



COMP9311: Database Systems

Data Modelling

(textbook: chapters 3 and 4)

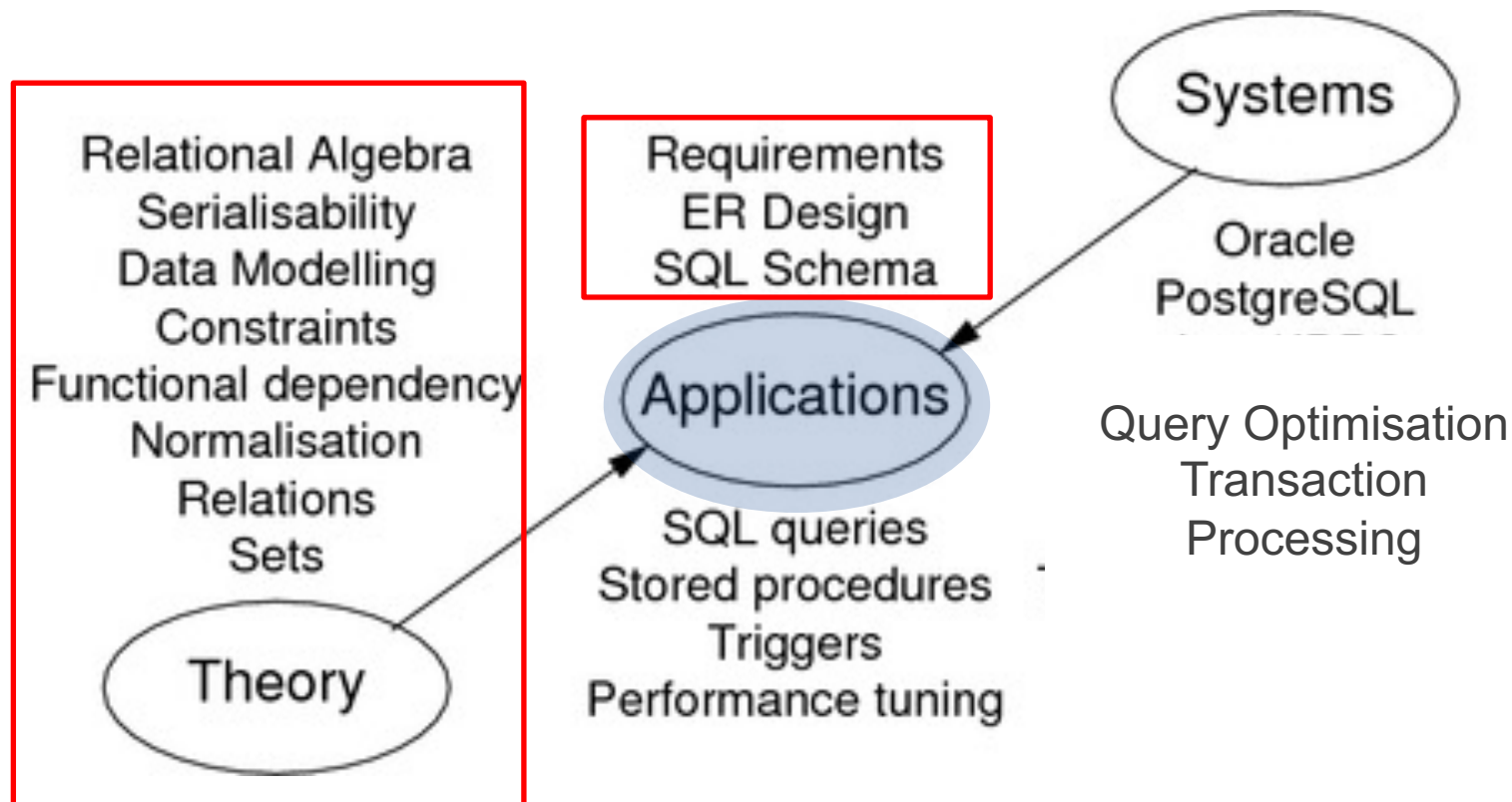
Term 3 2022

Week 1 Data Modelling

By Helen Paik, CSE UNSW

Disclaimer: the course materials are sourced from previous offerings of COMP9311 and COMP3311

Overview of the Databases Field



Database Application Development

A variation on standard software engineering process:

- analyse application requirements
- develop a data model to meet these requirements
- define operations (transactions) on this model
- implement the data model as relational schema
- implement operations via SQL and procedural PLs
- construct an interface to these operations
- At some point, populate the database (may be via interface)

Data Modelling

Aims of data modelling:

- describe what *information* is contained in the database
(e.g., entities: students, courses, accounts, branches, patients, ...)
- describe *relationships* between data items
(e.g., John is enrolled in COMP3311, Andrew's account is held at Coogee)
- describe *constraints* on data
(e.g., 7-digit IDs, students can enrol in no more than four courses per term)

Data modelling is a *design* process

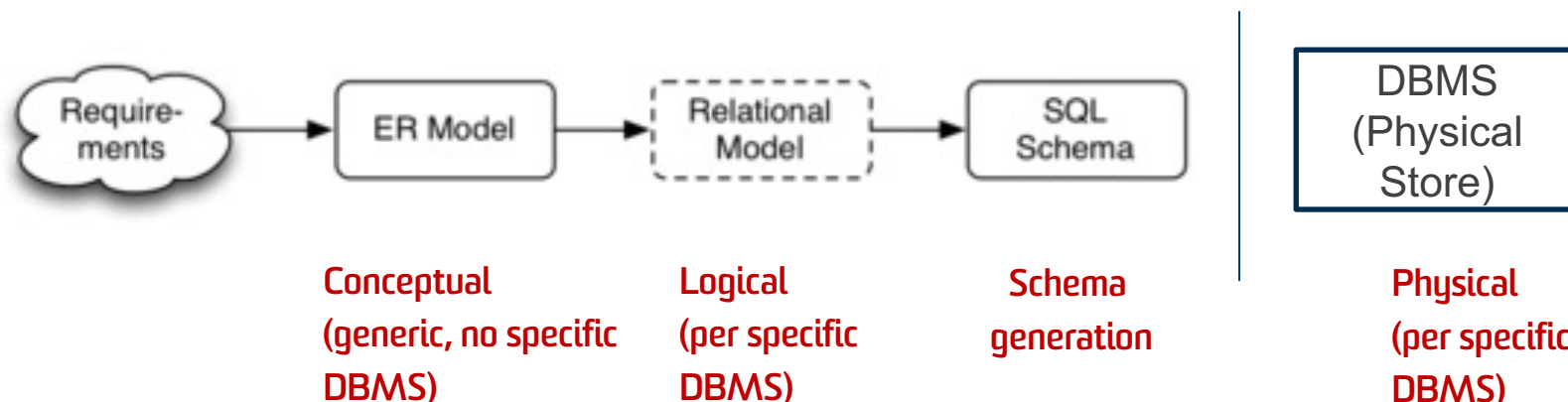
- converts **requirements** into a data **model**

Data Modelling

Kinds of data models:

- *conceptual*: abstract, high-level data model, e.g., ER, ODL (object data language) – user friendly
- *logical*: concrete, for implementation in specific DBMS, e.g., relational
- *physical*: internal file storage (inside a specific DBMS)

Strategy: design using abstract model; map to logical model, DBMS takes care of the physical model



Some Design Ideas

Consider the following when you work through a design exercise:

- start simple ... evolve design as problem better understood
- identify objects (and their properties), then relationships
- most designs involve kinds (classes) of people
- *keywords in requirements suggest data/relationships*
(rule-of-thumb: nouns → data, verbs → relationships)
- don't confuse operations/actions with relationships
(operation: he **buys** a book; relationship: the book **is owned** by him)
- consider all possible data, not just what is available

Example - Gmail Data Model

Consider the Google Mail System:

Let's develop an **informal** data model for it by identifying:

- the data items involved (objects and their attributes)
- relationships between these data items
- constraints on the data and relationships

Quality of Designs

There is no single "best" design for a given application.

