# Database Management System

Lecture 3

Database Design – ER Model and ER model to Relational Schemas

* Some materials adapted from R. Ramakrishnan, J. Gehrke and Shawn Bowers

# Today's Agenda

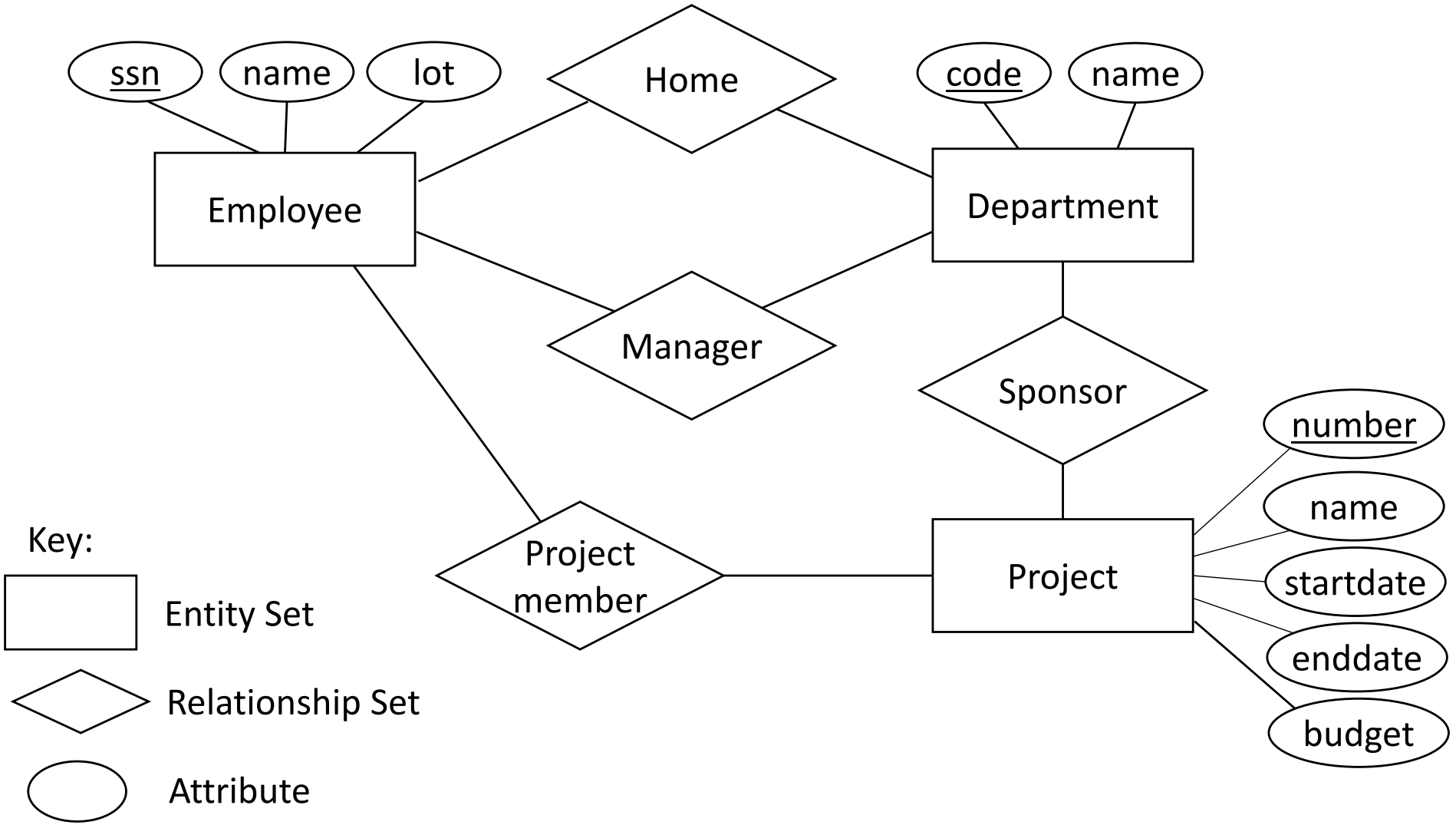
- ER Model
- ER Model to Relational

# ER Model

# Conceptual Data Modeling

- Similar to software design ...
  - requirements gathering and analysis
  - *application architecture and design*
  - implementation and testing
  - maintenance
- database design involves multiple steps (esp. prior to creating table definitions)
- Here we focus on conceptual design using the Entity- Relationship Model
  - similar to the use of UML diagrams for software design

# Entity-Relationship (ER) Diagram [Chen 71]



# Terminology

- An ”Entity” is an object that can be distinguished from other object
  - e.g., the individual “John Smith” or a particular company
  - described using a set of attribute-value pairs
  - one or more attributes as ids (i.e., keys)
- An “*Entity Set*” is a collection of similar entities
  - e.g., the set of employee entities
  - defined by the type of attributes and relationships entites of the set are characterized by
  - often informally called an “entity” if everyone knows we are talking about the schema
- An Entity Set is also be called an “*Entity Type*”

# Terminology

- A "*Relationship*" is an association among 2 or more entities
  - e.g., John's *home department* is Pharmacy 2
  - just as entities are instances of entity sets, relationships are
  - instances of relationship sets ...
- A "*Relationship Set*" is a collection of similar relationships
  - e.g., the set of *home department* relationships
  - defined by the participating entity types and other constraints
  - often informally called a "relationship" if everyone knows we are talking about the schema
- A Relationship Set is also be called a "*Relationship Type*"

# Entity-Relationship Model vs. Relational Model

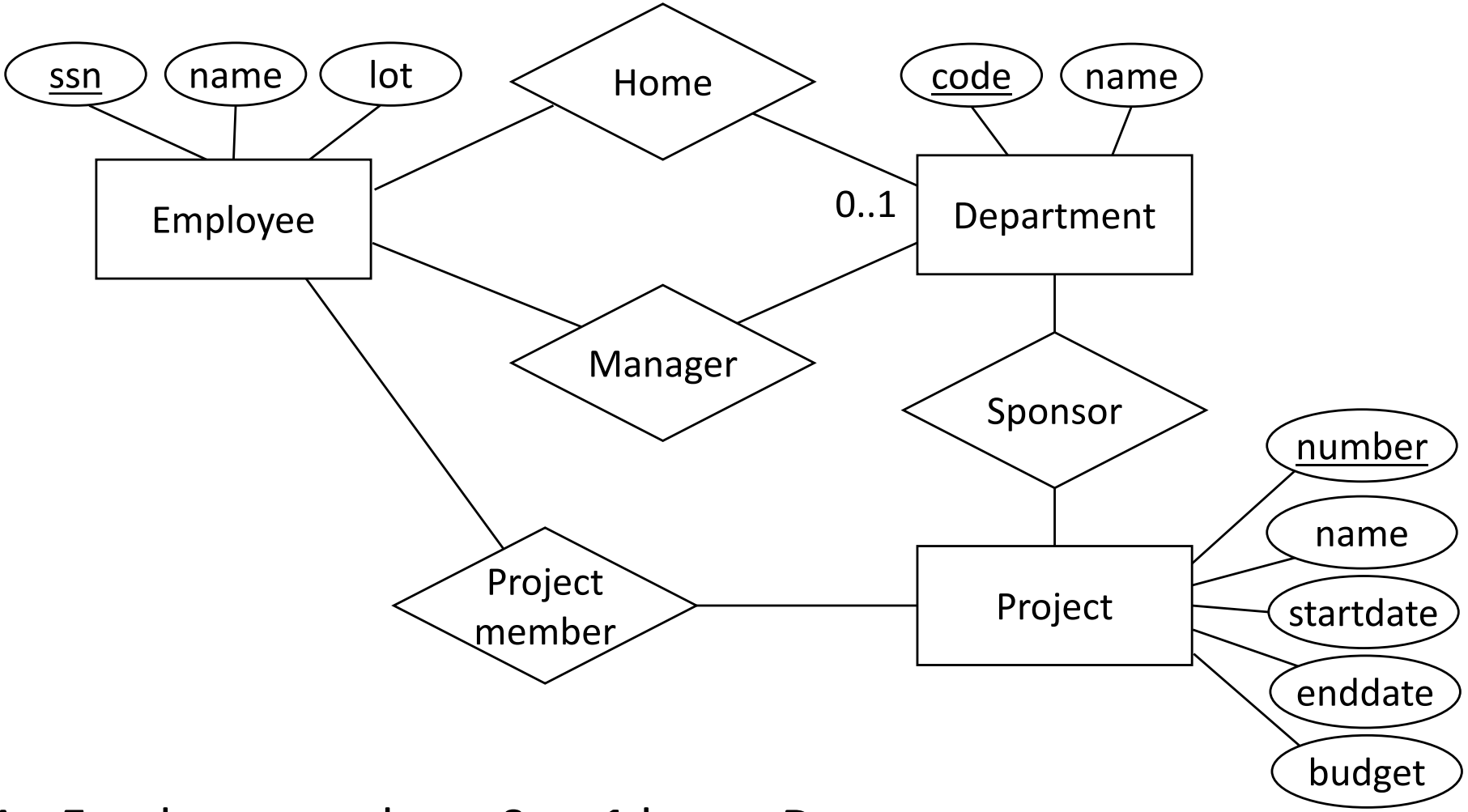
- *A different data model than the relational model*
  - different constructs for modeling schemas and data
  - DBMS systems (in various forms) have even been built on the ER model ... though primarily used as a design tool
- The relational model has:
  - *Tables* (relations) with attributes, primary keys, foreign keys, rows, values
- The ER model has:
  - Entities and Entity Sets with attributes and entity identifiers (like keys)
  - Relationships and Relationship Sets with cardinality constraints, roles, attributes, etc.

# Mapping ER models to Relational Schemas

Employee(ssn, name, lot, home-dept)
ProjectMember(ssn, number)
Department(code, name, manager)
Project(number, name, startdate, enddate, budget, sponsor)

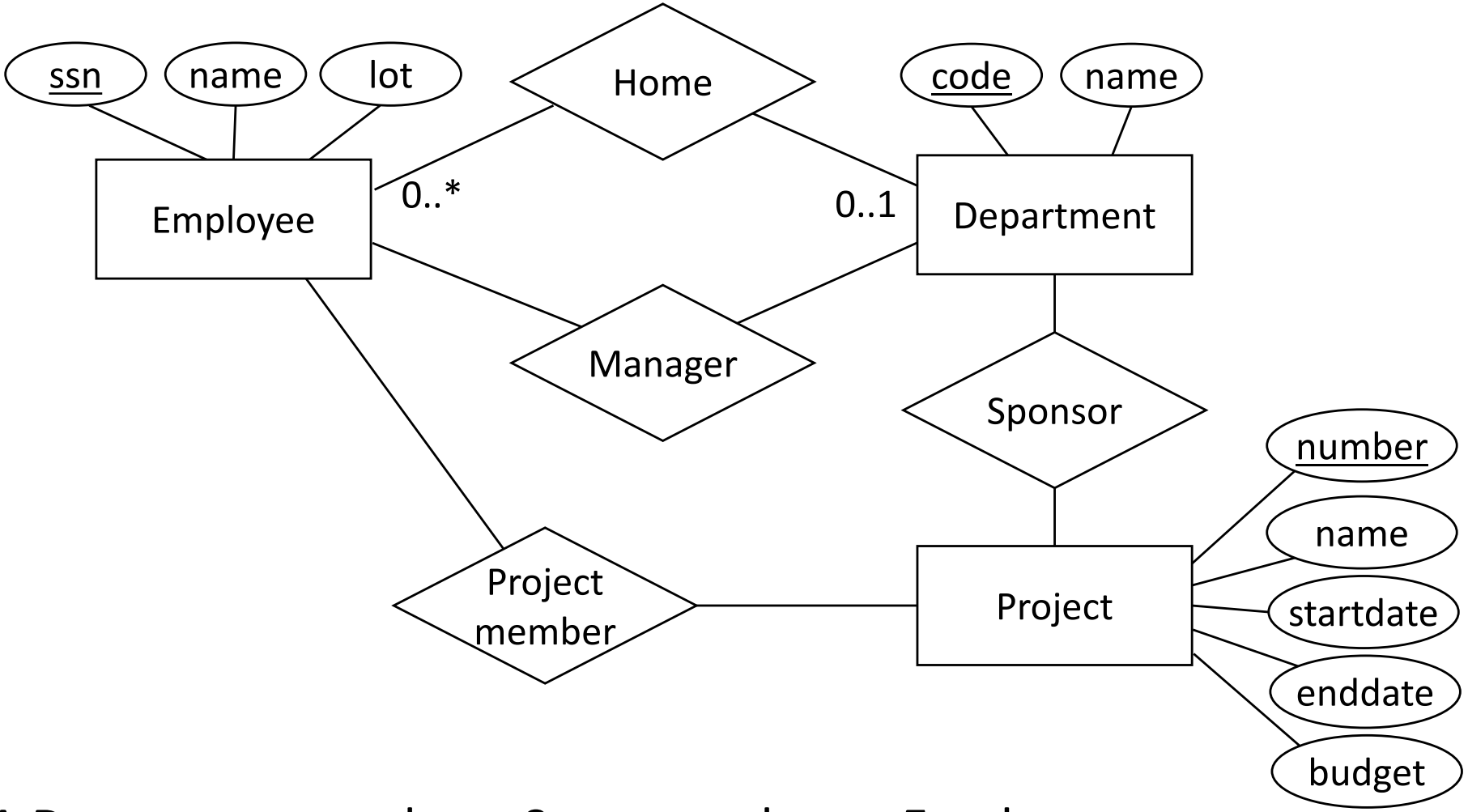
- This mapping assumes:
  - Employees have one home department
  - Departments have one manager
  - Employees can participate in many projects

