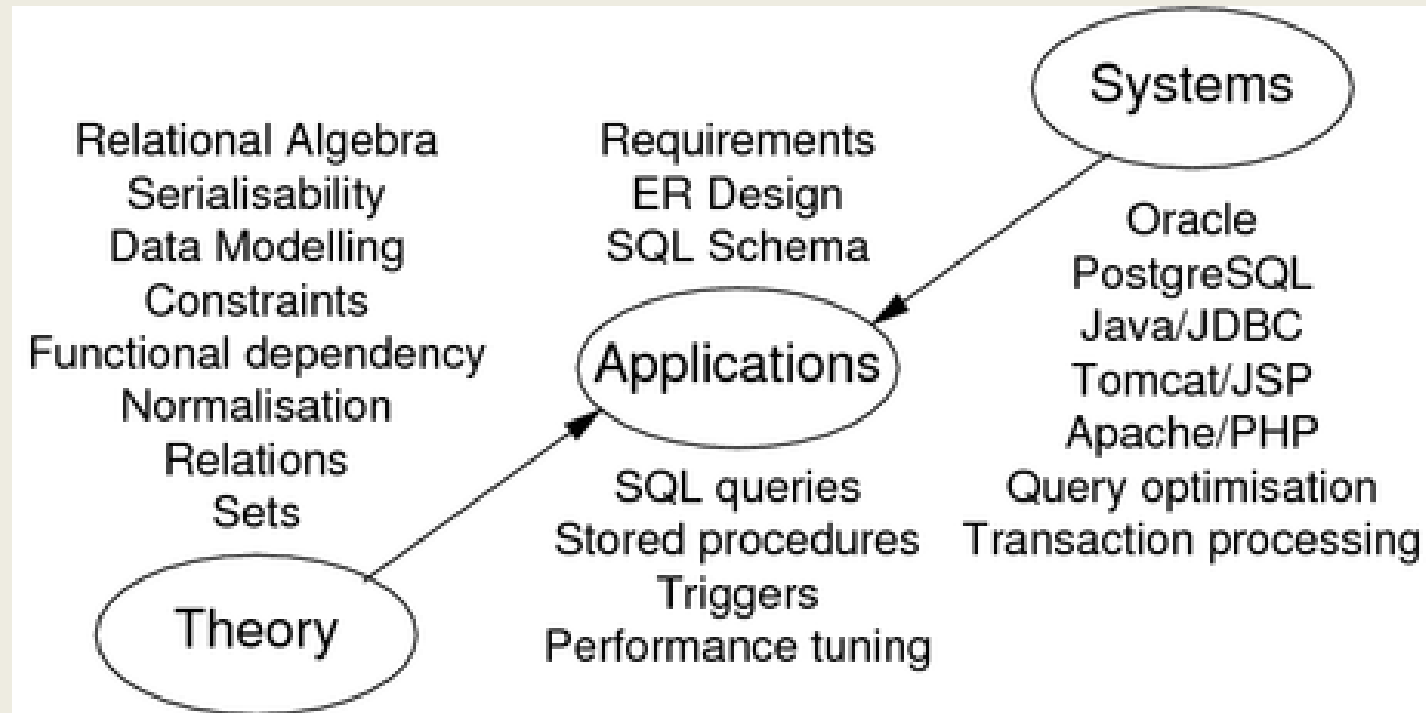


# Conceptual Database Design

Textbook: chapter 3 & 4

# 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
- implement the data model as relational schema
- implement transactions via SQL and procedural programming languages
- construct an interface to these transactions
- 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 Commonwealth Bank)
- describe *constraints* on data  
(e.g., 7-digit IDs)

Data modelling is a *design* process

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

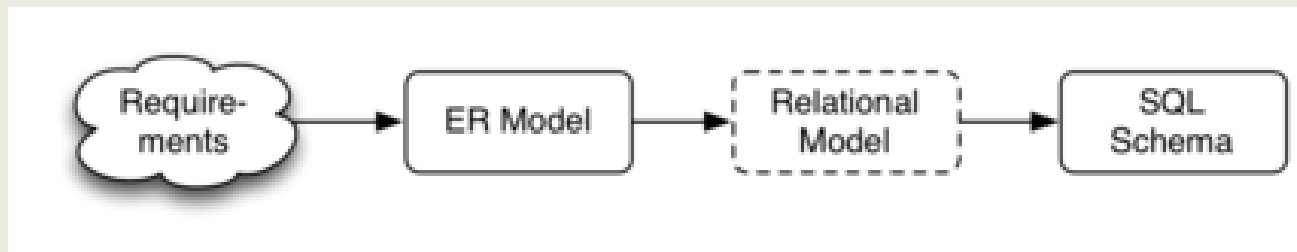
# Data Modelling

---

Kinds of data models:

- *logical*: abstract, for conceptual design, e.g., ER, ODL (object data language)
- *physical*: record-based, for implementation, e.g., relational

Strategy: design using abstract model; map to physical model



# Some Design Ideas

---

Consider the following while we work through exercises:

- start simple ... evolve design as problem better understood
- identify objects (and their properties), then relationships
- *keywords in requirements suggest data/relationships*  
*(rule-of-thumb: nouns → data, verbs → 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 (ER) Model

---

The world is viewed as **a collection of inter-related entities**.

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

- *attribute*: data item describing a property of interest
- *entity*: collection of attributes describing object of interest
- *relationship*: association between entities (objects)

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

Lecture notes use simple notations. We can use this as ‘COMP9311 standard’.



# Entity-Relationship Model

---

The Entity-Relationship (ER) model is a high-level conceptual data model (Chen in 1966).

ER is used mainly as a design tool.

# Entity and Attributes

---

*Entities* represent things in the real world.

*Attributes* describe properties of entities.

Attributes may be

- simple(atomic) e.g. sex = 'Female', or
- composite e.g. name consists of title (Dr), Initials (C.C.), family name (Chen).

# Entity and Attributes<sub>(cont)</sub>

---

Each entity has values for each attribute.

Attributes may be

- *single-valued* e.g. student number, name, or
- *multivalued* e.g. keywords = neural networks, computer graphics, databases.

# Entity and Attributes<sub>(cont)</sub>

---

An attribute can have a null value if, for example:

- there is no suitable value e.g. a student may have no interests: keywords = NULL
- the true value is not known e.g. the marriage date of a person is not known: marriage date = NULL.

A derived attribute is one whose value can be derived from other attributes and entities. e.g. number of students.

# Entity and Attributes<sub>(cont)</sub>

---

An *entity* type is a set of entities with the same attributes.

It is described by an *entity* schema: a name and a list of attributes.

The set of individual entity *instances* at a particular moment in time is called an extension of the entity type.

# Entity and Attributes<sub>(cont)</sub>

---

Schema (Intension)	RESEARCHER Name, Payroll_no, No_of_students, Keywords	DEPARTMENT Name
Instances (Extension)	(Dr C.C. Chen, 230-0013, 3, Neural Networks) (Dr R. Wilkinson, 231-0091, 1, Databases)	Computer Science Psychology Management

# Entity and Attributes<sub>(cont)</sub>

---

An entity type usually has a *key*: a set of attributes that uniquely identifies an entity. For example:

- {payroll number} is a key of RESEARCHER,
- {name} is a key of DEPARTMENT.

There may be more than one possible key.

An important constraint is the **key constraint**: in any extension of the entity type, there cannot be two entities having the same values for their key attributes.

# Key

---

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

- natural (e.g., name+address+birthday) or artificial (e.g., SSN)

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

*Primary key* = candidate key chosen by DB designer

Keys are indicated in ER diagrams by underlining



# Entity and Attributes<sub>(cont)</sub>

---

Composite and multi-valued attributes:

- We can describe schema with composite attributes using ()'s and with multi-valued attributes using {}'s. e.g.

