Database Design

How do you like to see the Spy Database? As tables?

Agent(agent_id,first,middle,last,address,city,country,salary,clearance_id)

Mission(mission_id, name, access_id, team_id, mission_status)

Affiliation(aff_id, title, description)

AffiliationRel(aff_id, agent_id, affiliation_strength)

LanguageRel(lang_id, agent_id)

Language(lang_id, language)

SecurityClearance(sc_id, sc_level, description)

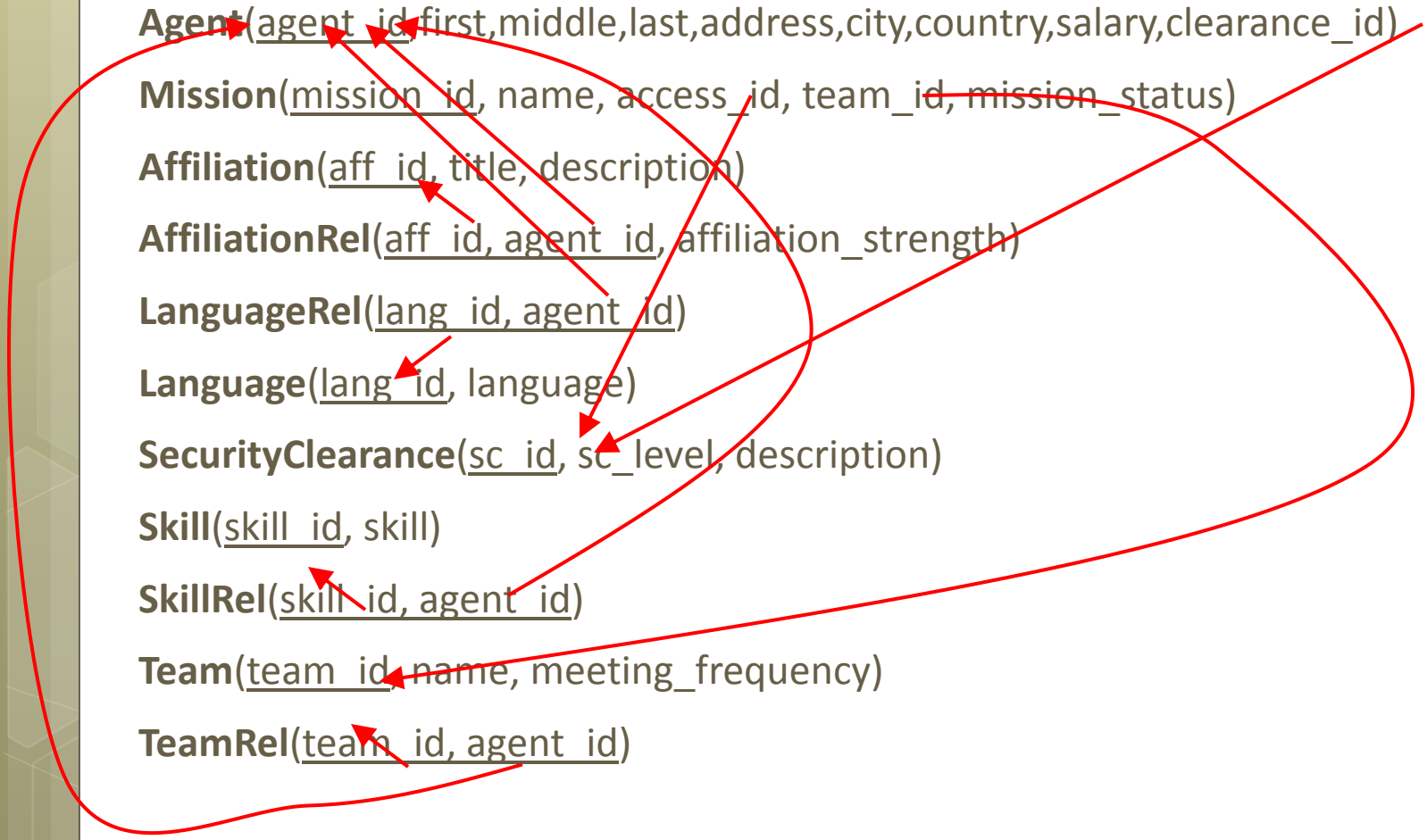
Skill(skill_id, skill)

SkillRel(skill_id, agent_id)

Team(team_id, name, meeting_frequency)

TeamRel(team_id, agent_id)

How do you like to see the Spy Database? As tables with foreign keys?



Agent(agent_id,first,middle,last,address,city,country,salary,clearance_id)

Mission(mission_id, name, access_id, team_id, mission_status)

Affiliation(aff_id, title, description)

AffiliationRel(aff_id, agent_id, affiliation_strength)

LanguageRel(lang_id, agent_id)

Language(lang_id, language)

SecurityClearance(sc_id, sc_level, description)

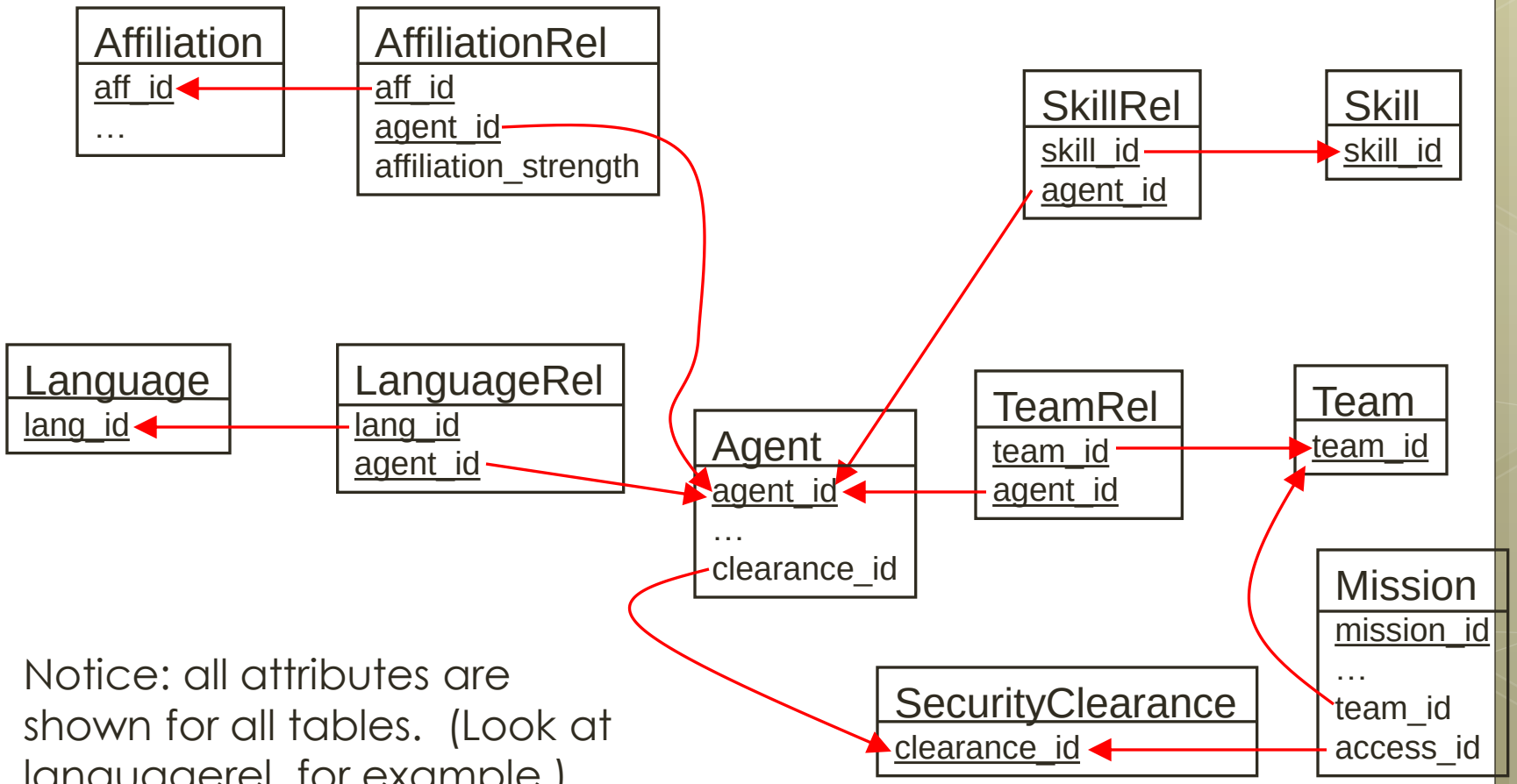
Skill(skill_id, skill)

SkillRel(skill_id, agent_id)

Team(team_id, name, meeting_frequency)

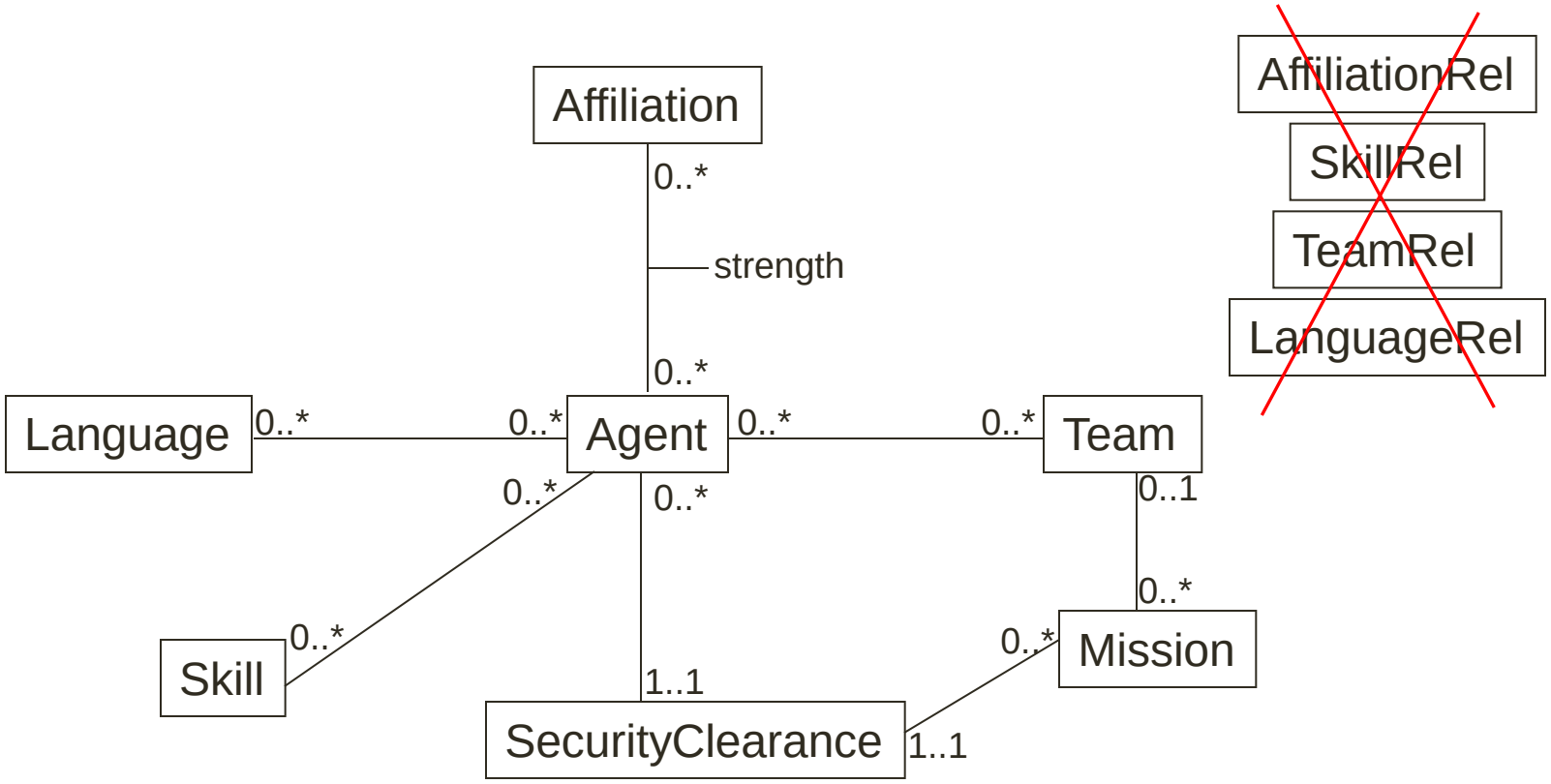
TeamRel(team_id, agent_id)

How do you like to see the Spy Database? Relationship Diagram – of tables?



Notice: all attributes are shown for all tables. (Look at languagerel, for example.)

How do you like to see the Spy Database? Entity-Relationship Diagram (ERD)?

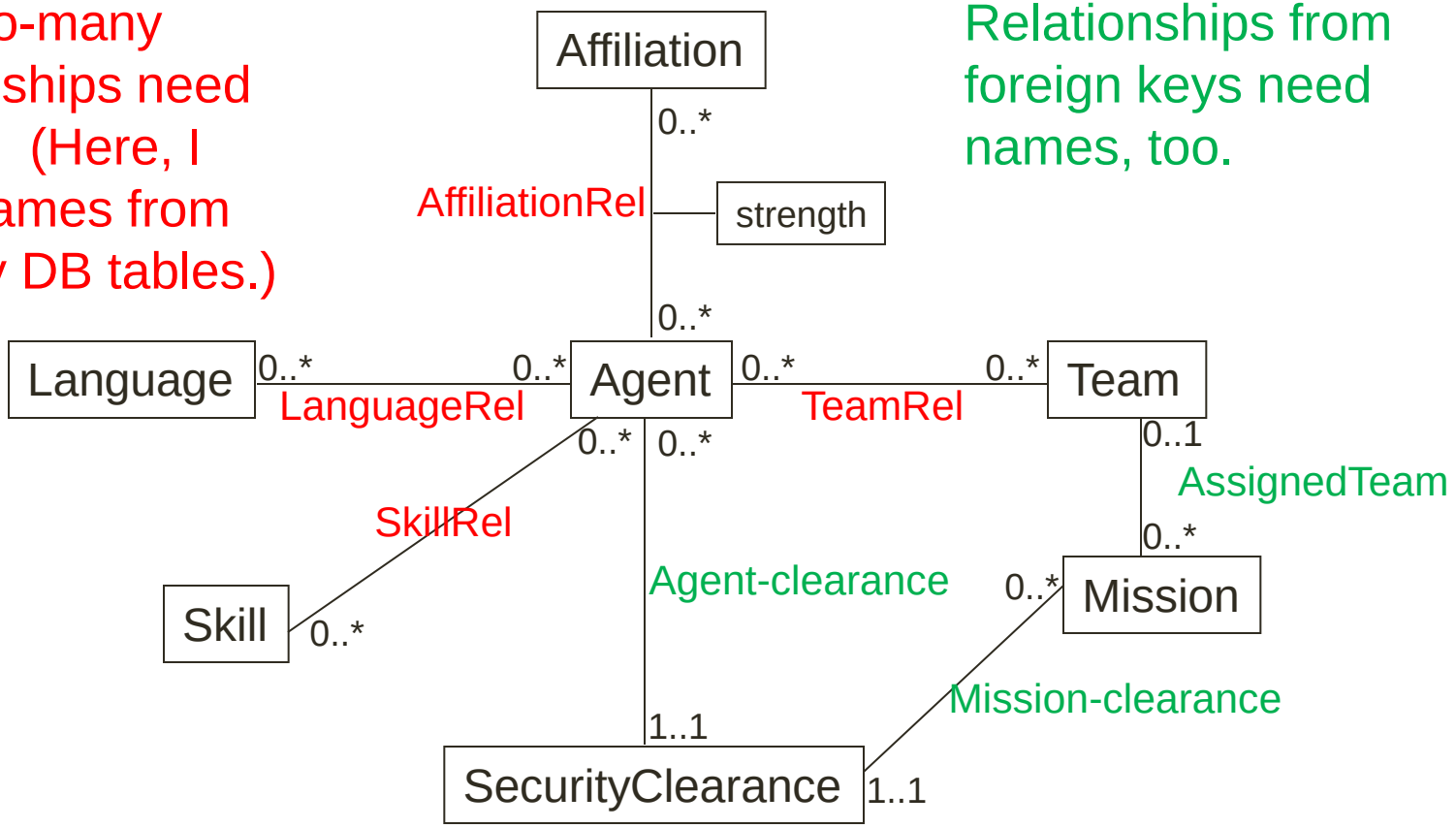


Rectangles: Entities Lines: Relationships (with cardinalities)

Entity-Relationship Diagram (ERD)

Many-to-many relationships need names. (Here, I used names from the Spy DB tables.)

Relationships from foreign keys need names, too.



Notice: relationships are just lines; no attributes needed. languagerel just connects language & agent.

Conceptual Database Design Using the Entity-Relationship (ER) Model

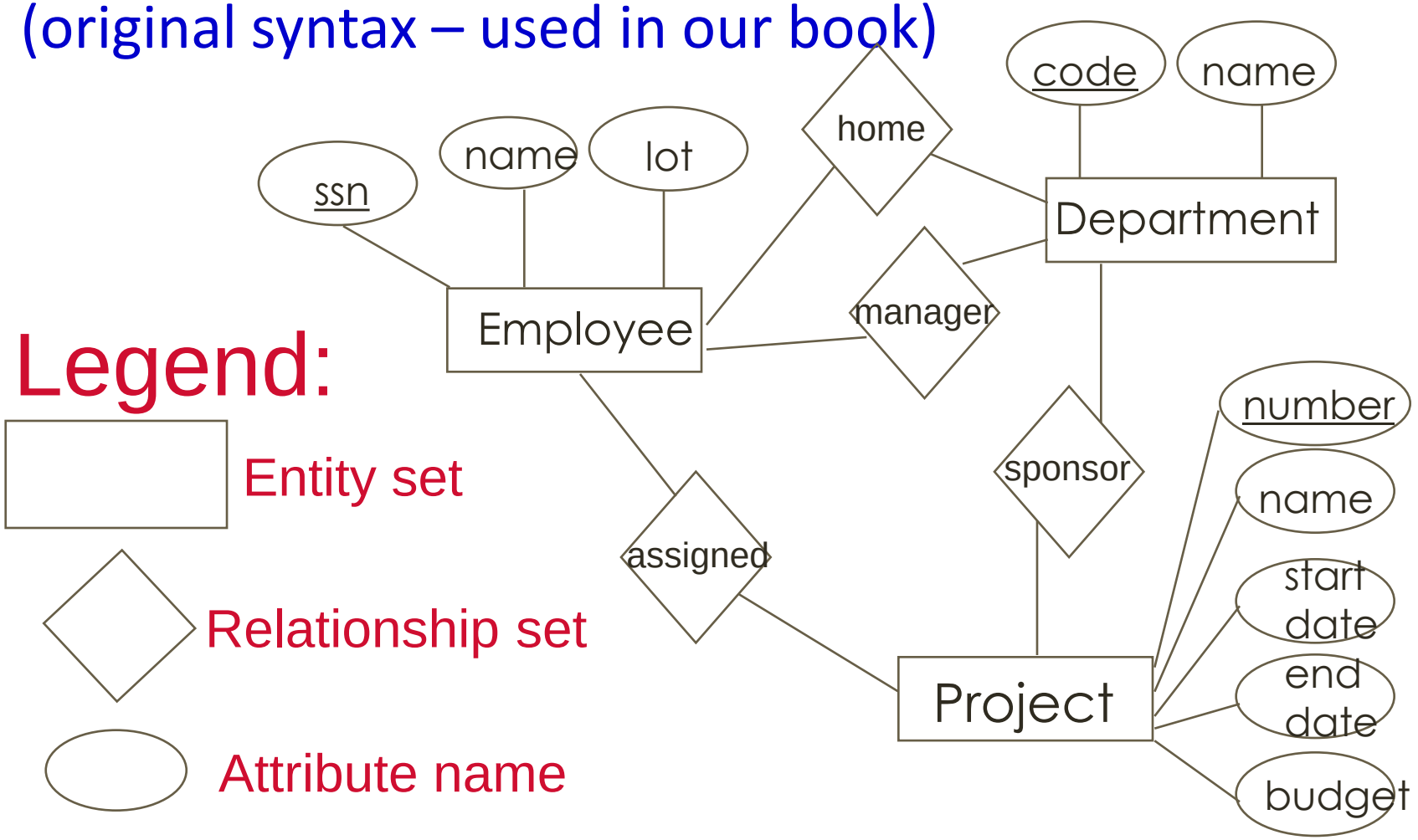
Overview of Database Design

- Conceptual design: (ER Model is used for this.)
  - What are the **entities** and **relationships** we need?
- Logical design:
  - Transform ER design to Relational Schema
- Schema Refinement: (Normalization)
  - Check relational schema for redundancies and related anomalies.
- Physical Database Design and Tuning:
  - Consider typical workloads; (sometimes) modify the database design; select file types and indexes.

