.







before



after

# 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.