HALMSTAD UNIVERSITY

DS4001 Databases (7.5 credits)

Lecture 5 – Entity-Relationship Diagrams

Yuantao Fan
yuantao.fan@hh.se

Halmstad University

Overview

- Schema
- Database design flow
- Entity-Relationship diagram
  - Domain description
  - Entities, multiple entities, weak entities
  - Attributes of an entity, or a relationship
  - Relationships
  - Multiplicity
  - Relations in ER-diagrams
- Examples

Designing Databases with Entity-Relationship (ER) Diagram

- We have learnt how to use DDL to implement a databse design
  - Given tables
  - Create a database via SQL
  - Query for data

- Knowing the concept of constraints
  - Primary keys, prevent duplicate values
  - Foreign keys (reference constraints)

- Design a database based on a domain description

Schemas

- A database schema is a collection of relation:

```
Teaches(tid, cid, hours)
    tid -> Teacher.tid
    cid -> Course.cid
    hours > 0
```

- You can write SQL code for implementing the design

```
CREATE TABLE Teaches(
    tid tinyint(4) NOT NULL,
    cid tinyint(4) NOT NULL,
    hours INT,
    CHECK (hours > 0),
    FOREIGN KEY (tid) REFERENCES Teacher(tid),
    FOREIGN KEY (cid) REFERENCES Course(cid)
);
```

tid	cid	hours
11	1	80
11	2	100
22	4	50
33	4	50
44	3	100

Schema

- Why the following databse design is bad?

Schedule

tid	cid	c_name	date	time	room	nn_seats
33	4	Databases	2030-01-23	10:15 - 12:00	D415	50
33	4	Databases	2030-01-24	08:15 - 10:00	D415	50
11	3	Mathematics	2030-01-24	13:15 - 15:00	D208	30
11	3	Mathematics	2030-01-25	13:15 - 15:00	D415	50

Schema

- Why the following databse design is bad?

  Schedule

tid	cid	c_name	date	time	room	nn_seats
33	4	Databases	2030-01-23	10:15 - 12:00	D415	50
33	4	Databases	2030-01-24	08:15 - 10:00	D415	50
11	3	Mathematics	2030-01-24	13:15 - 15:00	D208	30
11	3	Mathematics	2030-01-25	13:15 - 15:00	D415	50

- Redundancy
  - Duplicates in the number of seats for room D415
- Update anomaly & delete anomaly
  - Change nn_seats in one row but not others
  - nn_seats information is gone if all bookings of D415 is removed

Decomposing the table

Schedule

tid	cid	date	time	room
33	4	2030-01-23	10:15 - 12:00	D415
33	4	2030-01-24	08:15 - 10:00	D415
11	3	2030-01-24	13:15 - 15:00	D208
11	3	2030-01-25	13:15 - 15:00	D415

Nice and simple when tables are small

Room

room	nn_seats
D415	50
D208	30

Course

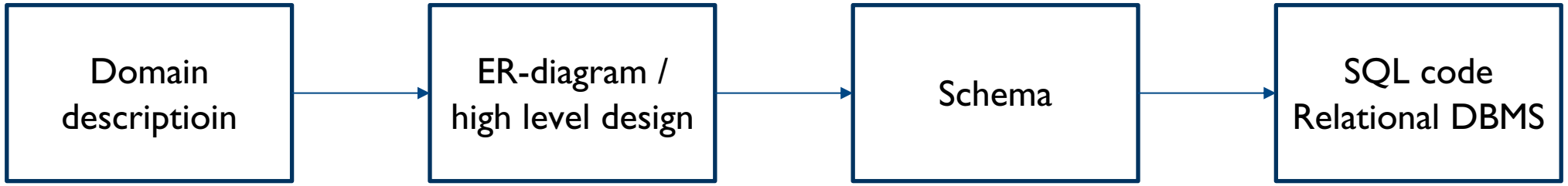
cid	course_name
3	Discrete Mathematics
4	Databases

Domain descriptions

- The domain description of a database is an informal description of everything that a database should contain
- Provided by the stakeholders (experts in this domain), or your clients
- Written in natural language
  - Depending on it's structure, maybe difficult to generate SQL code without error
- Has certain abstraction
  - Ambigous on some details
- Development of the databases is likely to be iterative
  - Contious meetings between developers and the clients

Modelling Domains

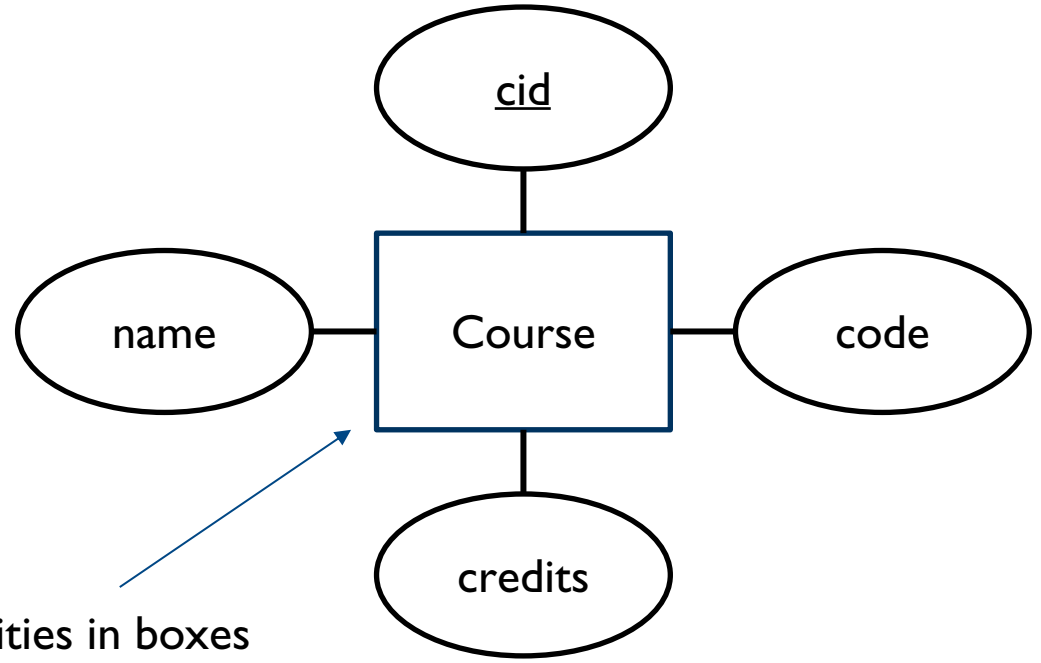
- Model the domain for developing databaseas
- A model contains formal, well-defined definitions
- Given a description on the domain
  - Classrooms can be booked for courses on working days during daytime
- How to model a domain?
- Problems
  - Directly write SQL code for implmenting databases according the description is prone to error
  - Hard to present it efficienty when the tables are large



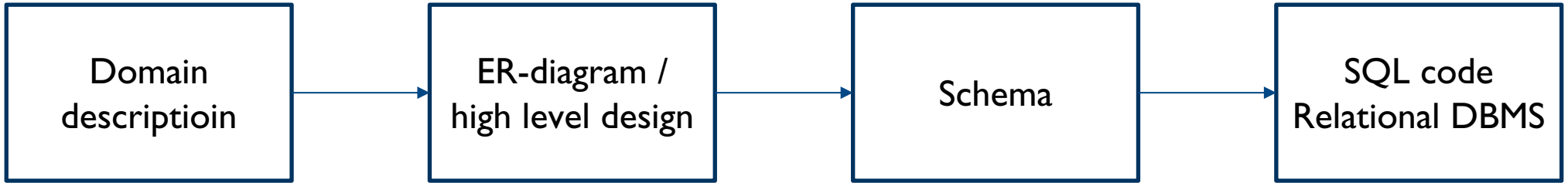
The Entity-Relationship (ER) Model

- An ER model describes interrelated things of interest in a specific domain of knowledge.
  - Entities (including its attributes)
  - Relationships between different entities
- Entities are concepts (or things) from the domain
  - Teacher, Courses, Students etc.
  - Attributes are properties of the entities
- Relationship connect entities
  - Teachers teach courses etc.

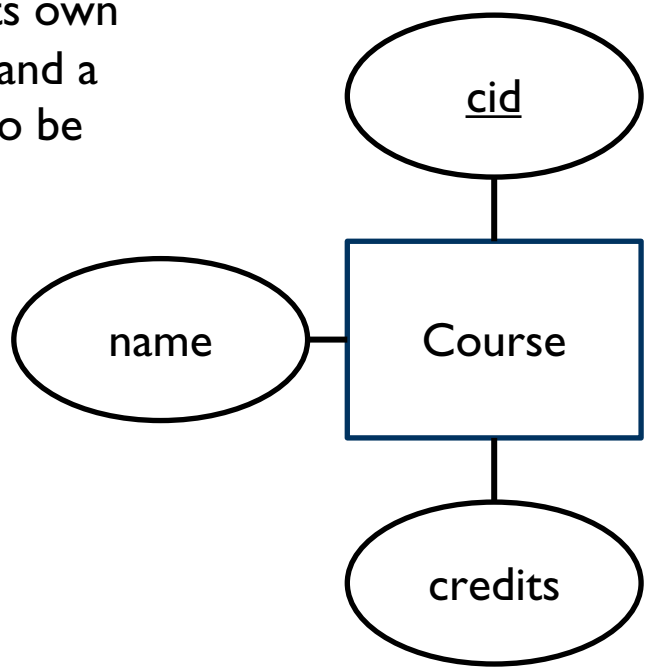
An ER diagram



From ER model to relational schema



"Each Course has its own name, a unique ID, and a number of credits to be acquired."

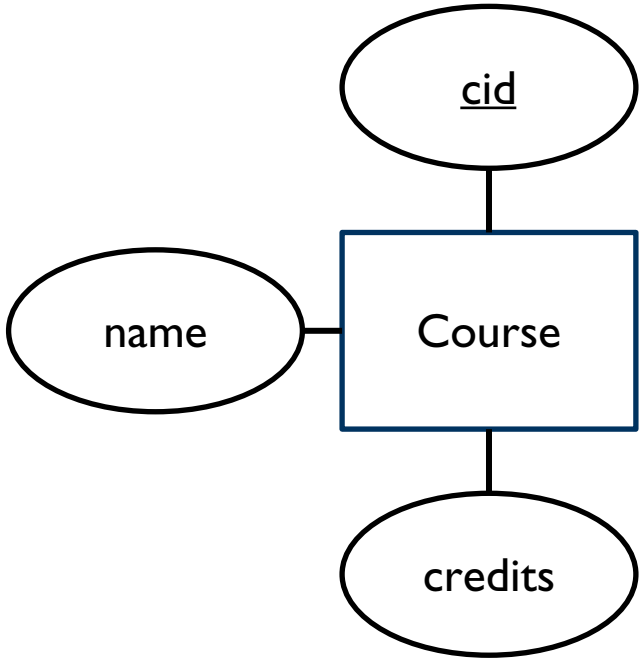


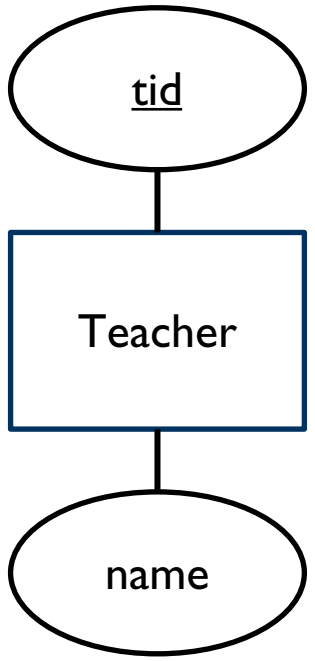
Courses(cid, name, credits)

```
CREATE TABLE Course(
    cid tinyint(4),
    name TEXT ,
    credits tinyint(4),
    PRIMARY KEY (cid)
);
```

