



Conceptual and Logical Database Design

Abdu Alawini

University of Illinois at Urbana-Champaign

CS411: Database Systems

Overview of Database Design

- **Conceptual design:** (ER & UML Models are used for this.)
 - What are the **entities** and **relationships** we need?
- **Logical design:**
 - Transform ER design to Relational Schema
- **Schema Refinement:** (Normalization) **<-next lecture**
 - 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.



Conceptual Database Design

From user/application requirements to Entity-Relationship Diagrams OR
Unified Modeling Language design

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

- Relational 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 uniqueness or cardinality constraints

Entity Relationship Model Unified Modeling Language

ER Model

- Proposed by Peter Chen in 1976
- Gives us a language to specify
 - What information the database must hold
 - How the bits of information relate to one another

UML Model

- UML is a standard language for designing software systems
 - also used for DB design
- created by the Object Management Group (OMG)
- UML 1.0 specification draft was proposed to the OMG in early 1997.

Entity-Relationship Diagram (original syntax)

Legend:



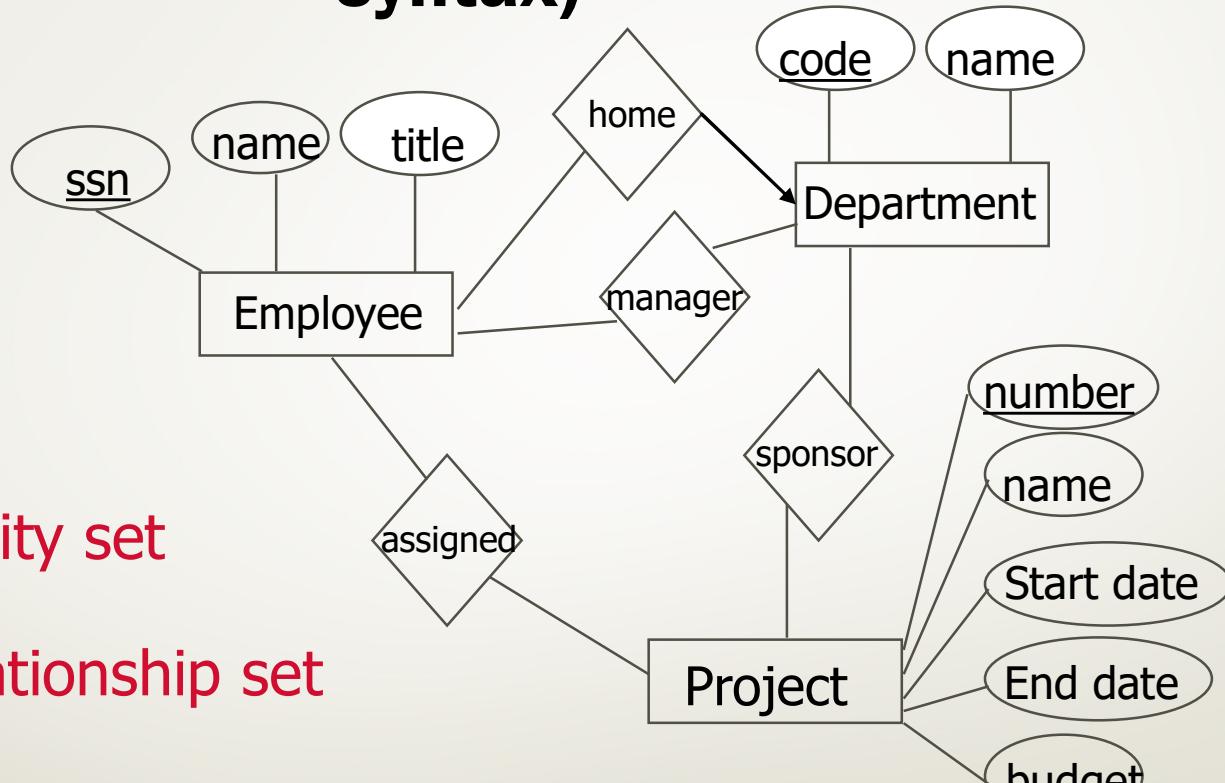
Entity set



Relationship set



Attribute name



Definitions

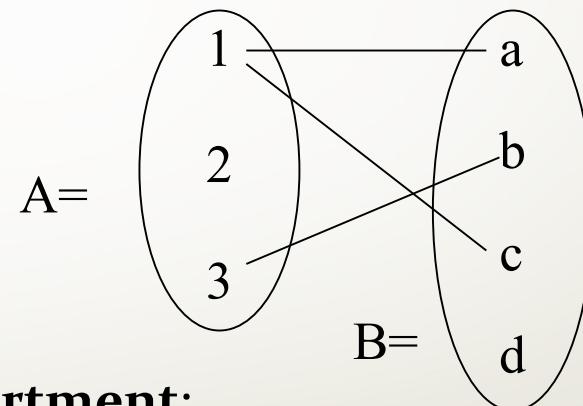
- Entity: Real-world object distinguishable from other objects.
An entity is described using a set of *attributes*.
- Entity Set: A collection of similar entities. E.g., all employees.
(often referred to as just entity, which blurs the distinction between type and collection)
- Relationship: Association among 2 or more entities. E.g., Kristin's **home department** is Research & Development.
- Relationship Set: Collection of similar relationships. E.g., Home
(often referred to as just relationship)

Relationships

- Formal definition:
 - if A, B are sets, then a relation R is a subset of $A \times B$
- $A = \{1, 2, 3\}$, $B = \{a, b, c, d\}$,
 $R = \{(1, a), (1, c), (3, b)\}$

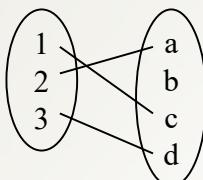
Same story w/ entity sets

sponsor is a subset of **Project x Department**:

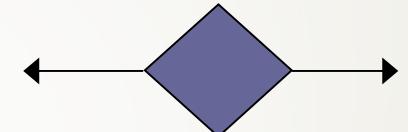


Multiplicity of ER Relationships

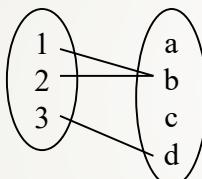
- one-one:



One on LHS/RHS
connected to at most
one on RHS/LHS



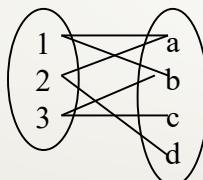
- many-one:



One on LHS
connected to at most
one on RHS



- many-many:

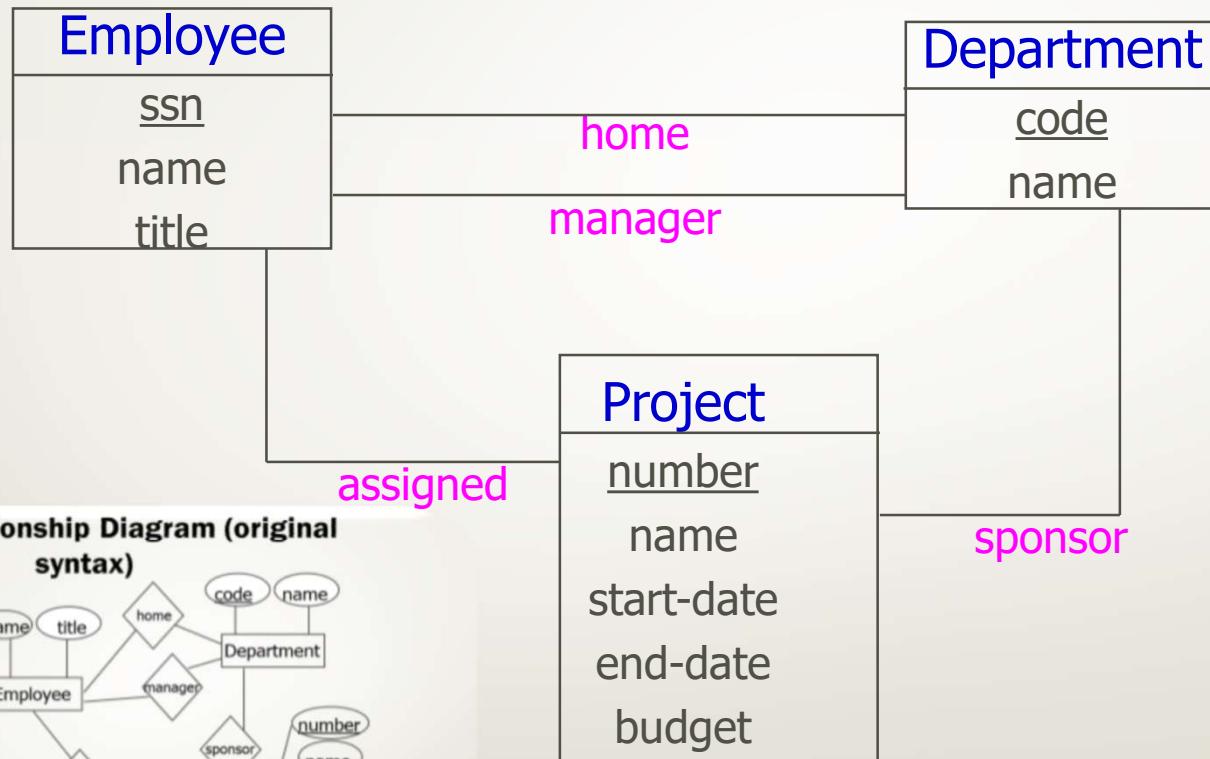


No constraints



- Multiplicity can be shown with arrows
- Arrow = at most 1

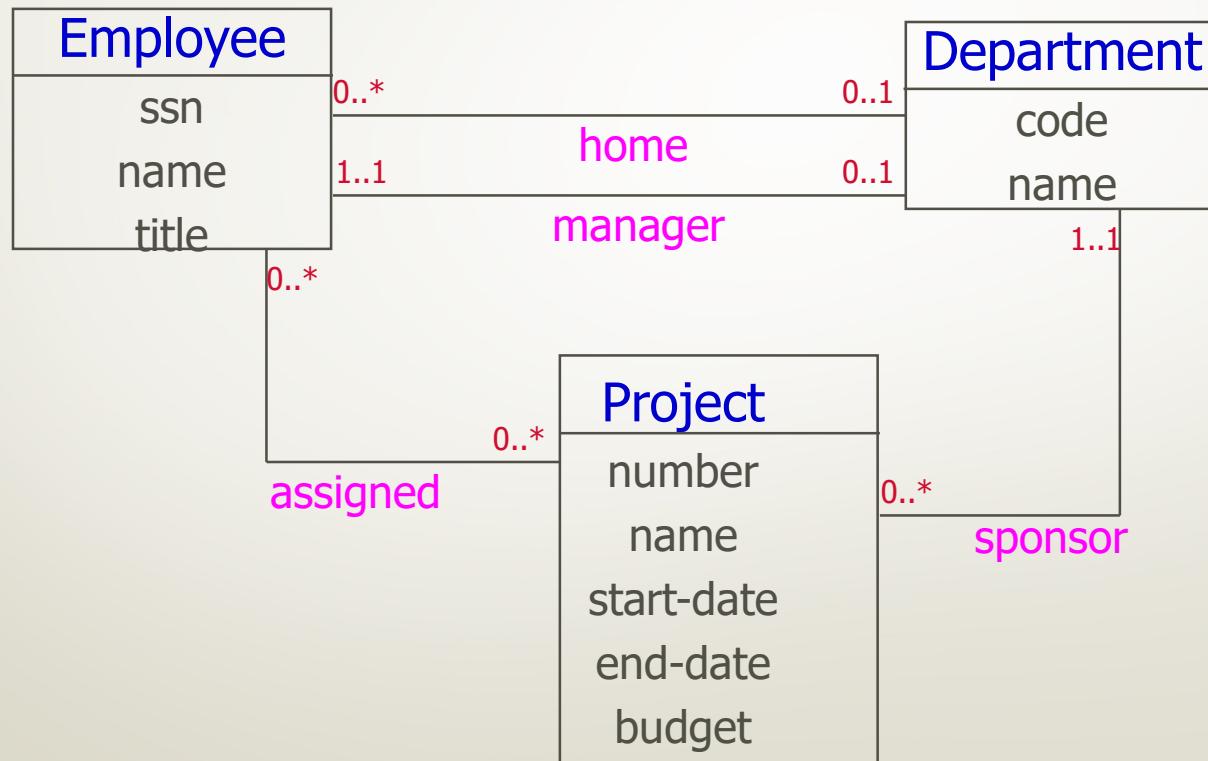
UML version of the same E-R Diagram



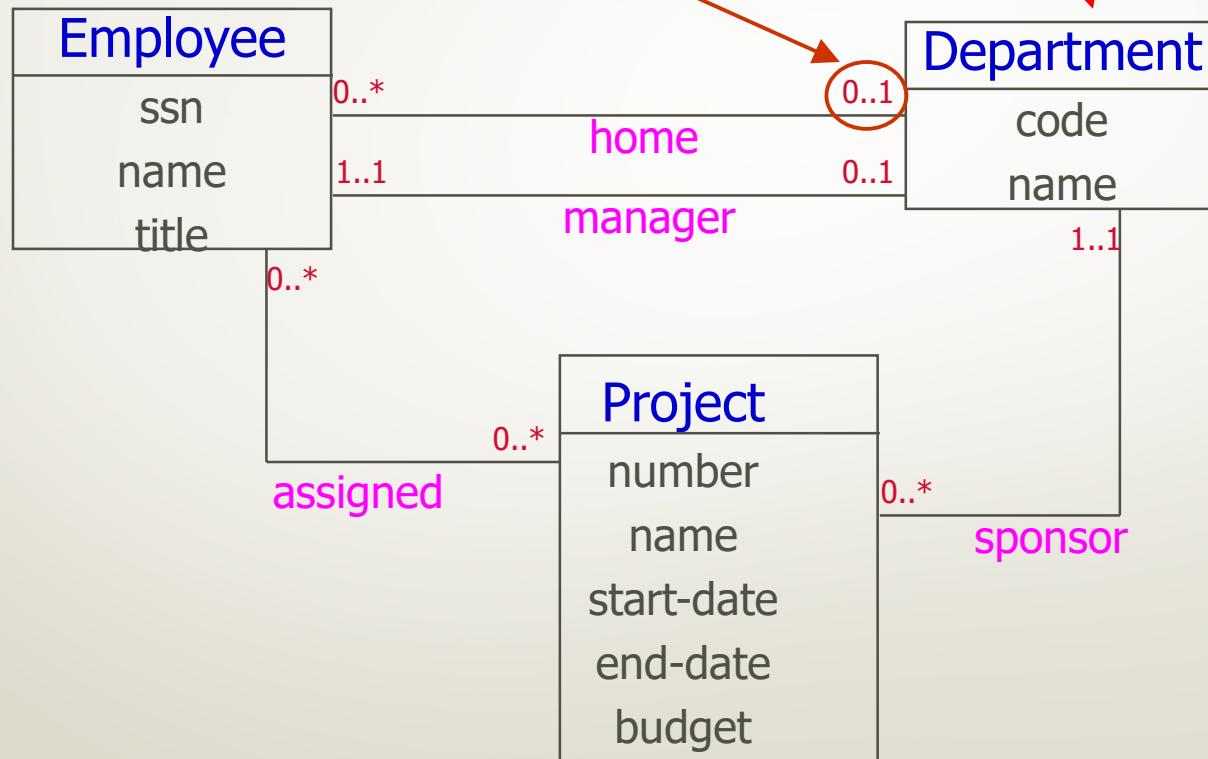
Entity-Relationship Diagram (original syntax)

