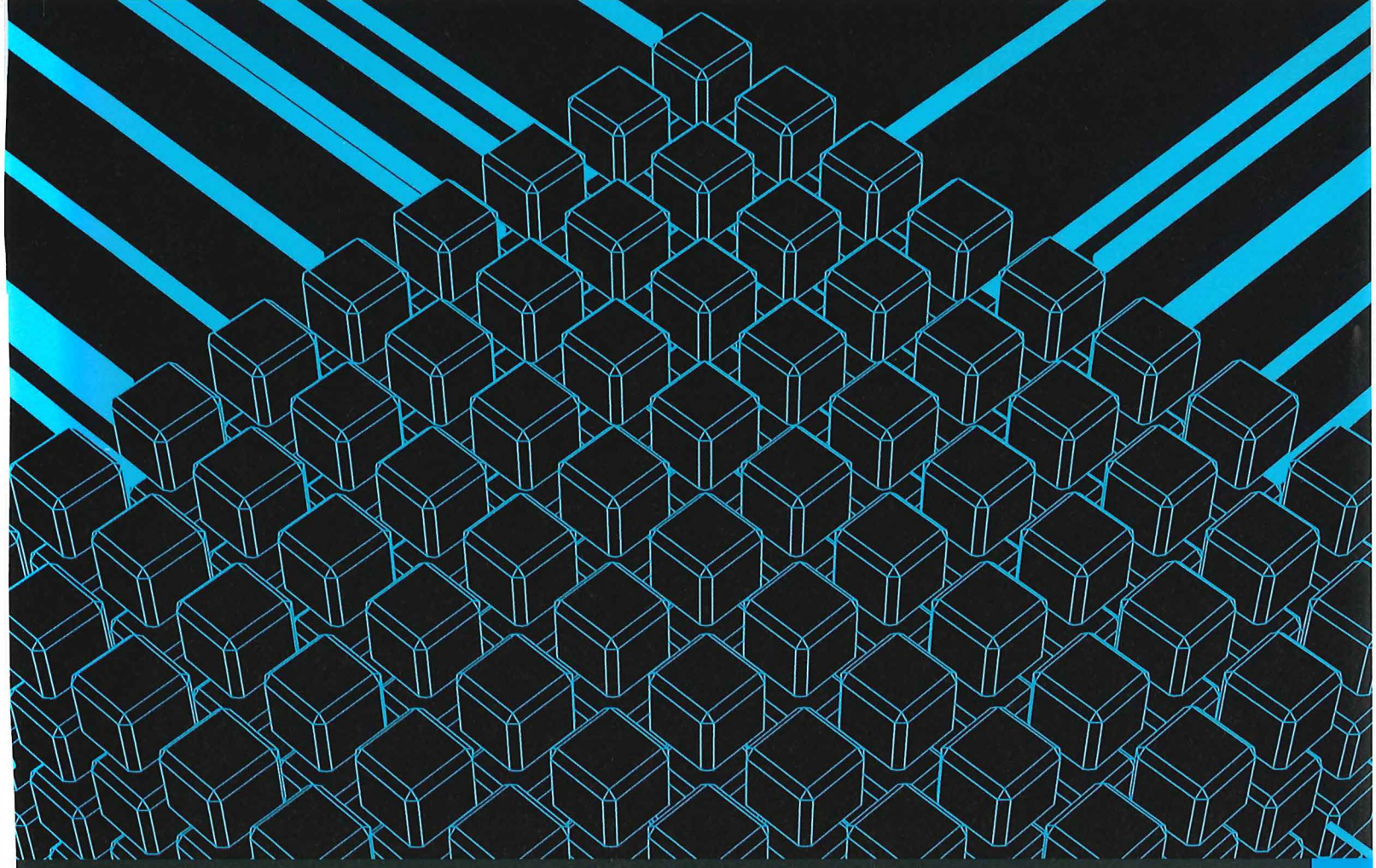



# Chapter 14: Databases

## Learning objectives

***By the end of this chapter you will:***

- understand what is meant by a database
- know that a database can be paper-based or electronic
- understand what databases are used for
- understand what is meant by database structure, including tables, records and fields
- understand what is meant by a single table (flat file) database
- understand that different data types exist and how to choose a suitable data type for a field
- understand what is meant by a primary key and how to choose a suitable primary key
- understand what is meant by a query and how to perform a query-by-example
- understand that data in a database is only of use if it is reliable
- be able to explain the consequences of unreliable data in a database.

