T04 Prashansa Singh

# INFO20003 Tutorial 4

Starting ~2.20 pm

Today's tutorial

- Tute 3 Q3 Review
- Multivalued & composite Attributes
- Unary Relationships
- Simple & Bus case study
  - group work

Github Link

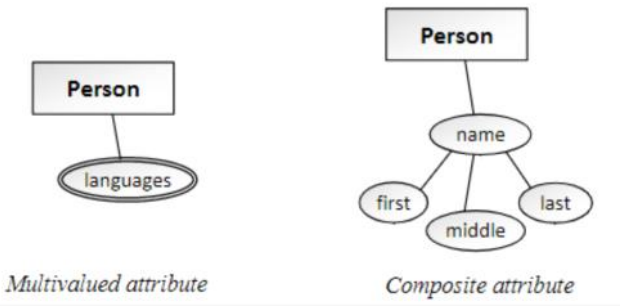
- https://github.com/Prashansa-Singh/INFO20003-Sem1-2022

Have a read of the case studies for today's tute while waiting :)

## Multivalued & Composite Attributes

- **Multivalued attributes**
  - Can store multiple values
  - Values are all of the same type
    - e.g. phone numbers all have same data type

- **Composite attributes**
  - Have hidden structure
  - Can be broken down in multiple components
  - Usually of different types (but not always)
    - e.g. For "Address" composite attribute
    - Would use different data types to store postcode, street number, street name etc




*Multivalued attribute* *Composite attribute*

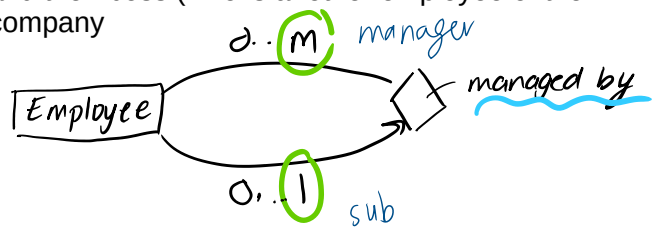
## Unary Relationships

- Unary relationship = a relationship between an entity and itself
- Relationship can be
  - one-to-one
  - one-to-many
  - many-to-many

- If the two sides of the relationship have different constraints e.g. one-to-many :
  - Remember to **label** the two sides of the relationship!

**Examples**

- the “managed by” relationship between an employee and their boss (who is another employee of the company




- What are the lower and upper bounds of each side of the relationship?

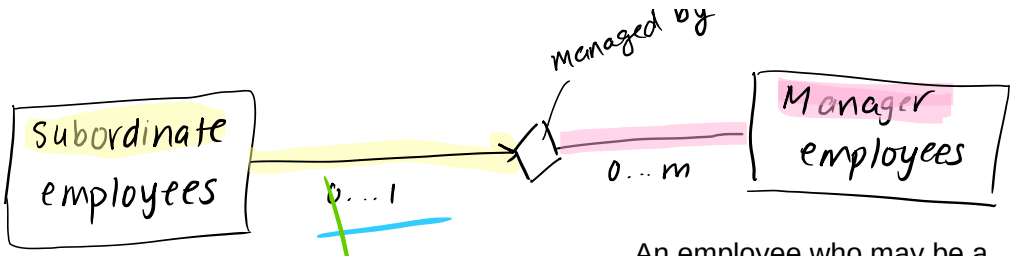
**How to read:**

- An employee can be managed by 0 to 1 employees
- An employee can manage 0 to many employees

- Label the relationship if needed
  - Easy way to figure out where the labels go:
  - Think about it as a **binary** relationship:

The two roles of an employee can be

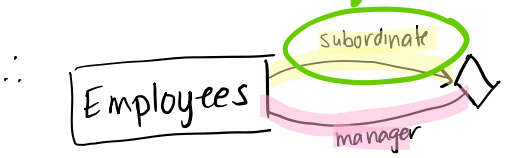
- subordinate or manager:
- (but there are employees who don't have a manager and employees that don't have any subordinates. hence, lower bounds are 0 for both)




An employee who may be a subordinate is managed by **0 to 1** manager employees

An employee who may be a manager can manage **0 to many** subordinate employees

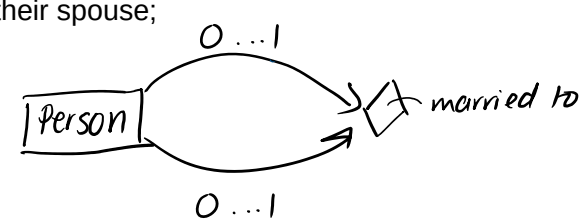
Add labels to each side of the relationship corresponding to the entity on the same side



