Databases and Info Systems

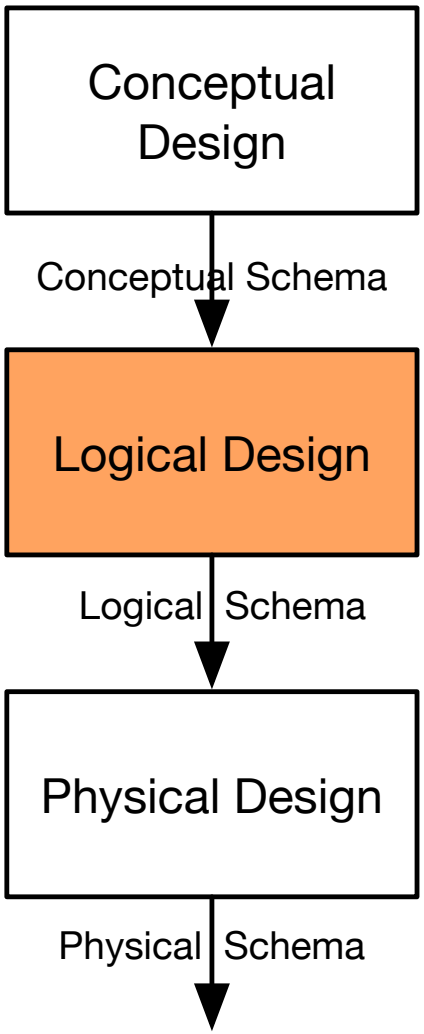
Logical Design

Dr. Seán Russell
sean.russell@ucd.ie,

School of Computer Science,
University College Dublin

March 24, 2020

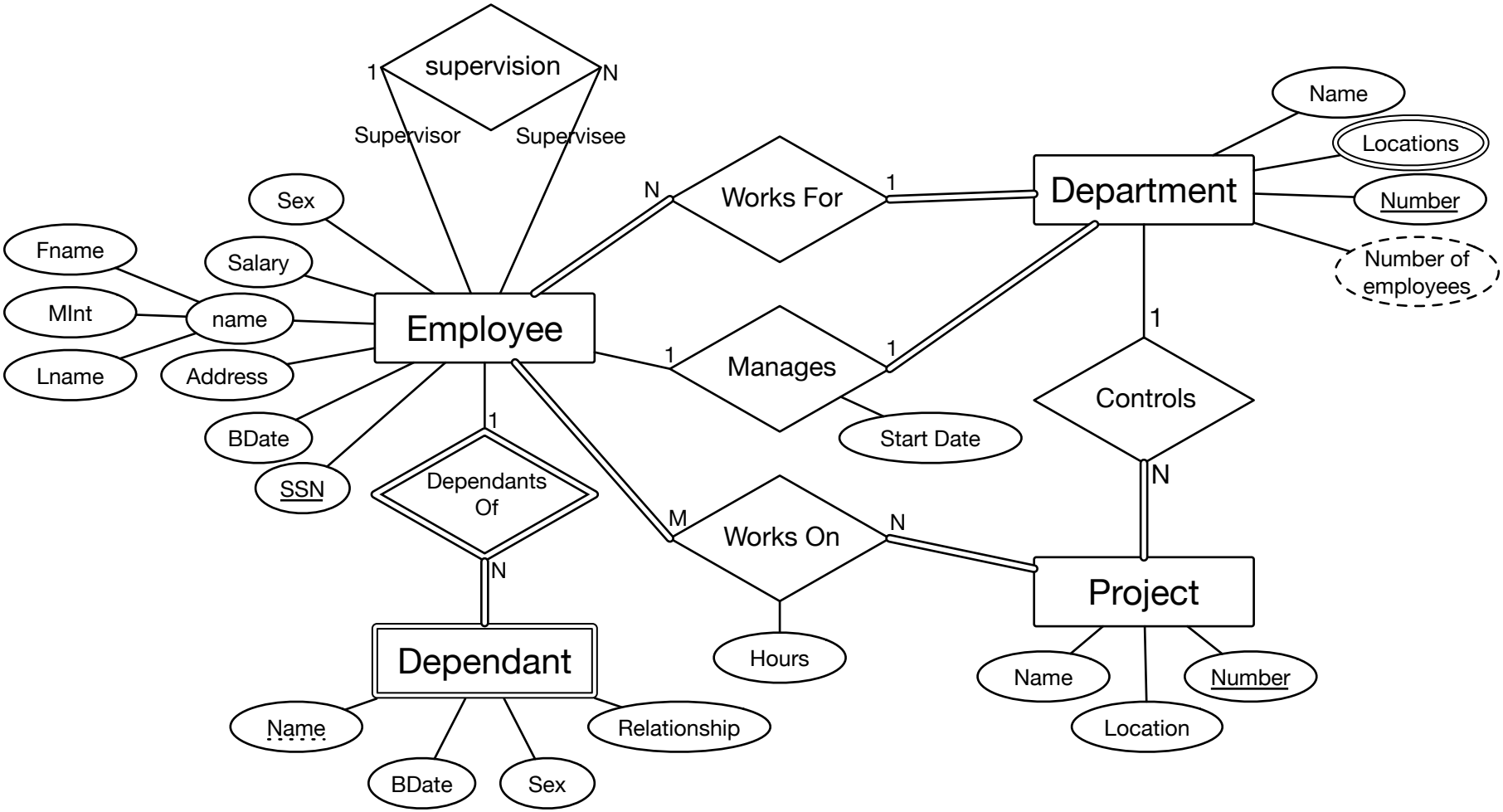
Logical Design



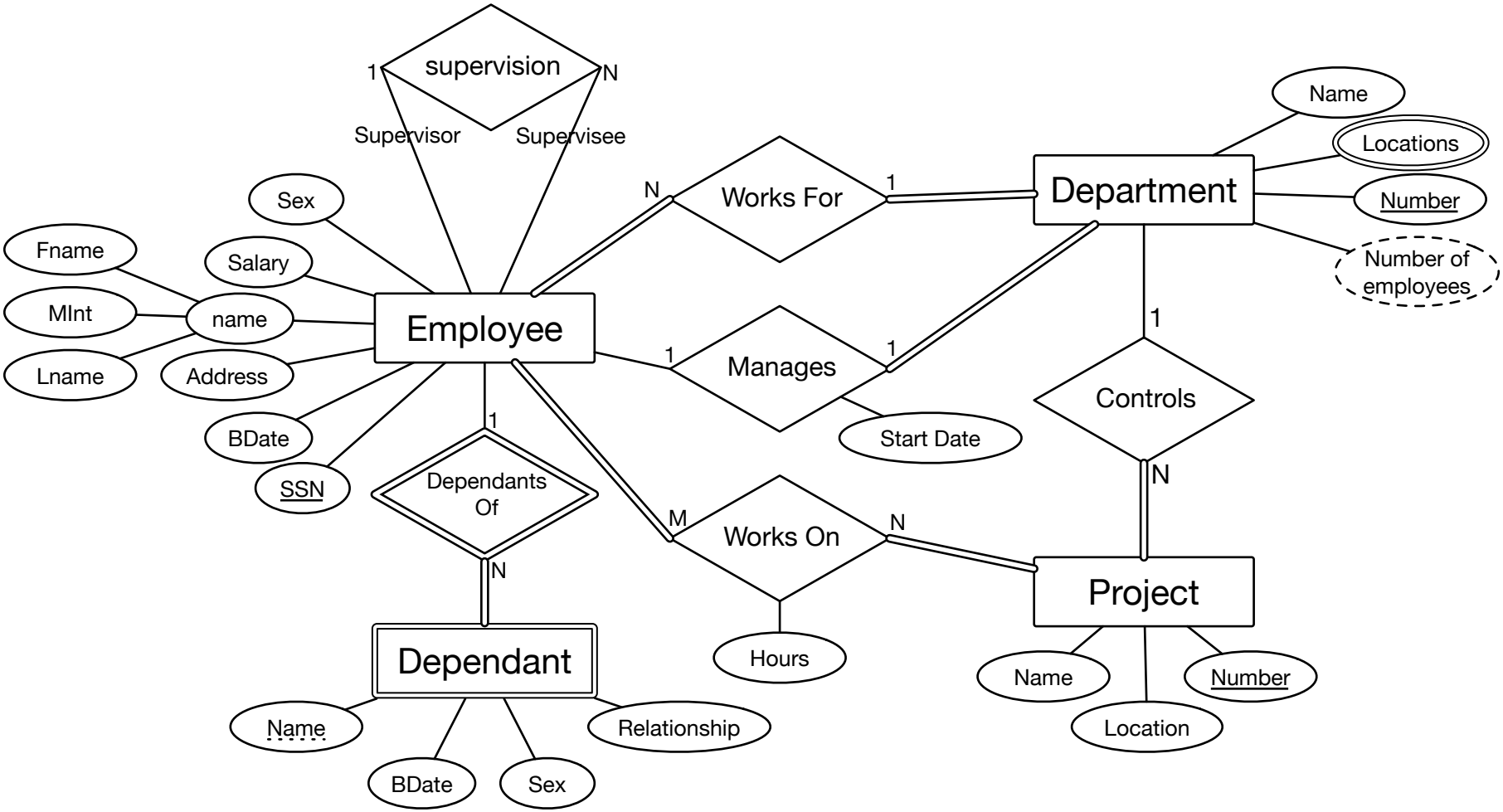
- An ER/EER diagram is a **conceptual schema** of the database.
- Next, we need to map this to a **logical schema**: i.e. the tables and attributes that will be created to store the data in the database

Table of Contents

- In this lecture, we will look at a method to convert an ER/EER diagram into a logical model (i.e. the relations that will be used to create a database)
- This follows an 8-step algorithm
  1. Mapping Regular Entity Types
  2. Mapping Weak Entity Types
  3. Mapping 1:1 Relationships
  4. Mapping 1:N Relationships
  5. Mapping M:N Relationships
  6. Mapping Multivalued Attributes
  7. Mapping N-ary Relationships
  8. Mapping supertypes/subtypes


supervision
1
N
Supervisor
Supervisee
Sex
Fname
Salary
MInt
name
Lname
Address
BDate
SSN
Employee
N
Works For
1
Department
Name
Locations
Number
Number of employees
1
Manages
1
Start Date
1
Controls
N
1
Dependants Of
N
Dependant
Name
BDate
Sex
Relationship
M
Works On
N
Hours
Project
Name
Number
Location

- We begin with regular (strong) entity types.
- Create a relation for each strong entity.
- Include all simple attributes:
  - For any composite attributes, only include its component attributes.
  - For multi-valued attributes, leave these out until later.
- Select a primary key (possibly a composite primary key consisting of multiple attributes).
- We do not consider foreign keys and relationships until later in the process.


