Computing and Web Development with Foundation

**Programme:** Computing & Web Development

**Module:** CPU5002

**Module Title:** Database Theory and Practice

**Assignment Title:** Database Case Study

**Student Identification Number:** 1401501

**Tutor Name: Andrew Parker**

**Date of Submission: Sunday 11th December 2016**

Contents Page

[1. Abstract 4](#_Toc469243101)

[2. Introduction 5](#_Toc469243102)

[3. Database Design 5](#_Toc469243103)

[3.1 Planning 5](#_Toc469243104)

[3.2 Normalisation 6](#_Toc469243105)

[3.3 First Normal Form (1NF) 6](#_Toc469243106)

[3.4 Second Normal Form (2NF) 7](#_Toc469243107)

[3.5 Third Normal Form (3NF) 8](#_Toc469243108)

[4. Entity Relationship Diagram 8](#_Toc469243109)

[5. Database Creation 8](#_Toc469243110)

[6. Database Dictionary 12](#_Toc469243111)

[7. Database Test Queries 12](#_Toc469243112)

[8. C.R.U.D Operations 16](#_Toc469243113)

[Lecture Table 16](#_Toc469243114)

[Course Table 16](#_Toc469243115)

[Tutor Table 17](#_Toc469243116)

[Lecture Table cont 19](#_Toc469243117)

[9. Conclusion 21](#_Toc469243118)

[10. Appendices 22](#_Toc469243119)

[Appendix One 22](#_Toc469243120)

[Appendix Two 23](#_Toc469243121)

[Appendix Three 24](#_Toc469243122)

[Appendix Four 24](#_Toc469243123)

[Appendix Five 25](#_Toc469243124)

[Appendix Six 25](#_Toc469243125)

[11. Bibliography 26](#_Toc469243126)

# Abstract

The following report covers the process and techniques used to create a modern relational database from a single data file. Design and data models are discussed with a description of the normalisation process from unnormalised to third normalised form. Test data is interrogated to perform specific scenarios ensuring the database is correctly designed with good performance and efficiently. Examples are provided giving a visual aid to the techniques used, with a conclusion demonstrating the importance of the process.

# Introduction

The University department holds a spreadsheet database containing timetable information for staff members. This shows who teaches at what time, which module and the number of hours they work. This database is an example of a flat file database system which is inefficient holding a number of duplicated tuples. A replacement database is required to aid production and efficiency within the department.

The database shown in appendix 1 is an example of the flat file database system used by the department. Although the database appears to be well presented, it offers little database structure, relational management, indexing, data control, integrity or security.

The following case study report details the various steps and stages which redesigns the flat file database into a useable and efficient relational database system.

# Database Design

Designing a database system involves a variety of methods and procedures where each stage builds on the previous to create a final product. To design an effective and efficient relational database good decision making techniques are required. With database design being unique to the application, a good understanding of the client requirements is of the up most importance. In this case, the requirement and purpose of the database is to provide efficiency and data integrity, which surpasses the previous system.

## Planning

Planning is an important factor when designing a database. Initial steps in the planning process is to review the current system identifying any problematic areas or limitations which will need improvement. At this initial stage, it is important to gather and review as much information related to the old system as possible, especially key areas which staff and management have issues with. (Microsoft, 2016)

During the review of the system, it is essential to identify the key objects or entities which will be managed by the database, these objects can be recorded in such a way that they represent the database system from a visual point of view. Using several different database modelling tools or simply drawn onto paper, this reference acts as a modelling tool which is used throughout the implementation of the database system. (Microsoft, 2016)

Using the database model, created from the previous step, will highlight the type of information which is to be stored for each object. This information will often become the columns within the table of that object. Identifying the relationships or associations between data is an essential part of designing process. A good design plan will often result in a solid representation of data, data relationships and constraints on data which is relevant to the client. (Connolly & Begg, 2014).

*“The logical design of the database, including the tables and the relationships between them, is the core of an optimized relational database. A good logical database design can lay the foundation for optimal database and application performance. A poor logical database design can hinder the performance of the whole system.”* (Microsoft, 2016)

## Normalisation

Normalisation is another database design technique used to further examine the functional dependencies between attributes. Using several tests, data is grouped into close relationships to minimize duplication, redundancy and improve data integrity.

“Normalization is a formal technique for analysing relations based on their primary key (or candidate key) and functional dependencies.” (Connolly & Begg, 2014)

The database shown in appendix one is classed as unnormalised. Data in unnormalised or 0NF form is subject to data duplication, insert, delete and amendment anomalies with slow performance and large file space. This database can be improved using the normalisation technique.

## First Normal Form (1NF)

The first step to this technique is called First Normal Form (1NF). At this stage a discrepancy was found within the data as shown in figure 1.



Figure - Data discrepancy

The routing fundamentals hours were different in each occurrence, this was an example of an insert anomaly and was corrected ensuring both values were identical.

Reaching 1NF is a fundamentally important part of the database design structure, if this stage is not correct the database could suffer from anomalies. Initially all data from the planning stage is entered into a table format where each object or entity has its related attributes contained within. An example of this is shown in appendix 2 where data has been taken from appendix 1.

For the database to be in 1NF several rules need to be met:

1. There should be no repeated data
2. Data is broken down into an atomic state
3. Each row is unique with a single value at the intersection of each row and column.
4. Each field is given a unique name.

Referring to appendix 2 we can see the database designer has broken the 0NF data into two tables. The first entity has been named Course containing the attributes Module Code, Module Title and Hours these attributes are directly related to the entity Course with a key attribute being the module code. In the second table the entity has been named Tutor with the attributes firstname, lastname, Room Occurrence and Year. Module Code, which has been identified as a key attribute, is used from the Course entity to define the relationship between the two tables. This method of separation reduces redundant data and progresses the normalisation process further than performing the flattening process. All data is now fully compliant to the rules of 1st Normal Form.

## Second Normal Form (2NF)

Second Normal Form or 2NF is based on attributes having full functional dependency. To reach 2NF It is said that “*A relation that is in first normal form and every non-primary-key attribute is fully functionally dependant on the primary key”* (Connolly & Begg, 2014)

An example of this is shown in appendix 3. The entity course is automatically in 2NF as the attributes are fully functionally dependant on the module code which is a unique single attribute. Module title is functionally dependant on the module code, therefore, if there is no module code it can be said there would be no hours or year. The Tutor entity has been given a unique single identification of staffId which meets the 2NF rule of every attribute being fully functionally dependant on a unique key.



