The Relational Data Model

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.

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

Each row of the table is called a tuple of the relation.

N.B. There is no ordering of column or rows.

Example

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

Above two tables are the same relation ---- Player

Relational Data Model

Mathematically,

- a *domain D* is a set of atomic values (having some fixed data type) which represent 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 disallowed!

Relational Data Model

A *tuple*, $t(A_1,A_2, ...,A_n)$, is a point in $dom(A_1) \times \ldots \times dom(A_n)$ where each $dom(A_j)$ is the domain of A_j.

A *relation (or a relation instance)* is a set of tuples: a subset of $dom(A_1) \times \ldots \times dom(A_n)$.

A relation schema is used to describe a relation.

The *degree* of a relation is the number of attributes of its relation schema.

Relational Data Model vs ER Model

Relation schema (intension) ⇄ entity or relationship type schema (intension).

attributes ⇄ attributes

tuple ⇄ instance of entity/relationship

relation (instance, extension) ⇄ entity/relationship extension

composite and multivalued attributes are allowed in ER model, but not allowed in relational data model.

Keys

Keys are used to identify tuples in a relation.

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

Note that this is a property of the relation that does not depend on the current relation instance.

A *candidate key* is a superkey, none of whose *proper* subsets is a superkey.

Keys are determined by the applications.

Keys

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

E.g. if {Name} is unique then it is a candidate key for PLAYER; otherwise we need to use the whole tuple or create a candidate key, say PID.

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

{Name, Goals} is a superkey but not a candidate key if {Name} is a key.

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, usually better to choose a primary key with a single attribute or a small number of attributes.

Integrity constraints

There are several kinds of integrity constraints that are an integral part of the relational model:

Key constraint: candidate key values must be unique for every relation instance.

Entity integrity: an attribute that is part of a primary key cannot be NULL.

Referential integrity: The third kind has to do with “foreign keys”.

Keys

Foreign keys are used to refer to a tuple in another relation.

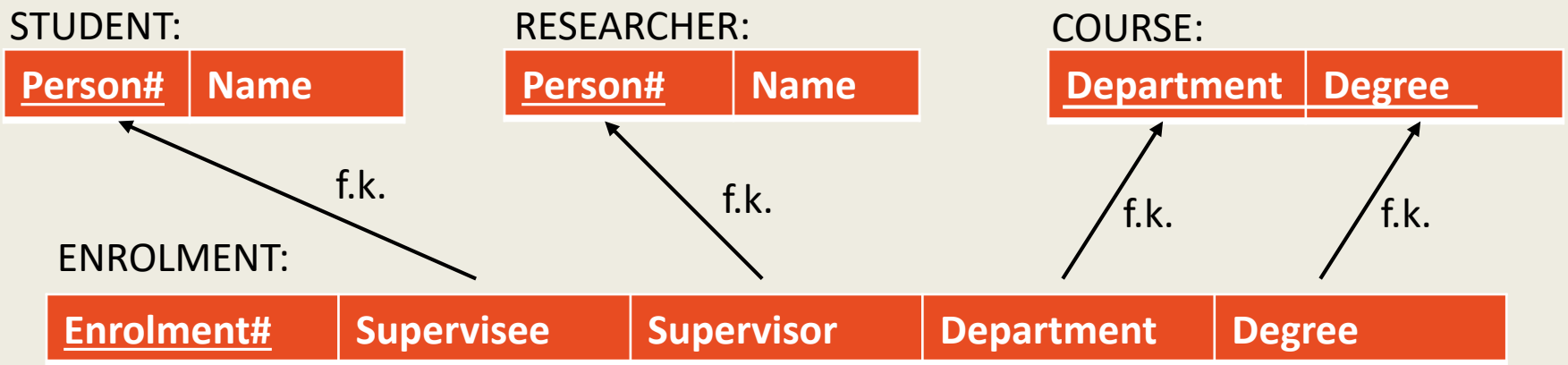
A set, *FK*, of attributes from a relation schema R1 may be a foreign *key* if

- 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.

Checking constraints on updates

- To maintain the integrity of the database, we need to check that integrity constraints will not be violated before proceeding with an update.
- Example: Suppose we have the following schema with foreign keys as shown:



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#	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.

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.

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#	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? No. Violates a key constraint.
   Action? Reject or allow the user to correct.

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#	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? No. Violates the entity integrity constraint.
   Action: Reject or correct.

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

Action: Reject, correct or accept after insertion of person number 6.

Deletions

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

Examples:

RESEARCHER:

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

1. Delete tuple with Person# = 2 from RESEARCHER

   Allowed? No. Violates the referential integrity.

   Action: Reject, correct or modify the ENROLMENT tuple by the following actions:

Deletions

deleting it (note that the this requires another integrity check, possibly causing a cascade of deletions), or

setting the foreign key value to NULL (note this can’t be done if it is part of a primary key), or

setting the foreign key value to another

acceptable value.

Modifications

If the modified attribute is a

- primary key: this is similar to deleting and then reinserting.
- foreign key: check that the new value refers to an existing tuple.
- neither: no problems can arise.

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.

