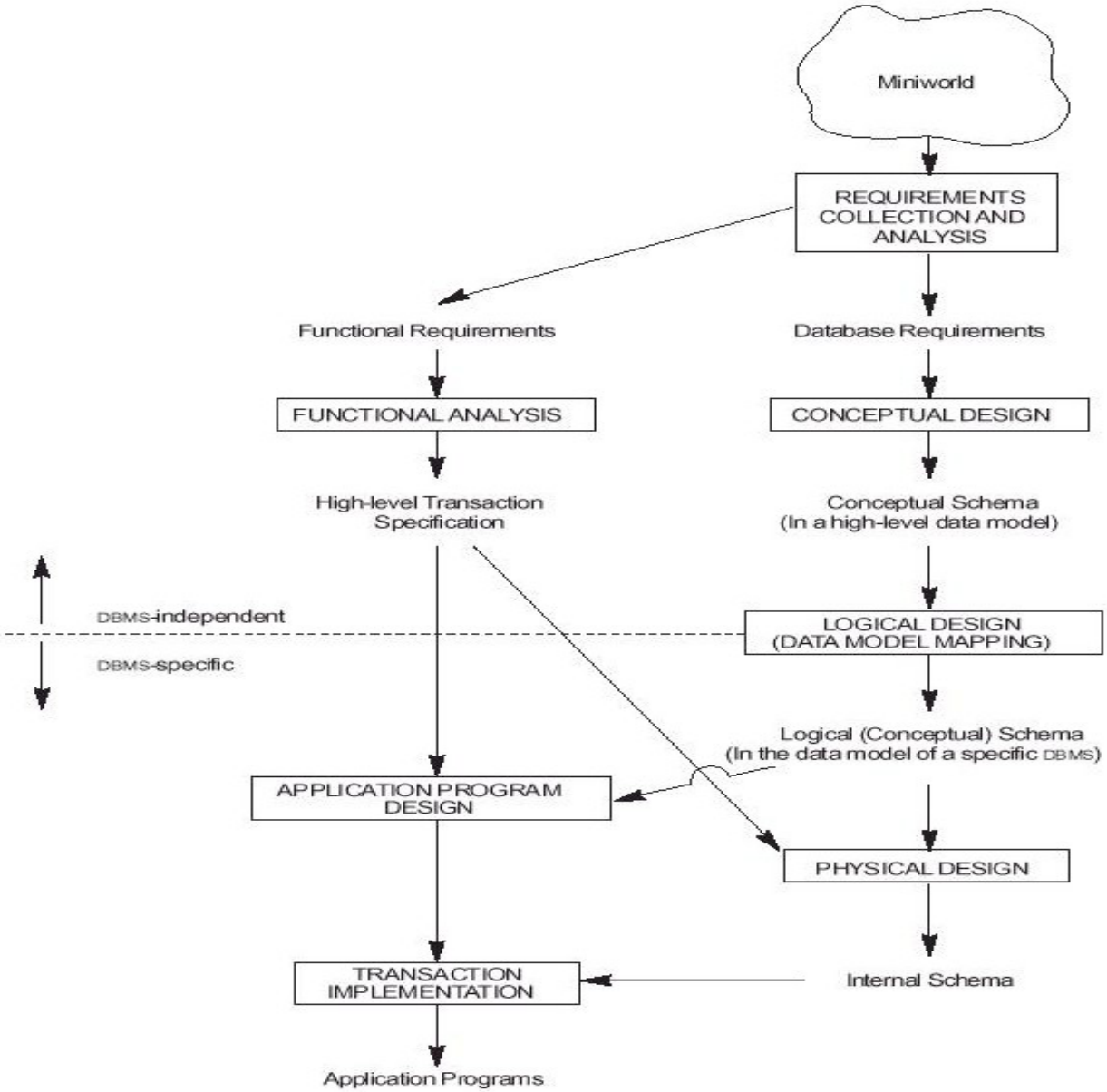
# Review of E-R Data Model

# Database Design

- Goal: specification of database schema
- Methodology:
  - Use E-R model to get a high-level graphical view of essential components of enterprise and how they are related
  - Convert E-R diagram to DDL
- E-R Model: enterprise viewed as set of
  - Entities
  - Relationships among entities


Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Database Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

# Overview of Database Design

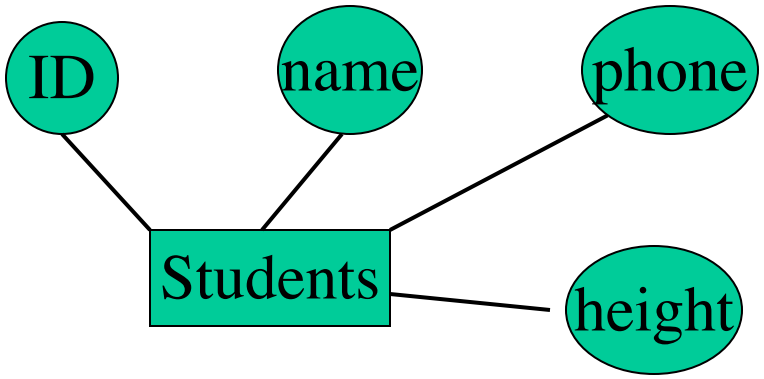
* Conceptual design: ER Model is used at this stage, which is then translated to a relational schema.
- Schema Refinement: (Normalization) Check relational schema for redundancies and related anomalies.
- Physical Database Design: Consider typical workloads and further refine (index, path access, etc) the database design.

# Conceptual Design

- What are the *entities* and *relationships* in the enterprise?
- What information about these entities and relationships should we store in the database?
- What are the *integrity constraints* that hold?
- Represent this information pictorially in *ER diagrams*, then map ER diagram into a relational schema.

# ER Diagrams

- Entities: real world "objects", = "thing."
  - e.g. Students, Courses, Routes, Climbers. Entity sets are drawn as rectangles.
- Entity Type: set of "similar" entities/objects.
- Attributes: properties typically used as column names, similar to fields of a struct
  - e.g. Name, Age, Height. Drawn as ovals.

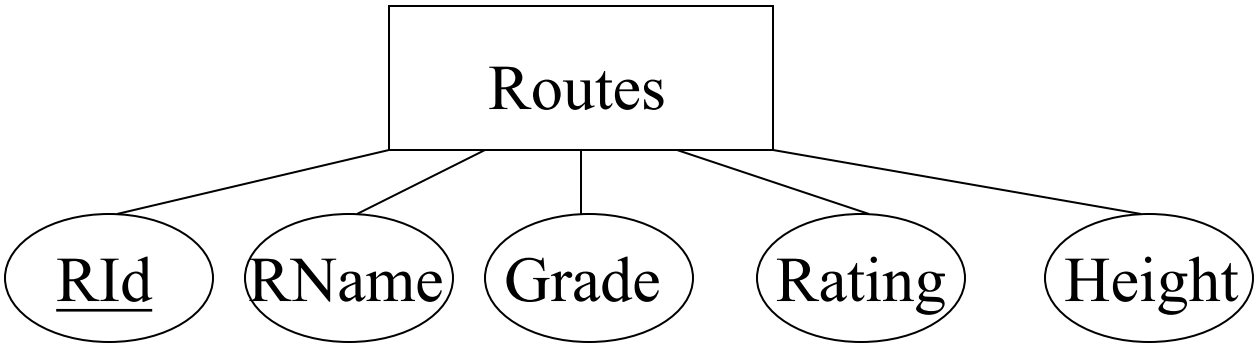


# Relationships

- Connect two or more entity sets.
- Represented by diamonds.

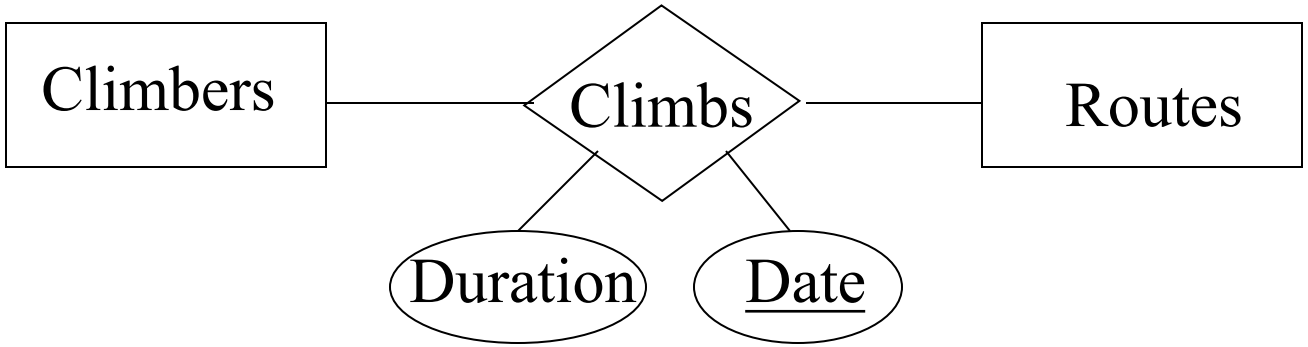
# Drawing Entity Type

- The term “entity” and “entity type” are often confused. Boxes represent *sets* of entities.
- To draw an entity set we connect it with its attributes and indicate the key by underlining it:



# Drawing Relationships

- Relationship types are represented by diamonds.
- They connect the entities they relate, and may additionally have attributes.



# Structural Constraints – one way to express semantics of relationships

**Structural constraints on relationships:**

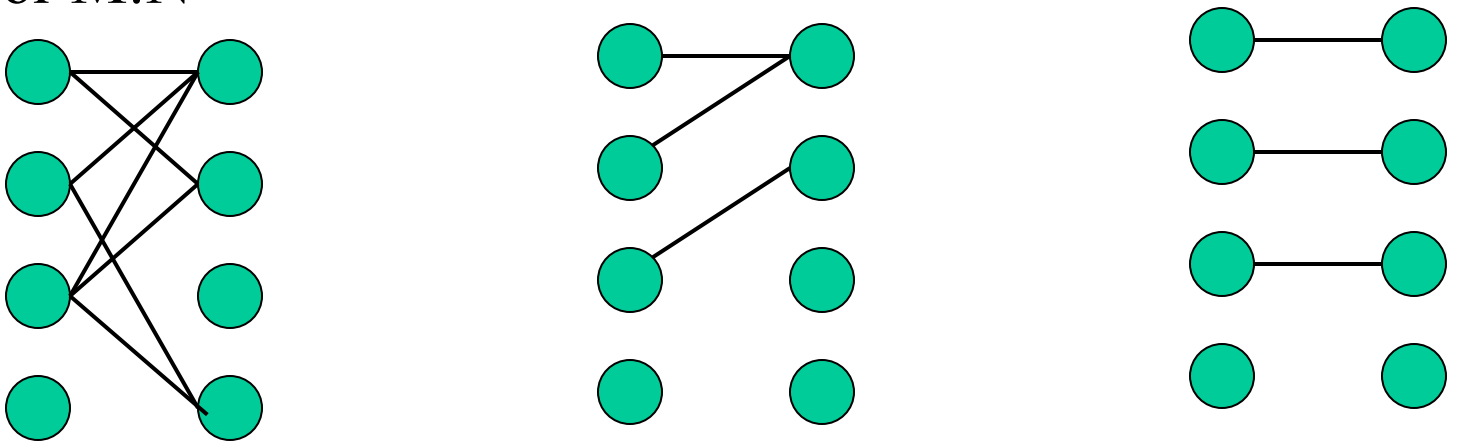
- **Cardinality ratio** (of a binary relationship): 1:1, 1:N, N:1, or M:N

- **Participation constraint** (on each participating entity type)

# Structural Constraints – one way to express semantics of relationships

**Structural constraints on relationships:**

- **Cardinality ratio** (of a binary relationship): 1:1, 1:N, N:1, or M:N



- **Participation constraint** (on each participating entity type)

# Structural Constraints – one way to express semantics of relationships

**Structural constraints on relationships:**

- **Cardinality ratio** (of a binary relationship): 1:1, 1:N, N:1, or M:N

  **SHOWN BY PLACING APPROPRIATE NUMBER ON THE LINK.**

- **Participation constraint** (on each participating entity type):
  - total (called *existence dependency*)
  - partial.

  **SHOWN BY DOUBLE LINING THE LINK**

# Alternative (min, max) notation for structural constraint

- Specified on *each participation* of an entity type E in a relationship type R
- Specifies that each entity e in E participates in *at least* min and *at most* max relationship instances in R
- Default(no constraint): min=0, max=n
- Must have min≤max, min≥0, max ≥1
- Derived from the knowledge of mini-world constraints

# Alternative (min, max) notation for structural constraints:

Examples:

- A degree program can have any # of student majoring in and a student can major in at most 4 different programs at the same school.

