The Entity/Relationship (ER) Model & DB Design

Gary KL Tam

Department of Computer Science
Swansea University

Conventions and examples adapted from book:
Modern Database Management

Overview

A primary goal of database design is to decide what tables to create. Usually there are two principles:

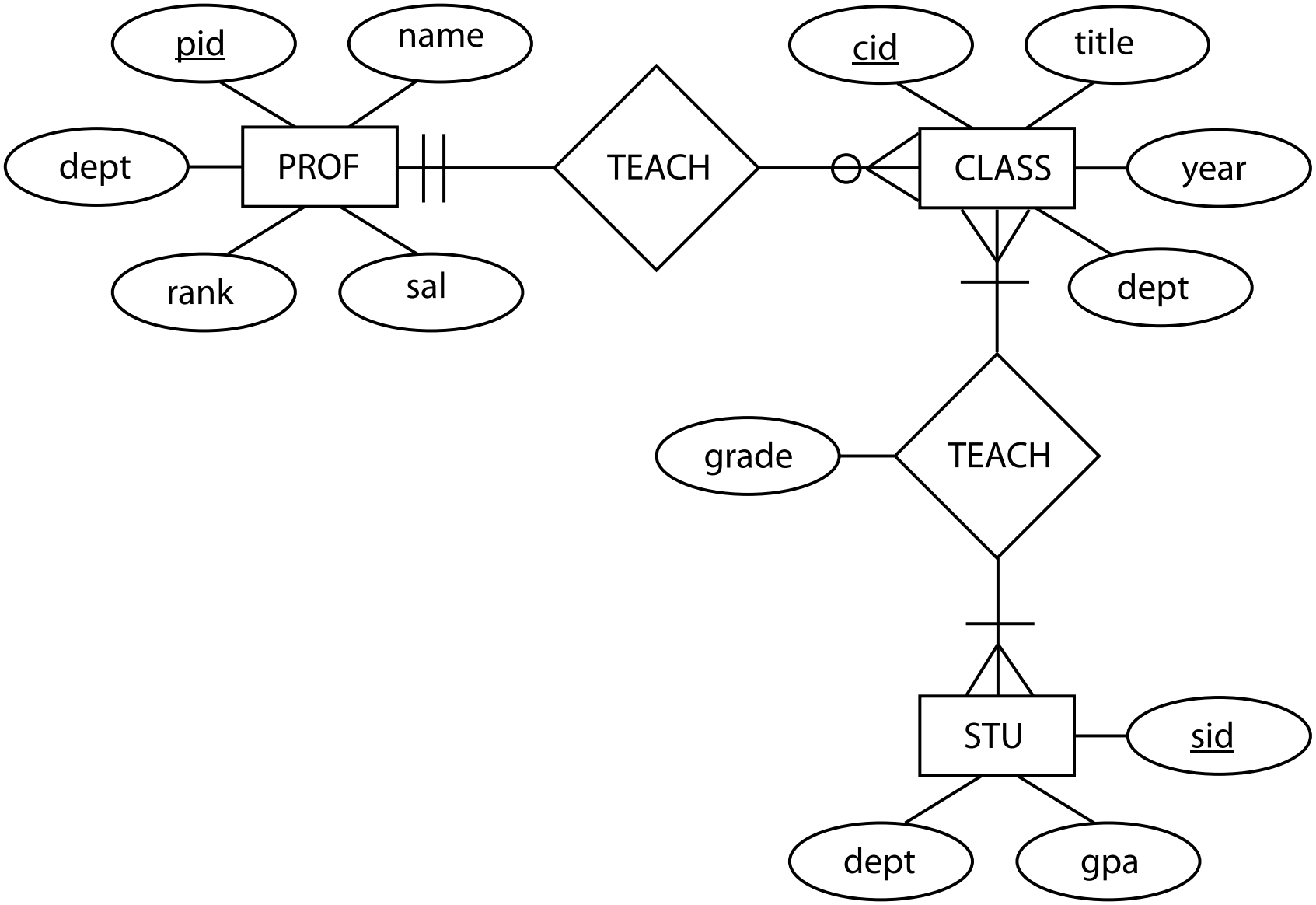
1. Capture all the information that needs to be captured by the underlying application.
2. Achieve the above with little redundancy.

The first principle is enforced with an entity relationship (ER) diagram, while the second with normalization.

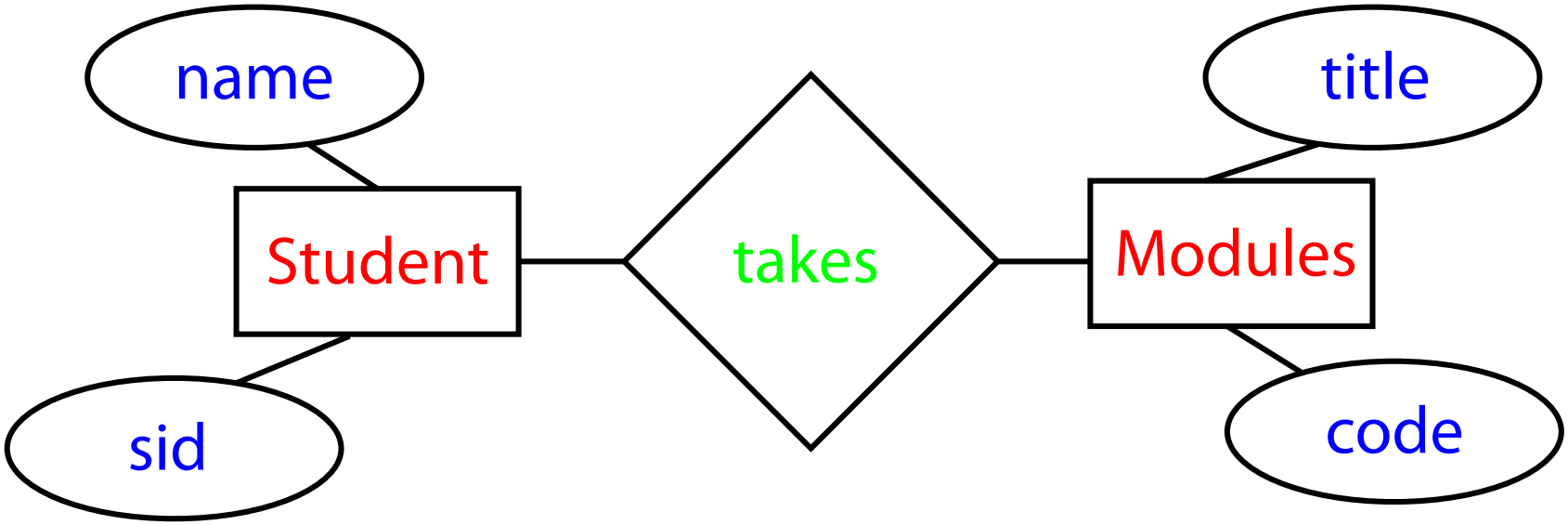
This lecture focuses on the ER diagram.