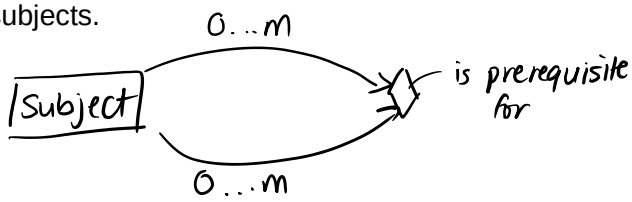

An employee can be managed by 0 to 1 employees.
An employee can manage 0 to many employees.

**Other Examples!**

- the “married to” relationship between a person and their spouse;




- the “prerequisite” relationship between university subjects.




- **Classify** the type of relationship

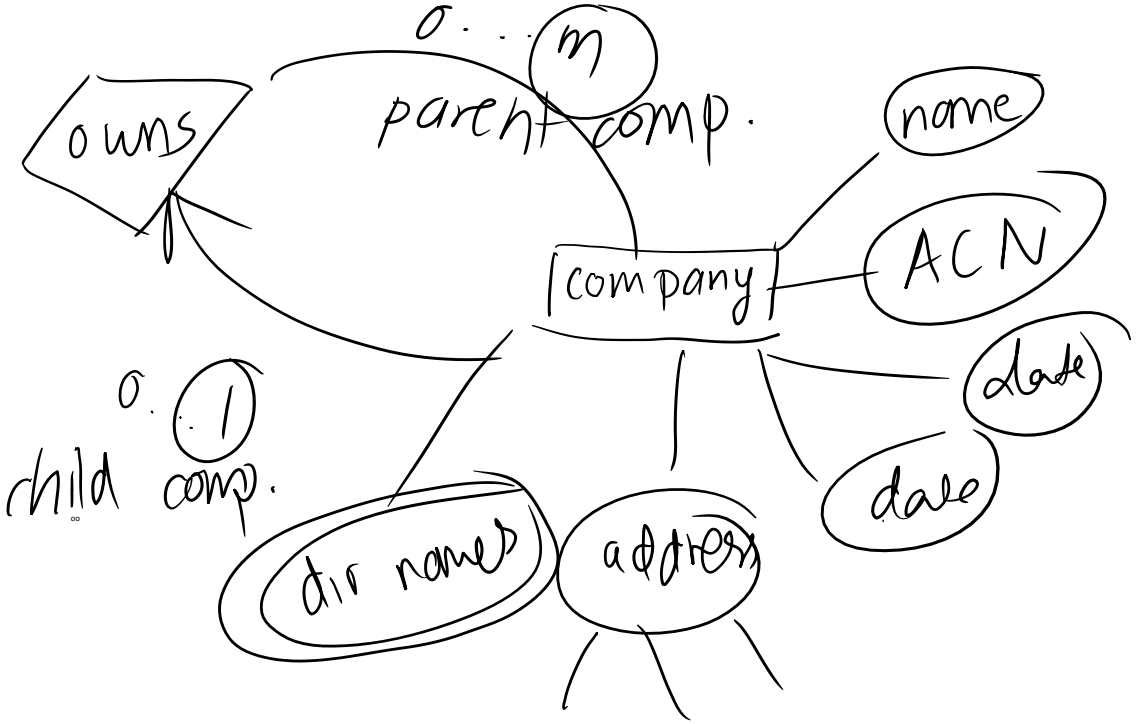
- How to read:
  - **Married to**
    - A person can be married to 0 to 1 people
    - A person can marry 0 to 1 people
  - **Prerequisite**
    - A subject can be a prerequisite for 0 to many subjects

- A subject can have 0 to many prerequisite subjects

**2. Practising these concepts:**

Australia's corporate regulator, ASIC, stores a range of information about companies, including the name, the nine-digit ACN (Australian Company Number), the date of registration and deregistration, and the names of the company's directors. Every company has a registered address, made up of the street address, suburb, state and postcode. A company may be owned by another company; in this situation ASIC keeps track of the company's parent company.

Use this information to model a "company" entity using Chen's notation.



