AQA Computer Science GCSE

3.7 Relational databases and structured query language (SQL)

Advanced Notes

3.7.1 Relational databases

A database is an organised and structured collection of data that can be easily stored, searched, and updated.

A table holds related data in rows and columns, like a spreadsheet. A relational database is a database with multiple tables that are linked together using keys. The use of a relational database facilitates the elimination of data inconsistency (where data points are stored in more than one place, but with different values) and data redundancy (where data is unnecessarily duplicated in more than one place, which can lead to inconsistencies and wasted storage space).

Summary table

Term	Description
Database	A structured collection of data
Table	A collection of related data entries. This data is held in rows and columns, like a spreadsheet.
Record	A row in a table - one complete set of data about a single item
Field	A column in a table - stores one type of data (e.g. name, ID)
Primary Key	A field (or combination of fields) that uniquely identifies each record in a table.
Foreign Key	A field (or combination of fields) in one table that links to the primary key in another table

Primary and foreign keys

A primary key is an attribute that provides a unique identifier for every record in a database table. When tables are linked by a shared attribute, the attribute must be a primary key in one table and is called a foreign key in the other.

A foreign key is an attribute in a table which is the primary key in another, related table.

If it is not possible to form a primary key from just one field type, it is possible to combine field types to form what is called a composite primary key.

Example: The following database tables are related to one another.

Table: Flights

FlightNo 🔑	PilotNo 🔑	Destination
ESY8876	65587	Paphos
RYN4133	13584	Dublin
BRI1101	20547	Munich
ESY5655	65587	Edinburgh
BRI8989	20547	Athens

Table: Pilots

PilotNo 🔑	PilotName
65587	Adam Triston
13584	Charlotte Green
20547	Orville Wright

The primary key in `Pilots` is `PilotNo` and is `FlightNo` in `Flights`. The tables are linked by the shared attribute `PilotNo`. This makes `PilotNo` a foreign key in `Flights`.

3.7.2 Structured Query Language (SQL)

SQL is a language used to search for, manage, and manipulate data in a relational database.

There are four main SQL commands: SELECT, UPDATE, INSERT and DELETE.

The SELECT command

SELECT is used for retrieving data from a database table. Commands take the following form:

```
SELECT <attribute> FROM <table> WHERE <condition> AND <condition>
                    ORDER BY <ASC/DESC>
```

To select several attributes, separate their names with commas. Note that you can use the AND keyword as many times as needed (if at all), to specify additional conditions that must be met. The ORDER BY clause is optional. Let's use the following table as an example.

Table: Flights

FlightNo 🔑	PilotNo	Destination
ESY8876	13584	Glasgow
ESY1225	13584	Swansea
BRI1101	20547	Berlin

```
SELECT FlightNo FROM Flights WHERE Destination = 'Berlin'
>> BRI1101

SELECT FlightNo FROM Flights WHERE PilotNo = '13584' AND Destination
= 'Swansea'
>> ESY1225

SELECT FlightNo, Destination FROM Flights WHERE PilotNo = '13584'
AND Destination = 'Glasgow'
>> ESY8876, Glasgow

SELECT Destination FROM Flights WHERE PilotNo = '13584' ORDER BY
FlightNo DESC
>> Glasgow
   Swansea
```

The UPDATE command

This command is used in databases for modifying the attributes of an existing record and takes the form:

```
UPDATE <table> SET <attribute1> = <value1>
WHERE <attribute2> = <value2>
```

Table: Students			
StudentNo 🔑	**Name**	**Email**	**Year**
55685	Aaron Aaronson	a.a.aaronson@outlook.com	1
55887	Beth Hunter	elisabeth.h@gmail.com	2
55622	Sam Cooper	samc00per@hotmail.com	1

```
UPDATE Students SET Email = 'beth24@yahoo.co.uk' WHERE StudentNo =
55887
UPDATE Students SET Name = 'Samuel Cooper' WHERE StudentNo = 55622
```

Once the two UPDATE commands above have been carried out on the table above, the table looks like this:

Table: Students			
StudentNo 🔑	**Name**	**Email**	**Year**
55685	Aaron Aaronson	a.a.aaronson@outlook.com	1
55887	Beth Hunter	beth24@yahoo.co.uk	2
55622	Samuel Cooper	samc00per@hotmail.com	1

UPDATE commands usually use the table's primary key to identify which entities to update but can use more general conditions which would update all of the entities that meet the condition.

```
UPDATE Students SET Year = 2 WHERE StudentNO < 55700
```

Table: Students			
StudentNo 🔑	**Name**	**Email**	**Year**
55685	Aaron Aaronson	a.a.aaronson@outlook.com	2
55887	Beth Hunter	beth24@yahoo.co.uk	2
55622	Samuel Cooper	samc00per@hotmail.com	2

SQL keywords are not case sensitive, but data values might be.