A third entity was formed named Lecture, this contains data directly related to the lecture information. Using the Module code as a unique key attribute again meets the 2NF rule. An example why this has been done is to say that a room cannot exist without the module code, the same applies for the attribute occurrence. Further amendments can be made but at this stage the rules for 2NF are fully met.

## Third Normal Form (3NF)

Third Normal Form is *“A relation that is in first and second normal for and in which no non-primary -key attribute is transitively dependant on the primary key.”* (Connolly & Begg, 2014)

Reaching 3NF ensures there are no partial or transitive dependencies within the entity ensuring that any update, delete or insert anomalies are removed. To complete this stage of normalisation each entity is given a unique primary key which can be hidden from the end user, which improves security. Foreign keys have been specified and used to create the final relationships between each entity which is shown in appendix four and the following relationship diagram.

# Entity Relationship Diagram

An entity relationship diagram graphically represents the association between entities and attributes. Using the Unified Modelling Language (UML) relating attributes can be listed below the entity name giving an easy to understand view of the database association. Included in the UML diagram is the multiplicity of the association which is described both ways using a connecting line and a verb describing the representation. An example of an ER Diagram is shown in appendix 5. Using this diagram the database designer has shown each part of the related data sets and presented them in an easy to understand relationship which is common practice within the database design. (Jewett, n.d.)



# Database Creation

Upon completion of the data normalisation the database can be created. There are several ways this can be achieved through a few variety of database creation tools such as Microsoft Access, MySQL, NoSQL, MongoDB or even Microsoft Excel.

The specification of this database from the database design model states that this database is to be a relational and for the creation of this the database the designer has chosen to use MariaDB which is a modern enterprise level open source database application. (MariaDB, n.d.)

The creation of the database will be through a command line interface (CLI), this is being chosen to further develop the skills and knowledge of the SQL language which will be used to develop the database figure 2 shows the CLI.

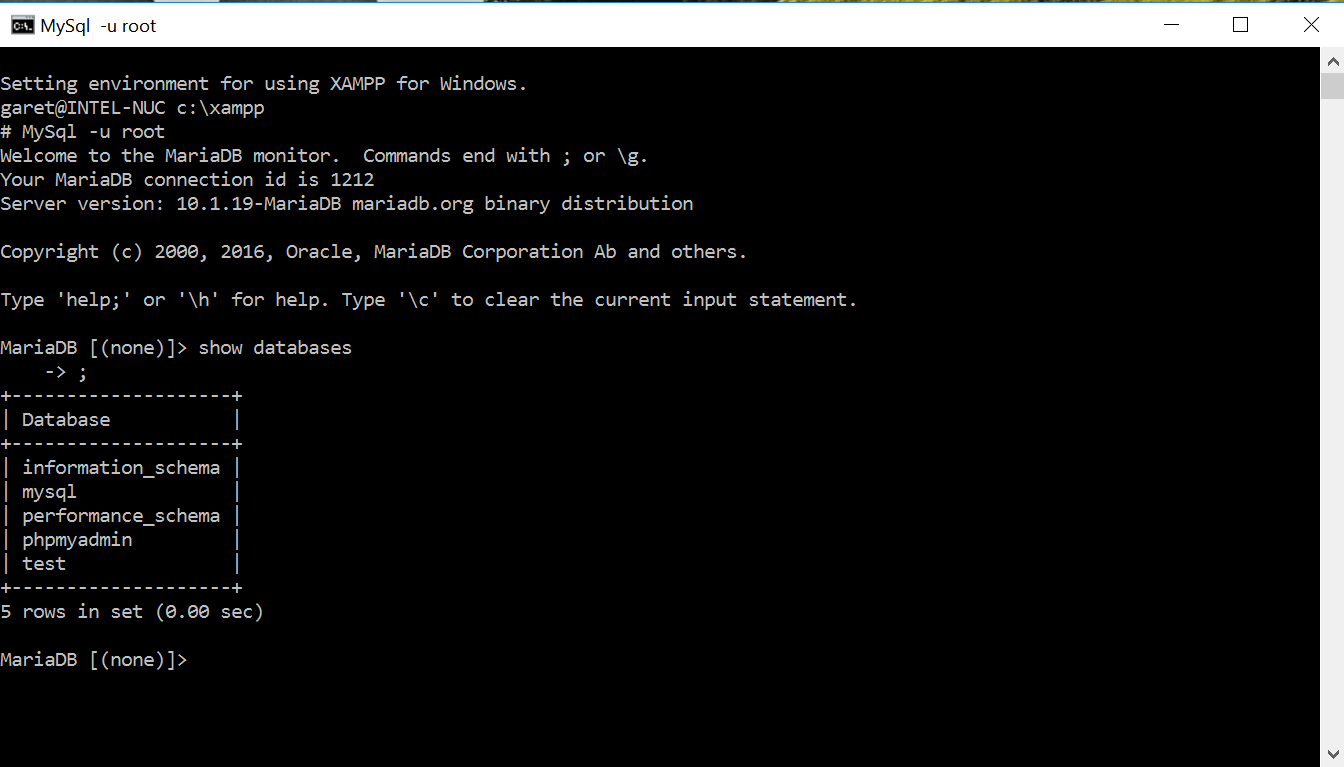


Figure - MariaDB CLI

Creating the database is a simple case of using the CREATE DATABASE command, the database has been named ComputingDB as this relates to the Computing Department, figure 3 below shows the creation of the database.

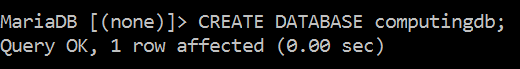
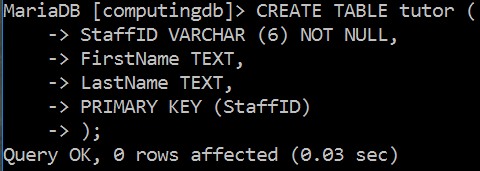
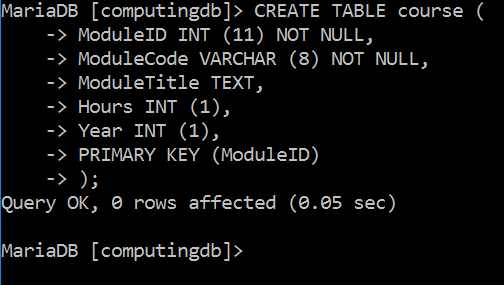


Figure - Database Creation

With the database created, the internal structure needs to be defined, to complete this a command of CREATE TABLE is used as shown in figure 4. It is important to create the tables to the schema plan. In this instance the course table is created followed by the tutor table. Once the Primary keys are set for the latter, the final Lecture table can be created as this uses foreign keys creating a relationship to the course and tutor tables.