# 14.01 What is a database?

A database is a collection of **data** that is stored in a logical and organised manner. Databases are often electronic, but they do not have to be. Many databases are paper-based, for example a catalogue or a telephone directory. In fact, any collection of data can be called a database if the data is stored logically and in a structured format.

**KEY TERM**

**Data** – raw facts and figures.

Databases can be used in a number of ways, for example:

- Patient medical records held at a hospital are used by doctors and nurses to help treat patients.
- Search engines store data about websites, such as their name, web address and their content; users can quickly search for and find the information they are looking for.
- TV program listings contain the TV channels available, the programmes they are to show and their viewing time.
- Telephone directories contain the names, addresses and telephone numbers of people and businesses.
- Online shop databases store data about all the products the shop sells so that customers can browse for and order items.
- In schools, databases can store data about pupils such as their name, date of birth and address. They could store attendance registers. They could also be used by teachers to record classwork marks and exam results.
- A library stores each book's data, such as title, genre and author, so that someone can easily search for a book.



An electronic database is a database that is created and stored on a computer. In an electronic database a computer program can be used to quickly select the data that we require. Electronic databases are widely used because of the advantages they offer to the user (see Table 14.01).

Table 14.01

| Advantage | Example |
|---|---|
| The data can easily be sorted in a specific way, for example alphabetically or from highest to lowest | A sort could be used to arrange products in an online shop by price, popularity or average customer review. It could also be used to arrange marks in a teacher's mark book by exam score |
| The data we need can be found quickly | In a library, it may take us time to find the book we want. If the library has an electronic database of books, we could search for the book and the database would tell us exactly where to find the book in the library<br><br>Also, we may not remember the name of the book, only the author. If we search for an author in the database we can see what books they have written and this may prompt our memory for the one we were looking for |

| Advantage | Example |
|---|---|
| Data can be filtered to retrieve only the data we require | An online provider of television programs and movies would have a filter on their database. If we like science fiction movies we can just request, by selecting the Science Fiction category, to see those movies in the database |
| Data can easily be shared over the internet or using a portable storage device such as an external hard drive | A clothing retail chain, with a number of branches, may want stores to know the stock levels in a warehouse. The warehouse could share a database to show each branch the levels of stock left of the products that are sold. This way, the retail branches always know what clothing products are in stock |
| Data can be validated to make sure that data entered is possible, realistic and sensible | Validation could be used in an electronic database of patient records in a hospital. If there was no validation present on the patient's date of birth it could be entered as 1st January 2001, 01/01/2001, 01/01/01 or even 1/1/01. When a doctor wanted to search for patients born on 01/01/2001, it would only pick up records where the data had been entered in this way, giving a false figure. If we force the user to enter the data in the format DD/MM/YYYY then all the date-of-birth data would have to be entered in this way, making the database more reliable. This is more difficult to do with a paper-based system, in which rules can be suggested, but not enforced |
| A backup copy of the database can easily be made | A school could store a backup copy of all the students' records on a portable storage device. It could even keep this backup copy in fire-proof box. If the school were ever to suffer a network failure or even a fire and lose all the student records, they could use the backup copy to get the student data back again. A paper-based database can be copied but it might be very time consuming to copy, especially if the data runs to thousands of documents |
| It saves space compared to a paper database | A bank could store thousands of records in one electronic database, on a computer or portable storage device. A paper-based database would need a great number of filing cabinets or other storage methods to store thousands of records. This could take up a lot of valuable space in an office |



Storage devices and validation are mentioned in Table 14.01. If you would like to learn more about storage devices, see Chapter 8. If you would like to learn more about validation, see Chapter 13.