# Entity and Attributes<sub>(cont)</sub>

---

## CAR

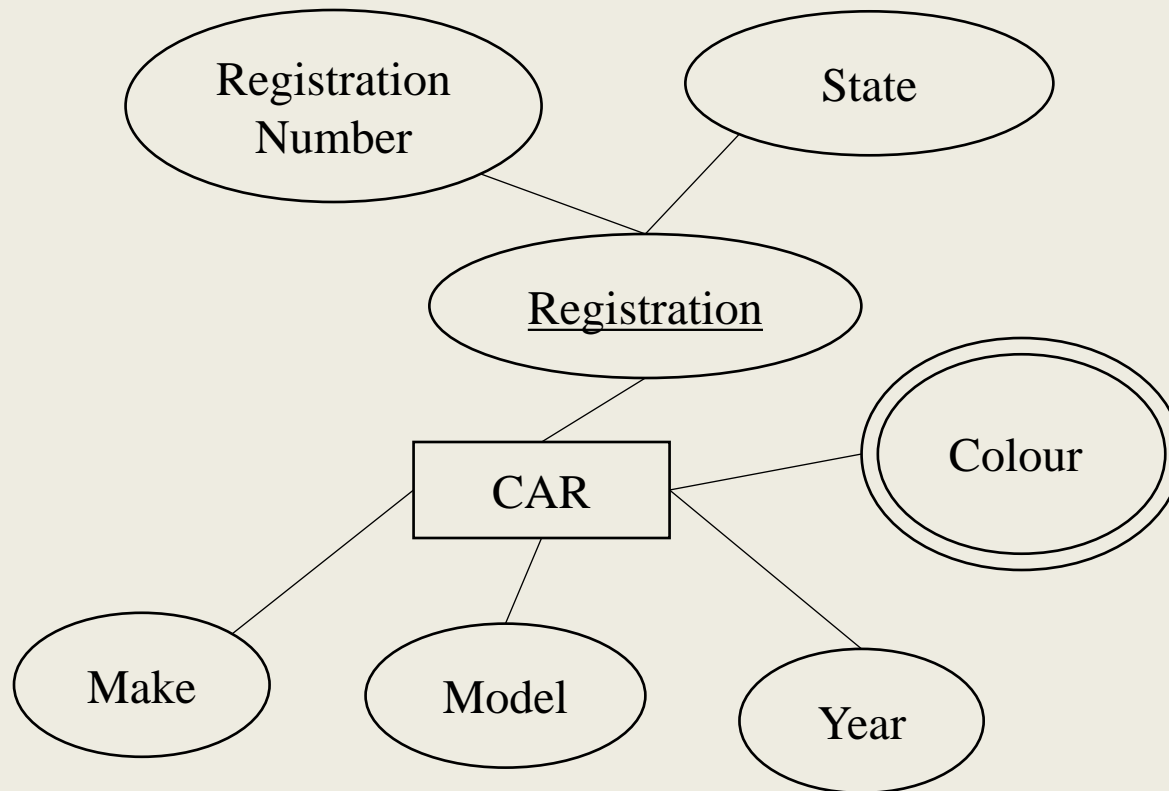
**Registration(Registration No, State), Make, Model, Year, {Colour}**

((ARQ) 595, Vic), Datsun, 120Y, 1972, {green})  
((8HR) 696, WA), Mazda, 929, 1979, {grey, black})

# Entity and Attributes<sub>(cont)</sub>

---

Entities and their attributes can also be described with Entity-Relationship Diagrams (ERDs). e.g.



# Relationships

---

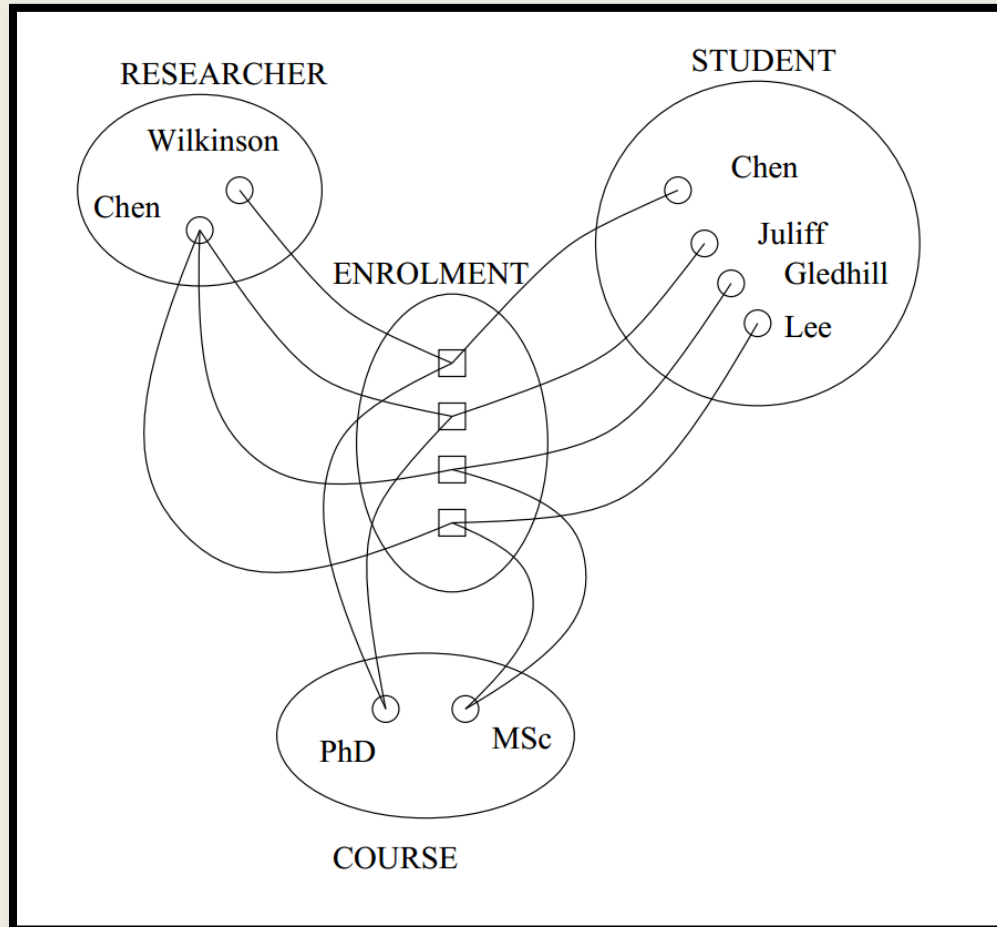
A *relationship* represents an association between things / entities.

The *degree* of a relationship is the number of participating entity types. For example,

- ENROLMENT could be a ternary (degree 3) relationship between RESEARCHER, STUDENT and COURSE.

# Relationships<sub>(cont)</sub>

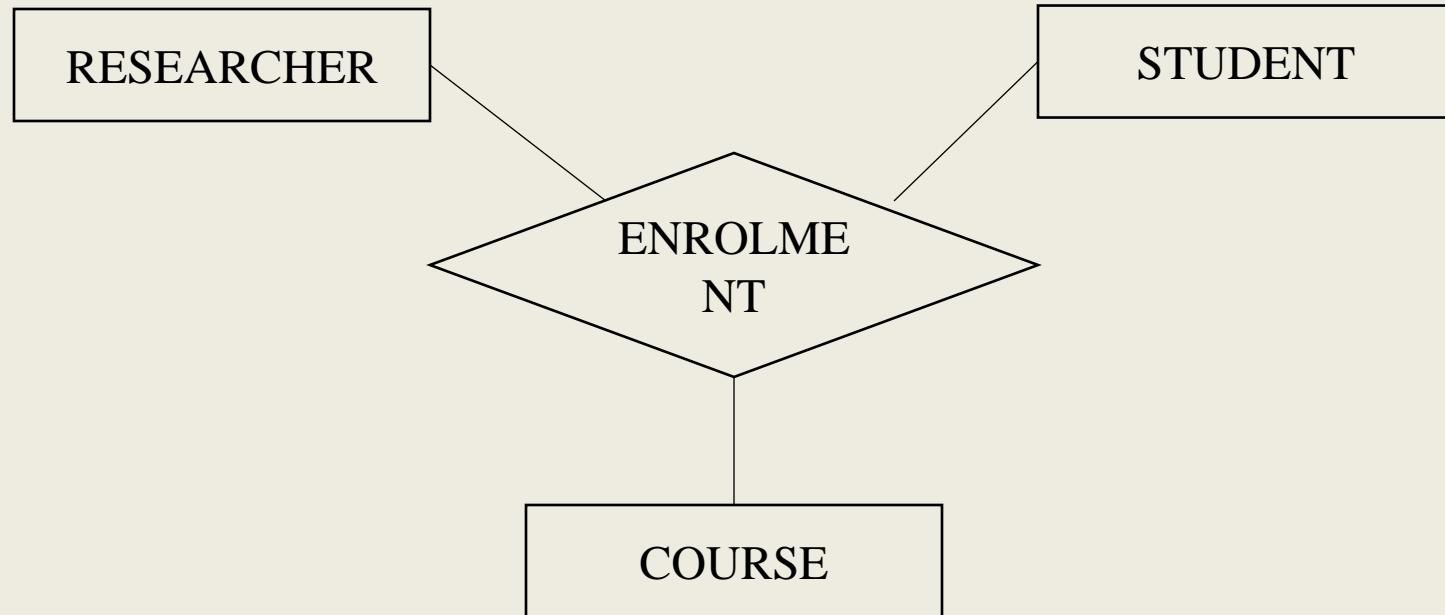
We can illustrate this using an occurrence diagram:



# Relationships<sub>(cont)</sub>

---

Entities and their relationships can also be represented using Entity-Relationship diagrams:



# Relationships<sub>(cont)</sub>

---

Each entity type that participates in a relationship plays a particular *role* in the relationship.