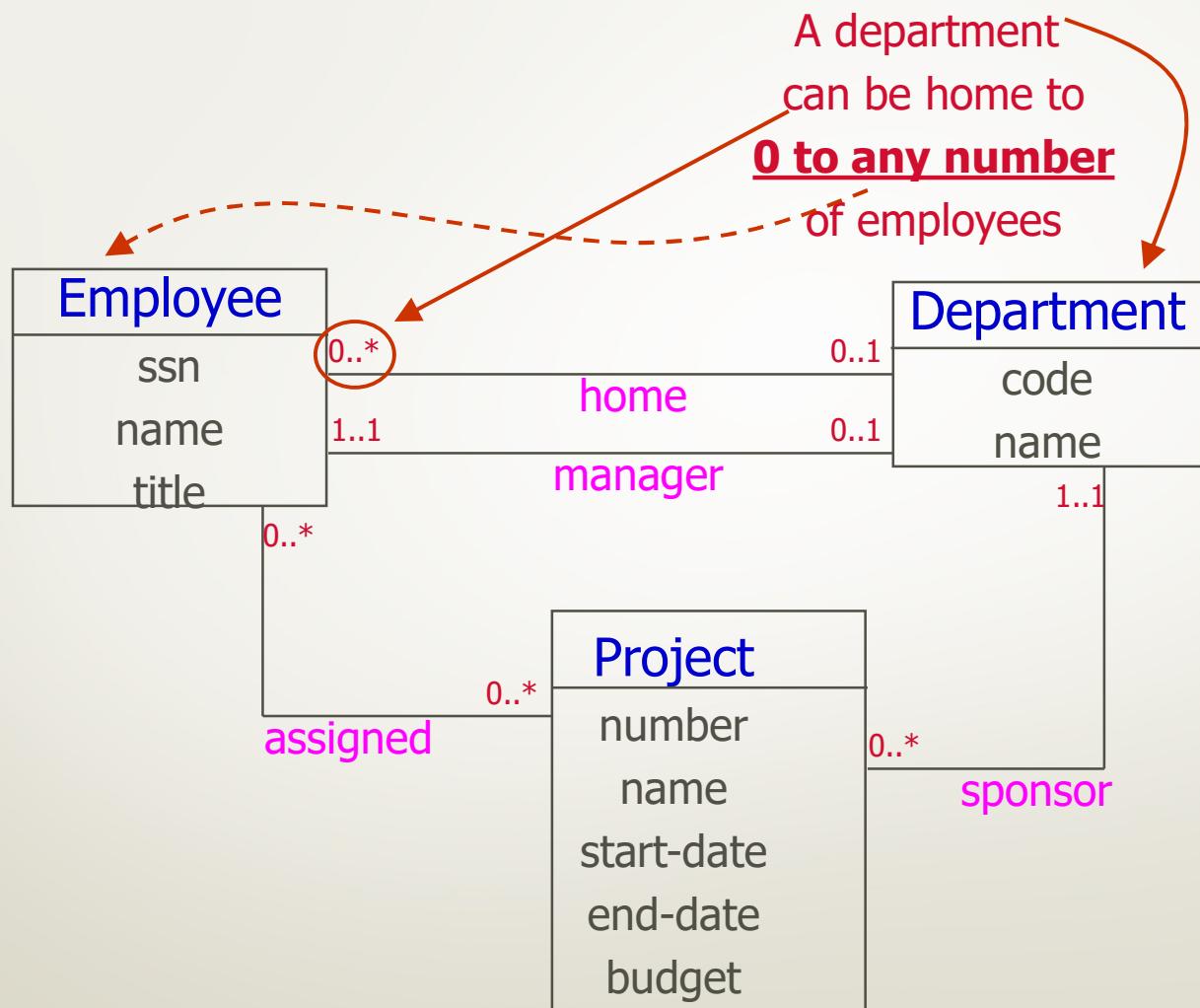
Legend:
Entity set
Relationship set
Attribute name