- An ER diagram is a pictorial representation of the information that can be captured by a database.
- Such a "picture" serves three purposes:
  - It allows database professionals to describe an overall design concisely yet accurately.
  - (Most of) it can be easily transformed into the relational schema.
  - It may help identifying some problems in a design.

An ER Diagram



- Basic Concepts: entites and their relationships along with the attributes describing them.



Entity

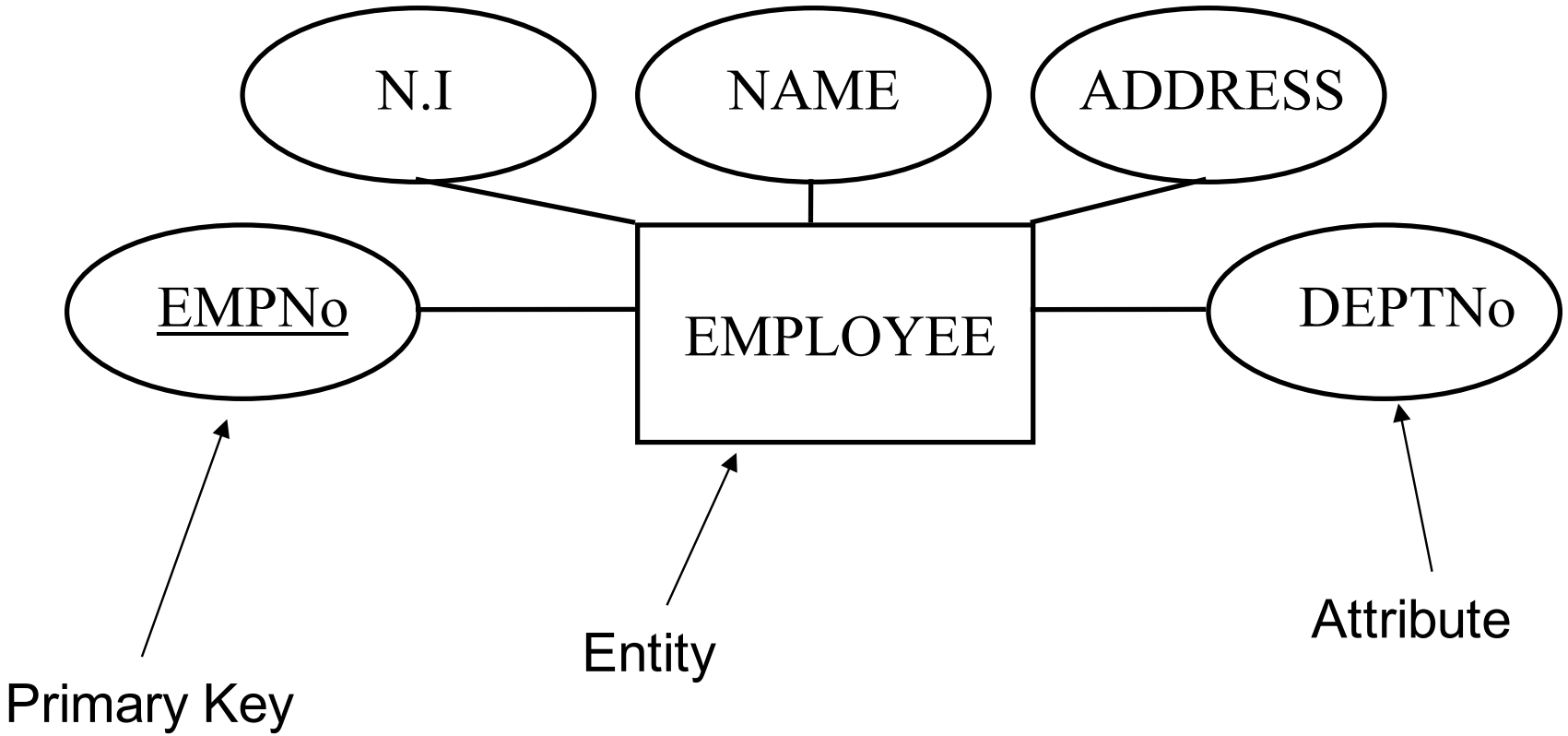
- An entity is a thing of interest in the real world.
- For example:

Person	EMPLOYEE, STUDENT, PATIENT
Place	STATE, REGION, COUNTRY
Object	MACHINE, BUILDING, CAR
Event	SALE, REGISTRATION, RENEWAL
Concept	ACCOUNT, COURSE, DEPARTMENT

- Instances of a particular entity:
  Gary Tam, Mark Jones are instances of Lecturer

