# Chapter 7:
# Data modeling using the Entity-Relationship (ER) model

# Last time: Three levels of data models

- Conceptual (high-level)
  - for end users / analysts
- Physical (low-level)
  - for programmers
- In between: Representational (implementation) data model
- Today: conceptual model:

  *Entity-relationship (ER) model*

# Entity-Relationship Model

- conceptual data model
- **ER-model** is used to create a *conceptual (database) schema*
  - *Popular high-level model*
  - *Database design tools employ ER concepts*
- Independent of the *representational* model (and of the kind of DBMS)
- The schema will be translated into a *logical schema* (e.g., relational schema, MS-access tables)
- The diagrammatic notation associated with the ER model is known as **ER diagrams**

# Steps of design and implementation

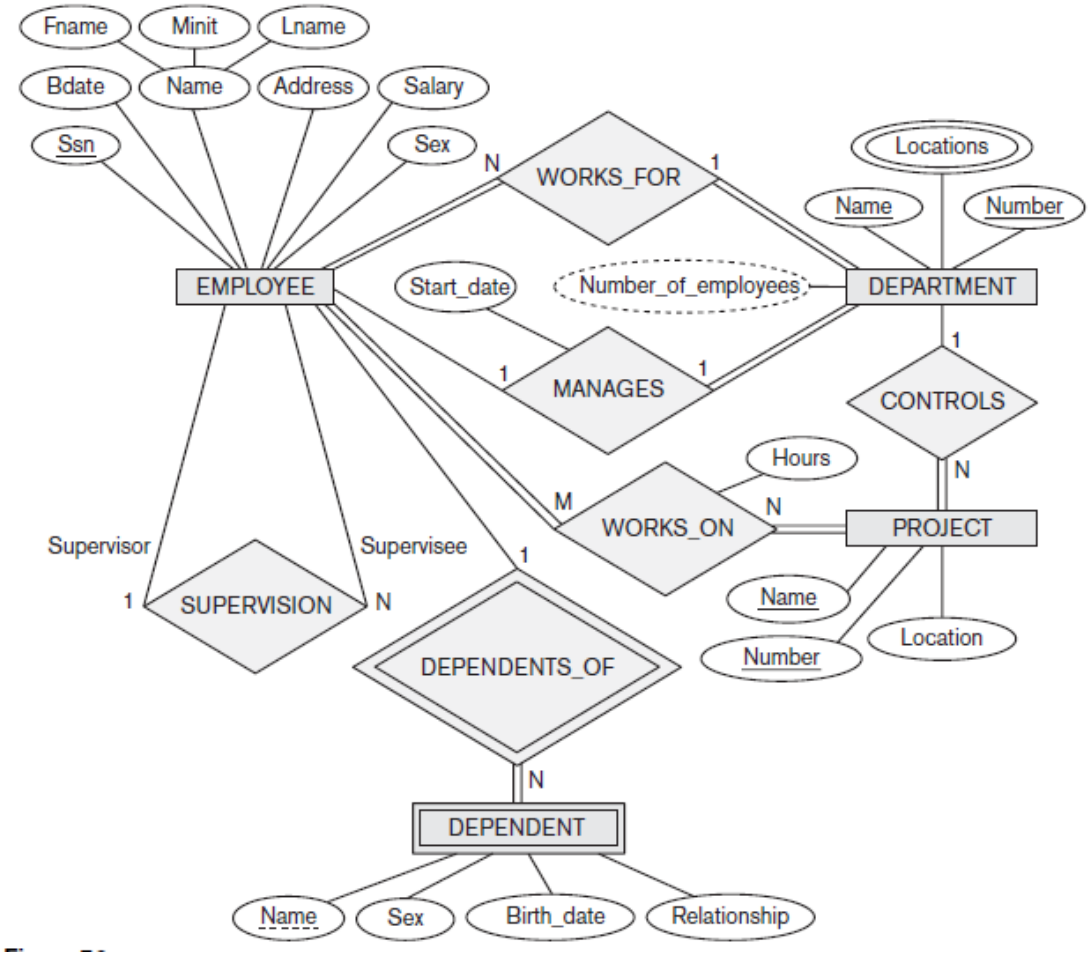
- Once requirements are gathered, the conceptual schema is built
  - This does not include technical details but user requirements mapped to entity types, relationships and constraints
- Next step is the implementation using a DBMS
  - (semi)automated procedures offered for converting conceptual schemas to database schemas (logical design or data model mapping)
- Last step is the physical design phase
  - internal storage structures, file organizations, indexes, access paths, and physical design parameters for the database files are specified
  - application programs are designed and implemented as database transactions corresponding to the high level transaction specifications.

# What we will learn today…

Suppose designers, after long meetings with all users, come up with the following scenario and rules (description of the **miniworld**):

- The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
- A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
- We store each employee's name, Social Security number, address, salary, gender, and birth date. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. We keep track of the current number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee (who is another employee).
- We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent's first name, gender, birth date, and relationship to the employee.

# What we will learn today…



# ER model

- Relevant information from the mini world is described in terms of:
  - Entities
  - Attributes
  - Relationships

# Entities

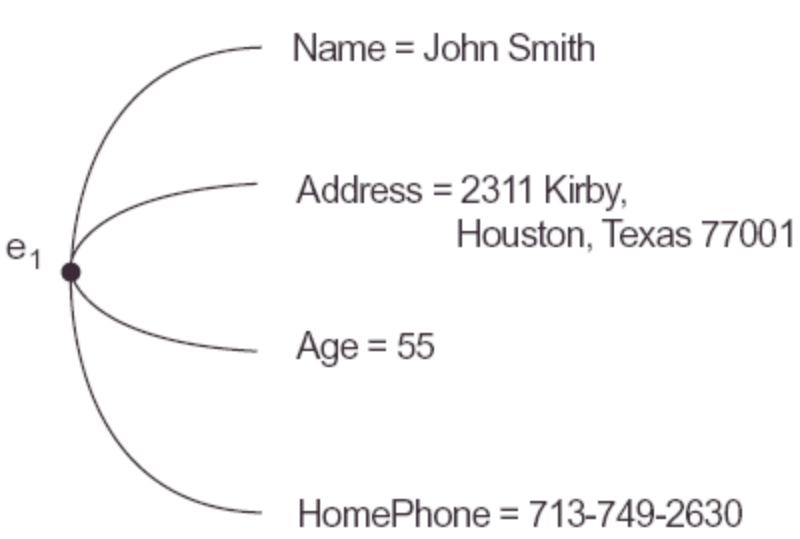
- An entity in the ER model is the representation of a “thing” in the real world that exists on its own.
  - Tangibles: a *person, a house, a car*
  - Intangibles : a *department, a job, a course*

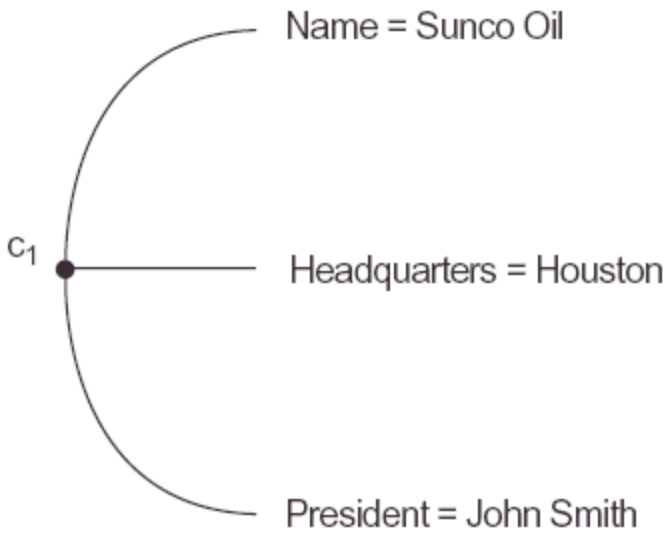
# Attributes

- each entity has *attributes* that describe the characteristics of that specific entity

- A particular entity (instance) has *values* for all its attributes

# Entities and attributes



# Kinds of attributes

- Composite
  - an attribute that can be split:
    - address = “parkstraat 12a, 5243 GE, Hoogestande”
    - “parkstraat” - “12” - “a” - “5243 GE” - “Hoogestande”
    - attribute-hierarchy
- Atomic
  - cannot be split
    - age = 23

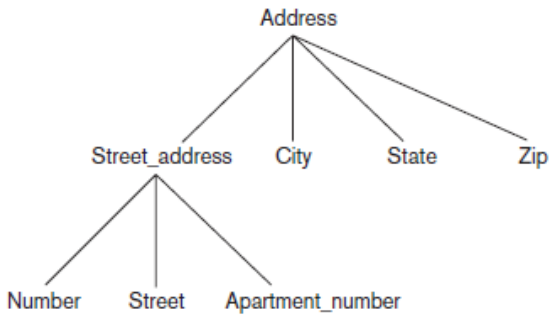



**Figure 7.4**
A hierarchy of composite attributes.

# Kinds of attributes

- Single-valued versus multi-valued
  - age = 23 (single-valued)
  - Phone.nr = 043-2235541 and 06-2355534
  - hobby = swimming and piano and party
- Stored versus derived
  - age derived from date of birth
- Complex attributes:
  multi-valued and composite

# Null-values

- An attribute can have a special value *null* (or *nil*). The meaning of the value depends on the context.
- Possible meanings of null:
  - *Not applicable / cannot be determined*
  - *Should be known but is missing*

# Entity-Types

- In a database there exist groups of *similar* entities:
  - having the same attributes
  - having different values for the attributes

- An **entity-type** represents a *collection of entities with the same attributes*.

# Entity-Set

- The set of entities of a certain entity-type that occur at a given moment in a database is the *entity-set*.
- *entity-type* = intension
- *entity-set* = extension