Multiple Entities

- Entities are named singular, while relations are in plural



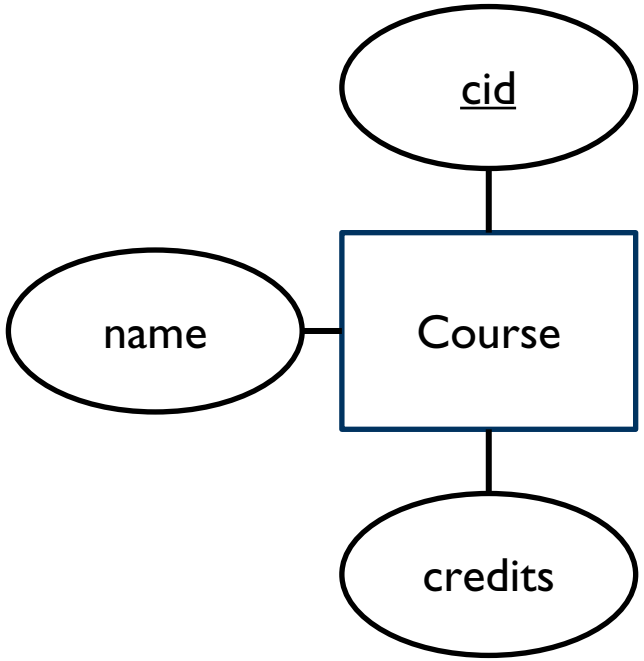


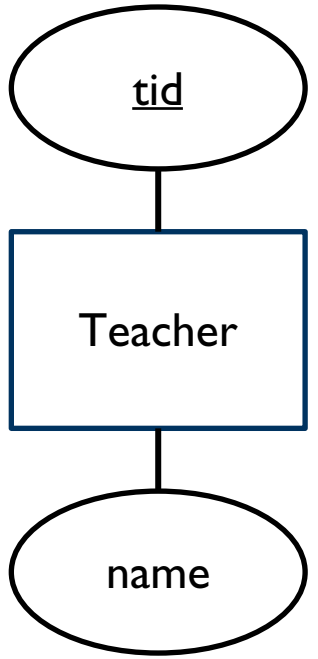
Courses(<u>cid</u>, name, credits)

Teachers(<u>tid</u>, name)

Relationship between two entities

- It is required to assign teachers onto the course to fulfill tasks
- How can it be implemented?
- Add attributes?





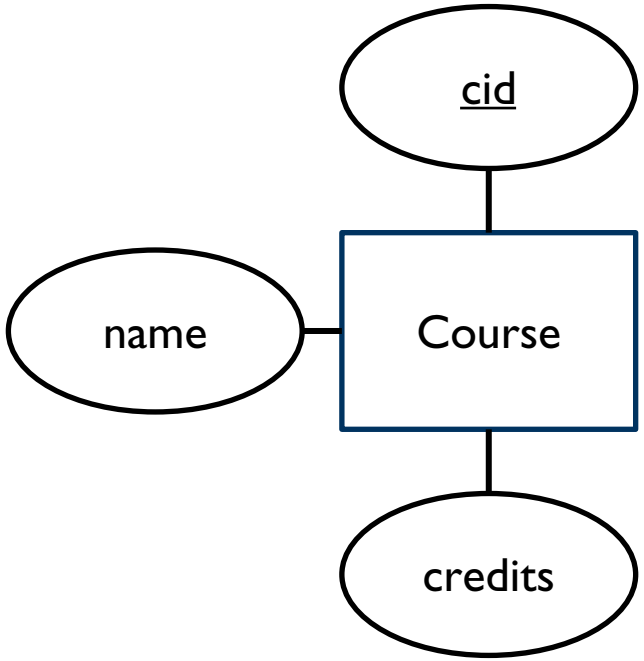
Courses(<u>cid</u>, name, credits)

Teachers(<u>tid</u>, name)

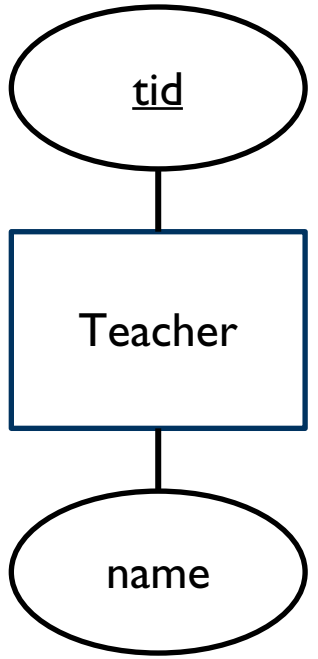
Relationship between two entities

- It is required to assign teachers onto the course to fulfill tasks
- How can it be implemented?
- Add attributes?
- How about a a new relation?
  - A binary relation with (course, teacher) pairs

Courses(<u>cid</u>, name, credits)
Teachers(<u>tid</u>, name)
TaughtBy(<u>course</u>, <u>teacher</u>)
course -> Courses.cid
teacher -> Teachers.tid



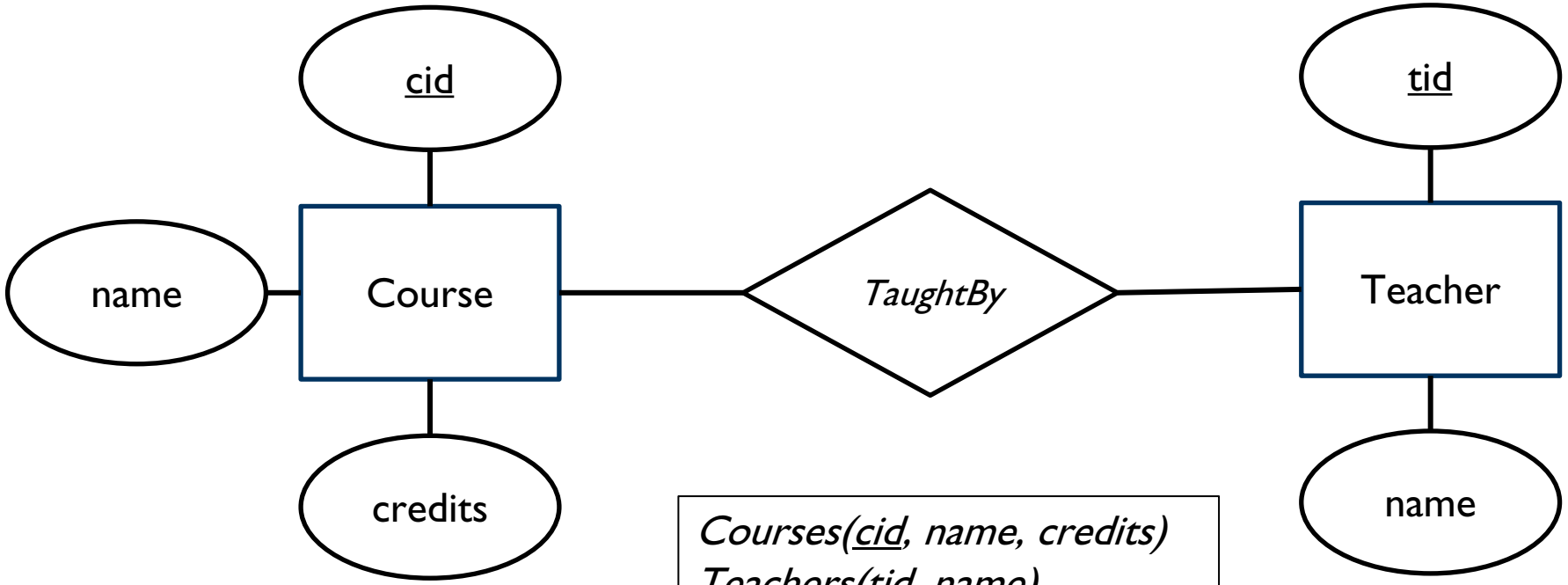
Courses(<u>cid</u>, name, credits)



Teachers(<u>tid</u>, name)

Multiple Entities

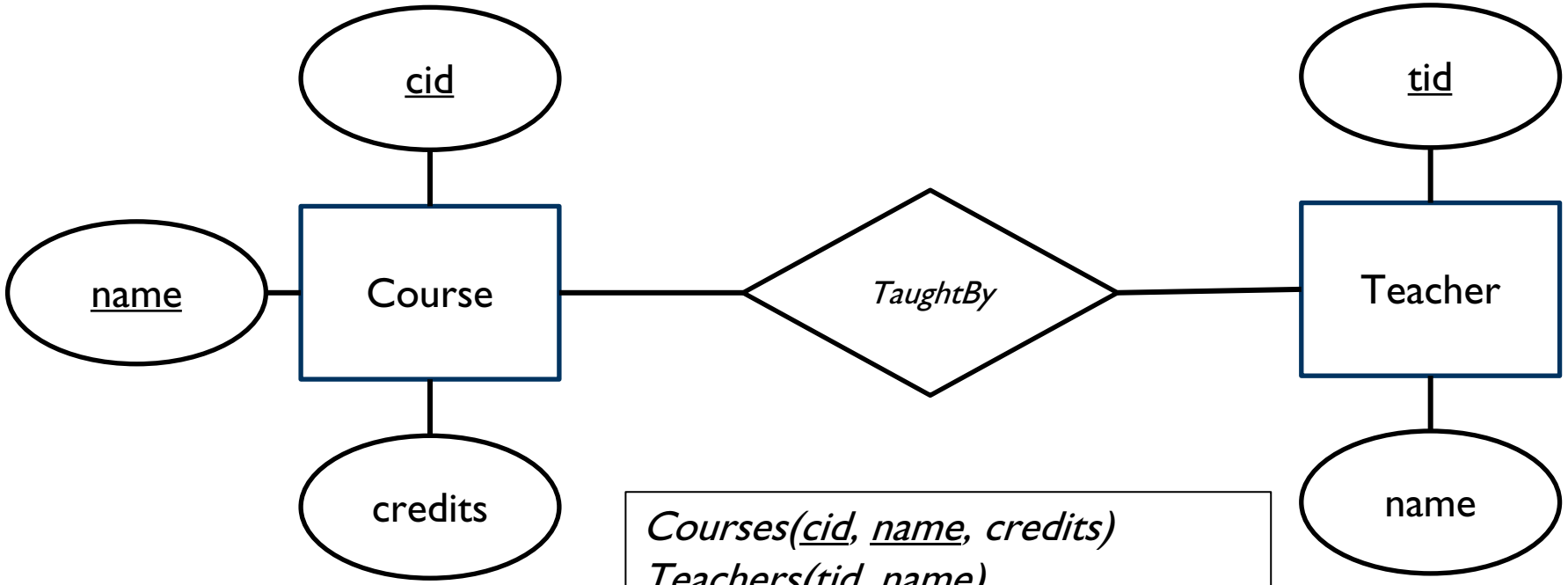
- Relationships in diamond-shpes
- Name describe the relationship



Courses(<u>cid</u>, name, credits)
Teachers(<u>tid</u>, name)
TaughtBy(<u>course</u>, <u>teacher</u>)
course -> Courses.cid
teacher -> Teachers.tid

