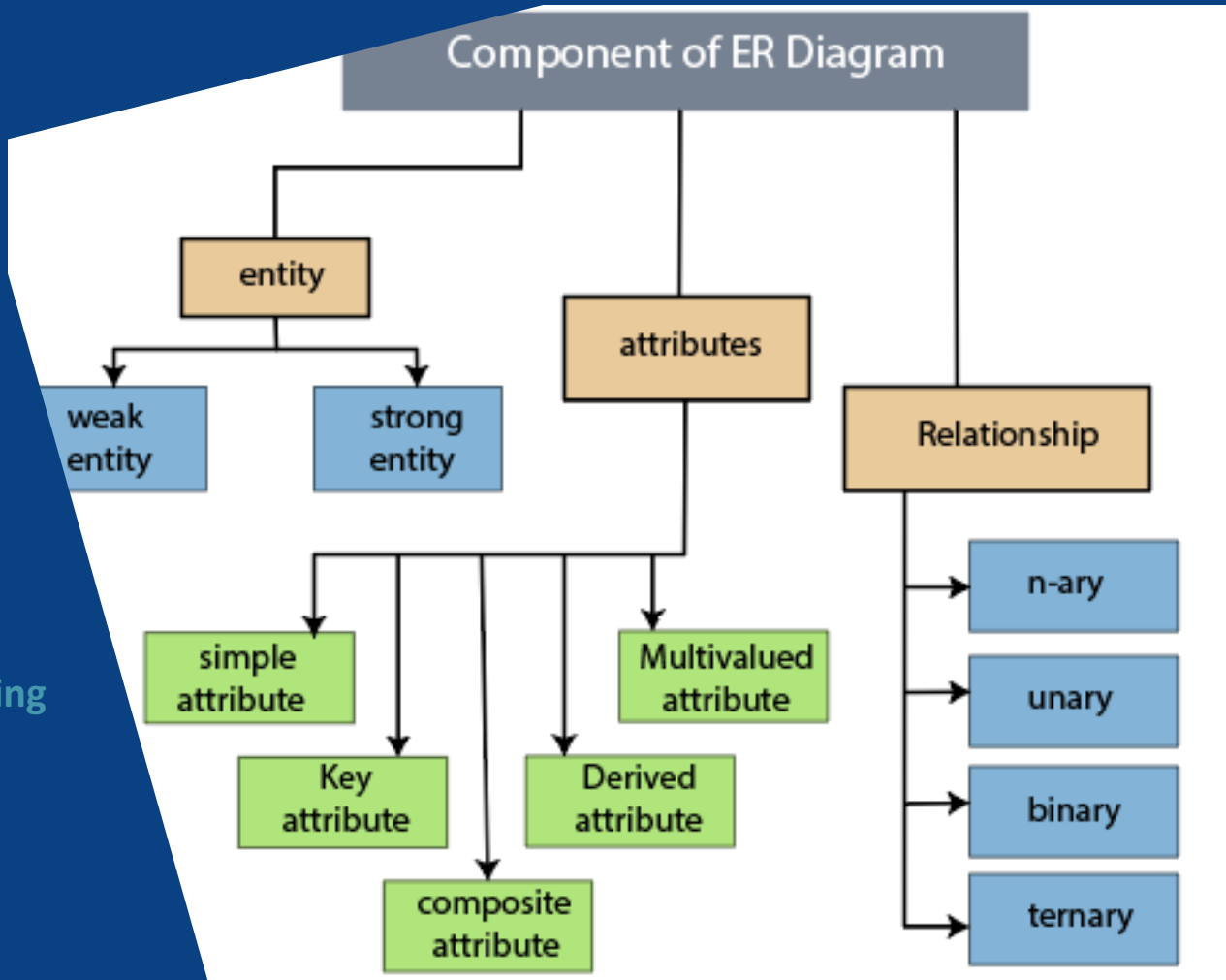
THE UNIVERSITY OF MELBOURNE

Introduction to Data Modelling

Database Systems & Information Modelling
INFO90002

Week 2 – Data Modelling
Dr Tanya Linden



Announcements

Assignment 1 - Time to form a team!

- Teams of 4
- All team members must belong to the same tute
- Register your team with your tutor
  - Email student numbers, names for all 4 students
  - Your tutor will respond with the group number, e.g. if you are in Wed 3.15 pm tute, you will get a response of Wed3.15pm-1 (or -2, or -3…)

Data Modeling and Data Models

Data modeling refers to the process of creating a specific data model for a determined problem domain

- Data modeling is an iterative, progressive process

A **data model** is a relatively simple representation of more complex real-world data structures

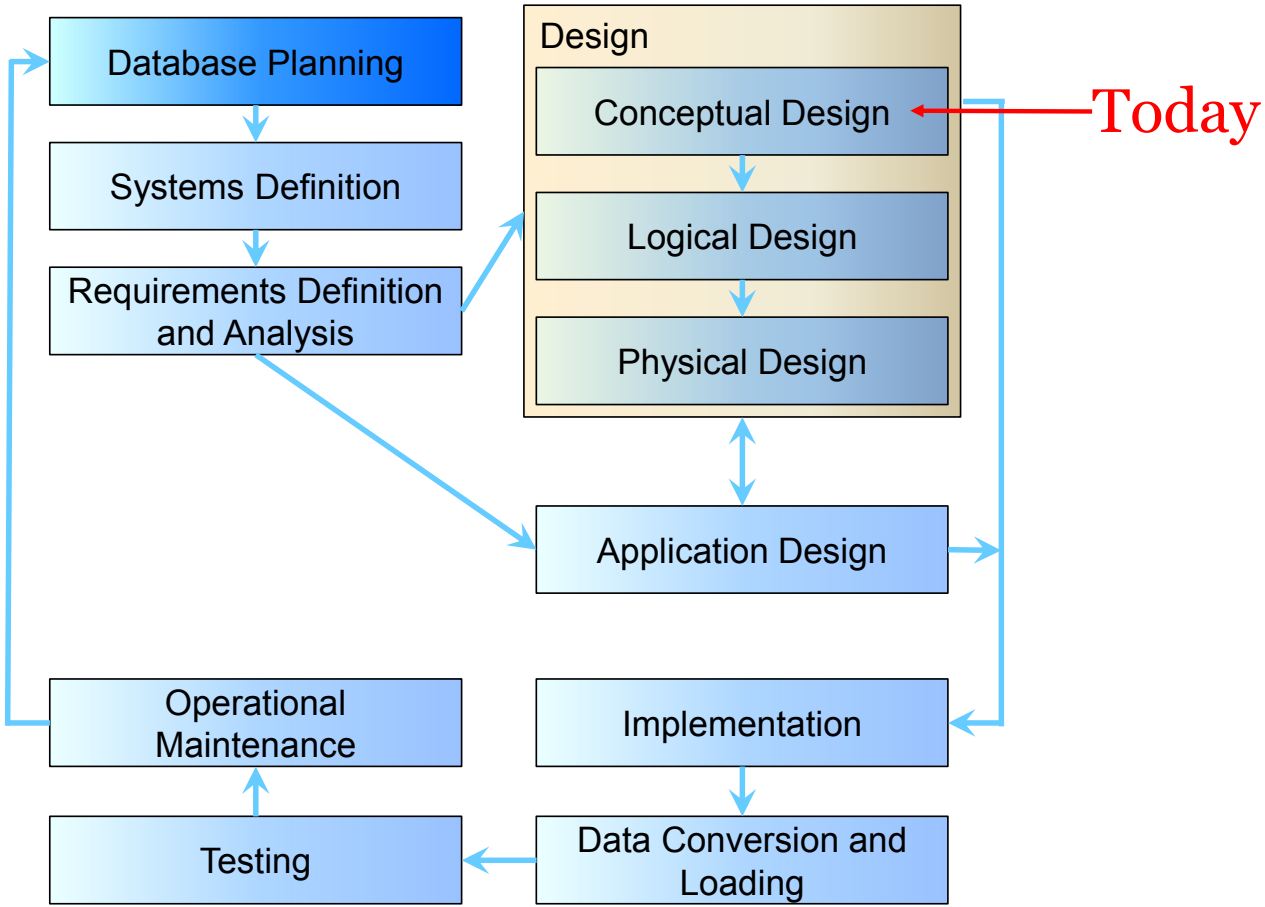
Database designers make use of existing data-modeling constructs and powerful database design tools that diminish the potential for errors in database modeling

Data models are a communication tool

Applications are created to manage data and to help transform data into information, but data is viewed in different ways by different people

When a good database blueprint is available, it does not matter that an applications programmer's view of the data is different from that of the manager or the end user

Database Development Lifecycle: Review



The Entity-Relationship Model

ER model:
entity-relationship model

(實體)

An ER Model / ER Diagram **does not** actually store any data

- An ER Model is a plan

↳ How store the data

It a plan how data represent and store in database

Entity Relationship Diagram

- A **graphical** representation of the ER Model

★ whatever you need to store about is your entity

★ entity relates to each other in some way

Basic ER modeling concepts

everything you need to store data about

- Uses **entities** to represent people, objects, events…

entities

- Identifies **relationships** between various entities

these entities related in some way

- Based around **business rules** of the organisation: What are the ***integrity constraints*** that hold? Example:

→ constraints for entities

  - in any given semester a student may enrol in up to 5 subjects

ER Model: Entity and its attributes

we describe entity with attributes

↳ Because the same entity could be different base on domain

Entity: Real-world object distinguishable from other objects. An entity is **described** (in DB) using a set of *attributes.*