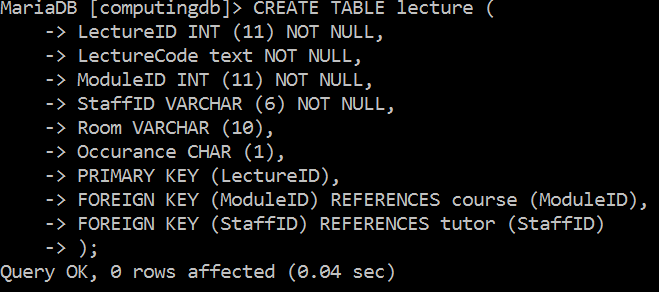
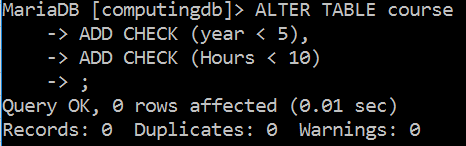
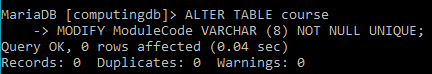


Figure - Table creation

The creation of the tables in figure 4 shows the SQL (Structured Query Language) syntax. The SQL commands are in capital letters while the attributes are in camel case notation. Each line represents an attribute with the attribute name, data type with size in brackets followed by a constraint. The NOT NULL value has been used to indicate the attribute cannot contain a null value. Check constraints are also added to the database attributes which ensures database insert anomalies are reduced to an absolute minimum. This can be done at the same time as the CREATE statement is issued, this example shows the use of the ALTER statement to add these constraints. Figure 5 shows the CHECK and UNIQUE constraint being applied.





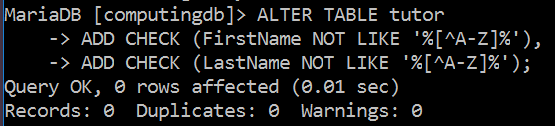


Figure - CHECK Constraint

This CHECK statement will provide a constraint on the data entered into the attribute year ensuring a value less than 5 is entered and hours is less than 10. The first and last name attributes from the tutor entity have a CHECK constraint to ensure only text is entered which will reduce INSERT anomalies. This particular version of Mariadb will not allow the check function to perform, however this check process will work with other database management systems.

Entering the data into the database uses the INSERT statement, this is a simple process which requires a statement shown in figure 6.

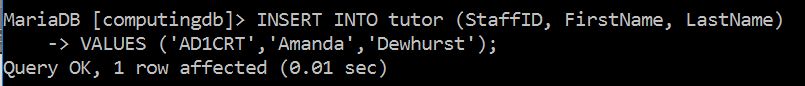
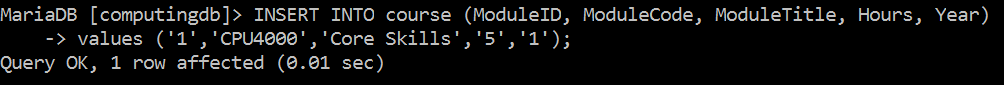
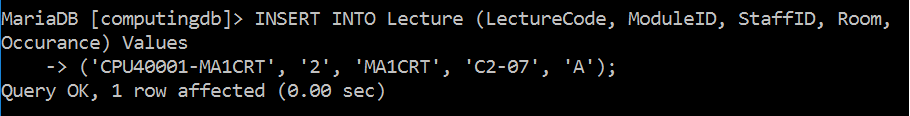
 

Figure - Insert Example

There are two methods of INSERT INTO statement as shown in figure 6, the first provides a rudimental check to ensure the data is entered into the correct attribute by firstly specifying the attributes followed by the values. The second method however, allows the user to enter the data directly into the database without the need to specify the attribute name. This could cause INSERT anomalies if information is entered incorrectly and has been avoided for this the creation of the database.

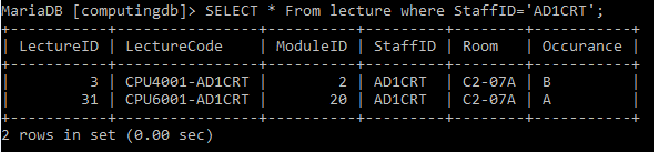
# Database Dictionary

The creation of the database dictionary is designed as a reference guide to the database administration team. The dictionary provides information relating to the object relationships, data types primary keys, foreign keys, general structure and a brief description of the created database. A database dictionary has been provided as a reference guide for the database created within the report, this is shown in appendix six. (techtarget, n.d.)

# Database Test Queries

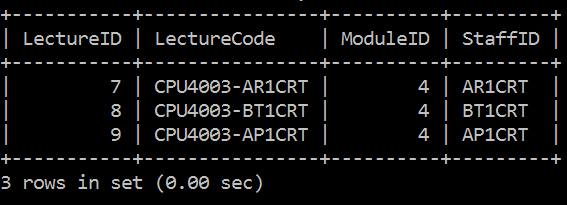
A number of example queries were performed to ensure the database was functioning correctly.