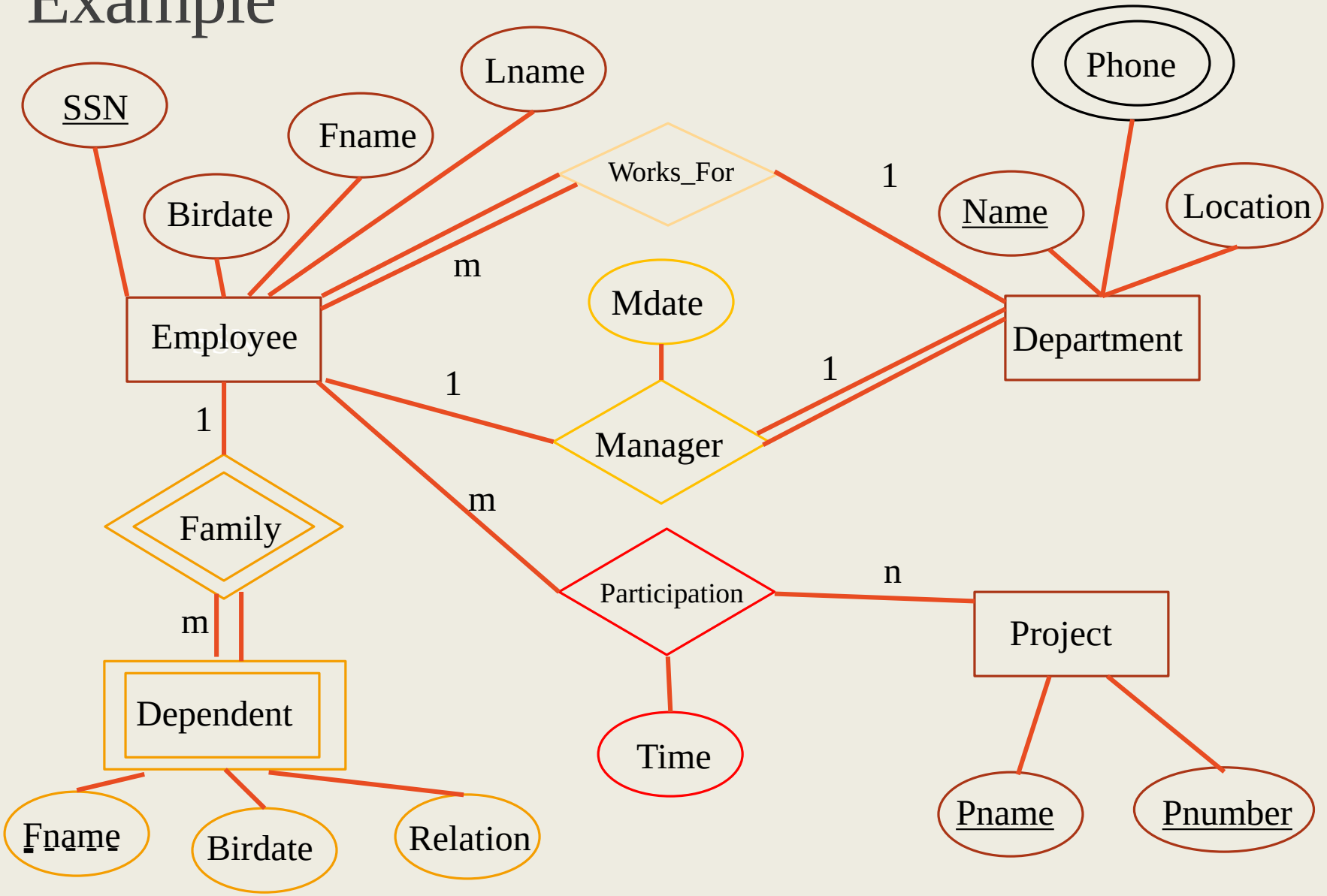
ER to Relational Data Model Mapping

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.

Example


SSN
Lname
Fname
Birdate
Works_For
1
Phone
Name
Location
m
Mdate
Employee
Department
1
1
1
Manager
Family
m
Participation
n
Project
m
Dependent
Time
Fname
Birdate
Relation
Pname
Pnumber

ER to RDM

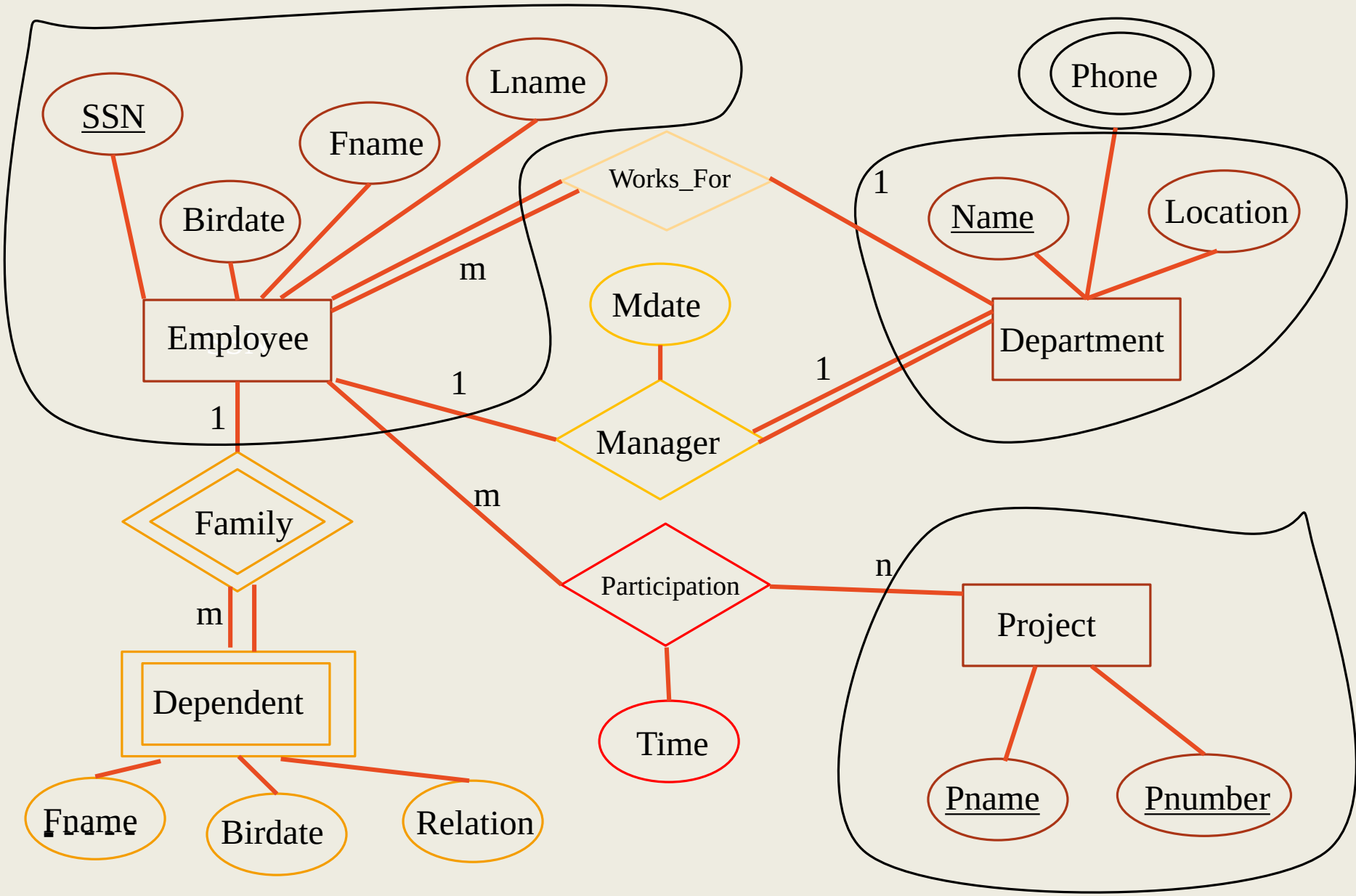
Step 1 : For each regular (not weak) entity type E, create a relation R with

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

ER to RDM

Step 1a : For each specialised entity type E, with parent entity type P, create a relation R with

- Attributes : The attributes of the key of P, plus the simple attributes of E.
- Key : The key of P.