Most important aspects of a design (data model):

- correctness (satisfies requirements accurately)
- completeness (all reqs covered, all assumptions explicit)
- consistency (no contradictory statements)

Potential inadequacies in a design:

- omits information that needs to be included
- contains redundant information (\Rightarrow inconsistency)
- leads to an inefficient implementation
- violates syntactic or semantic rules of data model

Entity-Relationship Data Modelling

In ER, The world is viewed as **a collection of inter-related "entities"**.

ER has **three** major modelling **constructs**:

- *entity*: objects (“things”) in your world that you are interested
 - *Person, Restaurants, Books, University Courses, ...*
- *attribute*: data item describing a property of interest
 - Person (name, phone number, DOB, ...)
- *relationship*: association between entities (objects)
 - Person dines-at Restaurant

Entity-Relationship (ER) Diagrams

ER diagrams are a graphical tool for data modelling.

An ER diagram consists of:

- a collection of *entity set* definitions
- a collection of *relationship set* definitions
- *attributes* associated with entity and relationship sets
- connections between entity and relationship sets

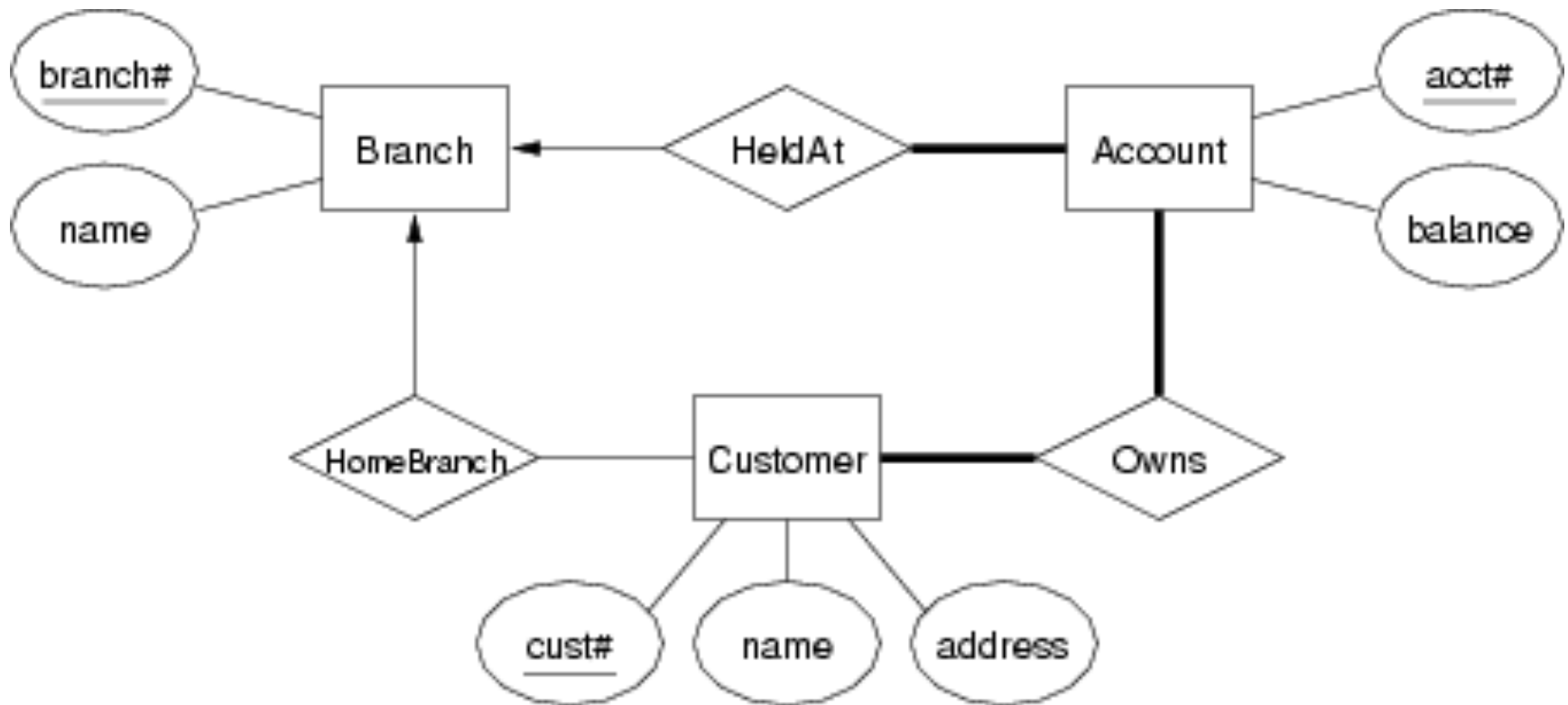
Terminology: when discussing "entity sets", we frequently say just "entity"

The ER model is not a standard, so many variations exist.

Lecture notes use simple notations -> as 'COMP9311 standard'.

