Callum Owen-Bridge
15002504
COMI009AZ2016/7
Database Technology

Table of contents

1.0 Introduction

This report will demonstrate how a new database system will be created for Ace Training and the features of the database will be explained. The database must be secure and protected to keep personal details safe, secure and have limited access to details depending on the role in the university. The database will be easy to access and navigate for students, staff and prospective students who might look at the website as their first point of contact.

In this report, the creation and features of the database will be shown and explained. This will be done using diagrams, descriptions and SQL code.

2.0 History of Databases

Introduction

Database is a collection of information specifically organised, which allows a user or computer to quickly access the information to amend information, store records or make a query. A system which accesses the database to make a response for a query made by the user is called a database management system, abbreviated to DBMS. (The Editors of Encyclopaedia Britannica, 2016).

2.1 File Based database model

This database model was created by Herman Hollerith Herman, as he "conceived the idea that data could be represented by holes punched in paper cards then tabulated by machine." (Flat file database, 2016). This idea was then improved on in the 1980, where it became popular on multiple operating systems. In 2010 this database model became popular for use in content management systems as it allows web developers to change content directly, (Flat file database, 2016).

A flat file database model stores information as a single table, for example figure 1. A flat file can be plain text, which usually contains one record per line with each field separated using delimiters such as commas using software such as notepad, or a spreadsheet from an application such as Microsoft Office Excel shown in figure 1 below. (Flat file database, 2016).

File based databases can be used as configuration files to create operating systems, software applications and websites, (Tuffill, 2016). "flat file databases are used internally by various computer applications to store data related to configuration. Most of the applications permit users to store and retrieve information from flat files based on a predefined set of fields." (Divestopedia and Institute, 2016).

Name	Money In	Money Out	Date	Balance
Mr Smith	£ 50.00		01/09/2016	£ 100.00
Miss Jones		£ 10.00	02/09/2016	£ 50.00

Figure 1 above, shows an example of a flat file database containing two entities and five attributes.

An entity is an object that a user wants to store information about, such as Mr Smith in figure 1. An attribute is a field within the entity record, such as the date in figure 1.

2.1.1 Advantages

Advantages of a flat file database are that they are clear to search and recognise information, as there is a record per line with delimiters separating columns, (Tuffill, 2016). All the records can be stored in one place. This type of database is simple to understand, create and use.

2.1.2 Disadvantages

Disadvantages of a flat file database are that there is potential for duplications to occur as there is no mechanism coded into the system to prevent multiple records from occurring. It is hard to change data formats, for example if the date column had to be formatted to an American date format, for example from 01/09/2016 to 09/01/2016, then each record would have to be changed one at a time, which is time consuming. This type of database cannot restrict what type of data the user can see once the flat file is accessed all records and fields are viewable, so confidential information can be seen by any user. (Teach-ICT A level computing OCR exam board – features of a flat file database, no date). If more than two flat files are used and contain the same fields for a record such as someone's email address, then all files containing this field would have to be manually modified. (databasedev, 2003).

2.1.3 Recommendations

Flat file models are recommended for creating applications or for storing a small amount of records, using spreadsheets.

2.2 Hierarchical database model

A hierarchical database model organises information into a tree-like structure. The information is stored as records, which links to other tables of the tree such as the Staff table linking to the Equipment table by the field Emp No, shown in figure 2. This hierarchical model was created by IBM together with North American Rockwell in the 1960s. The structure forms a one-to-many relationship or a parent-child relationship, this is where one table is linked to many tables by one field type, for example employee number used in the tables in figure 2. However, each child table can only have one parent table. (Hierarchical database model, 2016).

To access data from a hierarchical model the user must start from the root node of the tree, and queries must pass through the nodes to move down to the next node, for example if figure 3 is used the user will start from Staff and move from node to node to access Equip type. (Hierarchical database model, 2016). Hierarchical database models are still used in healthcare to record patients within hospitals, for example if they entered a hospital and had an appointment for an MRI scan or to record a patients' medical record. To create this type of database a plan of the links must first be made to keep the database efficient and clear.

Staff				Equipment		Equip Type			Start Time	
Emp No	First Name	Last Name	Job Title	Emp No	Type	Type	Serial No	Date Bought	Emp No	Time
1500	Sam	Smith	Admin	1500	Mobile	Mobile	19534	15/09/2009	1500	08:30:00
1501	Sarah	Jones	Engineer	1501	Tool Bag	Tool Bag	98834	10/01/2001	1501	12:00:00
1502	Adam	Brown	Engineer	1502	Laptop	Laptop	78455	01/04/2016	1502	17:20:00

Figure 2 above, shows an example of hierarchical database model, the parent is the Staff table and the child to this table is the equipment table, they are linked by the Emp No field.

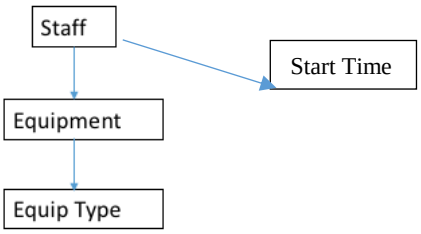


Figure 3 above shows the tables in tree structure. The root node is Staff. Each line represents a link.

2.2.1 Advantages

The advantages of a hierarchical database model are that it is simple database model, and is very fast to access information at the top of the tree, (Hierarchical data model, no date). The record fields are split into separate tables or nodes therefore making the data clearer to read, this also allows for security measures to be enabled preventing certain users from accessing information, for example preventing users from seeing an employee's address or bank information. As Taylor explains that QBE (Query-By-Example) can be used to restrict data, "The security module ensures that researchers working in one clinic do not get access to data from another clinic. The security can be based on a flexible taxonomy structure that allows ordinary users to access data from individual clinics and super users to access data from all clinics." (Taylor, 2003)

2.2.2 Disadvantages

The disadvantages of a hierarchical database model are the database can become very slow when accessing data in lower entities, such as at the Equip Type section of Figure 3. Searching for data requires another system called database management system which has to search the whole tree until the data is found which makes searches very slow, (Hierarchical data model, no date). A rule of this database model is that all nodes must be accessed through the root node, which can make adding and deleting very complex.

2.2.3 Recommendations

The recommended use for this database model is for applications in hospitals to hold records of patients, businesses to keep track of employees and the education Department to keep records of students and teachers. (Database models – hierarchical model, 2001).

2.3 Network database model

This database model is not restricted to being a hierarchy, the nodes are known as objects and relationship types are known as arcs. This model was created by Charles Bachman and was developed and published in 1969. It is based on a many-to-many relationship, so this allows each record to have multiple links to other tables, this forms a network structure shown in figure 4. (Network model, 2015).

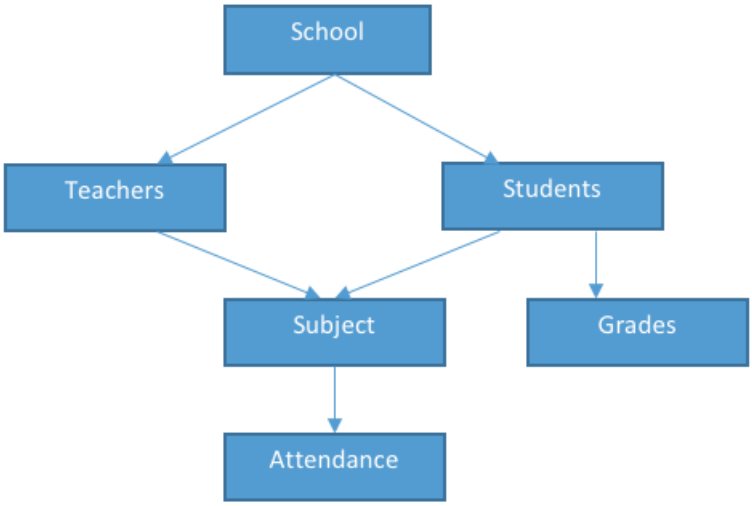


Figure 4 above shows the Network data model and how it uses different relationship models.

2.3.1 Advantages

The advantages of a network database model are that its simple to design, understand and access. This model can use many relationship types, therefore can be used in more real-life situations. (Arora, 1975).

2.3.2 Disadvantages

The disadvantages of a network database model are due to the complex relationships the system becomes more complex, therefore maintaining the system becomes much more difficult, such as amending records. Changing the structure of the database can be very complex due to the many links between records. (Arora, 1975).

2.3.3 Recommendations

This database model can be used for many real-life situations such as within businesses, to show where employees are situated, what equipment they use, their wage and their attendance.

2.4 Relational database model

The relational database model was created by Edgar F. Codd. “In the relational model of a database, all data is represented in terms of tuples, grouped into relations.” (Relational model, 2016). A tuple is a limited list of ordered elements. Relational databases usually use SQL, which is a structured query language, this is where users provide the information the database contains and its purpose, the management software will then sort the information into a structure for storing and retrieving information, (Relational model, 2016).

EmpNo	Name	DeptName	DeptNo
50	James	Admin	100
51	Sam	Accounting	200
52	Adam	Accounting	200
53	Emily	Admin	100

EmpNo	Name	DeptNo
50	James	100
51	Sam	200
52	Adam	200
53	Emily	100

DeptNo	DeptName
100	Admin
200	Accounting
200	Accounting
100	Admin

Figure 5 above shows an example of tables used in a relational database model

In in first table of figure 5, the information is more difficult to alter because if both employees Sam and Adam leave the Accounting department will be lost. However, if the relational database model is used this can be prevented. In the second table of figure 5, the department number is used to link to another table containing the department names, so if James was transferred to the accounting department only the department number must change or if both Sam and Adam left, the accounting department will remain. The links between tables are created by using key fields with a unique identity such as DeptNo.

2.4.1 Advantages

The advantages of a relational database model are, due to data only being entered once, the storage of the database is more efficient there is less likely to be duplication errors. More complex queries can be made due to a language called structured query language which allows programmers to update entities. Due to the ability to spilt the data into more tables, the extra tables can be protected so that when a user accesses the database only those with authorisation can access these tables. For example, an employee could see what their timetable is but not their colleagues. (Teach-ICT A level computing OCR exam board relational database advantage, no date).

2.4.2 Disadvantages

The disadvantages of a relational database model are that they are quite expensive to run, as a programmer will be needed to set it up using a structured query language and an administrator will be needed to help maintain the database. All the information should either be typed or entered as a flat file which can be time consuming and if any confidential information is being used a secure method should be used to prevent unauthorised user access. The amount of data that can be entered into a field, is restricted. This can cause problems with names, (Martin, 2016).

2.4.3 Recommendations

The relational database can be used for many situations such a library, organisations, hospitals and education. An example of the use in hospitals would be if a patient makes an appointment with the doctor, no one will be able to access the patients records only the doctor, however the receptionist will be able to book the appointment and can view who the patient is going to see.

2.5 Object Orientated database model

The object orientated database model was created through research in the 1970s. This database model represents real world entities in the form of objects, such as the objects used in object orientated programming, instead of tables.

Each object has an identity, state and behaviour. An object identity is used by the system to identify objects, which is not visible to the user. An object's state is where the object's data is contained, for example an object's name, shown in in figure 6. The object's behaviour contains the methods which manipulates the database.

Figure 6 above shows how an object orientated database model

In figure 6, the first table is an object called Record, its instance is the table to the right of it, this is the state of the object, the information for the record. The arrow shows how the first object links to another object by one attribute, in this case ID.

2.5.1 Advantages

The advantages of an object orientated database model are, query languages do not have to be used as the information is accessed through the objects, which allows for easier navigation. The user doesn't have to add primary keys to tuples, as the object orientated database model uses software which adds unique values to the objects without the user's implementation and so the user cannot see this information reducing the risk of the same identity values being used. (Obasanjo, 2001). This database model is based on real world objects. This database model has a much more improved performance compared to the previous models. Due to the efficiency of this model it has a much larger storage capacity compared to the other models and a much higher access speed. (Name, 2010).

2.5.2 Disadvantages

The disadvantages of an object orientated database model are that this model is dependent on one specific type of language, so to access the data "a specific language using a specific API" (Obasanjo, 2001). There is a lack of standards with this model as "There is no universally agreed data model", (Thakur, no date). There is a lack of security options, as the user cannot restrict certain objects. (Thakur, no date).

2.5.3 Recommendations

This database model is currently being used in "The Chicago Stock Exchange manages stock trades via a Versant ODBMS.", (Obasanjo, 2001). "Ajou University Medical Center in South Korea uses InterSystems' Cach¡§¡§ ODBMS to support all hospital functions including mission-critical departments.", (Obasanjo, 2001).

3.0 Scenario

3.1 Introduction

This section is about the analysis of Ace Training's database brief. It is important to first analyse the brief; to acquire the client's requirement, to prevent any inconsistencies in the database structure and to view any rules the client requires to have incorporated into the database. First, the database requirements will be discussed and then the business rules and operations that need to be incorporated into the database.

Database Requirements

3.2 Course Details

The database will store details for each course that is provided by Ace Training.
After analysing the brief, the details for each course that require storing are:

- Name
- Department
- Credit value
- Course Code
- Start date
- End Date
- Year of study *

The bullet point with an asterisk are extra details required to be stored in the database.

3.2.1 Tutor Requirements

The following are requirements needed by the tutor when accessing the database. The tutor must be able to create courses, so they must be able to add the name, Department, Credit value, Course code, Start date, End Date. The tutor must be able to enrol students using an online form and must be able to authorise student enrolments.

3.2.2 Student Requirements

The student will be able view details of the courses, such as course name, course code, their attendance, credit value, End date, start date, department and year of study. Students would not be able to amend or delete these details.

3.3 Resource Details

The database will store details for each resource that is provided by the tutor.
After analysing the brief, the details for each resource that require storing are:

- Title
- Type
- Student availability
- Shared with

3.3.1 Tutor Requirements

The following are requirements needed by the tutor when accessing the database. The tutor must be able to add, delete and amend resources in the database. The tutor must be able add the name, type, allow student availability or not and allow availability within date range, and can share the resources with other departments.

3.3.2 Student Requirements

The student must be able to access the resources and view them. The students cannot amend, add or delete the resources.

3.4 Fee Details

The database will store details for each fee that a student has.
After analysing the brief, the details for each fee that require storing are:

- Total
- Duration
- Amount paid

3.4.1 Tutor Requirements

The following are requirements needed by the tutor when accessing the database. The tutor must not be able view the fee details of a student or amend any details.

3.4.2 Student Requirements

The student must be able to view fee details, such as the total, duration and amount paid. The student must not be able to amend, add or delete the fee details. Other students should not be able to view other student's fees.

3.5 Quiz Details

The database will store details for each quiz that is provided by Tutor.
After analysing the brief, the details for each quiz that require storing are:

- Name
- quizID

3.5.1 Tutor Requirements

The following are requirements needed by the tutor when accessing the database. The tutor must be able to add quizzes to the courses. The tutor can add, amend or delete the quizzes, such as adding the quiz name and questions.

3.5.2 Student Requirements

The students must be able to access the quiz available on their course(s), they can only view and complete the quiz. The student must not be able to add, amend or delete the quiz details.

3.6 Subtype to the entity quiz

The database will store questions for each quiz that is provided by Tutor.
After analysing the brief, the details for questions that require storing are:

- fill in the blank
- multiple choice
- true or false

3.6.1 Tutor Requirements

The following are requirements needed by the tutor when accessing the database. The tutor must be able to add questions. The tutor can add, amend or delete the questions

3.6.2 Student Requirements

The students must be able to access the quiz available on their course(s), they can only view and complete the questions. The student must not be able to add, amend or delete the questions.

3.7 Grade Details

The database will store details for each score that is provided by the Tutor.
After analysing the brief, the details for each score that requires storing are:

- Score
- Completion

3.7.1 Tutor Requirements

The following are requirements needed by the tutor when accessing the database. The tutor must be able to view the student scores and whether students have completed the quiz. The tutor must be able to view all the scores of all the students who have enrolled for the course.

3.7.2 Student Requirements

The students must be able to see their own grades for each quiz taken. The students cannot amend, add or delete any grade details. The students must not be able to view other student grades.

3.8 Student Details

The database will store details for each student. After analysing the brief, the details for each student that require storing are:

- Forename
- Surname
- AddressL1
- AddressL2
- Postcode*
- Town City*
- Country*
- Date of Birth
- Student ID*
- Email Address
- Phone Number*
- Next of kin details
- Course(s) enrolled
- National Insurance*
- Passport number
- Visa Number *
- Visa expiry date
- Registered*

The bullet point with an asterisk are extra details required to be stored in the database.

3.8.1 Tutor Requirements

The following are requirements needed by the tutor when accessing the database. The tutor can only view part of student details such as, name student ID and email address. The tutor cannot amend, delete or add the student details.

3.8.2 Student Requirements

The student must be able to amend their student details, such as their name, address, date of birth, email address, phone number, national insurance number, passport number and visa expiry date. The students must not be able to create or amend their own student ID. The students must not be able to view other student details.

3.9 Tutor Details

The database will store details for each tutor. After analysing the brief, the details for each tutor that require storing are:

- Forename(s)
- Surname
- Date of Birth
- AddressL1
- AddressL2
- Postcode*
- Town City*
- Country *
- Tutor ID*
- Email Address
- Phone Number*
- National Insurance
- Office Number
- Extension number

The bullet point with an asterisk are extra details required to be stored in the database.

3.9.1 Tutor Requirements

The following are requirements needed by the tutor when accessing the database. The tutor must be able to amend their tutor details. The tutor cannot add details to this entity as the administrators can only create tutors' details. The tutors can view other tutor details such as, name, email address, phone number, office number and extension number. The remaining details should remain private from other tutors.

3.9.2 Student Requirements

The student should be able to view the tutor's name, email address and office number. The other details of the tutor should be kept private from students such as, address, date of birth, tutor ID, phone number, National Insurance number and Extension number. The Students must not be able to amend, add or delete tutor details.

3.9.3 Administration Requirements

The administrators must be able to create tutors and add them to the database.

3.10 Next of Kin Details

The database will store details for each next of kin provided. After analysing the brief, the details that require storing are:

- Forename
- Surname
- AddressL1
- AddressL2
- Postcode
- Town City
- Phone Number
- Relationship

3.10.1 Tutor Requirements

The following are requirements needed by the tutor when accessing the database. The tutor must be able to amend next of kin details, such as name, address phone number and their relationship. Other tutors must not be able to view other tutors', students or administrators' next of kin details.

3.10.2 Student Requirements

The student must be able to amend next of kin details, such as name, address phone number and their relationship. Other students must not be able to view other tutors', students or administrator's next of kin details.

3.10.3 Administrator Requirements

The administrator must be able to amend next of kin details, such as name, address, phone number and their relationship. Administrators must be able to view other tutors', students or administrator's next of kin details to be able to amend any faults.