# Entity-Types

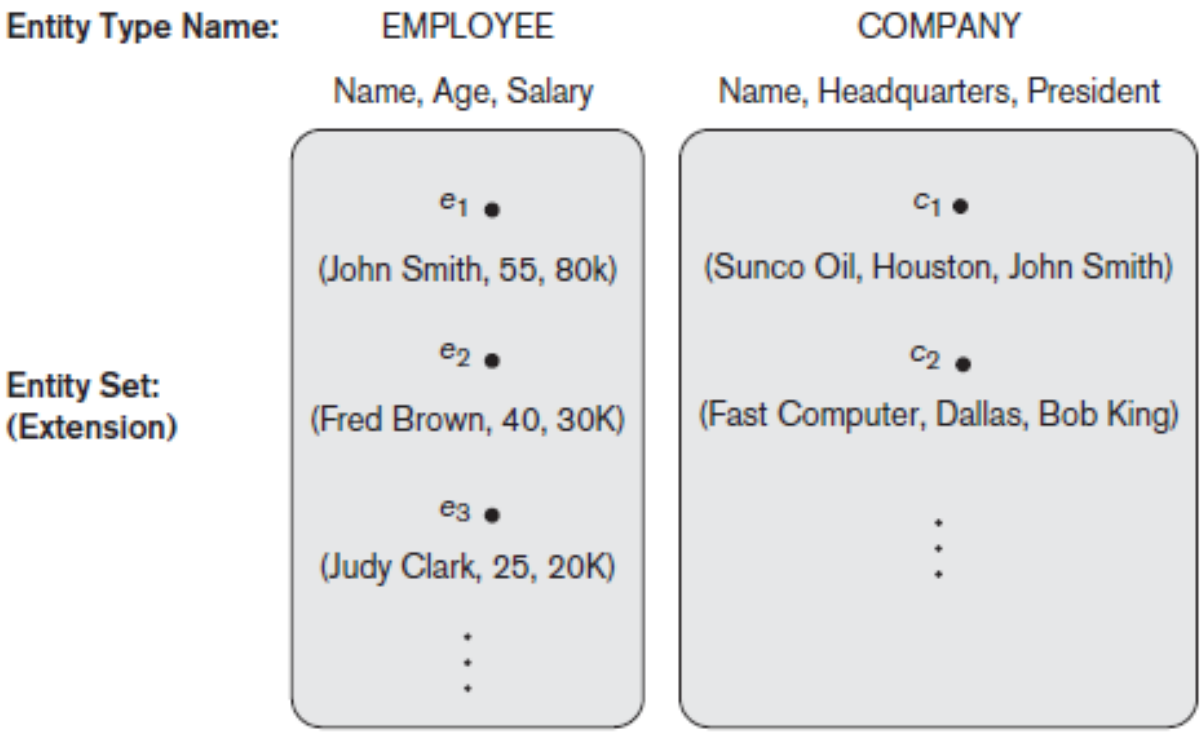


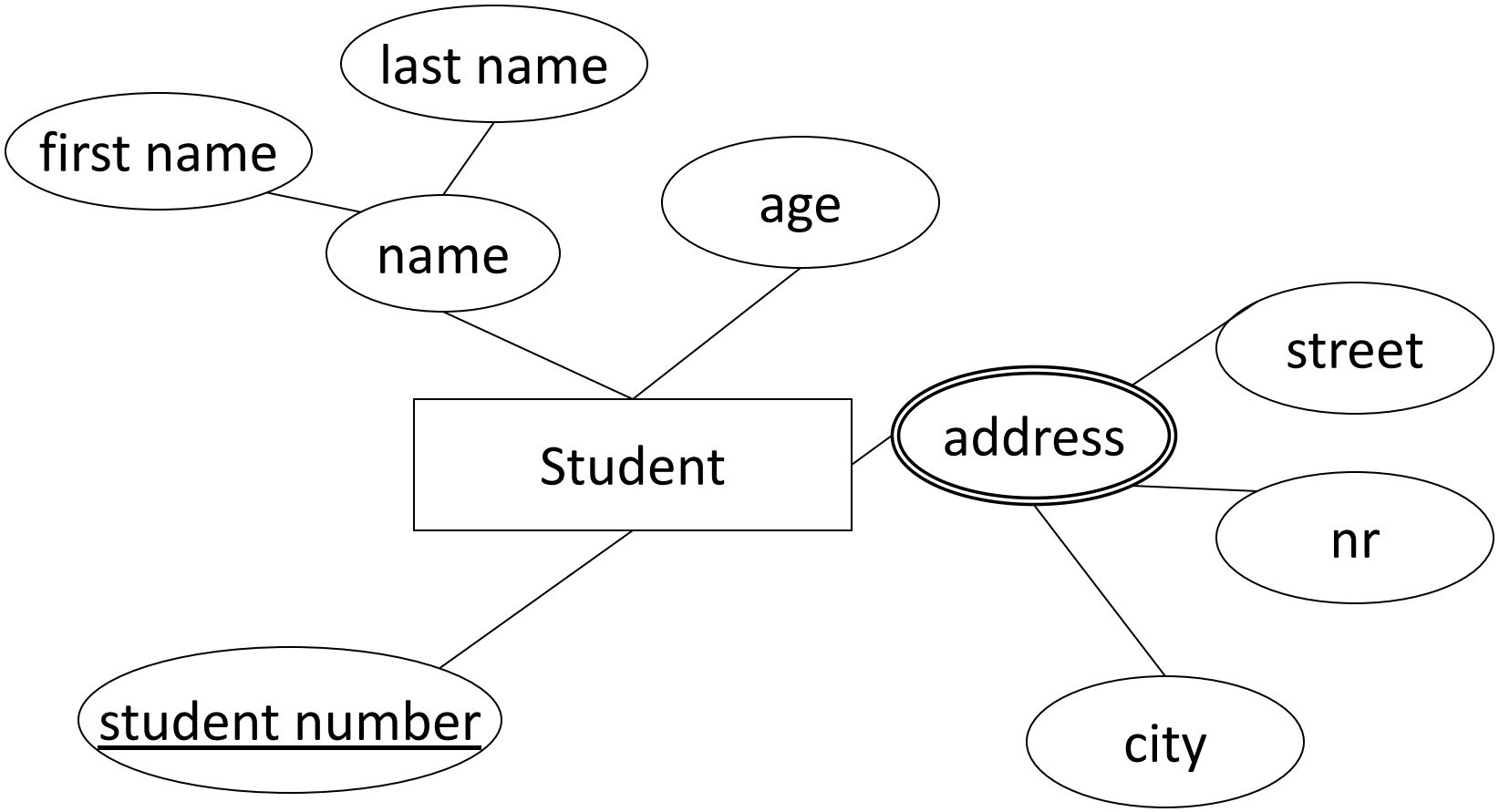

**Figure 7.6**
Two entity types, EMPLOYEE and COMPANY, and some member entities of each.

# Entity-Types Representation

- In an **ER-schema** entity-types are determined by a unique *name* and a set of *attribute-names.*

- In an **ER-diagram** an entity-type is represented by a rectangle, containing the name, to which ellipses are connected, containing the attribute names. (multivalued: double ellipses)

# Entity-Type


last name
first name
name
age
Student
address
street
nr
student number
city

# Key attribute

- A **key attribute** is an attribute that *is unique* for each entity in all possibly occurring entity-sets *(uniqueness constraint)*
- For example, the Name attribute is a key of the COMPANY entity type. For the PERSON entity type, a typical key attribute is Ssn
- Sometimes several attributes together form a key, meaning that the combination of the attribute values must be distinct for each entity.
  - The best way to represent this in an ER model is to define a composite attribute and designate it as a key attribute for the entity type
  - Such a composite attribute must be minimal
  - In ER diagrammatic notation, each key attribute has its name underlined inside the oval

# Key attributes

- An entity-type can have more than one key (separate keys)
- An entity-type can also have no keys (i.e. weak entity-type).
- There is no concept of primary key, unlike in relational models (in ER diagrams, we can have more than one key)

# Attribute value set (domain)

- Every atomic attribute is associated with a value domain
  - natural numbers
  - [16,67]
  - texts of up to 50 characters
  - yes/no
- This is not represented in the ER-diagram

# Initial Conceptual design of the COMPANY database (slide 5)

- An entity type DEPARTMENT with attributes Name, Number, Locations, Manager, and Manager_start_date.
- Locations is the only multivalued attribute.
- We can specify that both Name and Number are (separate) key attributes because each was specified to be unique.

# Initial Conceptual design of the COMPANY database

- An entity type PROJECT with attributes Name, Number, Location, and Controlling_department.
- Both Name and Number are (separate) key attributes.

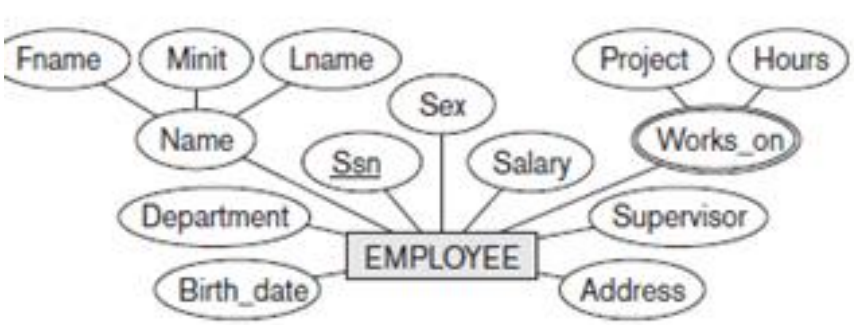
# Initial Conceptual design of the COMPANY database

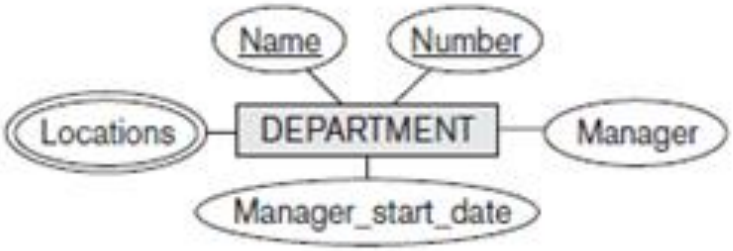
- An entity type EMPLOYEE with attributes Name, Ssn, Sex, Address, Salary, Birth_date, Department, and Supervisor.
- Both Name and Address may be composite attributes;
  - however, this was not specified in the requirements.
  - We must go back to the users

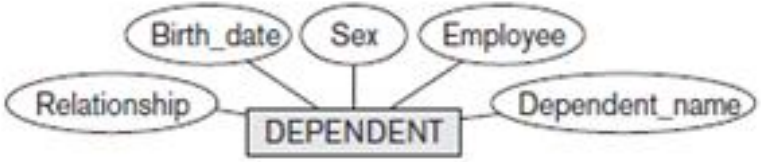
# Initial Conceptual design of the COMPANY database

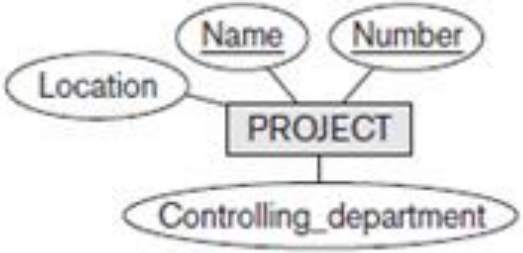
- An entity type DEPENDENT with attributes Employee, Dependent_name, Sex, Birth_date, and Relationship (to the employee).

# Initial Conceptual design of the COMPANY database



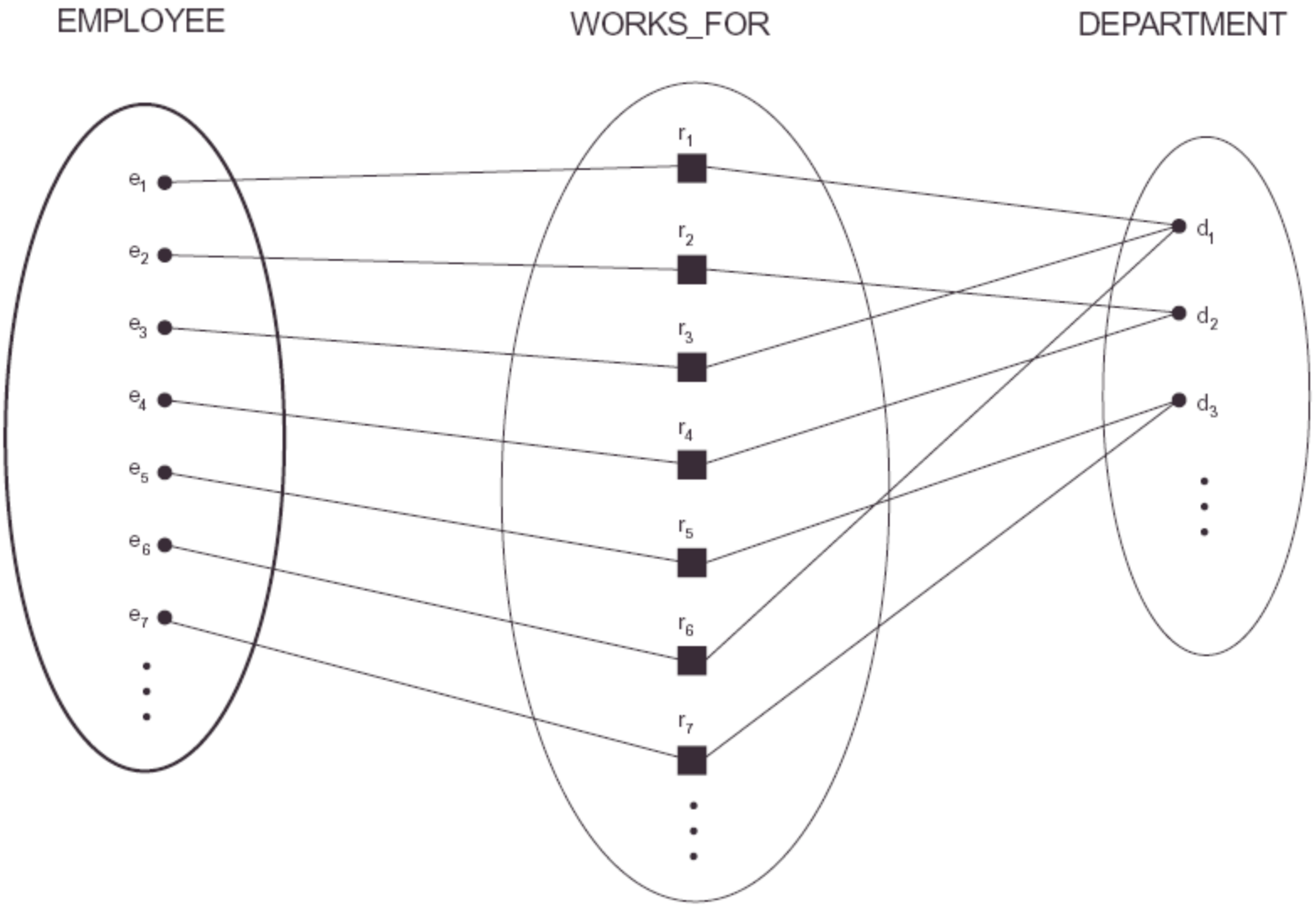



# Relationships

- A *relationship* is the representation of a relevant, existing association between two (or more) entities in the mini world.
  - Student i99883 follows ‘databases’
  - FHS is a faculty of the UM

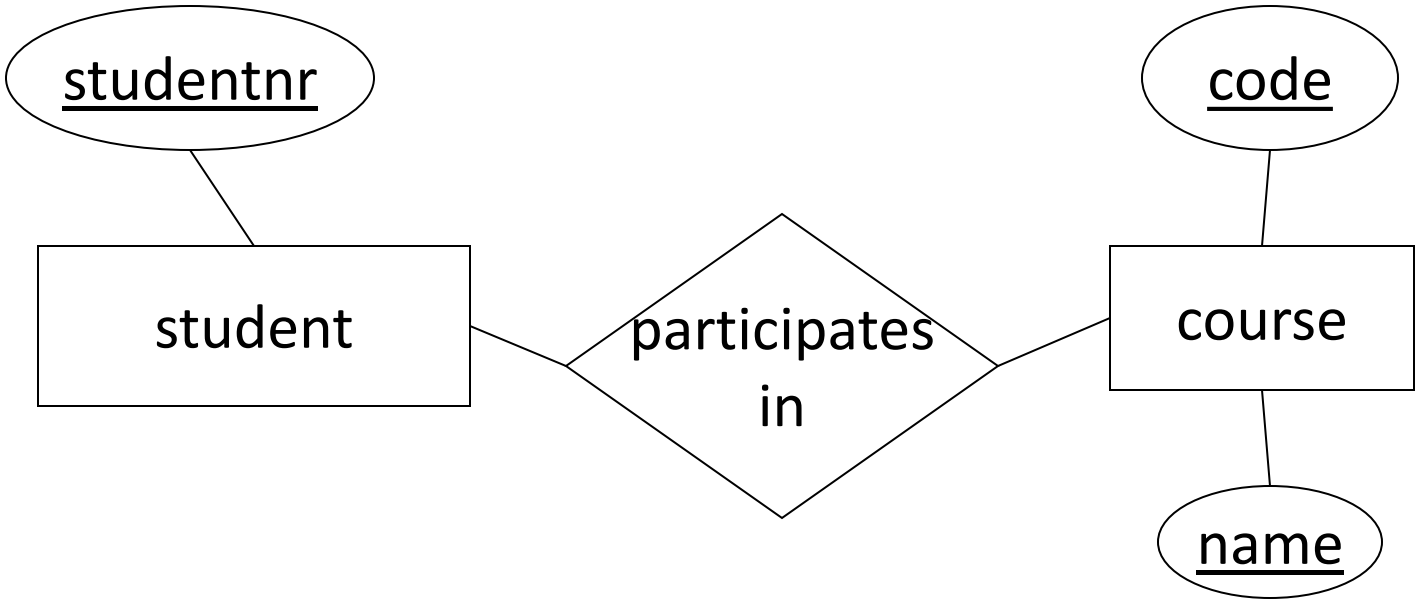

EMPLOYEE
WORKS_FOR
DEPARTMENT
e1
e2
e3
e4
e5
e6
e7
r1
r2
r3
r4
r5
r6
r7
d1
d2
d3

# Relationship-Types

- Each relation-type has a *name*
- examples:
  - Faculty **is-faculty-of** University
  - Student **participates-in** Course