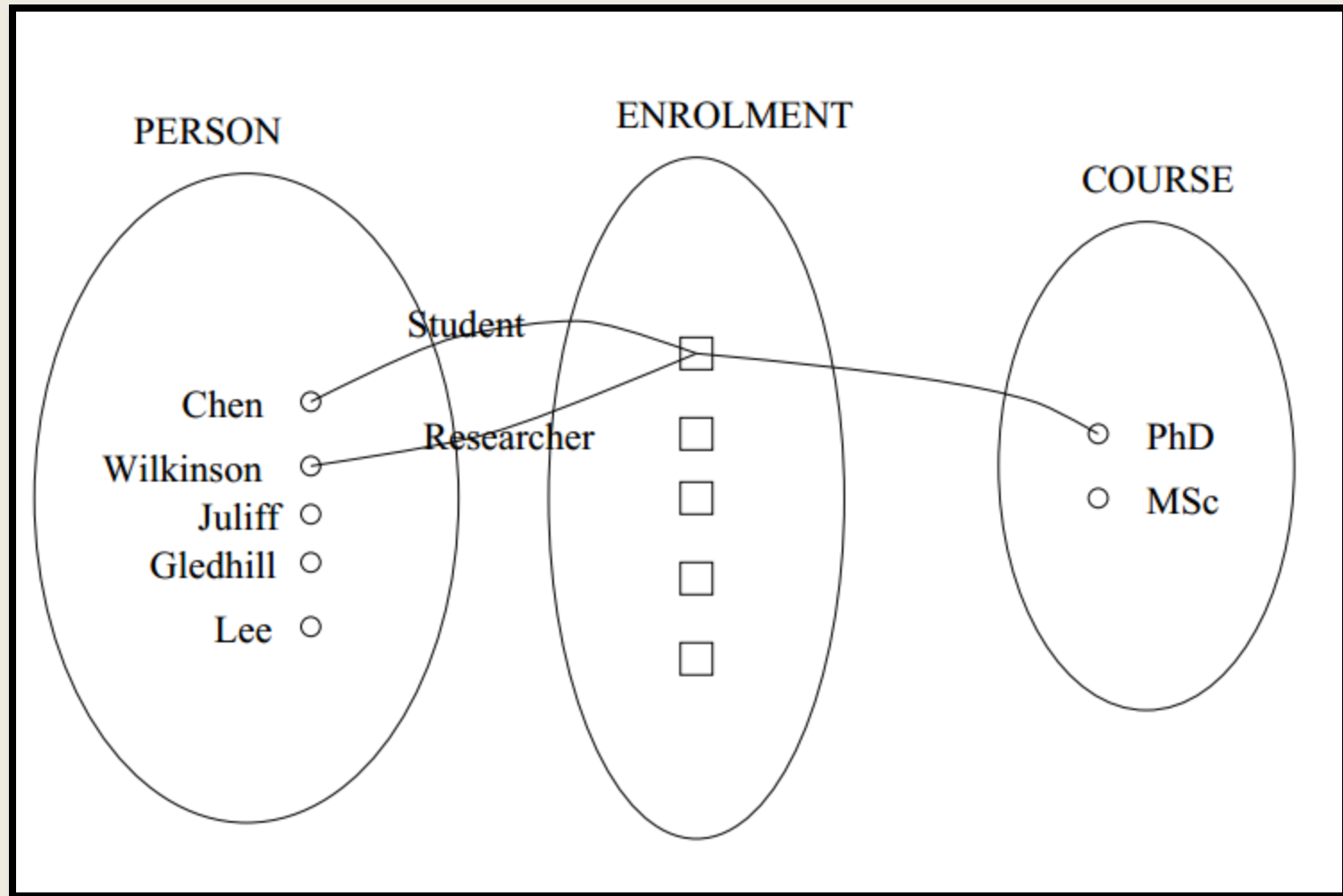
An entity type can play

- different roles in different relationships, or
- more than one role in a relationship.

A role name can be used to distinguish these.

For example, ENROLMENT could be a relationship between PERSON(as researcher), PERSON(as student) and COURSE as in the diagram below:

# Relationships<sub>(cont)</sub>

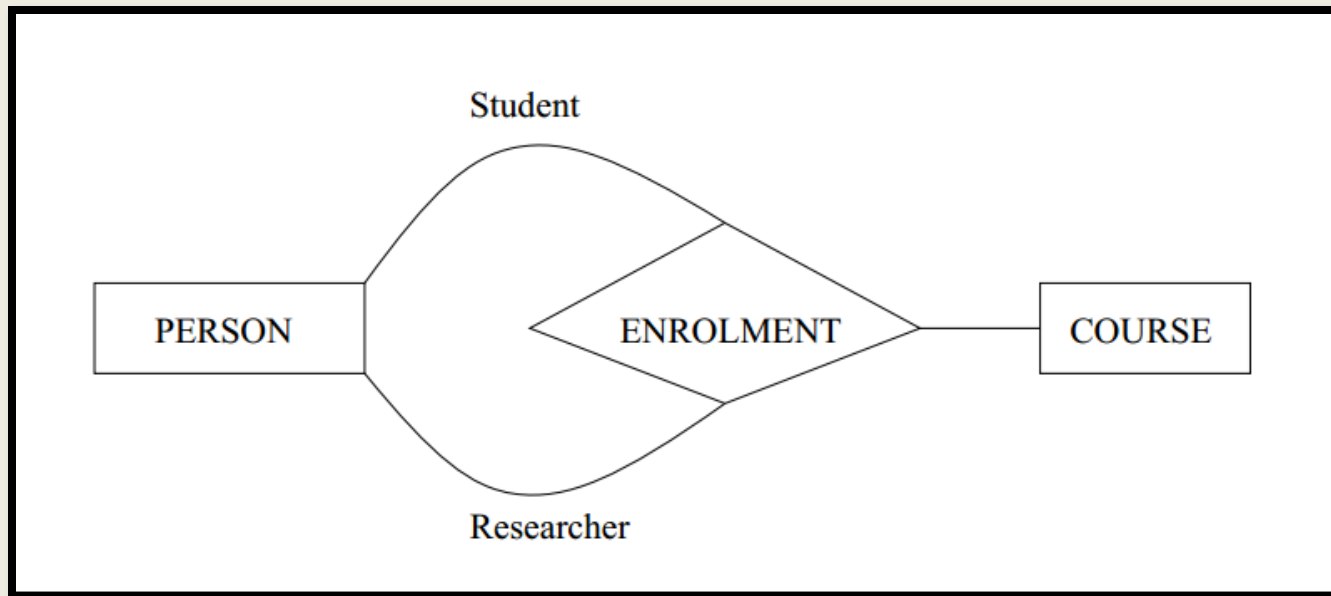




# Relationships<sub>(cont)</sub>

---

Or, using an ERD:



This is called a recursive relationship.

# Weak entity types

---

Some entity types do not have a key of their own.

Such entity types are called weak entity types.

Entities of a weak entity type can be identified by a partial key and by being related to another entity type - *owner*.

The relationship type between a weak entity type to its owner is the *identifying relationship* of the weak entity type.

# Weak entity types<sub>(cont)</sub>

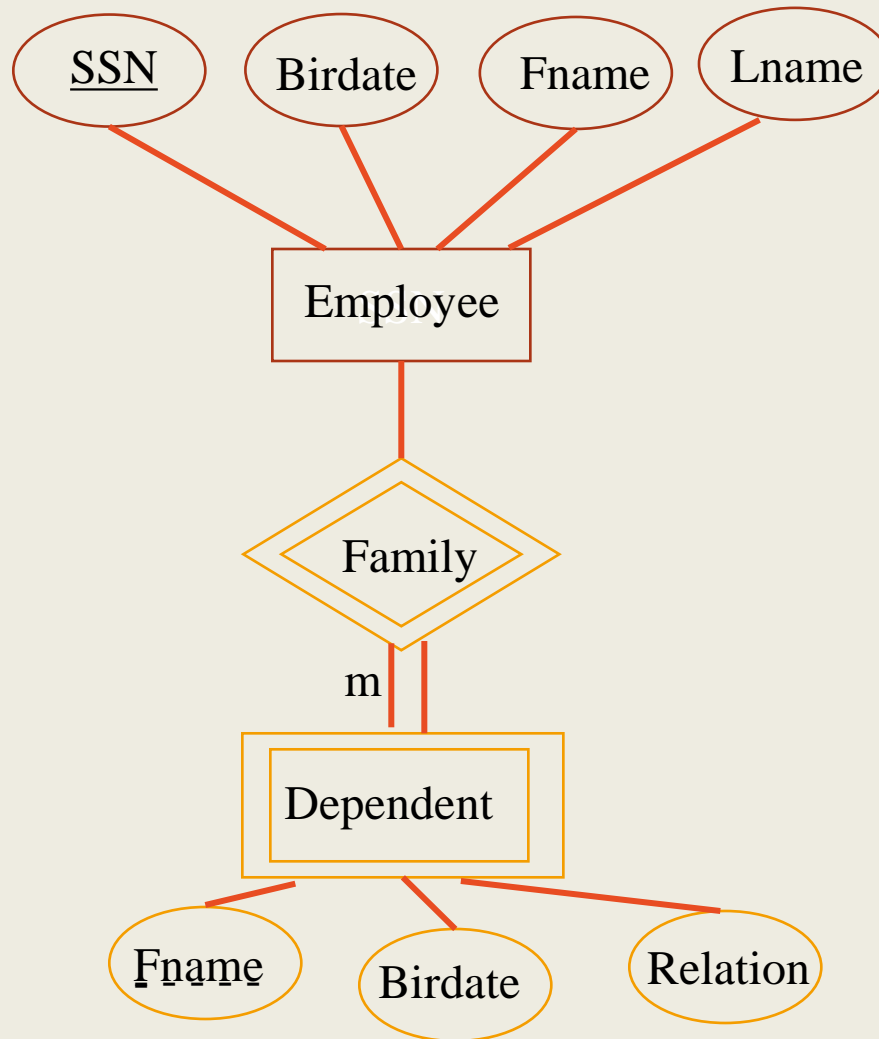
---

For example, a TAX PAYER entity may be related to several DEPENDENT, identified by their names.