Entity-Relationship Model is a different model than the Relational Model

- Relation model has:
  - tables (relations) with attributes, keys, foreign keys, domain definitions for attributes
- Entity-Relationship model has:
  - Entities and entity sets with attributes, keys, and domain definitions for attributes
  - Relationships among entities and relationship sets with cardinality constraints

Entity-Relationship Diagram (original syntax – used in our book)



Definitions

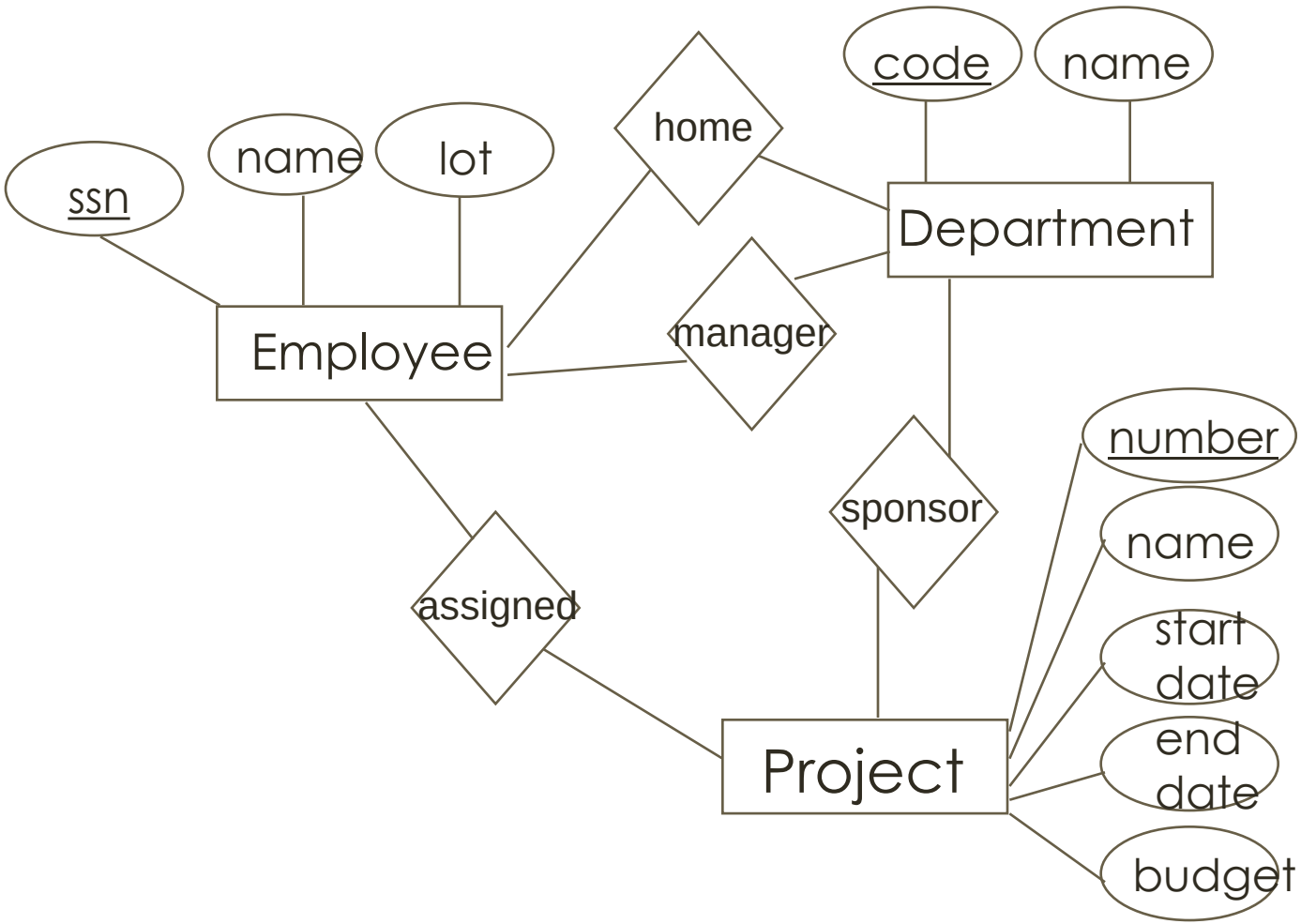
- *Entity*: Real-world object distinguishable from other objects. (An employee named John Smith, for example.)
  - An entity is described using a set of *attributes*.
- *Entity Set*: A collection of similar entities. (All employees in the company, for example.)
- There is a related concept that was not explicitly mentioned in the original ER paper. The entity type is the definition of the entity – the name, the names and types of all the attribtes, and key(s). You can think of the entity types as part of the schema – expressed in the ER model.

Definitions

- *Relationship*: Association among 2 or more entities. John's home department is Pharmacy, for example. John is an entity. Pharmacy is an entity.

- *Relationship Set*: Collection of similar relationships. The home department relationship set consists of all indications of a home department for an employee.
- There is a related concept of relationship type which is the name of the relationship, the indication of the participants in the relationship, the cardinality of the participants, and the role names (if present). Relationship type is part of the schema.

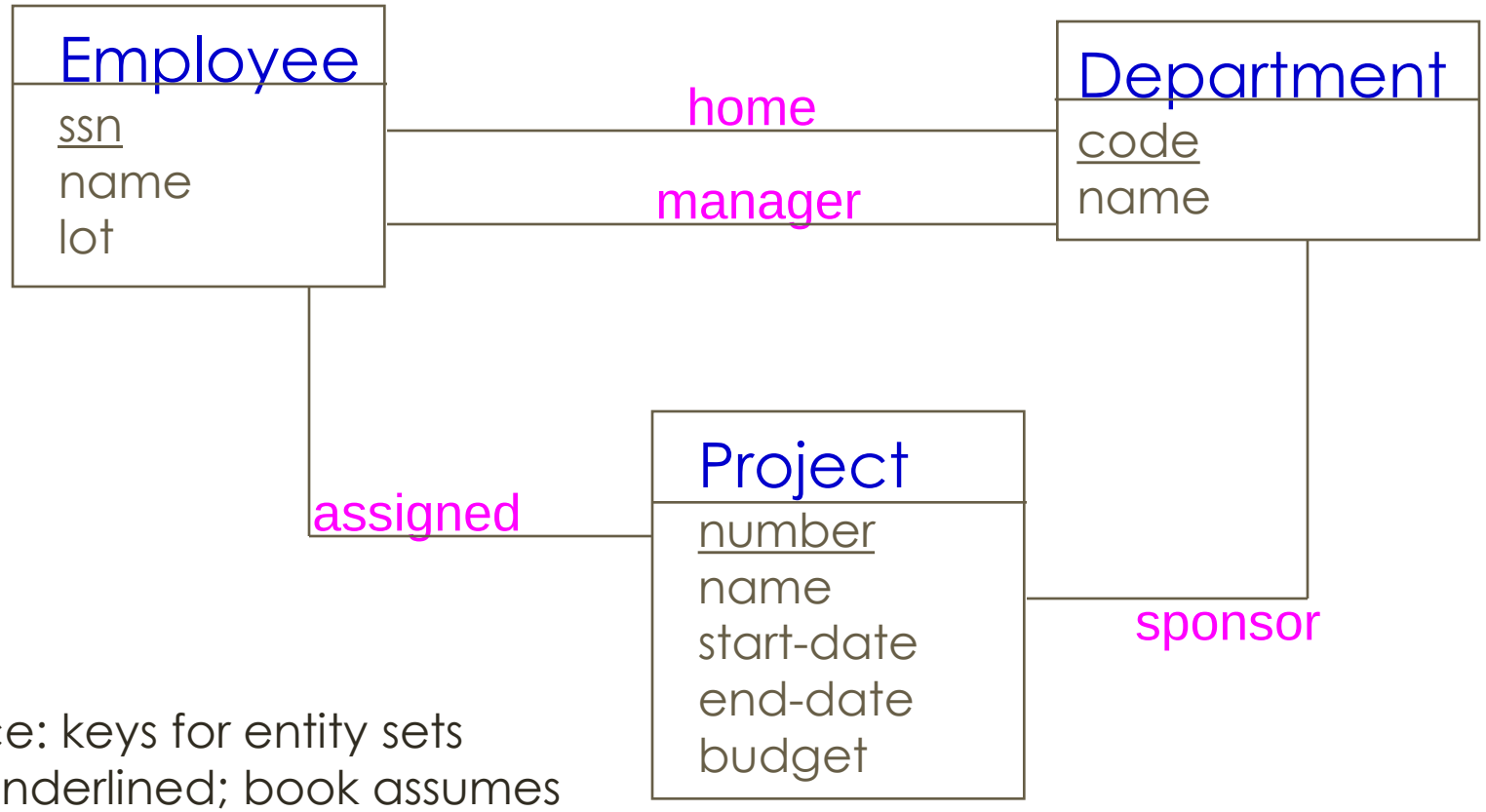
Entity-Relationship Diagram (repeated)



UML notation for the same E-R Diagram

(w/o cardinalities)

UML: Unified Modeling Language – for OO Design



Notice: keys for entity sets are underlined; book assumes just one key for each entity set.

Equivalent Relational Schema

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

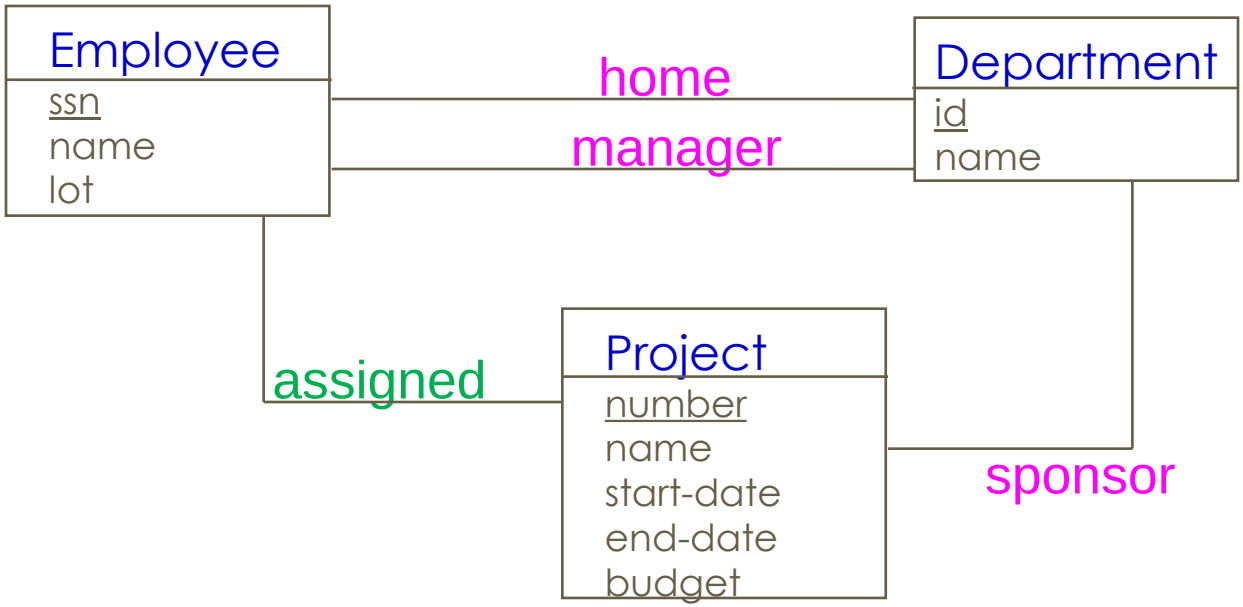
Project-team(ssn, number)

Department (id, name, manager)

Project (number, name, start-date, end-date, budget, sponsor)

extra table to represent a many-to-many relationship

Notice: attributes are added to various tables to represent relationships.



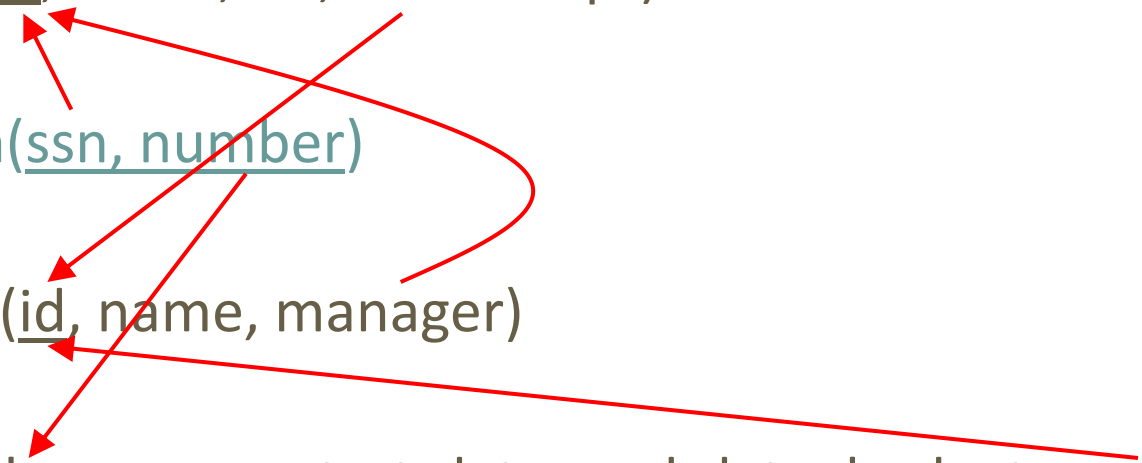
Equivalent Relational Schema - with foreign keys shown

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

Project-team(ssn, number)

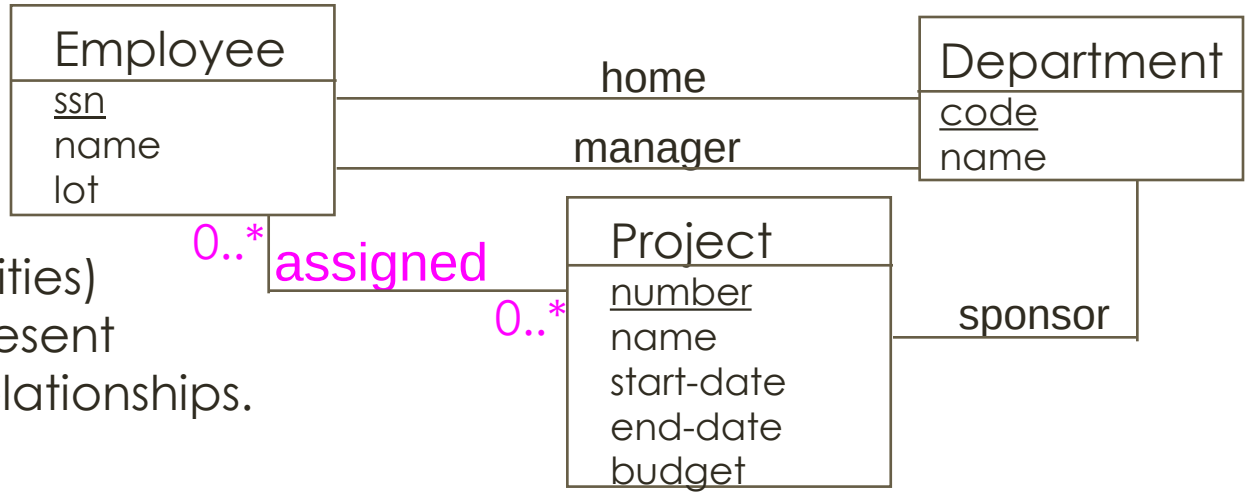
Department (id, name, manager)

Project (number, name, start-date, end-date, budget, sponsor)



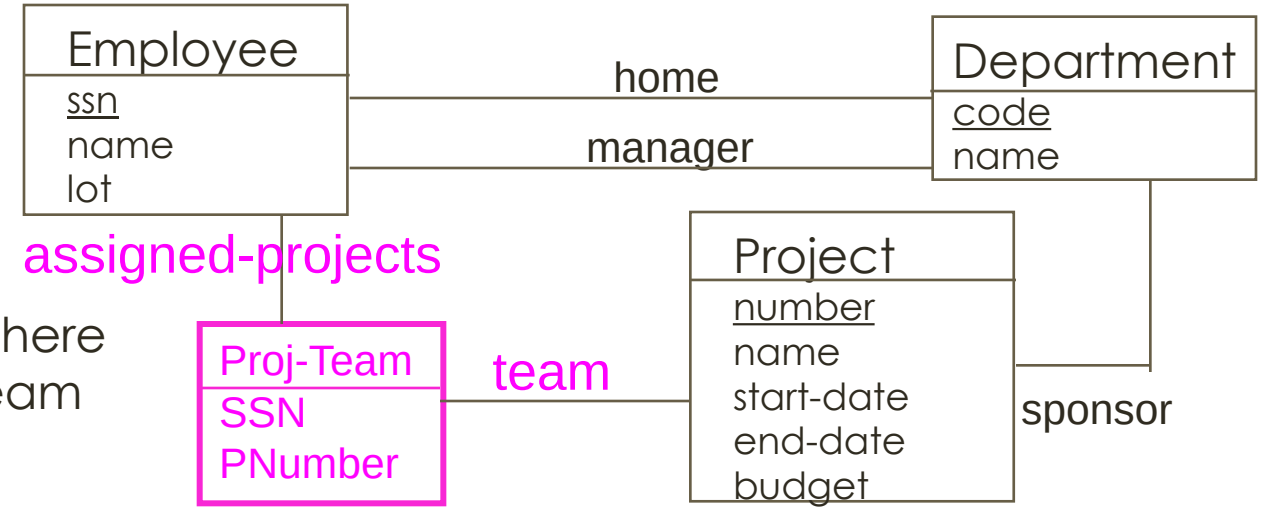
Many-to-many relationship sets (e.g., assigned)

ERD



Just the line (with the right cardinalities) is enough to represent many-to-many relationships.

Relational DB Diagram



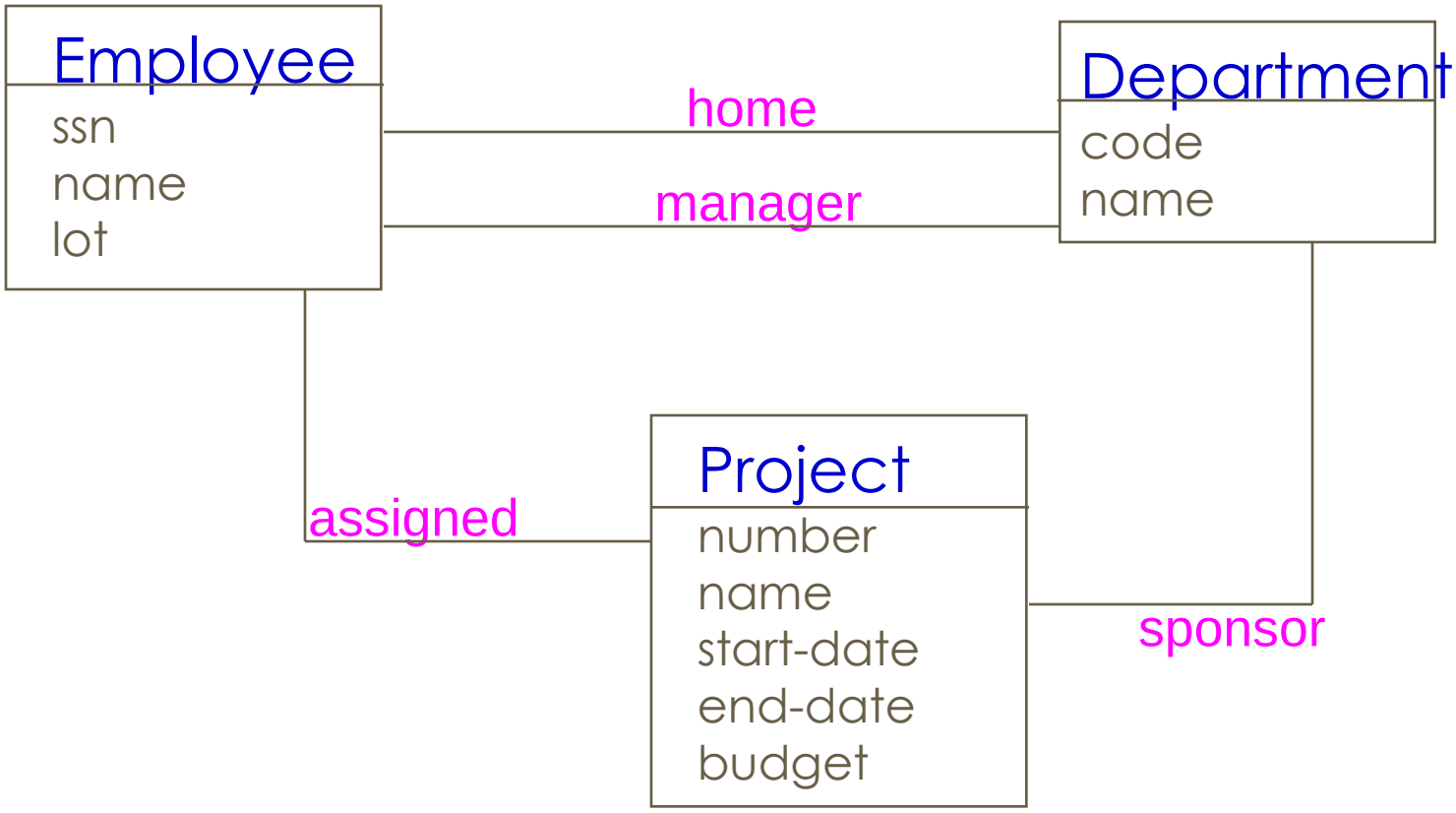
We see a box for each table; thus there is a box for Proj-Team (a table for the many-to-many).