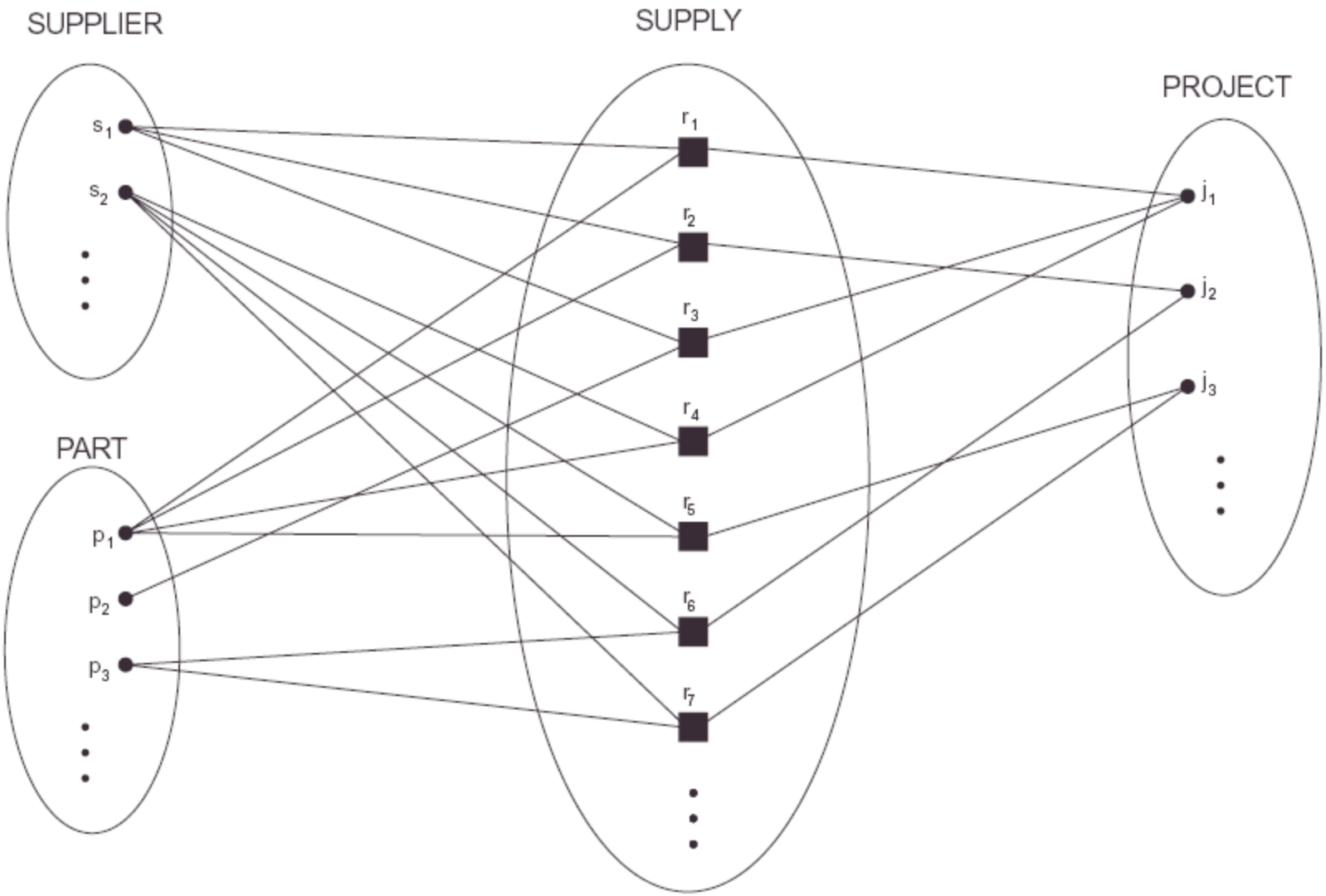
- In ER-diagrams relation-types are represented by a diamond-shape containing the name, connected to the participating entity-types

# Relation-Types



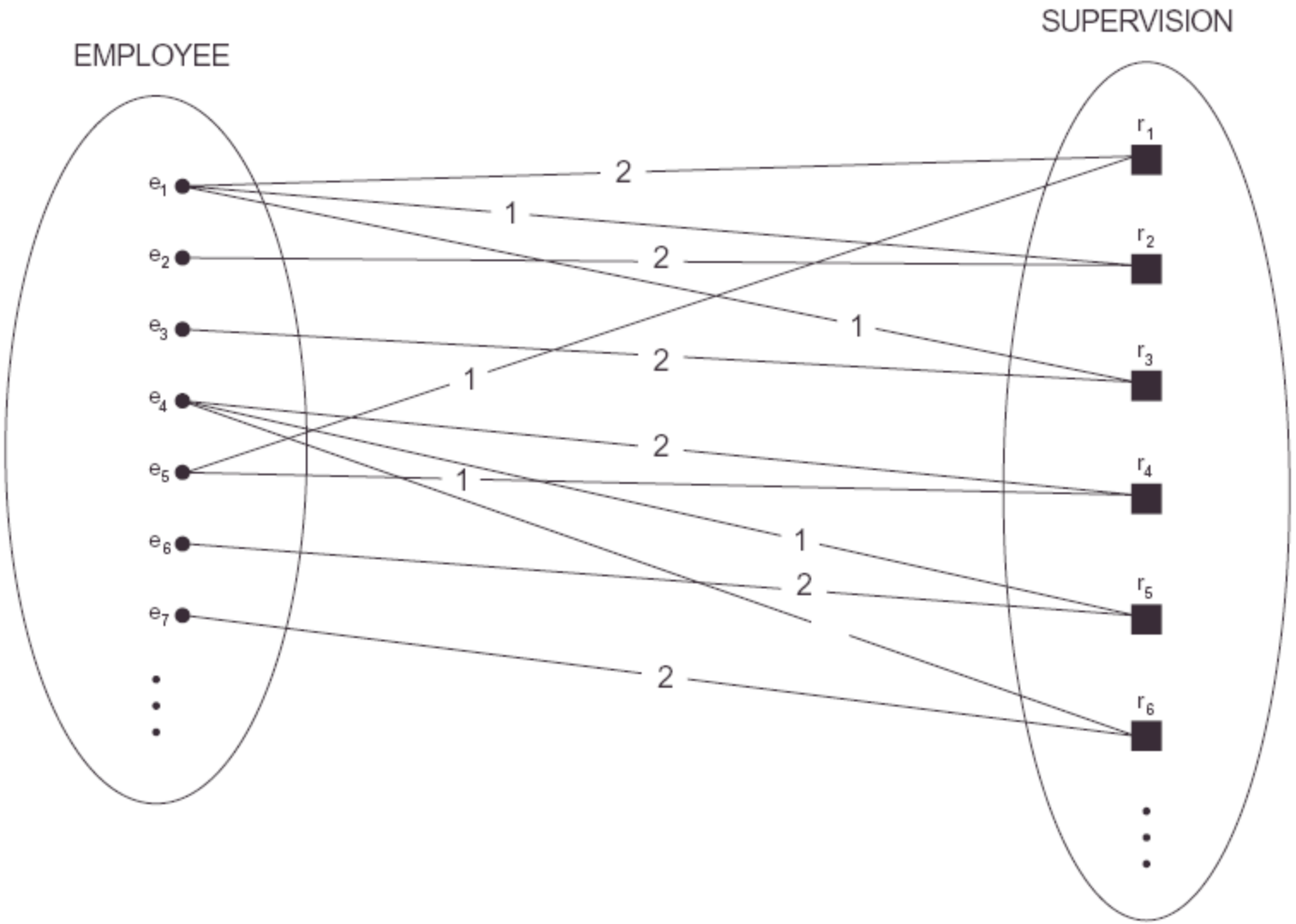
# Degree of relation-types

- **Degree** = number of participations by entity-types
  - binary = degree 2
  - ternary = degree 3
- Every degree (> 1) can occur
- In most cases the degree is 2


SUPPLIER
SUPPLY
PROJECT
PART
s1
s2
p1
p2
p3
r1
r2
r3
r4
r5
r6
r7
j1
j2
j3

# Recursive relationship-types

- **Roles**: each entity-type plays a *role* in a relation-type. This role can be mentioned explicitly in a diagram, although this is not necessary if the relationship links two separate entity types.
  - The role names can be the relationship types themselves
- For example, in the WORKS_FOR relationship type, EMPLOYEE plays the role of *employee* or *worker* and DEPARTMENT plays the role of *department* or *employer*.
- An entity-type can play multiple roles simultaneously in a relation-type:
  - recursive relation-type
- In that case it **must** be mentioned what the roles are of such an entity




Employee plays two roles: supervisor (2) and supervisee (1)

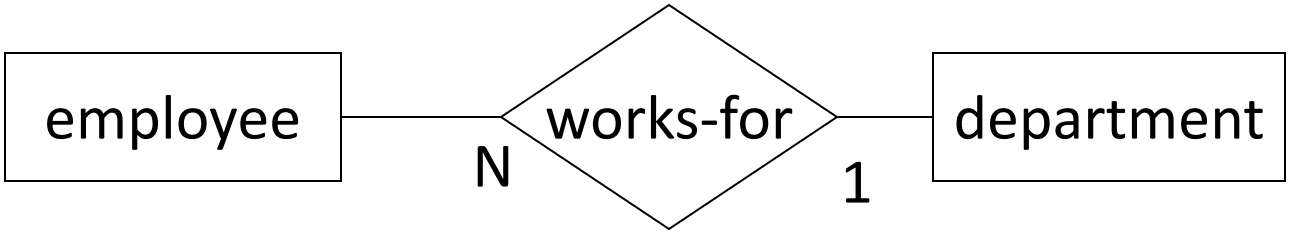
# Constraints on binary relationship types

- Relationship types usually have certain constraints that limit the possible combinations of entities that may participate in the corresponding relationship set.
- These constraints are determined from the miniworld situation that the relationships represent.
- For example,  if the company has a rule that each employee must work for exactly one department, then we would like to describe this constraint in the schema.
- We can distinguish two main types of binary relationship constraints:
  - **cardinality ratio**
  - **participation.**

# Cardinality ratio

- To how many relation-instances can each entity participate?
  - The ratio for university:faculty is 1:N
    (a faculty can only belong to one university, but a university can have more than one faculty)
- Possible ratios are: 1:1, 1:N, N:1, N:M
- These values are written next to the diamonds in the diagram

# Cardinality ratio




- Be aware: in some books the 1 and N are swapped!
- Always carefully check what these numbers mean.
- In our convention, we read preferentially from left-to-right
- or top-to-bottom.

  N does not mean a max. It means ‘any number’

# Participation constraints and existence dependencies

- An entity-type participates **totally** if every entity of that type participates in at least one relation-instance
- If not, the participation is **partial**
- Totality is indicated by a **double line** in the diagram

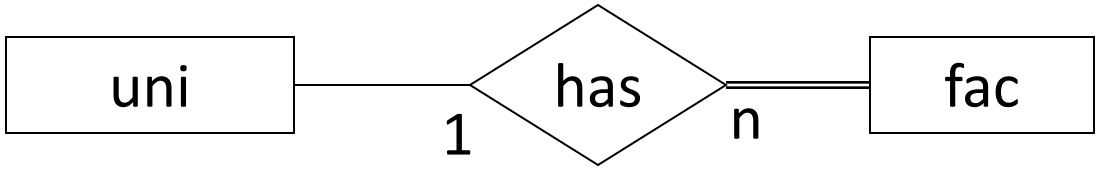
# Participation constraints and existence dependencies

Example:

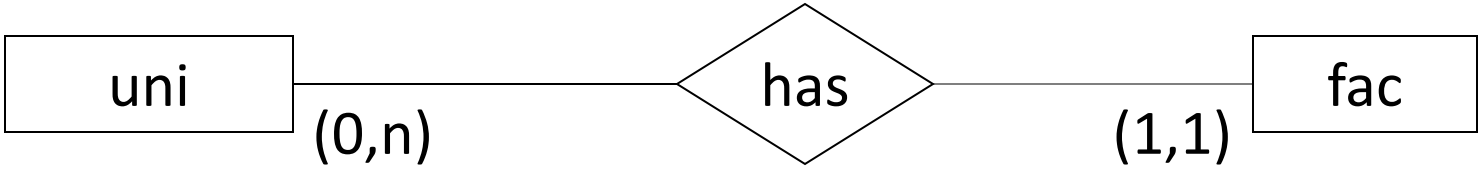
- An employee entity can exist only if it participates in at least one WORKS_FOR relationship instance.
  - the participation of EMPLOYEE in WORKS_FOR is called **total participation**, (*the total set* of employee entities must be related to a department entity via WORKS_FOR).
    - Total participation is also called **existence dependency**.
- We do not expect every employee to manage a department, so the participation of EMPLOYEE in the MANAGES relationship type is **partial**, meaning that *some* or *part of the set of* employee entities are related to some department entity via MANAGES, but not necessarily all

# Alternative Notation

**Cardinality and participation form the *structural restrictions on relation-types***




- Write at the entity-type the *minimum* and *maximum* number of participations per entity allowed:



# Attributes of relationship-types

- Also relation-types can have attributes
  - Student id122773 participates_in databases *in the academic year 2012/2013*
- Indicated by ellipses connected to the diamond
- There are no keys for relation-types
- In 1:1 or 1:N relations: the attributes can move to an entity-type

