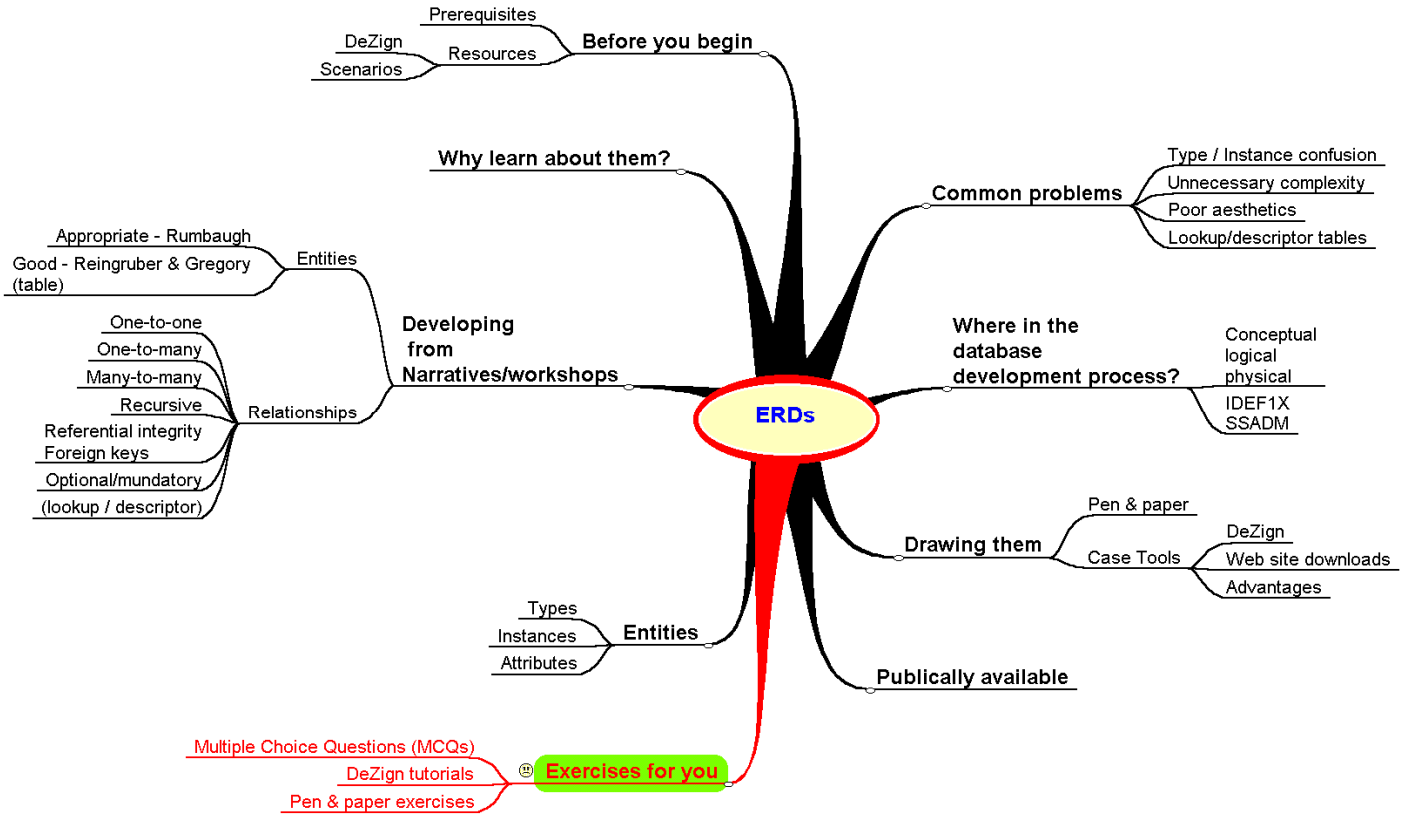
An Introduction to Entity Relationship Diagrams (ERDs)

Written by: Robin Beaumont e-mail: robin@organplayers.co.uk

Date last updated: 09/04/2007 14:23

Version: 5



How this document should be used:
This document has been designed to be suitable for web based and face-to-face teaching. The text has been made to be as interactive as possible with exercises, Multiple Choice Questions (MCQs) and web based exercises.

If you are using this document as part of a web-based course you are urged to use the online discussion board to discuss the issues raised in this document and share your solutions with other students.

Who this document is aimed at:
This document is aimed for two types of people:

- Those who wish to become involved in database development but are not interested in the nuts and bolts of programming, such people are commonly called domain experts and act a bridges between a professional group (e.g. medics, Solicitors etc) to which they belong and IT experts.
- As an introduction for those just beginning professional computer science courses

I hope you enjoy working through this document.

Robin Beaumont

Contents

1. Before you start

1.1 Prerequisites

This document assumes that you have the following knowledge and skills:

Skills:

That you have used the following features of a Database Management System (DBMS) such as Access to:

- Create Tables
- Create Relationships (and therefore know about the relationship window)
- Create simple Queries in the query design window
- Create a simple form

Knowledge:

You should also be able to describe what the following concepts mean:

- Tables, indexes and Fields
- Relationships (not a detailed description)
- Forms
- Queries

If you have completed the ECDL (European Computer Driving Licence) you will have covered these topics. If not I recommend that you do so now. You can take the ECDL either at a local college (in the UK) or as a distance Learning course. There are also some very good books guiding you through the ECDL.

1.2 Required Resources

This document provides you with the knowledge and skills to be able to develop and draw ERDs using a pen and paper it does not describe how to create and maintain ERDs using a specialist tool such as a case tool (DeZign, MagicDraw, System Architect etc.). Please see http://www.robin-beaumont.co.uk/virtualclassroom/contents.htm section 11 for various CASE tool tutorials.

I do recommend that you either complete either the DeZign or MagicDraw tutorial while, you work through this document.

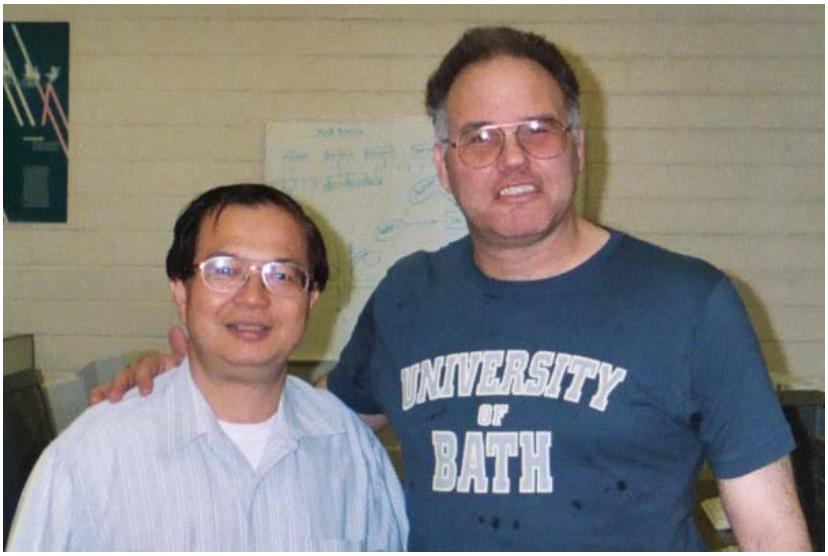
2. Learning Outcomes

This document aims to provide you with the following skills and information. After you have completed it you should come back to these points, ticking off those you feel happy with.

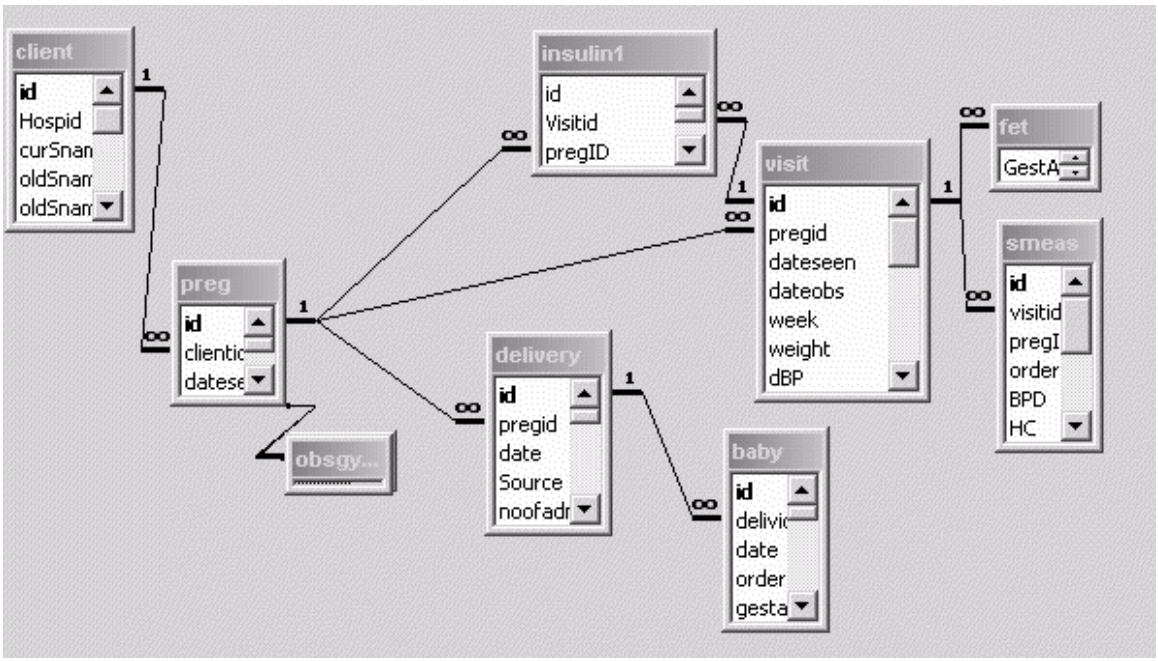
Learning outcome	Tick box
Be aware of the history of ERD diagraming	
Be able to list the reasons why the ability to develop ERDs is important for database developers and those involved in database development	
Be able to provide a definition and example of an entity type	
Be able to provide a definition and example of an entity (instance)	
Be able to provide a definition and example of an attribute	
Be able to explain the relationship between entity type, entity instance and attribute	
Be able to discuss the importance of context in identifying entity types	
Be able to produce a list of candidate entity types from a narrative description	
Be able to discuss the various criteria you may use to help refine an initial list of entity types	
Be able to suggest criteria that entity types should conform to	
Be aware of the use of workshops and informal ERDs to develop ERDs in a group setting	
Be able to provide a definition of relationship within the ERD context	
Be able to describe the Parent/Child concept	
Be able to provide an example of how the relationship concept is implemented within a relational DBMS (e.g. Access)	
Be able to explain the existence (mandatory) dependency constraint	
Be able to explain the optional dependency constraint	
Be able to describe what referential integrity is	
Be able to provide examples and explain what a one to one relationship is	
Be able to provide examples and explain what a one to many relationship is	
Be able to provide examples and explain what a many to many relationship is	
Be able to discuss the main issues in the confusion concerning conceptual, logical and physical data models	
Be able to evaluate the quality of an ERD from the perspective of undue complexity and aesthetics	
Be able to describe the pragmatic approach which is usually adopted when developing successive ERDs	
Be able to describe the additional functions a CASE tool offers over drawing programs	
Be able to provide examples of various models that already exist and are available to the public for a number of situations	
Be able to describe recursion in ERDs	
Be able to describe how recursion is often removed from ERDs as they are refined	
Be able to indicate that the process of normalising to fifth business normal form removes recursive relationships	
Be able to contrast the main differences between small scale and large scale database projects	
Be able to describe a process for developing a small database	
Be able to demonstrate the central part ERDs play in all database development methods	

3. Introduction

This document will introduce you to a technique used to provide a diagrammatic description of certain aspects of the data requirements for a database. The technique is called entity relationship modelling and uses Entity Relationship Diagrams (**ERDs**) as the main method. It is a particular kind of data modelling and is one of the oldest techniques around having been developed in the 1970's by Peter Chen who is very much still alive and continues to produce important research (http://www.csc.lsu.edu/~chen/chen.html). You can read his original paper at http://www.csc.lsu.edu/~chen/pdf/erd.pdf. One of my online students was motivated enough to go and see him when he was working through a former version of this document (he did live close by).



You may have come across such diagrams already using a variety of databases including Microsoft Access. The example below shows an example from a database to collect research data from those who suffer from diabetes and are also pregnant (don't worry about understanding the diagram at this stage):



3.1 Why Learn About ERDs?

This document is very much about getting you to learn what ERDs are and how to produce them. You may ask why, so here are a few reasons I believe it is important for you to do so:

- These diagrams form the basis of most database design methods. If you ever are involved in database design (i.e. data modelling) you will therefore need to understand them because if they are wrong (i.e. the blueprint), the database that is built from them will also be wrong!
- These diagrams form the basis for several more trendy techniques that you will come across from systems developers and learn about in subsequent documents, most notably UML.

- These diagrams provide a method of analysing a situation that forces you to adopt a stance of a typical database modeller. By using them you begin to realise how their minds work and also begin to appreciate why some databases are so problematic.

These are only a few of the reasons that I personally believe this skill is so important.

Even if you do not intend to become a programmer or systems analyst you may want to become the type of person who is able to provide a bridge between your professional group – such as doctor, vet or solicitor – and those who develop or manage information systems. Such people are vitally important in developing usable information systems and there is a great shortage of them. Such a person is often referred to as a '**Domain Expert**'.

Before you start to learn what ERDs are and how to create them yourself there are some Multiple Choice Questions (MCQs) on the next page to see how much you have taken in so far.

Exercise 1. MCQs

1. From the list below choose **two reasons** why it is important for a 'domain expert' such as you to learn about the ERD method:

a. Provides insight into the mindset of database developers

b. Is the only method available to describe the data requirements

c. Provides credibility to IT personnel

d. Forms the basis of most data modelling techniques

e. Has been proved to be the most cost effective method of specifying data requirements

2. From the list below choose the **one option that describes the most desirable** 'domain expert' from the medical profession:

a. Someone who has developed several databases but knows little of database modelling or current issues in medicine

b. Someone who has little interest in how information may help the department

c. Someone who has problems working in a collaborative environment

d. Someone who has previously managed IT projects

e. Someone who has knowledge of data modelling techniques and currently works in the appropriate situation

3.2 What is an ERD?

Definition:

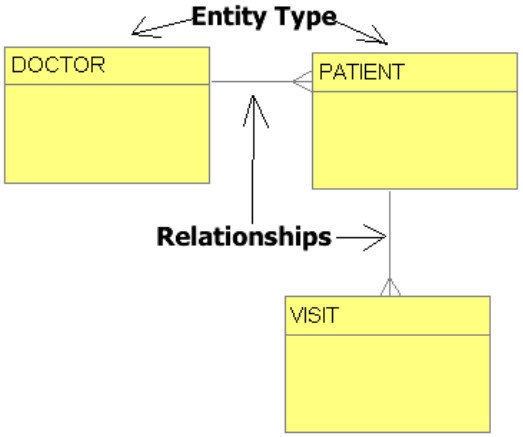
An ERD is a graphical description of the data for a particular database = a graphical data model. It represents the data at a high level of abstraction.

Some more clarification:

The term 'database' above usually implies a computer-based database; however, given the complexity of the ERD it might well be realised ('implemented') in the form of some type of paper based clerical system.

'High level of abstraction' means that it is not necessary to show details of the various fields or indexes, just the bare bones.

Opposite is a very simple ERD that shows three entity types and two relationships. On the following pages we will consider each of these elements separately.



Exercise 2. MCQs

1. ERDs provide a description of (one correct answer):

a. The processes that occur in the model
b. The various entities and their relationships in the model
c. The entities in the model
d. The processes and data requirements of a model
e. The User Interface aspects of the model

2. ERDs provide a (one correct answer):

a. A detailed description of the data within a data model
b. A detailed description of the proposed uses of a database
c. A guide to the training costs
d. A high level description of the data within a data model
e. A graphical picture that is of little use in developing a database

4. Entity types, instances and attributes

The first question is: what is an entity? However, unfortunately this is not an easy question to answer as the term 'entity' has been used to mean two different things. I have tried below to present these two different meanings as 'entity instance' and 'entity type', adapted from Reingruber & Gregory 1994.

Some definitions of Entities and Entity Types

"An entity is a 'thing' or 'concept' or 'object'. Well, most of the time".

An entity [type] is not a single 'thing' but rather a representation of like or similar things that share characteristics (properties). For example, King Lear and Hamlet are both plays and have properties such as name, author, cast of actors and lines of verse. The entity [type] describing these might be PLAY, with King Lear and Hamlet as examples of instances or occurrences of PLAY [i.e. each is an entity [instance]].

Entities involve information. The 'things' that entities represent are things about which the enterprise wants or needs to retain information. Therefore, while a data model should be an accurate representation of the business and its rules, we should never forget that data modelling is generally a precursor to design and development of structures intended to collect, store and dispense data." (p64)

Every entity [instance] in an entity type has the same set of attributes (characteristics).

The following are some definitions provided by various writers also listed in Reingruber & Gregory 1994 p63:

Thomas Bruce (1992): "Any distinguishable person, place, thing, event, or concept, about which information is kept"

Peter Chen (1976): "A thing which can be distinctly identified"

CJ Date (1986): "Any distinguishable object that is to be represented in a database"

James Martin (1989): "...anything about which we store information (e.g. supplier, machine tool, employee, utility pole, airline seat, etc). For each entity type, certain attributes are stored"

The bottom line is that:

- **An entity type** is a template or blueprint about which you are interested in collecting information (e.g. man, desk etc).
- **An entity instance)** is a single example (=instance) of an entity type (e.g. man=Robin Beaumont; desk=my desk in the upstairs study etc). People often say this is a "real world example".
- Both of the above consist of a **name** and a set of **attributes**.

