# 8 Databases

In this chapter, you will learn about

- the limitations of a file-based approach to storage and retrieval of data
- the features of a relational database that overcome the limitations of a file-based approach
- the terminology associated with a relational database model
- entity-relationship (E-R) diagrams to document database design
- normalisation to third normal form (3NF)
- producing a normalised database design for a given set of data or tables
- the features provided by a database management system (DBMS)
- the software tools provided by a DBMS
- the creation and modification of a database structure using a database definition language (DDL)
- queries and the maintenance of a database using a database manipulation language (DML)
- using SQL as a DDL and as a DML
- how to understand a given SQL script
- how to write an SQL script.

# 8.1 Database concepts

## WHAT YOU SHOULD ALREADY KNOW

Try these three questions before you read the first part of this chapter.

1 Databases are commonly used to store large amounts of data in a well organised way. Identify **three** databases that are storing information about you.

2 **a)** Name **three** of the most commonly used database management systems.

**b)** Give **four** benefits of using a database.

3 Relational databases use their own terminology.

**a)** Explain what the terms *122* and *122* mean.

**b)** Identify and explain the meaning of **three** or more terms used with relational databases.

**c)** What is a normalised relational database?

## Key terms

**Database** – a structured collection of items of data that can be accessed by different applications programs.

**Relational database** – a database where the data items are linked by internal pointers.

**Table** – a group of similar data, in a database, with rows for each instance of an entity and columns for each attribute.

**Record (database)** – a row in a table in a database.

**Field** – a column in a table in a database.

**Tuple** – one instance of an entity, which is represented by a row in a table.

**Entity** – anything that can have data stored about it, for example, a person, place, event, thing.

**Attribute (database)** – an individual data item stored for an entity, for example, for a person, attributes could include name, address, date of birth.

**Candidate key** – an attribute or smallest set of attributes in a table where no tuple has the same value.

**Primary key** – a unique identifier for a table. It is a special case of a candidate key.

**Secondary key** – a candidate key that is an alternative to the primary key.

**Foreign key** – a set of attributes in one table that refer to the primary key in another table.

**Relationship** – situation in which one table in a database has a foreign key that refers to a primary key in another table in the database.

**Referential integrity** – property of a database that does not contain any values of a foreign key that are not matched to the corresponding primary key.

**Index (database)** – a data structure built from one or more columns in a database table to speed up searching for data.

**Entity-relationship (E-R) model or E-R diagram** – a graphical representation of a database and the relationships between the entities.

**Normalisation (database)** – the process of organising data to be stored in a database into two or more tables and relationships between the tables, so that data redundancy is minimised.

**First normal form (1NF)** – the status of a relational database in which entities do not contain repeated groups of attributes.

**Second normal form (2NF)** – the status of a relational database in which entities are in 1NF and any non-key attributes depend upon the primary key.

**Third normal form (3NF)** – the status of a relational database in which entities are in 2NF and all non-key attributes are independent.

**Composite key** – a set of attributes that form a primary key to provide a unique identifier for a table.

## 8.1.1 The limitations of a file-based approach

A file is a collection of items of data. It can be structured as a collection of records, where each record is made up of fields containing data about the same ‘thing’. Individual elements of data can be called data items.

When a program is used for data processing, the organisation of any records used depends on how the program is written. Records can be fixed or variable in length and each record may also contain information about its structure, for example, the number of fields or the length of the record. If these records are to be processed by another program, that program must be written to use the exact same record structure. If the structure is changed by one program, the other program must be rewritten as well. This can cause problems if updating programs is not carefully managed.

For example, a business keeps separate payroll files and sales files. Each file is used by a different application.

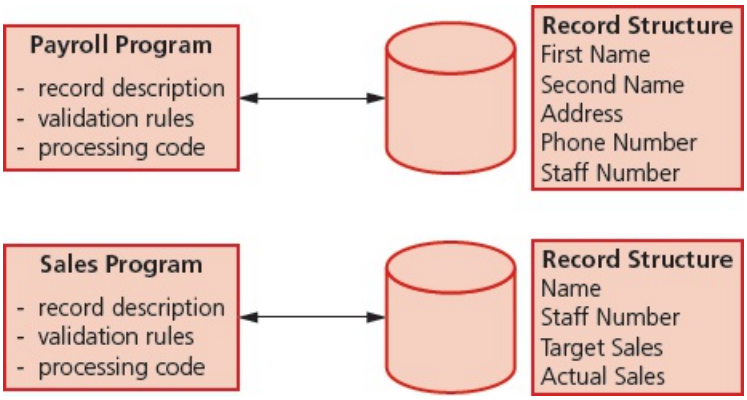



**Figure 8.1** File-based approach

Several problems have occurred using this file-based approach. The name of a member of staff and their staff number are stored twice. The way the staff name is stored is different for each program. If the staff number was changed by the payroll program and not by the sales program, these fields may contain different values for the same member of staff. The fields in the two files are also in a different order: the staff number is the fifth field in the payroll record and the second field in the sales file.

A file-based approach is limited because

- storage space is wasted when data items are duplicated by the separate applications and some data is redundant
- data can be altered by one application and not by another; it then becomes inconsistent
- enquiries available can depend on the structure of the data and the software used so the data is not independent.

### ACTIVITY 8A

Match the problems with the payroll and sales system to the limitations of a file-based approach set out above.