What data do we need to record a relationship?

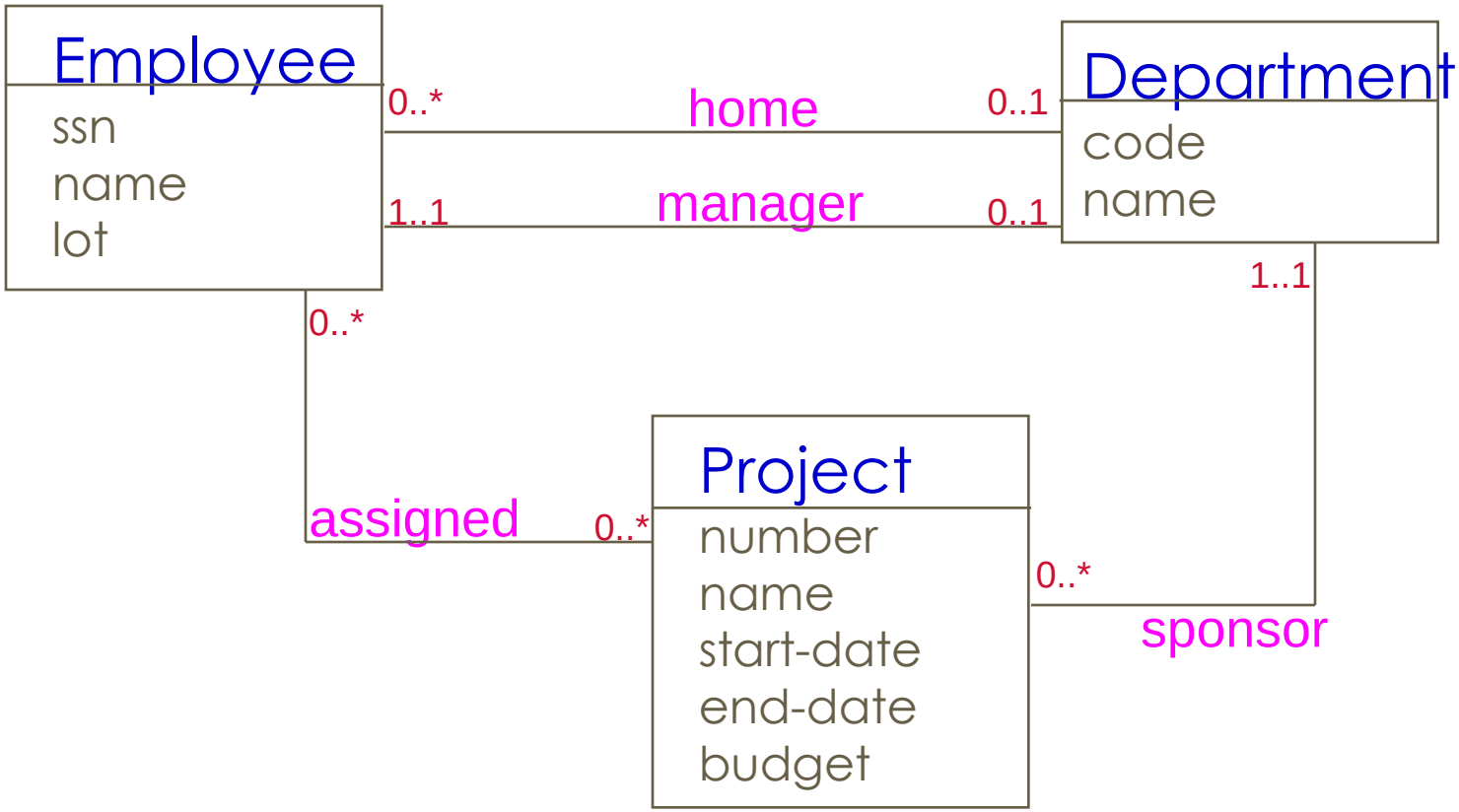
We must indicate which employee and which department are to be connected (for each relationship).

We need the key value for an employee and the key value for the department – stored together – to represent the relationship. (Like skillrel, languagerel, and teamrel.)

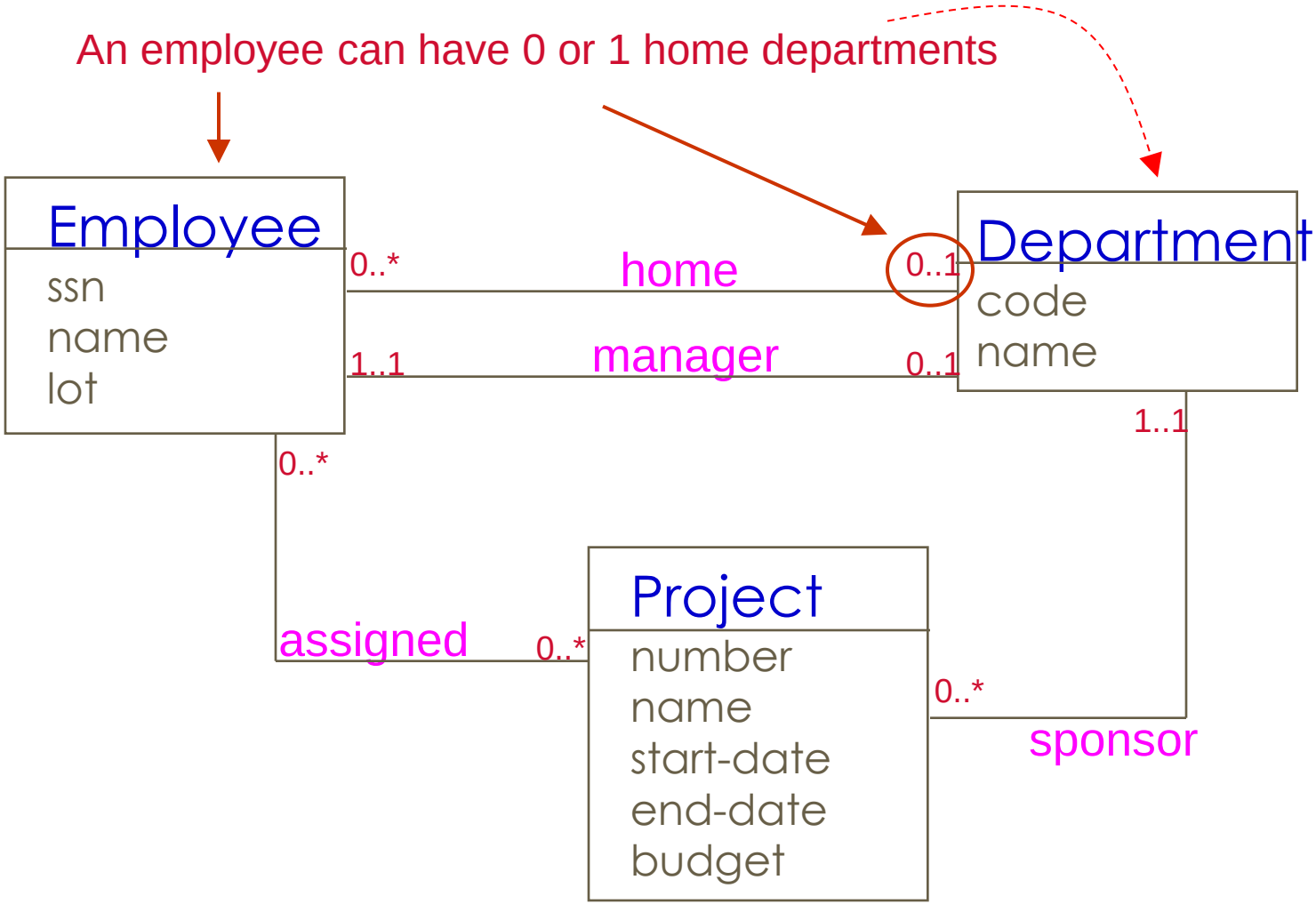
Cardinality Constraints on Relationship sets: How many entities can participate?



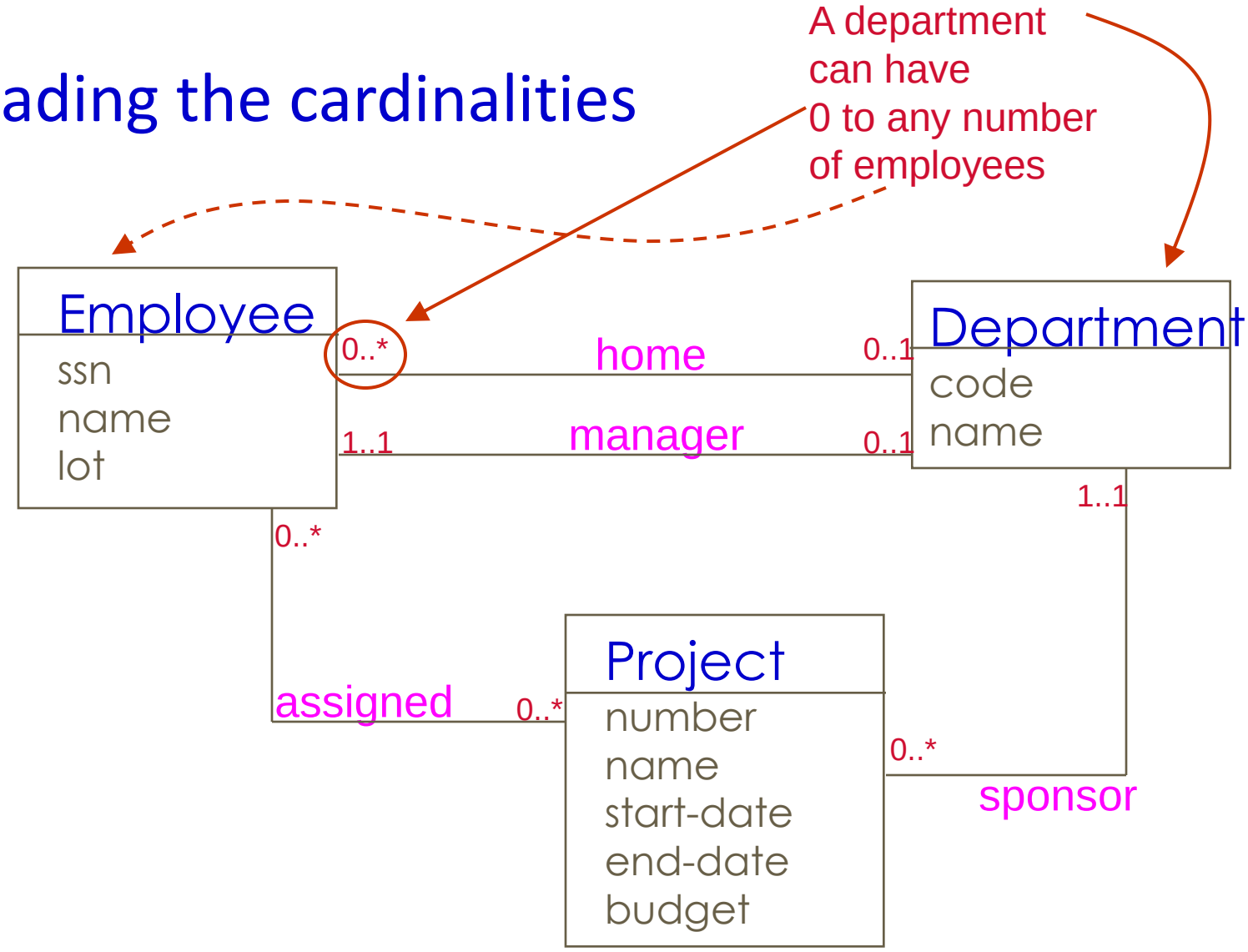
Cardinality Constraints on Relationship sets
How many entities can participate?



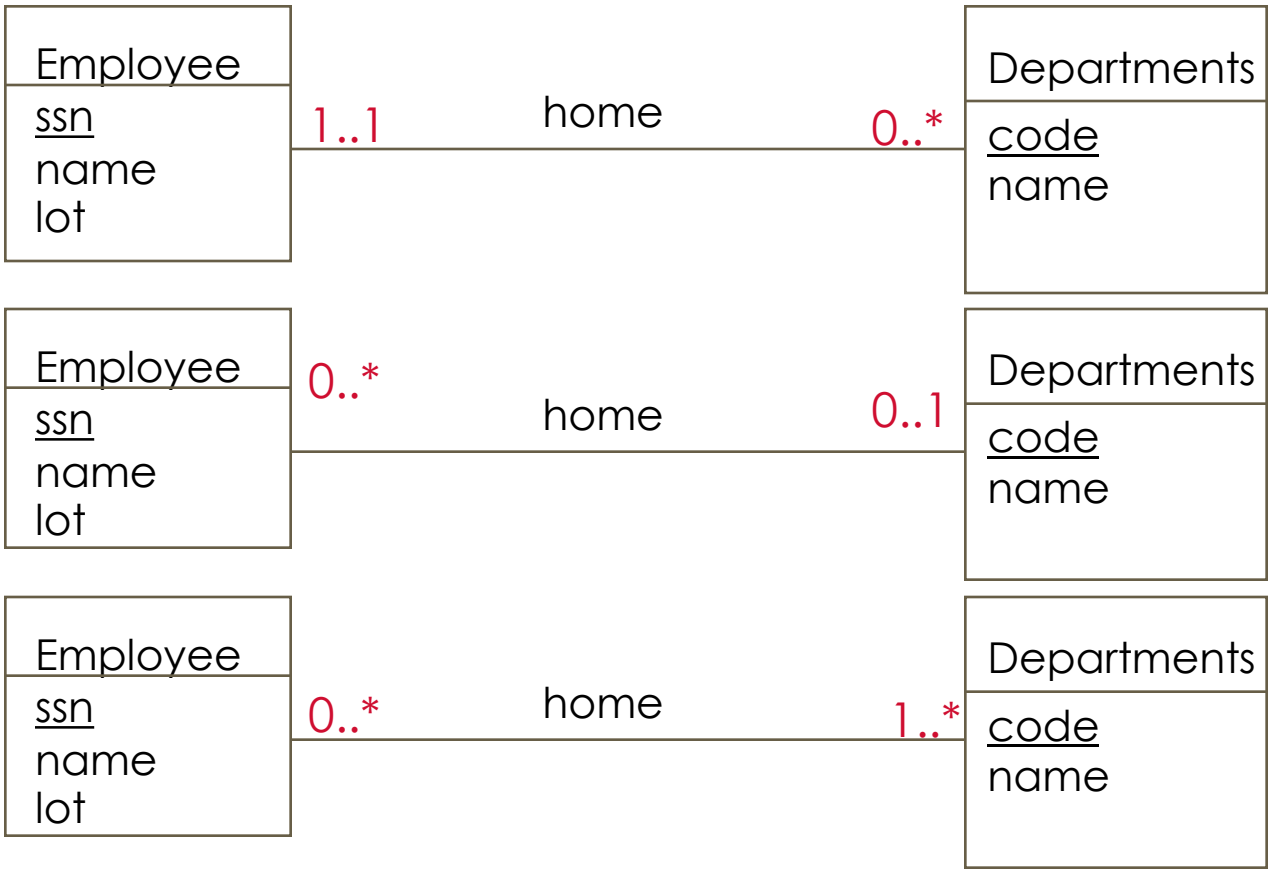
Reading the cardinalities (using UML conventions)



Reading the cardinalities

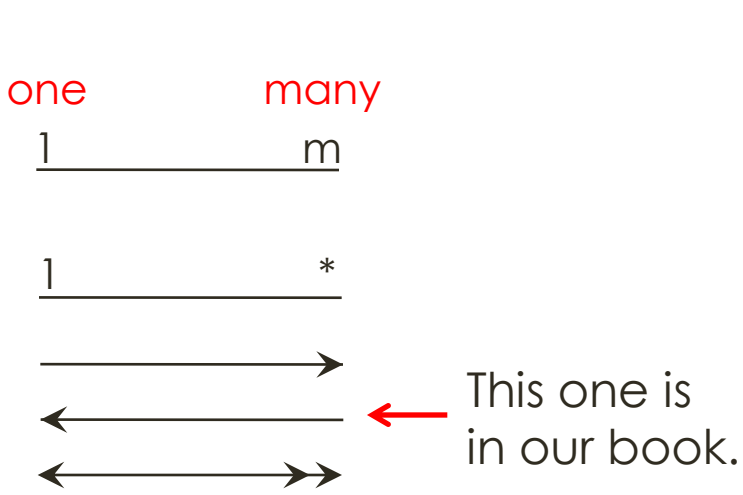


Unified Modeling Language (UML):
Class Diagram

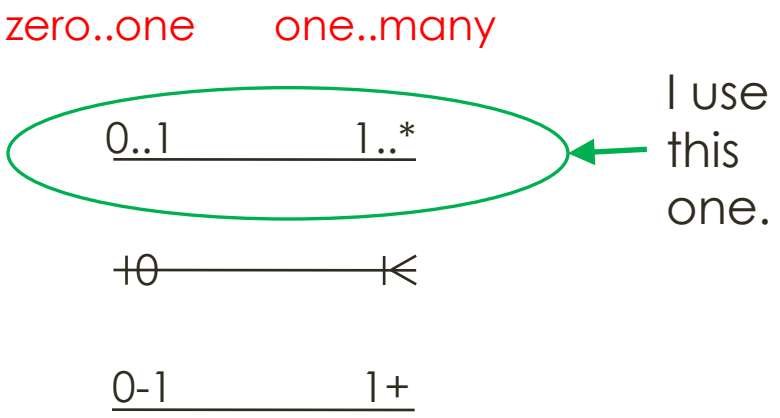


Which one is right? We must discover the semantics of the application!

Various notation for "one-to-many"



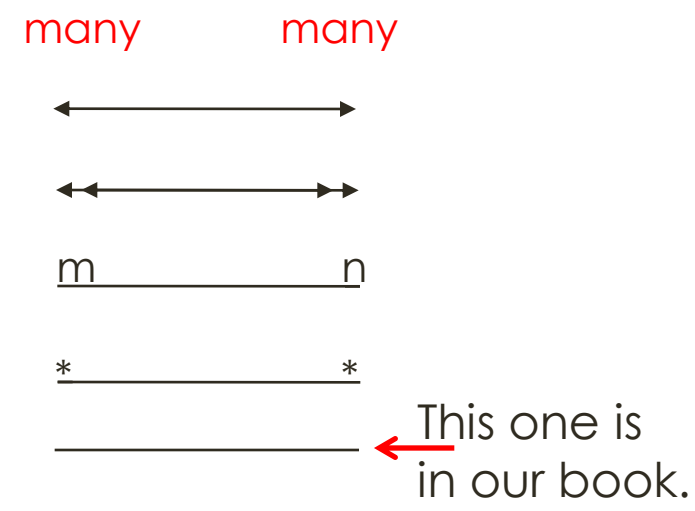
maximum cardinalities only



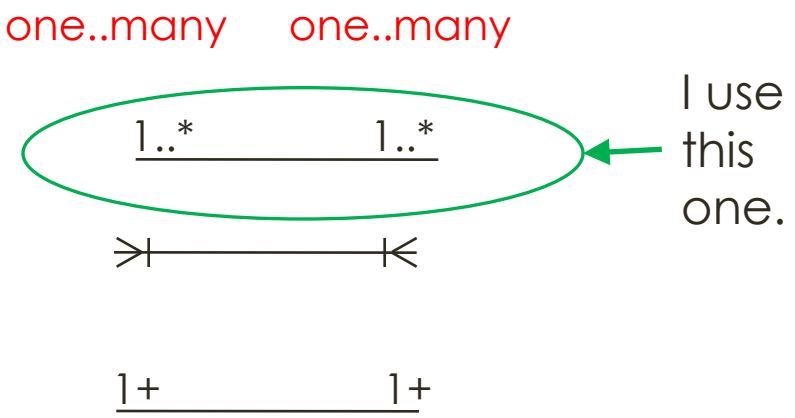
minimum and maximum cardinalities

Note: UML notation allows any sort of cardinality. DBMSs are limited to enforcing a max of 1 or many and a min of 0 or 1.