# Alternative (min, max) notation for structural constraints:

Examples:

- A degree program can have any # of student majoring in and a student can major in at most 4 different programs at the same school.
  - Specify (0,4) for participation of Student in MajorsIn
  - Specify (0,N) for participation of Program in MajorsIn.

# Alternative (min, max) notation for structural constraints:

Examples:

- A degree program can have any # of student majoring in and a student can major in at most 4 different programs at the same school.
  - Specify (0,4) for participation of Student in MajorsIn
  - Specify (0,N) for participation of Program in MajorsIn.

- A program can be offered to any number of student, but a student majors in exactly one program.

# Alternative (min, max) notation for structural constraints:

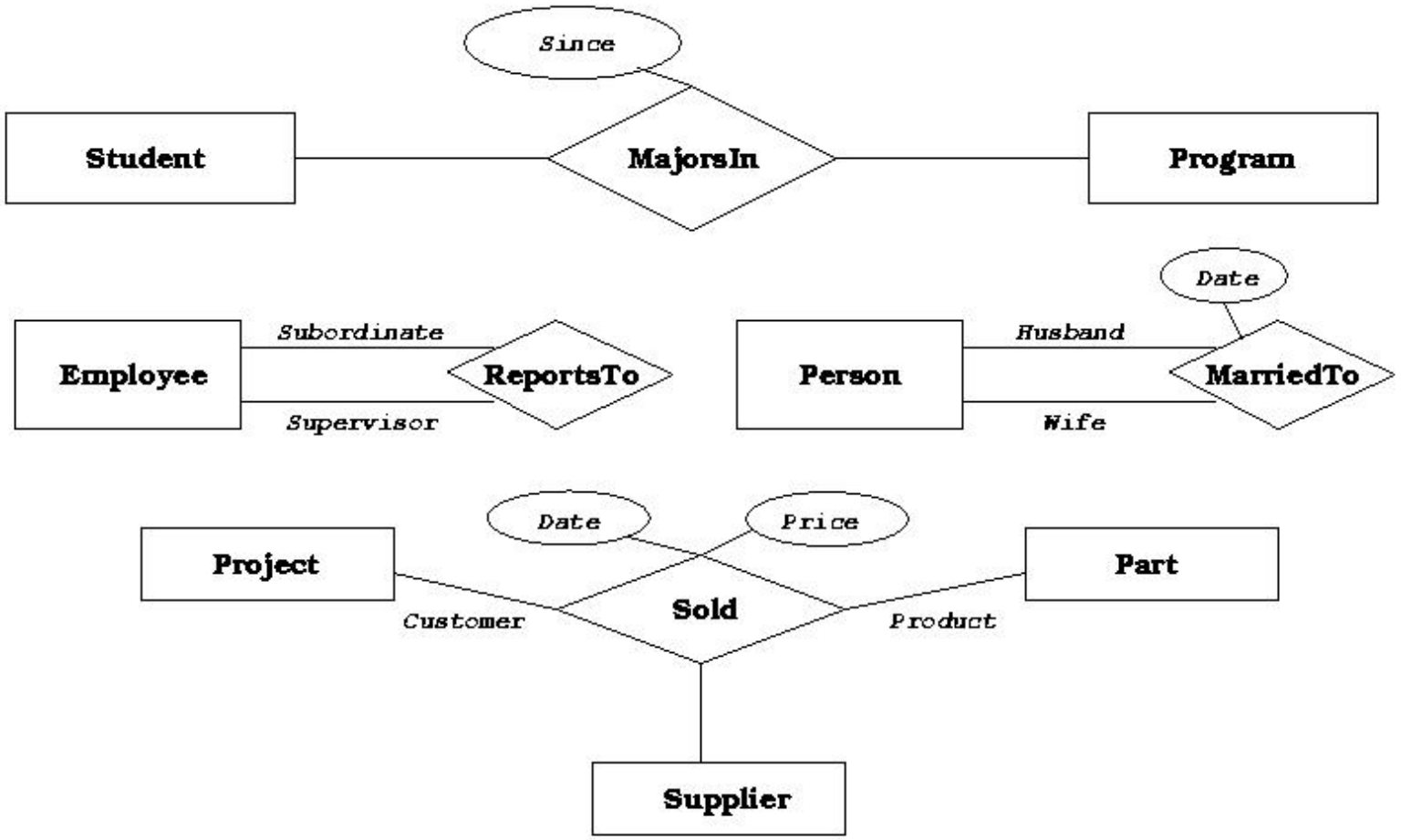
Examples:

- A degree program can have any # of student majoring in and a student can major in at most 4 different programs at the same school.
  - Specify (0,4) for participation of Student in MajorsIn
  - Specify (0,N) for participation of Program in MajorsIn.

- A program can be offered to any number of student, but a student majors in exactly one program.
  - Specify (1,1) for participation of Student in MajorsIn.
  - Specify (0,N) for participation of Program in MajorsIn.

# Graphical Representation

- Roles are edges labeled with role names (omitted if role name = name of entity set). Most attributes have been omitted.

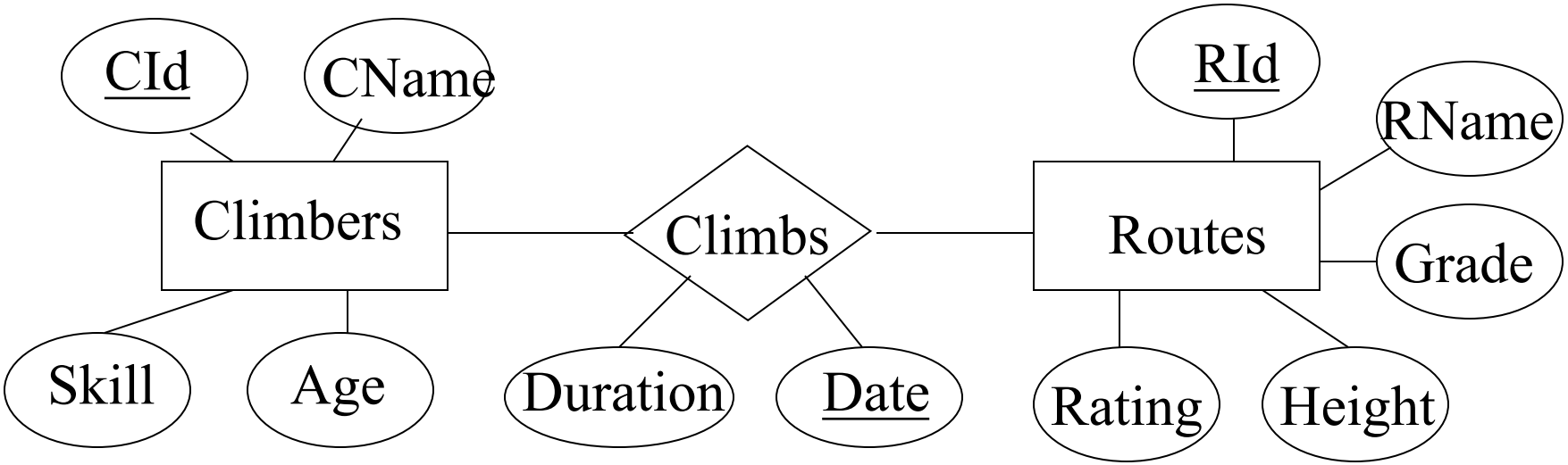


# The whole diagram

Note:

lines connect entities to attributes and relationships to entities and to attributes.

No connection between attributes to attributes, entities to entities or relationships to relationships.



# Weak Entity Types

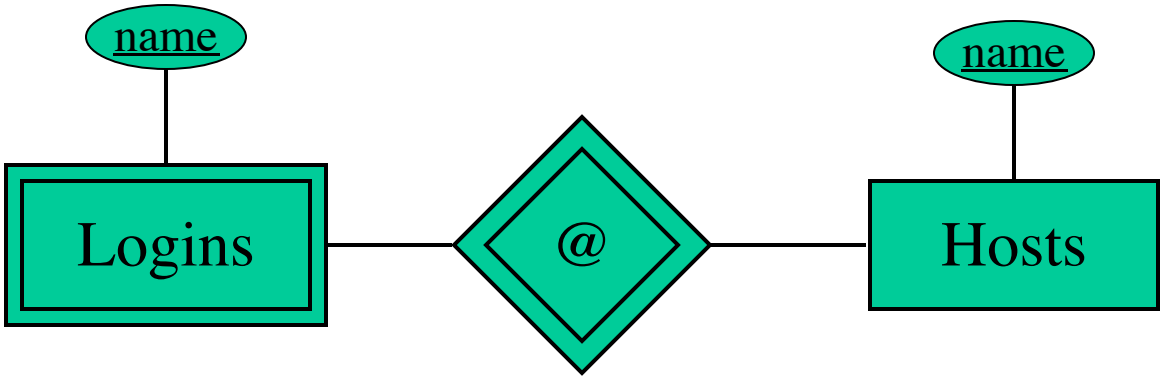
Sometimes an E.T. *E*'s key comes not (completely) from its own attributes, but from the keys of one or more E.T.'s to which *E* is linked by a *supporting (identifying)* many-one relationship.

- Called a *weak* E.S.
- Represented by putting double rectangle around *E* and a double diamond around each supporting relationship.
- Many-one-ness of supporting relationship (includes 1-1) essential.
  - With many-many, we wouldn't know which entity provided the key value.
- "Exactly one" also essential, or else we might not be able to extract key attributes by following the supporting relationship.

# Weak Entity Types

- An entity that does not have a key attribute
- A weak entity must participate in an identifying relationship type with an owner or identifying entity type
- Entities are identified by the combination of:
  - A partial key of the weak entity type
  - The particular entity they are related to in the identifying entity type

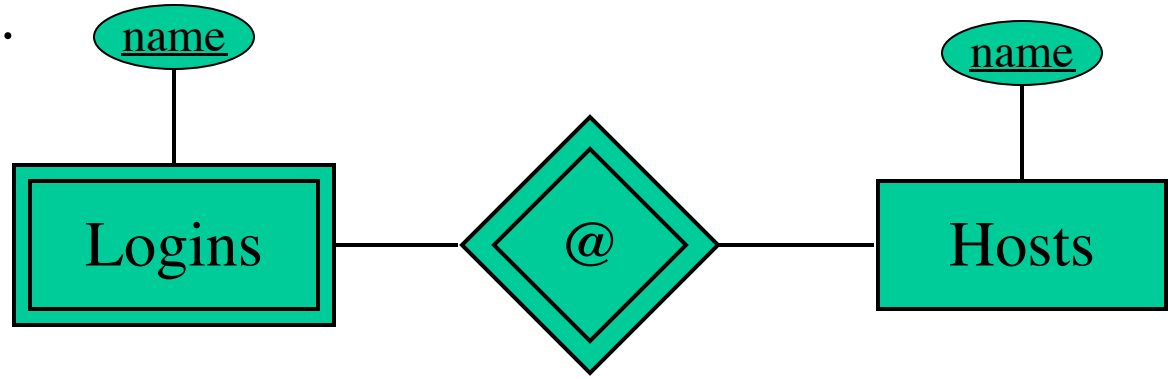
# Example: Logins (Email Addresses)



# Example: Logins (Email Addresses)

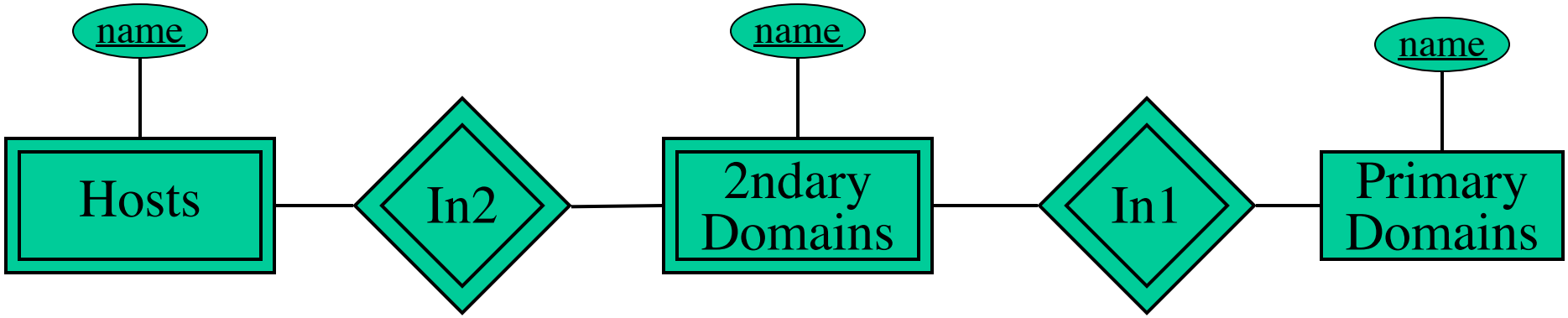
Login name = user name + host name, e.g., `abc@cs.umn.edu`.

