AQA Computer Science A-Level

4.11.1 Big Data

Intermediate Notes

Specification:

4.11.1 Big Data:

Know that 'Big Data' is a catch-all term for data that won't fit the usual containers. Big Data can be described in terms of:

- Volume – too big to fit into a single server
- Velocity – streaming data, milliseconds to seconds to respond
- Variety – data in many forms such as structured, unstructured, text, multimedia

Know that when data sizes are so big as not to fit on to a single server:

- The processing must be distributed across more than one machine
- Functional programming is a solution, because it makes it easier to write correct and efficient distributed code

Know what features of functional programming make it easier to write:

- Correct code
- Code that can be distributed to run across more than one server

Be familiar with the:

- Fact-based model for representing data
- Graph schema for capturing the structure of the dataset
- Nodes, edges and properties in graph schema

Big Data

The name “big data” is a catch-all term for data that doesn’t fit the usual containers. The three defining features of big data can be remembered as “the three Vs”:

Volume	Velocity	Variety
There is too much data for it all to fit on a conventional hard drive or even a server. Data has to be stored over multiple servers, each of which is composed of many hard drives.	Data on the servers is created and modified rapidly. The servers must respond to frequently changing data within a matter of milliseconds.	The data held on the servers consists of many different types of data from binary files to multimedia files like photos and videos.

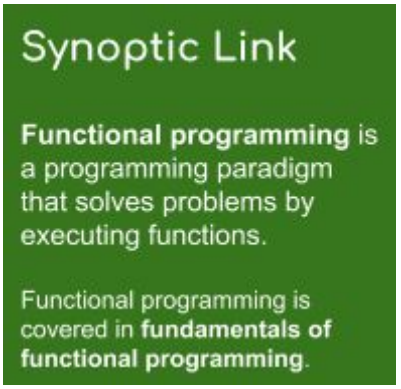
While you may think that the volume of big data is its most challenging attribute, it turns out that big data’s lack of structure causes the most trouble, making it difficult to analyse the data.

In order to extract useful information from big data, machine learning techniques must be used to discern patterns in the data.

When data is stored over multiple servers, as is the case with big data, the processing associated with using the data must also be split across multiple machines. This would be incredibly difficult with conventional programming paradigms as the machines would all have to be synchronised to ensure that no data is overwritten or otherwise damaged.

Functional programming

Functional programming is a solution to the problem of processing data over multiple machines. It is easier to write correct, efficient, distributed code with functional programming than with procedural programming techniques.



The fact-based model for representing data

Because big data doesn't conform to the row and column format typically used to represent data, it must be represented in a different manner. One way of representing big data is with the fact-based model.

In the fact-based model, each individual piece of information is stored as a fact. Facts never change once created and can't be overwritten.

Stored with each fact is the date and time at which a piece of information was recorded. Seeing as facts are never deleted or overwritten, multiple different values could be held for the same attribute. This is where timestamps come in, allowing a computer to discern which value is the most recent.

For example, the two facts above contain the number and colour of a house at two different points in time. The house was green in 2010 but re-painted white in 2016. If this information were stored in a dataset and the colour of house number 42 was queried, the two timestamps would be compared and the most recent information (`Colour: White`) would be returned.

As facts are not overwritable, using the fact-based model for storing big data reduces the risk of accidentally losing data due to human error.

Representing big data using graph schema

Graph schema uses graphs consisting of circles and arrows to graphically represent the structure of a dataset. Circles in a graph represent entities and can contain information about an entity (called properties).

Arrows are used to represent relationships between entities and are labelled with a brief description of the relationship.

The example below shows three entities represented in a graph schema as three circles. The properties of each entity (`breed` and `name`) are listed inside of the circles. Arrows linking the circles represent the relationships between the circles.

Timestamps are rarely included in graph schema diagrams, instead you should assume that each circle contains the most recent information available.