# Attributes of relationship-types

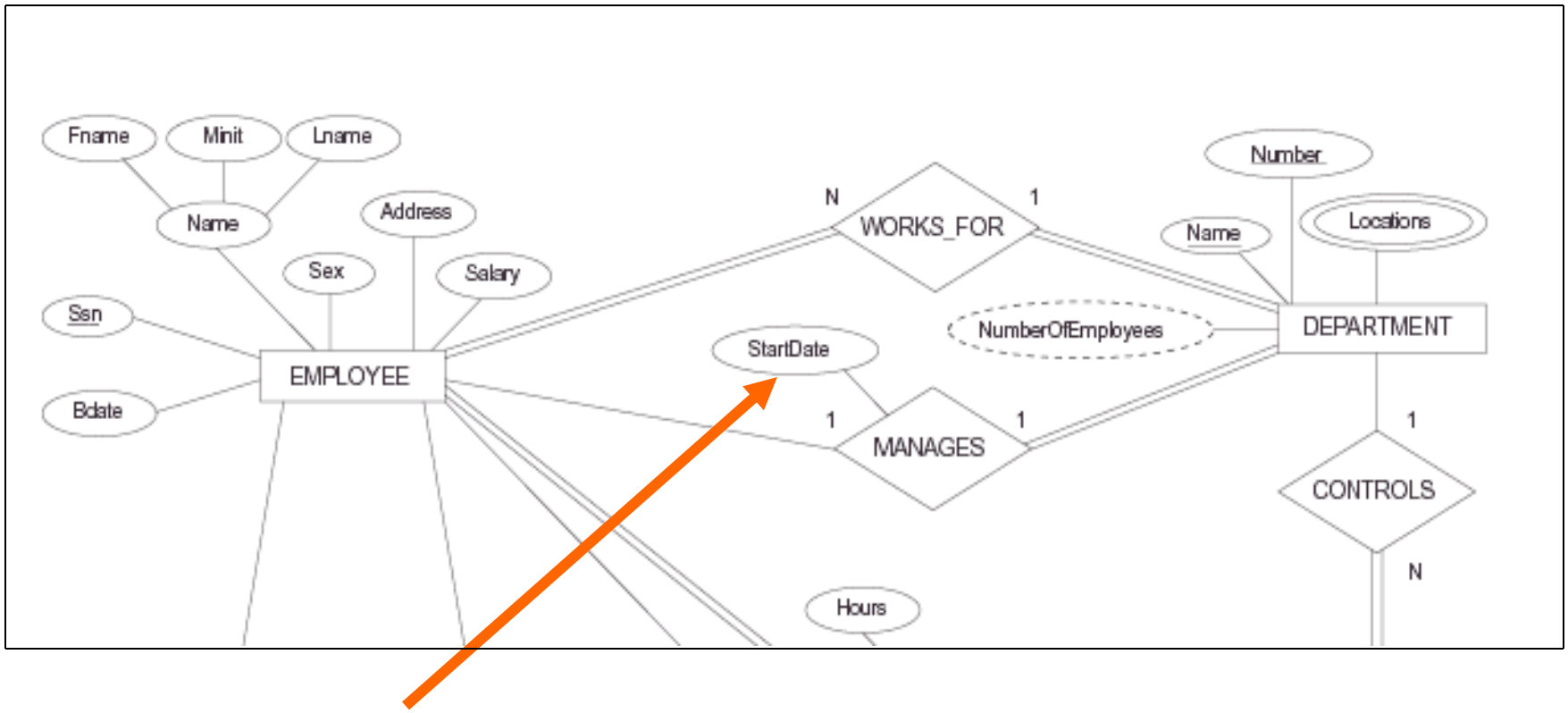
In a relationship STUDENT_STUDIES, suppose we want to have an attribute *start_date.*

- What is the cardinality ratio of the relationship?
- Can *start_date* move to the STUDENT or FACULTY entity type?

# Attributes of relationship-types

What happens in the 1:1 case?

What about the M:N?




May move to EMPLOYEE or to DEPARTMENT

# Weak entity-types

- The entities we examined so far, all have key attributes **(strong entity types)**
- An entity-type W without a key is called **weak entity type**
- Entities belonging to a weak entity type are identified by being related to specific entities from another entity type in combination with one of their attribute values (**identifying or owner entity type**).
- A weak entity type always has a *total participation constraint* (existence dependency) with respect to its identifying relationship

# Weak entity-types

- Consider the entity type DEPENDENT, related to EMPLOYEE (1:N relationship).
- The attributes of DEPENDENT are Name, Birth_date, Sex, and Relationship (to the employee).
- Two dependents of *two distinct employees* may, by chance, have the same values for Name, Birth_date, Sex, and Relationship, but they are still distinct entities.
- They are identified as distinct entities only after determining the *particular employee entity* to which each dependent is related. Each employee entity is said to *own* the dependent entities that are related to it.

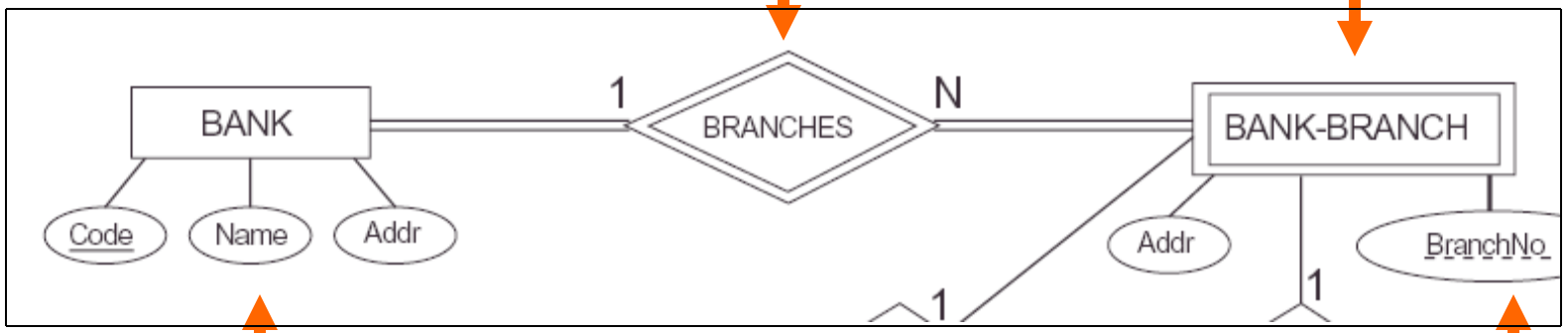
# Weak entity-types

- A weak entity type normally has a **partial key**, which is the attribute that can uniquely identify weak entities that are *related to the same owner entity*.

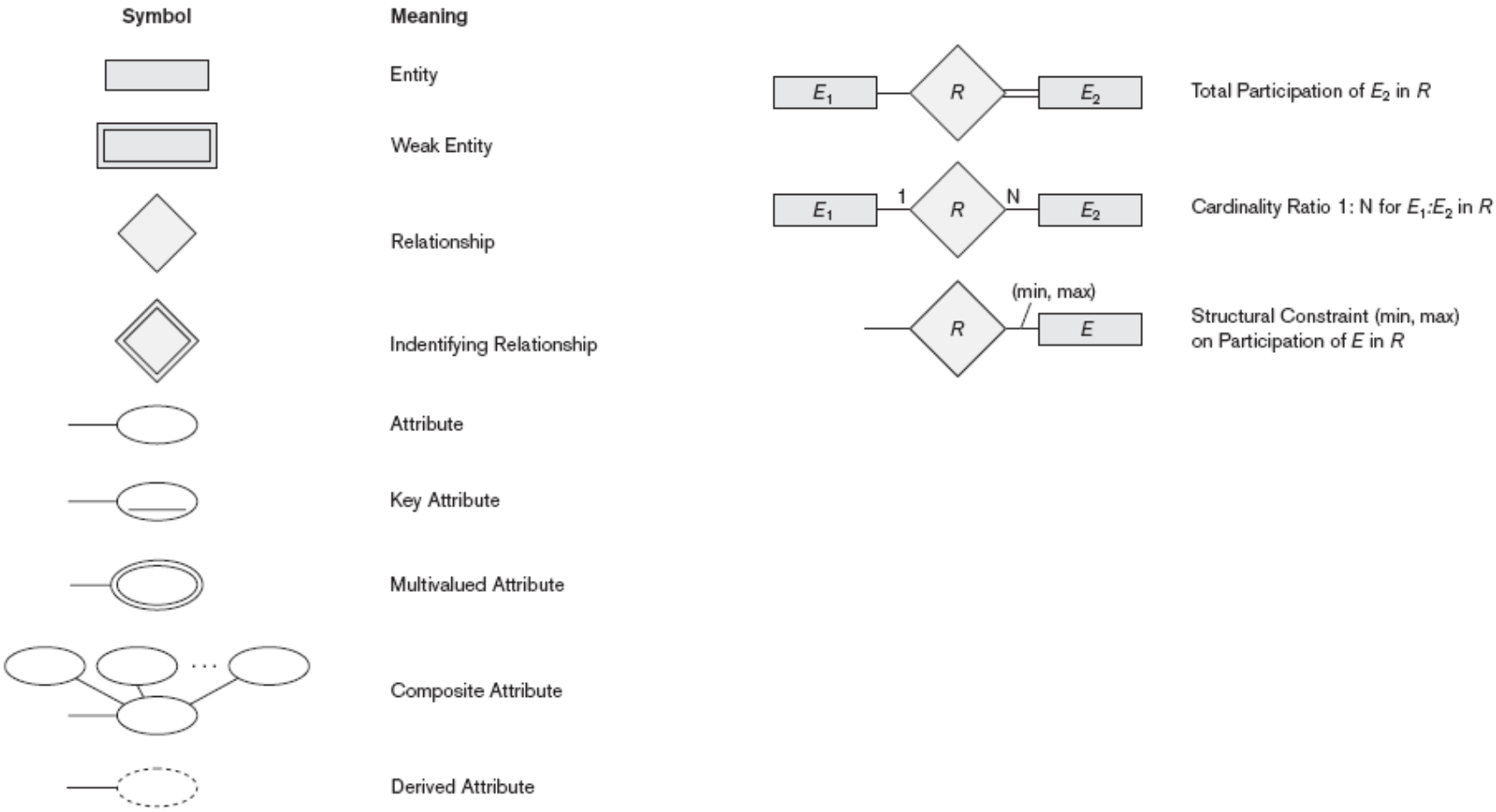
- If we assume that no two dependents of the same employee ever have the same first name, the attribute Name of DEPENDENT is the partial key (a composite attribute can also be the partial key, if necessary).

- In ER diagrams, weak entities and identifying relationships are distinguished by surrounding their boxes and diamonds with double lines. The partial key attribute is underlined with a dashed or dotted line.

