

Introduction to database systems

In this week, we will cover the following topics:

- *Data: what (and examples of where) it is.*
- *How data might be stored on a computer.*
- *Examples of where data might be useful.*
- *Spreadsheet data storage.*
- *Distinction between data and derived information.*
- *What databases are.*
- *What relational database systems are.*
- *How relational approach sits with other approaches to databases.*

... and will result in the following learning outcomes:

- *An appreciation of what data is and how structured data can be stored.*
 - *Knowledge that data is very important.*
 - *An appreciation of how databases are different to more traditional data storage approaches.*
 - *An understanding that relational databases, the focus of this course, are not the only kind of database that exist.*
-

Table of Contents

1.1	Data everywhere	3
1.2	File system data organization	3
1.2.1	Freight delivery scenario	4
1.3	A legacy data management system	5
1.3.1	Obtaining 'information' from stored data	6
1.4	Databases	7
1.4.1	Database definitions	7
1.4.2	Purpose and motivation for using databases	8
1.4.3	Databases are 'part of' a database management system	8
1.4.4	Database types: a heads-up.....	9
1.4.5	Database data modeling.....	11
1.5	Summary	11
Figure 1:	Example index card	4
Figure 2:	Typical spreadsheet application	5
Figure 3:	Example issues concerning data.....	6
Figure 4:	Stackoverflow "Databases" are not DBMS.....	9
Figure 5:	Databases and DBMS examples	9
Table 1:	Basic 'find truck TypeC' algorithm	5
Table 2:	Different definitions of 'Database'	7

1.1 Data everywhere

Data is absolutely everywhere. Data is contained in familiar everyday objects, for example: newspapers or books; in photographs and digital images, and therefore in videos as a moving collection of 'frames'; within the pages of an accountant's spreadsheet; in a recorded voice, which is increasingly well-recognized by computer algorithms such that people now speak to their devices. Data is everywhere.

A useful starting point, then, if data is everywhere, is to decide what *kind* of data you **will** be focusing on in this course... and which kind of data you **won't** be focusing on. Note, this does not mean that the kind of data you won't be focusing on is any less important. By the time you complete your degree you will have a much better idea yourself about nuances of different kinds of data.

The kind of data you won't be thinking about is **unstructured data**, so let us explain what that is, then this will help us explain **structured data**, which is *the* essential reason for using databases.

- *Unstructured data*: this is data that has **no pre-defined design** structure. A typical example is a piece of written text. When people write they follow a set of rules, such as 'write from left to right' or 'write from right to left', depending on the language they are writing in, or choosing words from a relevant dictionary. However, the rules can be used, creatively. The result is that the style of written text takes on a large number of different *forms*. Of course, for those who like to read books, this is a good thing as it is nice to have choice!
- *Structured data*: at other times it is convenient to have data that is **highly organised**. For example, when we go to choose our book, from a shop or a library, it might be less pleasurable if the ordering of the books themselves are highly unstructured, say randomly, especially if, like in a university library, the book collection is very large. This is why in your university library a system of ordering is used – i.e., a method, or set of rules, by which the books and subjects are ordered, such that they are easier to find. There are a number of such methods. In the UK, a system known as Dewy Decimal Classification is often used, although there are also many other classification systems.

As you will learn on this course, although there is no strictly right or wrong way to organize data, the choices made concerning the design of the organization scheme, technically referred to as a *schema*, determine how easily the data can be exploited.

1.2 File system data organization

Before we start to look in more depth at computational database systems, we should continue to think about the organization of data as *separate* to the physical system being used.

File systems, these days, are strongly associated with computers. For example, operating systems have well defined ways of organizing and recording **files**, which according to the Oxford Dictionary of Computing are "*Information held on a backing store in order (a) to enable*

it to persist beyond the time of execution... to (b) overcome limitations in main memory". In other words, files are organised units of computer memory.

File systems, however, can also be **manual**, and computer file systems are based on similar kinds of principles as manual files systems. In fact, we usually image a computer to do lots of *clever* things, like multiplying very large numbers or finding their nth root, all at lightning speed. Much of the time, however, the computer is mimicking what we imagine a filing clerk would be doing in the office of a large company – moving backwards and forwards between lots of filing cabinets, removing files, taking notes from (or recording information on) the files, before rushing back to the filing cabinet and placing the file in exactly the same position, according to the system of file organization being used.

