Tutorial 8: Practice Exam Questions

Informatics 1 Data & Analysis — Notes on Solutions

Week 10, Semester 2, 2017/18

Read this first: it is not the same as the other tutorials

Following the strike by university staff I have changed around some of the tutorial work. The material usually in Tutorial 8 has now moved a week later and will appear in Tutorial 9, by which time the topics it addresses will have been covered in lectures. In its place this week I am providing some exam question practice.

This worksheet includes two questions, both taken from the 2016/17 resit examination.

- Before your tutorial, attempt both questions. Write out your answers on paper and with all your working: just as you would expect to do in the exam itself.
- Bring your written work to the tutorial.
- At the tutorial you will use the marking guidelines from the original exam to assess these solutions. You will be working with others and with the help of your tutor.

You don’t need to do this work under exam conditions. As usual with Inf1-DA tutorials, this is formative assessment entirely for your feedback and learning. No marks here will directly contribute to your final grade. Because of this you can freely ask for help on the questions, discuss on Piazza, and talk about your work with other students. Please do. The aim here is for you to get practice answering questions, find out what’s involved, and prepare for doing this well in the exam itself.

If you have problems with some parts of the questions then write down what it is that you find challenging and ask your tutor about this in the meeting.

If you are uncertain about the meaning of some question, or think that there are several possible answers, then write all this down and raise it for discussion in the tutorial.

As usual, it’s important both for your learning and other students in the group that you come to tutorials properly prepared. Students who have not attempted the exam questions will be sent away from the tutorial to do them elsewhere and return later.

Tutorials are a key part of course participation: if you are ill or otherwise unable to attend one week then email your tutor, and if possible attend another tutorial group in the same week.

Please send any corrections and suggestions to Ian.Stark@ed.ac.uk
Solution Notes

This document contains the original tutorial questions together with some notes on sample solutions. These notes highlight key points, indicate possible errors, and report on some of the ways people approached the questions in the exam itself.

These are not entirely “model” answers; instead, they show some possible solutions. After all, not every question necessarily has a single “right” answer. There can be multiple correct ways to write a database query, explain a concept, or construct an example. These notes include some variants on answers, but still cannot cover every possible correct alternative.

Exam Marking

Every exam paper shows how marks are assigned between different parts of each question. Within that, the number of marks awarded for each part depends on the quality of the answer submitted. If there is no answer given at all, or the answer given is wholly incorrect, then it receives zero. If the answer is entirely correct then it gets the full total. If it is partially correct it receives some of the marks.

Inf1-DA always uses positive marking: marks are awarded for things present in an answer, never deducted for absence. In Inf1-DA there’s also no scaling or “curving” to fit results to some profile — the mark for a script depends only on what’s written there, not on the performance of anyone else in the exam.
1. [This question is worth a total of 40 marks.]

(a) An online app store uses a relational database to keep track of everything in the store. One of the tables holds information about all apps in the store’s online catalogue, and is defined as follows.

```
create table app (
    title varchar(100) not null, -- Each app has a different title
    version integer not null, -- An app might have several versions
    author varchar(100) not null, -- One author can write multiple apps
    cataloguecode varchar(10) not null, -- Unique identifier for items in the store
    price integer not null, -- Price in pennies
    primary key (cataloguecode)
)
```

As you can see from the comments, the store might hold several versions of the same app, but they will always have different catalogue codes.

The following five terms have precise meanings when applied to a relational database. For each one, give a one-sentence description of its meaning and an example based on the `app` table declaration above.

(i) Superkey
(ii) Key
(iii) Composite key
(iv) Candidate key
(v) Primary key

[10 marks]

QUESTION CONTINUES ON NEXT PAGE
The following entity-relationship diagram shows part of a conceptual design document for another database, this time for a bookshop.

![ER Diagram](image_url)

The diagram is accompanied by the following information.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RRP</td>
<td>Recommended Retail Price: Sale price suggested by the publisher. Record as an integer number of pence.</td>
</tr>
</tbody>
</table>

Names are required, email and web addresses are optional; and all of these are assumed to fit within 300 characters.