# 8.1.2 The advantages of a relational database over a file-based approach

What is a database? There are many different definitions of a database, such as:

> … A (large) collection of data items and links between them, structured in a way that allows it to be accessed by a number of different applications programs. The term is also used loosely to describe any collection of data.
>
> *BCS Glossary of Computing, 14th Edition*
>
> … An electronic filing cabinet which allows the user to perform various tasks including: adding new empty files, inserting data into existing files, retrieving data from existing files, updating data in existing files and cross-referencing data in files.
>
> *An Introduction to Database Systems (sixth edition) by CJ Date*

More straightforwardly, a **database** is a structured collection of items of data that can be accessed by different applications programs. Data stored in databases is structured as a collection of records, where each record is made up of fields containing data about the same 'thing'. A **relational database** is a database in which the data items are linked by internal pointers.

Using the same example as previously, a business keeps a database for payroll and sales data. A payroll application is used for the payroll and a sales processing application is used for sales.

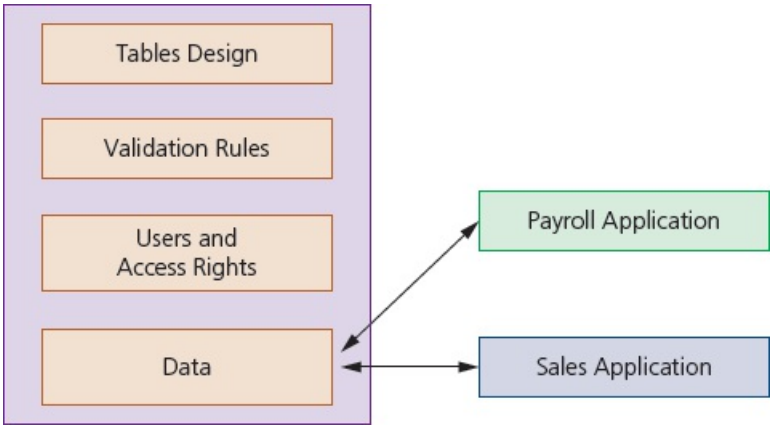



**Figure 8.2** Database approach

The problems that occurred using the file-based approach have been solved. The name of a member of staff and their staff number are only stored once. So, any changes made to the data by the payroll application will be seen by the sales processing application and vice versa. The fields are the same and in the same order.

A database approach is beneficial because

- storage space is not wasted as data items are only stored once, meaning little or no redundant data
- data altered in one application is available in another application, so the data is consistent
- enquiries available are not dependent on the structure of the data and the software used, so the data is independent.

## 8.1.3 Relational database model terminology

In order to rigorously define the structure of a relational database we need to be able to understand and use the terminology associated with a relational database.

A relational database data structure can look similar to a file-based structure as it also consists of records and fields. A **table** is a group of similar data, in a database, with rows for each instance of an entity and columns for each attribute. A **record** is a row in a table in a database. A **field** is a column in a table in a database.

For example, a database of students in a school could contain the table **Student** with a record for each student that contains the fields **First Name**, **Second Name**, **Date of Birth** and **Class ID**.

| First Name | Second Name | Date Of Birth | Class ID |
|---|---|---|---|
| Noor | Baig | 09/22/2010 | 7A |
| Ahmed | Sayed | 06/11/2010 | 7B |
| Tahir | Hassan | 01/30/2011 | 7A |

← row is a record

↑
column is a field

**Table 8.1** Part of a student table

Now data is independent of the program processing it. The terms record and field are also used in file processing, so there is more rigorous terminology used specifically for relational databases. Files of data are replaced by tables, with each row of a table representing a record (a **tuple**, sometimes called a logical record or an occurrence of an **entity**). Each column of the table is an **attribute** that can also be referred to as a field.

An entity is anything that can have data stored about it, such as a person, place, event or object. An attribute is an individual data item stored for an entity; to use the same example as before, for a student attributes could include first name, second name, date of birth and class. As stated before, a table is a group of similar data, in a database, with rows for each instance of an entity and columns for each attribute. A tuple is one instance of an entity, which is represented by a row in a table.

| First Name | Second Name | Date Of Birth | Class ID |
|---|---|---|---|
| Noor | Baig | 09/22/2010 | 7A |
| Ahmed | Sayed | 06/11/2010 | 7B |
| Tahir | Hassan | 01/30/2011 | 7A |

← each row is a tuple

↑
each column is an attribute

**Table 8.2** Part of a table for student entity

Data is shared between applications using the database. In order to ensure the consistency of data updating is controlled or automatic, so that any copies of a data item are changed to the new value. Also, in order to reduce the number of copies of a data item to a minimum, a relational database uses pointers between tables. These pointers are keys that provide relationships between tables.

There are different types of keys.

- A **candidate key** is an attribute or smallest set of attributes in a table where no tuple has the same value.
- A **primary key** is a unique identifier for a table, it is a special case of a candidate key.
- A **secondary key** is a candidate key that is an alternative to the primary key.
- A **foreign key** is a set of attributes in one table that refer to the primary key in another table.

For example, a database of chemical elements contains a table **Elements** with attributes **Symbol**, **Name** and **Atomic Weight**. As all these attributes are unique to each element, all are candidate keys. One of these could be chosen as the primary key, for example Symbol. Then the other two attributes, Name and Atomic Weight, would be secondary keys.

all attributes are candidate keys

| Symbol | Name | Atomic Weight |
|---|---|---|
| H | Hydrogen | 1.008 |
| Li | Lithium | 6.94 |
| Na | Sodium | 22.990 |

Symbol is the primary key

Name and Atomic Weight are secondary keys

**Table 8.3** Part of a table of elements

Most tables have only one candidate key, which is used as the primary key. For example, the student table could have an extra attribute Student ID, which is unique to each student.

| Student ID | First Name | Second Name | Date Of Birth | Class ID |
|---|---|---|---|---|
| S1276 | Noor | Baig | 09/22/2010 | 7A |
| S1277 | Ahmed | Sayed | 06/11/2010 | 7B |
| S2199 | Tahir | Hassan | 01/30/2011 | 7A |

Student ID is the primary key and the candidate key

**Table 8.4** Part of a table for student entity

## *Relationships*

A **relationship** is formed when one table in a database has a foreign key that refers to a primary key in another table in the database. In order to ensure **referential integrity** the database must not contain any values of a foreign key that are not matched to the corresponding primary key.

