DATABASE DEsign and development

National 5 Computing Science

2020

****

Contents

[Analysis 3](#_Toc534796540)

[Design & GDPR 4](#_Toc534796541)

[Entities 5](#_Toc534796542)

[Attributes 5](#_Toc534796543)

[Relationships 5](#_Toc534796544)

[Entity relationship diagrams 6](#_Toc534796545)

[Data dictionary 7](#_Toc534796546)

[Designing a solution to a query 11](#_Toc534796547)

[Implementation 14](#_Toc534796548)

[Referential integrity 14](#_Toc534796549)

[**SQL**  15](#_Toc534796550)

[**Testing** 17](#_Toc534796551)

[**Evaluation** 17](#_Toc534796552)

Alison Cowie

Portobello High School, Edinburgh

## Analysis

The development of a database has several phases. These include:

**1. Analysis**

**2. Design**

**3. Implementation**

**4. Testing**

**5. Evaluation**

For example, Mr McLeod is Depute at Craigfar High School. He wants to make a database of school activities (clubs) so that, at any point in time, he can print out information (a report) about any activity and which teacher runs it

Analysis: Mr McLeod meets with Ms Lyall who knows how to make databases. They create a document called the **database specification** that lists the **end-user requirements** (what it needs to be like for Mr McLeod to use it – he’s not an **expert** user) and the **functional requirements** of the database (what it needs to do for Mr McLeod)

Mr McLeod decides he would like a button that will let him search for an activity. The search will result in a report (display) of the activity details. He would also like a button that lets him search for a teacher. The search will result in a report (display) of all the activities that the teacher is running

Design: Ms Lyall will consider the **implications of the *General Data Protection Regulation*** for the database she is about to make

She will draw the **relationship between the entities** that will appear in the database. At the moment the two entities are ACTIVITY and TEACHER. Then she will draw up a **data dictionary** for the database and **plan the scripts** (programs) for all the operations that Mr McLeod requires. Some of the scripts will be automated within the database software she is going to use so she won’t need to write all the code

Implementation: Once the design (the plan) for the database is complete, Ms Lyall will create the tables for each entity in the database, create the attributes for each table and include their type, size and any appropriate validation. Then she will populate the database with actual data (the relevant details of the activities and teachers). This might mean typing in the data from scratch or being able to import data from other files e.g. teacher details. Then she will include program code to carry out the operations that Mr McLeod has asked for – the **end-user** and **functional requirements of the database**

Testing: Ms Lyall **will be testing the database continually** while she is working. When she is finished, Mr McLeod will be involved in testing it to make sure that it works as he specified. This may involve going back to previous phases and making changes (iteration). Once it is complete, Ms Lyall will arrange for the database to installed in such a way that Mr McLeod can use it

Evaluation: This is to decide whether the database is **fit for purpose**. That means: Does it meet its **end-user** and **functional requirements**? Is it **robust?** Is it **reliable** in that the **output is accurate**?

## Design & GDPR

General Data Protection Regulation 2018

Companies and organisations like the government hold a lot of personal data about individuals. To protect this data and people’s rights when storing, transmitting and exporting it, the *General Data Protection Regulation* was passed by Parliament to protect peoples’ privacy.

In short the 7 principles of GDPR:

1. Lawfulness, fairness and transparency
2. Purpose limitation
3. Data minimisation
4. Accuracy
5. Storage limitation
6. Integrity and confidentiality (security)
7. Accountability

The above are explained in more detail at the following web address (URL):-

<https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/principles/>

**Exceptions to the Act**

If any data is held by the ***government***, ***police***, ***security forces*** or the ***Inland Revenue***, then you are denied access to the data.

**Data Users** are the employees that work for organisations/companies that have access to / make use of the information held about individuals.

**Data Controllers** are the companies, organisations or people who store personal information about individuals. Data Controllers must be registered with the **Data Protection Commissioner**. Data Controllers must apply for permission to store data and state what data they want to store and for what purpose.

**Information Commissioner** at the ***Information Commissioner’s Office (ICO)*** is the person you contact if you wish to raise a complaint about the accuracy of data. They regulate the Freedom of Information Act. As well as operating an advice service to address general enquiries, it promotes good practice in data protection by raising awareness of organisational responsibilities across all sectors.

