University of London International Programmes

CO2209 Coursework assignment 2
2015–2016

Important

Your coursework assignment should be submitted as a single PDF file, using the following file-naming conventions:

FamilyName_SRN_COxxxxcw#.pdf (e.g. Zuckerberg_920000000_CO3323cw2.pdf)

- **FamilyName** is your family name (also known as last name or surname) as it appears in your student record (check your student portal)
- **SRN** is your Student Reference Number; for example, 920000000
- **COXXXX** is the course number, for example CO2209, and
- **cw#** is either cw1 (coursework 1) or cw2 (coursework 2).

It should take between 20 and 40 hours to complete, depending on how much you already know about the topics. There are some easy parts, and some which are more challenging.

Each part of the coursework assignment has been chosen to help you understand some key issues in the subject of databases. It should be undertaken with the *subject guide*, *Volume I*, at hand. There are six Appendices at the back to supplement the information in the *subject guide*.

The best way to approach this coursework assignment is to look over the whole thing first, and get an idea of what you want to concentrate on as you read the *subject guide* and other materials such as your textbook. **Part B** covers some of the fundamental ideas of database theory and most of it can be done independently of **Part A**, so you might wish to do those parts of **B** right away. If you have never encountered relational database ideas before, the terms will be unfamiliar and it will take some time for them to become part of your everyday working inventory of ideas – learning these definitions by heart might be a good strategy to start with, because this will help you gain a deeper conceptual understanding of them as you do the coursework assignment.

Background

In the first coursework assignment, we created a 'toy' database, consisting of four tables and a few dozen rows. Real databases are typically many orders of magnitude greater. In this coursework assignment, our practical task will be to download such a database, and use it to learn more about real databases.

We are going to download the 'Mondial' database, which carries information about the countries of the world. This is supposedly based on the CIA's World Fact Book.

Go to this website: http://www.dbis.informatik.uni-goettingen.de/Mondial/ .

Find the paragraph entitled 'Generating the Database under MySQL' and click on each of the following links.
http://www.dbis.informatik.uni-goettingen.de/Mondial/OtherDBMSs/mondial-schema-mysql.sql

http://www.dbis.informatik.uni-goettingen.de/Mondial/OtherDBMSs/mondial-inputs-mysql.sql

You do not download the data in the database base directly, but rather download statements in the first file which will create the tables (called mondial-schema-mysql.sql), and then statements in the second file (called mondial-inputs-mysql.sql), which will populate them.

The second file is about 1.5 mbytes in size, and consists of over 20,000 INSERT INTO statements.

These files may be downloaded and submitted to a 'front end' processor for your database, if you are using one (that is, the statements themselves will be processed).

Or, the whole files, which are displayed when you click their links, may be copied and pasted into a text editor (you will probably need to add a new file name extension to the files to do this; for example, changing mondial-inputs-mysql.sql to mondial-inputs-mysql.sql.txt) and then the statements in those files copied and pasted directly to the MySQL command line processor if you are running MySQL in command line mode. (Of course, you will first do this with the schema creation file, and then with the inputs file.)

Note: This is a large database, which may take up to half an hour to download. If you are using Linux, and if you encounter a problem with the download, come to the forum and see if your problem has been answered there.

From this same website (http://www.dbis.informatik.uni-goettingen.de/Mondial/) download the documents which display the structure of the database: there are three, all of which carry the same information but which show it in different ways:

A 'Referential Dependency' diagram:
-- [http://www.dbis.informatik.uni-goettingen.de/Mondial/mondial-abh.pdf]

An Entity/Relationship diagram:
-- [http://www.dbis.informatik.uni-goettingen.de/Mondial/mondial-ER.pdf]

A Relational Schema:
-- [http://www.dbis.informatik.uni-goettingen.de/Mondial/mondial-RS.pdf]

These documents let us see how the data in the separate tables is related; namely, which tables hold data relating to the same things. Note that the Relational Schema shown here is only a broad outline schema. It does not show datatypes for the attributes; or which attributes are Primary or Foreign Keys; or other constraints.

The Entity/Relationship diagram is a very simple one, which omits cardinality and participation constraints. Do not worry too much about these documents until you think you have understood the concepts of data dependency, functional dependency, keys and normalisation. They will be useful for constructing SQL statements to answer queries which require you to know which tables are linked to which other tables.

Important note about the Mondial database: the value of this database is that it is not a toy one. However, it is definitely out-of-date, and was inaccurate even when first put up on the internet. (Remember: all large data sets must be assumed to be 'dirty'.)