The DELETE command

As you might expect, the DELETE command is used for removing records from a database. The commands take the following form:

```
DELETE FROM <table> WHERE <condition>
```

Table: Cars				
Model 🔑	**Manufacturer** 🔑	**Price**	**Year**	**Sold**
`Polo`	Volkswagen	4995	2010	TRUE
`i10`	Hyundai	5225	2013	FALSE
`Fiesta`	Ford	3995	2009	TRUE

```
DELETE FROM Cars WHERE Sold = TRUE
```

Table: Cars				
Model 🔑	**Manufacturer** 🔑	**Price**	**Year**	**Sold**
`i10`	Hyundai	5225	2013	FALSE

The INSERT command

When using SQL to add new records to an existing table, the INSERT command is used. The command usually takes the form

```
INSERT INTO <table> (<attribute1>, <attribute2>, …) VALUES
(<value1>, <value2>, …)
```

but can be simplified to

```
INSERT INTO <table> VALUES (<value1>, <value2>, …)
```

when all of the attributes in the table are being used in the correct order.

For example, executing the following commands would add two new records to the `Cars` table.

```
INSERT INTO Cars VALUES (“KA”, “Ford”, 3999, 2010, FALSE)
INSERT INTO Cars (Model, Year, Manufacturer) VALUES (“E-Type”, 1970,
“Jaguar”)
```

The first command inserts values into all attributes in the correct order. The second command inserts only some values in a different order, so must list columns.

Wildcards

Wildcards can be used in SQL commands to specify any possible value. For example, rather than selecting a specific attribute in a SELECT command, a wildcard could be used to return all attributes.

In SQL, wildcards are usually notated with an asterisk. For example, using the original `Cars` table from before the delete command:

```
SELECT * FROM Cars WHERE Price > 4000
>> [Polo, Volkswagen, 4995, 2010, TRUE], [Hyundai, 5225, 2013,
FALSE]
```

The DELETE command is a bit of a special case when it comes to wildcards. The commands `DELETE FROM Cars` and `DELETE * FROM Cars` would do the same job of deleting all entries in the Cars table.

The SELECT command for related tables

You may be required to extract data from up to two related tables. This section will demonstrate how to query multiple tables using SQL.

The command usually takes the form:

```
SELECT <table1.attribute> FROM <table1>, <table2>
              WHERE <join-criteria>
```

The `<join-criteria>` statement is used to link the two tables as part of the query. It is typically created by setting the foreign key of one table equal to the primary key of the other. As before, the `WHERE` statement can also contain other conditions using the AND keyword.

In SQL an attribute specific to a table is denoted as `<table_name.attribute_name>`.

The `Students` table and a related `Teachers` table will be used to show examples of selecting from multiple tables. The `Students` table includes `TeacherID` as a foreign key, linking to the `TeacherID` primary key in the `Teachers` table.

Table: Students				
StudentNo 🔑	**Name**	**Email**	**Year**	**TeacherID**
55685	Aaron Aaronson	a.a.aaronson@outlook.com	1	01
55887	Beth Hunter	elisabeth.h@gmail.com	2	02
55622	Sam Cooper	samc00per@hotmail.com	1	01

Table: Teachers		
TeacherID 🔑	**Name**	**Class**
01	Mr. Davies	13D
02	Ms. Smith	12C

Example 1: Selecting a Field from One Table

The following example shows how a single field is selected from one table:

```
SELECT Students.StudentNo FROM Students, Teachers
WHERE Students.TeacherID = Teachers.TeacherID
>> 55685, 55887, 55622
```

However, it is actually equivalent to:
`SELECT StudentNo FROM Students`

We only need to include this line:
`WHERE Students.TeacherID = Teachers.TeacherID`
If we are also filtering using data from `Teachers`.

Example 2: Selecting multiple Fields from Both Tables
You can retrieve data from both tables with a single query. Suppose you want to retrieve each student's name and the name of their teacher:

```
SELECT Students.Name, Teachers.Name FROM Students, Teachers
WHERE Students.TeacherID = Teachers.TeacherID
>> Aaron Aaronson, Mr.Davies
   Beth Hunter, Ms.Smith
   Sam Cooper, Mr. Davies
```

Example 3: Filtering With a Condition from Another Table
In exams, you may have to filter the results extracted using a condition from another table. Suppose you want to retrieve the emails of students in class 13D:

```
SELECT Students.Email FROM Students, Teachers WHERE Teachers.Class =
'13D' AND Students.TeacherID = Teachers.TeacherID
>> a.a.aaronson@outlook.com, samc00per@hotmail.com
```

Example 4: Ordering Results
As with a single-table (flat-file) database, the results of a `SELECT` query can be ordered. However, they can be ordered by an attribute from either table.

The following SQL command returns the `StudentNo` and `Name` of any `Students` that have `Mr. Davies` as their teacher and orders it into ascending alphabetical order of the students' `Name`.

```
SELECT Students.Name, Students.StudentNo FROM Students, Teachers
WHERE Teachers.Name = 'Mr. Davies' AND Teachers.TeacherID =
Students.TeacherID ORDER BY Students.Name ASC
>> Aaron Aaronson, 55685
   Sam Cooper, 55622
```