- A "login" entity corresponds to a user name on a particular host, but the passwd table doesn't record the host, just the user name, e.g., `abc`.
- Key for a login = the user name at the host (which is unique for that host only) + the IP address of the host (which is unique globally).



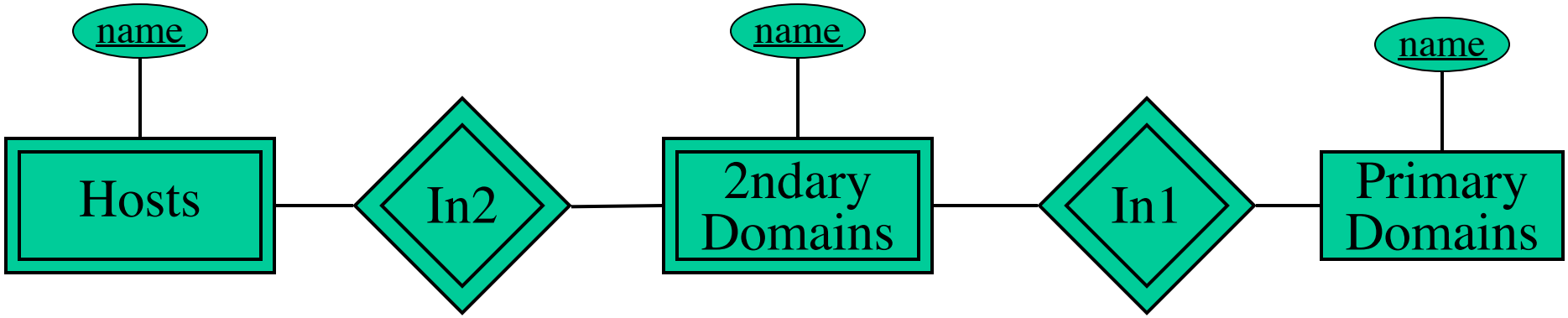
# Example: Chain of "Weakness"

Consider IP addresses consisting of a primary domain (e.g., `edu`), subdomain (e.g., `umn`), and host (e.g., `cs`).



# Example: Chain of "Weakness"

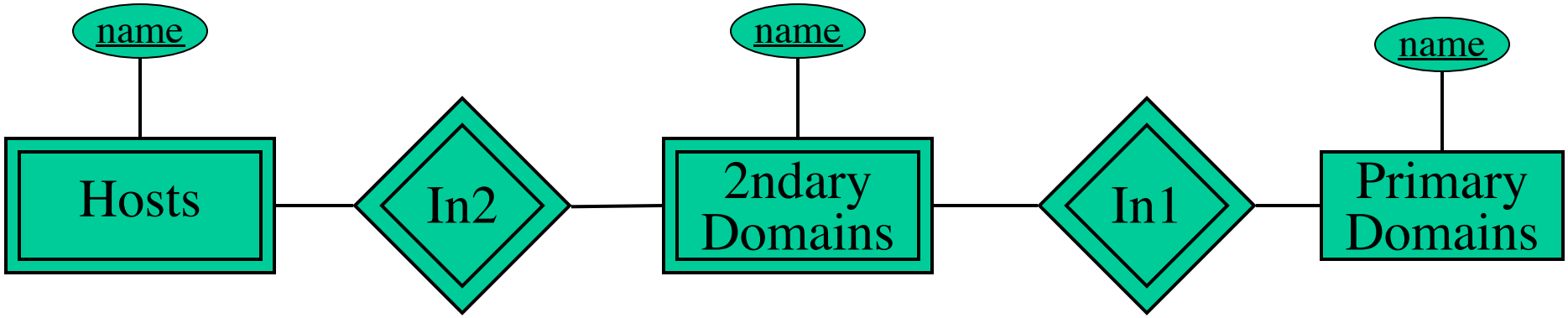
Consider IP addresses consisting of a primary domain (e.g., `edu`), subdomain (e.g., `umn`), and host (e.g., `cs`).




- Key for primary domain = its name.

# Example: Chain of "Weakness"

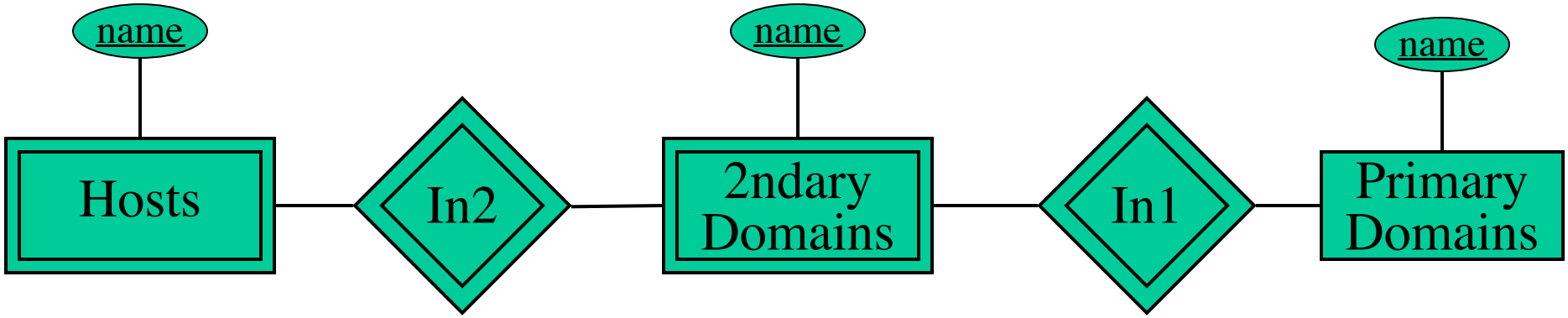
Consider IP addresses consisting of a primary domain (e.g., `edu`), subdomain (e.g., `umn`), and host (e.g., `cs`).




- Key for primary domain = its name.
- Key for secondary domain = its name + name of primary domain.

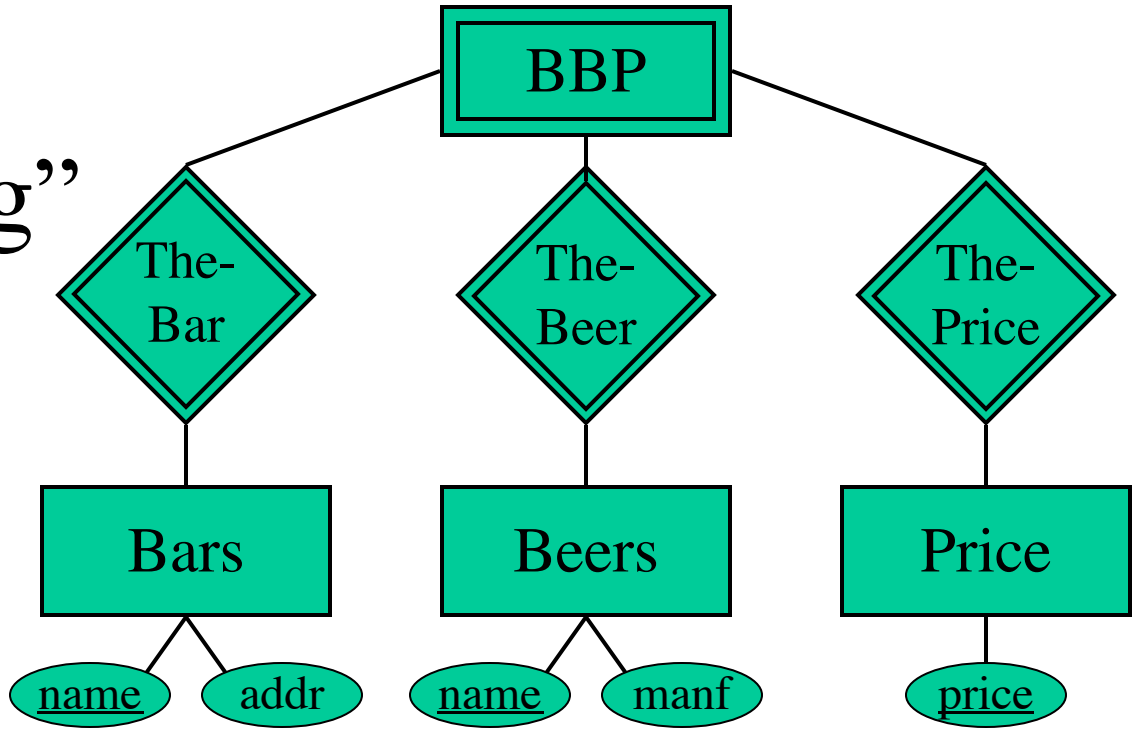
# Example: Chain of "Weakness"

Consider IP addresses consisting of a primary domain (e.g., `edu`), subdomain (e.g., `umn`), and host (e.g., `cs`).

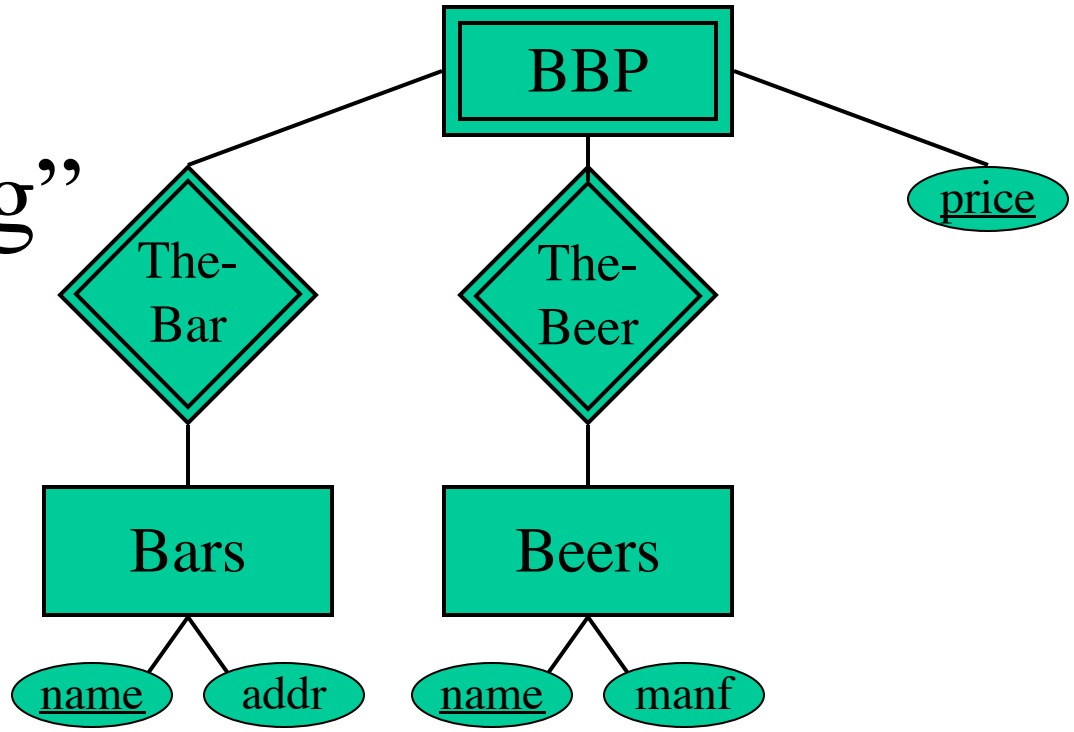



- Key for primary domain = its name.
- Key for secondary domain = its name + name of primary domain.
- Key for host = its name + key of secondary domain = its name + name of secondary domain + name of primary domain.

# All "Connecting" Entity Sets Are Weak



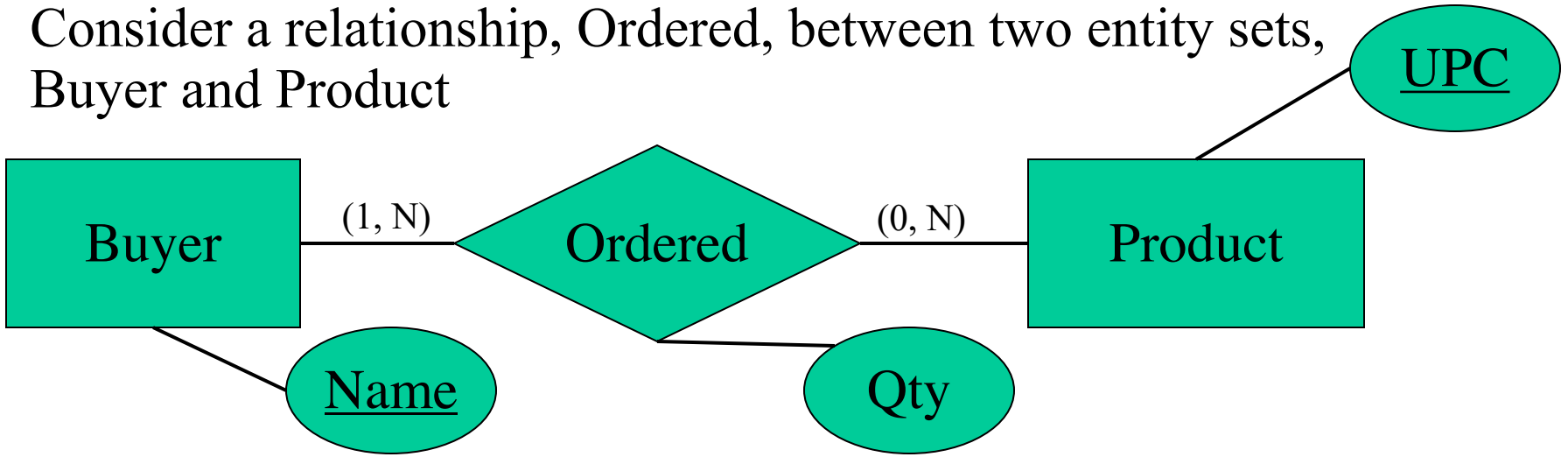
# All "Connecting" Entity Sets Are Weak




- In this special case, where bar and beer determine a price, we can omit `price` from the key, and remove the double diamond from `ThePrice`.
- Better: `price` is attribute of `BBP`.