Additionally, its designers made at least one poor choice, in the examiner's opinion: they have field names (attributes, or columns) which are the same as relation names. So there is a relation called 'Country', and in some of the other relations, there is an attribute called 'Country'. The 'Country' field of these other relations is a Foreign Key for Country.Code. It would have been a better idea to label these attributes 'Country-Code'. We will look at a way to improve this.

Also: note that the Mondial database does **not** enforce Foreign Key integrity. (That is, it would be possible to have a 'Country' field in a relation which has no matching 'Code' field in the relation Country.)

Coursework assignment 2

NOTE: If you have questions about any of these coursework assignments, or need help in completing them, please use the discussion forum.

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

A1(a) Write a short report briefly describing your past experience with database management systems, including any courses you have taken relevant to this subject; and any practical experience (paid or otherwise) you have had with databases. This need not be longer than a few sentences.

A1(b) Write a short report, after you have set up and populated your database, describing this experience. This may be just a few sentences if all went perfectly smoothly, or it may be up to two pages if there were problems you needed to solve.

[3 hours (for downloading and creating, plus writing), 5 marks]

Tip: in using MySQL, if your operating system is Windows, you may want to make the DOS Command Prompt window larger than the default value. Go here to see how to do this: http://superuser.com/questions/401621/how-can-i-widen-the-windows-7-command-prompt-window

(Please note: You may have to reboot after doing this for the new window size to take effect.)

A2. Compiling a description of the tables you have downloaded

This compilation will provide basic information about the database tables that you have set up. If you ever get a job as a Database administrator, or have to work with a database that you yourself did not create, you will have to do something like this first. Combined with the Referential Dependency diagram, E/R diagram, and Relational Schemas you have already downloaded, you would have the materials you need to start to understand your database's structure.

You do not actually have to write anything here, but rather, just copy in the results of running some commands:

To do this, you will need the SQL commands
SHOW TABLES; and
DESCRIBE <tablename>; and
SELECT COUNT(*) FROM <tablename>.

Note that for '<tablename>' you will need to substitute the names of each of the tables listed by SHOW TABLES. If you are working with MySQL directly from the DOS prompt, use a word processor or a simple text editor to make this a very quick operation. Note that to paste into the Command Prompt, you need to right-click – CTRL-V does not work!

It will be useful to be able to output what you see on the screen to an output file; you can use the command 'tee' to do this, as follows (user input in bold):

mysql> **TEE** D: OutputLog.txt – whatever shows on the screen is also copied to the file OutputLog.txt which the examiner has placed on their D: disc in this example, but which can be located anywhere you like.

mysql> **SHOW TABLES;** – information about the tables will be sent to OutputLog.txt as well as being shown on the screen;
mysql> **DESCRIBE BORDERS**;
mysql>**SELECT COUNT(*) FROM BORDERS;**
mysql> **SHOW INDEX FROM BORDERS;**
- and so on ... for the first **five** relations in this database (COUNTRY to DESERT);

mysql> **notee;** – turns it off;

A2(a) Include the requested information for the first five tables, plus the size of each of these tables, which you can find out by executing the commands below.

To see the size in megabytes of each of your tables, do this:

```
SELECT
    table_schema as `Database`,
    table_name AS `Table`,
    round(((data_length + index_length) / 1024 / 1024), 2) `Size in MB`
FROM information_schema.TABLES
WHERE table_schema = 'mondial'
ORDER BY (data_length + index_length) DESC ;
```

A2(b) 1. What is the total size of the Mondial database?
2.What are the two largest relations in the Mondial database in terms of total bytes?
3. For any two relations (in any database), is it the case that the relation with the largest cardinality must be the largest in terms of total bytes of data?
4. For any two relations, is it the case that the relation with the largest degree (number of columns) must be the largest in terms of total bytes of data?
5. If, given two relations, one is larger than the other in terms of both cardinality and degree, is it necessarily larger than the second one, in terms of total bytes of data?

Please start this answer on a new page.

[2 hours,5 marks]

A3. Queries on the Mondial database

A note about SQL: SQL's tables do not conform completely to the definition of relations. In particular, the tables which result from a query can have duplicate tuples (rows), which in most cases is not what we want. To avoid this, always use the **DISTINCT** keyword, as in **SELECT DISTINCT.**

A3(a) What is the query that will list names of the countries which (according to the database) are members of the EU? (Use the table Ismember.)

A3(b) What is the query that will list the names of the countries which are members of both the EU and NATO?

