1 DESIGNING A DATABASE

There is a well developed theory underlying relational databases, entire books have been written on this subject. There are a number of principles to keep in mind when designing a relational database which will be illustrated in this workbook.

The first stage is to consider carefully what information you wish to record, the form in which you wish to record it and what data you want to retrieve from the database.

We will design a database for use in an administrative department in the University. A department needs to keep records about students, which modules they are taking and the marks they get for those particular modules.

If this were implemented as a Flat File Database, there would need to be a record containing details about each student and also details about the modules. If a student were taking 6 modules their name would have to be repeated six times in the database alongside the six module details. This is not very efficient and the larger the database grows the more difficult it becomes to search and manipulate.

In a relational database the details of the students would be kept in a separate table from those of each module. In this way the tables can be linked together by a common field. The linking field is the only repetition between the tables making it much more efficient.

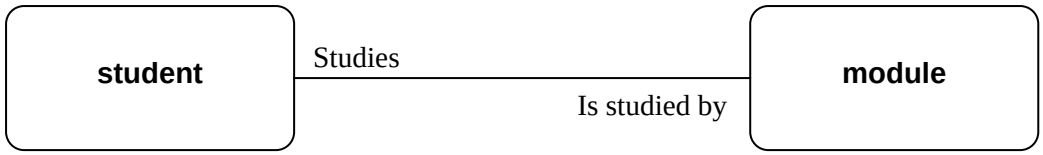
1.1 What subjects are covered by the data

Subjects are also known as **Entities**. We will draw an **Entity-Relationship-Attribute** diagram to identify the entities in this database. An entity is represented as a box with rounded corners and a name in the singular.

Two entities can be identified in this example, **student** and **module** as information about each student and each module is required

1.2 Relationships between entities

The next stage is to decide how the entities are related to each other. A relationship is a significant relationship between two entities. It is represented by a line that joins two entity boxes. A relationship has a name and a degree.



The relationship in this example is that a student studies a module and a module is studied by a student.

1.3 Degrees of relationships

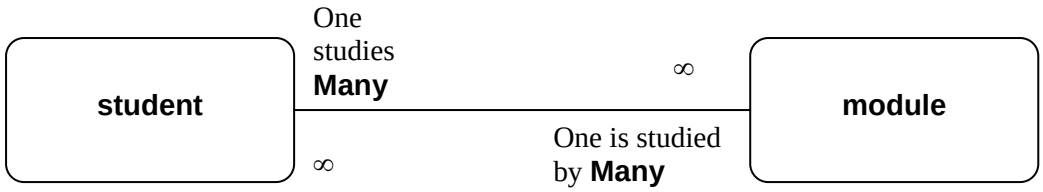
A degree of a relationship can be classed as **one to one**, **one to many** or **many to many**.

One to One relationships are fairly rare in database design. They occur when one occurrence of an entity has a relationship with only one occurrence of another entity. An example of this is that at **one** particular time a patient can occupy **one** bed in a ward and **one** bed can only be occupied by **one** patient at any one time.

One to Many relationships occur when one entity can be related many times to another entity. An example of this is that a book can be published by only **one** publisher but a publisher can publish **many** other books.

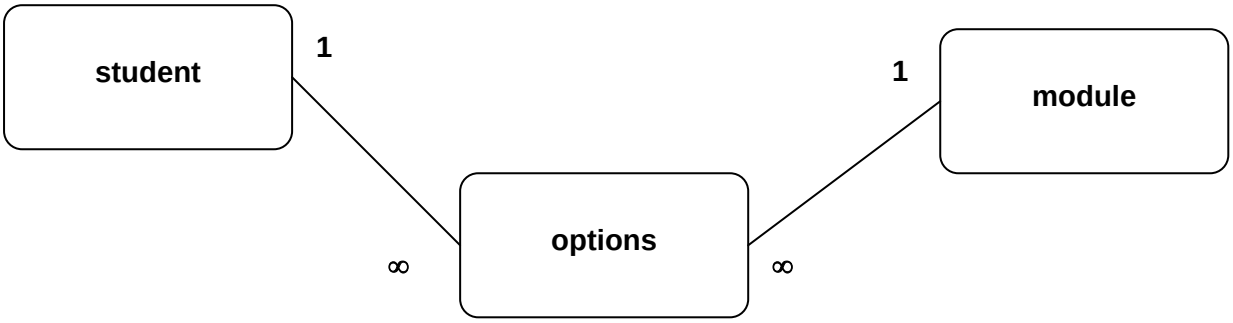
Many to Many relationships occur where there can be many occurrences of one entity related to many occurrences of another entity. An example of this is a book may have **many** authors and the author may have written **many** books. In this case the degree of the relationship is Many to Many.

In this example one student can study many modules and one module can be studied by many different students therefore the degree of the relationship is Many to Many.