Various notations for “many-to-many”

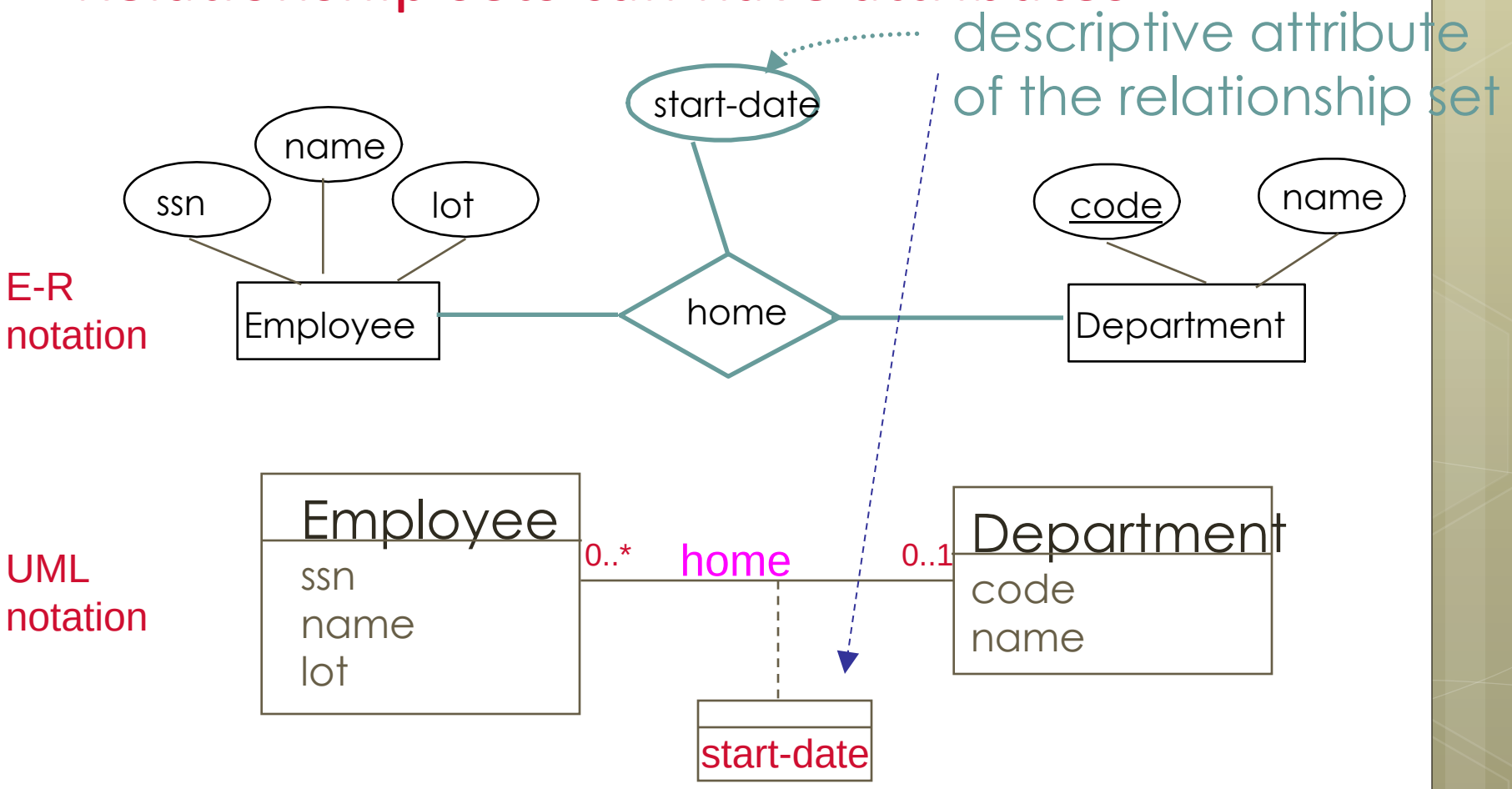


maximum cardinalities only

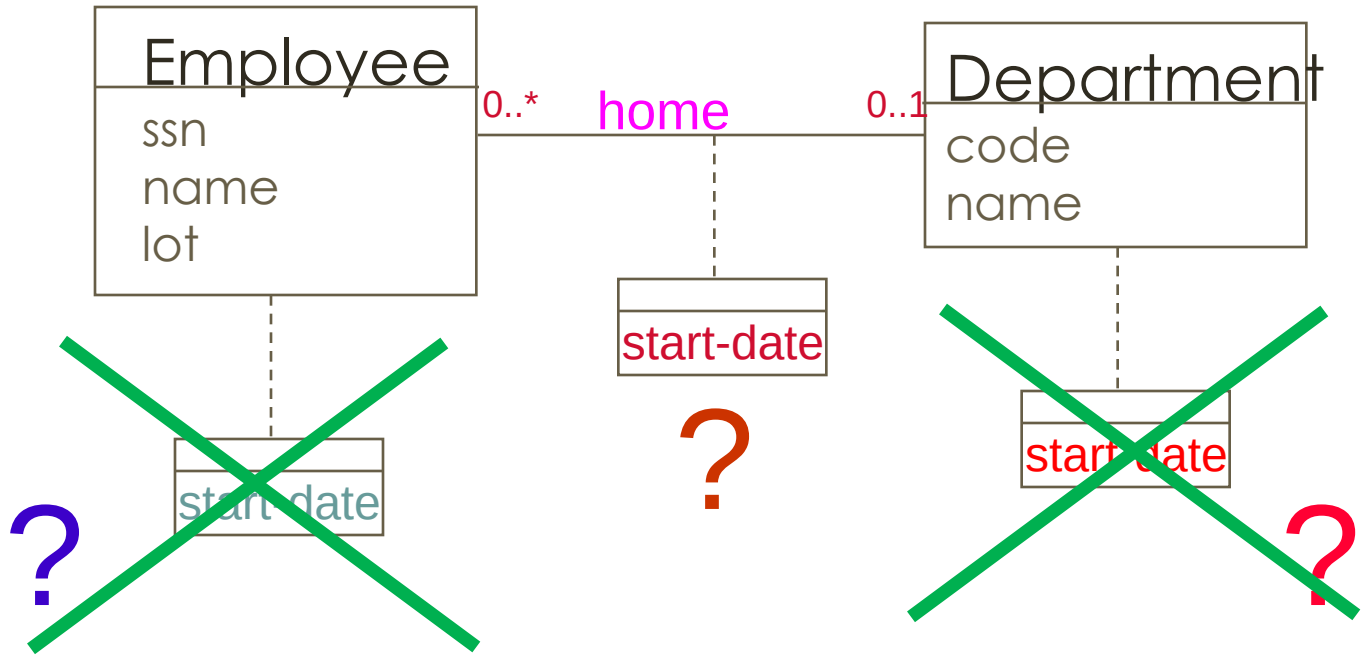


minimum and maximum cardinalities

Relationship sets can have attributes

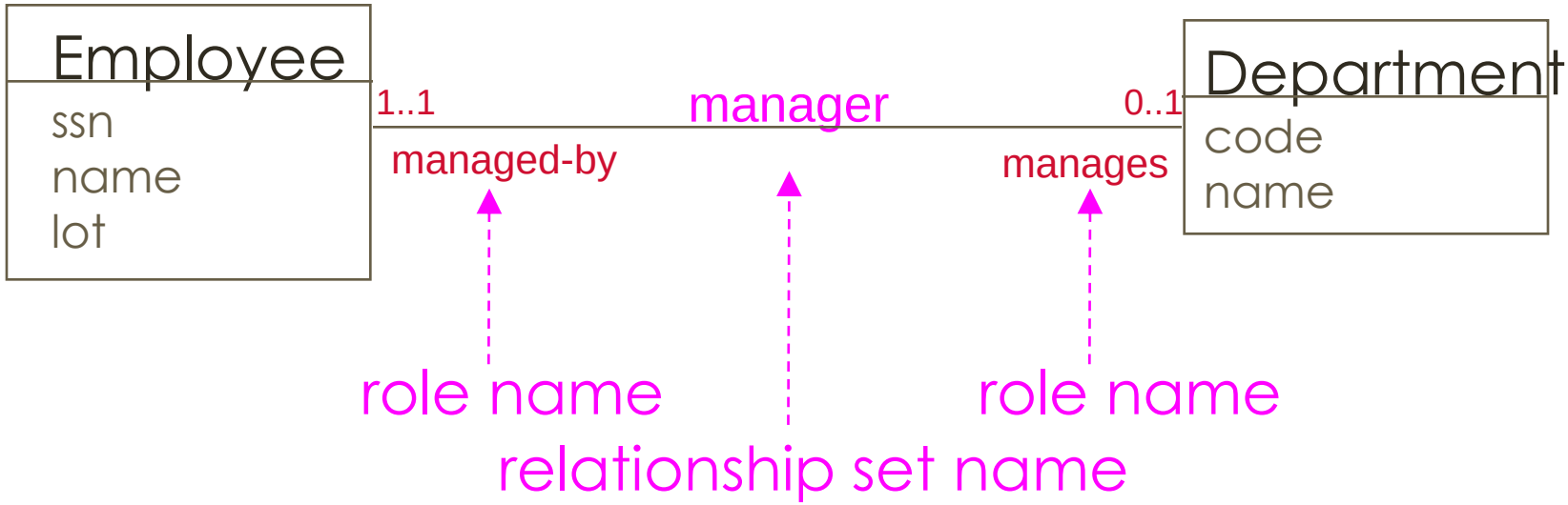


Try all three locations for the attributes; which one makes sense?

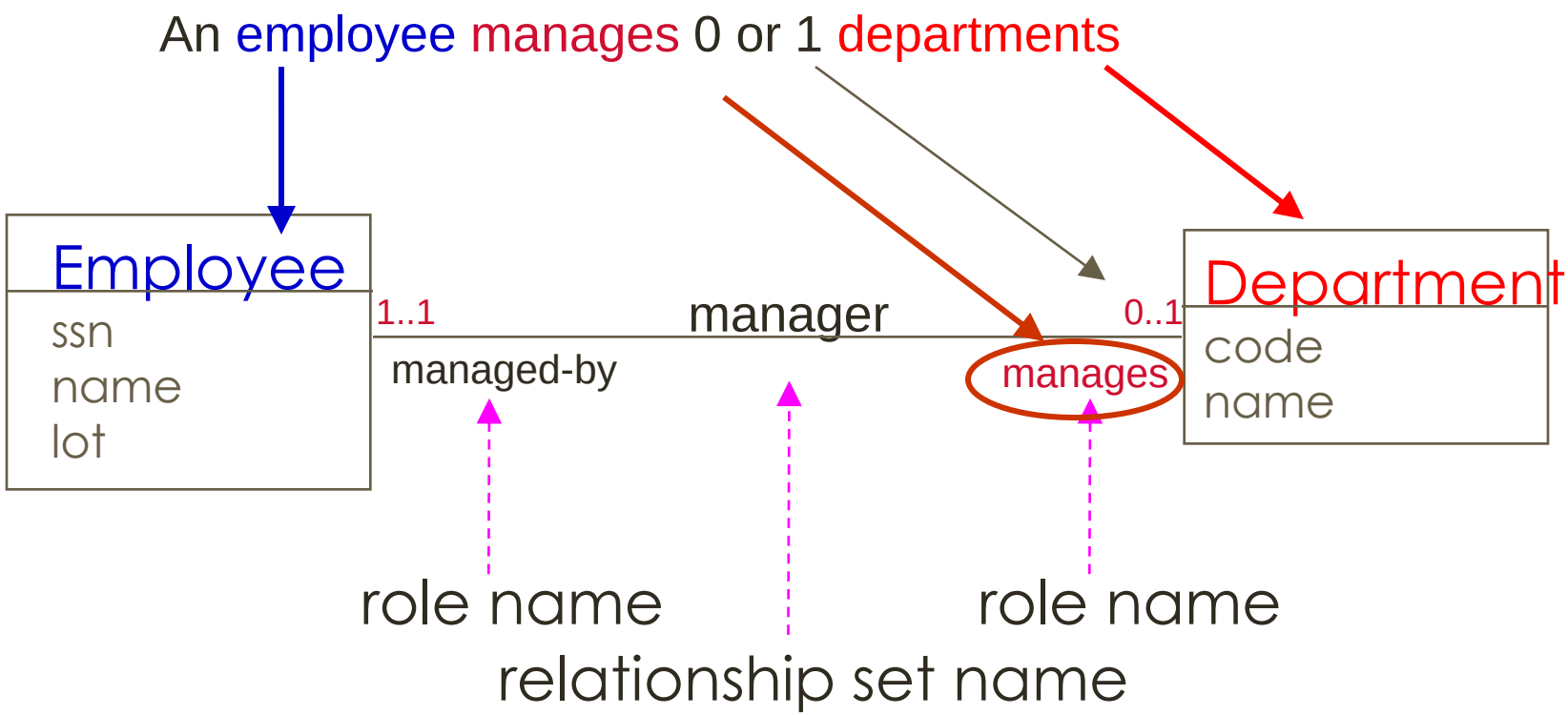


Relationship sets can have role names

(in addition to the name of the relationship set)

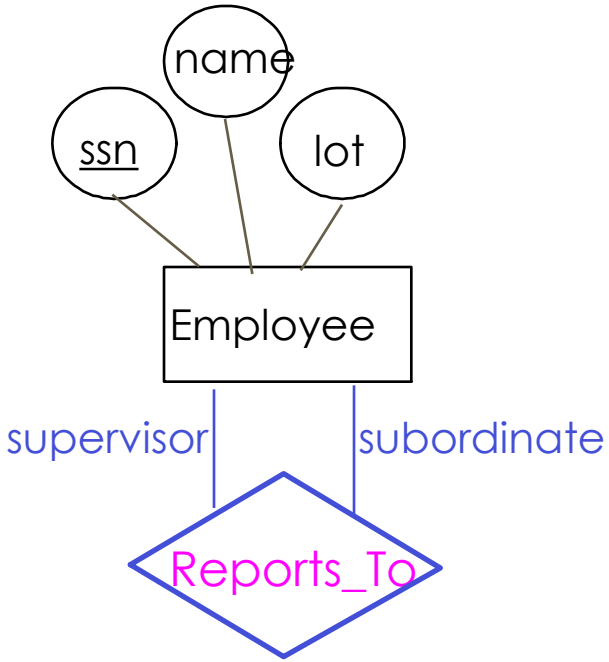


Example: reading role names

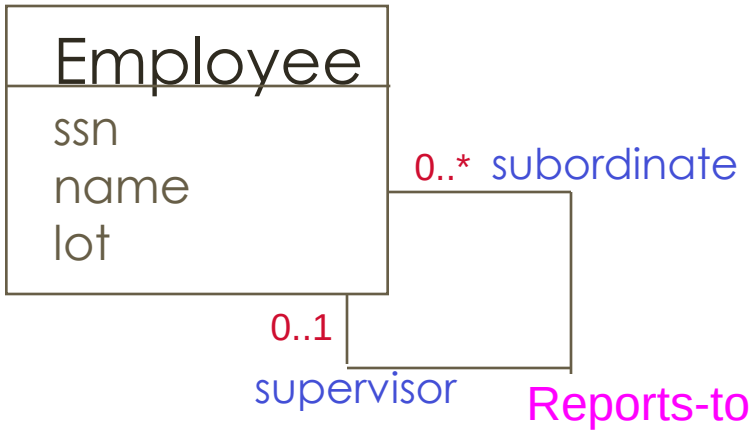


Same entity sets can participate in different "roles" for the same relationship set

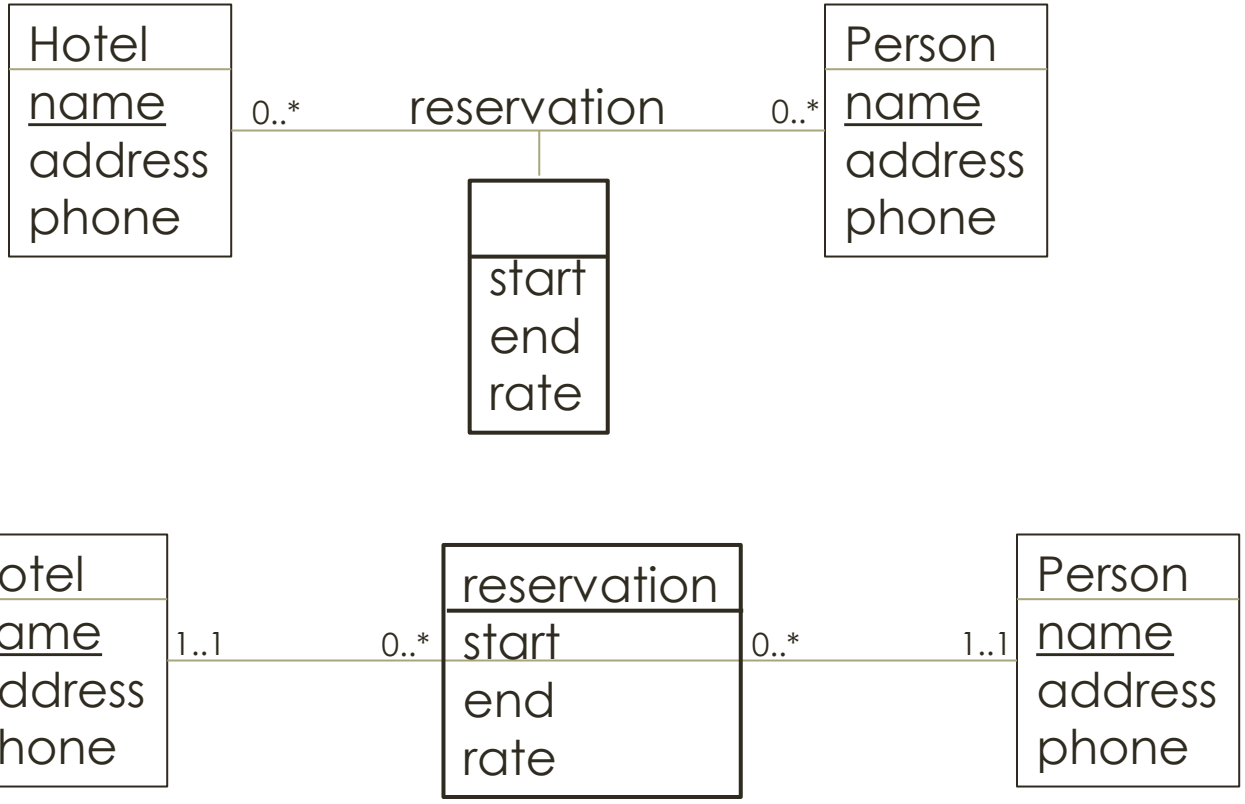
E-R notation



UML notation



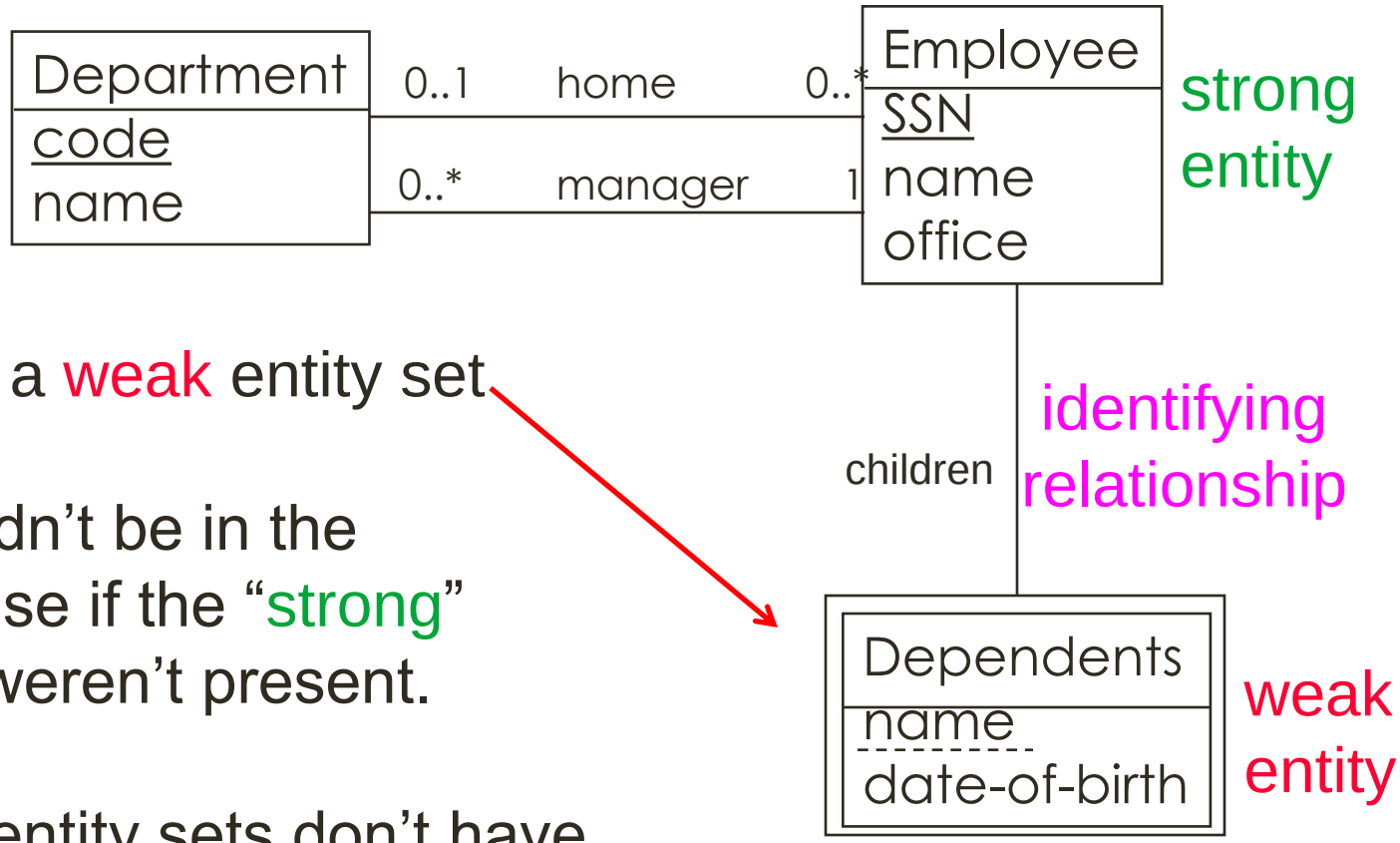
Relationships can be turned into entities
Which model to you prefer?