# ER Cardinality Constraints



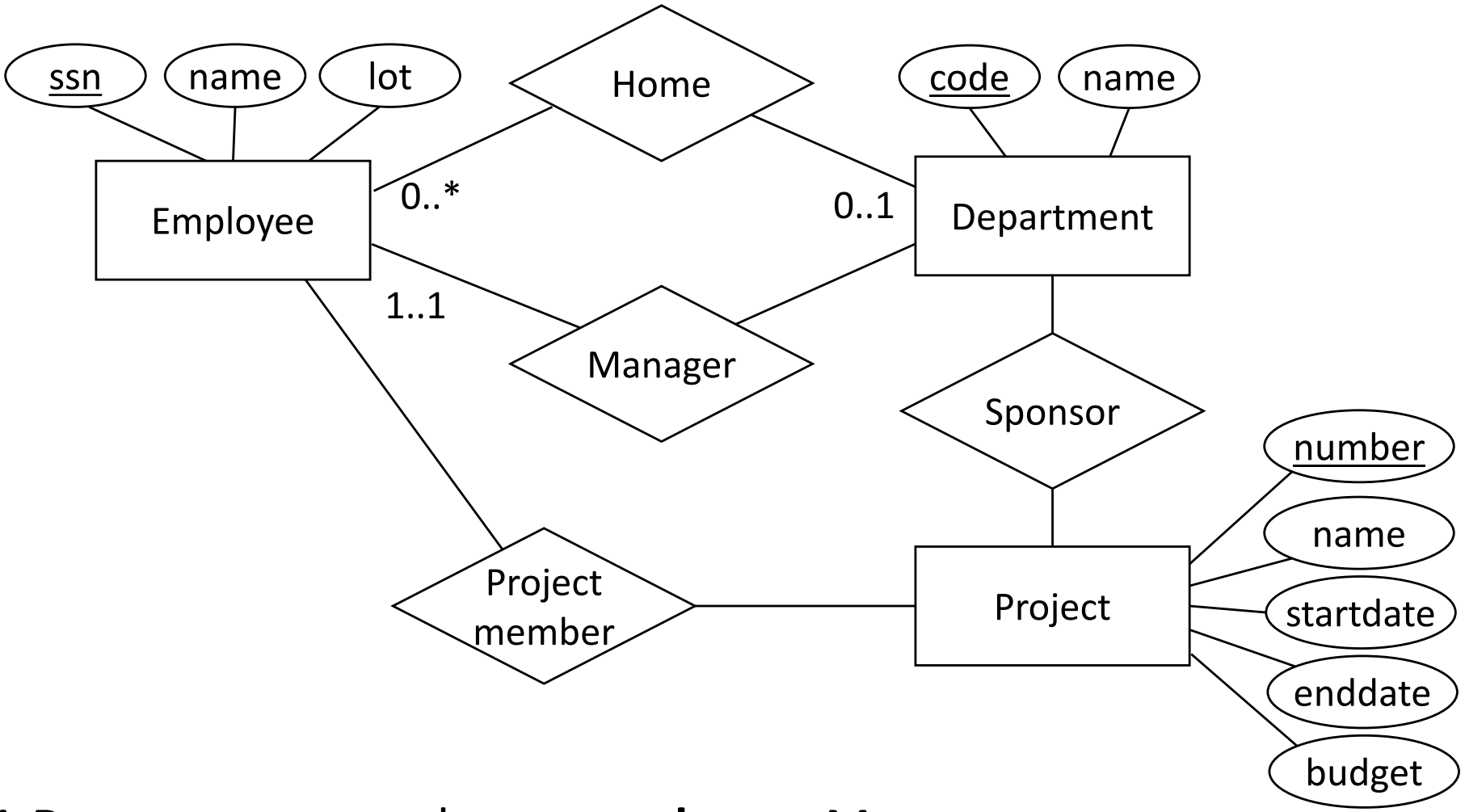

An Employee can have **0 or 1** home Department

# ER Cardinality Constraints



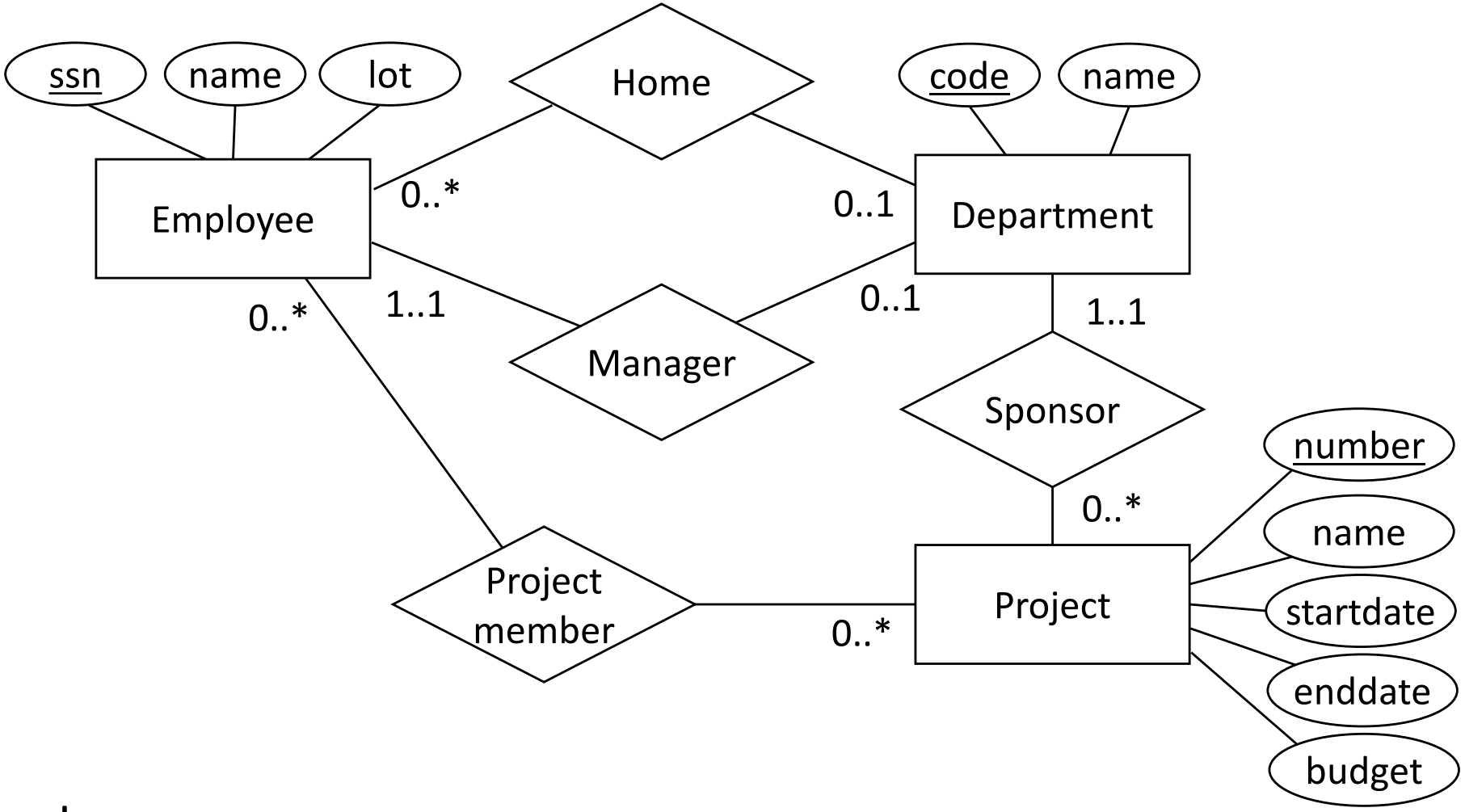

A Departemnt can have **0 or many** home Employees

# ER Cardinality Constraints




A Department must have **exactly one** Manager

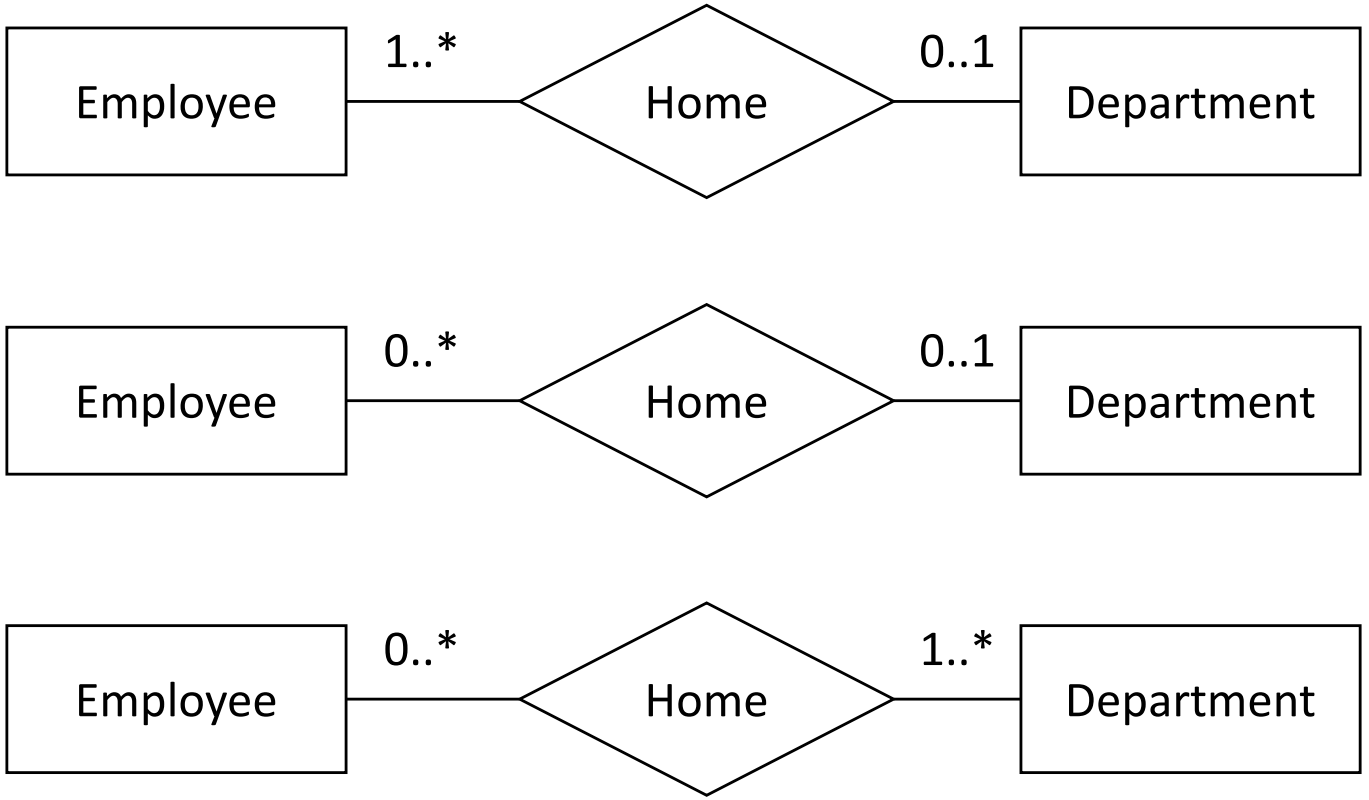
# ER Cardinality Constraints




and so on…

# ER Cardinality Constraints

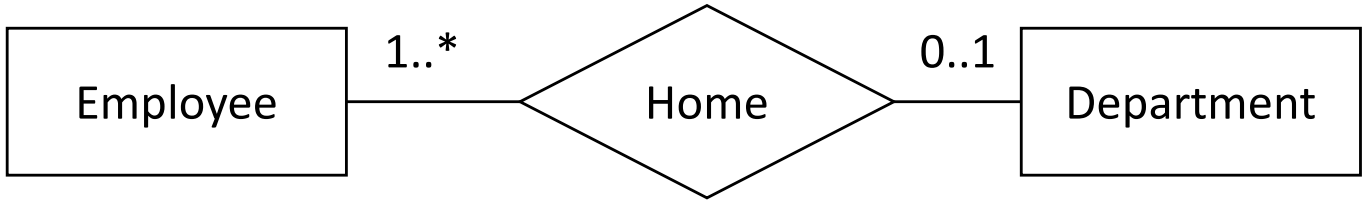
- Which one is correct?