1. Search for a staff member and return the classes they are teaching.



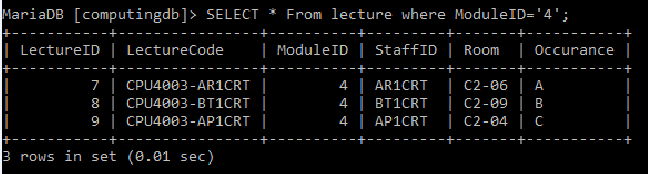
Example

1. Search for a module and return information about who is teaching it.



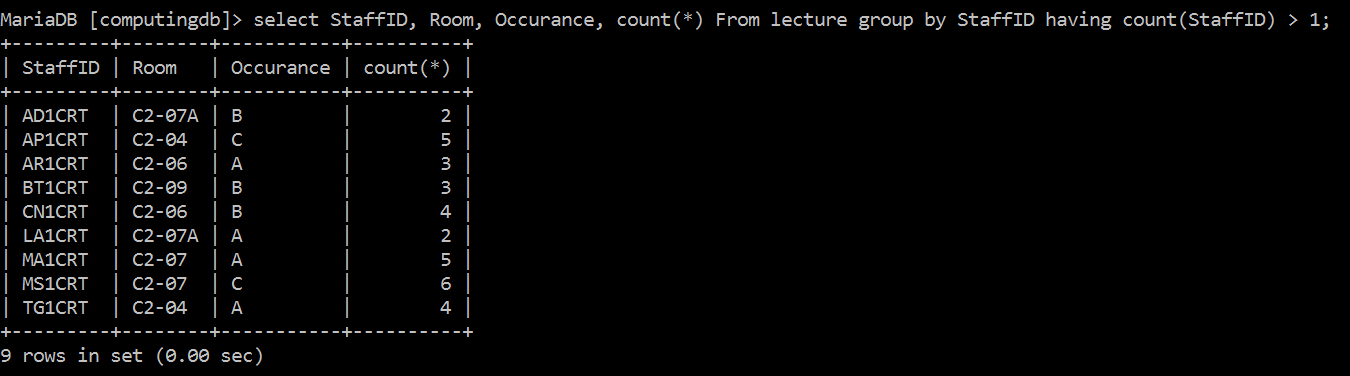
Example

1. Search for a module and return the information about the staff who is teaching it and the staff location



Example

1. Display all staff who teach more than two modules



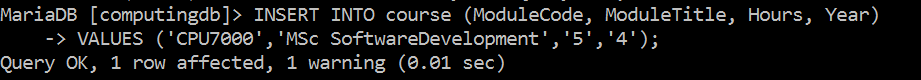
Example

1. Show all modules that do not currently have staff assigned including the occurrence information



Example

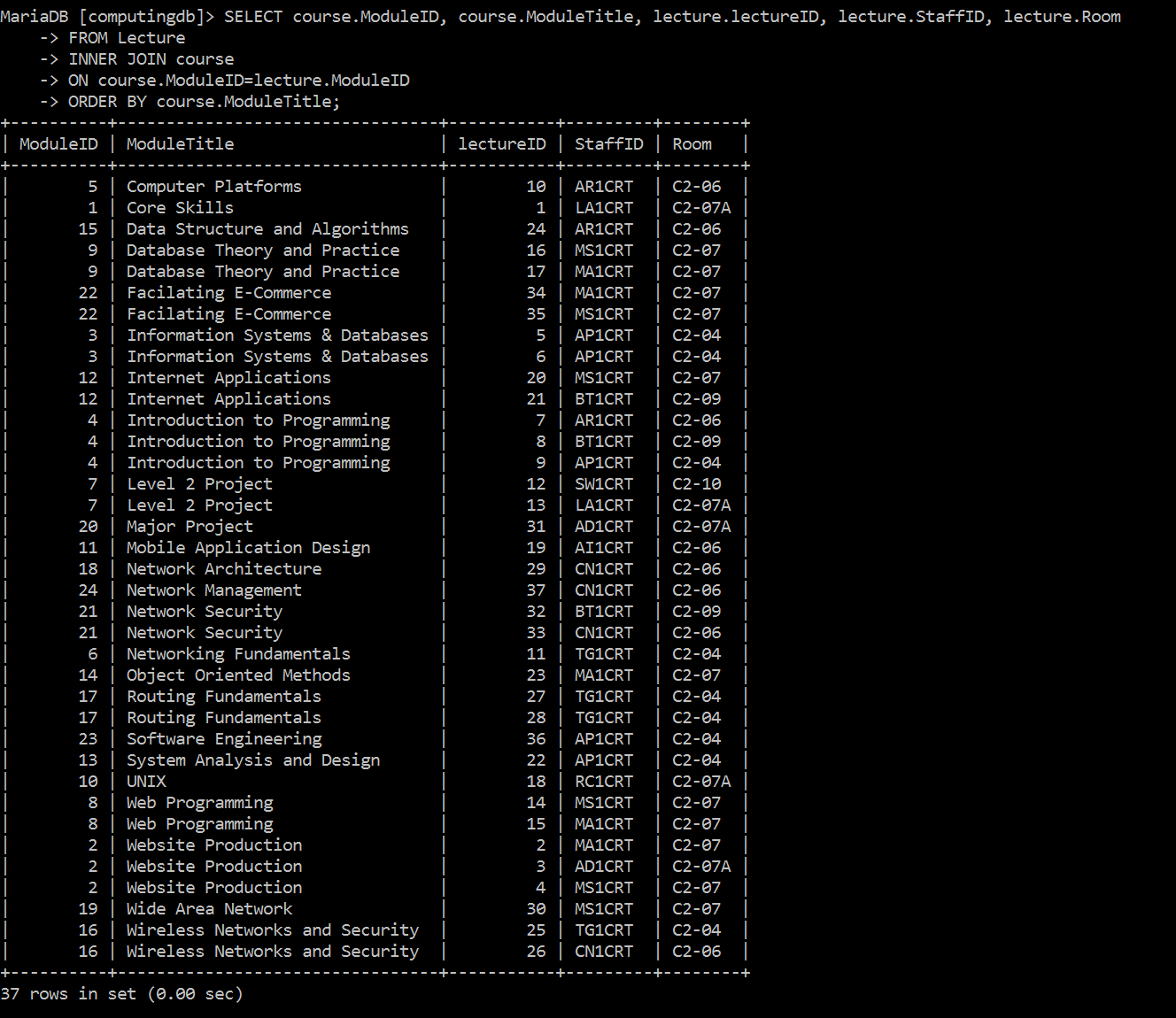
1. Add a module or make changes to any existing modules