Compound keys and relationshipes

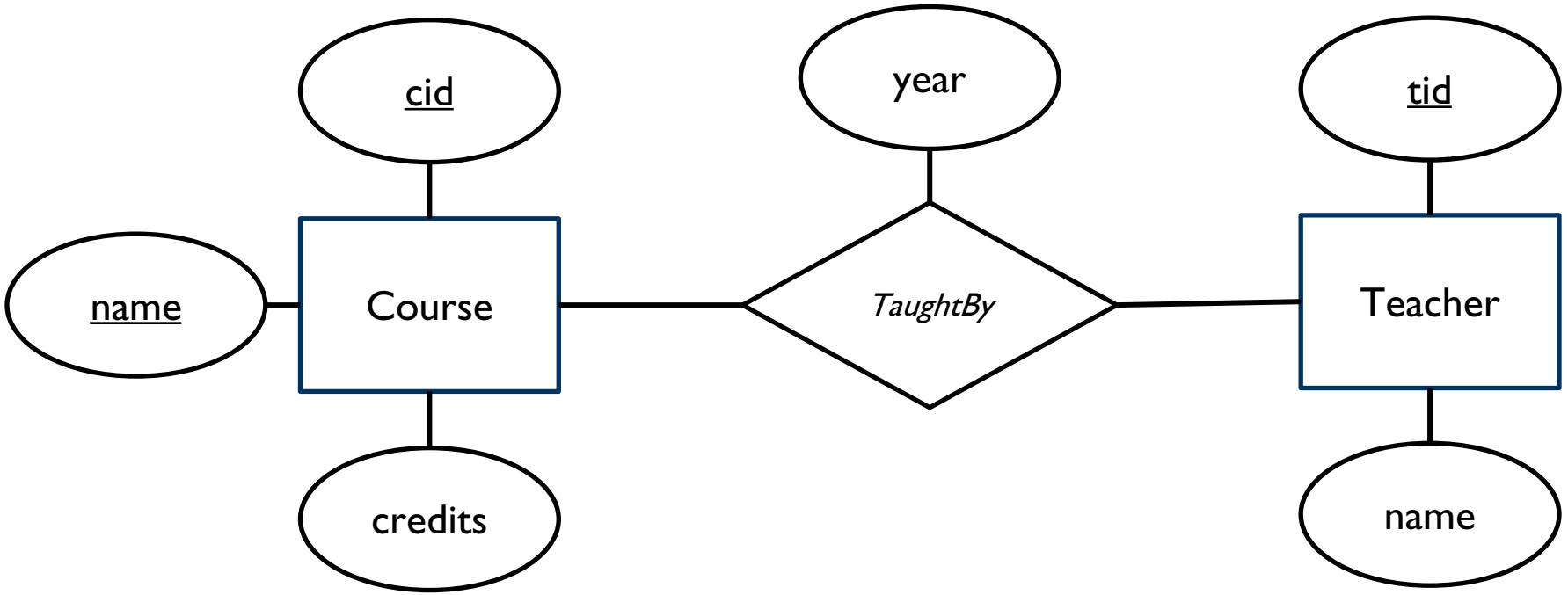
- Include whole key of both relations



Courses(<u>cid</u>, <u>name</u>, credits)
Teachers(<u>tid</u>, name)
TaughtBy(<u>course</u>, <u>c_name</u>, <u>teacher</u>)
course -> Courses.(cid, name)
teacher -> Teachers.tid

Attributes of relationships

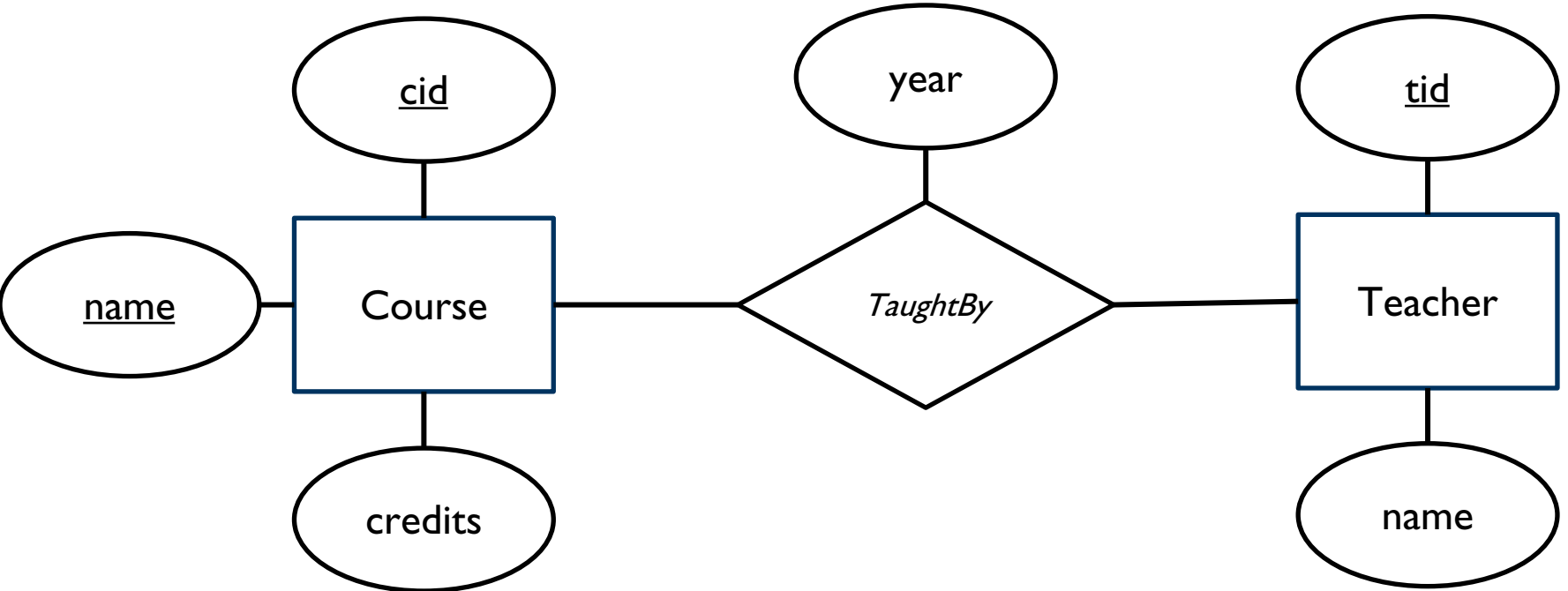
- Specifiy the year of responsible teacher of each course
  - Is year an attribute of course? teacher? or the relationship?



Attributes of relationships

- Specifiy the year of responsible teacher of each course
  - Is year an attribute of course? teacher? or the relationship?
- Note that relationship can never have key attributes
  - Always identified by the related entities
- Identify attributes on relationships in domains
  - A might have a B in/at/for/to a C, where A and C are entities, and z is an attribute

Courses(cid, name, credits)
Teachers(tid, name)
TaughtBy(course, c_name, teacher, year)
course -> Courses.(cid, name)
teacher -> Teachers.tid



e.g. Teacher can be assigned with hours in courses

Degree of Relationships

- The degree of a relationship tupe is the number of participating entity types
- The WORKS_FOR relationship is of degree two - binary
- The SUPPLY relationship is of degree three - ternary

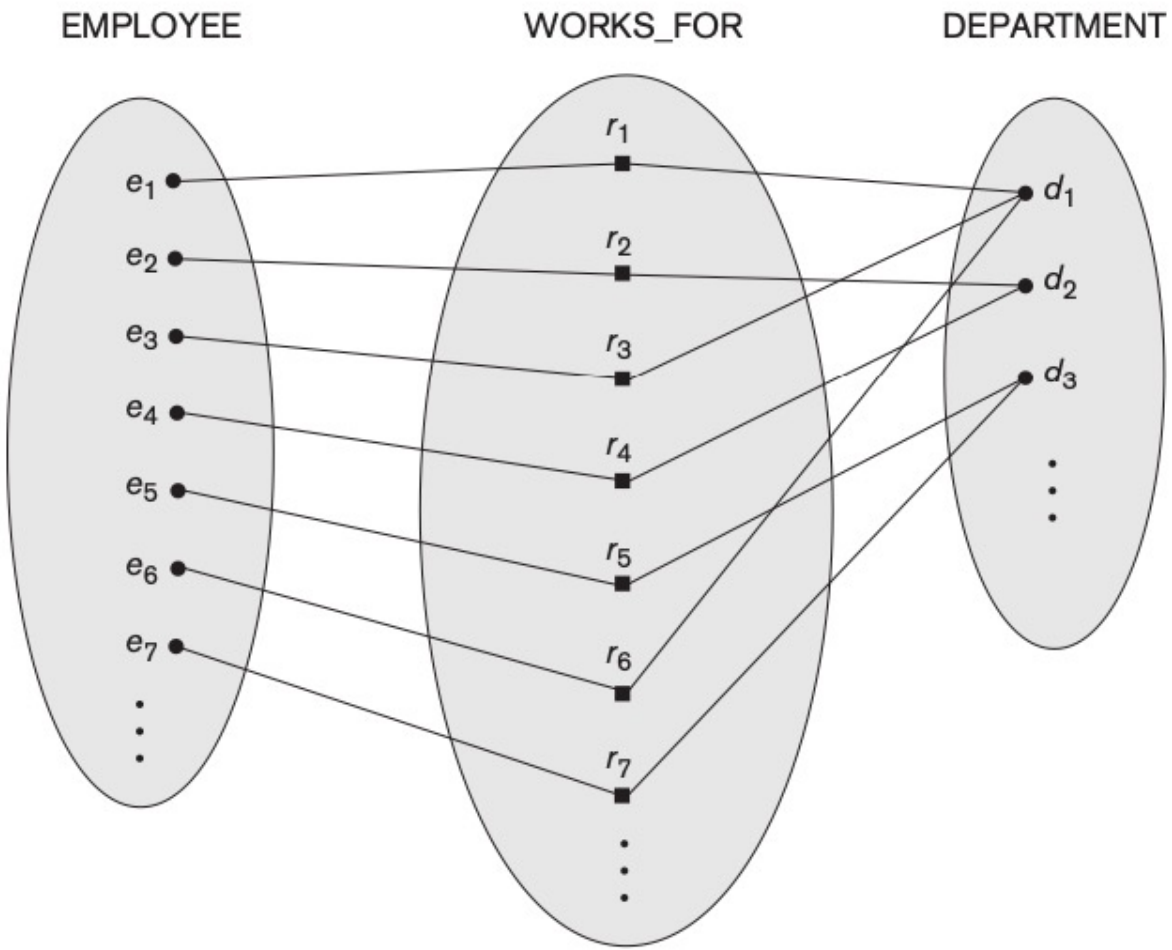
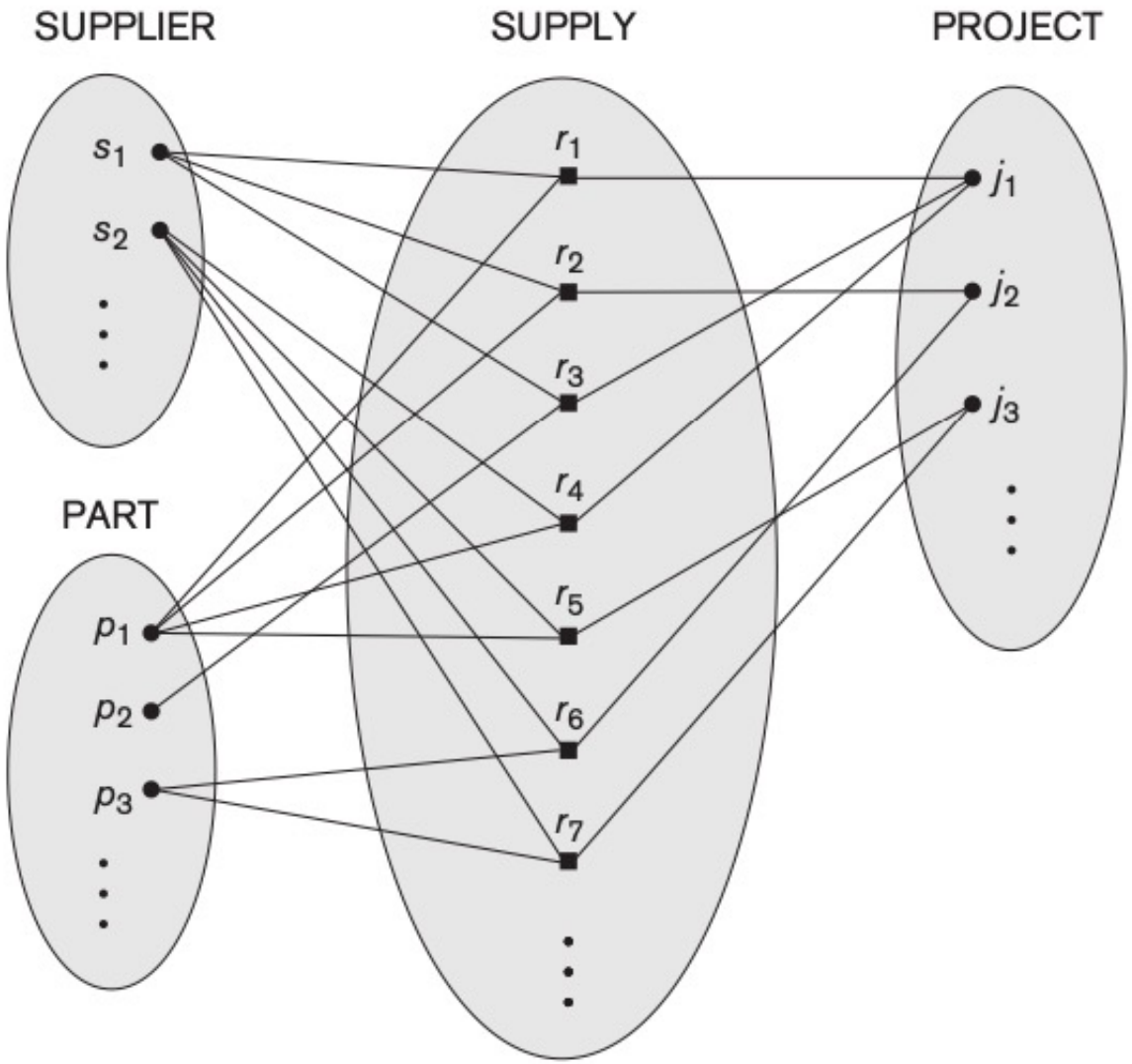