3.11 Institution Details

The database will store details for the institution provided by Ace Training. After analysing the brief, the details for institution that require storing are:

- Name
- AddressL1
- AddressL2
- Postcode
- Town City
- Phone Number

3.11.1 Tutor Requirements

The tutor can only view the details of the institution. The tutor must not be able to amend the institution details.

3.11.2 Student Requirements

The student can only view the details of the institution. The student must not be able to amend the institution details.

3.12 Administrator Details

The database will store details for each administrator. After analysing the brief, the details for each administrator that require storing are:

- Surname
- Forename
- Phone Number
- Email address
- AddressL1
- AddressL2
- Postcode
- Town City
- country
- Phone Number
- Date of Birth

It is recommended to include the administrator details because administrators are required to maintain the database. Details of the administrator will be needed by the tutors, if there are any issues found.

3.12.1 Tutor Requirements

The tutor can only view part of the details of the administrator, such as name, email address and phone number. The tutor must not be able to amend, add or delete the administrator details.

3.12.2 Student Requirements

The student cannot view the details of the administrator. The student must not be able to amend, add or delete the administrator details.

3.12.3 Administrator Requirements

The administrator must be able to amend their details, such as their name, address, date of birth, email address and phone number. The administrator must not be able to create or amend their own administrator ID.

3.13 Register Details

The database will store details for each register taken. After analysing the brief, the details for each register that require storing are:

- Time
- Date
- Seminar room
- Lecture room
- Lab room
- Attended

3.13.1 Tutor Requirements

The following are requirements needed by the tutor when accessing the database. The tutor must be able to add to the register details. However, the tutor cannot be able to delete or update register entries.

3.13.2 Student Requirements

The student must be able to view their register details, but must not be able to view other student details. Student must not be able to update or delete register entries

3.13.3 Administrator Requirements

The administrator must be able to amend the register details.

3.14 Student Progression Details

The database will store details for each of the student's progression. After analysing the brief, the details for student progression that require storing are:

- Average knowledge
- Course progression

3.14.1 Tutor Requirements

The tutor can only view these details. The tutor must not be able to amend or delete these details.

3.14.2 Student Requirements

The student cannot view these details. The student must not be able to amend, add or delete the student progression details.

3.12 Business rules and Business Operations

A business rule is a database restriction; these rules prevent users from viewing, updating or deleting data they aren't authorised to access.
A business operation is a database action, such as allowing students to view their own grades.
These rules and operations are required to implement the database.

Below is a table containing the business rules for the database.

Tutor Business Rules

BR	Tutor cannot insert, update, delete student details
BR	Tutor cannot delete course
BR	Tutor cannot be deleted
BR	Tutor cannot add themselves as a course tutor
BR	Tutor can only view institution details
BR	Tutor cannot update, delete register details
BR	Tutor cannot insert, update, delete tutorID, officeNumber, extensionNumber, emailAddress in tutor details
BR	Tutor cannot insert, update, delete administrator details
BR	Tutor can only view forename, surname, emailAddress, phoneNumber in administrator details
BR	Tutor can only view studentID, forename, surname, emailAddress in student details.

Student Business Rules

BR	Student cannot update, insert delete institution details
BR	Student cannot view other student's grades
BR	Student cannot enrol for already undertaken course
BR	Student cannot be deleted
BR	Student cannot enrol for new course with outstanding fees
BR	Student cannot insert, update, delete: studentID, email address, feeTotal, feeDuration, feeAmountPaid; in student details.
BR	Student cannot insert, delete update register details
BR	Student cannot insert, update, delete course details
BR	Student cannot enrol themselves for a course
BR	Student cannot insert, delete, update quiz details
BR	Student cannot insert, delete, update resource details
BR	Student cannot insert, update, delete grade details
BR	Student cannot insert, update, delete tutor details
BR	Student can only select forename, surname, officeNumber, emailAddress and phoneNumber in tutor details.

Administrator Business Rules

BR	Administrator cannot delete staff
BR	Administrator cannot delete students

Below is a table containing the business operations for the database.

Administrator Business Operations

BO	Administrator can create tutors
BO	Administrator has full database access
BO	Courses can be added
BO	Courses can be deleted

Tutor Business Operations

BO	Tutor can authorise student enrolment
BO	An Individual can register as a tutor
BO	Tutor can insert, update, select resources
BO	Tutor can make resources available to students
BO	Tutor can share resources to other courses
BO	Tutor can select, insert, update course details
BO	Tutor can select institution details
BO	Tutor can select, insert, update quiz details
BO	Tutor can select, insert, update grade details
BO	Tutor can view spProgression, spAverageKnowledge in studentCourse details
BO	Tutor can select tutor details
BO	Tutor insert, update forename, surname, addressL1, addressL2, townCity, postcode, nationalInsurance, dateOfBirth, nkForename, nkSurname, nkAddressL1, nkAddressL2, nkTownCity, nkPostcode, nkRelationship, nkPhoneNumber in tutor details
BO	Tutor can view forename, surname, phoneNumber, emailAddress, of administrator details
BO	Tutor can view and insert register details
BO	Student insert, update forename, surname, addressL1, addressL2, townCity, postcode, nationalInsurance, visaExpiryDate, passportNumber, dateOfBirth, nkForename, nkSurname, nkAddressL1, nkAddressL2, nkTownCity, nkPostcode, nkRelationship, nkPhoneNumber student details

Student Business Operations

BO	Student can view their grades
BO	Student can view available resources
BO	Student can view course details
BO	Student can access Quiz
BO	Student can view their fee information
BO	Student can be unregistered
BO	Student can view attendance

4.0 Conceptual Design

4.1 Introduction

In this section a conceptual design of the database will be constructed to show a simple layout of how the database will be structured. A justification on the relationships made will be discussed and this section will end with a conclusion.

4.2 Model Explanation

The diagram script in chapter 4.3.1 is used to show the relationships between entities, the attributes entities contain and any synonyms used for the entities. The diagram script also allows the conceptual design to be designed less complex.

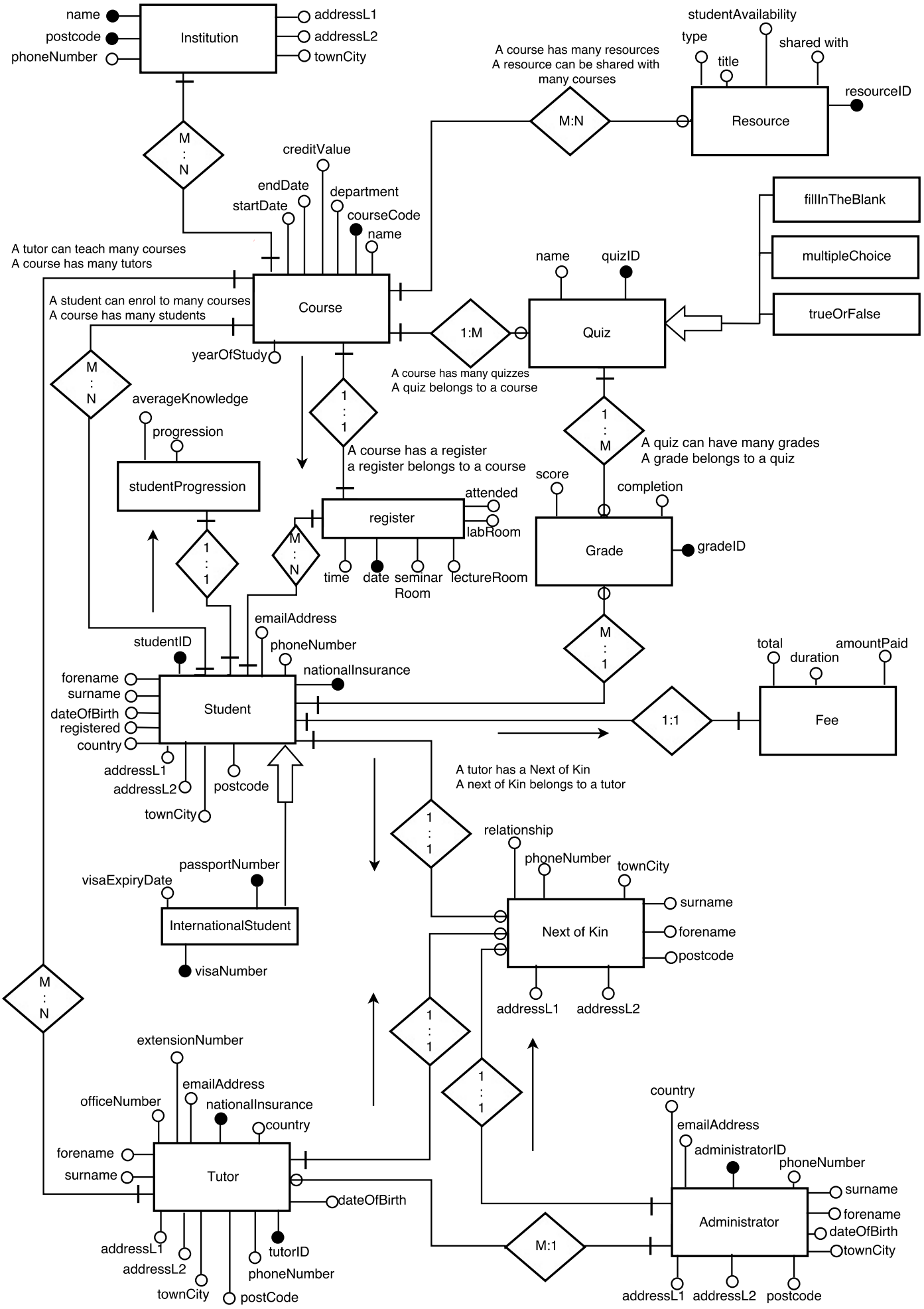
The conceptual design in chapter 4.3.2 is used to display how all the entities relate to one another, as well as their relationship types. This design is also used if the person it is designed for is not trained in databases.

4.3 Conceptual Design and Diagram Script

4.3.1 Diagram Script

Entity	Attribute	Synonym	Relation
Student	forename, surname, townCity, addressL1, addressL2, postcode, country, dateOfBirth, studentID, emailAddress, phoneNumber, nationalInsurance, registered	Pupil	International student, Next of Kin, Course, Fee, Grade, student Progression, register
International Student	visaExpiryDate, passportNumber, visaNumber		Student
Tutor	forename, surname, townCity, addressL1, addressL2, postcode, country, dateOfBirth, emailAddress, phoneNumber, nationalInsurance, officeNumber, extensionNumber, tutorID	Teacher, Course Tutor	Next of Kin, Course, Administrator
Course	department, name, courseCode, startDate, endDate, creditValue, yearOfStudy	Subject	Resource, Quiz, Student, Tutor, Institution, register
Quiz	name, quizID	Test, Exam	Course, Question, Grade
Grade	score, completion, gradeID	Score	Quiz, Student
Question	fillInTheBlank, trueOrFalse, multipleChoice		Quiz
Student Progression	averageKnowledge, studentProgression	Progression	Student
Fee	total, amountPaid, duration	Cost	Student
Resource	type, title, studentAvailability, sharedWith, resourceID	Documents	Course
Institution	name, townCity, addressL1, addressL2, postcode, phoneNumber	Training centre	Course
Administrator	forename, surname, emailAddress, administratorID, phoneNumber, townCity, addressL1, addressL2, postcode, dateOfBirth, country	Admin	Tutor, next of kin
NextOfKin	forename, surname, townCity, addressL1, addressL2, postcode, relationship, phoneNumber		Student, Tutor, administrator
register	date, time, lectureRoom, seminarRoom, labRoom, attended		Student, course

4.3.2 Conceptual Design



4.4 Justification of Relationships and Cardinalities

4.4.1 There are two types of attributes:

- Identifying attribute: an attribute that uniquely identifies the entity, for example student ID and transaction ID. This attribute type is shown by a black circle as shown in chapter 4.3.2 conceptual design.
- Describing attribute: an attribute which describes the entity such as Name and Department. This attribute type is shown by a white circle as shown in chapter 4.3.2 conceptual design.

4.4.2 Relationship types

- One to One: a tutor teaches a subject and a subject is taught by a tutor. Represented as 1:1 in the diamonds of the conceptual design in chapter 4.3.2.

- One to many: a student has many grades and many grades belong to a student. Represented as 1:M in the diamonds of the conceptual design in chapter 4.3.2.

- Many to many: a student is enrolled to many courses and a course has many students. Represented as M:N in the diamonds of the conceptual design in chapter 4.3.2.

- Mandatory: must be associated with, for example a student must be enrolled to a course. Represented as a small line in the conceptual design in chapter 4.3.2.

- Optional: may not be associated with, for example a student may not have a next of kin. Represented as an empty circle in the conceptual design in chapter 4.3.2.

- Generalisation: an object is a subtype, for example an international student is a subtype of the super type student. Represented by an arrow shown in the conceptual design in chapter 4.3.2.

- Recursive: where entities relate to each other but are under the same entity name for example, a manager is a type of staff but also manages many staff. This relationship is usually one to many.

4.4.3 Relationship Justification

- Institution to course
    - Many to many relationships
    - An institution will have many courses and a course and belong to many different institutions.

- Course to Resource
    - Many to many relationships
    - A course can have many resources and a resource can be shared to many courses.

- Course to Quiz
    - One to many relationship
    - A course has many quizzes and a quiz belongs to a course

- Course to Register
    - One to one relationship
    - A course may have a register and a register will belong to a course

- Course to Student
    - Many to many relationship
    - A course has many students and a student can attend many courses

- Course to Tutor
  - Many to many relationship
  - A course can have many tutors and a tutor can teach many courses.

- Quiz to Question
  - generalisation relationship
  - As a quiz contains many different questions and a question will belong to a certain quiz.

- Quiz to Grade
  - one to many relationship
  - a quiz can have many student grades and a student grade is for a quiz.

- Student to Grade
  - one to many relationship
  - A student can have many grades and a grade is for a student.

- Student to Fee
  - One to one relationship
  - A student can have a single fee and a fee can belong to a student
- Student to Next of Kin
  - one to one relationship
  - A student has a next of kin and a next of kin relates to a student

- Student to International student
  - Generalisation relationship
  - International student is a type of student

- Student to Student Progression
  - One to one relationship
  - A student will have a set of progression results per course taken and a set of progression results will be recorded for a student per course taken.

- Student to Register
  - Many to many relationship
  - A student may be registered to different registers and a register will hold many students.

- Tutor to Next of Kin
  - One to one relationship
  - A tutor has a next of kin and a next of kin relates to a tutor.

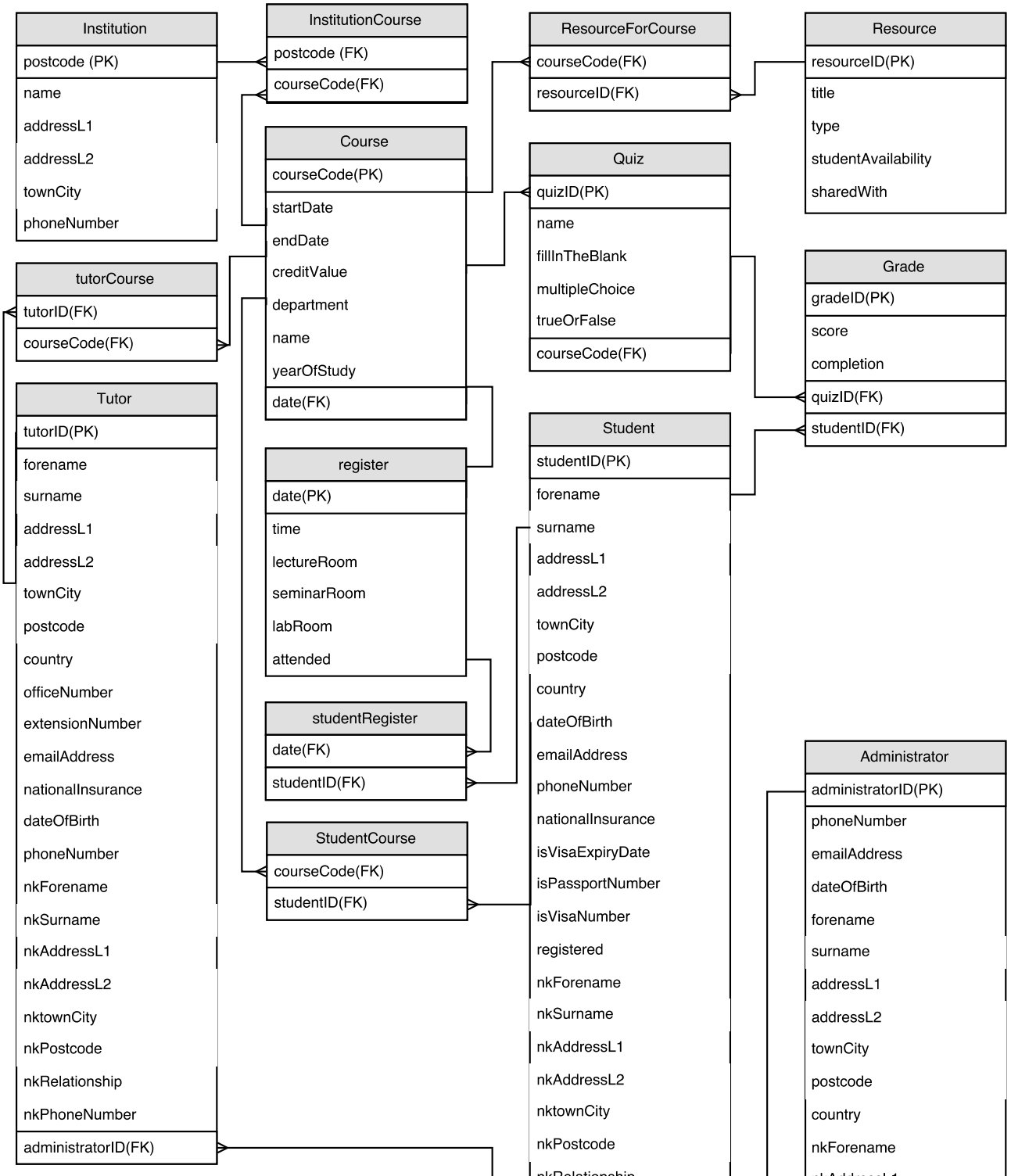
- Tutor to Course
  - Many to many relationship
  - A tutor teaches different course and a course can be taught by many tutors.

- Administrator to Tutor
  - One to many relationship
  - An administrator will create tutors and tutors will refer to an administrator.

- Administrator to Next of Kin
  - One to one relationship
  - An administrator has a next of kin and a next of kin relates to an administrator.

5.0 Logical Design

In this chapter the logical design will be presented. The logical design is created using the conceptual design. In the logical design, many to one and most one to one relationships are allowed, therefore many relationships have been altered such as the relationship between student and register.