Exercise 1: Draw an ERD

- Professors have a SSN, a name and an age and their SSNs uniquely identify them.
- Professors teach courses. Each course is coordinated by one professor.
- Courses are uniquely identified by their courseID, and they have a name.
- Choose cardinalities for all your relationships.
- Modify your design so that a course can be taught by a team of one or more professors.
- Modify your design to that you describe courses separate from classes. Add quarters (e.g., Fall 2012) to your design.

Weak Entity Sets (and Identifying Relationship sets)



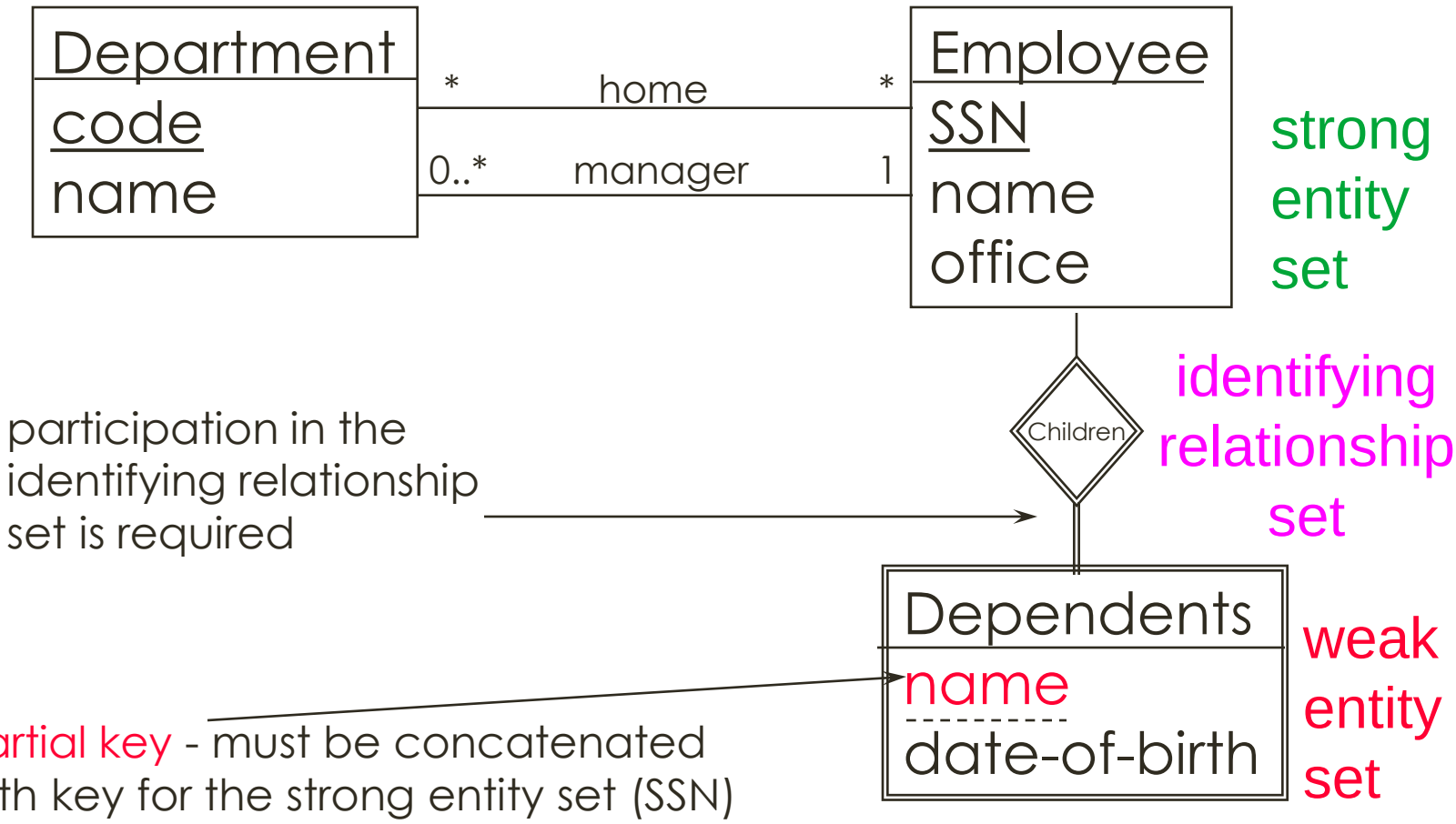
This is a weak entity set

It wouldn't be in the database if the "strong" entity weren't present.

Weak entity sets don't have keys on their own.

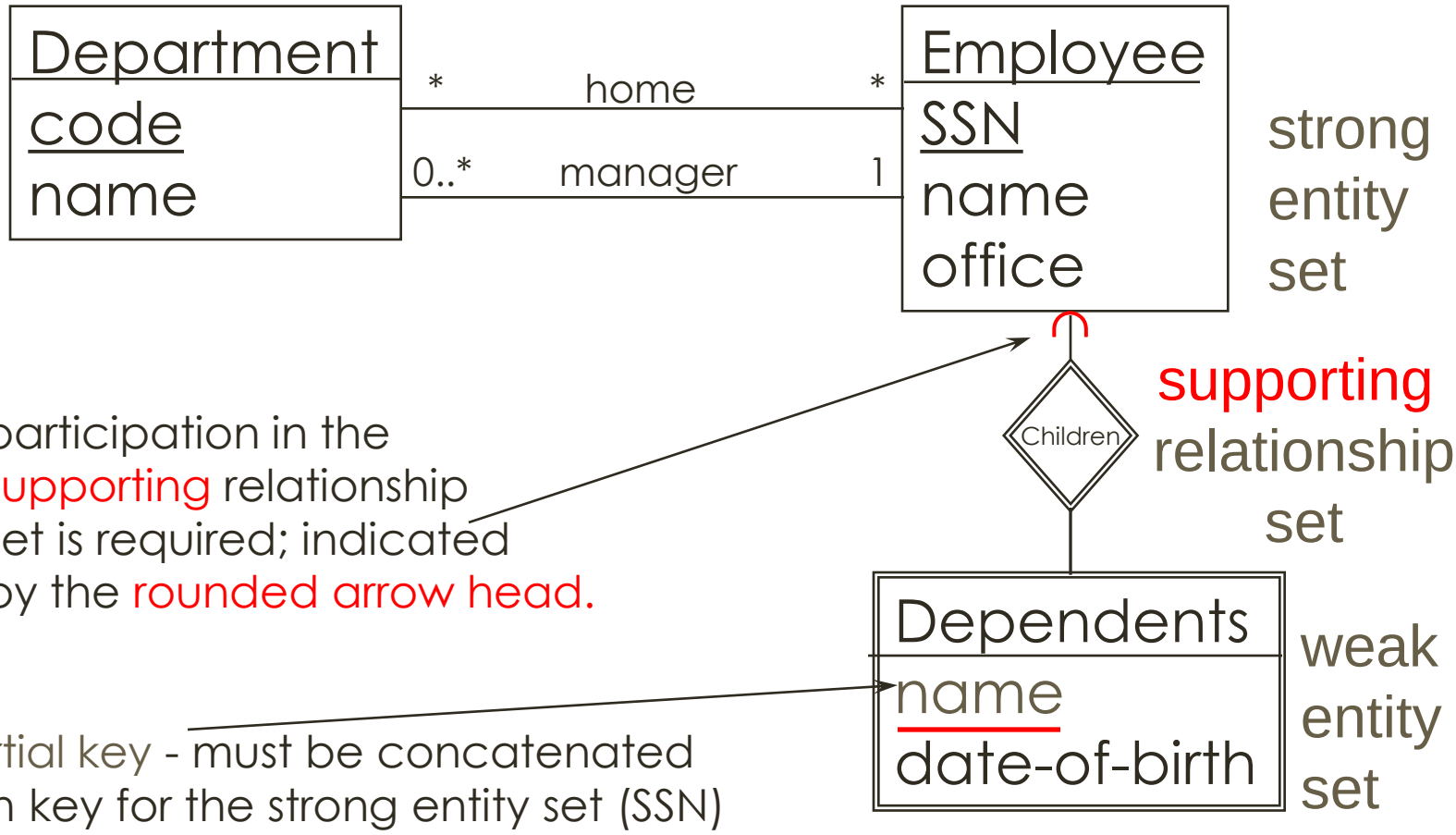
Weak Entity Sets (& Identifying Relationship sets)

Alternative Notation

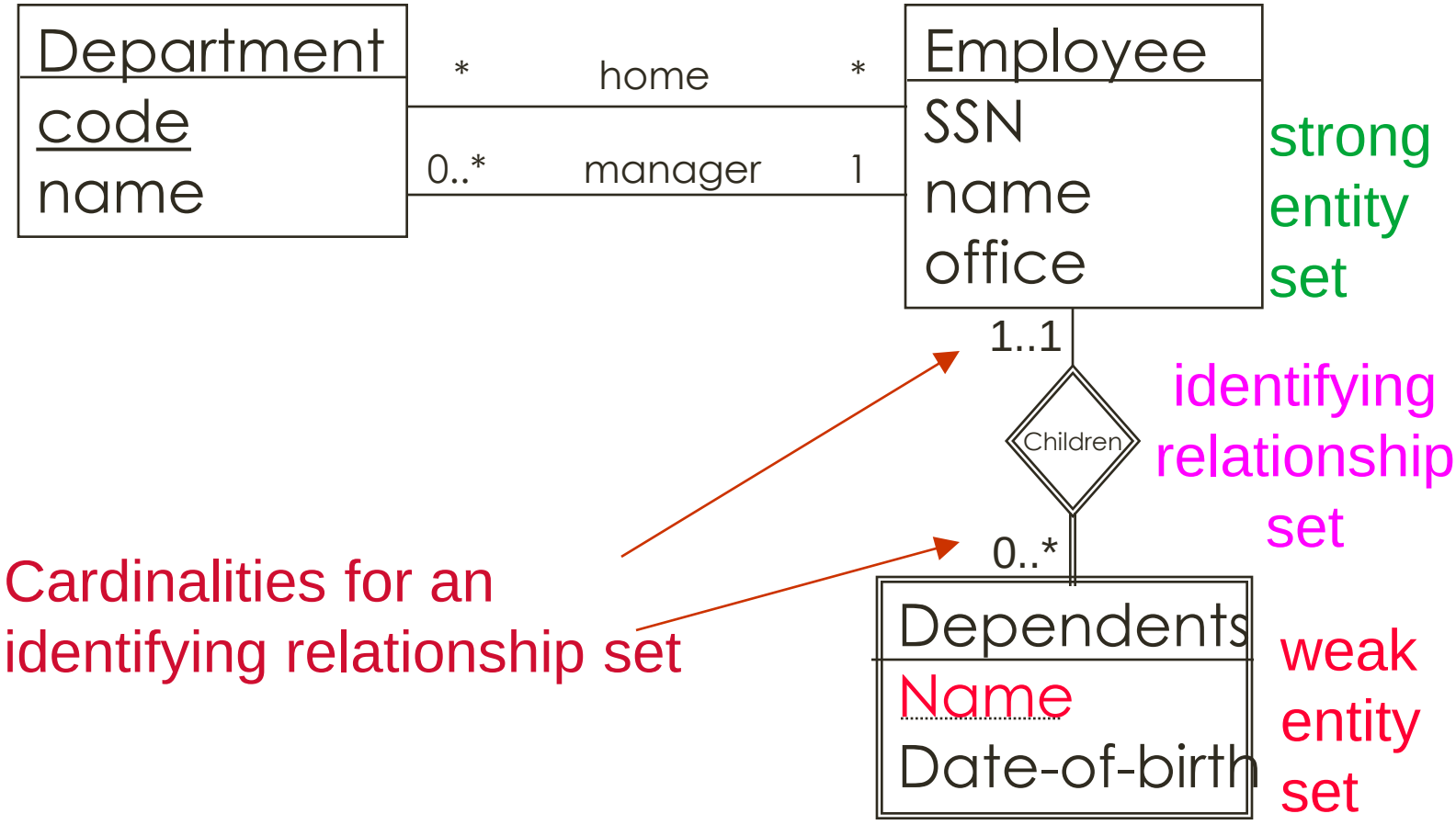


Weak Entity Sets (& Supporting Relationship sets)

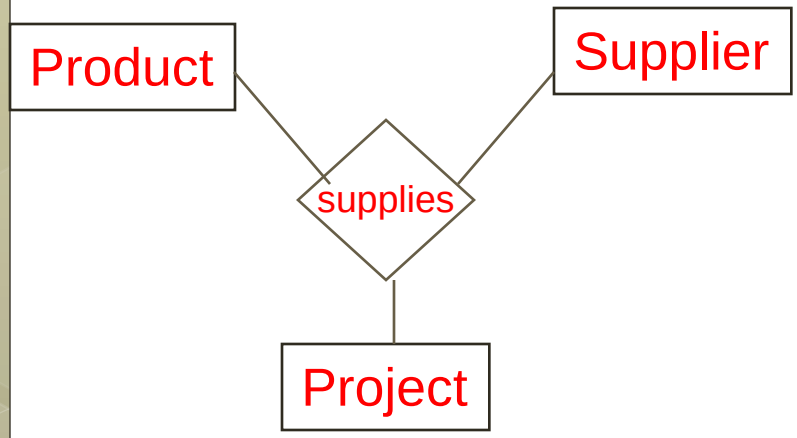
Notation and terminology in our book shown in red font

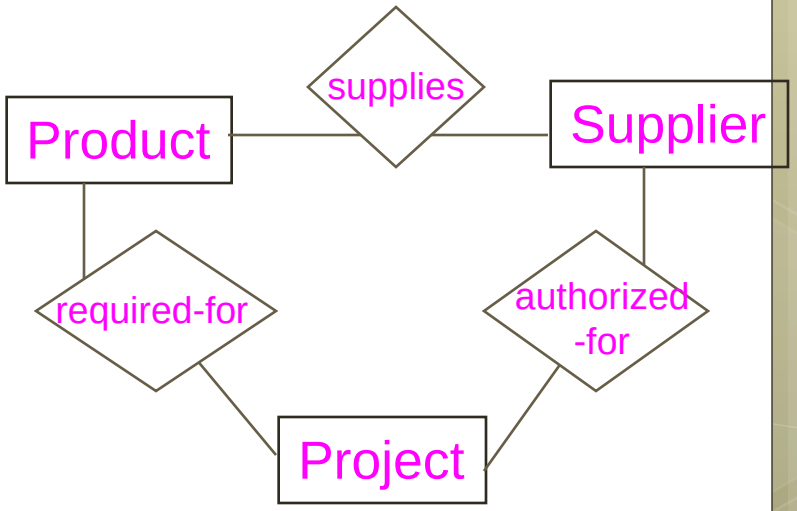


Weak Entity Sets and Identifying Relationship sets:
Alternative Notation (cont.)



Ternary vs. Binary Relationship sets





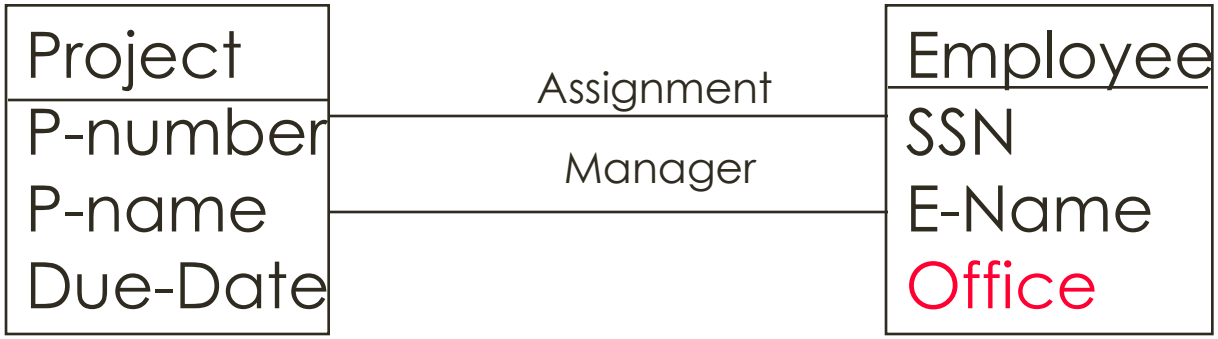
These two schemas are not equivalent!
When would we use a ternary relationship set?
When would we use three binary relationship sets?

Binary vs. Ternary Relationship sets (Cont.)

- The ternary relationship set means that a Supplier must be authorized to supply a particular part to a particular project. e.g., Office-Depot can supply laser printer paper to project 112. Office-Max can supply paper clips to Project 112. Office-Max can supply pencils to project 115. (But based on that much information, Office-Max can't supply pencils to 112.)
- The three binary relationship sets each represent something distinct. A Supplier can be authorized to supply certain products (Office-Max can supply pencils). A Project can require certain products (112 needs pencils). And a Supplier can be authorized to supply a certain project. (Office-Max supplies 112) Therefore: Office-Max can supply pencils to 112.

Duality: entity ⟺ value

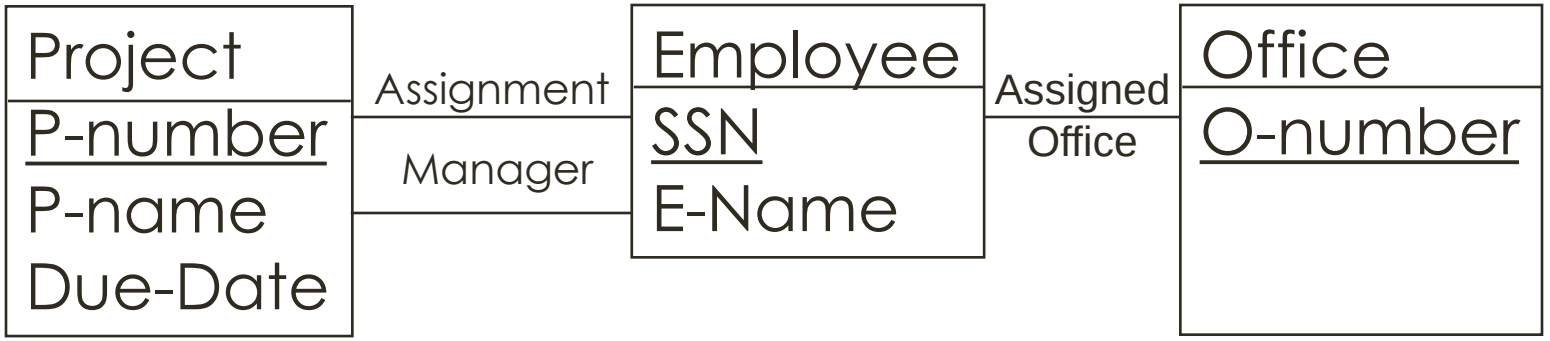
and attribute ⟺ relationship



Should Office be an attribute of Employee? or a separate entity set? Most attributes can be "promoted" to an entity set and some entities can be "demoted" to an attribute value.