The above ideas go back to ancient Greece, but I won't bore you with such details now.

Attributes are basically characteristics. Traditionally these attributes were easily measurable such as weight, height, name (text used to be represented as a series of numbers – ASCII codes) etc. However there is no reason why an attribute could not be something like a passport photo or ECG recording.

A number of examples will hopefully make things a little clearer.

Doctor = Entity type

Why is this an entity type?

Because it as a name

Because it has a set of unique attributes i.e. surname, forename, date of birth, GMC number, salary, gender etc.

Why is this a type rather than an instance?

Because it possesses these attributes but they do not have specific values

Dr Dow smith (GMC number 39200456) born 19/12/1956 = entity instance of doctor

Why is this an entity instance of Doctor Entity type?

Because it has the **same set of attributes** as the doctor entity type i.e. surname, forename, date of birth, GMC number, salary, age etc.

And each attribute has a specific value

Another way of thinking about the bond between an entity type and instance is to think of the type as being the column headings (= attribute names) for a number of rows each of which is an instance of the entity type.

Doctor					
surname	**forename**	**date of birth**	**GMC number**	**salary**	**gender**
smith	Dow	19/12/1956	39200456	89K	male
Coates	Jill	03/05/1966	5748337	67K	female
Worsley	Alan	11/07/1970	578493	80K	male

Doctor entity type with 3 entity instances of Doctor

Doctor					
surname	**forename**	**date of birth**	**GMC number**	**salary**	**gender**

Doctor entity type with no entity instances of Doctor

Doctor					
surname	**forename**	**date of birth**	**GMC number**	**salary**	**gender**
smith	Dow	19/12/1956	39200456	89K	male
Coates	Jill	03/05/1966	5748337	67K	female
Worsley	Alan	11/07/1970	578493	80K	Male
Brown	John	29/10/1055	4958576	45K	Male

Doctor entity type with 4 entity instances of Doctor

Coates	Jill	03/05/1966	5748337	67K	female
Worsley	Alan	11/07/1970	578493	80K	Male
Brown	John	29/10/1055	4958576	45K	Male

3 entity instances of Doctor

Now can you see why an entity type is like a template for entity instances.

Below are some examples of entity types and instances for three different contexts:

A typical general practitioner planning on collecting information about patients:

Entity type	Entity instance example(s) providing values for some of the attributes
Doctor	Angus Wallace living in the UK
Patient	Joe Bloggs in Newcastle , Alan Smith London
Visit	12/06/2001 at 4pm Prospect House Newcastle
Prescription	Valium 5mg eight times a day
Advice leaflet	Chronic back pain leaflet given to Mrs Smith last friday

The university administrator thinking about developing a course database:

Entity type	Entity instance example(s) providing values for some of the attributes
Module Coordinator	Robin Beaumont, Joe Brand
Student	Paul Whatling
Tutor	Ruth Brown
Module	Introduction to Informatics to run in 2003

A garage collecting detailed information about car components:

Entity type	Entity instance example(s) providing values for some of the attributes
Spark Plug	Rover, model type 49532
Engine	122 Brake Horsepower
Cooling system	17 pints capacity
Gearbox	Second gear ratio 7.55:1
Propeller shaft	Hardy Spicer type

Are the above examples correct?

While the above examples probably list a selection of the important entity types for each of the contexts, there is no absolute way of being able to say they are correct. Rumbaugh et al (1991, p21) makes this important point when discussing object identification; which for the moment you can assume is the same as an entity instance:

“We define an object as a concept, abstraction, or thing with crisp boundaries and meaning for the problem at hand. Objects serve two purposes: they promote understanding of the real world and provide a practical basis for computer implementation. Decomposition of a problem into objects depends on judgment and the nature of the problem. There is no one correct presentation.”

This is clear from considering the last example concerning car component details. Assume that another garage owner who was less concerned with collecting detailed information for each component only wanted one item of information recorded for each component. In this instance we would probably have CAR as the entity type and each of the entity types listed above relegated to attributes of the CAR entity type. There is no definitive correct answer; we only know by talking to the actual garage owner. **Entities are largely defined by the context.**

Exercise 3. Entity Types and Instances

Spend five minutes filling in the tables below. List some entity types along with examples of their entity instances. Some of your entity types may have no or several instances. The important thing is that each instance possesses the same set of attributes as the entity type.

	Entity type name:					
Attributes names:						
Attributes values:						

	Entity type name:					
Attributes names:						
Attributes values:						

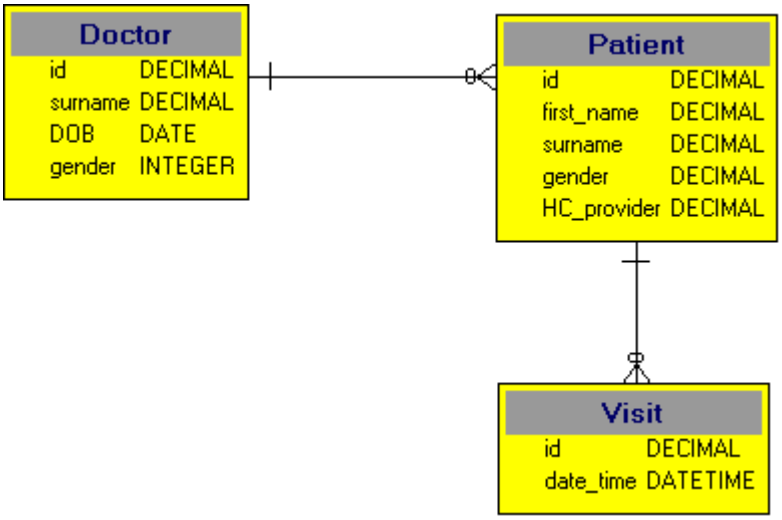
	Entity type name:					
Attributes names:						
Attributes values:						

	Entity type name:					
Attributes names:						
Attributes values:						

4.1 Showing Attributes in ERDs

The diagram below shows the ERD displayed in the previous section but with details of several attributes for each of the entity types. For example, the entity type DOCTOR has four attributes displayed: id, surname, DOB and gender. Beside each one you will notice a term such as DECIMAL or DATE etc. This is just indicating what type of data each attribute is.

To work successfully through this document you do not need to know the various data types that each attribute can take, such as decimal etc. You only need to be aware that you can show an ERD either just with the entity type names showing or details of the attributes as well; I personally prefer the former layout as it is far less cluttered.



You may have begun to feel that there is some similarity between the concepts we have been discussing above and various concepts you will have come across when using a database management system such as Access.

In fact there are two very basic similarities:

Table definitions (in a database) = Entity types (in an ERD)

Records in a table (in a database) = Entity instances (in an ERD)

It must be realised that this is only a rough guide as there are numerous exceptions to the rule, but it does help to think of the concepts in the above manner.

4.2 Summary

- An ERD consists of entity types and relationships.
- An entity type may be bound to zero or more entities instances (you can therefore have an entity type with no instances).
- An entity type possesses a number of attributes (e.g. a GP possesses name, date of birth, GMC number, salary, etc).
- Each entity instance possesses the same set of attributes as the associated entity type, where now each attribute possesses a particular value (e.g. GP = Smith; date of birth = 21/09/78, etc).

We will now look in more depth at the very important task of identifying entity types for a particular context, but before that here are some MCQs for you to answer.

Exercise 4. MCQs

1. Which of the following provides the best description of an entity **type** (select one)?
   a. A specific concrete object with a defined set of processes (e.g. John Brown with diabetes)
   b. A value given to a particular attribute (e.g. height - 230 cm)
   c. A thing that we wish to collect data about where zero or more, possibly real world examples of it may exist
   d. A template for a group of things with the same set of characteristics that may exist in the real world
   e. An undefined concept that needs further clarification

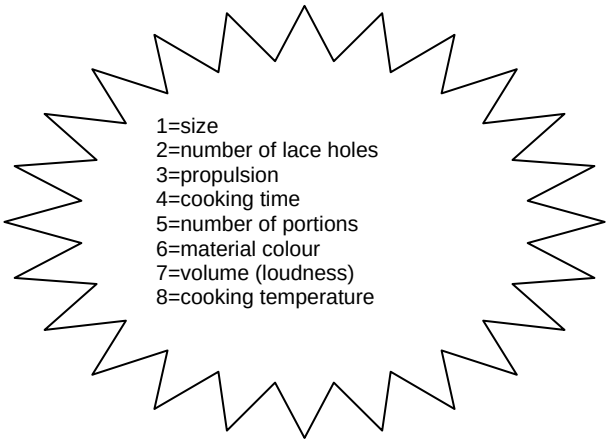
2. Which of the following provides the best description of an entity **instance** (select one)?
   a. A specific concrete object with a defined set of processes (e.g. John Brown with diabetes)
   b. A value given to a particular attribute (e.g. height - 230 cm)
   c. A thing that we wish to collect data about where zero or more, possibly real world examples of it may exist
   d. A template for a group of things with the same set of characteristics that may exist in the real world
   e. An undefined concept that needs further clarification

3. From the following list, select **two** entity instances:
   a. Steinway piano model D design template
   b. McGill type forceps as used in surgical operations
   c. Tony Blair (British prime minister)
   d. PIII type chip that is found in many modern PCs
   e. An advice leaflet for chronic back pain issued to Mrs Smith on 02/10/2002

4. Which one of the following statements is true (select one)?
   a. An ERD can only display entity type names.
   b. Attributes are a rarely considered aspect of entities.
   c. Attributes must always be displayed in ERDs.
   d. Attributes equate to table names in a database.
   e. Attributes can optionally be displayed in ERDs.

5. Select from the following the best list of attributes for the entity SHOE for someone working in a shoe shop (select one answer):
   a. 1, 2, 6
   b. 1, 2, 6, 3
   c. 1, 2
   d. 1, 6
   e. 1, 2, 3, 4, 5, 6

6. Select from the following the best list of attributes for the entity RECIPE for someone following a recipe at home (select one):
   a. 4, 5, 7, 8
   b. 4, 5
   c. 4, 5, 6
   d. 4, 5, 8
   e. 4, 5, 6, 8



4.3 Identifying Entity Types

While there is much written concerning ERD theory, far less has been written about actually developing the models. Two of the most common ways of identifying entity types for developing ERDs (i.e. data models) are:

- Narrative descriptions of requirements
- Workshops

Both of the above methods rely to some extent on **expert domain knowledge** to identify additional, and reclassify identified, entity types. We will now look at the first method, using a narrative description of the system requirements on which to base the development of an ERD.

Rather than give a brief description of each stage and then visit each in turn, I will just take you on a journey and then reflect upon what we will have done in a summary at the end. So now let's start with the narrative description.

4.3.1 Analysing a Narrative Description of Requirements

Frequently the first thing to be produced when someone has the idea of developing a database is a paper document describing what he or she wants. This often includes something like the following:

We wish to develop an information system (database) for our hospital in which:

"Patients are treated in a single ward by the doctors assigned to them. Usually each patient will be assigned a single doctor, but in rare cases they will have two. Patients either pay for their treatment directly or through an insurance company.

Healthcare assistants also attend to the patients, and a number of these are associated with each ward.

Initially the system will be concerned solely with drug treatment. Each patient is required to take a variety of drugs a certain number of times per day and for varying lengths of time.

The system must record details concerning patient treatment and staff payment. Some staff are paid part time, and doctors and care assistants work varying amounts of overtime at varying rates (subject to grade).

The system will also need to track what treatments are required for which patients and when, and it should be capable of calculating the cost of treatment per week for each patient.

When users use the system they will be able to print out as well as view on screen the results. (Although it is currently unclear to what use this information will be put.) "
(taken from http://www.umsl.edu/~sauter/analysis/er/er_intro.html and expanded)

The first stage is to pick out all the nouns (names) in the above passage; this provides a good baseline from which to consider possible entity types.

Exercise 5. Marking Nouns

> Mark in the above narrative all the nouns (names) you see.

You can see in bold below the possible set of nouns (names) that I have come up with:

*"**Patients** are treated in a single **ward** by the **doctors** assigned to them. Usually each **patient** will be assigned a single **doctor**, but in rare cases they will have two. **Patients** either pay for their **treatment** directly or through an **insurance company**.*

***Healthcare assistants** also attend to the **patients**, a number of these are associated with each **ward**.*

*Initially the **system** will be concerned solely with **drug treatment**. Each **patient** is required to take a variety of **drugs** a certain number of **times** per **day** and for varying **lengths** of **time**.*

*The **system** must record **details** concerning **patient treatment** and **staff payment**. Some **staff** are paid part **time** and **doctors** and **care assistants** work varying amounts of **overtime** at varying **rates** (subject to **grade**).*

*The **system** will also need to track what **treatments** are required for which **patients** and when and it should be capable of calculating the **cost** of **treatment** per **week** for each **patient**.*

*When the **users** use the **system** they will be able to print out as well as view on **screen** the **results**. (Although it is currently unclear to what **use** this **information** will be put.)"*

(taken from http://www.umsl.edu/~sauter/analysis/er/er_intro.html and expanded)

Gathering together all the above nouns produces the following list, not in any specific order:

• Patients	• System
• Doctors	• Users
• Healthcare assistants	• Cost
• Care assistants	• Time
• Ward	• Day
• Drug treatment	• Lengths
• Drugs	• Details
• Patient treatment	• Overtime
• Treatment	• Rates
• Staff payment	• Grade
• Staff	• Week
• Insurance company	Results
• Screen	• Use
•	• Information

-

The next stage is to consider which of these nouns are appropriate entity types to include in the ERD.

4.3.2 Identifying Appropriate Entity Types

Rumbaugh et al (1991 p152-3, repeated in Blaha & Rumbaugh 2005 p 185 -6) provides some guidance as to how to identify appropriate entity types. He suggests that you consider the following issues:

- **Redundant entity types** Two entity types may express the same information (i.e. two names for the same concept, or **synonyms**). In the above example DOCTORS and HEALTHCARE ASSISTANTS might be considered to be just entity instances of the entity type called STAFF; however, this is unlikely in the above example because we know from our own expert domain knowledge that doctors are concerned with prescribing whereas healthcare assistants are not. In addition the information we plan to collect is specifically concerned with prescribing. We therefore consider the entity type STAFF to be redundant, at least for the time being; possibly when the system is developed further such considerations can be re-visited.
- **Irrelevant entity types** If an entity type has little or nothing to do with the problem, it should possibly be left out. In the above example one would need to clarify with the person requesting the system if they need information stored about INSURANCE COMPANIES. Also if the system is initially only concerned with drug treatments, is there any point collecting information about other treatments?
- **Vague entity types** In a narrative description, often words are used indiscriminately. In the above example the description states that *"Initially the system will be concerned solely with drug treatment"* yet the following paragraphs refer to PATIENT TREATMENT and also TREATMENT. Are these three different entity types or not? Again we can only find out by discussing this issue with the person who requested the database. We would probably suggest that we have two entity types called DRUG TREATMENT and TREATMENT where TREATMENT is concerned with information about any non-drug intervention and probably not included in the initial ERD.
- **Entity types that are really attributes** Often an initial entity type is an attribute. In the above, COST has been classified as an entity type yet it is probably an attribute of TREATMENT or DRUG TREATMENT or PATIENT (renaming it BILL).
- **Multiple roles become entity types** "The name of a entity type should reflect its intrinsic nature and not a role that it plays. For example, OWNER would be a poor name for a entity type in a car manufacturer's database. What if a list of drivers is added latter? What about persons who lease cars? The proper class is PERSON (or possibly CUSTOMER), which assumes various different roles, such as owner, driver, lessee." (Blaha & Rumbaugh 2005 p 186).
- **Implementation information** The ERD is concerned with defining the data that needs to be stored. It is not concerned with the hardware or processing of the data. Therefore, in the above example SYSTEM, SCREEN and USER can be ignored. Similarly RESULTS are an outcome of processing data. Perhaps from your knowledge of Access or other databases, you will realise that such things as RESULTS and reports are just a process of using a query or report within the database management system.

Another type of problem occurs resulting from the existence of **homonyms.** This is where two entity types with the same name actually mean different things. For example the entity TREATMENT may mean very different things to different healthcare professionals. In this instance it may be necessary to add one or more additional entity types to express the different concepts more clearly (e.g. DRUG_TREATMENT and ART_THERAPY etc).

So what have we ended up with after all the above deliberations? The following is the revised list of entity types:

- Patients, Doctors, Drug treatment, Drugs

By concentrating on the drug aspect of treatment and considering each of Rumbaugh's guidelines we have reduced the list from 27 to 4 for a possible initial ERD.

Exercise 6. Identifying Entity Types from a Narrative

Time: 60 minutes

Look through the DopeHead scenario, found in the scenarios handout, and:

1. Mark all the nouns and produce a list of them.

2. Using Rumbaugh's guidelines revise your list to identify possible entity types.

4.3.3 Good and Bad Entity Types

While the previous page was concerned with identifying entity types, we will now look at a few of the many guidelines suggested by Reingruber & Gregory 1994 (p65-77) for what makes good entity types.