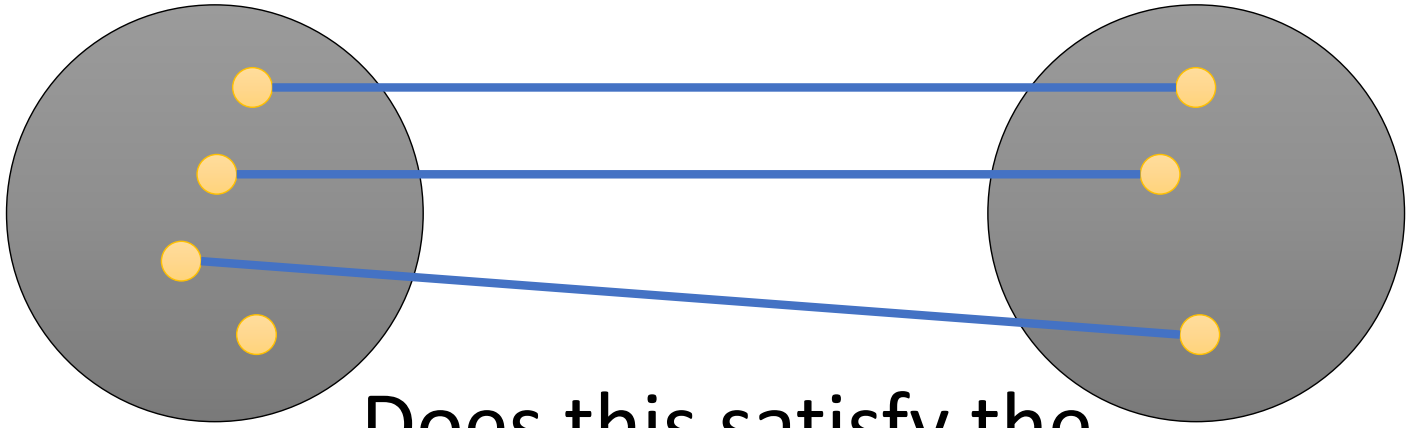

Must discover the semantics of the applications

# ER Cardinality Constraints (1/5)

- Constraints are expressed over Entity Sets and Relationship Sets
- Constraints on the members of the corresponding Entities and Relationships



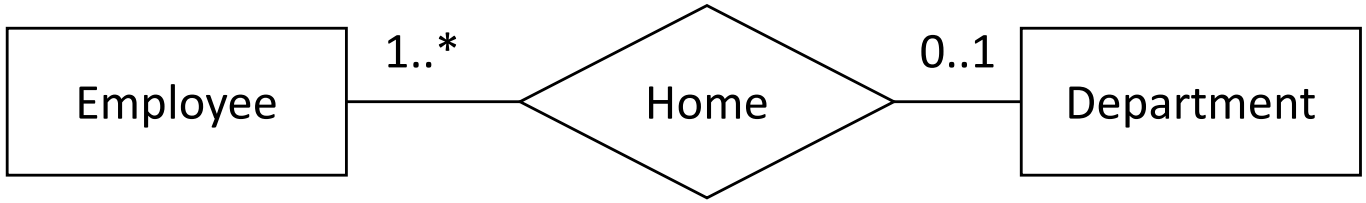

home relationship



Does this satisfy the cardinality constraints?

# ER Cardinality Constraints (2/5)

- Constraints are expressed over Entity Sets and Relationship Sets
- Constraints on the members of the corresponding Entities and Relationships

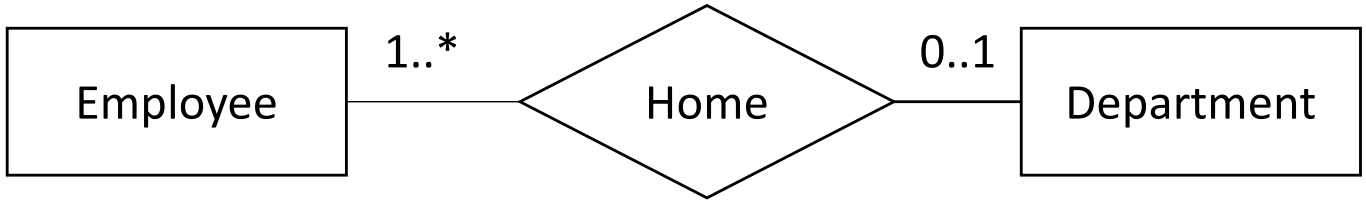



home relationship

How about now?

# ER Cardinality Constraints (3/5)

- Constraints are expressed over Entity Sets and Relationship Sets
- Constraints on the members of the corresponding Entities and Relationships



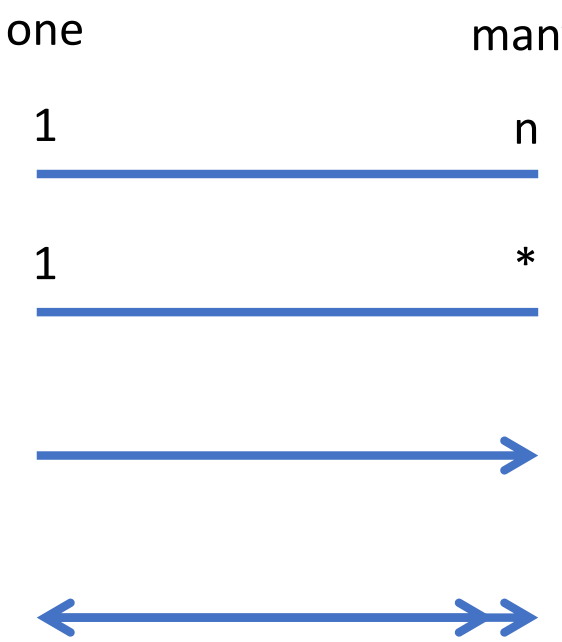

home relationship

And now?

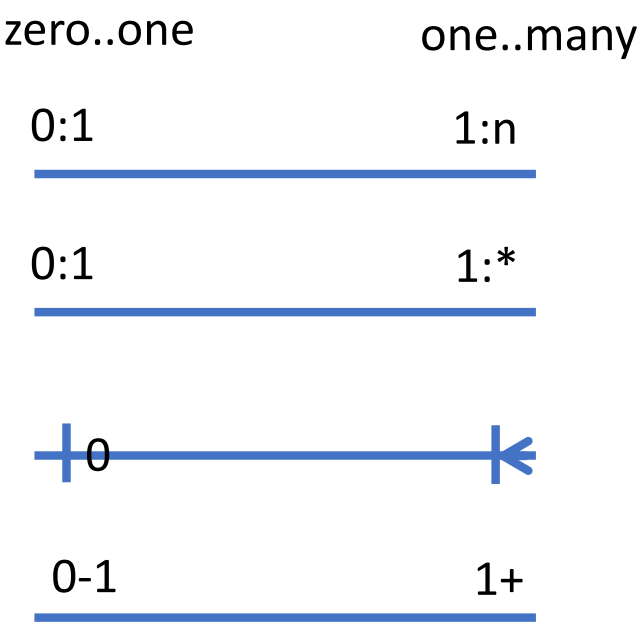
# ER Cardinality Constraints (4/5)

- There are various notations used for writing cardinality constraints ...

## Examples of “One to Many” constraints




Maximum cardinalities only

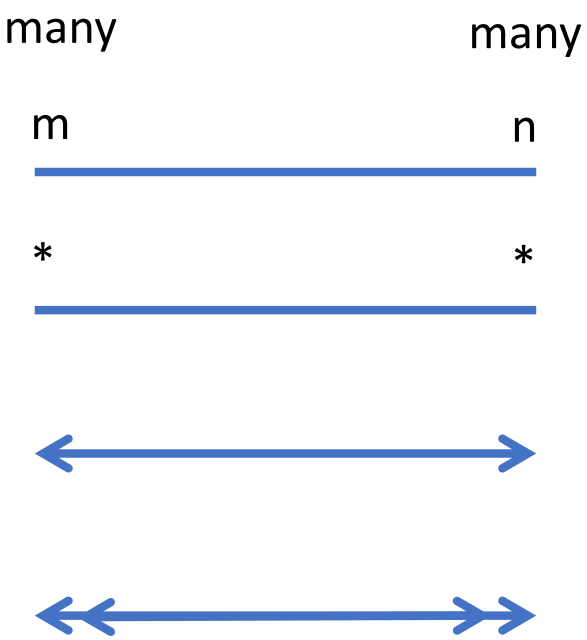



Maximum and maximum cardinalities

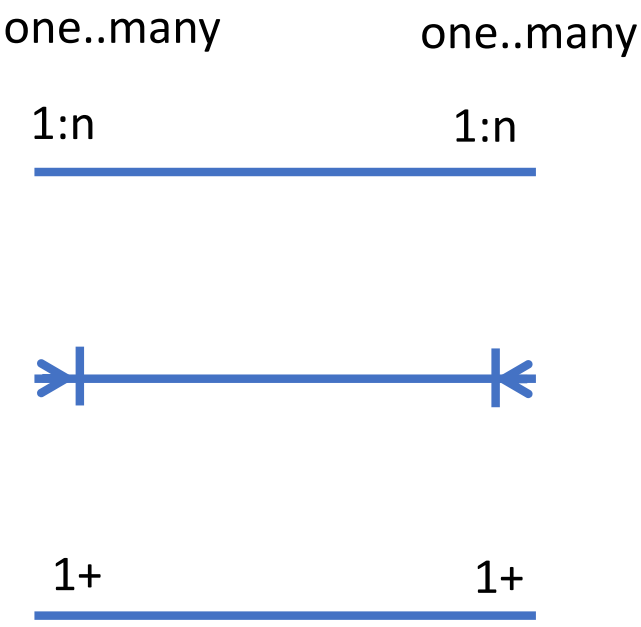
# ER Cardinality Constraints (5/5)

- There are various notations used for writing cardinality constraints ...