### Entities

An **entity** is a single person, place, or thing about which data can be stored e.g. a car, Spain, flight 4A07

### Attributes

An entity will have attributes. In the example below, the car entity has four attributes: make, model, colour, registration number

make: vauxhall

model: zafira

colour: grey

registration number: S012 GHJ

The attributes have values that can be edited.

Each collection of attributes describing an entity is called a record. If we made a database of 1200 cars, we would have 1200 records, each with four attributes

### Relationships

We are surrounded by entities that have relationships with each other. The relationships can be described as

* ONE to ONE
* ONE to MANY
* MANY to MANY

For National 5 we only look at ONE to MANY relationships

For example, ONE car has MANY repairs. You might think – but many cars have many repairs – but that is not what the relationship is saying. It means that one specific repair (for example, the brake pads on Vauxhall Zafira SO12 GHJ replaced on 5.3.17) was for ONE car but that ONE car might have MANY different (unique) repairs

CAR

REPAIR

For example, the relationship between mother and children is ONE to MANY. You might think – but many mothers have many children – but that is not what the relationship is saying. It means that one specific child (for example, Zoltan Albard born on 12.8.2001) is the child of ONE mother but that ONE mother might have MANY different (unique) children

CHILD

MOTHER

### Entity relationship diagrams

A flat file database is shown below – it has 9 attributes. It would be better if the data is split into two linked tables with a relationship created between them. At the moment, if Miss Chess changed her name to Mrs Board, every entry for her would have to be amended which could easily lead to errors (loss of data integrity)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Activity ID*** | ***Title*** | ***Age Range*** | ***Venue*** | ***Teacher ID*** | ***Name*** | ***Teaching Room*** | ***Subject*** | ***Full Time*** |
| *ASC2745* | *Revision* | *11-18* | *Room 101* | *T87* | *Miss Chess* | *214* | *Maths* | *No* |
| *Susp1024* | *Water sports* | *14-18* | *Porty beach* | *T56* | *Mr Splash* | *303* | *PE* | *Yes* |
| *Susp1013* | *Adventure* | *11-15* | *Crazy Ways* | *T14* | *Ms Yeehah* | *504* | *Geography* | *Yes* |
| *Susp1067* | *Make a mural* | *11-18* | *PHS* | *T39* | *Mrs Colourist* | *612* | *Art* | *No* |
| *Susp1018* | *Cycling* | *16-18* | *Glen Tress* | *T44* | *Mr Spokes* | *405* | *CDT* | *Yes* |
| *ASC2543* | *Board games* | *11-14* | *Room 102* | *T87* | *Miss Chess* | *214* | *Maths* | *No* |
| *ASC2716* | *Cycling* | *14-17* | *PHS* | *T44* | *Mr Spokes* | *405* | *CDT* | *Yes* |
| *ASC3487* | *Adventure* | *15-18* | *PHS* | *T56* | *Mr Splash* | *303* | *PE* | *Yes* |

If the data shown in the flat file database was held in two tables as a relational database, the **entity relationship** would be shown as follows:

ONE teacher offers MANY activities

TEACHER

ACTIVITY

The two tables would be:

Foreign key is used to link the two tables together – it is the primary key in the other table

Primary key uniquely identifies each record in this table

Primary key uniquely identifies each record in this table

**Activity ID**

**Title**

**Age Range**

**Venue**

**Teacher ID\***

**Teacher ID**

**Name**

**Teaching Room**

**Subject**

**Full Time**

ACTIVITY table

TEACHER table

### Data dictionary

A data dictionary should include the following:

#### Entity names

For this example, the entity names are ACTIVITY and TEACHER

#### Attribute (field) names

The attribute (field) names for this example are:

*Activity ID*, *Title*, *Age Range*, *Venue*, *Teacher ID*, *Name*, *Teaching Room*, *Subject*, *Full Time*

#### Primary and foreign key

***Primary key field:*** this is the field (attribute) that **uniquely identifies each record** in a database e.g. account number, ID, car registration number, etc.

In this example, the primary keys are Activity ID for the ACTIVITY table and Teacher ID for the TEACHER table

***Foreign key field:*** this is used to link one table to another table in a database i.e. set up relationships between tables.