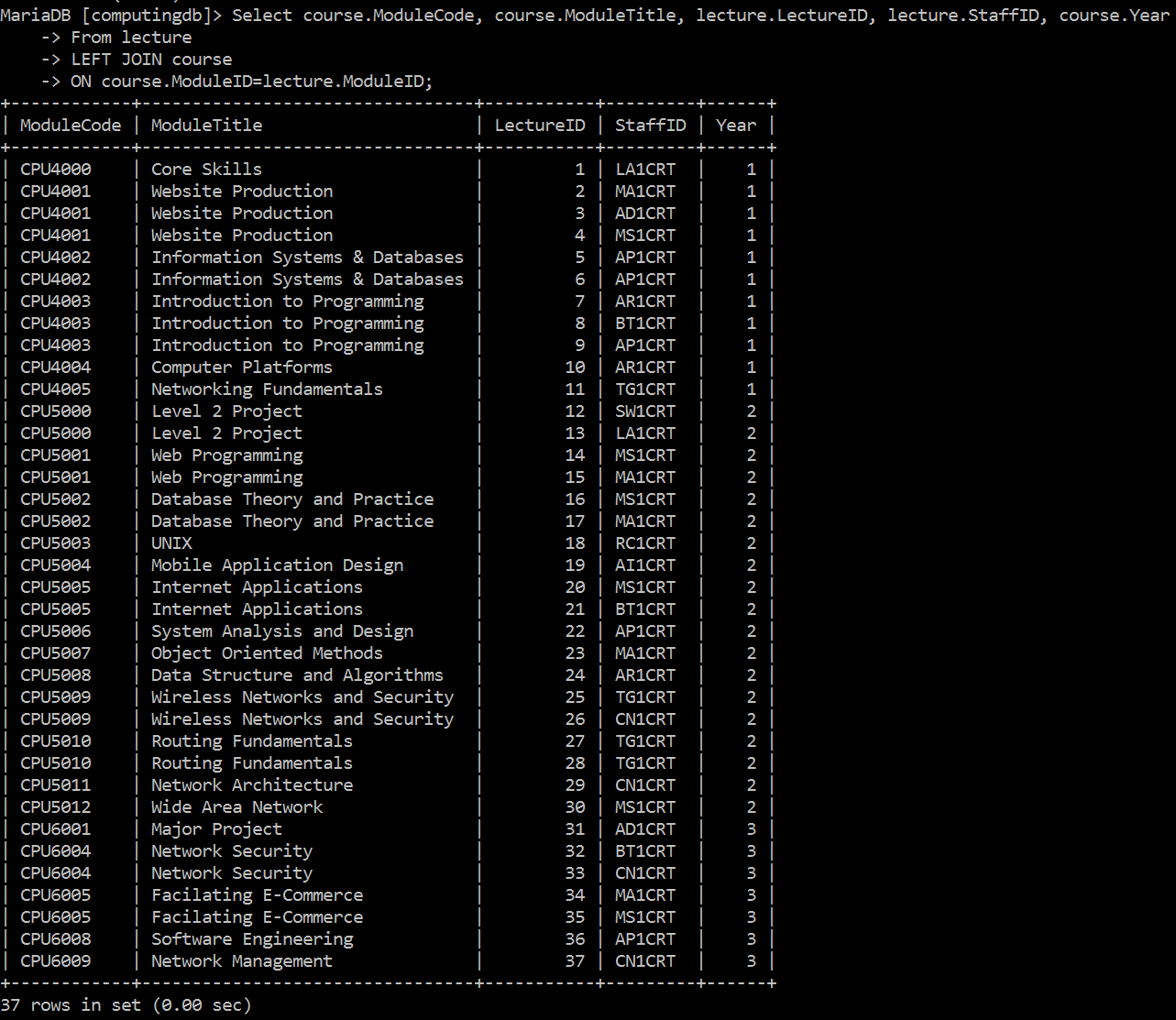
Example

1. Show examples of using joins to retrieve information from multiple tables

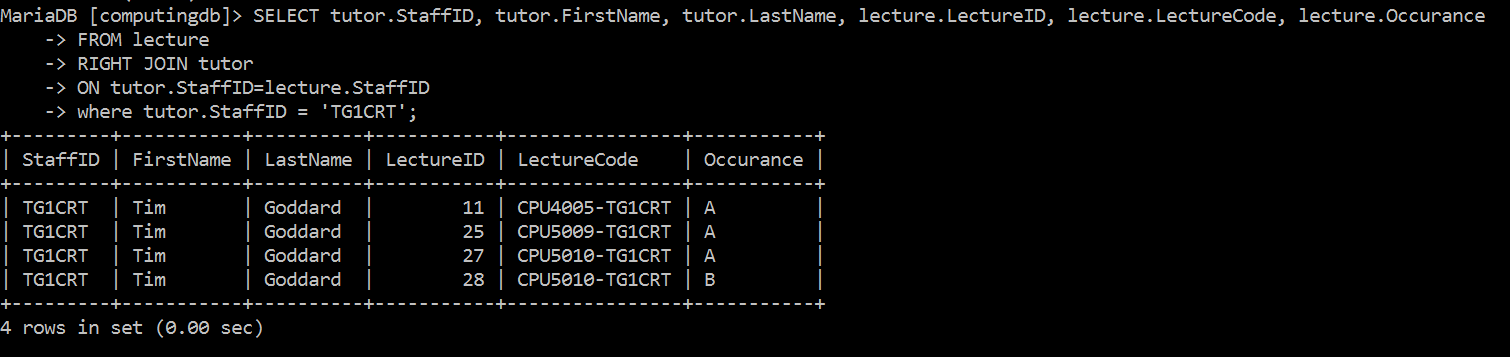
Inner Join Example:



Left Join Example

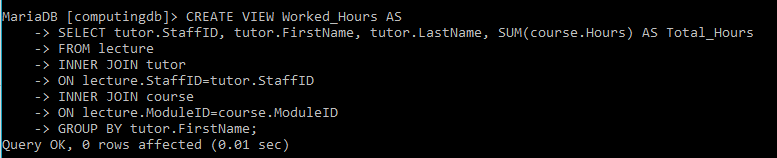


Right Join example

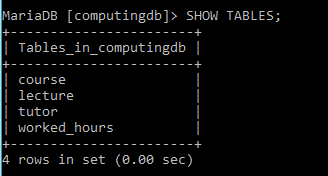


Example

1. The total number of hours worked by each member of staff should be calculated and stored as a view.







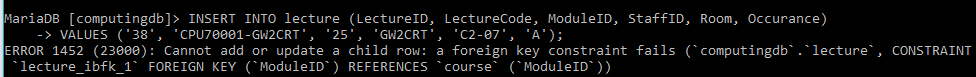
Example

# C.R.U.D Operations

To ensure the database is functioning correctly a test data set is also produced which ensures the four basic functions of Create, Update, Read and Delete (CRUD) work correctly. This routine can also check the constraints within the database.