## Examples of “Many to Many” constraints

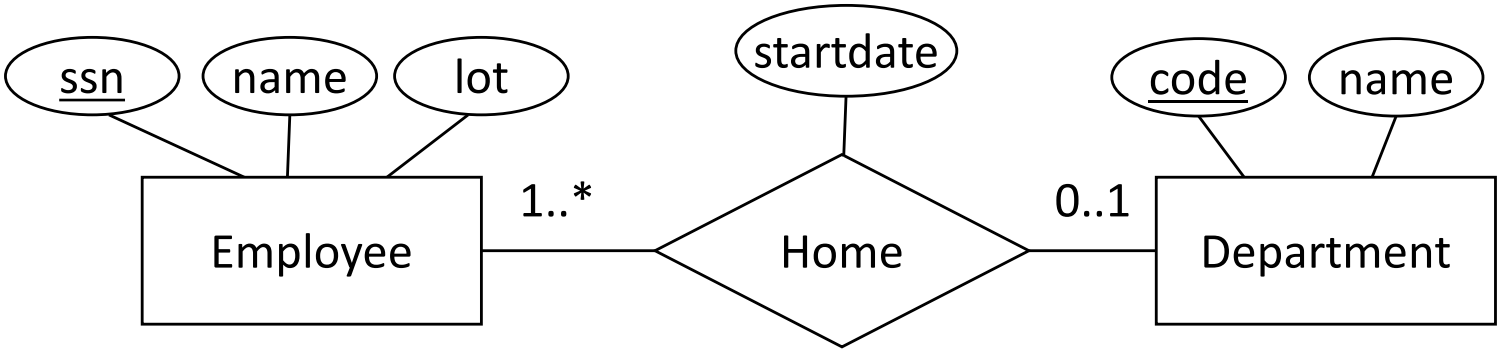



Maximum cardinalities only




Maximum and maximum cardinalities

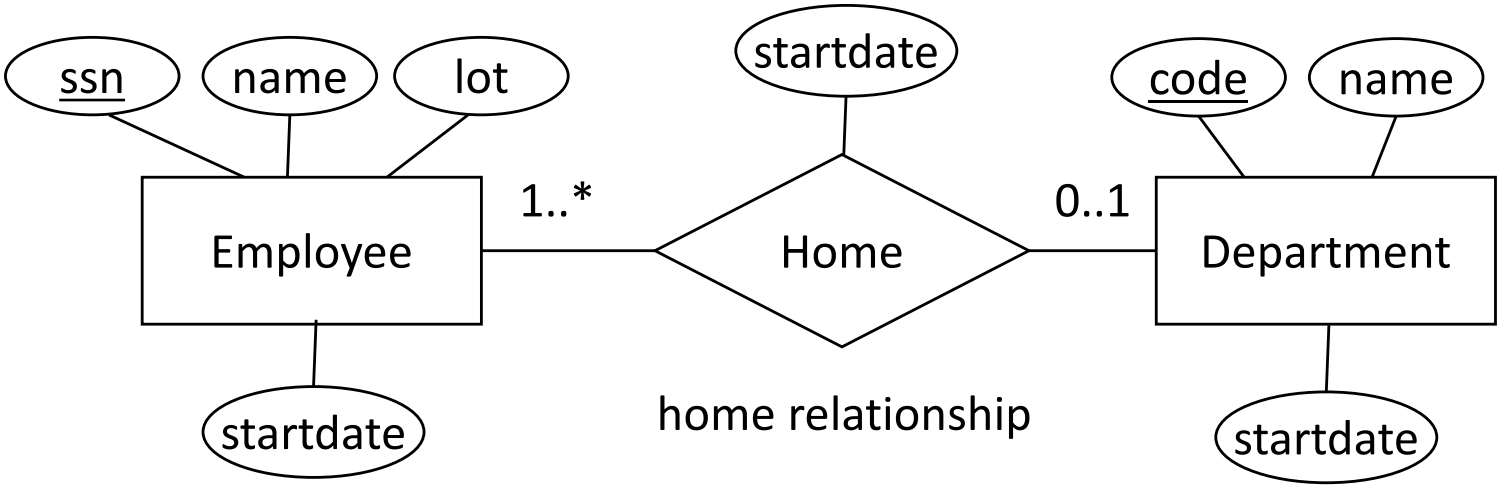
# Relationship Set Attributes (1/4)




home relationship

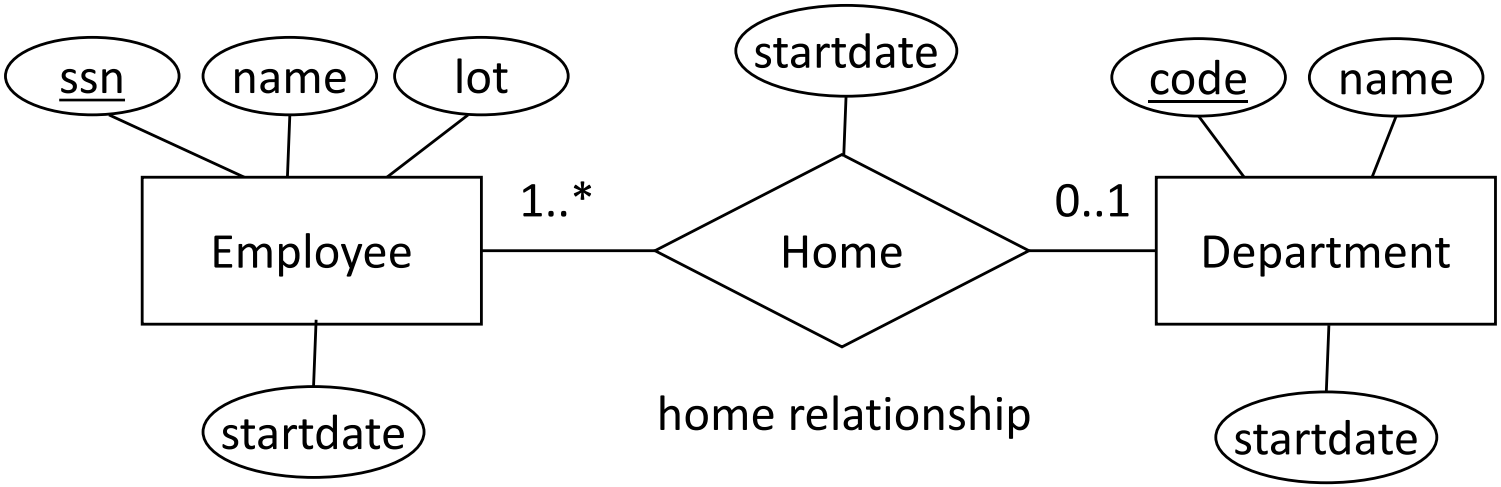
- Each instance of the relationship has a value for the attribute ...

# Relationship Set Attributes (2/4)




- Which one makes sence?

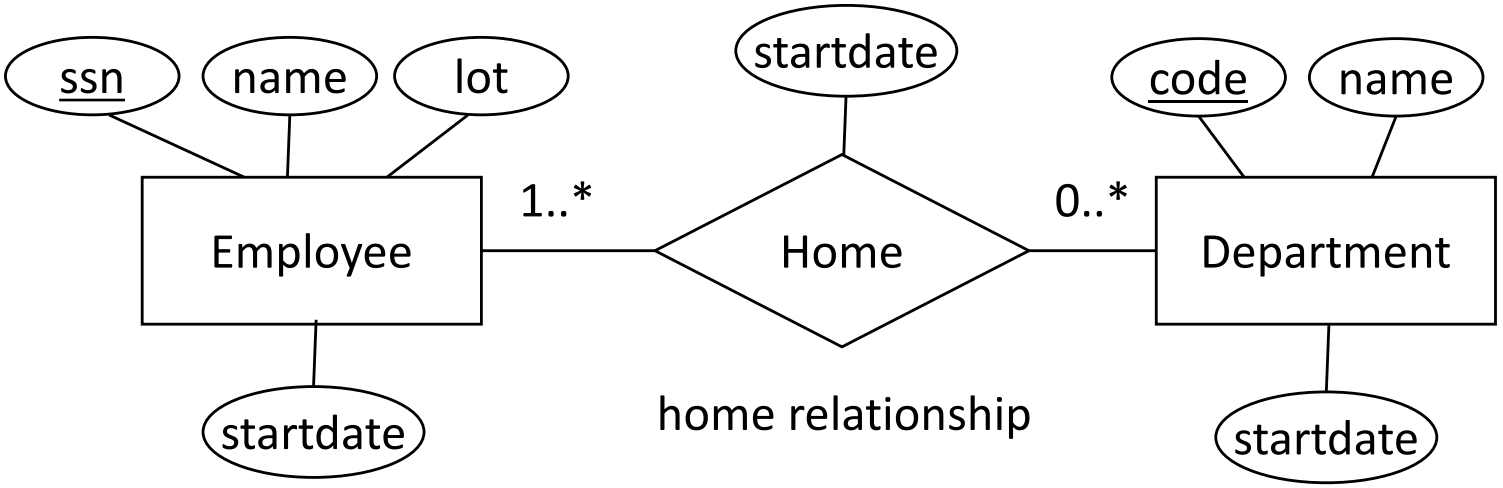
# Relationship Set Attributes (3/4)




- Because Employees have *zero or one* home departments,
- ... start date will work as an Empoyee or home attribute

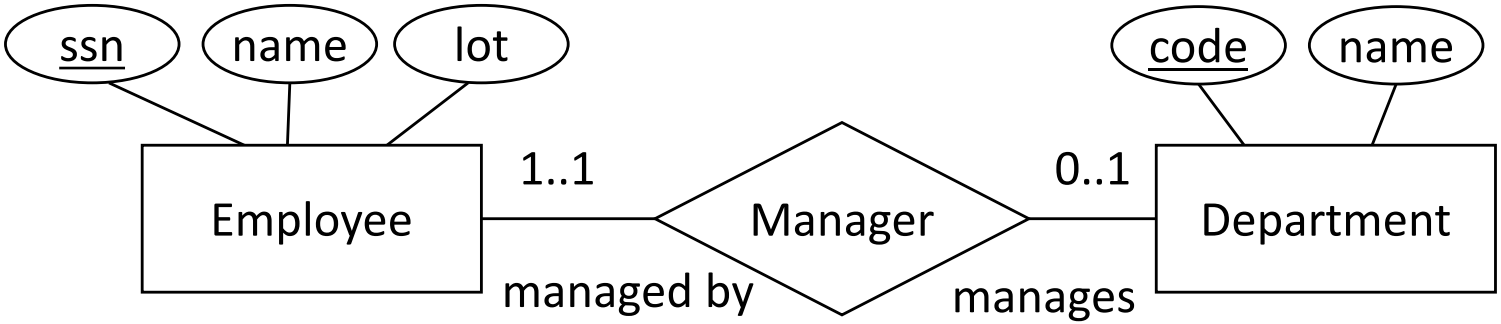
- *What about startdate at Department?*

# Relationship Set Attributes (4/4)




- What about now?
- ... Since employees can have multiple home departments, it must be a *relationship attribute*

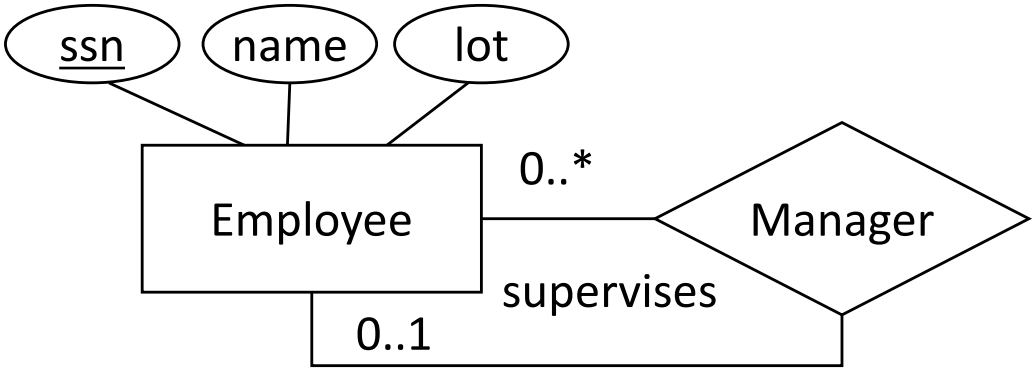
# Relationship Set Role Names




- Relationships can have role names
  - An employee "**manages**" zero or one department
  - A department is "**managed by**" exactly one employee

Note: some notations use the opposite side of the relationship set to specify cardinality and roles

# Relationship Set

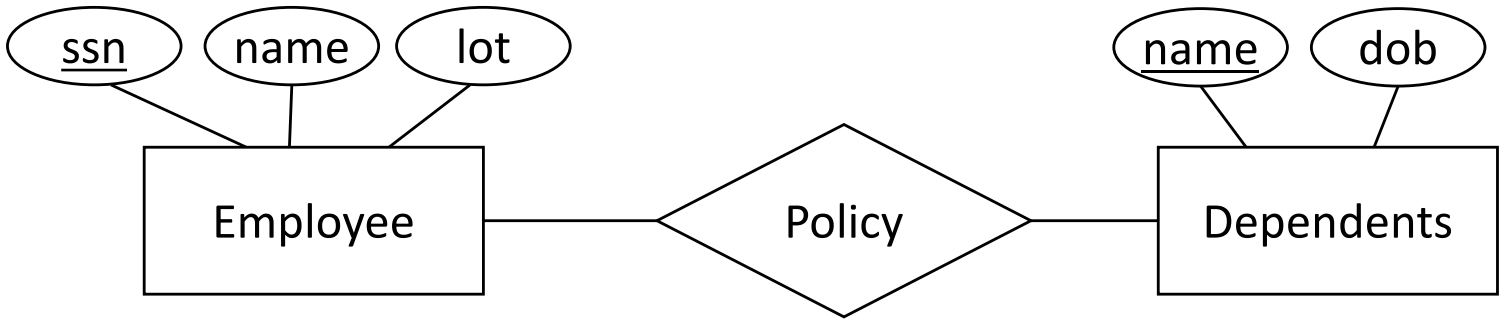



- Entity sets can participate in different roles for the same relationship set

# Exercise

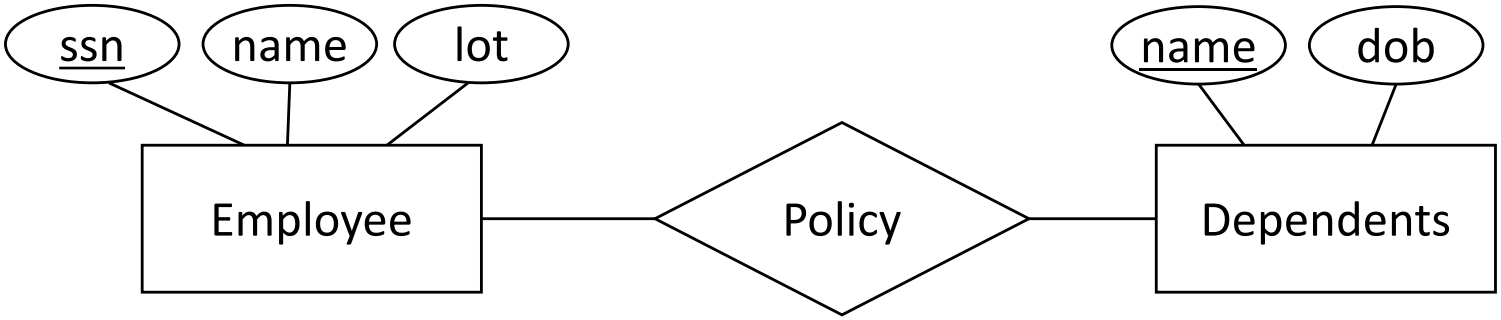
- Form groups of 2
- Draw an ERD for a database that stores information about *Students*, *Faculty*, *Courses*, and *Course Offerings*
  - Faculty can serve as a “course coordinator”
  - Faculty can be qualified to teach a course
  - Courses can have other courses as prerequisites
  - One or more faculty can teach course offerings
- Identify entity sets, attributes, and keys
- Identify relationships (and roles, if needed)
- Define cardinality constraints