Most databases include more than one table. For example, a school database could contain the table **Student** and another table **Class** that contains the **Class ID**, the **Teacher Name** and **Location** of classroom. Only values for **Class ID** that are stored in the **Class** table can be used as the foreign key in the **Student** table.

| Student ID | First Name | Second Name | Date Of Birth | Class ID |
|---|---|---|---|---|
| S1276 | Noor | Baig | 09/22/2010 | 7A |
| S1277 | Ahmed | Sayed | 06/11/2010 | 7B |
| S2199 | Tahir | Hassan | 01/30/2011 | 7A |

↑
Class ID is the foreign key

**Table 8.5** Part of a table for student entity

| Class ID | Teacher Name | Location |
|---|---|---|
| 7A | Mr Khan | Floor 2 Room 3 |
| 7B | Miss Malik | Floor 2 Room 4 |
| 7C | Miss Gill | Floor 2 Room 5 |

↑
Class ID is the primary key

**Table 8.6** Part of a table for class entity

Relationships can take several forms

- one-to-one, 1:1
- one-to-many, 1:m
- many-to-one, m:1
- many-to-many, m:m.

The relationship between **Student** and **Class** is many-to-one, as one value of the attribute **Class ID** may appear many times in the **Student** table but only once in the **Class** table.

In order to speed up searching for data, an **index** can be used. This is a data structure built from one or more columns in a database table. The **Student** table could be indexed on **Class**, **Second Name** and **First Name** to provide class lists in alphabetical order of **Second Name**.

## 8.1.4 Entity-relationship (E-R) diagrams

An **E-R diagram** can be used to document the design of a database. This provides an easily understandable visual representation of how the entities in a database are related.

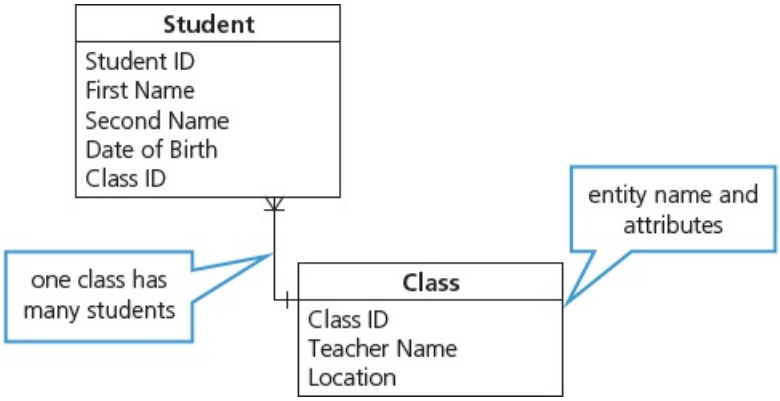



**Figure 8.3** E-R diagram for school database

Relationships may be mandatory or optional. For example, in a workroom with desks, each employee has one desk, but there could be spare desks. The relationship between desk and employee is zero or one, so this relationship is optional. The relationship between mother and child is mandatory because every mother must have at least one child, so the relationship is one or many. The type of relationship and whether it is mandatory or optional gives the cardinality of the relationship. The cardinality of relationships is shown in Figure 8.4.

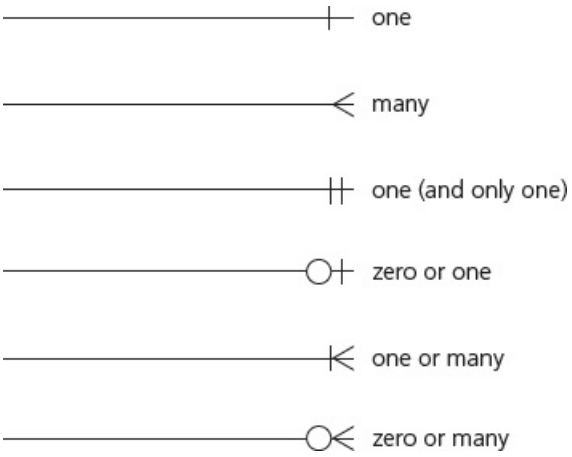



**Figure 8.4** Cardinality of relationships

### ACTIVITY 8B

The School database will also include the following details about each teacher:

- teaching licence number
- date of birth
- address.

A teacher can have more than one class. A table **Teacher** is to be added.

List the attributes for this new table. Show the change that should be made to the attributes in the **Class** table.

Draw the new E-R diagram for the three tables in the database.

## EXTENSION ACTIVITY 8A

In small groups, identify suitable entity relationships for each example of cardinality shown above. Explain your findings to another group or the whole class.

# 8.1.5 The normalisation process

**Normalisation** is used to construct a relational database that has integrity and in which data redundancy is reduced. Tables that are not normalised will be larger. As more data is stored, it will be harder to update the database when changes are made and more difficult to extract the required data to answer queries.