# Weak entity-types

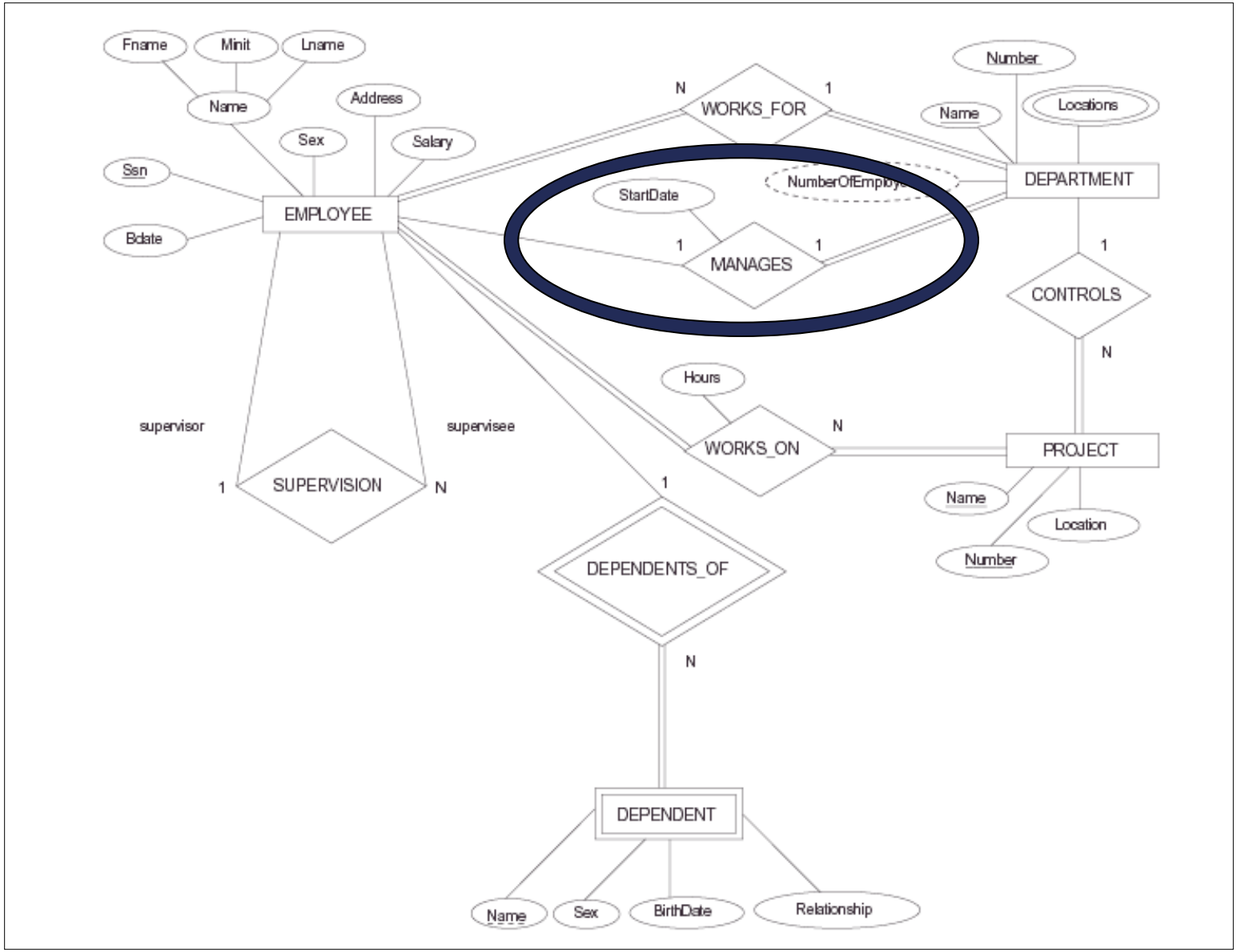


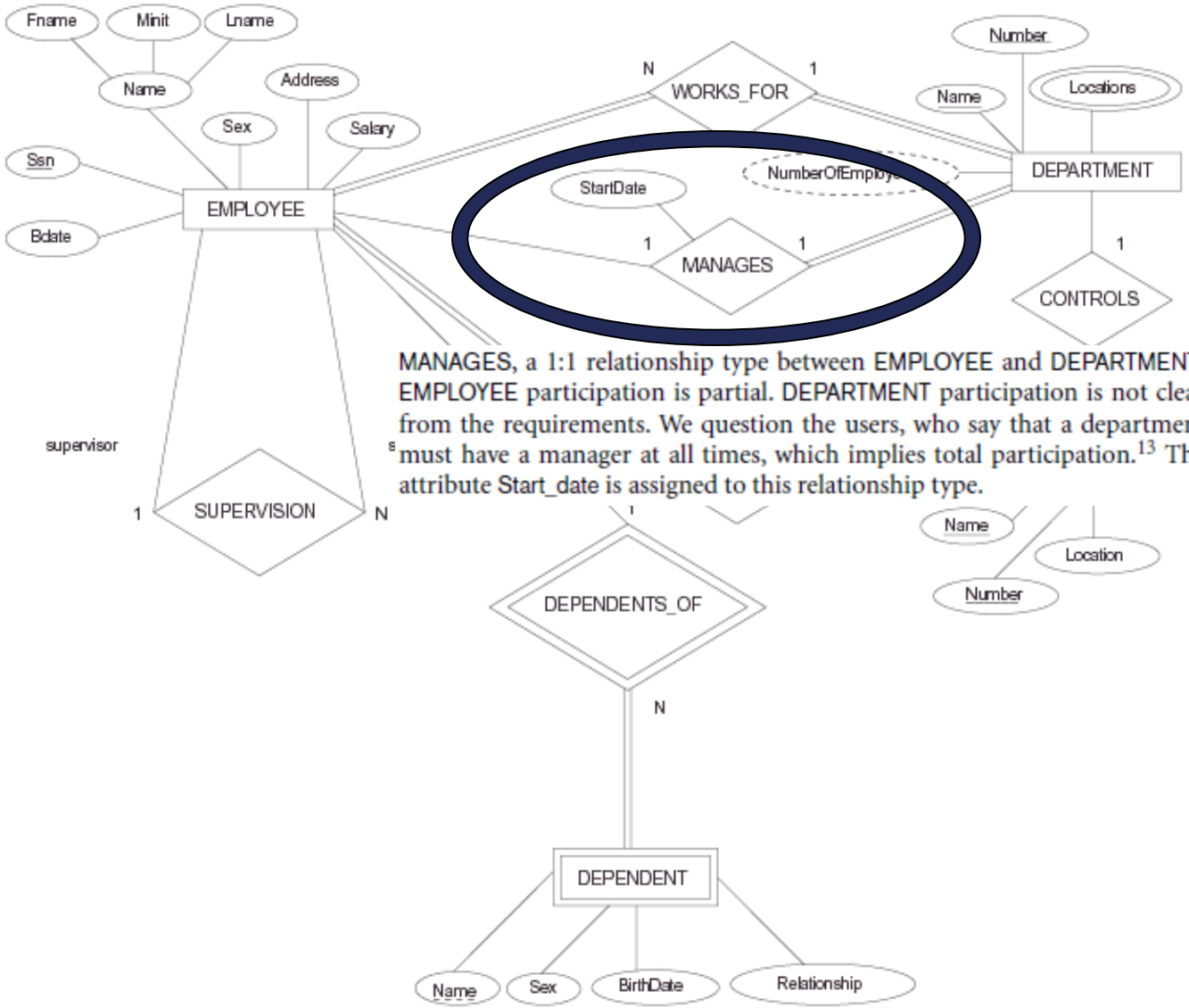
# Summary of notation in ER diagrams



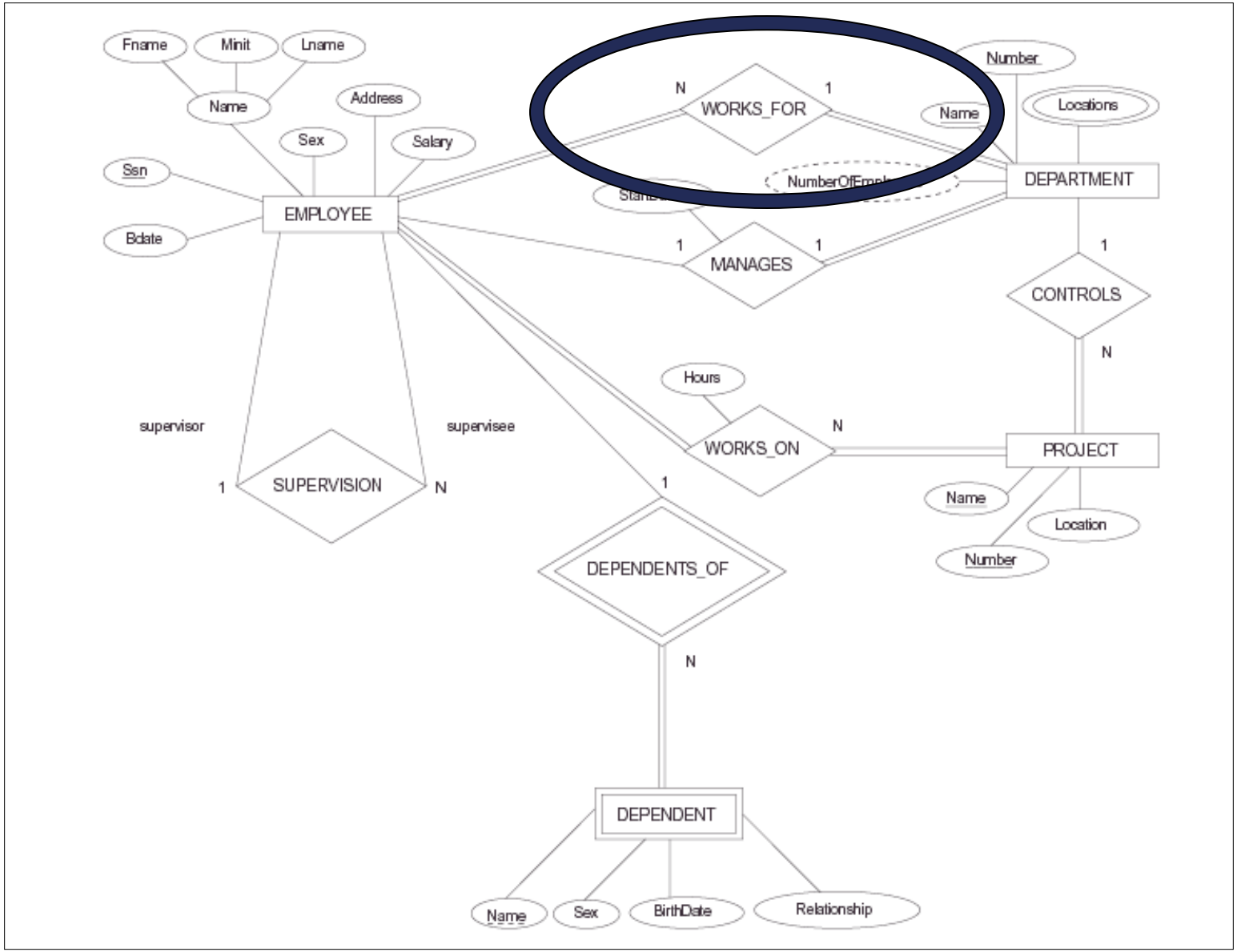
# Designing a complete ER-diagram

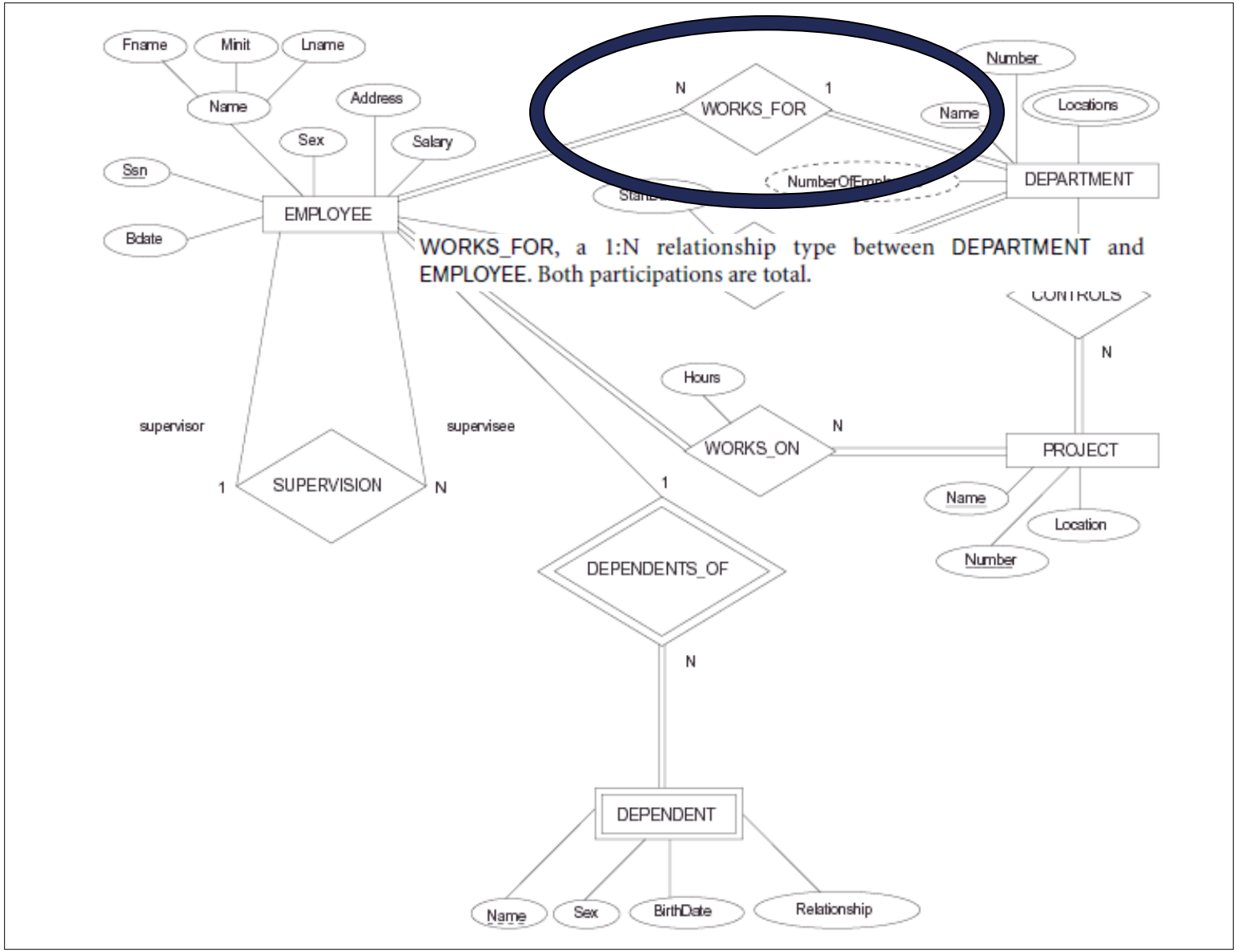
- Using these tools a complete ER-diagram for some task can be designed.
- It can be an iterative task: designing, going back to the users, re-designing, etc. Questions might be:
  - Should initial attributes better be modeled as relationship types?
  - What are the cardinalities of the relationship?
  - Should relationships be total?
  - Should entities be split into identifying and weak entities?
- This leads to a final ER-diagram, like the one given at the beginning of this lecture:

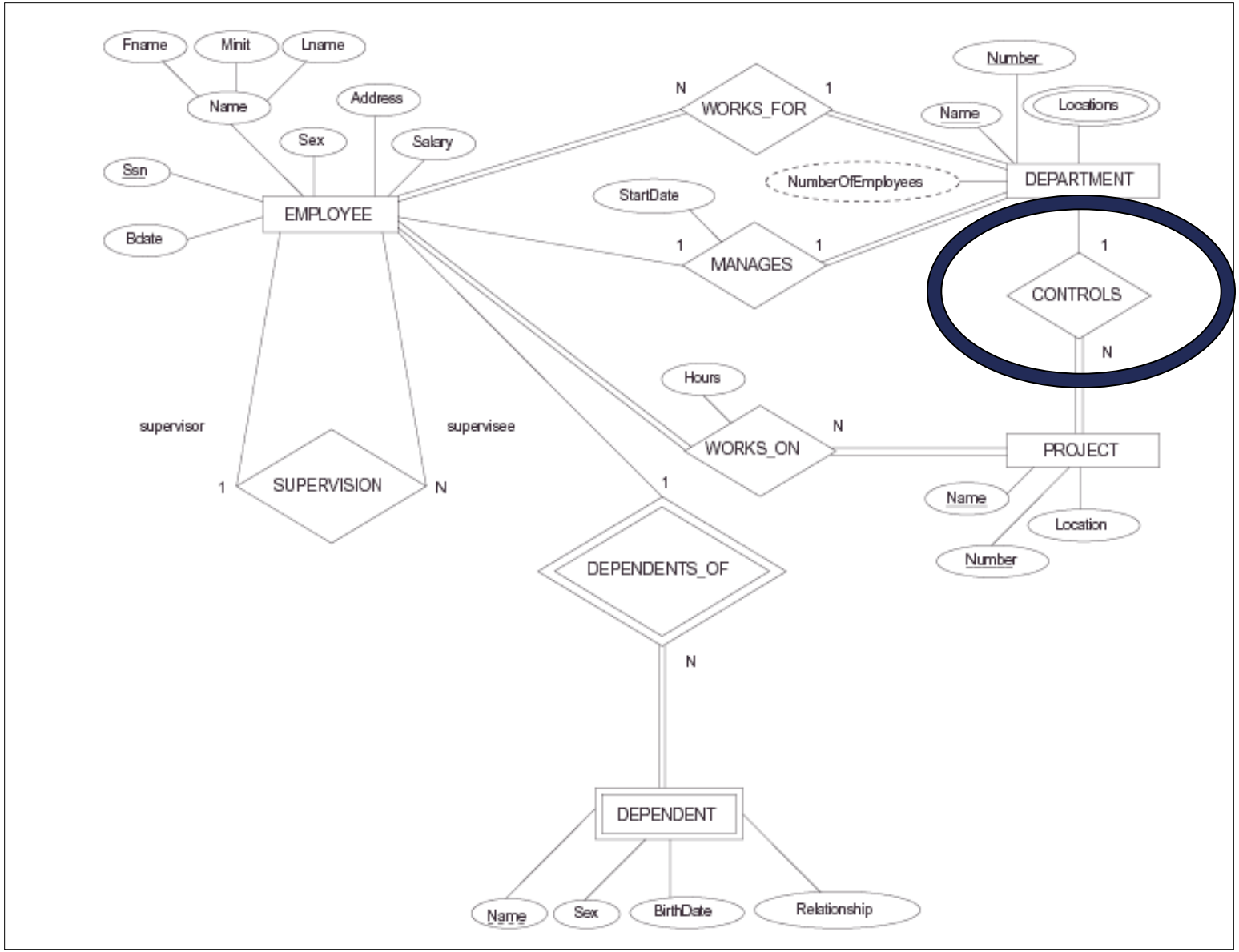

Fname
Minit
Lname
Name
Address
Sex
Salary
Ssn
Bdate
EMPLOYEE
N
WORKS_FOR
1
Number
Name
Locations
DEPARTMENT
StartDate
NumberOfEmployees
1
MANAGES
1
1
CONTROLS
N
Hours
supervisor
supervisee
WORKS_ON
N
PROJECT
1
SUPERVISION
N
Name
Location
Number
1
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
BirthDate
Relationship