# Weak Entity Sets (1/)



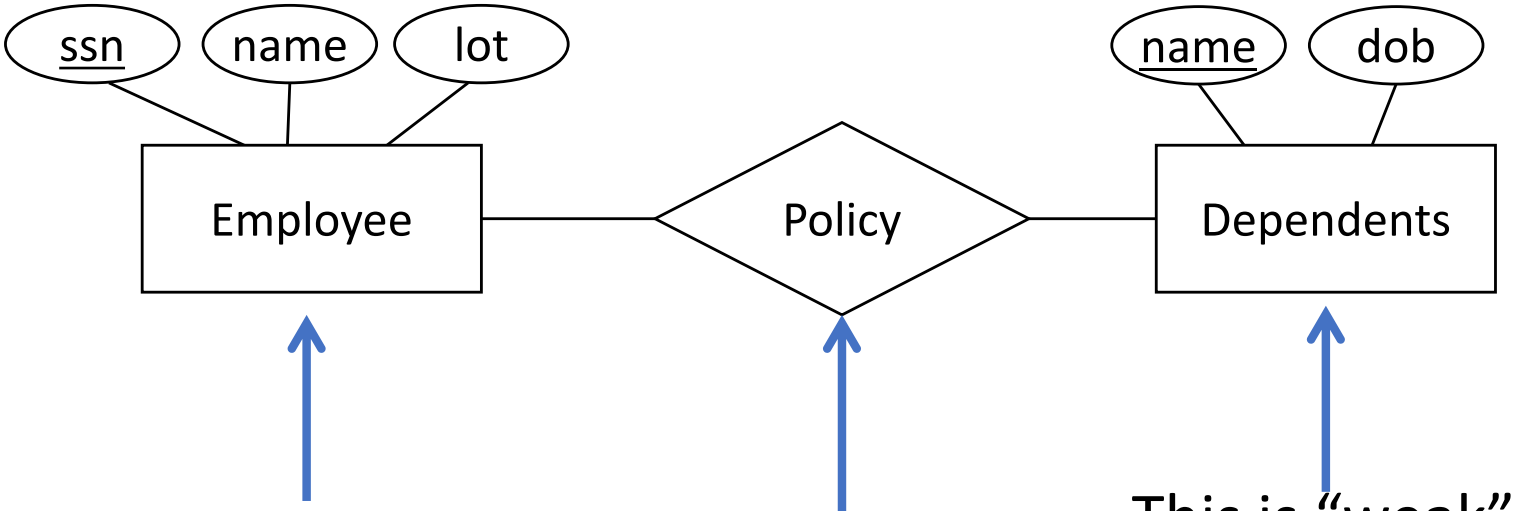

- In this model assume
  - we need to record the insurance policies of employees
  - we need to track dependents w.r.t. the policies
  - we only need to store the name and date-of-birth of dependents (and nothing else)
  - that, e.g., when employees leave, we no longer track their policies or dependents

# Weak Entity Sets (2/)




- Note that we only identify dependents through their corresponding employees
  - we assume dependents of an employee have unique names
  - but different employees could have dependents with the same name ... since names are not guaranteed to be unique, e.g.,“John Smith”

# Weak Entity Sets (3/)



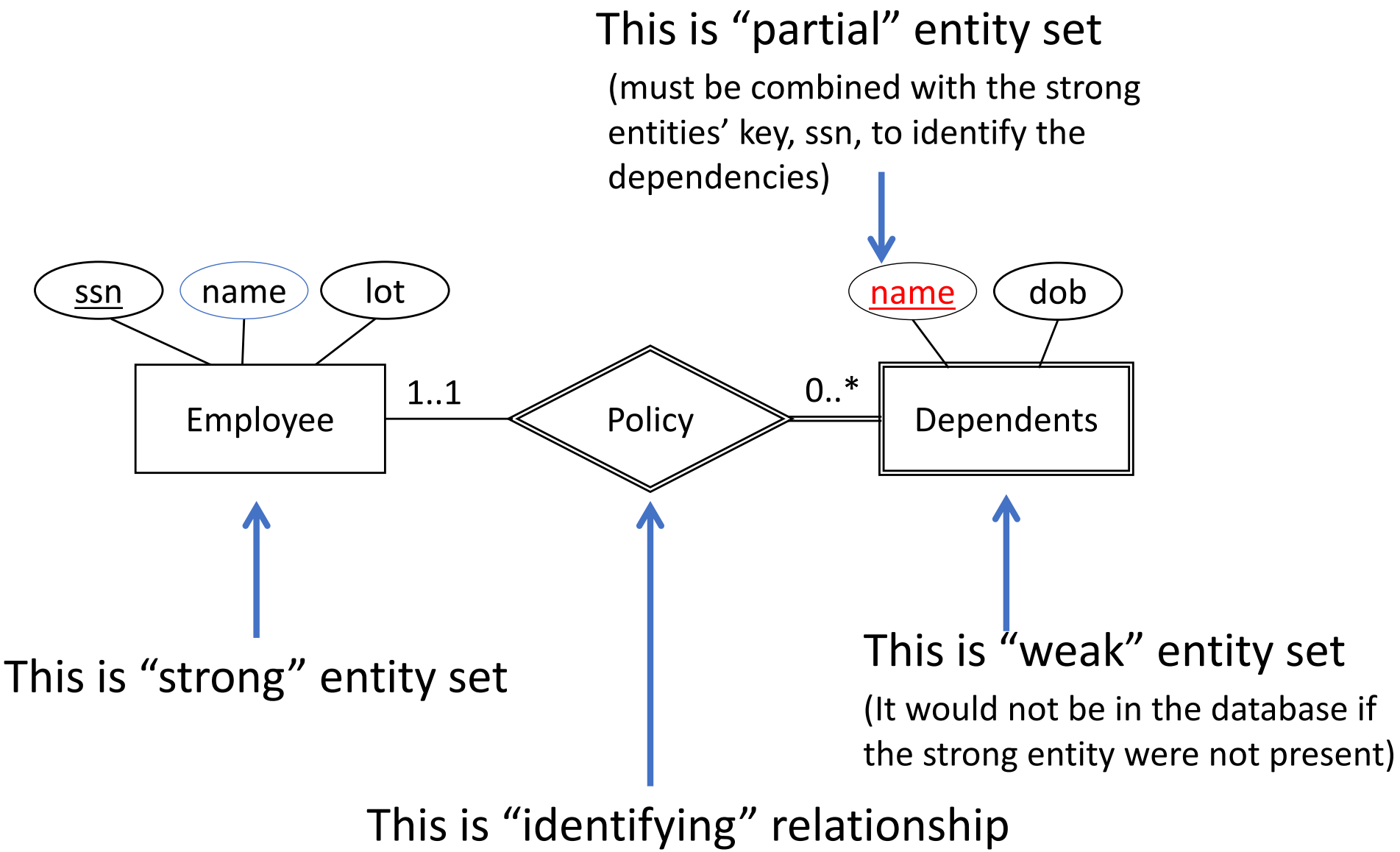

This is “strong” entity set

This is “weak” entity set

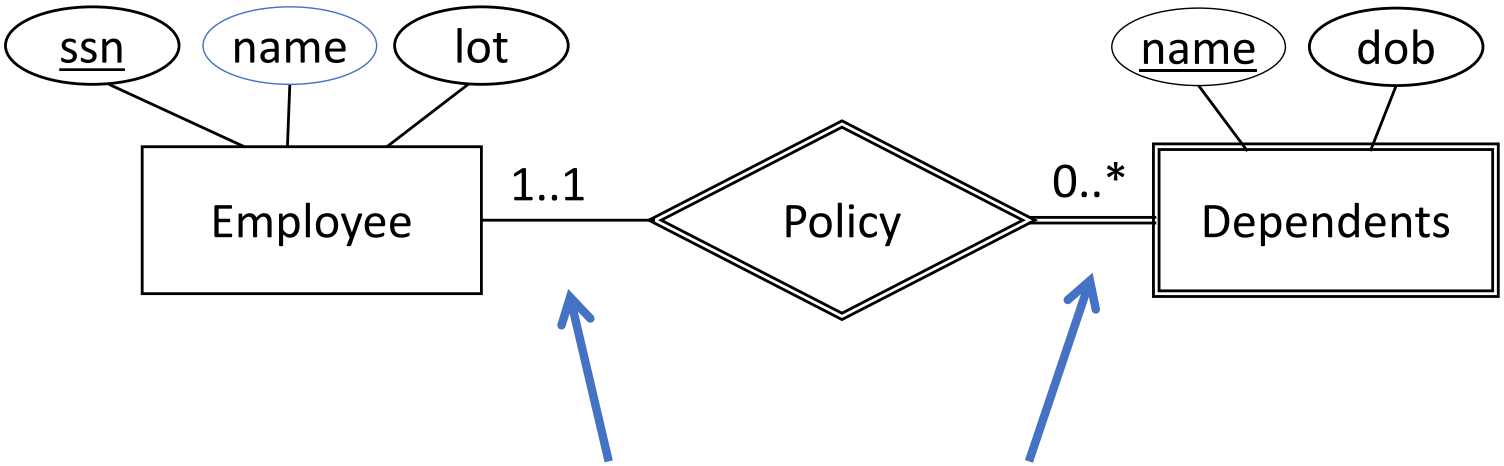
(It would not be in the database if the strong entity were not present)

This is “identifying” relationship

# Weak Entity Sets (4/)



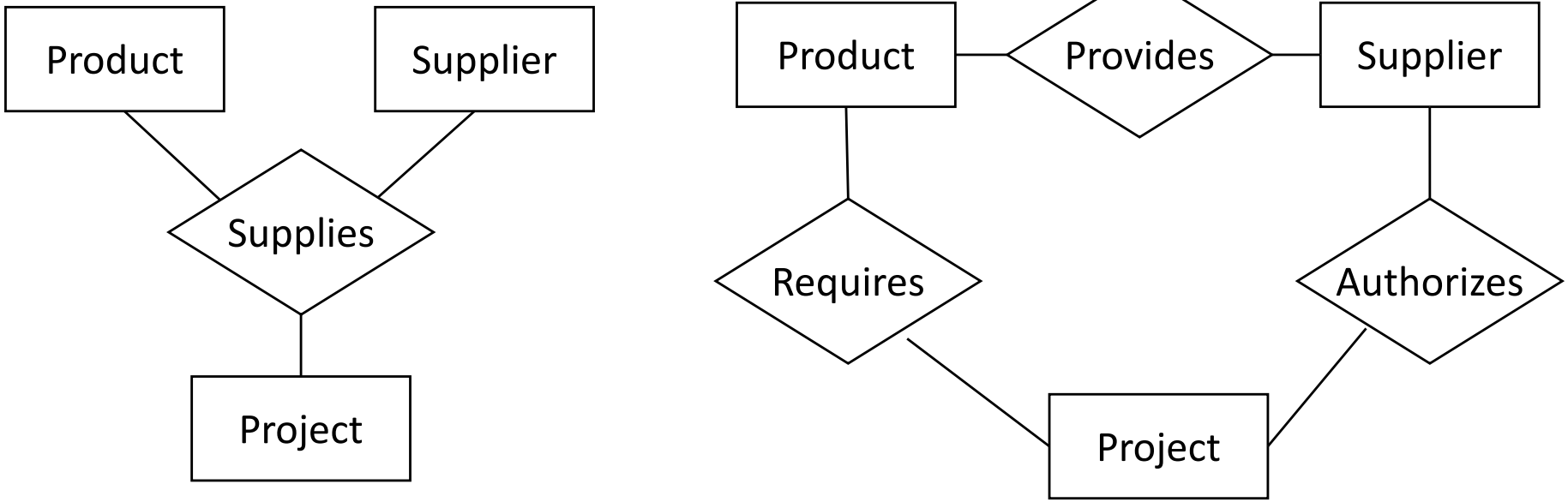
# Weak Entity Sets (5/)