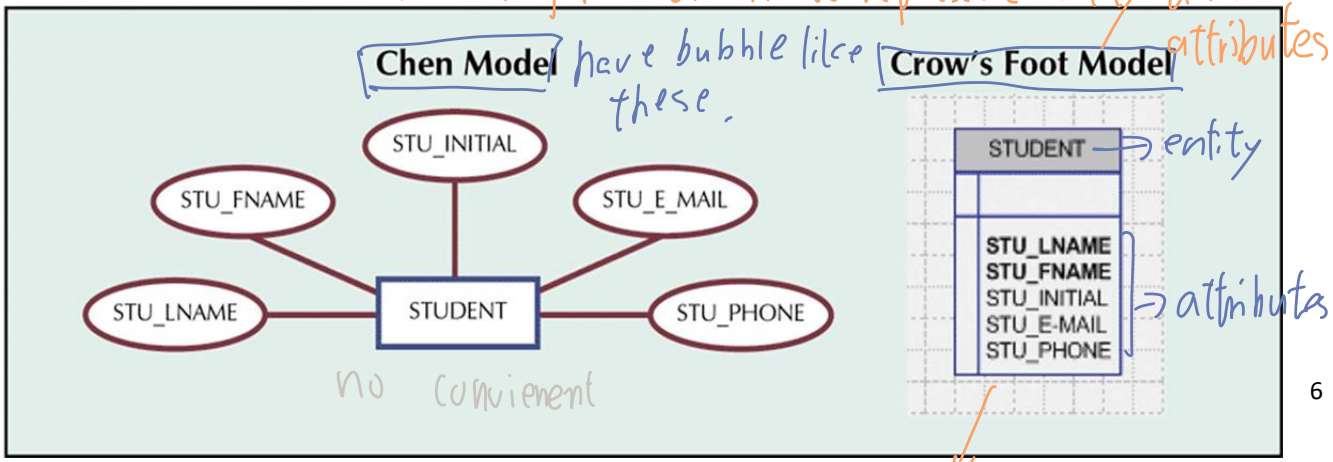
Refers to the *entity set* and not to a single entity occurrence

Corresponds to a table and not to a row in the relational environment

In both the Chen and Crow's Foot models, an entity is represented by a rectangle containing the entity's name

entities are labeled in database

two major notation to represent entity and attributes



Source: Coronel and Morris
Figure 4.1 STUDENT Entity

Attributes

each entity is table, and value is attributes.

Attribute names must not have spaces (use alpha-numeric and underscore or use *camelCase* for readability)

Attributes have a domain, i.e. the set of possible values for a given attribute

Domain is described by the company's business rules

attribute has domain

Examples:

this is the rule, which constraints between 0 ~ 100

- Attribute **Mark** at any Australian University must be **between 0 and 100** inclusive
- Attribute **Grade** at the University of Melbourne can only have one of the following values H1, H2A, H2B, H3, P, N, NH
- Attribute **Grade** at Deakin or Swinburne Universities can only have one of the following values HD, D, C, P, N
- Attribute **LastName** can contain only alphabetical characters and dashes

Attributes may share a domain

- Phone numbers of staff and customers adhere to the same rules, therefore sharing the same domain

Required attributes – the attributes that must have a value (NOT NULL) → cannot be null

Optional attributes may be left blank

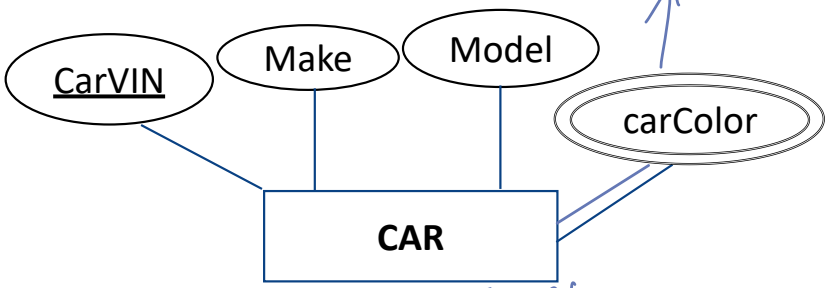
↓ it's ok to left back

Attributes (cont)

Single-valued vs **Multi-valued** attributes

A **single-valued attribute** can have only a single value,
e.g. *year_of_birth*, *carRego*, *gender*

A **multi-valued attribute** can have many values,
e.g. a car may have several colours, one for a roof, another for body and another for trim



In Chen's notation a multi-valued attribute is shown with a double line border (in some Chen notations shown in bold or connected by double line)

Later-on multi-value attributes need to be replaced, e.g. attribute carColor may be split into carBodyColor, carTrimColor, carRoofColor

- Not always a good solution to replace one attribute with several when we don't know how many possible values to replace with (e.g. employee qualifications)

Attributes (cont)

multi-value: same type of value

compositive attribute: contains different type of value

sub attribute: if seperate or not: depends on your requirement

Composite vs Simple attributes

A **composite attribute** can be further subdivided into additional attributes

- *Address* contains *street number*, *street name*, *city*, *state*, *postcode*, *country*

A **simple attribute** cannot be subdivided further, e.g. *gender*, *age*, *year_of_birth* (but not *date_of_birth*)

composite: we can seperate date_of_birth into year, month and day

Derived attribute – can be calculated using other attributes, data or functions

- Example, *age* is a derived attribute from *DOB* and *current date CURDATE()*

Case:
When you meet the people with same name.

Identifier / Key attribute / Primary key

An identifier is an attribute(s) that uniquely identifies an instance of an entity

- Also called a **Primary Key**

Rules

- Every entity must have an identifier
- Every instance of an entity must have a unique identifier
- No duplicates
- The value of an identifier cannot be empty / null

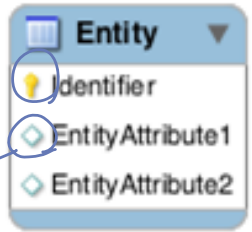
Chen notation – underlined

In Crow's Foot marked as PK

Conventions of ER Modelling (Workbench)

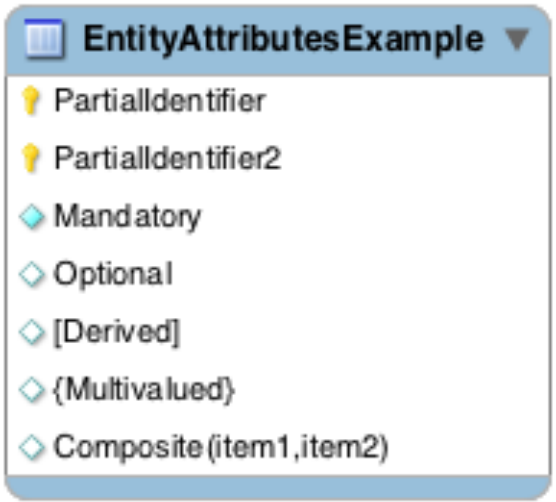
primary key → yellow key

Entity



optional attribute

Attributes



Identifier or key:

- Fully identifies an instance

Partial Identifier:

- Identifies an instance in conjunction with one or more partial identifiers
- **Attributes types:**
  - Mandatory (blue diamond)
  - Optional (empty diamond)
  - Derived []
    - [Age]
  - Multivalued {}
    - {CarColour}
  - Composite ()
    - Name (First, Middle, Last)
    - Address (Street, Suburb, Postcode)

ER Model: Relationship

Relationship: Association between two or more entities.

- Example: John *places* a Pizza order.

***Relationship Set*:** Collection of relationships of the same type.

- Example: Customers *place* orders.

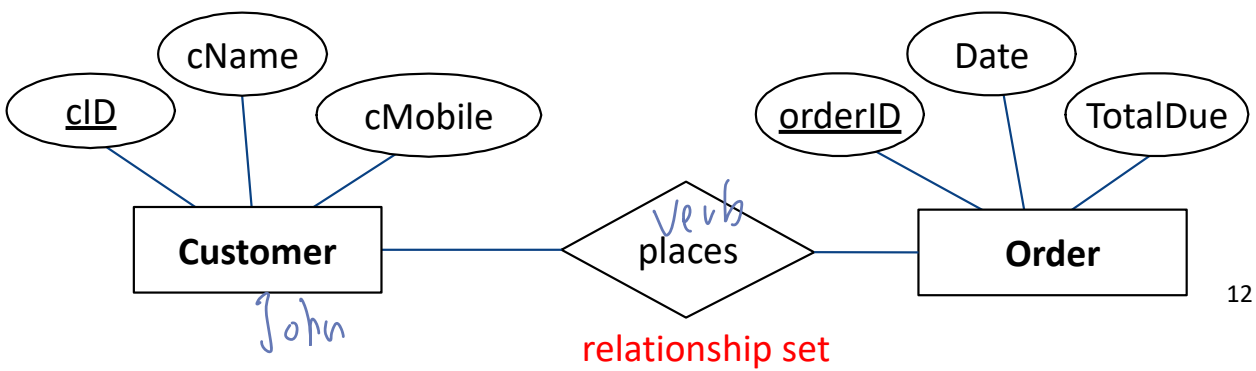
Connected entities are called participants

Relationships between entities always operate in both directions

- A customer places orders
- An order belongs to a customer

What notation is used here?

↳ Chen relation



Connectivity and Cardinality

how many entities on the right relate to how many entities on your left.

1:1
1:M
M:M

Connectivity

- Used to describe the relationship classification, e.g. 1:M
  - 1 customer can place MANY orders
  - 1 order is placed by 1 customer

many can also be 0.