Key:

PK: Primary Key
FK: Foreign Key
nk: next of Kin
sp: studentProgression
ID: identification
L: Line
is: international student

6.0 Normalisation

In this chapter a cumulative design and normalisation tables will be presented and described.

6.1 Cumulative Design

A cumulative design is a list of all the relations with rules of how the foreign keys function within the relations.

Institution (postcode, Name, addressL1, addressL2, townCity, phoneNumber)

InstitutionCourse (postcode, courseCode)
postcode → Institution **Update Cascade, Delete Restrict**
courseCode → Course **Update Cascade, Delete Restrict**

Tutor (tutorID, forename, surname, addressL1, addressL2, townCity, postcode, country, officeNumber, extensionNumber, emailAddress, nationalInsurance, dateOfBirth, phoneNumber, nksurname, nkforename, nkAddressL1, nkAddressL2, nkTownCity, nkPostcode, nkRelationship, nkPhoneNumber, administrationID)
administrationID→Administrator **Update Cascade, Delete Restrict**

TutorCourse (tutorID, courseCode)
tutorID→ tutor **Update Cascade, Delete Cascade**
courseCode → Course **Update Cascade, Delete Restrict**

Administrator (administrationID, phoneNumber, emailAddress, forename, surname, addressL1, addressL2, townCity, postcode, country, dateOfBirth, nksurname, nkforename, nkAddressL1, nkAddressL2, nkTownCity, nkPostcode, nkRelationship, nkPhoneNumber)

Course (courseCode, startDate, endDate, creditValue, department, name, yearOfStudy, date)
date → register **Update Cascade, Delete Restrict**

Student (studentID, Name, phoneNumber, emailAddress, forename, surname, addressL1, addressL2, townCity, postcode, country, dateOfBirth, nationalInsurance, visaExpiryDate, passportNumber, visaNumber, registered, nkAddressL1, nkAddressL2, nkTownCity, nkPostcode, nkRelationship, nkPhoneNumber)

StudentCourse (courseCode, studentID, spAverageKnowledge, spProgression)
studentID → Student **Update Cascade, Delete Restrict**
courseCode →Course **Update Cascade, Delete Restrict**

Resource (resourceID, name, type, studentAvailability, sharedWith)

ResourceForCourse (resourceID, courseCode)
resourceID →Resource **Update Cascade, Delete Restrict**
courseCode → Course **Update Cascade, Delete Restrict**

Quiz (quizID, name, fillInTheBlank, trueOrFalse, multipleChoice, courseCode)
courseCode → Course **Update Cascade, Delete Restrict**

Grade (gradeID, score, completion, quizID, studentID)
quizID → Quiz **Update Cascade, Delete Restrict**
studentID → Student **Update Cascade, Delete Restrict**

register (date, time, lectureRoom, seminarRoom, labRoom, attended)

studentRegister (date, studentID)
studentID → Student **Update Cascade, Delete Restrict**
date → register **Update Cascade, Delete Restrict**

6.2 Normalisation

Normalisation of data is carried out to reduce or eliminate data redundancy, make the database simple to access and amend and to arrange data in a logical way.

UNF

Field	Value 1	Value 2
institutionName	Hope	
institutionAddressL1	32 Close	
institutionAddressL2		
institutionTownCity	Liverpool	
institutionPostcode	L15 6BE	
institutionPhoneNumber	01744 252633	
studentID	1501	
studentEmailAddress	1501@hope.ac.uk	
studentPhoneNumber	1744648259	
studentNationalInsurance	hg5944l	
studentForename	Hayley	
studentSurname	Smith	
Student AddressL1	58 Avenue	
student AddressL2		
studentTownCity	Liverpool	
student Postcode	L32 6SW	
studentDateOfBirth	05/10/1998	
studentVisaExpiryDate	15/09/2018	
studentVisaNumber	B5125410	
studentCountry	United Kingdom	
studentPassportNumber	df595222g	
studentnkForename	Peter	
studentnkSurname	Griffin	
studentnkAddressL1	58 Avenue	
studentnkAddressL2		
studentnkTownCity	Liverpool	
studentnkPostcode	L32 6SW	
studentnkRelationship	Father	
studentnkPhoneNumber	0151 895632	
registered	yes	
feeTotal	27000	
feeDuration	3	
feeAmountPaid	5000	
spAverageKnowledge	86	87
spProgression	75	80
registerDate	15/12/2016	15/12/2016
registerTime	11:00	12:00
registerLectureRoom	FML100	
registerSeminarRoom		FML400
registerLabRoom		
registerAttended	YES	YES
CourseCode	CS1651	EE8526
CourseStartDate	09/10/2015	09/10/2015
CourseEndDate	15/06/2018	15/06/2018
courseYearOfStudy	2	2
CourseCreditValue	120	120
CourseName	Computer Science	Electronic Engineering
Department	Maths and Computer Science	Robotics
resourceID	CS7523	EE9651
resourceType	Powerpoint	PDF
resourceTitle	Coding	Robotics
studentAvailability	Avaialble	unavailable
sharedWith	Electronic Engineering	
quizID	CS8520	EE4567
quizname	Portfolio 5	Robotics
questionFillInTheBlank	Question 1	Question 1
questionTrueOrFalse	Question 2	Question 2
questionMultipleChoice	Question 3	Question 3
gradeID	49876	57346
score	87	93
completion	100	100
tutorID	8520	9630
tutoremailAddress	8520@hope.ac.uk	9630@hope.ac.uk
tutorphoneNumber	0151 852456	0151 789842
tutornationalInsurance	sd4956k	gh7892u
tutorForename	Anthony	Julie
tutorSurname	Jones	Williams
tutor AddressL1	25 Drive	782 Avenue
tutorAddressL2		
tutorTownCity	Manchester	Liverpool
tutor Postcode	M45 7NE	L32 7NW
tutorCountry	United Kingdom	United Kingdom
tutorDateOfBirth	15/01/1957	20/09/1964
tutorExtensionNumber	252	500
tutorOfficeNumber	401	415
tutornkForename	Dave	Tom
tutornkSurname	Stallings	Henderson
tutornkAddressL1	582 Drive	782 Avenue
tutornkAddressL2		
tutornkTownCity	Manchester	Liverpool
tutornkPostcode	M58 7NE	L32 7NW
tutornkRelationship	Father	Uncle
tutornkPhoneNumber	0151 489752	01744 654895
administratorID	4956	1245
administratorPhoneNumber	0151753984	0151461327
administratorEmailAddress	4956@hope.ac.uk	1245@hope.ac.uk
administratorAddressL1	59 Thames Street	24 Balsham Road
administratorAddressL2		
administratorTownCity	Chester	London
administratorPostcode	IV24 2WA	DN11 7RJ
administratorForename	Christopher	Stewart
administratorSurname	Jones	Holmes
administratorDateOfBirth	01/17/1951	11/26/1956
administratornkForename	Jake	Jennifer
administratornkSurname	Dixon	Bray
administratornkAddressL1	1236 Cedar Street	82 Wressle Road
administratornkAddressL2		
administratornkTownCity	London	Manchester
administratornkPostcode	TN27 1SD	NP6 4LF
administratorCountry	United Kingdom	United Kingdom
administratornkRelationship	Cousin	Mother
administratornkPhoneNumbe	07748471590	07975870907

This UNF table is the un-normal form. This is where all attributes, fields and data are collected and placed into one table. studentID is the primary key for this table.

1NF

institutionStudentTransaction Table

Field	Value 1	Value 2
institutionName	Hope	
institutionAddressL1	32 Close	
institutionAddressL2		
institutionTownCity	Liverpool	
institutionPostcode	L15 6BE	
institutionPhoneNumber	01744 252633	
studentID	1501	
studentEmailAddress	1501@hope.ac.uk	
studentPhoneNumber	1744648259	
studentNationalInsurance	hg5944l	
studentForename	Hayley	
studentSurname	Smith	
Student AddressL1	58 Avenue	
student AddressL2		
studentTownCity	Liverpool	
student Postcode	L32 6SW	
studentDateOfBirth	05/10/1998	
studentVisaExpiryDate	15/09/2018	
studentPassportNumber	df595222g	
studentnkForename	Peter	
studentnkSurname	Griffin	
studentnkAddressL1	58 Avenue	
studentnkAddressL2		
studentnkTownCity	Liverpool	
studentnkPostcode	L32 6SW	
studentCountry	United Kingdom	
studentnkRelationship	Father	
studentnkPhoneNumber	0151 895632	
studentVisaNumber	B5125410	
registered	yes	
feeTotal	27000	
feeDuration	3	
feeAmountPaid	5000	

studentCourseTutorAdmin Table

studentID	1501	
CourseCode	CS1651	EE8526
CourseStartDate	09/10/2015	09/10/2015
CourseEndDate	15/06/2018	15/06/2018
courseYearOfStudy	2	2
CourseCreditValue	120	120
CourseName	Computer Science	Electronic Engineering
Department	Maths and Computer Science	Robotics
registerDate	15/12/2016	15/12/2016
registerTime	11:00	12:00
registerLectureRoom	FML100	
registerSeminarRoom		FML400
registerLabRoom		
registerAttended	YES	YES
resourceID	CS7523	EE9651
resourceType	Powerpoint	PDF
resourceTitle	Coding	Robotics
studentAvailability	Avaialble	unavailable
sharedWith	Electronic Engineering	
quizID	CS8520	EE4567
quizname	Portfolio 5	Robotics
questionFillInTheBlank	Question 1	Question 1
questionTrueOrFalse	Question 2	Question 2
questionMultipleChoice	Question 3	Question 3
gradeID	49876	57346
score	87	93
completion	100	100
tutorID	8520	9630
tutoremailAddress	8520@hope.ac.uk	9630@hope.ac.uk
tutorphoneNumber	0151 852456	0151 789842
tutornationalInsurance	sd4956k	gh7892u
tutorForename	Anthony	Julie
tutorSurname	Jones	Williams
tutor AddressL1	25 Drive	782 Avenue
tutorAddressL2		
tutorTownCity	Manchester	Liverpool
tutor Postcode	M45 7NE	L32 7NW
tutorDateOfBirth	15/01/1957	20/09/1964
tutorExtensionNumber	252	500
tutorOfficeNumber	401	415
tutornkForename	Dave	Tom
tutornkSurname	Stallings	Henderson
tutornkAddressL1	582 Drive	782 Avenue
tutornkAddressL2		
tutornkTownCity	Manchester	Liverpool
tutornkPostcode	M58 7NE	L32 7NW
tutorCountry	United Kingdom	United Kingdom
tutornkRelationship	Father	Uncle
tutornkPhoneNumber	0151 489752	01744 654895
administratorID	4956	1245
administratorPhoneNumber	0151753984	0151461327
administratorEmailAddress	4956@hope.ac.uk	1245@hope.ac.uk
administratorAddressL1	59 Thames Street	24 Balsham Road
administratorAddressL2		
administratorTownCity	Chester	London
administratorPostcode	IV24 2WA	DN11 7RJ
administratorForename	Christopher	Stewart
administratorSurname	Jones	Holmes
administratorDateOfBirth	01/17/1951	11/26/1956
administratornkForename	Jake	Jennifer
administratornkSurname	Dixon	Bray
administratornkAddressL1	1236 Cedar Street	82 Wressle Road
administratornkAddressL2		
administratornkTownCity	London	Manchester
administratornkPostcode	TN27 1SD	NP6 4LF
administratorCountry	United Kingdom	United Kingdom
administratornkRelationship	Cousin	Mother
administratornkPhoneNumbe	07748471590	07975870907
spAverageKnowledge	86	87
spProgression	75	80

This is the 1 NF table, first normal form. This is where repeating attributes are separated into tables. Repeating attributes is where an entity's attributes uses more than one column, for example tutorForename uses two columns and contains two attributes; Anthony and Julie. The keys studentID and courseCode form a compound key in the second table.

2NF

institutionStudentTransaction Table

Field	Value 1	Value 2
institutionName	Hope	
institutionAddressL1	32 Close	
institutionAddressL2		
institutionTownCity	Liverpool	
institutionPostcode	L15 6BE	
institutionPhoneNumber	01744 252633	
studentID	1501	
studentEmailAddress	1501@hope.ac.uk	
studentPhoneNumber	1744648259	
studentNationalInsurance	hg5944l	
studentForename	Hayley	
studentSurname	Smith	
Student AddressL1	58 Avenue	
student AddressL2		
studentTownCity	Liverpool	
student Postcode	L32 6SW	
studentDateOfBirth	05/10/1998	
studentVisaExpiryDate	15/09/2018	
studentPassportNumber	df595222g	
studentnkForename	Peter	
studentnkSurname	Griffin	
studentnkAddressL1	58 Avenue	
studentnkAddressL2		
studentnkTownCity	Liverpool	
studentnkPostcode	L32 6SW	
studentCountry	United Kingdom	
studentnkRelationship	Father	
studentnkPhoneNumber	0151 895632	
studentVisaNumber	B5125410	
registered	yes	
feeTotal	27000	
feeDuration	3	
feeAmountPaid	5000	

studentCourseTable

studentID	1501	
CourseCode	CS1651	EE8526
spAverageKnowledge	86	87
spProgression	75	80
registerDate	15/12/2016	15/12/2016
registerTime	11:00	12:00
registerLectureRoom	FML100	
registerSeminarRoom		FML400
registerAttended	YES	YES
registerLabRoom		
courseYearOfStudy	2	2

courseTable

CourseCode	CS1651	EE8526
CourseStartDate	09/10/2015	09/10/2015
CourseEndDate	15/06/2018	15/06/2018
CourseCreditValue	120	120
CourseName	Computer Science	Electronic Engineering
Department	Maths and Computer Science	Robotics
resourceID	CS7523	EE9651
resourceType	Powerpoint	PDF
resourceTitle	Coding	Robotics
studentAvailability	Avaialble	unavailable
sharedWith	Electronic Engineering	
quizID	CS8520	EE4567
quizname	Portfolio 5	Robotics
questionFillInTheBlank	Question 1	Question 1
questionTrueOrFalse	Question 2	Question 2
questionMultipleChoice	Question 3	Question 3
gradeID	49876	57346
score	87	93
completion	100	100

tutorAdmin Table

CourseCode	CS1651	EE8526
tutorID	8520	9630
tutoremailAddress	8520@hope.ac.uk	9630@hope.ac.uk
tutorphoneNumber	0151 852456	0151 789842
tutornationalInsurance	sd4956k	gh7892u
tutorForename	Anthony	Julie
tutorSurname	Jones	Williams
tutor AddressL1	25 Drive	782 Avenue
tutorAddressL2		
tutorTownCity	Manchester	Liverpool
tutor Postcode	M45 7NE	L32 7NW
tutorDateOfBirth	15/01/1957	20/09/1964
tutorExtensionNumber	252	500
tutorOfficeNumber	401	415
tutornkForename	Dave	Tom
tutornkSurname	Stallings	Henderson
tutornkAddressL1	582 Drive	782 Avenue
tutornkAddressL2		
tutornkTownCity	Manchester	Liverpool
tutornkPostcode	M58 7NE	L32 7NW
tutorCountry	United Kingdom	United Kingdom
tutornkRelationship	Father	Uncle
tutornkPhoneNumber	0151 489752	01744 654895
administratorID	4956	1245
administratorPhoneNumber	0151753984	0151461327
administratorEmailAddress	4956@hope.ac.uk	1245@hope.ac.uk
administratorPhoneNumber	0151753984	0151461327
administratorEmailAddress	4956@hope.ac.uk	1245@hope.ac.uk
administratorAddressL1	59 Thames Street	24 Balsham Road
administratorAddressL2		
administratorTownCity	Chester	London
administratorPostcode	IV24 2WA	DN11 7RJ
administratorForename	Christopher	Stewart
administratorSurname	Jones	Holmes
administratorDateOfBirth	01/17/1951	11/26/1956
administratornkForename	Jake	Jennifer
administratornkSurname	Dixon	Bray
administratornkAddressL1	1236 Cedar Street	82 Wressle Road
administratornkAddressL2		
administratornkTownCity	London	Manchester
administratornkPostcode	TN27 1SD	NP6 4LF
administratorCountry	United Kingdom	United Kingdom
administratornkRelationship	Cousin	Mother
administratornkPhoneNumbe	07748471590	07975870907

This is the 2NF table, second normal form. This is where fields are checked for dependencies on compound keys. The compound key in this normalisation is studentID and courseCode. Other fields in this table where then compared for dependencies to the two keys. If they are only partially dependent on the compound key, the field and attributes are put into another table with the key that they are fully dependent on.