Cardinalities for
an identifying relationship set

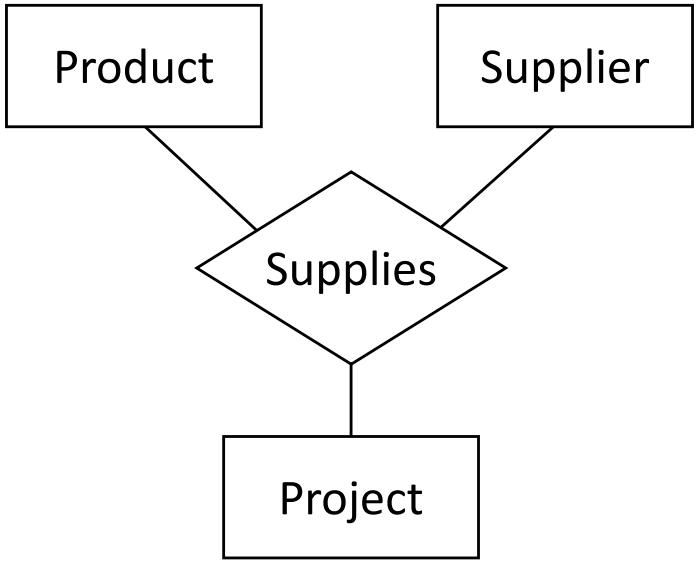
They **must** be like this

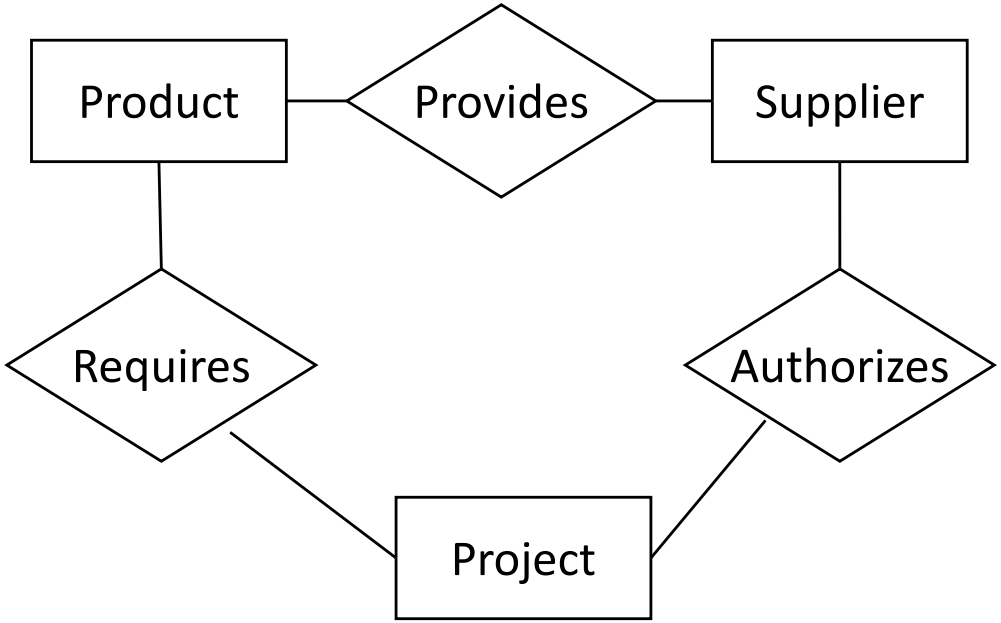
# Ternary versus Binary Relationships (1/4)




- These two schemas are not equivalent!
- When would we use a ternary relationship set?
- When would we use three binary relationship sets?

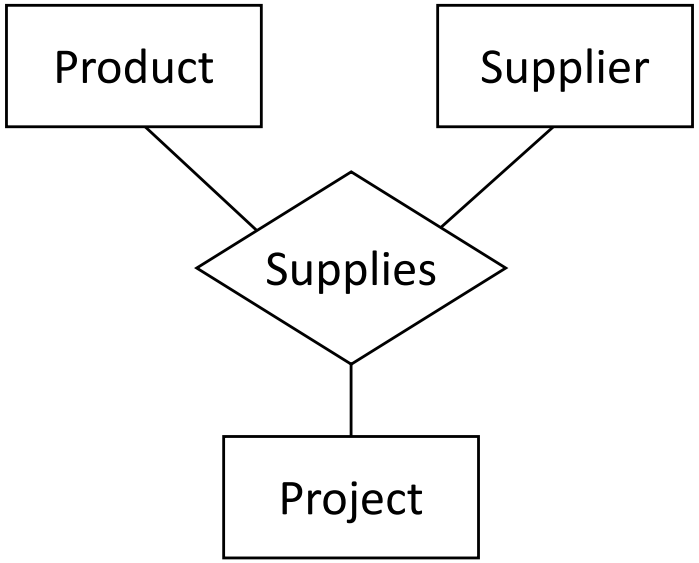
# Ternary versus Binary Relationships (2/4)

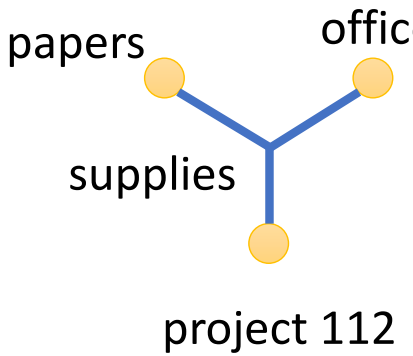


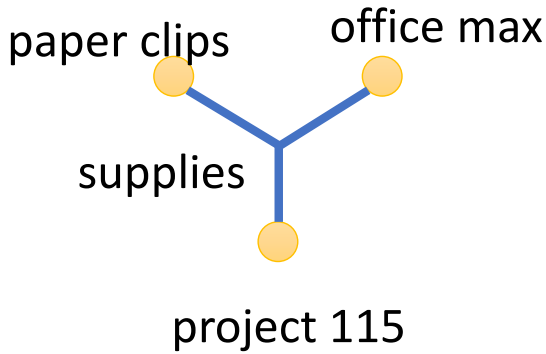




- This *ternary* relationship set means that a supplier must be authorized to supply a particular part to a particular project
- For example
  - “office depot” can supply “printer paper” to “project 112”
  - “office max” can supply “paper clips” to “project 115”
  - but this does not imply “office max” can supply “paper clips” to “112”

# Ternary versus Binary Relationships (3/4)

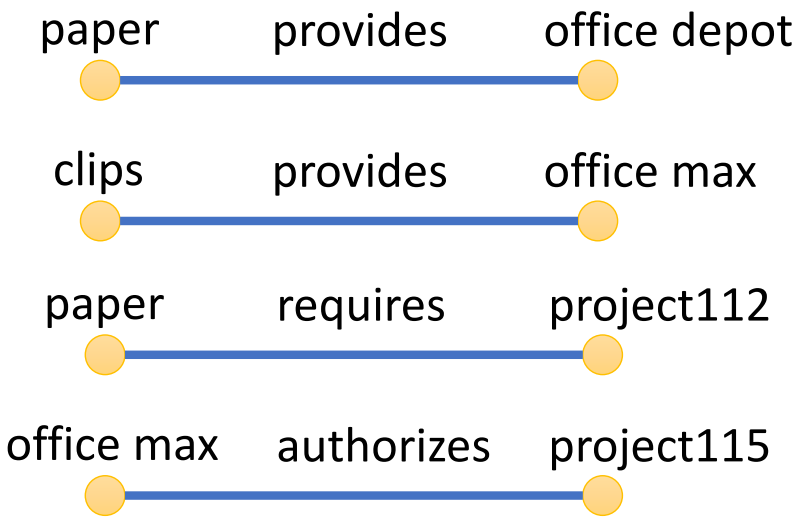


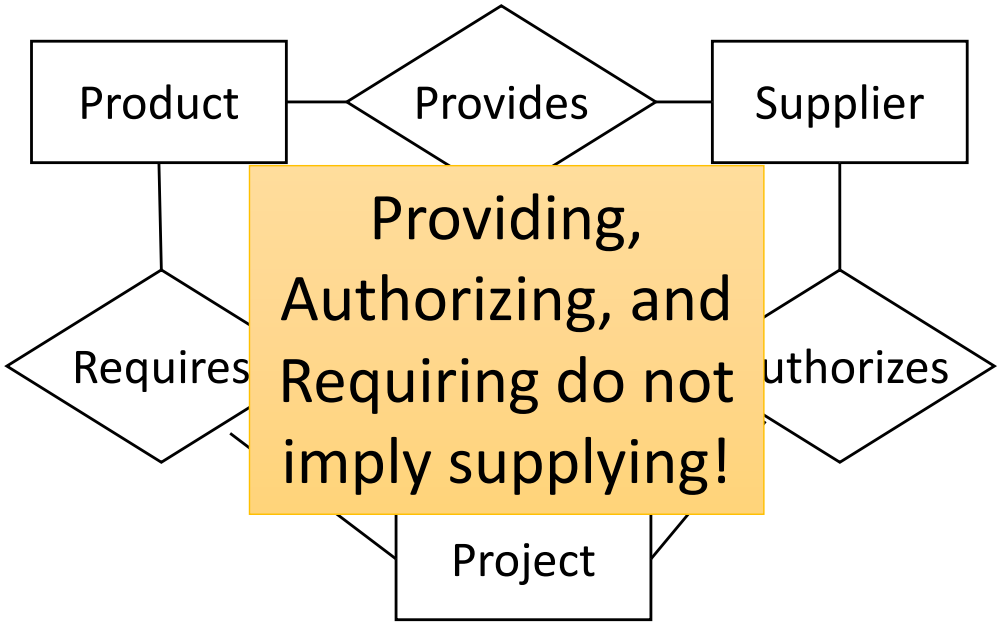




- This *ternary* relationship set means that a supplier must be authorized to supply a particular part to a particular project
- For example
  - “office depot” can supply “printer paper” to “project 112”
  - “office max” can supply “paper clips” to “project 115”
  - but this does not imply “office max” can supply “paper clips” to “112”

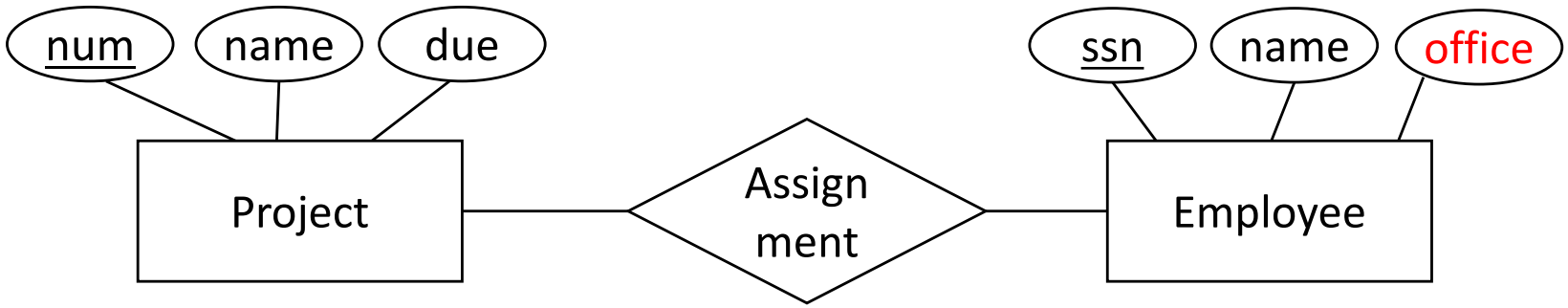
# Ternary versus Binary Relationships (4/4)



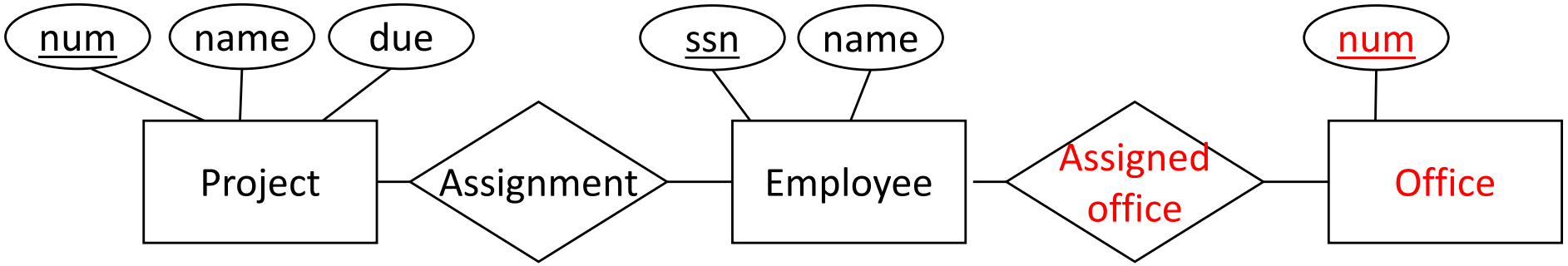

- Each binary relationship set represents something *distinct*
  - a supplier can provide certain products (*office max can provide pencils*)
  - a project can require certain products (*112 requires pencils*)
  - a supplier can be authorized to use a certain supplier (*112 is authorized to use office max*)
  - therefore, we might assume that office-max supplies pencils to 112