1.2.1 Freight delivery scenario

Let us therefore consider an illustrative, manual file system scenario, based on the example of a freight company. The freight company delivers a wide range of products from various different warehouses to various different retail outlets. The freight company has a number of resources, such as a fleet of trucks and employees who drive the trucks etc. Lots of data is stored in a large filing system, which is based on simple index cards [who invented index cards]. A simple such card is shown below. It contains information about a given driver, which is useful for the clerk staff when they need to allocate a driver to a certain job.

Driver Name	John Brown
Truck Types	TypeB, TypeC, TypeA
Salary	£15 per hour
Location	Sheffield
.	.

Figure 1: Example index card

For example, a job is required to deliver 10 tons of freight, which is currently situated in Chesterfield (30 minutes' drive from Sheffield), requiring vehicle TypeC, to be delivered to Inverness, which will take about 7.5 hours according to the mapping service used by the company. So far, John Brown looks like a good candidate. The clerk then might look at the number of hours he has worked this week (the company has a policy that drives cannot work over 60 hours). It turns out that John Brown has only worked 30, and so the job is then allocated to him. Consider the simple example of finding a TypeC vehicle, but the potential amount of search required:

1. Start at random position in the filing system
2. N = 0
3. Take out card
4. Read each datum
 - a. Is the type 'Truck Types'
 - i. Does TypeC exist? (yes/no)
5. Is the type Location
 - a. Is the distance to freight pick-up < 50 km (yes/no)
6. If number of yes' = 2 END
Else
Take out next card GOTO step '1'

7. Put card back

Table 1: Basic 'find truck TypeC' algorithm

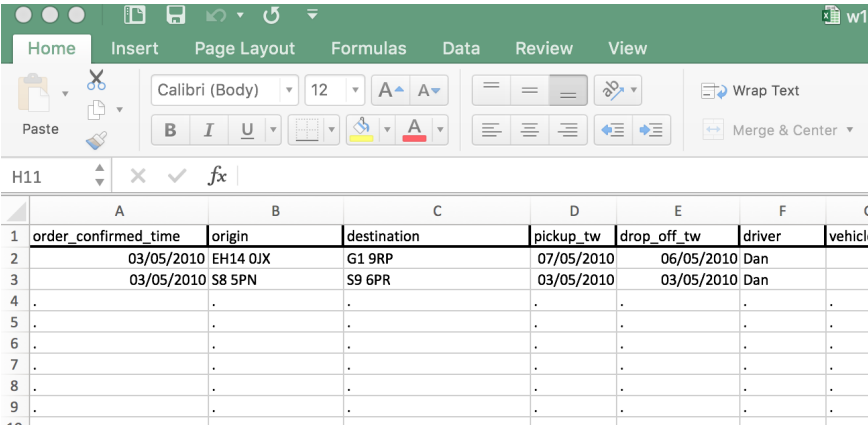
The design of the filing system will impact on how quickly and efficiently the clerk can process information and allocate jobs. In this case, for example, assuming that each job will require a certain vehicle type, then organizing the file system around vehicle types might be more efficient. In the above case, in other words, searching inside a 'TypeC' filing cabinet, or folder, would improve efficiency.

1.3 A legacy data management system

Returning to our freight company example, imagine a time when the company first began trading as a small business. The fleet size was just 2 trucks. Due to the small scale of the business, only one admin person was required to take orders, input them into a spreadsheet, and allocate the driver and vehicle. The required information to input is the time the job is confirmed to the client, the origin of the freight, the destination of the freight, when the freight needs to be collected from the origin (pick-up time window), when the freight needs to be dropped off at the destination (drop-off time window), the driver allocated to the job and the vehicle he will be using.

order_confirmed_time	origin	destination	pickup_tw	drop_off_tw	driver	vehicle
----------------------	--------	-------------	-----------	-------------	--------	---------

Typically, such information is accumulated into a spreadsheet application:



	A	B	C	D	E	F	G
1	order_confirmed_time	origin	destination	pickup_tw	drop_off_tw	driver	vehicle
2	03/05/2010	EH14 OJX	G1 9RP	07/05/2010	06/05/2010	Dan	
3	03/05/2010	S8 5PN	S9 6PR	03/05/2010	03/05/2010	Dan	
4
5
6
7
8
9
10

Figure 2: Typical spreadsheet application

Note the error in cell E2 (or it could be D2) where the drop_off_tw entry is mistakenly put before the pick-up entry, a physical impossibility. This is a classic disadvantage of this kind of data management: there is no way of checking the accuracy and validity of the data manual input of data.