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

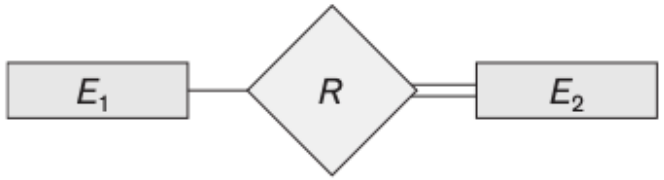
Relationship Degree

- The degree of a relationship tupe is the number of participating entity types
- The WORKS_FOR relationship is of degree two - binary
- The SUPPLY relationship is of degree three - ternary

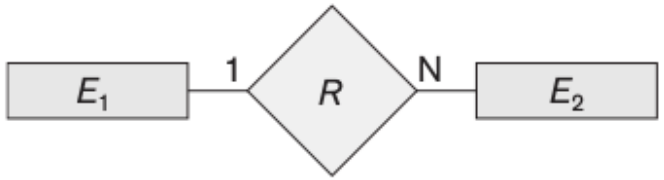


Multiplicity

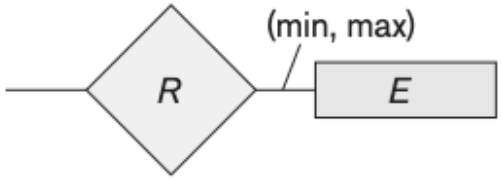
- Example relationship models
  - Each course has a single teacher
  - Each course has at least one teahcer
  - Each teacher has a single course



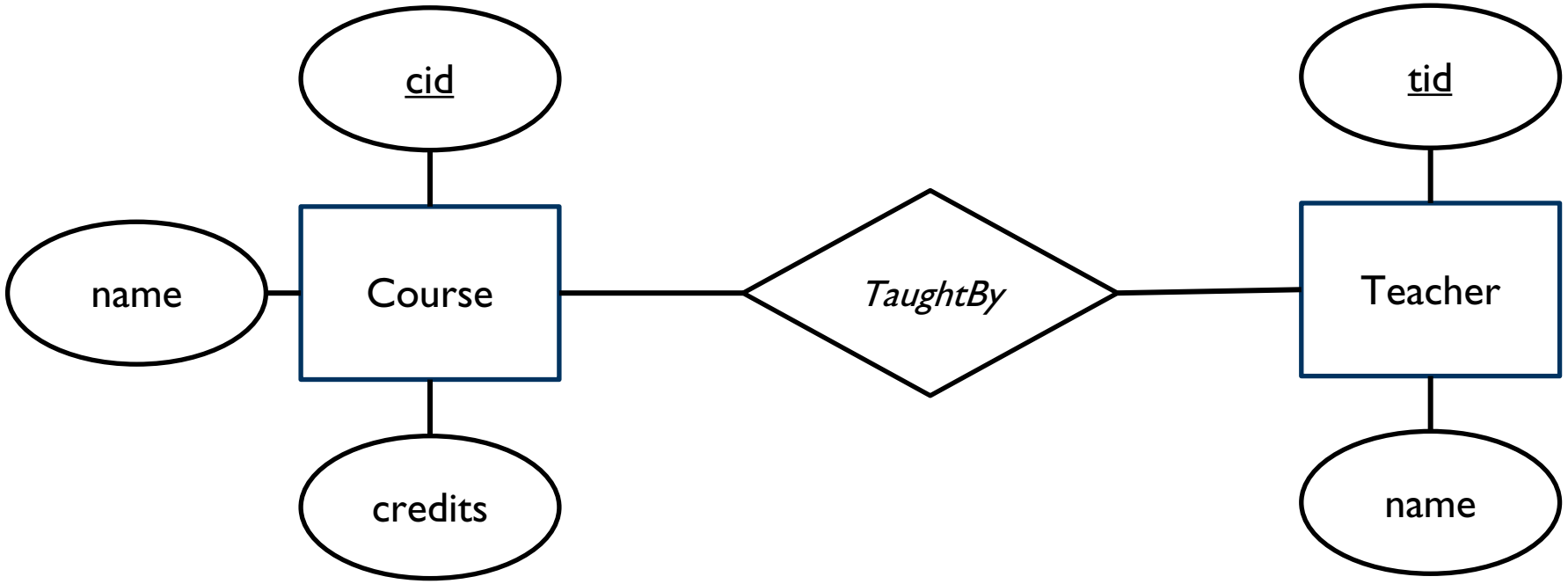
Total Participation of E_2 in R



Cardinality Ratio 1: N for E_1 : E_2 in R

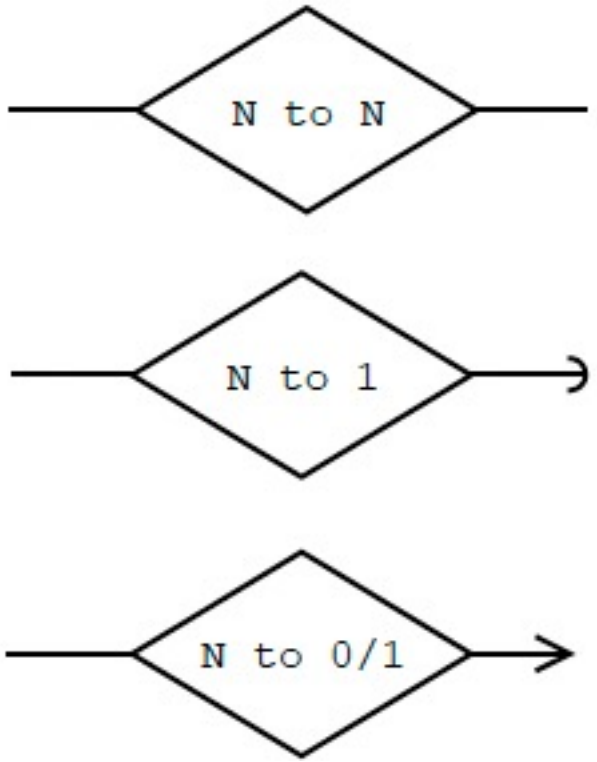


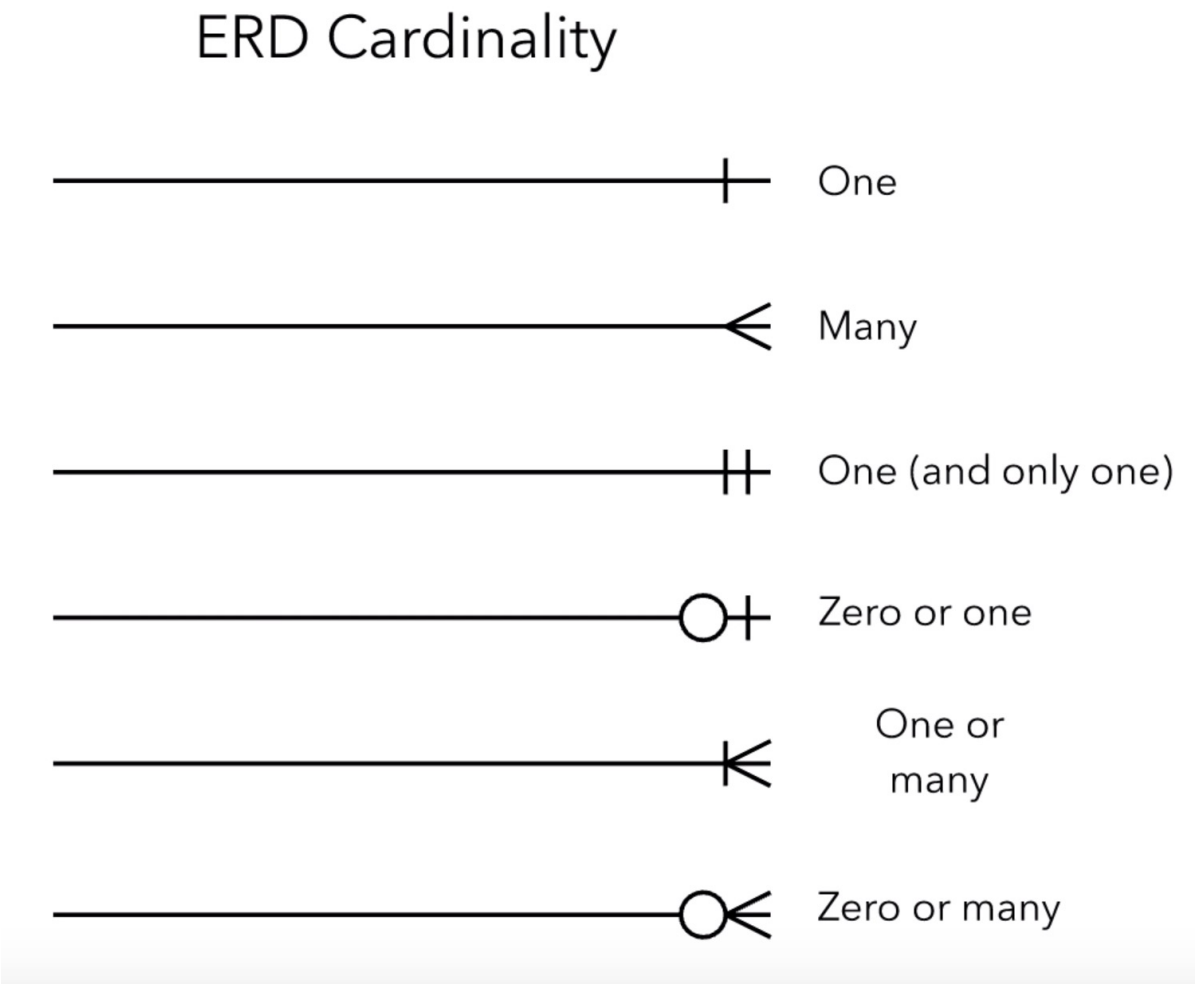
Structural Constraint (min, max) on Participation of E in R



