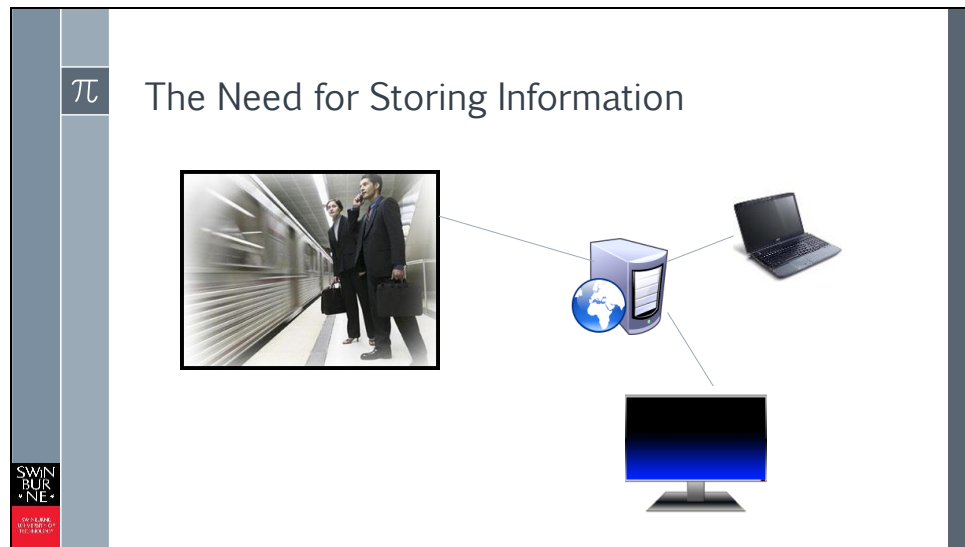


This introductory module discusses the main categories of data storage with some examples.



Everything humans do relies on information. Just to board a train, we need a timetable and directions to the correct platform.

We rely on our mobile devices for information in many situations. Once we sit down to work on our computers and laptops, we access even more information about our customers, our services, government regulations, competitors and much more.

The information we use is not all the same – Twitter feeds are very different from customer orders.

All of this information is brought to us by software, which interacts with data storage software. The type of data storage product we use and the structure we choose to store the data in – called a data model – depends on the type of information.

The slide features a vertical bar on the left with a Greek letter  $\pi$  and a Swinburne University of Technology logo. The main content is a list of data storage methods:

- › Unstructured Data
  - Every piece of data is different
  - Data types vary, if they exist
    - › Often text-based
- › Semi-structured Data
  - Irregular or incomplete
  - May have structure, but some of it may change
- › Structured Data
  - Has a rigid Schema

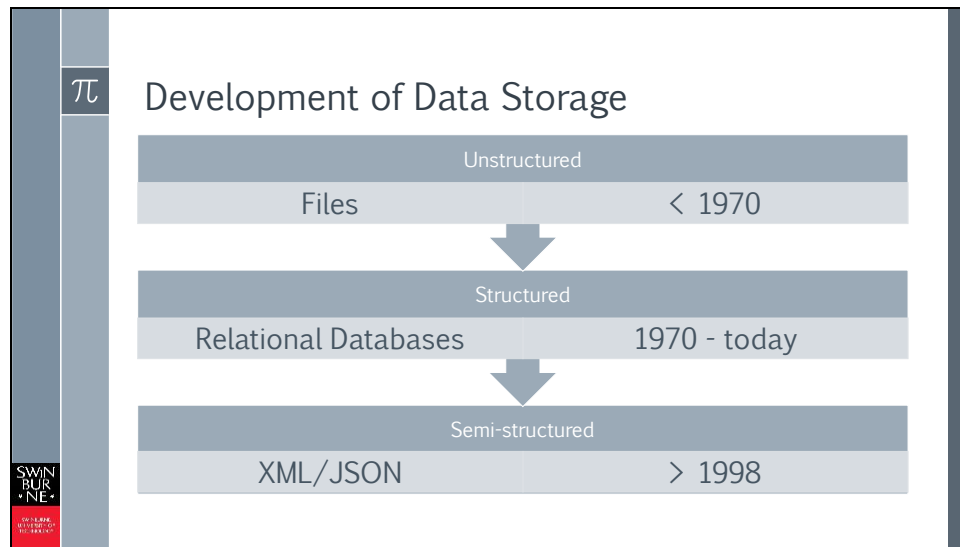
A callout box points to the word 'Schema' with the text: 'Description of the structure or model of the data'.

On a high level, the storage formats we use to store our data can be subdivided into unstructured, semi-structured and structured.

Structured data means every little piece of our data is defined with a type and tagged with a name. There are no surprises. If there is new data that does not fit into this pattern, we cannot store it.

Unstructured data has no predefined form. Every collection of data is presented differently. You can think of unstructured data as a collection of documents, such as letters. Every letter has many words, but the location of each word is different in each letter. Even if there are numbers in the text, we don't define them – we don't say the number on page 1 is a person's age and therefore an integer. It may appear in one letter but not another. Since we can't define any numbers, we consider the whole document a string.