In this example, DEPENDENT is called a weak entity, {Name} is a partial key for it. The identifying relationship between DEPENDENT and TAX PAYER is IS DEPENDENT OF. TAX PAYER is said to *own* DEPENDENT.

# Weak entity types<sub>(cont)</sub>

---



# Constraints on relationship types

---

Relationship types usually have certain constraints that limit the possible combinations of entities participating in relationship instances.

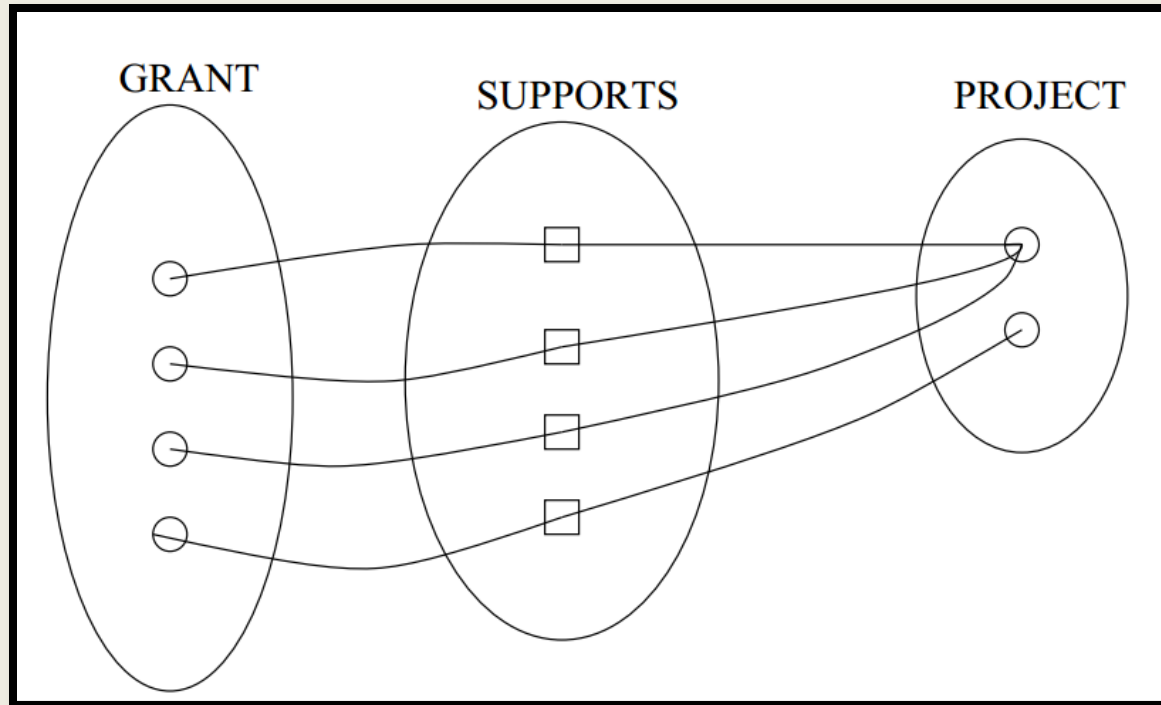
They should reflect the correct factors

*Cardinality ratio constraint*: specifies the number of relationship instances an entity can participate in.

Example: A research grant supports only one research project, but a research project may be supported by many grants. PROJECT:GRANT is a 1 : N relationship.

# Constraints on relationship types<sub>(cont)</sub>

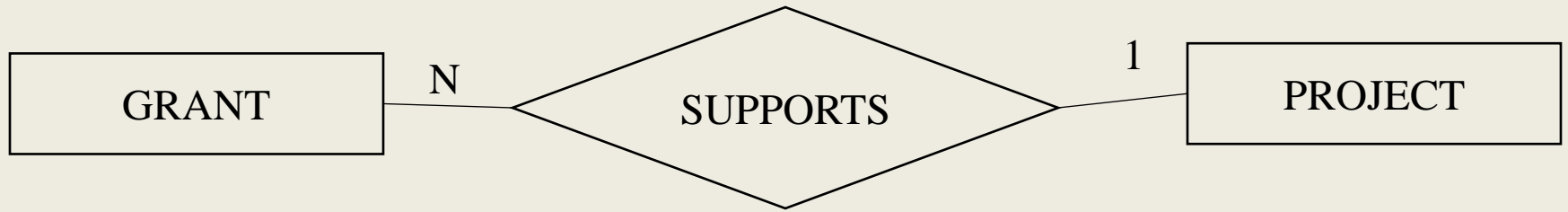
This is illustrated in the occurrence diagram below:



# Constraints on relationship types<sub>(cont)</sub>

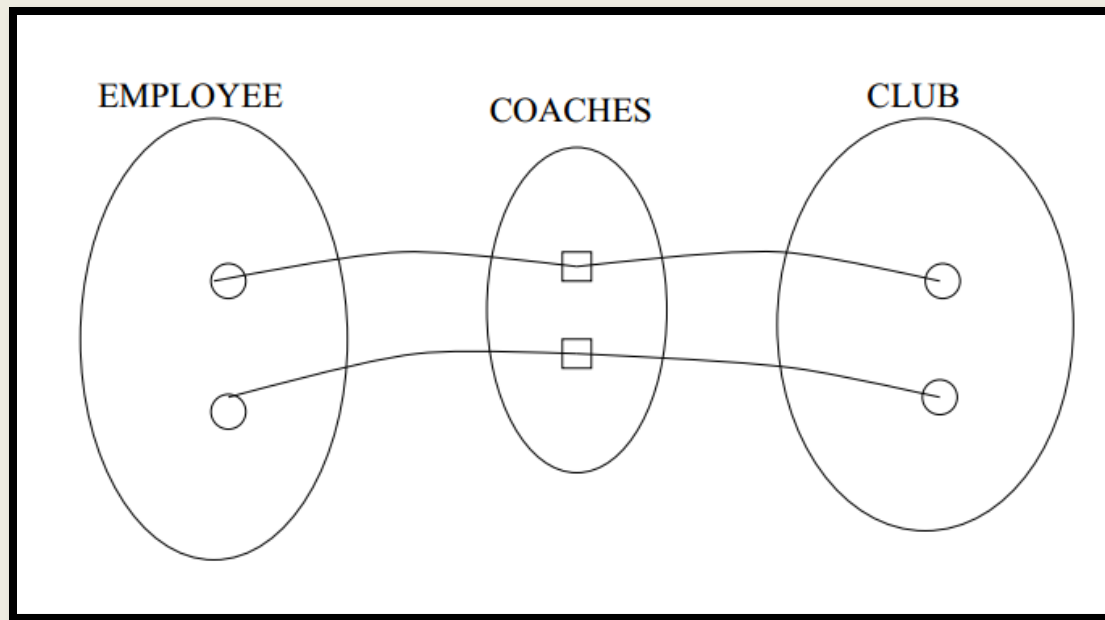
---

We can also show this in an ERD:



# Constraints on relationship types<sub>(cont)</sub>

Example: Consider a database of AFL (here substitute your favourite team sport) statistics. The relationship of head coaches to clubs is an example of a 1 : 1 relationship.





# Constraints on relationship types<sub>(cont)</sub>

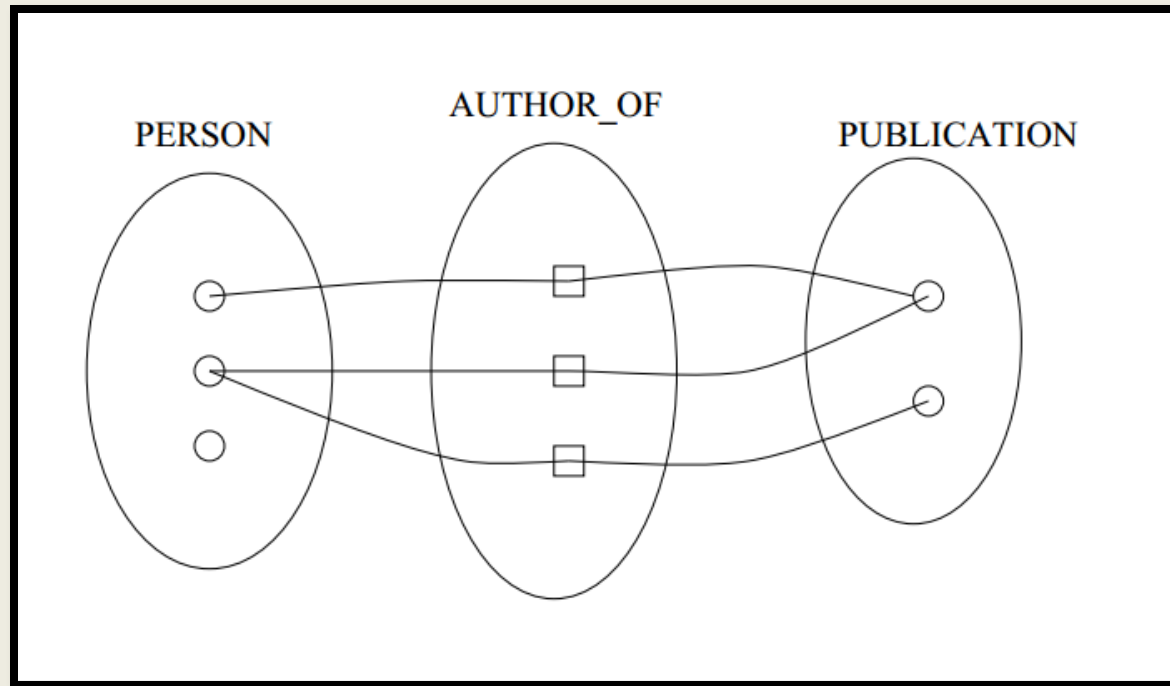
---

With an ERD:



# Constraints on relationship types<sub>(cont)</sub>

Example: An example of an N : M relationship is authorship of publications:



# Constraints on relationship types<sub>(cont)</sub>

---

The equivalent ERD:



# Constraints on relationship types<sub>(cont)</sub>

---

Another kind of constraint that can be represented using the ER model is a

- *Participation constraint*: participation of an entity in a relationship can be:
  - *total*: every entity must participate e.g. every publication has an author.
  - *partial*: not necessarily total. e.g. not every person has publications.

# Constraints on relationship types<sub>(cont)</sub>

---

This can be shown with an ERD like the one below:



# Attributes of relationship types

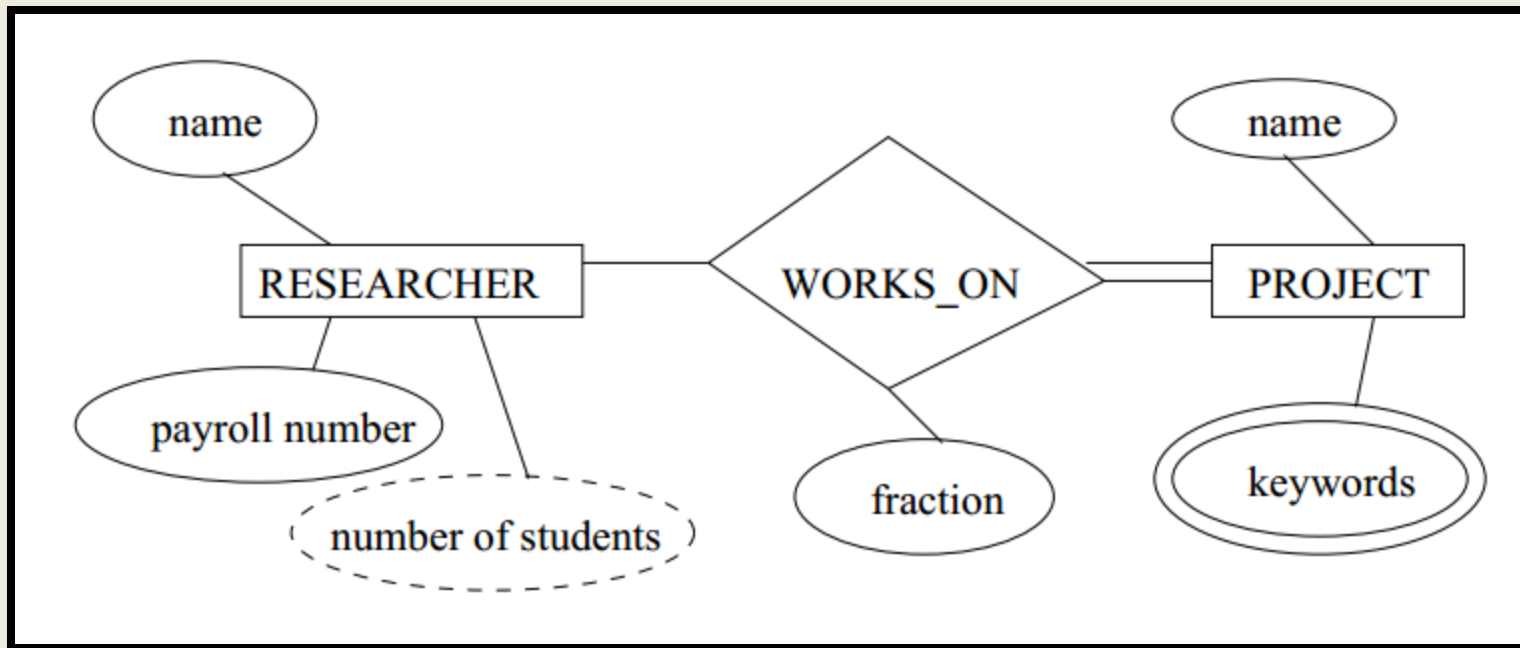
---

Relationship types can have attributes – for example,

- a researcher may work on several projects. The fraction of her time devoted to a particular project could be an attribute of the WORKS ON relationship type.

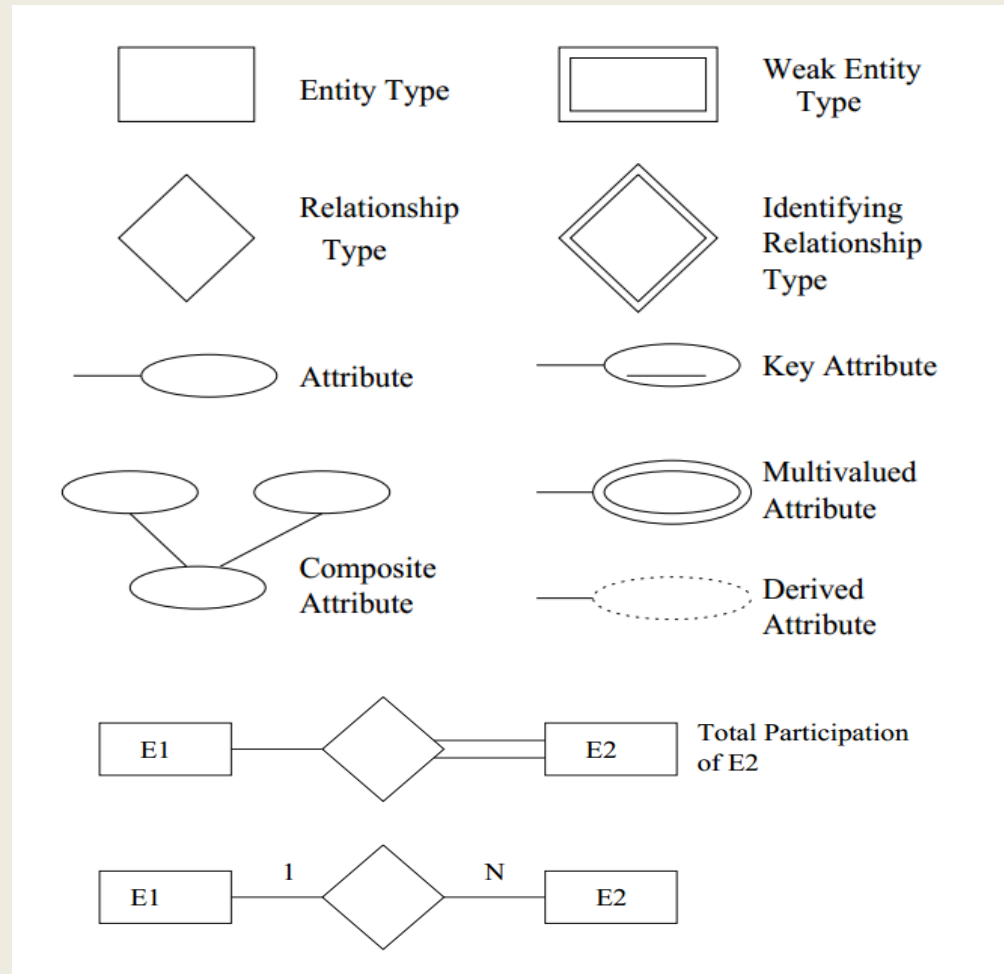
This can be shown in an ERD as below:

# Attributes of relationship types<sub>(cont)</sub>



# Attributes of relationship types<sub>(cont)</sub>

The notation used for ERDs is summarised in Elmasre/Navathe Figure 3.15.





# Enhanced ER (EER) model

---

Designers must use additionally modelling concepts to

- represent the requirements from applications as accurately and explicitly as possible.

# Enhanced ER (EER) model<sub>(cont)</sub>

---

There are many extensions to the ER model. We will look at one:

- *Specialisation*: the process of defining a set of subclasses of an entity type; this entity type is called the superclass of the specialization.
- *Generalisation*: a reverse process of specialisation.

A subclass inherits all the attributes of the superclasses.