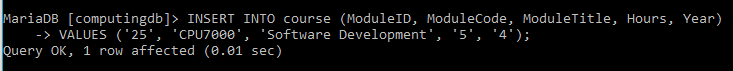
The first table to test is the Lecture table:

### Lecture Table

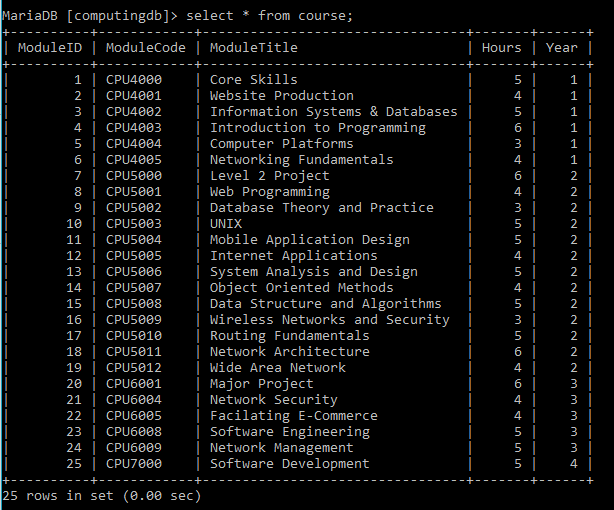


This Violation is caused as the Foreign key constraint has failed, to overcome this violation Course and Tutor tables will need to be updated as they contain the Primary Keys relating to the Foreign keys within the Lecture table. Due to the violation, the course table followed by the tutor table will be updated before continuing the CRUD procedure.

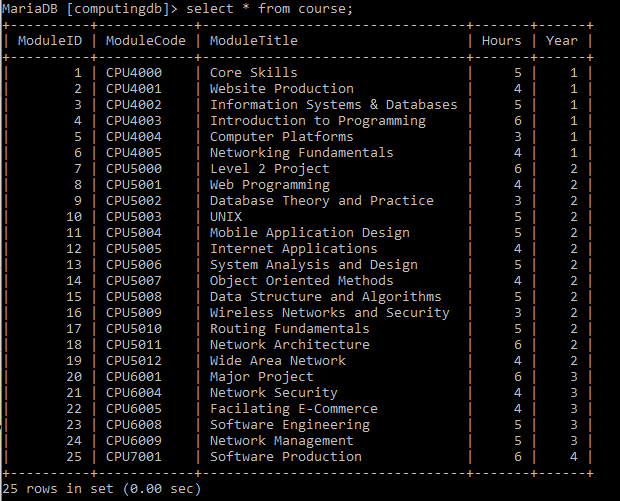
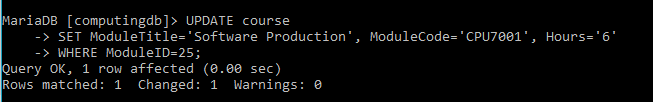
### Course Table

Create - 

Read



Update



Delete

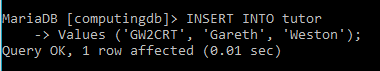


To complete the testing procedure of the course, table a check is made to ensure the constraints are working correctly. This shows there is a duplicate entry for the Module Code CPU6009 and will not allow a second tuple to be entered. This is known as a UNIQUE constraint where each entry in this attribute is to unique.

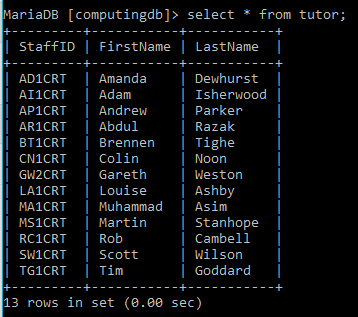


### Tutor Table

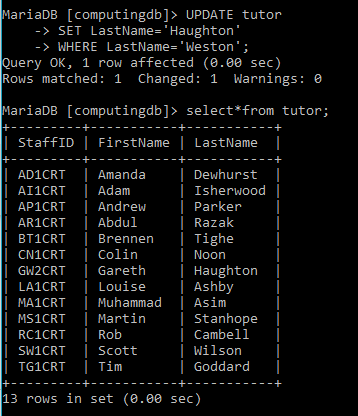
Create



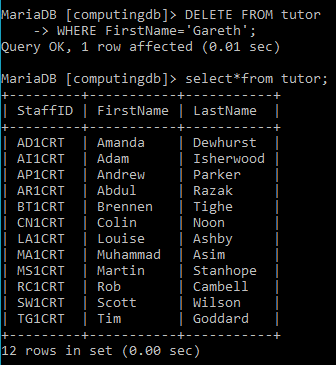
Read



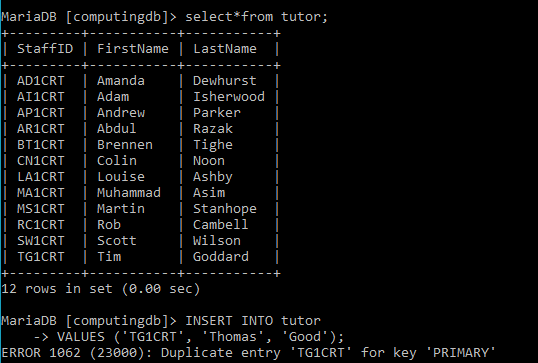
Update



Delete



To complete the testing procedure of the tutor table, a constraint check is made. This violation shows the Primary Key constraint. The attribute staffID is assigned as the primary key which ensures all entries are unique.

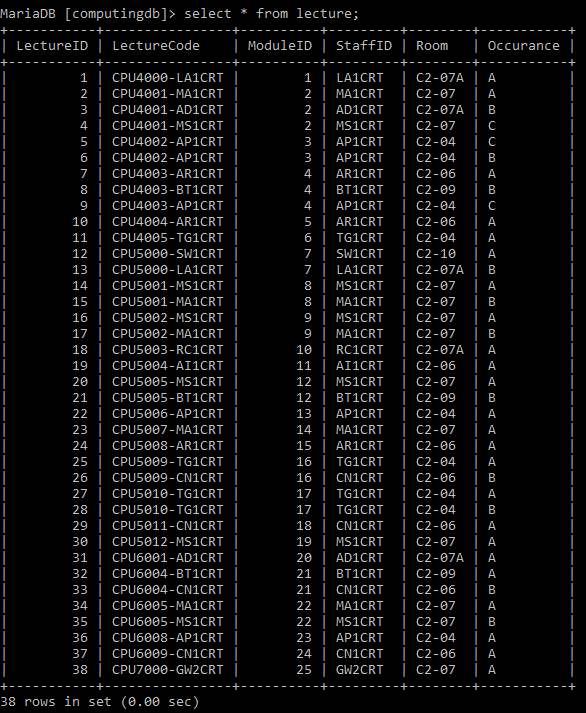


### Lecture Table cont

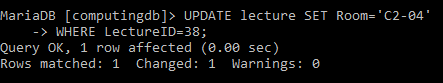
Create



Read



Update



Delete



# Conclusion

To Improve the university departments data storage issue, the flat file system needed to be replaced with a full relational database system. Using a number of data models and database design techniques the departments data was passed through various tests, each reaching a different stage of normalisation. Upon completion of each stage the data has less redundancy and greater integrity leading to the creation of an efficient database.