Semi-structured data has some structure, which is often a tree. If the tree structure is defined, the semi-structured document has a schema. A schema is a definition of the pattern of the document. If a schema exists, all documents stored under that schema must have that same pattern. In semi-structured data, the schemas are not defined to the last detail. They allow for variation of some parts of the document. Structured data has a rigid schema – and all documents that match it have precisely defined parts.



Traditionally, data processing was done manually on unstructured data. When computing became more mainstream, automated processing led to a structured approach when E. F. Codd developed the relational model in 1970. The relational model has been researched for forty years and many standards and implementations exist. In the mid-1990s, when object-oriented programming languages became very popular, object-oriented databases began to emerge. The initial implementations, however, had performance problems and were never widely accepted.

Around the same time, the emergence of the World-Wide Web opened up possibilities for mining information from diverse sources. The lack of common structure of such information led to the need for semi-structured approaches. XML was finally ratified by the World Wide Web Consortium in 1998, and a large number of applications and libraries for xml parsing started to emerge. XML-based databases were also developed. Over time, the wordy syntax of XML was regarded as a major drawback and JSON was adopted as a less bulky alternative.

$\pi$  Unstructured Data

ORDER

John Lee has ordered 5 tablets on 13 January. They cost \$119 each.

DELIVERY NOTE

John Lee's 5 tablets were sent by truck on 5 February to 22 Boundary Lane Camberwell.

Invoice for John Lee	
5 tablets	\$595.0
GST	\$59.5
Total	\$654.5
Due date 16 March 2016	

Invoices can be seen as unstructured or semi-structured.

SWINBURNE UNIVERSITY OF TECHNOLOGY

Manual information management used files (paper-based or electronic). Paper-based files were used widely up to 1970. The first computer programs emulated the paper-based notes at first, and human operators had to interpret the files. Soon it became clear that there was great potential for further automation – if the data were to be structured, computer programs could make sense of them themselves, possibly even extracting information and communicating it to other computers without human intervention.

Documents like invoices can be seen as structured or semi-structured. They usually follow an implicit schema – the due date is always on the same line, as are the prices and totals. Some invoices also contain additional information as text, which would add an unstructured part and make them semi-structured.

$\pi$  Unstructured Data - Redundancy

ORDER


John Lee has ordered 5 tablets on 13 January. They cost \$119 each.

DELIVERY NOTE

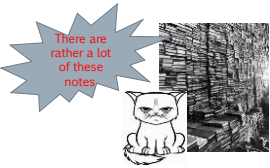
John Lee's 4 tablets for **\$135** were sent by truck on 5 February to 22 Boundary Lane Camberwell.

Invoice for John Lee

5 tablets	\$595.0
GST	\$59.5
Total	\$654.5
Due date 16 March 2016	



What?? Did we forget his discount??



There are rather a lot of these notes


Apart from the problem of computers having a hard time extracting facts from unstructured data, most often unstructured data leads to redundancy – unnecessary duplication. It is also a good way of filling up storage space. When there are several versions of the same piece of information, at some point there will be inconsistency. After a while, the different files will contradict each other – one says John Lee's tablets cost \$119, the other \$135.

$\pi$


## Unstructured Data – Search Efficiency

2015 ORDERS

John Lee ordered 5 tablets on 13 January. They cost \$119 each and were delivered on 5 February.  
Peter Brown ordered 10 PCs on 20 January. They cost \$450 each and were delivered on 27 January.  
Stephen Nguyen ordered 5 printers on 1 February. They cost \$250 each and were delivered on 7 February.  
...



How many laptops did we sell in 2015??



How long can this take?

SWINBURNE  
UNIVERSITY  
OF TECHNOLOGY

Computers have made searching documents faster compared to humans reading through them, but unstructured data is most often freetext and difficult for a program to interpret.

In our list of orders, even if the program identifies the numbers in the text correctly, it is very hard for a computer program to tell whether the product we are selling is a tablet, a PC or a printer. To do this, we would need sophisticated word matching software and is still error-prone.

Searching for a keyword in a large document is a time-consuming affair because the search is sequential – it has to go through every letter, space and number in sequence.