Multiplicity

- ER-diagrams

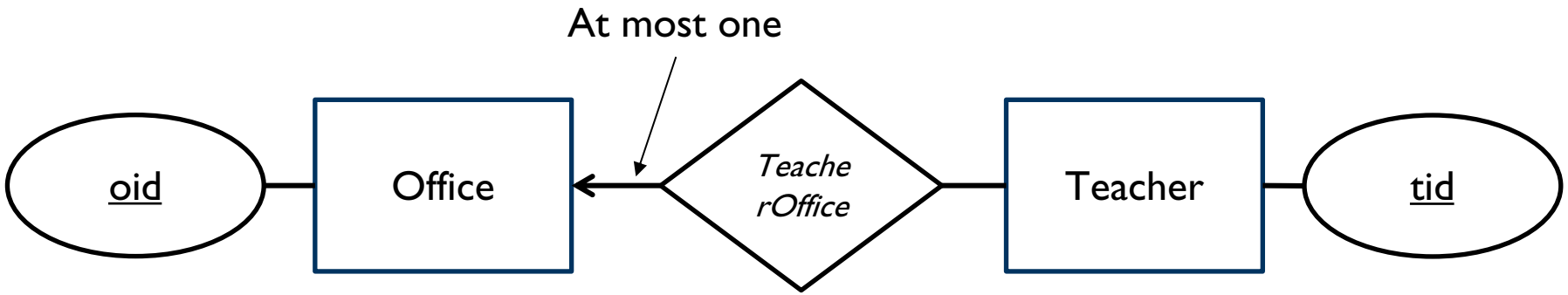




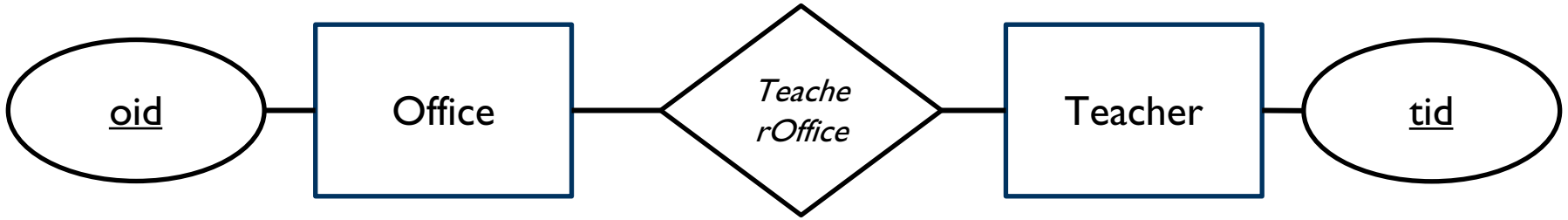
Cardinalities represent the relaitonships between databases

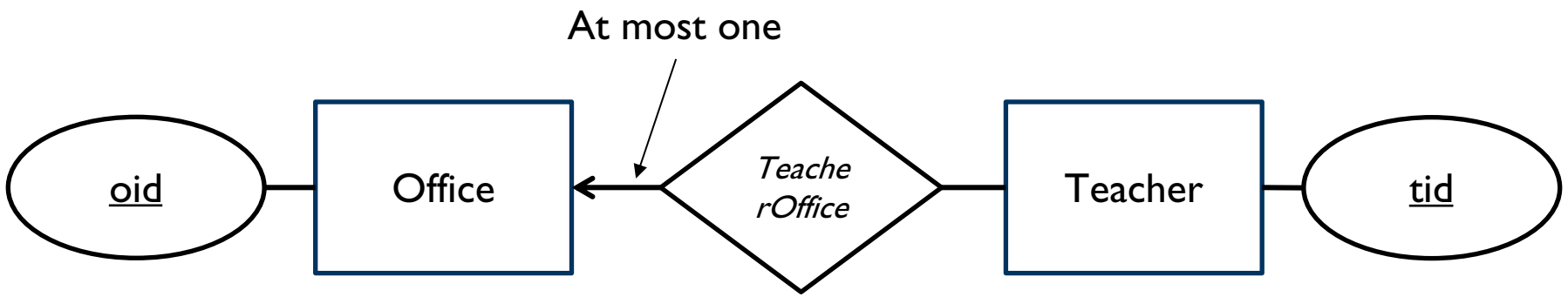
Multiplicity

- Identifying many-to-exactly-one in domains
  - With the form "Every X has a Y" (Y is another entity)
  - Ambigous cases, e.g. "Xs have Ys", each X can have multiple Ys, or each X has one Y.
  - Use many-to-one relationships for attributes that can be more accurately modelled as an entity
- Most-to-at-most-one
  - Some teachers have an office
  - A teachers may have an office

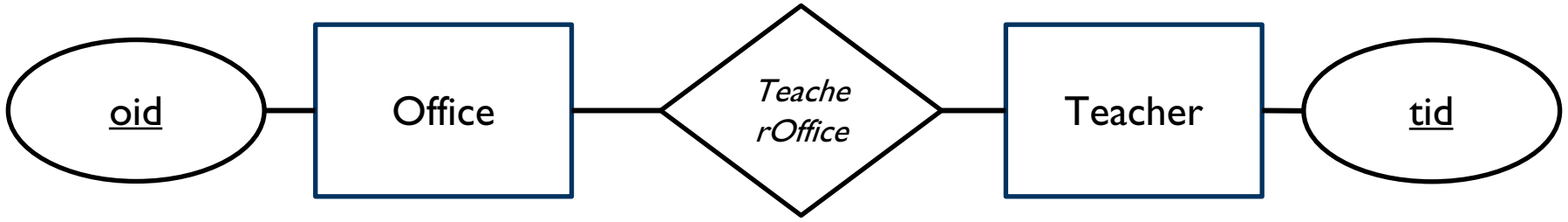


Multiplicity

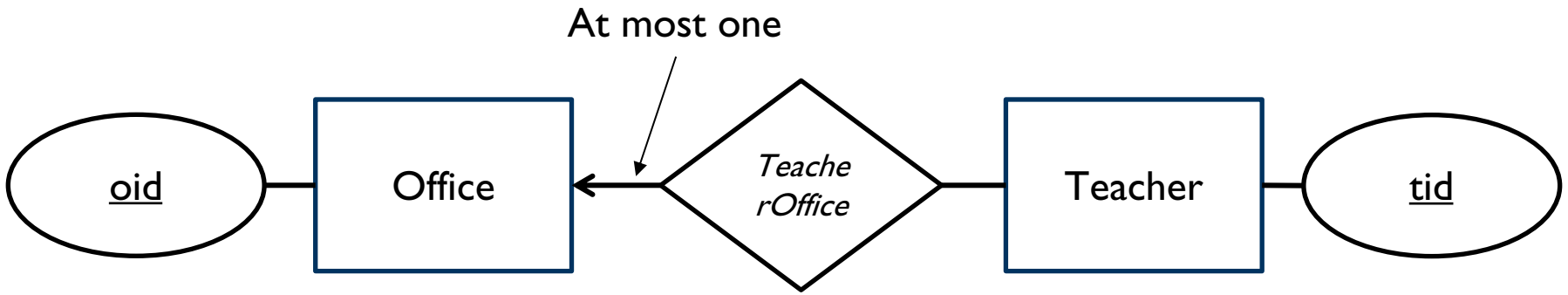




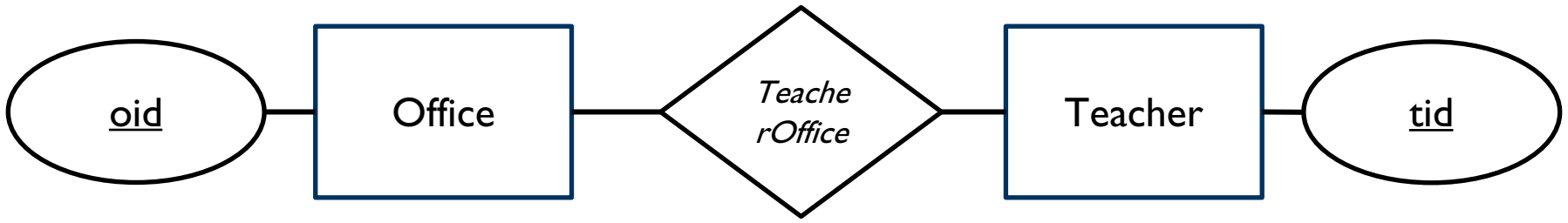
Multiplicity



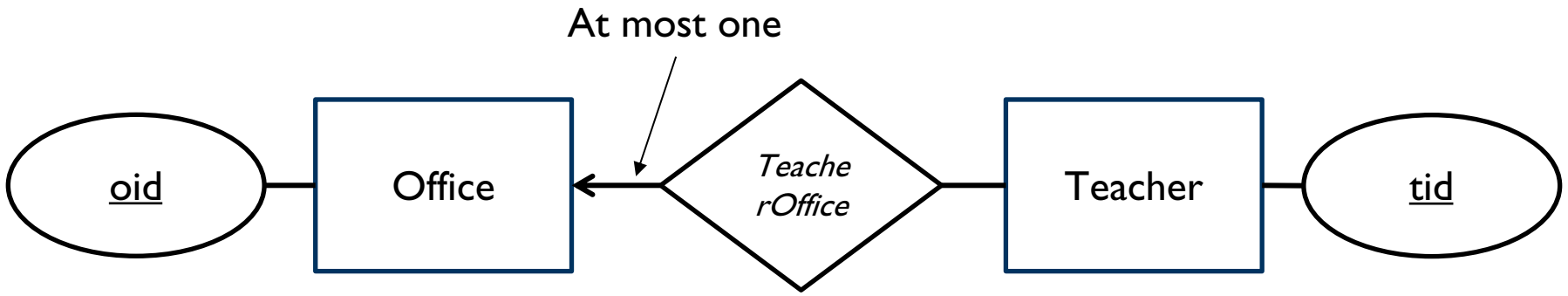
Teachers(<u>tid</u>)
Office(<u>oid</u>)
TeacherOffice(<u>teacher</u>, <u>office</u>)
office -> Office.oid
teacher -> Teachers.tid



