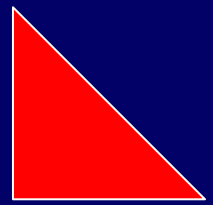
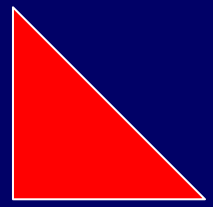


Data Analysis 1

Unit 2.1

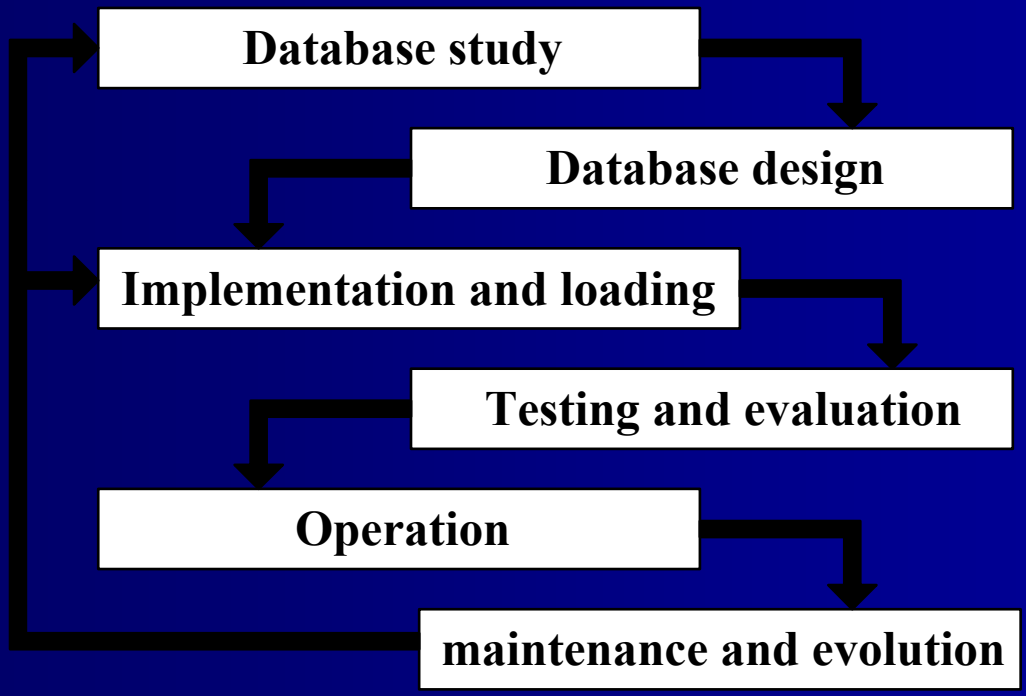
Entity Relationship Modelling



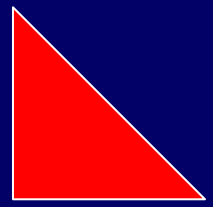
Overview

- Database Analysis Life Cycle
- Components of an Entity Relationship Diagram
- What is a relationship?
- Entities, attributes, and relationships in a system
- The degree of a relationship
- Construct an Entity Relationship Diagram