For example, if the **School** database is held in a single table it could look like this:

| Student ID | First Name | Second Name | Date Of Birth | Class ID | Location | Teacher Name | Licence Number | Address | Teacher Date Of Birth |
|---|---|---|---|---|---|---|---|---|---|
| S1276 | Noor | Baig | 09/22/2010 | 7A | Floor 2 Room 3 | Mr Khan | 37952 | School House 1 | 03/27/1985 |
| S1277 | Ahmad | Sayed | 06/11/2010 | 7B | Floor 2 Room 4 | Miss Malik | 68943 | School House 2 | 12/14/1988 |
| S1299 | Tahir | Hassan | 01/30/2011 | 7A | Floor 2 Room 3 | Mr Khan | 37952 | School House 1 | 03/27/1985 |

**Table 8.7**

This could cause problems when alterations are made to the records. Every time a new student is added, the teacher's name, address, licence number, date of birth, and the location of the classroom need to be added as well. If Mr Khan leaves the school and is replaced by another teacher, then every record containing his name and other details needs to be changed. If all the students from Class 7B leave, then all the details about Class 7B will be lost.

The rules for normalisation are set out as follows.

1. **First normal form (1NF)** – entities do not contain repeated groups of attributes.
2. **Second normal form (2NF)** – entities are in 1NF and any non-key attributes depend upon the primary key. There are no partial dependencies.
3. **Third normal form (3NF)** – entities are in 2NF and all non-key attributes are independent. The table contains no non-key dependencies.

When the database is in 3NF, all attributes in a table depend upon the key, the whole key and nothing but the key.

The School database also includes subject choices for each student. For this database to be normalised, the process is:

| Student ID | First Name | Second Name | Date Of Birth | Subject Name | Subject Teacher | Class ID | Location | Teacher Name | Licence Number | Address | Teacher Date Of Birth |
|---|---|---|---|---|---|---|---|---|---|---|---|
| S1276 | Noor | Baig | 09/22/2010 | Maths, History, Geography | Mr Yee, Miss Wu, Mr Khan | 7A | Floor 2 Room 3 | Mr Khan | 37952 | School House 1 | 03/27/1985 |
| S1277 | Ahmad | Sayed | 06/11/2010 | Maths, Science, Geography | Mr Yee, Miss Yo, Mr Khan | 7B | Floor 2 Room 4 | Miss Malik | 68943 | School House 2 | 12/14/1988 |
| S1299 | Tahir | Hassan | 01/30/2011 | Maths, Science, History | Mr Yee, Miss Yo, Miss Wu | 7A | Floor 2 Room 3 | Mr Khan | 37952 | School House 1 | 03/27/1985 |

**Table 8.8**

## *First normal form (1NF)*

The un-normalised School database can be represented as follows.

**STUDENT**(StudentID, FirstName, SecondName, DateOfBirth, SubjectName, SubjectTeacher, SubjectName, SubjectTeacher, SubjectName, SubjectTeacher, ClassID, Location, TeacherName, LicenceNumber, Address, TeacherDateOfBirth).

**STUDENT** is the table name; the attributes are listed in order and the primary key is underlined.

The student's subjects and the subject teacher are the repeating attributes. For the database to be in first normal form, these need to be removed to a separate table and linked to the original table with a foreign key.

| Student ID | First Name | Second Name | Date Of Birth | Class ID | Location | Teacher Name | Licence Number | Address | Teacher Date Of Birth |
|---|---|---|---|---|---|---|---|---|---|
| S1276 | Noor | Baig | 09/22/2010 | 7A | Floor 2 Room 3 | Mr Khan | 37952 | School House 1 | 03/27/1985 |
| S1277 | Ahmad | Sayed | 06/11/2010 | 7B | Floor 2 Room 4 | Miss Malik | 68943 | School House 2 | 12/14/1988 |
| S1299 | Tahir | Hassan | 01/30/2011 | 7A | Floor 2 Room 3 | Mr Khan | 37952 | School House 1 | 03/27/1985 |

| Student ID | Subject Name | Subject Teacher |
|---|---|---|
| S1276 | Maths | Mr Yee |
| S1276 | History | Miss Wu |
| S1276 | Geography | Mr Khan |
| S1277 | Maths | Mr Yee |
| S1277 | Science | Miss Yo |
| S1277 | Geography | Mr Khan |
| S1299 | Maths | Mr Yee |
| S1299 | Science | Miss Yo |
| S1299 | History | Miss Wu |

**Table 8.9** School database in 1NF

The School database can now be represented in 1NF as follows.

**STUDENT**(StudentID, FirstName, SecondName, DateOfBirth, ClassID, Location, TeacherName, LicenceNumber, Address, TeacherDateOfBirth).

**STUDENTSUBJECT**(StudentID, SubjectName, SubjectTeacher).

The primary key for the **STUDENTSUBJECT** table is a **composite key** formed from the two attributes **StudentID** and **SubjectName**; the attribute **StudentID** is also a foreign key that links to the **STUDENT** table.

## *Second normal form (2NF)*

There are now two tables; in the **STUDENTSUBJECT** table the primary key is a composite key and the **SubjectTeacher** is only dependent on the **SubjectName** part of the primary key. This is a partial dependence and needs to be removed by introducing a third table, **SUBJECT**.

| Student ID | First Name | Second Name | Date Of Birth | Class ID | Location | Teacher Name | Licence Number | Address | Teacher Date Of Birth |
|---|---|---|---|---|---|---|---|---|---|
| S1276 | Noor | Baig | 09/22/2010 | 7A | Floor 2 Room 3 | Mr Khan | 37952 | School House 1 | 03/27/1985 |
| S1277 | Ahmad | Sayed | 06/11/2010 | 7B | Floor 2 Room 4 | Miss Malik | 68943 | School House 2 | 12/14/1988 |
| S1299 | Tahir | Hassan | 01/30/2011 | 7A | Floor 2 Room 3 | Mr Khan | 37952 | School House 1 | 03/27/1985 |

| Student ID | Subject Name |
|---|---|
| S1276 | Maths |
| S1276 | History |
| S1276 | Geography |
| S1277 | Maths |
| S1277 | Science |
| S1277 | Geography |
| S1299 | Maths |
| S1299 | Science |
| S1299 | History |

| Subject Name | Subject Teacher |
|---|---|
| Maths | Mr Yee |
| History | Miss Wu |
| Geography | Mr Khan |
| Science | Miss Yo |

**Table 8.10** School database in 2NF

The School database can now be represented in 2NF as follows.