Multiplicity

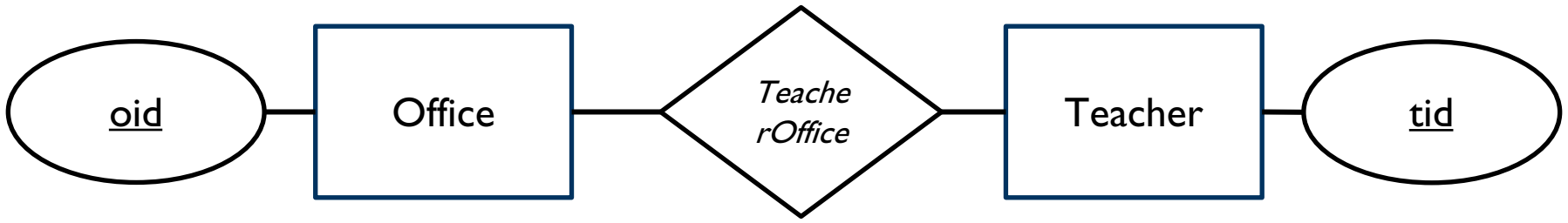


Teachers(<u>tid</u>)
Office(<u>oid</u>)
TeacherOffice(<u>teacher</u>, <u>office</u>)
office -> Office.oid
teacher -> Teachers.tid

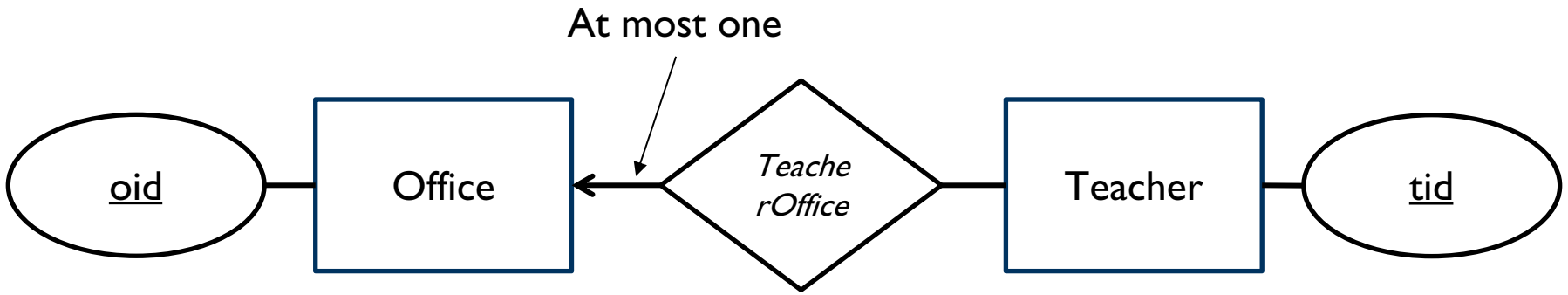


Teachers(<u>tid</u>)
Office(<u>oid</u>)
TeacherOffice(<u>teacher</u>, office)
office -> Office.oid
teacher -> Teachers.tid

Multiplicity

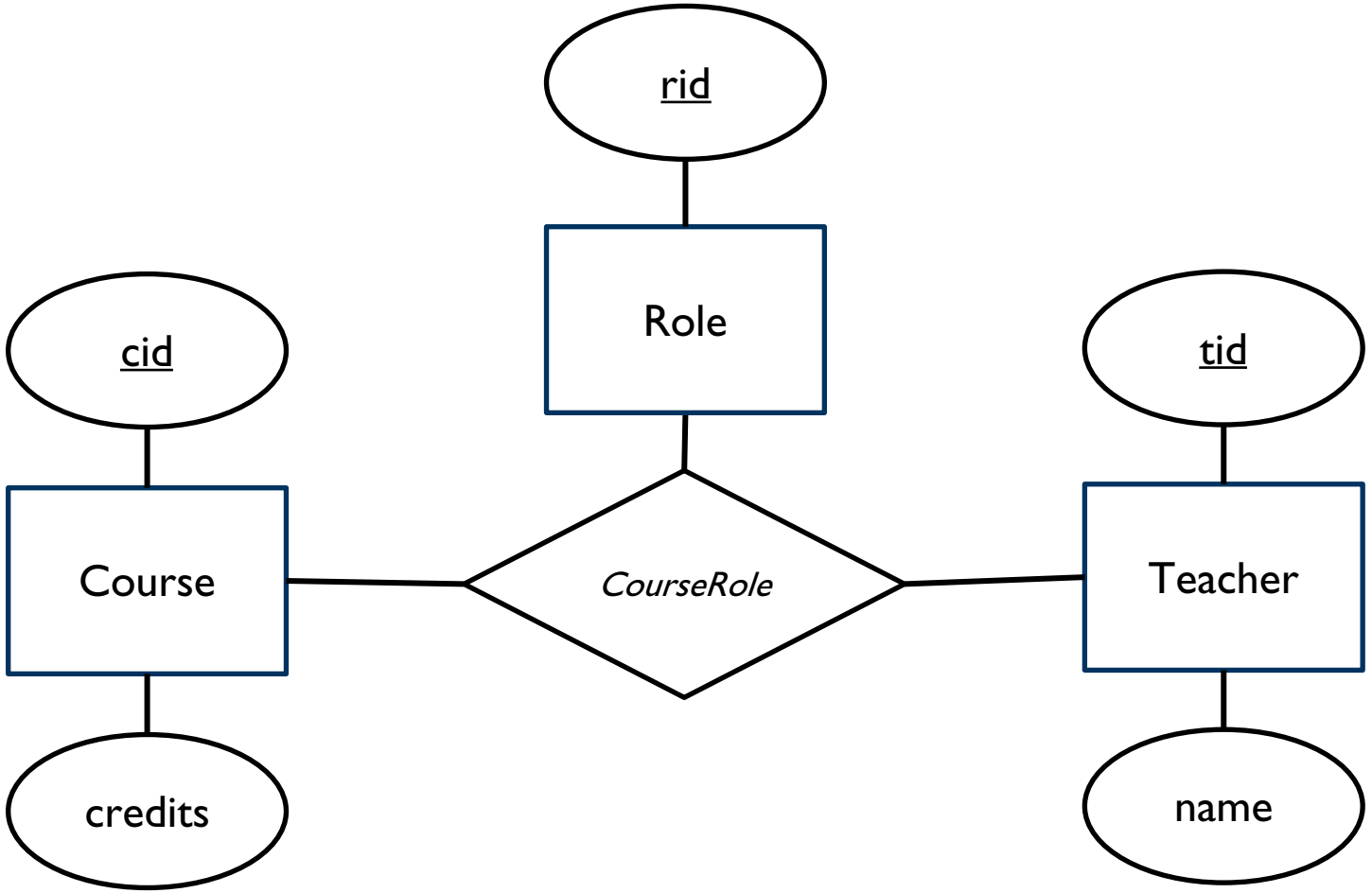


Teachers(<u>tid</u>)
Office(<u>oid</u>)
TeacherOffice(<u>teacher</u>, <u>office</u>)
office -> Office.oid
teacher -> Teachers.tid



Teachers(<u>tid</u>)
Office(<u>oid</u>)
TeacherOffice(<u>teacher</u>, office (or null))
office -> Office.oid
teacher -> Teachers.tid

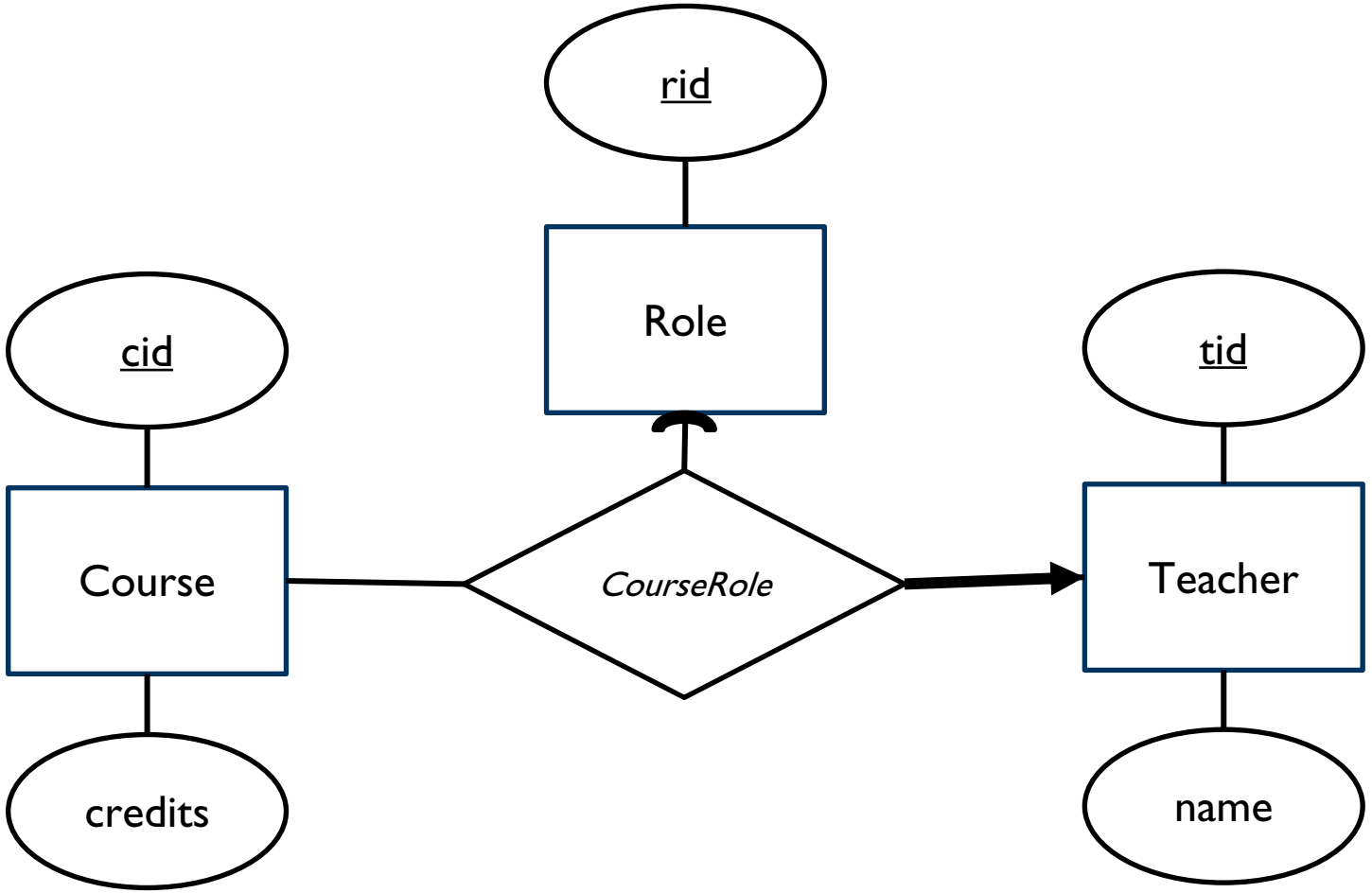
Multiway relationship



Courses(<u>cid</u>, credits)
Teachers(<u>tid</u>, name)
Role(<u>rid</u>)
TaughtBy(<u>course</u>, <u>teacher</u>, role)
course -> Courses.cid
teacher -> Teachers.tid
role→ Roles.rid

Key pairs (*<u>course, teacher</u>*) ensures assignment of any number teachers with any number of courses, for each association we need to select a valid role