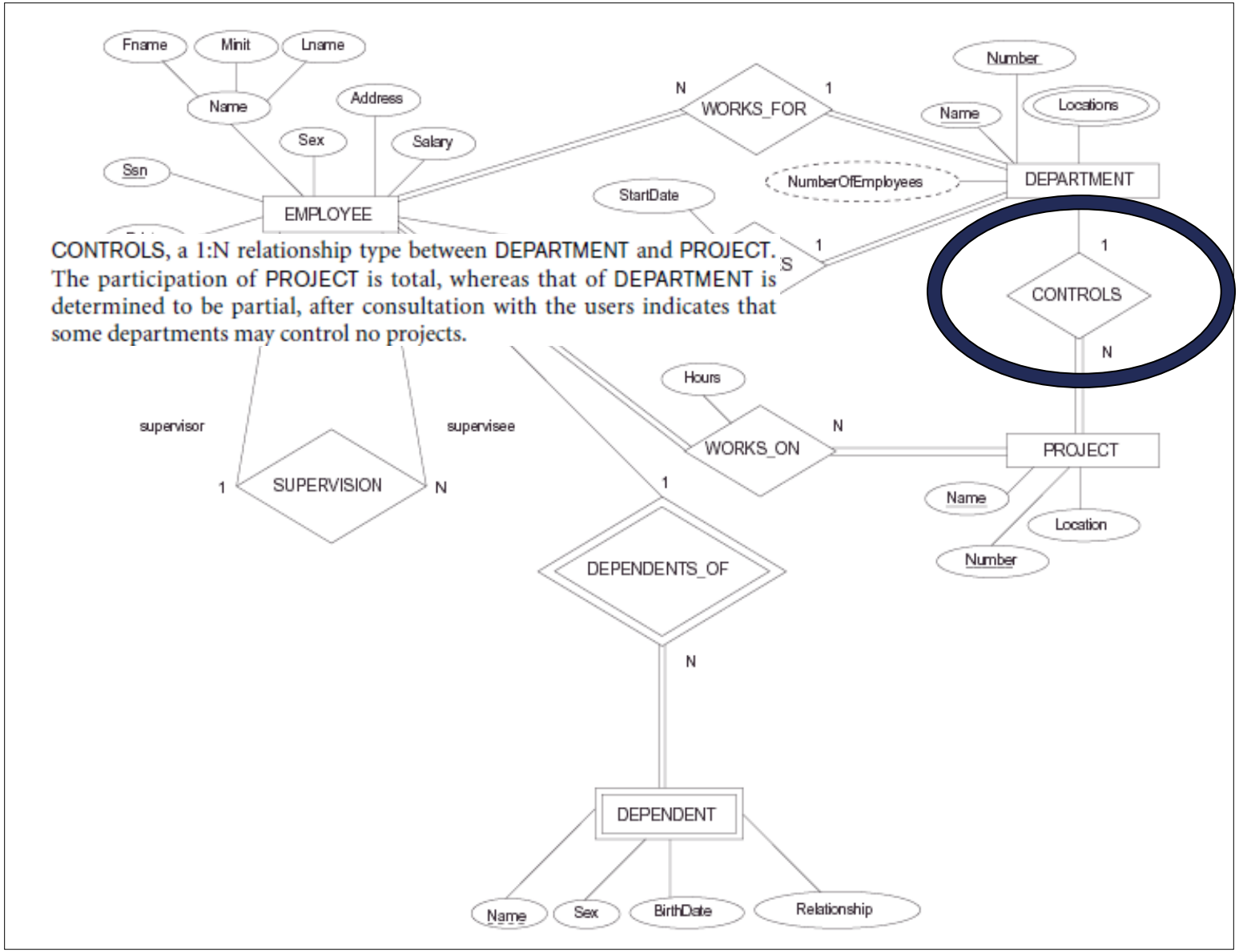


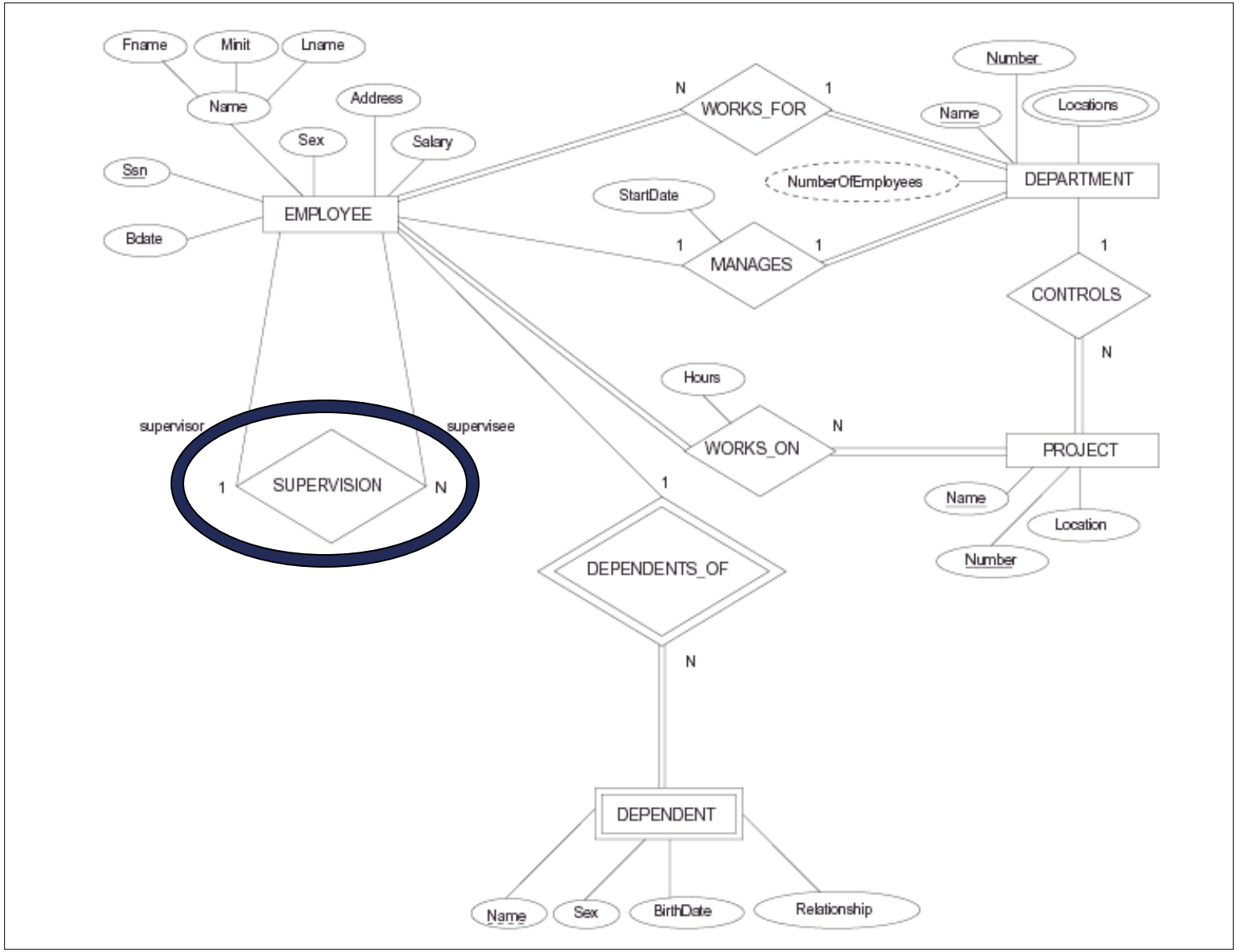

MANAGES, a 1:1 relationship type between EMPLOYEE and DEPARTMENT. EMPLOYEE participation is partial. DEPARTMENT participation is not clear from the requirements. We question the users, who say that a department must have a manager at all times, which implies total participation.[13] The attribute Start_date is assigned to this relationship type.

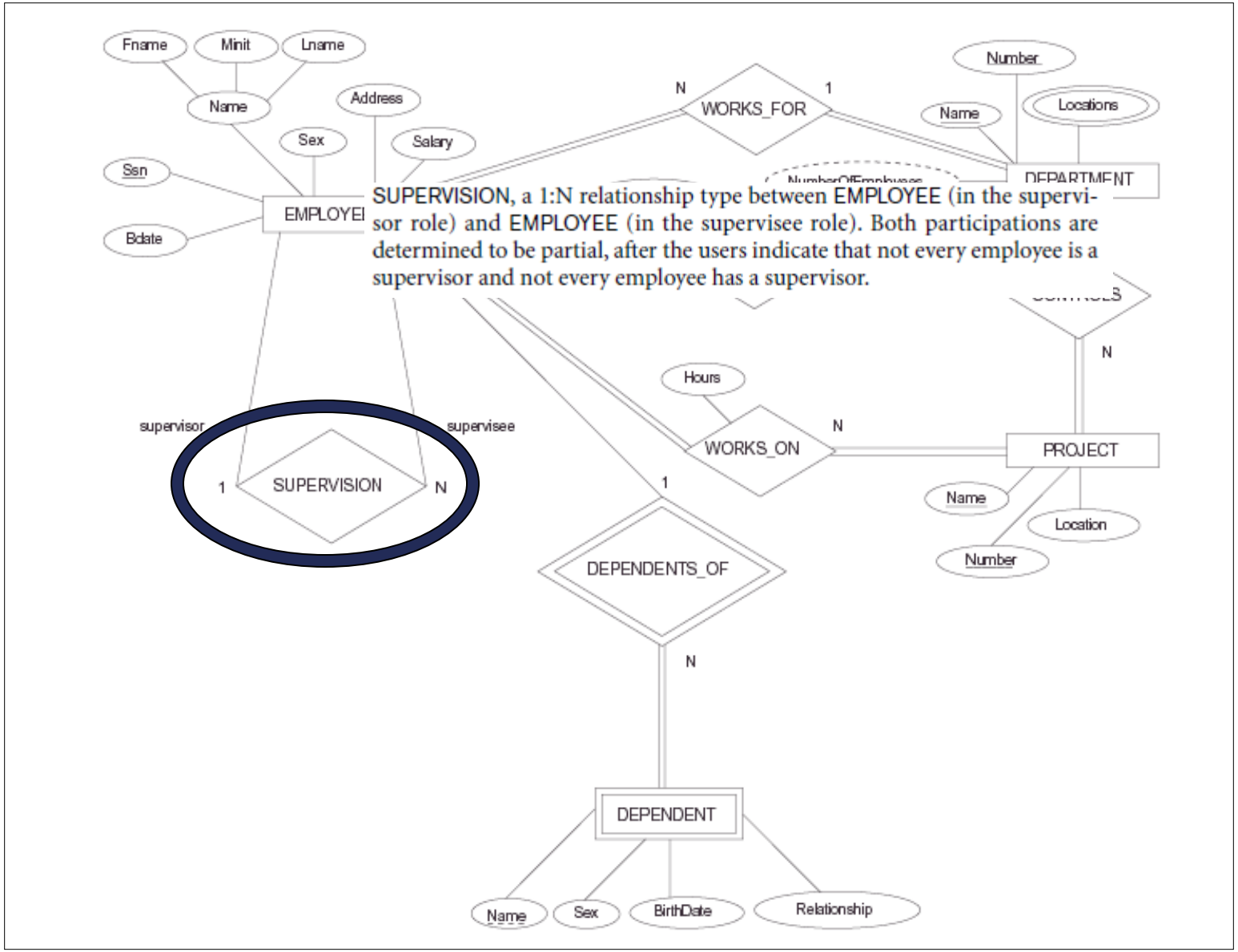

Fname
Minit
Lname
Name
Address
Sex
Salary
Ssn
Bdate
EMPLOYEE
N
WORKS_FOR
1
Number
Name
Locations
DEPARTMENT
NumberOfEmpl
1
MANAGES
1
1
CONTROLS
N
Hours
supervisor
supervisee
WORKS_ON
N
PROJECT
1
SUPERVISION
N
Name
Location
Number
1
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
BirthDate
Relationship