Furthermore, as the business grows, so too does the amount of data. Imagine a situation, several years later, where the spreadsheet has grown to reflect a business that has increased in size (50 vehicles, 40 drivers, 10 admin staff and a business analyst). The sheer size of the spreadsheet has grown to and now contains thousands of rows. The business analyst wants

to obtain a profile of the data for specific companies. Although it can be possible to use the search facilities within a spreadsheet application, doing so becomes tedious and inefficient.

Even more problematic is the ease with which data is updated. The company has reached a size whereby orders come in thick and fast. While one order is being placed another one comes in. The recruitment of the additional admin staff provides enough capacity to deal with the orders, but the data management system, based on a spreadsheet, simply does not support this volume of orders:

- The use of a shared spreadsheet easily results in overwritten orders.
- Allocating spreadsheets to separate admin staff results in a fragmented system and the task of maintaining the data becomes time consuming and therefore costly.

These problems are not just “awkward”. In a very real sense for the freight company the data, held in the spreadsheet, *is* the business; it would not function without it – the data is *critical*.

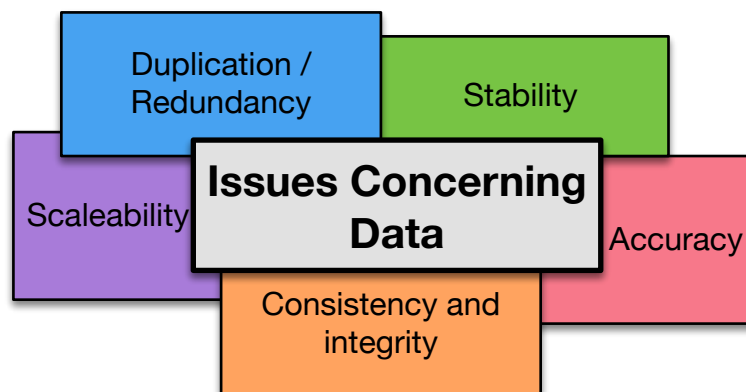


Figure 3: Example issues concerning data

1.3.1 Obtaining ‘information’ from stored data

It is useful to distinguish between the terms data and information.

Cross-referencing: Typically, with any kind of data, we are not *interested* in all of the data all of the time, which is why we store data for later use. Furthermore, the nuances of data use and retrieval cannot easily be mapped directly to a file system design, at the design stage. As an example, imagine that a new vehicle has recently been designed. The freight company wants to buy a number of such vehicles. They will become represented on index cards as TypeF, but these changes do not get made to the file system before the vehicles are bought (there is no TypeF cabinet). By law, in order to Drive a TypeF vehicle, drivers must hold licenses for vehicle TypeA and vehicle TypeB. In order to know if a Driver is licensed to drive TypeF, the clerk then has to check all of the drivers in the TypeA cabinet, and all of the drivers in the TypeB cabinet, to find out who is a ‘TypeF driver’ i.e:

*Which drivers are available who can drive vehicle TypeA
...of the drivers who can also drive vehicle TypeB?*

This is referencing TypeA drivers from TypeB drivers, or is otherwise known as *cross-referencing*. A specific problem with the need to cross-reference data that is held within a file based system, is that cross-referencing encourages the creation of intermediate files and the need to process the existing data files synchronously (i.e. assuming that by the time file 2 is read, that the information already copied from file 1 is not out of date). The essential problem with files is that they are stored separately and isolated from each other and processing them in the way required is difficult, especially when cross-referencing several different files.

Aggregating information: It is important to make the distinction between **data** and its derived **information**. What we mean, in the current context, is that information is based on the processing of a set of datum (singular for the plural 'data'). Continuing with our freight company example, we might need an answer to a simple question:

What is the total amount of driver salary?

The salary information is, of course, indexed on the card of each employee/driver. Therefore, in the file based system assumed here, we need to read each and every card, extract the data, perhaps by reading it out to a file, and then sum the data in this created file.

Fragmentation: each separate file is an isolated place in computer memory. Cross-referencing data in order to access subsets of data is difficult and the creation of temporary files is encouraged. Processing files in a file system is, therefore, possible but it is awkward and difficult and prone to error.