## 14.02 Database structure

**SYLLABUS CHECK**

Define a single-table database from given data storage requirements.

Before creating a database there needs to be some consideration given to its structure. It must be decided what data it will hold and how much data needs to be stored. This will affect the size of the database and how much storage space it will need.

**KEY TERM**

**Table** – a structure in which data is stored in a database.

**Flat file database** – a database whose records are stored in a single table.

**Record** – a collection of data about one single item in a database.

**Field** – one piece of data about an item in a database.

A **table** is the simplest structure in which to hold data. For example, the data in a contact list might be just names, phone numbers and birthdays.

| Name | Phone number | Birthday |
|---|---|---|
| John | 08881-764538 | 2nd October |
| Venkhat | 08881-983726 | 13th June |
| Fatmia | 08881-229220 | 21st May |
| Tanjung | 08881-123987 | 27th September |
| Steffi | 08881-445589 | 8th February |

Figure 14.01 A paper-based database of a contact list

An electronic database is basically a digital filing system. Databases store data in tables. A database with a single table is called a **flat file database**. The tables are made up of **records** and **fields**. A record is a collection of data about one single item in a database. A field is one piece of data about an item in a database.



A record

| Student Name | Phone number | Birthday |
|---|---|---|
| John | 08881-7645378 | 02/10 |
| Venkhat | 08881-983726 | 13/06 |
| Fatima | 08881-229220 | 21/05 |
| Tanjung | 08881-123987 | 27/09 |
| Steffi | 08881-445589 | 08/02 |

A field

Figure 14.02 A simple electronic database storing a contact list

**TEST YOURSELF**

Look at the example of the contact list database:

1 How many tables does it have?
2 Is it a flat file database?
3 How many records does it have?
4 How many fields does it have?

## 14.03 Data types

**KEY TERM**

**Data type** – the format of the data in a field.

A data type is a format that we can give to data in a field. When entering data into a database there we need to consider the appropriate **data type** for each field. The data type of a field

will depend on the data that is being stored in that field. Various data types are shown in Table 14.02.

Table 14.02

| Data type | Description | Example |
|---|---|---|
| Text (alphanumeric) | Any characters or symbols on a keyboard | Hello |
| Integer | A whole number that does not have any decimal places | 100 |
| Real | A number that does have decimal places | 1.01 |
| Date/time | Dates and times | 01/01/2015 12:30 |
| Boolean | Data that has only two possible values | Yes/no<br>True/False |

Consider the simple student database in Figure 14.03.

| Student Name | Date of Birth | Attempts at test | Average Score | Re-sit? |
|---|---|---|---|---|
| Peter | 07/08 | 3 | 8.6 | No |
| Amy | 28/02 | 2 | 9.5 | No |
| Vikram | 07/08 | 3 | 9.1 | No |
| Seema | 29/09 | 3 | 6.3 | Yes |
| Amy | 06/03 | 2 | 9.1 | No |

Figure 14.03

The database contains five fields. Each field requires a different data type.

Table 14.03

| Field name | Data type |
|---|---|
| Student Name | Text |
| Date of Birth | Date/time |
| Attempts at test | Integer |
| Average Score | Real |
| Re-sit? | Boolean |

It is very important to choose the correct data type for each field. If the wrong data type is chosen, data could be stored incorrectly. For example, in the student database, if the data type for the average score is specified as an integer, the scores will not be stored correctly. This is because only a whole number would be stored. Peter's score of 8.6 may be rounded up to 9 or truncated to 8. It could not be stored as 8.6 as that is not a whole number. The database would store an incorrect average score for Peter.

Data sometimes needs to be stored using a different data type than we might immediately think. For example, telephone numbers are often stored as text. In some countries, telephone numbers start with a 0. However, computers do not recognise 0s that occur at the beginning of integers, and would remove them from the data. The telephone number 01234 567890 stored as an integer would read 1234 567890. This means the telephone numbers would be missing their first digit and would be incorrect data. Specifying the data type as text overcomes this problem as the computer will now recognise the 0. Additionally, using a text data type allows spaces to be inserted in the phone numbers. This makes them easier to read.