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

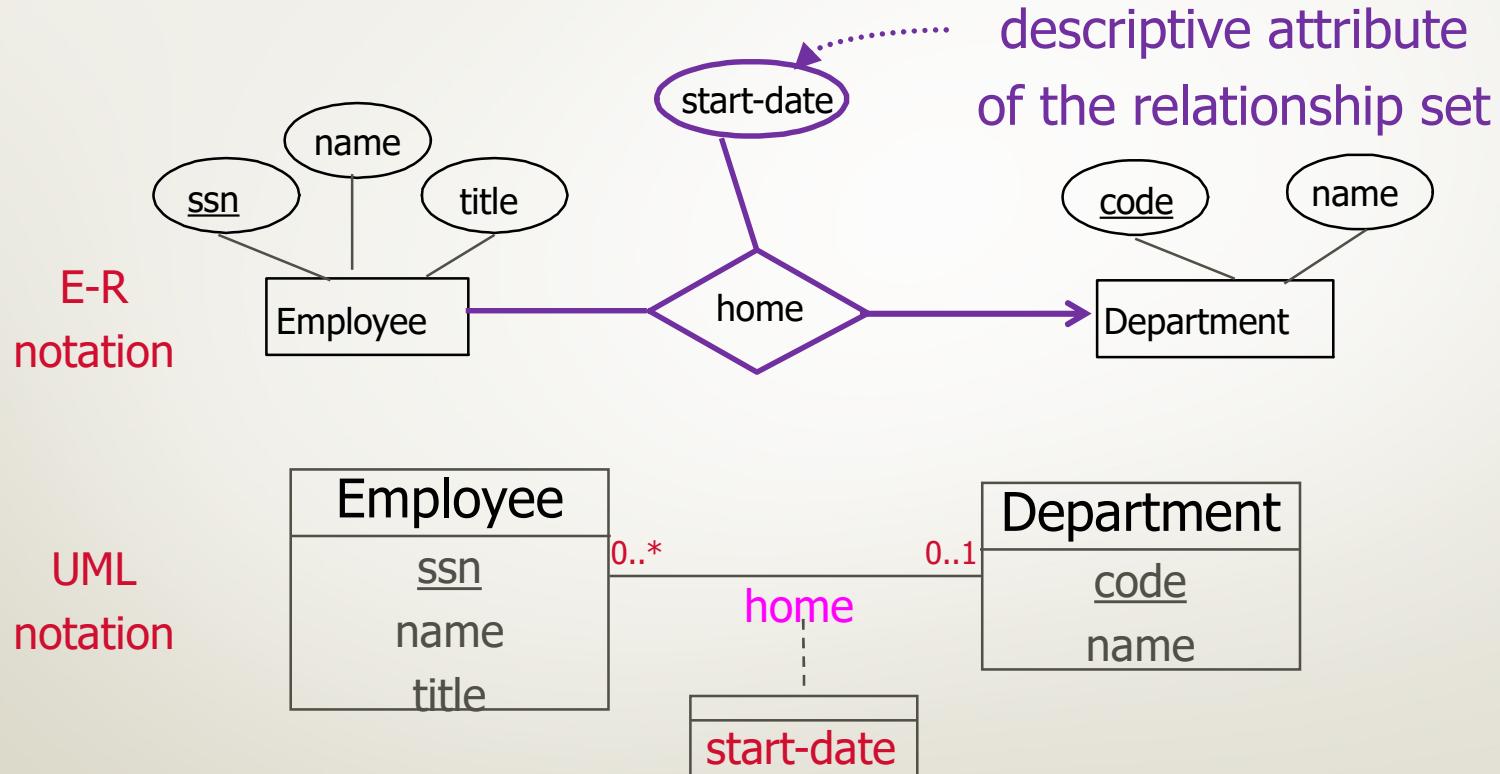


An employee can have **0 or 1** home departments



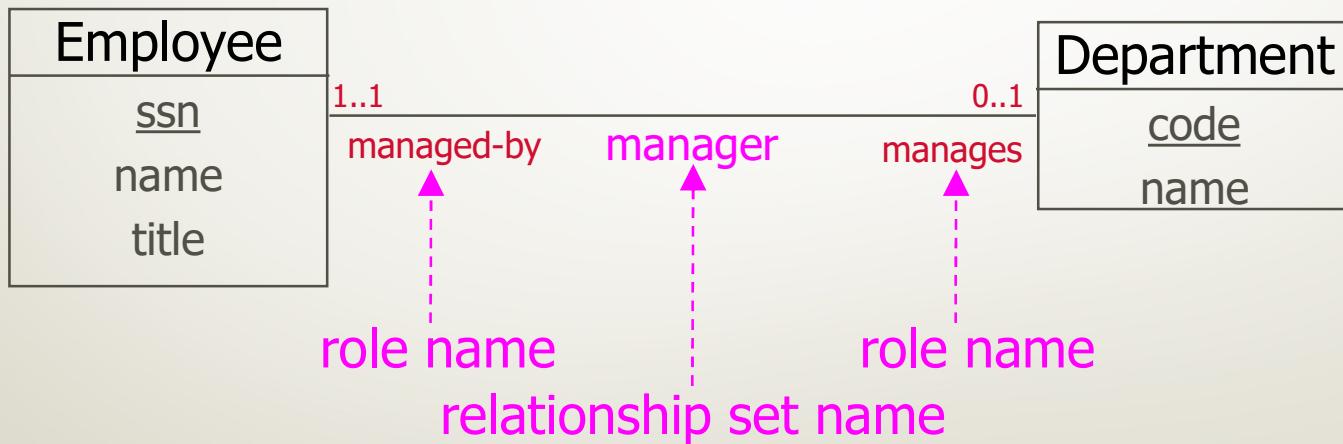


Relationship sets can have attributes

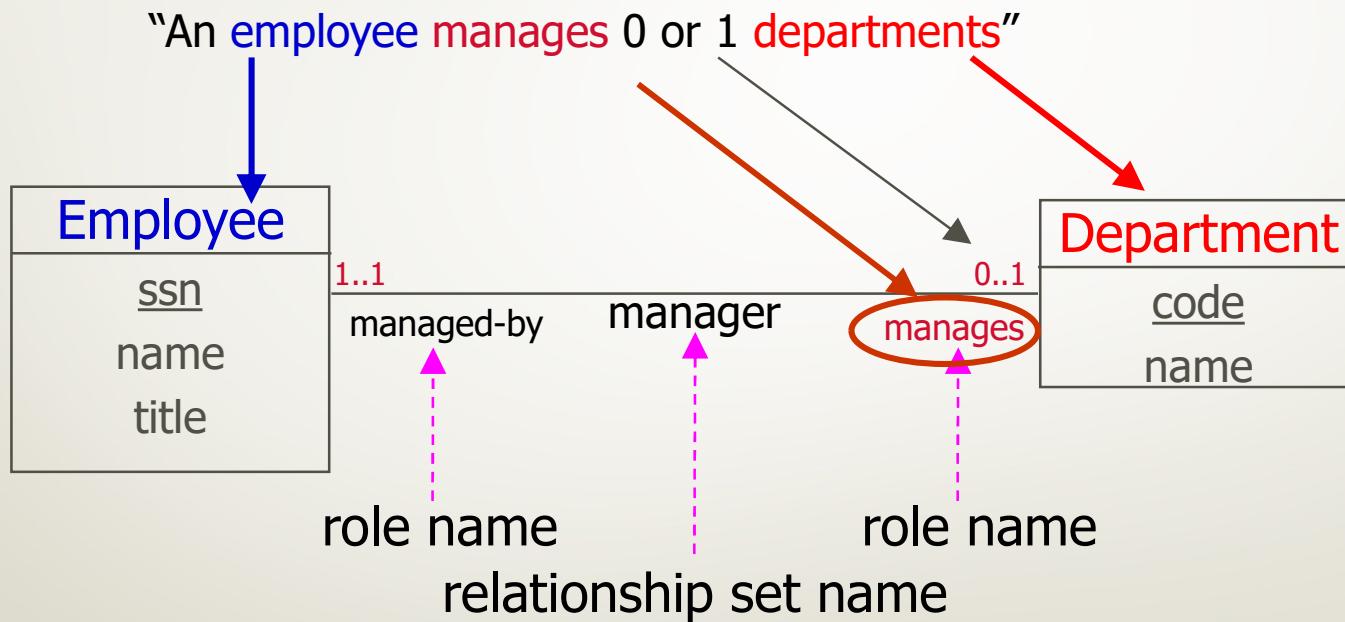


Relationship sets can have **role names**

(in addition to the name of the relationship set)

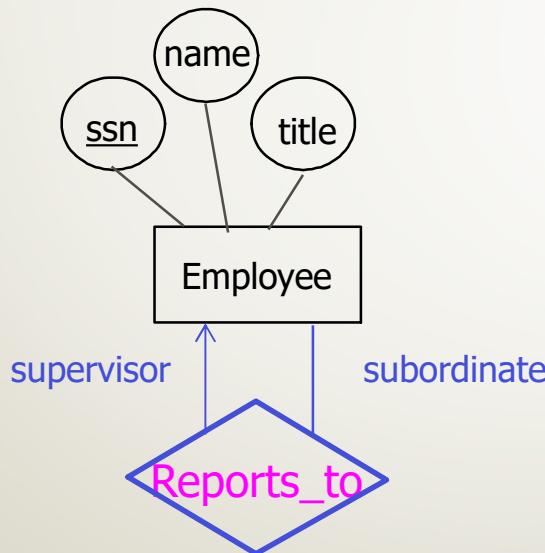


Example: reading role names

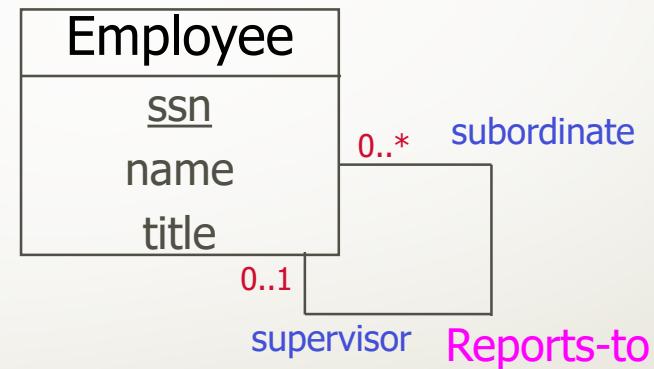


Same entity sets can participate in different “roles” for the same relationship set

E-R
notation



UML
notation



Constraints in ER diagram

- Recall that a constraint is an assertion about the database that must be true at all times
- Part of the database schema = structure
(so it must be part of the ER diagram)
- Very important in database design

Modeling Constraints

Finding constraints is part of the modeling process.