Cardinality

- Expresses the specific (minimum and maximum) number of entity occurrences associated with one occurrence of the related entity, e.g.
  - 1 customer can place between 0 to MANY orders
  - 1 order is placed by 1 and only 1 customer (i.e. cannot be 0 customers or 2 or 3 customers owning the same order)
  - A credit card account can have up to 2 cards associated with it (minimum 1 card, maximum 2 cards) 1..2 or 1,2
  - One credit card belongs to 1 account only

depends on business rule

can not be 0. would only be 1

min
max

business rule

Established by very concise statements known as *business rules*

E-R Models and Business Rules

Consider some **business rules:**

- A student must only be enrolled in 1 course at any time
- A student may enrol in up to 5 subjects at one time (semester)
- A subject must only have 1 coordinator
- An employee must only have 1 tax file number

Some business rules are **common** amongst many organisations

Some business rules are **unique** to a single organisation

Very few organisations have identical business rules

(Lucky for IT professionals – or jobs in the IT industry are in trouble)

Your ERD design and subsequent database design **must meet the needs** of the organisation's business rules

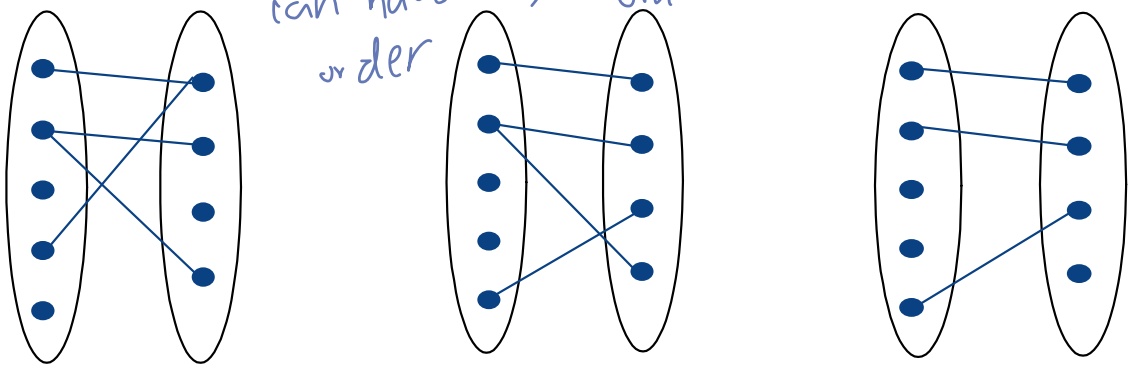
Connectivity / Relationship Types

Connectivity determines the number of entities taking part in the relationship set (how many from each side)

Types of relationships:

customer can have many order

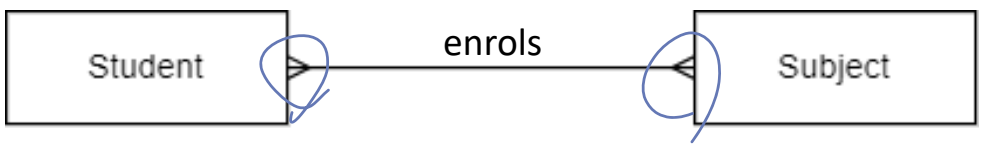
one order belongs to one customer



Many-to-Many	**One-to-Many**	**One-to-One**
Examples: Student enrols in Subject	Customer places Order	School is headed by Principal

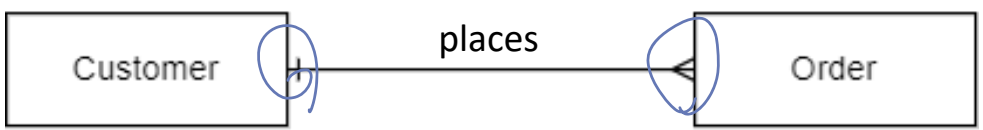
*Note, the above diagrams **do not depict cardinality**.*

Crow's foot notation



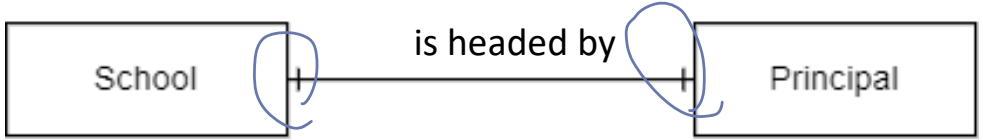
Many-to-Many

Student enrols in many Subjects
Subject has many enrolled Students



One-to-Many

Customer places many Orders
Order belongs to one Customer



One-to-One

School is headed by one Principal
Principal presides over one School

Connectivity Exercises

department ⊢< employee

Draw Crow's foot diagrams based on the following statements:

1. A department in an organisation employs several employees but an employee can be employed by one department only.
2. A company Gift Warehouse is made up of several departments with each department employing several people but each employee belongs to one department only. The salespeople working for the Sales department look after sales orders placed by customers. Each sales order is controlled by only one salesperson. Each customer can place multiple orders but each order belongs to one customer. Each order can contain multiple products and each product can appear in many orders.

1. Department ⊢< Employee

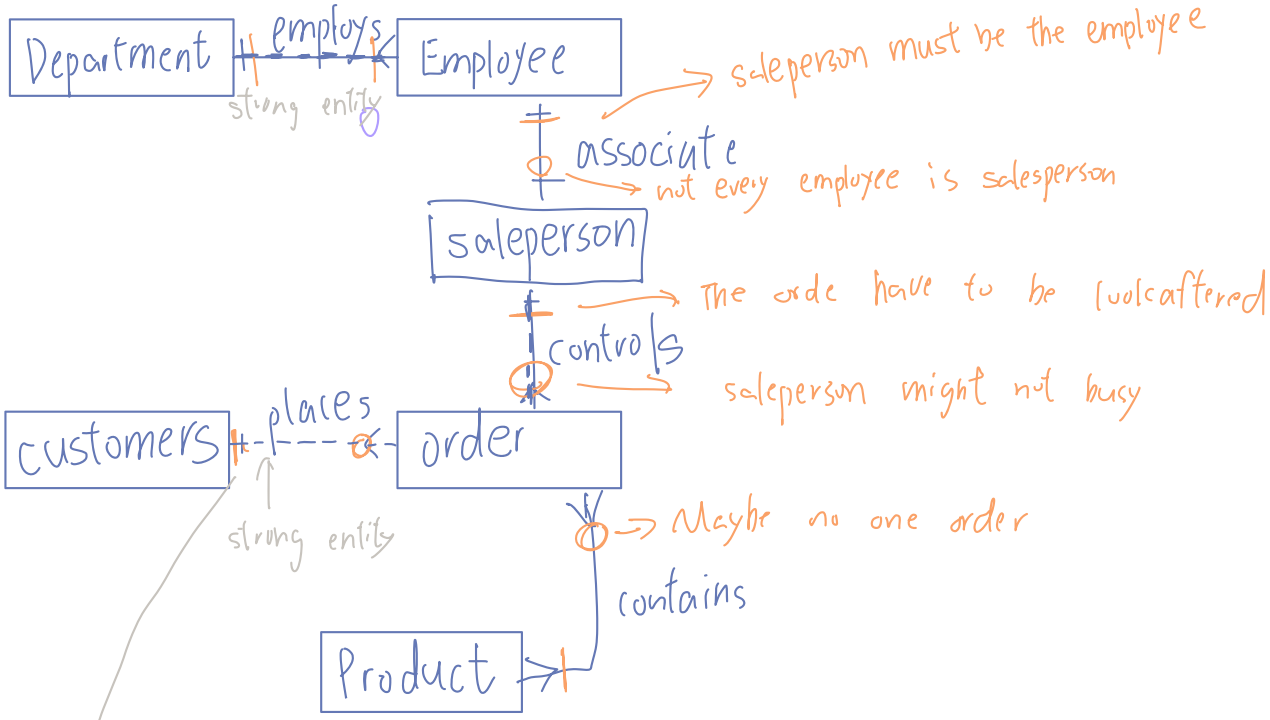
Department ⊢< employee

sales

customer ⊢< order

product

2.



- customers might have many order (0 ∞) optional
  but order must belongs to only one customer

- mandatory means that existance of the instance in another entity is required.

  ★ notice the requirement

  It's mandatory or optional depend on requirement

Relationship Participation / Cardinality

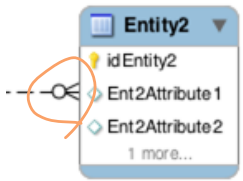
Optional:

- One entity occurrence does not require a corresponding entity occurrence in a particular relationship
  - E.g. a customer may place several orders means that a customer may have 0 or more orders placed.

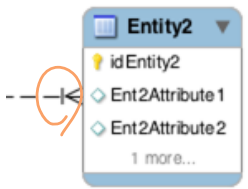
Mandatory:

- One entity occurrence requires a corresponding entity occurrence in a particular relationship
  - E.g. Every order must belong to a customer, i.e. an order cannot exist on its own without being placed by a customer
  - A customer must place at least one order (How can this be implemented? Can we create a customer before the order is placed?)

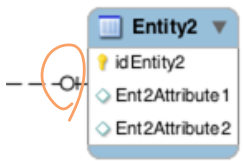
Cardinality



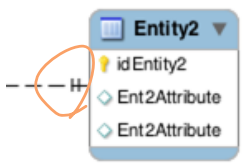
Optional Many
Partial participation



Mandatory Many
Total participation



Optional One
Partial participation



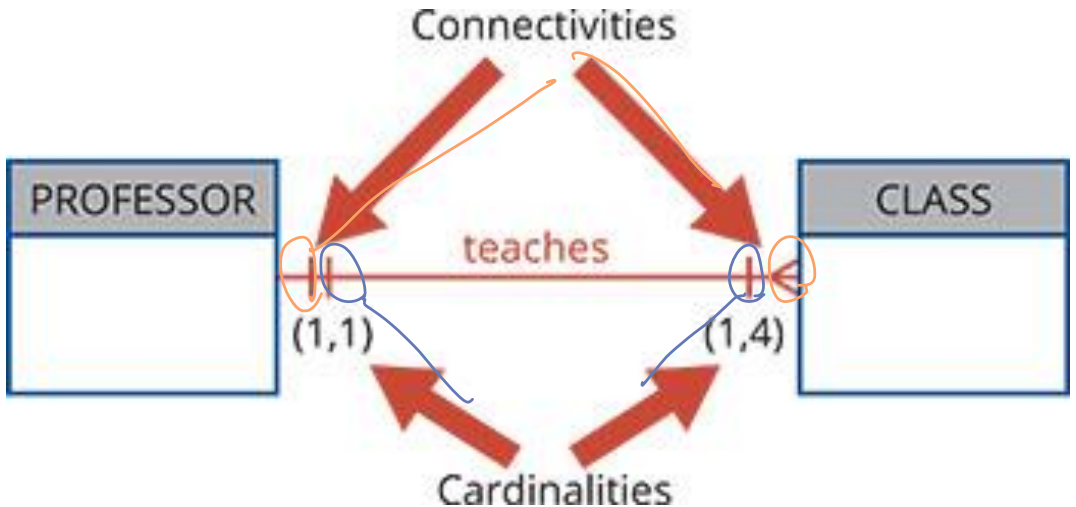
Mandatory One
Total participation

Cardinality

Cardinality expresses the minimum and maximum number of entity occurrences associated with one occurrence of the related entity

In some ER notations, cardinality is indicated by placing the appropriate numbers beside the entities, using the format (x,y)

- The first value represents the minimum number of associated entities and the second value represents the maximum number of associated entities



*In our solutions we will **not** be adding (min, max)*

Source: Coronel and Morris
Figure 4.7 Connectivity and Cardinality in an ERD

Cardinality Exercises

Enhance Crow's foot diagrams with cardinality details:

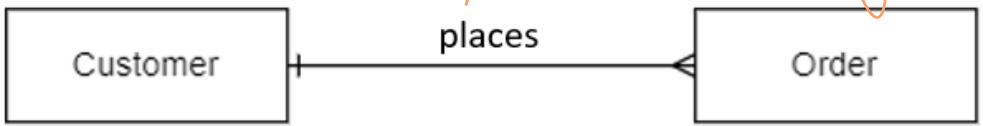
1. A department in an organisation employs several employees but an employee can be employed by one department only.
2. A company Gift Warehouse is made up of several departments with each department employing several people but each employee belongs to one department only. The salespeople working for the Sales department look after sales orders placed by customers. Each sales order is controlled by only one salesperson. Each customer can place multiple orders but each order belongs to one customer. Each order can contain multiple products and each product can appear in many orders.