SSN
Lname
Fname
Birdate
Works_For
Phone
1
Name
Location
m
Mdate
Employee
Department
1
1
1
Manager
Family
m
Participation
n
Project
m
Dependent
Time
Fname
Birdate
Relation
Pname
Pnumber

Employee

SSN	Fname	Lname	Birdate

Department

Name	Location

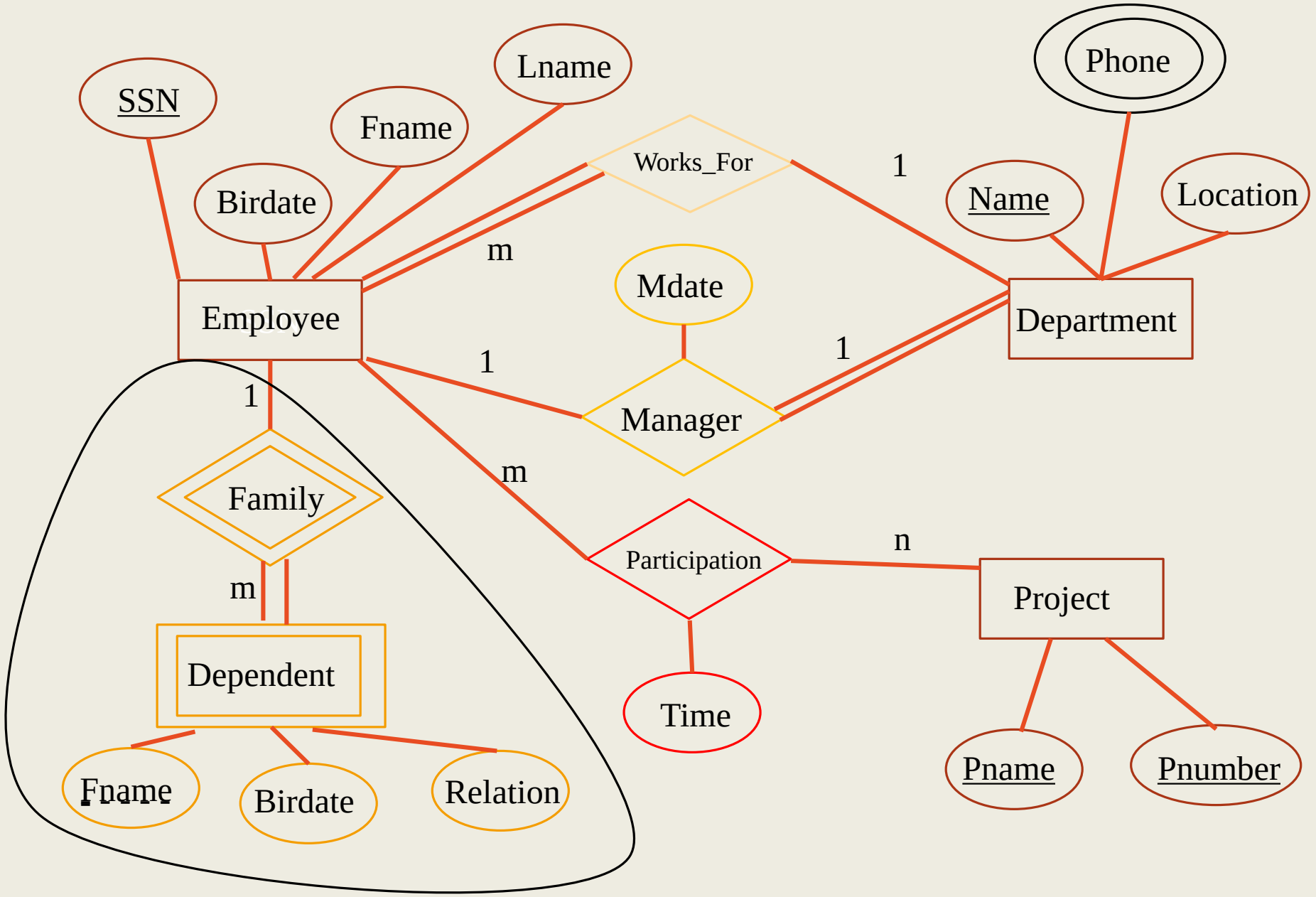
Project

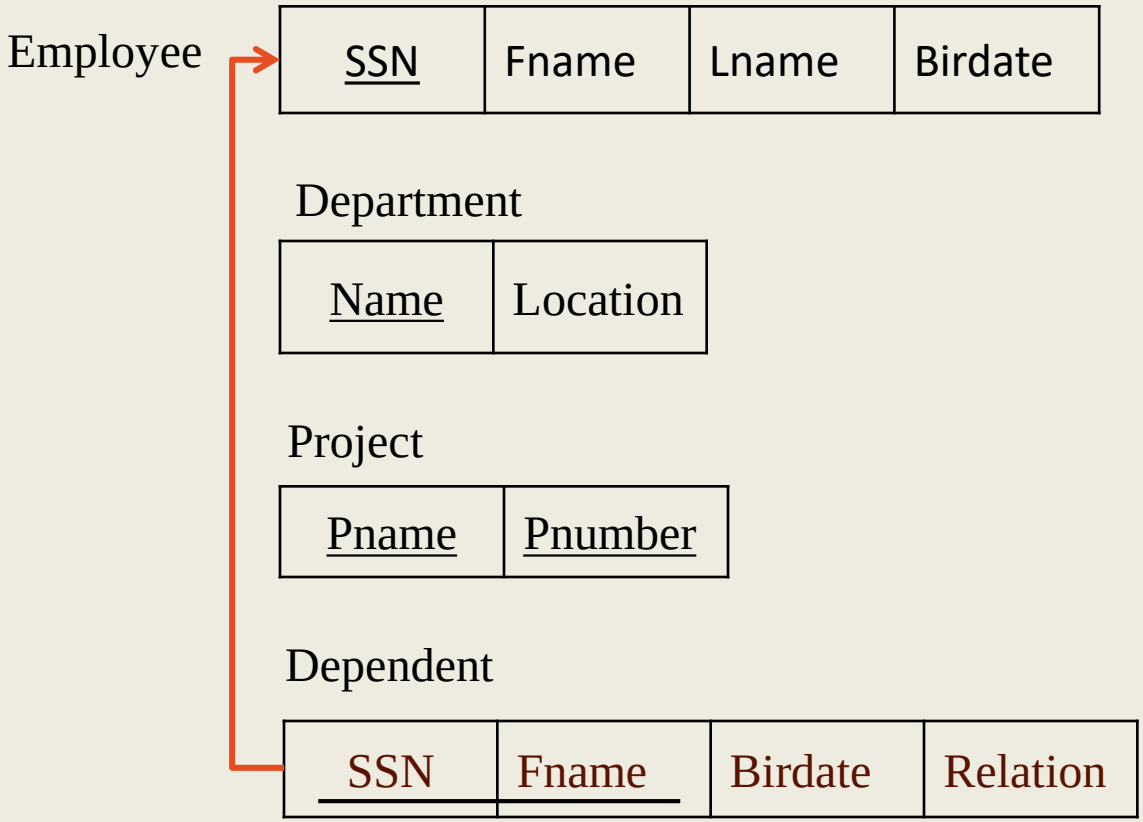
Pname	Pnumber

ER to RDM

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

- Attributes : All simple attributes (and simple components of composite attributes) of W, and include as a foreign key the prime attributes of the relation derived from E.
- Key : The foreign key plus the partial key of W.