The initial planning of the database is crucial, this acts as a guide throughout the process. An entity relationship diagram is produced from the normalised data which gives the database designer an insight into how each object is connected.

Creating the database includes the entry of structured data into the database application, this allows the user to query or interrogate the data for a specified result. There are a number of varying database applications which could be used for this purpose, however, mariadb was chosen for its open source stature and free availability.

Test queries are run on the database to ensure the correct functionality with Integrity checks to ensure the database structure is free from any possible anomalies. This entire process will lead to an efficient database system which will continue to be amended, manipulated and updated for a number of years.

# Appendices

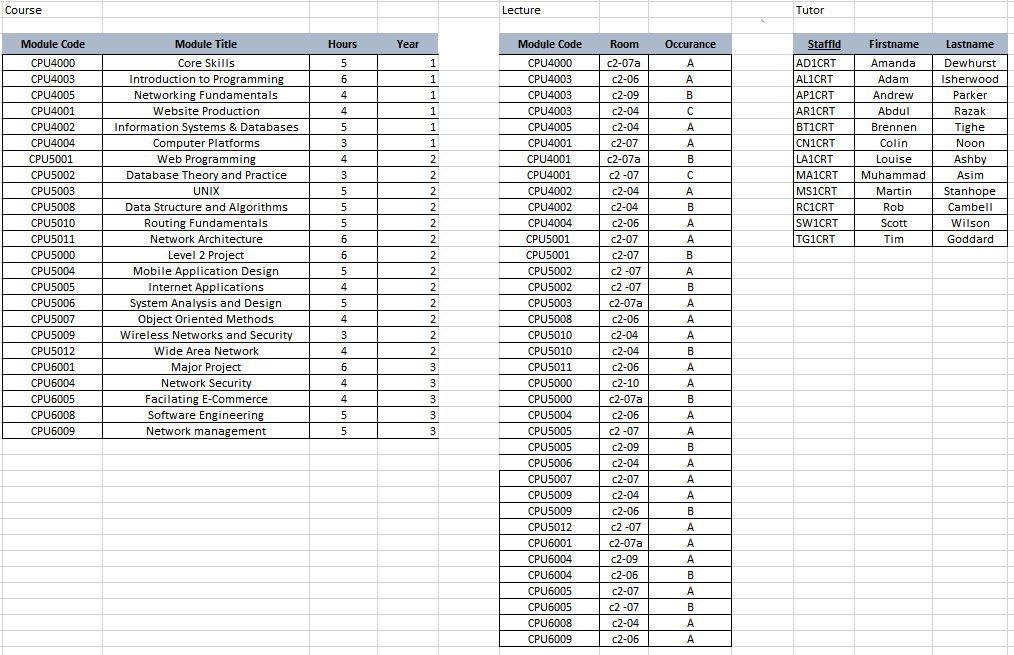
### Appendix One



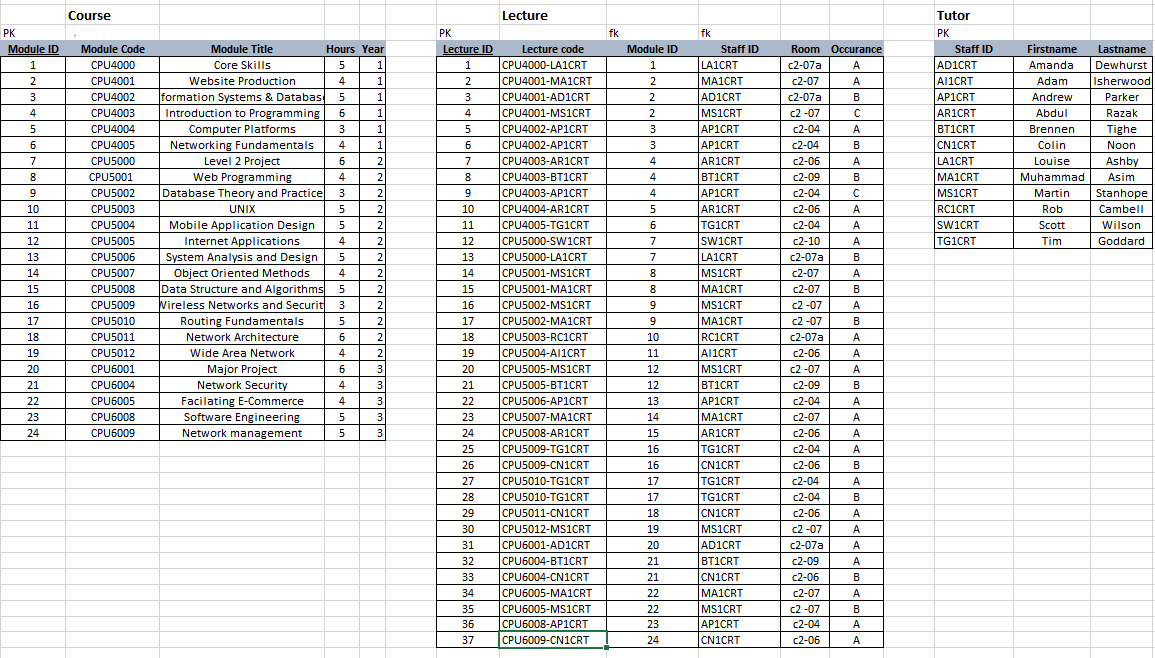
### Appendix Two



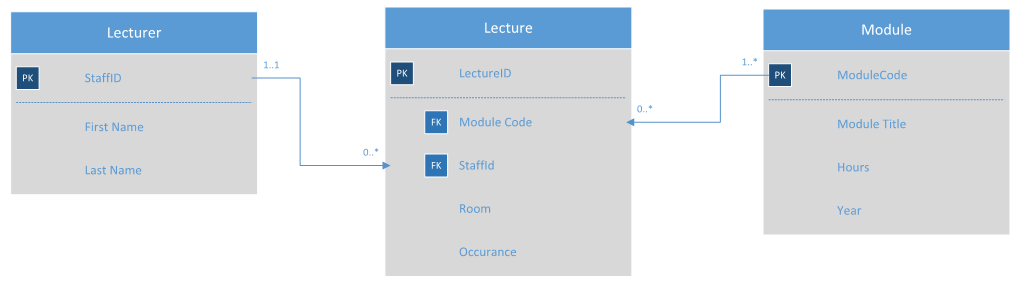
### Appendix Three



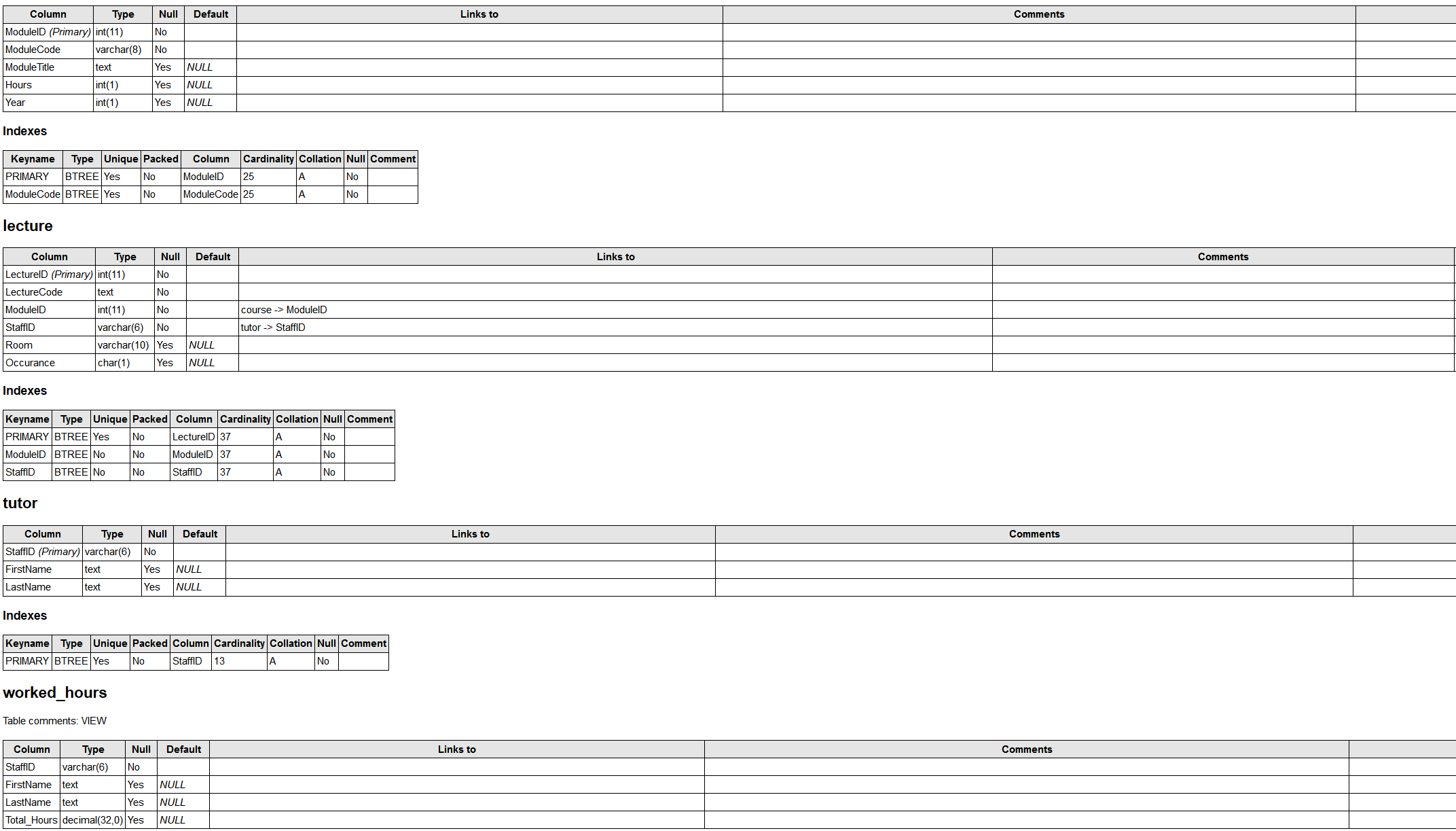
### Appendix Four

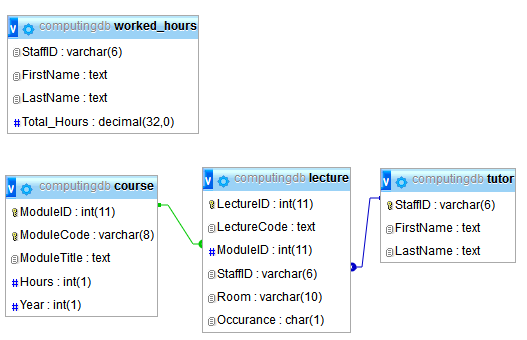


### Appendix Five



### Appendix Six





# Bibliography

Connolly, T. & Begg, C., 2014. *Database Systems A Practical Approach to Desigh, Implemementation and Management.* 6th ed. Harlow, United Kingdom: Pearson.