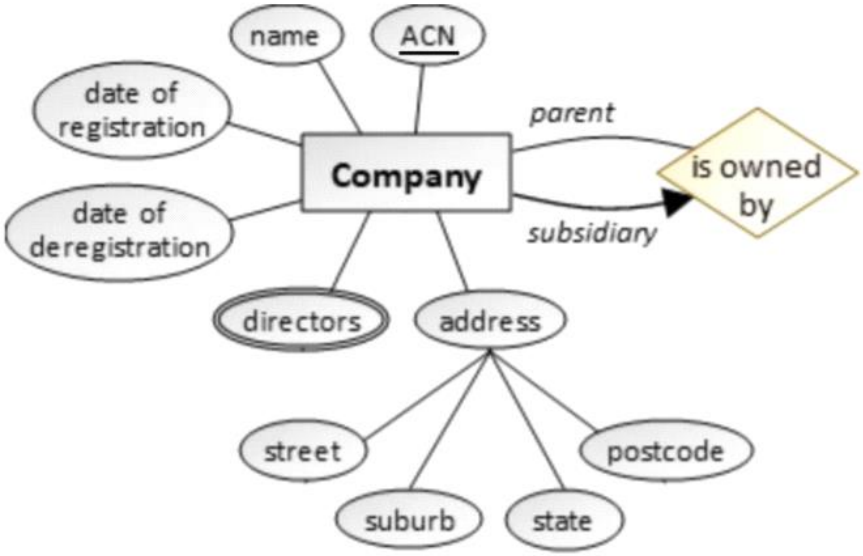

How to figure out labels:
- A company can own 0 to many companies
- A company can be owned by 0 to 1 companies


A company (which may be a parent company) can own 0 to many child companies

A company (which may be a child company) can be owned by 0 or 1 parent comapnies.

There may be companies that have no child companies.

There may be companies that have no parent companies. Hence, both lower bounds are 0.




**3. Consider the following case study:**

A bus company owns a number of buses. Each bus is allocated to a particular route, although some routes may have several buses. Each route passes through a number of towns. One or more drivers are allocated to each stage of a route, which corresponds to a journey through some or all of the towns on a route. Some of the towns have a depot where buses are kept – each bus always returns to its allocated depot at the end of the day.

Each of the buses is identified by its registration number and can carry different numbers of passengers, since the vehicles vary in size and can be single or double-decked. Each route is identified by a route number and information is available on the average number of passengers carried per day for each route. Drivers have an employee number, name, address, and sometimes a telephone number, and the names of the training courses they have completed need to be stored.

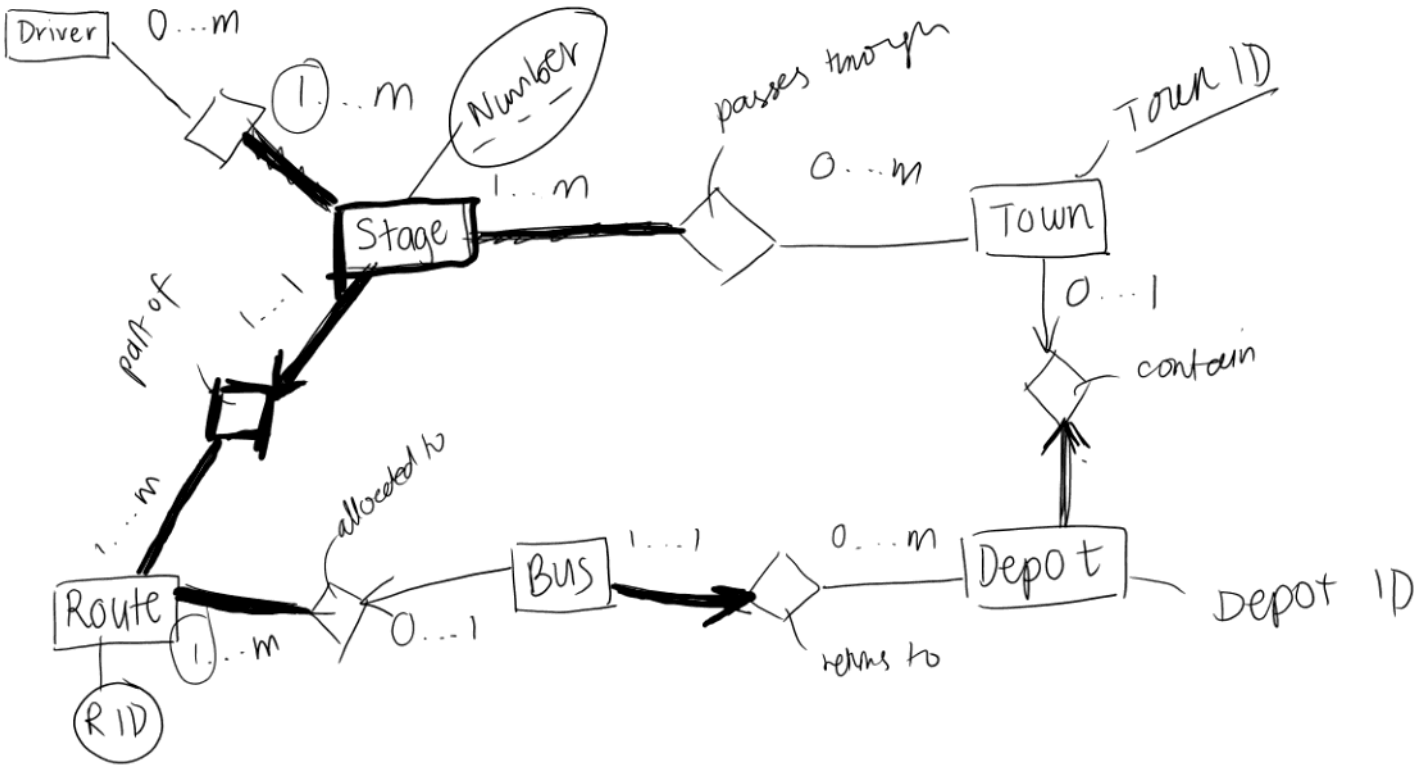
a. Identify the entities.
b. Identify the relationships (use business rules to identify relationships). State all the key constraints and participation constraints.
c. Draw a conceptual model and populate entities with appropriate attributes (use Chen's notation).
d. Discuss the logical modelling of the Driver entity.