3NF

institution Table

Field	Value 1	Value 2
institutionName	Hope	
institutionAddressL1	32 Close	
institutionAddressL2		
institutionTownCity	Liverpool	
institutionPostcode	L15 6BE	
institutionPhoneNumber	01744 252633	

student Table

studentID	1501	
studentEmailAddress	1501@hope.ac.uk	
studentPhoneNumber	1744648259	
studentNationalInsurance	hg5944l	
studentForename	Hayley	
studentSurname	Smith	
Student AddressL1	58 Avenue	
student AddressL2		
studentTownCity	Liverpool	
student Postcode	L32 6SW	
studentDateOfBirth	05/10/1998	
studentVisaExpiryDate	15/09/2018	
studentPassportNumber	df595222g	
studentnkForename	Peter	
studentnkSurname	Griffin	
studentnkAddressL1	58 Avenue	
studentnkAddressL2		
studentnkTownCity	Liverpool	
studentnkPostcode	L32 6SW	
studentCountry	United Kingdom	
studentnkRelationship	Father	
studentnkPhoneNumber	0151 895632	
studentVisaNumber	B5125410	
registered	yes	
feeTotal	27000	
feeDuration	3	
feeAmountPaid	5000	
institutionPostcode	L15 6BE	

studentCourseTable

studentID	1501	
CourseCode	CS1651	EE8526
spAverageKnowledge	86	87
spProgression	75	80
registerDate	15/12/2016	15/12/2016
courseYearOfStudy	2	2

register Table

registerDate	15/12/2016	15/12/2016
registerTime	11:00	12:00
registerLectureRoom	FML100	
registerSeminarRoom		FML400
registerAttended	YES	YES
registerLabRoom		

course Table

CourseCode	CS1651	EE8526
CourseStartDate	09/10/2015	09/10/2015
CourseEndDate	15/06/2018	15/06/2018
CourseCreditValue	120	120
CourseName	Computer Science	Electronic Engineering
Department	Maths and Computer Science	Robotics
resourceID	CS7523	EE9651
quizID	CS8520	EE4567
gradeID	49876	57346
registerDate	15/12/2016	15/12/2016

resource Table

resourceID	CS7523	EE9651
resourceType	Powerpoint	PDF
resourceTitle	Coding	Robotics
studentAvailability	Avaialble	unavailable
sharedWith	Electronic Engineering	

quiz Table

quizID	CS8520	EE4567
quizname	Portfolio 5	Robotics
questionFillInTheBlank	Question 1	Question 1
questionTrueOrFalse	Question 2	Question 2
questionMultipleChoice	Question 3	Question 3

grade Table

gradeID	49876	57346
score	87	93
completion	100	100

tutor Table

CourseCode	CS1651	EE8526
tutorID	8520	9630
tutoremailAddress	8520@hope.ac.uk	9630@hope.ac.uk
tutorphoneNumber	0151 852456	0151 789842
tutornationalInsurance	sd4956k	gh7892u
tutorForename	Anthony	Julie
tutorSurname	Jones	Williams
tutor AddressL1	25 Drive	782 Avenue
tutorAddressL2		
tutorTownCity	Manchester	Liverpool
tutor Postcode	M45 7NE	L32 7NW
tutorDateOfBirth	15/01/1957	20/09/1964
tutorExtensionNumber	252	500
tutorOfficeNumber	401	415
tutornkForename	Dave	Tom
tutornkSurname	Stallings	Henderson
tutornkAddressL1	582 Drive	782 Avenue
tutornkAddressL2		
tutornkTownCity	Manchester	Liverpool
tutornkPostcode	M58 7NE	L32 7NW
tutorCountry	United Kingdom	United Kingdom
tutornkRelationship	Father	Uncle
tutornkPhoneNumber	0151 489752	01744 654895
administratorID	4956	1245

administrator Table

studentID	1501	
administratorID	4956	1245
administratorPhoneNumber	0151753984	0151461327
administratorEmailAddress	4956@hope.ac.uk	1245@hope.ac.uk
administratorAddressL1	59 Thames Street	24 Balsham Road
administratorAddressL2		
administratorTownCity	Chester	London
administratorPostcode	IV24 2WA	DN11 7RJ
administratorForename	Christopher	Stewart
administratorSurname	Jones	Holmes
administratorDateOfBirth	01/17/1951	11/26/1956
administratornkForename	Jake	Jennifer
administratornkSurname	Dixon	Bray
administratornkAddressL1	1236 Cedar Street	82 Wressle Road
administratornkAddressL2		
administratornkTownCity	London	Manchester
administratornkPostcode	TN27 1SD	NP6 4LF
administratorCountry	United Kingdom	United Kingdom
administratornkRelationship	Cousin	Mother
administratornkPhoneNumbe	07748471590	07975870907

This is 3NF table, third normal form. The tables are checked for any fields which could be a possible key such as gradeID. They are then put into their own tables and leave a foreign key in the table they come from. The tables are then searched for non-key attributes which depend on other non-key attributes.

7.0 Implementation

In this section a physical design will be presented and the SQL statements will be presented along with the explanation of the code.

7.1 Physical Design

The tables below, show the physical design of the implementation stage. These tables help to identify key types, table entities, field names for the individual entities, data types and constraints for the field name.

Physical Design

Institution

Field Name	Data Type	Field Size	Format	Key Type	Required	Index	Constraint
postcode	varchar	15		primary	yes	yes (duplicates no)	unique
name	varchar	20			yes	yes	not null
addressL1	varchar	50	123 aaaaa		yes	yes	not null
addressL2	varchar	50	aaaaa		no	no	
townCity	varchar	25			yes	yes	not null
phoneNumber	integer	20		candidate	yes	yes (no duplicates)	unique

InstitutionCourse

Field Name	Data Type	Field Size	Format	Key Type	Required	Index	Constraint
postcode	varchar	15		Foreign	yes	yes (duplicates ok)	not null
courseCode	char	15	AA11111	Foreign	yes	yes (duplicates ok)	not null

FK postcode → Institution **Update Cascade, Delete Restrict**

FK courseCode → Course **Update Cascade, Delete Restrict**

Course

Field Name	Data Type	Field Size	Format	Key Type	Required	Index	Constraint
courseCode	char	15	AA11111	primary	yes	yes(no duplicates)	unique
startDate	date		11/11/1111		yes	yes	not null
endDate	date		11/11/1111		no	no	
creditValue	integer	3			yes	yes	not null
department	varchar	50			yes	yes	not null
name	varchar	50		secondary	yes	yes (duplicates ok)	not null
date	date		11/11/1111	foreign	yes	yes (duplicates ok)	not null

FK date → register **Update Cascade, Delete Restrict**

resourceForCourse

Field Name	Data Type	Field Size	Format	Key Type	Required	Index	Constraint
resourceID	smallInt			Foreign	yes	yes (duplicates ok)	not null
courseCode	char	15	AA11111	Foreign	yes	yes (duplicates ok)	not null

FK resourceID →Resource **Update Cascade, Delete Restrict**

FK courseCode → Course **Update Cascade, Delete Restrict**

resource

Field Name	Data Type	Field Size	Format	Key Type	Required	Index	Constraint
resourceID	smallInt		AutoNumber	primary	yes	yes(no duplicates)	unique
title	varchar	254		secondary	yes	yes (duplicates ok)	not null
type	varchar	20			no	no	
studentAvailability	char	1			yes- y/n	yes	not null
sharedWith	varchar	50			yes	yes	not null

quiz

Field Name	Data Type	Field Size	Format	Key Type	Required	Index	Constraint
quizID	smallInt		AutoNumber	primary	yes	yes(no duplicates)	unique
name	varchar	50		candidate	yes	yes (no duplicates)	unique
fillInTheBlank	varchar				no	no	
multipleChoice	varchar				no	no	
trueOrFalse	varchar				no	no	
courseCode	char	15	AA11111	Foreign	yes	yes (duplicates ok)	not null

FK courseCode → Course **Update Cascade, Delete Restrict**

grade

Field Name	Data Type	Field Size	Format	Key Type	Required	Index	Constraint
gradeID	smallInt		AutoNumber	primary	yes	yes(no duplicates)	unique
score	varchar				yes	no	not null
completion	varchar				yes	yes	not null
quizID	smallInt			Foreign	yes	yes (duplicates ok)	not null
studentID	smallInt			Foreign	yes	yes (duplicates ok)	not null

FK quizID → Quiz **Update Cascade, Delete Restrict**

FK studentID → Student **Update Cascade, Delete Restrict**

studentCourse

Field Name	Data Type	Field Size	Format	Key Type	Required	Index	Constraint
courseCode	char	15	AA11111	Foreign	yes	yes (duplicates ok)	not null
studentID	smallInt			Foreign	yes	yes (duplicates ok)	not null
spaverageKnowledge	varchar				yes	yes	not null
spstudentProgression	varchar				yes	yes	not null

FK studentID → Student **Update Cascade, Delete Restrict**

FK courseCode →Course **Update Cascade, Delete Restrict**

register

Field Name	Data Type	Field Size	Format	Key Type	Required	Index	Constraint
date	date		11/11/1111	primary	yes	yes(no duplicates)	unique
time	time		11:11		yes		not null
lectureRoom	varchar		aaa111	secondary	yes	yes (duplicates ok)	not null
seminarRoom	varchar		aaa111	secondary	yes	yes (duplicates ok)	not null
attended	char	1			yes - y/n	yes	not null

studentRegister

Field Name	Data Type	Field Size	Format	Key Type	Required	Index	Constraint
date	date		11/11/1111	Foreign	yes	yes (duplicates ok)	not null
studentID	smallInt			Foreign	yes	yes (duplicates ok)	not null

FK studentID → Student **Update Cascade, Delete Restrict**

FK date → register **Update Cascade, Delete Restrict**

tutorCourse

Field Name	Data Type	Field Size	Format	Key Type	Required	Index	Constraint
tutorID	smallInt		AutoNumber	Foreign	yes	yes (duplicates ok)	not null
courseCode	char	15	AA11111	Foreign	yes	yes (duplicates ok)	not null

FK tutorID→ tutor **Update Cascade, Delete Cascade**

FK courseCode → Course **Update Cascade, Delete Restrict**

tutor

Field Name	Data Type	Field Size	Format	Key Type	Required	Index	Constraint
tutorID	smallInt		AutoNumber	primary	yes	yes(no duplicates)	unique
forename	varchar	25			no	no	
surname	varchar	25		secondary	yes	yes (duplicates ok)	not null
addressL1	varchar	50	123 aaaa		yes	yes	not null
addressL2	varchar	50			no	no	
townCity	varchar	25			yes	yes	not null
postcode	varchar	15			yes	yes	not null
country	varchar	255			yes	yes	not null
officeNumber	integer	20			yes	yes	not null
extensionNumber	integer	20		secondary	yes	yes (duplicates ok)	not null
emailAddress	varchar	254	11111@aaa.com/11111@aaa.co.uk	candidate	yes	yes(no duplicates)	unique
nationalInsurance	varchar	20		candidate	yes	yes(no duplicates)	unique
dateOfBirth	date		11/11/1111		yes	no	not null
phoneNumber	integer	20		secondary	yes	yes (duplicates ok)	not null
nkForename	varchar	25			no	no	
nkSurname	varchar	25		secondary	yes	yes (duplicates ok)	not null
nkAddressL1	varchar	50	123 aaaa		yes	yes	not null
nkAddressL2	varchar	50			no	no	
nkTownCity	varchar	25			yes	yes	not null
nkPostcode	varchar	15			yes	yes	not null
nkRelationship	varchar	20			yes	no	not null
nkPhoneNumber	integer	20		secondary	yes	yes (duplicates ok)	not null
administratorID	smallInt			Foreign	yes	yes (duplicates ok)	not null

FK administrationID→Administration **Update Cascade, Delete Restrict**

student

Field Name	Data Type	Field Size	Format	Key Type	Required	Index	Constraint
studentID	smallInt		AutoNumber	primary	yes	yes(no duplicates)	unique
forename	varchar	25			yes	no	
surname	varchar	25		secondary	yes	yes (duplicates ok)	not null
addressL1	varchar	50	123 aaaa		yes	yes	not null
addressL2	varchar	50			no	no	
townCity	varchar	25			yes	yes	not null
postCode	varchar	15			yes	yes	not null
country	varchar	255			yes	yes	not null
dateOfBirth	date		11/11/1111		yes	yes	not null
emailAddress	varchar	254	11111@aaa.com/11111@aaa.co.uk	candidate	yes	yes(no duplicates)	unique
phoneNumber	integer	20		secondary	yes	yes (duplicates ok)	not null
nationalInsurance	varchar	20		candidate	yes	yes(no duplicates)	unique
visaExpiryDate	date		11/11/1111		yes	yes	not null
visaNumber	varchar	8		candidate	yes	yes	unique
passportNumber	varchar	15		candidate	yes	yes(no duplicates)	unique
registered	varchar	15			yes	yes	not null
nkForename	varchar	25			no	no	
nkSurname	varchar	25		secondary	yes	yes (duplicates ok)	not null
nkAddressL1	varchar	50	123 aaaa		yes	yes	not null
nkAddressL2	varchar	50			no	no	
nkTownCity	varchar	25			yes	yes	not null
nkPostCode	varchar	15			yes	yes	not null
nkRelationship	varchar	20			no	no	
nkPhoneNumber	integer	20			yes	yes	not null
feeTotal	decimal	7, 2			yes	yes	not null
feeDuration	varchar	2			yes	yes	not null
feeAmountPaid	decimal	7, 2			yes	yes	not null

administrator							
Field Name	Data Type	Field Size	Format	Key Type	Required	Index	Constraint
administratorID	smallInt		AutoNumber	primary	yes	yes(no duplicates)	unique
phoneNumber	integer	20		secondary	yes	yes (duplicates ok)	not null
emailAddress	varchar	254	11111@aaa.com/11111@aaa.co.uk	candidate	yes	yes(no duplicates)	unique
dateOfBirth	date		11/11/1111		yes	yes	not null
forename	varchar	25			no	no	
surname	varchar	25		secondary	yes	yes (duplicates ok)	not null
addressL1	varchar	50	123 aaaa		yes	yes	not null
addressL2	varchar	50			no	no	
townCity	varchar	25			yes	yes	not null
postCode	varchar	15			yes	yes	not null
country	varchar	255			yes	yes	not null
nkForename	varchar	25			no	no	
nkSurname	varchar	25		secondary	yes	yes (duplicates ok)	not null
nkAddressL1	varchar	50	123 aaaa		yes	yes	not null
nkAddressL2	varchar	50			no	no	
nkTownCity	varchar	25			yes	yes	not null
nkPostCode	varchar	15			yes	yes	not null
nkRelationship	varchar	20			no	no	
nkPhoneNumber	integer	20			yes	yes	not null

For most identification numbers, smallint has been chosen as they usually contain numerical characters and the maximum length of characters is 32,767. courseCode has data type CHAR() as both numerical and letters can be used for the code, this helps with identifying which subject the code belongs to.

For date, related fields the data type date has been used as this will store data in a date format. The format of this field can be changed.

For many fields varchar, has been chosen as a data type as the maximum number of characters that can be entered can be adjusted, therefore reducing storage space. For example, for email address the recommended character size is 254 as the email address can vary in character length however, for a postcode 15 has been recommended as a postcode does not usually exceed this value.

7.2 SQL- Structured Query Language

In this section of the chapter, the SQL code will be presented and SQL statements will be described.

7.2.1 SQL Statements

- CREATE DATABASE mysql– creates a database called mysql
- USE mysql – uses a specified database called mysql
- CREATE TABLE mysql – creates a table in the database called mysql
- INSERT INTO – inserts values into tables
  VALUES (); – the values to be inserted into tables
- SHOW TABLES - shows tables within the current database
- CREATE USER 'mysql'@'localhost' IDENTIFIED BY 'mysql'; - creates a user and user password

7.2.3 Code for Creating Users

```
/* creating users */
CREATE USER 'student'@'localhost'
IDENTIFIED BY 'mypass';

CREATE USER 'tutor'@'localhost'
IDENTIFIED BY 'mypass';

CREATE USER 'administrator'@'localhost'
IDENTIFIED BY 'mypass';

USE mysql;
SELECT user, password FROM user;
```

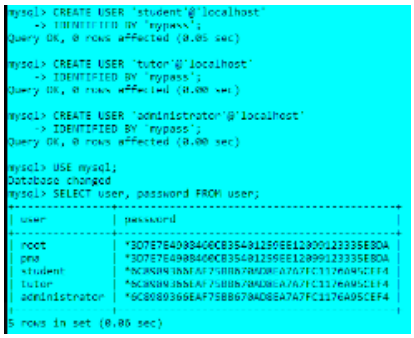


Figure 1 user table

Figure 1 on the right shows a table of users entered. This shows that the code works.

7.2.4 Code for Creating Database

```
/* creating the database */

CREATE DATABASE aceTraining;
USE aceTraining;
```

7.2.5 Code for Creating Tables

```
/* creating database tables */

CREATE TABLE institution
   (name varchar(20) not null,
    addressL1 varchar(50) not null,
    addressL2 varchar(50),
    townCity varchar(25) not null,
    postcode varchar(15) not null unique,
    phoneNumber integer(20) not null unique,

    primary key(postcode)
   );

CREATE TABLE institutionCourse
   (postcode varchar(15) not null,
    courseCode char(15) not null,

    foreign key(postcode) references institution(postcode)
    ON UPDATE CASCADE ON DELETE RESTRICT,
    foreign key(courseCode) references course(courseCode)
    ON UPDATE CASCADE ON DELETE RESTRICT
   );
```

```
CREATE TABLE course
   (courseCode char(15) not null unique,
    startDate date not null,
    endDate date,
    yearOfStudy smallint not null,
    creditValue integer(3) not null,
    department varchar(50) not null,
    name varchar(50) not null,
    date date not null,

    INDEX codes(courseCode);
    primary key(courseCode),
    foreign key(date) references register(date)
    ON UPDATE CASCADE ON DELETE RESTRICT
   );
CREATE TABLE resourceForCourse
   (resourceID smallint not null,
    courseCode char(15) not null,

    foreign key(resourceID) references resource(resourceID)
    ON UPDATE CASCADE ON DELETE RESTRICT,
    foreign key(courseCode) references course(courseCode)
    ON UPDATE CASCADE ON DELETE RESTRICT
   );
CREATE TABLE resource
   (resourceID smallint auto_increment,
    title varchar(50) not null,
    type varchar(20) not null,
    studentAvailability char(1) not null,
    sharedWith varchar(50) not null,

    primary key(resourceID)
   );
CREATE TABLE quiz
   (quizID smallint auto_increment,
    name varchar(50) not null unique,
    courseCode char(15) not null,
    fillInTheBlank text,
    multipleChoice text,
    trueOrFalse text,

    primary key(quizID),
    foreign key(courseCode) references course(courseCode)
    ON UPDATE CASCADE ON DELETE RESTRICT
   );
CREATE TABLE grade
   (gradeID smallint auto_increment,
    score varchar(4) not null,
    completion varchar(4) not null,
    studentID smallint not null,
    quizID smallint not null,

    primary key(gradeID),
    foreign key(studentID) references student(studentID)
    ON UPDATE CASCADE ON DELETE RESTRICT,
    foreign key(quizID) references quiz(quizID)
    ON UPDATE CASCADE ON DELETE RESTRICT
   );
CREATE TABLE studentCourse
   (studentID smallint not null,
    courseCode char(15) not null,
    spAverageKnowledge varchar(20) not null,
    spProgression varchar(20) not null,

    foreign key(studentID) references student(studentID)
```

```
    ON UPDATE CASCADE ON DELETE RESTRICT,
    foreign key(courseCode) references course(courseCode)
    ON UPDATE CASCADE ON DELETE RESTRICT
   );
CREATE TABLE register
   (date date not null unique,
    time time not null,
    lectureRoom varchar(20) not null,
    seminarRoom varchar(20) not null,
    labRoom varchar(20) not null,
    attended char(1) not null,

    primary key(date)
   );
CREATE TABLE studentRegister
   (date date not null,
    studentID smallint not null,

    foreign key(studentID) references student(studentID)
    ON UPDATE CASCADE ON DELETE RESTRICT,
    foreign key(date) references register(date)
    ON UPDATE CASCADE ON DELETE RESTRICT
   );
CREATE TABLE tutorCourse
   (tutorID smallint not null,
    courseCode char(15) not null,

    foreign key(tutorID) references tutor(tutorID)
    ON UPDATE CASCADE ON DELETE CASCADE,
    foreign key(courseCode) references course(courseCode)
    ON UPDATE CASCADE ON DELETE RESTRICT
   );
CREATE TABLE tutor
   (tutorID smallint auto_increment,
    forename varchar(25),
    surname varchar (25) not null,
    addressL1 varchar(50) not null,
    addressL2 varchar(50),
    townCity varchar(25) not null,
    postcode varchar(15) not null,
    country varchar(255) not null,
    officeNumber integer(20) not null,
    extensionNumber integer(20) not null,
    emailAddress varchar(254) not null unique,
    nationalInsurance varchar(20) not null unique,
    dateOfBirth date not null,
    phoneNumber integer(20) not null,
    nkForename varchar(25),
    nkSurname varchar (25) not null,
    nkAddressL1 varchar(50) not null,
    nkAddressL2 varchar(50),
    nkTownCity varchar(25) not null,
    nkPostcode varchar(15) not null,
    nkRelationship varchar(20) not null,
    nkPhoneNumber integer(20) not null,
    administratorID smallint not null unique,

    primary key(tutorID),
    foreign key(administratorId) references administrator(administratorId)
    ON UPDATE CASCADE ON DELETE RESTRICT
   );
CREATE TABLE student
   (studentID smallint auto_increment,
    forename varchar(25),
    surname varchar (25) not null,
```

```
    addressL1 varchar(50) not null,
    addressL2 varchar(50),
    townCity varchar(25) not null,
    postcode varchar(15) not null,
    country varchar(255) not null,
    dateOfBirth date not null,
    emailAddress varchar(254) not null unique,
    phoneNumber integer(20) not null,
    nationalInsurance varchar(20) not null unique,
    visaExpiryDate date not null,
    visaNumber varchar(8) not null,
    passportNumber varchar(15) not null unique,
    registered varchar(15) not null,
    nkForename varchar(25),
    nkSurname varchar (25) not null,
    nkAddressL1 varchar(50) not null,
    nkAddressL2 varchar(50),
    nkTownCity varchar(25) not null,
    nkPostcode varchar(15) not null,
    nkRelationship varchar(20) not null,
    nkPhoneNumber integer(20) not null,
    feeTotal decimal(7,2) not null,
    feeDuration varchar(2) not null,
    feeAmountPaid decimal(7,2) not null,

    primary key(studentID)
   );
CREATE TABLE administrator
   (administratorID smallint auto_increment,
    forename varchar(25),
    surname varchar (25) not null,
    emailAddress varchar(254) not null unique,
    phoneNumber integer(20) not null,
    addressL1 varchar(50) not null,
    addressL2 varchar(50),
    townCity varchar(25) not null,
    postcode varchar(15) not null,
    country varchar(255) not null,
    dateOfBirth date not null,
    nkForename varchar(25),
    nkSurname varchar (25) not null,
    nkAddressL1 varchar(50) not null,
    nkAddressL2 varchar(50),
    nkTownCity varchar(25) not null,
    nkPostcode varchar(15) not null,
    nkRelationship varchar(20) not null,
    nkPhoneNumber integer(20) not null,

    INDEX adminNames(surname);
    primary key(administratorID)
   );
```