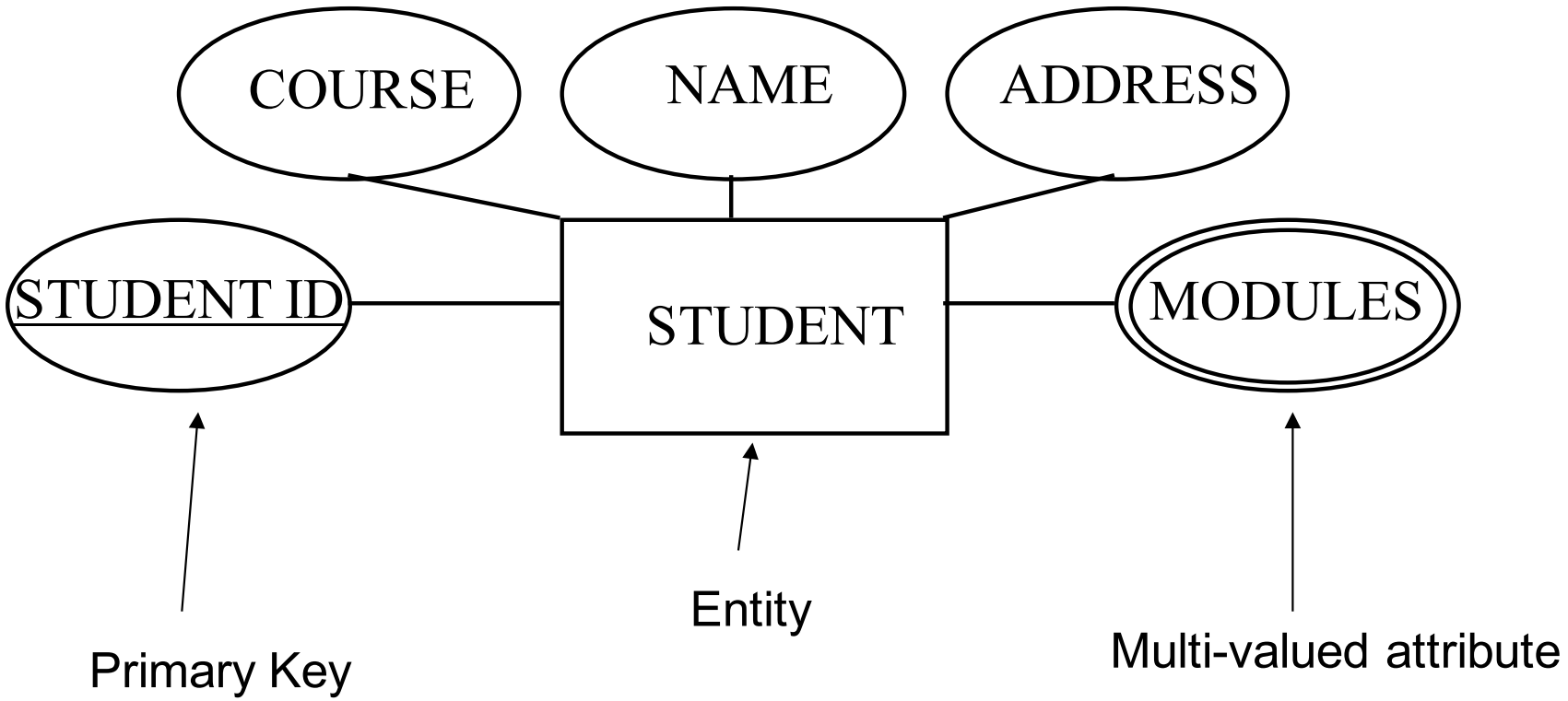
Attributes

- Each entity has a number of attributes associated with it.
- Attributes are facts, aspects, properties or details about an entity
- primary key attribute is underlined



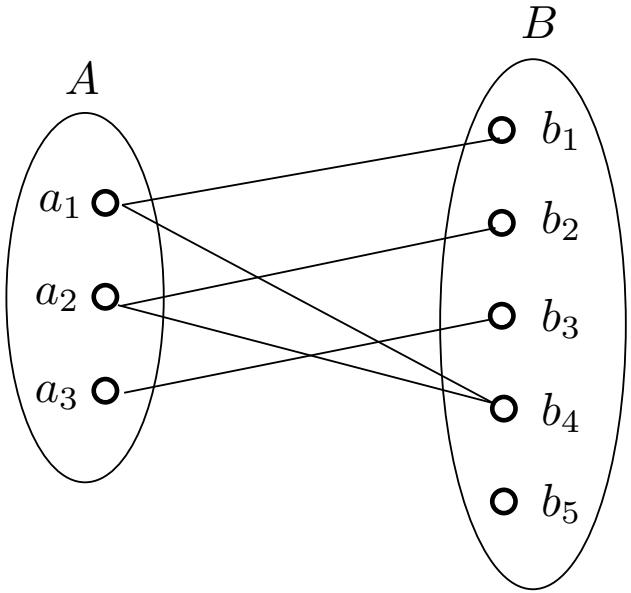
Multi-valued Attributes

- A single valued attribute will have a single value.
- A multi-valued attribute will have a set of values.
- multi-valued attribute is represented as double ellipse.



Relationship

- A instance of relation is an association between one or more entity instances.
- A relationship R is a collection of these instances of relation.



$$R = \{(a_1, b_1), (a_1, b_4), (a_2, b_2), (a_2, b_4), (a_3, b_3)\}$$

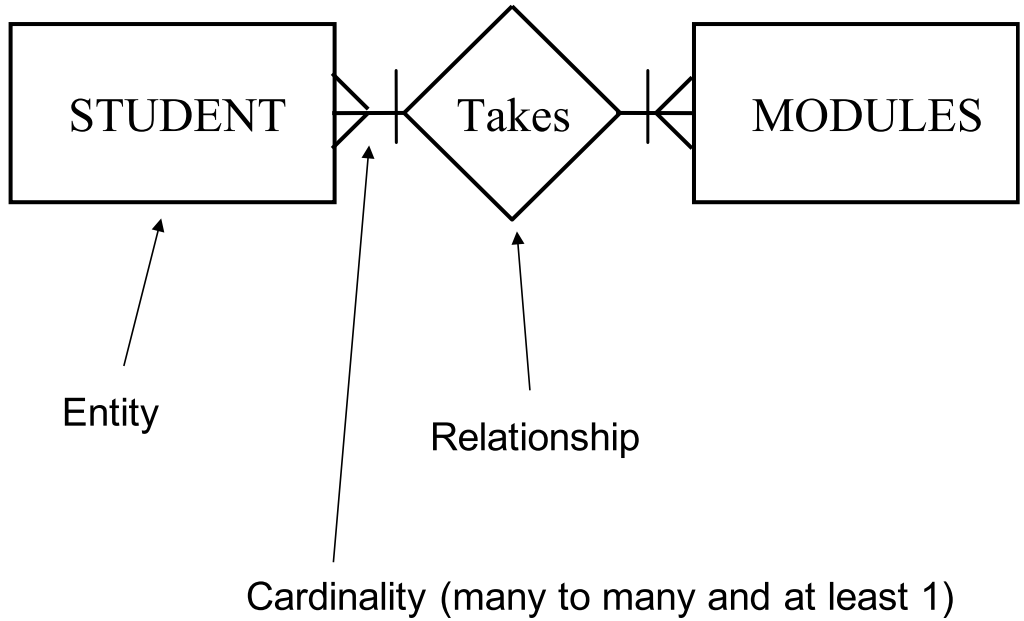
- Example:
  - each student takes several modules
  - each module is taught by a lecturer
  - each employee works for a single department

Relationship in ER

A relationship in ER diagram has the followings:

- name
- possibly some attributes
- degree
- cardinality and participation constraints