Note: the foreign key in one of the tables will be the primary key in another table

Helpful hint: “the primary key from the ONE table is used as the foreign key in the MANY table”

#### Attribute (field) types

##### Text

e.g. EH9 1EF, 07652 915845, Jane

(N.b. a telephone number is text – it has a lead zero, may contain a space and would not be used in calculations)

##### Number

e.g. 95, 0.645 Also £32.95 (formatted as currency)

##### Date

e.g. 15/04/2017

##### Time

e.g. 23:30, 06:00

##### Boolean

TRUE/FALSE

#### Attribute size

For example, a text attribute might be maximum 20 characters in length, written as text (20)

#### Validation

The process of checking that data entered into a system is of the correct type and structure

1. *Presence check*

checks that data has been entered/data field not left blank

1. *Restricted choice*

the user presented with a list of options to choose from e.g. from a drop-down menu

1. *Field length check*

specifies the size of a field e.g. number of characters allowed in a field

1. *Range check*

specifies the range of values a field can hold e.g. range 0 to 100 inclusive

Altogether, the data dictionary might look like this:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Table** | **Attribute name** | **PK/FK** | **Type (size)** | **Unique** | **Required** | **Validation** | **Format** | **Sample data** |
| ***ACTIVITY*** | ***Activity ID*** | *PK* | *Text (8)* | *Yes* | *Yes* | *Presence check* |  | *ASC2745* |
| ***Title*** |  | *Text (20)* |  | *Yes* |  |  | *Revision* |
| ***Age Range*** |  | *Text (5)* |  |  |  |  | *11-18* |
| ***Venue*** |  | *Text (15)* |  | *Yes* |  |  | *Room 101* |
| ***Teacher ID*** | *FK* | *Lookup* | *No* | *Yes* | *Lookup from TEACHER* |  | *T87* |
| ***TEACHER*** | ***Teacher ID*** | *PK* | *Text (4)* | *Yes* | *Yes* | *Presence check* |  | *T87* |
| ***Name*** |  | *Text (40)* |  |  | *Restricted choice* |  | *Miss Chess* |
| ***Teaching Room*** |  | *Text (3)* |  | *Yes* | *Field length check* |  | *214* |
| ***Subject*** |  | *Text (20)* |  |  |  |  | *Maths* |
| ***Full Time*** |  | *Boolean* |  |  |  | *Checkbox* | *No* |

Here’s another example:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Table** | **Attribute name** | **PK/FK** | **Type (size)** | **Unique** | | **Required** | **Validation** | **Format** | **Sample data** |
| ***CUSTOMER*** | ***Customer ID*** | *PK* | *Text (8)* | | *Yes* | *Yes* | *Presence check* |  | *C01456AR* |
| ***Title*** |  | *Text (3)* | |  |  | *Restricted choice* |  | *Mrs* |
| ***Name*** |  | *Text (30)* | |  | *Yes* |  |  | *Amy Bryce* |
| ***Address*** |  | *Text (150)* | |  | *Yes* |  |  | *16 Main St, Craigfar, CK9 1RF* |
| ***Age*** |  | *Number* | |  |  | *Range check*  *(16 to 100)* | *000* | *37* |
| ***ORDER*** | ***Order ID*** | *PK* | *Text (4)* | | *Yes* | *Yes* | *Presence check* |  | *T87* |
| ***Order date*** |  | *Date* | |  |  | *Restricted choice* | *dd/mm/yy* | *12/05/17* |
| ***Total to pay*** |  | *Number* | |  | *Yes* | *Field length check* | *£0000.00* | *£36.95* |
| ***Customer ID*** | *FK* | *Lookup* | | *No* | *Yes* | *Lookup from CUSTOMER* |  | *C01456AR* |

Note. The foreign key is not unique in the MANY table For example, there may be many orders with the same Customer ID on them