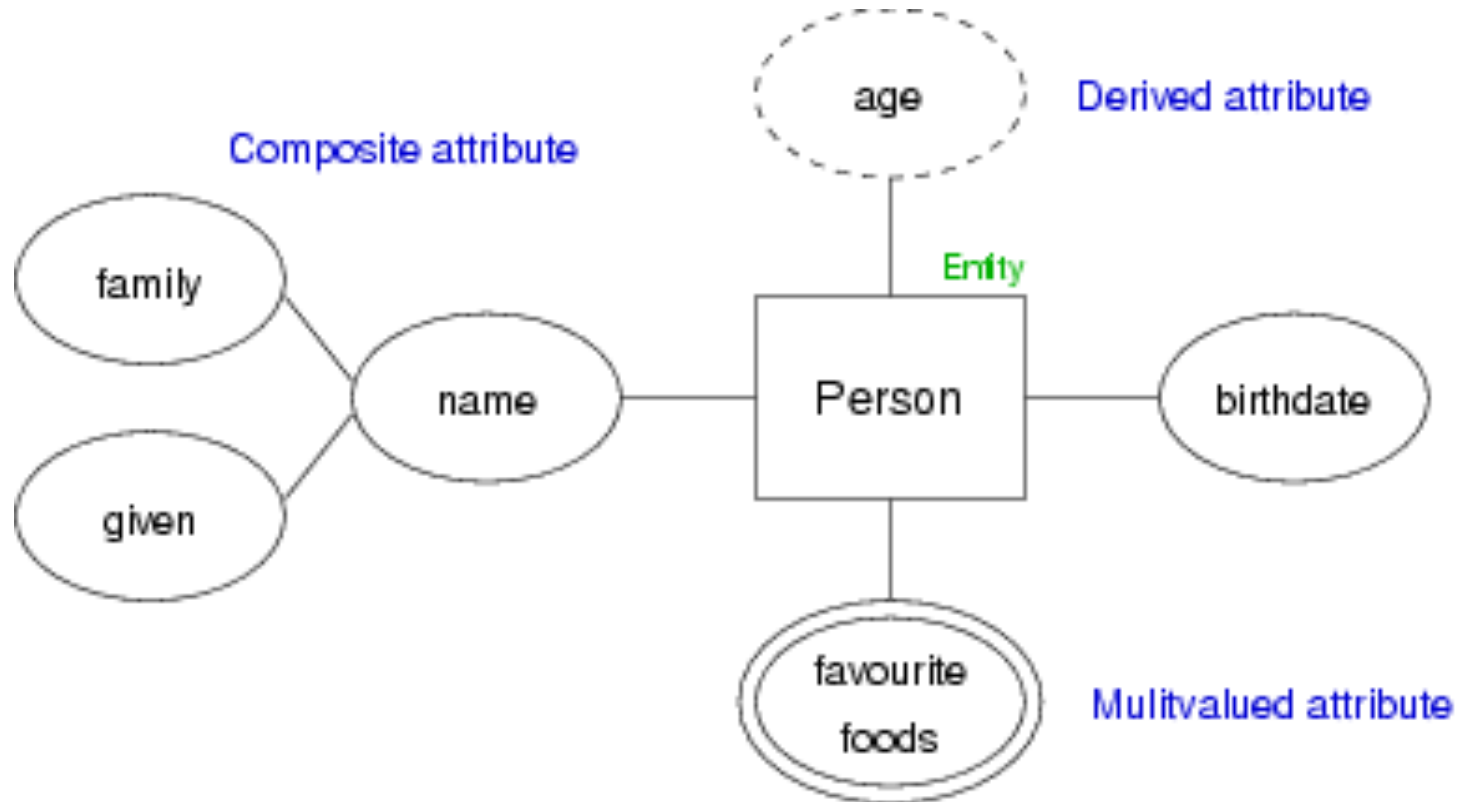
Entity-Relationships

Example ER Diagram: entities, attributes, relationships/connections



Entity-Relationships

Example of attribute notations



Entity Sets and Entity Type

An *entity set* can be viewed as either:

- a set of entities with the same set of attributes
- an abstract description of a class of entities -> a.k.a. *Type (Entity Type)*

Entity Type Name:

EMPLOYEE

COMPANY

Name, Age, Salary

Name, Headquarters, President

**Entity Set:
(Extension)**

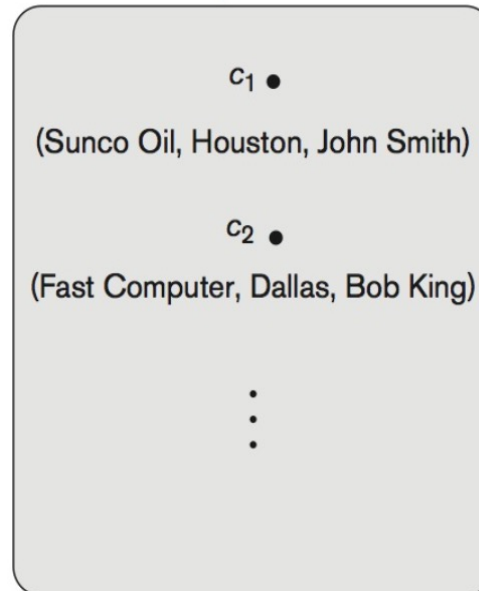
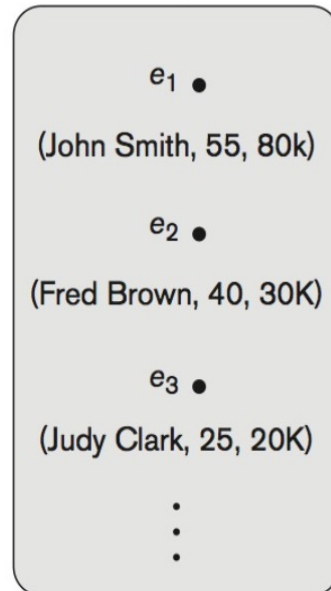


Figure 3.6

Two entity types, EMPLOYEE and COMPANY, and some member entities of each.

Entity Sets and Keys

Entities of an entity type, say EMPLOYEE needs a **key** to distinguish each other in a set.

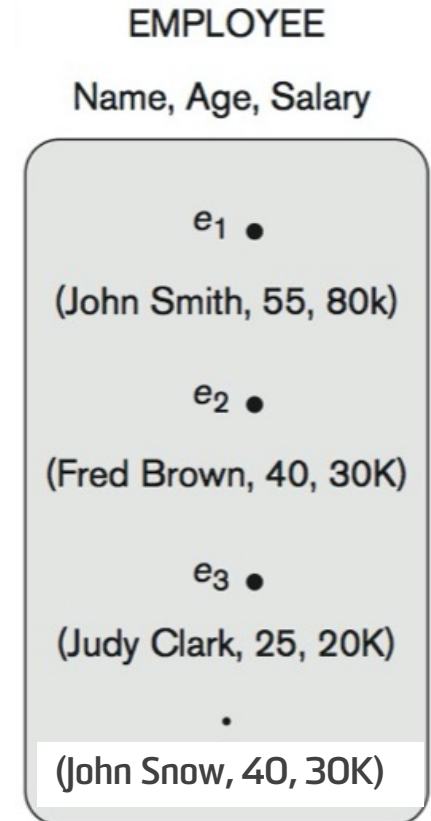
Key (superkey): any set of attributes whose set of values are distinct over entity set

- natural (e.g., name+age+salary) or artificial (e.g., employee number)

Candidate key = minimal superkey (no subset is a key)

Primary key = candidate key chosen by DB designer later in the development stage

Keys are indicated in ER diagrams by underlining



Relationship Sets

Relationship: an association among several entities

- e.g., Customer(9876) **is the owner of** Account(12345)
- e.g., Student(0001) **is enrolled in** Course (9311)

Relationship set: collection of relationships of the same type

- *Degree* = # entities involved in reln (in ER model, ≥ 2)
- *Cardinality* = # associated entities on each side of reln
- *Participation* = must every entity be in the relationship