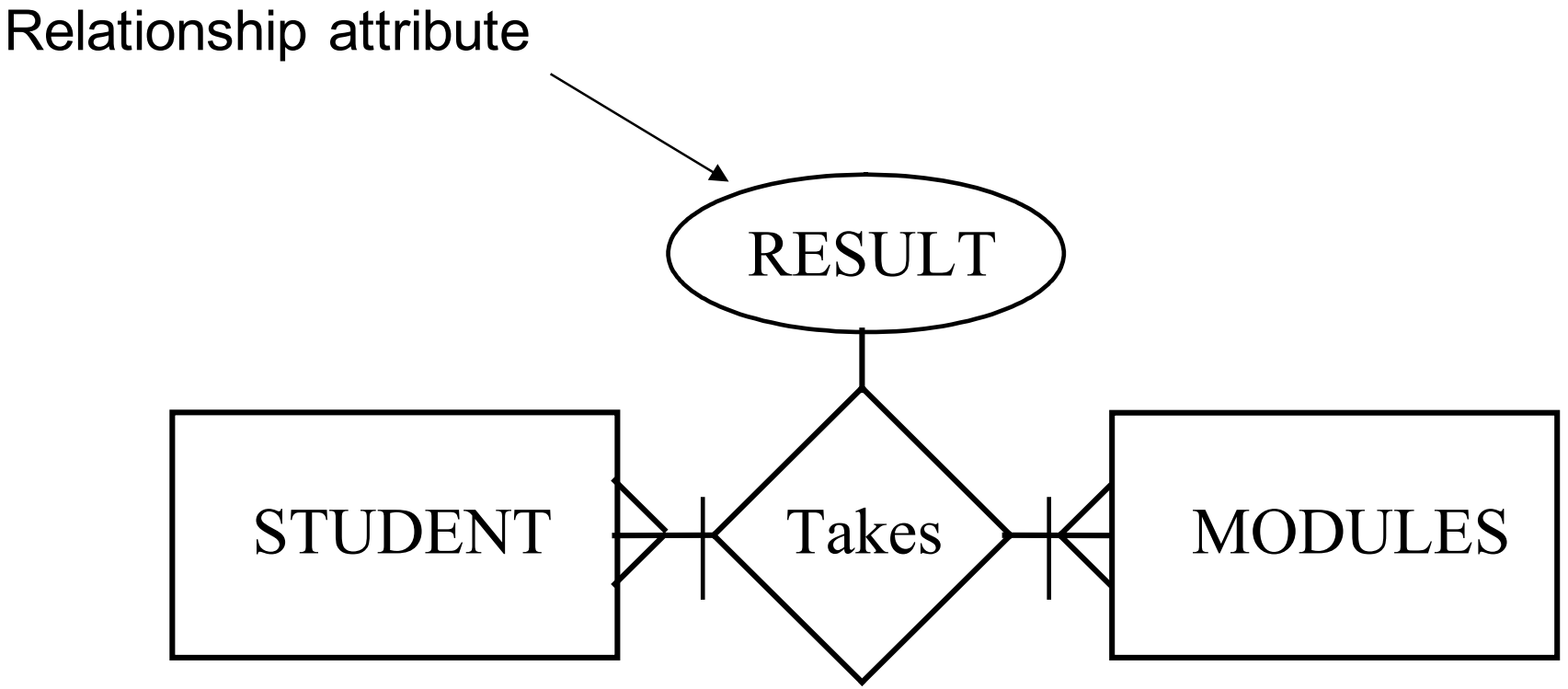
Example: each student takes several modules



Relationship Attributes

Attribute: RESULT

- not an attribute of student, why?
- not an attribute of module, why?

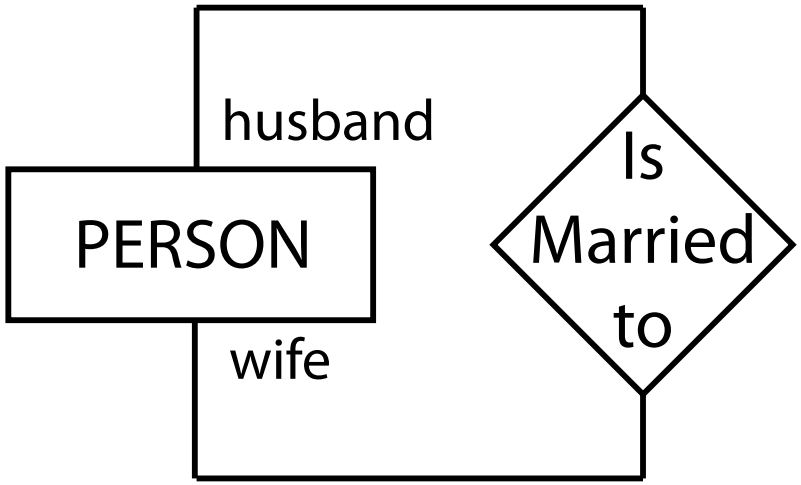


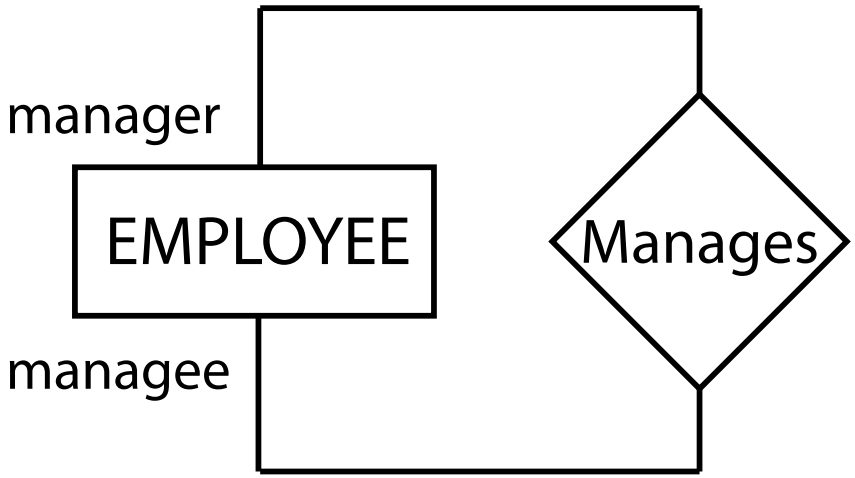
Degree of Relationship

- A degree - the number of entities that participate
- Example:

Unary Relationship:

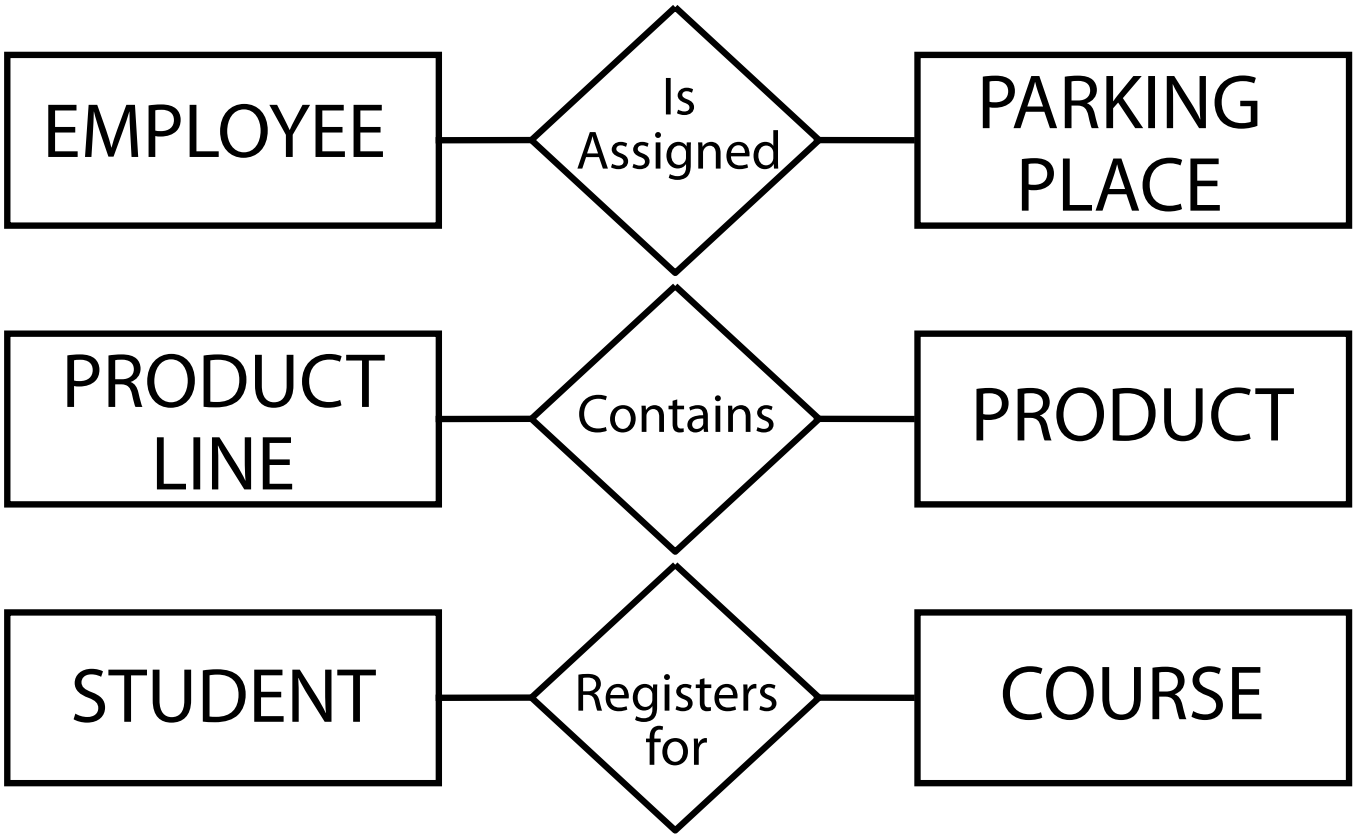
- usually specify the roles





Degree of Relationship

Binary Relationship (most common)



There is relationship with higher degree, e.g. ternary relationship, which is not required in this course.

The next few slides will mainly discuss binary relationships.
We can impose two types of constrains.

Cardinality constraint:

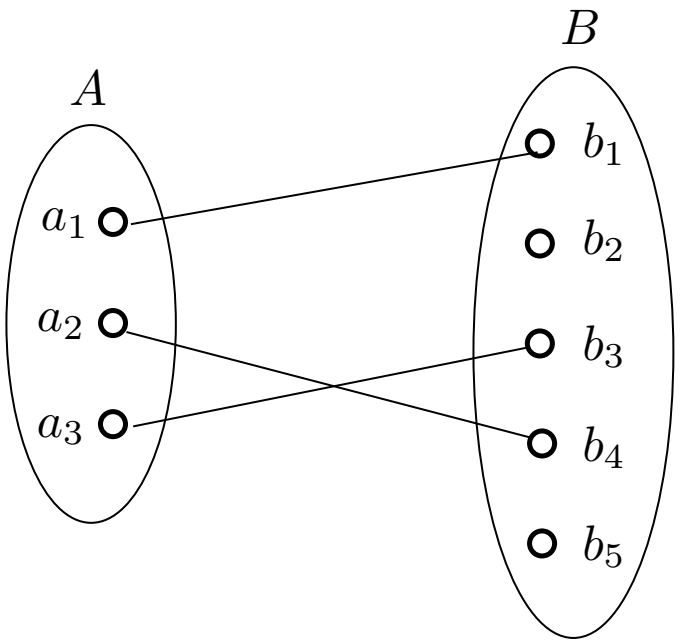
- One-to-one
- One-to-many (or conversely, many-to-one).
- Many-to-many

Participation constraint:

- Total
- Partial

One-to-One

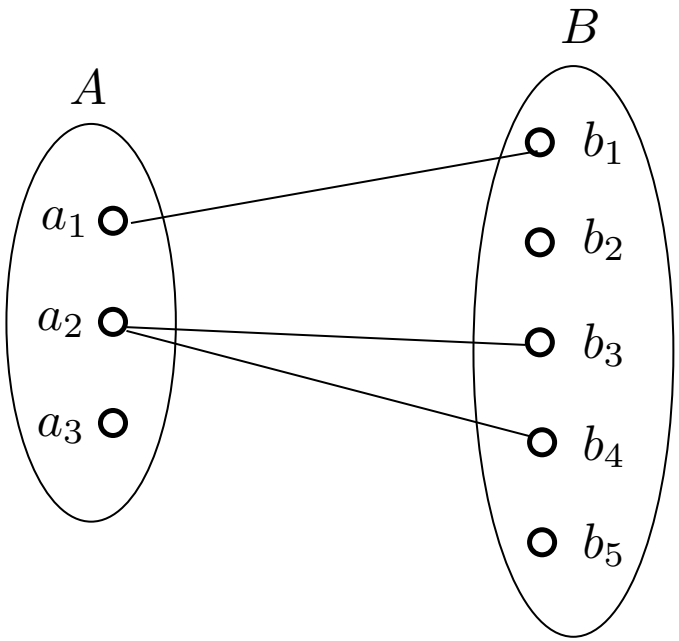
A relationship R between entity A and B is one-to-one if every instance in entity A and B can participate in at most one relationship instance in R.