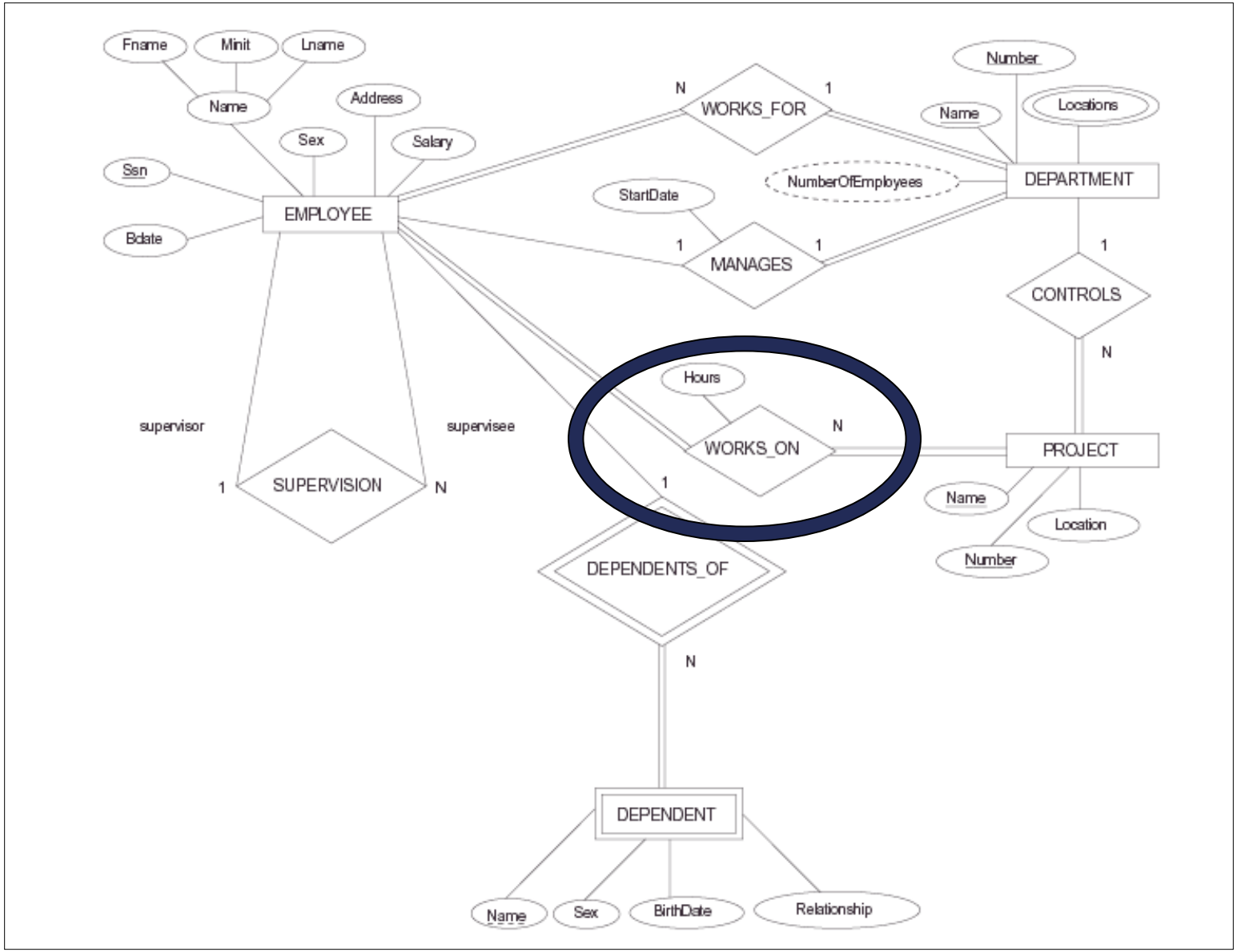

Fname
Minit
Lname
Name
Address
Sex
Salary
Ssn
Bdate
EMPLOYEE
N
WORKS_FOR
1
Number
Name
Locations
DEPARTMENT
WORKS_FOR, a 1:N relationship type between DEPARTMENT and EMPLOYEE. Both participations are total.
CONTROLS
N
Hours
supervisor
supervisee
WORKS_ON
N
PROJECT
1
SUPERVISION
N
Name
Location
Number
1
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
BirthDate
Relationship

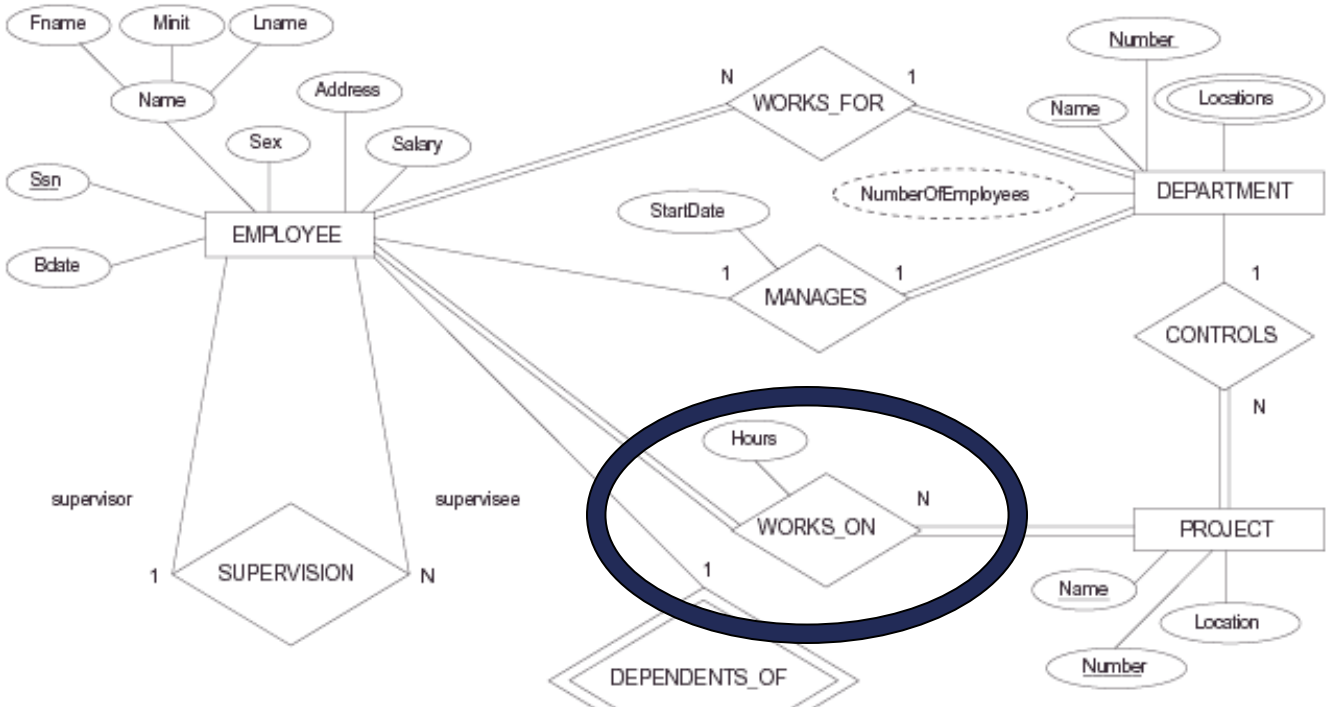

Fname
Minit
Lname
Name
Address
Sex
Salary
Ssn
Bdate
EMPLOYEE
N
WORKS_FOR
1
Number
Name
Locations
DEPARTMENT
NumberOfEmployees
StartDate
1
MANAGES
1
1
CONTROLS
N
Hours
supervisor
supervisee
WORKS_ON
N
PROJECT
1
SUPERVISION
N
Name
Location
Number
1
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
BirthDate
Relationship


Fname
Minit
Lname
Name
Address
Sex
Salary
Ssn
EMPLOYEE
N
WORKS_FOR
1
Number
Name
Locations
DEPARTMENT
NumberOfEmployees
StartDate
1
CONTROLS
N
CONTROLS, a 1:N relationship type between DEPARTMENT and PROJECT. The participation of PROJECT is total, whereas that of DEPARTMENT is determined to be partial, after consultation with the users indicates that some departments may control no projects.
Hours
supervisor
supervisee
1
SUPERVISION
N
WORKS_ON
N
PROJECT
Name
Location
Number
1
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
BirthDate
Relationship

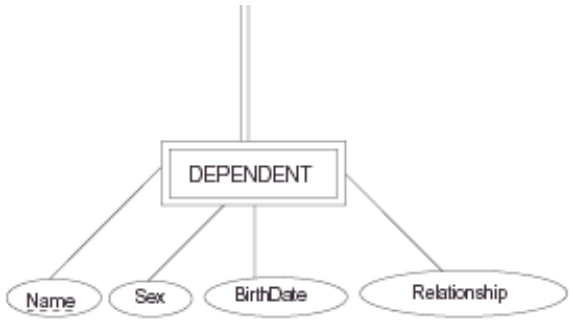

Fname
Minit
Lname
Name
Address
Sex
Salary
Ssn
Bdate
EMPLOYEE
N
WORKS_FOR
1
Number
Name
Locations
DEPARTMENT
NumberOfEmployees
StartDate
1
MANAGES
1
1
CONTROLS
N
supervisor
supervisee
1
SUPERVISION
N
Hours
WORKS_ON
N
PROJECT
Name
Location
Number
1
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
BirthDate
Relationship

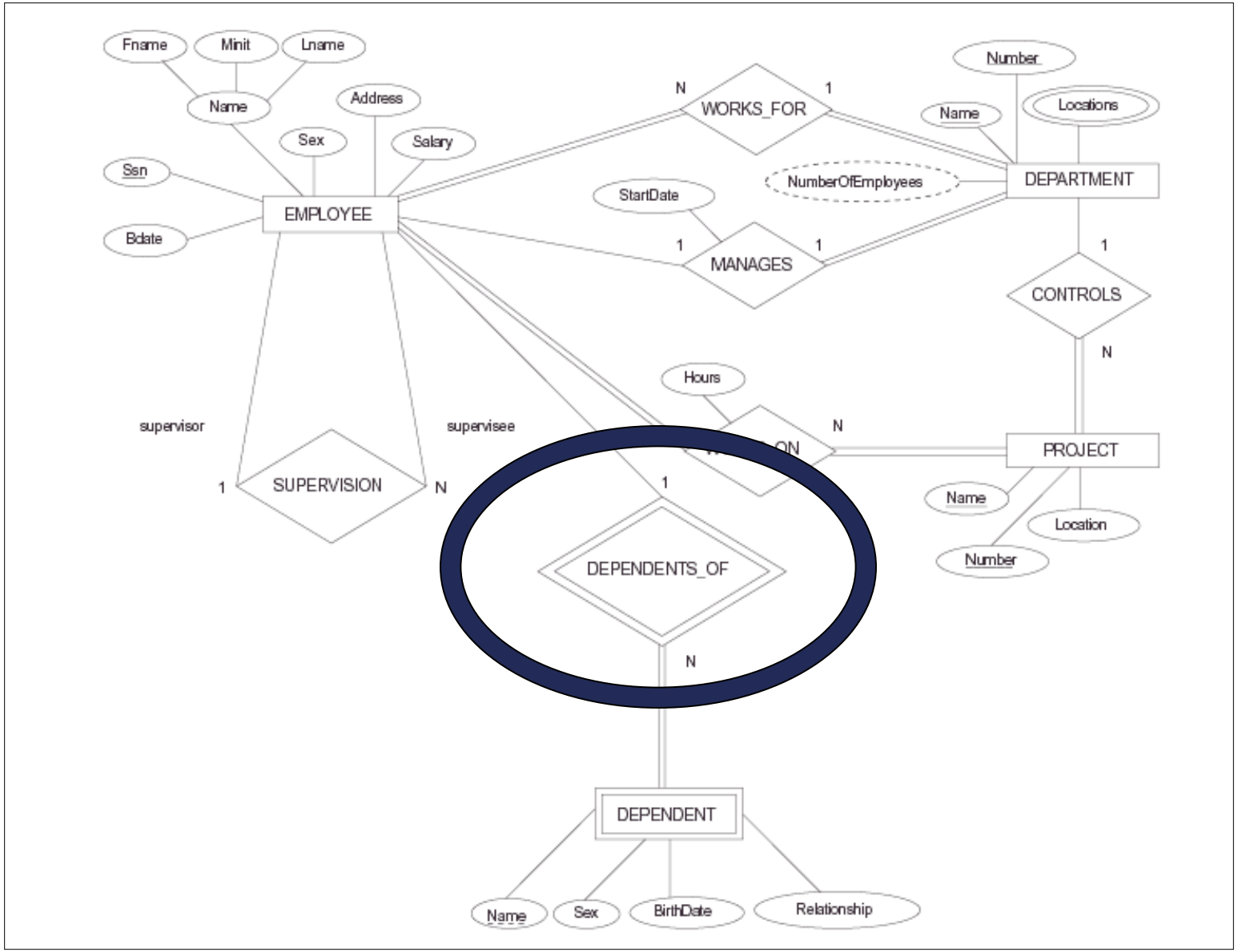

Fname
Minit
Lname
Name
Address
Sex
Salary
Ssn
Bdate
EMPLOYEE
N
WORKS_FOR
1
Number
Name
Locations
DEPARTMENT
SUPERVISION, a 1:N relationship type between EMPLOYEE (in the supervisor role) and EMPLOYEE (in the supervisee role). Both participations are determined to be partial, after the users indicate that not every employee is a supervisor and not every employee has a supervisor.
supervisor
supervisee
1
SUPERVISION
N
Hours
WORKS_ON
N
PROJECT
Name
Location
Number
1
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
BirthDate
Relationship