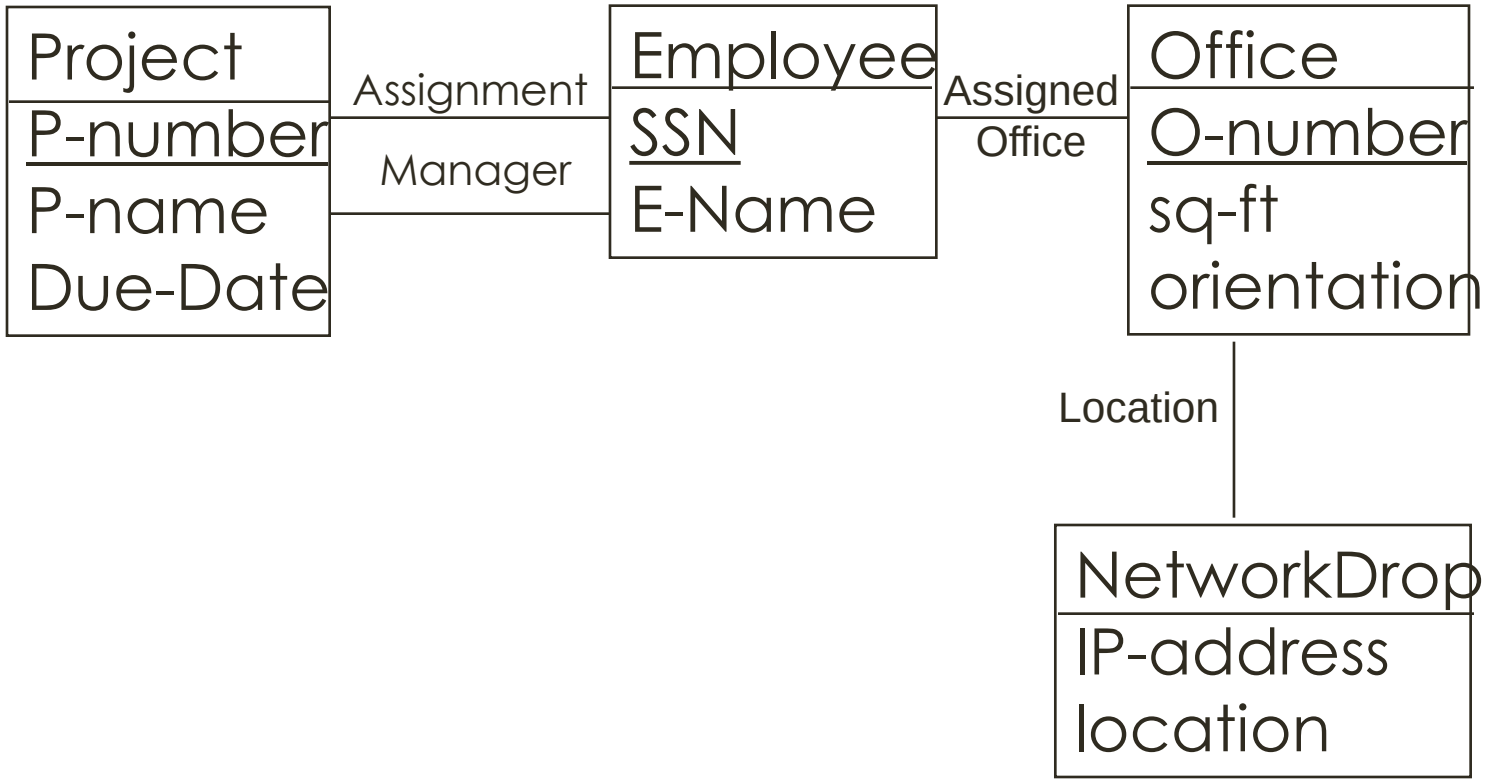
This explains why there are so many different ways to design a schema.

Why should Office be an entity?

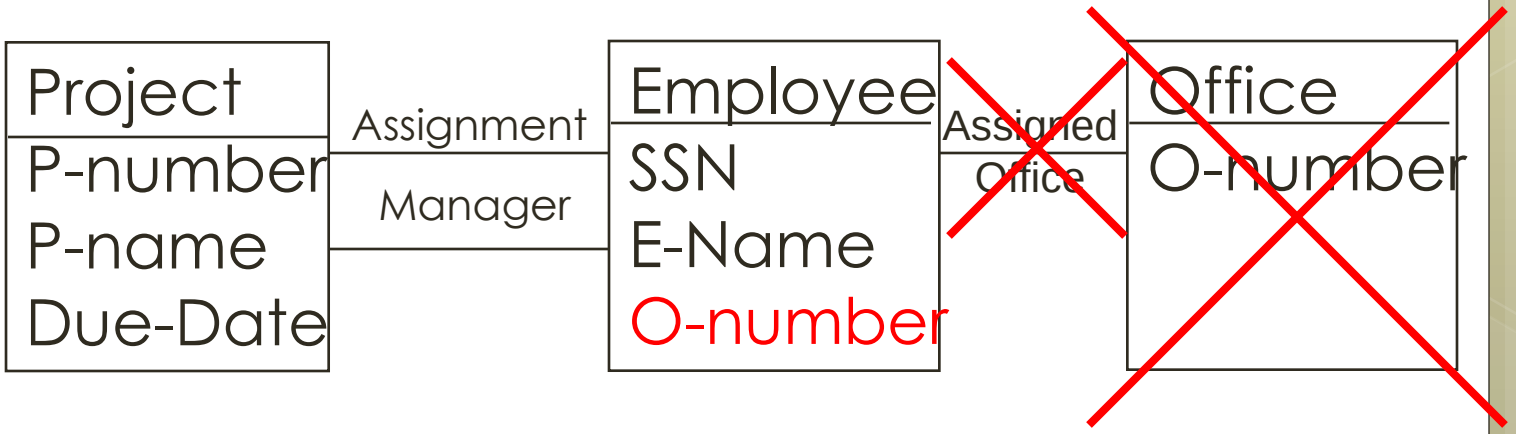


- If an employee can have more than one office
- If you want other attributes of Office
- If an office needs to participate in other relationships such as a relationship connecting to furniture or telephones or network drops (located in the office)

Office – as an entity (with attributes & relationships)



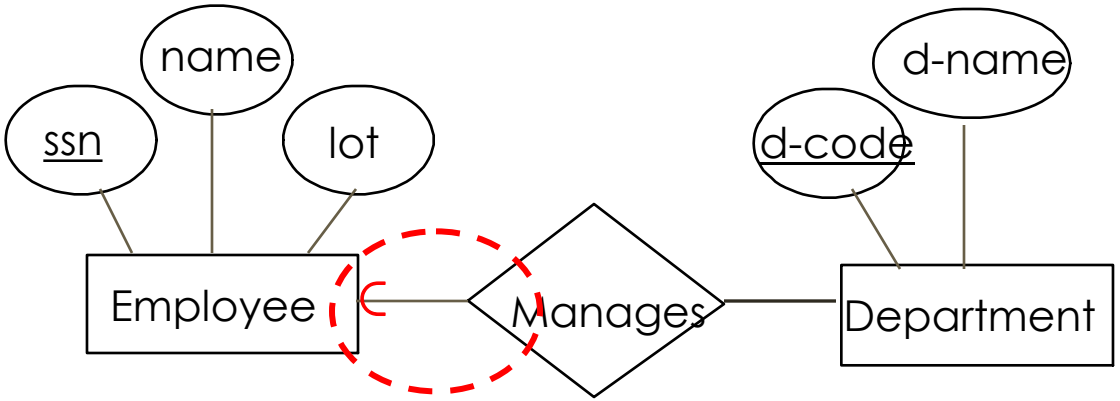
Why should Office be an attribute of Employee?



- It is faster to access office number; there is no join required.
- Schema is a little simpler; one less table

What the book calls referential integrity
Must have one participant – cupped arrow

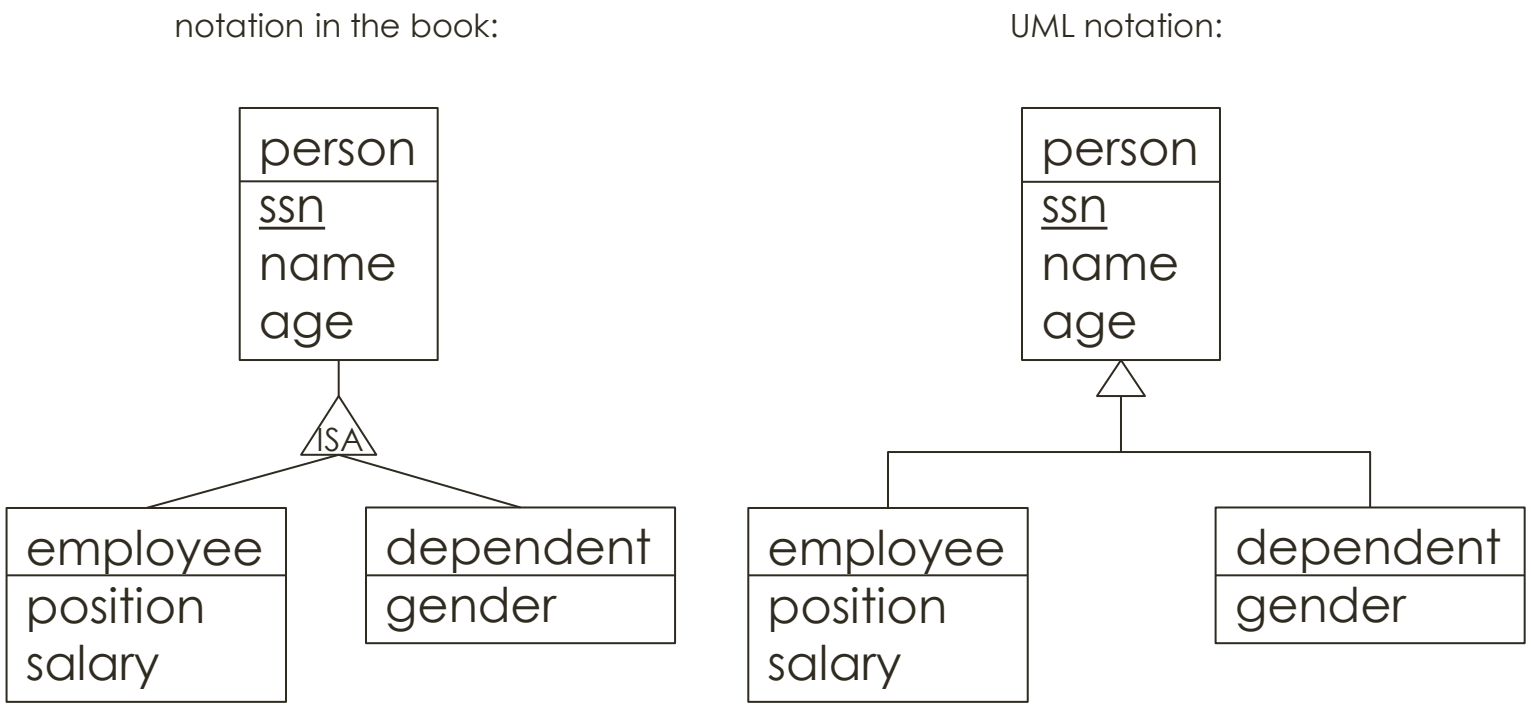
Cupped arrow circled here



is equivalent to card. of 1 shown here.

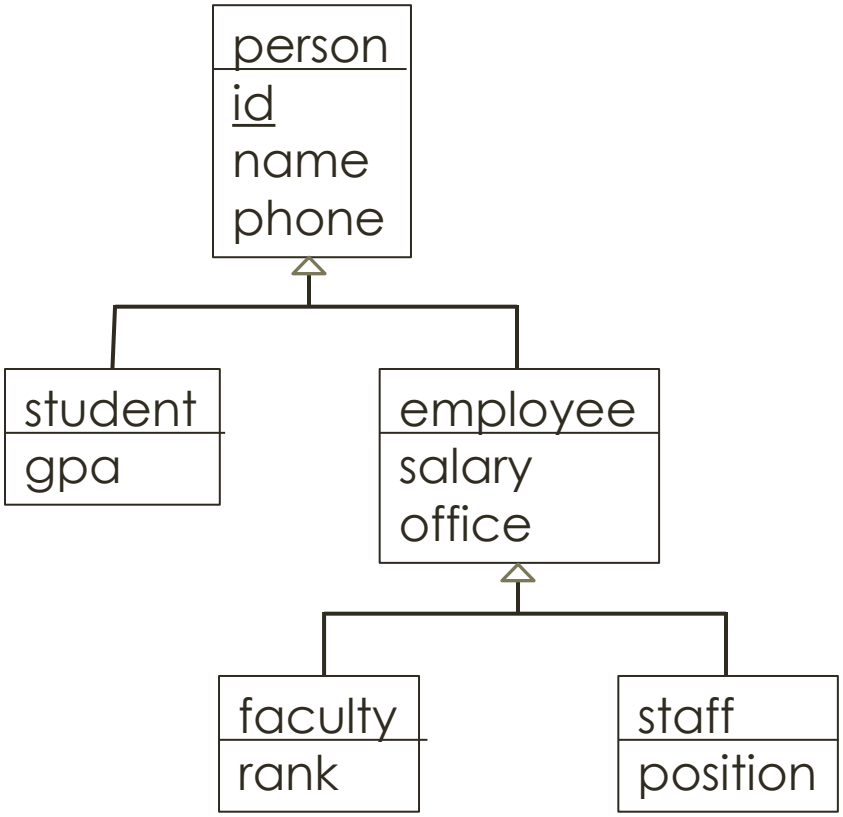
Each dept has at most one manager, according to the *key constraint* on Manages. d-code can be the key for Manages.

ISA relationship among entities



The person entity is the superclass; the employee and dependent entities are the subclasses. They inherit all of the attributes (and relationships) from the person.
Note: subclasses don't need a key; they'll use key of superclass.

ISA relationships among entity sets



Only the root of the ISA hierarchy has a key.
(All students and employees have id as their key, here.)

A real-world entity has multiple components – in the appropriate entity sets.

(This is different from OO PLs where an object has exactly one type.)

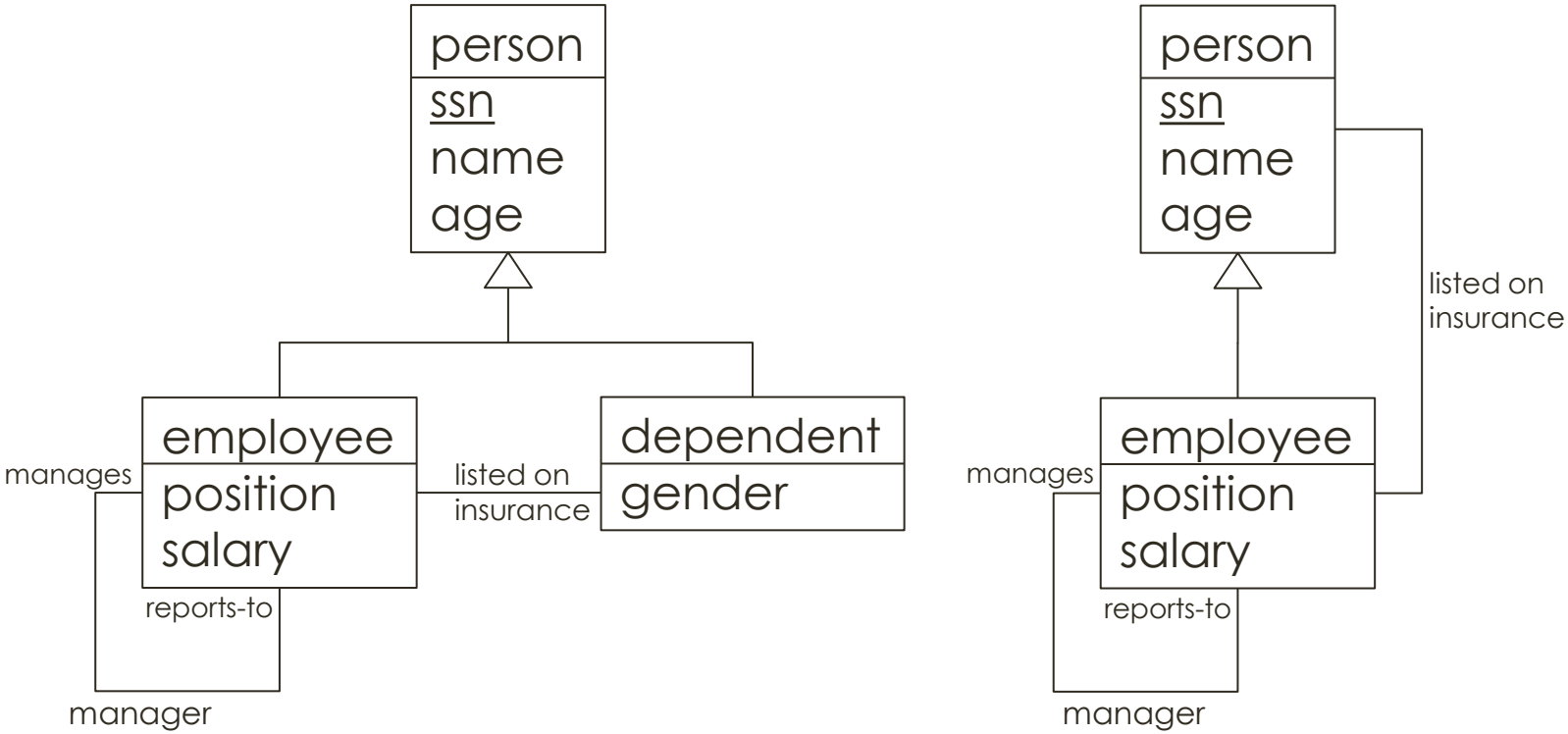
Exercise 2 (or do sports teams or …)

- A club has a name, office and phone; it is uniquely identified by its name.
- Clubs sponsor events. Each event has one main club as sponsor, and may have other clubs as co-sponsors.
- An event has a title, date, location and description; it is uniquely identified by the title and date.

Draw an Entity-Relationship Diagram (ERD).

- Extend your design to add departments; allow there to be two kinds of clubs: clubs with departmental affiliation where the department provides a lump sum of funding each year, clubs with no affiliation (and no funding). (Use ISA)

Several modeling options for dependent



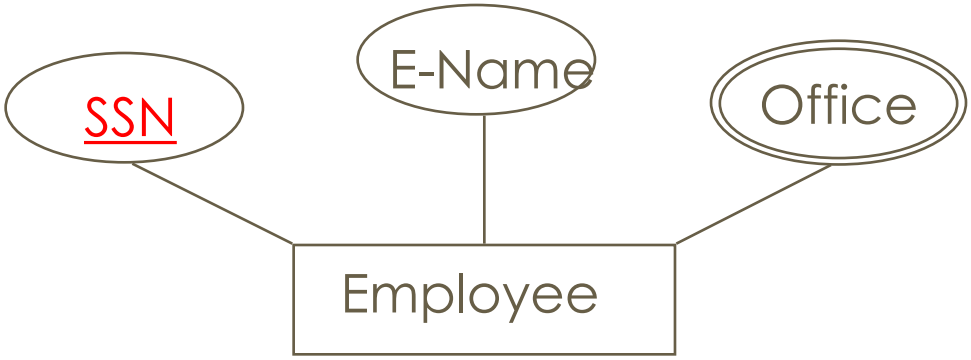
Should we model dependent separately from person? Or not?
Note: in this schema, dependent is NOT weak; it has ssn as a key.

