Informatics 1: Data & Analysis
Session 2012–2013, Semester 2

Coursework Assignment Student Notes

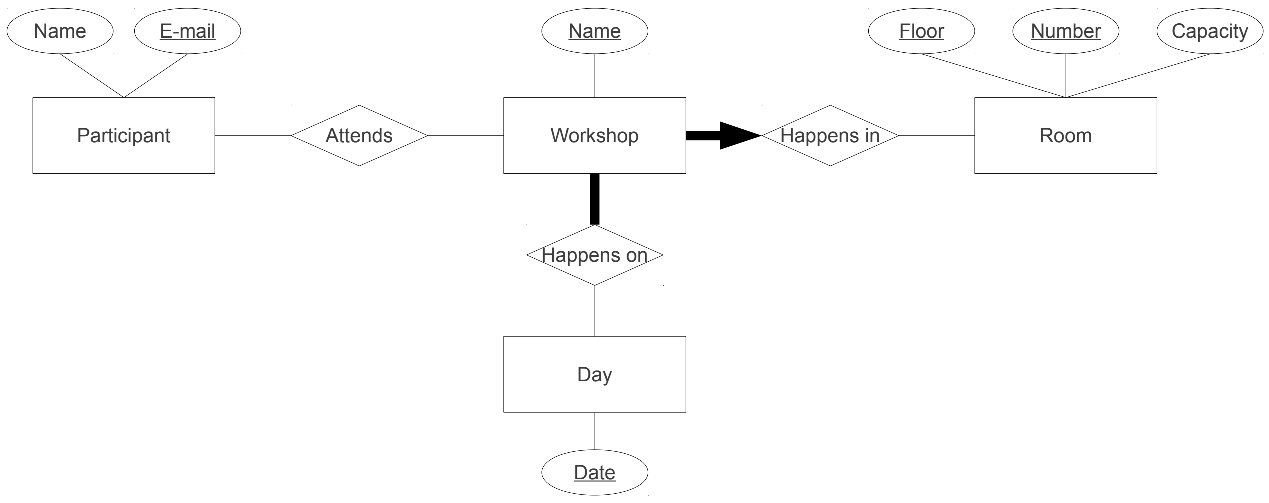
The coursework assignment for Inf1-DA 2012–2013 is the resit examination paper from August 2011. These are notes for students on the course, to help you review your work on the assignment, and with comments based on scripts submitted in the actual resit examination. Although these notes contains information about solutions, they are not necessarily model answers. They are provided as guidance to assess your own answers, not always to dictate what is "the" answer.

If you find an error in these notes, please email me Ian.Stark@ed.ac.uk

Thank you for your participation in Informatics 1 this year.

Ian Stark
2013-03-24

1. (a) Here is a suitable entity-relationship diagram.



One incorrect alternative is to assign a “date” attribute to workshops: the question specifically mentions multiple-day workshops.

As well as the entities and relationships, it’s important to include the underlined primary keys — notice the composite key of Floor and Number for rooms — the thick lines of two participation constraints for Workshop in Happens in and Happens on, and the arrowhead of a key constraint for Workshop in Happens on.

(b) (i) A *key* is a minimal set of attributes whose values uniquely identify an item in an entity set. For example, the e-mail address of a participant is a key for the corresponding entity set.

(ii) A *composite key* is a key that includes more than one attribute. For example, the floor and room number of a meeting room.

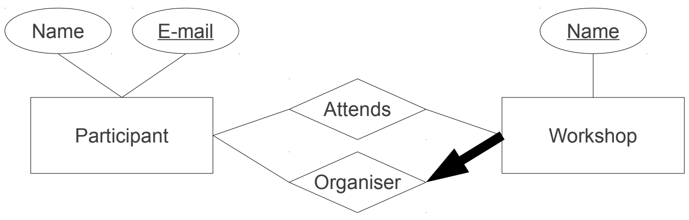
(iii) *Total participation* is a requirement that every element of an entity set must appear at least once in a particular relationship. For example, the requirements that every workshop must happen on at least one date, and in some meeting room.