Figure 2 below shows a list of tables in the database aceTraining. This also shows that the code works.

```
mysql> SHOW tables;
+------------------------+
| Tables_in_acetraining |
+------------------------+
| administrator          |
| course                 |
| grade                  |
| institution            |
| institutioncourse      |
| quiz                   |
| register               |
| resource               |
| resourceforcourse      |
| student                |
| studentcourse          |
| studentregister        |
| tutor                  |
| tutorcourse            |
+------------------------+
14 rows in set (0.06 sec)
```

Figure 2 tables in the database

8.0 Security and Contingency

In this chapter security and contingency will be defined. Then the security and contingency of the database will be discussed. Justifications to the grants applied are found in chapter 3.

8.1 Security

The database must be secure as personal private information of users must be stored. To restrict access to certain information, restrictions and grants can be applied to a user type that prevents the user from accessing areas of the database. For example, for student the will only be granted access to student related data, can view basic information of the tutor and will not be able to delete any data and will not be able to amend certain areas of the database such as tutor details and grades.

8.1.1 Applying restrictions to users

Restrictions are applied to users to prevent misuse of the database and keep personal details private. An administrator account is required as they will be maintaining the database.

```
/* creating users */
CREATE USER 'student'@'localhost'
IDENTIFIED BY 'mypass';

CREATE USER 'tutor'@'localhost'
IDENTIFIED BY 'mypass';

CREATE USER 'administrator'@'localhost'
IDENTIFIED BY 'mypass';
```

```
/* specifying grants for users */
/*student*/
GRANT SELECT (postcode, name, addressL1, addressL2, townCity, phoneNumber) ON
aceTraining.institution TO 'student'@'localhost';

GRANT SELECT (studentID, forename, surname, addressL1, addressL2, townCity,
postcode, country, dateOfBirth, emailAddress, phoneNumber, nationalInsurance,
visaExpiryDate, visaNumber, passportNumber, registered, nkForename, nkSurname,
nkAddressL1, nkAddressL2, nkTownCity, nkPostcode, nkRelationship, nkPhoneNumber,
feeTotal, feeDuration, feeAmountPaid), INSERT (forename, surname, addressL1,
addressL2, townCity, postcode, country, dateOfBirth, phoneNumber,
nationalInsurance, visaExpiryDate, PassportNumber, nkForename, nkSurname,
nkAddressL1, nkAddressL2, nkTownCity, nkPostcode, nkRelationship, nkPhoneNumber),
UPDATE(forename, surname, addressL1, addressL2, townCity, postcode, dateOfBirth,
phoneNumber, nationalInsurance, visaExpiryDate, visaNumber, PassportNumber,
nkForename, nkSurname, nkAddressL1, nkAddressL2, nkTownCity, nkPostcode,
nkRelationship, nkPhoneNumber) ON aceTraining.student TO 'student'@'localhost';

GRANT SELECT (spAverageKnowledge, spProgression) ON aceTraining.studentCourse TO
'student'@'localhost';

GRANT SELECT (date, time, lectureRoom, seminarRoom, labRoom, attended) ON
aceTraining.register TO 'student'@'localhost';

GRANT SELECT (courseCode, startDate, endDate, yearOfStudy, creditValue, department,
name) ON aceTraining.course TO 'student'@'localhost';

GRANT SELECT (title, type) ON aceTraining.resource TO 'student'@'localhost';

GRANT SELECT (name, fillInTheBlank, multipleChoice, trueOrFalse) ON
aceTraining.quiz TO 'student'@'localhost';

GRANT SELECT (score, completion) ON aceTraining.grade TO 'student'@'localhost';

GRANT SELECT (forename, surname, officeNumber, emailAddress, phoneNumber) ON
aceTraining.tutor TO 'student'@'localhost';
```

```
/*tutor*/
GRANT SELECT (postcode, name, addressL1, addressL2, townCity, phoneNumber) ON
aceTraining.institution TO 'tutor'@'localhost';

GRANT SELECT (studentID, forename, surname, emailAddress) ON aceTraining.student TO
'tutor'@'localhost';

GRANT SELECT (spAverageKnowledge, spProgression), INSERT (spAverageKnowledge,
spProgression), UPDATE (spAverageKnowledge, spProgression) ON
aceTraining.studentCourse TO 'tutor'@'localhost';

GRANT SELECT (date, time, lectureRoom, seminarRoom, labRoom, attended), INSERT
(date, time, lectureRoom, seminarRoom, labRoom, attended) ON aceTraining.register
TO 'tutor'@'localhost';

GRANT SELECT (courseCode, startDate, endDate, yearOfStudy, creditValue, department,
name), INSERT (courseCode, startDate, endDate, yearOfStudy, creditValue,
department, name), UPDATE (startDate, endDate, yearOfStudy, creditValue,
department, name) ON aceTraining.course TO 'tutor'@'localhost';

GRANT SELECT (title, type, studentAvailability, sharedWith), INSERT (title, type,
studentAvailability, sharedWith), UPDATE (title, type, studentAvailability,
sharedWith) ON aceTraining.resource TO 'tutor'@'localhost';

GRANT SELECT (name, fillInTheBlank, multipleChoice, trueOrFalse), INSERT (name,
fillInTheBlank, multipleChoice, trueOrFalse), UPDATE (name, fillInTheBlank,
multipleChoice, trueOrFalse) ON aceTraining.quiz TO 'tutor'@'localhost';

GRANT SELECT (score, completion), INSERT (score, completion), UPDATE (score,
completion) ON aceTraining.grade TO 'tutor'@'localhost';

GRANT SELECT (tutorID, forename, surname, addressL1, addressL2, townCity, postcode,
country, officeNumber, extensionNumber, emailAddress, nationalInsurance,
dateOfBirth, phoneNumber, nkForename, nkSurname, nkAddressL1, nkAddressL2,
nkTownCity, nkPostcode, nkRelationship, nkPhoneNumber), INSERT(forename, surname,
addressL1, addressL2, townCity, postcode, country, nationalInsurance, dateOfBirth,
phoneNumber, nkForename, nkSurname, nkAddressL1, nkAddressL2, nkTownCity,
nkPostcode, nkRelationship, nkPhoneNumber), UPDATE(forename, surname, addressL1,
addressL2, townCity, postcode, country, nationalInsurance, dateOfBirth,
phoneNumber, nkForename, nkSurname, nkAddressL1, nkAddressL2, nkTownCity,
nkPostcode, nkRelationship, nkPhoneNumber)ON aceTraining.tutor TO
'tutor'@'localhost';

GRANT SELECT(phoneNumber, emailAddress, forename, surname) ON
aceTraining.administrator TO 'tutor'@'localhost';
/*administrator*/
GRANT ALL ON aceTraining.* TO 'administrator'@'localhost';
```

Figures 3 and 4 show the code for grants and that they have been successfully entered.

Figure 3 Grants applied in code

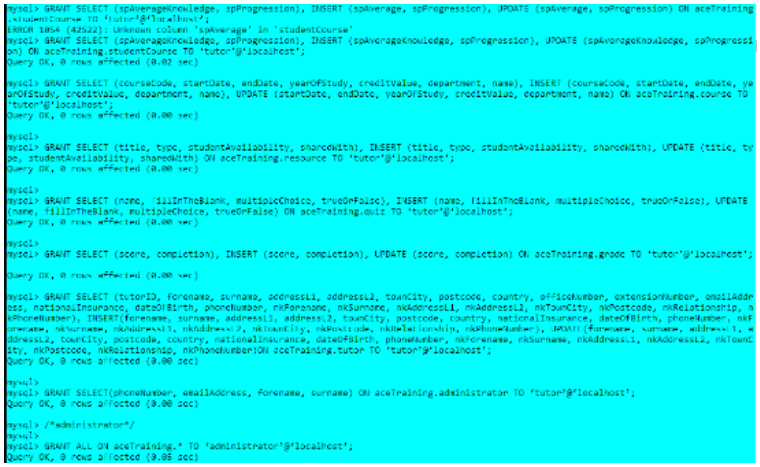


Figure 4 Grants applied in code

8.1.2 Testing Security

Student User Testing

```
INSERT INTO student
   (studentID)
VALUES
   (1501);

SELECT nationalInsurance FROM student;

SELECT spAverageKnowledge FROM studentCourse;

SELECT date FROM register;

SELECT startDate FROM course;

SELECT title FROM resource;

SELECT fillInTheBlank FROM quiz;

SELECT score FROM grade;

SELECT officeNumber FROM tutor;
```

Administrator User Testing

```
SELECT nationalInsurance FROM student;
INSERT INTO student
   (nationalInsurance)
VALUES
   ('PB48526G');

SELECT spAverageKnowledge FROM studentCourse;
INSERT INTO studentCourse
   (spAverageKnowledge)
VALUES
   ('100');

SELECT date FROM register;
INSERT INTO register
   (time)
VALUES
   (1000);

SELECT startDate FROM course;
INSERT INTO course
   (courseName)
VALUES
   ('Mathematics');

SELECT fillInTheBlank FROM quiz;
INSERT INTO quiz
   (fillInTheBlank)
VALUES
   ('Qustion10');

SELECT score FROM grade;
INSERT INTO grade
   (score)
VALUES
   ('10');
```

```
SELECT officeNumber FROM tutor;
INSERT INTO tutor
   (phoneNumber)
VALUES
   ('0548745210');

SELECT name FROM institution;
INSERT INTO institution
   (phoneNumber)
VALUES
   ('4785120500');

SELECT title FROM resource;
INSERT INTO resource
   (title)
VALUES
   ('maths_test');

SELECT phoneNumber FROM administrator;
INSERT INTO administrator
   (phoneNumber)
VALUES
   ('7552105230');
SELECT postcode FROM institutionCourse;
SELECT courseCode FROM resourceForCourse;
SELECT date FROM studentRegister;
SELECT courseCode FROM tutorCourse;
```

Tutor User Testing

```
SELECT name FROM institution;

SELECT nationalInsurance FROM student;

SELECT forename FROM student;

SELECT spAverageKnowledge FROM studentCourse;
INSERT INTO studentCourse
   (SpProgression)
VALUES
   ('75');

INSERT INTO register
   (lectureRoom, attended)
VALUES
   ('FML100', 'y');
SELECT lectureRoom, attended FROM register;

INSERT INTO register
   (lectureRoom)
VALUES
   ('FML100');
SELECT lectureRoom FROM register;

INSERT INTO resource
   (type, title)
VALUES
   ('Powerpoint', 'Coding');

SELECT title FROM resource;
INSERT INTO quiz
   (name)
VALUES
   ('Portfolio 5');
SELECT name FROM quiz;
```

```
INSERT INTO grade
   (score, completion)
VALUES
   ('93', '100');
SELECT score, completion FROM grade;

INSERT INTO tutor
   (forename, surname)
VALUES
   ('Anthony', 'Jones');

SELECT forename FROM tutor;

SELECT phoneNumber FROM administrator;
```

Evidence of these tests working for each user could not be acquired due to errors in the programming environment, as shown below in figure 5.

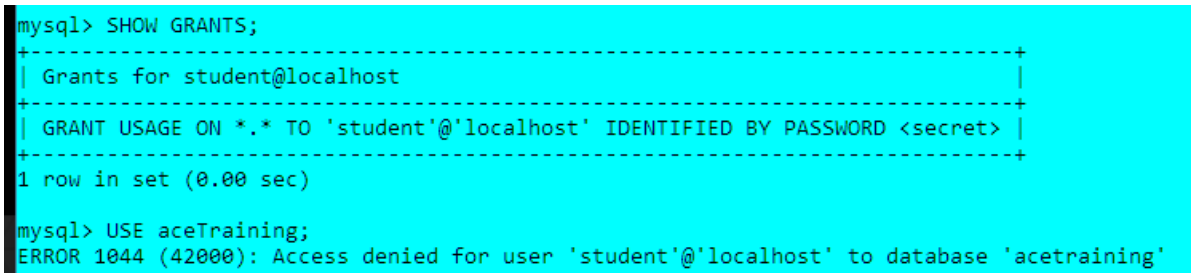


Figure 5 error message

8.2 Contingency

Introduction

Contingency is a future event which can occur but cannot be predicted. Contingency is important as any issues can be found. This helps to create solutions to these issues before the issue occurs, or possibly remove the issue. Such as possible hard drive failure, solution would be to use better quality hard drives or store spare hard drives.

8.2.1 Data back-up methods

Magnetic Tape

A tape drive is a storage device that uses magnetic tape, to store data. This storage method is slow as the magnetic tape must wind onto reels.

Advantages:

- Tape cartridges can be added as many times as required
- This method has a low power consumption and does not require low temperatures to function

Disadvantages:

- Cannot be shared on a network
- Cannot be recycled
- This method is slow
- The tape is very fragile

RAID

In this storage option, there are different levels, however depending on the level the data is stored on storage devices in several ways. The higher the RAID level the less failures and read errors that will occur. There are seven levels ranging from zero to six.

Advantages:

- If a drive fails it can be switched for another whilst the system is still on
- Reading and writing of data is done simultaneously
- A fast method for accessing data.

Disadvantages:

- Does not keep data secure
- Requires higher power consumption and low temperatures to function
- This method is complex to maintain

Cloud Based

In this storage option, data is stored in an off-site data centre. This data is accessed by the internet. This storage option allows for any stored data to be accessed securely from any location. This option is usually subscription based.

Advantages:

- Data can be encrypted
- No maintenance of hardware is required

Disadvantages:

- This method is slow to do large data backups
- Data is not stored on site
- Data may not be secure

Off-Site

In this storage option, data is stored in a similar way to the cloud based storage. This option can be built by the company, therefore removing subscription options. Off-site storage is also known as vaulting as data is transferred to the servers as a backup, for data recovery.

Advantages:
- Data could be more secure than previous options
- Data transfers could be much faster and reliable
- Data is easier to share

Disadvantages:
- Can require high power consumptions and low temperature conditions to function at optimal levels depending on hardware used
- Connection to the site is required, which could cost more due to distance and required bandwidth.

Network

Advantages:
- Can be done on site
- Can be much faster and reliable
- Much easier to share data between departments and buildings
- Can be more secure

Disadvantages:
- Could require low temperature conditions which could be hard to accomplish on site.
- The network could be compromised; therefore, people could access the data.

The recommended backup method

The recommended backup method for this database is a network based storage. This is because this method can be easily maintained and accessed. It is much faster and more reliable than other storage methods as the data can be accessed as soon as the user logs into the database.

8.2.2 Operation

For small-scale use, such as a small business, an on-site back up method would be recommended as this can be accessed on the network and is more cost effective. Back-up frequency should be daily as the data changes would not be as often as a large business.

For a large-scale use, such as a large business, an off-site back up method would be recommended as the storage can be expanded. Back-Up frequency should be hourly as the data changes will be quite high.

8.2.3 Cost Implications

The cost implications to backing up data are:
- Failures, this requires possible new hardware and technicians to fix any faults.
- Some back up methods require high power consumption and low temperature conditions, so causing high electricity bills.
- Technicians are required to maintain the storage.