A3(c) What is the query that will list the names of the countries which are members of the EU but not of NATO? (Hint: do not use <>; use NOT IN. MySQL does not have specific operators for INTERSECT or MINUS, but these operations can be done using IN or EXISTS.)

A3(d) What is the query that will list the total population for all countries (all countries' populations added together)?

A3(e) What is the query that will list the name of the country with the highest population density? (Note: population density =population/area)

A3(f) What is the query that will list the world's languages, and for each one, the total number of countries where each is represented? (Hint: use **GROUP BY**.)

A3(g) What is the query that will list the world's languages which are present in at least five countries, and for each one, the total number of countries where it is represented? (Hint: use **GROUP BY** and **HAVING.**)

[10 hours, A3(a) 2 marks; A3(b)–A3(g) 3 marks each. Total: 20 marks]

A4 Getting help

Most database access does not consist of *ad hoc* queries typed in via the command line or a user-friendly front end, but from computer programs.

Assume you wanted to access a MySQL database from a Python program. Find and give the URLs of three websites which host tutorials on how to do this, plus the URLs of two YouTube videos which explain how to access a MySQL database from a Python program. You should read the tutorials and watch the videos.

Write a brief (no more than half a page) commentary on which two of these you think would be most useful to you, justifying your answer.

Please start this answer on a new page.

[1–2 hours, 5 marks]

B1 Determining Candidate keys, and choosing one as the Primary key

Being able to recognise which attributes or combinations of attributes make up a Candidate key – something that uniquely identifies each tuple, that is, is necessarily different for each tuple – is one of the skills that you need to learn in this course. It is not difficult, but may take some practice. Some students get the wrong idea that the Primary key is the 'most important' attribute or combination of attributes; namely, 'what the table is giving information on'. This is not necessarily so.

A Candidate key uniquely identifies a row (tuple). You can check your choice of Candidate keys by asking: 'Could any two rows have the same values for their Candidate keys, as I have chosen them?' If so, you have made the wrong choice.

A Candidate key is a 'possible key'. If there can be more than one possible Key for a table, then we have to choose one as the Primary key. Note that a Primary key is also a Candidate key; although it is common to hear the phrase 'Candidate key' used to refer only to those Candidate keys that were not chosen to be the Primary key.

Designating a Primary key protects a table from one kind of integrity violation. Consider the database that holds your University of London student registration number. There will be one table in which the attribute with this number is the Primary key, with other attributes holding your name and other information about you. If Primary key integrity (also called 'Entity integrity') was not enforced, then there could be two different people with the same registration number. (Another way of putting it is this: the Primary key designation automatically enforces the UNIQUE constraint. But whereas you can designate more than one attribute (column) as UNIQUE, you can have only one Primary key.)

You may wish to review the *subject guide, Volume 1*, pp.49–52, before starting this part of the coursework assignment.

The relations that follow describe boats, boatyards (the unique 'home port' for boats, where each one is registered); and who has rented each boat at what time.

REGISTRATION in **BOATS** uniquely identifies each individual boat, as does NAME for **BOATYARDS.** However, the names in NAME for **BOATS** are not necessarily unique, since two boats may have the same name. A boat will have only one boatyard at a time listed as its 'home port', although it can transfer from one to the other.

BOATS

REGISTRATION	NAME	OWNER	TYPE	HOMEPORT	LENGTH
CU3977364	El Tigre	Rafael Fernandez	Diesel engine	Havana Bay	40
US9709763	Crescent Moon	Bill Sykes	Sail	Havana Bay	35
US9976445	Barracuda II	Rafael Fernandez	Sail	Miami	40
UK077733	Golden Hind	Francis Drake	Sail	Miami	90
UK097766	Victory	Horatio Nelson	Sail	Key West	100
DE9737373	Scharnhorst	Rudy Dutschke	Diesel	Miami	35
MEX9884441	El TIgre	Juanita Perez	Sail	Cancun	60

BOATYARDS

NAME	MANAGER
Havana Bay	Fidel C. Ruz
Miami	Oswaldo Bosch
Cancun	Pablo Escobar
Key West	NULL

RENTALS

BOAT	HIRED-TO	DATE-OF-DEPARTURE	DATE-OF-RETURN
CU3977364	Hector Munoz	2015-06-24	2015-07-24
US9709763	Bob Hope	2015-07-11	2015-09-25
CU3977364	Pepe Diaz	2015-08-12	2015-09-02
UK077733	Shala Subbodi	2015-10-01	NULL
CU3977364	Hector Munoz	2015-10-13	NULL
DE9737373	Piotr Androff	2015-02-24	2015-03-20

B1(a) What are the Candidate keys of each of these relations? (Note: the Primary key is also a Candidate key.)