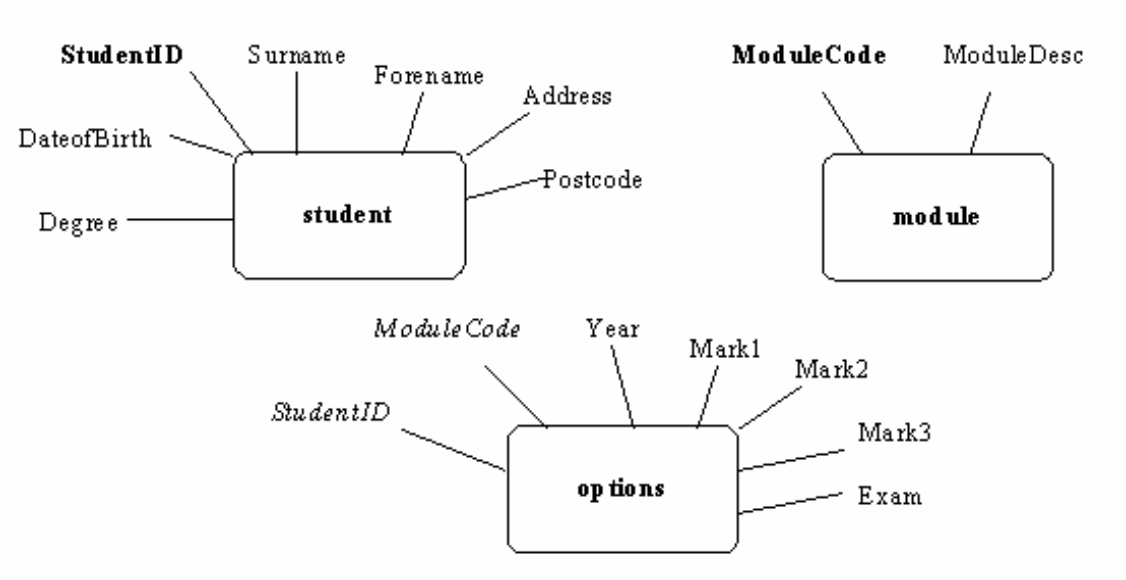
Many to Many relationships occur frequently in database design but unfortunately relational databases do not allow many to many relationships. They need to be split into two one to many relationships. This is done by creating another entity.

In this example another entity called **options** is created. This entity provides a link between the student and the modules they choose and will hold the student code, module code and marks received for the modules. It is helpful to think which field in a table will make the record unique. The student code will appear only once in **student** but could appear many times in **options** with different module codes. Similarly each module code will appear only once in **module** but could appear many times in **options** with different student codes.



1.4 Attributes of entities

An attribute is a property that describes an entity. They are important as the values of the attributes are the data that is stored in the tables. The attributes become the fields (columns) of a table.
In this example attributes have been assigned to each entity. Attributes in bold type are fields which will uniquely identify a particular record in the table.



1.5 Deriving database tables

Databases are structured into tables. A table consists of a set of records (rows) and each record contains a number of distinct fields (columns). The final stage is to translate the design from the diagram into a list of tables with appropriate fields.

The **entities** become the table names.

The **attributes** become the field names. There must be at least one attribute in each table that will enable it to be linked with another table in the database otherwise no relationship will be possible.

1.6 Primary and foreign keys

A Primary Key is allocated to a field (or group of fields) that uniquely identifies each record in a table. A relational database table should not contain any records which are exactly the same. The Primary Key is often a product code or ID number and determines the order of the records in a table; it also speeds up the querying of a table.

Each student can be uniquely identified by their UCAS number, **StudentID** which will become the primary key of the **student** table.

Each module has a unique module code, **Mcode** which will become the primary key of the **module** table.

The primary key in the options table is a composite key because it comprises of more than one field. For any one record in the options table to be unique the combination of the **StudentID** and **Mcode** fields must be unique. These fields are called **Foreign keys** as they are both primary keys in other tables in the database.

Table and field names for the administrative database:

student	**options**	**module**
StudentID *	***StudentID** ****	**ModuleCode ***
Surname	***ModuleCode** ****	ModuleDesc
Forename	Year	
Address	Mark1	
Postcode	Mark2	
DateOfBirth	Mark3	
Degree	Exam	

***** Primary key field
italics - Foreign key field

1.7 Field names and table names

There are some guidelines for naming fields, controls, objects and tables in Access:

Fields, controls and objects:

Can be up to 64 characters long.

Can include any combination of letters, numbers, spaces, and special characters except a period (.), an exclamation mark (!), an accent grave (`), and brackets ([ ]).

Can't start with leading spaces.

Can't include control characters (ASCII values 0 through 31).

Can't include a double quotation mark (") in table, view, or stored procedure names in a Microsoft Access project.

Must be unique within a single table.

Tables:

Can be up to 64 characters long.

Can contain spaces, however this is not recommended for Visual Basic programming purposes.

1.8 Access data types

Fields must be defined before data can be entered into a table. When defining fields in a table the type of data in that field also needs to be identified. Data can be in various forms which have different characteristics. There are ten data types in Access namely **Text, Memo, Number, Date/Time, Currency, AutoNumber, Yes/No, OLE objects, Hyperlink** and **Lookup Wizard**.

Text - Letters, Numbers or other keyboard symbols. Limited to 255 characters.

Memo - Large amounts of text. Limited to 64,000 characters.

Number - Numerical data.

Date/Time - Any valid date or time.

Currency - Monetary values.

AutoNumber - A numeric value which is automatically increased for each new record.

Yes/No - Boolean values. Allow Yes/No or True/False values.

OLE Objects - Objects such as graphics or spreadsheets which are from other Windows applications that support Object Linking and Embedding.

Hyperlink - Text used as a link to a file or page of data.

Lookup Wizard - Permit the choice of a value from another table or from a list of values from a list or combo box.