supervision
1
N
Supervisor
Supervisee
Sex
Fname
Salary
MInt
name
Lname
Address
BDate
SSN
Employee
N
Works For
1
Department
Name
Locations
Number
Number of employees
1
Manages
1
Start Date
1
Controls
N
1
Dependants Of
N
Dependant
Name
BDate
Sex
Relationship
M
Works On
N
Hours
Project
Name
Number
Location

Employee

Fname	Minit	Lname	SSN	Bdate	Address	Salary	Sex

Department

Dname	Dnumber

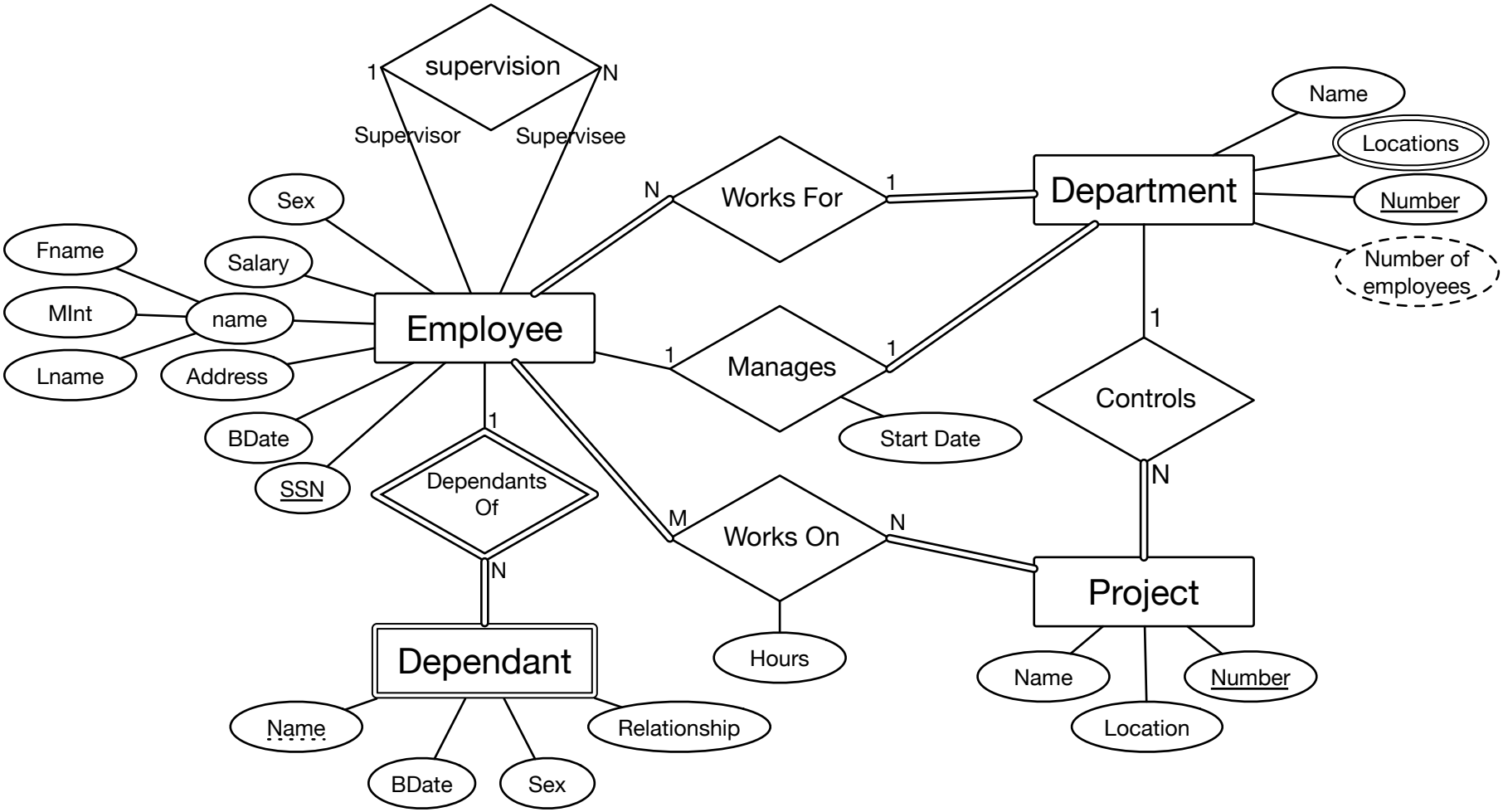
Project

Pname	Pnumber	Plocation

Notes

- Composite attribute "Name" from EMPLOYEE is not included: only its component attributes: Fname (first name), Minit (middle initial) and Lname (last name)
- For DEPARTMENT, either Dname or Dnumber would be OK to choose as a primary key. We choose only one (primary keys are underlined)
- We do not yet include the multi-valued attribute "Locations" for DEPARTMENT (we will do that later)
- No relationships/foreign keys are included yet