B1(b) A boatyard can have at most one manager. So why wouldn't MANAGER be a good choice as the Primary key for the relation **BOATYARDS**?

B1(c) What would be the best choice of Primary key for the relation **BOATS** (assume that we cannot change the structure of a table (to add or modify columns)? Justify your choice. (Hint: what are the desirable qualities for a Primary key, besides being able to uniquely identify a tuple?)

B1(d) Why is REGISTRATION (alone) not a Candidate Key in **RENTALS**?

B1(e) Which attribute in **RENTALS** should be made a 'Foreign key', and which attribute in which other relation should it reference?

A frequent query run on this database is to find the HOMEPORT of a boat with a given registration number. An example would be **SELECT** HOMEPORT **FROM** BOATS **WHERE** REGISTRATION ='US9709763';

Suppose we redesigned this database to include the home port boatyard of a boat in the **BOATYARDS** relation, instead of in the **BOATS** relation. (In other words, instead of showing, for each boat, its homeport boatyard, we would show, for each boatyard, the boats that made it their homeport. The homeport-boat link would just be transferred to another relation.

There are two ways we could do this.

BOATYARDS-A (Note that we would have to make the ALLBOATS column of type string.)

NAME	MANAGER	ALLBOATS
Havana Bay	Fidel C. Ruz	'CU3977364, US9709763'
Miami	Oswaldo Bosch	'US9709763, UK077733, DE9737373'
Cancun	Pablo Escobar	'MEX9884441'
Key West	NULL	'UK097766'

NOTE: The examiner has populated these alternate relations with only the data shown in the original version of the relation.

BOATYARDS-B

NAME	MANAGER	BOAT-1	BOAT-2	BOAT-3	BOAT-4	BOAT-5	BOAT-6
Havana Bay	Fidel C. Ruz	CU3977364	US9709763	NULL	NULL	NULL	NULL
Miami	Oswaldo Bosch	US9709763	UK077733	DE9737373	NULL	NULL	NULL
Cancun	Pablo Escobar	MEX9884441	NULL	NULL	NULL	NULL	NULL

Key West	NULL	UK097766	NULL	NULL	NULL	NULL	NULL

Now, look at the relations **BOATS, BOATYARDS-A** and **BOATYARDS-B**, and write SQL **queries** for all three to answer the question: 'Find the NAME of the Boatyard where the boat US7305736 is based.'

For this exercise, assume that **BOATYARDS-B** can hold information on a maximum of 10 boats (that is, the last column is named BOAT-10).

For querying **BOATYARDS-A,** see here: (http://stackoverflow.com/questions/4122193/how-to-search-for-rows-containing-a-substring/)

B1(f) BOATS :

B1(g) BOATYARDS-A :

B1(h) BOATYARDS-B :

B1(i): Which of the three designs is easiest to query?

B1(j): Suppose we want to extend the capacity of **BOATYARDS-B** to hold information on more than 10 boats. What will we need to do?

[3 hours, 10 marks]

B2: Normalisation

The basic idea of database normalisation (US 'normalization') is that a single 'fact' – a dependency among data items – should be stored in one place only, and independently of other facts. (Note also that the word 'normalise' has many completely different meanings. Do not confuse database normalisation with other uses.)

When we design a set of tables, we sometimes find that we have stored the same fact in more than one place. (For instance, a customer's address may be stored both in a table dealing with billing, and also in a table dealing with marketing. If the customer changes his address, we can end up with inconsistent data.)

Sometimes we find that it is necessarily to violate the rules of normalisation, for performance purposes. That is, we deliberately store the same data in more than one place, in order to avoid the performance hit of doing a JOIN to retrieve the data. However, the initial design of a database should be according to the principles of normalisation, so that any violation of these principles is a conscious one of which we are aware.

We begin the process of normalisation by identifying the Functional (one-valued from one data item to another) dependencies among the data; and also noting any dependencies that are multi-valued in both directions. Remember that we are only interested in dependencies that we cannot deduce from other dependencies and that do not depend on other data items for their existence – the 'direct' dependencies. In the table COURSES below, for example, EXAM-DATE and LECTURER have a relationship, since values for them appear in the same tuple, but it is not one we would record independently.