# Relationship Vs Weak Entities

- Consider a relationship, Ordered, between two entity sets, Buyer and Product




- How can we add Shipments to track Buyer's order status?

# Design Issues

Setting: client has (possibly vague) idea of what they want. You must design a database that represents these thoughts and only these thoughts.

## Avoid redundancy

= saying the same thing more than once.

- Wastes space and encourages inconsistency.

## Example

# Design Issues

Setting: client has (possibly vague) idea of what they want. You must design a database that represents these thoughts and only these thoughts.

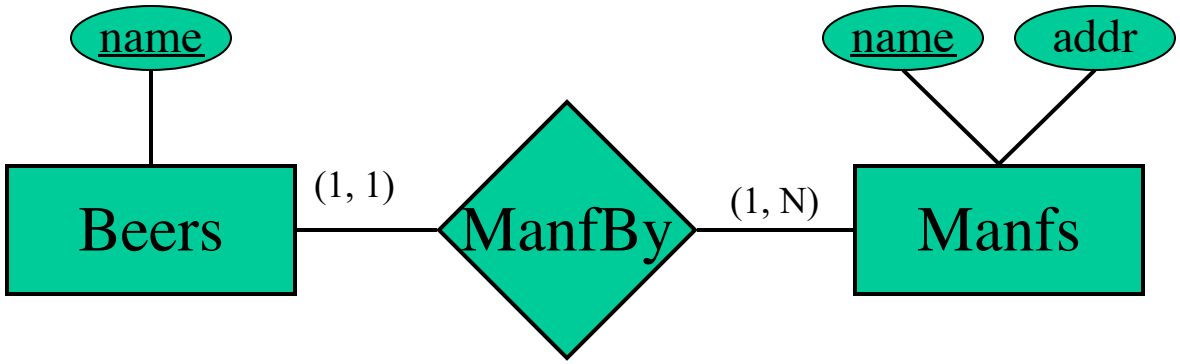
# Avoid redundancy

= saying the same thing more than once.

•Wastes space and encourages inconsistency.

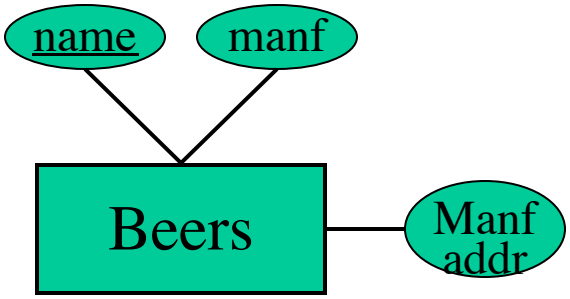
# Example

Good:

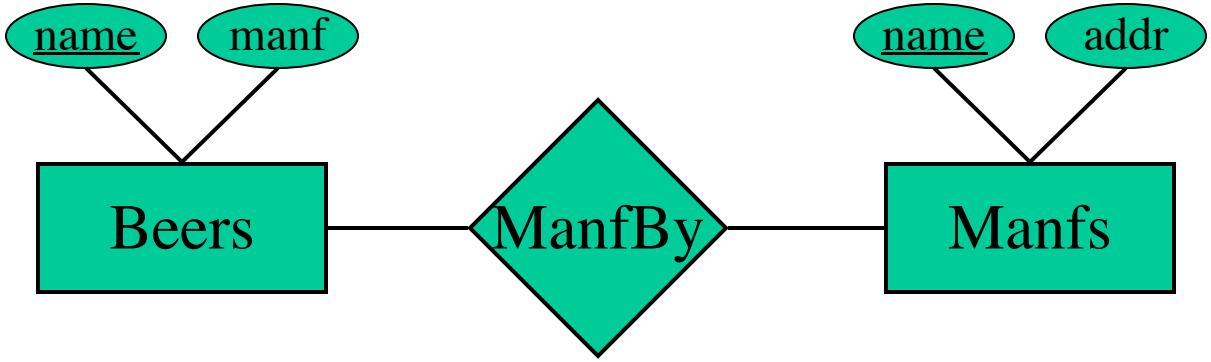


# Example

Bad: repeats manufacturer address for each beer they manufacture.




Bad: manufacturer's name said twice.



# Use Schema to Enforce Constraints

- The design ***schema*** should enforce as many constraints as possible.
  - Don't rely on future data to follow assumptions.

# Example

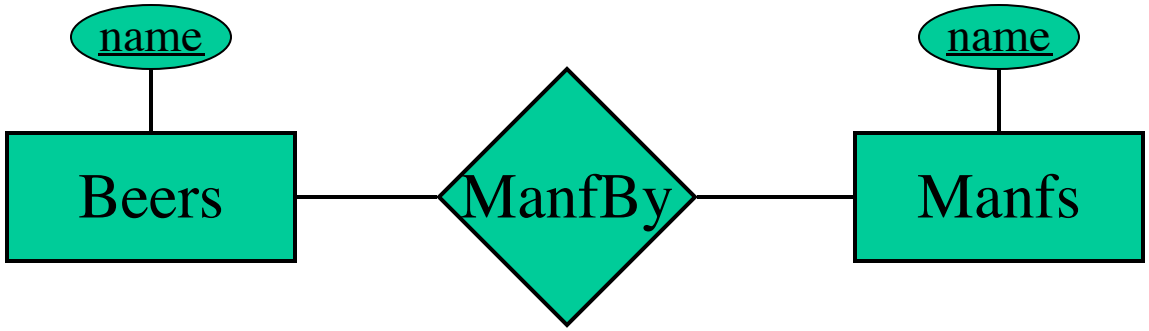
- If registrar wants to associate only one instructor with a course,
- don't allow sets of instructors and count on departments to enter only one instructor per course.

# Entity Sets Vs. Attributes

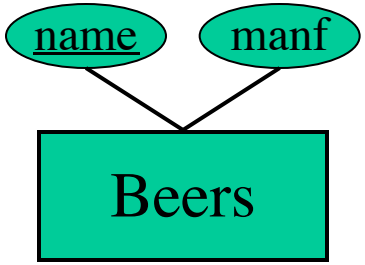
You may be unsure which concepts are worthy of being entity sets, and which are handled more simply as attributes.

- Don't create needless entity sets to make project "larger."

Bad:




Good:



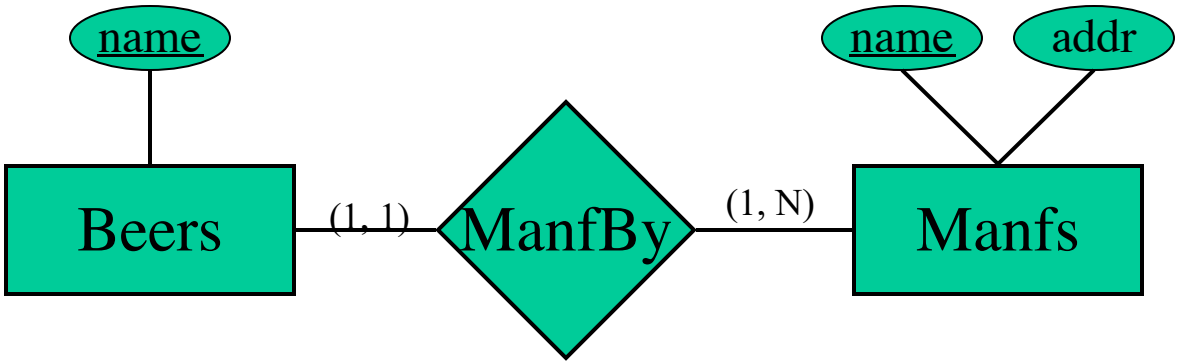
# Intuitive Rule for E.S. Vs. Attribute

Make an entity set only if it either:

1. Is more than a name of something; *i.e.*, it has nonkey attributes or relationships with a number of different entity sets, or
2. Is the "many" in a many-one relationship.

# Example

The following design illustrates both points:




- *Manfs* deserves to be an E.S. because we record *addr*, a nonkey attribute.
- *Beers* deserves to be an E.S. because it is at the "many" end.
  - If not, we would have to make "set of beers" an attribute of *Manfs* – something we avoid doing, although some may tell you it is OK in E/R model.

# Don't Overuse Weak E.T.

- There is a tendency to feel that no E.T. has its entities uniquely determined without following some relationships.
- However, in practice, we almost always create unique ID's to compensate: social-security numbers, VIN's, etc.

# LIMITATIONS OF THE ER MODEL

- No relationship may be defined between an entity type and a relationship type
- No relationship may be defined between an entity type and a collection of entity types from which any one type may participate
  - e.g. Entity type1 : POLICY-HOLDER may be an individual, multiple individuals, one organization, or many organizations
  - Entity type2 : POLICY

# LIMITATIONS OF THE ER MODEL

- No constraints (e.g., exclusion, co-existence etc.) among relationship types
  - NIAM model, UML class diagrams allow them
  - UML (Unified Modeling Language) is a standard way to specify, construct, and document systems that use object-oriented code such as Java, C++

# Extended Entity-Relationship (EER)Model

- Incorporates Set-subset Relationships
- Incorporates Generalization Hierarchies
- Constraints:
  - Coverage Constraints: partial vs. total
  - Disjointedness Constraint: disjoint vs. overlapping

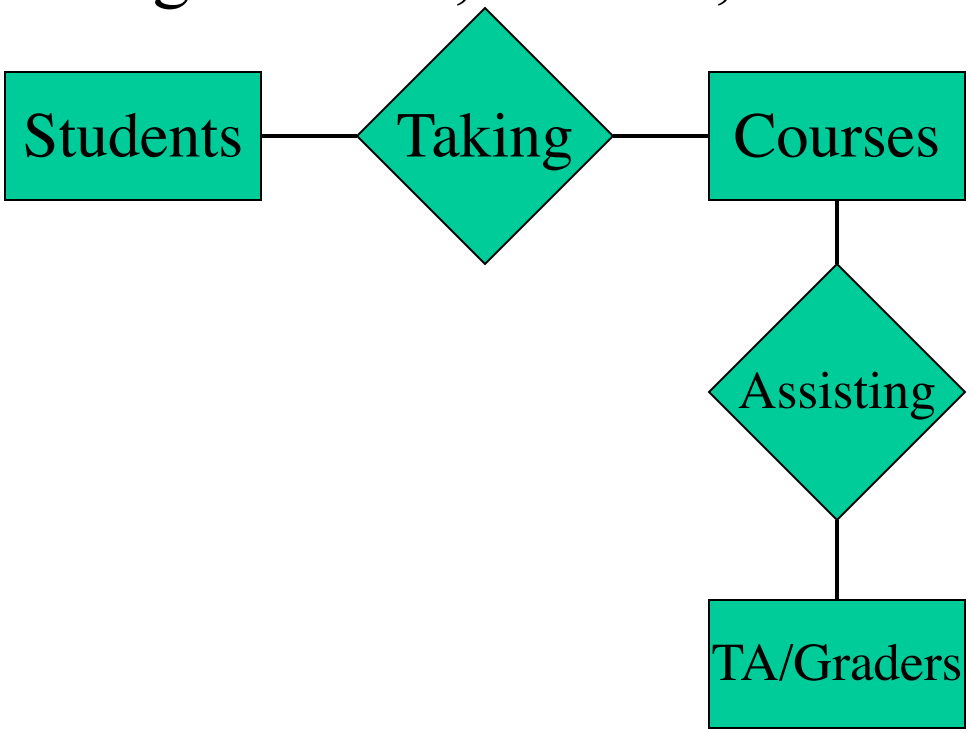
Rest of the lecture discusses HOW THE ER MODEL CAN BE EXTENDED WITH

- **Set-subset** relationships
- **Generalization** Hierarchies
- how we can impose **further notation** on them

# Multi-way Relationships

Usually binary relationships (connecting two E.T.) suffice.