- For any weak entity types, create a relation.
- As with step 1, include any simple attributes it has (including component attributes of composite attributes).
- Include the primary key of the owner entity type as a foreign key.
- The primary key of this relation is a combination of this key and the weak entity's partial key.
- If the owner is also a weak entity type, that should be mapped first (so that we know its primary key).
- Usually, we use the CASCADE option for referential integrity, because the weak entity cannot exist without its owner


supervision
1
N
Supervisor
Supervisee
Sex
Fname
Salary
MInt
name
Lname
Address
BDate
SSN
Employee
N
Works For
1
Department
Name
Locations
Number
Number of employees
1
Manages
1
Start Date
1
Controls
N
1
Dependants Of
N
Dependant
Name
BDate
Sex
Relationship
M
Works On
N
Hours
Project
Name
Number
Location

Dependant

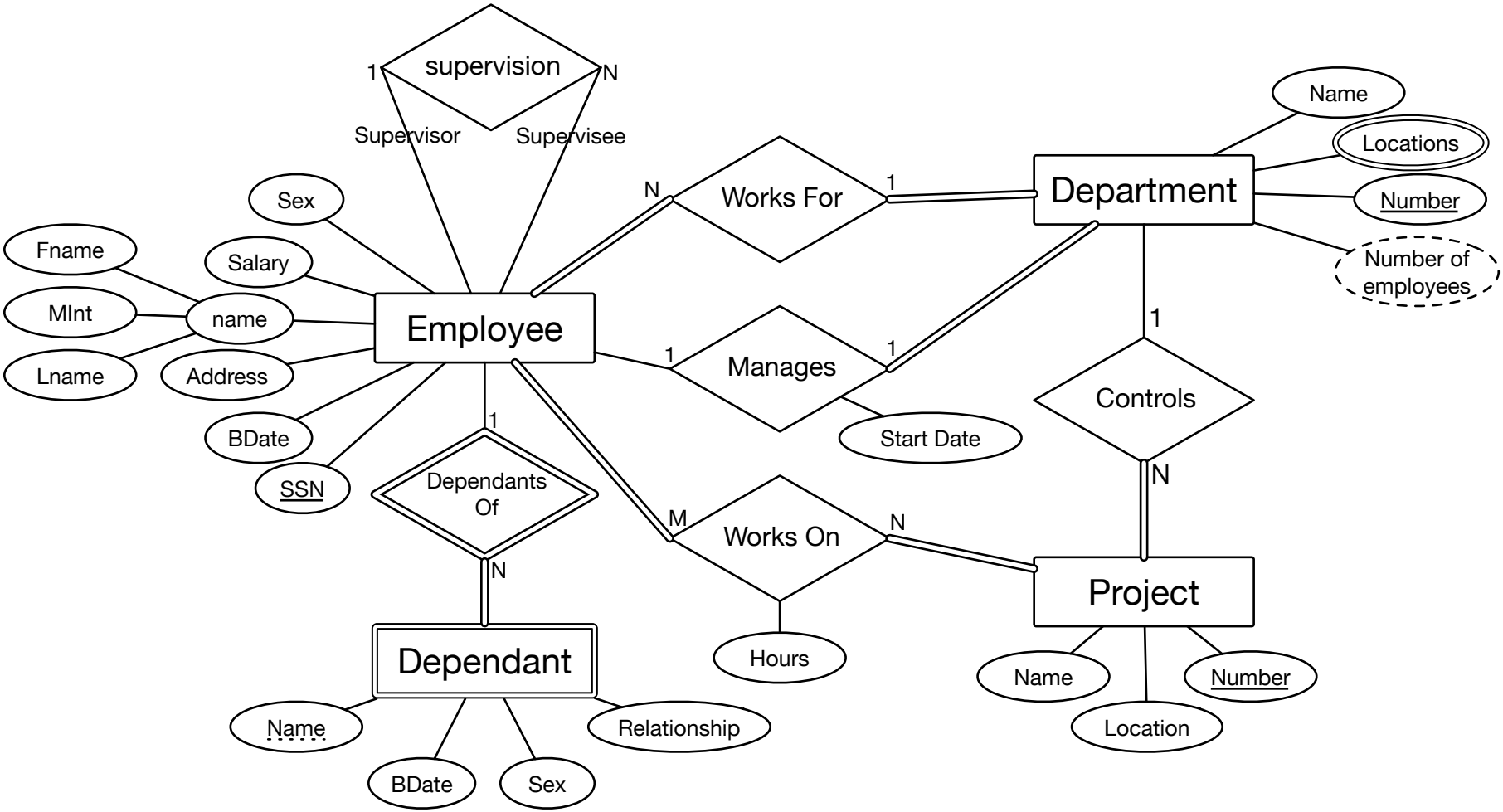
ESSN	Dependant_name	Sex	Bdate	Relationship

- Essn is a foreign key to the "Ssn" attribute in EMPLOYEE (and so we show it in italics).
- The primary key of DEPENDENT is the combination of its owner's primary key (Essn) and its own partial key (Dependent_name)

- To map a 1:1 relationship, there are 3 approaches to choose from. The first is most common
  1. **Foreign key approach**: Choose one of the entity types, and include the primary key of the other as a foreign key
  2. **Merge the relations**: if both entity types have mandatory participation in the relationship, they can be merged into one relation
  3. **Relationship relation**: Create a separate table to represent the relationship. This is rare for 1:1 relationships but very common for M:N relationships, as we will see later

Foreign Key Approach

- Foreign key approach: Choose one of the entity types, and include the primary key of the other as a foreign key.
- It is best to include the foreign key in an entity type that has mandatory participation in the relationship (avoids storing many NULL values).
- If the relationship has any simple attributes, they can be included in the same relation as the foreign key.