9.0 Testing

Introduction

Testing is where the database is tested for any faults, such as students being able to delete records. This testing is done to keep the database secure and personal details private from other users. The grants for each user will be tested

9.1 Student User Testing

In chapter 12, appendix, there will be diagrams showing code successfully entered for creating the database, users, tables and grants and the code used to create the database.

```
1. GRANT SELECT (postcode, name, addressL1, addressL2, townCity, phoneNumber) ON
   aceTraining.institution TO 'student'@'localhost';

SELECT name FROM institution;

2. GRANT SELECT (studentID, forename, surname, addressL1, addressL2, townCity,
   postcode, country, dateOfBirth, emailAddress, phoneNumber, nationalInsurance,
   visaExpiryDate, visaNumber, passportNumber, registered, nkForename, nkSurname,
   nkAddressL1, nkAddressL2, nkTownCity, nkPostcode, nkRelationship, nkPhoneNumber,
   feeTotal, feeDuration, feeAmountPaid), INSERT (forename, surname, addressL1,
   addressL2, townCity, postcode, country, dateOfBirth, phoneNumber,
   nationalInsurance, visaExpiryDate, PassportNumber, nkForename, nkSurname,
   nkAddressL1, nkAddressL2, nkTownCity, nkPostcode, nkRelationship,
   nkPhoneNumber), UPDATE(forename, surname, addressL1, addressL2, townCity,
   postcode, dateOfBirth, phoneNumber, nationalInsurance, visaExpiryDate,
   visaNumber, PassportNumber, nkForename, nkSurname, nkAddressL1, nkAddressL2,
   nkTownCity, nkPostcode, nkRelationship, nkPhoneNumber) ON aceTraining.student TO
   'student'@'localhost';

INSERT INTO student
   (studentID)
VALUES
   (1501);

SELECT nationalInsurance FROM student;

3. GRANT SELECT (spAverageKnowledge, spProgression) ON aceTraining.studentCourse TO
   'student'@'localhost';

SELECT spAverageKnowledge FROM studentCourse;

4. GRANT SELECT (date, time, lectureRoom, seminarRoom, labRoom, attended) ON
   aceTraining.register TO 'student'@'localhost';

SELECT date FROM register;

5. GRANT SELECT (courseCode, startDate, endDate, yearOfStudy, creditValue,
   department, name) ON aceTraining.course TO 'student'@'localhost';

SELECT startDate FROM course;

6. GRANT SELECT (title, type) ON aceTraining.resource TO 'student'@'localhost';

SELECT title FROM resource;

7. GRANT SELECT (name, fillInTheBlank, multipleChoice, trueOrFalse) ON
   aceTraining.quiz TO 'student'@'localhost';

SELECT fillInTheBlank FROM quiz;
```

```
8. GRANT SELECT (score, completion) ON aceTraining.grade TO 'student'@'localhost';
SELECT score FROM grade;

9. GRANT SELECT (forename, surname, officeNumber, emailAddress, phoneNumber) ON
   aceTraining.tutor TO 'student'@'localhost';
SELECT officeNumber FROM tutor;
```

9.1.1 Expectations

The expectations from the test code used for student are:

- Test number two should display an error as the student has not be granted access to insert students, however the select code should work.
- The rest of the tests should work successfully.

9.2 Administrator User Testing

```
10.   GRANT ALL ON aceTraining.* TO 'administrator'@'localhost';

SELECT nationalInsurance FROM student;
INSERT INTO student
   (nationalInsurance)
VALUES
   ('PB48526G');

SELECT spAverageKnowledge FROM studentCourse;
INSERT INTO studentCourse
   (spAverageKnowledge)
VALUES
   ('100');

SELECT date FROM register;
INSERT INTO register
   (time)
VALUES
   (1000);

SELECT startDate FROM course;
INSERT INTO course
   (courseName)
VALUES
   ('Mathematics');

SELECT fillInTheBlank FROM quiz;
INSERT INTO quiz
   (fillInTheBlank)
VALUES
   ('Qustion10');

SELECT score FROM grade;
INSERT INTO grade
   (score)
VALUES
   ('10');

SELECT officeNumber FROM tutor;
INSERT INTO tutor
   (phoneNumber)
VALUES
   ('0548745210');

SELECT name FROM institution;
INSERT INTO institution
   (phoneNumber)
VALUES
   ('4785120500');
```

```
SELECT title FROM resource;

INSERT INTO resource
   (title)
VALUES
   ('maths_test');

SELECT phoneNumber FROM administrator;
INSERT INTO administrator
   (phoneNumber)
VALUES
   ('7552105230');

SELECT postcode FROM institutionCourse;
SELECT courseCode FROM resourceForCourse;
SELECT date FROM studentRegister;
SELECT courseCode FROM tutorCourse;
```

9.2.1 Expectations

The expectations from the test code used for administrator are:

- All the tests should work successfully, as the administrator has full access to the database.

9.3 Tutor User Testing

```
11.   GRANT SELECT (postcode, name, addressL1, addressL2, townCity, phoneNumber) ON
   aceTraining.institution TO 'tutor'@'localhost';

SELECT name FROM institution;

12.   GRANT SELECT (studentID, forename, surname, emailAddress) ON
   aceTraining.student TO 'tutor'@'localhost';

SELECT nationalInsurance FROM student;
SELECT forename FROM student;


13.   GRANT SELECT (spAverageKnowledge, spProgression), INSERT (spAverageKnowledge,
   spProgression), UPDATE (spAverageKnowledge, spProgression) ON
   aceTraining.studentCourse TO 'tutor'@'localhost';

SELECT spAverageKnowledge FROM studentCourse;
INSERT INTO studentCourse
   (SpProgression)
VALUES
   ('75');

14.   GRANT SELECT (date, time, lectureRoom, seminarRoom, labRoom, attended),
   INSERT (date, time, lectureRoom, seminarRoom, labRoom, attended) ON
   aceTraining.register TO 'tutor'@'localhost';

INSERT INTO register
   (lectureRoom, attended)
VALUES
   ('FML100', 'y');
SELECT lectureRoom, attended FROM register;

15.   GRANT SELECT (courseCode, startDate, endDate, yearOfStudy, creditValue,
   department, name), INSERT (courseCode, startDate, endDate, yearOfStudy,
   creditValue, department, name), UPDATE (startDate, endDate, yearOfStudy,
   creditValue, department, name) ON aceTraining.course TO 'tutor'@'localhost';

UPDATE course
SET startDate= 09/09/17
WHERE courseCode= ''
```

```
16.   GRANT SELECT (title, type, studentAvailability, sharedWith), INSERT (title,
   type, studentAvailability, sharedWith), UPDATE (title, type,
   studentAvailability, sharedWith) ON aceTraining.resource TO 'tutor'@'localhost';

INSERT INTO resource
   (type, title)
VALUES
   ('Powerpoint', 'Coding');

SELECT title FROM resource;

17.   GRANT SELECT (name, fillInTheBlank, multipleChoice, trueOrFalse), INSERT
   (name, fillInTheBlank, multipleChoice, trueOrFalse), UPDATE (name,
   fillInTheBlank, multipleChoice, trueOrFalse) ON aceTraining.quiz TO
   'tutor'@'localhost';

INSERT INTO quiz
   (name)
VALUES
   ('Portfolio 5');

SELECT name FROM quiz;

18.   GRANT SELECT (score, completion), INSERT (score, completion), UPDATE (score,
   completion) ON aceTraining.grade TO 'tutor'@'localhost';

INSERT INTO grade
   (score, completion)
VALUES
   ('93', '100');

SELECT score, completion FROM grade;

19.   GRANT SELECT (tutorID, forename, surname, addressL1, addressL2, townCity,
   postcode, country, officeNumber, extensionNumber, emailAddress,
   nationalInsurance, dateOfBirth, phoneNumber, nkForename, nkSurname, nkAddressL1,
   nkAddressL2, nkTownCity, nkPostcode, nkRelationship, nkPhoneNumber),
   INSERT(forename, surname, addressL1, addressL2, townCity, postcode, country,
   nationalInsurance, dateOfBirth, phoneNumber, nkForename, nkSurname, nkAddressL1,
   nkAddressL2, nkTownCity, nkPostcode, nkRelationship, nkPhoneNumber),
   UPDATE(forename, surname, addressL1, addressL2, townCity, postcode, country,
   nationalInsurance, dateOfBirth, phoneNumber, nkForename, nkSurname, nkAddressL1,
   nkAddressL2, nkTownCity, nkPostcode, nkRelationship, nkPhoneNumber)ON
   aceTraining.tutor TO 'tutor'@'localhost';

INSERT INTO tutor
   (forename, surname)
VALUES
   ('Anthony', 'Jones');

SELECT forename FROM tutor;

20.   GRANT SELECT(phoneNumber, emailAddress, forename, surname) ON
   aceTraining.administrator TO 'tutor'@'localhost';

SELECT phoneNumber FROM administrator;
```

9.3.1 Expectations

The expectations from the test code used for tutor are:

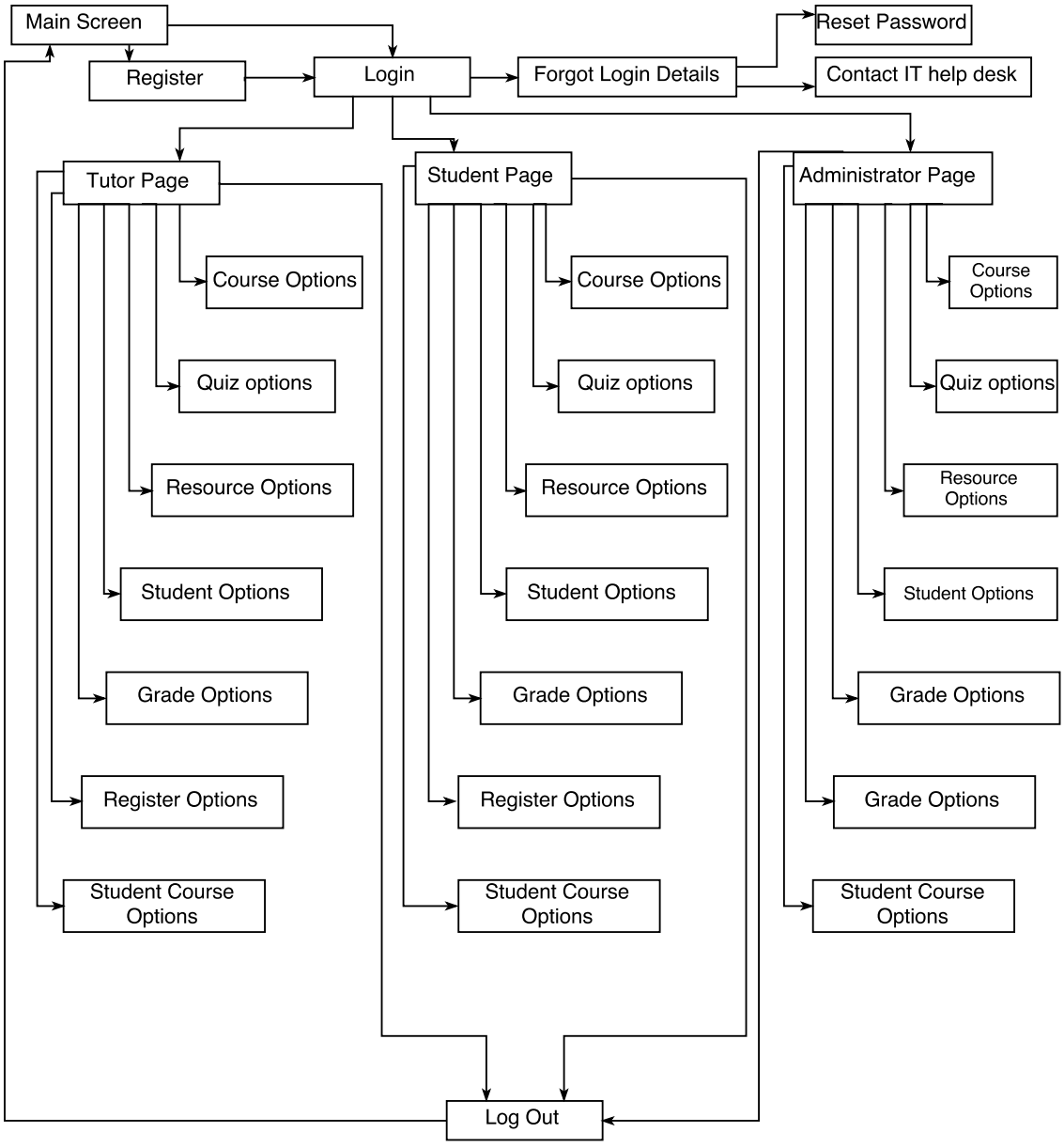
- The first select code for test code 12 should fail, as the tutor is not granted access to view the student's national insurance, however they can view their forename.
- The rest of the tests should work successfully.

10.0 Graphical User Interface

10.1 Introduction

This chapter is about a graphical user interface for the database. In this chapter the interface will be described and diagrams will show how it is formed. This stage is important as it will be a method of how users access the database, if it does not function properly the database will be unusable.

10.2 Navigation



10.2.1 Navigation Diagram Explanation

The navigation diagram in chapter 10.2 is used to show how the database can be navigated by users. In the tutor login section, the tutor will be able to view and insert in all options. the tutor can update some options. the tutor can navigate into each option and return to the tutor page.
In the student login section, the student can only view all options and navigate back to the student homepage. The student can insert in student options, for student details.

In the administrator login page, the administrator can view, insert, update and delete in all options and can view both login sections. The administrator has this ability as they must maintain the database and fix and faults found.

10.3 Web Page design

In this section the design of the graphical user interface will be shown and described.

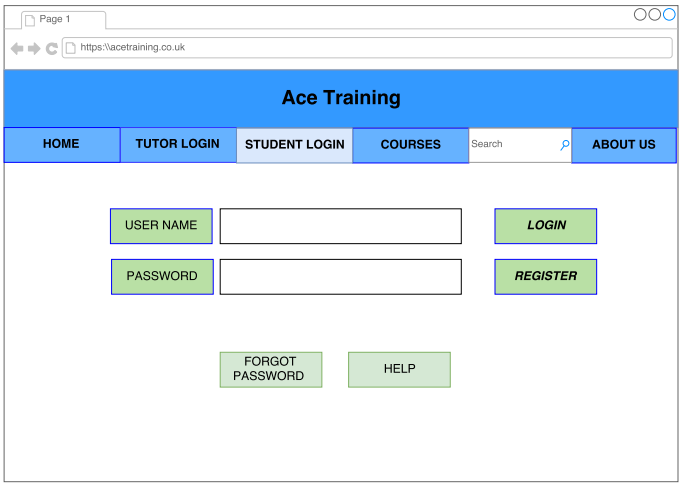


Figure 6 student login page

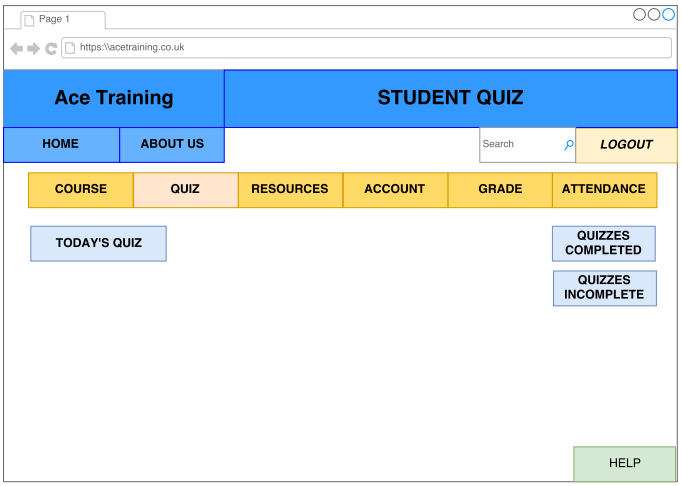


Figure 7 student quiz page

In figure 6 above the login page is shown for students. This is simple and clear, therefore not confusing the user on what to do. However, a help option and forgot password option are present if the user is experiencing any problems. in this login page the student can login to their student account, change their password, the user can register as a student and the user can acquire help if needed.

In figure 7 on the right a section of the student's page is shown. Here the student can navigate to view their course, account details, grades achieved and more. The student can amend the account details. Figure 7 shows the quiz section of the student page, here the student can access the current quiz, view old quizzes and any quizzes they have not completed.

The student has an option to logout of their account and access help if they experience issues. This webpage clearly indicates which section of the student page is accessed and other options are clearly highlighted.

In figure 8 the student registration page is shown. This shows how the data capture forms will be used and displayed. The page is clearly laid out and is not congested. There is a search bar present, this can be used to search the website. Drop down menus are used to help quickly find the selection. Fields have been clearly marked with an asterisk, stating that they are required to be filled in before continuing. There is a back option if the user needs to quit or alter previous information.

Options for home, help and about us are present this improves navigation and ease of use for the user. The about us page will allow the user to view the contact details of Ace Training, improving communication.

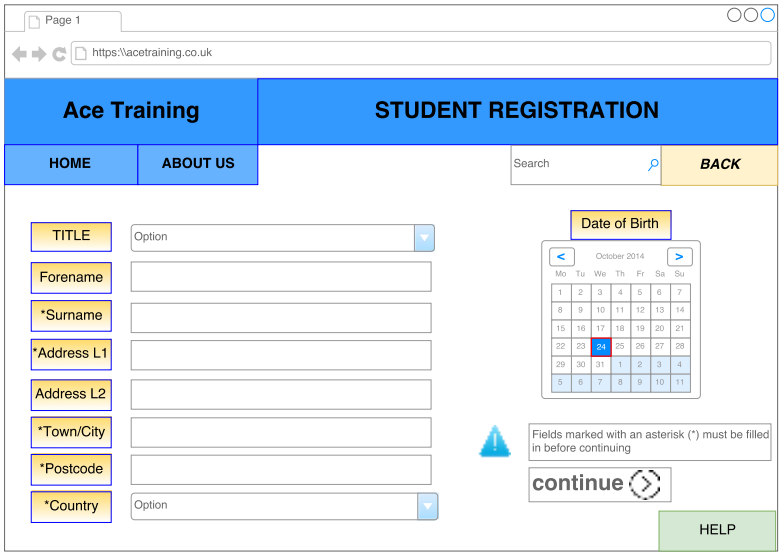


Figure 8 student registration page

In figure 9 below, the diagram shows a query being made. The tutor is searching the database for administrators. This then displayed the administrators name with their contact number. The administrator names could also become links to a webpage displaying the viewable details of the administrator such as their office location.

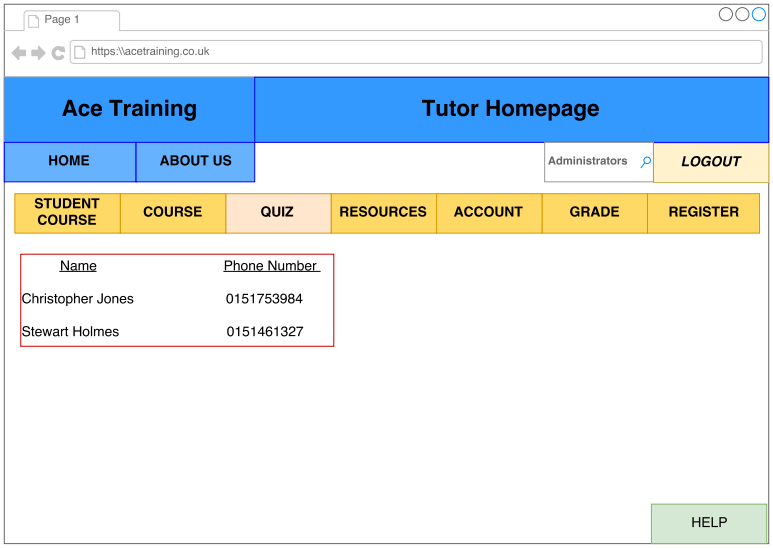


Figure 9 query

10.4 Data Capture

Below are data capture forms. These forms are used to collect data for the database.