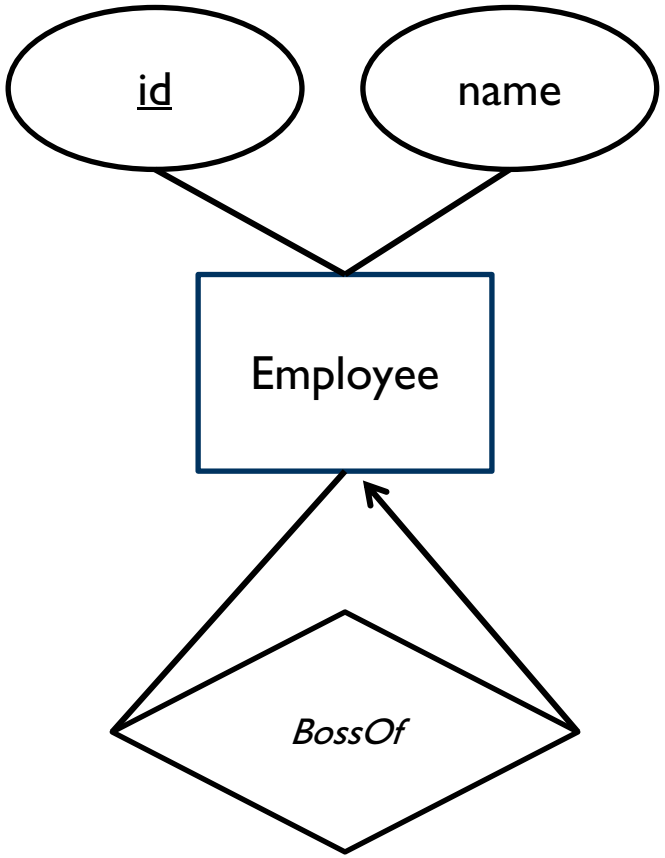
Multiway relationship



Courses(cid, credits)
Teachers(tid, name)
Role(rid)
TaughtBy(course, teacher, role)
course -> Courses.cid
teacher -> Teachers.tid
role→ Roles.rid

At most one (teacher, role) pair per course

Translating self relationships

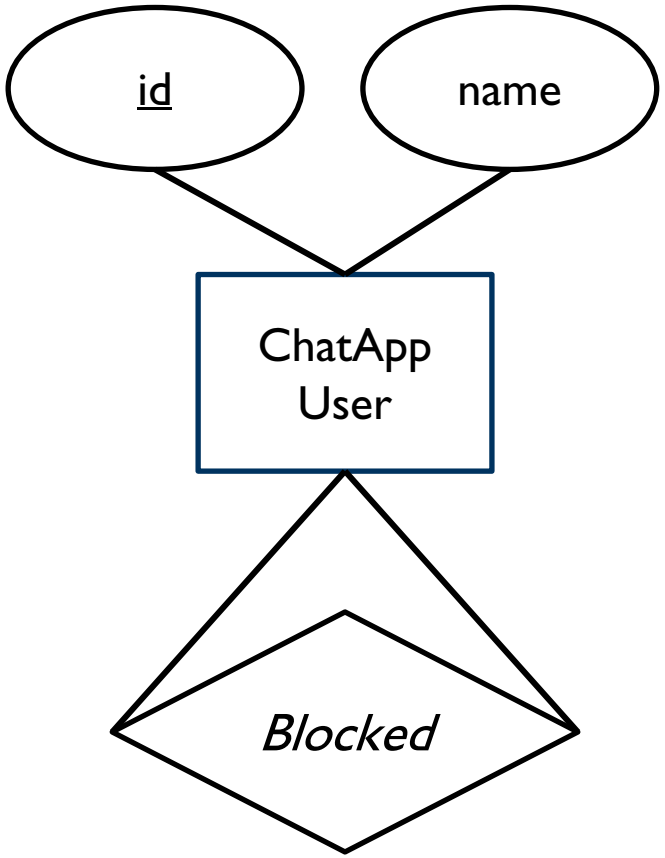


- Similar to how other relationships are handled
- Make the self-referencing nullable

Employees(id, name, boss (or null))
boss -> Emplyees.id

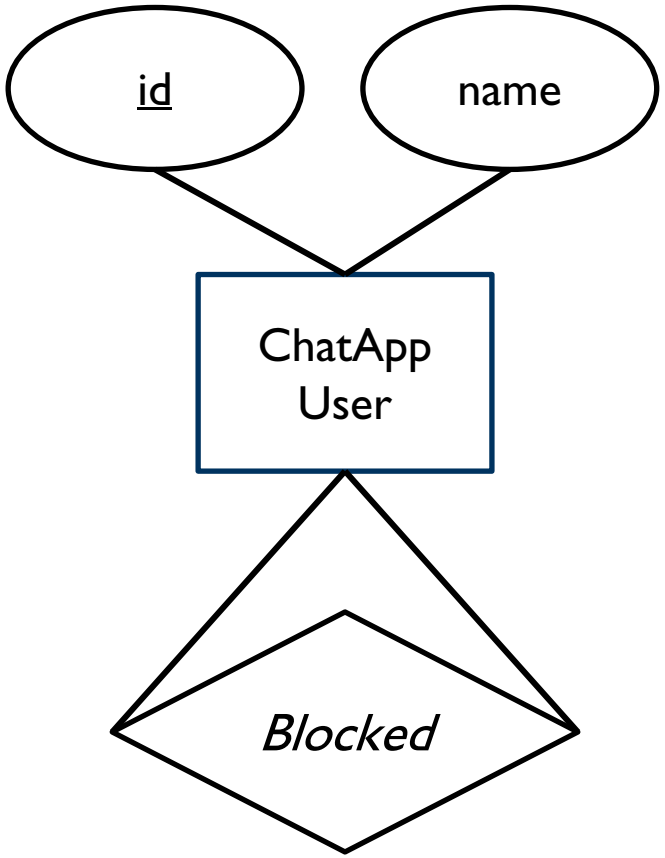
```
CREATE Table Employees(
        id INT PRIMARY KEY,
        name TEXT NOT NULL,
        boss INT REFERENCES Employees
);
```

Translating self relationships



- How to model Users can block other users

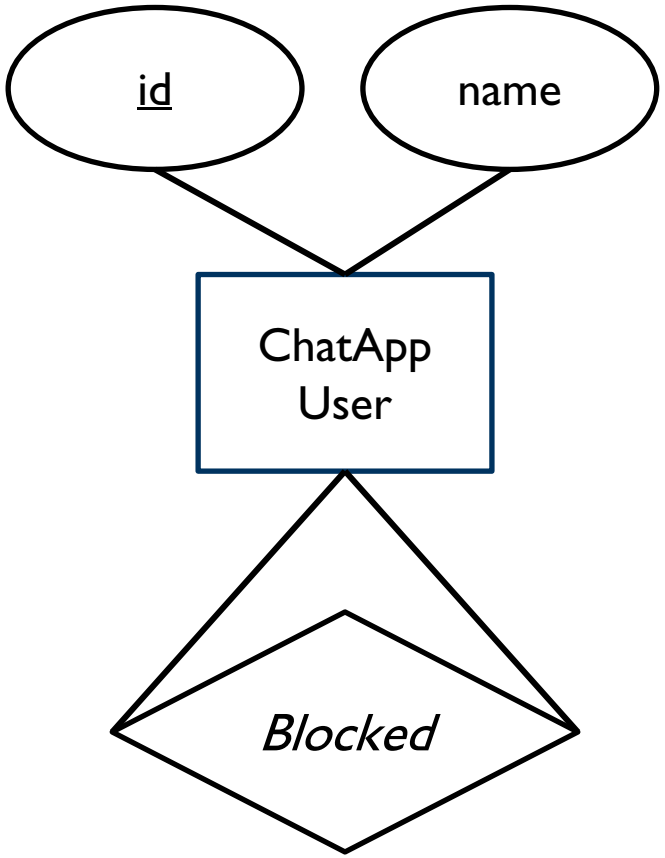
Translating self relationships



- How to model Users can block other users?
- Cardinality of the relationship?

ChatAppUsers(id, name)
Blocked(blocking, blocked)
blocking -> ChatAppUsers.id
blocked -> ChatAppUsers.id

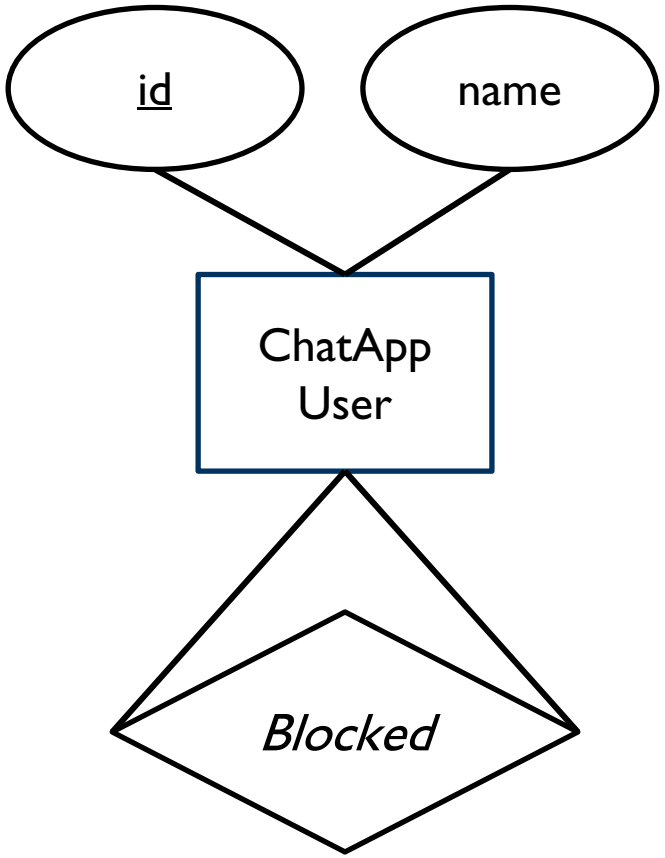
Translating self relationships



- How to model Users can block other users?
- Cardinality of the relationship?
  - Many to Many

ChatAppUsers(id, name)
Blocked(blocking, blocked)
blocking -> ChatAppUsers.id
blocked -> ChatAppUsers.id

Translating self relationships



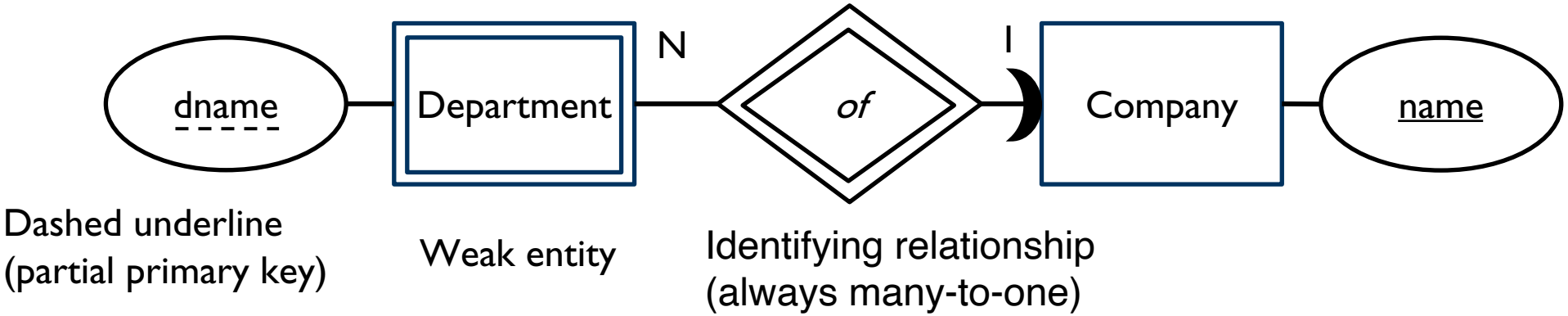
- How to model Users can block other users?
- Cardinality of the relationship?
  - Many to Many

ChatAppUsers(id, name)
Blocked(blocking, blocked)
blocking -> ChatAppUsers.id
blocked -> ChatAppUsers.id

- Limitation of self-relationships in ER-diagrams?
  - Self-referrencing (I block myself?)
  - Cycles
- Identifying self-relations in domain
  - With the form “X has … to another X”