Example: Husbands and wifes

One-to-Many

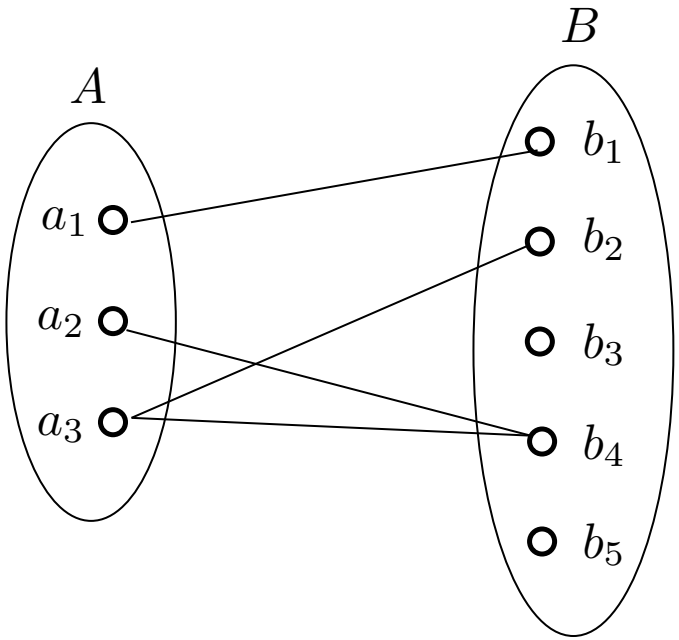
A relationship R between entity A and B is one-to-many if every instance in entity A can participate in any number of relation instance in R, but an instance in entity B can participate in at most one relation instance in R.



Example: Parents and Children.
Many-to-one is defined analogously.

Many-to-Many

A relationship R between entity A and B is many-to-many if every instance in entity A and B can participate in any number of relation instance in R.



Example: Students and modules.

Cardinality constraint:

- One-to-one
- One-to-many (or conversely, many-to-one).
- Many-to-many

Participation constraint:

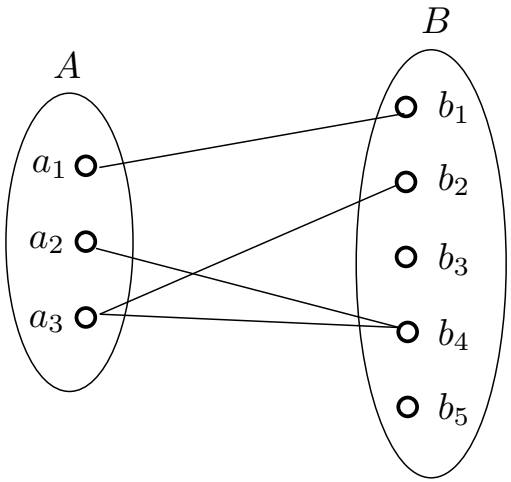
- Total
- Partial

Next we talk about participation constraints.

Total/Partial Participation

- Let R be the relationship between entity A and B.
- The participation of A is total if every instance of A must participate in at least one relation instance in R.
- Otherwise, the participation of A is partial.
- Likewise, we can define total or partial participation of B.

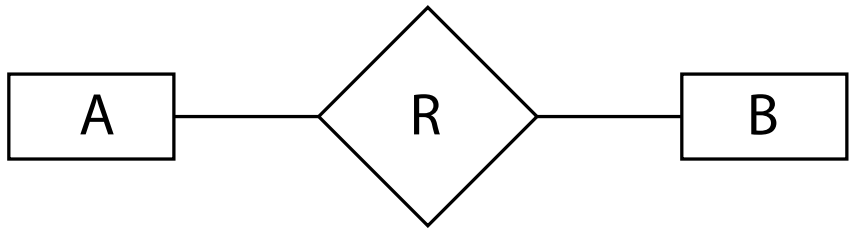
Example: Participation of A is total, that of B is partial.



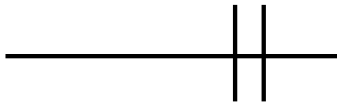
Example: Patents and Professors

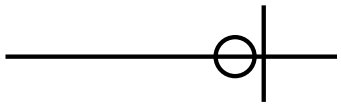
Relationship in ER Diagrams

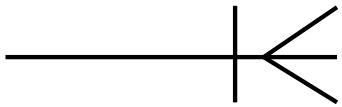
Basic representation of a binary relationship R between entity A and B

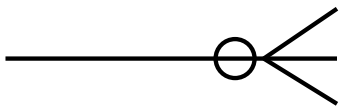


Symbols:


... to one with total participation


... to one with partial participation

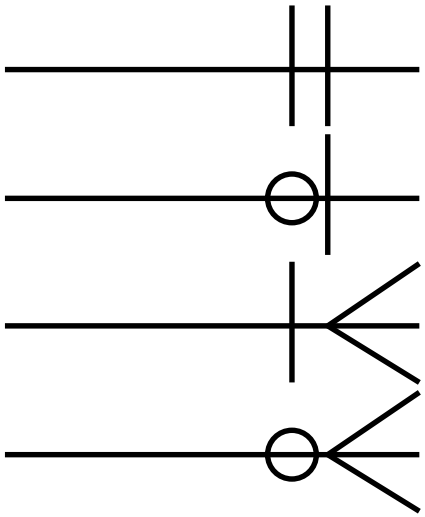

... to many with total participation


... to many with partial participation

Tips and Tricks

1. Cardinality and participation symbols are drawn at opposite side of relationship. Look across.
2. Roughly follows a certain grammar:

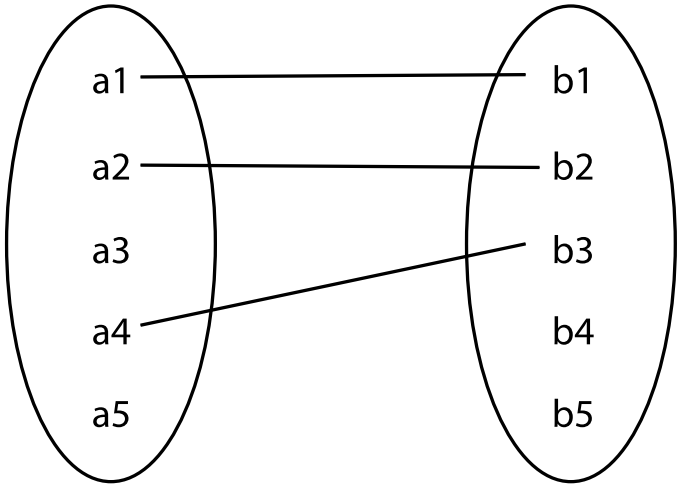
<entity><min cardinality><relation><max cardinality><entity>
min cardinality : may (partial) / must (total)
max cardinality : at most one / one (or more) / many

