Teaching Guide: Relational databases and structured query language (SQL)

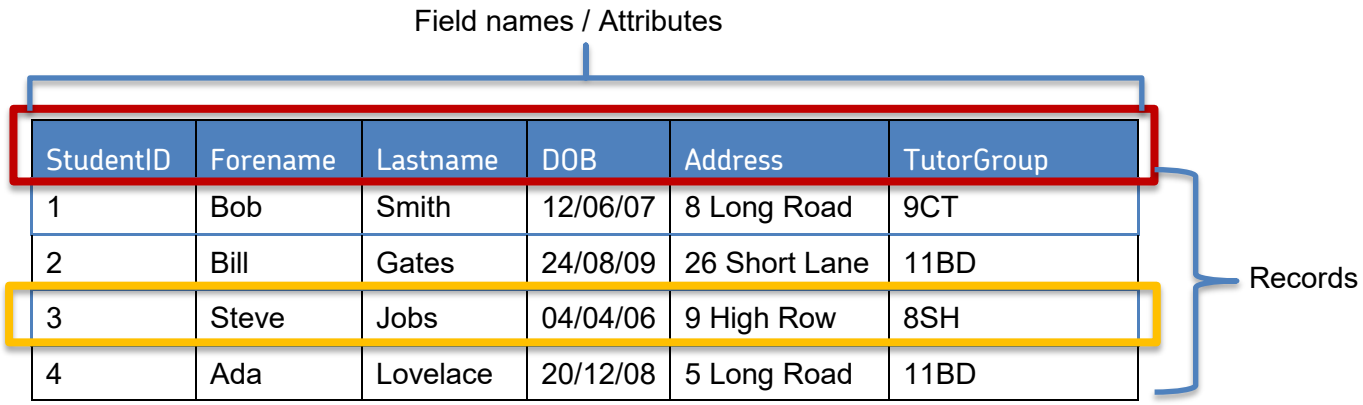
This resource is designed to help you to understand the key concepts surrounding section *3.7.1 Relational databases* and *3.7.2 Structured query language (SQL)* of the 8525 GCSE Computer Science specification.

Introduction to databases

Databases are a key resource for both programmers and computer science practitioners, eg all websites that have some form of login usually have a database that supports it.

How data is laid out in a database

Data in a database is stored in a **table.** Each table has a number of key elements:



StudentID	Forename	Lastname	DOB	Address	TutorGroup
1	Bob	Smith	12/06/07	8 Long Road	9CT
2	Bill	Gates	24/08/09	26 Short Lane	11BD
3	Steve	Jobs	04/04/06	9 High Row	8SH
4	Ada	Lovelace	20/12/08	5 Long Road	11BD

In the table example above, the field names are shown at the top of the table in blue. Field names are used to refer to pieces of information within a record in a table. Every record in a particular table will have the same fields within it.

A record is a number of related pieces of information grouped together eg all the information that a school holds about a particular student.

Relating this information to students is usually better by forming sentences, e.g:

- Every student has a LastName *field*
- All the information about one student is grouped together in a *record*
- All the students are grouped together in a *table*

Types of Database

Databases come in two main types:

- Flat file databases - where data is stored in a single table
- Relational databases - where data is stored in multiple linked tables.

Benefits of relational databases over flat file databases

The advantage of keeping all the information in one table (flat file), is that it is easier to set up. The disadvantages are that it is harder to manage and takes up more space.

Flat file databases store all the information in one table. This usually means that the same data is included multiple times, which can lead to inconsistencies in the data, potential mistakes and a database that is larger than it need be.

A relational database stores related information in separate tables. This means that individual pieces of information are only stored once, thus preventing any inconsistency in the data and also saving space. For example, in a relational database for a school, the data for the teacher of class 11BD, Mrs Burgess, would be stored only once, no matter how many students she taught.

Any duplicated information that is then not needed is, by definition, redundant. Any data that is seen to be duplicated within a database is called **REDUNDANCY**.

Linking information together

When a relational database is designed, the method used to link information in different tables together uses two special fields within the tables; a *primary key* and a *foreign key*. Consider the following example:

This is called an *entity relationship diagram*. An *entity* is described as a thing, object, person or relationship. The diagram is a way of showing how all the information links together. An entity can have any number of *attributes* (*attribute* is another name for *field*).

In this example, the ‘crow’s feet’ demonstrate the link between the tables. We have three tables in the example above. A Students table, a Teachers table and a linking table called StudentTeacher. As each student can be taught by many teachers and each teacher can teach many students we need the linking table to store which teachers teach which students.

A primary key is a field that is unique for each record within a table. The way of linking the primary key with another table, is by using a foreign key. A foreign key is

an attribute in one table that contains the value of a primary key in another table. Primary keys **cannot** be duplicated whereas foreign keys can be stored multiple times in the same table.

In the example above possible field names for the primary keys are:

Table Name	Primary Key
Students	StudentID
StudentTeacher	StudentID, TeacherID
Teachers	TeacherID

Notice how the linked table has a *composite primary key* made up of the primary key fields from the individual tables it is linking together.

When they appear in a linked table they are called foreign keys rather than primary keys and their value can be duplicated.

In a composite key each individual value from the primary key of a single table can be repeated but not the combination of the primary keys. (If there were a repeated composite value of StudentID, TeacherID this would mean that the same student was being taught twice by the same teacher: if this was the case then there would need to be another primary key from say the Class table, showing that a student was taught by the same teacher twice, but in different classes).