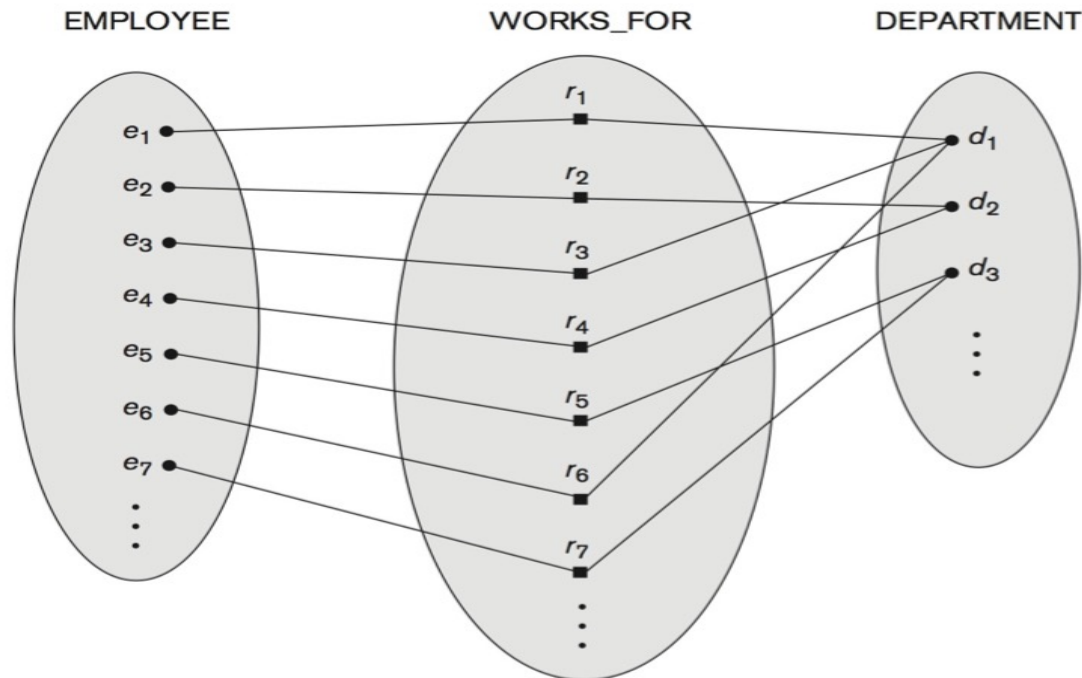
Example: relationship participation



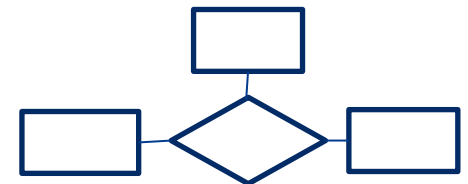
How to think about the relationships more concretely:



Entity “sets”, Relationship “sets” and their memberships

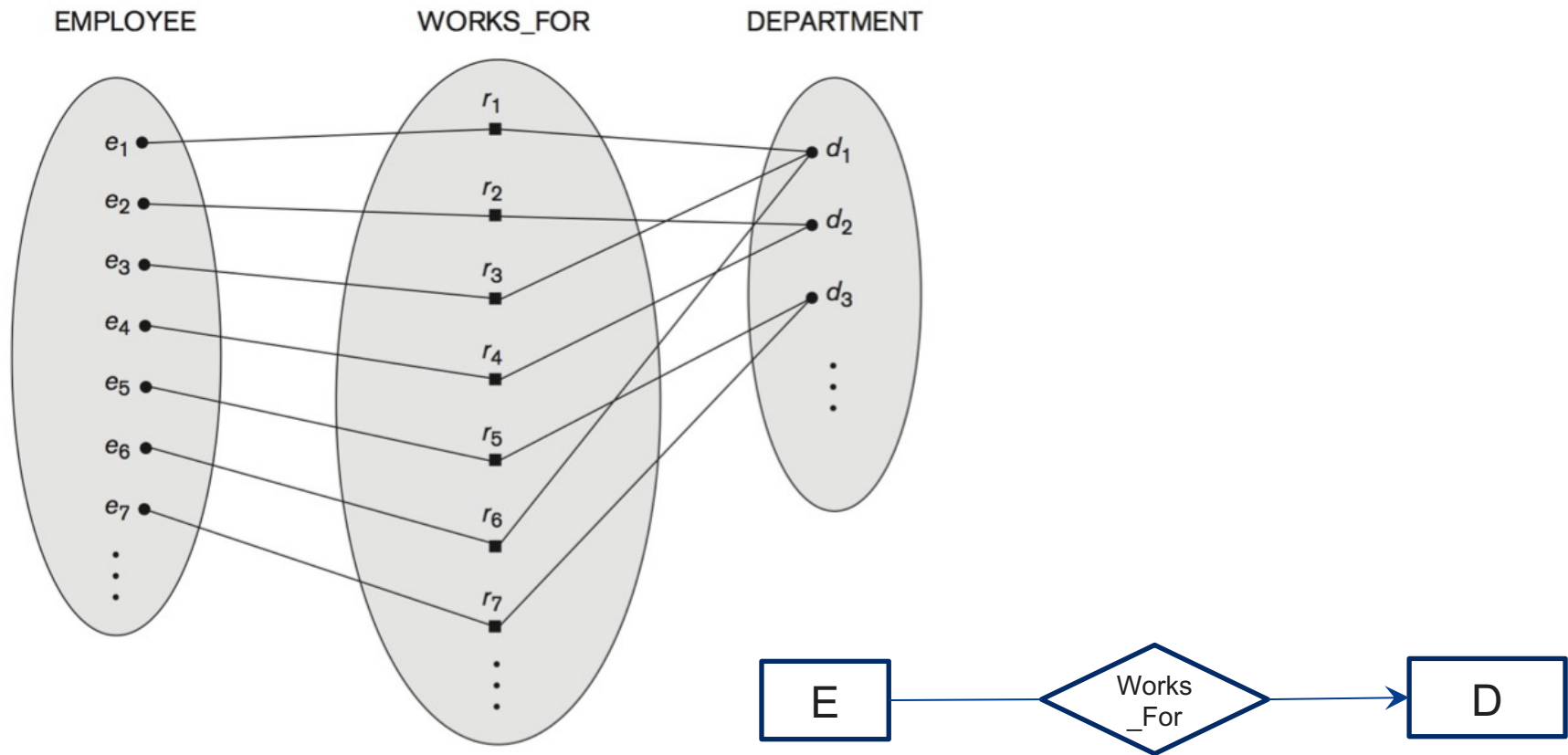


First ... Number of Degree in this relationship set?