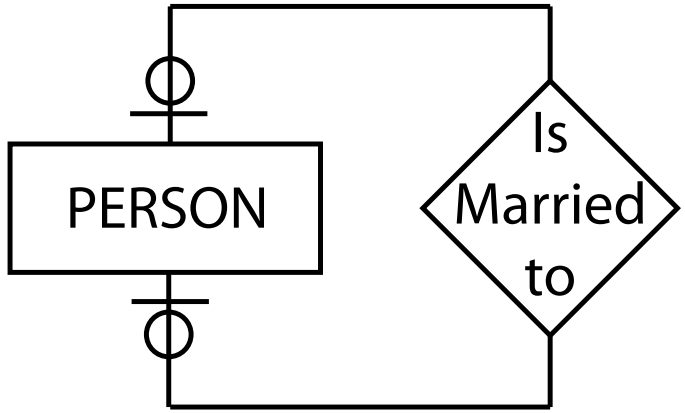


- circle - zero (min)
- | - one (min/max)
- branch - many (max)

Mapping and Participation

One-to-One:



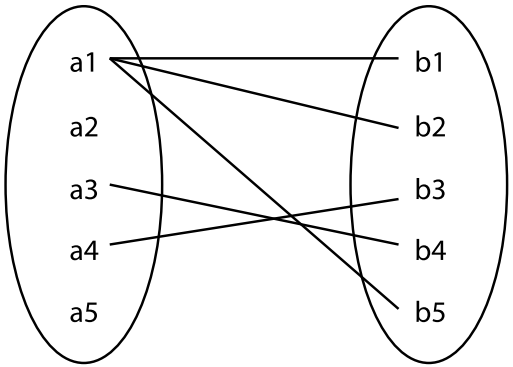


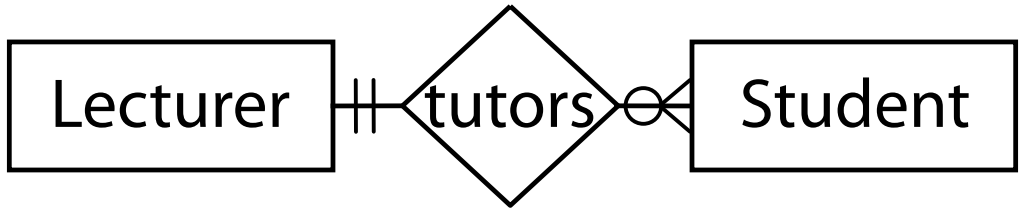
Read as:
A person may not marry any one. A person (husband/wife) can marry to at most one person (as wife/husband).

Or simply: A person may marry to one another person.

Mapping and Participation

One-to-Many:



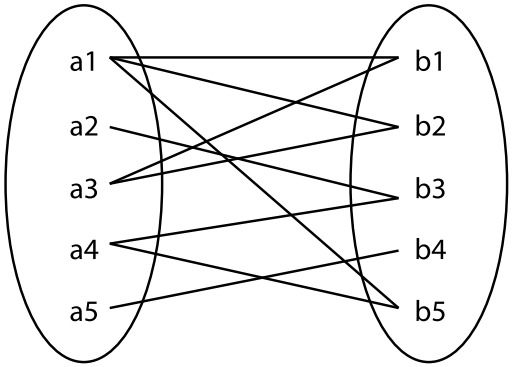


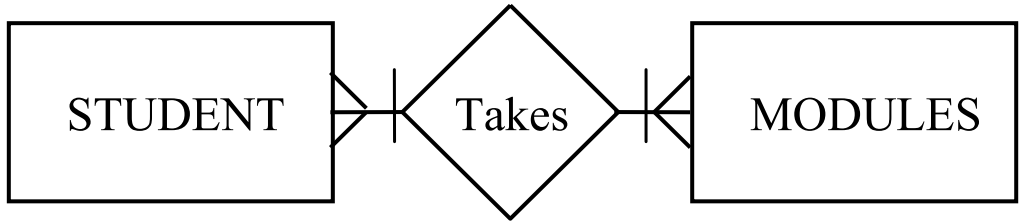
Read as:
Lecturers can tutor many students, but some lecturers may not tutor any students at all. Every student has 1 lecturer as a tutor.

Or simply: A lecturer may tutor many students. A student has exactly one/must have one lecturer as tutor.

Mapping and Participation

Many-to-Many:





From this diagram we read:
Each student takes many modules (and must take at least 1).
Each module is taken by many students (and must be taken by at least 1).

Or simply: Each student must take one (or more) modules. Each module must have one (or more) students.

Subtypes and Inheritance

Several types of employees

- Hourly employee: emp#, name, address, date hired, hourly rate
- Salaried employee: emp#, name, address, date hired, annual rate, share option
- Contract employee: emp#, name, address, date hired, contract#, daily rate