1.4 Databases

1.4.1 Database definitions

Let's consider some definitions:

"a structured set of data held in a computer, especially one that is accessible in various ways."

(Oxford Dictionary of English)

"A database is an organized collection of data. It is the collection of schemas, tables, queries, reports, views, and other objects. The data are typically organized to model aspects of reality in a way that supports processes requiring information..."

(<https://en.wikipedia.org/wiki/Database>)

"A shared collection of logically related data and its description, designed to meet the needs of an organisation".

(Connolly and Begg, 2015)

"... a body of information held within a computer system using the facilities of a database management system."

(Oxford Dictionary of Computing, 1997)

Table 2: Different definitions of 'Database'.

So, there are different ways we can define a database. As Ritchie states: “... unfortunately, like many terms used in computer technology, there is considerable variation in interpretations placed on the word ‘database’. These range from the least prescriptive, ‘a collection of data’ – to the over-specific: ‘a large centralised shared data repository’, which seems to exclude a database on a personal computer”. (Ritchie, 2002).

Furthermore, we have already seen that it is possible to store data without a database, and we gave an example of using a spreadsheet application, noting some general problems related to file-based storage. The flipside to these problems, i.e., the motivation and purpose for the development of solutions to them, helps further define what databases are.

1.4.2 Purpose and motivation for using databases

- **Accessibility and ease of updating:** as the size of a company and, therefore, its data develops and grows, the accessibility to the data from the point of view of a growing number of individual user’s perspective should not change.
- **Accuracy and consistency:** should be facilitated by a database. We have seen how, within a spreadsheet application, errors are easily introduced, due to a lack of heuristics regarding data entry. Accuracy of data, for example, is afforded by a database that enforces consistency according to the application domain. In other words, domain rules can be imposed on the form of the data to help prevent problems such as an order’s delivery date preceding the data the order was initially placed.
- **Robustness:** data can survive catastrophic system failures, such as crashes, or even hard drive meltdowns, and that during such events none of the survived data is corrupt in anyway. Databases thus provide a strength to our data, providing confidence in the business as a whole that its main asset, critical data, cannot be damaged in any way.
- **Scalability:** This is a general property reflecting all of the above, but in light of the fact that data can grow from humble beginnings, i.e., small-sized, to large, to **HUGE**. When the accessibility, ease of updating, accuracy, consistency and robustness of data remain unchanged as a consequence of growth, then the database has proven scalability.
- **Security:** not mentioned above, but there can be good reasons (sensitivity of company data, privacy of medical health records) that data security is upheld.

1.4.3 Databases are ‘part of’ a database management system

During your studying endeavours you will no doubt become stuck on various things. It is important to understand, firstly, that this is normal and, secondly, that throughout your career (as a Software Developer, Software Architect) you will often find yourself browsing sites such as **stackoverflow**, which is a very useful resource in such situations. The style of the conversions, there, however, are very informal and we will take a look at one now:

Choosing the right database: MySQL vs. Everything else

A FAST, FLEXIBLE & SCALABLE PATH
Tools, sample code, libraries and more for your IoT project.

▲ 17 It would seem these days that everyone just goes with MySQL because that's just what everyone goes with. I'm working on a web application that will be handling a large quantity of incoming data and am wondering if I should "just go with MySQL" or if I should take a look at other open-source databases or even commercial databases?

▼

★ 6 Thanks, Ben

EDIT: Should have mentioned, am looking for optimal performance, integration with ruby + rails running on debian 5 and money is tight although if it will save money in the long run I would consider making an investment into something more expensive.

sql mysql oracle postgresql database

Figure 4: Stackoverflow "Databases" are not DBMS

Note the title of the 'Choosing the right database: MySQL vs. Everything else'. The post then goes on to presume, initially, that MySQL is a kind database. This is not the case, which is not a criticism of the post (there is no need to be semantically correct with every term you use when having an online conversation, that would be silly), but we use it here to point out that databases are separate to database management systems.