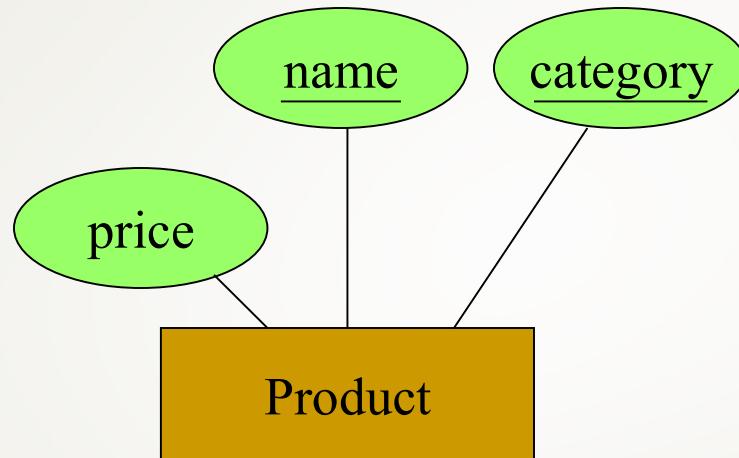
Commonly used constraints:

- **Keys:** attributes that identify entities in an entity set
e.g., social security number uniquely identifies a person.
- **Referential integrity constraints:** relationship-based constraints
e.g., if you work for a company, it must exist in the database.
- **Domain constraints:** peoples' ages are between 0 and 150.
- **General constraints:** all others (at most 50 students enroll in a class)

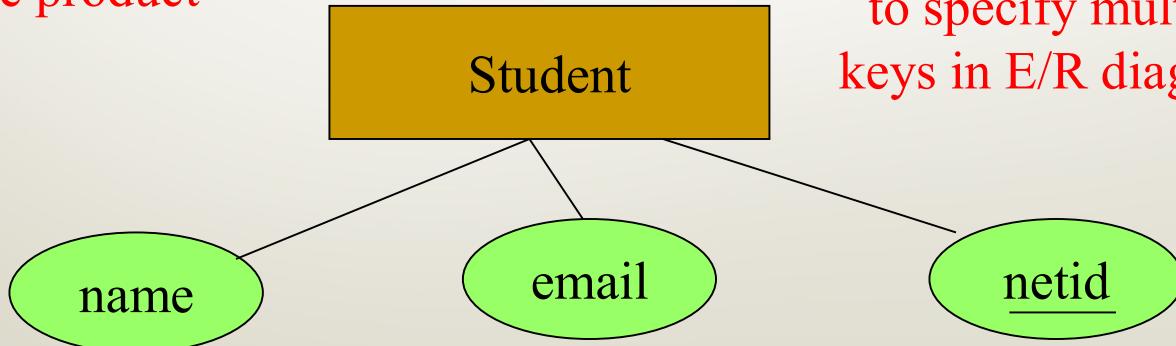
Keys in E/R Diagrams

Underline:

This means “name”
and “category”
together uniquely
determine product



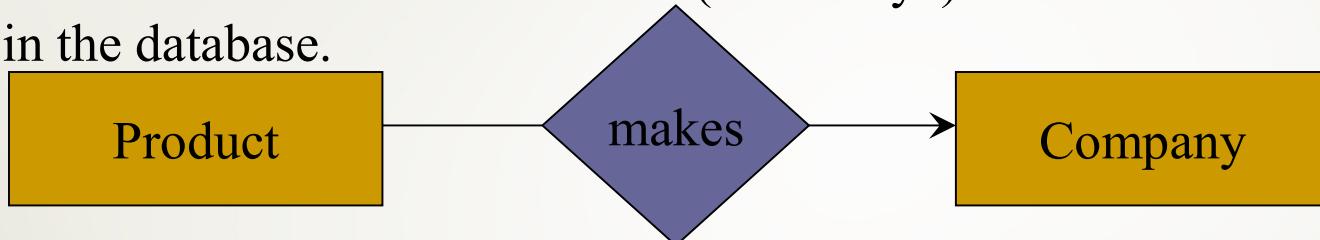
No formal way
to specify multiple
keys in E/R diagrams



Referential Integrity Constraints

Recall: the arrow meant “at most one”.

Each Product must be related to (“made by”) at most one Company in the database.



Wouldn’t it be weird if a product was not associated with any company?



This says “exactly one”.

Each Product must be related to (“made by”) exactly one Company in the database.