DB Analysis Life Cycle

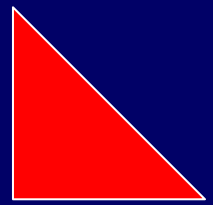


Database Analysis Life Cycle



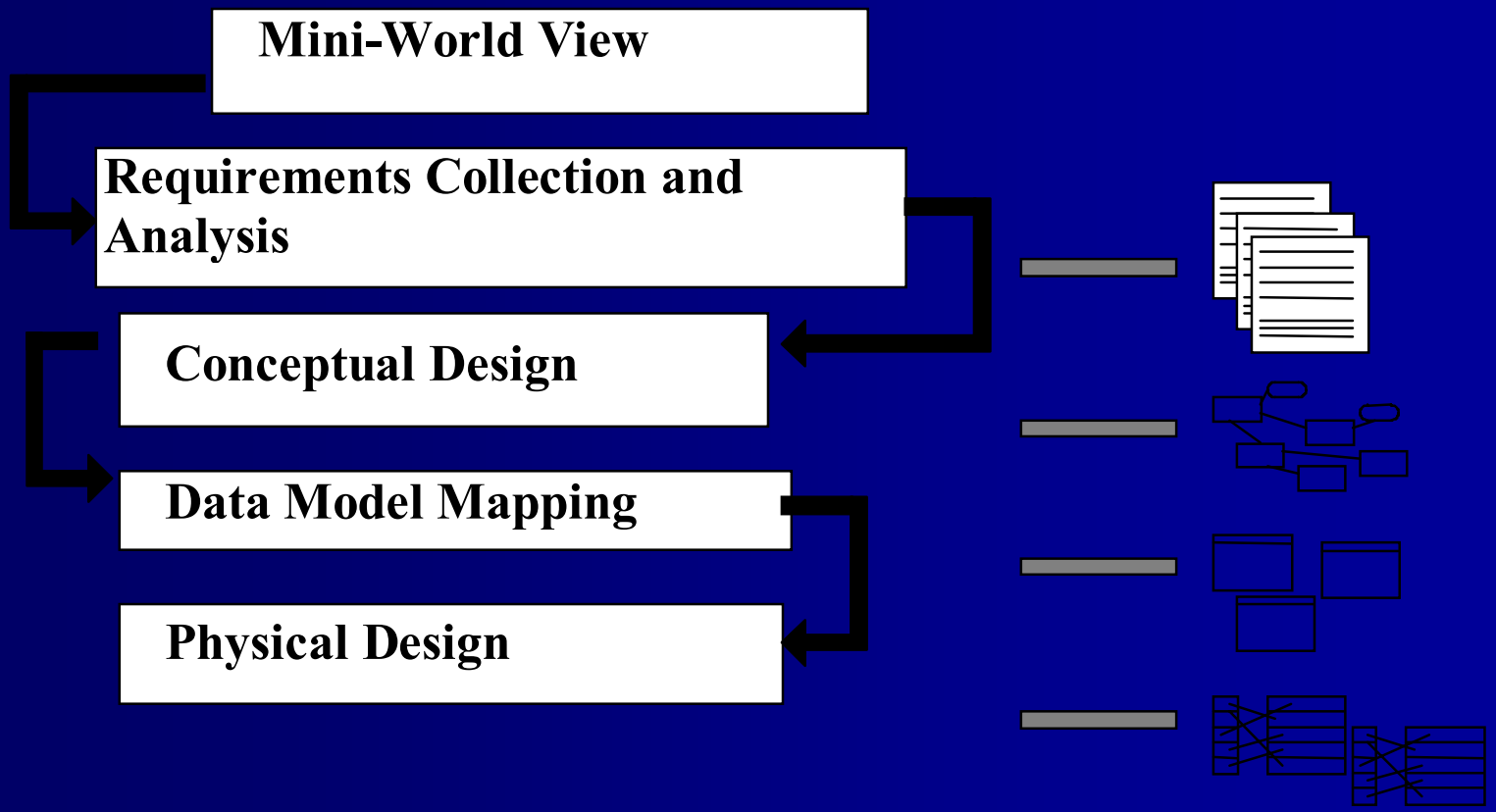
- Database Initial study
  - Analyse the company situation
  - Define problems and constrains
  - Define objectives
  - Define scope and boundaries
- Database Design
  - Create the conceptual design
  - Create the logical design
  - Create the physical design
- Implementation and loading
  - Install the DBMS and create the database(s)
  - Load or convert the data

Analysis Life Cycle continued...

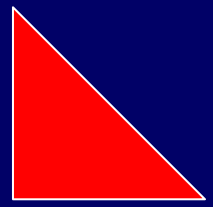


- Testing and evaluation
  - Test the database
  - Fine-tune the database
  - Evaluate the database and its application programs
- Operation
  - Produce the required information flow
- Maintenance and evolution
  - Introduce changes
  - Make enhancements

Database Design

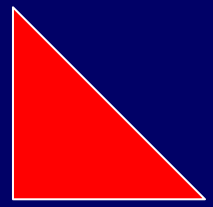


Database Design



- Conceptual Design
  - Entity Relationship modelling and normalisation
  - Data model verification
- DBMS software selection
- Data Model Mapping
  - Logical design
  - Translate conceptual model into definitions for tables, views...
- Physical design
  - Storage structures and access paths - optimize performance
  - Distributed database design

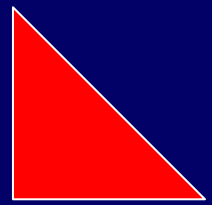
Entity Relationship Modelling



Entity Relationship (ER) modelling

- is a design tool
- is a graphical representation of the database system
- provides a high-level conceptual data model
- supports the user's perception of the data
- is DBMS and hardware independent
- had many variants
- is composed of entities, attributes, and relationships