Entity type names:

- Should be **unique** for the particular model. (This does not usually apply to attribute names which only need to be unique to a particular entity type.)
- Should be a **singular noun** (e.g. PATIENT not PATIENTS).
- Should be **self-explanatory** to those reading the ERD.
- Should follow any **naming conventions** locally defined by the modellers. For example, two common constraints are that:
    - They should be written in **UPPER CASE**.
    - Possible spaces should be represented by **the _ character** (eg DRUG_TREATMENT rather than DRUG TREATMENT).
- Should **not** be a name of an **individual object** (e.g. Freeman Hospital Newcastle upon Tyne).
- Should **not** express **more than one concept** (e.g. EQUIPMENT/BED).
- Should be **documented** in the system design specification. The description for each entity type should be clear, unambiguous and supplemented with examples.

When carrying out this process it is a good idea to draw up a table similar to the one below to make sure you carry out the process, the initial proposed entity type name on the left and the final entity type name on the right.

Initial entity type name:	Unique	Singular noun	Self-explanatory	UPPER CASE	The _ character	Not a individual object	Not more than one concept	Documented	Final entity type name
Patients									
Doctors									
Drug treatment									
Drugs									

Exercise 7. Clarifying Entity Types

Time: 60 minutes

1. Looking back at the hospital example, consider the following four possible entities. Use the above guidelines to edit them appropriately:

    Patients

    Doctors

    Drug treatment

    Drugs

2. Revisit the initial set of entity types you defined for the DopeHead scenario and edit them accordingly. It may help to draw up a table similar to the one above.

4.3.4 Warning about the 'is a Kind of' Situation

To start with, people are often confused about the relationship between entity types and entity instances thinking that an entity instance "is a kind of" entity type. An example will help to make this clearer. Someone might say that for the entity type ANIMAL entity instances in this case might be BIRD or HUMAN. This is not usually the case. The reason for this is because thinking about what information you might want to collect about BIRDs and HUMANs would probably be different in several respects for each of them. Would you really want to collect information about the size of wingspan for a HUMAN or the IQ of a particular bird? In other words the set of attributes for each is different. If you are unsure of this go back to Exercise three on page 11

Unfortunately it is not always as simple as this. Here is the argument for keeping BIRD and HUMAN as entity instances of HUMAN. Suppose you were an antique dealer and just wished to collect information about the instances of ANIMAL statue you had. In this case you would probably just want to instance of ANIMAL statues you had rather than animal specific, information in which the amount and type would vary for each. In this situation you are happy about keeping the same type of information about each of the statues regardless of what they represent, probably just species name would do for your attribute name. **The context dictates what entity types you will find**.

4.3.5 Workshops

The alternative to using a paper document to partially derive a set of entity types is to hold a series of interactive workshops. This requires commitment, including the willingness to learn, from the person(s) who have requested the system (i.e. database). They need to understand to a limited extent what you are learning now!

Often a mixed approach can be taken, using a paper document as the starting point, to a workshop. Running workshops requires good group working skills, and must be carried out in a non-threatening manner. Often the modellers need to show considerable tact when people get hung up on what the modellers themselves know to be irrelevant issues (e.g. font size/colour, type of screen or computer etc) for the particular task at hand.

Now that we have looked at how to create an initial list of entity types we will consider the actual mechanics of drawing entities.

4.3.6 Drawing ERDs

It must be remembered that database modelling, of which ERDs are a particular tool, is a discipline that has only been in existence for about a quarter of a century. I was recently looking at a book on systems analysis written in 1984 (Daniels & Yeates) and only managed to find 2 of the 300 pages dedicated to databases, and within those two pages no mention was made of ERD diagramming.

There are a large number of different ERD diagramming styles. And this document is not concerned with you knowing any particular one intimately but rather being aware of the general style and meaning of them.

Learning to draw ERD's is usually a pen and paper exercise to begin with and I would strongly encourage you to practice using this simple technique to begin with - be warned you'll also need a larger rubber! Drawing informal ERDs and discussing them is a very good way of developing them in a measured and appropriate manner, flip charts, or electronic white boards are excellent in this respect.

Other documents on my web site provide detailed tutorials on how to use specialised software (called case tools) to draw ERDs. http://www.robin-beaumont.co.uk/virtualclassroom/contents.htm

Exercise (8.) MCQ

1. Which of the following types of software (applications) is most suitable for developing ERDs?

a. Desktop drawing package (e.g. Smartdraw)
b. Desktop publishing application to ease the use of textual explanations
c. Anything with drawing capabilities (e.g. Microsoft Word)
d. A CASE tool
e. A spreadsheet

4.4 Summary

Specifically we have gone through the following stages to come up with a list of initial entity types for both the hospital and Dopehead scenarios and obviously we can also use this method to produce a list of candidate entity types from any narrative description. The list below should remind you of the steps you went through:

1. Identifying nouns and drawing up a list
2. Removing:
    a. Redundant entity types (e.g. synonyms)
    b. Irrelevant entity types
    c. Vague entity types
    d. Entity types that are really attributes
    e. Roles amalgamated into entity types
    f. Implementation information
3. Adding entity types due to homonyms
4. Considering Reingruber & Gregory 1994's guidelines:
    a. Initial entity type name
    b. Unique
    c. Singular noun
    d. Self-explanatory
    e. UPPER CASE
    f. The _ character
    g. Not an individual object
    h. Not more than one concept
    i. Documented
    j. Final entity type name

We could have also used a more interactive approach with clients in a workshop environment, and developed ERD's with them.

After a few MCQs we will move onto the second aspect of ERDs, the relationships. If you feel like one this would be a good time to take a break.

Exercise 9. MCQs

1. Which one of the following best describes the technique used to identify entity types in a narrative?

a. Identification of verbs, the application of various criteria and standards (eg naming conventions etc) to refine the list and then the creation of a list of appropriately named entity types
b. Identification of nouns then the application of ISO standards to create a list of appropriately named entity types
c. Identification of verbs then the application of standards (eg naming conventions etc) to create a list of appropriately named entity types
d. Identification of nouns, the application of various criteria (eg Reingruber & Gregory 1994's) and standards (eg naming conventions etc) to refine the list and then the creation of a list of appropriately named entity types
e. Identification of a few important nouns then the application of standards (eg naming conventions etc) to create a list of appropriately named entity types

2. Which of the following criteria are true (choose three)?

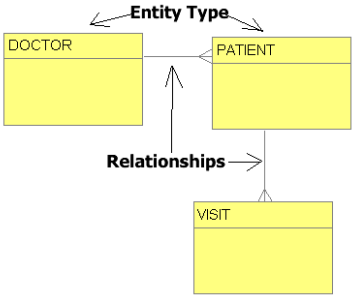
a. Attribute names must always be unique for a particular model.
b. Entity type names must be unique for a particular model.
c. Attribute names must be unique for a particular entity type within a model.
d. An entity type should only express one concept.
e. Entity type names should be singular or plural nouns.

Space for you to make additional notes about Entity types and Instances:

5. Relationships?

At the beginning of this document I stated that ERDs consist of entity types and relationships. We have already looked at entity types and will now move on to consider relationships. Just as with Entities you can consider Relationships at the Type or Instance level but because I find it easier to think of them at the instance level I will only discuss them at this level.

Relationships are shown on ERDs by lines, usually between entity types. A relationship is a method of linking together entity types. Within an ERD, each entity type has one or more relationships with a number of other entity types. A relationship is optionally (see Reingruber & Gregory 1994 p174) given a name which is a **verb** (action). For example:



DOCTOR **has** several PATIENT(s)

PATIENT **visits** a DOCTOR

The ERD opposite shows the two above relationships:

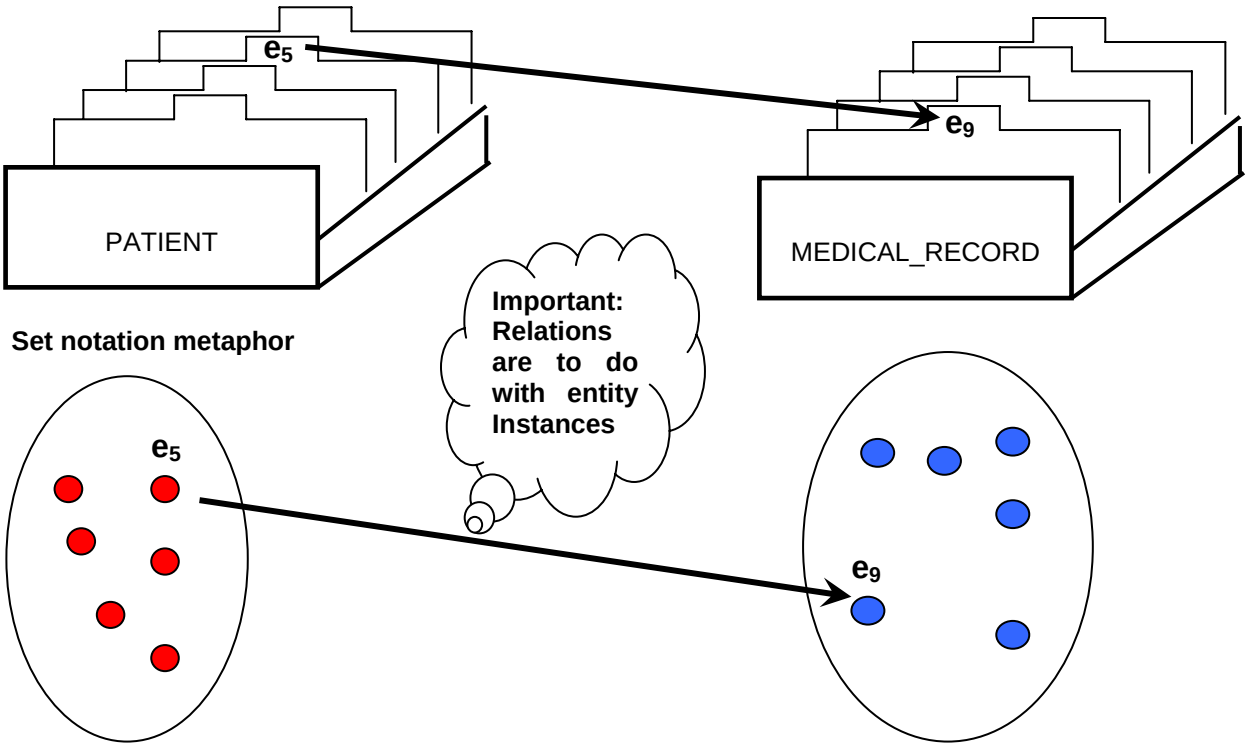
5.1 What Does it Mean?

The presence of a relationship between two entity types implies that one is linked to the other. One is said to be the **parent** while the other is the **child**. Consider the following situation:

PATIENT **owns** MEDICAL_RECORD

Here the situation is that a PATIENT has a relationship to MEDICAL_RECORD. I have illustrated this in two ways below (idea taken from Carter 1995 p34). PATIENT instance 5 (e_5) is associated with MEDICAL_RECORD instance 9 (e_9). You could substitute the words 'linked to' or 'related to' for 'associated with' in the previous sentence.

Index card metaphor



The section below is not usually included when discussing relationships within ERDs. This is because it is considered to be too low level; ERDs are meant to be at a high conceptual level. (Notice I have tried to avoid discussing attribute characteristics such as integers, text lengths etc) However, I have found when teaching this topic that students can only grasp the meaning of a relationship in an ERD when they are given a low level explanation. In other words, what does a relationship mean at the end of the day in a database such as Access?

5.2 Foreign Keys - How it all Works

Patient			
surname	**forename**	**date of birth**	**id**
jones	Amanda	19/12/1956	004
Oaks	gale	03/05/1966	005
kowal	Andrew	11/07/1970	006
bull	Chris	29/10/1055	007
....	....	...	...

Foreign key

Medical_record			
id	**source**	**date**	**Patient_ID**
9	GP	19/12/1956	005
10	St james hospital	03/05/1966	....
11	GP	11/07/1970	...
12	Istanbul	29/10/1055	...

So how do you actually relate two entity types? It is remarkably simple; you add a reference to the parent entity (instance) in the child entity instance by adding another attribute. This reference in the child entity is called a **foreign key**. To understand why it has this name, it is necessary to understand a little more about attributes.

Consider a typical set of attributes for a PATIENT entity type: name, address, date of birth etc. If we have a large group of PATIENT entity instances, it is unlikely that we could uniquely identify one from any one of these attributes. What we need is a unique identifier such as a 'patient number'. Such an attribute that possesses a unique value for each entity instance for a particular entity type is called a **primary key** it is often given the attribute name **ID**.

If we copy the particular value of this **primary key** into the child entity instances they will be linked. What we have done is to copy the key value into an entity foreign to it, hence the 'foreign'.

Taking the above example with PATIENT and MEDICAL_RECORD, the MEDICAL_RECORD entity instance 9 will possess a foreign key (attribute), probably called something like patient_id with the value 5 to link it back to the particular PATIENT entity instance with which it has a relationship.

5.3 What Name do you give a Foreign Key Attribute?

This is basically up to you. As a rule of thumb it is a good idea to give it the same name as the attribute (i.e. the primary key) from which it was copied. Two different entity types can have the same attribute name(s). In the above example the attribute called 'doctor_id' in the DOCTOR entity type can also exist in the MEDICAL_RECORD entity type. Sometimes it is sensible to add a prefix or suffix to the 'foreign' attribute indicating that the attribute is a foreign key such as 'FKdoctor_id'.

Do not worry too much about the foreign key idea at the moment. You will understand it at the end of the subsection when we have worked through a number of examples.

There are a number of types of relationships, each of which we will now investigate separately. But before that we will take a quick look at one important aspect of relationships: dependency.

5.4 What are Dependency and Referential Integrity?

Dependency is the concept of having something rely upon something else. For the reliance to be fulfilled the thing that one depends upon must exist. In ERDs most relationships are of the parent/child type where an instance of a child entity is dependent upon a particular instance of a parent entity (called existence dependency).

I previously described a relationship between the patient (parent) and medical record (child) entity types shown below. Each patient has a single medical record.

Repeating once again how this works; a value in the key index attribute PATIENT entity instance of 5 is copied into the appropriate medical record entity instance into a specific attribute called the foreign key. Note this is not an optional requirement for the link to exist; this must be the case. If you imagine the value being 'copied' into the foreign key attribute you can see how the term 'dependency' is so apt. You can't have a non-existent value copied into the MEDICAL_RECORD for a non-existent patient. Or can you? Let's consider some interesting possibilities:

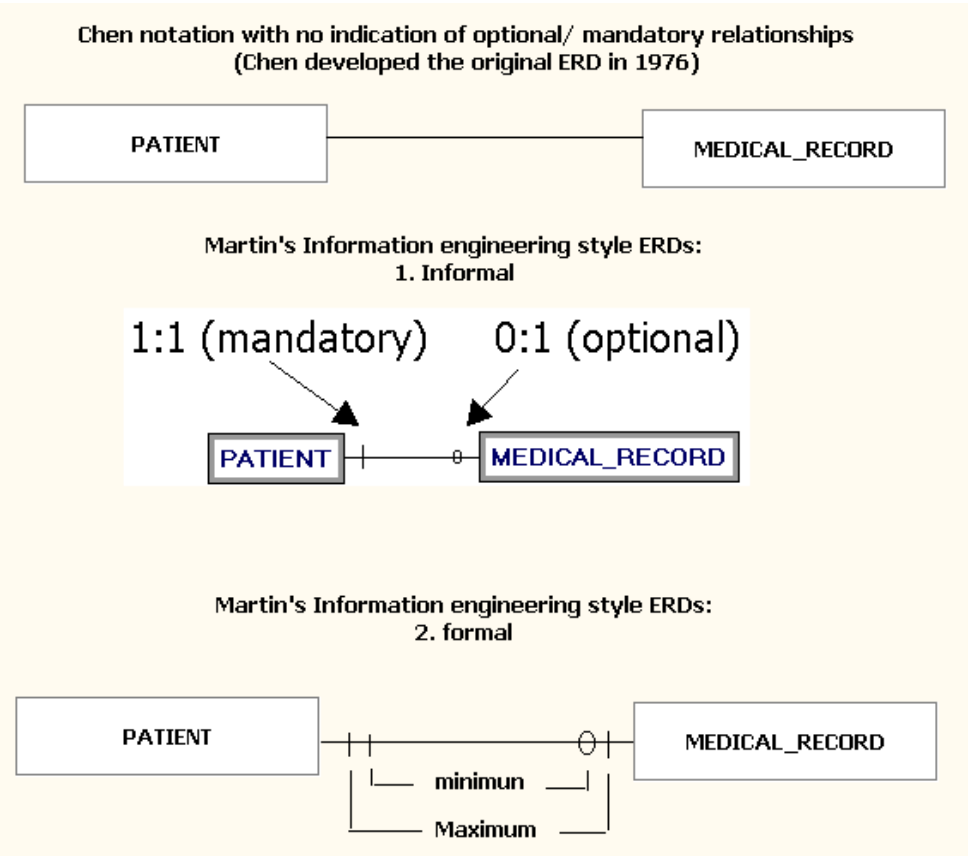
- If you allow the foreign key attribute to take an empty value (technically called a **null value** – for our purposes they are identical), a particular MEDICAL_RECORD instance can exist without an associated PATIENT instance linked to it. Is this sensible? This situation is called an **optional relationship**. According to Reingruber & Gregory 1994 (p 185) as well as other writers, such relationships should be avoided.
- If you disallow the foreign key attribute to take an empty value a particular MEDICAL_RECORD instance can only exist after the appropriate PATIENT instance has been created. (You can't copy what you don't have!) This is called a **mandatory relationship** (**also called an existence dependency**).
- If you change the value of the PATIENT key field in the relationship the MEDICAL_RECORD foreign key value will need to be updated appropriately to keep the relationship intact.
- If you delete the associated PATIENT entity instance what becomes of the orphaned child records that once were related to it? You have three options: delete all the child instances, change their foreign key values to null or do nothing with them. Clearly the third option is not appropriate, as it would make no sense because the whole purpose of the value is to indicate a relationship.