Foreign Key

It's a primary key from the other entity which you are linking

Customer places an order

Customer may place many orders



How do we show which customer an order belongs to?

- Place customerID as an attribute in Order
- customerID becomes a **Foreign Key** (FK) for Order

Customer

primary

Customer ID	Surname	First name	Address	Postcode	Credit card
C2045	Smith	Fred	1 John St, Hawthorn	3122	1234234534564567
C2048	Nguyen	Vincent	2/7 Oak Ave, Altona	3018	4554123423457899
C2146	Davis	Liz	32 Lyle St, Toorak	3142	4564564578970022

3 order

PRIMARY Key

customer has many order

Order

Order No	Customer ID	Date	Time	Deliver
3224	C2045	1/09/2021	17:35	The bel
3228	C2045	3/09/2021	18:42	The bel
3236	C2048	3/09/2021	21:05	
3248	C2045	4/09/2021	15:20	The bel

FOREIGN Key

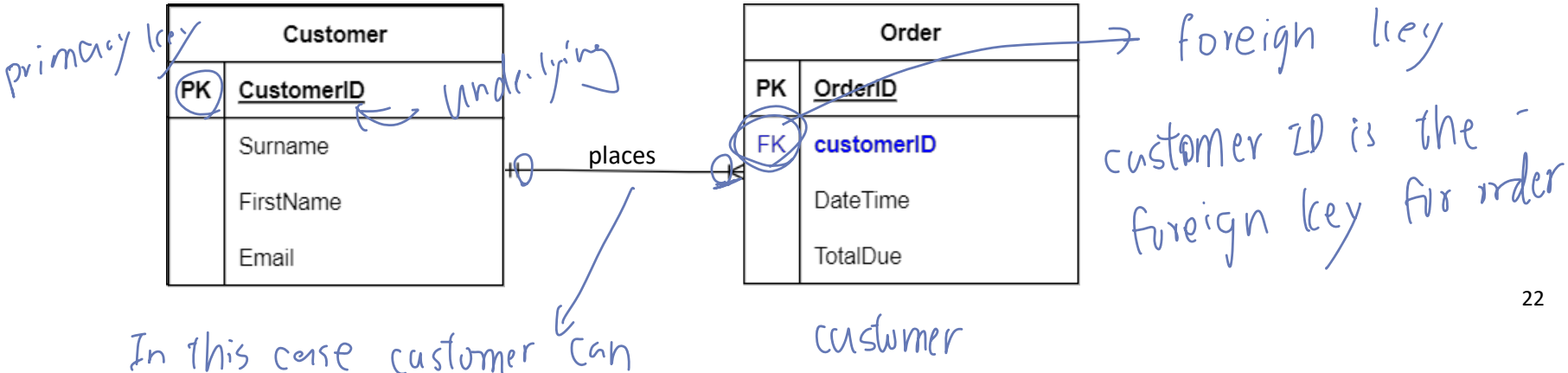
Foreign Key (cont)

Foreign Key is an attribute of the entity that refers to the PRIMARY KEY in another entity allowing to link those entities.

FK *always* goes to the *MANY side*

- Customer may place MANY Orders, therefore customerID becomes a Foreign Key of the Order entity

Exercise: what is the cardinality depicted on this diagram? Do you agree with it?



be registed only they placed the order (mandatory)

Strong and Weak entities

An entity is a person, place, object, event or concept.

Entities can be classified into two groups

- **Strong** Entities
- **Weak** Entities

A **strong entity** is : it has it's own unique identifier

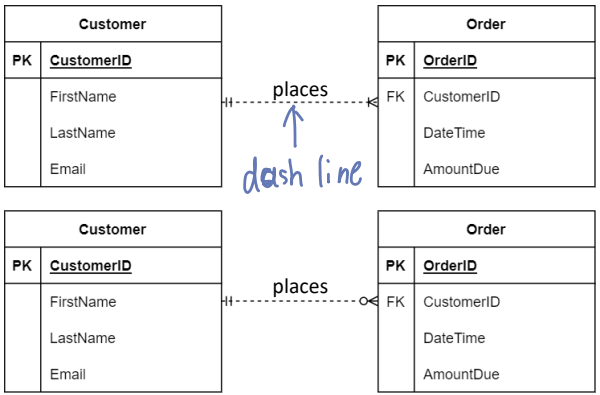
"An entity type that exists independently of other entity types"
(Hoffer Prescott McFadden)

A **strong entity** can be identified by its own attributes, meaning a unique identifier can be chosen from its own attributes

All previous examples had **Strong Entities**

- Department, Employee
- Student, Subject
- Customer, Order

Strong entities are connected by non-identifying / weak relationships, represented by dashed line in Crow's foot



Weak entity

A weak entity is an entity that meets 2 conditions

- The entity **cannot exist** without the entity with which it has a relationship
- The entity has a unique identifier that is partially or totally derived from the parent entity in the relationship

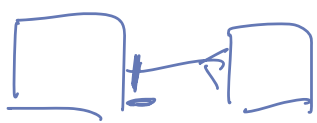
Owner entity set and weak entity set must participate in a *one-to-many* relationship set (one owner, many weak entities)

In Chen notation weak entities are represented by bold border or double border rectangle

In Crow's foot it is depicted through primary keys and solid line for the relationship.

Such relationship is called *identifying* or *strong.*

Weak entity set must have mandatory participation in this relationship set.



Example

Buildings at a university contain rooms.

Each building has a building name and building code (e.g. Melbourne Connect – MC, Doug McDonnell – DM) – strong entity

Each room has room number (e.g. 4.05, 2.01), room category (staff office, lecture theatre, lab) and capacity

Can the attribute roomNo be used as a unique identifier?

No, because several buildings have the same room numbers

Building	
PK	bCode
	bName

Room doesn't exist without building

Room	
PK	RoomNo ???
	Category
	Capacity

no unique

Example (cont.)

Relationship: the room is weak entity

- One building contains many rooms
- A room is contained in one building only

So we need a *composite identifier* for the room entity, i.e. bCode+roomNo own primary key but not unique

Problem: bCode is not an attribute of room

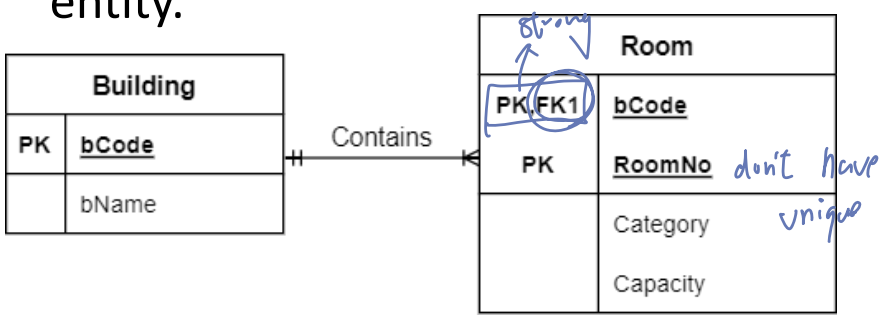
Solution: Room borrows identifier from building; bCode is a Foreign key of Room.

Room is a **weak** entity

- The identifier of Room is **partially or fully** defined by identifier(s) from other **related entities**

The relationship between a weak entity and its "parent" is a **strong / identifying** relationship and is denoted by a *solid* line. don't need to be strong, it can also be weak entity

"Parent" can be a strong or another weak entity.



Weak entity

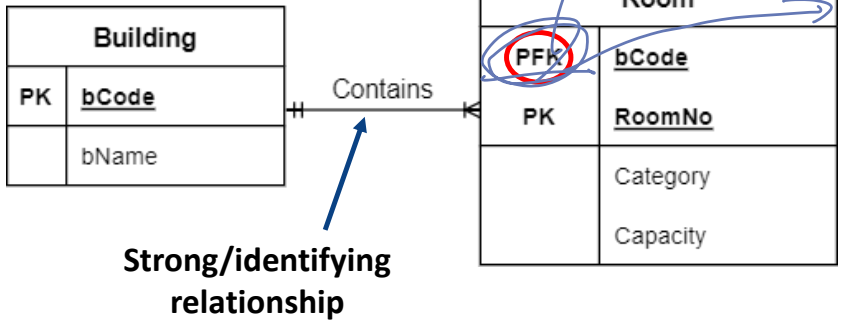
A weak entity cannot exist without its parent entity

It has a composite PK, part of which is borrowed from another entity (it's the best way to keep it unique)

The borrowed part of the identifier is PK of another entity and becomes a Foreign Key (FK) in the weak entity. It is called Primary Foreign Key (PFK)

In this example parent entity is a strong entity

Weak entity can be a parent of another weak entity

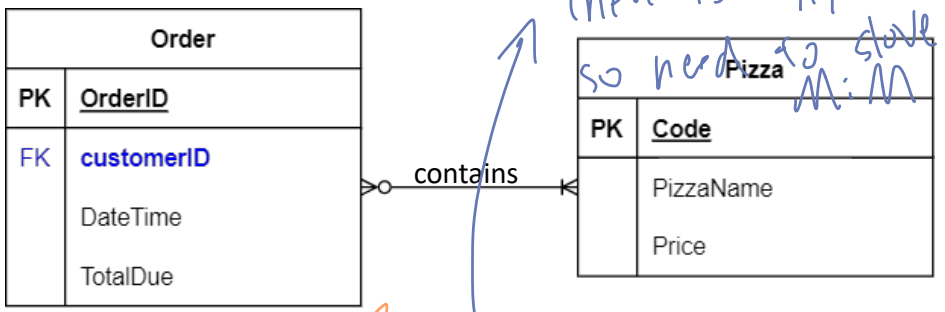


M:M and Weak entity

An order contains pizza(s)