SSN
Fname
Lname
Birdate
Works_For
1
Phone
Name
Location
m
Mdate
Employee
Department
1
1
Manager
1
Family
m
Participation
n
Project
m
Dependent
Time
Fname
Birdate
Relation
Pname
Pnumber

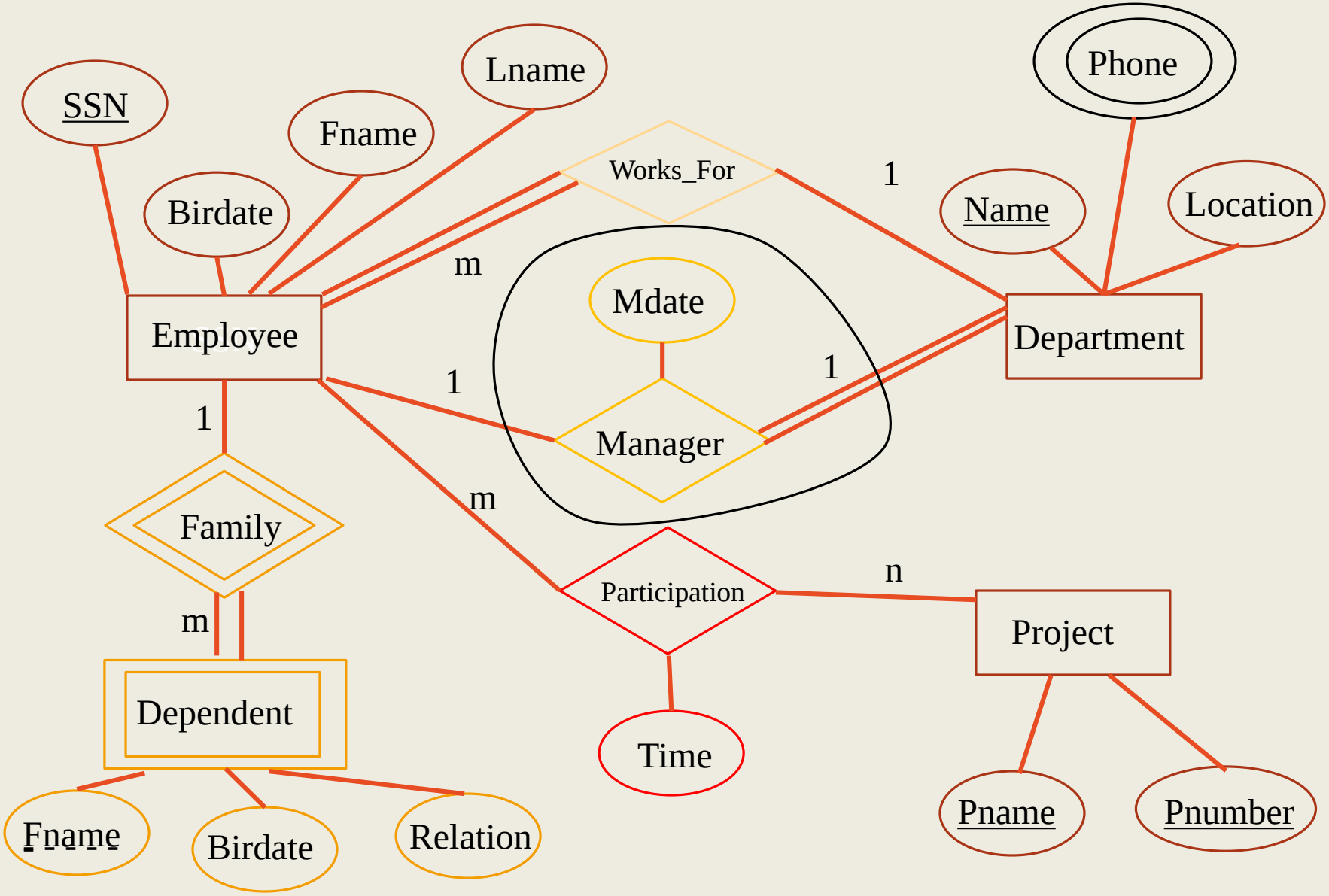

Employee
SSN
Fname
Lname
Birdate
Department
Name
Location
Project
Pname
Pnumber
Dependent
SSN
Fname
Birdate
Relation