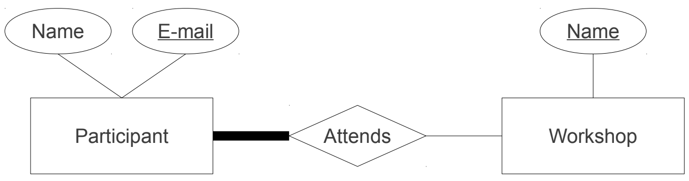
(iv) A *key constraint* is a requirement that each element of an entity set may appear at most once in a particular relationship. For example, the requirement that every workshop must be allocated no more than one room, even if it runs for more than one day.

A total participation constraint is shown by a thick line joining the participating entity to the relationship. A key constraint is shown by an arrowhead on the line joining the entity to the relationship.

(c) (i) Two of these are readily presented in an ER diagram: the requirement for workshop organisers, and for every participant to register for at least one workshop. Making sure that no participant registers for two workshops on the same day is much harder to incorporate within an ER diagram, and might be better done as a dynamic check when people make their registrations.

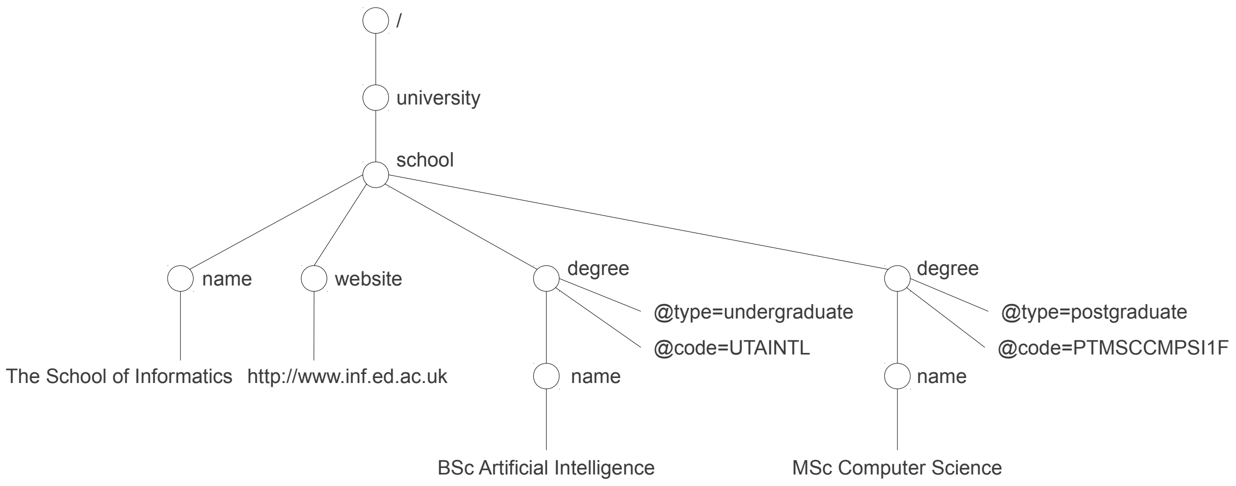
(ii) Here are suitable modifications to the ER diagram.





Recording organisers requires an additional relationship, with both a participation and a key constraint; requiring participants to be registered for at least one workshop can be represented as a participation constraint on the existing Attends relationship.

2. (a) Here is the XPath tree model.



Notice that the following are all distinct: leaves with general text; internal nodes with their type; and attributes with a type and value.

(b) The following is a suitable DTD.

```
<!DOCTYPE university [
<!ELEMENT university   (school+) >
<!ELEMENT school       (name,website,degree+) >
<!ELEMENT name         (#PCDATA) >
<!ELEMENT website      (#PCDATA) >
<!ELEMENT degree       (name) >
<!ATTLIST degree
    type (undergraduate|postgraduate)  #REQUIRED
    code CDATA                         #REQUIRED >
]>
```

Some small variations are possible. For example: the ATTLIST might be split into two statements rather than one; and elements could use $*$ rather than $+$, allowing for empty lists of schools or degrees.