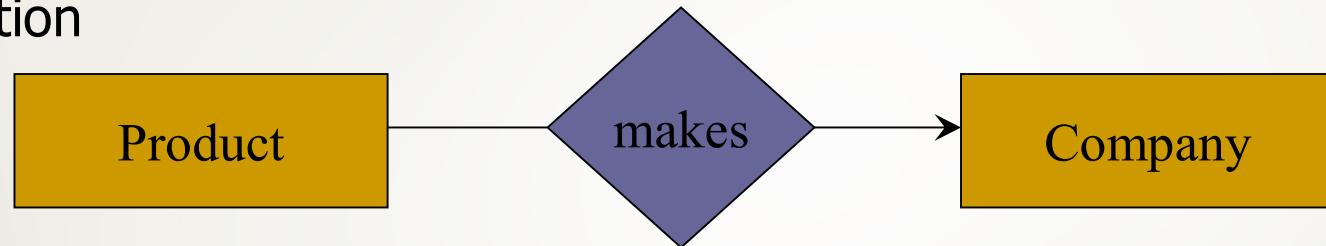
Arrow = at most 1

Semicircle = exactly 1

Referential Integrity Constraints

UML: Aggregation

ER
Notation



UML
Notation

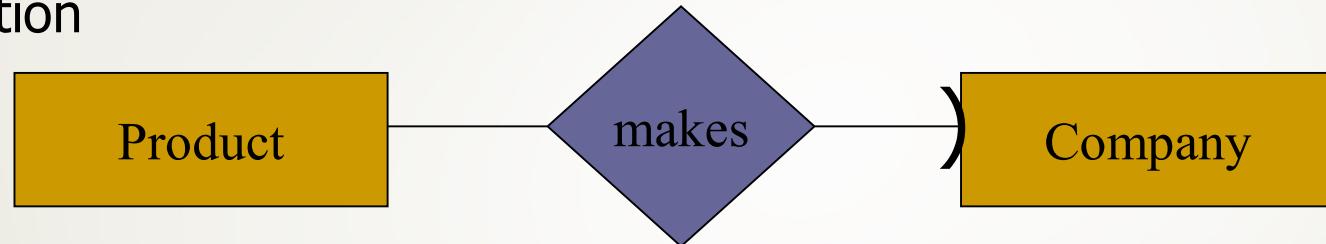


Referential Integrity Constraints

UML: Composition

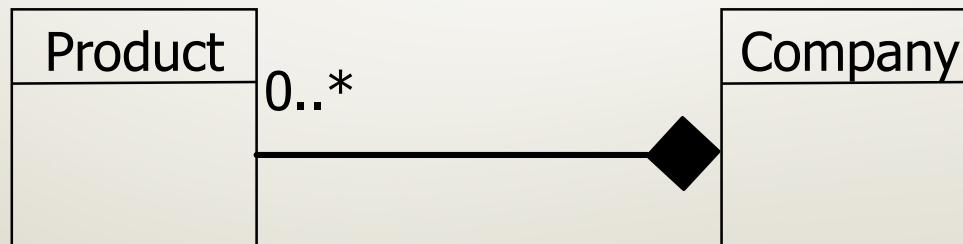
ER

Notation



UML

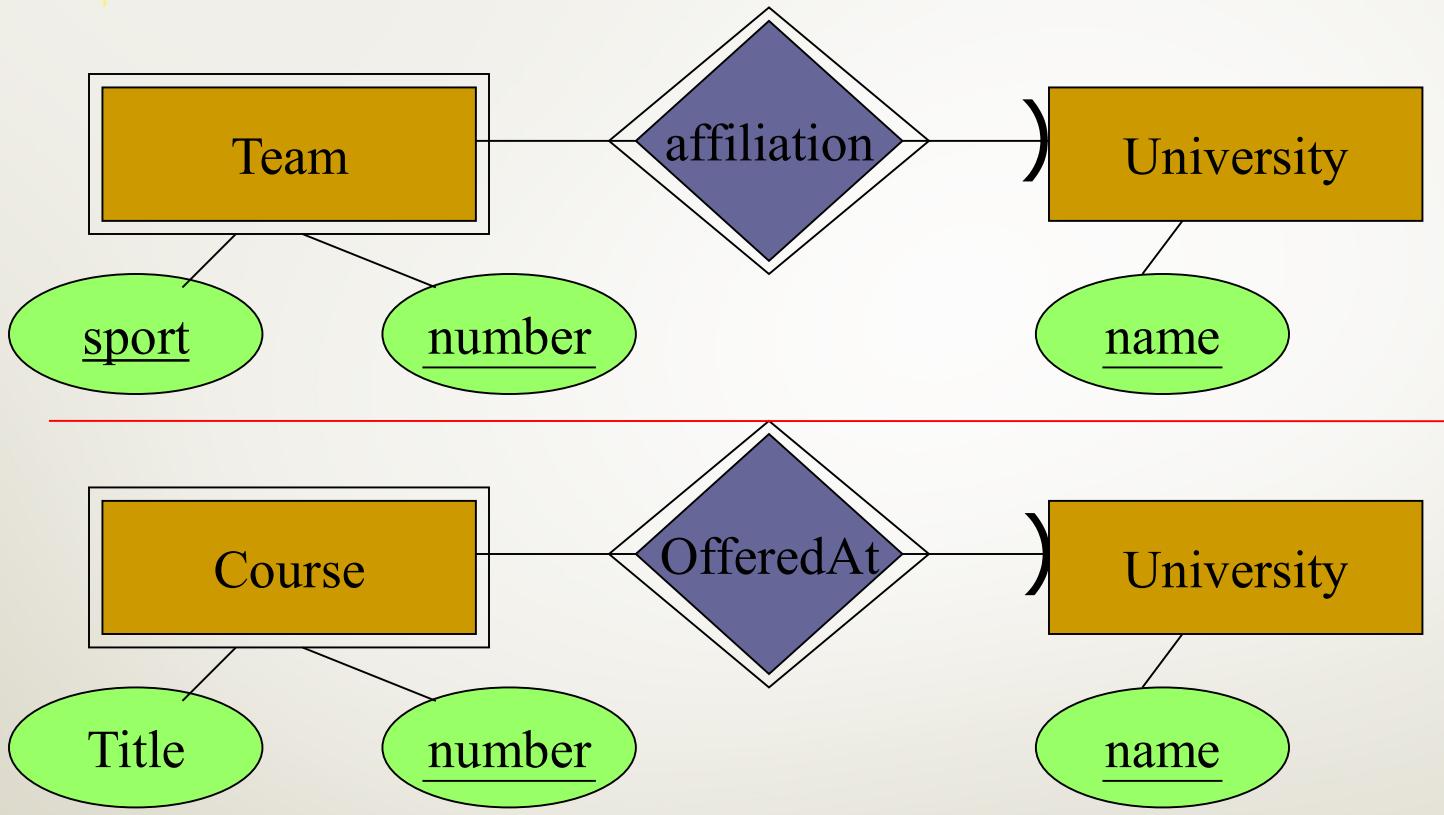
Notation



Weak Entity Sets

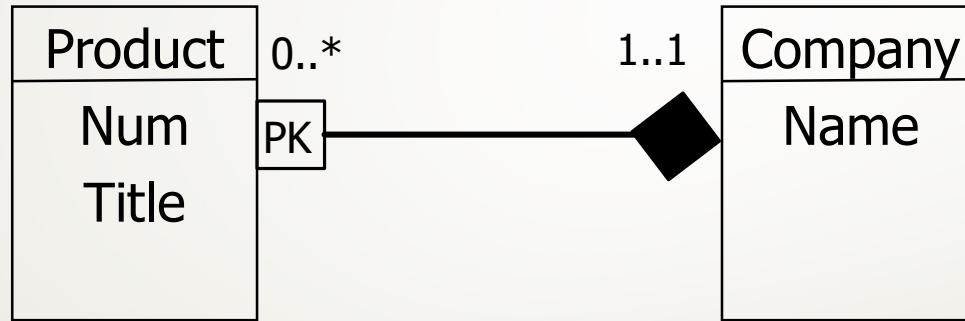
- Occasionally, entities of an entity set need “help” to identify them uniquely.
- Entity set E is *weak* if in order to identify entities of E uniquely, we need to follow one or more many-one relationships from E and include the key of the related entity sets.
- Note: not an is-a relationship because E is not a “subclass” of F : Univ and Team

Notations for weak entity set



- “University” is a “supporting entity set” for “Team”.
- “Affiliation” is a “supporting relationship”.