**STUDENT**(StudentID, FirstName, SecondName, DateOfBirth, ClassID, Location, TeacherName, LicenceNumber, Address, TeacherDateOfBirth)

**STUDENTSUBJECT**(StudentID, SubjectName)

**SUBJECT**(SubjectName, SubjectTeacher)

## *Third normal form (3NF)*

There are now three tables. In the **STUDENT** table, the attributes **Location** and **TeacherName** depend upon the attribute **ClassID** and the attributes **LicenceNumber**, **Address** and **TeacherDateOfBirth** depend upon the attribute **TeacherName**. These are non-key dependencies that need to be removed to ensure that the database is in 3NF.

At this stage it is also worth inspecting the database and its contents to consider any other problems that could arise, such as the following:

- Teacher names might not be unique; therefore, it is better to use the licence number as a primary key.
- Teachers can be both class teachers and subject teachers; these need to be combined in one table.

| Student ID | First Name | Second Name | Date Of Birth | Class ID |
|---|---|---|---|---|
| S1276 | Noor | Baig | 09/22/2010 | 7A |
| S1277 | Ahmad | Sayed | 06/11/2010 | 7B |
| S1299 | Tahir | Hassan | 01/30/2011 | 7A |

| Licence Number | Teacher Name | Address | Teacher Date Of Birth |
|---|---|---|---|
| 37952 | Mr Khan | School House 1 | 03/27/1985 |
| 68943 | Miss Malik | School House 2 | 12/14/1988 |
| 35859 | Mr Yee | School House 1 | 10/07/1985 |
| 77248 | Miss Yo | School House 2 | 05/05/1987 |
| 72691 | Miss Wu | School House 2 | 11/21/1989 |
| 37952 | Mr Khan | School House 1 | 03/27/1985 |

| Class ID | Location | Licence Number |
|---|---|---|
| 7A | Floor 2 Room 3 | 37952 |
| 7B | Floor 2 Room 4 | 68943 |

| Student ID | Subject Name |
|---|---|
| S1276 | Maths |
| S1276 | History |
| S1276 | Geography |
| S1277 | Maths |
| S1277 | Science |
| S1277 | Geography |

| | |
|---|---|
| S1299 | Maths |
| S1299 | Science |
| S1299 | History |

| Subject Name | Licence Number |
|---|---|
| Maths | 35859 |
| History | 72691 |
| Geography | 37952 |
| Maths | 77248 |

**Table 8.11** School database in 3NF

The improved School database can now be represented in 3NF as follows.

**STUDENT**(StudentID, FirstName, SecondName, DateOfBirth,)

**CLASS**(ClassID, Location, LicenceNumber)

**TEACHER**(LicenceNumber, TeacherName, Address, TeacherDateOfBirth)

**STUDENTSUBJECT**(StudentID, SubjectName)

**SUBJECT**(SubjectName, LicenceNumber).

## ACTIVITY 8C

Construct an E-R diagram to represent the database structure of the fully normalised school database shown above.

## EXTENSION ACTIVITY 8B

Discuss any other possible problems that could occur with this database.

Hint: look at the subject table and think about subjects that could have more than one teacher or different levels. Identify an improved database structure that could solve the problem.

The School database example showed at each stage why the database was not normalised. Here is another example for you to try.

A database has been set up as a single table to store employees of a business and their contacts. Part of the database is shown below.

| Employee Number | Employee Name | Position | Contact Number | Contact Name | Contact Email Address |
|---|---|---|---|---|---|
| 7001 | James Tey | Financial Director | 28 | Mary Jones | mary@xyz.com |
| | | | 31 | James Smith | james@pqr.com |
| | | | 17 | Mishal Hussani | mh@xyz.com |
| 7002 | Paul Leigh | Accountant | 19 | Mary Cheung | mch@abc.com |
| | | | 27 | Dean Knight | knd@swz.com |
| 7011 | Suzy Mey | Personnel Manager | 28 | Mary Jones | mary@xyz.com |

**Table 8.12** Un-normalised employee database

This table is not in 1NF because there are repeating attributes and the table is not in 3NF because there are non-key dependencies. The employee database can be represented as:

**EMPLOYEE**(EmployeeNumber, EmployeeName, Position, ContactNumber, ContactName, ContactEmailAddress).

Where **EmployeeNumber** is the primary key **ContactNumber**, **ContactName** and **ContactEmailAddress** may be repeated as often as required.

## ACTIVITY 8D

Normalise the Employee database and show the new tables. Draw the E-R diagram for the normalised database.

## ACTIVITY 8E

**1 a) i)** Describe the limitations of a file-based approach to storage and retrieval of data.

**ii)** Give **two** benefits of using a database management system.

**b)** A new relational database is to be developed. The developer needs to produce a normalised database design.

**i)** Explain what is meant by *normalisation*.

**ii)** Describe the process of normalisation.

**2** A warehouse stores parts for cars for several manufacturers. A database stores the following data for each part:
Part number, part description, date last ordered, minimum order level, manufacturer name, manufacturer address, manufacturer contact details, position in warehouse, number in stock

**a)** Design a fully normalised database for the parts.

**b)** Draw the E-R diagram.

# 8.2 Database management systems (DBMSs)

## WHAT YOU SHOULD ALREADY KNOW

Try these two questions before you read the second part of this chapter.