# Enhanced ER (EER) model<sub>(cont)</sub>

---

A specialisation involves the following aspects:

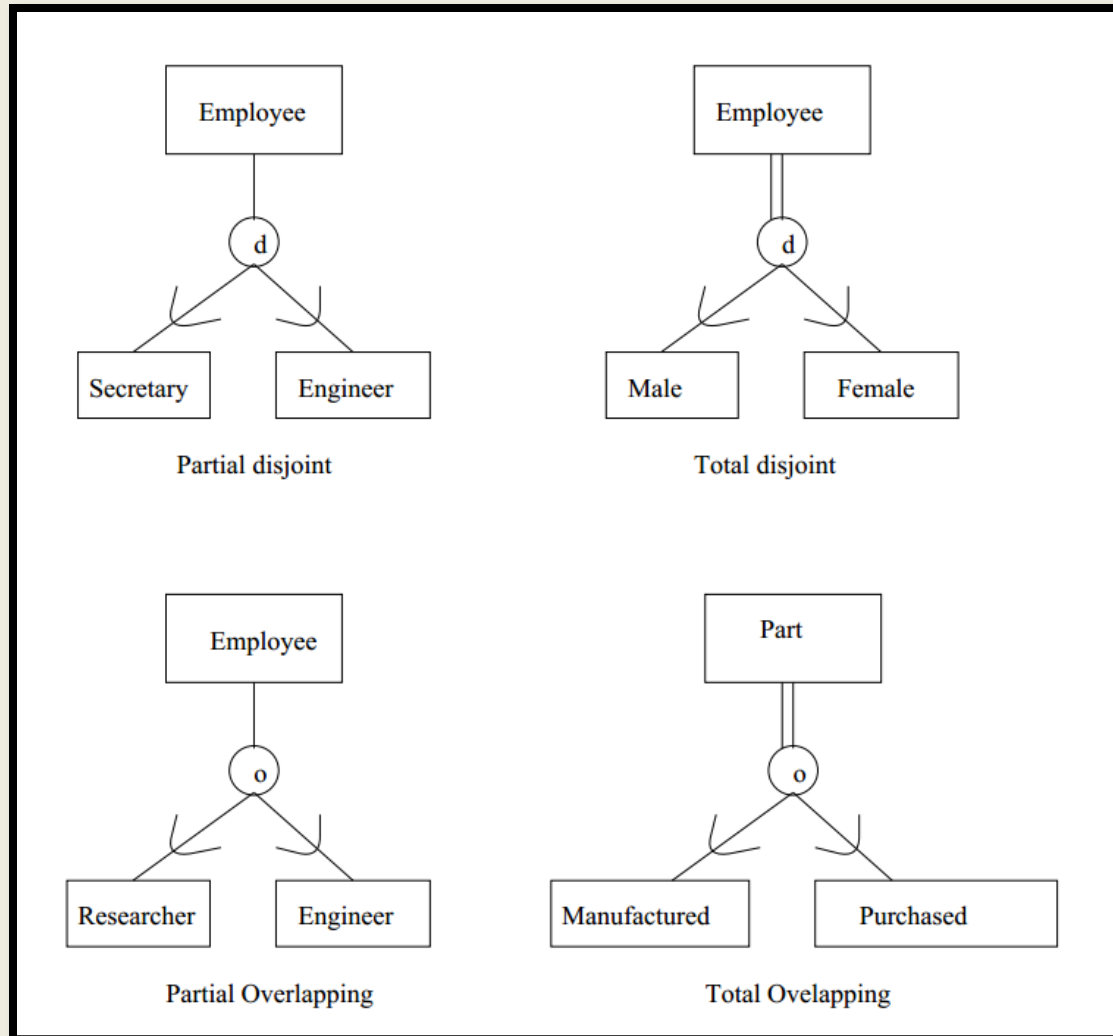
- Define a set of subclasses of an entity type.
- Associate additional specific attributes with each subclass.
- Establish additional specific relationship types between each subclass and other entity types, or other subclasses.

A subclass may have multiple superclasses.

A specialisation:

- may be either total or partial; and
- may be either disjoint or overlapping.

# Enhanced ER (EER) model<sub>(cont)</sub>



# Design Principles

---

Faithfulness: reflect reality.

Avoid redundancy.

Picking the right kind of element.

# Exercise:

---

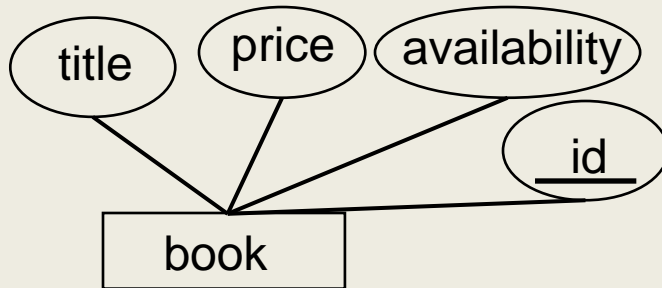
Please draw an ER diagram for UNSW library based on the following specifications.

- A book is uniquely identified by its book id. For each book, we also record its title, price, and availability.
- A reader is uniquely identified by his/her reader id and we also record his/her name, phone number, expire date and address. The address is composed of street and suburb.
- A publisher is uniquely identified by its publisher id. For each publisher, the name is also recorded.
- An author is uniquely identified by his/her author id. For each author, the name, phone number and birth date are also recorded.
- A reader can borrow zero or more books and a book can be borrowed by zero or more readers. Thus, we need to record the starting date and ending date for the borrowing relationship.
- A publisher can publish zero or more books and a book is published by exactly one publisher. We also need to record the date of publication.
- An author can write zero or more books and a book is written by one or more authors.

# Exercise:

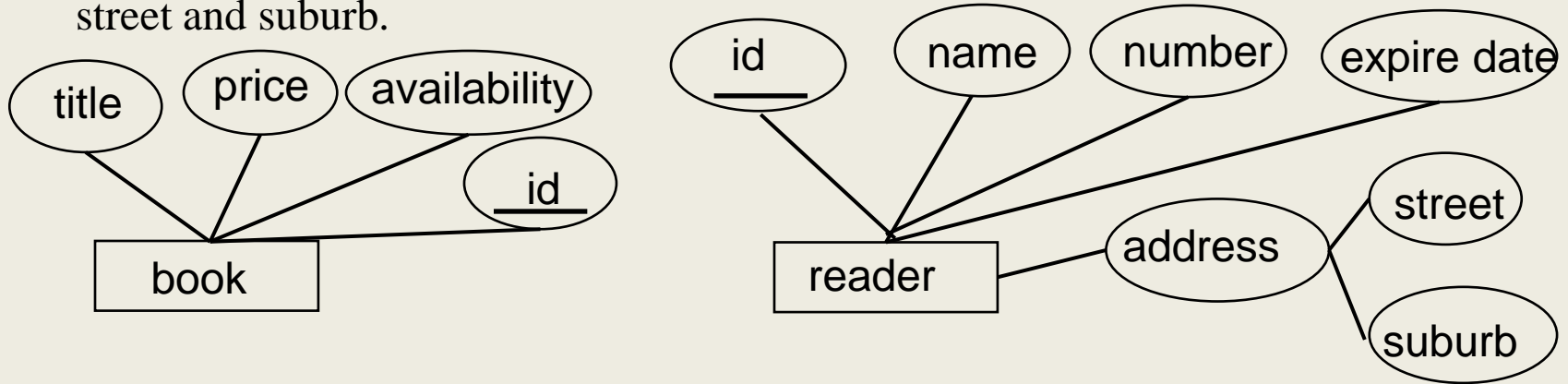
---

- A book is uniquely identified by its book id. For each book, we also record its title, price, and availability.