Having identified the Functional dependencies and the both-way multi-valued dependencies, we then try to arrange the data so that every determinant (the attribute or combination of attributes on the left-hand side of a Functional dependency diagram) is a Candidate key of a table. If there are mutual multi-valued dependencies, each of them goes into its own table. There can arise situations in which further work may be required, or in which we have to choose between two less-than-optimal arrangements, but they are rare.

Please start this coursework assignment answer on a new page.

B2 Coursework assignment

Consider the following relation, which contains information on Subjects taught at a university. For each Subject, there are one or more tutorial sessions, overseen by a Tutorial Assistant, which take place in a Tutorial Room. Every room can seat a maximum number of people, and this information is recorded in Room-Size. Each subject has a single Exam Date, and one Lecturer in overall charge of the subject. A Lecturer may be in charge of more than one Subject. This lecturer has an Office, whose location is recorded.

Thus we can see from the table below that the subject Physics 101 has three tutorial sessions; one of which is led by J. Chan; and takes place in the Newton building in Room 34; which can seat 45 people. This subject will be examined on 2016-06-02, and lecturer in charge of it is B. Rosen, whose office is in room 233 of the Feynman building.

COURSES

PRIMARY KEY: SUBJECT + TUTORIAL-ASSISTANT

SUBJECT	TUTORIAL-ASSISTANT	TUTORIAL ROOM	ROOM-SIZE	EXAM-DATE	LECTURER	OFFICE
Physics 101	J. Chan	Newton 34	45	2016-06-02	B. Rosen	Feynman 233
Physics 101	B. Horowitz	Newton 32	40	2016-06-02	B Rosen	Feynman 233
Physics 101	M. Jalazi	Newton 35	30	2016-06-02	B. Rosen	Feynman 233
Chemistry 102	C. Daniels	Pauling 73	35	2016-06-12	M. Hafiz	Lavoisier 04
Chemistry 102	F. Kaur	Pauling 54	40	2016-06-12	M. Hafiz	Lavoisier 04
Physics 102	J. Chan	Newton 43	30	2016-05-29	L. Liu	Feynman 248
Physics 102	M. Harrison	Newton 35	30	2016-05-29	L. Liu	Feynman 248
Biochemistry 1	F. Kaur	Pauling 34	45	2016-06-10	B. Martin	Salk 237

B2(a). Identify the Functional Dependencies in this relation. One has been done for you.

SUBJECT → EXAM-DATE A subject has just one examination date.

B2(b). Change the schema so that the data in this table is recast into several tables, all in Boyce-Codd Normal Form, specifying the Primary keys of each new table, and showing the extension of the resulting relations. (In Boyce-Codd Normal Form, all determinants are Candidate keys, usually Primary keys.) This will involve splitting the original table into more than one table. Do not invent new attributes! **Please start this answer on a new page.**

A note on Normal Forms: are there 'higher' Normal Forms beyond Boyce-Codd? Yes, but all that this means is that occasionally, you can run across relations which are in BCNF, but which need to be split further, because of some particular (and unusual) feature they have. These are rare. In almost all cases, if a relation is in BCNF (which is just a refinement of Third Normal Form), it will also be in Fourth, Fifth, etc. Normal Form also.

[2 hours, 10 marks]

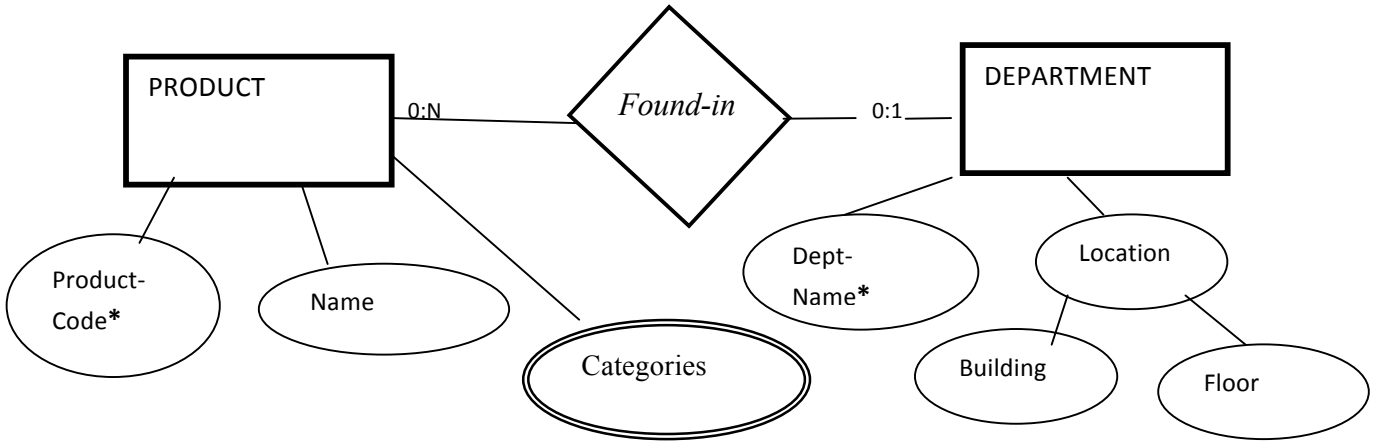
B3 E/R diagrams and designing a database