**1 a)** Name a database management system (DBMS) you have used.

**b)** Describe **three** tasks that you have used the DBMS for.

**2** Most DBMSs include back-up procedures and access rights to keep the data secure.

**a)** Describe what is meant by back-up.

**b)** Describe what is meant by access rights.

**c)** How do these features help to keep data secure?

## Key terms

**Database management system (DBMS)** – systems software for the definition, creation and manipulation of a database.

**Data management** – the organisation and maintenance of data in a database to provide the information required.

**Data dictionary** – a set of data that contains metadata (data about other data) for a database.

**Data modelling** – the analysis and definition of the data structures required in a database and to produce a data model.

**Logical schema** – a data model for a specific database that is independent of the DBMS used to build that database.

**Access rights (database)** – the permissions given to database users to access, modify or delete data.

**Developer interface** – feature of a DBMS that provides developers with the commands required for definition, creation and manipulation of a database.

**Structured query language (SQL)** – the standard query language used with relational databases for data definition and data modification.

**Query processor** – feature of a DBMS that processes and executes queries written in structured query language (SQL).

# 8.2.1 How a DBMS addresses the limitations of a file-based approach

## *Data redundancy issue*

This is solved by storing data in separate linked tables, which reduces the duplication of data as most items of data are only stored once. Items of data used to link tables by the use of foreign keys are stored more than once. The DBMS will flag any possible errors when any attempt is made to accidentally delete this type of item.

## *Data inconsistency issue*

This is also solved by storing most items of data only once, allowing updated items to be seen by all applications. As data is not inconsistent, the integrity of the data stored is improved. Consistent data is easier to maintain as an item of data will only be changed once, not multiple times, by different applications.

## *Data dependency issue*

Data is independent of the applications using the database, so changes made to the structure of the data will be managed by the DBMS and have little or no effect on the applications using the database. Any fields or tables added to or removed from the database will not affect the applications that do not use those fields/tables, as each application only has access to the fields/tables it requires.

Information from a database is more easily available in a form that is required so it is not dependent on the structure of the data and the application used. A DBMS usually includes facilities to query the data stored using a defined query language or a query-by-example facility.

## *The DBMS approach*

A **DBMS** uses a more structured approach to the **management**, organisation and maintenance of data in a database. An already-defined data structure can be used to set up and create the database. The entry of new data, the storage of data, the alteration and deletion of data are all managed by the DBMS.

A DBMS uses a **data dictionary** to store the metadata, including the definition of tables, attributes, relationships between tables and any indexing. The data dictionary can also define the validation rules used for the entry of data and contain data about the physical storage of the data. The use of a data dictionary improves the integrity of the data stored, helping to ensure that it is accurate, complete and consistent.

**Data modelling** is an important tool used to show the data structure of a database. An E-R diagram is an example of a data model. A **logical schema** is a data model for a specific database that is independent of the DBMS used to build the database.

A DBMS helps to provide data security to prevent the unwanted alteration, corruption, deletion or sharing of data with others that have no right to access it.

Security measures taken by a DBMS can include

- using usernames and passwords to prevent unauthorised access to the database
- using access rights to manage the actions authorised users can take, for example, users could read/write/delete, or read only, or append only
- using **access rights** to manage the parts of the database they have access to, for example, the provisions of different views of the data for different users to allow only certain users access to some tables
- automatic creation and scheduling of regular back-ups
- encryption of the data stored
- automatic creation of an audit trail or activity log to record the actions taken by users of the database.

# 8.2.2 The use and purpose of DBMS software tools

## *Developer interface*

The **developer interface** allows a developer to write queries in **structured query language (SQL)** rather than using query-by-example. These queries are then processed and executed by the **query processor**. This allows the construction of more complex queries to interrogate the database.

## *Query processor*

The query processor takes a query written in SQL and processes it. The query processor includes a DDL interpreter, a DML compiler and a query evaluation engine. Any DDL statements are interpreted and recorded in the database's data dictionary. DML statements are compiled into low level instructions that are executed by the query evaluation engine. The DML compiler will also optimise the query.

### ACTIVITY 8F

1 a) Describe how a DBMS overcomes the limitations of a file-based approach to the storage and retrieval of data.

b) Describe how a DBMS ensures that data stored in a database is secure.

2 a) Describe **three** features provided by a DBMS.

b) A school stores timetabling data for all pupils and classes.
Which features could a DBMS use to ensure that the administrators, teachers and pupils can only see the information available to them?

# 8.3 Data definition language (DDL) and data manipulation language (DML)

## WHAT YOU SHOULD ALREADY KNOW

Try this exercise before you read the third part of this chapter.
Using a DBMS with a graphical user interface (GUI), create the student database used in Section 8.1.5.
Write the following queries using a query-by-example form.

1. A list of all the teachers and their subjects.
2. A list of the pupils in class 7A in alphabetical order of second name.
3. A list of the students studying each subject.

You may want to save this database to practise your SQL commands.

## Key terms

**Data definition language (DDL)** – a language used to create, modify and remove the data structures that form a database.

**Data manipulation language (DML)** – a language used to add, modify, delete and retrieve the data stored in a relational database.