# Duality: Entity vs Value / Attribute vs Relationship (1/3)



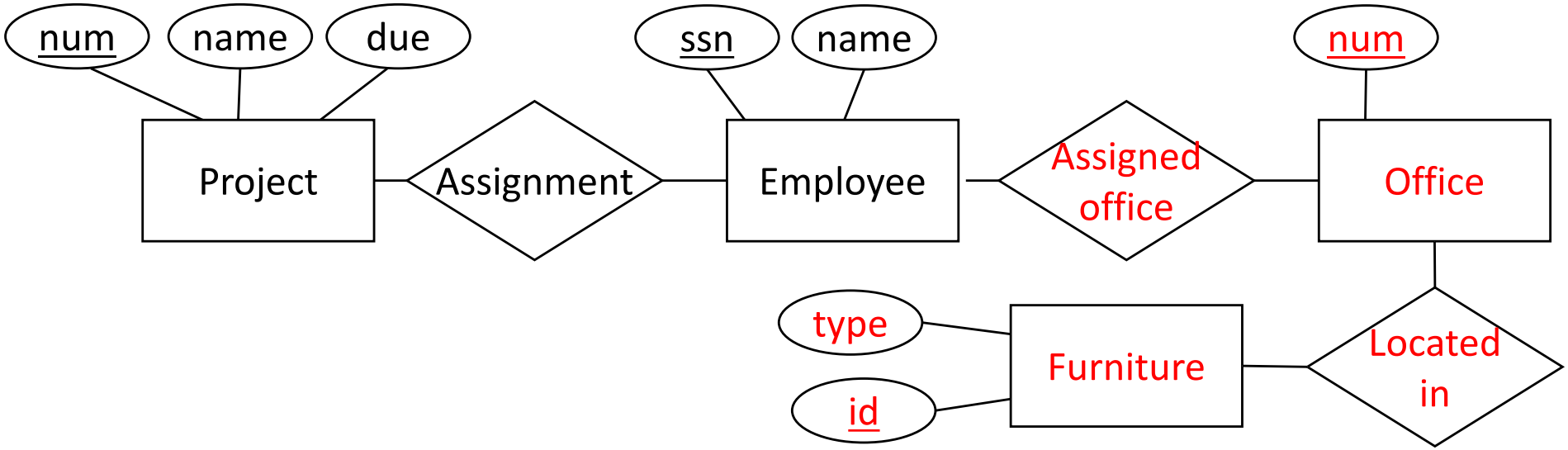

- Should office be an attribute of Employee?
- Should office be a separate Entity Set?
  - Most attributes can be “promoted” to an Entity Set ... and some Entities can be “demoted” to an attribute value
  - This is one reason why there are so many different ways to design a schema

# Duality: Entity vs Value / Attribute vs Relationship (2/3)




- What are some reasons to model Office as an Entity Set?
  - An employee can have more than one office
  - There are other attributes of Office
  - Office needs to participate in other relationship sets
    - E.g., a relationship set connecting to furniture or telephones or network drops (located in the office)

# Duality: Entity vs Value / Attribute vs Relationship (3/3)




- Example when Office should be an entity set
  - Office needs to participate in other relationship sets

# Translating
# ERDs to Relational Schemas

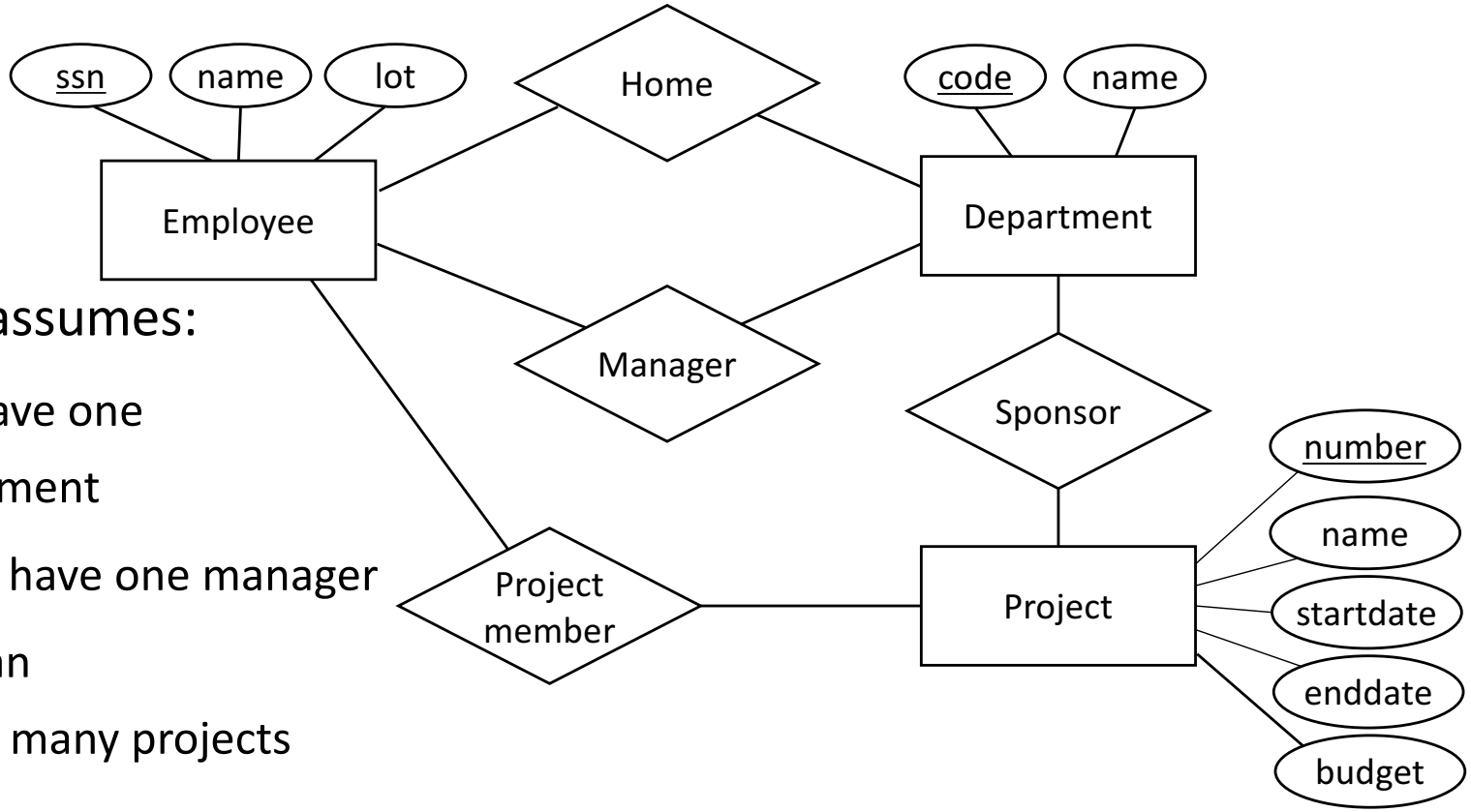
# Translating ERDs to Relational Schemas

Employee(ssn, name, lot, home-dept)

ProjectMember(ssn, number)

Department(code, name, manager)

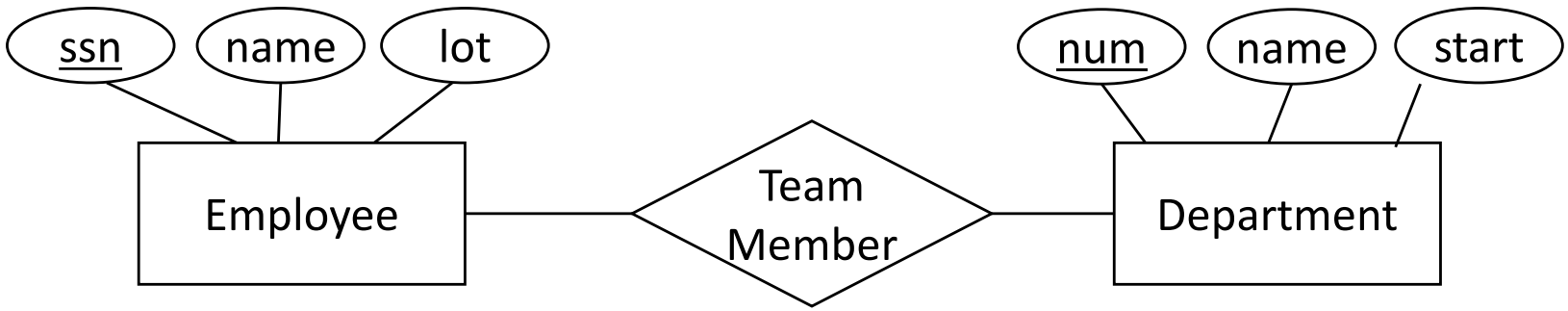
Project(number, name, startdate, enddate, budget, sponsor)




- This mapping assumes:
  - Employees have one home department
  - Departments have one manager
  - Employees can participate in many projects

# Translating Relationship Sets (1/7)

- For relationship sets
  - we must indicate which entities we want to have connected
  - e.g., we need the *key values* for employees and teams stored together to represent the relationship
  - these could be in an existing table that represents one of the involved entities ...
  - ... or in a new table introduced explicitly to represent the relationship



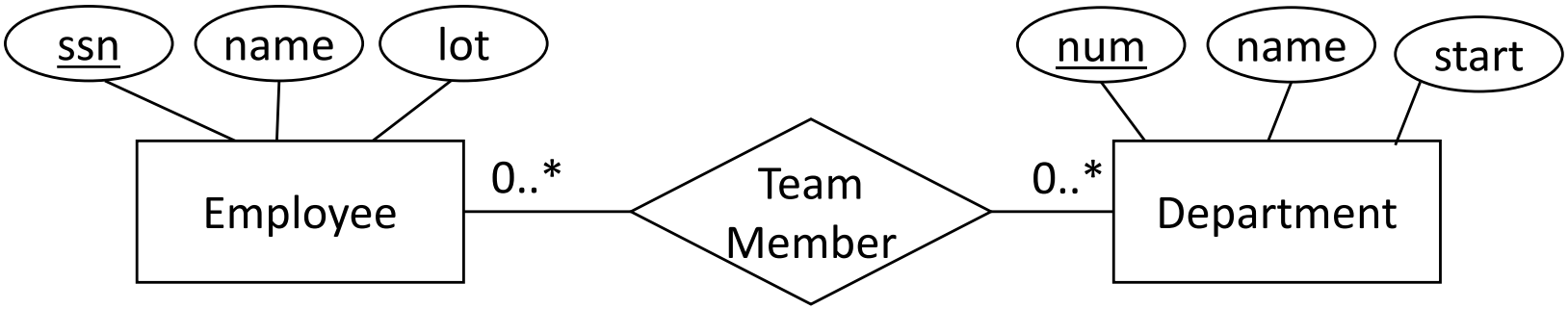
# Translating Relationship Sets (2/7)

- For many-to-many relationship sets
  - create a new table to represent the relationship
- For example:

TeamMember(ssn, num)

... with two foreign keys

Employee(ssn, name, lot)
Team(num, name, start)

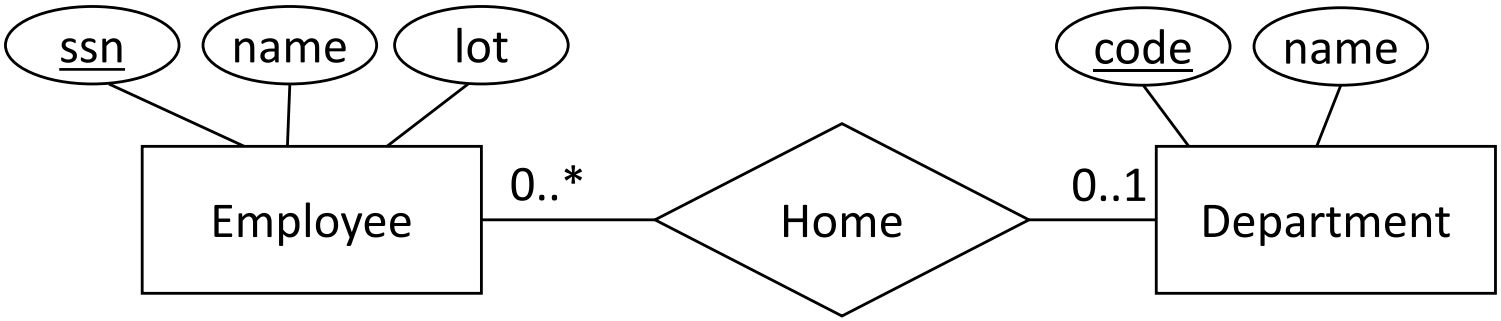


# Translating Relationship Sets (3/7)

- For one-to-many relationship sets
  - introduce a foreign key to the “many” side of the relationship
- For example:

  Department(<u>code</u>, name)
  Employee(<u>ssn</u>, name, lot, **homedept**)

  ... where **homedept** is a foreign key (referencing Department code)

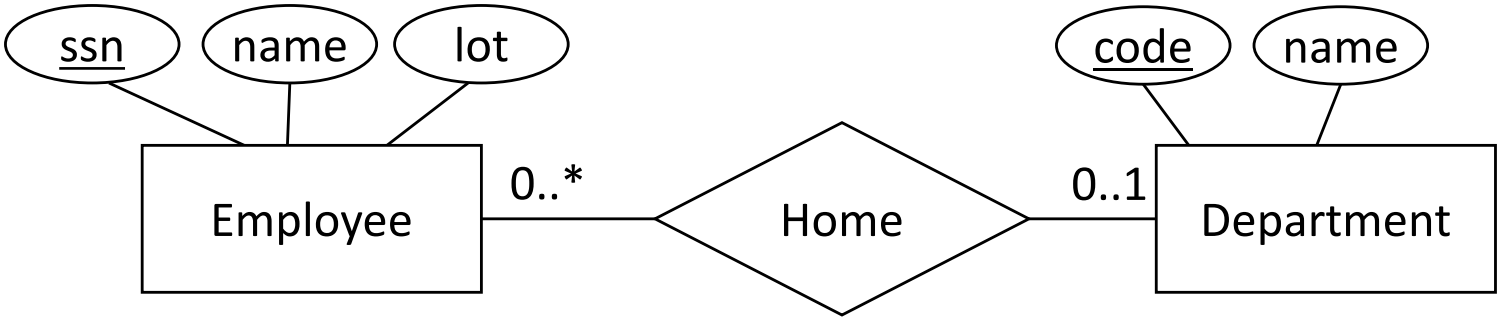


# Translating Relationship Sets (4/7)

- Alternatively, for one-to-many relationship sets
  - Create a new table (like in many-to-many relationships)
  - For example:

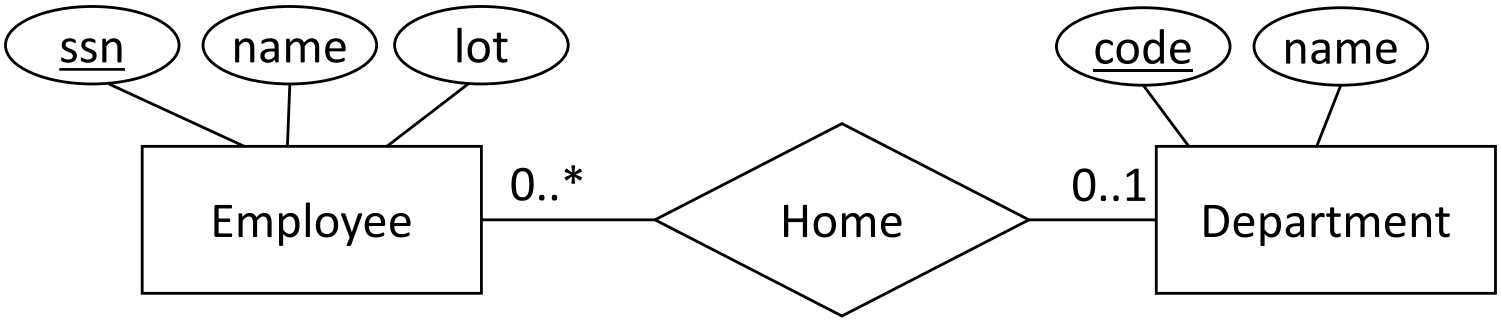
HomeDepartment(<u>ssn</u>, code) ... note that ssn is the key
Department(<u>code</u>, name)
Employee(<u>ssn</u>, name, lot)

What are the tradeoffs between these approaches? ...



# Translating Relationship Sets (5/7)

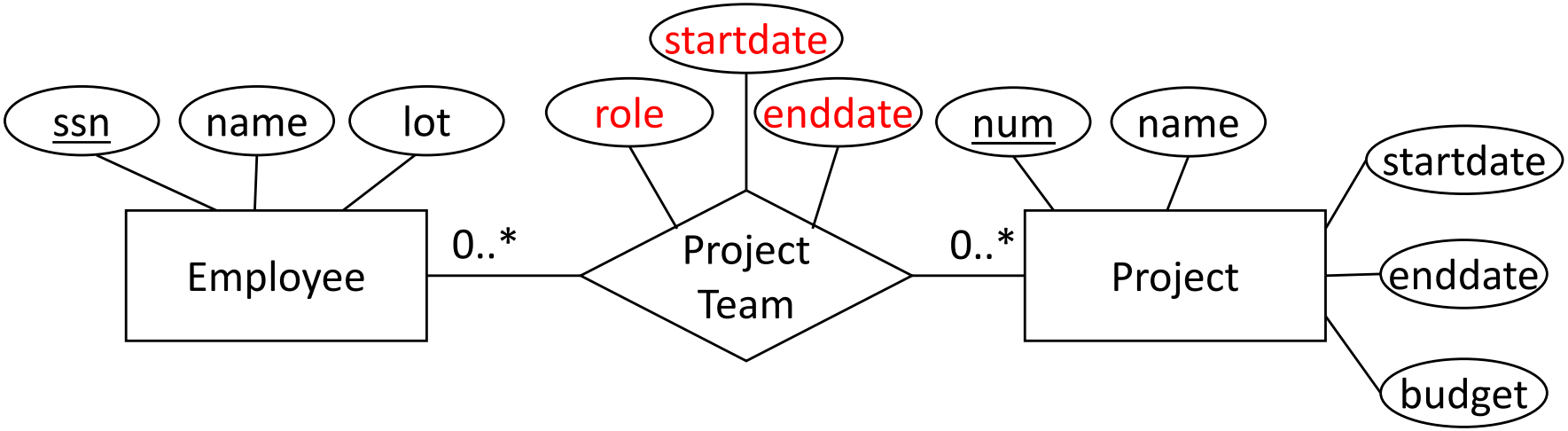
- Creating a new table from one-to-many relationships
  - requires a Join to obtain an employees home department
  - Each Employee **ssn** value associated with a home department is ***stored twice***
  - ... in the Employee and HomeDepartment table



# Translating Relationship Sets (6/7)

- When a many-many relationship set has attributes
  - Put them in the table that represents the relationship

ProjectTeam(num, ssn, role, startdate, enddate)
Project(num, name, startdate, enddate, budget)
Employee(ssn, name, office)



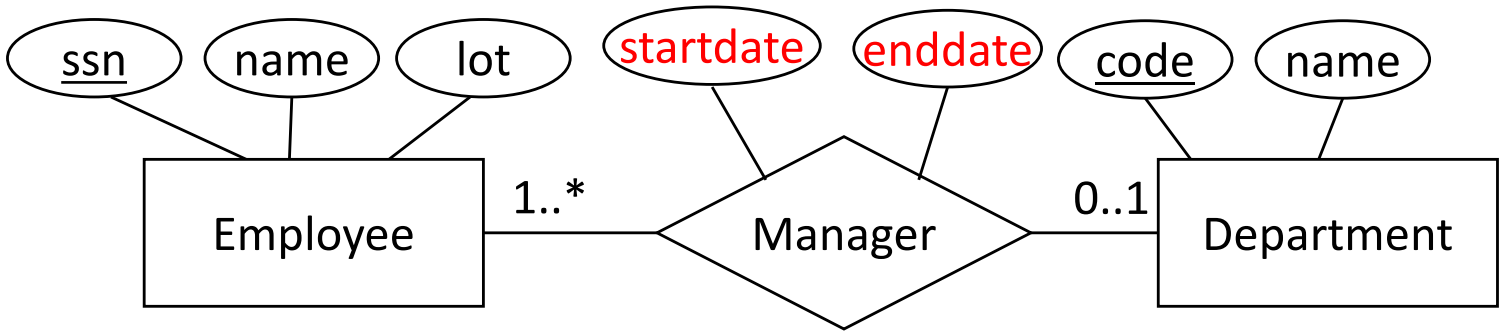
# Translating Relationship Sets (7/7)

- When a one-to-many relationship set has attributes
  - Add them to the table where the relationship is represented

Department(<u>code</u>, name, manager, startdate, enddate)
Employee(<u>ssn</u>, name, office, dept)

or else …

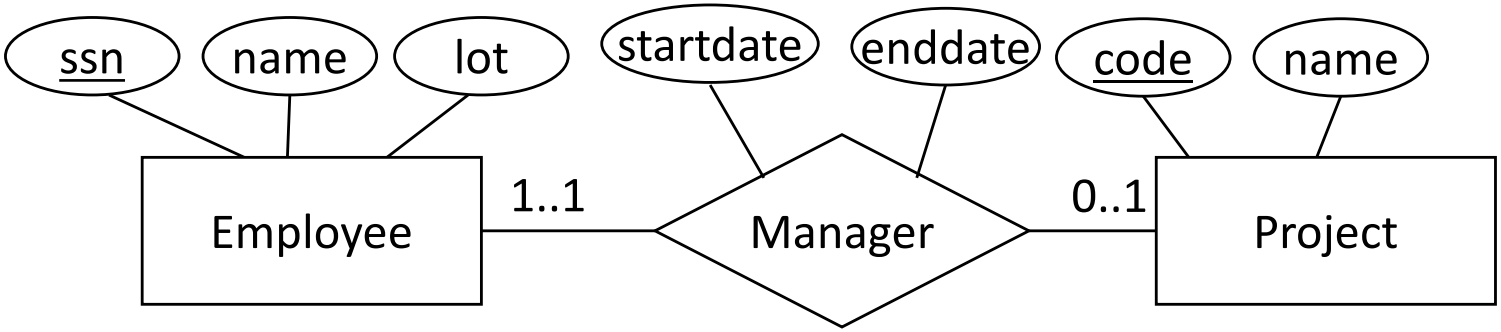
Manager(<u>code</u>, ssn, startdate, enddate)



# Participation Constraints in SQL

- We can require any table to be in a binary relationship using a foreign key
  - by constraining the attribute to be NOT NULL

```
CREATE TABLE Department (
   code INTEGER,    name VARCHAR(20),    manager VARCHAR(9) NOT NULL,
   startdate DATE,    enddate DATE,              PRIMARY KEY (code),
   FOREIGN KEY (manager) REFERENCES Employee,
   ON DELETE NO ACTION
);
```



# Translating Weak Entity Sets

- Weak entity sets and identifying relationship sets are translated into *a single table*
- We must include the key of the strong entity as a foreign key
- The key for the table is the key of the strong entity plus the partial key
- When the owner entity is deleted, all owned weak entities must also be deleted

```
CREATE TABLE Policy (
    name VARCHAR(20),
    age INTEGER,
    ssn VARCHAR(11) NOT NULL,
    PRIMARY KEY (name, ssn) REFERENCES Employee,
    ON DELETE CASCADE
);
```

# Summary of the Translation [Elmasri & Navathe]

- Create table and choose key for each entity set, include (single-valued) attributes
- Create table for each weak entity set, include (single-valued) attributes and the key of the owner as a foreign key. Set the key as foreign key plus partial key.
- For each one-to-one relationship set, add a foreign key to one of the entity sets involved in the relationship (a foreign key to the other entity in the relationship)*
- For each one-to-many relationship set, add a foreign key to the entity set on the many side of the relationship (to reference the entity set on the one side of the relationship)*
- For each many-to-many relationship set, create a new table. Include a foreign key for each participant entity set in the relationship set.The key for the new table is the set of all such foreign keys.
- For each multi-valued attribute, construct a separate table. Repeat the key for the entity in this new table. It serves as both a key for this table and a foreign key to the original table for the entity.

* Unless the relationship set has attributes, in which case create a new table for the relationship set

# For Next Week

- Review – Quiz on the material
  - Ch. 2-2.5
  - Ch. 3.5
- Reading assignments
  - Ch. 19 to 19.6
- Be sure you understand