**Group work**

- Important to focus on entities & relationships
  - Don't spend too much time writing in attributes
- But think about primary keys, multivalued and composite attributes

Many possible conceptual models are acceptable!
Subjective!
This is just one possibility

1) Highlight the entities we see in the text




- What PKs can we see in the text
  - What do we do about the entities that don't have PKs?
  - Are there any weak entities?

- Let's look closely at the Driver entity
  - Does it have any multivalued or composite attributes?

**Some constraints to note:**

- Realise that stage of route is an entity
  - this can be difficult because we aren't told any attributes about it
  - But because it has relationships with other entities e.g. drivers, it is good to model it as an entity
    - e.g. text says "drivers allocated to each stage of route"

- Each stage can have more than one driver as one driver can drive through more than one stage
- Driver may not be allocated to any stage of a route
  - e.g. driver is new or on-leave

- Each stage of route goes through at least one town

- A bus may be assigned to 0 to 1 routes
  - e.g. bus under repair
  - Should lower bound be 0 or 1?
    - Ambiguous - both are ok.
    - Justify your assumptions

- All routes and route stages are active. They all must have at least 1 bus and 1 driver
  - We only are storing the active bus routes in our model of a bus company. Wouldn't be very useful to store inactive routes

- A stage of a route must pass through at least 1 town
  - As bus stops must be located in a town
- But some towns may have no busses passing through
  - e.g. the town only has a depot but no bus stops

- Depot can hold 0 to many busses (0 because depot might be new and have no allocated buses yet)
- Each bus must be assigned to a depot
  - Otherwise where would we store it while it is not being driven?
  - So lower bound is 1