Each pizza can be in many orders

M:M



M:M in a conceptual model – must be resolved for future steps

Logical and Physical models cannot represent M:M

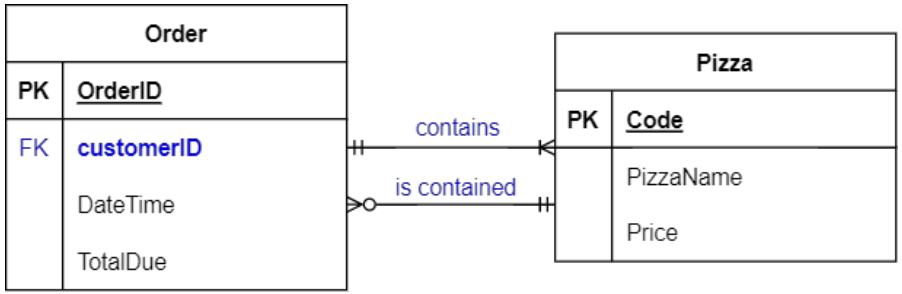
M:M cannot be implemented in the database

Where do you place ordered quantity?

M:M must be replaced by two 1:M

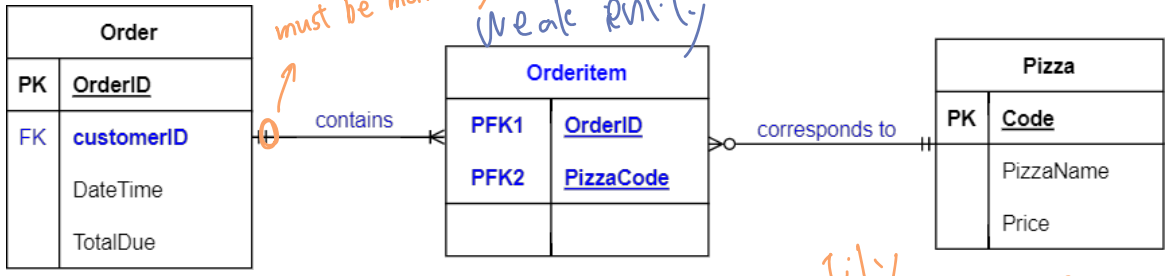
M:M and Weak entity (cont.)

Replace M:M by two 1:M

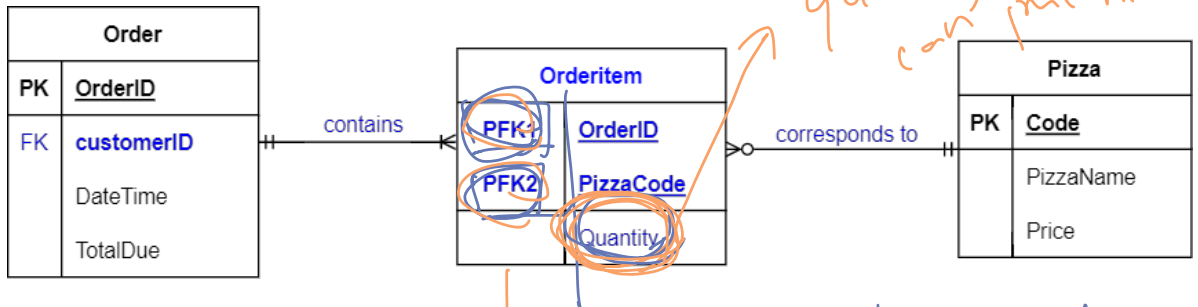


Insert a weak entity to resolve M:M

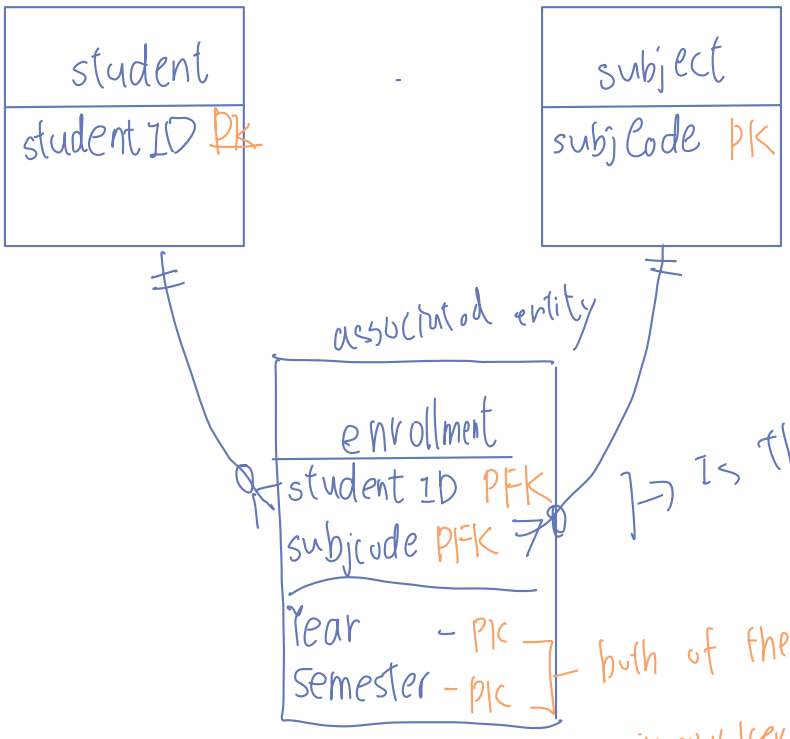
- Specify PFKs



Decide whether the weak entity has its own attributes



student and subject are M:M



orderitem cannot exist without pizza and order table.

Here to check PFK is unique or not

for this example PFK1 and PFK2 is make unique, so dont need another primary key

Is this unique ino in this class

↳ ex: student fail the subject, so it need primary key

both of them are primary key

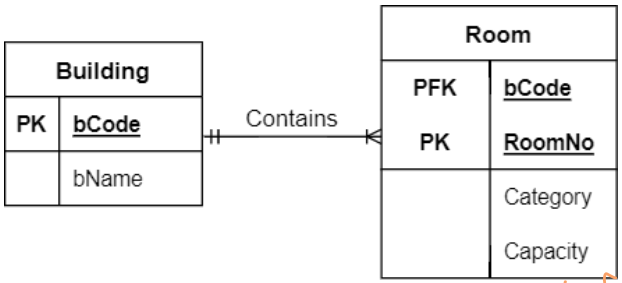
primary key should be unique.

neither of them is unique when it exists alone

Borrowing FK for PK

A parent entity can be a strong entity or another weak entity.

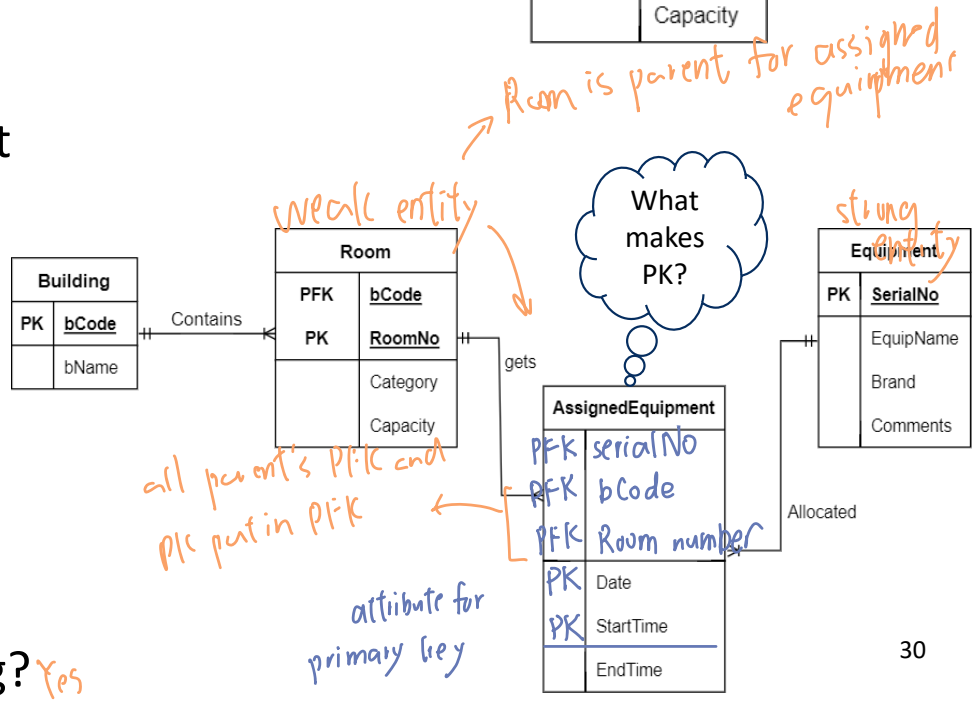
Example: building is a parent entity which is strong.



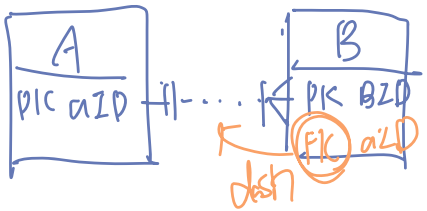
The company uses portable electronic equipment and accessories (e.g. data projectors, screens, microphones, speakers). Each room has many equipment pieces assigned to it, each piece of equipment may appear in different rooms at different times.

Room – Equipment is M:M, resolved with a weak entity AssignedEquipment.

Will AssignedEquipment borrow PK from Building? Yes



Exercise



Which one is a weak entity on this diagram?

Determine the PK of each entity.

C has only one may.
so C only depend on D. So start C first.