To learn more about data types, see Chapter 11.

## 14.04 Primary keys

**SYLLABUS CHECK**

Choose a suitable primary key for a database table.

A table should have a field that will be a unique identifier for each record. This means that it will give each record a unique reference so that two records can be told apart. If two people in a database have the same name, a unique reference would be a way to tell the records apart. This unique identifier is called a **primary key.** It can also be known as a 'key field'.



**KEY TERM**

**Primary key** – a unique identifier for each record in a database.

**Index** – a list of values or items.

In Figure 14.03, there are two students called Amy. Also, both Peter and Vikram have the same birthday. Therefore, neither the Student Name field nor the Date of Birth field would be suitable as a primary key, as not all the data in those fields are unique. Similarly, no unique data exists in any of the three remaining fields. As it stands, this database has no suitable field to use as a primary key as no field has unique data.

To overcome this problem we would add an extra field to the database. Each student could be given a unique ID number that could be used to tell his or her records apart. This field would be ideal as a primary key, since every item of data in that field can be unique.

| Student ID | Student Name | Date of Birth | Attempts at test | Average Score | Re-sit? |
|---|---|---|---|---|---|
| 1001 | Peter | 07/08 | 3 | 8.6 | No |
| 1002 | Amy | 28/02 | 2 | 9.5 | No |
| 1003 | Vikram | 07/08 | 3 | 9.1 | No |
| 1004 | Seema | 29/09 | 3 | 6.3 | Yes |
| 1005 | Amy | 06/03 | 2 | 9.1 | No |

Primary key

Figure 14.04 Database of students with a primary key added

Now each student can be uniquely identified. The key acts also as an **index** by which all records can be ordered and kept individual.

**TEST YOURSELF**

Consider this database about books:

| Book title | Genre | Author | Reviewer's score | In print? | ISBN |
|---|---|---|---|---|---|
| Great Journeys | Travel | Kendo Takahashi | 3.5 | No | 1863827662 |
| Computer Science Projects | Computer Science | Sarah Lawrey | 4.8 | Yes | 22990398272 |
| Great Journeys | Astronomy | Jan Timmins | 5.0 | Yes | 94837263728 |
| Computing is Fun! | Computing | Sarah Lawrey | 4.0 | Yes | 94220333723 |
| 100 Best Films | Film and TV | Dheeraj Saltani | 3.5 | No | 83726372688 |

1 Can you identify the correct data type required for each field?
2 Is there a suitable field for a primary key or would an extra field need to be added?

## 14.05 Relational databases



An extension topic to the syllabus would be to look at databases that have more than one table.

A **relational database** is a database that has more than one table. The tables are related (linked) using their primary keys. For example, a veterinary practice might have a relational database with three tables, as shown in Table 14.04.

Table 14.04

| Table | Primary key | Data stored |
|---|---|---|
| Owner | Owner Number | Data about each owner, such as their name and address. The Owner Number would be included in the Pet table to link the owner to the pet |
| Pet | Pet Number | Data about each pet, such as its name and breed. This table would also contain the Owner Number to link the owner to the pet |
| Appointment | Appointment Number | Data about each appointment, such as the time and date. The Pet Number would also appear in the table to link the pet to the appointment |

**KEY TERM**

**Relational database** – a database that has several tables linked by primary key fields.
**Foreign key** – a primary key that appears in another table in a relational database.

The database structure would look like Figure 14.06.