supervision
1
N
Supervisor
Supervisee
Sex
Fname
Salary
MInt
name
Lname
Address
BDate
SSN
Employee
N
Works For
1
Department
Name
Locations
Number
Number of employees
1
Manages
1
Start Date
1
Controls
N
1
Dependants Of
N
M
Works On
N
Project
Hours
Dependant
Name
BDate
Sex
Relationship
Name
Number
Location

Department

Dname	Dnumber	*Mgr_SSN*	Mgr_start_date

- Department has mandatory participation in the Manages relationship
- Therefore it makes most sense to include a foreign key in Department that refers to the primary key of Employee
- If we instead included a foreign key in Employee that referred to Department, then this would be NULL for all the employees that are not managers: because we wish to avoid too many NULL values, it's better to include the foreign key in Department

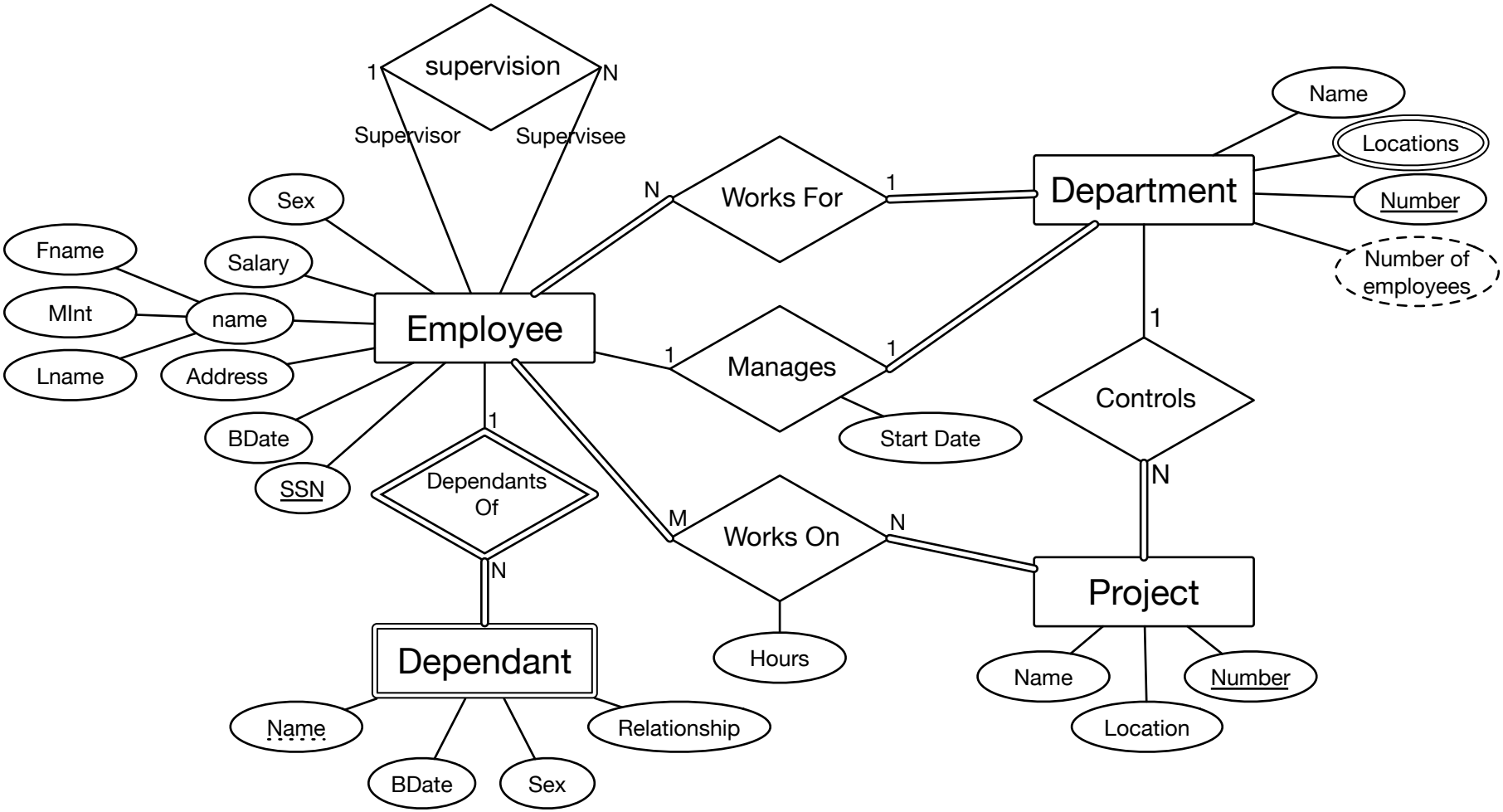
Department

Dname	Dnumber	*Mgr_SSN*	Mgr_start_date

- The `start_date` attribute of the `Manages` relationship can also be included as an attribute

- Because the participation is mandatory, we would use a NOT NULL constraint on the foreign key when we turn this into a database

- In the entity type on the N side, include a foreign key that refers to the primary key of the entity on the 1 side
- If the relationship has simple attributes, these can also be included in the same relation
- Alternatively, we could use a separate table to store the relationship, but again this is rare for 1:N relationships.
- It might be more useful if only a few tuples participate in the relationship, to avoid many NULL values.