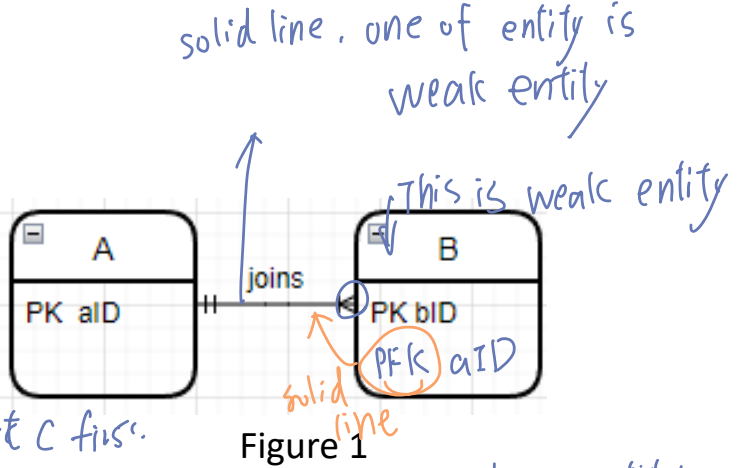


Figure 1

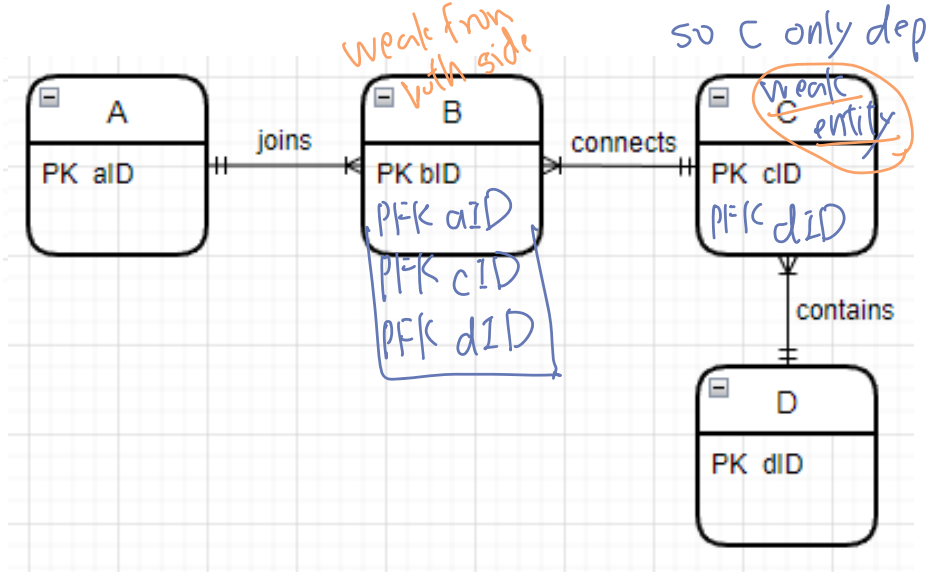


Figure 3

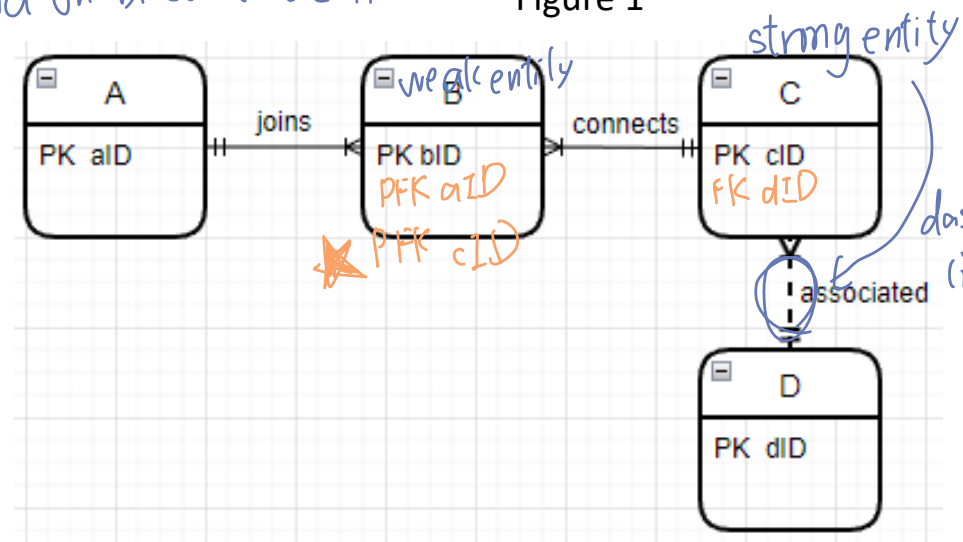
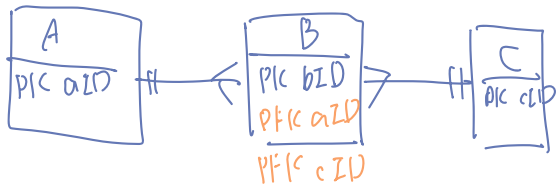
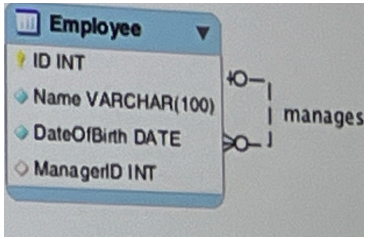


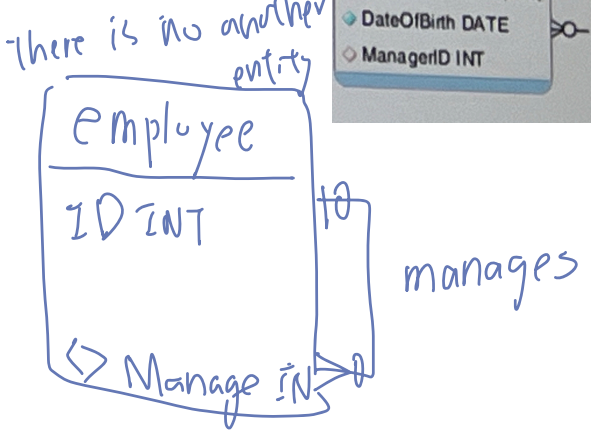
Figure 4



M:M → two 1:M

Relationship Degree





Indicates number of associated entities or participants

Unary relationship

- Association is maintained within a single entity
- Also known as recursive relationship
- E.g. an employee within an employee entity is the manager for one or more employees

Binary relationship (most common)

- Two entities are associated

Ternary relationship

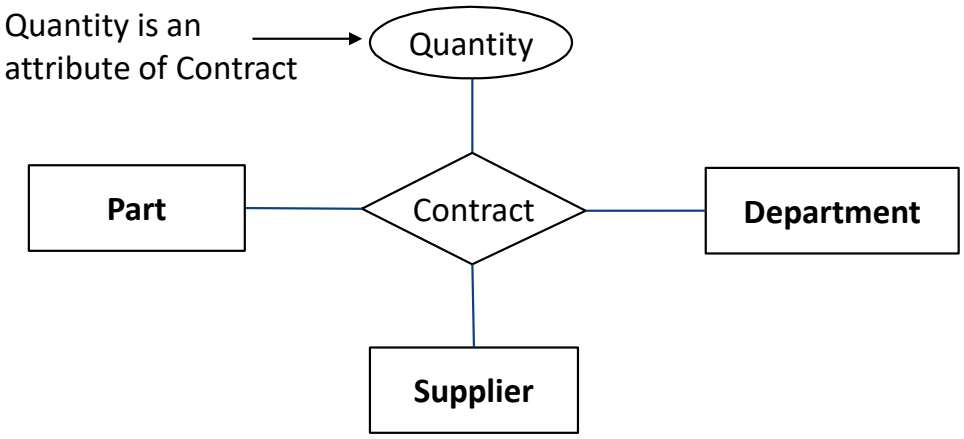
- Three entities are associated

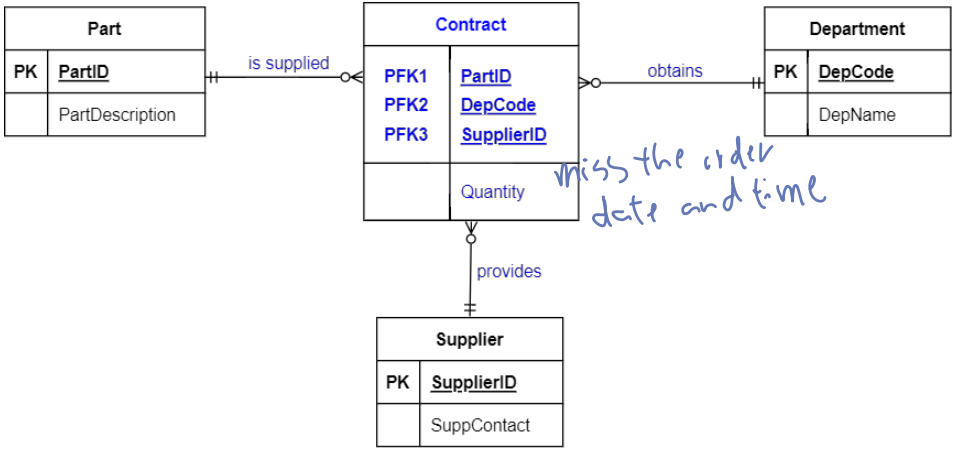
… N-ary

- N entities are associated

Ternary Relationships

In general, we can have **n**-ary relationships, and relationships can have attributes





This is a ternary relationship with one relationship attribute.

Hint: Count the number of contributing entities to determine the n-ary relationship name

"Departments will use one or more suppliers to obtain one or more parts for producing the product"

Developing an E-R Diagram

Iterative Process

1. List the major entities in the system.
2. Represent the entities graphically by a rectangle.
3. Search for relationships between the entities and represent them graphically with the proper symbol (e.g. a diamond if Chen's notation is used).
4. Add attributes; remember to establish the primary key for every entity.
5. Model relationship connectivity between each pair of entities.
6. Model relationship cardinalities between each pair of entities (i.e. the minimum and maximum participation).
7. Determine whether there are weak entities. Refine M:M relationships. If required, refine the connectivity and cardinality of entities affected.
8. Verify the ERD you have created by going through each component you have created from Steps (1) to (7). Ensure that they **properly represent the business rules** of the system you are developing the database for.