Jewett, T., n.d. *Basic structure: associations.* [Online]   
Available at: http://www.tomjewett.com/dbdesign/dbdesign.php?page=association.php  
[Accessed 24th November 2016].

MariaDB, n.d. *About Us.* [Online]   
Available at: https://mariadb.com/about-us  
[Accessed 25th November 2016].

Microsoft, 2016. *Developing a Database Plan.* [Online]   
Available at: https://technet.microsoft.com/en-us/library/ms177460(v=sql.105).aspx  
[Accessed 19th November 2016].

Teach ICT, n.d. *Databases.* [Online]   
Available at: http://www.teach-ict.com/as\_as\_computing/ocr/H447/F453/3\_3\_9/database\_design/miniweb/index.htm  
[Accessed 20th November 2016].

techtarget, n.d. *Data Dictionary.* [Online]   
Available at: http://searchsoa.techtarget.com/definition/data-dictionary  
[Accessed 6th December 2016].

W3Schools, n.d. *SQL Constraints.* [Online]   
Available at: http://www.w3schools.com/sql/sql\_constraints.asp  
[Accessed 24th November 2016].

W3Schools, n.d. *SQL Quick Reference.* [Online]   
Available at: http://www.w3schools.com/sql/sql\_quickref.asp  
[Accessed 4th December 2016].

W3Schools, n.d. *SQL SUM Function.* [Online]   
Available at: http://www.w3schools.com/sql/sql\_func\_sum.asp  
[Accessed 4th December 2016].

W3Schools, n.d. *SQL Tutorial.* [Online]   
Available at: http://www.w3schools.com/sql/default.asp  
[Accessed 6th December 2016].