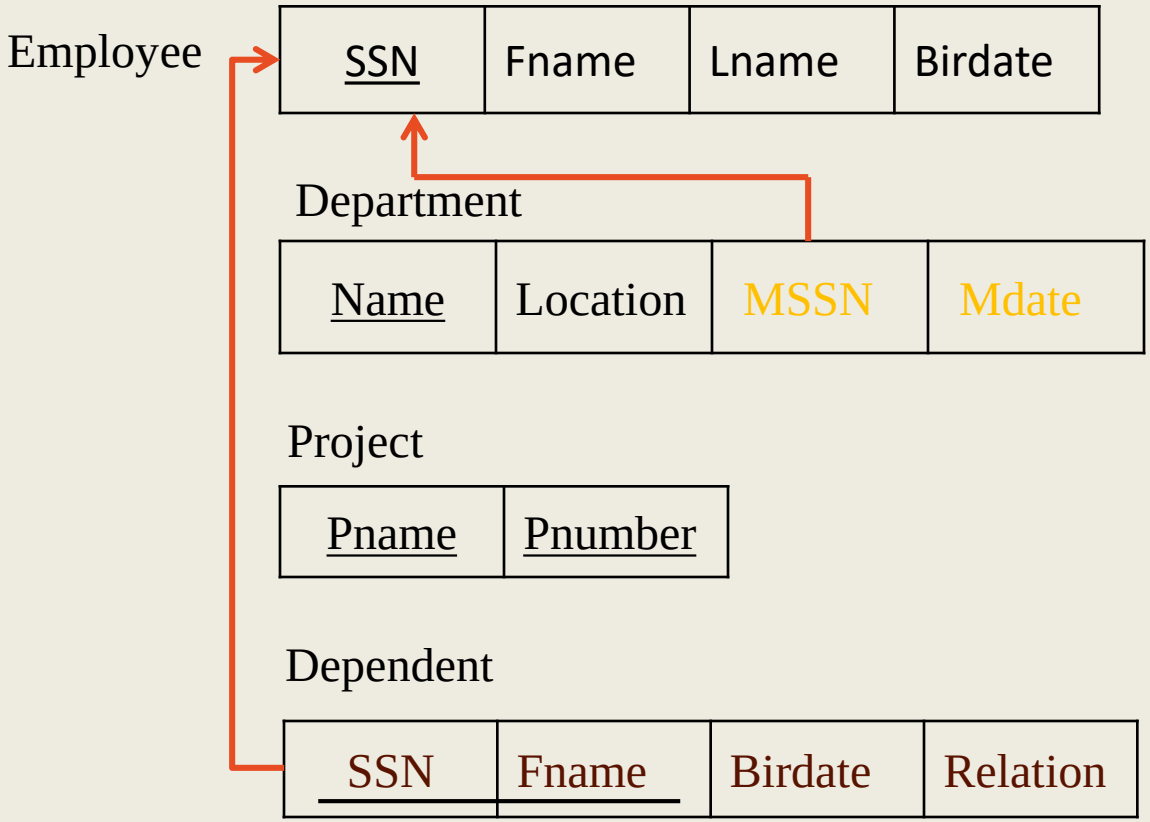
ER to RDM

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 (prefer one that participates totally), say S.
- Add the attributes of the primary key of T to S as a foreign key.
- Add the simple attributes (and simple components of composite attributes) of B as attributes of S.

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


SSN
Fname
Lname
Birdate
Works_For
Phone
Name
Location
Employee
Mdate
Department
Manager
Family
Participation
Project
Dependent
Time
Fname
Birdate
Relation
Pname
Pnumber
1
m
1
1
1
m
m
n

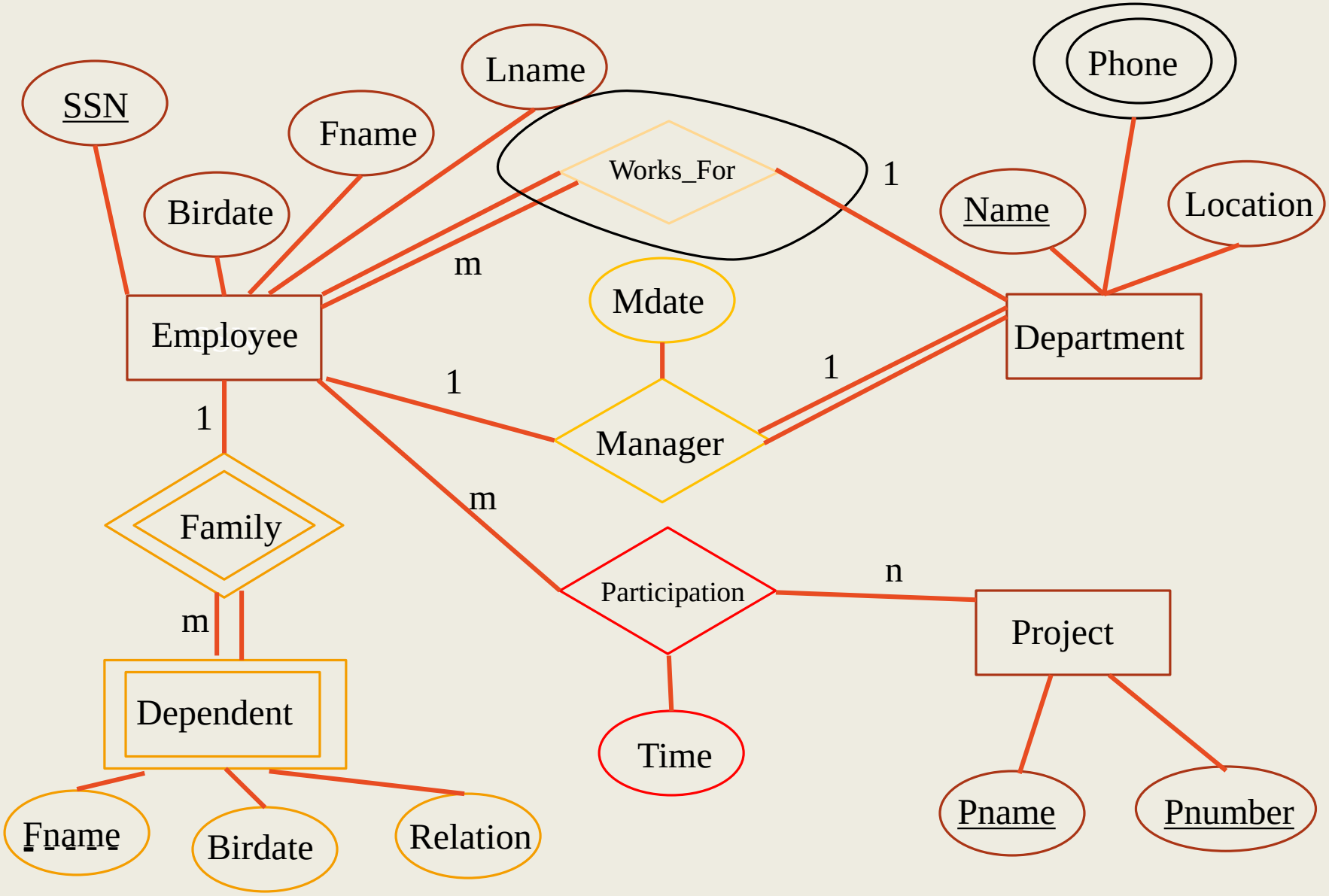

Employee
SSN
Fname
Lname
Birdate
Department
Name
Location
MSSN
Mdate
Project
Pname
Pnumber
Dependent
SSN
Fname
Birdate
Relation