### Designing a solution to a query

#### Multiple tables

Suppose the flat file database shown here …

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Activity ID*** | ***Title*** | ***Age Range*** | ***Venue*** | ***Teacher ID*** | ***Name*** | ***Teaching Room*** | ***Subject*** | ***Full Time*** |
| *ASC2745* | *Revision* | *11-18* | *Room 101* | *T87* | *Miss Chess* | *214* | *Maths* | *No* |
| *Susp1024* | *Water sports* | *14-18* | *Porty beach* | *T56* | *Mr Splash* | *303* | *PE* | *Yes* |
| *Susp1013* | *Adventure* | *11-15* | *Crazy Ways* | *T14* | *Ms Yeehah* | *504* | *Geography* | *Yes* |
| *Susp1067* | *Make a mural* | *11-18* | *PHS* | *T39* | *Mrs Colourist* | *612* | *Art* | *No* |
| *Susp1018* | *Cycling* | *16-18* | *Glen Tress* | *T44* | *Mr Spokes* | *405* | *CDT* | *Yes* |
| *ASC2543* | *Board games* | *11-14* | *Room 102* | *T87* | *Miss Chess* | *214* | *Maths* | *No* |
| *ASC2716* | *Cycling* | *14-17* | *PHS* | *T44* | *Mr Spokes* | *405* | *CDT* | *Yes* |
| *ASC3487* | *Adventure* | *15-18* | *PHS* | *T56* | *Mr Splash* | *303* | *PE* | *Yes* |

… has been implemented as a **relational database** with **two tables** of records as shown here…

Activity ID: ASC2745

Title: Revision

Age Range: 11-18

Venue: Room 101

Teacher ID: T87

Activity ID: Susp1024

Title: Water sports

Age Range: 14-18

Venue: Porty beach

Teacher ID: T56

Activity ID: Susp1013

Title: Adventure

Age Range: 11-15

Venue: Crazy Ways

Teacher ID: T14

Activity ID: Susp1067

Title: Make a mural

Age Range: 11-18

Venue: PHS

Teacher ID: T39

Activity ID: Susp1018

Title: Cycling

Age Range: 16-18

Venue: Glen Tress

Teacher ID: T44

Activity ID: ASC2543

Title: Board games

Age Range: 11-14

Venue: Room 102

Teacher ID: T87

Activity ID: ASC2716

Title: Cycling

Age Range: 14-17

Venue: PHS

Teacher ID: T44

Activity ID: ASC3487

Title: Adventure

Age Range: 15-18

Venue: PHS

Teacher ID: T56

Teacher ID: T18

Name: Mr Donne

Teaching Room: 100

Subject: English

Full Time: Yes

Teacher ID: T87

Name: Miss Chess

Teaching Room: 214

Subject: Maths

Full Time: No

Teacher ID: T62

Name: Mr Kildare

Teaching Room: 101

Subject: History

Full Time: Yes

Teacher ID: T56

Name: Mr Splash

Teaching Room: 303

Subject: PE

Full Time: Yes

Teacher ID: T14

Name: Ms Yeehah

Teaching Room: 504

Subject: Geography

Full Time: Yes

Teacher ID: T39

Name: Mrs Colourist

Teaching Room: 612

Subject: Art

Full Time: No

Teacher ID: T75

Name: Miss Overby

Teaching Room: 518

Subject: HE

Full Time: No

Teacher ID: T44

Name: Mr Spokes

Teaching Room: 405

Subject: CDT

Full Time: Yes

… and Mr McLeod wanted to create a report that showed the names of all the **full time** teachers who offered **adventure** activities. Which teachers would the report show?

The report would show *Ms Yeehah and Mr Splash*

There is a programming language called **SQL** that is used to query, insert, update and modify data in a database. SQL stands for **Structured Query Language**.

If we were to use this language to find these teachers, then our SQL code would look like this

**SELECT** Teacher.Name

**FROM** Teacher

**INNER JOIN** Activity //this makes the link between the two tables (Teacher and Activity)

**ON** Teacher.Teacher ID = Activity.Teacher ID

**WHERE** Teacher.Full Time = Yes;

#### **Fields**

Sometimes the word “field” is used instead of “attribute”

#### **Search criteria**

***Simple search*** (query) (find) - searching on **one** field (attribute) only

e.g. search for all records with field (forename = “James”)

**SELECT** \* **FROM** Teacher

**WHERE** Teacher.Forename = “James”;

***Complex search*** (query) - searching on **two or more** fields (attributes)

e.g. search for all records with (surname = “McRury”) AND (purchased = true)

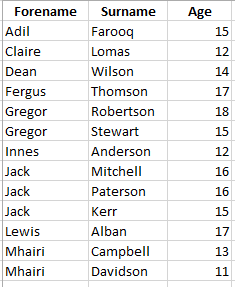
**SELECT** \* **FROM** Teacher

**WHERE** Teacher.Surname = “McRury” **AND** Teacher.Purchased = True;

#### Sort order

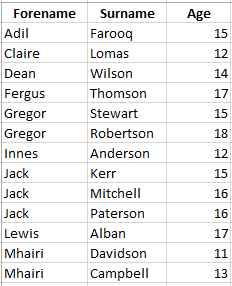
Sort in alphabetical, numerical, chronological order (ascending/descending)