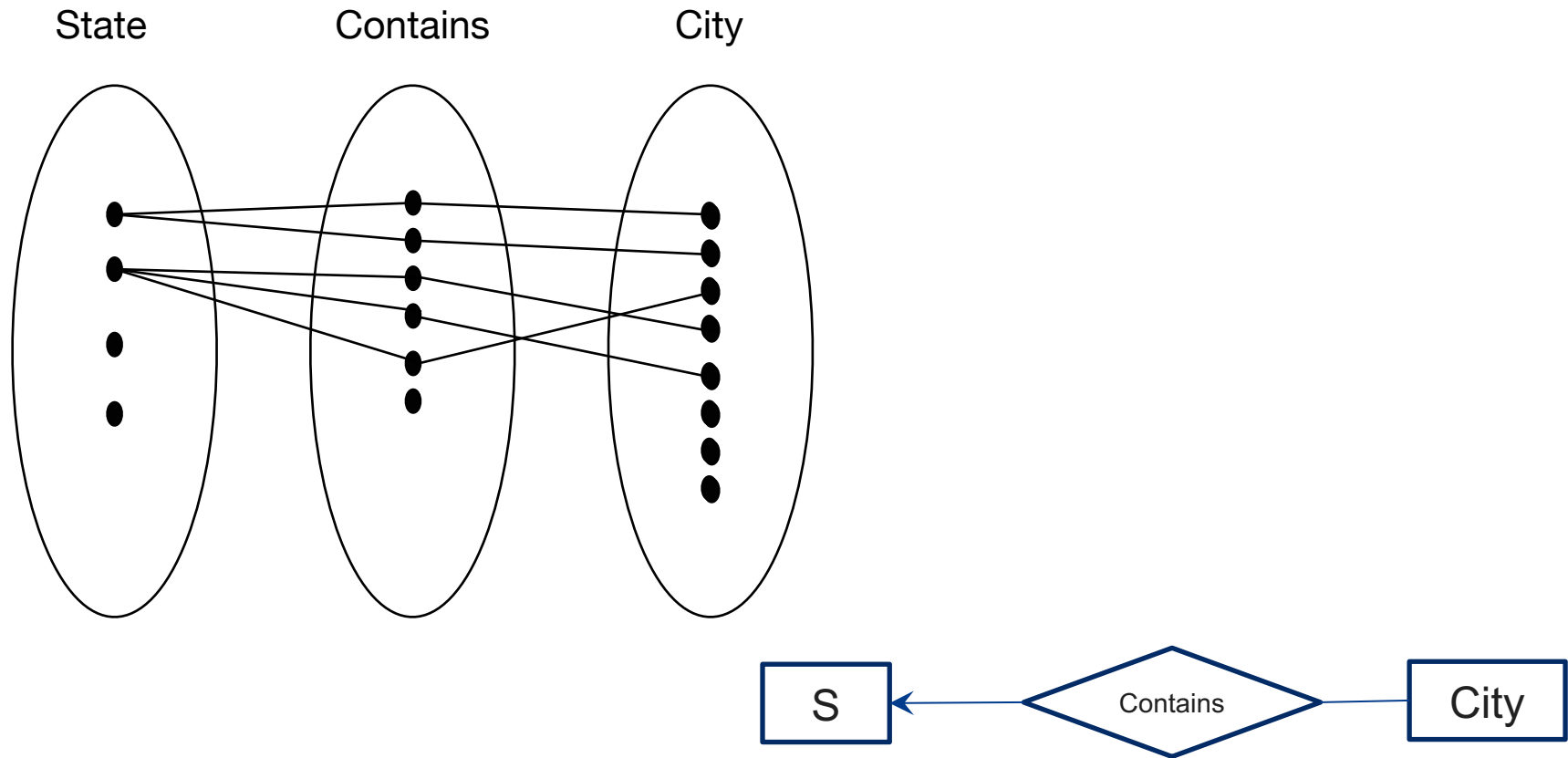
Placing 'constraints' on the relationships

Cardinality (think how many number of *relationship instances* that an entity can participate?) – this should depend on your requirements



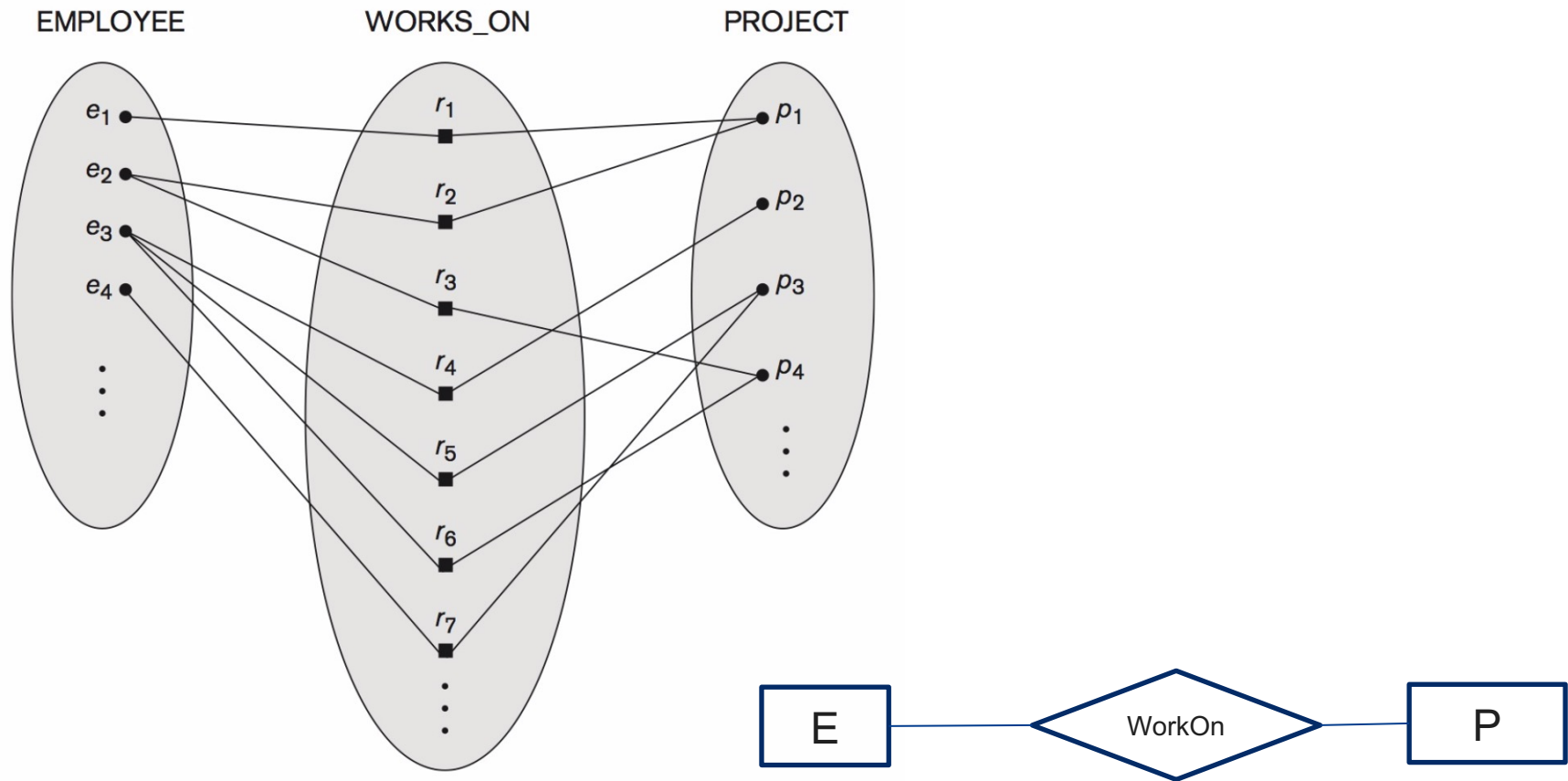
Placing 'constraints' on the relationships

Cardinality (think how many number of *relationship instances* that an entity can participate?) – this should depend on your requirements