Background

E/R diagrams are still a frequently-encountered way of giving a broad visual overview of a database. Some designers start their design by drawing one. For example, the Mondial database has a simplified E/R diagram. You need to know how to read an E/R diagram, even if you do not choose to begin your database design with them.

The E/R diagram below shows the 'found-in' relationship between products and departments in a large shopping complex. Every Product has a unique Product-Code, a Name, and is tagged with one or more Categories (such as 'for pets', 'needs batteries', 'flammable'). Departments have a unique Dept-Name, and are located on a specific floor of a specific building. A Department may stock many Products, but a given Product will be stocked in just one Department.

Note that in this diagram, unique entity-instance identifiers have been marked with an asterisk.



B3(a) From E/R diagram to relations.

Design a set of relations which could hold the information shown in your Entity/Relationship diagram above. Do **not** add any attributes which are **not** mentioned in the diagram above, but **do** include any attributes which **are** mentioned. Each relation should have a name, and the columns (fields, attributes) making up the primary key should be indicated. Use the example in **Appendix II** as a model. Also look at **Appendix III** for further information on how to generate a relational schema, or to check one that you have already created. Note: **you do not** have to specify the data types in your schema; nor do you have to implement it using MySQL. Please start this answer on a new page.

[2 hours, 5 marks]

B3(b) Finding the Data dependencies

Identify all of the Functional dependencies among the attributes of your relations. Also note any multi-valued dependencies among direct dependencies. Use the example in **Appendix III** as a model. Please start these answers on a new page.

[2 hours, 5 marks]

B4 Relational design

A company wants to design a database to hold information on its employees. An employee can be either male or female, and can be full-time, or part-time. It identifies employees by Employee-Number, and wants to record the Department they work in. It wants to be able to run queries of the following type: List the employee-numbers of all part-time female employees who work in the Shoe department.

Someone has proposed the following schema:

MALE-FULL-TIME

EMPLOYEE-NUMBER	DEPARTMENT

MALE-PART-TIME

EMPLOYEE-NUMBER	DEPARTMENT

FEMALE-FULL-TIME

EMPLOYEE-NUMBER	DEPARTMENT

FEMALE-PART-TIME

EMPLOYEE-NUMBER	DEPARTMENT

This schema requires four relations. Propose an alternate schema which can hold the same information in just one relation.

[1 hour, 5 marks]

B5 Examination errors

Download and read the *Examiners' reports* for the 2014 and 2015 **CO2209 Database systems** examinations, both Zone A and Zone B, and the examination papers for both zones. Write a brief (half page, maximum one page) summary of the topics many students seemed to have difficulty with, as identified in these reports. Consider both examinations and both reports for both years. Be sure to note what topics students found difficult in both years.

[2 hours, 10 marks]

B6 Alternatives to the Relational model

The relational model has been an overwhelming success in the database world, because it is both conceptually simple, mathematically sound, and can model so many situations, even ones that at first glance do not seem to fit into the 'square data' relational model.

One of those situations is where the data output of an object-oriented language must be fitted into the format of relations. At the time that the subject guide was written, there was a good deal of interest and research in 'object-oriented' databases, and 'object-relational' databases. It appeared then that the relational model might be eclipsed by developments in this field. (See *subject guide Volume 2*, pp. 47–51.) That did not happen.

However, the problems did not go away. And new ones appeared. Within the last few years, with the enormous growth of the Web and online processing of documents with very heterogeneous structures, a new family of databases has emerged, called, loosely, 'NoSQL' database systems. We have also seen the development of the 'self-describing' XML mark-up language (and alternatives, like JSON and YAML) for inter-system data transfer.

First watch this video**: https://www.youtube.com/watch?v=qI_g07C_Q5I**

Then research and write brief (no more than two pages) essays giving an overview of the following.

