

(Free) Sample 2

Taken from:

E-book 2: Conceptual Data Models

Knowledge River Ltd



(Free) Sample 2

Welcome to **Knowledge River** – the on-line e-learning store for Computer Science practitioners, academics and students.

Thank you for taking the time to download and examine this free sample – based on one of our popular e-learning products. In this free sample we have tried to give you a flavour of the full document upon which it is based. If you like what you see and would like to purchase the full document then please follow the instructions on our website: www.knowledge-river.co.uk

Whilst every effort has been made to ensure the accuracy of this material and any included software code has been tested carefully, they are only intended for instructional and illustrative purposes and are not guaranteed for any particular application, purpose or function. Knowledge River Ltd does not offer any warranties or representations, nor do we accept any liabilities with respect to any code examples included here.

Please note that Knowledge River provides Computer Science educational material in electronic format – we do not offer an on-line IT consultancy or help-desk function and so we cannot get into detailed correspondence on specific technical issues. However, we would appreciate general feedback and comments on what you think of the material, its layout, its usefulness and how it could be improved etc.

Please email us at: **feedback@knowledge-river.co.uk**

Copyright © Knowledge River Ltd 2006-7

All rights reserved.

Please see the accompanying notes on the Knowledge River Ltd usage policy.

Knowledge River Ltd - Usage Policy (How to use our products)

We hope you enjoy using these e-learning products and get a lot of benefit by doing so. However, these electronic resources do represent a very considerable investment in intellectual effort, time and money on our behalf and naturally we want to protect our intellectual property. These products are meant for *your own personal educational use* and should not be modified, altered, edited, re-sold or transferred to a third party.

Please use and enjoy our work but please respect our efforts too.

The table below summarizes what you can and cannot do with our e-learning products:

Action	Allowed?	Notes
Reading	Yes	Obviously! On-screen or off-screen (see printing).
Printing / Making a Hardcopy	Yes (Limited)	Only single copies for <i>your own personal use</i> . Multiple copies and/or photocopies intended for distribution or sale to third parties is NOT allowed.
Editing / Updating / Extending	No	The e-learning products you purchase from Knowledge River Ltd must NOT be edited or modified in any way. This is how they were intended to be and this is how we want them to stay. Edits are our responsibility.
Copying (Electronic)	Yes (Limited)	You may copy the purchased e-learning product for <i>your own personal use</i> within your own computing equipment and storage devices (eg on a desktop and laptop with a backup on a portable storage device). However, you must NOT make multiple copies of these products with the intention of passing these copies onto third parties (so doing copies for your friends or putting copies on a server for sale or distribution is not allowed). In short, keep one or two copies for <i>your own personal convenience</i> but do not give copies to other people. This rule applies regardless of whether or not you charge money for it.
Transferring / Passing On	No	Each e-learning product purchased is intended for <i>one individual</i> – so please do not pass copies onto third parties – let them purchase their own. This rule applies even if there is no money involved or financial gain for yourself – please do not pass these products around – ever. Knowledge River products are non-transferable.
Re-selling	No	It goes without saying that this is never allowed – taking a copy of our work and then selling it on is simply theft and is immoral - please don't even think about doing this.
Anything Else	No	Anything not already covered is not allowed – please treat our company and products with respect. Enjoy.

The full *E-book 2* contents are:

1. What exactly is a conceptual data model?
2. Is there a step before the conceptual data model?
3. Where else can I get information for my conceptual data model?
4. What are the user's data requirements?
5. Why do these requirements matter?
6. How does abstraction relate data models to reality?
7. What makes a good data model?
8. What happens if the data model is wrong?
9. How do you know it is wrong?
10. At what stage do we build a conceptual data model?
11. Who is responsible for building the conceptual data model?
12. What is the client's role in all this?
13. How many conceptual data models will be needed?
14. What is an ERD?
15. What is the history of the ERD?
16. What are the components of an ERD?
17. Are all entities equal?
18. Are all attributes equal?
19. Are all relationships equal?
20. Why are there different syntax details?
21. What is the difference between an ERD and an ERM?
22. Are there any guidelines to selecting your entities?
23. Are there any guidelines to selecting your attributes?
24. Are there any guidelines to selecting your primary keys?
25. What differentiates constraints from assumptions?
26. What are entity *super-types* and *sub-types*?
27. What are the problems of entity sub-typing?
28. Are there any pitfalls in building data models?
29. What about Many-to-Many relationships?
30. What is *not* on a finished ER Model?
31. How do you know when the ER model is finished?
32. What do you do with a finished ER model?