(c) (i) //school/name/text()
Not //name/text(), which will get both school and degree names.

(ii) //degree[@type='undergraduate']/name/text()

(iii) //school[degree/@type='postgraduate']/website/text() *or*
//degree[@type='postgraduate']/../website/text()

Again, various alternatives are possible: naming every node on the path rather than using the descendant axis "//"; more use of the parent axis ".."; or even long-form queries rather than the standard abbreviations.

(d) Here is an example SQL data declaration for the two tables, which captures the information from the XML document.

```
create table Schools (
  name     varchar(80),
  website  varchar(80),
  primary key (name)
)
```

```
create table Degrees (
  name   varchar(80),
  type   varchar(20),
  code   varchar(20),
  school varchar(80),
  primary key (code),
  foreign  key (school) references Schools (name) on delete cascade
)
```

Variations are possible: all sorts of possible field lengths, where anything reasonable will do; and anything could reasonably be labelled **not null**. Some things, however, are vital: for example, the **foreign key** reference between Degrees and School.

Notice the **on delete cascade** declaration, which indicates that if the entry for a school is deleted, its associated degrees should also be removed.

(e) (i) **SELECT** name **FROM** Schools

(ii) **SELECT** name **FROM** Degrees **WHERE** type = "undergraduate"

(iii) **SELECT** S.website **FROM** Degrees D, Schools S
WHERE D.type = "postgraduate" **AND** D.school = S.name

3. (a) The *information retrieval task* is to find those documents relevant to a user query from among some large collection of documents.

   For example, searching for previous legal rulings relevant to a certain topic from a judicial archive. The judicial archive is the document collection; the query is some words related to the topic; and the previous rulings are the relevant documents to be retrieved.

   Another example might be to search for recipes in a user-contributed online repository. The online repository is the document collection; the query is some words, perhaps ingredients or the name of a dish; and the recipes are the relevant documents to be retrieved.

   Any appropriate example will do.

   (b) *Precision* records what proportion of the documents retrieved do in fact match the query; *recall* is the proportion of relevant documents in the collection which are successfully retrieved.

   (c)
   - *TP* is *True Positives*, the number of relevant documents correctly returned.
   - *FP* is *False Positives*, the number of irrelevant documents returned.
   - *FN* is *False Negatives*, the number of relevant documents incorrectly rejected.

   It's important to answer the question precisely here: give both the name and definition in each case.

   (d) These tables set out the performance results for each retrieval system, and the calculation required for precision and recall of both.

X	Relevant	Not relevant	Total
Retrieved	40	360	400
Not retrieved	10	1090	1100
Total	50	1450	1500

Y	Relevant	Not relevant	Total
Retrieved	15	15	30
Not retrieved	35	1435	1470
Total	50	1450	1500

$$\text{System } X \text{ precision } P = \frac{40}{400} = 0.1 \qquad \text{System } Y \text{ precision } P = \frac{15}{30} = 0.5$$

$$\text{System } X \text{ recall } R = \frac{40}{50} = 0.8 \qquad \text{System } Y \text{ recall } R = \frac{15}{50} = 0.3$$

   (e) Here is the standard formula defining F-score.

$$F_\alpha = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}}$$

   (f) For the retrieval of legal judgements, recall is of particular importance (you really don't want to miss anything), so value of α below 0.5, say 0.2, might be appropriate.

   For a recipe website, precision seems likely more important than recall — there may be far more recipes on any topic than you need to see, but any returned should match the query as well as possible. A value of α above 0.5, say 0.9, might be appropriate.

   If you have a different example, the value and justification should fit that. This could include the harmonic mean at 0.5 if that's appropriate.