B6(a) the 'CAP theorem'

B6(b) the 'entity-attribute-value' model (start here: https://en.wikipedia.org/wiki/Entity-attribute-value_model);

Note that YouTube may have useful videos on these subjects.

[Video: 1 hour, essays 2 x 2 hours, 20 marks]
[TOTAL 100 marks]

[END OF COURSEWORK ASSIGNMENT 2]

APPENDIX I: E/R Modelling conventions

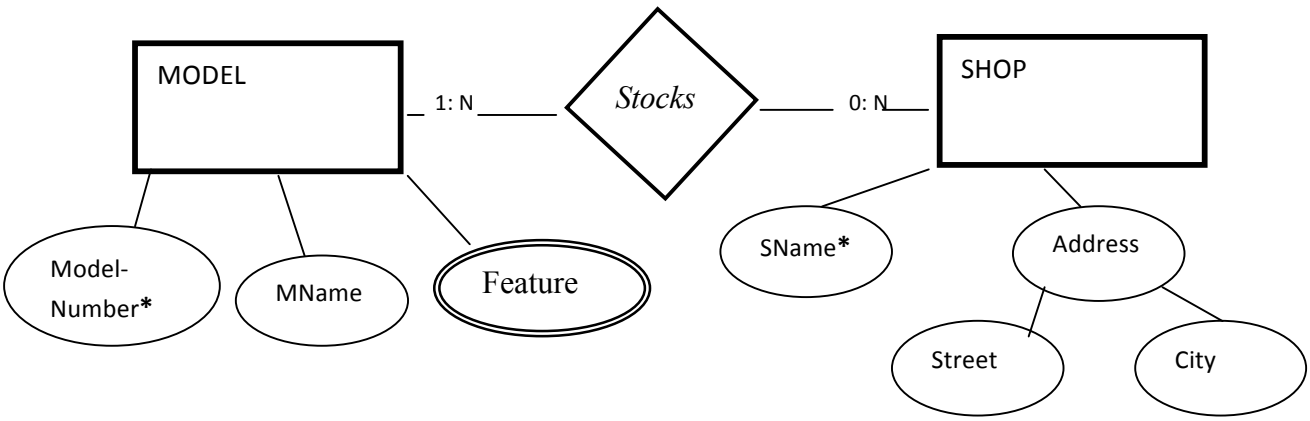
Use the following example as a model for your own E/R diagrams. (There are many variants of Entity/Relationship diagrams. This is one of the most common.) There are additional refinements to E/R diagrams with which you should be familiar, discussed in the subject guide. (In particular, make sure you know how entity sub-types and how 'weak entities' can be represented.)

This illustrates part of a database dealing with shops that sell model airplanes.

For each model, there is a unique Model-Number, a name, and one or more 'features', such as 'not for children under 3', 'requires assembly', 'takes batteries', etc. For each shop, there is a unique name, and an address, which consists of a street, such as '34 Lyons Lane', and a City, such as 'Leeds'. We want to record these attributes, plus the *relationship* 'stocks' between shops and models.

A shop will not exist in the database unless it stocks at least one model, but a model may exist in the database even if no shop stocks it. In this example, these constraints are shown by 'minimum: maximum' numbers, where 'N' means there is no upper limit (the usual case). (Existence and cardinality constraints are sometimes indicated by 'crow's feet' and single versus double lines.)

Note that the 1: N or 0: N indicators are on the **opposite** side of the 'relationship' diamond to the entity types they are describing. This is sometimes called 'look-across' notation. If we wanted to show information about a relationship itself – for example, 'date-first-stocked' – we could either add an attribute-bubble to the relationship diamond, or treat 'stocked' as an entity type in itself (since we are going to make it into a table anyway). Both conventions are common.



Note: attribute names and entity-type names are normally given in the singular. Thus 'MODEL' not 'MODELS'.

Appendix II: Generating a Relational schema from an E/R diagram

To generate a relational schema from the diagram in **Appendix I**, we first create at least one relation for each entity type, whose Primary key will be the entity-identifier attribute. (In this example, such attributes have an asterisk beside their names.) All attributes (including composite attributes, such as the 'Address' attribute for shops), which have only one value per entity instance, can go into this relation.

An attribute which can have multiple instances, such as the 'Feature' attribute for Models, must have a separate relation, whose attributes will be the entity-identifier and the appropriate attribute value, both of which will make up a composite Primary key.