Like this sample, the full document is in PDF format and can be purchased individually at a modest price or you may like to buy several different documents and get *very substantial discounts* – full details on our website. Thank you once again for taking the time to examine this **Knowledge River** educational material – please have a look at our website or drop us an email.

Below are three examples...

What are the user's data requirements?

Via the above requirements collection and scoping stage, the user will express what they wish to see in the database and give some indication of how that data will be used. Much of this interaction between modeller and domain expert will be informal and loosely expressed. It is the modeller's task to very accurately establish the (persistent) data requirements of the organisation (or that part of the organisation) who requested a database and how those data items interact. This means turning informal and loose descriptions into firm, accurate and semantically clear:

- Entities
- Attributes
- Relationships
- Constraints
- Assumptions

As an example, a doctor may loosely say that each patient occupies one bed in each ward at a time and that the average patient is attended by two nurses at any one time. As a data modeller you need to establish:

- What constitutes a patient – an out-patient, an in-patient or both?
- What constitutes a ward – are Intensive Care and Accident & Emergency wards?
- What constitutes a nurse – what about auxiliary nurses, ward sisters or midwives?
- What constitutes a doctor – consultants, senior house officers, junior doctors, all?
- What constitutes a bed – ward-based, temporary beds in corridors, others?
- What constitutes attendance – being on the ward, being on-call (on or off-site)?

The point being made here is that natural language descriptions (like English) are notoriously vague and open to misinterpretation – that is why in most areas of science and engineering formal languages like mathematics, logic and schematic diagrams are used – because then any semantic uncertainties can be ironed out.

As you will shortly see, conceptual data models adopt this rigorous approach by having their own specialized notation made up of entities, attributes and relationships (plus other items). The mechanics of building conceptual data models is covered in depth soon.

Building a complete, correct and semantically accurate data model based on often vague and informal user inputs is not an easy task and takes many iterations. But it is vital that it is done properly as it forms the foundation for all subsequent database activities.

What are the components of an ERD?

A typical ERD will comprise:

- Entity Types
- Attributes
- Relationships

The ERD is itself part of an **ER Model** – which also includes a list of constraints and assumptions. We will return to ER models shortly but first we get to grips with the mechanics of the ERD.

An **entity type** is a logical or physical object that has its own independent existence within the universe of discourse and is of interest to the modeller. If we assume a universe of discourse comprising a university, then typical *physical* examples could be: student, tutor, room, desk, projector and computer. Typical *logical* examples could include: grade, extension, qualification, module and course. It is quite normal to mix physical and logical entities on the same ERD – they are treated as being the same. In a banking scenario, *physical* entities could be: customer, employee, branch or manager while *logical* entities could include loan, mortgage, overdraft or account.

Each entity type will have one or more **entity instantiations** (an instantiation is simply an example of the type). For example, if we have an entity type called *module*, instantiations of that module could include: Java Programming, Networking Fundamentals, Advanced C++ or Database Development. For the entity type *student* we could have instantiations of Stuart, Simon, Susan, Steven or Sarah etc while the type *qualification* could have instantiations of BSc, MSc, PhD, MA, MBA etc.

Each entity instantiation must be uniquely identifiable in some way (this is related to primary keys which we cover later). Entity instantiations are also known as entity occurrences.

So entity **types** are classes of individual objects known as entity **instantiations**. The ERD always shows entity types (the instantiations are never shown on an ERD but become rows in the database tables later when the database is implemented).

Each entity type (and so all of its instantiations) will have a set of **attributes**. An attribute is simply a property of an entity that is of interest to the modeller. Each attribute is defined on a **domain** (a set of values that may be assigned to that particular attribute). Attributes can share common domains but they can also have their own user-defined domains.

See below:

(Free) Sample 2

Here are some examples:

Entity Type	Attributes
Student	Student ID, First Name, Last Name, Address, Telephone, Mobile
Tutor	Staff ID, First Name, Last Name, Department, Room, Extension
Module	Module Code, Title, Description, Start Date, End Date, Credits
Room	Number, Floor, Building, Seating Capacity, Type
Course	Course Code, Title, Director, Duration, Description

Each of these attributes will be defined over a domain (here we show just a selection):

Attribute	Domain	Comments
Student ID	Integer	Fixed format 8 digit number
Staff ID	Integer	Fixed format 8 digit number
Student First Name	Variable Character (20)	Holds 0-20 characters
Address	Variable Character (100)	Holds 0-100 characters
Telephone	Character (12)	Holds 0-12 characters (numbers)
Module Code	Character (4)	Fixed format 4 character like T123
Duration	Integer IN (1, 2, 3)	Value must be 1, 2 or 3 only
Credits	Integer IN (10, 30, 60, 90)	Value must be 10, 30, 60 or 90 only

So far we have covered entities and their attributes. The final part of the ERD trilogy concerns **relationships**.