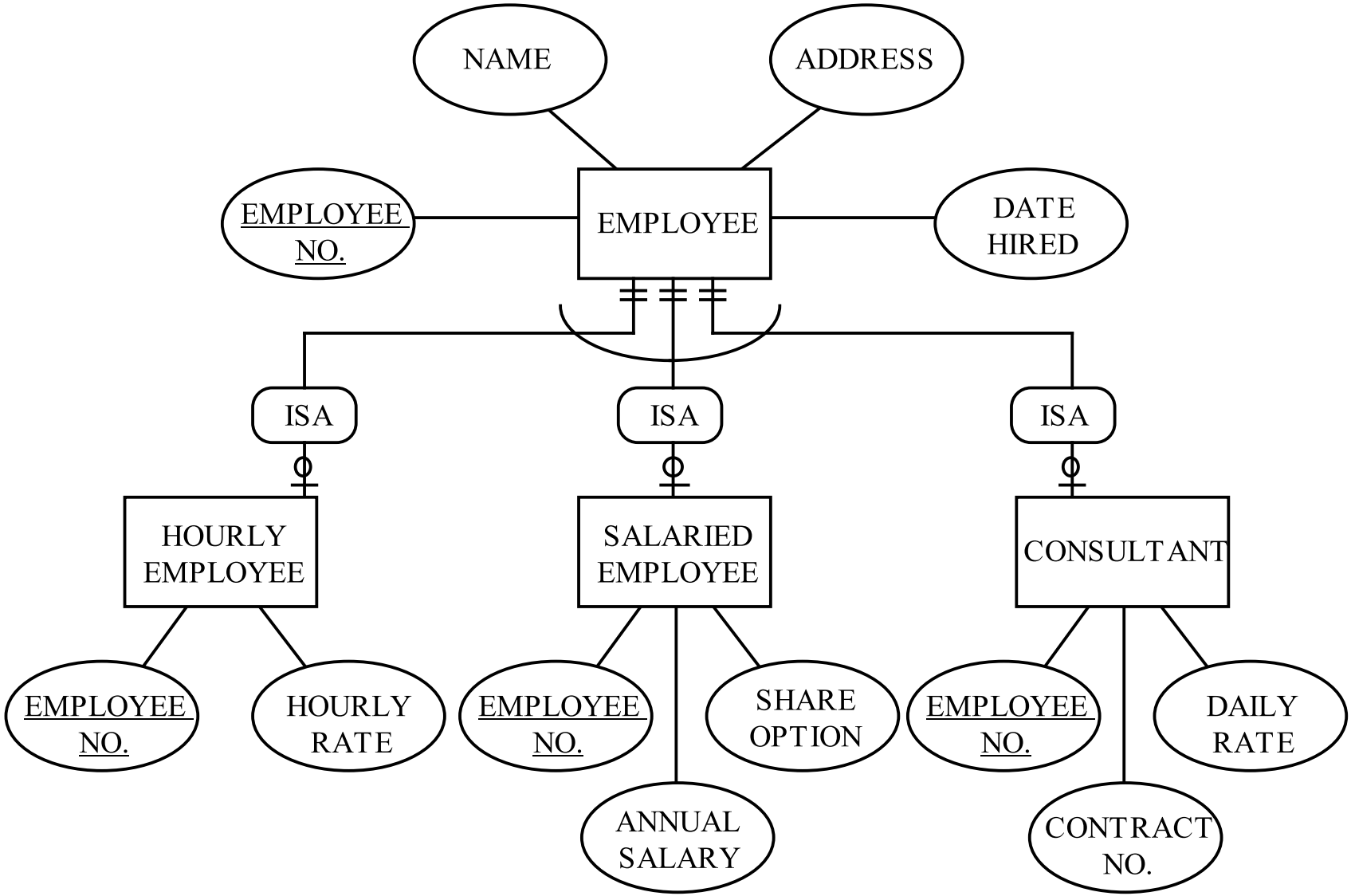
Options:

1. Define a single entity EMPLOYEE using all possible attributes.
2. Define separate entity for each type of EMPLOYEE.
3. Define a supertype EMPLOYEE with subtypes: HOURLY EMP, SALARIED EMP, CONSULTANT.

Which option will you adopt? Why?

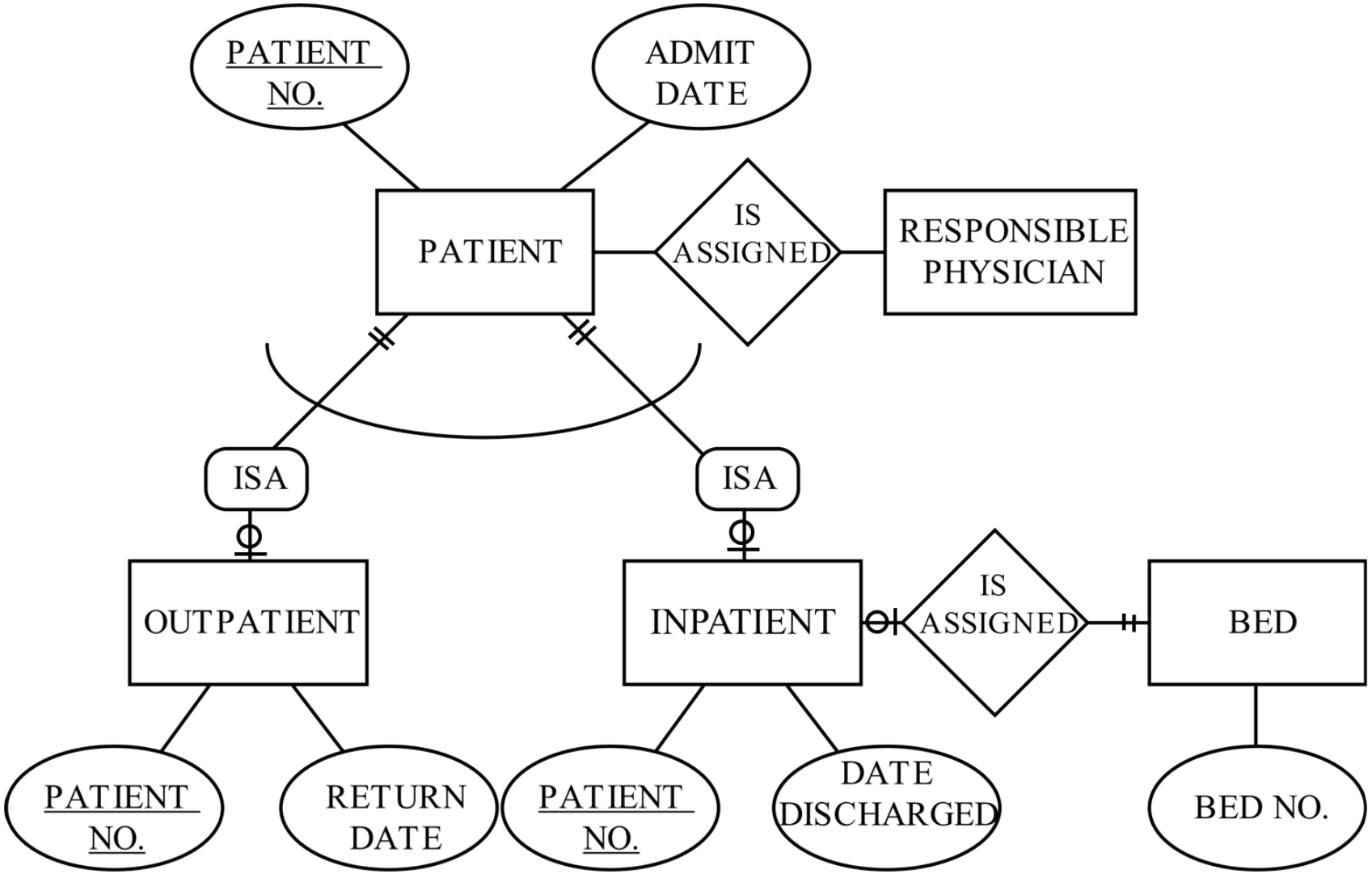
ISA relationship

Curved line in the diagram denotes an exclusive relationship - exactly one is required for each instance of EMPLOYEE.



Subtypes and Relations

The subtypes can also have relations that only apply to them.



Non-Exclusive Subtype

An instance can have more than one subtype

Example: 4x4 Lorry requires

- 4x4 subtype
- Lorry subtype

Motorbike?

- Ok $\rightarrow$ allows us to data modelling even part of the business is not developed / understood.