# Exercise:

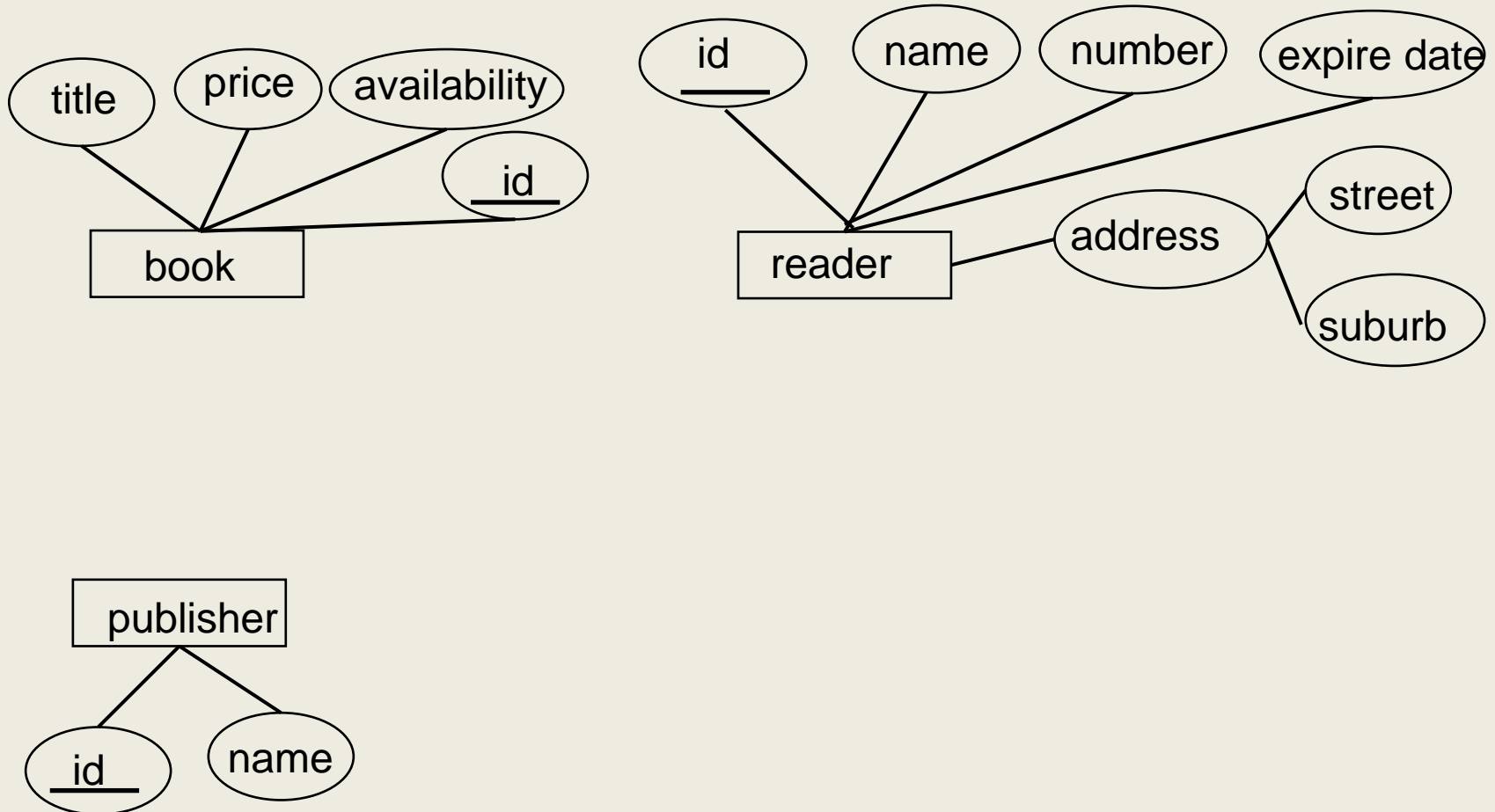
- A reader is uniquely identified by his/her reader id and we also record his/her name, phone number, expire date and address. The address is composed of street and suburb.





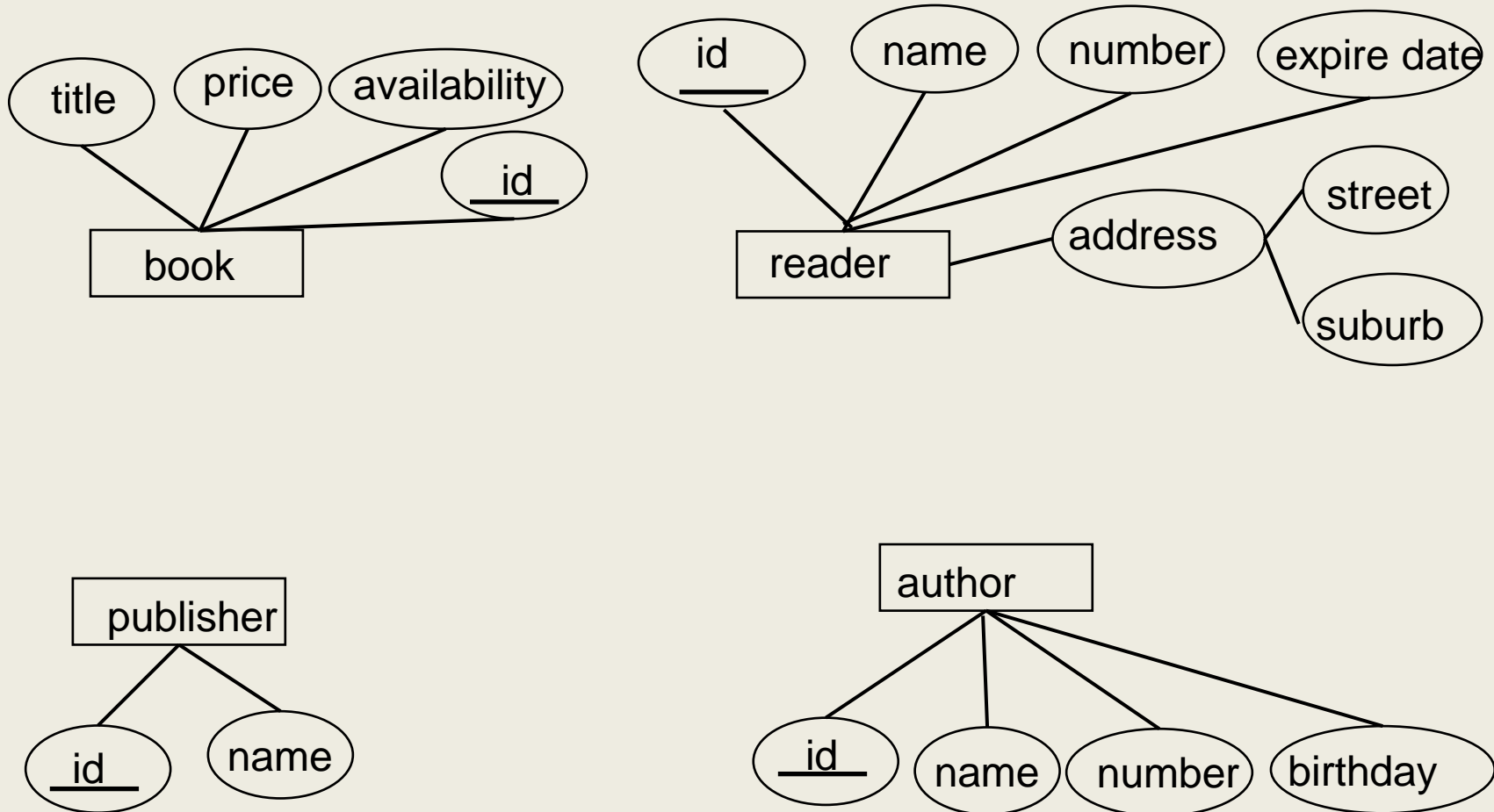
# Exercise:

- A publisher is uniquely identified by its publisher id. For each publisher, the name is also recorded.



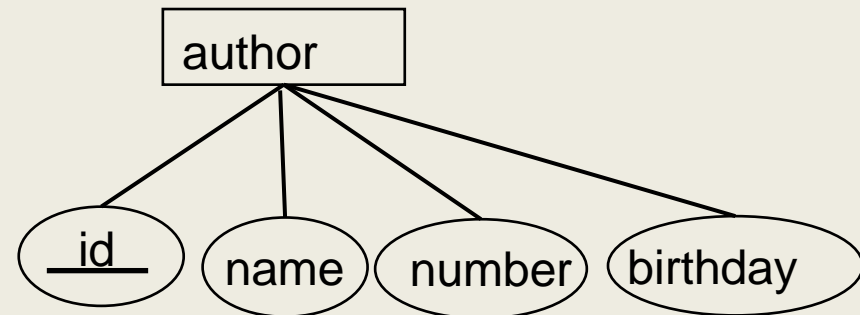
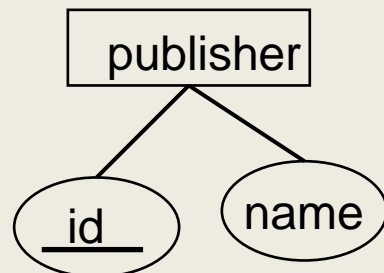
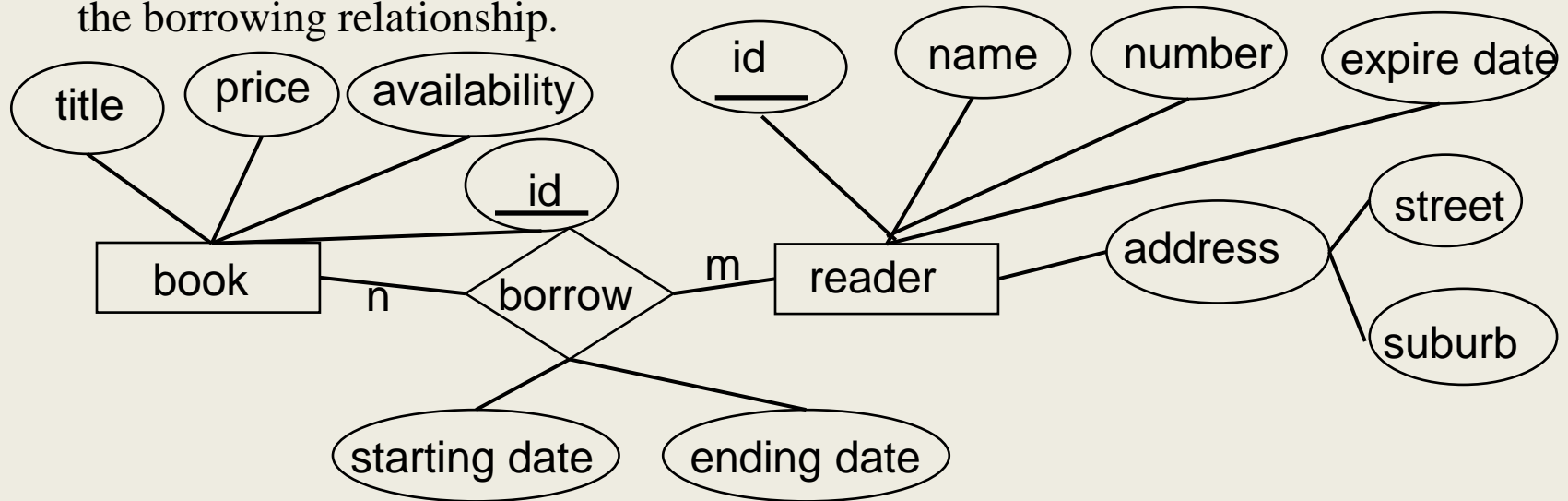
# Exercise:

- An author is uniquely identified by his/her author id. For each author, the name, phone number and birth date are also recorded.