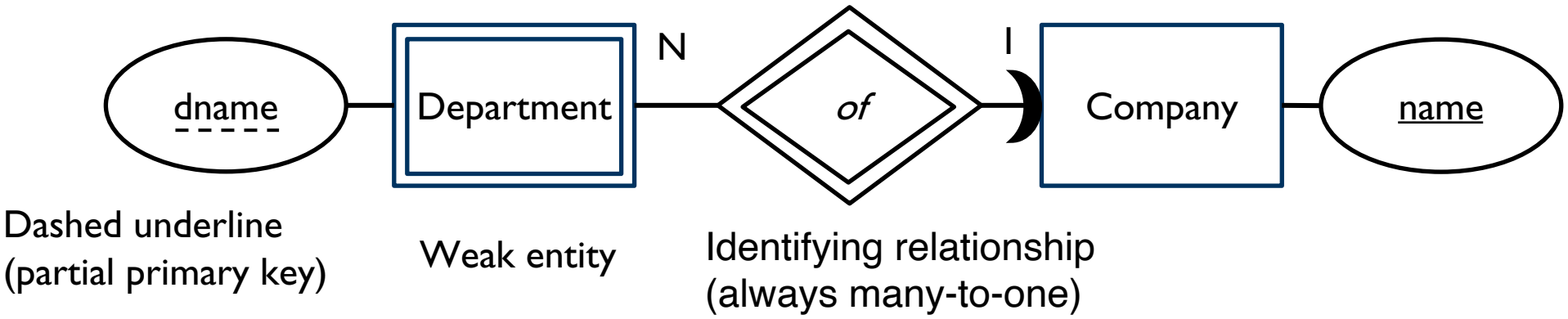
Weak Entities

- A weak entity can be identified uniquely only by considering the primary key of another (owner) entity
  - In other words, it cannot be identified only by its own attributes
  - The owner entity set and weak entity set must participate in a one to many relationship set (one owner, many weak entities)
  - Weak entity set must have total participation in this identifying relationship set
  - Weak entities have only a partial key (dashed underline)



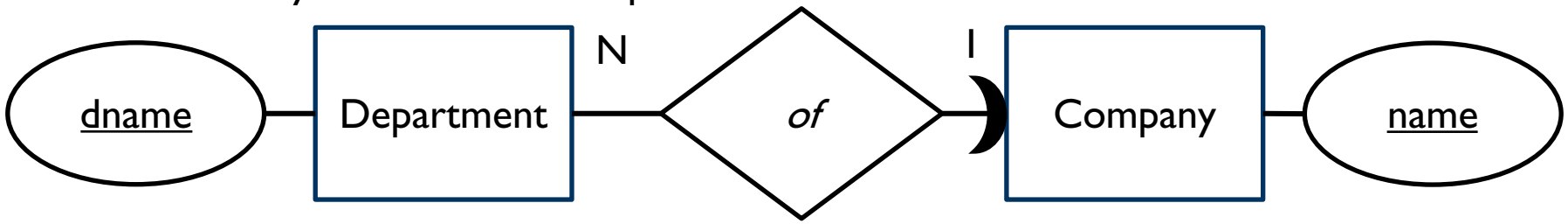
Weak Entities

- A weak entity can be identified uniquely only by considering the primary key of another (owner) entity
- Identifying relationship is when the existence of a row in a child table depends on a row in a parent table
  - Make the foreign key part of the child's primary key
  - logical relationship is that the child cannot exist without the parent



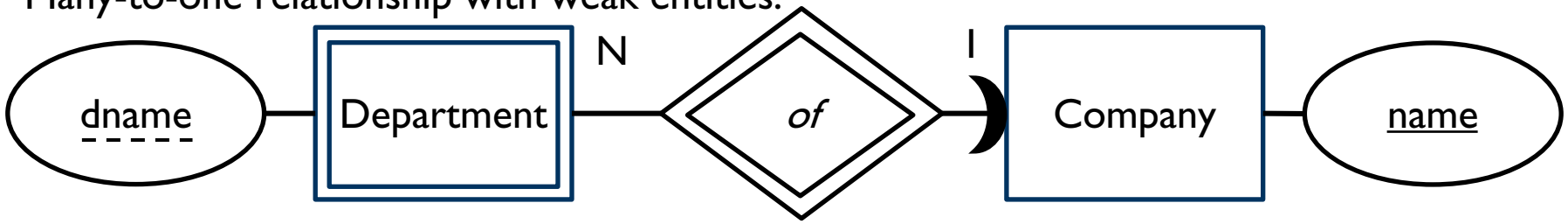
Example translating weak entities

Common Many-to-one relationship:



Companies(<u>name</u>)
department(<u>dname</u>, company)
company->companies.name

Many-to-one relationship with weak entities:



Companies(<u>name</u>)
department(<u>dname</u>, <u>company</u>)
company->companies.name

- Identifying weak entities in domain descriptions
  - If attributes determined for an entity are not sufficient to identify members
  - E.g. Student id are unique within classes

ER Diagram Examples

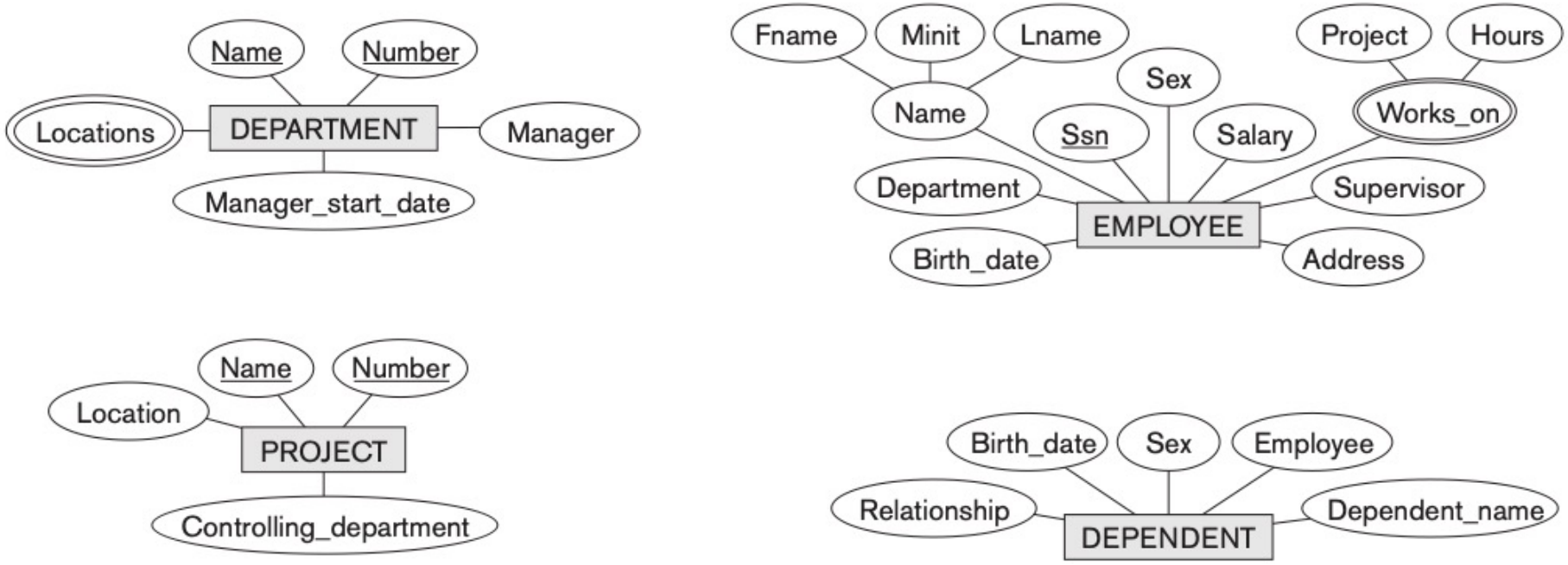


Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

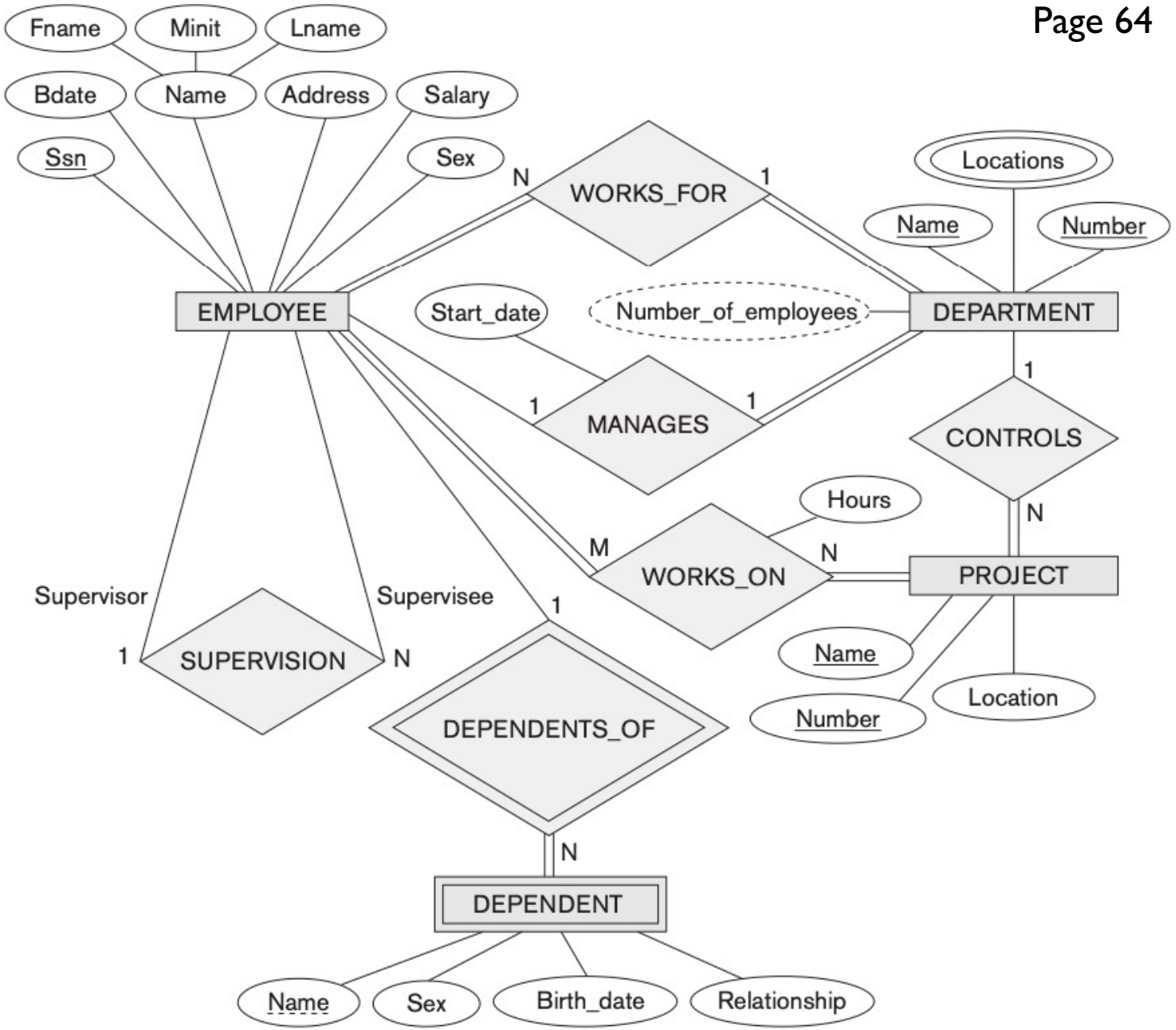


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 3.14.