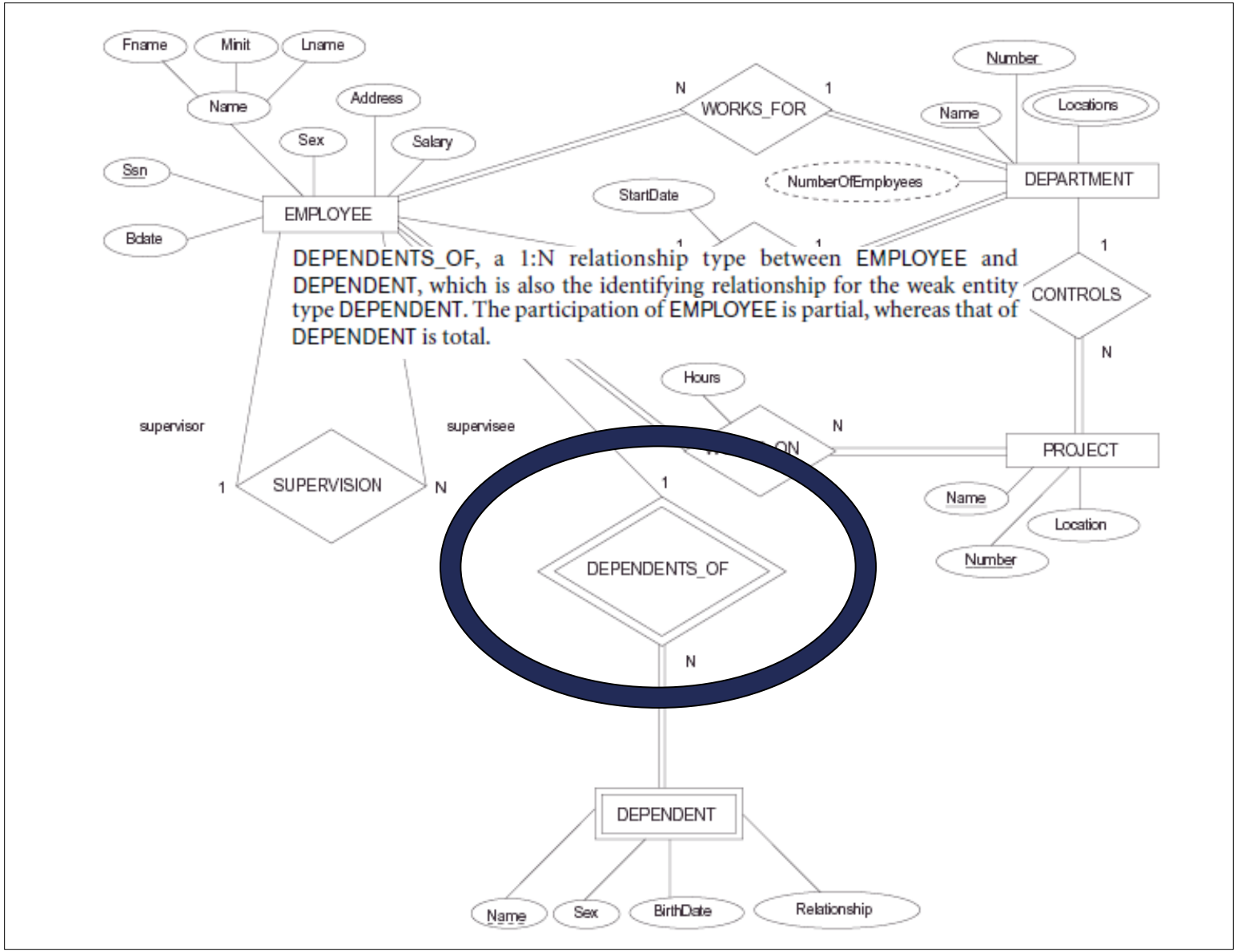

Fname
Minit
Lname
Name
Address
Sex
Salary
Ssn
Bdate
EMPLOYEE
N
WORKS_FOR
1
Number
Name
Locations
DEPARTMENT
NumberOfEmployees
StartDate
1
MANAGES
1
1
CONTROLS
N
Hours
N
WORKS_ON
PROJECT
Name
Location
Number
supervisor
supervisee
1
SUPERVISION
N
1
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
BirthDate
Relationship

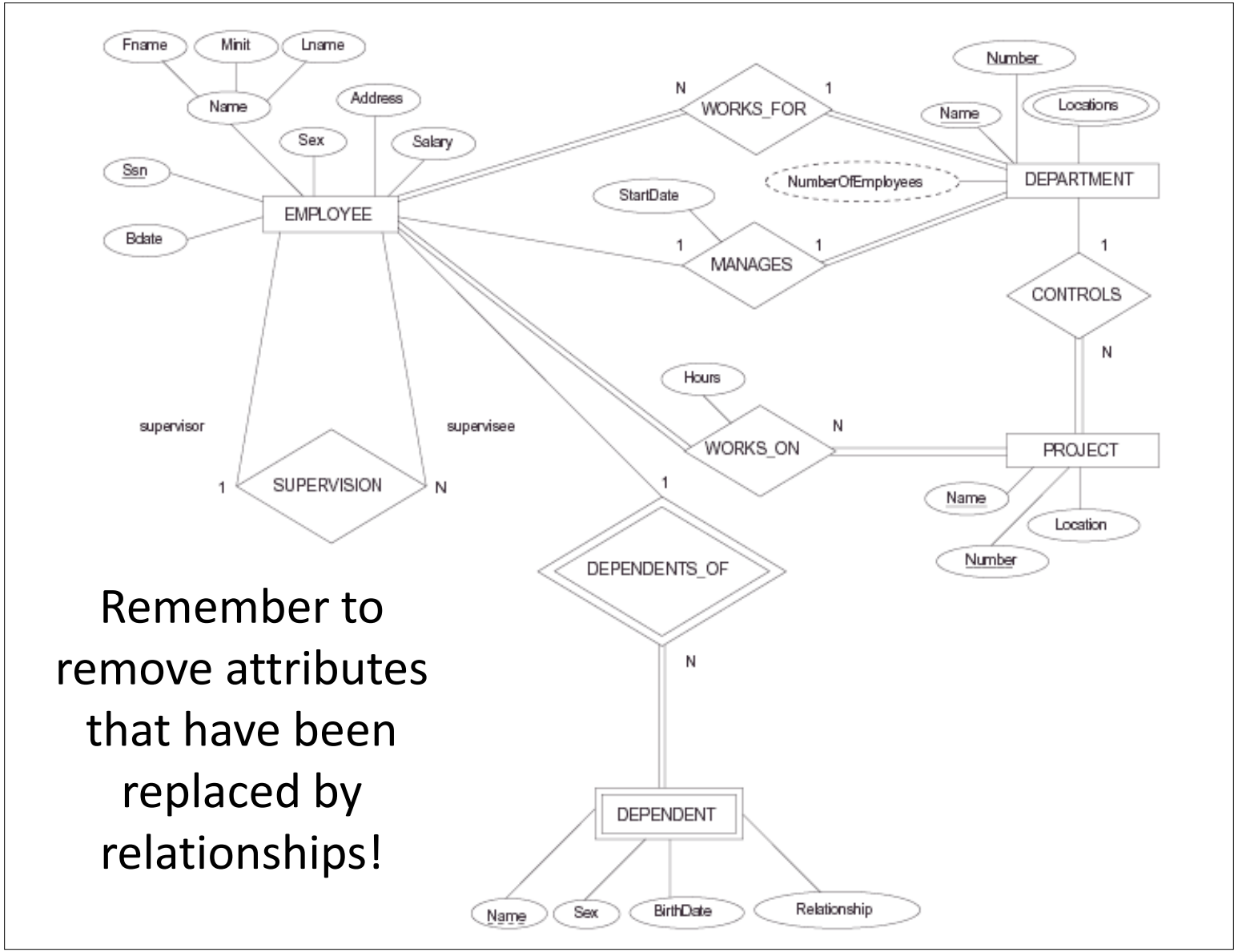



WORKS_ON, determined to be an M:N relationship type with attribute Hours, after the users indicate that a project can have several employees working on it. Both participations are determined to be total.




Fname
Minit
Lname
Name
Address
Sex
Salary
Ssn
Bdate
EMPLOYEE
N
WORKS_FOR
1
Number
Name
Locations
DEPARTMENT
NumberOfEmployees
StartDate
1
MANAGES
1
1
CONTROLS
N
Hours
N
ON
PROJECT
Name
Location
Number
supervisor
supervisee
1
SUPERVISION
N
1
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
BirthDate
Relationship


Fname
Minit
Lname
Name
Address
Sex
Salary
Ssn
Bdate
EMPLOYEE
N
WORKS_FOR
1
Number
Name
Locations
DEPARTMENT
NumberOfEmployees
StartDate
1
1
1
CONTROLS
N
DEPENDENTS_OF, a 1:N relationship type between EMPLOYEE and DEPENDENT, which is also the identifying relationship for the weak entity type DEPENDENT. The participation of EMPLOYEE is partial, whereas that of DEPENDENT is total.
Hours
supervisor
supervisee
N
ON
PROJECT
1
SUPERVISION
N
1
Name
Location
Number
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
BirthDate
Relationship




Remember to remove attributes that have been replaced by relationships!

# Other restrictions

- The ER-diagram and schema cannot represent all restrictions from the mini world
- Those restrictions, however, can be extremely important when building the database application
- Write down these restrictions as a note to the diagram and schema

# ER-schema Conventions

- Singular names to entity types
- ENTITY_TYPE and RELATIONSHIP_TYPE names are uppercase letters
- Attributes have first letter capitalized
- Role names are lowercase letters

# From text to ER-schema

- Often, you will create an ER diagram based on a textual description. There are some rules-of-thumb that can help you in this task

- Noun: entity-type
  - In our **university**, **student**s follow **course**s that are given by a **teacher**.

# From text to ER-schema

- Verb that connects nouns: relation type
  - In our university, students **follow** courses that are **given** by a teacher.
- Adjective: attribute at entity-type
  - The **large** shop sells **expensive** cars.

# From text to ER-schema

- Adverb: attribute at relation-type
  - The student **successfully** finishes the course.
  - The car is rent by the client **for a certain period of time**.
- Relationship names read from left-to-right or top-to-down.
  - We should have written HAS_DEPENDENTS instead of DEPENENTS_OF

# From text to ER-schema

- When you are done, inspect the diagram and improve it where possible:
  - Identify possible keys
  - Identify weak entity-types
  - Attributes that appear in many entities should better become entities on their own
  - Entity types with a single attribute, linked only to one other entity type, should better be denoted as an attribute of that entity type (Ex: Gender)
  - Identify and remove redundant relationships
  - Identify redundancies (a relationship being described by an attribute)

# From text to ER-schema

- Describe the domain of all attributes
- Write down all restrictions that cannot be represented in the diagram
- Describe all operations on the data that are needed for the database application (data entry, removal, etc.)

# Alternative notation

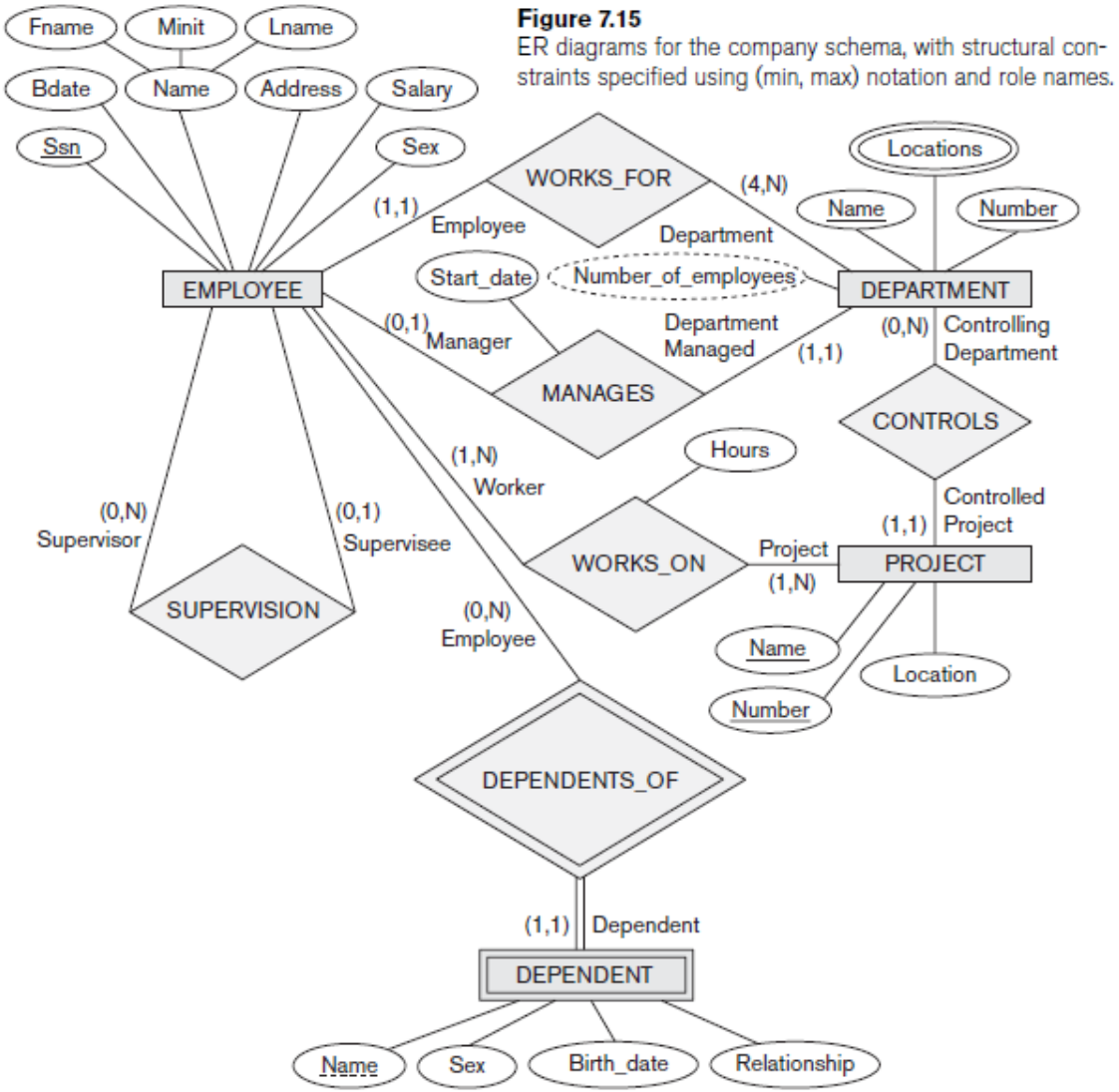



**Figure 7.15**
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.