The actual database software in which a particular database resides is designed to manage all the above problems. It basically keeps all the foreign key values ('references') up to date and stops you from damaging the relationships once you have set up the database. This process is called maintaining **referential integrity**.

Now that we have some idea of dependency we will look at the simplest type of relationship, called a one to one relationship.

5.5 What is a One to One Relationship?

We have already looked at a one to one relationship on the previous page, that concerning a PATIENT and MEDICAL_RECORD. The important aspect of such a relationship is to ensure that you have established which is the parent and which is the child entity type by way of making it, if at all possible, a mandatory relationship. For example, in the above situation we say that a MEDICAL_RECORD entity instance cannot exist without it being associated with a PATIENT entity instance. However, it is possible for a PATIENT entity instance to exist without an associated MEDICAL_RECORD entity instance.



Mandatory one to one relationships are often indicated as 1:1 while optional one to one relationships are shown as 0:1.

The diagram below shows several different types of ERD notation. In the original ERD notation, optional/mandatory preferences were not indicated.

5.6 What is a One to Many Relationship?

This type of relationship is the most common within databases. Examples include:

- DOCTOR has many PATIENTs
- STUDENT takes many COURSEs
- PERSON has many ACCOUNTs
- GARDEN contains many PLANTs

Patient			
surname	**forename**	**date of birth**	**id**
jones	Amanda	19/12/1956	004
Oaks	gale	03/05/1966	005
kowal	Andrew	11/07/1970	006
bull	Chris	29/10/1055	007
....	....	...	...

Foreign key

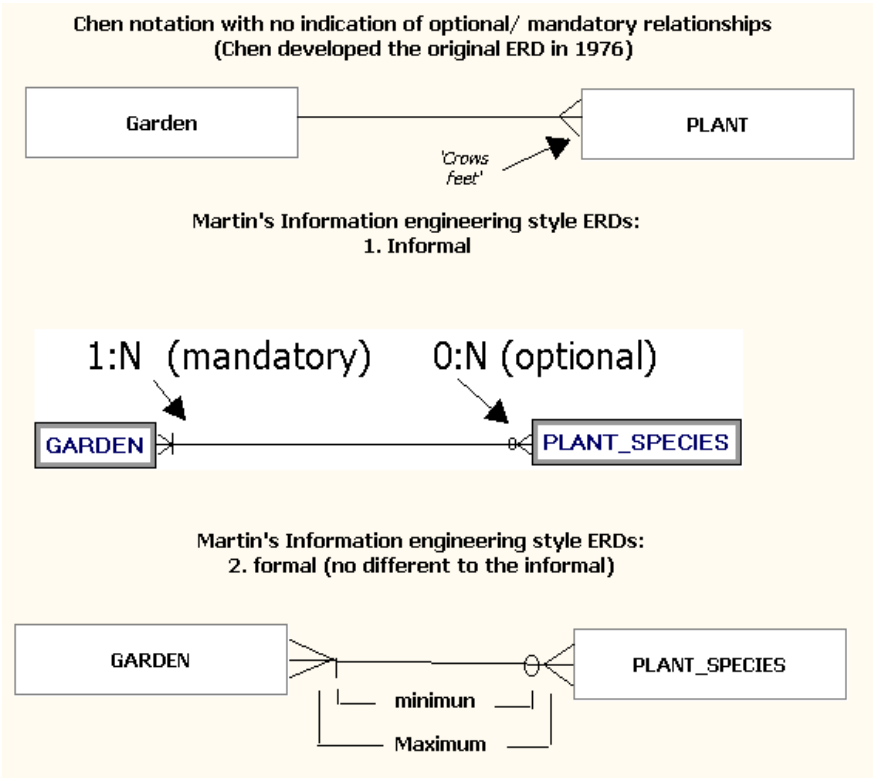
Medical_record			
id	**source**	**date**	**Patient_ID**
9	GP	19/12/1956	005
10	St james hospital	03/05/1966	005
11	GP	11/07/1970	...
12	Istanbul	29/10/1055	...

The parent is the 'one' side of the relationship while the child is represented on the 'many' side. In this instance one parent entity instance can be associated with possibly many child entity instances. The 'many' aspect is shown on an ERD diagram by 'crows feet'. See below for an example.

When you are drawing a one to many relationship you are also explicitly stating a parent child relationship. If you delete a particular parent instance and it has child instances associated with it they will more likely than not be deleted as well. (This is technically called cascade delete). However you can specify a 'restricted deletion'. But then you could end up with child instances that are no longer associated with a parent instance. Lets take our Patent / medical_record example again and this time assume that a PATIENT can now have multiple MEDICAL_RECORD(s). Suppose we would now like our patient Oaks to have two medical records this is easily achieved by linking multiple entity instances in the Medical record entity to the appropriate patient instance. In other words now two entity instances have the same foreign key value.

5.7 Optionality

As with the one to one relationship, it is possible to have either a mandatory one to many ('1:N') or optional one to many ('0:N') relationship. Once again, if at all possible, all optional relationships should be investigated thoroughly to see if they can be converted to mandatory relationships. Reingruber & Gregory 1994 provide details.



5.8 How Many is Many?

This does not matter initially when designing the ERD. But eventually you might want to consider the following two aspects:

- What is the maximum likely number of child entity instances going to be for any parent entity instance?
- What is the average number of child instances for each parent instance (e.g. 1:5 or 1:50 etc)?

Exercise 10. Drawing Entity Types using DeZign - optional

Time: 60 minutes

Work through the **first** DeZign tutorial document. For details of how to obtain this tutorial see the first section of this document - required resources.

Exercise 11. Drawing Relationships

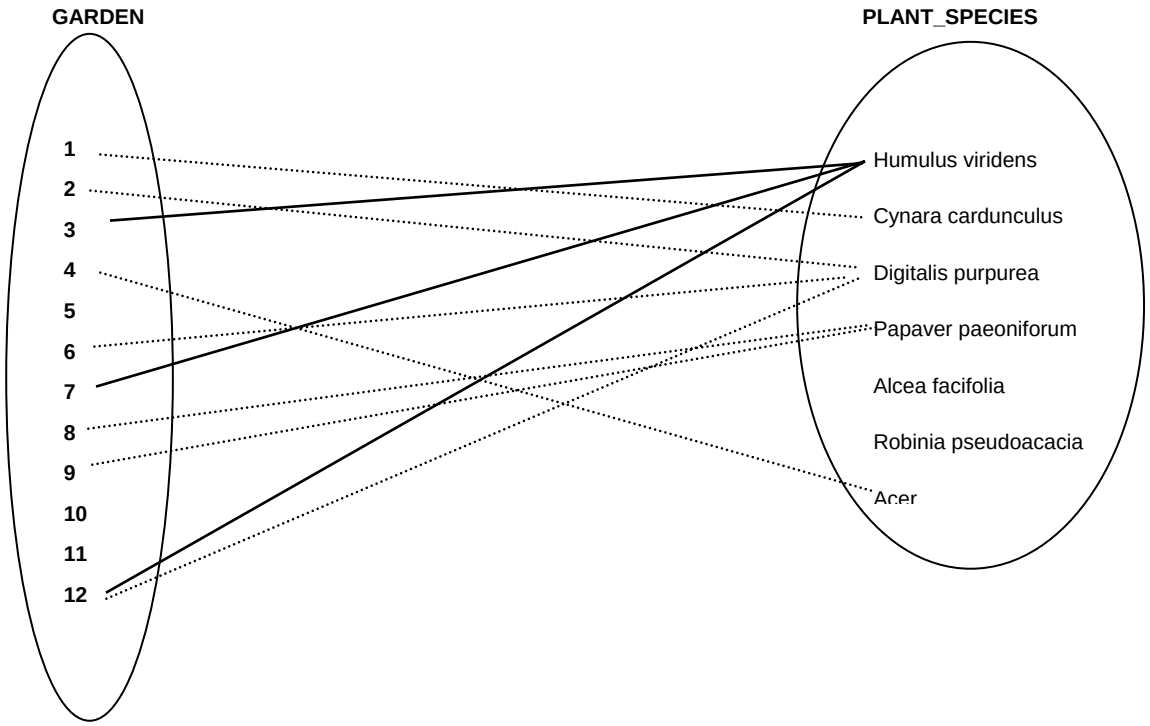
Time: 240 minutes

1. Work through the second tutorial in the DeZign tutorial document.

2. Define and then draw the various relationships (you have identified the entity types in a previous exercise!) that you have identified in the DopeHead scenario. You can do this either in pen and paper or using the DeZign software.

5.9 What is a Many to Many Relationship?

The last page showed a relationship between GARDEN and PLANT_SPECIES as having crows feet at each end of the relationship line. This is a many to many relationship which is often written 'N:M'. The diagram below provides an example:



In this example not only can one PLANT_SPECIES instance be associated with several GARDEN instances but also a GARDEN instance can be associated with several PLANT_SPECIES instances. For example the PLANT_SPECIES Humulus viridens might be in garden numbers 3, 7 and 12 while the gardens also contain other species including Cynara cardunculus, Digitalis purpurea, Papaver paeoniforum, Alcea facifolia, Robinia pseudoacacia and an Acer. The above situation is complex to say the least!

Because of the far reaching consequences of many to many relationships they are often broken down into two, separate one to many relationships. In this example you might end up with GARDEN having many PLANTs, each of which is a type of PLANT SPECIES, which I have renamed NAME.

Now notice that we have two parent entity types, GARDEN and NAME, which can exist independent of each other. In other words we have a set of GARDEN details and also a set of NAME details. This is not the end of the story as we will probably want to be able to model the possibility of a plant having a history of names, as some plants get renamed regularly. **It is always advisable to convert many to many relationships to several one to many relationships after the initial designs.** If you are interested in finding out how many to many relationships are implemented in databases see http://www.robin-beaumont.co.uk/virtualclassroom/chap7/s3/dbcon2.pdf.

Exercise (12.) MCQs

1. Which of the following statements is true (one correct answer)?
    a. A relationship within an ERD is always a link between various entity instances. It is implemented within a database by the use of a foreign key.
    b. A relationship within an ERD is always a link between various entity instances. It is implemented within a database by the use of a primary key.
    c. A relationship within an ERD is always a link between two entity instances within an entity type. It is implemented within a database by the use of a foreign key.
    d. A relationship within an ERD is always a link between two entity instances within an entity type. It is implemented within a database by the use of a primary key.
    e. A relationship within an ERD is always a link between two fields within an entity instance. It is implemented within a database by the use of a primary key.

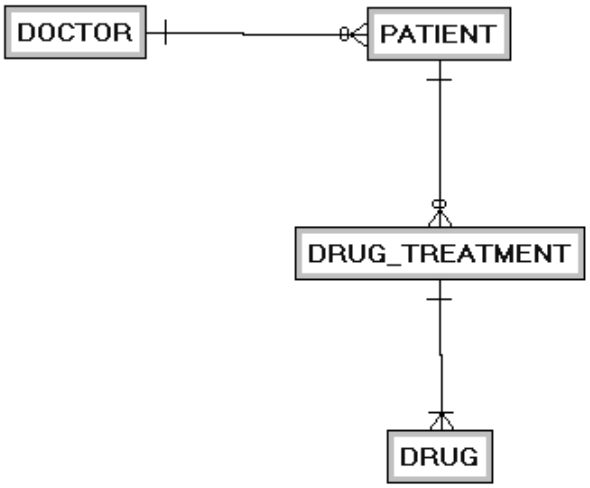
2. Which of the following best describes an optional relationship (one correct answer)?
    a. An optional relationship is one where a foreign key attribute can only take a specific value. For example, patient 001 may have a medical record 003 but no other.
    b. An optional relationship is one where a foreign key attribute can take a null value. For example, a patient may or may not have a medical record.
    c. An optional relationship is one where a foreign key attribute can take any value except null. For example, a patient may have a medical record or a dummy record.
    d. An optional relationship is one where a foreign key attribute may or may not exist. For example, a patient may have a medical record.
    e. An optional relationship is one where a foreign key attribute can take any value regardless of the values in the associated primary key. For example, a patient may have a specific medical record, an undefined one or none at all (depending upon your interpretation).

3. Which of the following statements is correct (one correct answer)?
    a. A mandatory relationship is also called external dependency. A mandatory one to one relationship implies that one instance of entity A is always associated with one instance of B. A mandatory one to many relationship implies that one instance of entity A is always associated with zero or more instances of entity B.
    b. A mandatory relationship is also called existence dependency. A mandatory one to one relationship implies that one instance of entity A is always associated with one instance of B. A mandatory one to many relationship implies that one instance of entity A is always associated with zero or more instances of entity B.
    c. A mandatory relationship is also called existence dependency. A mandatory one to one relationship implies that one instance of entity A is always associated with one instance of B. A mandatory one to many relationship implies that one instance of entity A is always associated with one or more instances of entity B.
    d. A mandatory relationship is also called external dependency. A mandatory one to one relationship implies that one instance of entity A is associated with zero or one instance of B. A mandatory one to many relationship implies that one instance of entity A is always associated with one or more instances of entity B.
    e. A mandatory relationship is also called existence dependency. A mandatory one to one relationship implies that one instance of entity A is associated with zero or one instance of B. A mandatory one to many relationship implies that one instance of entity A is always associated with zero or more instances of entity B.

6. Some Common Mistakes in ERDs

6.1 Confusing Instances and Types

In the example at the beginning of this document we ended up with four entity types: DOCTOR, PATIENT, DRUG_TREATMENT and DRUG. If you carried out the exercises correctly you should now have the ERD shown below.



It may be worthwhile for you to have a look at the original narrative as well; we have certainly cut it down! The important sentence is:

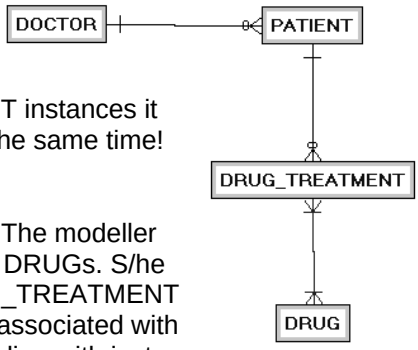
*"Initially the **system** will be concerned solely with **drug treatment**. Each **patient** is required to take a variety of **drugs** a certain number of times per day and for varying lengths of time."*

According to our ERD at left:

- A DRUG_TREATMENT instance is related to many DRUG instances.
- A DRUG instance is related to a single DRUG_TREATMENT instance.

For a variety of reasons this is the best way of modelling the relationship. However, let's consider another way the relationships may have been modelled.

A different modeller argues that while a particular DRUG_TREATMENT has one or more DRUGs a particular DRUG is associated with one or more DRUG_TREATMENTs. While this may seem more logical it should be realised that now if we change the particular value in an entity instance in DRUG we may (depending upon how many DRUG_TREATMENT instances it is related to) be changing several DRUG_TREATMENT entity instances at the same time!

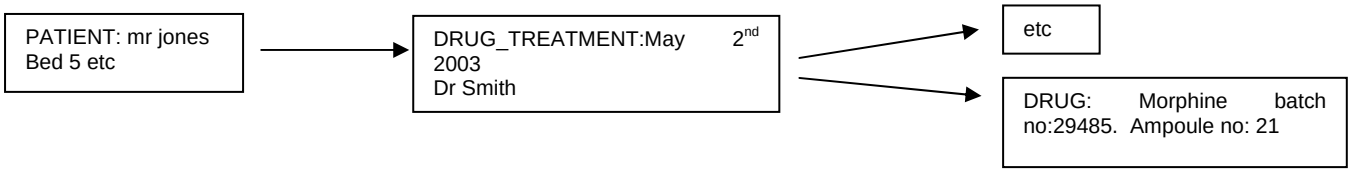


Why has the confusion occurred? The problem is the result of two things:

1. The modeller getting confused between entity types and entity instances. The modeller is correct in thinking that a particular DRUG_TREATMENT has one or more DRUGs. S/he is working at the instance level (good) and thinking about a particular DRUG_TREATMENT for a particular patient. However, s/he then states that a particular DRUG is associated with one or more DRUG_TREATMENTs. Here s/he has forgotten that s/he is dealing with instances. While we are talking about a particular patient's particular drug treatment, s/he has now moved to the level of thinking about all patients' relationships to drugs. In this case the following is correct, A PATIENT takes many DRUG(s) and a DRUG is associated with many patients thus:

Where we are thinking of DRUG as just a name of the drug, for example Mr Jones and Miss Smith take Penicillin. This is not the situation in the scenario where the DRUG is associated with a particular PATIENT and DRUG_TREATMENT DRUG here refers to a specific dose given to a PATIENT. It makes no sense for more than one PATIENT to share a single dose of something!

One method of avoiding this confusion is to create fake instances with attributes and values. For example:



2. Another reason why the modeller may have become confused is concerned with the concept of the lookup table, which we will discuss next.