**SQL script** – a list of SQL commands that perform a given task, often stored in a file for reuse.

## 8.3.1 Industry standard methods for building and modifying a database

DBMSs use a **data definition language (DDL)** to create, modify and remove the data structures that form a relational database. DDL statements are written as a script that uses syntax similar to a computer program.

DBMSs use a **data manipulation language (DML)** to add, modify, delete and retrieve the data stored in a relational database. DML statements are written in a script that is similar to a computer program.

These languages have different functions: DDL is used for working on the relational database structure, whereas DML is used to work with the data stored in the relational database.

Most DBMSs use structured query language (SQL) for both data definition and data manipulation. SQL was developed in the 1970s and since then it has been adopted as an industry standard.

## 8.3.2 SQL (DDL) commands and scripts

In order to be able to understand and write SQL, you should have practical experience of writing **SQL scripts**. There are many applications that allow you to do this. For example, MySQL and SQLite are freely available ones. When using any SQL application it is important that you check the commands available to use as these may differ slightly from those listed below.

You will need to be able to understand and use the following DDL commands.

| SQL (DDL) command | Description |
|---|---|
| CREATE DATABASE | Creates a database |
| CREATE TABLE | Creates a table definition |
| ALTER TABLE | Changes the definition of a table |
| PRIMARY KEY | Adds a primary key to a table |
| FOREIGN KEY … REFERENCES … | Adds a foreign key to a table |

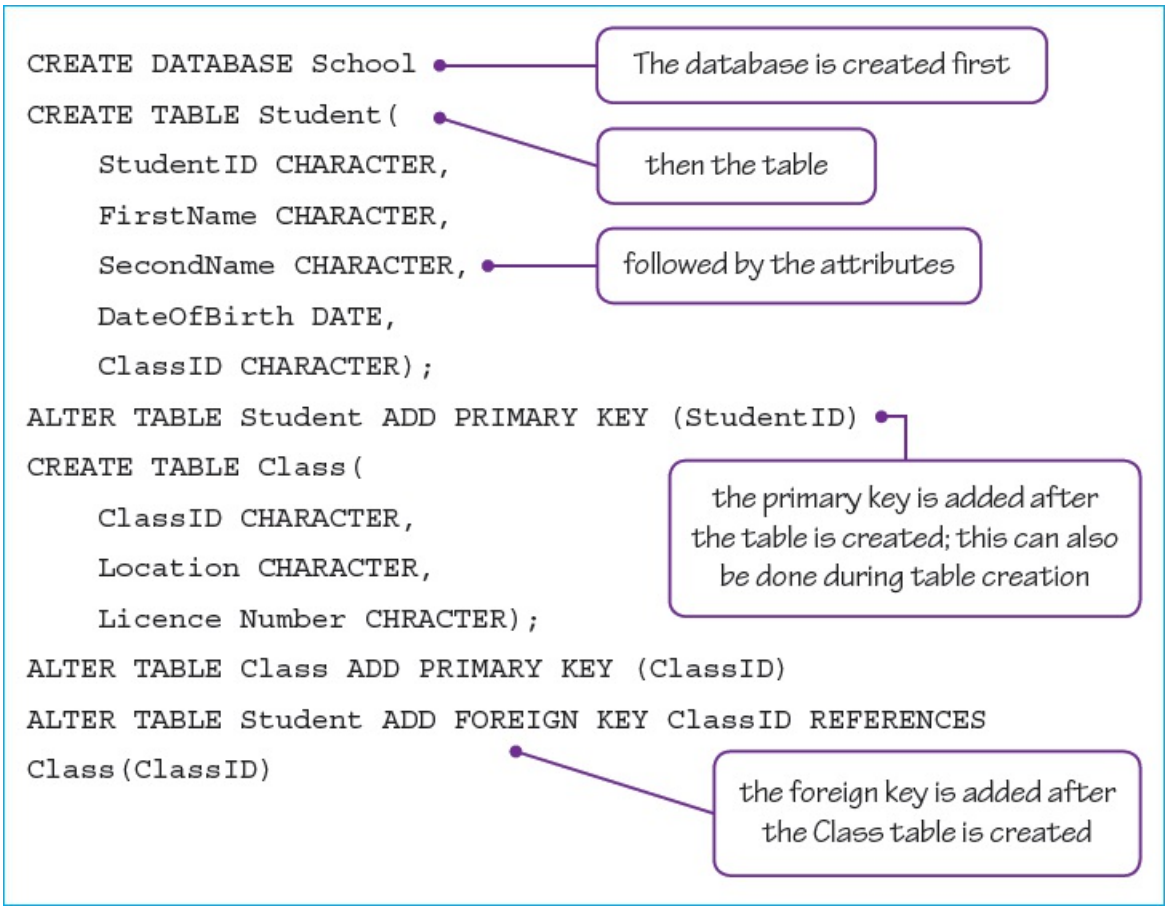
**Table 8.13** DDL commands

You also need to be familiar with the following data types used for attributes in SQL.

| Data types for attributes | Description |
|---|---|
| CHARACTER | Fixed length text |
| VARCHAR(n) | Variable length text |
| BOOLEAN | True or False; SQL uses the integers 1 and 0 |
| INTEGER | Whole number |
| REAL | Number with decimal places |
| DATE | A date usually formatted as YYYY-MM-DD |
| TIME | A time usually formatted as HH:MM:SS |

**Table 8.14** Data types for attributes

Here are some examples of DDL that could have been used when the school database was created.




## ACTIVITY 8G

Create the Teacher table and add the **Licence Number** as a foreign key to the Class table.

## 8.3.3 SQL (DML) commands and scripts

In order to be able to understand and write SQL, you should have practical experience of writing SQL scripts and queries. There are many applications that allow you to do this. Again, MySQL and SQLite are freely available ones. You can also write SQL commands in Access. When using any SQL application, it is important that you check the commands available to use as these may differ slightly from those listed below.