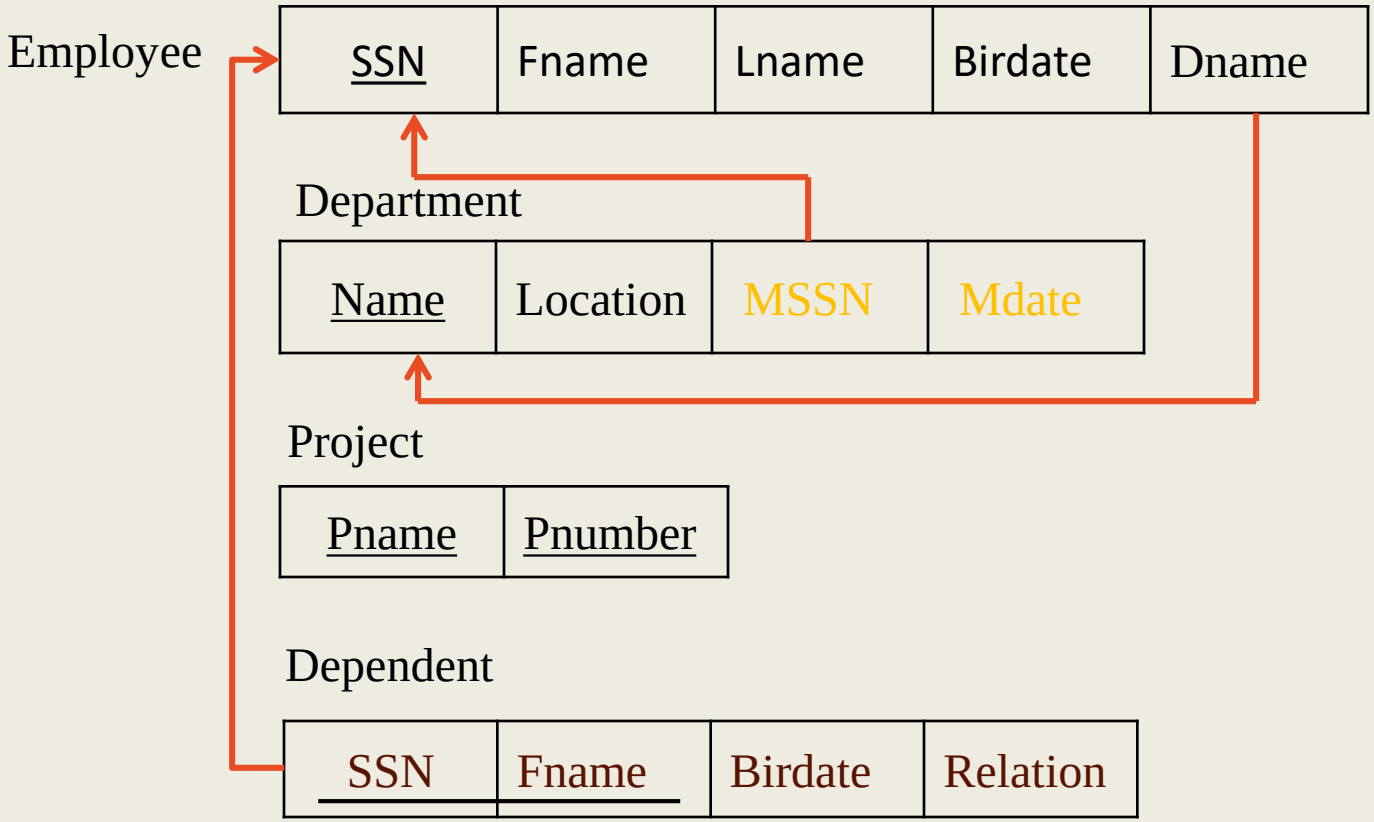
ER to RDM

Step 4 : For each regular 1:N relationship type B.

- Let E and F be the participating entity types.
- Let E by the entity type on the 1 side, F the one on the N side.
- Let S and T be the corresponding relations.
- Add the attributes of the primary key of S to T as a foreign key.
- Add to T any simple attributes (or simple components of composite attributes) of the relationship.

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

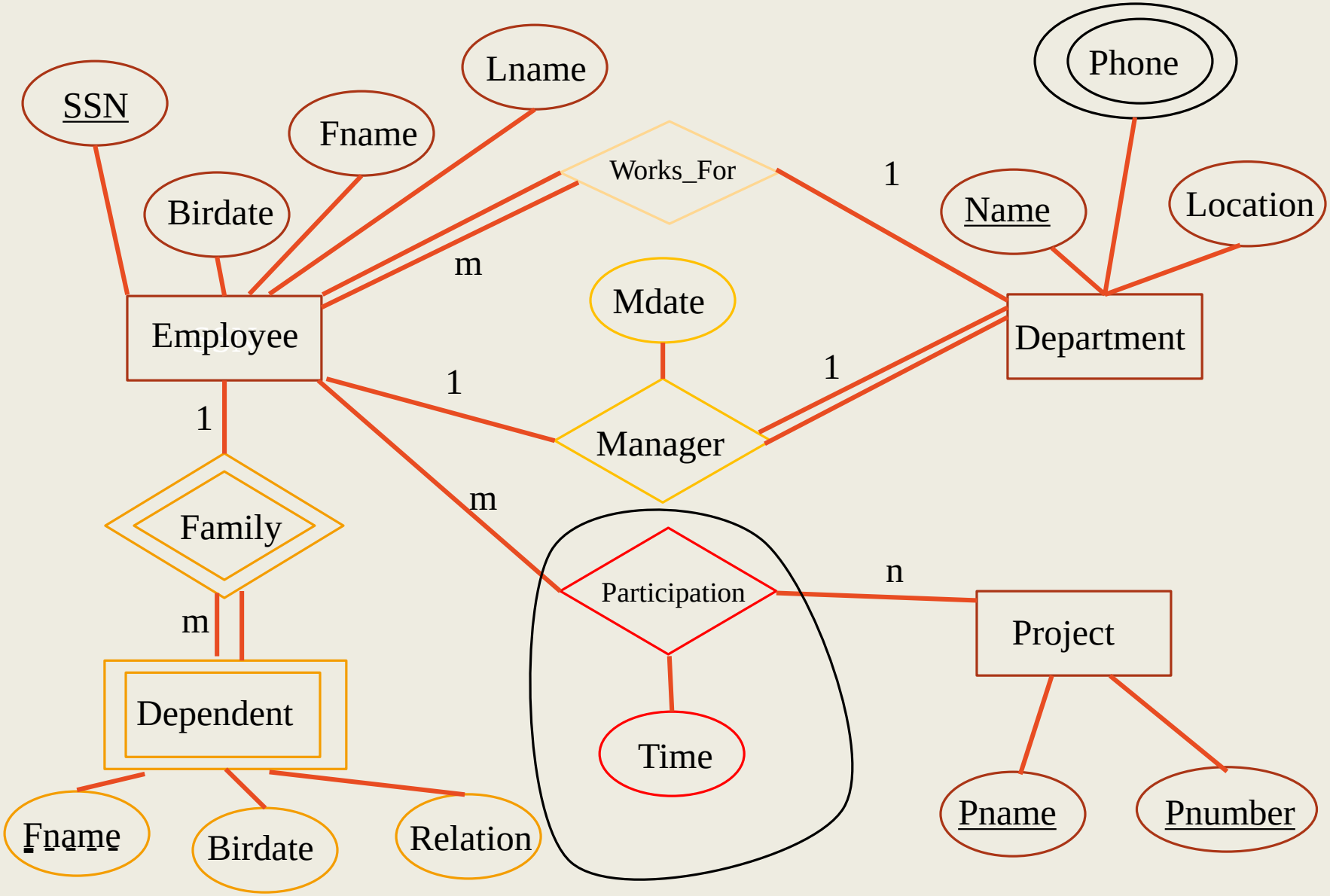

SSN
Lname
Fname
Phone
Works_For
1
Birdate
Name
Location
m
Mdate
Employee
Department
1
1
1
Manager
Family
m
Participation
n
Project
m
Dependent
Time
Fname
Birdate
Relation
Pname
Pnumber