- However, there are some cases where three or more E.T. must be connected by one relationship.
- Example: relationship among students, courses, TA's (and graders).

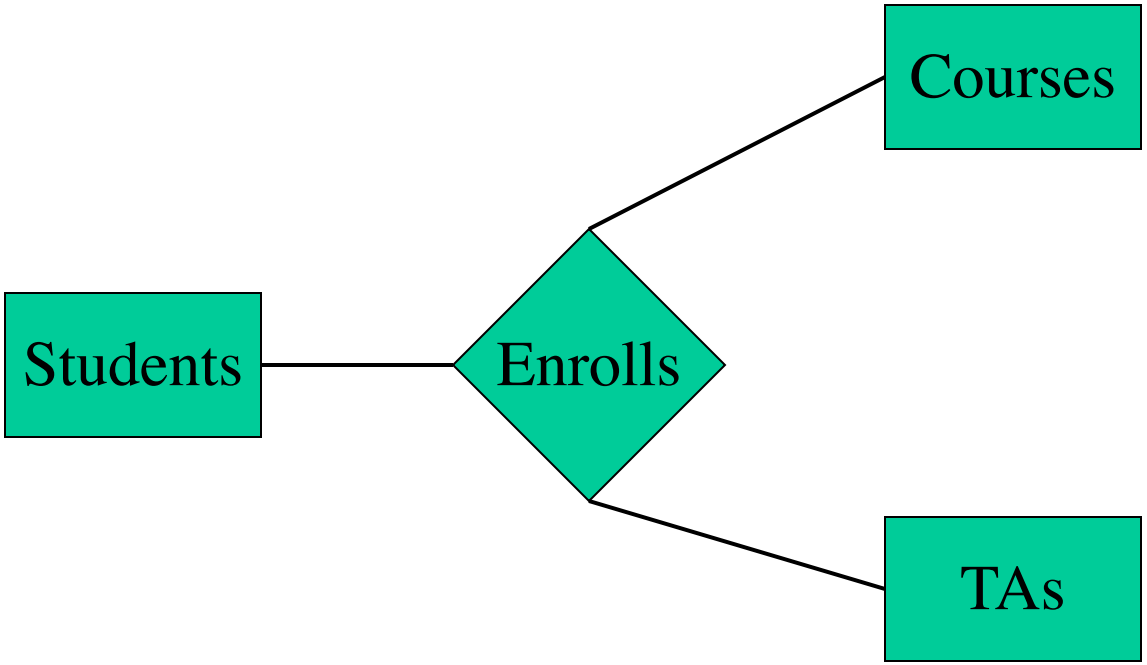


# Example

- This binary relationship OK if each TA (or grader) is a TA of all students. Connection student-TA is *only* via the course.
- But what if students were divided into sections, each headed by a TA?

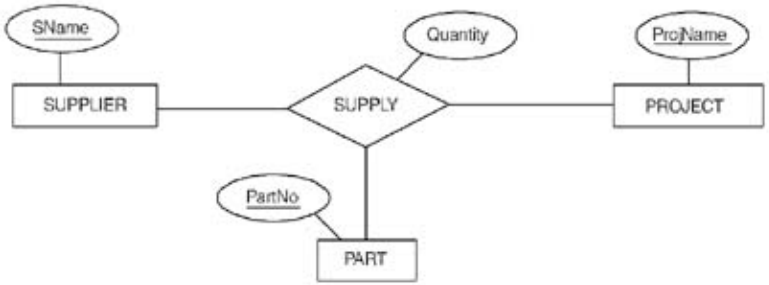
# Example

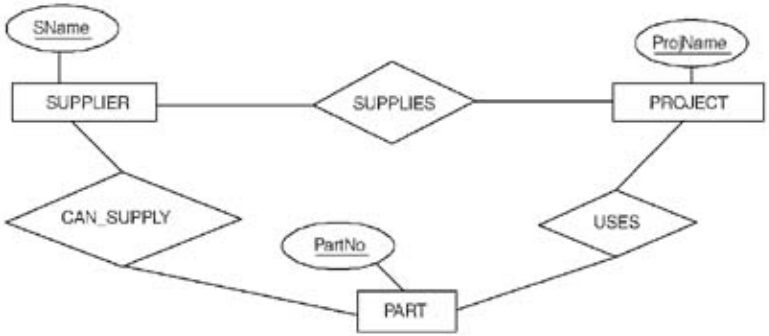
- This binary relationship OK if each TA (or grader) is a TA of all students. Connection student-TA is *only* via the course.
- But what if students were divided into sections, each headed by a TA?
  - Then, a student in CS 6360 would be related to only one of the TA's for CS6360. Which one?
- Need a 3-way relationship to tell.

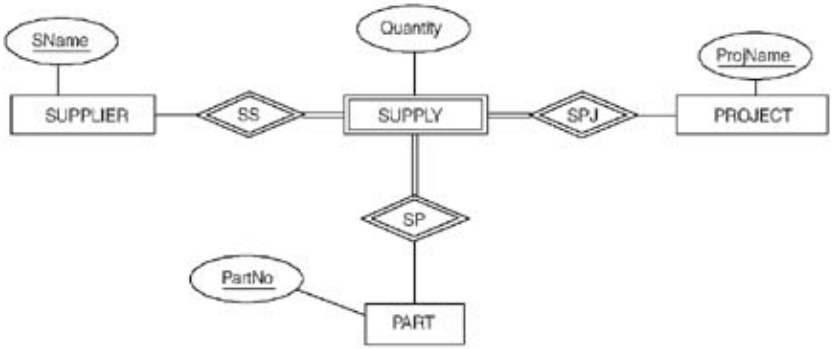



| Students | Courses | TAs |
|---|---|---|
| Ann | CS6360 | Jan |
| Sue | CS6360 | Pat |
| Bob | CS6360 | Jan |
| Linda | CS6360 | Mike |
| … | … | … |