Entities



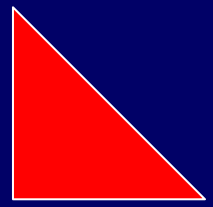
- An entity is any object in the system that we want to model and store information about
  - Individual objects are called entities
  - Groups of the same type of objects are called entity types or entity sets
  - Entities are represented by rectangles (either with round or square corners)

Chen's notation

other notations

- There are two types of entities; weak and strong entity types.

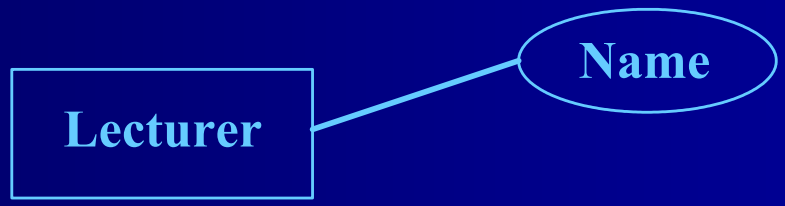
Attribute



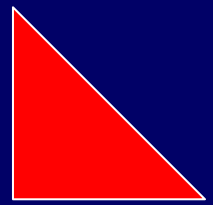
- All the data relating to an entity is held in its attributes.
- An attribute is a property of an entity.
- Each attribute can have any value from its domain.
- Each entity within an entity type:
  - May have any number of attributes.
  - Can have different attribute values than that in any other entity.
  - Have the same number of attributes.

Attribute cont...

- Attributes can be
  - simple or composite
  - single-valued or multi-valued
- Attributes can be shown on ER models
  - They appear inside ovals and are attached to their entity.
  - Note that entity types can have a large number of attributes... If all are shown then the diagrams would be confusing. Only show an attribute if it adds information to the ER diagram, or clarifies a point.

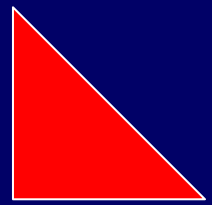


Keys



- A key is a data item that allows us to uniquely identify individual occurrences or an entity type.
- A candidate key is an attribute or set of attributes that uniquely identifies individual occurrences or an entity type.
- An entity type may have one or more possible candidate keys, the one which is selected is known as the primary key.
- A composite key is a candidate key that consists of two or more attributes
- The name of each primary key attribute is underlined.

Relationships



- A *relationship type* is a meaningful association between entity types
- A *relationship* is an association of entities where the association includes one entity from each participating entity type.
- Relationship types are represented on the ER diagram by a series of lines.
- As always, there are many notations in use today...

Relationships cont...

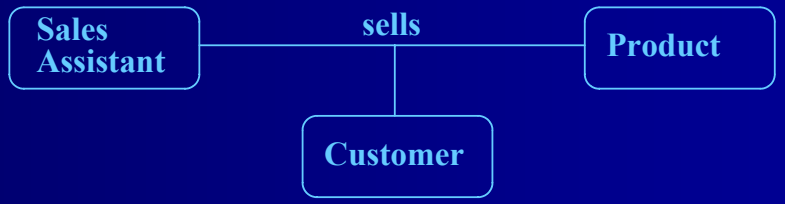
- In the original Chen notation, the relationship is placed inside a diamond, e.g. managers manage employees:

- For this module, we will use an alternative notation, where the relationship is a label on the line. The meaning is identical

Degree of a Relationship

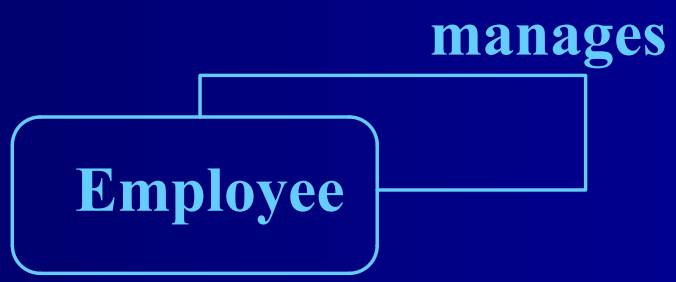
- The number of participating entities in a relationship is known as the degree of the relationship.
- If there are two entity types involved it is a *binary* relationship type

- If there are three entity types involved it is a *ternary* relationship type



Degree of a Relationship cont...

