Coursework commentaries 2015–2016

CO2209 Database systems – Coursework assignment 1

General remarks

Most students completed all of the coursework. Where students lost marks, it was generally either because of a 'No attempt' on certain questions or an incorrect answer on one or more of the SQL queries. The most important advice arising from this year's coursework is: start early! Many – if not the majority – of the 'No attempts' were on essay-style questions that were not difficult but may have been rather time-consuming. It is hard to escape the conclusion that some students delayed doing the coursework until they were under great time pressure.

Comments on specific questions

The coursework was divided into two sections, which might be called 'practical' and 'theoretical'. The first was a 'doing' part, the second a 'write about' part.

The first section of the coursework assignment required students to download and set up a database management system package (MySQL), and then implement a 'toy' database with it. They then had to describe this database graphically with an entity-relationship diagram, find out how to get help from experienced database users and run a number of queries on the database they had implemented.

Downloading the MySQL package did not present major problems for anyone this year – indeed one student wrote: 'I installed Ubuntu Server on the VM then installed MySQL. The way to do this from Ubuntu is pish posh.' (Although 'pish posh' means 'absurd, irrelevant, redundant, presumably it was not difficult.) However, it is important for students to understand that they need to provide more detail about downloading and installing MySQL.

Most of the entity-relationship diagrams were done well, although several students did not understand that the cardinality/participation indicators are 'look across' with respect to the entity type and the relationship (for more information, see Chen, 2002). This is a perennial problem, perhaps because it is counterintuitive.

Accessing online help was easy, as intended. However, some of the problems students chose to report on were rather simple. A very few students simply posted a web link without including commentary on the problems presented and solved there.

SQL queries were generally done correctly, but the usual problems occurred. SQL can be deceptive because it is not difficult to formulate a query that will return results, but the results may not be the intended ones. More than a few students seemed to simply accept whatever their queries brought back, without actually looking at the original relations and seeing if the results of their SQL matched the results of doing the query 'by hand'. On a large dataset, similar to the one in the second coursework assignment, this is not practical. But the advantage of the 'toy' database in this coursework is that every query can first be done without the computer – and this **should** be done.

One query, number 20, did give some students unnecessary problems. Querying for 'null' values has to use the keyword 'is', not '=', but a surprisingly large number of students thought that the problem was that everyone has a birthdate so that birthdate cannot be 'null'. And yet 'null' can be used to mean 'unknown' or 'not available'. That is, the data can exist, but we may not know what it is.

In a few cases, the SQL proposed was correct, but the answers returned were wrong because the data was entered incorrectly. This may be a real problem in a database with millions of records, but there is no excuse, when implementing a database with just a couple of dozen tuples, for not carefully checking that the data entered is the data that was supposed to be entered.

The sections of the coursework following the SQL queries explored issues of normalisation, functional dependency and choice of primary key in relation design. Although most coursework submissions were adequate in this area, there were still far too many examples of confusion about what functional dependency means and what a primary key is.

Two examples of student confusion, taken directly from submitted coursework, illustrate the importance of consulting the subject guide and/or recommended textbooks when for basic definitions:

1. 'A functional dependency describes the relation of attributes. For example the woodcraftskills of rangers.'
2. 'A determinant is any attribute in a database table that you can use to determine the other attributes of the same row. For example the birthday of a ranger can be used to find out the firstname and surname.'

Both of these definitions sound reasonable, but they are wrong.

Too many coursework assignments gave the wrong choices for primary keys. There is a set of simple practical tests that can be applied to a proposed relational schema with respect to the correct choice of primary key. Does it keep out tuples ('rows') that should be kept out, and does it permit the entry of tuples that should be let in? Does a proposed primary key allow two different sponsoring teams to do maintenance on the same park area on the same date, when one of the requirements is that this not be possible? (This will happen if 'sponsoring team' is made part of the primary key.) Does it prevent a firm from sponsoring more than one park area? (This will happen if the primary key is made up only of the sponsoring firm.)

Since understanding relational design is at the heart of understanding modern databases, students should take extra care to check their understanding of these issues. In particular, they should make use of the online forum, one of the great advantages of this course, to discuss their understanding of any issue they find confusing.

Conclusion

In summary, most students' coursework assignments showed serious effort. The errors that occurred appeared to be the result of hasty work: not checking that data was entered correctly in the first place, not checking that SQL query answers matched reality and working from memory when dealing with the key concepts of functional dependency, normalisation and primary keys. The lessons are clear: for your coursework assignments, start early, work slowly and carefully, and do not rely on 'common-sense' definitions of critical terms.

Coursework commentaries 2015–2016

CO2209 Database systems – Coursework assignment 2

General remarks

This coursework assignment was generally done well, although some students did not attempt all parts of the assignment. A number of students would have benefited from using the online forum, where help can be found not only from lecturers but also from fellow students, some of whom are very knowledgeable professionals.