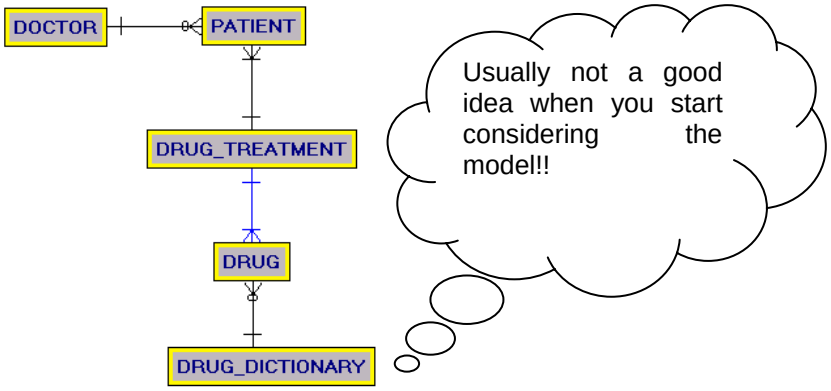
6.2 Including Lookup/Descriptor Entities

While the topic of lookup tables is very much concerned with the low level database design, the concept seems often to impinge upon people's decision making when they are just beginning to draw ERDs. This is probably because we are all familiar with the concept and therefore feel that it must be represented somewhere in the ERD.

A lookup table is something used by the users of databases to provide a pick list of possible entries for a particular attribute. It might be something like a disease code or name of a local doctor.

Consider the ERD on the previous page PATIENT -> DRUG_TREATMENT -> DRUG. Let's assume a particular doctor wishes to have the ability to select a drug from a pre-defined drug list, which seems entirely reasonable.

We could add another entity type to the ERD, DRUG_DICTIONARY, which would provide certain values to any particular DRUG entity instance. In other words certain attribute values for any DRUG entity instance would be obtained by looking up and copying the values from the DRUG_DICTIONARY entity. You might think that this is just another one to many relationship, but look below and think about it:



In this situation, however, we do not want real referential integrity as we don't want the deletion of a particular DRUG_DICTIONARY instance at any time to delete any of the DRUG entity instances that may possess the value copied from the DRUG_DICTIONARY.

This process of copying values from another table but not having referential integrity constraints is usually how a **lookup** table works. Such details as lookup tables are not usually shown in ERDs, at least not the first few you develop. Thinking about it this makes sense because just looking at the above ERD there are several other places we could have lookup tables: one to show the doctors, one for the drug treatments and even one for selecting patients. The ERD would start to look very cluttered.

Another reason why it makes sense not to show the lookup tables is because frequently these tables are simply queries on the attribute you are currently working with. For example you may want to add a new drug attribute value for a new instance of a DRUG based upon all the values that exist in the instances that already exist. You may remember that when we started to consider which where appropriate entity types, one of the exclusion criteria where those possible entity types we came up with that were simply the result of processing information from other entity types.

Just one final warning: although I have indicated that it is not a good idea to show lookup tables in an ERD, there may be occasion to show the odd one. For example, in the above the DRUG_DICTIONARY might be something important in your model such as the British National Formulary or something similar. You would probably then include it and add some text in the ERD providing additional information.

You should begin to realise now that there is not one correct or perfect ERD. Much depends upon the unique situation in which each is developed.

Lookup tables are also called descriptor tables by modellers who specialise in developing a special kind of database called a data warehouse. (Nelson G S 2001 Implementing a Dimensional Data Warehouse with the SAS System SUGI26 proceedings. See: http://www2.sas.com/proceedings/sugi22/DATAWARE/PAPER129.PDF)

Hernandez 1997 p344-349 Calls them Validation tables and recommends that such entity types be distinguished in the ERD from other entity types by having a distinctive marking.

Some software tools such as System Architect (SA) differentiate between the display of identifying and non-identifying relation lines by drawing an identifying relation line as a solid line and a non-identifying relation line as a dash line. An *identifying relation* indicates that the parent entity identifies the child entity. A *non-identifying relation* indicates that the child entity does not rely on the parent entity for its identification in this instance the parent can be specified as mandatory or optional when the relation is non-identifying. (Much of the information from this last paragraph has been taken from the excellent SA help file).

6.3 Unnecessary Complexity

An historical aside – Occam's Razor

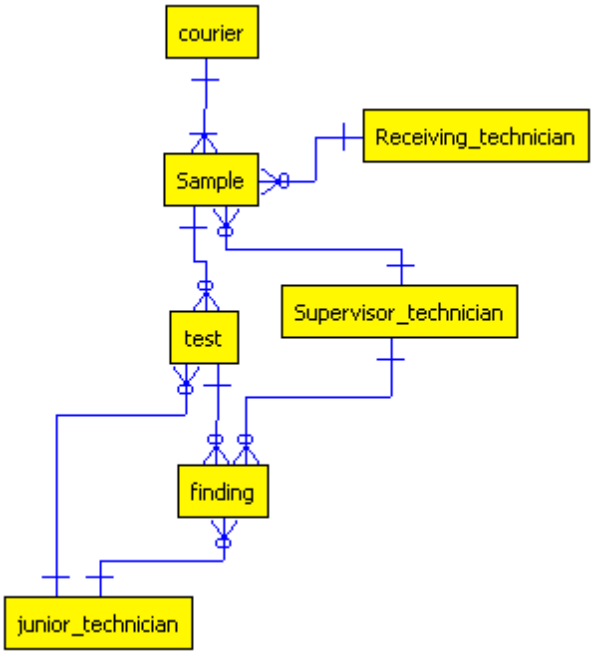
Entia non sunt multiplicanda præter necessitatem – entities ought not to be multiplied except from necessity

Which means that all unnecessary facts or constituents in the subject being analysed are to be eliminated. Occam dissected every question as with a razor. William of Occam, the Doctor *Singularis et Invincibilis* (d. 1349), the great Franciscan scholastic philosopher, was probably born at Ockham, Surrey, UK. Occam being the latinized form of the name.

From Ebenezzer Cobham Brewers Dictionary of Phase & Fable (Wordsworth edition revised by Ivor Evans 1993)

One of the main problems with ERDs is the desire to introduce a multitude of relationships and entities when thoughtfulness (along with a large amount of scrap paper!) can usually produce a model with much more clarity. Here are some examples.

Unnecessary entities and relationships

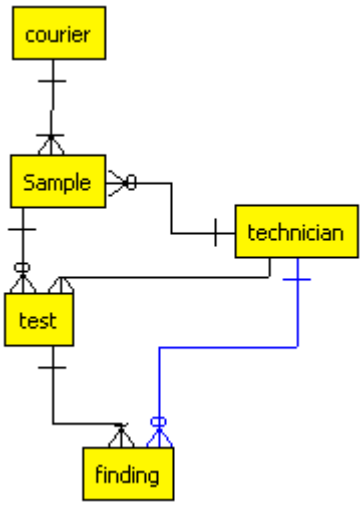


Consider the following:

- A COURIER delivers one or more SAMPLEs.
- Each sample can have zero or more TESTs carried out on it.
- Each TEST produces zero or more FINDINGs.
- Each sample is signed for by the RECEIVING_TECHNICIAN.
- The sample is then managed by a SUPERVISOR_TECHNICIAN who also confirms any FINDINGs.
- The TESTs are carried out by JUNIOR_TECHNICIANs.

From the above information one possible ERD is shown opposite. This all looks very complex. The entity types COURIER, SAMPLE, TEST and FINDING seem sufficiently different. However, how different are the RECEIVING_TECHNICIAN, SUPERVISING_TECHNICIAN and JUNIOR_TECHNICIAN? I even wonder that on a Sunday afternoon they could in real life be the same person.

Obviously the modeller has tried to present the information that is collected at each stage concerning the TECHNICIAN input, but I wonder if the following might be just as suitable.



To indicate that a particular grade of technician should only carry out certain tasks a small amount of text could be added to the ERD and it is important to realise that a ERD can not express all the complexities you may wish to model.

Looking again at the ERD there still appears to be a large number of relationships from the TECHNICIAN entity to various others. Do we need all these? To help answer this question I would like to digress slightly and consider two different aspects:

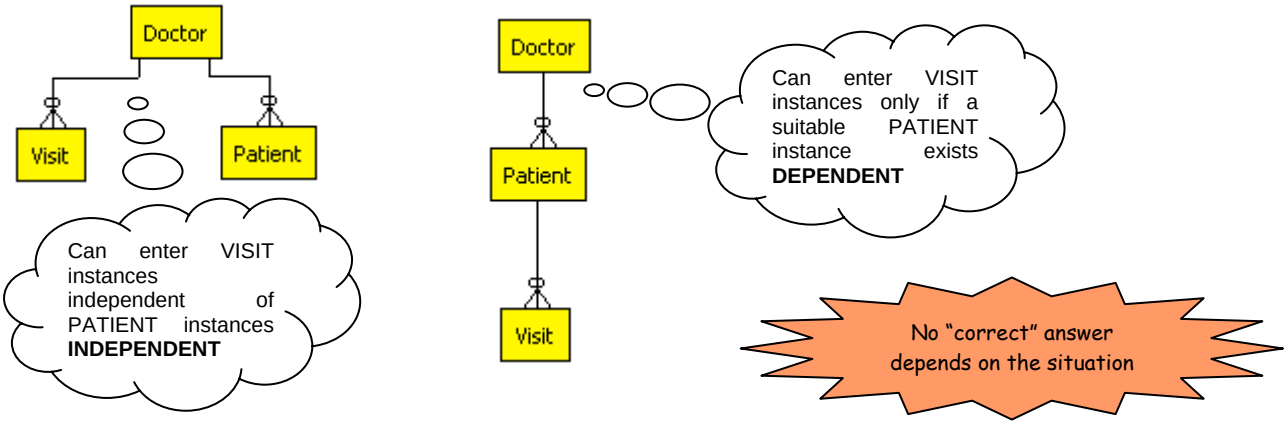
- **Independent versus sequential relationships**
- **Multiple level relationships**

Avoid loops of relationships - one may be redundant

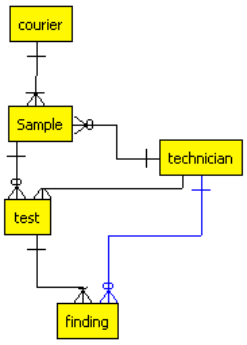
Carter 1995 p.49 recommends that if there are a number of relationships which form a loop in your ERD if possible the loop should be broken. For example in the SAMPLE -> TECHNICIAN -> TEST situation in the ERD above you should consider very carefully if you need all three relationships. In the above ERD you do but there are situations where you do not. For example think about the three entity types CUSTOMER, INVOICE and PAYMENT and assume we have relationships between all three. However in this situation the relationship between Customer and PAYMENT is no really necessary as any payment information can be obtained by going through the Invoice which obviously will exist before a payment is made.

Independent versus sequential relationships

Consider the following two ERDs:



Multiple level relationships

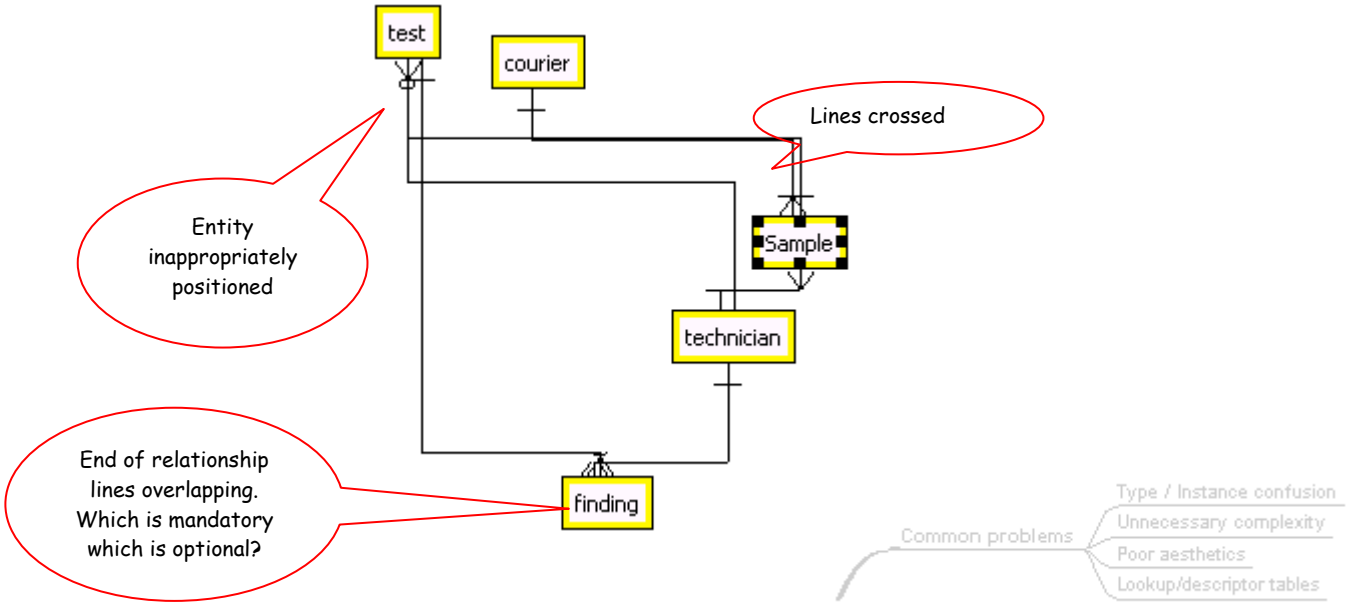


This is the situation in the COURIER-> SAMPLE ->TEST-> FINDING ERD where the TECHNICIAN entity has a relationship at three levels. Considering the information above about dependency, you should begin to realise that if we had a mandatory relationship (1:n) rather than an optional one (0:n) the situation would be made easier. Why? Because with a mandatory relationship we know that a parent instance would always possess at least one child instance. Let's presume the ERD opposite was changed so that the relationships between SAMPLE ->TEST and TEST and FINDING were mandatory. If this were the case we would know that every SAMPLE would be related to at least one TEST and FINDING instance. Given this information we might then revise the ERD to have only one relationship from TECHNICIAN which would then probably be to FINDING.

While the above is one possible method of making the ERD much clearer, it might not be appropriate to the situation. The question is: what is the real situation and once again what is one trying to achieve?

6.4 Poor Aesthetics

The diagram below shows a poor version of the above ERD. It is unsatisfactory because the entities and relationships are poorly positioned.



7. The Relationship between Narrative Descriptions and ERDs

Throughout this document I have used narrative descriptions as the basis for designing ERDs. This is usually the case. Importantly the resulting ERDs, after discussion with those who wrote the narratives, are clearer than the original textual documents, at least I always find them to be so. This is probably because I'm very much a graphical rather than a textual person. I always find myself drawing family trees when reading a Russian novel to keep track of the characters!

We spent some time at the beginning of this document identifying entity types and relationships in narratives. A brief summary (partially derived from the DeZign help file) of the information presented on those is given below.

One-to-One (1:1) Relationship

This is when at most one instance of an entity A is associated with one instance of entity B. For example: "Employees in the company **are each** assigned their own office." For each employee, there exists a unique office, and for each office there exists a unique employee.

One-to-Many (1:N) Relationship

In this type of relationship, for one instance of entity A there are zero, one or many instances of entity B, but for one instance of entity B, there is only one instance of entity A. For example: "A department **has many** employees but each employee **is only** assigned to one department."

Many-to-Many (M:N) Relationship

This relationship, sometimes called non-specific, is when for one instance of entity A there are zero, one or many instances of entity B, and for one instance of entity B there are zero, one or many instances of entity A. For example: "Employees can be assigned to no more than two projects at the same time; projects must have assigned at least three employees." A single employee can be assigned to **many** projects; conversely, a single project can have assigned to it **many** employees.

Optionality

The existence of an entity in a relationship is defined as either mandatory or optional. If an instance of an entity must always occur for an entity to be included in a relationship, then it is mandatory. An example of mandatory existence is the statement "Every project **must** be managed by a single department."

Mandatory = must

If the instance of the entity is not required, it is optional. An example of optional existence is the statement "Employees **may be** assigned to work on projects."

Optional = maybe

The following exercise gives you some practice at working between narrative descriptions and ERDs.

Exercise 13. Developing ERDs from Narratives

Time: 60 minutes

These excellent examples have been taken from David McLean's (D.McLean@doc.mmu.ac.uk) home pages. He is a senior lecturer in computing at Manchester Metropolitan University (http://www.doc.mmu.ac.uk/online/SAD/T06/erdexes.htm no longer active June 2006).

A) Produce an ERD for the following scenario:

This scenario describes the activities of the Science and Technology faculty in a university in northwest England. Students take courses in the university. The courses are run by departments. It is possible for a course to be run by more than one department. Each course consists of a number of units. Lecturers teach the units, and in some circumstances more than one lecturer will teach a unit. Lecturers of course teach on more than one unit. Since the introduction of 'modular' courses, it is possible for one unit to be included on more than one course. A lecturer may be a course leader for a particular course.

B) Produce an ERD for the following scenario:

A farmer wishes to keep computerised records on the milk and calf production of the dairy herd. All calves produced are sold and not added to the dairy herd. Each cow has a name and date of birth, and will produce milk for a lactation period after the birth of a calf or calves. Milk recordings for each cow in terms of litres are taken each day. The information required for each pregnancy of a cow are the bull's name, date of mating, date of birth of calf or calves and each calf's sex and birth weight. The system is to provide the following information to the farmer:

- Details of all births of calves attributed to each bull
- Milk yield of a cow over a particular pregnancy (note: a lactation period is associated with each pregnancy)

C) Produce an ERD for the following scenario:

Olde Worlde Homes (OWH) is a regional building company employing both permanent and contract staff. The main activity of the company is building houses that recapture the feeling of bygone days on land it acquires cheaply. All matters of policy (eg what land should be purchased) are determined by management. Once the land has been purchased, the land is known as a 'site' and details of it such as area, cost, vendor etc are kept so that management decisions can be assessed.