Sorting on more than ***one*** field: sorts on first field, then sorts within the groups of the second field, and then within the groups of the third and so on



This table has been sorted on one field (attribute)

It has been sorted on Forename in alphabetical ascending order (A to Z)

This table has been sorted on ***two*** fields (attributes) – first on Forename in alphabetical ascending order and then on Age in numerical ascending order

How to tell: if you look at the Gregors, Jacks and Mhairis, their surnames are sometimes ascending, sometimes descending but their ages (in their little groups) are all increasing

## Implementation

### Referential integrity

Advantages of a relational database over a flat-file database:

In the flat file database shown previously, if Miss Chess changed her name to Mrs Chequer, then someone would have to change every entry for Miss Chess. They are likely to make mistakes and, especially in a large database, this would quickly lead to the data becoming **inaccurate** and **unreliable**.

In the relational database, there would be just one record for Miss Chess. There would be two activity records linked to her record and, if her name needed to be amended (e.g. she became Mrs Chequer), then only the record about her would need to be amended and all the activities linked to her would automatically see her new name.

So the advantage of a **relational database with linked tables** is that it **avoids modification errors for data that is duplicated** that would occur if the data was held in a flat database. Links can be made from data in one table to data that already exists in another table to provide different pathways through the data. The implementation should ensure that the tables are linked together in the most efficient and useful way so that any necessary reports can be generated.

### 

### **SQL**

There is a programming language called **SQL** that is used to query, insert, update and modify data in a database. It stands for **Structured Query Language**. The following examples have been copied from W3Schools.com

#### **SELECT**

**FROM**

For example: SELECT CustomerName, City

FROM Customers;

CustomerName, City are **attributes** (fields) in the **table** called Customers

**WHERE**

For example: SELECT CustomerName

FROM Customers

WHERE Country = 'Mexico ';

CustomerName, City, Country are **attributes** (fields) in the **table** called Customers

**AND, OR, <, >, =**

For example:

SELECT \* // \* means all the records

FROM Customers  
WHERE Country='Germany' AND City='Berlin';

SELECT \*

FROM Customers  
WHERE City='Berlin' OR City='München';

SELECT Name

FROM Students

WHERE Age >= 14 AND Age <= 18;

**ORDER BY** (max of two fields)

For example:

SELECT \*

FROM Customers  
ORDER BY Country DESC;

SELECT \*

FROM Customers  
ORDER BY Country ASC, CustomerName DESC;

#### **INSERT**

For example:

INSERT INTO Customers (CustomerName, City, Country)  
VALUES ('Cardinal', 'Stavanger', 'Norway');

#### **UPDATE**

For example:

UPDATE Customers  
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'  
WHERE CustomerID = 1;

#### **DELETE**

For example:

DELETE FROM Customers  
WHERE CustomerName='Alfreds Futterkiste';

#### **INNER JOIN** (Equi-join between tables)

For example:

SELECT Orders.OrderID, Customers.CustomerName  
FROM Orders  
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

## **Testing**

This would involve testing the SQL operations to ensure that they work correctly and produce the correct output

A test table could be drawn up showing the test data, the expected output and the actual output in the form of screenshots. This test table could be referred to in support of your evaluation

## **Evaluation**

Finally, the complete database is evaluated in terms of whether it is **fit for purpose**. This means making a judgement as to whether it meets its **specification** by delivering its **functional requirements** (see Analysis). You would also judge whether it meets the **end-user requirements**. You would consider your test results and draw conclusions from them.

For example, you might say “The functional requirements Mr McLeod asked for was a button that would let him search for an activity. The search was to result in a report (display) of the activity details. He also wanted a button that let him search for a teacher. The search was to result in a report (display) of all the activities that the teacher is running. My database allowed Mr Mcleod to carry out these operations and the **output was accurate** (see test results) so my database is **fit for purpose** in that sense. However, my user interface was unnecessarily complicated for Mr McLeod (a novice user) to use so I would say that it did not fully meet the end-user requirements”

### 

### 