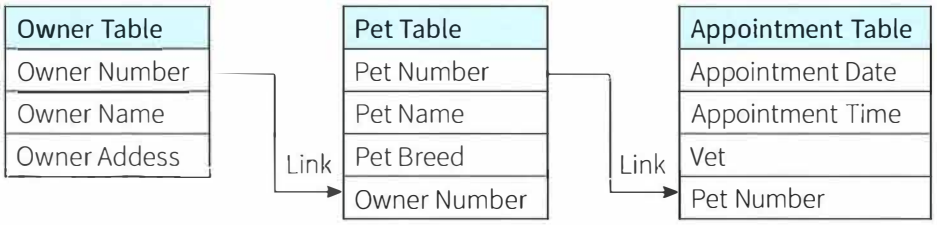


Figure 14.06 A relational database in a veterinary surgery

As the tables are linked, all relevant data can be seen easily:

- By looking up an owner, we could see which pets and appointments they have.
- By looking up a pet, we could see who their owner is and which appointments they have.
- By looking up an appointment, we could see which pet and owner the appointment is with.

When a primary key appears in another table it is called a **foreign key**.

Although a relational database is more complex in structure, it has several advantages over a flat file database.

Table 14.05

| Advantage | Example |
|---|---|
| Data only needs to be entered once, but it can be used repeatedly within the database | An owner might have several pets. Each time a new pet's data is entered, the owner's data does not need to be re-entered. This is because the Owner Number in the Pet table provides a link to the owner's details. This reduces duplication of data and the size of the database. With a flat file database all the owner's data would have to be entered each time for every pet they have |
| Entering data only once reduces the chance of data errors | If we had to enter all the owner and pet data every time the pet had an appointment there is the possibility that on occasion the data would be entered incorrectly. With a flat file database this could lead to inconsistency in the data |
| If a data entry error has occurred the data only has to be corrected once | If the pet's name had been entered incorrectly, we would only need to correct the name in the Pets table. With a flat file database we would also have to correct the name in each record in the Appointments table where that pet appeared |



## 14.06 Performing searches

**SYLLABUS CHECK**

Perform a query-by-example from given search criteria.

Databases can be very useful for searching through data. A search in a database is known as a **query**.

**KEY TERM**

**Query** – a method of searching a database.
**SQL** – a specialised programming language for maintaining databases and generating queries.
**Query-by-example** – a method of searching a database by stating the criteria to be searched.

When we search a database we use search criteria and we can do this by using **query-by-example**. In a query-by-example we choose each field we want to appear in the results and set any criteria we want to apply to that field.

There are a few items of data we need to include in our query-by-example, these are:

- The table
- The field
- The criteria
- Any sorting that we want to apply
- If we require the field to appear in the results or not.

We can structure this in a table type format to show what we want to perform as a search. For example, if we wanted to search our student database for those students who have taken less than three attempts at the test and have an average score of 9 or above. We want our results to only show the name of the student and no further data. Our query-by-example would look like this:

| **Table:** | **Students** | **Students** | **Students** |
|---|---|---|---|
| Field: | Student Name | Attempts at test | Average Score |
| Sort: | | | |
| Criteria: | | <3 | =>9 |
| | | | |
| Show: | ☑ | ☐ | ☐ |

By placing 'less than 3' in the criteria for the number of attempts at the test and 'equal to or greater than 9' in the criteria, this will provide us with the students that meet the criteria requested. You will note that the only box ticked in the 'Show:' section is for Student Name. This is because the only field we wanted to see in our search result is the name of the student.

If we wanted to see students that have taken less than three attempts or had an average score of 9 or above we would change our query slightly to this:

| **Table:** | **Students** | **Students** | **Students** |
|---|---|---|---|
| Field: | Student Name | Attempts at test | Average Score |
| Sort: | | | |
| Criteria: | | <3 | |
| | | | =>9 |
| Show: | ☑ | ☐ | ☐ |

You will note that the criteria for the average score now appears in the second criteria box. Each criteria that is placed in the first box will mean the results given must meet all of the criteria requested. If a criteria is placed in the second criteria box, this means that the results must meet either the first or the second criteria.