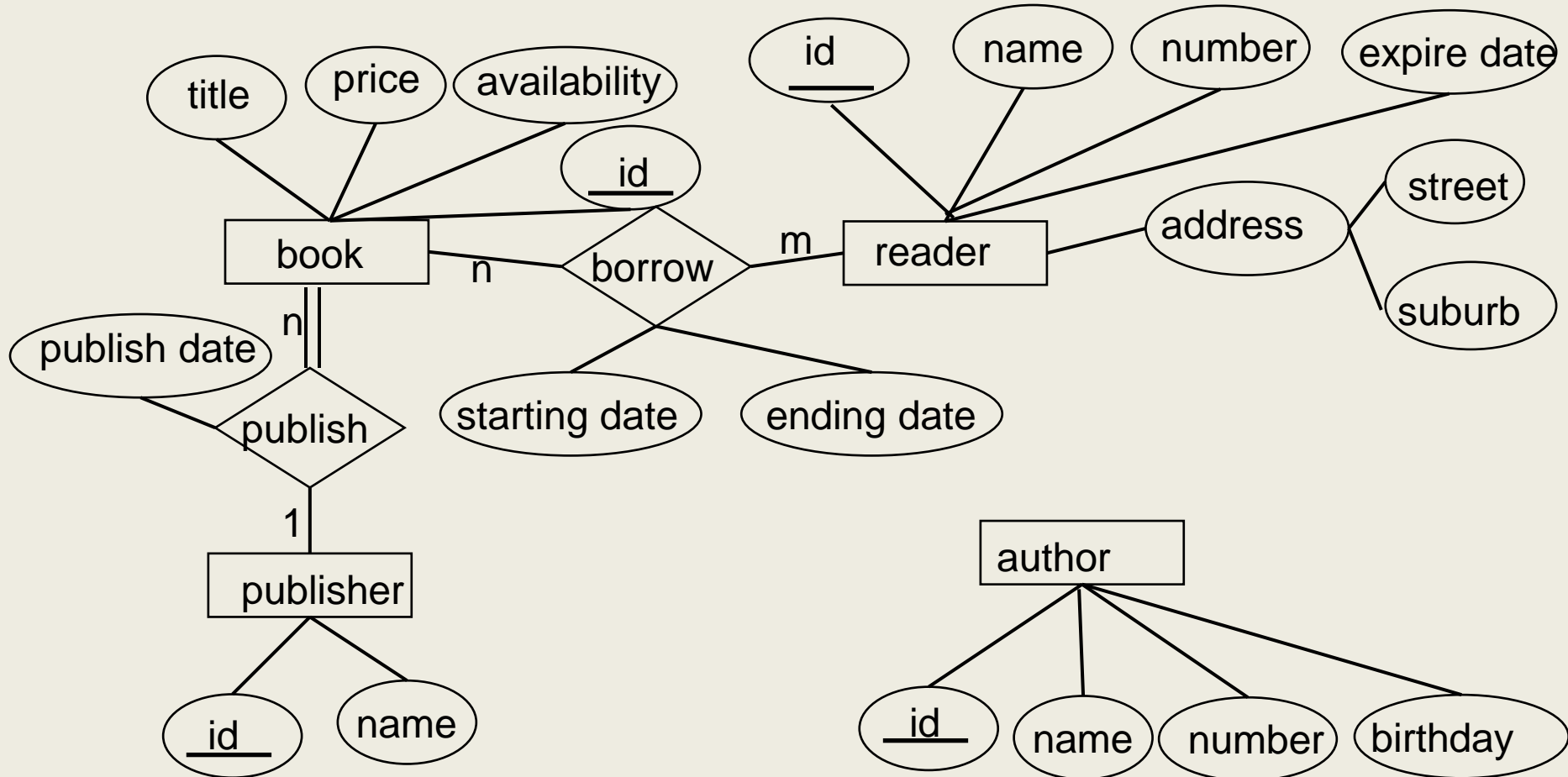
# Exercise:

- A reader can borrow zero or more books and a book can be borrowed by zero or more readers. Thus, we need to record the starting date and ending date for the borrowing relationship.



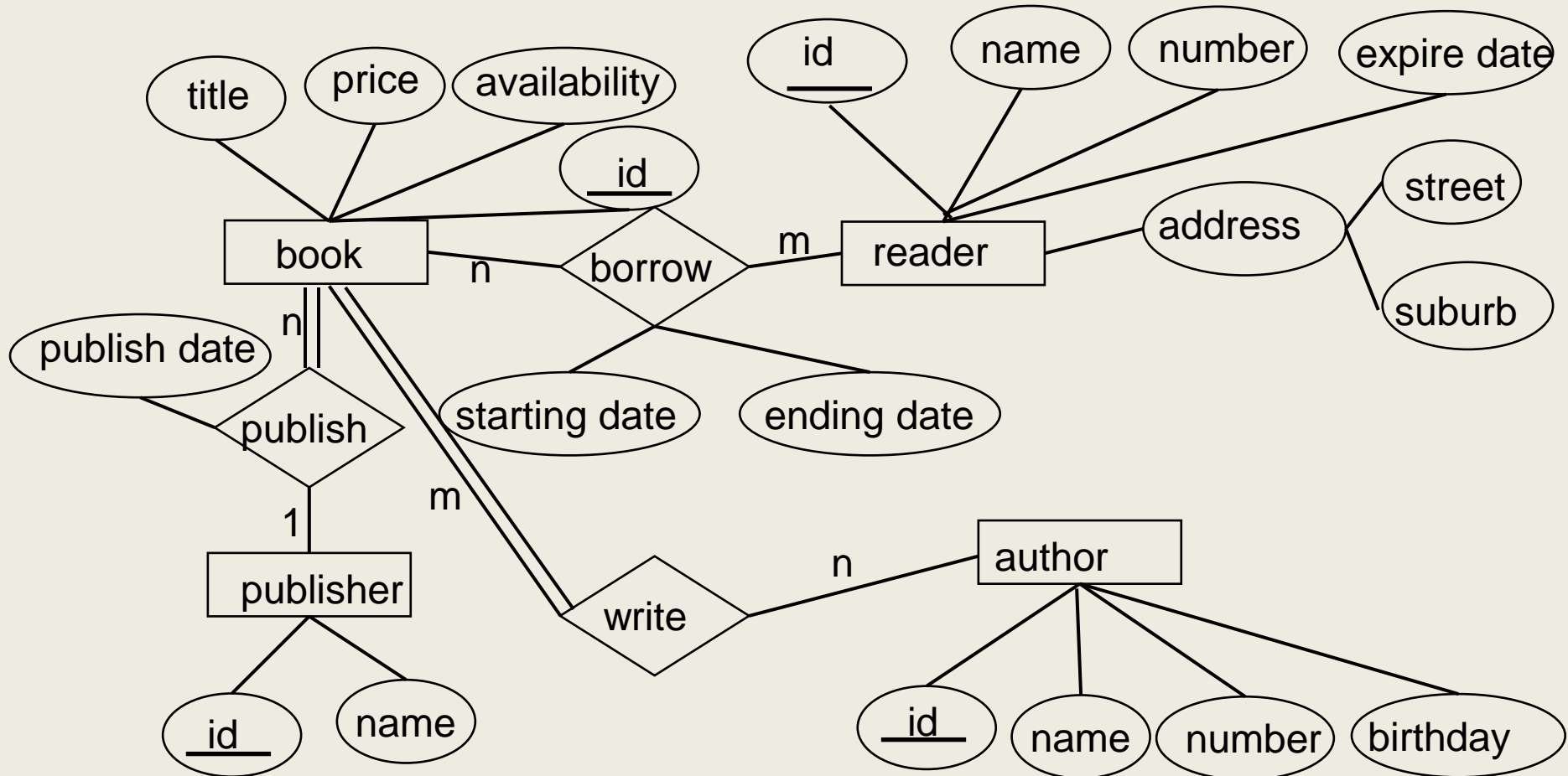
# Exercise:

- A publisher can publish zero or more books and a book is published by exactly one publisher. We also need to record the date of publication.



# Exercise:

- An author can write zero or more books and a book is written by one or more authors.



# Learning Outcome:

---

- Given application requirements, design ER-diagram to accurately reflect the requirements