The linked StudentTeacher table would look something like the example below:

StudentID	TeacherID	ClassCode	SubjectCode
1	9	8B	Phy
2	6	7A	CS
1	3	10C	Ma
4	9	8B	Phy
3	3	10C	Ma
1	6	7A	CS
3	1	9D	Ch

Notice how, at the moment, this table contains redundant data. The subject code and class code are repeated within the table. This would be a badly designed database if it was left like this. In reality additional tables would be created to store the subjects and the classes, with the subject and class primary keys added to this table, thus removing the duplication.

SQL

When you have a lot of related information within a database, the challenge is to find it, insert (add) more information, update (change) it or delete it and to do all of these things quickly and efficiently. SQL is a programming language that is used to manipulate information within a database.

Retrieval of data from a database - the `SELECT` Statement

The SELECT statement is the simplest of SQL statements and is used to find information within a database.

A SELECT statement comes in four main parts:

Key Term	Description
`SELECT`	The information you want to find within the database
`FROM`	The table(s) that contain(s) the information
`WHERE`	The criteria you need to find the information
`ORDER BY`	This is how the data is organized (sorted) when it is retrieved.

Example:

```
SELECT
    Forename, Lastname
FROM
    Students
WHERE
    StudentID < 10
ORDER BY
    Lastname, Forename ASC
```

This example will return only the `Forename` and `Lastname` of all the students from the `Students` table who have a `StudentID` of less than 10.

Cross table queries

When you want to return data from more than one table there are two ways it can be done. The original method is to use a `WHERE` statement but the more modern method is to use a `JOIN` statement.

For the purposes of the examinations it will not matter which method pupils use in their answers.

The original method

When combining data from multiple tables the same SELECT statement is used:

```
SELECT
    Students.Forename, Students.Lastname,
    Teachers.Teachername
FROM
    Teachers, StudentTeacher, Students
WHERE
    Students.StudentID = StudentTeacher.StudentID
AND StudentTeacher.TeacherID = Teachers.TeacherID
```

The more modern method

```
SELECT Students.Forename, Students.Lastname,
Teachers.Teachername
FROM Teachers
INNER JOIN (Students INNER JOIN StudentTeacher ON
Students.StudentID = StudentTeacher.StudentID) ON
Teachers.TeacherID = studentTeacher.TeacherID
```

Both versions of this example will return the `Forename` and `Lastname` of all of the students in the `Students` table along with their teacher names from the `Teachers` table. The results will not be specifically sorted.

Ordering information

When retrieving data from a database, you can specify the order in which the data is returned. This is done using the **`ORDER BY`** command as we saw earlier.

The **`ORDER BY`** command has two options, namely `ASC` and `DESC` (if you have **`ORDER BY`** but don't give either `ASC` or `DESC` then `ASC` is used):

Ordering data in ascending order	Ordering data in descending order
`ORDER BY Lastname ASC`	`ORDER BY Lastname DESC`

You can also order data by multiple fields:

```
SELECT
    Forename, Lastname, TutorGroup
FROM
    Students
ORDER BY
    TutorGroup, Lastname
```

In the above example, data is first organized by `TutorGroup` and then by `Lastname` within each tutor group.

Inserting data into a database

In order to add information into a table within a database using SQL, an INSERT statement is used.

The SQL INSERT statement has the following structure:

```
INSERT INTO
    Table (field1, field2, field3, …)
VALUES
    (value1, value2, value3, …)
```

Example:

```
INSERT INTO
    Students (Lastname, Forename, TutorGroup)
VALUES
    ('Smith', 'Bob', '10RB')
```

Notice the use of quotation marks when entering string or text data.

Removing data from a database

In order to delete information from a database using SQL, a delete statement is used.

The SQL DELETE statement has the following structure:

```
DELETE FROM
    Table
WHERE
    Condition
```

Example:

```
DELETE FROM
    Students
WHERE
    StudentID = 1
```

This example would delete one record if a record with a StudentID of 1 existed or zero records if it did not exist.

Editing/updating data in a database

In order to edit or update information within a database using SQL, an UPDATE statement is used.

The SQL UPDATE statement has the following structure:

```
UPDATE
    Table
SET
    field1 = value1, field2 = value2, …
WHERE
    Condition
```

Example:

```
UPDATE
    Students
SET
    Forename = 'Bob', Lastname = 'Smith'
WHERE
    StudentID = 1
```

You can also use an UPDATE statement to update multiple records at once by using the following format:

```
UPDATE
    Students
SET
    TutorGroup = '12HT'
WHERE
    TutorGroup = '11BD'
```

Glossary of key terms

Key Term	Definition
Attribute	an individual fact, detail or characteristic of an entity (also commonly called a field)
Database	a persistent store of related information
Entity	a thing, person, object or relationship about which data can be collected
Field	a single piece of information about an entity (see attribute)
Flat file database	a database containing a single table
Foreign key	a field in the table which contains the value of a primary key in another table (since the primary key is unique this means that the foreign key identifies, or links to, just one record
Primary key	a unique piece of information within a table to identify a record
Redundancy	the duplication of data within a database
Record	a collection of fields/attributes about the same thing, person, object or relationship in a table
Relational database	a database with multiple linked tables
Table	a collection of related data