In some senses, the relationships between the data are more valuable than the data itself. For example, if you were given a list of patients, a list of wards and a list of doctors in a hospital it would not inform you greatly (lots of data but little information). However, if you were then given the relationships between patients, wards and doctors (which doctors treat which patients in which wards) then the information content rises dramatically and your knowledge has advanced considerably.

Relationship types specify some form of interaction or association between entity types. In the hospital example, the relationship 'Treats' links the entities Doctor and Patient while the relationship 'Resides' links Patient and Ward. Just as with entities, we also have **relationship instantiations** (occurrences) which are particular examples of that relationship type (so Mrs Brown *resides* on Ward 7 and Dr. Smith *treats* her).

Other relationship **type** examples (in italics) would be: Student *registers* for Course and *enrols* on a Module where s/he is *taught* by a tutor.

A relationship **instantiation** would be: Stuart *registers* for BSc Computer Science and *enrols* on Java Programming where he is *taught* by Dr. Roberts.

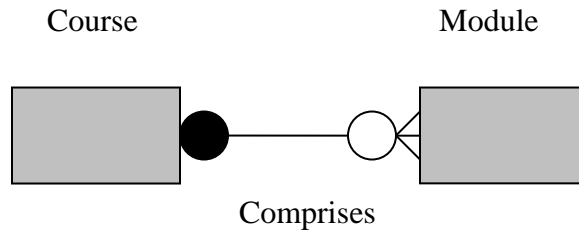
Shortly, we will see how these ideas can be put together using diagrammatic notation.

Why are there different syntax details?

So there we have it – entities, attributes and relationships. It is important that you realize that there are different syntax (symbols/drawing) conventions used when constructing an ERD. The semantics (meaning) is always the same but the same ERD can look quite radically different.

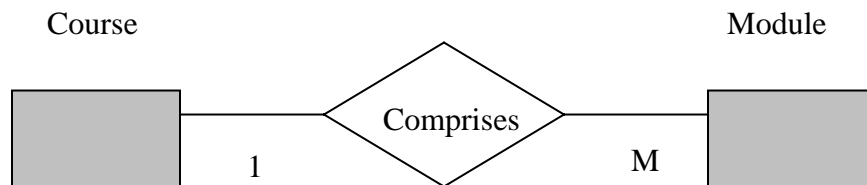
Syntax version 1

This is the version already used. We have boxes for entities and simple lines for relationships. Optional participation conditions use a clear circle and mandatory a black circle. The cardinality is represented by a simple line ('one') or a 'crow's foot' ('many'). Attributes do not appear directly on the actual ERD. Here is another example:



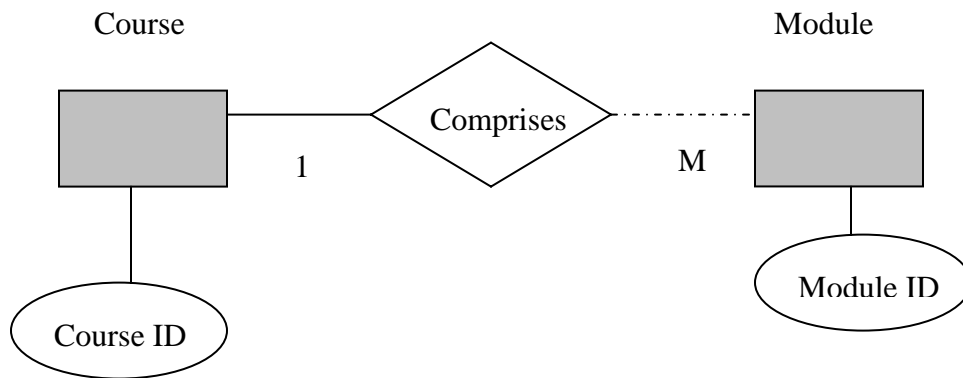
Syntax version 2

Another form keeps the boxes for the entities but uses diamonds for the relationships, writes the cardinality on the diagram:



Syntax version 3

Another form uses full lines for mandatory and dotted lines for optional participation conditions (and some even add the attributes directly onto the ERD):



In summary – just be aware that even though every ERD will contain entities, attributes and relationships (the semantic content) the precise drawing conventions may well differ.

We will use the first syntax version.