Finally, there must a separate relation for all relationships which are N: N, as this one is above. The key of this relation will be the entity identifiers of the two participating entity types. (1: N relationships are rare, but when they occur, they can be implemented in the relation of the 'N' entity. For example, if a model could only be stocked in one shop, we could record that shop's name along with the model name in the model relation.)

So, we would have the following relations.

MODEL

Attributes: MODEL-NUMBER, MNAME
Primary Key: MODEL-NUMBER

MODEL-FEATURES

Attributes: MODEL-NUMBER, FEATURE
Primary key: MODEL-NUMBER + FEATURE

Note: an alternative – ***bad*** -- design here would be:

Attributes: MODEL-NUMBER, FEATURE-1, FEATURE-2, FEATURE-3, FEATURE-4…
Primary-Key: Model-Number

This is a very poor design for several reasons (although it is one which novices often choose):

1. We do not know how many features any given model will have. If a new model has more features than we have feature attributes, we will have to restructure the whole table by adding a new column. In the original design this is never necessary. A model could have fifty features, and we would just add fifty rows to the Features table.

2. Consider querying this table. In the original design, a typical SQL statement to find all models requiring assembling would be:

   SELECT Model-Number **FROM** MODEL-FEATURES WHERE Feature = ' requires assembly';

In the alternative design, we would have to write:

SELECT MODEL-NUMBER FROM **MODEL-FEATURES WHERE** Feature-1 =‘ requires assembly’ **OR** Feature-2 = ‘ requires assembly’ **OR** Feature-3 = ‘ requires assembly’ **OR** Feature-4 = ‘ requires assembly’;

An even worse design would be to create a single ‘Feature’ attribute, but holding all features as one giant string, like this:

Attributes: MODEL-NUMBER, FEATURE

Primary Key MODEL-NUMBER

Not only would this make a very ugly print-out, but it would require us to use an inefficient SQL feature (LIKE %substring%) to form the query above. And it will not work at all if the attribute in question is numeric or of type date.

SHOP

Attributes: SNAME, STREET, CITY
Primary key: SNAME

NOTE: ‘Address’ is ‘flattened’ here. The relational model has no provision for recognising composite attributes. That is, you could not have a relation header that looked like this:

	ADDRESS		
SNAME	STREET	CITY	

MODEL-IN-SHOP

Attributes: MODEL-NUMBER, SNAME

Primary key: MODEL-NUMBER + SNAME

Appendix III: Functional dependency diagrams

An alternative approach to creating a relational schema from a description of data, is to identify all the functional dependencies among the data items. This avoids the sometimes confusing issue of 'what is an entity type', but needs to be supplemented, if used, by identifying any two-way multi-valued dependencies that are not themselves determinants.

In the E/R model in Appendix II, the Functional dependencies are:

Model-Number → MName For a given Model-Number, there is just one MName.

SName →Street For a given SName, there is just one Street.
SName →City For a given SName, there is just one City.

Where determinants are the same, we can combine FDs into one relation.

The two-way multi-valued dependencies are:

Model-Number and Feature multi-determine each other: for a given Model-Number there can be many features, and for a given Feature there can be many Model-Numbers.

SName and Model-Number multi-determine each other: for a given SName there can be many Model-Numbers, and for a given Model-Number there can be many SNames.

Sometimes multi-valued dependencies are shown this way, using a double-headed arrow:

Model-Number ↠Feature

Feature ↠Model-Number

SName ↠Model-Number

Model-Number ↠SName

So Model-Number and SName must be candidate keys, because they are determinants, each in its own relation (and we will choose them as Primary keys), and Model-Number + Feature, and SName + Model-Number will be composite candidate keys, each pair in its own relations (and we will choose the composites as Primary keys). Where we have two-way multi-valued dependencies, we will need a new, separate relation. (Trying to put multi-valued dependencies into a 'master relation' -- which is made up of a Primary key and a number of attributes functionally-dependent on that key -- is probably the most common mistake of novices.)

We still end up with the same four relations as we did working directly from the E/R diagram. Identifying the Functional dependencies is a good way to check on your relational schema if you do start from an E/R diagram.

E/R diagrams and Functional dependency diagrams have no particular value in themselves: they are just tools to help us design and document databases.

All of this may seem overly formal. But without going through a few exercises like this, novices often create very poor database designs. After a while, if you work with databases, seeing the functional and multi-valued dependencies among data items, and being able to choose valid Primary keys, will become an instinct, and you will design normalised tables without even thinking about it. Good luck!

-- End of Appendices --

[END OF COURSEWORK ASSIGNMENT 2 (including Appendices)]