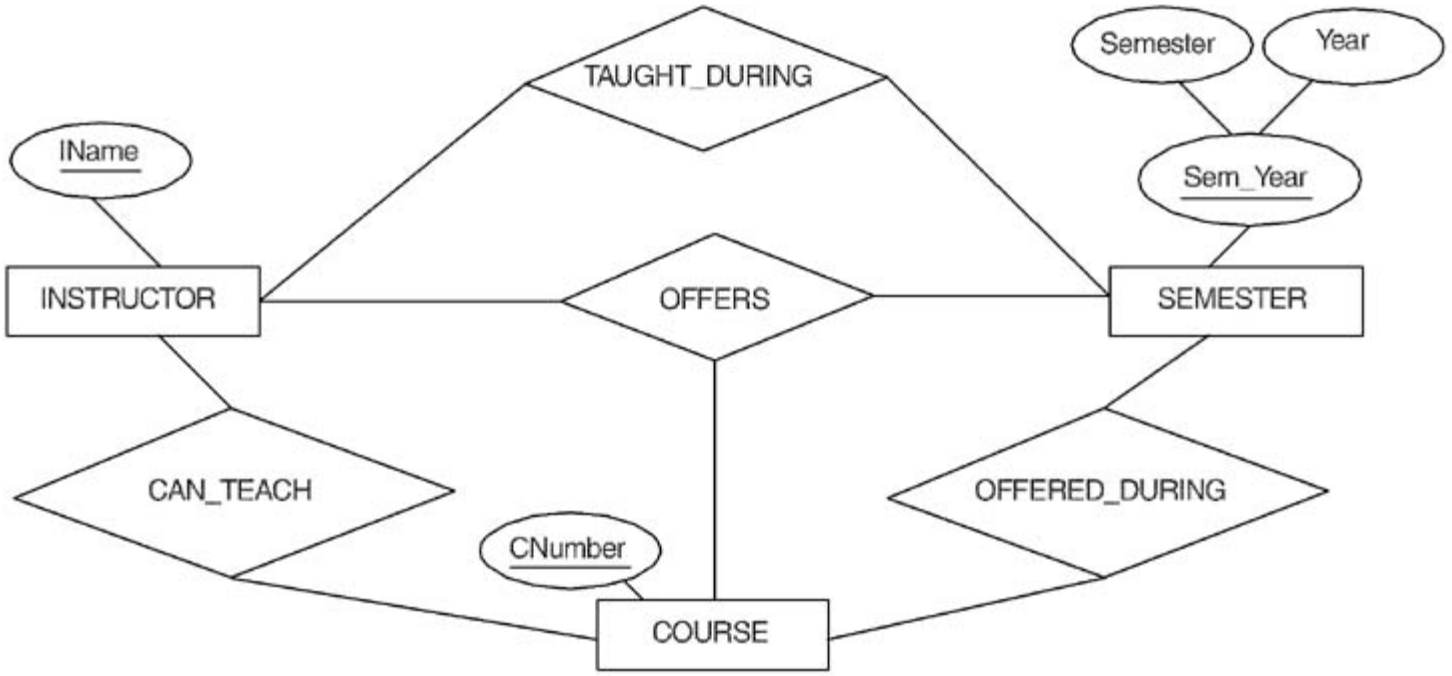
# TERNARY RELATIONSHIPS



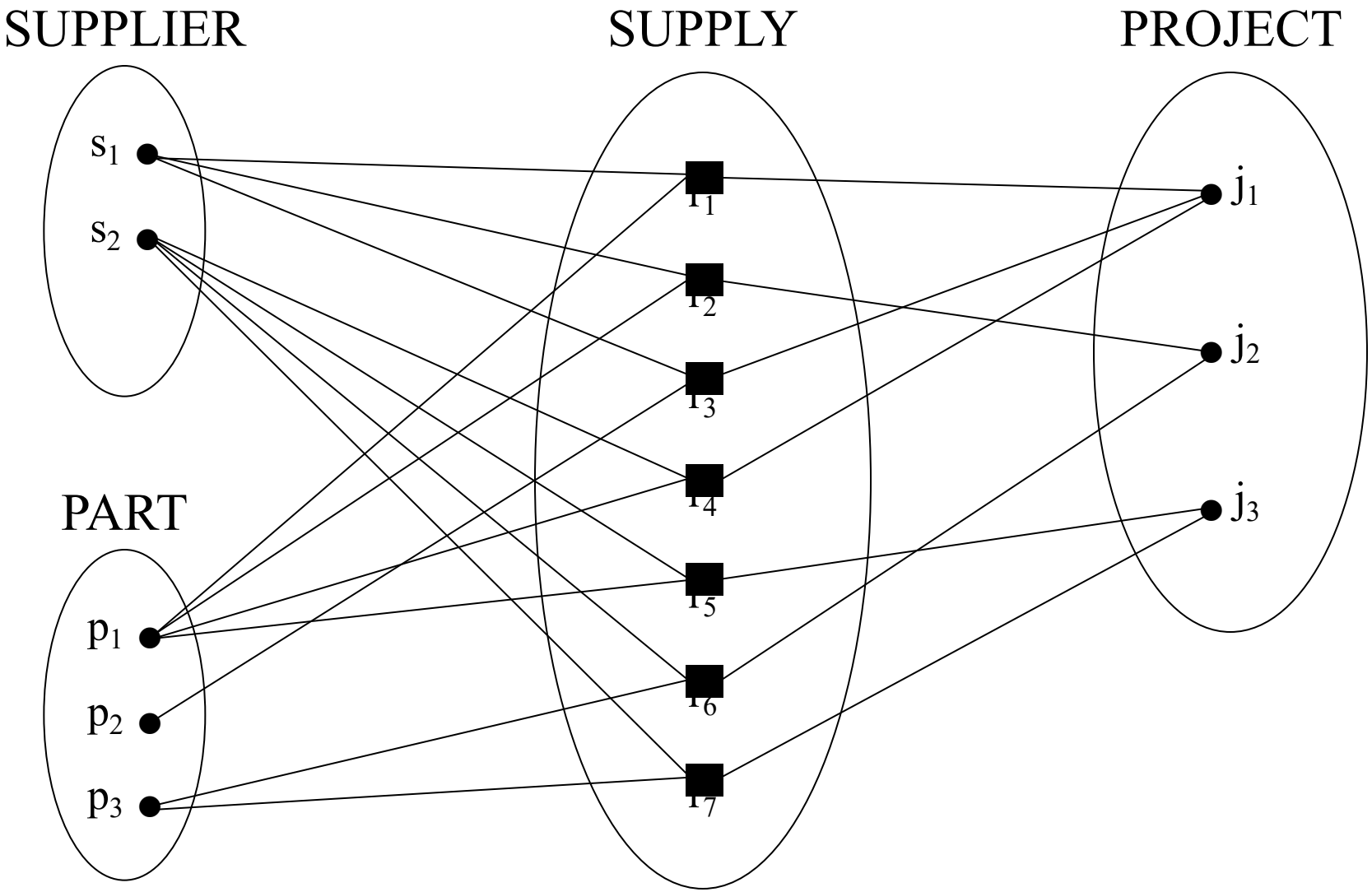



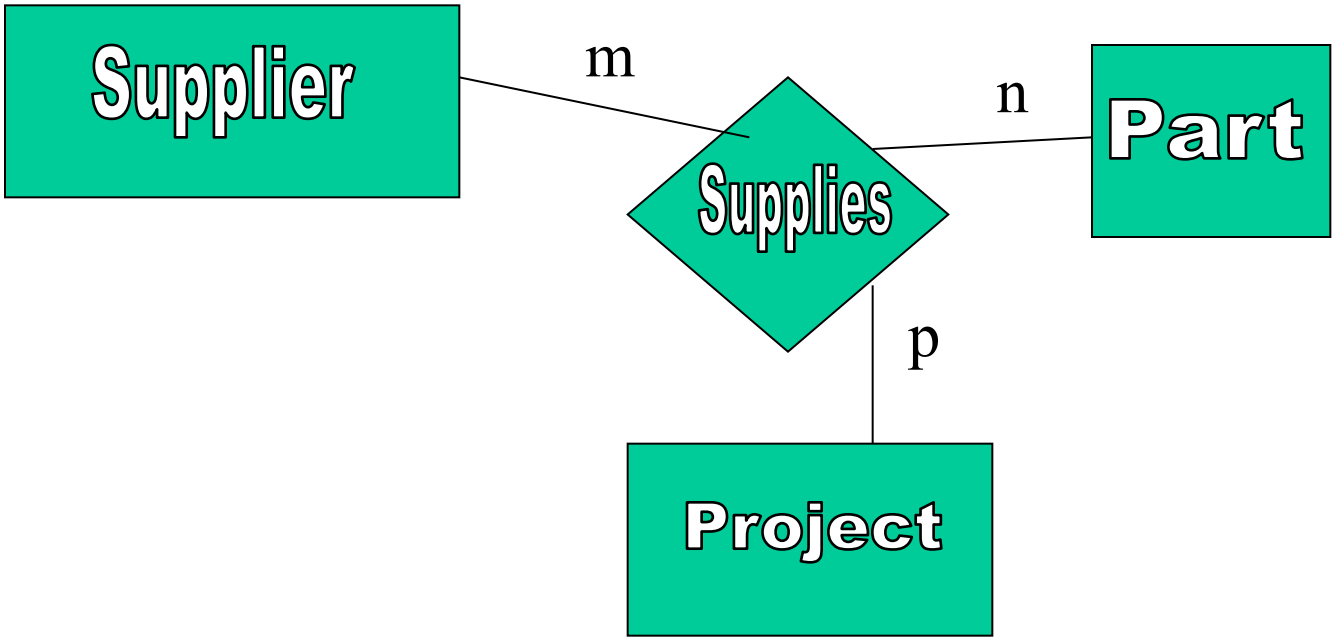
# TERNARY VS. BINARY RELATIONSHIPS



# TERNARY RELATIONSHIP-
# Instance Diagram

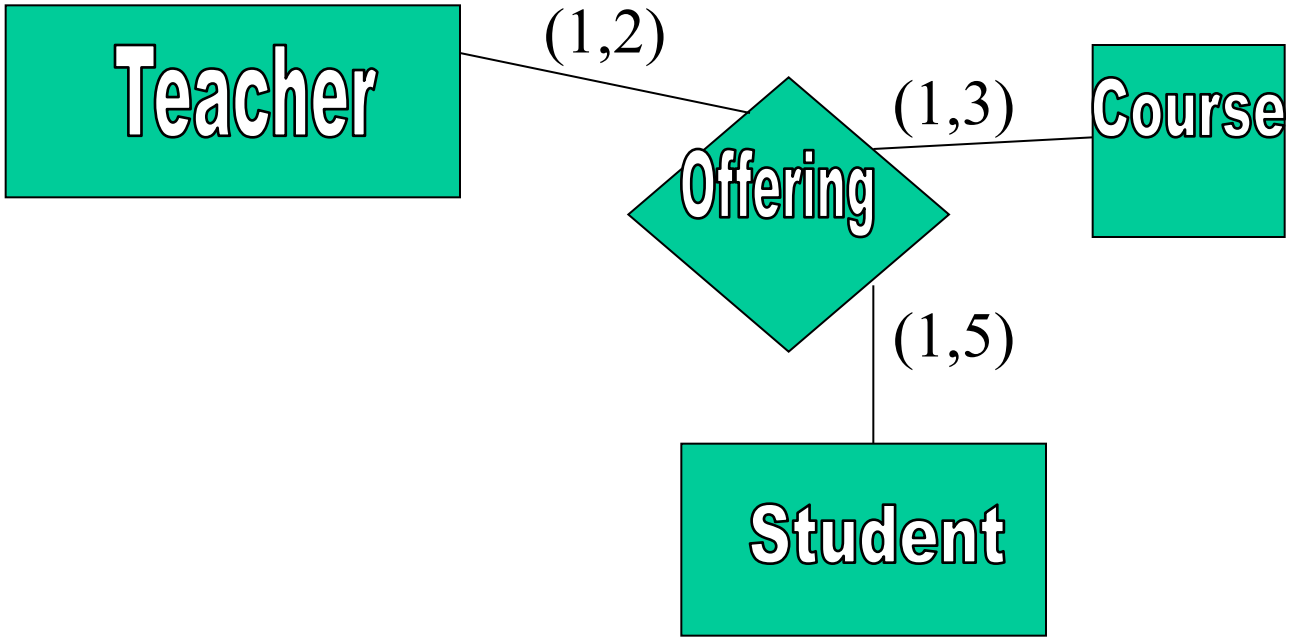


# Problem with constraints on higher order relationship types




What does it mean to put m:n:p on the three arms of the relationship ? It is essentially meaningless.

# The (min,max) notation for higher order relationship type constraints

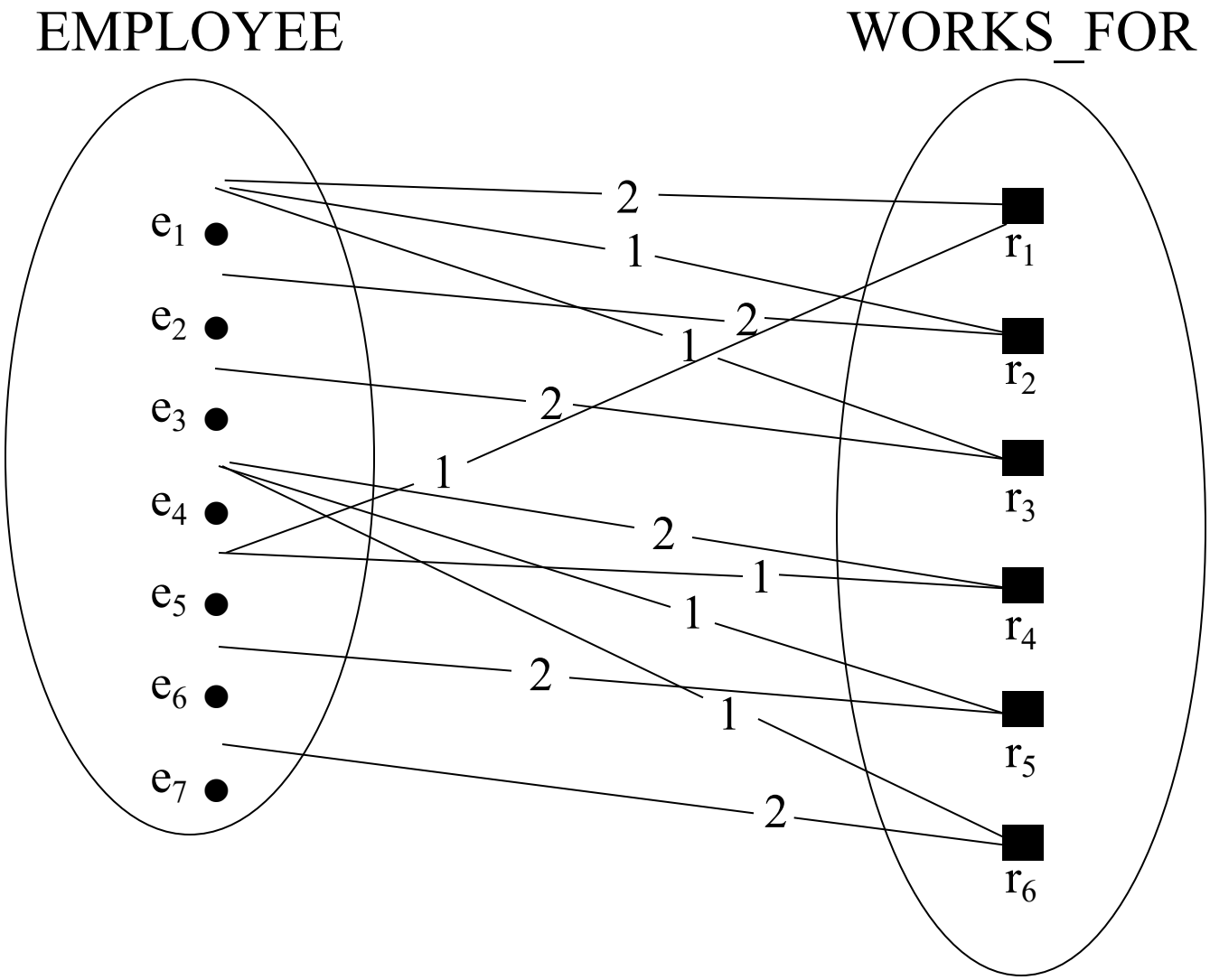



A Teacher can offer min 1 and max 2 Offerings

A Course may have 1 to 3 Offerings

A Student may enroll in from 1 to 5 Offerings

# RECURSIVE RELATIONSHIP SUPERVISION



# Some of the Currently Available Automated Database Design Tools

| COMPANY | TOOL | FUNCTIONALITY |
|---|---|---|
| Embarcadero Technologies | ER Studio | Database Modeling in ER and IDEF1X |
| | DB Artisan | Database administration and space and security management |
| Oracle | Developer 2000 and Designer 2000 | Database modeling, application development |
| Popkin Software | System Architect 2001 | Data modeling, object modeling, process modeling, structured analysis/design |
| Platinum Technology | Platinum Enterprice Modeling Suite: Erwin, BPWin, Paradigm Plus | Data, process, and business component modeling |
| Persistence Inc. | Pwertier | Mapping from O-O to relational model |
| Rational | Rational Rose | Modeling in UML and application generation in C++ and JAVA |
| Rogue Ware | RW Metro | Mapping from O-O to relational model |
| Resolution Ltd. | Xcase | Conceptual modeling up to code maintenance |
| Sybase | Enterprise Application Suite | Data modeling, business logic modeling |
| Visio | Visio Enterprise | Data modeling, design and reengineering Visual Basic and Visual C++ |