Conceptual Design Using the ER Model

Design choices:

- Should a concept be modelled as an entity or an attribute?
- Should a concept be modelled as an entity or a relationship?
- Should we model relationships as unary, binary, ternary, n-ary?

Constraints in the ER Model:

- A lot of data semantics can (and should) be captured (check business rules)

Entity vs Attribute

Example:

Should "*address*" be an attribute of Customer or an entity (related to Customer)?

Answer:

Depends upon how we want to use address information, and the semantics of the data:

- If we have several addresses per customer (delivery, billing, postal), *address* must be an entity
- If the structure (city, street, etc.) is important, *address* could be modeled as an entity

Notes on the ER design

ER design is *subjective*. There are often many ways to model a given scenario!

Where to start – noun-verb analysis

Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:

- Entity vs. attribute, entity vs. relationship (in Chen), unary, binary or n-ary relationship.

There is no standard notation (we will focus on Crow's foot, but you were introduced to **Chen's** notation)

Community Toy Library – noun-verb analysis

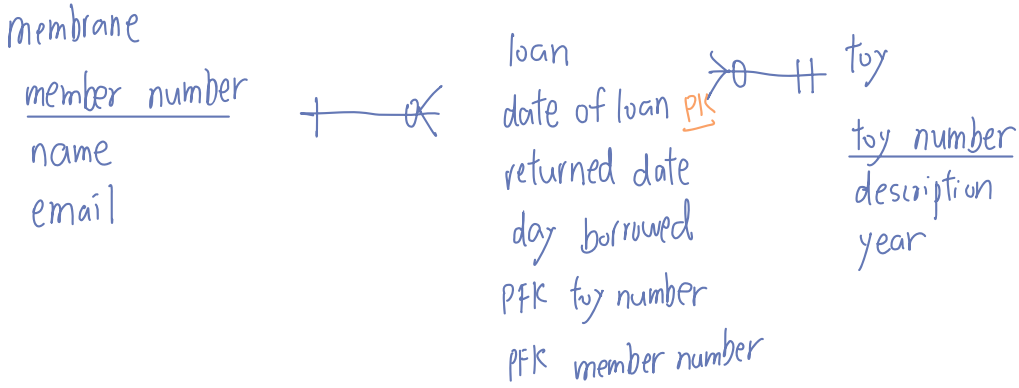
Sample Business Narrative:

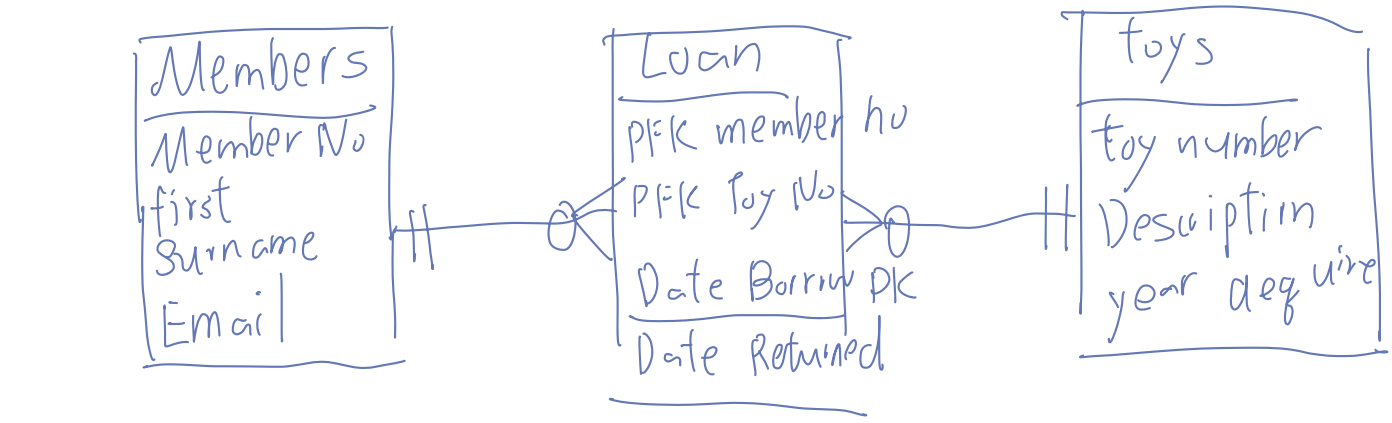
Community Toy Library has many **Members**. The library records a member's **name** and **email**. Each new Member is assigned a sequential **member number**.

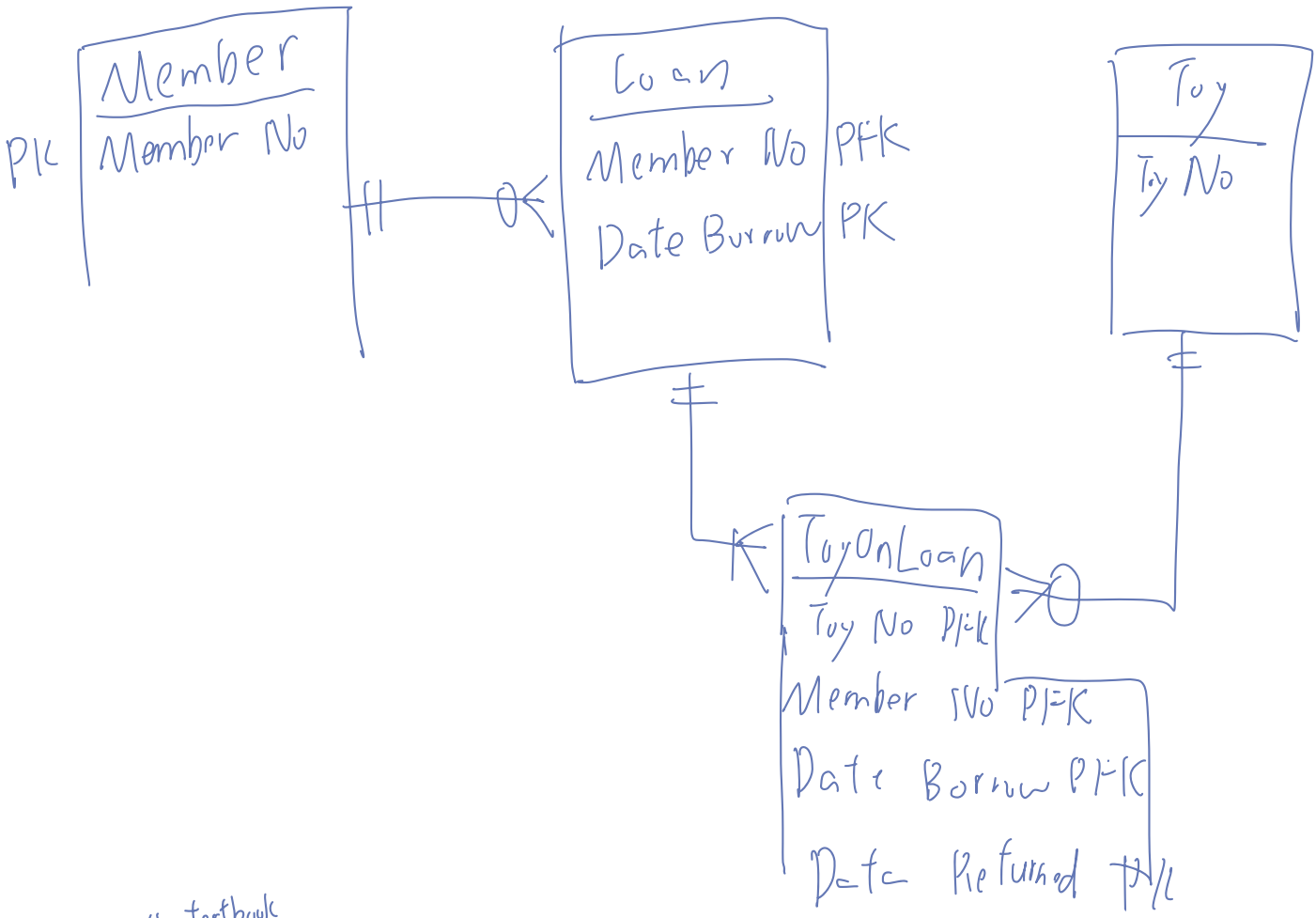
Over years the library acquired a number of **toys**. Every toy in the library has a unique **toy number.** Each toy has a **description** and **year** of acquisition.

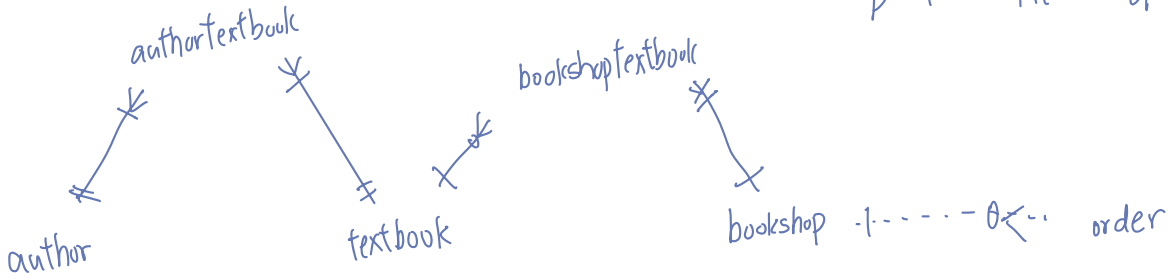
Each time a toy is loaned to a member, the **date of the loan** is recorded. When the toy is returned, the **returned date** is recorded and the number of **days borrowed** is determined.