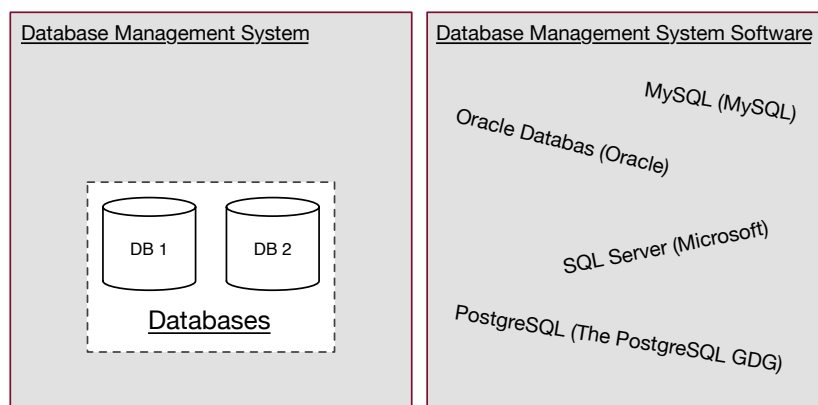


Figure 5: Databases and DBMS examples

1.4.4 Database types: a heads-up

Here, by *type* of database we refer to the fundamental/underlying mechanisms that are used to store and retrieve data.

It has not been mentioned, yet, that the current course is actually quite biased, or highly focused (for some good reasons), on a specific database approach; i.e., the course deal mostly with **relational** databases, which are currently still *the* most common types of database used. In the *relational* database approach, for example, we will learn that data is stored using *tabular-relational* mechanisms.

However, just as with other aspects of your degree, such as programming, where you might (again for good reasons) focus on a particular computer programming language, like Java, you

should at the same time be mindful of other approaches ideas concerning underlying database mechanism.

In celebrating the emergence of the relational database concept as the leading approach in database systems, Ritchie (2002) states that "... with the rapid growth in computer performance and the development of improved disk-accessing techniques, this [processor-intensive] disadvantage was soon overcome. The success of the relational database is such that it is the autonomous choice for virtually all new development work".

Ritchie also describes some 'early' database approaches, such as the 'Hierarchy' database approach, which is associated with production process involving hierarchical component assemblies. For example, cars have various component parts (wheels, engines), which can be broken down into further parts all the way down to 'atomic' features such as nuts and bolts, in a hierarchical way. Another 'earlier' approach is the network approach.

However, in computer science arguments as to the eventual success of one technique over another are not very often that compelling, especially when judged at a later date, as things change quickly, but where 'new' approaches often borrow concepts from earlier ones perhaps ahead of their time. For example, since the publication of Ritchie (2002), a movement known as NoSQL has started to gain popularity, due mainly to the specific data problems it addresses, more effectively, than the relational approach. Interestingly, e.g., the NoSQL movement draws inspiration from some of the "earlier" network approaches to which Ritchie (2002) refers.

We will revisit the NoSQL movement in a later week. The main point here is that you should be aware of the various different kinds of database approaches that exist, and that the reason they exist is that relational databases are not always (they often can be) the best choice. Like all software technologies, relational databases are part of an ecosystem. The best choice of tools often depends of the nuances of the task in hand. We therefore list numerous kinds of database approaches below.

- Hierarchical
- Network
- Object-Oriented
- NoSQL
- Relational

However, the good reasons for studying relational databases are as follows:

- While relational databases are not the only option, they are still very commonly used, both in academia and industry.
 - It is therefore important to be familiar with prevailing approaches in order to marketable from the point of view of the job market.
- Many relational database management systems operate on (and require familiarity with) similar principles. This is not necessarily the case with other systems. Learning the relational approach, therefore, has quite a broad *payback*.

- It might be argued that the relational approach is easier to learn. This does not mean that you should not have the ambition to look at 'more difficult' technologies at some point, just that relational approaches be a good place to start; they will provide a foundation that you will be able to build on as you progress through your course.

1.4.5 Database data modeling

Modelling start from page 4 Ritchie

Data and programs coupling

Modelling

1.5 Summary

- Vast amounts of data that are available to be exploited and converted into useful information.
- Structured data is useful to distinguish from unstructured data.
- Structured data organization can take different forms, but traditionally in businesses had been organised according to some form of file system storage.
- Duplication of data, it's scalability, accuracy and integrity are examples of limiting factors related to file-based storage, which motivates the use of databases.
- We noted the relational databases exist alongside other database types (e.g., hierarchical, object-orientated and noSQL), but that relational databases remain the most widely used database technology today.

~~~~~