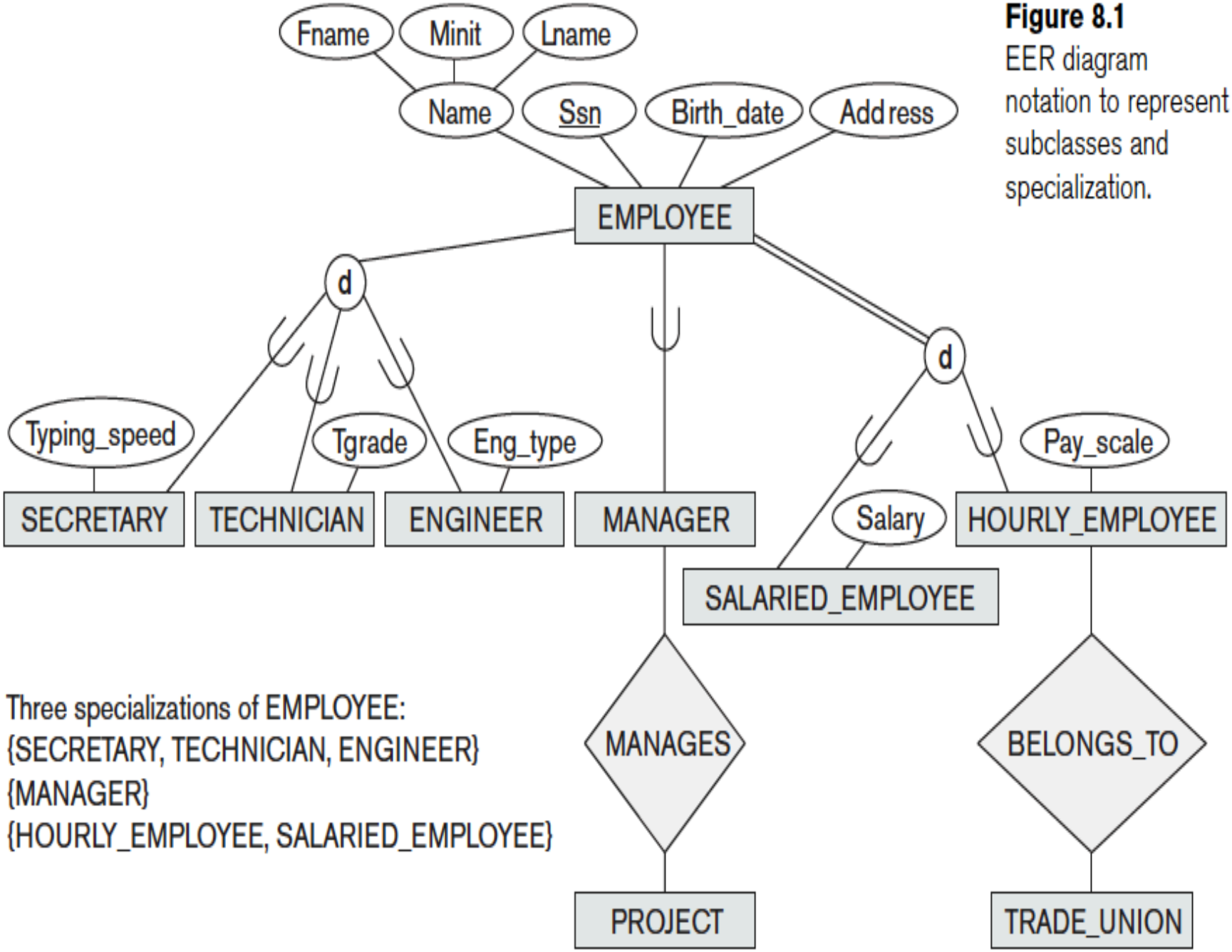



**Figure 8.1**
EER diagram notation to represent subclasses and specialization.

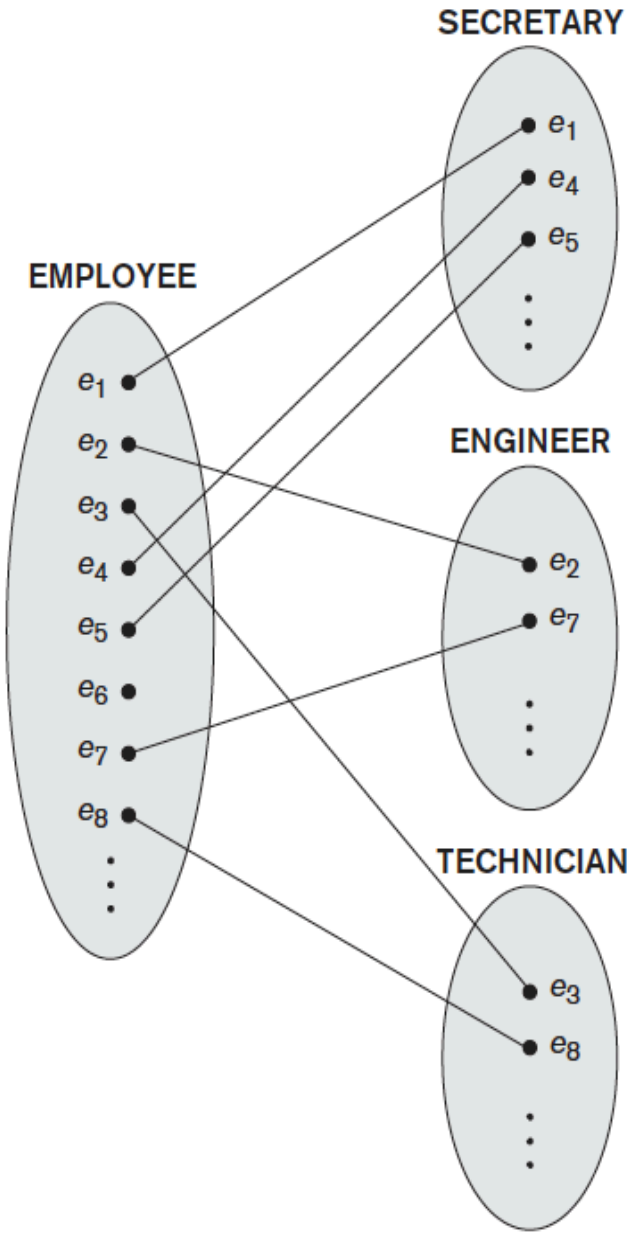



**Figure 8.2**
Instances of a specialization.

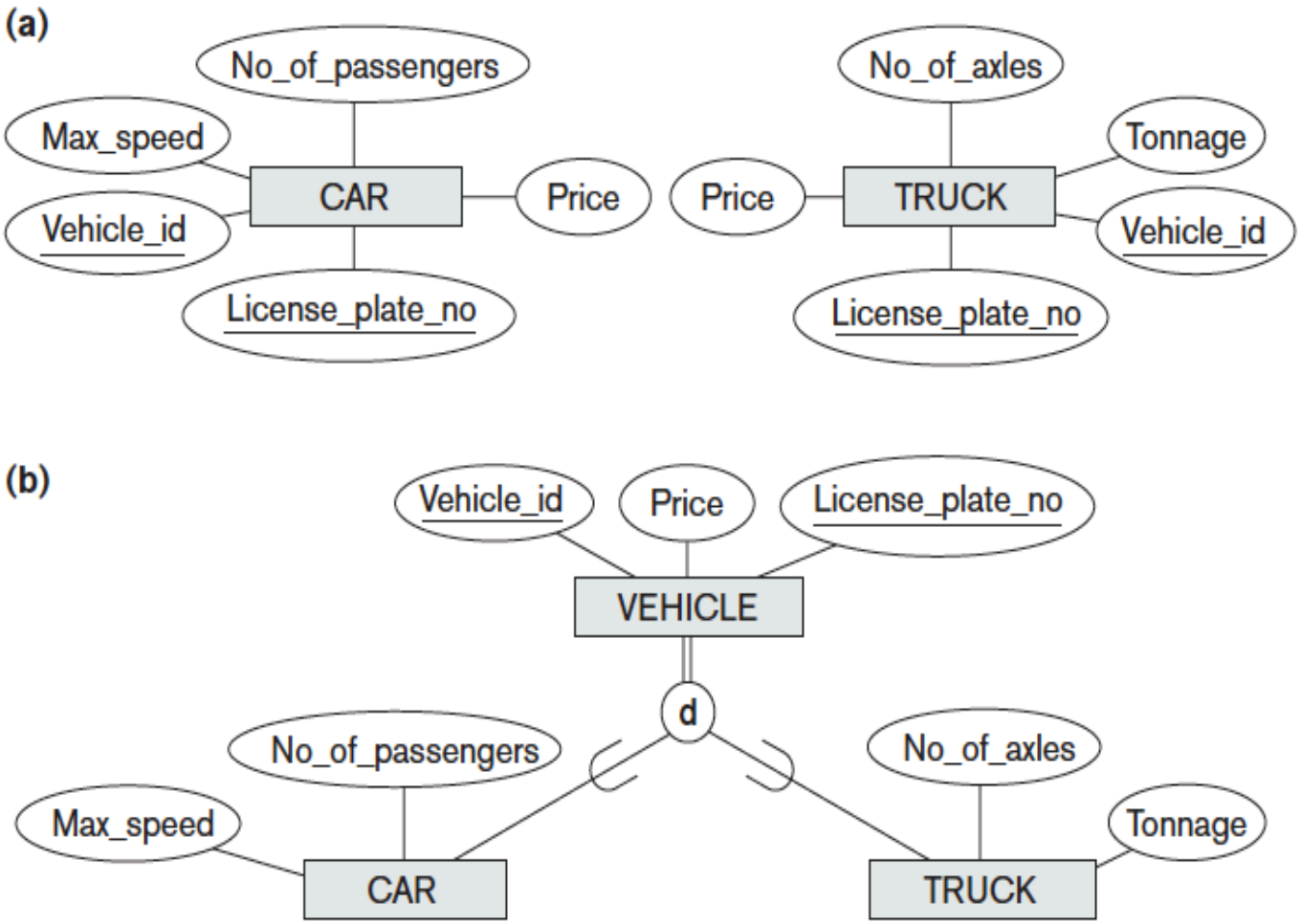
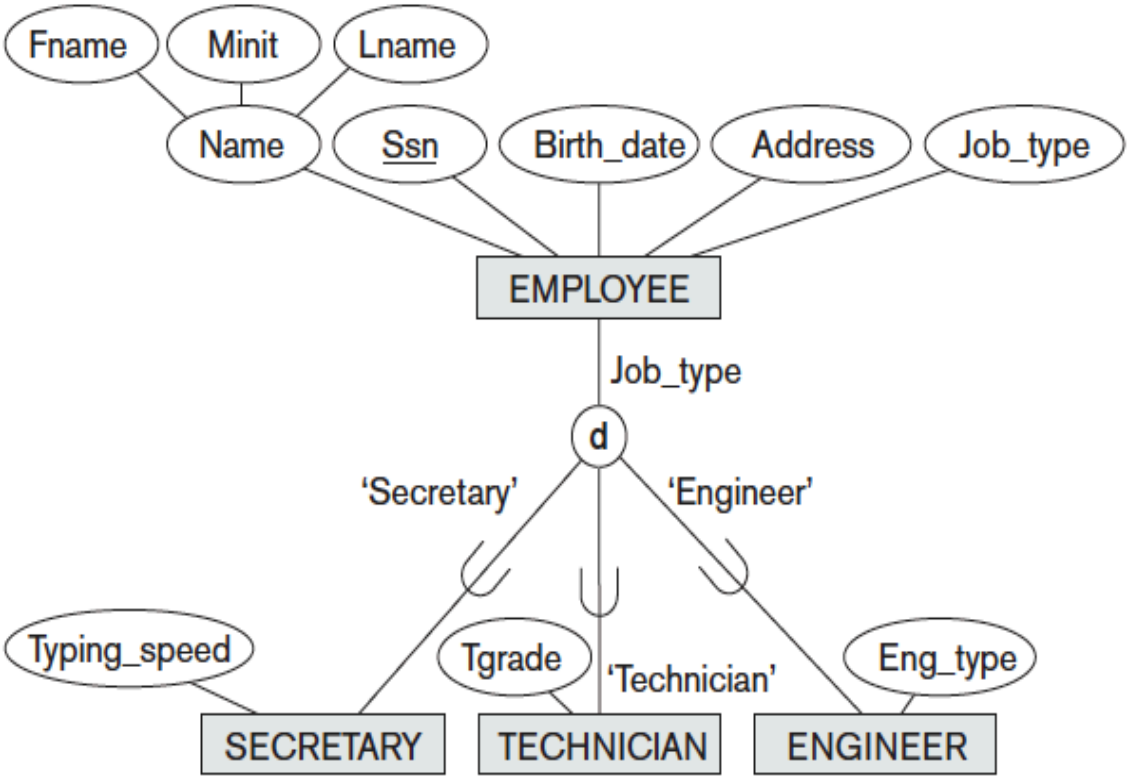



**Figure 8.3**
Generalization. (a) Two entity types, CAR and TRUCK. (b) Generalizing CAR and TRUCK into the superclass VEHICLE.