Each loan is for a single toy (for now).


membrane
member number
name
email
loan
date of loan PK
returned date
day borrowed
PFK toy number
PFK member number
toy
toy number
description
year


Members
Member No
first
Surname
Email
Loan
PFK member no
PFK Toy No
Date Borrow PK
Date Returned
toys
toy number
Desciptin
year aequire


Member
PK
Member No
Loan
Member No PFK
Date Burrow PK
Toy
Toy No
ToyOnLoan
Toy No PFK
Member No PFK
Date Borrow PFK
Date Returned

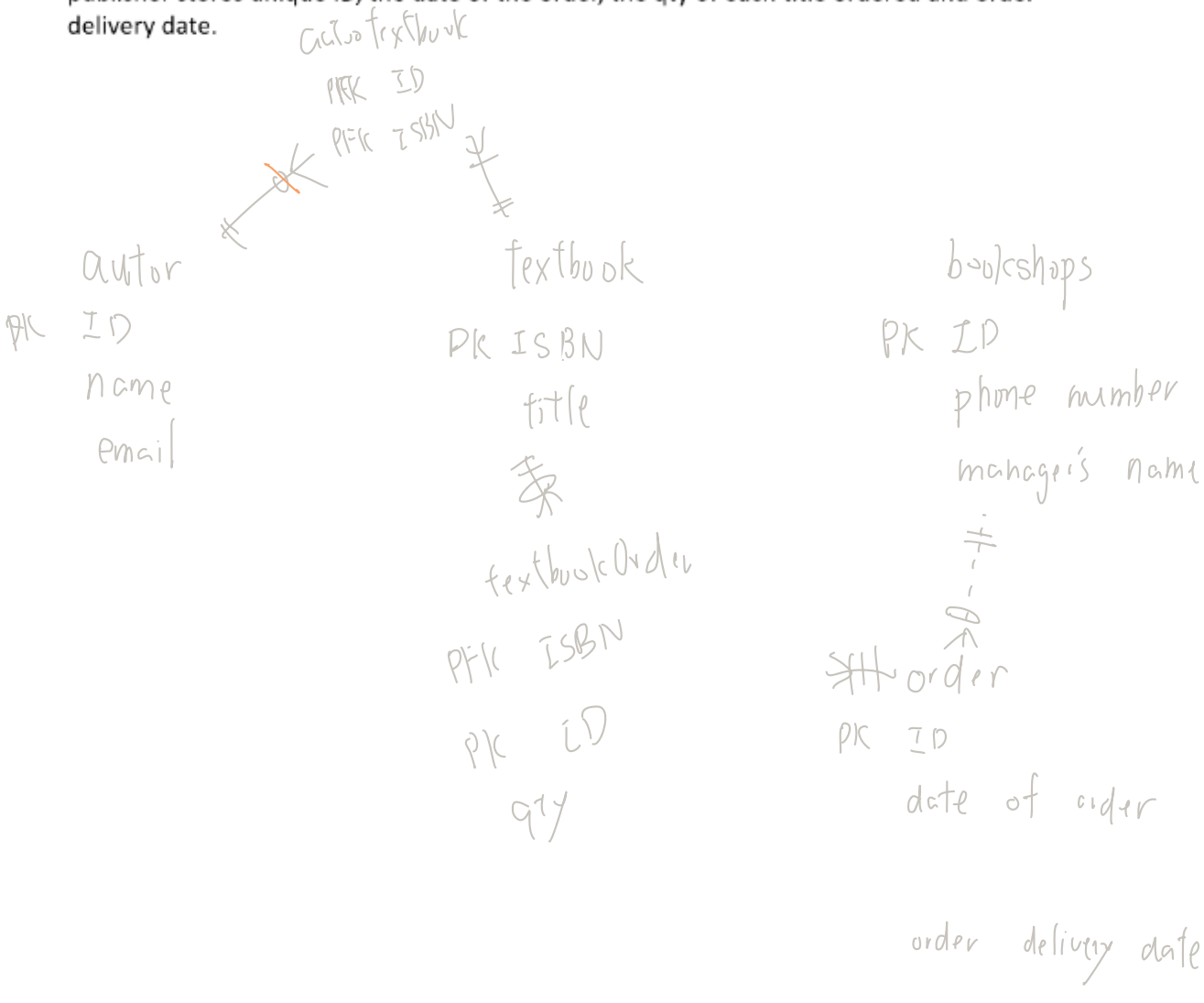

authortextbook
bookshoptextbook
author
textbook
bookshop
order

Exercise

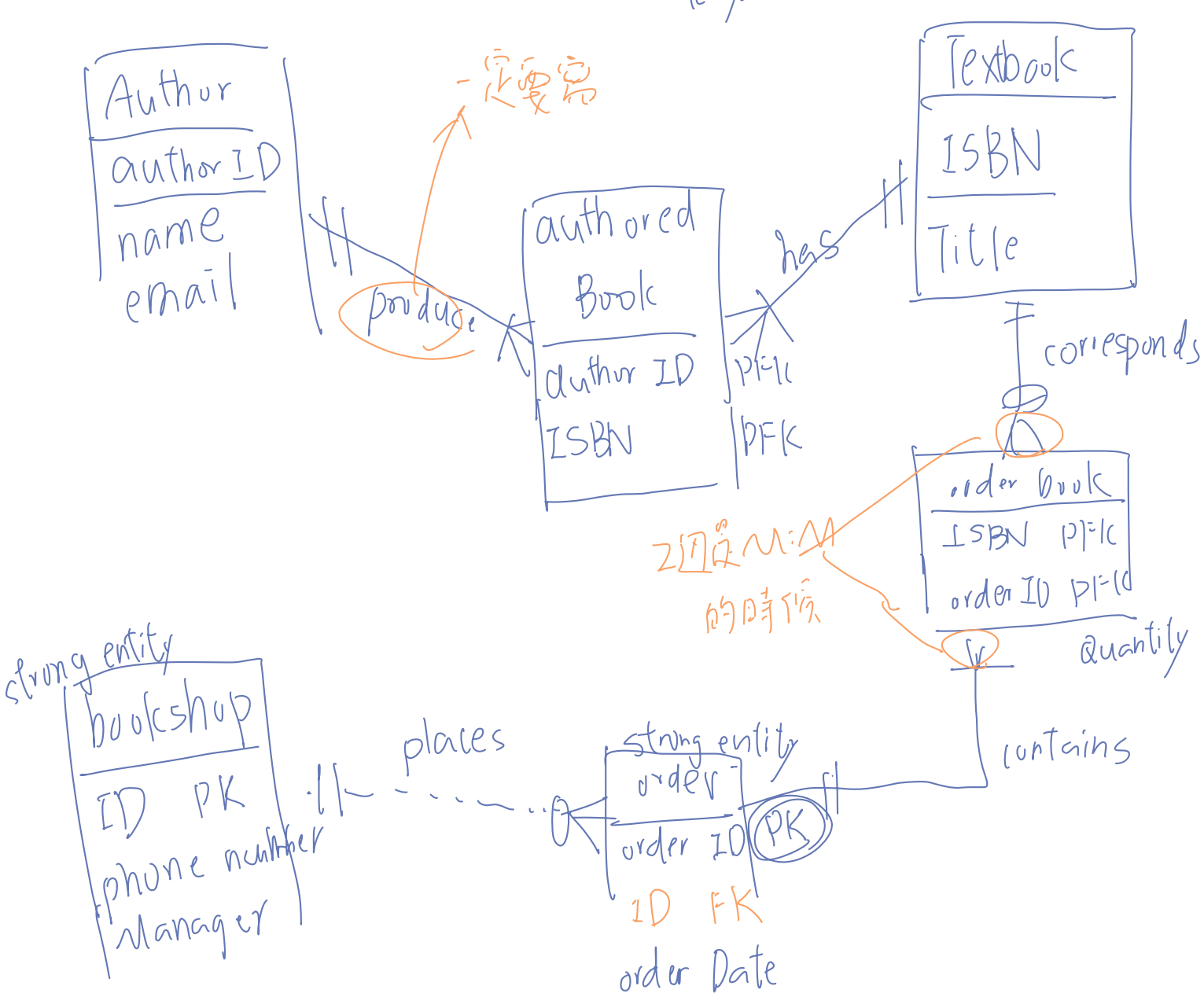
Uni Publishing specialises on textbooks for University subjects. They have several authors employed who produce 2-3 textbooks a year. For each textbook the publisher records a title and a unique ISBN code. For each author they keep an ID, a Name and an email. Sometimes 2 or more authors collaborate on the same textbook.

The publisher supplies textbooks to a number of university bookshops located at different campuses. They identify each bookshop by an ID, phone number and they also store the manager's name.

Each bookshop regularly orders textbooks from Uni Publishing. For each order the publisher stores unique ID, the date of the order, the qty of each title ordered and order delivery date.



Because ISBN is unique, so it doesn't need any primary key.


Author
authorID
name
email
produce
authored
Book
authorID
ISBN
PFK
PFK
has
Textbook
ISBN
Title
corresponds
order Book
ISBN PFK
orderID PFK
Quantity
contains
strong entity
bookshop
ID PK
phone number
Manager
places
strong entity
order
orderID
PK
ID FK
order Date

Identify Entities and Attributes

Element	Circle best alternative
Community Toy Library	StrongEntity / WeakEntity / Attribute / Calculated / Ignore
Member	StrongEntity / WeakEntity / Attribute / Calculated / Ignore
MemberNo	StrongEntity / WeakEntity / Attribute / Calculated / Ignore
Firstname	StrongEntity / WeakEntity / Attribute / Calculated / Ignore
Surname	StrongEntity / WeakEntity / Attribute / Calculated / Ignore
Member email	StrongEntity / WeakEntity / Attribute / Calculated / Ignore
Toy	StrongEntity / WeakEntity / Attribute / Calculated / Ignore
ToyNo	StrongEntity / WeakEntity / Attribute / Calculated / Ignore
Description	StrongEntity / WeakEntity / Attribute / Calculated / Ignore
YearAquired	StrongEntity / WeakEntity / Attribute / Calculated / Ignore
DateBorrowed	StrongEntity / WeakEntity / Attribute / Calculated / Ignore
DateReturned	StrongEntity / WeakEntity / Attribute / Calculated / Ignore
DaysBorrowed	StrongEntity / WeakEntity / Attribute / Calculated / Ignore

What's examinable*

Need to be able to draw conceptual diagrams on your own

- **Given a problem, *determine entities, attributes, identifiers, relationships***
- **What is primary key, foreign key and PFK?**
- **Connectivity and cardinality**
- **Do we need a weak entity?**
- **Mark *non-identifying* relationships**

* All material is examinable – these are the suggested key skills you would need to demonstrate in an exam scenario

THE UNIVERSITY OF MELBOURNE

Thank you