Multi-valued Attribute

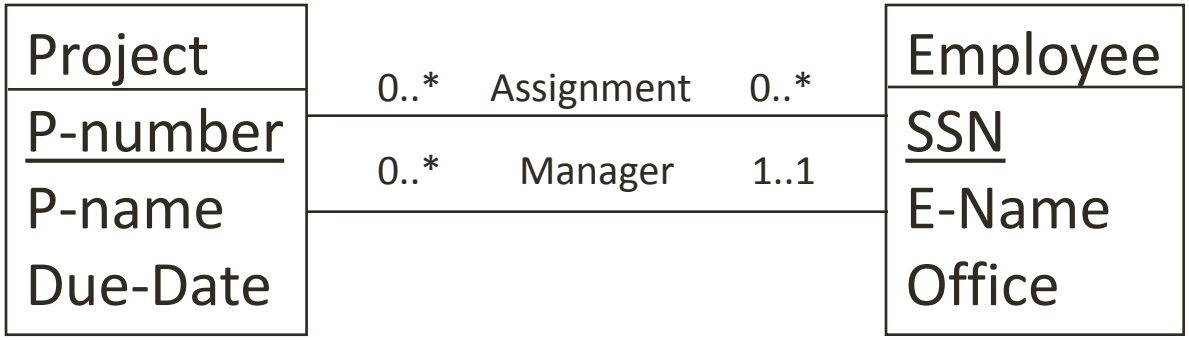
An attribute that can have multiple values for an entity.

Sometimes indicated by double circle, as shown.

This is an extension to the original ER model.

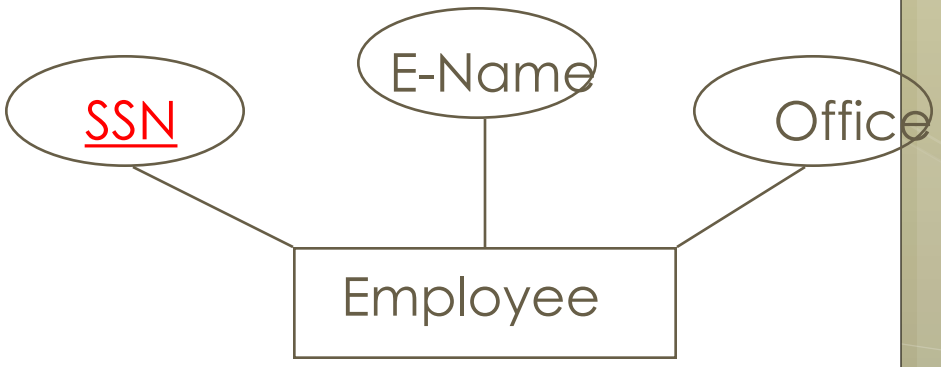


Converting ER to Relational Schema



1. Translate each entity into a table, with keys.

- Entity :
  - can be represented as a table in the relational model
  - has a key … which becomes a key for the table




```
CREATE TABLE Employee
    (SSN CHAR(11) NOT NULL,
    E-Name CHAR(20),
    Office  INTEGER,
    PRIMARY KEY  (SSN))
```

2. Create a table for the multi-valued attribute.

Note: a relational DBMS may or may not allow multi-valued attributes.

How many offices can one employee have?

Just one

Project(P-number, P-name, Due-Date)
Employee(SSN, E-Name, Office)

vs.

More than one; define extra table.

Project(P-number, P-name, Due-Date)
Employee(SSN, E-Name)
Office-Assignment(SSN, Office)

Sample Data

Just one

Project(P-number, P-name, Due-Date)

Employee(SSN, E-Name, **Office**)

12 Smith O-105

15 Wei O-110

More than one; define extra table.

Project(P-number, P-name, Due-Date)

Employee(SSN, E-Name)

12 Smith

15 Wei

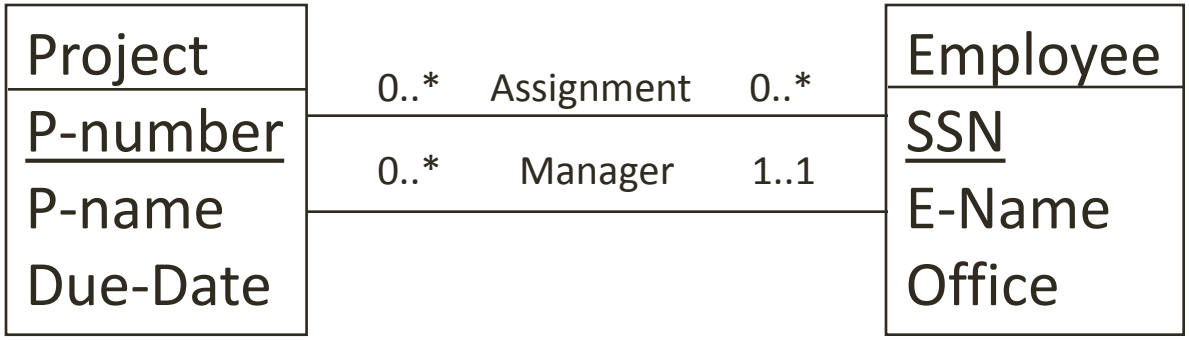
Office-Assignment(SSN, Office)

12 O-105

12 O-106

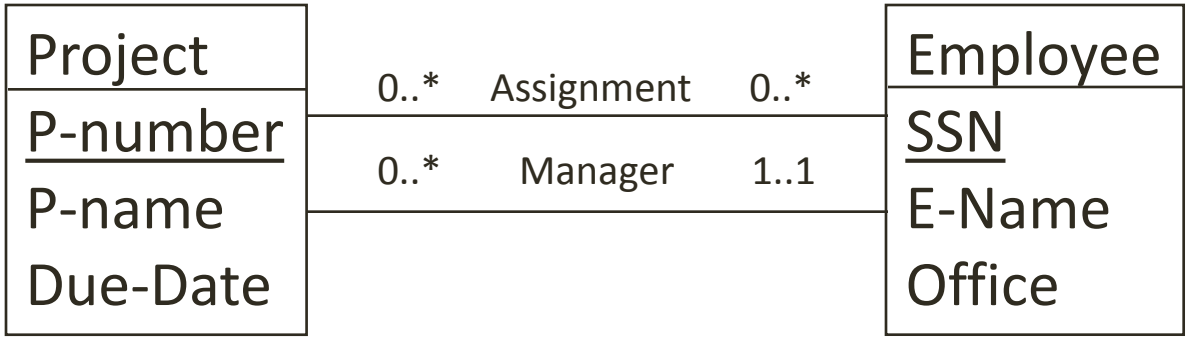
15 O-110

3. Translate each **many-to-many** relationship set into a table



What are the attributes and
what is the key for Assignment?

Project(P-number, P-name, Due-Date)
Employee(SSN, E-Name, Office)

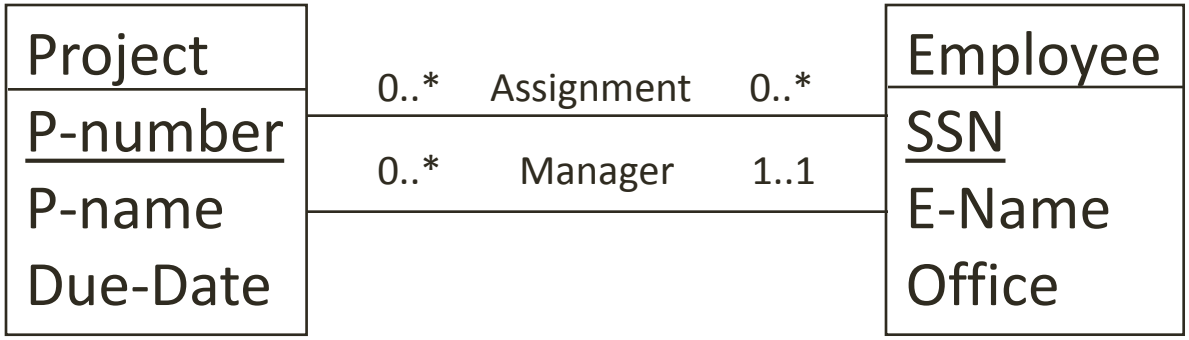


Answer: Assignment(P-Number, SSN)

P-Number is a foreign key for Project
SSN is a foreign key for Employee

Project(P-Number, P-Due-Date)
Employee(SSN, E-Name, Office)

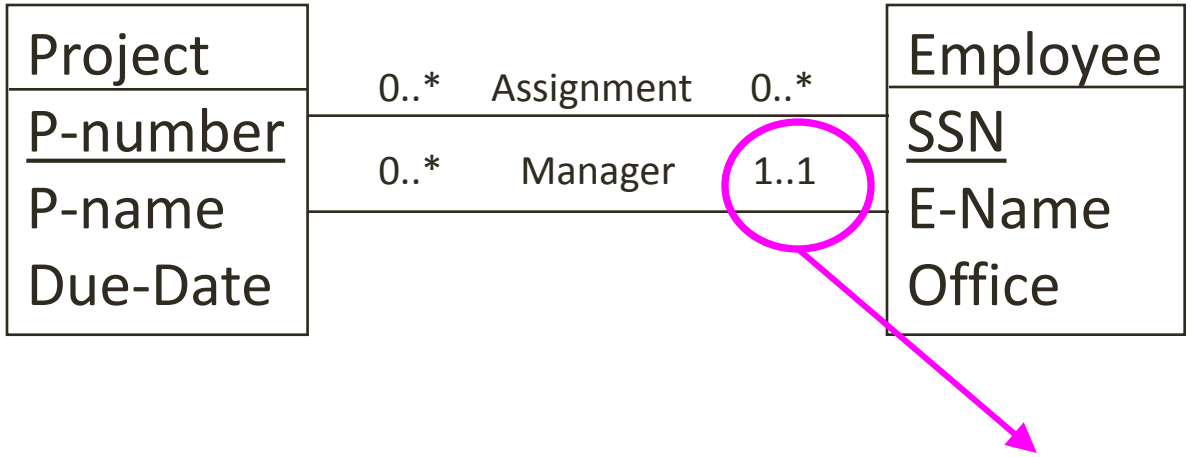
What do we do with a one-to-many relationship?



For example, what do we do with Manager?

Project(P-number, P-name, Due-Date)
Employee(SSN, E-Name, Office)

4. Create a foreign key for a 1-to-many relationship.

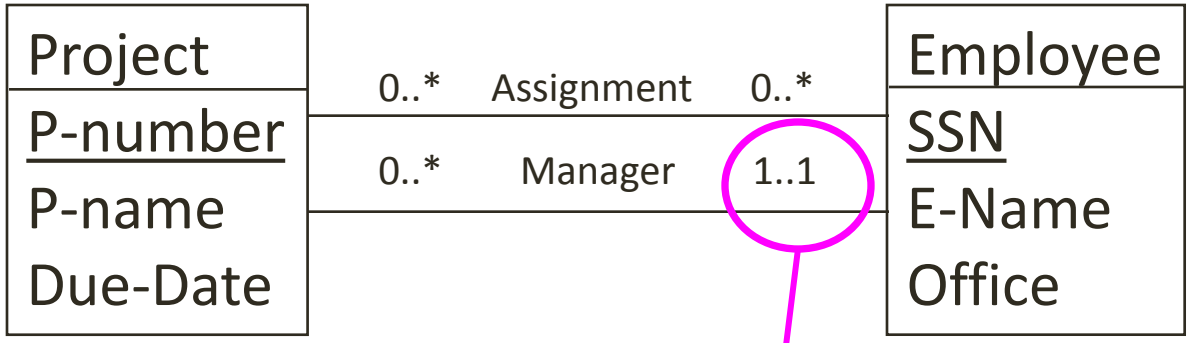


Project(P-number, P-name, Due-Date, Manager)
Employee(SSN, E-Name, Office)

Manager is a foreign key (referencing the Employee relation)

value of Manager must match an SSN in Employee

4. Or...Create a table for a 1-many relationship.



Project(<u>P-number</u>, P-name, Due-Date, Manager)
Employee(<u>SSN</u>, E-Name, Office)

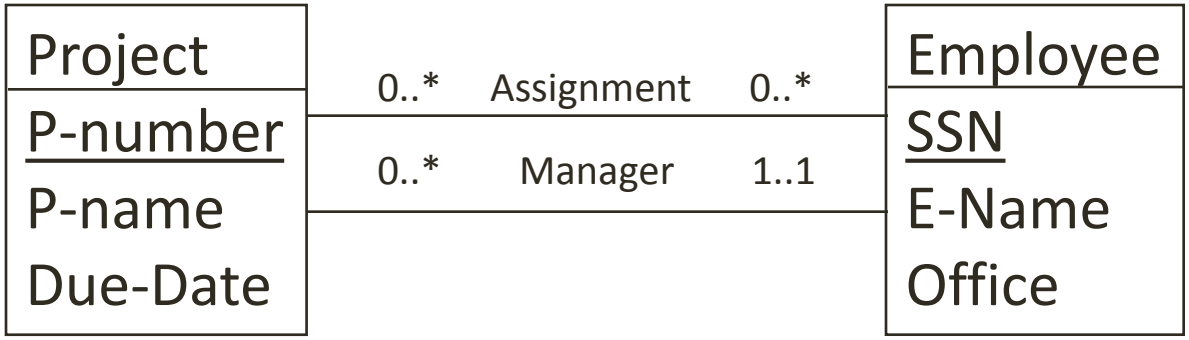
vs.

Project(<u>P-number</u>, P-name, Due-Date)
Employee(<u>SSN</u>, E-Name, Office)
Manager(<u>P-number</u>, SSN)

What are the tradeoffs between these two?

Note:
P-number
is the key
for Manager

What if SSN is the key for Manager?



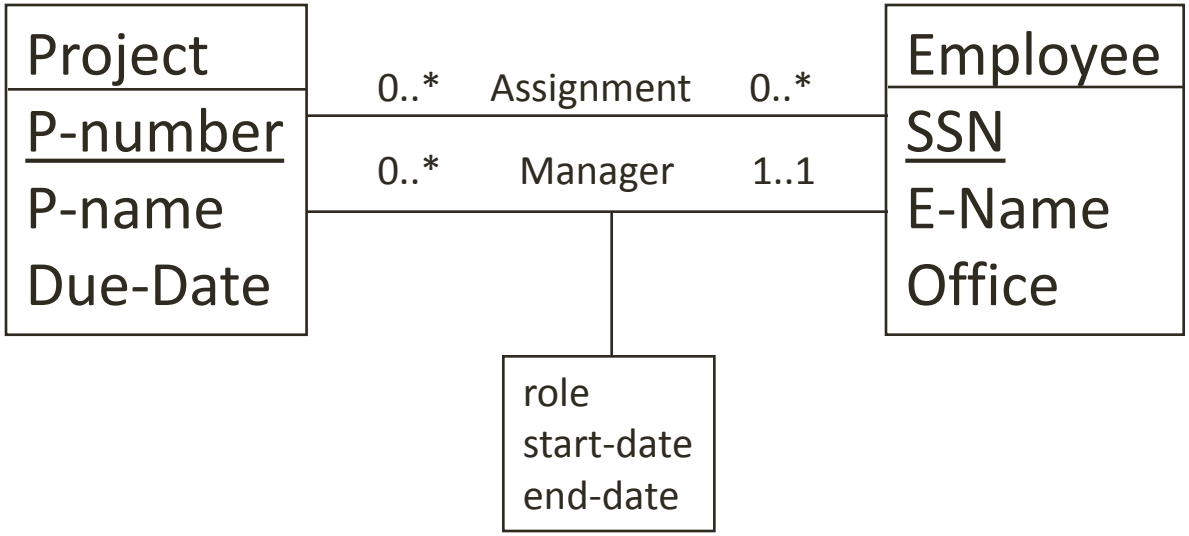
Project (P-number, P-name, Due-Date)
Employee (SSN, E-Name, Office, Managed-project)

vs.

Project (P-number, P-name, Due-Date)
Employee (SSN, E-Name, Office)
Manager (P-number, SSN)

We end up with the wrong semantics; a person can only manage one project.

What if a many-to-many relationship has attributes?

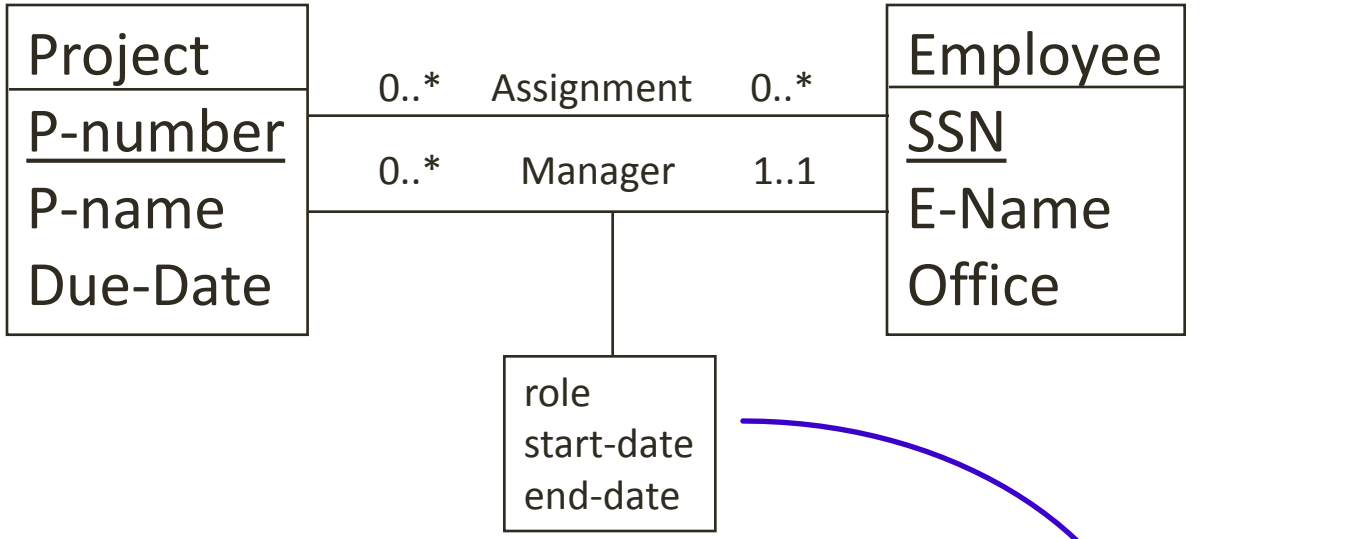


- Assignment(P-number, SSN)

Project(P-number, P-name, Due-Date)

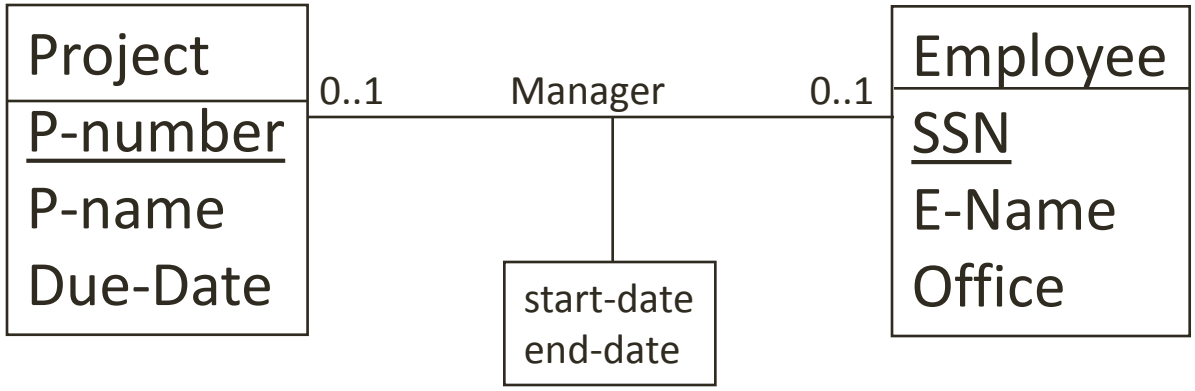
Employee(SSN, E-Name, Office)

Add attributes to the table for the relationship



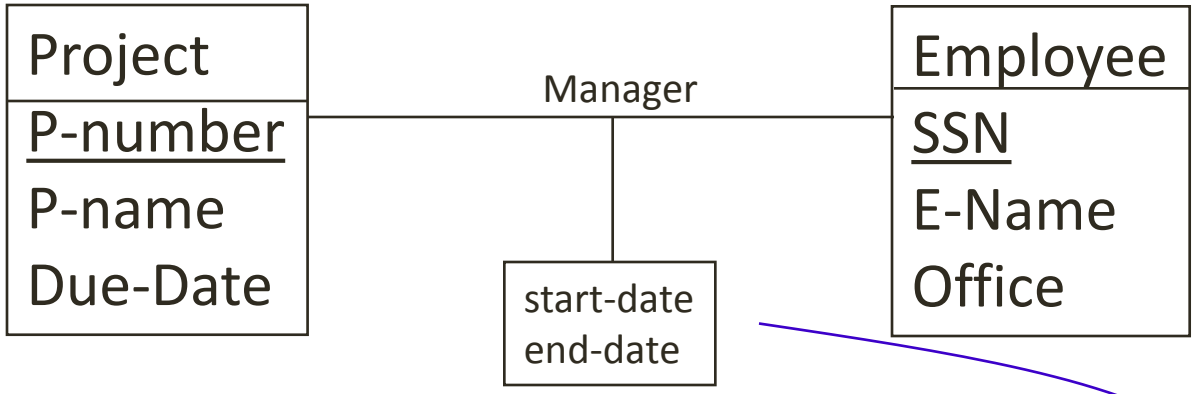
Assignment(P-number, SSN, role, start-date, end-date)
Project(P-number, P-name, Due-Date)
Employee(SSN, E-Name, Office)

What if a 1-to-many relationship has an attribute?