supervision
1
N
Supervisor
Supervisee
Sex
Fname
Salary
MInt
name
Lname
Address
BDate
SSN
Employee
N
Works For
1
Department
Name
Locations
Number
Number of employees
1
Manages
1
Start Date
1
Controls
N
1
Dependants Of
N
M
Works On
N
Project
Hours
Dependant
Name
BDate
Sex
Relationship
Name
Number
Location

Employee

Fname	Minit	Lname	SSN	Bdate	Address	Salary	Sex	*Dnum*	*Super_SSN*

Project

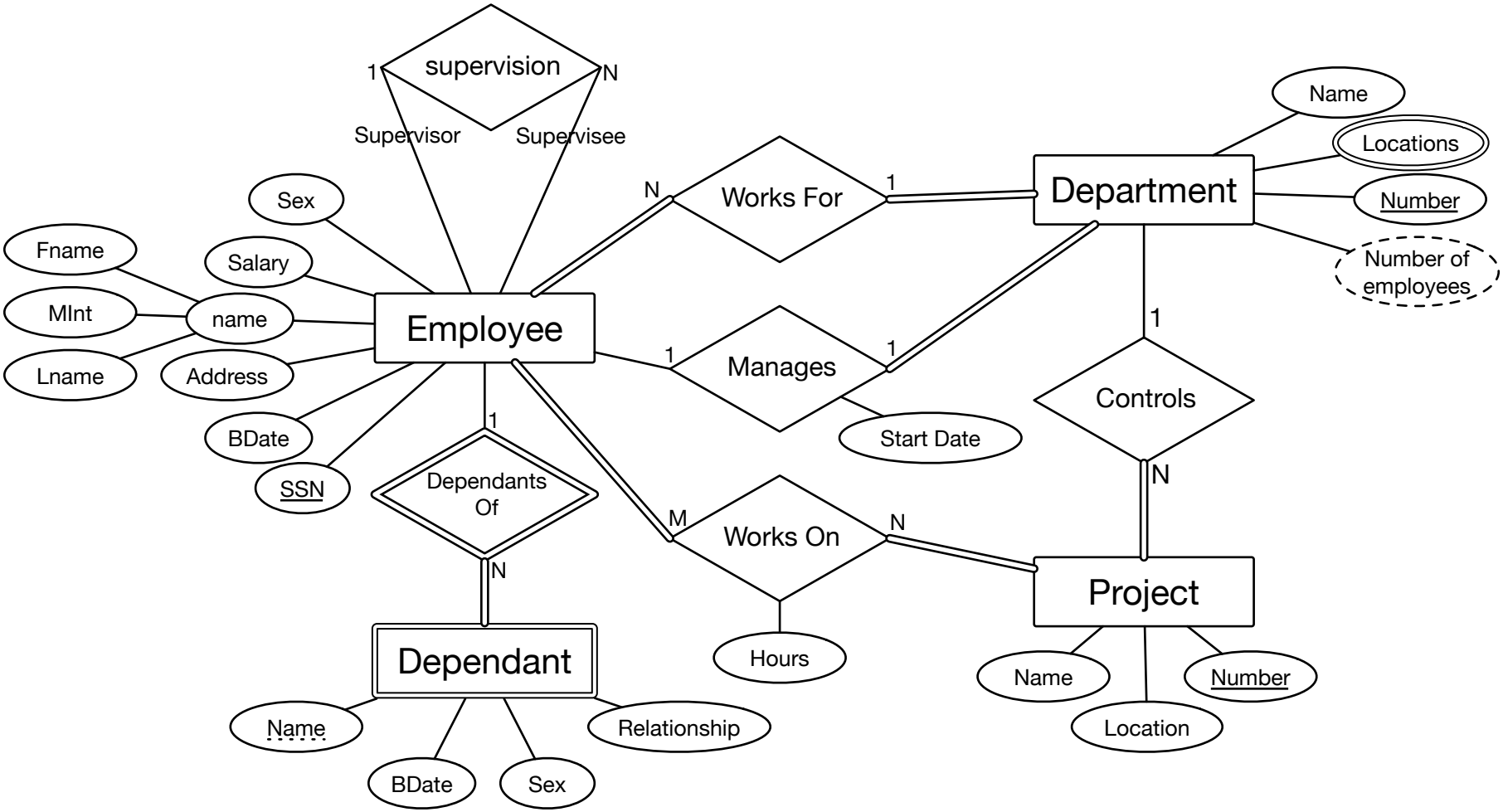
Pname	Pnumber	Plocation	*Dnum*

- In the `Employee` relation, we include the primary key of `Department` to show the department they work for as a foreign key
- In the `Employee` relation, we include the `Super_ssn` attribute to be a foreign key that refers to the `Ssn` attribute of each employee's supervisor (who is also an employee). As this is an optional relationship, it can be NULL for employees that don't have a supervisor.
- In the `Project` relation, we include the foreign key of `Department` to show which department controls a project. Again, this will be NOT NULL because every project must be controlled by a department.