Several house are built on the site; the number and style are determined by the location, the land size, target market and the physical assets of the land (eg trees or ponds). It follows from this that all houses are built to an individual design ('architect designed' according to the sales literature). Information on each house such as number of bedrooms, style, sale price etc is recorded, again so that management can review their decision making and also monitor costs.

In order to maintain quality, only building materials which have been approved by the quality assurance manager can be used to build the houses. Records are kept of the approved materials. All the materials which are used to build a particular house are itemised and then purchased from the cheapest supplier. The items (bricks, cement, kitchen units etc) are noted on a list which also contains details of the supplier and cost. A list of suppliers containing nature of business, telephone number and address is maintained. Once the houses have been built, they are marketed directly by OWH.

Houses are sold directly to the purchaser (or in some cases joint purchasers) without involving estate agents. Details of purchasers are kept on file.

D) The following is a brief description of a library.

Borrowers take books out on loan. A borrower may take out up to five books at any one time. Several copies of the same book are held for books which are continually in demand. Borrowers may make reservations for titles which are out on loan.

Items which need to be stored include borrower number, borrower name, date borrowed, ISBN, acquisition number (allocated when a book is purchased), date acquired, title, author, borrower making reservation, and date reserved.

The library needs to be able to do the following:

1. check whether a book returned has been reserved
2. check which borrower has a particular book out on loan
3. check which books a borrower has out on loan
4. check whether a copy of a particular title is in the library

Draw an ERD to represent the library. Resolve the many-to-many relationships and suggest identifiers and the main attributes for each entity.

State any assumptions you make.

Do not turn the page until you have tried the above exercises!!

Discussion of Above Exercises

David McLean thoughtfully does not only provide solutions to the above exercises but also information as to how he came to the solutions. I have taken the liberty of expanding his comments, as clearly his material would be supplemented by face to face explanations.

A) University course scenario

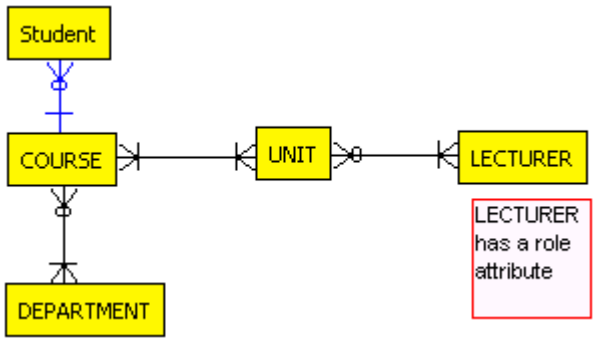
1. Students take courses ~~in the university~~. (see text)
2. The courses are run by departments.
3. It is possible for a course to be run by more than one department.
4. Each course consists of a number of units.
5. Lecturers teach the units,
6. in some circumstances more than one lecturer will teach a unit.
7. Lecturers teach on more than one unit.
8. Since the introduction of 'modular' courses, it is possible for one unit to be included on more than one course.
9. A lecturer may be a course leader for a particular course.

The main facts abstracted from the scenario are given opposite.

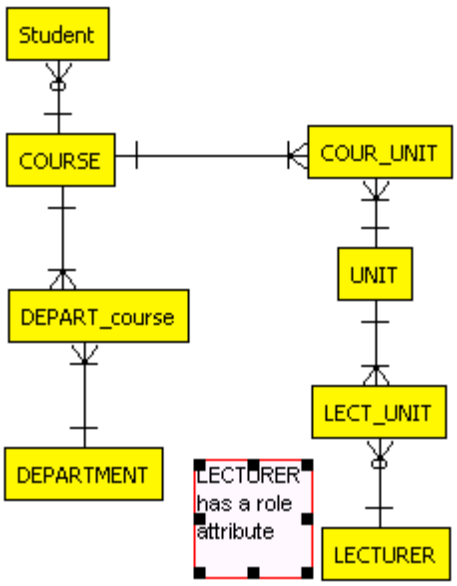
Firstly notice that the list does not include “the Science and Technology faculty in a university in northwest England” (ie the University). This is because there is no need to include an entity type that only has one instance and includes all the others.

Considering the facts, we come out with the ERD below; this is only one of several possible solutions based upon how you interpreted the narrative. You may have even added some extra ‘domain specific’ knowledge such as a relationship between DEPARTMENT and LECTURER.:

You will notice that there are several many to many relationships.



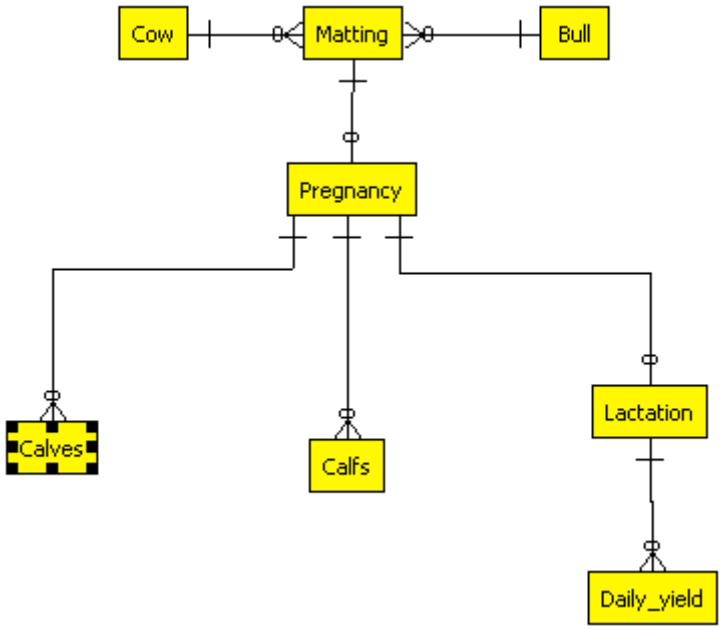
There are several many to many relationships in the ERD which we could develop by adding additional entities:



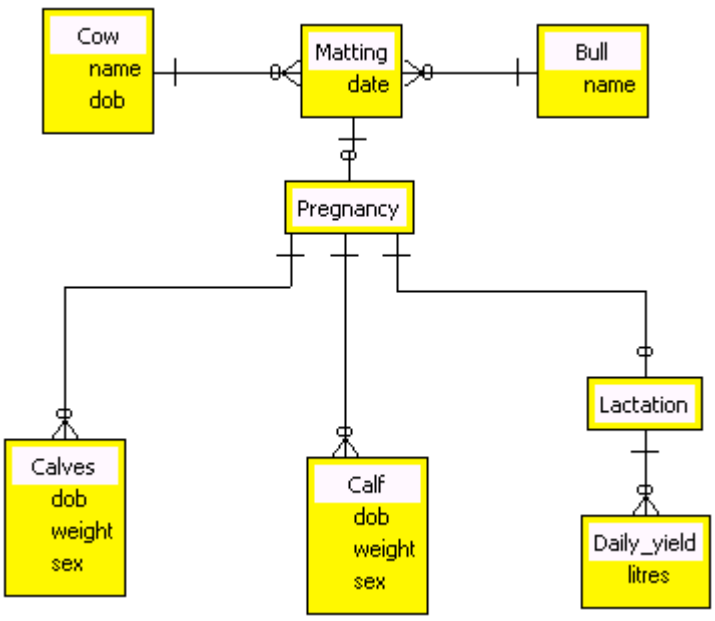
B) Dairy farmer scenario

1. All calves produced are sold and not added to the dairy herd.
2. Each cow has a name and date of birth, and will produce milk for a lactation period after the birth of a calf or calves.
3. Milk recordings for each cow in terms of litres are taken each day.
4. The information required for each pregnancy of a cow are the bull's name, date of mating, date of birth of calf or calves and each calf's sex and birth weight.
5. The system is to provide the following information to the farmer:
   - Details of all births of calves attributed to each bull
   - Milk yield of a cow over a particular pregnancy (note: a lactation period is associated with each pregnancy).

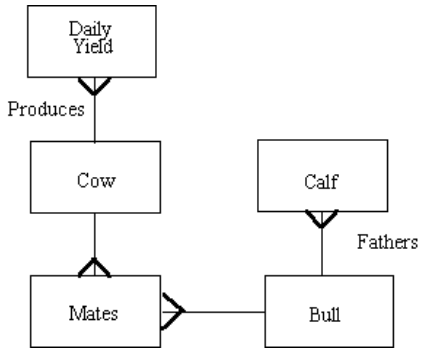
Initially we have a many to many relationship between Cow and Bull. This is easily resolved to Cow has zero or more MATING(s) and Bull has zero or more MATING(s). All the other relationships appear to be of the one to many variety; each MATING can result in a pregnancy which will result in zero or more CALVES and CALFS. There will also be a LACTATION period.



The diagram below shows the attributes that are described in the scenario:

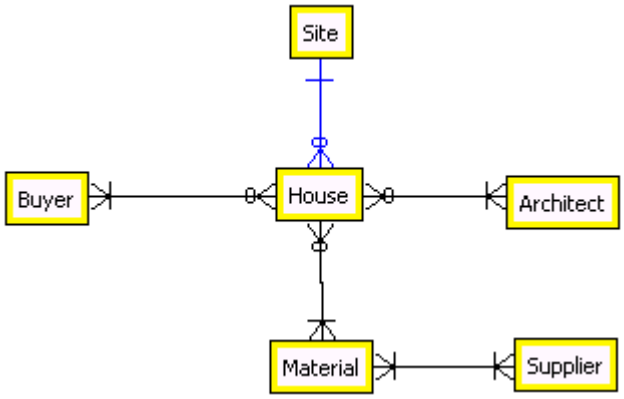


David McLean provides the following possible solution which is simpler than mine and probably does the job as well. Quoting David, “by reading the question carefully, the key features are calves, bulls, milk yield and cows”:



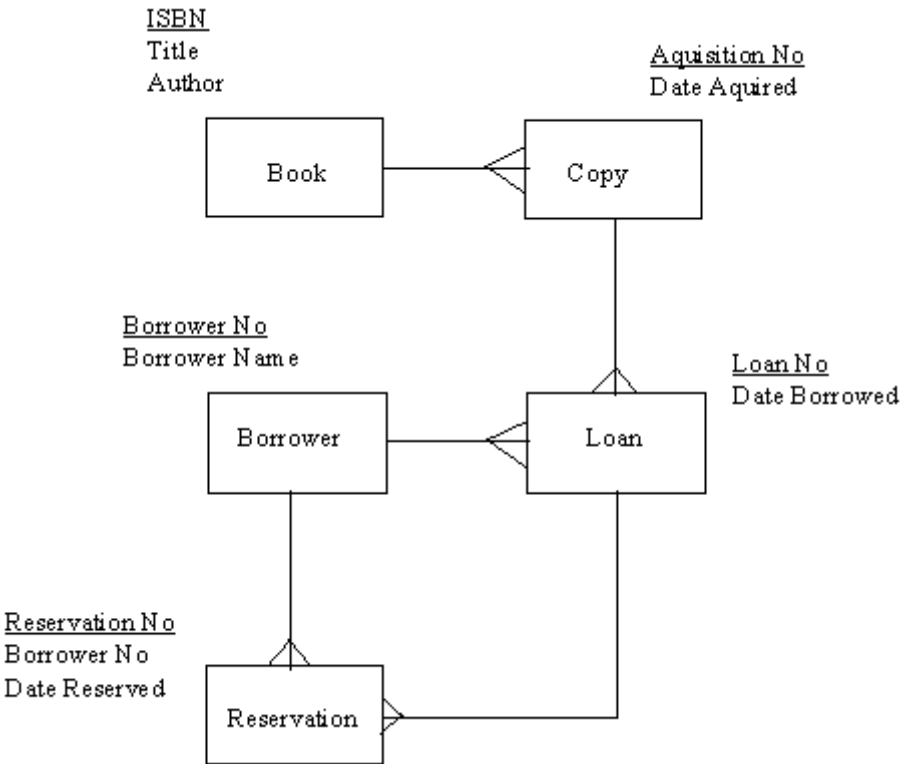
C) Olde Worlde Homes (OWH) scenario

I have just reproduced David's solution below. In the final version it would have been appropriate to convert the four many to many relationships to several one to many relationships by adding intermediate entity types such as HOUSE_ARCHITECT between HOUSE and ARCHITECT etc.



D) Library scenario

Once again I have just reproduced David's solution below.



You now have a good grasp of the basics of ERDs. But before we finish all the practical aspects of ERDs I would like to introduce the idea of relationships that occur within a single entity type.

8. Recursion in ERDs

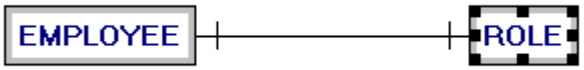
Recursive relationships are also called **unary** or **involuted** relationships.

So far we have discussed examples of relationships between entity types; however, it is possible to have links within an entity. Such a relationship is called recursive.

The usual example given is that of an EMPLOYEE entity type that has a relationship with itself called supervisor. In other words an EMPLOYEE entity instance can relate to another EMPLOYEE entity instance in a supervisor role.

Clearly a 'one to many' or 'many to many' recursive relationship may also exist. (see Carter 1995 p61 - 68, Elmasri & Navathe 1989 p. 49) Most writers suggest that these recursive relationships are acceptable in early models but should be converted to 'one to many' or 'one to one' relationships, as the model is refined. For example the above recursive relationship could be modelled by introducing a new entity type called ROLE.

Finkelstein (1989), amongst others, has recommended a process to remove recursive relationships basically by creating a 'role' entity type as in the above example. Furthermore he specified that models where such relationships had been removed were defined as being in **fifth business normal form**:



"An entity is in fifth business normal form if its dependencies on occurrences of the same entity or entity type have been moved into a STRUCTURE entity"
(Finkelstein 1989 quoted in Reingruber & Gregory 1994 p207)

Very briefly, when designing a database one aims for a highly normalised (basically structured) design but how you achieve this is a matter of dispute. Each normal form represents a greater degree of data structure. This usually involves breaking entity types up from those that have a large number of attributes to several with a smaller number of attributes.

I have not discussed the process of normalisation in this document as there are two conflicting schools of thought about how you go about developing these eventually highly normalised ERDs. One school of thought recommends the method I have described in this document, you take a narrative and abstract the nouns etc and constantly refine your model. A conflicting more formal approach suggests that you subject an initial model to a formal process of normalisation to obtain the highly normalised ERD. If you wish to find out more about normalisation see http://www.robin-beaumont.co.uk/virtualclassroom/chap7/s7/index.htm

Whichever method you use it should now be clear to you that developing ERDs is an iterative process, starting with very rough models and gradually refining them. The topic of the next section firstly considers how this process has been refined into a number of stages and then secondly where ERDs fit in the whole database development process, but first a quick exercise.

Exercise 14. MCQ

Which of the following are correct statements about recursion (choose two)?

a. Recursion is a valuable modelling technique in detailed data models.

b. Recursion is added when making a data model comply to fifth business normal form.

c. Recursion is added at the end of the modelling process.

d. Recursion is modelled by introducing 'structure' entity types in the latter data models.

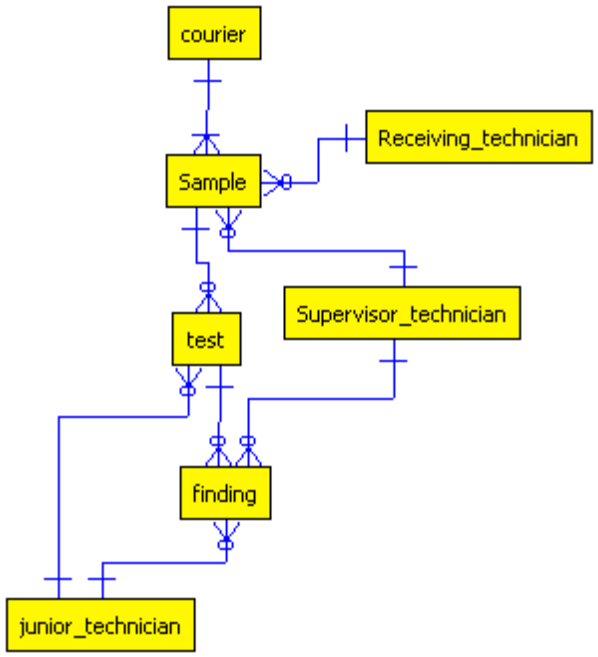
e. Recursion is something that is allowed at the beginning of the modelling process.

Exercise 15. Recursion

Time: 10 minutes

Consider this ERD. Think where recursion is implied and suggest a solution.

.



9. Conceptual, Logical and Physical Data Models

Much has been written about 'conceptual', 'logical' and 'physical' data models. Unfortunately most of it is utter dribble! The problem is often made even worse because of this unsatisfactory situation, by well meaning writers who continually develop other categories of models such as Cook & Daniels' (1994 p10) essential, specification and implementation models.

The division of logical and conceptual data models was primarily done to help those developing databases.

Many years ago, in 1978 to be exact, the ANSI/X3/SPARC committee produced a standards document specifying what a database management system (DBMS) should look like. Within it they specified, very sensibly, that there should be three levels:

- **External ('User') views** (ie what the doctor sees of the database is a different view from that of a secretary)
- **Conceptual** schema (the ERD, etc)
- **Internal schema** (the actual 0 and 1 digits which comprise files in the computer)

The idea was that most people worked at the top two levels, which is now taken for granted; nowadays you write a database using a DBMS. No one in his or her right mind would write a database using basic C++ or any other programming language.