Weak entity set in UML

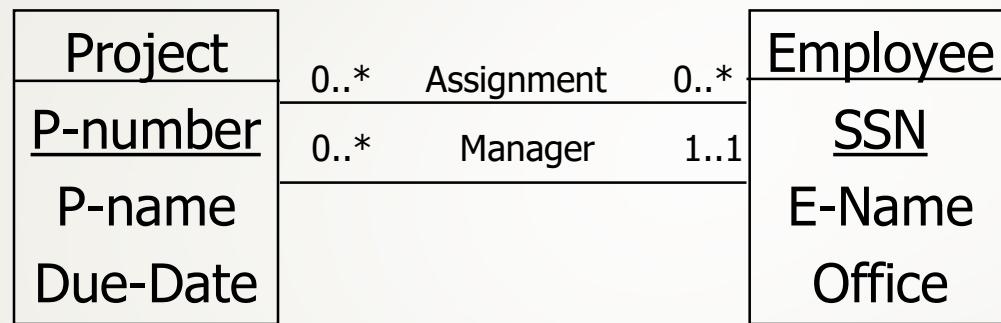




Logical Database Design

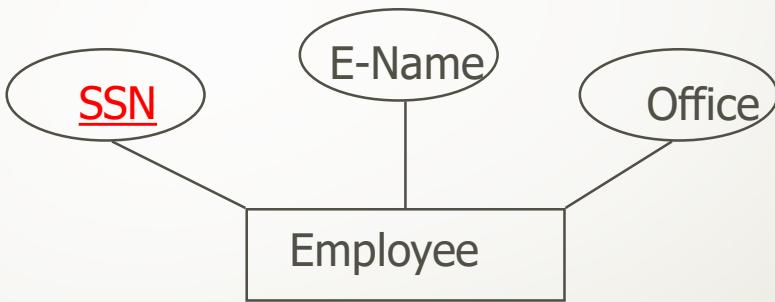
From Entity-Relationship Diagrams OR Unified Modeling Language design
to Relational Database Schema

Converting ER to Relational Schema



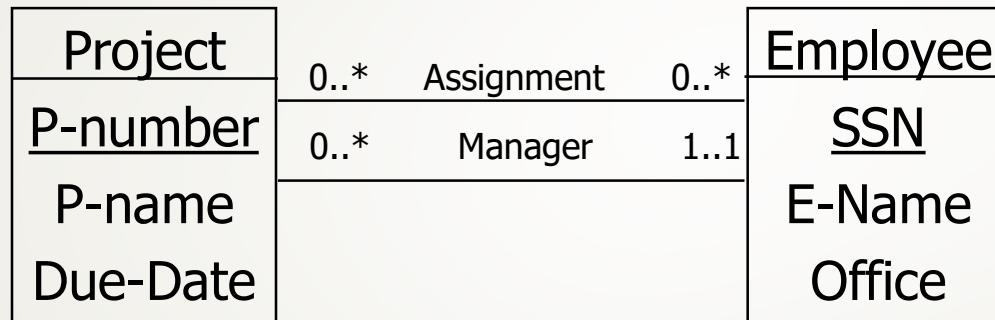
Translate each entity set into a table, with keys.