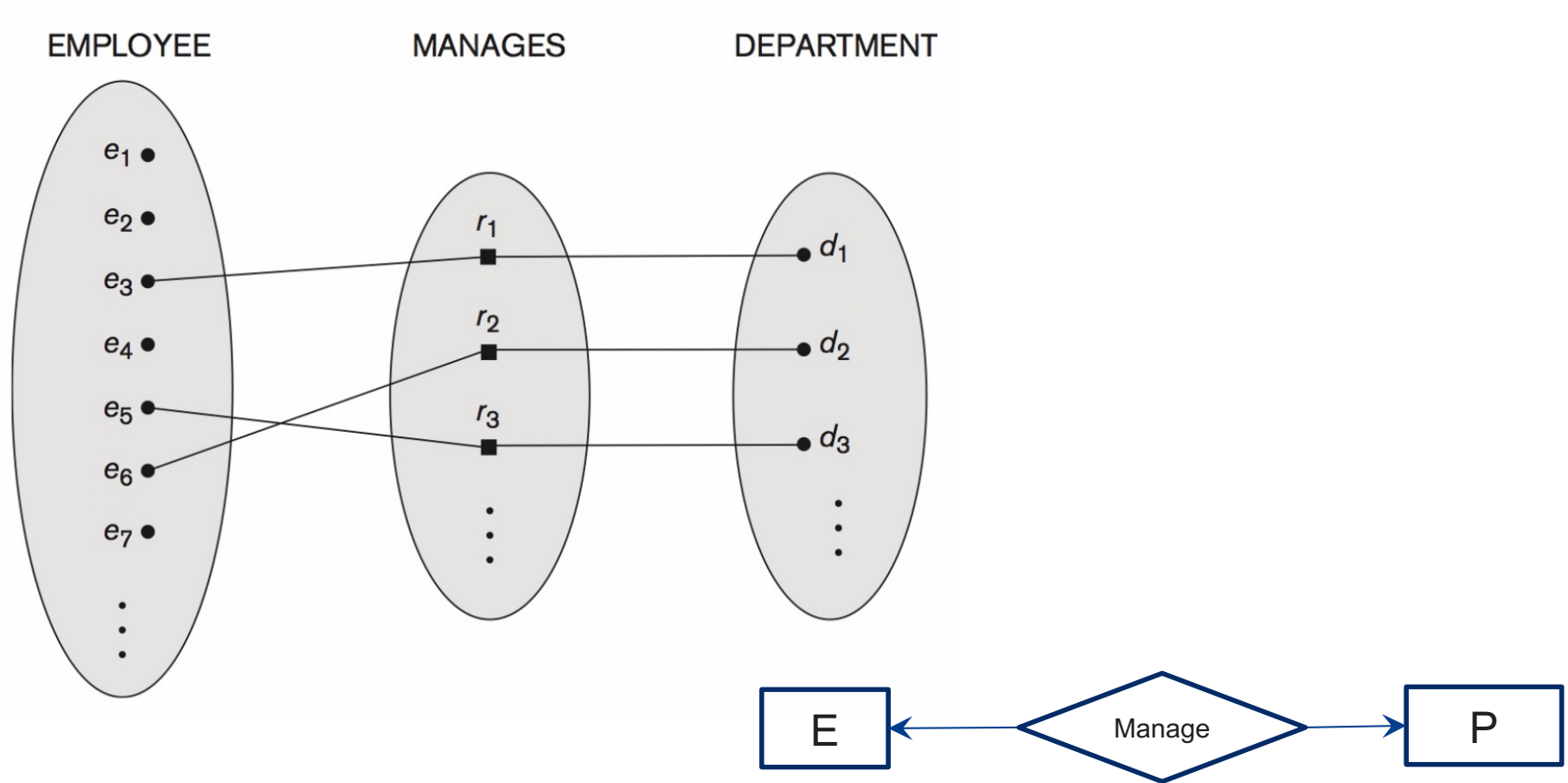
Placing 'constraints' on the relationships

Cardinality (think how many number of *relationship instances* that an entity can participate?) – this should depend on your requirements



Placing 'constraints' on the relationships

Cardinality (think how many number of *relationship instances* that an entity can participate?) – this should depend on your requirements