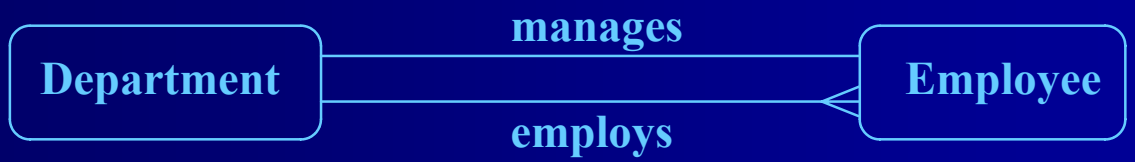
- It is possible to have a n-ary relationship (e.g. quaternary or unary).
- Unary relationships are also known as a *recursive* relationship.



- It is a relationship where the same entity participates more than once in different roles.
- In the example above we are saying that employees are managed by employees.
- If we wanted more information about who manages whom, we could introduce a second entity type called manager.

Degree of a Relationship

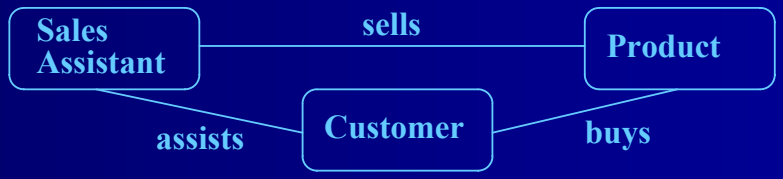
- It is also possible to have entities associated through two or more distinct relationships.



- In the representation we use it is not possible to have attributes as part of a relationship. To support this other entity types need to be developed.

Replacing ternary relationships

When ternary relationships occurs in an ER model they should always be removed before finishing the model. Sometimes the relationships can be replaced by a series of binary relationships that link pairs of the original ternary relationship.



- This can result in the loss of some information - It is no longer clear which sales assistant sold a customer a particular product.
- Try replacing the ternary relationship with an entity type and a set of binary relationships.

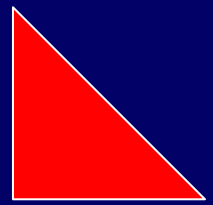
Replacing Ternary relationships cont...

Relationships are usually verbs, so name the new entity type by the relationship verb rewritten as a noun.

- The relationship *sells* can become the entity type *sale*.

- So a sales assistant can be linked to a specific customer and both of them to the sale of a particular product.
- This process also works for higher order relationships.

Cardinality



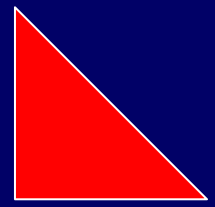
- Most relationships are not one-to-one
  - For example, a manager usually manages more than one employee
- This is described by the *cardinality* of the relationship, for which there are four possible categories.
  - One to one (1:1) relationship
  - One to many (1:m) relationship
  - Many to one (m:1) relationship
  - Many to many (m:n) relationship
- On an ER diagram, if the end of a relationship is straight, it represents 1, while a "crow's foot" end represents many.

Cardinality cont...

- A one to one relationship - a man can only marry one woman, and a woman can only marry one man, so it is a one to one (1:1) relationship

- A one to may relationship - one manager manages many employees, but eachemployee only has one manager, so it is a one to many (1:n) relationship

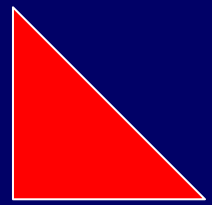
Cardinality cont...



- A many to one relationship - many students study one course. They do not study more than one course, so it is a many to one (m:1) relationship

- A many to many relationship - One lecturer teaches many students and astudent is taught by many lecturers, so it is a many to many (m:n) relationship

Optionality

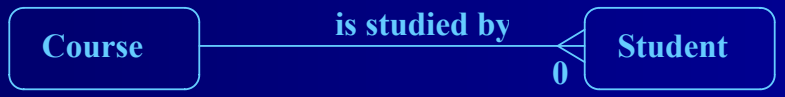


A relationship can be option or mandatory.