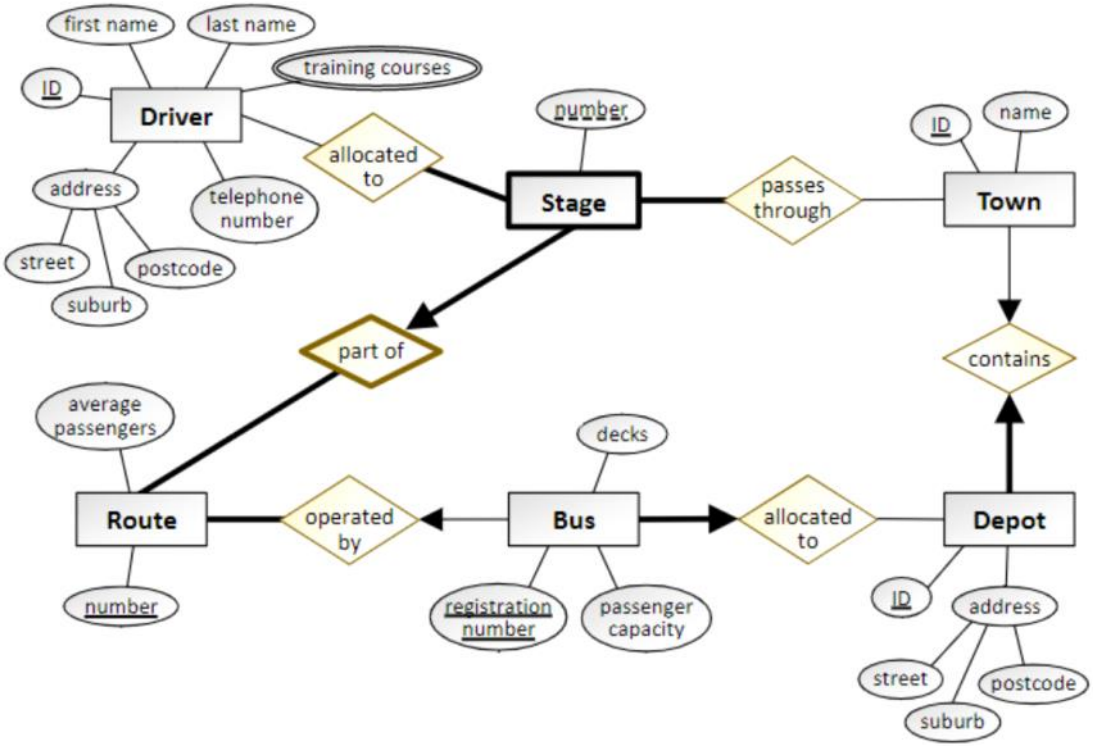
- You could argue that Depot could be stored as weak entity. Just write in your assumptions

- But what might be a good partial key? The text hasn't specified it
- Depot Number?
  - But there's only at most one depot per town so all of them would have number 1
  - So it's not very useful to know the Depot number
- Town Name/ID?
  - But we are already getting this as a FK in the logical model because there of the relationship between Town and Depot

- We have taught in the lectures that weak entities have partial keys. **But it's actually ok if weak entites don't have any partial keys sometimes.**
  - For this Depot example, it can still be uniquely identified by the Town ID (PFK) it will get in the logical/physical design stages
    - This is because there's at most 1 depot in a town.
  - But for the Cinema-Screen example last week, there can be multiple screens in one cinema. So we do need a partial key like Screen number

- So it's a choice you can make
  - Either make Depot weak
  - Or just have Depot as a strong entity as done in the solutions

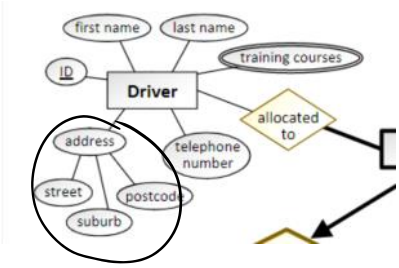
- Usually we avoid circular dependencies in models
  - But sometimes it's ok
  - There's no way to store all the information we need to without doing this
  - Try removing one of these relationships and notice we will lose information




d. Discuss the logical modelling of the Driver entity.

**Rules - To go from conceptual to logical:**

- **Flatten multivalued and composite attributes**
- **CamelCase**




- **Composite attributes** are resolved by adding the component parts of the attribute directly to the entity.

  Driver (DriverID PK, FirstName, LastName, AddressStreet, AddressSuburb, AddressPostcode, PhoneNumber)

- **Resolving multivalued attributes**
  - 2 Ways
  - Method 1:
    - If small and known number of values for the attribute
    - e.g. work phone, home phone and mobile phone
    - **Can resolve in same way as a composite attribute.**

- Method 2:
  - If number of values is unknown and/or large/infinite
    - Need to know number of columns in schema beforehand
    - Can't just keep adding columns as we go
  - Can create a **lookup table**!
    - Create a **new** table with a PFK referring to the PK of the table that the multivalued attribute belongs to
    - And the column storing the values of the multivalued attribute is also made a PK

For training courses, there is no limit to the number of courses a driver can take. So must create a new 'DriverCourses' table:

**DriverTrainingCourses**
**(DriverID PFK,TrainingCourseName PK)**

Both should be PKs as drivers can take multiple courses and the same course can be completed by multiple drivers.

**So final logical design of Driver is:**

**Driver** (DriverID PK, FirstName, LastName, AddressStreet, AddressSuburb,AddressPostcode, PhoneNumber)

**DriverTrainingCourses**
(DriverID PFK,TrainingCourseName PK)

- **Classify all the relationships**
- **Will do rest of logical and physical models for this case inside this week's Lab**