Project(P-number, P-name, Due-Date, Manager)
Employee(SSN, E-Name, Office)

Add attributes to the table for the relationship

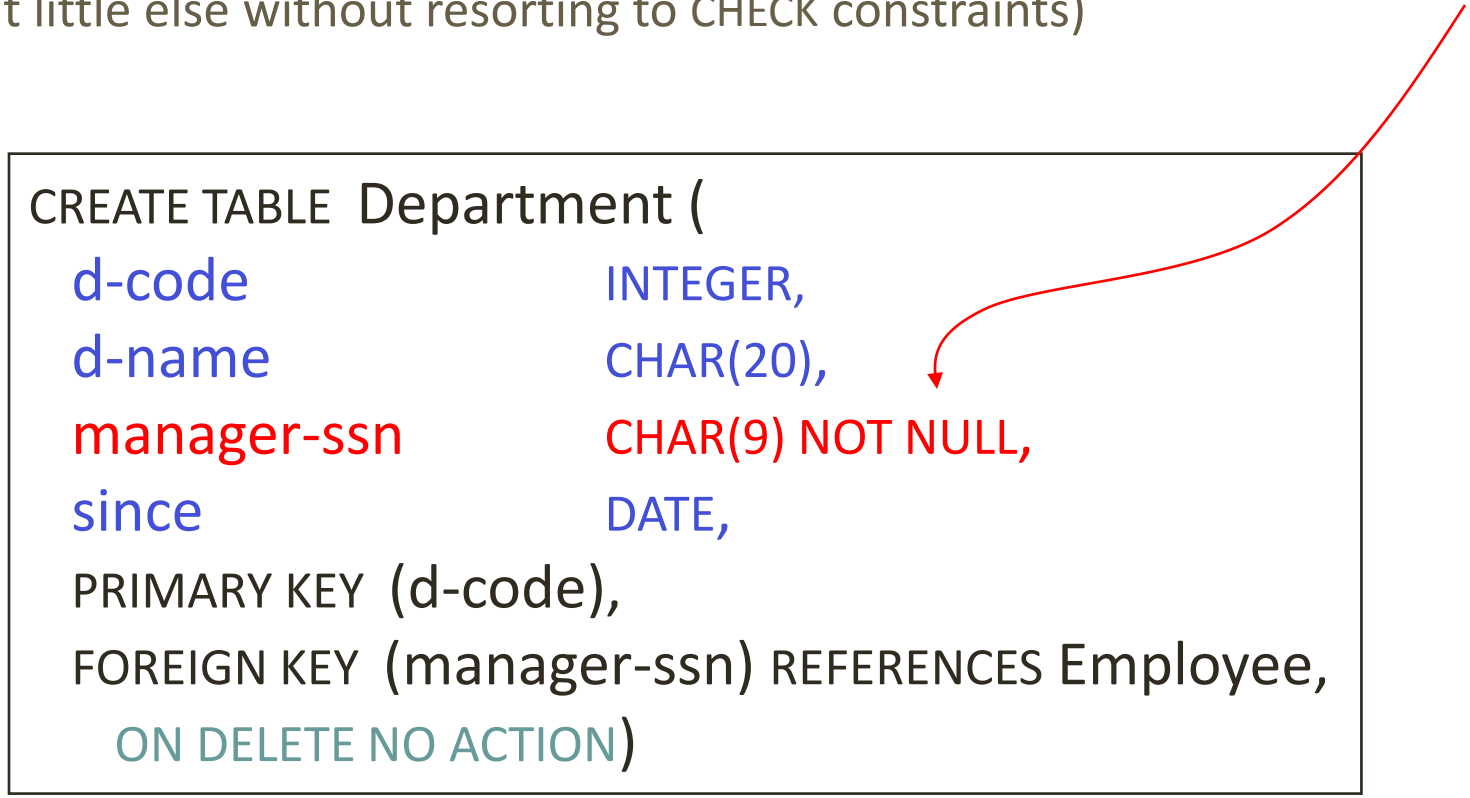


Project(P-number, P-name, Due-Date, Manager, start-date, end-date)

Employee(SSN, E-Name, Office)

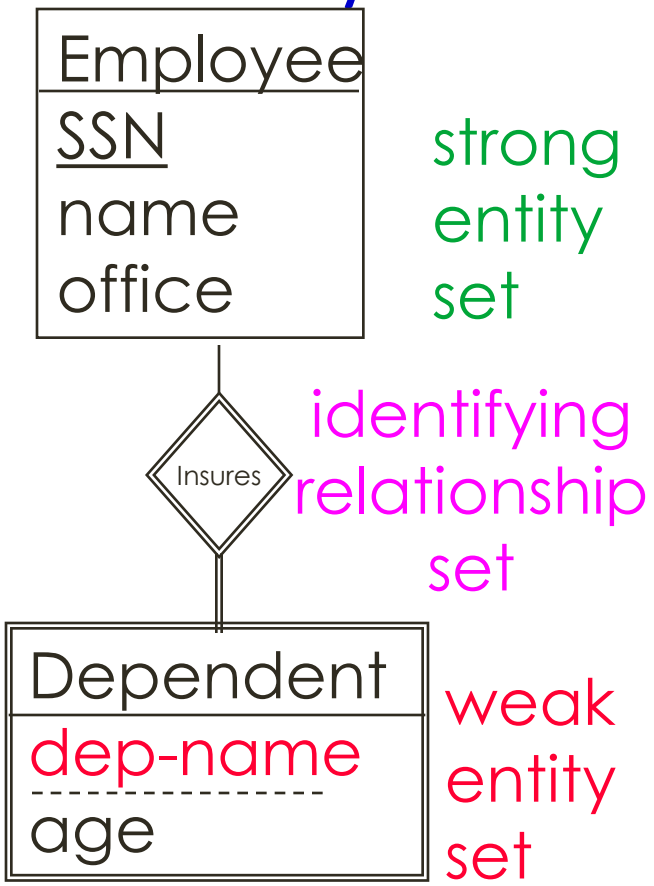
Referential integrity constraints in SQL

We can require any table to be in a binary relationship using a foreign key which is required to be NOT NULL
(but little else without resorting to CHECK constraints)

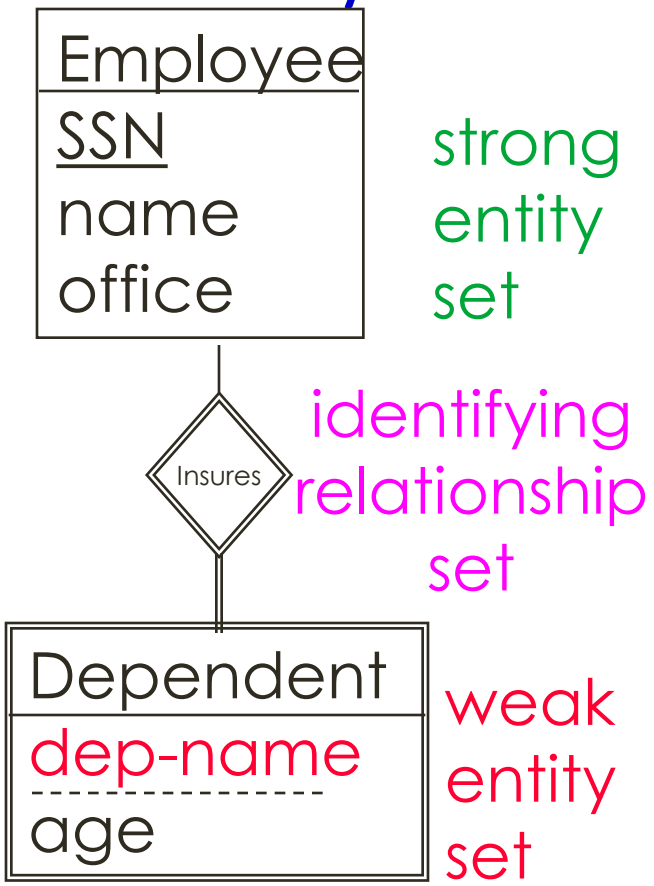


```
CREATE TABLE Department (
  d-code            INTEGER,
  d-name            CHAR(20),
  manager-ssn       CHAR(9) NOT NULL,
  since             DATE,
  PRIMARY KEY (d-code),
  FOREIGN KEY (manager-ssn) REFERENCES Employee,
    ON DELETE NO ACTION)
```

Weak Entity Sets



Weak Entity Sets



Employee(ssn, name, office)

foreign key

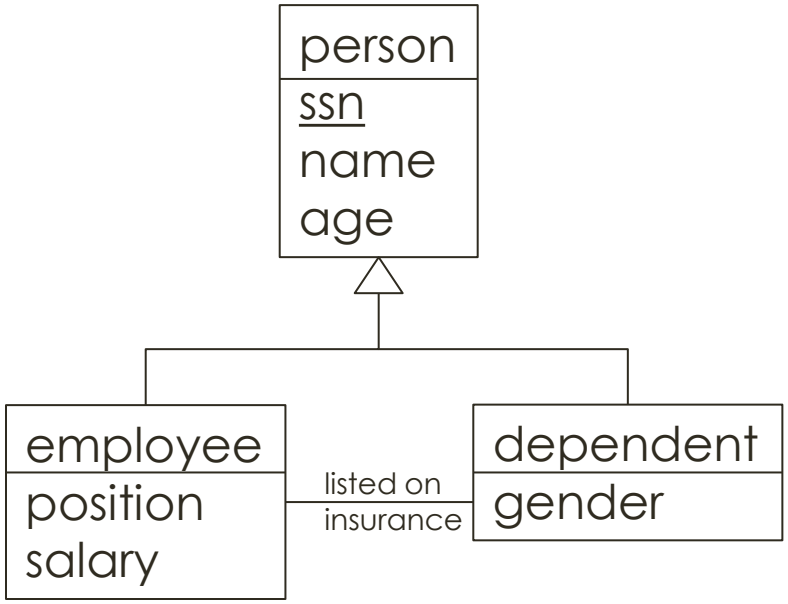
Dependent(ssn, dep-name, age)

Translating Weak Entity Sets

- Weak entity sets and supporting/identifying relationship sets are translated into a single table. Must include key of (all) supporting/strong entity set and they must be part of the key for the weak entity. They are also foreign keys.
- When the owner entity is deleted, all owned weak entities should also be deleted.

```
CREATE TABLE Dependent (
  dep-name        CHAR(20),
  age             INTEGER,
  ssn             CHAR(11) NOT NULL,
  PRIMARY KEY (dep-name, ssn),
  FOREIGN KEY (ssn) REFERENCES Employee, ON DELETE CASCADE)
```

Translating entities involved in ISA

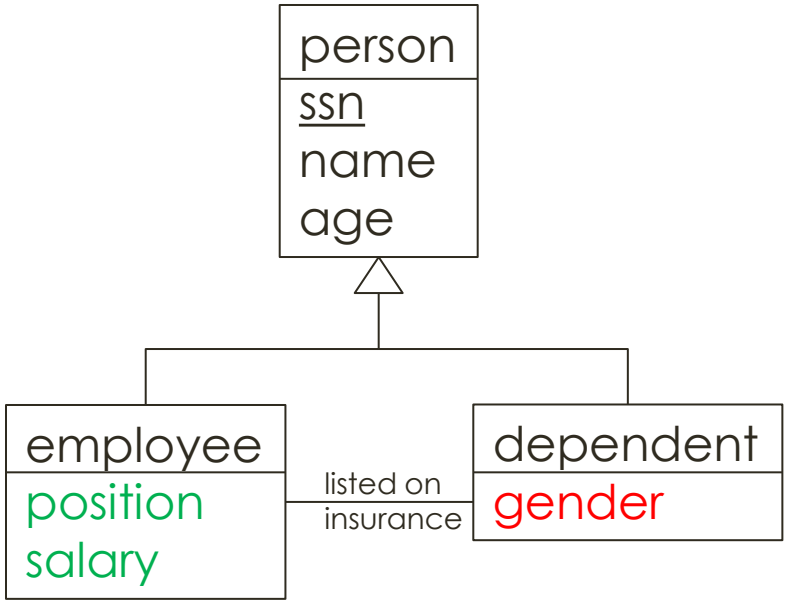


You have three choices:

1. One table: person with all attributes
2. Two tables: employee and dependent with all attributes from person copied down
3. Three tables: one for person, employee, and dependent. Add ssn to employee and dependent

Note: in this schema, dependent is NOT weak. It has ssn as a key because of the ISA relationship.

Option 1: (E/R approach) one table for each entity types (person, employee, ependent)



Advantages:

1. there can be persons who are not employees or dependents
2. no redundancy
3. no need to use nulls
4. matches the conceptual model

Disadvantages:

1. joins are required to get all of the attributes

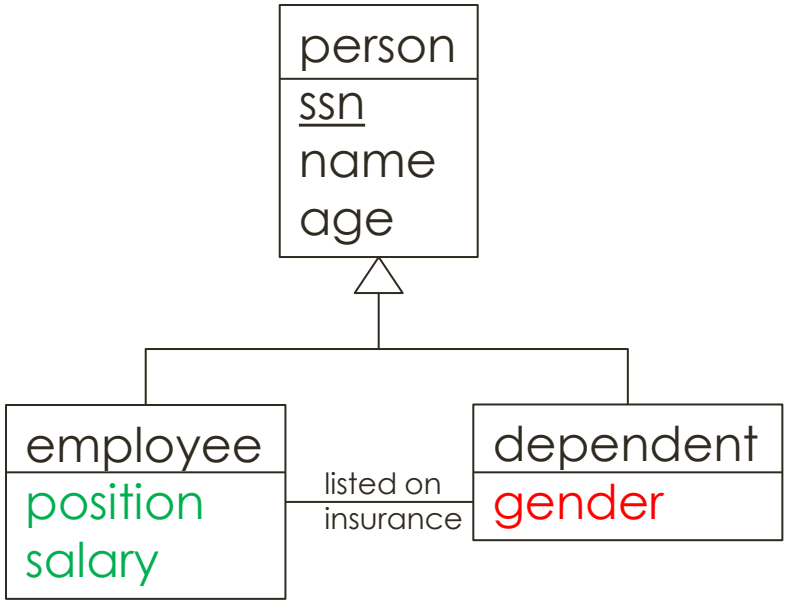
person(ssn, name, age)

employee(ssn, position, salary) where ssn refs person.ssn

dependent(ssn, gender, insurer) where ssn refs person.ssn
where insurer references employee.ssn

Note: this works (for the "listed on insurance" relationship) if each dependent is on just one employee's insurance.

Option 2: (OO approach) one table for each subtree rooted at the top



Advantages:

1. each person is in just one table
2. no need to use nulls
3. minimum amount of space

Disadvantages:

1. unions are required to get persons (for example)
2. lots of tables

person(ssn, name, age)

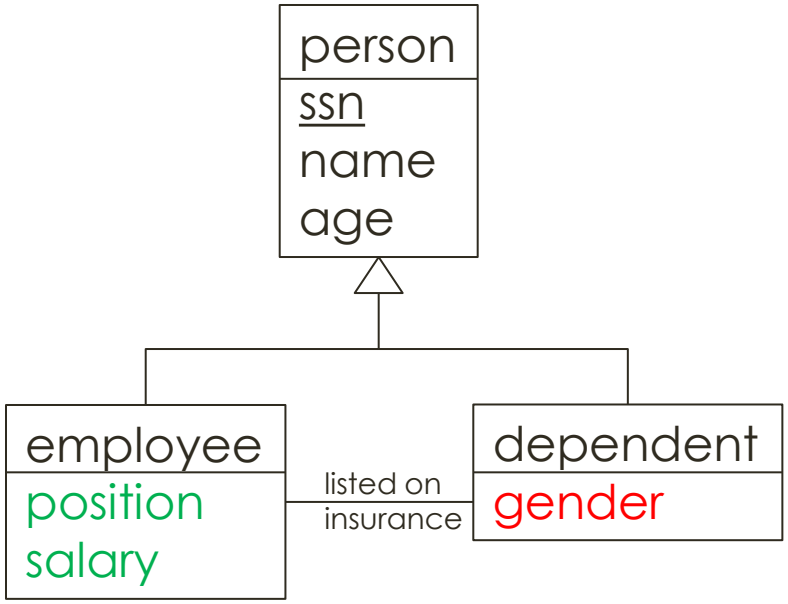
pemployee(ssn, name, age, position, salary) where ssn refs person.ssn

pdependent(ssn, name, age, gender, insurer)

Now, the foreign key for "insurer" may point to several tables but you can't do that with a foreign key. You need a constraint or trigger.

pemployeedep(ssn, name, age, position, salary, gender, insurer)

Option 3 (nulls): Use one table (for person) with all attributes from all subclasses

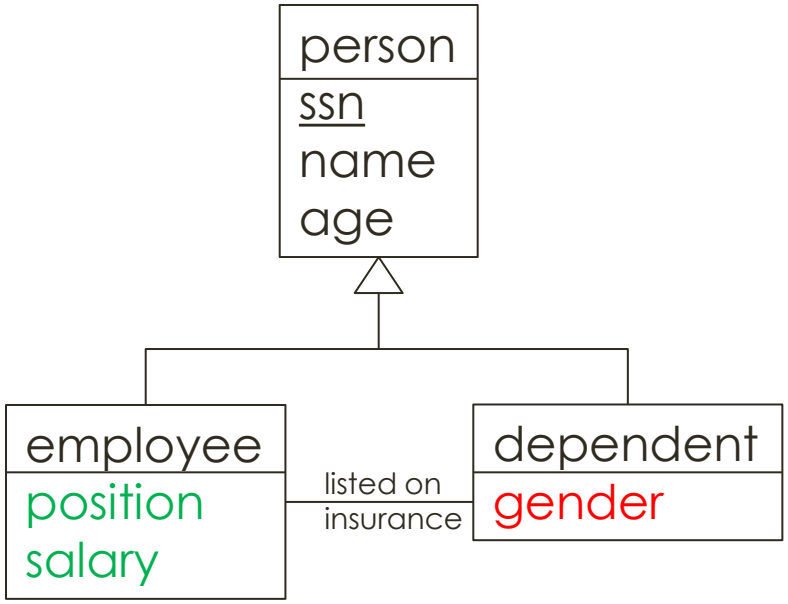


Advantages:
1. no join needed
2. no redundant attributes

Disadvantages:
1. may have lots of nulls
2. slightly more difficult to find just employee or just dependent

person(ssn, name, age, position, salary, gender)
listed-on-insurance(emp-ssn, dep-ssn) where
emp-ssn references person.ssn and
dep-ssn references dep-ssn

Simplify: Use employee & dependent if no one will ever be just a person. (Use only tables you need.)



Advantages:
1. fewer nulls
2. easy to get just employee or just dependent

Disadvantages:
1. Not possible to have a person who is not an employee or person
2. name and age are stored redundantly

employee(ssn, name, age, position, salary)

dependent(ssn, name, age, gender, insurer)

where insurer references employee.ssn

This works if each dependent is on just one insurance.

Translation Steps: ER to Tables

- Create table and choose key for each entity; include single-valued attributes.
- Create table for each weak entity; include single-valued attributes. Include key of supporting entity(ies) as a foreign key in the weak entity. Set key as foreign key of owner plus local, partial key. (No need to translate supporting relationships.)
- For each 1:1 relationship, 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 1:N relationship, add a foreign key to the entity set on the N-side of the relationship (to reference the entity set on the 1-side of the relationship).

Translation Steps: ER to Tables (cont.)

- For each M:N 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 will serve as both the key for this table as well as a foreign key to the original table for the entity.
- This algorithm from Elmasri & Navathe, *Fundamentals of Database Systems, 6th Edition.* pp. 285-296. They have slightly different options for translating ISA relationships.

Exercise 3: Translate one or more of your ERDs to relational tables