**Figure 8.4**
EER diagram notation for an attribute-defined specialization on Job_type.



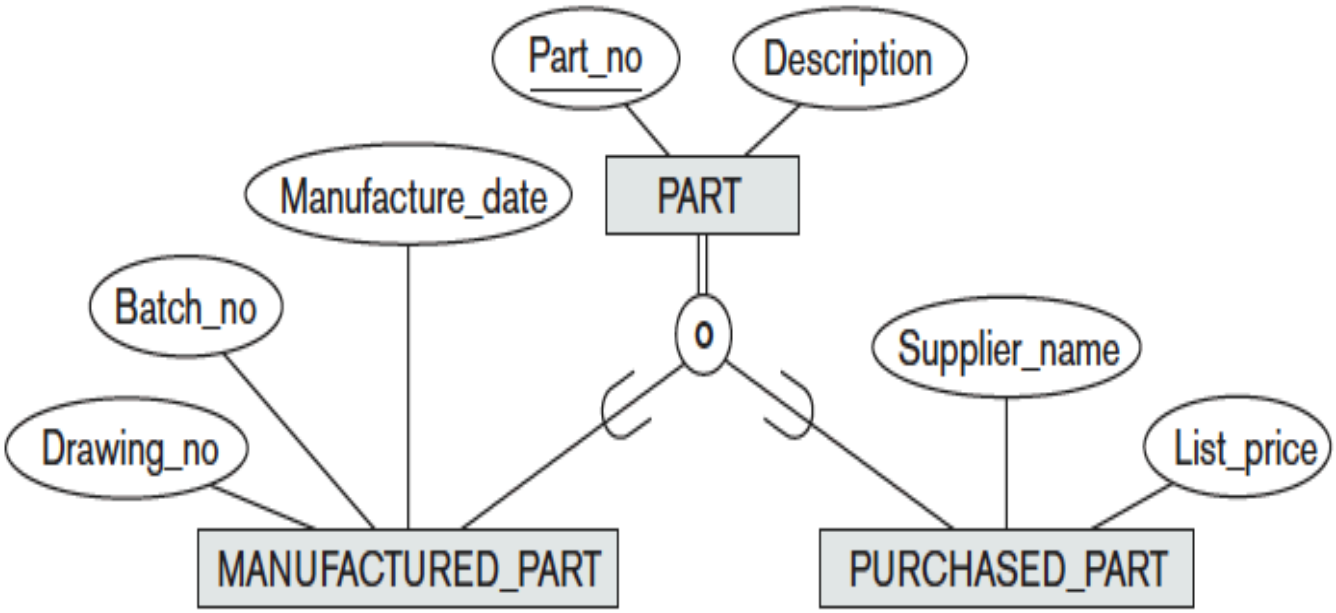
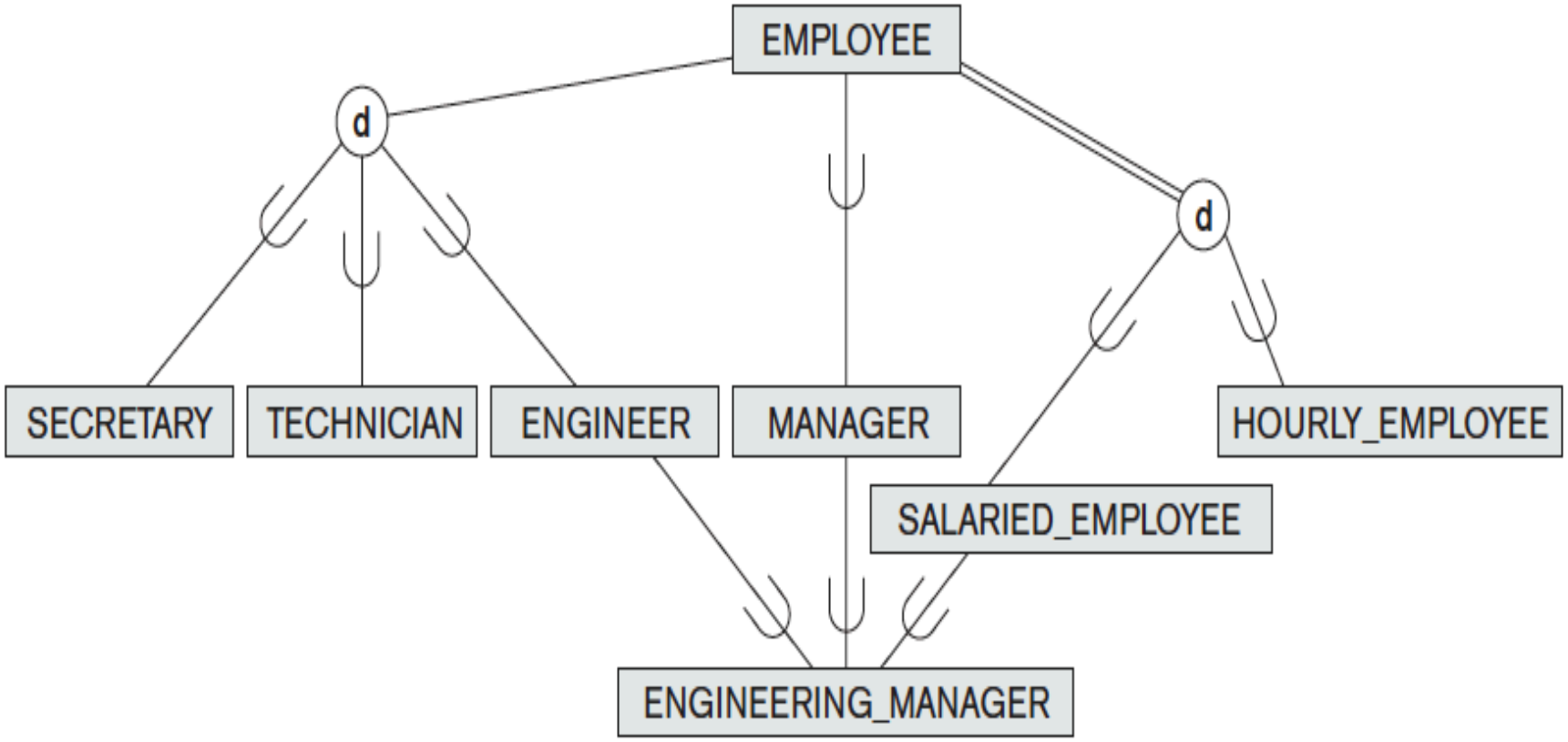



**Figure 8.5**
EER diagram notation for an overlapping (nondisjoint) specialization.

**Figure 8.6**
A specialization lattice with shared subclass ENGINEERING_MANAGER.



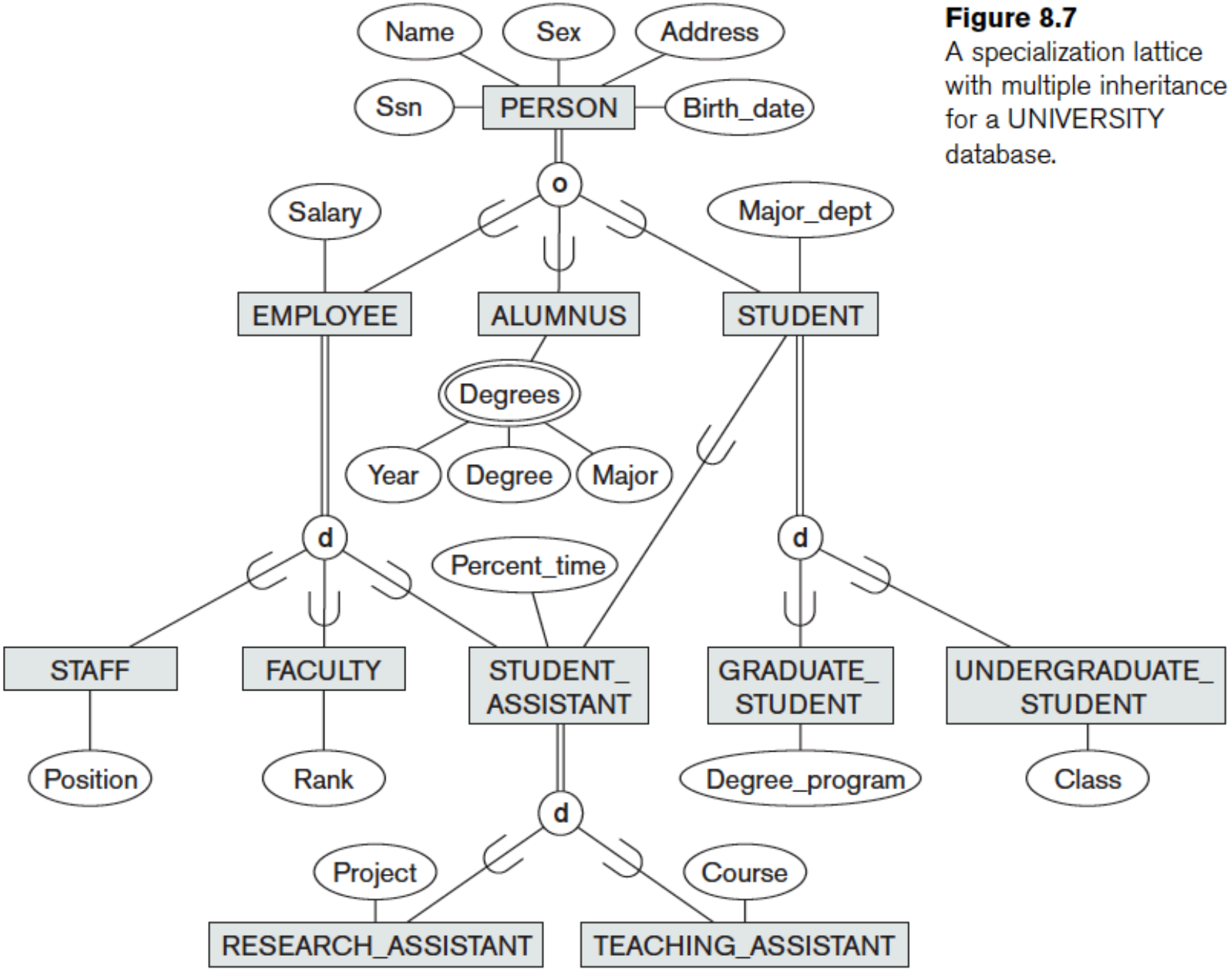



**Figure 8.7**
A specialization lattice with multiple inheritance for a UNIVERSITY database.

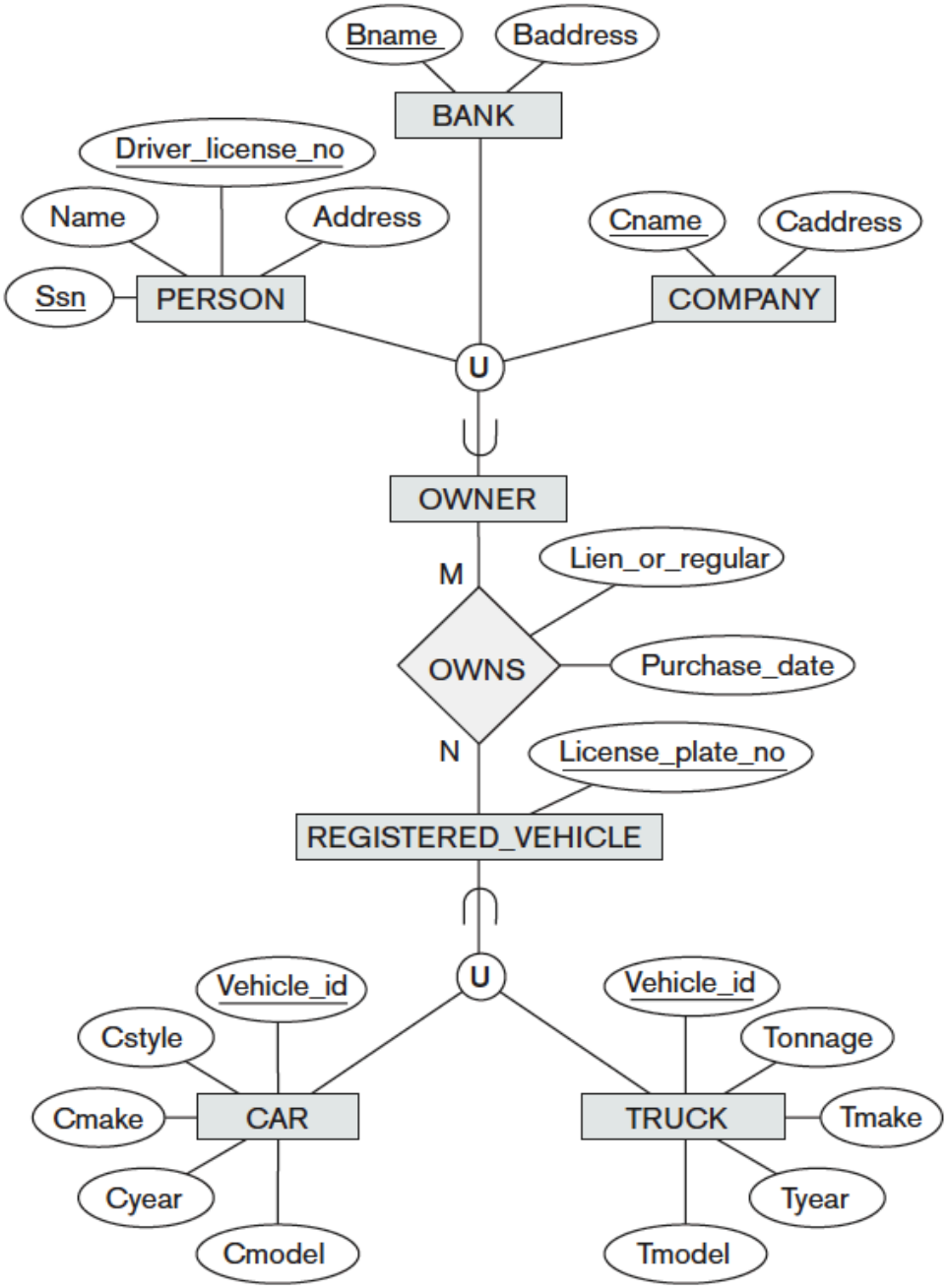



**Figure 8.8**
Two categories (union types): OWNER and REGISTERED_VEHICLE.