- For a M:N relationship, it is not possible to simply include a foreign key in one of the entity types involved
  - A foreign key can only store one value, which would mean that each entity can only be related to one other entity. But here, all entities can be linked to many other entities
- Instead, we must create a new relation that represents the relationship
- We can also use this approach for 1:1 and 1:N relationships, but this is rarely done in practice

Relation to represent Relationship

- Include foreign keys to refer to both of the entity types involved
- The primary key of this table will be the combination of both of these foreign keys
- If the relationship has any simple attribute, include them in this new relation

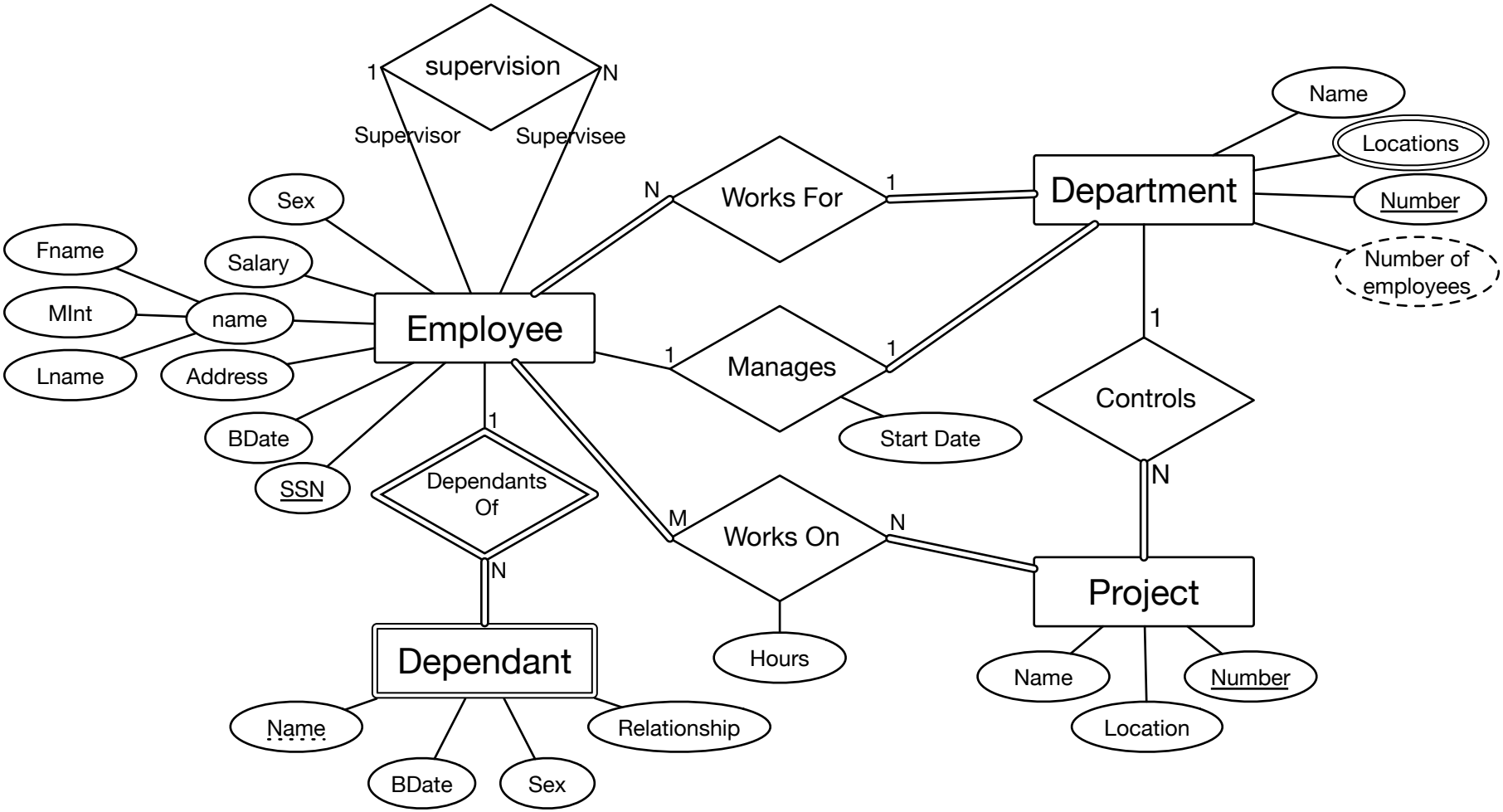

supervision
1
N
Supervisor
Supervisee
Sex
Fname
Salary
MInt
name
Lname
Address
BDate
SSN
Employee
N
Works For
1
Department
Name
Locations
Number
Number of employees
1
Manages
1
Start Date
1
Controls
N
1
Dependants Of
N
Dependant
Name
BDate
Sex
Relationship
M
Works On
N
Hours
Project
Name
Number
Location

`works_on`

ESSN	*Pnum*	Hours

- The `works_on` relationship involves an `Employee` and a `Project`, so we include foreign keys that refer to both of these.
- The primary key of `works_on` is a combined primary key that includes both of the foreign keys.
- The attribute `Hours` is also added to this relation.
- Because `works_on` depends on both, we would use a CASCADE option for referential integrity

- For each multi-valued attribute, create a new relation to represent it
  - If the multi-valued attribute is a composite attribute, we include only its simple components
- This new relation contains:
  - An attribute to store the multi-valued attribute itself (or several attributes if it was a composite attribute)
  - A foreign key that refers to the entity type that the attribute belongs to.
- The primary key of this relation is the combination of the foreign key and the attribute itself.


supervision
1
N
Supervisor
Supervisee
Sex
Fname
Salary
MInt
name
Lname
Address
BDate
SSN
Employee
N
Works For
1
Department
Name
Locations
Number
Number of employees
1
Manages
1
Start Date
1
Controls
N
1
Dependants Of
N
Dependant
Name
BDate
Sex
Relationship
M
Works On
N
Hours
Project
Name
Number
Location