- Entity set:
 - 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))
```

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

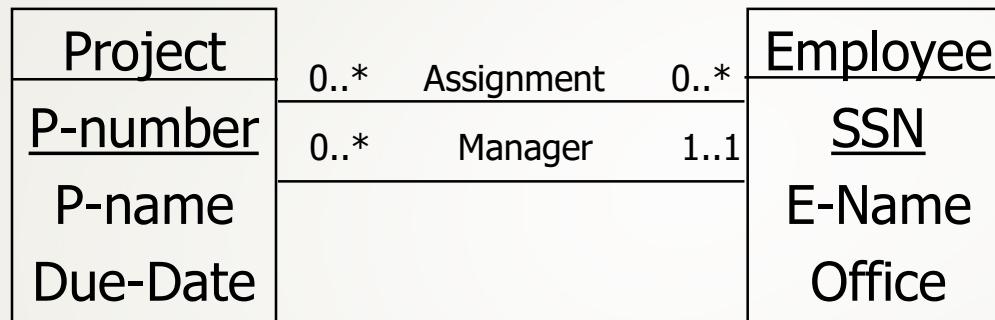


What are the attributes and what is the key for Assignment?

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

Employee(SSN, E-Name, Office)

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



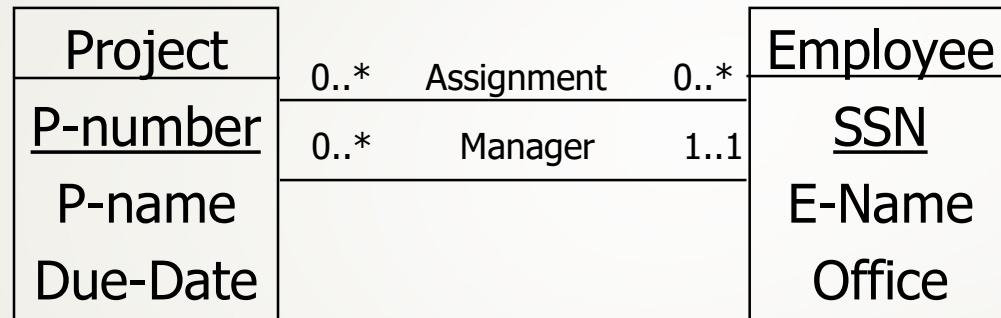
Answer: Assignment(P-Number, SSN)

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 should we do with each **one-to-many** relationship set?

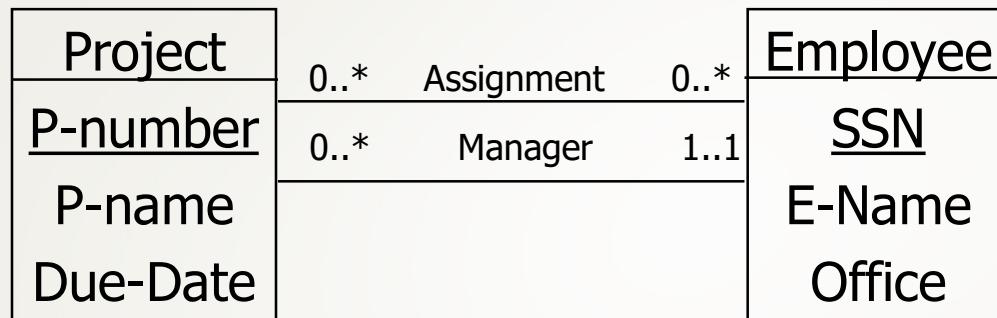


Manager (?)

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

Employee(SSN, E-Name, Office)

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



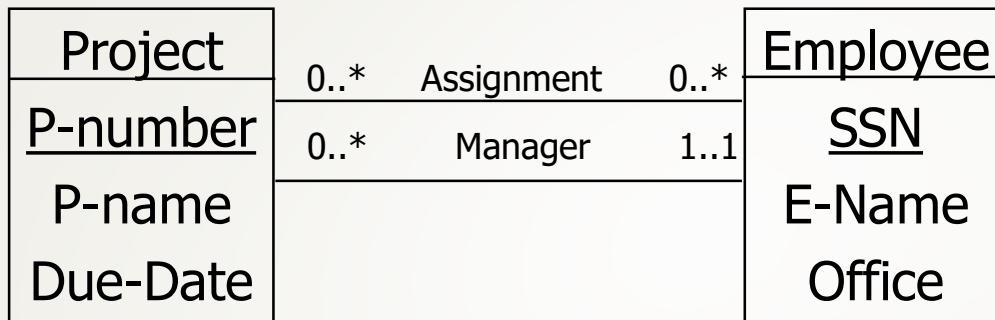
Project(P-number, P-name, Due-Date, **MgrSSN**)

Employee(SSN, E-Name, Office)

MgrSSN is a foreign key (referencing the Employee relation)

value of Manager must match an SSN

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



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

Employee(SSN, E-Name, Office)

vs.

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

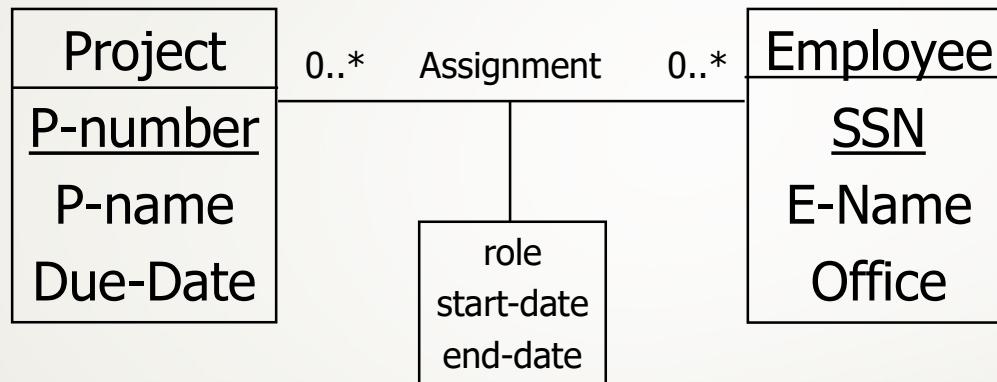
Employee(SSN, E-Name, Office)

Manager(P-number, SSN)

What are the tradeoffs between these two?

Note:
P-number
is the key
for Manager

What do we do when a **many-to-many** relationship set has an attribute?

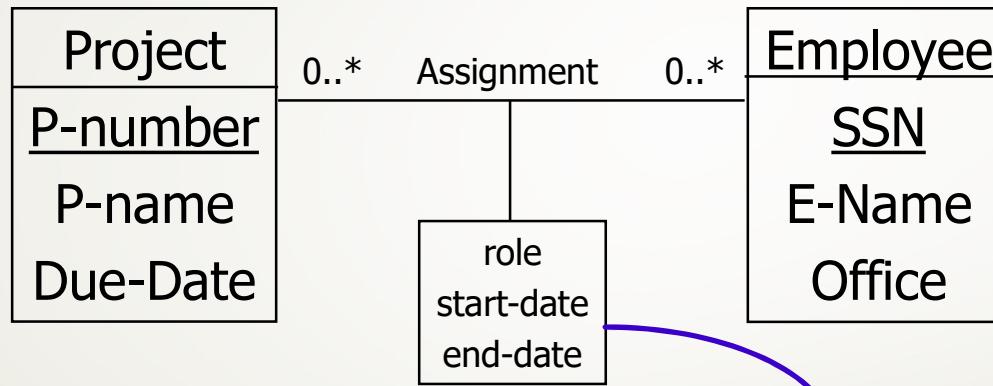


Assignment(P-number, SSN)

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

Employee(SSN, E-Name, Office)

What do we do when a **many-to-many** relationship set has an attribute?

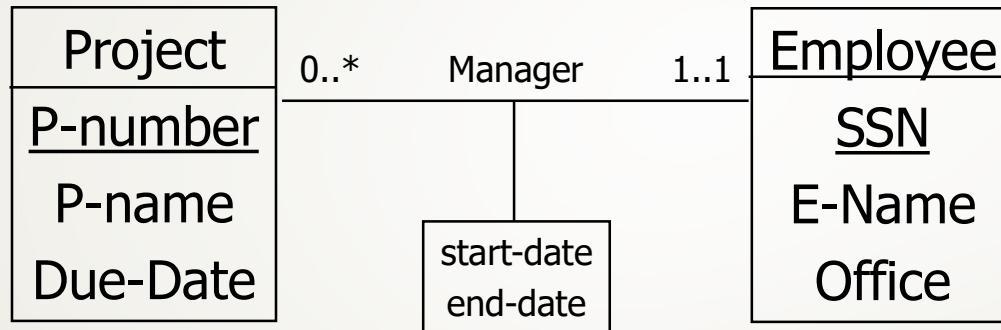


Assignment(P-number, SSN, role, start-date, end-date)

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

Employee(SSN, E-Name, Office)

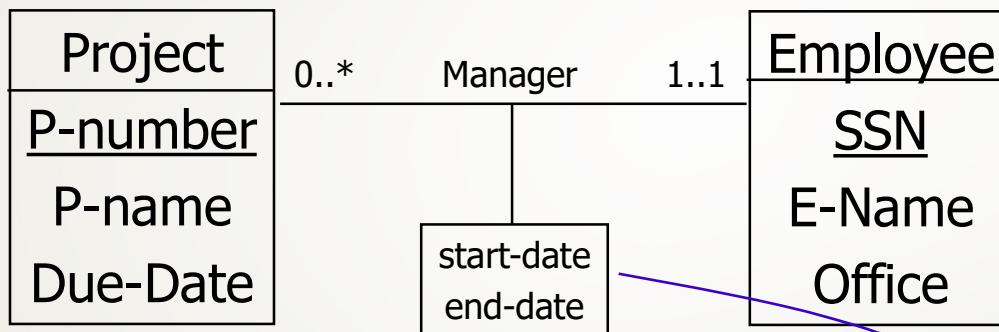
What do we do when a **1-to-many** relationship set has an attribute?



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

Employee(SSN, E-Name, Office)

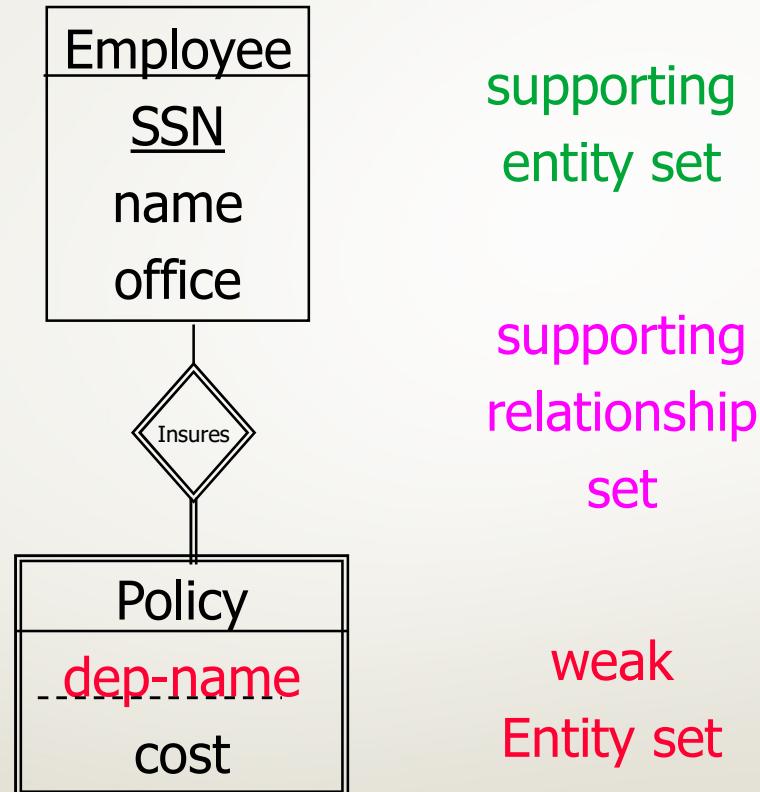
What do we do when a **1-to-many** relationship set has an attribute?



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

Employee(SSN, E-Name, Office)

Weak Entity Sets



Translating Weak Entity Sets

- Weak entity sets and supporting relationship sets are translated into a single table. Must include key of supporting entity set, as a foreign key.
- When the owner entity is deleted, all owned weak entities must also be deleted.

```
CREATE TABLE Insurance_Policy (
    dep-name      CHAR(20),
    cost          REAL,
    ssn          CHAR(11) NOT NULL,
    PRIMARY KEY (dep-name, ssn),
    FOREIGN KEY (ssn) REFERENCES Employee,
    ON DELETE CASCADE)
```