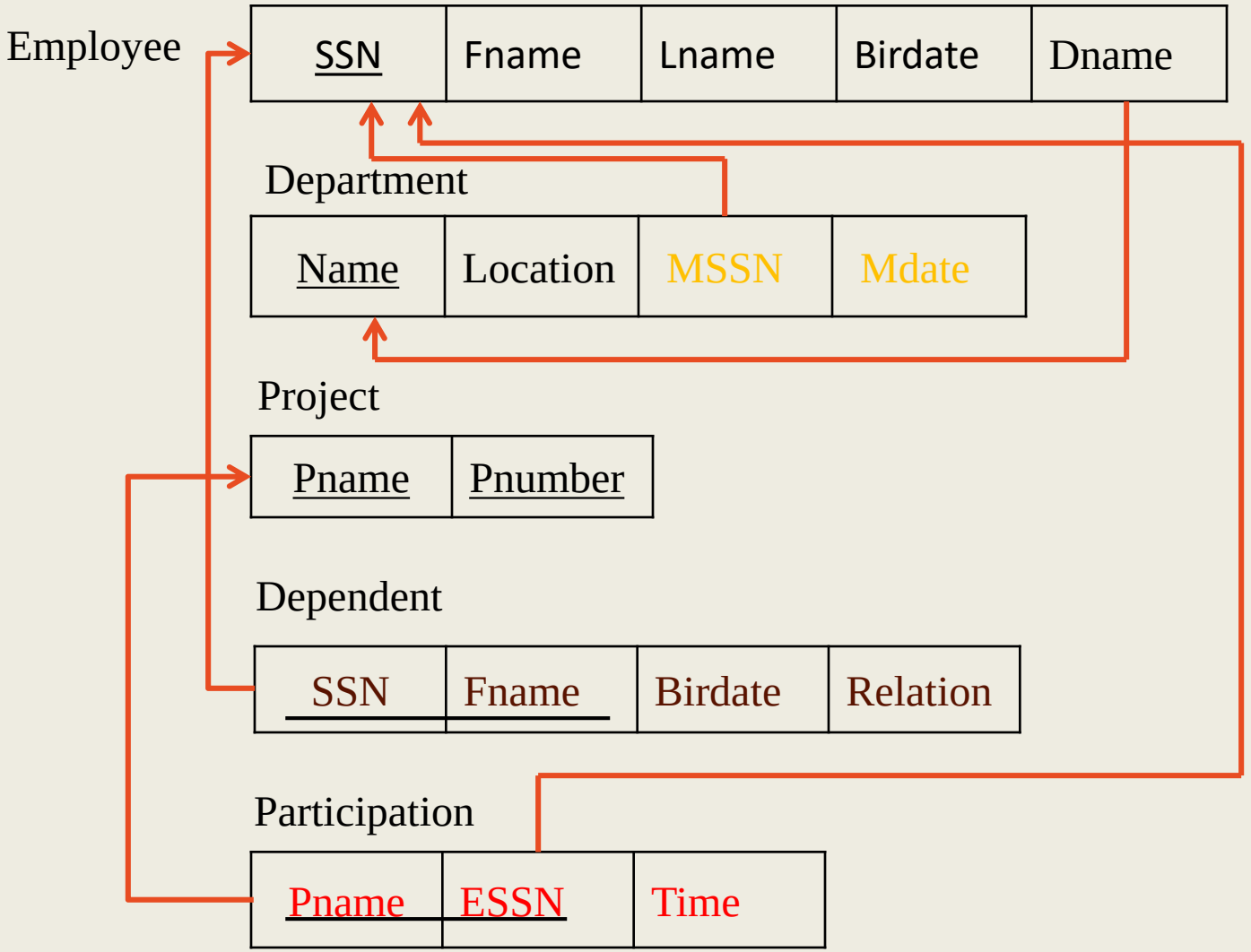

Employee
SSN
Fname
Lname
Birdate
Dname
Department
Name
Location
MSSN
Mdate
Project
Pname
Pnumber
Dependent
SSN
Fname
Birdate
Relation

ER to RDM

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

- Attributes : The key of S and the key of T as foreign keys, plus the simple attributes (and simple components of composite attributes) of B.
- Key : The key of S and the key of T.