Many databases use a special-purpose programming language called **SQL** (Structured Query Language). SQL allows a database programmer to create queries very similar to writing a sentence. For example, this SQL query would search for 'Seema' in our student database:

```
SELECT * FROM Students WHERE Name = 'Seema'
```

Table 14.06

| Keyword | Meaning | Description |
|---|---|---|
| SELECT | The command that is being performed | Records are being selected |
| FROM | Specifies the table that will be searched | In this query, the table is Students |
| WHERE | Specifies the field that we want to search for the criteria | In this query, the field is Name |
| = 'Seema' | The criterion that we are searching for | This query searches for 'Seema' in the Name field of the Students table |

The criteria can be chosen using one or more of several operators, shown in Table 14.07.

Table 14.07

| Operator | Description | Example |
|---|---|---|
| = | Equal to | Eye colour = brown |
| > | Greater than | Number in stock > 10 |
| < | Less than | Number sold < 5 |
| >= | Greater than or equal to | Height >= 1.5 |
| <= | Less than or equal to | Average test score <= 5 |
| AND | Combines two criteria | Eye colour = brown AND Height > 1.5 |
| OR | Either one criterion or another | Eye colour = brown OR Eye colour = blue |
| NOT | Excluding this criterion | Eye colour NOT brown |



Consider the student database we examined in Figure 14.04.

We could perform a search for all students who need to re-sit a test:

Table 14.08

| Table | Students |
|---|---|
| Criterion | Re-sit? = 'Yes' |

The equivalent SQL query would be:

```
SELECT * FROM Students WHERE Re-sit? = 'Yes'
```

**TEST YOURSELF**

1. Which student records would be returned if this search was performed?
2. Why would the remaining student records not be returned?

We could perform a search for all students whose average score is over 9:

Table 14.09

| Table | Students |
|---|---|
| Criterion | Average Score > 9 |

The equivalent SQL query would be:

SQL – a specialised programming language for maintaining databases and generating queries SELECT * FROM Students WHERE Average Score > 9

**TEST YOURSELF**

1 Which student records would be returned if this search was performed?
2 Why would the remaining student records not be returned?

We could perform a search for all students whose average scores are over 5 and who attempted the test less than 3 times:

Table 14.10

| Table | Students |
|---|---|
| Criteria | Average Score > 9 AND Attempts at test < 3 |

The equivalent SQL query would be:

SELECT * FROM Students WHERE Average Score > 9 AND Attempts at test < 3

**TEST YOURSELF**

1 Which student records would be returned if this search was performed?
2 Why would the remaining student records not be returned?
3 Using the student database in Figure 14.04, create an SQL query or a query table for each of the following searches:
   - **a** Students who have scored higher than 5 on average.
   - **b** Students who do not need to re-sit a test.
   - **c** Students who have scored less than 9 on average or those who have taken the test 3 times.
   - **d** Students who have scored more than 8 on average and taken the test less than 3 times, or those who have scored less than 9 on average and taken the test twice.

There may be more than one possible way of creating a query for some searches.



## 14.07 Data validation

The data in a database is only useful if it is accurate and reliable. Any data that is entered incorrectly can cause query results to be incorrect and unreliable.

Consider the student database in Figure 14.04 once again. Suppose Peter's name had been incorrectly entered as Peta. If the teacher using the database used a query to search Peter's record, the search would return no results. This is because the name Peter does not exist in the database.

Similarly, if the re-sit field for Peter had been left blank (neither Yes nor No has been entered), then a query looking for those students who do not need to re-sit a test would not include Peter in its results. Peter may need to retake the test, but the teacher would not know.

Data validation is the process of automatically checking whether the data entered is possible, realistic and sensible. Validation does not make sure that the data is correct, incorrect data could still be entered that passes the validation. It eliminates the entering of data that is not realistic or sensible. This is done by making sure that the data entered follows certain rules. For example, in our student database, Vikram has an average score of 9.1. The validation on the Average Score field is that the number entered must be between 0 and 10. Suppose his score had been entered as 1.9. The data would be incorrect, but as far as the database is concerned, quite possible as it passes the validation.

To learn more about the ways in which data entered can be validated, see Chapter 13.