Tutor Information

Forename	
*Surname	
*AddressL1	
AddressL2	
*Town/City	
*Postcode	
*Country	
Date of Birth	
*Email Address	
*Phone Number	
*National Insurance	
Office Number	
Extension Number	
Course	

Next Of Kin Details

Forename	
*Surname	
*Relationship	
*Phone Number	
*Address L1	
Address L2	
*Town/City	
*Postcode	

* Required fields

Administrator Information

Forename	
*Surname	
*AddressL1	
AddressL2	
*Town/City	
*Postcode	
Date of Birth	
*Email Address	
*Phone Number	
*National Insurance	

Next Of Kin Details

Forename	
*Surname	
*Relationship	
*Phone Number	
*Address L1	
Address L2	
*Town/City	
*Postcode	

* Required fields

Institution Information

Name	
*Phone Number	
*AddressL1	
AddressL2	
*Town/City	
*Postcode	

* Required fields

Course Information

*CourseCode	
*Start Date	
End Date	
*Credit Value	
Year of Study	
*Department	
*Course Name	

* Required fields

Student Course Information

*Course Code	
*Student ID	
*Student Progression	
*Student Average Knowledge	

* Required fields

Register Information

*CourseCode	
*Date	
*Time	
Lecture Room	
Seminar Room	
Lab Room	
*Student Attended	

* Required fields

Grade Information

*quiz ID	
*Score	
*Completion	
*Student ID	

* Required fields

Student Information

Forename	
*Surname	
*AddressL1	
AddressL2	
*Town/City	
*Postcode	
*Country	
Date of Birth	
*Email Address	
*Phone Number	
*National Insurance	
Visa Expiry Date	
Visa Number	
Passport Number	

Next Of Kin Details

Forename	
*Surname	
*Relationship	
*Phone Number	
*Address L1	
Address L2	
*Town/City	
*Postcode	
*Course 1	
Course 2	

* Required fields

11.0 References

Bibliography

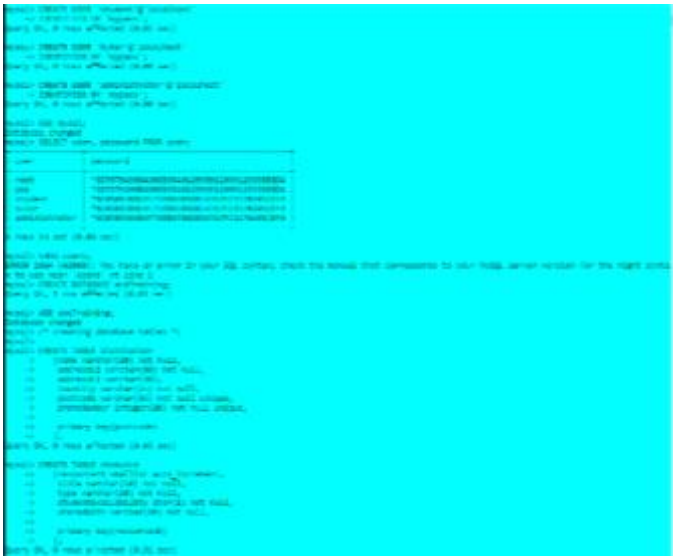
Arora, K. (1975) *Network model and their advantages and disadvantages | data models* [online]. Available from: <http://dbmsenotes.blogspot.co.uk/2014/03/comparison-of-data-models-data-models.html> [accessed 4 October 2016].
databasedev, 2015 (2003) *Flat file database design vs. Relational database design* [online]. Available from: <http://www.databasedev.co.uk/flatfile-vs-rdbms.html> [accessed 4 October 2016].
Database models - hierarchical model (2001) [online] sales@atpl.net.au. Available from: <https://sielearning.tafensw.edu.au/toolboxes/Database_Administration/content/models/hierarchical_model.htm> [accessed 14 October 2016].
Divestopedia and Institute, S. (2016) *What is a flat file database? - definition from Techopedia* [online] Techopedia.com. Available from: <https://www.techopedia.com/definition/7231/flat-file-database-database> [accessed 4 October 2016].
Flat file database (2016) In: *Wikipedia*, vol. Wikimedia Foundation
Hierarchical database model (2016) In: *Wikipedia*, vol. Wikimedia Foundation
Hierarchical data model (n.d.) [online] Database Management. Available from: <http://databasemanagement.wikia.com/wiki/Category:Hierarchical_Data_Model> [accessed 4 October 2016].
Martin, A. (2016) *Disadvantages of a relational database* [online] Techwalla. Available from: <https://www.techwalla.com/articles/disadvantages-of-a-relational-database> [accessed 5 October 2016].
Name (2010) *The evolution of database* [online] All About Databases. Available from: <https://mhaadi.wordpress.com/2010/10/18/the-evolution-of-database/> [accessed 7 October 2016].
Network model (2015) In: *Wikipedia*, vol. Wikimedia Foundation
Obasanjo, D. (2001) *Why Aren't you using an object oriented database management system?* [online]. Available from: <http://www.25hoursaday.com/whyarentyouusinganoodbms.html> [accessed 5 October 2016].
Relational model (2016) In: *Wikipedia*, vol. Wikimedia Foundation
Taylor, M. (2003) Hierarchical data security in a query-by-example interface for a shared database. *Journal of biomedical informatics.* [online], 35(3), pp.171–7. Available from: <https://www.ncbi.nlm.nih.gov/pubmed/12669980> [accessed 7 October 2016].
Teach-ICT A level computing OCR exam board - features of flat file database (n.d.) [online]. Available from: <http://www.teach-ict.com/as_as_computing/ocr/H447/F453/3_3_9/database_design/miniweb/pg5.htm> [accessed 4 October 2016].
Teach-ICT A level computing OCR exam board - relational database advantage (n.d.) [online]. Available from: <http://www.teach-ict.com/as_as_computing/ocr/H447/F453/3_3_9/database_design/miniweb/pg8.htm> [accessed 5 October 2016].
Thakur, D. (n.d.) *What is object oriented database (OODB)? Advantages and disadvantages of OODBMSS* [online]. Available from: <http://ecomputernotes.com/database-system/adv-database/object-oriented-database-oodb> [accessed 5 October 2016].
The Editors of Encyclopædia Britannica (2016) Database | computer science, In: *Encyclopædia Britannica*, vol. Encyclopædia Britannica
Tuffill, S. (2016) *Advantages & disadvantages of flat file databases* [online] Techwalla. Available from: <https://www.techwalla.com/articles/advantages-disadvantages-of-flat-file-databases> [accessed 4 October 2016].

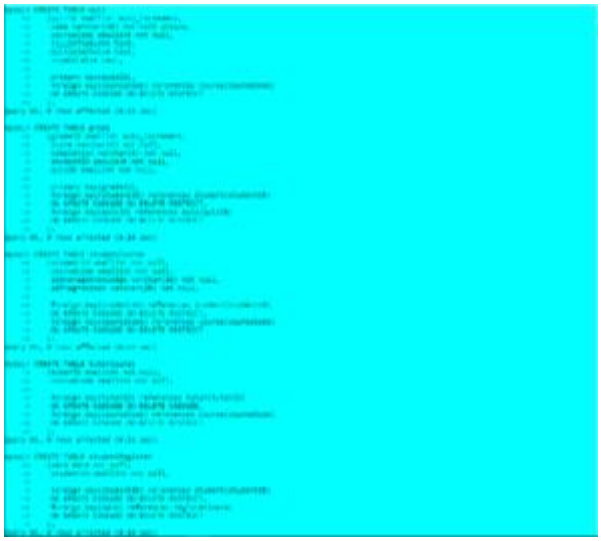
Citations, Quotes & Annotations

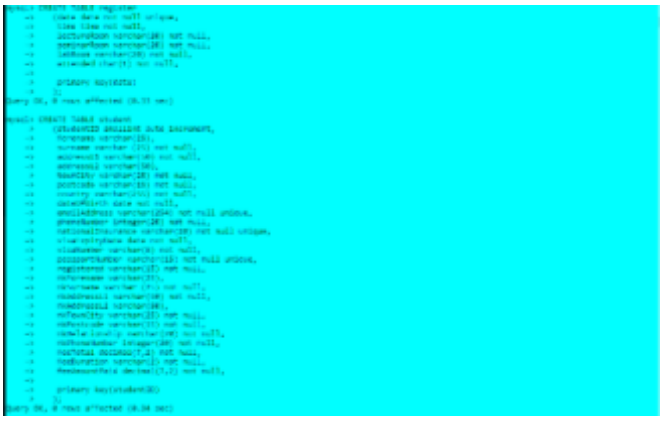
Arora, K. (1975) *Network model and their advantages and disadvantages | data models* [online]. Available from: <http://dbmsenotes.blogspot.co.uk/2014/03/comparison-of-data-models-data-models.html> [accessed 4 October 2016].

(Arora, 1975)
databasedev, 2015 (2003) *Flat file database design vs. Relational database design* [online]. Available from: <http://www.databasedev.co.uk/flatfile-vs-rdbms.html> [accessed 4 October 2016].
(databasedev, 2003)
Database models - hierarchical model (2001) [online] sales@atpl.net.au. Available from: <https://sielearning.tafensw.edu.au/toolboxes/Database_Administration/content/models/hierarchical_model.htm> [accessed 14 October 2016].
(*Database models - hierarchical model*, 2001)
Divestopedia and Institute, S. (2016) *What is a flat file database? - definition from Techopedia* [online] Techopedia.com. Available from: <https://www.techopedia.com/definition/7231/flat-file-database-database> [accessed 4 October 2016].
(Divestopedia and Institute, 2016)
"flat file databases are used internally by various computer applications to store data related to configuration. Most of the applications permit users to store and retrieve information from flat files based on a predefined set of fields." (Divestopedia and Institute, 2016)
Flat file database (2016) In: *Wikipedia*, vol. Wikimedia Foundation
(*Flat file database*, 2016)
"conceived the idea that data could be represented by holes punched in paper cards then tabulated by machine." (*Flat file database*, 2016)
Hierarchical database model (2016) In: *Wikipedia*, vol. Wikimedia Foundation
(*Hierarchical database model*, 2016)
Hierarchical data model (n.d.) [online] Database Management. Available from: <http://databasemanagement.wikia.com/wiki/Category:Hierarchical_Data_Model> [accessed 4 October 2016].
(*Hierarchical data model*, n.d.)
Martin, A. (2016) *Disadvantages of a relational database* [online] Techwalla. Available from: <https://www.techwalla.com/articles/disadvantages-of-a-relational-database> [accessed 5 October 2016].
(Martin, 2016)
Name (2010) *The evolution of database* [online] All About Databases. Available from: <https://mhaadi.wordpress.com/2010/10/18/the-evolution-of-database/> [accessed 7 October 2016].
(Name, 2010)
Network model (2015) In: *Wikipedia*, vol. Wikimedia Foundation
(*Network model*, 2015)
Obasanjo, D. (2001) *Why Aren't you using an object oriented database management system?* [online]. Available from: <http://www.25hoursaday.com/whyarentyouusinganoodbms.html> [accessed 5 October 2016].
(Obasanjo, 2001)
"a specific language using a specific API" (Obasanjo, 2001)
"Ajou University Medical Center in South Korea uses InterSystems' Cach¡§¡§ ODBMS to support all hospital functions including mission-critical departments" (Obasanjo, 2001)
"The Chicago Stock Exchange manages stock trades via a Versant ODBMS." (Obasanjo, 2001)
Relational model (2016) In: *Wikipedia*, vol. Wikimedia Foundation
(*Relational model*, 2016)
"In the relational model of a database, all data is represented in terms of tuples, grouped into relations" (*Relational model*, 2016)
Taylor, M. (2003) Hierarchical data security in a query-by-example interface for a shared database. *Journal of biomedical informatics.* [online], 35(3), pp.171–7. Available from: <https://www.ncbi.nlm.nih.gov/pubmed/12669980> [accessed 7 October 2016].
(Taylor, 2003)
"The security module ensures that researchers working in one clinic do not get access to data from another clinic. The security can be based on a flexible taxonomy structure that allows ordinary users to access data from individual clinics and super users to access data from all clinics." (Taylor, 2003)
Teach-ICT A level computing OCR exam board - features of flat file database (n.d.) [online]. Available from: <http://www.teach-ict.com/as_as_computing/ocr/H447/F453/3_3_9/database_design/miniweb/pg5.htm> [accessed 4 October 2016].
(*Teach-ICT A level computing OCR exam board - features of flat file database*, n.d.)
Teach-ICT A level computing OCR exam board - relational database advantage (n.d.) [online]. Available from: <http://www.teach-ict.com/as_as_computing/ocr/H447/F453/3_3_9/database_design/miniweb/pg8.htm> [accessed 5 October 2016].
(*Teach-ICT A level computing OCR exam board - relational database advantage*, n.d.)
Thakur, D. (n.d.) *What is object oriented database (OODB)? Advantages and disadvantages of OODBMSS* [online]. Available from: <http://ecomputernotes.com/database-system/adv-database/object-oriented-database-oodb> [accessed 5 October 2016].
(Thakur, n.d.)
"There is no universally agreed data model" (Thakur, n.d.)
The Editors of Encyclopædia Britannica (2016) Database | computer science, In: *Encyclopædia Britannica*, vol. Encyclopædia Britannica
(The Editors of Encyclopædia Britannica, 2016)
Tuffill, S. (2016) *Advantages & disadvantages of flat file databases* [online] Techwalla. Available from: <https://www.techwalla.com/articles/advantages-disadvantages-of-flat-file-databases> [accessed 4 October 2016].
(Tuffill, 2016)
(n.d.) [online]. Available from: <https://en.oxforddictionaries.com/definition/contingency> [accessed 11 December 2016].
(n.d.)

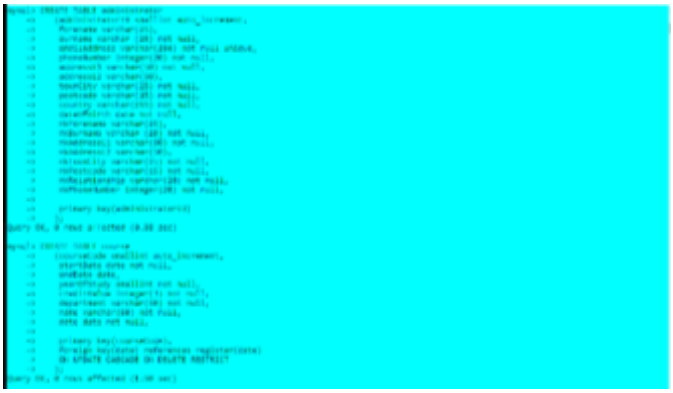
12.0 Appendices

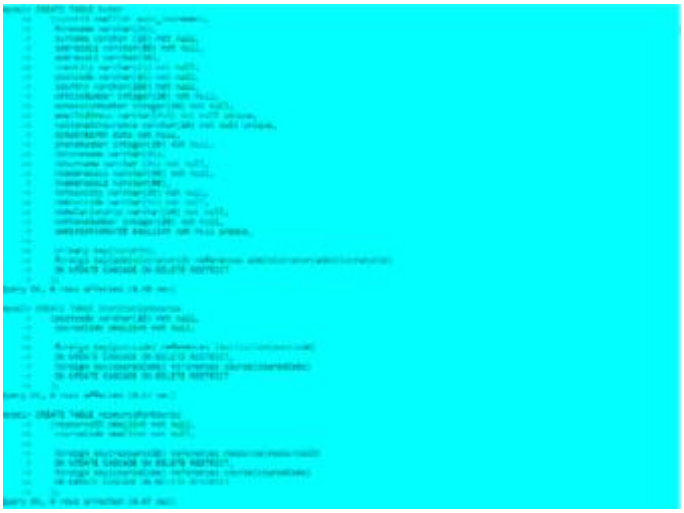


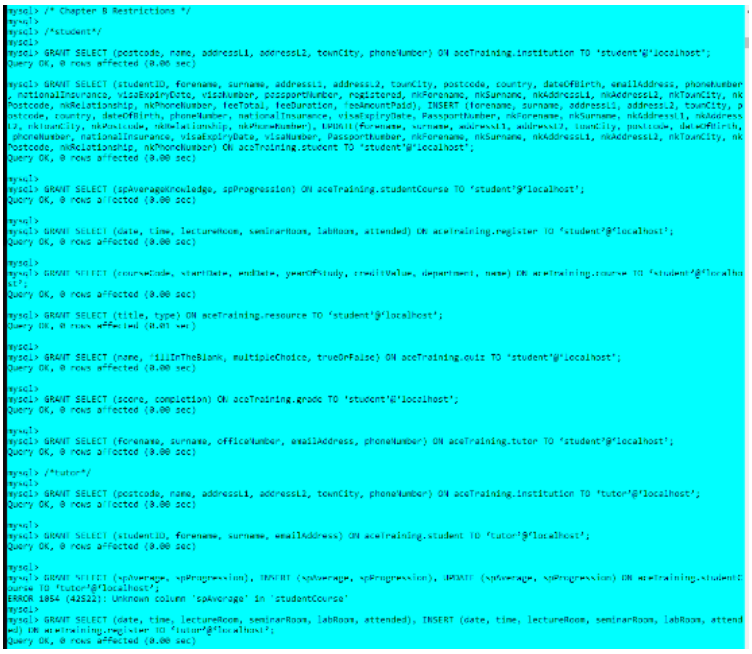


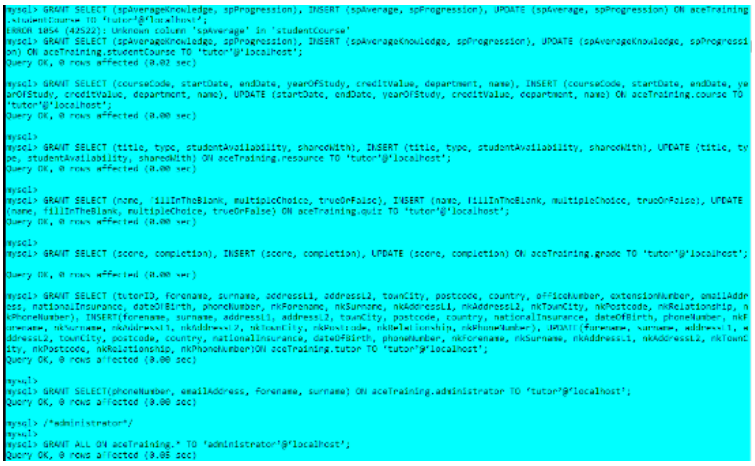


```
mysql> SHOW tables;
+------------------------+
| Tables_in_acetraining |
+------------------------+
| administrator          |
| course                 |
| grade                  |
| institution            |
| institutioncourse      |
| quiz                   |
| register               |
| resource               |
| resourceforcourse      |
| student                |
| studentcourse          |
| studentregister        |
| tutor                  |
| tutorcourse            |
+------------------------+
14 rows in set (0.06 sec)
```