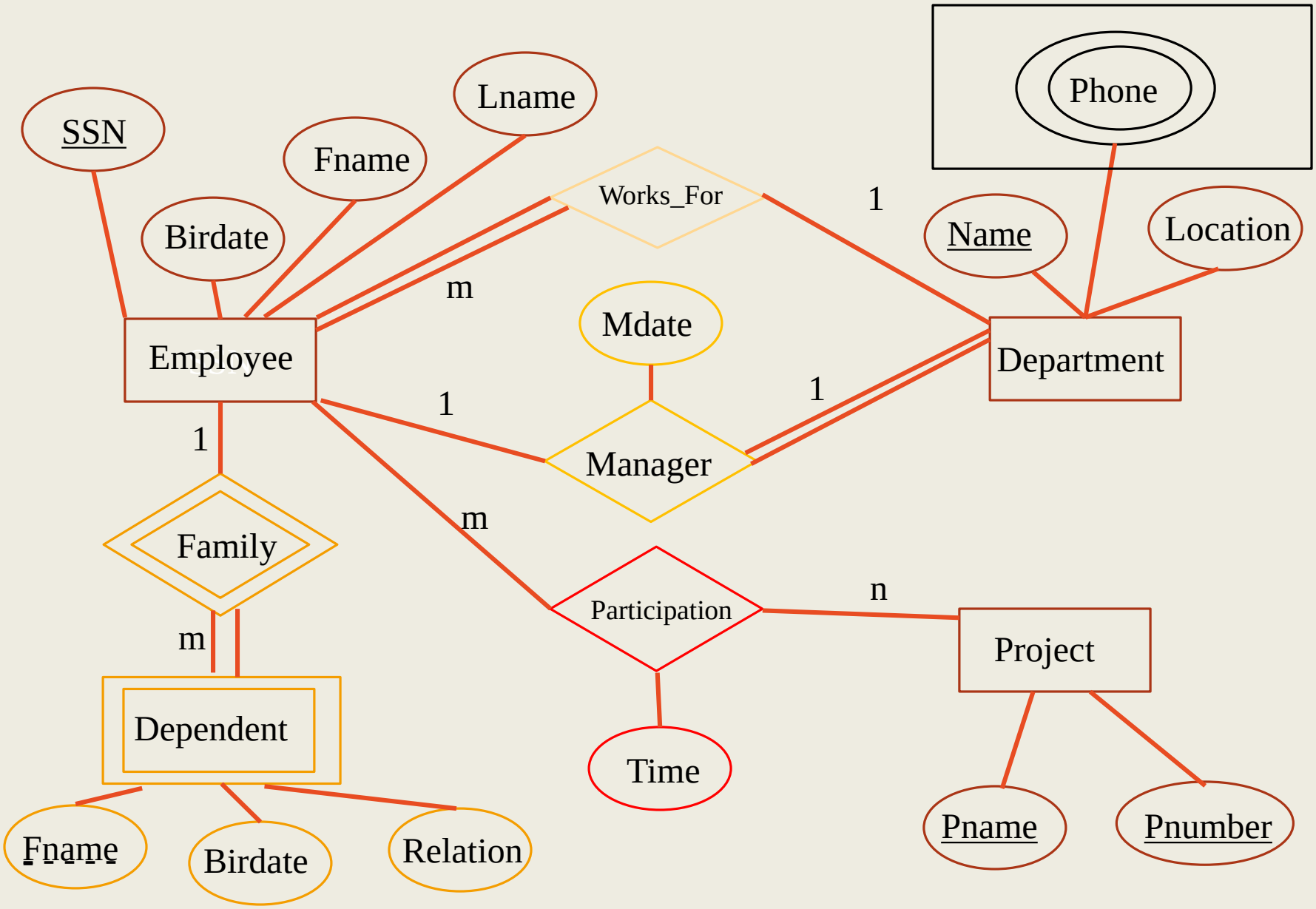

SSN
Fname
Lname
Birdate
Works_For
1
Phone
Name
Location
m
Mdate
Employee
Department
1
1
1
Manager
m
Family
Participation
n
Project
m
Dependent
Time
Fname
Birdate
Relation
Pname
Pnumber

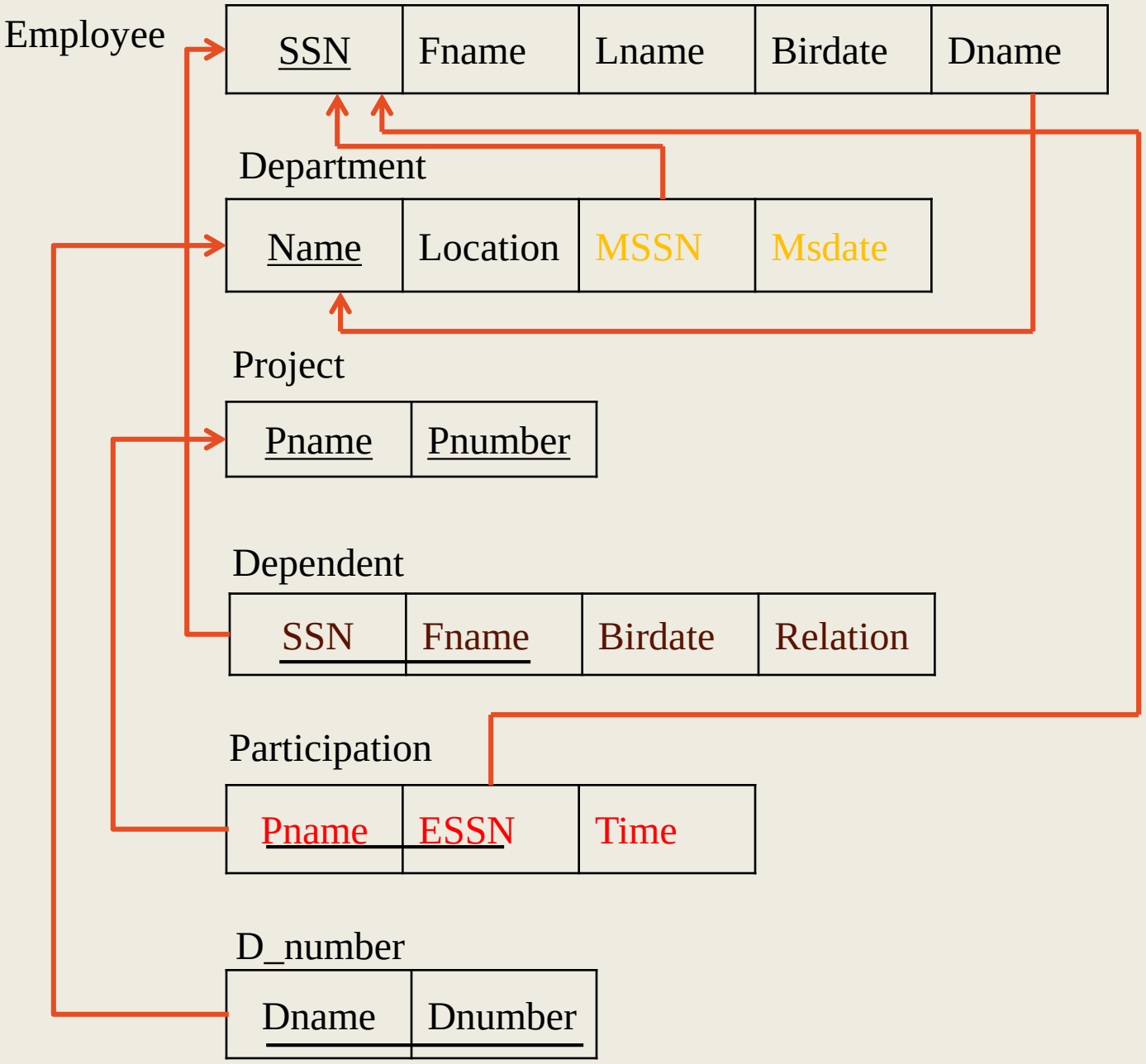

Employee
SSN
Fname
Lname
Birdate
Dname
Department
Name
Location
MSSN
Mdate
Project
Pname
Pnumber
Dependent
SSN
Fname
Birdate
Relation
Participation
Pname
ESSN
Time

ER to RDM

Step 6 : For each multivalued attribute A. Create a new relation R. Let A be an attribute of E.

- Attributes :
  1. A (if A is a simple attribute) together with the key of E as a foreign key.
  2. The simple components of A (if A is a composite attribute), together with the key of E as a foreign key.
- Key : All attributes.


SSN
Fname
Lname
Birdate
Works_For
1
Phone
Name
Location
m
Mdate
Employee
Department
1
1
Manager
1
Family
m
Participation
n
Project
m
Dependent
Time
Fname
Birdate
Relation
Pname
Pnumber


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

ER to RDM

Step 7 : For each n-ary relationship type (n > 2). Create a new relation with

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