Recursive (or self-referencing) relationships

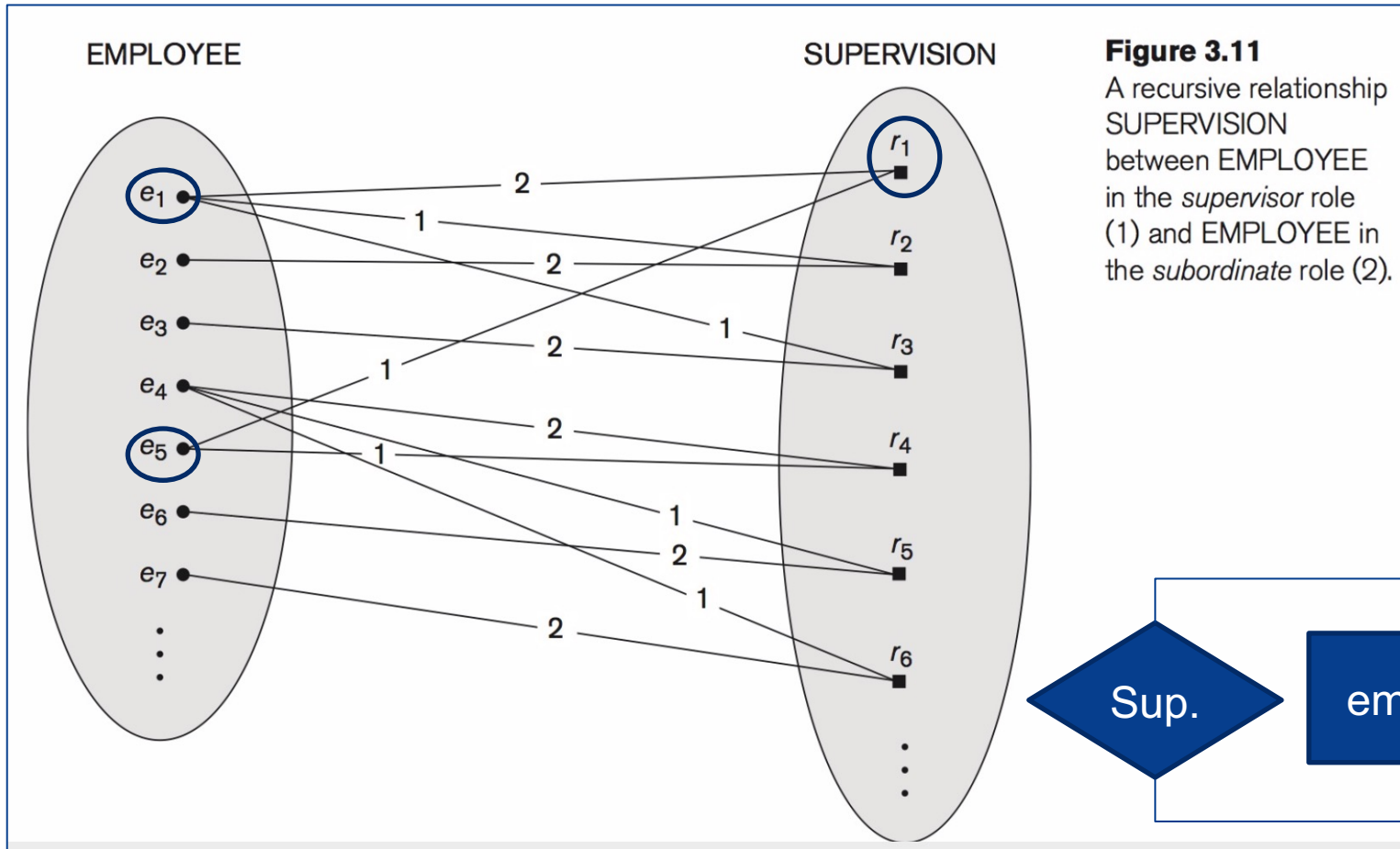


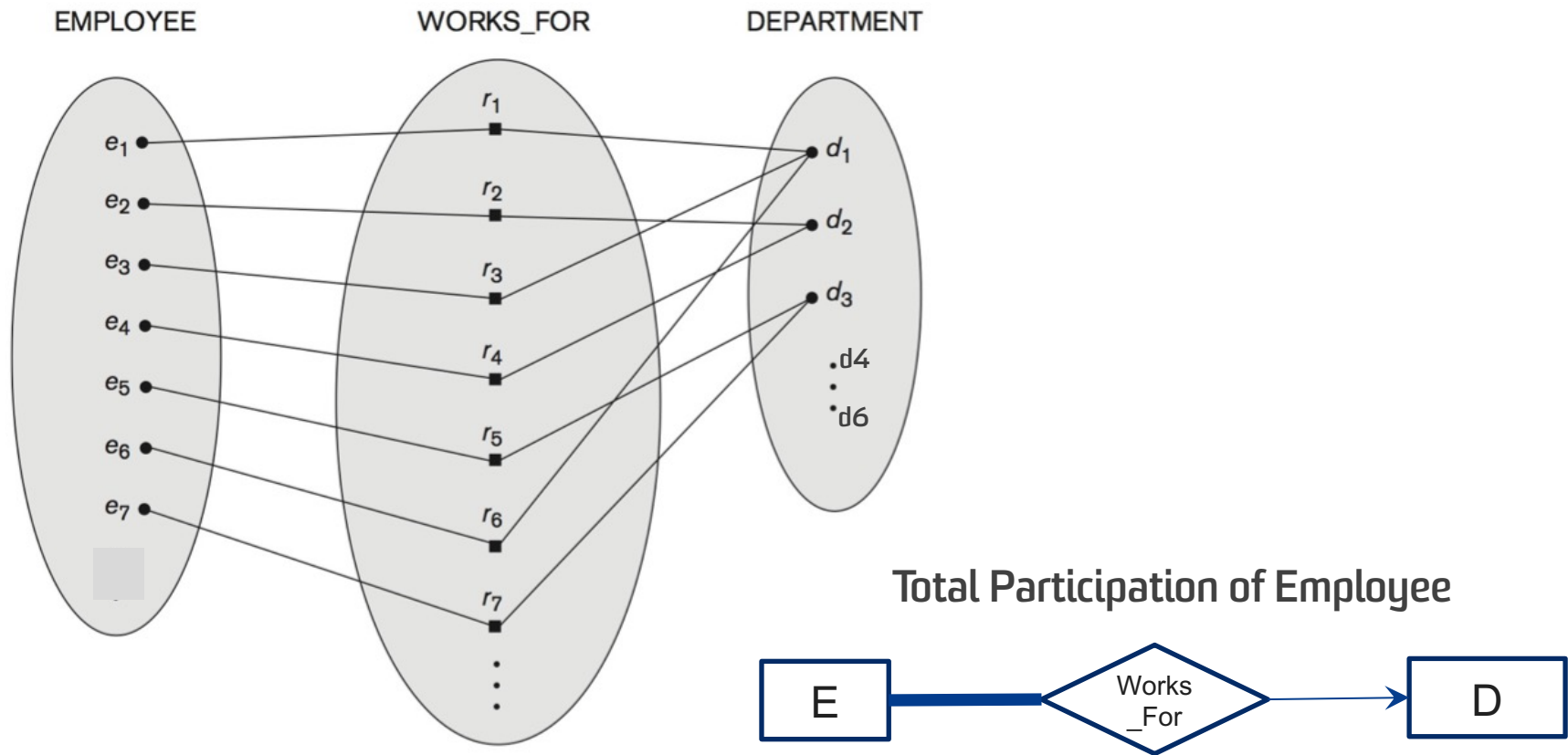
Figure 3.11

A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

What would the ER diagram look like in this scenario?

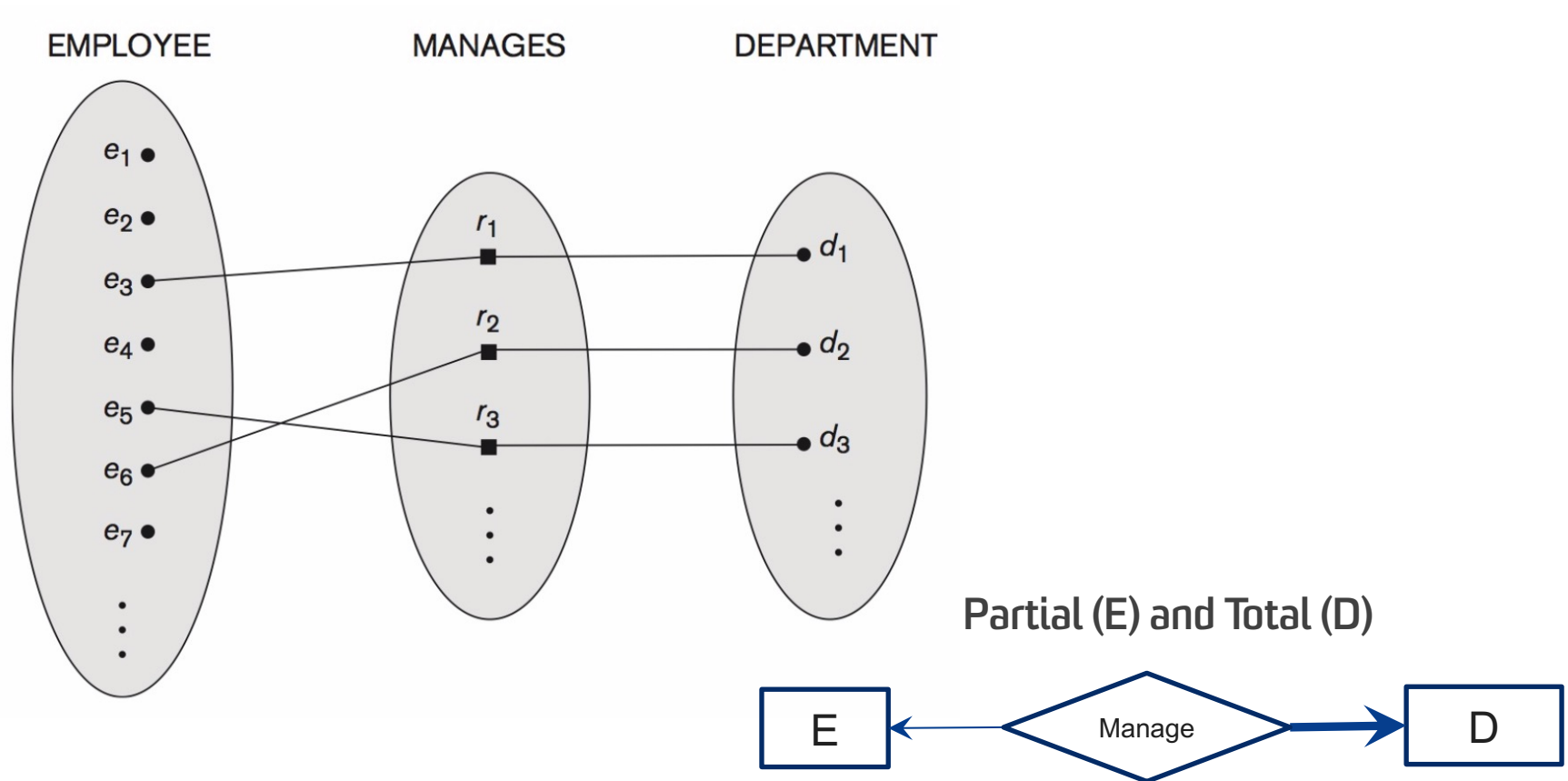
Placing 'constraints' on the relationships