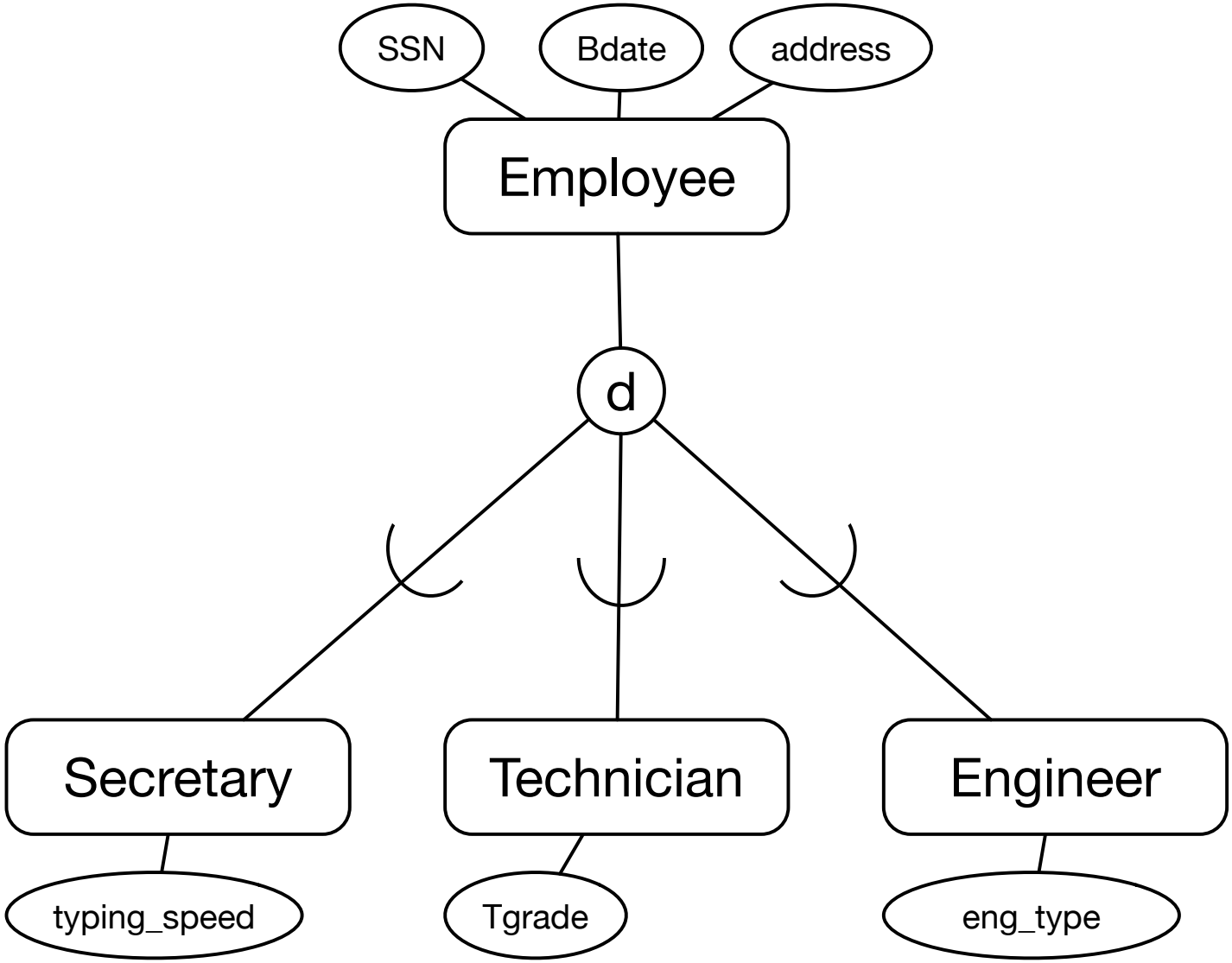
Dept_locations

Dnumber	Dlocation

- Department has a multivalued attribute named locations, which we represent as a new relation
- This includes an attribute Dlocation to store the location itself
- It also includes a foreign key Dnumber that refers to the department this location belongs to
- The primary key of Dept_locations is the combination of both of these
- Again we would use a CASCADE option because the department location depends on the existence of the department it belongs to

- If your model includes any relationships with more than 2 entity types, these must also be mapped to a new relation
- Include foreign keys to refer to all the participating entity types
- Include any attributes that the relationship has
- The primary key of this relation is usually the combination of the foreign keys

- There are a number of different options we can use for this mapping.
- The most general is:
  - Create a relation for the supertype.
  - For each subtype, create a relation and include a foreign key that refers to the supertype's primary key.
  - This foreign key attribute becomes the primary key of the subtype


SSN
Bdate
address
Employee
d
Secretary
Technician
Engineer
typing_speed
Tgrade
eng_type

Employee

Fname	Minit	Lname	SSN	Bdate	Address	Salary	Sex	*Dno*	*Super_SSN*

Secretary

SSN	typing_speed

Technician

SSN	Tgrade

Engineer

SSN	Eng_type

Summary

- After creating an ER/EER diagram to represent a data model, the next step is to map this to a relational schema that can be stored in a database
- By following the 8-step algorithm above, this can be achieved using a well-defined method
- At this stage, we can also think about the constraints that we can use in the final database