You will need to be able to understand and use the following DML commands.

| SQL (DML) query command | Description |
|---|---|
| SELECT FROM | Fetches data from a database. Queries always begin with SELECT. |
| WHERE | Includes only rows in a query that match a given condition |
| ORDER BY | Sorts the results from a query by a given column either alphabetically or numerically |
| GROUP BY | Arranges data into groups |
| INNER JOIN | Combines rows from different tables if the join condition is true |
| SUM | Returns the sum of all the values in the column |
| COUNT | Counts the number of rows where the column is not NUL |
| AVG | Returns the average value for a column with a numeric data type |

| SQL (DML) maintenance commands | Description |
|---|---|
| INSERT INTO | Adds new row(s) to a table |
| DELETE FROM | Removes row(s) from a table |
| UPDATE | Edits row(s) in a table |

**Table 8.15** DML commands

Here are some examples of DML that could have been used to query and update the school database.

This query will show, in alphabetical order of second name, the first and second names of all students in class 7A:

```
SELECT FirstName, SecondName
FROM Student
WHERE ClassID = '7A'
ORDER BY SecondName
```

This query will show the teacher's name and the subject taught:

```
SELECT Teacher.TeacherName AND Subject.SubjectName
FROM Teacher INNER JOIN Subject ON Teacher.
LicenceNumber = Subject.LicenceNumber
```

## ACTIVITY 8H

Create a query to show each student's First Name, Second Name and the subjects studied by each student.

This statement will insert a row into the Student table:

```
INSERT INTO Student VALUES(S1301, Peter, Probert,
06/06/2011, 7A)
```

If the values for all the columns are not known, then the table columns need to be specified before the values are inserted:

```
INSERT INTO Student(StudentID, FirstName, SecondName)
VALUES(S1301, Peter, Probert)
```

These statements will delete the specified row(s) from the Student table (take care: `DELETE FROM Student` will delete the whole table!):

```
DELETE FROM Student
WHERE StudentID = 'S1301'
```

The values for any column can be counted, totalled or averaged.

For example, if an extra column was added to the **STUDENTSUBJECT** table showing each student's exam mark in that subject, the following query could be used to total all of the students' exam marks:

```
SELECT SUM (ExamMark)
FROM STUDENTSUBJECT
```

## ACTIVITY 8I

Use the SQL statements AVG and COUNT to find the average mark and count how many marks have been recorded.

# End of chapter questions

**1** A database has been designed to store data about programmers and the programs they have developed.

These facts help to define the structure of the database:

– Each programmer works in a particular team.

– Each programmer has a unique first name.

– Each team has one or more programmer.

– Each program is for one customer only.

– Each programmer can work on any program.

– The number of days that each programmer has worked on a program is recorded.

The table ProgDev was the first attempt at designing the database.

| FirstName | Team | ProgramName | NoOfDays | Customer |
|---|---|---|---|---|
| Alice | WC | TV control | 3 | SKM |
| | | Ice alert | 2 | WZP |
| | | Digital camera | 6 | HNC |
| Charles | PC | Oil flow | 1 | GEB |
| | | Rescue Pack | 8 | BGF |
| Ahmad | QR | TV control | 2 | SKM |
| | | Accounts | 8 | ARC |
| | | Digital camera | 4 | HNC |
| | | Test Pack | 3 | GKN |

**a)** State why the table is **not** in first normal form (1NF).

[1]

**b)** The database design is changed to:

```
Programmer (FirstName, Team)
Program (FirstName, ProgramName, NoOf Days,
Customer)
```

| **Table:** Programmer | |
|---|---|
| FirstName | Team |

| | |
|---|---|
| | |
| | |

| Table: Program | | | |
|---|---|---|---|
| FirstName | ProgramName | NoOfDays | Customer |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

**c)** **i)** A relationship between the two tables has been implemented.
Explain how this has been done.

[2]

**ii)** Explain why the `Program` table is **not** in third normal form (3NF).

[2]

**iii)** Write the table definitions to give the database in 3NF.

[2]

**2** A school stores a large amount of data. This includes student attendance, qualification, and contact details. The school's software uses a file-based approach to store this data.

**a)** The school is considering changing to a DBMS.

**i)** State what DBMS stands for.

[1]

**ii)** Describe **two** ways in which the Database Administrator (DBA) could use the DBMS software to ensure the security of the student data.

[4]

**iii)** A feature of the DBMS software is a query processor.
Describe how the school secretary could use this software.

[2]

**iv)** The DBMS has replaced software that used a file-based approach with a relational database.
Describe how using a relational database has overcome the previous problems associated with a file-based approach.

[3]

**b)** The database design has three tables to store the classes that students attend.

**STUDENT**(StudentID, FirstName, LastName, Year, TutorGroup)
**CLASS**(ClassID, Subject)
**CLASS-GROUP**(StudentID, ClassID)

Primary keys are not shown.

There is a one-to-many relationship between **CLASS** and **CLASS-GROUP**.

**i)** Describe how this relationship is implemented.

[2]

**ii)** Describe the relationship between **CLASS-GROUP** and **STUDENT**.

[1]

**iii)** Write an SQL script to display the `StudentID` and `FirstName` of all students who are in the tutor group 10B.
Display the list in alphabetical order of `LastName`.

[4]

**iv)** Write an SQL script to display the `LastName` of all students who attend the class whose ClassID is CS1.

[4]

*Cambridge International AS & A Level Computer Science 9608*
*Paper 12 Q8 June 2016*