Participation (think "is it 'every (or total)' or 'some' entity instances" ?) – this should depend on your requirements



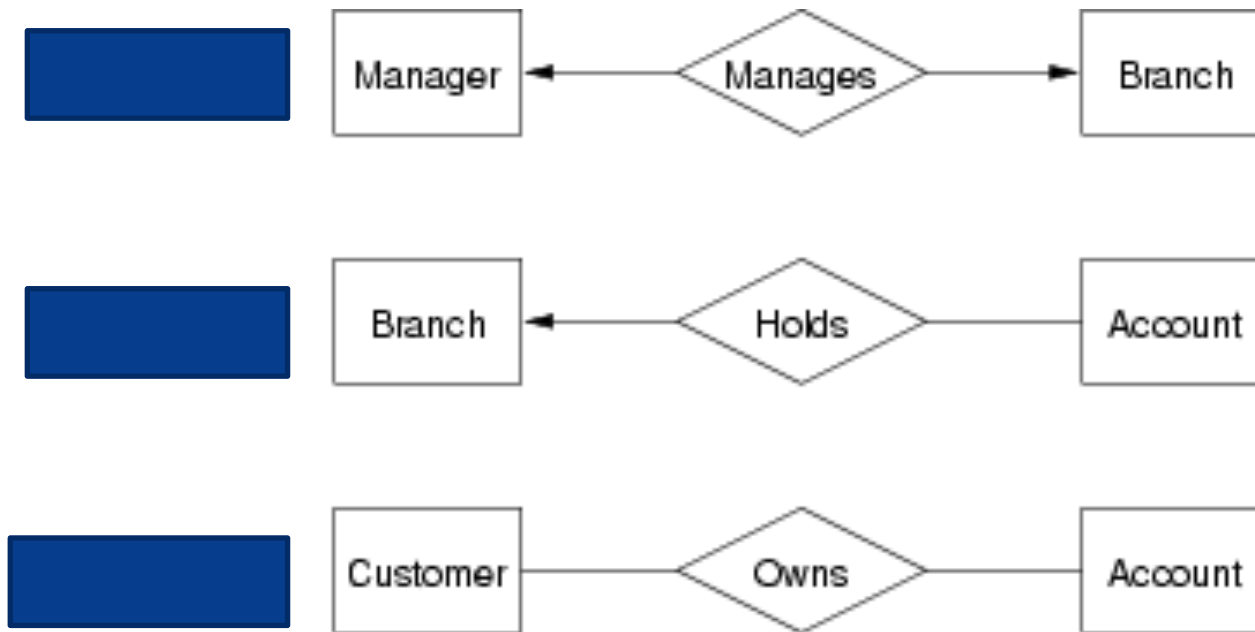
Placing 'constraints' on the relationships

Participation (think is it 'every (or total)' or 'some' entity instances ?) – this should depend on your requirements



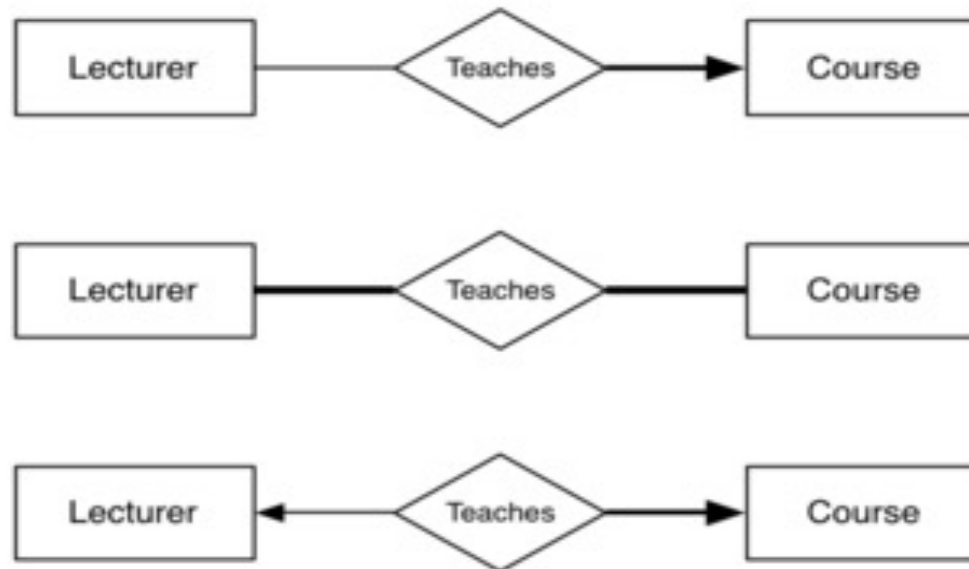
Exercise 2: Relationship Semantics

Relationships – degree and cardinality



Exercise 2: Relationship Semantics

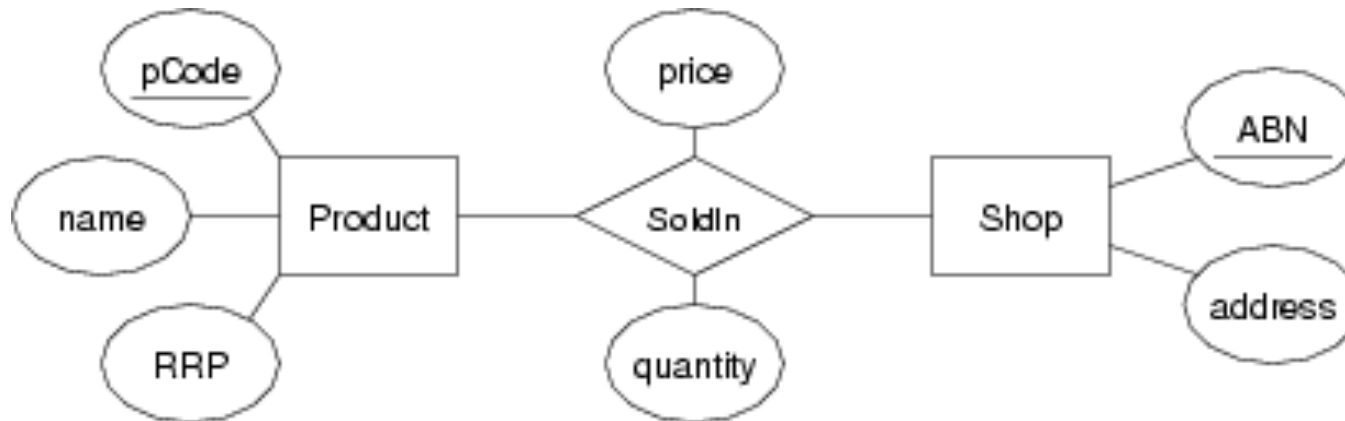
Describe precisely the scenarios implied by the following relationships:
(degree, cardinality and participation)



Relationship with attributes

In some cases, a relationship needs associated attributes.

Example:



(Price and quantity are related to products in a particular shop)