- If the relationship is mandatory
  - an entity at one end of the relationship must be related to an entity at the other end.
- The optionality can be different at each end of the relationship
  - For example, a student must be on a course. This is mandatory. To the relationship 'student studies course' is mandatory.
  - But a course can exist before any students have enrolled. Thus the relationship 'course is_studied_by student' is optional.
- To show optionality, put a circle or '0' at the 'optional end' of the relationship.

Optionality cont...

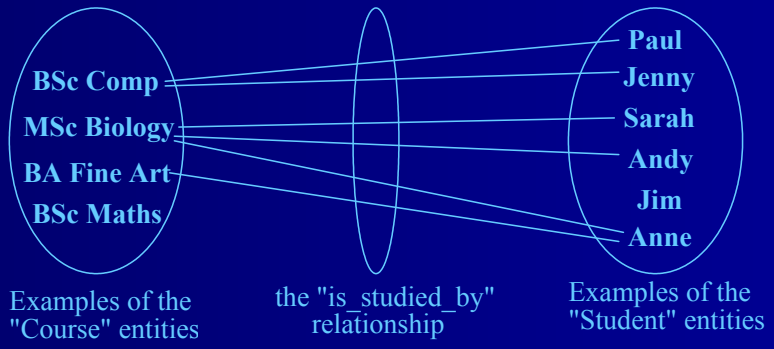
- As the optional relationship is 'course is_studied_by student', and the optional part of this is the student, then the 'O' goes at the student end of the relationship connection.



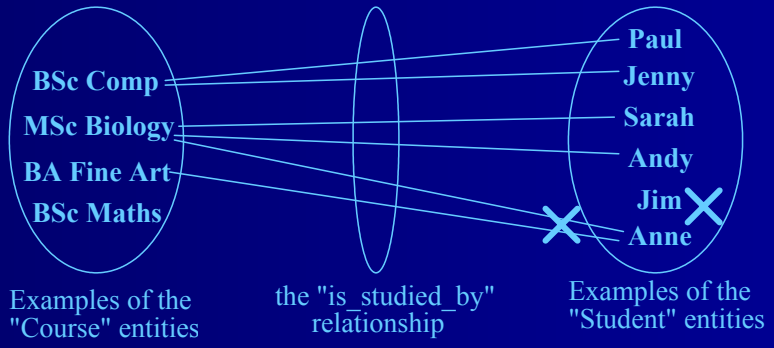
- It is important to know the optionality because you must ensure that whenever you create a new entity it has the required mandatory links.

Entity Sets

Sometimes it is useful to try out various examples of entities from an ER model. One reason for this is to confirm the correct cardinality and optionality of a relationship. We use an 'entity set diagram' to show entity examples graphically. Consider the example of 'course is_studied_by student'.

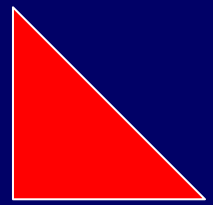


Confirming Correctness



- Use the diagram to show all possible relationship scenarios.
  - Go back to the requirements specification and check to see if they are allowed.
  - If not, then put a cross through the forbidden relationships
- This allows you to show the cardinality and optionality of the relationship

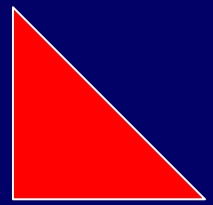
Deriving the relationship parameters



To check we have the correct parameters (sometimes also known as the degree) of a relationship, ask two questions:

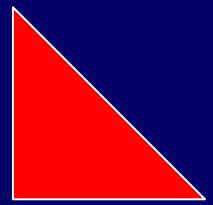
- One course is studied by how many students? Answer => 'zero or more'.
  - This gives us the degree at the 'student' end.
  - The answer 'zero or more' needs to be split into two parts.
    - The 'more' part means that the cardinality is 'many'.
    - The 'zero' part means that the relationship is 'optional'.
  - If the answer was 'one or more', then the relationship would be 'mandatory'.

Relationship parameters cont...



- One student studies how many courses? Answer => 'One'
  - This gives us the degree at the 'course' end of the relationship.
    - The answer 'one' means that the cardinality of this relationship is 1, and is 'mandatory'
  - If the answer had been 'zero or one', then the cardinality of the relationship would have been 1, and be 'optional'.