**TEST YOURSELF**

Inaccurate data in databases can have serious consequences.

1 What could happen in a hospital as a result of data in a database being unreliable?

2 What might be the consequences for an online shop whose products are in a database with inaccurate data?

## Summary

- A database is a collection of data stored in a logical and ordered manner. Databases can be either paper-based or electronic.
- All databases have a structure that allows the data held within them to be stored in a logical and orderly manner.
- Data is stored in tables consisting of records and fields.
- A database with a single table is known as a flat file database. A database with multiple tables is known as a relational database.
- A record is a collection of data about one single item in a database.
- A field is one piece of data about an item in a database.
- Each database field must be given a data type.
- Tables need a primary key, which is a field that holds data that uniquely identifies a record.
- A search for data in a database is known as a 'query'.
- SQL is a programming language that is used to maintain databases and to create queries. A common method of searching a database is using query by example.
- Data held within a database is only of use if it is reliable. Unreliable data can lead to serious consequences for users.

## Exam-style questions

1 Explain what a database is. (1 mark)

2 Describe two advantages of using an electronic database over a paper-based system. (2 marks)

3 Match the correct term with its definition: (5 marks)

| | |
|---|---|
| Field | A structure in which data is stored in a database |
| Query | A unique identifier for a record |
| Record | A collection of data about one single item in a database |
| Table | One piece of data about an item in a database |
| Primary Key | A method used to search a database |

4 Explain why it is important to give the correct data type to a field. (2 marks)

5 Bosire has a simple flat file database. Its table has two fields: name and telephone number.

**a** Explain why neither field is suitable as a primary key. What field could be added to create a primary key? (2 marks)

**b** Why might a user choose to create a flat file database rather than a relational database? (1 mark)

6 A database stores data in a table called Cars. The table structure includes fields called Colour, Engine Size and Number of Doors. Write an SQL query-by-example which searches for red or blue cars, with 5 doors and an engine size of more than 1600cc. (5 marks)

7 Why can unreliable data be of concern to a database user? (1 mark)

# Answers

The sample answers and marks provided are for the exam-style questions at the end of each chapter and have been written by the author. The way marks are awarded in examination may be different.

## Chapter 1 Data Representation

**1** The stopwatch would show 02:38:59.

(1 mark for correct hours)
(1 mark for correct minutes)
(1 mark for correct seconds)
**Total 3 marks**

**2** A byte is:

- A unit of data. (1 mark)
- Normally 8 bits. (1 mark)

**Total 2 marks**

**3** In hexadecimal notation it would be: C1E

(1 mark for each correct hex value)
**Total 3 marks**

**4** Two answers accepted from:

- It is easier for humans to read than binary. (1 mark)
- It is a much shorter way of representing a byte or bytes of data. (1 mark)
- It is easier to debug than binary. (1 mark)

**Total 2 marks**

**5** Two answers accepted from:

- Change the resolution of the image to a lower resolution. (1 mark)
- Use lossy compression to remove the redundant data from the image. (1 mark)
- Use lossless compression that will look for repeating patterns in the image. (1 mark)

(1 mark for each method)
(1 mark for explanation)
**Total 4 marks**

**6** The most suitable file extensions are:

| | .jpg | .mp4 | .mp3 | .csv |
|---|---|---|---|---|
| Storing holiday photos | ✓ | | | |
| Storing holiday videos | | ✓ | | |
| Storing a favourite song | | | ✓ | |
| Storing data to be imported into other files | | | | ✓ |

(1 mark per correct tick)
**Total 4 marks**

**7** Two answers accepted from:

- Each character will take up a byte of data. (1 mark)
- Each character has an individual code that can be represented in binary. (1 mark)
- Capital letters, lower case letters, numbers and punctuation all have their own binary code. (1 mark)
- Each binary code can also be presented in hexadecimal. (1 mark)
- The computer will process the binary code and reference it to the character set used (ASCII) to see what letter is pressed. (1 mark)

**Total 2 marks**