(b) The double line and arrow joining the Book entity to the Publishes relationship indicates two distinct types of constraint in this link.
   (i) What are the two constraints called?
   (ii) Which graphical feature indicates which constraint?
   (iii) What does each constraint mean?  

(c) Design an SQL data declaration of tables suitable to implement this entity-relationship model. You should take into account all the constraints shown in the ER diagram and its accompanying information. Include not null and on delete declarations where appropriate.
Notes on Question 1

(a) (i) A superkey is any set of fields whose values taken together uniquely identify a record. For example, the combination of author, title and version.

Something that was actually a key would also serve as a superkey — and gets the marks — but it’s generally clearer to use something that includes more than is necessary for a key

(ii) A key is a minimal set of fields whose values uniquely identify a record. For example, the combination of title and version here; alternatively, the cataloguecode on its own.

It’s essential to mention that a key is a set of fields, not necessarily just a single field, and that it is minimal for identifying a record.

(iii) A composite key is a key that includes more than one field. For example, the combination of title and version.

(iv) Where there are several keys available for a relation, each one is a candidate key. Here both \{title, version\} and \{cataloguecode\} are candidate keys.

(v) A primary key is the key chosen among all candidate keys to be used as the unique identifier of records in a particular database table. In this case the primary key is \{cataloguecode\}.

(b) (i) A key constraint and a total participation constraint.

(ii) The arrowhead indicates the key constraint, and the double line indicates the total participation constraint.

(iii) The key constraint means that each Book entity instance can participate in at most one Publishes relationship instance. That is, each book can be associated with at most one Author/Publisher pairing.

The total participation constraint means that each Book entity instance must participate in at least one Publishes relationship instance. That is, each book must be associated with at least one Author/Publisher pairing.

Together these mean that for each book the database must record exactly one author and one publisher.
(c) The following DDL code captures the ternary relationship and its constraints.

```sql
create table Book (
    ISBN varchar(13),
    authorname varchar(300) not null,
    publishername varchar(300) not null,
    rrp integer not null,
    primary key (ISBN),
    foreign key (authorname) references Author(name) on delete cascade,
    foreign key (publishername) references Publisher(name) on delete cascade
)

create table Author (
    name varchar(300),
    email varchar(300),
    primary key (name)
)

create table Publisher (
    name varchar(300),
    website varchar(300),
    primary key (name)
)
```

The **not null** declarations listed here are all essential. It’s also acceptable to put them on the primary key fields as well.

The **on delete** declarations for foreign keys must be **cascade** rather than **set null** to maintain the total participation constraint in the ER diagram.

The lengths of all **varchar** fields are given in the question: some students made up their own, which didn’t get such good marks.

This solution incorporates all material from the Publishes relationship in the Book table. This is possible because of the key constraint: each book has at most one associated author/publisher pair, and bringing those fields into the Book table builds that constraint into the database. Putting **not null** declarations on the **authorname** and **publishername** fields then enforces the total participation constraint, too.

Several students gave a separate table for the ternary Publishes relationship, but then did not capture the constraints on Book from the ER diagram. For full marks an answer had to express both constraints. In general, relationships only need their own table where there are no key constraints involved.

A few students linked tables up the wrong way round, with **foreign key** declarations in Author or Publisher tables that linked to a Book. That’s entirely incorrect. Given a specific author, you can’t follow a foreign key to “their” book, because they may have written several. The appropriate direction matches the arrow in the ER diagram: from a book, you can follow a reference to find out its author or publisher.
2. [This question is worth a total of 30 marks.]

The following small XML document is a marked-up version of a fragment from the play *Rosencrantz and Guildenstern Are Dead* by Tom Stoppard.

```xml
<speech speaker="Guildenstern">
  <line>
    <w>The</w>
    <w>colours</w>
    <w>red</w>
    <punct>,</punct>
    <w>blue</w>
    <w>and</w>
    <w>green</w>
    <w>are</w>
    <w>real</w>
  </line>
  <line>
    <w>The</w>
    <w>colour</w>
    <w>yellow</w>
    <w>is</w>
    <w>a</w>
    <w>mystical</w>
    <w>experience</w>
  </line>
</speech>
```

(a) Draw the XPath data model for this document as a tree. [8 marks]

(b) Write an XML DTD for a Speech document type to validate such speeches. Assume that every speech must have an identified speaker. [10 marks]

(c) Suppose a large XML document contains many speeches, nested inside Plays, Acts, Scenes and so on. Write XPath expressions to extract the following.