However, it soon became clear that within the conceptual level there was a great deal of variation. This level then became (by various writers and database software companies) divided into **conceptual, logical** and **physica**l levels. Unfortunately there is no standard definition or consensus as to what this actually means. Take for example the following situations:

- One person creates an ERD and provides a list of attributes for each entity type. While the foreign key attributes are not defined yet they are clearly marked by the relationship lines on the ERD. (You can usually work them out from the diagram.) In contrast, another person adds all the foreign key attributes in the narrative description of the data model.
- One person creates an ERD and provides lists of attributes, which specify detail data types (integer, string etc) while another does not.
- One person creates an ERD with 'many to many' and recursive relationships in it. Another 'corrects' all these types of relationships to 'one to many' or 'one to one' types.
- One person ignores the various lookup tables in the ERD while another adds the majority of them in.

If you took a straw poll of database developers they would say that a conceptual model does not have foreign keys explicitly defined as attributes, whereas for the second and third situations given above it would be much more difficult to gain any consensus about what level the model was at. My advice is not to worry. You always start with a conceptual model and end up with a physical one.

The important thing to realise is that the conceptual model (the ERD) will need tweaking into a physical model before it can be translated into a particular DBMS. All these models sit within the conceptual level of the ANSI/SPARC framework.

Obviously, if you have created an ERD within a DBMS such as Access, it must be a physical model.

In summary:

- If the ERD has foreign key attributes defined, it is probably at the physical level.
- If the ERD does not have complete attribute definitions, it is not at the physical level.
- If the ERD has been created within a DBMS, it usually must be at the physical level.
- Most models do not include foreign key attributes

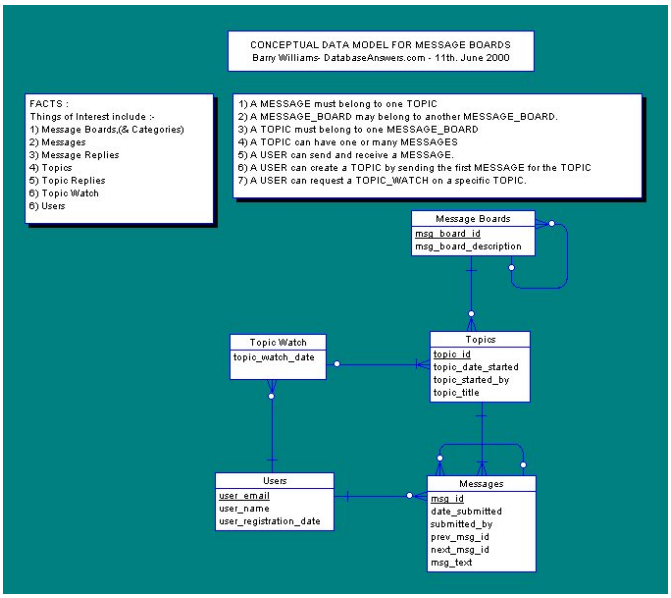
My advice to you is to be like Reingruber & Gregory, and the vast majority of actual database designers, by totally ignoring the arbitrary divisions. Just say you start off with a high level model and gradually add detail until it is possible to map the model directly onto the DBMS of your choice.

The problems described above are related to the history of the entity concept. For details see http://www.aisintl.com/case/method.html.

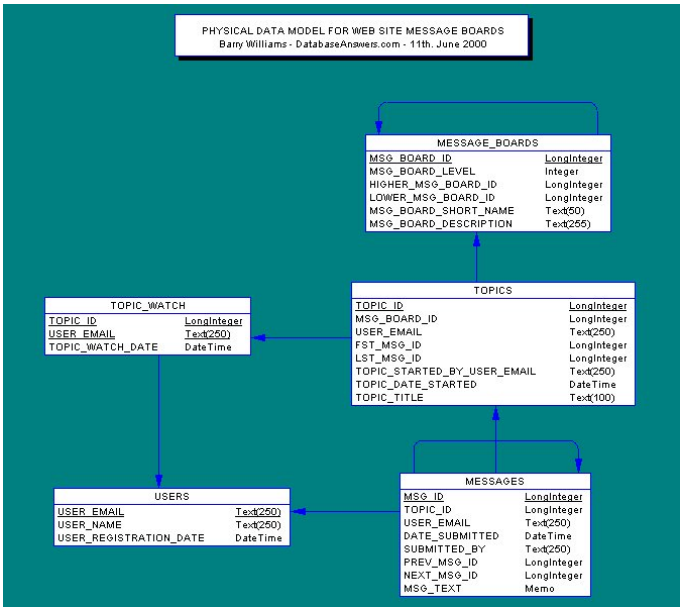
9.1 Electronic Message Board Example

Barry Williams has been generous enough to put a collection of data models on the web at http://www.databaseanswers.com/data_models/. The following is one of them and considers the development of an electronic message board. The author provides examples of what he considers to be the 'conceptual' and 'physical' data models as well as Access code to create the database.

Conceptual model



Logical model



In this instance both models have the same number of entity types; however, this is not always the case as you will have already realised with the situation concerning lookup tables. Notice also the addition of the foreign keys in the physical model. You can also download the SQL script to make this into an Access database at http://www.databaseanswers.com/data_models/msg_board_physical.htm.

Exercise 16. MCQ

Which of the following statements best describes conceptual, logical and physical data models?

a. A loosely defined set of terms, often used inappropriately, indicating the gradual progression from a high level data model to one which provides all the detail required to implement it in a specific database system.
b. A clearly defined set of terms, often used inappropriately, indicating the gradual progression from a data model containing attribute descriptions to one which provides all the detail required to implement it in a specific database system.
c. A loosely defined set of terms, often used inappropriately, indicating the gradual progression from a data model containing attribute descriptions to one which provides all the detail required to implement it in a specific database system.
d. A clearly defined set of terms, often used inappropriately, indicating the gradual progression from a high level data model to one which provides all the detail required to implement it in a Microsoft compliant database system.
e. A clearly defined set of terms, often used inappropriately, indicating the gradual progression from a high level data model to one which provides all the detail required to implement it in a referential database system.

10. Where do ERDs fit Into the Database Design Process?

At the beginning of this document we mentioned that ERDs play a central role in the design of databases. You should now see why this is the case; they define the actual database's structure. If this structure is wrong, no amount of tinkering around at the edges will solve the problem.

Obviously the way the database looks to the user (the user interface or UI) is also very important, and it is a very difficult balancing act to calculate the relative amount of attention that should be given to UI and data structure aspects in the real world with limited resources.

I would argue that the development of a good quality ERD (or a number of them for large databases) and associated data dictionary is a fundamental prerequisite for success.

Where exactly ERD modelling fits into the database design process depends upon the size of the project. We will consider two scenarios: the small personal database project and the large departmental database.

10.1 The Small Personal Database

This might be a small database for just you or perhaps for a research assistant as well. The method described below would not be suitable if you were planning on giving it to a team of people to use or if it were a 'mission critical application' as the IT people would say. To the right is the development method suggested by John Carter, a well respected database educator in the UK, in his book *Database Design and Programming with Access, SQL and Visual Basic*, McGraw Hill 2000 (p2).

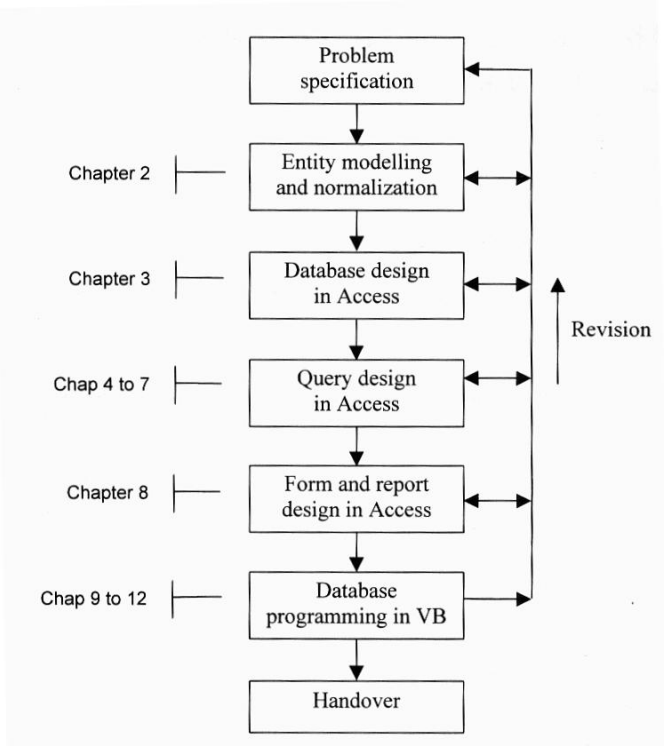


Fig. 1.1 The database system development life cycle of a typical database application.

10.2 Departmental and Hospital Systems

In this situation you are likely to go through a formal process in which you initiate either a formal procurement process for an existing system which might be adapted to your needs or the wholesale development of a new system. In either case a much more rigorous process will be followed than that shown above. There are numerous database method(ologie)s available for the management, procurement, specification and implementation of large systems. Most large computer systems companies have their own methods (e.g. Oracle, ICL, IBM etc) which are frequently based upon fairly standard techniques such as IDEF1X, SSADM or Star. All these techniques rely to a great extent upon ERD modelling; although the lines on the diagram might be a different shape for each, there is really very little difference between them.

In the UK, SSADM and "Structured Systems Analysis and Design" are registered trademarks of the Government Centre for Information Systems (CCTA) in the UK. The CCTA is responsible for the development of SSADM. IDEFiX is the USA equivalent.

Obviously these various method(ologies) are based upon a set of implicit assumptions which 'colour' their approach significantly. I will not discuss this here; the document concerned with the theories (http://www.robin-beaumont.co.uk/virtualclassroom/chap11/s4/sa1.pdf) behind systems methods will revisit this very important topic in more depth.

Exercise 17. Database Methodologies

Time: 30 minutes

1. Please visit some of the sites below. I do not want you to get any detailed knowledge but rather to become aware of such methods and how they differ from that described by Carter above.

IDEF1X (USA) National Institute of Standards and Technology (NIST): http://www.idef.com/idef1x.html

General list of different approaches:
http://en.wikipedia.org/wiki/List_of_software_development_philosophies

SSADM academic notes: http://www.comp.glam.ac.uk/pages/staff/tdhutchings/chapter4.html

2. Add a few of your own links below or if you are working through this document as part of a web based course why not share any good or bad links with your colleagues via the discussion board.

Exercise 18. MCQ

ERD modelling is which of the following (choose two answers)?

a. A highly objective method of obtaining data requirements

b. A technique developed over the last two decades

c. A process that requires knowledge of the context of the model to be developed

d. A very subjective method that produces highly personalised models

e. A technique only used in small-scale database development

11. Publicly Available Data Models (ERDs)

There are several publicly available data models. A few are given below.

You can see a health centre data model, along with numerous others at:
http://www.databaseanswers.com/data_models/health_centre/index.htm

The NHS Data Dictionary & Manual (July 2006): http://www.connectingforhealth.nhs.uk/datastandards/

The NHS Healthcare model: http://www.virtualtravelog.net/projects/Ontology/CBS/hier.htm

An excellent article describing why high level data modelling approaches sometimes fail (specifically the above models):
http://www.virtualtravelog.net/entries/2004/01/ontology_review_1_the_nhs_common_basic_specification_why_top_level_ontologies_dont_work.html

UNITED KINGDOM: Max Jones' PhD thesis on 'Formal generic Modelling' (no longer active July 2006)

EUROPE: The standards work of CEN/TC251

AUSTRALIA: http://www.aihw.gov.au/publications/index.cfm/title/9813

CANADA: http://secure.cihi.ca/cihiweb/splash.html

UNITED STATES:
CORBAMed, http://www.hipaanet.com/upin7-10.htm the healthcare domain taskforce of the Object Management Group

HL7

The US Military Medical Technology site http://www.military-medical-technology.com/

One of my Msc students (Luisa Whitelaw) kindly suggested the following additional links on the module discussion board:

http://support.esri.com/index.cfm?fa=downloads.dataModels.gateway

http://www.who.int/healthmetrics/library/south_africa_05apr.doc

http://www.nisc.co.za/databases

http://www.tama-sa.gov.za/terminology_sa_home.htm

http://www.doh.gov.za/nhis/docs/nchmis.htm

http://folk.uio.no/leopoldo/Publications/Papers/HCIS-1465JN_Finland.pdf

http://www.sita.co.za/documents/invitations/504_2006/RFB%20504EHR%20Strategy.doc

http://groups.csail.mit.edu/medg/courses/6872/2003/slides/lecture2-print.pdf

http://www.avert.org/safricastats.htm

http://www.tac.org.za/aidsstats.html (see links under section "further reading")

http://iri.columbia.edu/outreach/training/course/bamako1999/html/slide/ZAMO.ppt#263,10,Proposed outputs/results

http://www.musc.edu/infoservices/lanvision/Regulations_Policies_Guidelines/ASTMEMRGuide.pdf

Exercise 19. Public Data Models

Time: 30 minutes

1. Please visit some of the sites listed above. I do not want you to get any detailed knowledge but rather to become aware of such resources. Make sure you visit the NHS data dictionary and NHS Healthcare model sites, but don't attempt to understand it all fully!

2. Add a few of your own links below or if you are working through this document as part of a web based course why not share any good or bad links with your colleagues via the discussion board.

12. CASE Tools

You will have completed many of the tasks in this handout by drawing ERDS either with a pen and paper or using a CASE tool, the introduction to this document links you to some CASE tool tutorials.

CASE tools began to be developed in the 1980s, and many thought they would replace programmers. The idea was that you entered all the design details into the tool and it magically produced a finished system. Unfortunately this has never quite materialised. Good programmers are just as sought after now, if not more so, than they were 20 years ago. A new approach called '**Extreme Programming**' has taken on a few of the good aspects of the CASE tool approach, by bringing the programmer and the person(s) requesting the system much closer. You could say that the traditional massive design documents have now been replaced with fluid micro experiences where people work together in a problem solving manner, using a CASE tool as just one method to document their experiences.

12.1 What Advantage(s) do CASE Tools Offer Over Simple Diagramming Tools?

A diagramming tool, such as the picture drawing facilities in Word 2000, allows you to draw a diagram easily enough but that is where it stops. On the other hand a CASE tool is a **repository** (sometimes called an **encyclopaedia**) of information. This repository is formed from the actual diagrams and other data collection facilities contained in the CASE tool. This repository is in the form of a language (often in the form of a database itself) which allows the tool to do, amongst other things, the following:

- **Rule checking** - It checks that you have obeyed various rules while drawing the diagrams.
- **Consistency** - It checks that there is consistency between diagrams (entities with the same names across diagrams have the same relationships and attributes etc).
- **Reporting** - It can create various tabular reports from the information in the repository (eg the automatic creation of a data dictionary from a number of ERDs).
- **Database translation** - It allows the information (ERDs etc) to be converted to a number of database definition languages (eg Access 2000).
- **Reverse engineer** - It translates databases already developed into sets of ERDs and reports etc.

Most CASE tools provide support for several different database design method(ologie)s. Therefore a variety of diagramming styles are provided.

A list is provided below of some database modelling CASE tools. It was taken from http://www.databaseanswers.com/modelling_tools.htm:

PRODUCT	VENDOR	PRICE	COMMENTS
Argo UML now called Poseidon	http://www.gentleware.com.	Free - Open Source	For Modelling UML Diagrams - but not for the faint-hearted. Has a good export facility for diagrams including svg. Requires Java installed on the computer.
Case Studio	CharonWare	Free demo Download Maximum 6 entities), then $149 or $299	Looks impressive - if you try it, please let me have your comments. Here's a "5-Star" Review from Dan Horn on 27th. January 2002.
CASEWise	CASEWise		Aimed at Business Process Modelling, but has links to DataArchitect.
DataArchitect	Sybase	$2,000	If you are buying for the company, buy this one. DataArchitect is part of Power Designer, which is described below.
Database Design Tool (DDT)	from Jo Janssens	Free	A French version is also available. I found this link did not work on October 21st, 2002, but if you try it and find it works, please let me know.
Data Design Studio	Chilli Source	$99 downloaded or $189 on CD	DDS supports the ERD Modelling and reverse engineering is being added.