mysql> /* Chapter 8 Restrictions */
mysql>
mysql> /*student*/
mysql>
mysql> GRANT SELECT (postcode, name, addressL1, addressL2, townCity, phoneNumber) ON aceTraining.institution TO 'student'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT SELECT (studentID, forename, surname, addressL1, addressL2, townCity, postcode, country, dateOfBirth, emailAddress, phoneNumber
, nationalInsurance, visaExpiryDate, visaNumber, passportNumber, registered, fkForename, fkSurname, fkAddressL1, fkAddressL2, fkTownCity, fk
Postcode, fkRelationship, fkPhoneNumber, feeTotal, feeDuration, feeAmountPaid), INSERT (forename, surname, addressL1, addressL2, townCity, p
ostcode, country, dateOfBirth, phoneNumber, nationalInsurance, visaExpiryDate, PassportNumber, fkForename, fkSurname, fkAddressL1, fkAddress
L2, fkTownCity, fkPostcode, fkRelationship, fkPhoneNumber), UPDATE(forename, surname, addressL1, addressL2, townCity, postcode, dateOfBirth,
phoneNumber, nationalInsurance, visaExpiryDate, visaNumber, PassportNumber, fkForename, fkSurname, fkAddressL1, fkAddressL2, fkTownCity, fk
Postcode, fkRelationship, fkPhoneNumber) ON aceTraining.student TO 'student'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql>
mysql> GRANT SELECT (spAverageKnowledge, spProgression) ON aceTraining.studentCourse TO 'student'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql>
mysql> GRANT SELECT (date, time, lectureRoom, seminarRoom, labRoom, attended) ON aceTraining.register TO 'student'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql>
mysql> GRANT SELECT (courseCode, startDate, endDate, yearOfStudy, creditValue, department, name) ON aceTraining.course TO 'student'@'localho
st';
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT SELECT (title, type) ON aceTraining.resource TO 'student'@'localhost';
Query OK, 0 rows affected (0.01 sec)
mysql>
mysql> GRANT SELECT (name, fillInTheBlank, multipleChoice, trueOrFalse) ON aceTraining.quiz TO 'student'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql>
mysql> GRANT SELECT (score, completion) ON aceTraining.grade TO 'student'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql>
mysql> GRANT SELECT (forename, surname, officeNumber, emailAddress, phoneNumber) ON aceTraining.tutor TO 'student'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql> /*tutor*/
mysql>
mysql> GRANT SELECT (postcode, name, addressL1, addressL2, townCity, phoneNumber) ON aceTraining.institution TO 'tutor'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql>
mysql> GRANT SELECT (studentID, forename, surname, emailAddress) ON aceTraining.student TO 'tutor'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql>
mysql> GRANT SELECT (spAverage, spProgression), INSERT (spAverage, spProgression), UPDATE (spAverage, spProgression) ON aceTraining.student
Course TO 'tutor'@'localhost';
ERROR 1054 (42S22): Unknown column 'spAverage' in 'studentCourse'
mysql>
mysql> GRANT SELECT (date, time, lectureRoom, seminarRoom, labRoom, attended), INSERT (date, time, lectureRoom, seminarRoom, labRoom, attend
ed) ON aceTraining.register TO 'tutor'@'localhost';
Query OK, 0 rows affected (0.00 sec)


mysql> GRANT SELECT (spAverageKnowledge, spProgression), INSERT (spAverage, spProgression), UPDATE (spAverage, spProgression) ON aceTraining
.studentCourse TO 'tutor'@'localhost';
ERROR 1054 (42S22): Unknown column 'spAverage' in 'studentCourse'
mysql> GRANT SELECT (spAverageKnowledge, spProgression), INSERT (spAverageKnowledge, spProgression), UPDATE (spAverageKnowledge, spProgressi
on) ON aceTraining.studentCourse TO 'tutor'@'localhost';
Query OK, 0 rows affected (0.02 sec)
mysql> GRANT SELECT (courseCode, startDate, endDate, yearOfStudy, creditValue, department, name), INSERT (courseCode, startDate, endDate, ye
arOfStudy, creditValue, department, name), UPDATE (startDate, endDate, yearOfStudy, creditValue, department, name) ON aceTraining.course TO
'tutor'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql>
mysql> GRANT SELECT (title, type, studentAvailability, sharedWith), INSERT (title, type, studentAvailability, sharedWith), UPDATE (title, ty
pe, studentAvailability, sharedWith) ON aceTraining.resource TO 'tutor'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql>
mysql> GRANT SELECT (name, fillInTheBlank, multipleChoice, trueOrFalse), INSERT (name, fillInTheBlank, multipleChoice, trueOrFalse), UPDATE
(name, fillInTheBlank, multipleChoice, trueOrFalse) ON aceTraining.quiz TO 'tutor'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql>
mysql> GRANT SELECT (score, completion), INSERT (score, completion), UPDATE (score, completion) ON aceTraining.grade TO 'tutor'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT SELECT (tutorID, forename, surname, addressL1, addressL2, townCity, postcode, country, officeNumber, extensionNumber, emailAdde
ss, nationalInsurance, dateOfBirth, phoneNumber, fkForename, fkSurname, fkAddressL1, fkAddressL2, fkTownCity, fkPostcode, fkRelationship, f
kPhoneNumber), INSERT(forename, surname, addressL1, addressL2, townCity, postcode, country, nationalInsurance, dateOfBirth, phoneNumber, fkF
orename, fkSurname, fkAddressL1, fkAddressL2, fkTownCity, fkPostcode, fkRelationship, fkPhoneNumber), UPDATE(forename, surname, addressL1, a
ddressL2, townCity, postcode, country, nationalInsurance, dateOfBirth, phoneNumber, fkForename, fkSurname, fkAddressL1, fkAddressL2, fkTownC
ity, fkPostcode, fkRelationship, fkPhoneNumber)ON aceTraining.tutor TO 'tutor'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql>
mysql> GRANT SELECT(phoneNumber, emailAddress, forename, surname) ON aceTraining.administrator TO 'tutor'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql> /*administrator*/
mysql>
mysql> GRANT ALL ON aceTraining.* TO 'administrator'@'localhost';
Query OK, 0 rows affected (0.00 sec)

SQL Code to create the database for Ace Training

```
/* creating users */

CREATE USER 'student'@'localhost'
IDENTIFIED BY 'mypass';

CREATE USER 'tutor'@'localhost'
IDENTIFIED BY 'mypass';

CREATE USER 'administrator'@'localhost'
IDENTIFIED BY 'mypass';

USE mysql;

SELECT user, password FROM user;

/* creating the database */

CREATE DATABASE aceTraining;
USE aceTraining;

/* creating database tables */

CREATE TABLE institution
   (name varchar(20) not null,
    addressL1 varchar(50) not null,
    addressL2 varchar(50),
    townCity varchar(25) not null,
    postcode varchar(15) not null unique,
    phoneNumber integer(20) not null unique,

    primary key(postcode)
   );

CREATE TABLE institutionCourse
   (postcode varchar(15) not null,
    courseCode char(15) not null,

    foreign key(postcode) references institution(postcode)
    ON UPDATE CASCADE ON DELETE RESTRICT,
    foreign key(courseCode) references course(courseCode)
    ON UPDATE CASCADE ON DELETE RESTRICT
   );

CREATE TABLE course
   (courseCode char(15) not null unique,
    startDate date not null,
    endDate date,
    yearOfStudy smallint not null,
    creditValue integer(3) not null,
    department varchar(50) not null,
    name varchar(50) not null,
    date date not null,

    primary key(courseCode),
    foreign key(date) references register(date)
    ON UPDATE CASCADE ON DELETE RESTRICT
   );

CREATE TABLE resourceForCourse
   (resourceID smallint not null,
    courseCode char(15) not null,

    foreign key(resourceID) references resource(resourceID)
```

```
    ON UPDATE CASCADE ON DELETE RESTRICT,
    foreign key(courseCode) references course(courseCode)
    ON UPDATE CASCADE ON DELETE RESTRICT
   );

CREATE TABLE resource
   (resourceID smallint auto_increment,
    title varchar(50) not null,
    type varchar(20) not null,
    studentAvailability char(1) not null,
    sharedWith varchar(50) not null,

    primary key(resourceID)
   );

CREATE TABLE quiz
   (quizID smallint auto_increment,
    name varchar(50) not null unique,
    courseCode char(15) not null,
    fillInTheBlank text,
    multipleChoice text,
    trueOrFalse text,

    primary key(quizID),
    foreign key(courseCode) references course(courseCode)
    ON UPDATE CASCADE ON DELETE RESTRICT
   );

CREATE TABLE grade
   (gradeID smallint auto_increment,
    score varchar(4) not null,
    completion varchar(4) not null,
    studentID smallint not null,
    quizID smallint not null,

    primary key(gradeID),
    foreign key(studentID) references student(studentID)
    ON UPDATE CASCADE ON DELETE RESTRICT,
    foreign key(quizID) references quiz(quizID)
    ON UPDATE CASCADE ON DELETE RESTRICT
   );

CREATE TABLE studentCourse
   (studentID smallint not null,
    courseCode char(15) not null,
    spAverageKnowledge varchar(20) not null,
    spProgression varchar(20) not null,

    foreign key(studentID) references student(studentID)
    ON UPDATE CASCADE ON DELETE RESTRICT,
    foreign key(courseCode) references course(courseCode)
    ON UPDATE CASCADE ON DELETE RESTRICT
   );

CREATE TABLE register
   (date date not null unique,
    time time not null,
    lectureRoom varchar(20) not null,
    seminarRoom varchar(20) not null,
    labRoom varchar(20) not null,
    attended char(1) not null,

    primary key(date)
   );
```

```
CREATE TABLE studentRegister
   (date date not null,
    studentID smallint not null,

    foreign key(studentID) references student(studentID)
    ON UPDATE CASCADE ON DELETE RESTRICT,
    foreign key(date) references register(date)
    ON UPDATE CASCADE ON DELETE RESTRICT
   );

CREATE TABLE tutorCourse
   (tutorID smallint not null,
    courseCode char(15) not null,

    foreign key(tutorID) references tutor(tutorID)
    ON UPDATE CASCADE ON DELETE CASCADE,
    foreign key(courseCode) references course(courseCode)
    ON UPDATE CASCADE ON DELETE RESTRICT
   );

CREATE TABLE tutor
   (tutorID smallint auto_increment,
    forename varchar(25),
    surname varchar (25) not null,
    addressL1 varchar(50) not null,
    addressL2 varchar(50),
    townCity varchar(25) not null,
    postcode varchar(15) not null,
    country varchar(255) not null,
    officeNumber integer(20) not null,
    extensionNumber integer(20) not null,
    emailAddress varchar(254) not null unique,
    nationalInsurance varchar(20) not null unique,
    dateOfBirth date not null,
    phoneNumber integer(20) not null,
    nkForename varchar(25),
    nkSurname varchar (25) not null,
    nkAddressL1 varchar(50) not null,
    nkAddressL2 varchar(50),
    nkTownCity varchar(25) not null,
    nkPostcode varchar(15) not null,
    nkRelationship varchar(20) not null,
    nkPhoneNumber integer(20) not null,
    administratorID smallint not null unique,

    primary key(tutorID),
    foreign key(administratorId) references administrator(administratorId)
    ON UPDATE CASCADE ON DELETE RESTRICT
   );

CREATE TABLE student
   (studentID smallint auto_increment,
    forename varchar(25),
    surname varchar (25) not null,
    addressL1 varchar(50) not null,
    addressL2 varchar(50),
    townCity varchar(25) not null,
    postcode varchar(15) not null,
    country varchar(255) not null,
    dateOfBirth date not null,
    emailAddress varchar(254) not null unique,
    phoneNumber integer(20) not null,
    nationalInsurance varchar(20) not null unique,
    visaExpiryDate date not null,
    visaNumber varchar(8) not null,
```

```
    passportNumber varchar(15) not null unique,
    registered varchar(15) not null,
    nkForename varchar(25),
    nkSurname varchar (25) not null,
    nkAddressL1 varchar(50) not null,
    nkAddressL2 varchar(50),
    nkTownCity varchar(25) not null,
    nkPostcode varchar(15) not null,
    nkRelationship varchar(20) not null,
    nkPhoneNumber integer(20) not null,
    feeTotal decimal(7,2) not null,
    feeDuration varchar(2) not null,
    feeAmountPaid decimal(7,2) not null,

    primary key(studentID)
   );

CREATE TABLE administrator
   (administratorID smallint auto_increment,
    forename varchar(25),
    surname varchar (25) not null,
    emailAddress varchar(254) not null unique,
    phoneNumber integer(20) not null,
    addressL1 varchar(50) not null,
    addressL2 varchar(50),
    townCity varchar(25) not null,
    postcode varchar(15) not null,
    country varchar(255) not null,
    dateOfBirth date not null,
    nkForename varchar(25),
    nkSurname varchar (25) not null,
    nkAddressL1 varchar(50) not null,
    nkAddressL2 varchar(50),
    nkTownCity varchar(25) not null,
    nkPostcode varchar(15) not null,
    nkRelationship varchar(20) not null,
    nkPhoneNumber integer(20) not null,

    primary key(administratorID)
   );
```

/*student*/

```
GRANT SELECT (postcode, name, addressL1, addressL2, townCity, phoneNumber) ON
aceTraining.institution TO 'student'@'localhost';

GRANT SELECT (studentID, forename, surname, addressL1, addressL2, townCity,
postcode, country, dateOfBirth, emailAddress, phoneNumber, nationalInsurance,
visaExpiryDate, visaNumber, passportNumber, registered, nkForename, nkSurname,
nkAddressL1, nkAddressL2, nkTownCity, nkPostcode, nkRelationship, nkPhoneNumber,
feeTotal, feeDuration, feeAmountPaid), INSERT (forename, surname, addressL1,
addressL2, townCity, postcode, country, dateOfBirth, phoneNumber,
nationalInsurance, visaExpiryDate, PassportNumber, nkForename, nkSurname,
nkAddressL1, nkAddressL2, nkTownCity, nkPostcode, nkRelationship, nkPhoneNumber),
UPDATE(forename, surname, addressL1, addressL2, townCity, postcode, dateOfBirth,
phoneNumber, nationalInsurance, visaExpiryDate, visaNumber, PassportNumber,
nkForename, nkSurname, nkAddressL1, nkAddressL2, nkTownCity, nkPostcode,
nkRelationship, nkPhoneNumber) ON aceTraining.student TO 'student'@'localhost';

GRANT SELECT (spAverageKnowledge, spProgression) ON aceTraining.studentCourse TO
'student'@'localhost';

GRANT SELECT (date, time, lectureRoom, seminarRoom, labRoom, attended) ON
aceTraining.register TO 'student'@'localhost';
```

```
GRANT SELECT (courseCode, startDate, endDate, yearOfStudy, creditValue, department,
name) ON aceTraining.course TO 'student'@'localhost';

GRANT SELECT (title, type) ON aceTraining.resource TO 'student'@'localhost';

GRANT SELECT (name, fillInTheBlank, multipleChoice, trueOrFalse) ON
aceTraining.quiz TO 'student'@'localhost';

GRANT SELECT (score, completion) ON aceTraining.grade TO 'student'@'localhost';

GRANT SELECT (forename, surname, officeNumber, emailAddress, phoneNumber) ON
aceTraining.tutor TO 'student'@'localhost';
```

/*tutor*/

```
GRANT SELECT (postcode, name, addressL1, addressL2, townCity, phoneNumber) ON
aceTraining.institution TO 'tutor'@'localhost';

GRANT SELECT (studentID, forename, surname, emailAddress) ON aceTraining.student TO
'tutor'@'localhost';

GRANT SELECT (spAverageKnowledge, spProgression), INSERT (spAverageKnowledge,
spProgression), UPDATE (spAverageKnowledge, spProgression) ON
aceTraining.studentCourse TO 'tutor'@'localhost';

GRANT SELECT (date, time, lectureRoom, seminarRoom, labRoom, attended), INSERT
(date, time, lectureRoom, seminarRoom, labRoom, attended) ON aceTraining.register
TO 'tutor'@'localhost';

GRANT SELECT (courseCode, startDate, endDate, yearOfStudy, creditValue, department,
name), INSERT (courseCode, startDate, endDate, yearOfStudy, creditValue,
department, name), UPDATE (startDate, endDate, yearOfStudy, creditValue,
department, name) ON aceTraining.course TO 'tutor'@'localhost';

GRANT SELECT (title, type, studentAvailability, sharedWith), INSERT (title, type,
studentAvailability, sharedWith), UPDATE (title, type, studentAvailability,
sharedWith) ON aceTraining.resource TO 'tutor'@'localhost';

GRANT SELECT (name, fillInTheBlank, multipleChoice, trueOrFalse), INSERT (name,
fillInTheBlank, multipleChoice, trueOrFalse), UPDATE (name, fillInTheBlank,
multipleChoice, trueOrFalse) ON aceTraining.quiz TO 'tutor'@'localhost';

GRANT SELECT (score, completion), INSERT (score, completion), UPDATE (score,
completion) ON aceTraining.grade TO 'tutor'@'localhost';

GRANT SELECT (tutorID, forename, surname, addressL1, addressL2, townCity, postcode,
country, officeNumber, extensionNumber, emailAddress, nationalInsurance,
dateOfBirth, phoneNumber, nkForename, nkSurname, nkAddressL1, nkAddressL2,
nkTownCity, nkPostcode, nkRelationship, nkPhoneNumber), INSERT(forename, surname,
addressL1, addressL2, townCity, postcode, country, nationalInsurance, dateOfBirth,
phoneNumber, nkForename, nkSurname, nkAddressL1, nkAddressL2, nkTownCity,
nkPostcode, nkRelationship, nkPhoneNumber), UPDATE(forename, surname, addressL1,
addressL2, townCity, postcode, country, nationalInsurance, dateOfBirth,
phoneNumber, nkForename, nkSurname, nkAddressL1, nkAddressL2, nkTownCity,
nkPostcode, nkRelationship, nkPhoneNumber)ON aceTraining.tutor TO
'tutor'@'localhost';

GRANT SELECT(phoneNumber, emailAddress, forename, surname) ON
aceTraining.administrator TO 'tutor'@'localhost';
```

/*administrator*/

```
GRANT ALL ON aceTraining.* TO 'administrator'@'localhost';
```