(i) All lines spoken by Rosencrantz.

(ii) All speakers using the word “yellow” in a speech. [6 marks]

(d) Standard corpora for linguistic research like the British National Corpus bring together work from many sources. Building them requires balancing and sampling in order to ensure that they are representative.

Explain the meaning of balancing, sampling and representative here. [6 marks]
Notes on Question 2

(a) The document has the following tree structure in the XPath data model.

```
/speech[@speaker="Guildenstern"]
```

The colours red, blue and green are real. The colour yellow is a mystical experience.

In the exam itself most students drew this just fine. Even so, there are various points that need to be observed carefully for full marks.

- Nodes must be in correct left-to-right order — the only exception is the attribute node which can be attached in any direction.
- The attribute node "@speaker=..." must use the "@"-sign to show it is an attribute. Quotation marks are good, too, but I would not regard as essential here.
- There has to be a root node "/" at the top of the tree, above the root element "speech". This is a forward slash, not a backward slash.

(b) Here is an appropriate DTD.

```
<!ELEMENT speech (line)+  >
<!ELEMENT line  (w|punct)+ >
<!ELEMENT w    #PCDATA >
<!ELEMENT punct  #PCDATA >
<!ATTLIST speech speaker CDATA #REQUIRED >
```

The `speech` and `line` elements could possibly use "*" repetition instead of "+", although that then allows empty speeches and lines.

There are a few possible variations for the first two elements: for example `speech` could be `(line+)`, and `line` could be `(punct|w)+`.

The declarations can appear in any order.

Precise syntax is important here: `#PCDATA` and `#REQUIRED` both have a "#", while `CDATA` doesn’t.

The question states that every speech must have an identified speaker: so the `speaker` attribute on `speech` has to be declared as `#REQUIRED`.

(c) (i) All lines spoken by Rosencrantz:

```
//speech[@speaker='Rosencrantz']/line
```

This will return the lines as XML nodes, with the actual lines as subtrees of individual words and punctuation marks.
It’s also possible to pick out just the text from these, using variants like the following two.

```xml
//speech[@speaker='Rosencrantz']/line//text()
//speech[@speaker='Rosencrantz']/line/*/text()
```

Even so, each word and punctuation mark will still be returned separately. Extracting the complete sentence as pure text would require a more sophisticated query language. Some students returned only part of the sentences by selecting just words and not punctuation:

```xml
//speech[@speaker='Rosencrantz']/line/w
//speech[@speaker='Rosencrantz']/line/w/text()
```

That only received partial marks.

(ii) All speakers using the word “yellow” somewhere in a speech:

```xml
//speech[line/w/text()='yellow']/@speaker
```

Various abbreviations or variations on this are possible too, for example:

```xml
//speech[line/w/text()='yellow']/@speaker
//speech[.//w='yellow']/@speaker
//[w='yellow']/../../@speaker
```

Remember from lectures that “//speech[.//w='yellow']/@speaker” will not work because that “/” goes back up to the top of the tree, unlike “/” which works down from the current node. Some students used backward slash “\” rather than forward slash “/”. That’s an error, and meant those answers did not get full marks.

(d) **Balancing** means choosing a range of different types of sources for the corpus: books, newspapers, blogs, letters, etc.

**Sampling** refers to selecting texts at random from the chosen sources.

**Representative** A corpus is *representative* if it contains a similar mix of text to the language variant for which it is being developed.

Note that the question asks for explanations of three specific terms: in the original exam quite a few students just gave general descriptions of corpora. Even if the descriptions were correct, they didn’t answer the question.

All three of these are standard definitions from the course. Key points to note are that balancing is about selecting different types of source, not just choosing many sources; and that sampling must be random in some way.

Saying that a corpus is “representative” if it correctly represents the source material is circular. That’s not really enough for the marks: there needs to be some unpacking of what the word means.

One alternative statement is that a corpus is representative if it is suitable for making predictions about the original language of interest. That’s rather abstract compared to the “similar mix of text” explanation, but would still be fine.