PRODUCT	VENDOR	PRICE	COMMENTS
Dezign	Datanamic	Offers a free download or $139 to buy.	If you are buying for yourself, buy this one - now offers a free Tutorial.
Designer	Oracle	$200	Available from the Oracle Technology Network (free to join), offering unbeatable value at a price of $200
Dia	Gnome Office	Free - Open Source	Designed to be like Visio
Enterprise Architect	Sparx Systems	$99	UML Analysis & Design Tool and very affordable.
ER Creator	Model Creator (Danish)	About $100	Includes a Tutorial and a Trial version. Nice, easy to use, simple tool.
ER/Studio	Embarcadero	Trial Version available	A comparative Study of Database Design Tools is available (requires registration).
ERWin	Computer Associates		One of the Market Leaders
ImportER MySQL	Datanamic	Excellent value at $55	Reverse Engineering for MySQL
MagicDraw	No Magic inc.	Varies	Free educational versions of the Personal edition for educational institutions
Object Database Designer	Oracle	POA	Combines Object and Relational Modelling facilities
OR-Compass	Logic Works	POA	Object-Relational Modelling. I saw this mentioned in a User Group somewhere, but the only reference I found is in this 1998 article in DBMS magazine. Logic Works was taken over by Platinum, which was taken over by CA, and the CA Product Listing does not include OR-Compass. If anyone can give me a URL, I'd be very grateful.
Pacestar UML Diagrammer	PAcestar	Trial Version available $239	A diagrammer rather than a CASE tool
Platinum Repository	Computer Associates		This is a Repository, rather than a Modelling Tool, with a 'Where-used' facility, and it lets you add entity types, such as Business Function, Business Role and Script.
Power Designer	Sybase	$2,000	If you are buying for the company, buy this one. The Data Modelling Tool in this composite product is called "DataArchitect". Unfortunately, the design of the web site has changed recently and information about Data Architect is difficult to find. But the product is excellent, and I have found its powerful Reverse Engineering facilities to be very useful.
ProVision WorkBench	Proforma Corporation.	Prices on Application	"A Repository-based integrated business process and object modeling toolset." An interesting range of Data Model Exchange facilities.
QDesigner	Quest.	Pricing difficult to find on the Quest Web Site	From the 'Toad Company' - looks good and is well-recommended by at least one satisfied user - a downloadable Trial is available(rather substantial at 63MB).
SmartDraw	SmartDraw	$50	Strictly a drawing package, so no SQL creation.

PRODUCT	VENDOR	PRICE	COMMENTS
Telelogic System Architect 2007	Formerly Popkin now Telelogic	$3,000-5,000	Up until early 2007 a student edition of System Architect was available on CD-Rom which cost around $34 (ISBN: 0-07-293278-3). Unfortunately Telelogic, the company that have taken over Popkins, have no plans to develop the student edition and are singularly unhelpful. The alternative tool they recommend, Telelogic modeler at http://www.ilogix.com/sublevel.aspx?id=1756 can be freely downloaded, however this is a far inferior piece of software compared to the student edition of system architect. After registering you can still download an evaluation copy of Telelogic System Architect at the Telelogic web site.
Togethersoft	TogetherSoft	?	Large application, designed for large corporations
Toolkit for Conceptual Modeling (TCM)	University of Twente, Holland	Free and Downloadable	Very interesting combination of Tools but not for the PC as runs under various Unix platforms
Visio	Microsoft	Trial CD available	Many versions - Standard ($199), Technical ($399), Professional ($399) and Enterprise ($999)
Visual UML	Visual Object Modelers	Trial time limited download	Many versions
Visible analyst	Visible	?	Designed for large corporations?

I agree with the anonymous author above about System Architect being the best if your company can afford it however my note above detailing my dissatisfaction with Telelogic means that I would now recommend MagicDraw.

Exercise 20. CASE Tools

Time: 15 minutes

Visit the MagicDraw site listed above and then some of the others.

Exercise 21. MCQ

Which of the following are true statements about CASE tools (choose three)?

a. They provide rule checking.

b. They provide a method of automatically generating a user interface for a data model.

c. They are relatively cheap to purchase.

d. They provide a repository for information about the model from which reports can be generated.

e. Some provide reverse engineering capabilities.

13. Exercises

Time: 360 minutes (6 hrs)

It is very important to practice the skill of developing ERDs, and to help you I have provided a number of more complex exercises than those you have encountered as you have worked through this document.

Go to my main website and download the "Scenarios for practicing modelling techniques" document: http://www.robin-beaumont.co.uk/virtualclassroom/contents.htm

This document contains about ten scenarios.

Select two or three scenarios to:

• Develop an ERD for each scenario

• Provide a description of each entity type

• List a number of attributes for each of the entities plus a description of each attribute

• List a set of constraints / assumptions - You do not have the luxury of being able to question the client?

Hints:

Remember to use your own 'expert domain knowledge' where appropriate. But always state clearly any assumptions / constraints you have made for each of the models.

By the time you have finished with the model you should have:

• Removed all recursive relationships as described in the page concerned with recursion.

• Converted any many to many relationships to multiple one to many relationships.

14. MCQs

I have repeated all the MCQs together as a revision exercise for you.

1. ERDs provide a description of:

    a. The processes that occur in the model
    b. The various entities and their relationships in the model
    c. The entities in the model
    d. The processes and data requirements of a model
    e. The User Interface aspects of the model

2. ERDs provide a (one correct):

    a. Detailed description of the data within a data model
    b. Detailed description of the proposed uses of a database
    c. Guide to the training costs
    d. High level description of the data within a data model
    e. Graphical picture which is of little use in developing a database

3. From the list below choose two reasons why it is important for a 'domain expert' such as you to learn about the ERD method:

    a. Provides insight into the mindset of database developers
    b. Is the only method available to describe the data requirements
    c. Provides credibility to IT personnel
    d. Forms the basis of most data modelling techniques
    e. Has been proved to be the most cost effective method of specifying data requirements

4. From the list below choose the **one option that describes the most desirable** 'domain expert' from the medical profession:

    a. Someone who has developed several databases but knows little of database modelling or current issues in medicine
    b. Someone who has little interest in how information may help the department
    c. Someone who has problems working in a collaborative environment
    d. Someone who has previously managed IT projects
    e. Someone who has knowledge of data modelling techniques and currently works in the appropriate situation

5. Which of the following provides the best description of an entity **type** (select one)?
    a. A specific concrete object with a defined set of processes (e.g. John Brown with diabetes)
    b. A value given to a particular attribute (e.g. height - 230 cm)
    c. A thing that we wish to collect data about where zero or more, possibly real world examples of it may exist
    d. A template for a group of things with the same set of characteristics that may exist in the real world
    e. An undefined concept that needs further clarification

6. Which of the following provides the best description of an entity **instance** (select one)?
    a. A specific concrete object with a defined set of processes (e.g. John Brown with diabetes)
    b. A value given to a particular attribute (e.g. height - 230 cm)
    c. A thing that we wish to collect data about where zero or more, possibly real world examples of it may exist
    d. A template for a group of things with the same set of characteristics that may exist in the real world
    e. An undefined concept that needs further clarification

7. From the following list select **two** entity instances:
    a. Steinway piano model D design
    b. McGill type forceps as used in surgical operations
    c. Tony Blair (British prime minister)
    d. PIII type chip that is found in many modern PCs
    e. An advice leaflet for chronic back pain issued to Mrs Smith on 02/10/2002

8. Which one of the following statements is true (select one)?

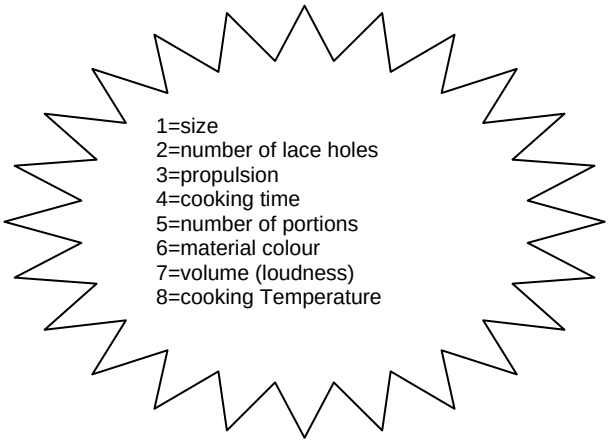
a. An ERD can only display entity type names.
b. Attributes are a rarely considered aspect of entities.
c. Attributes must always be displayed in ERDs.
d. Attributes equate to table names in a database.
e. Attributes can optionally be displayed in ERDs.

9. From the following, select the best list of attributes for the entity SHOE for someone working in a shoe shop (select one):

a. 1, 2, 6
b. 1, 2, 6, 3
c. 1, 2
d. 1, 6
e. 1, 2, 3, 4, 5, 6

10. From the following, select the best list of attributes for the entity RECIPE for someone following a recipe at home (select one):

a. 4, 5, 7, 8
b. 4, 5
c. 4, 5, 6
d. 4, 5, 8
e. 4, 5, 6, 8



11. Which of the following best describes the technique used to identify entity types in a narrative:

a. Identification of verbs, the application of various criteria and standards (eg naming conventions etc) to refine the list and then the creation of a list of appropriately named entity types
b. Identification of nouns then the application of ISO standards to create a list of appropriately named entity types
c. Identification of verbs then the application of standards (eg naming conventions etc) to create a list of appropriately named entity types
d. Identification of nouns, the application of various criteria (eg Reingruber & Gregory 1994's) and standards (eg naming conventions etc) to refine the list and then the creation of a list of appropriately named entity types
e. Identification of a few important nouns then the application of standards (eg naming conventions etc) to create a list of appropriately named entity types

12. Which of the following criteria are true (choose three)?

a. Attribute names must always be unique for a particular model.
b. Entity type names must be unique for a particular model.
c. Attribute names must be unique for a particular entity type within a model.
d. An entity type should only express one concept.
e. Entity type names should be singular or plural nouns.

13. Which of the following types of software (applications) is most suitable for developing ERDs?

a. Desktop drawing package (eg Smartdraw)
b. Desktop publishing application to ease the use of textual explanations
c. Anything with drawing capabilities (eg Microsoft Word)
d. A CASE tool
e. A spreadsheet

14. Which of the following statements is true?

a. A relationship within an ERD is always a link between various entity instances. It is implemented within a database by the use of a foreign key.
b. A relationship within an ERD is always a link between various entity instances. It is implemented within a database by the use of a primary key.
c. A relationship within an ERD is always a link between two entity instances within an entity type. It is implemented within a database by the use of a foreign key.
d. A relationship within an ERD is always a link between two entity instances within an entity type. It is implemented within a database by the use of a primary key.
e. A relationship within an ERD is always a link between two fields within an entity instance. It is implemented within a database by the use of a primary key.

15. Which of the following best describes an optional relationship?

a. An optional relationship is one where a foreign key attribute can only take a specific value. For example, patient 001 may have a medical record 003 but no other.
b. An optional relationship is one where a foreign key attribute can take a null value. For example, a patient may or may not have a medical record.
c. An optional relationship is one where a foreign key attribute can take any value except null. For example, a patient may have a medical record or a dummy record.
d. An optional relationship is one where a foreign key attribute may or may not exist. For example, a patient may have a medical record.
e. An optional relationship is one where a foreign key attribute can take any value regardless of the values in the associated primary key. For example, a patient may have a specific medical record, an undefined one or none at all (depending upon your interpretation).

16. Which of the following statements is correct (one only)?

a. A mandatory relationship is also called external dependency. A mandatory one to one relationship implies that one instance of entity A is always associated with one instance of B. A mandatory one to many relationship implies that one instance of entity A is always associated with zero or more instances of entity B.
b. A mandatory relationship is also called existence dependency. A mandatory one to one relationship implies that one instance of entity A is always associated with one instance of B. A mandatory one to many relationship implies that one instance of entity A is always associated with zero or more instances of entity B.
c. A mandatory relationship is also called existence dependency. A mandatory one to one relationship implies that one instance of entity A is always associated with one instance of B. A mandatory one to many relationship implies that one instance of entity A is always associated with one or more instances of entity B.
d. A mandatory relationship is also called external dependency. A mandatory one to one relationship implies that one instance of entity A is associated with zero or one instance of B. A mandatory one to many relationship implies that one instance of entity A is always associated with one or more instances of entity B.
e. A mandatory relationship is also called existence dependency. A mandatory one to one relationship implies that one instance of entity A is associated with zero or one instance of B. A mandatory one to many relationship implies that one instance of entity A is always associated with zero or more instances of entity B.

17. Which of the following statements best describes conceptual, logical and physical data models?

a. A loosely defined set of terms, often used inappropriately, indicating the gradual progression from a high level data model to one which provides all the detail required to implement it in a specific database system
b. A clearly defined set of terms, often used inappropriately, indicating the gradual progression from a data model containing attribute descriptions to one which provides all the detail required to implement it in a specific database system
c. A loosely defined set of terms, often used inappropriately, indicating the gradual progression from a data model containing attribute descriptions to one which provides all the detail required to implement it in a specific database system
d. A clearly defined set of terms, often used inappropriately, indicating the gradual progression from a high level data model to one which provides all the detail required to implement it in a Microsoft compliant database system
e. A clearly defined set of terms, often used inappropriately, indicating the gradual progression from a high level data model to one which provides all the detail required to implement it in a referential database system

18. Which of the following are true statements about CASE tools (choose three)?

a. They provide rule checking.
b. They provide a method of automatically generating a user interface for a data model.
c. They are relatively cheap to purchase.
d. They provide a repository for information about the model from which reports can be generated.
e. Some provide reverse engineering capabilities.

19. Which of the following are correct statements about recursion (choose two)?

a. Recursion is a valuable modelling technique in detailed data models.
b. Recursion is added when making a data model comply to fifth business normal form.
c. Recursion is added at the end of the modelling process.
d. Recursion is modelled by introducing ‘structure’ entity types in the latter data models.
e. Recursion is something that is allowed at the beginning of the modelling process.

20. ERD modelling is which of the following (choose two)?

a. A highly objective method of obtaining data requirements
b. A technique developed over the last two decades
c. A process that requires knowledge of the context of the model to be developed
d. A very subjective method that produces highly personalised models
e. A technique only used in small-scale database development

15. Summary

This document has introduced you to a number of new concepts and provided you with the skills to use them effectively to produce ERDs. You have seen how to define and refine a set of entity types for a given scenario. You have also considered in detail the various types of relationship that can exist between entity types.

The CASE tool concept has been introduced to you by way of seeing what a number of suppliers have to offer. Other documents provide practical exercises in using CASE tools.

I would now recommend that you return to the learning outcomes at the beginning of the document and see how much you have learnt!

16. References

Carter John 2000 Database design & programming with Access, SQL and visual basic McGraw Hill

Carter John 1995 The Relational Database. Chapman and Hall [An excellent introductory book]

Date C J. 1995 (6th ed.) An introduction to database systems. Addison-Wesley.

Elmasri R Navathe S B 1989 Fundamentals of database systems. Benjamin Cummings Wokingham UK.

Everest G 1986 Database management. McGraw-Hill. London.

Finkelstein Clive 1989 An introduction to information engineering. Addision-Wesley. Wokingham UK.

Finkelstein Clive 1992 Information engineering. Addision-Wesley. Wokingham UK.

Hernandez M J 1997 Database design for mere Mortals. Addison - Wesley

Martin James 1981 An end user's guide to data base. Prentice Hall [ISBN 0-13-277129-2]

Reingruber Michael C. Gregory William W 1994 The Data Modelling Handbook John Wiley & Sons Chichester

Rumbaugh J Blaha M Premerlani W et al 1991 Object-Oriented Modelling and design. Prentice Hall.

Blaha M Rumbaugh 2005 (2nd ed.) Object-Oriented Modelling and design with UML. [basically this is the seocnd edition of Rumbaugh et al 1991]Prentice Hall.

17. Links

University of Texas data modelling notes:
http://www.utexas.edu/cc/database/datamodeling/dm/objects.html

Links to database articles (applied information science web site):
http://www.aisintl.com/case/reading.html

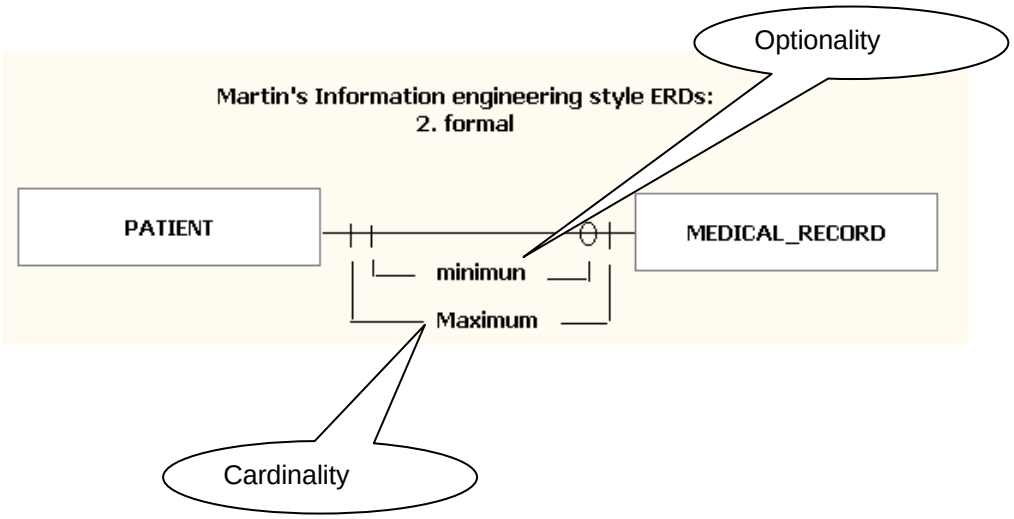
Wikipedia has a good article about ERDs:
http://en.wikipedia.org/wiki/Entity-relationship_model

Additional ones for you to add:

18. Appendix A Relationship Terminology

The following table is taken from Carter 1995 p40. Different writers use different words to describe the minimum and maximum constraint on a relationship; the synonyms are listed below. I have tried to avoid using most of the terms in this document to avoid confusion. I tend to refer to relationships as being “optional” (where the minimum number of participants is zero) or “mandatory” (where the minimum is one).

Source (writer)	Number of entities in relationship	Minimum number of participants	Maximum number of participants
Date			Degree
IEW		Optionality	Cardinality
D.C.C		Optionality	Degree
Ashworth			Degree
Eva		Optionality	Cardinality and degree
Kroenke	Degree	Minimal Cardinality	Maximal Cardinality
Bamford			Degree



Document details:

C:\HIcourseweb new\chap11\s9\erds_1.doc

End of document