Redundant relationships

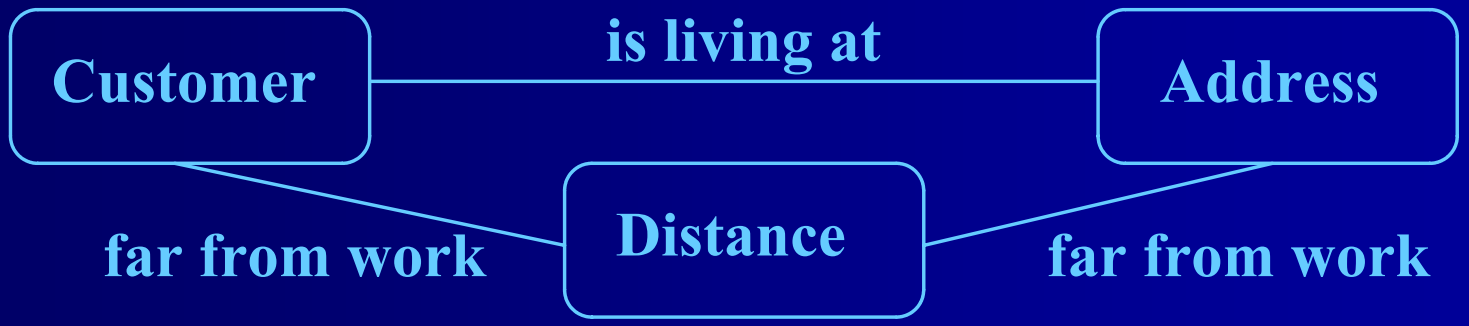


Some ER diagrams end up with a relationship loop.

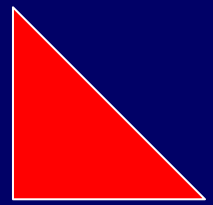
- check to see if it is possible to break the loop without losing info
- Given three entities A, B, C, where there are relations A-B, B-C, and C-A, check if it is possible to navigate between A and C via B. If it is possible, then A-C was a redundant relationship.
- Always check carefully for ways to simplify your ER diagram. It makes it easier to read the remaining information.

Redundant relationships example

- Consider entities 'customer' (customer details), 'address' (the address of a customer) and 'distance' (distance from the company to the customer address).



Splitting n:m Relationships



A many to many relationship in an ER model is not necessarily incorrect. They can be replaced using an intermediate entity. This should only be done where:

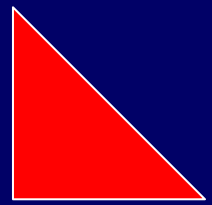
- the m:n relationship hides an entity
- the resulting ER diagram is easier to understand.

Splitting n:m Relationships - example

Consider the case of a car hire company. Customers hire cars, one customer hires many card and a car is hired by many customers.

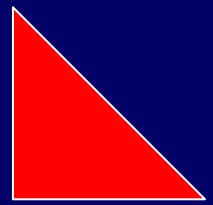
The many to many relationship can be broken down to reveal a 'hire' entity, which contains an attribute 'date of hire'.

Constructing an ER model - Entities



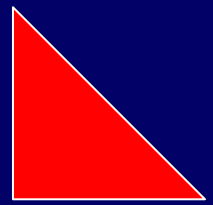
- Before beginning to draw the ER model, read the requirements specification carefully. Document any assumptions you need to make.
  1. Identify entities - list all potential entity types. These are the object of interest in the system. It is better to put too many entities in at this stage and them discard them later if necessary.
  2. Remove duplicate entities - Ensure that they really separate entity types or just two names for the same thing.
     - Also do not include the system as an entity type
     - e.g. if modelling a library, the entity types might be books, borrowers, etc.
     - The library is the system, thus should not be an entity type.

Constructing an ER model - Attributes



3. List the attributes of each entity (all properties to describe the entity which are relevant to the application).
   - Ensure that the entity types are really needed.
     - are any of them just attributes of another entity type?
     - if so keep them as attributes and cross them off the entity list.
   - Do not have attributes of one entity as attributes of another entity!
4. Mark the primary keys.
   - Which attributes uniquely identify instances of that entity type?
   - This may not be possible for some weak entities.

Constructing an ER model - Relationships



5. Define the relationships
   - Examine each entity type to see its relationship to the others.
6. Describe the cardinality and optionality of the relationships
   - Examine the constraints between participating entities.
7. Remove redundant relationships
   - Examine the ER model for redundant relationships.

ER modelling is an iterative process, so draw several versions, refining each one until you are happy with it. Note that there is no one right answer to the problem, but some solutions are better than others!