One thoughtful remark from a student who completed the coursework assignments to a high standard was a note that the database systems coursework required significantly more time than the coursework assignments in other subjects. It is worth reminding students that they should expect to spend around 30 hours per coursework assignment at stages 1 and 2.

A few coursework assignments showed signs of extremely hasty preparation – mislabelled questions, answers without question numbers or missed-out questions. Perhaps these were submitted in the last hours before the midnight deadline. One suggestion to prevent this: begin your coursework assignment by putting in all the question numbers, and then go back and answer each question. This will make sure that no question is mislabelled or overlooked.

Comments on specific questions

A1. Reporting your experience setting up this database, and with databases In general

In order to give students some experience with more than tiny 'toy databases', this part of the coursework assignment required downloading a substantial database. A few students found the relevant website that hosts the database a bit elusive, but everyone managed to download it in the end. Two students using XAMPP under Linux had some problems relating to timeout errors and port assignment clashes, but managed to solve these issues themselves.

A2. Compiling a description of the tables you have downloaded

This part of the coursework assignment directed students in how to reveal the total size of the Mondial world-facts database which they had downloaded, and then asked them some questions about the relationship between total database size, on the one hand, and the cardinality and degree of its relations, on the other. Most, but not all, students understood that total database size is a function not only of the number of columns and rows in each table, but also of the datatypes stored in the tables: strings and 'blobs' can take up large amounts of storage space, even in 'small' relations. The misunderstandings revealed by this part of the coursework highlighted the importance of practical experience in implementing databases. (The same lesson was illustrated in a later part of the coursework assignment, where it was clear that some students did not realise the enormous practical difference in a large database between adding a new tuple ('row') to a relation and adding a new attribute ('column').

A3. Queries on the Mondial database

Although SQL is not as complex as a typical programming language – the person using it does not have to make any decisions about designing data structures – it can still be tricky to use for all but the simplest queries. In particular, aggregate functions (like SUM, MAX, MIN, COUNT and AVG) have to be given special attention. It is very easy to formulate a query which 'works', in that it returns a set of data which 'looks right', but which is in fact wrong. GROUP BY and HAVING warrant special caution in queries. Although there were a good number of courseworks which got all of the SQL problems correct, there were also a large number where students fell into some well-known traps.

Rather than discuss each of the dozen or so problematic queries in turn, this report makes the following observation: 90 per cent of the errors students made could have been prevented if the student had bothered to check their answers against common sense and then against the database itself, or, where that was ruled out because of the size of some of the relations, against a user-generated subset of it.

Many of the erroneous queries returned results that should have been seen to be obviously untrue: the USA is not in the European Union, Afghanistan does not have the highest GNP in the world, etc.

So, the lesson to be drawn here is: once you have formulated a query which runs, check your results against your common-sense knowledge and then against the database itself.

A4. Getting help

This was not a difficult part of the coursework and was intended mainly to make sure that students know about the rich variety of sources of online help available. Students made some good critical comments about the respective value of the video presentations and online tutorials which had to be reported on.

B1–B4. Relational design

These sections contained the great majority of outright errors. The heart of relational database design is understanding dependencies, normal forms and primary key choice. These subjects are not difficult in the abstract, but can be difficult in practice. The ideas which underlie them are not intuitive and often go counter to the way connections among data have been experienced before encountering the relational model, both linguistically and visually. So it is not surprising that there were a significant number of courseworks which exhibited errors in identifying functional dependencies, normalising relations, identifying candidate keys and choosing a primary key from among them. Even some students who reported an extensive practical background in database management ran into problems when analysing data relationships and proposing a relational solution.

Some students proposed primary keys which were not minimal – that is, they included the appropriate attributes, but also included unnecessary attributes. Others clearly did not comprehend the basic idea of a functional dependency. Since these ideas are often presented in a quasi-mathematical way – as formal definitions – it is all too easy for even conscientious students to believe that memorising formal definitions is all that is required for an understanding of the topic. But this is not the case, and student difficulties here highlight the importance of providing many worked examples and practical examples of the principles being taught.

It was interesting to see, in one case, Armstrong's Inference Rules and Heath's Theorem apparently being applied to what was actually a very simple normalisation problem – did the student actually use them, or did they think they had to quote them? In any case, this was using an elephant gun to kill a mosquito.

B5. Examination errors

Completing this section of the coursework assignment meant reading a number of past examiners' reports on the database examinations. The aim here was to alert students to typical errors and omissions made by candidates sitting the examination, in the hope that being aware of them would help improve results on the next examination.

B6. Alternatives to the relational model

To complete this part of the coursework, students had to research and write brief essays on two topics: (1) the 'CAP theorem' (which addresses a problem arising from the enormous expansion of possible multiple simultaneous inputs to geographically dispersed databases) and (2) a particular alternative to the relational approach (the 'entity-attribute-value' model). Essays submitted varied greatly in quality – more than a few showed evidence of a cut-and-paste approach which, even where properly cited and referenced, does little to provide any 'added-value' from the student. However, it is to be hoped that even those who submitted rather cursory research benefited from a look at two very important current issues in the database field.