$\pi$

## Structured Data


Name	Product	Quantity	Delivered
John Lee	tablet	5	05/02/2016

Variable  
number of  
characters,  
max 30

Variable  
number of  
characters,  
max 20

Integer, max  
4 digits

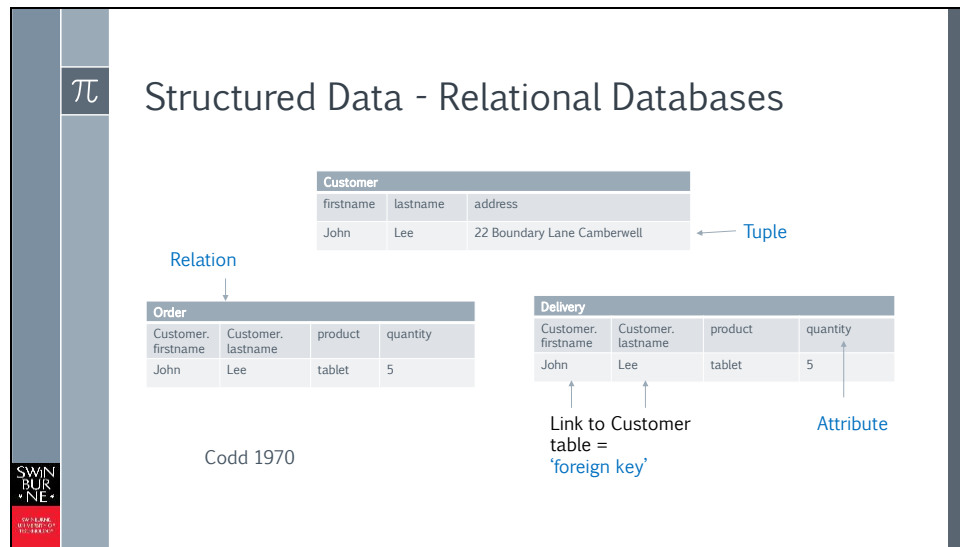
Date, local  
Australian  
format



When the storage format of the data is highly structured, each piece of data has an exactly defined type and name. Generally, there is a software that manages the data, and this software enforces the data type – if the date format is wrong, the date will not be accepted. This goes a long way towards avoiding errors. Even when the information is ‘chopped up’ into small pieces, the structure of the format still groups the pieces in a way that makes it rather obvious what the data is about. In this table, we can still guess that John Lee has received 5 tablets on 5 February 2016, and we do not even know the schema of this database.

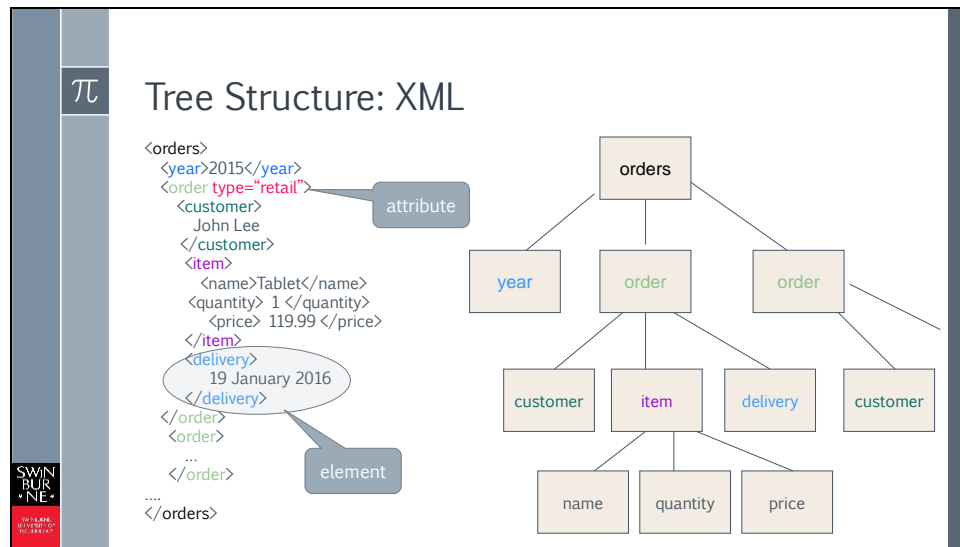
When you are working with structured data, it is best to have the schema at hand to interpret all the items correctly. The most common data model for structured data is the relational model. The schema of such data is most commonly presented as an ER diagram.





A relational database consists of several tables – called relations with rows that are called tuples. The tuples are defined so that they describe a single entity – a person, such as a customer, or an order, or a company. Because the relations are structured, they contain tuples with equivalent data – all rows of the table have the same attributes with the same data types, only different values.

There often are 'natural' relationships between the tables (called relations), and therefore the tables contain 'links' to other tables which are called foreign keys. In the relational model, one of the challenges is that we often need information that consists of data from several tables, and the tuples of several relations have to be combined. This is a complex process that can be computationally expensive. An entire query language called SQL has been developed to provide a multitude of ways to combine and filter data.



XML is an example of semi-structured data. It has elements that consist of tags that encase a value. An element can also be the value of another element – in this example the order tag's value is a group of elements. The group contains one customer element, one delivery element and one or more item elements. Because XML elements can be nested, they form a tree structure. The more enclosing tags there are, the lower the element is in the tree.

For computers to be able to read and interpret the document, the structure has to be defined. There are two technologies that can describe XML document formats: Document Type Definitions (DTDs) and XML Schema. While an XML document is considered all text, an XML Schema definition can define data types – so it could define the quantity as an integer and the price as a double. In this way, a program could convert the values of quantity and price to the appropriate number format and work with them.



$\pi$

## XML – Semi-structured

```

<orders>
  <year>2015</year>
  <order>
    John Lee has ordered 1 tablet that
    was delivered on 19 January 2016. It
    cost $119.99.
  </order>
  <order>
    ...
  </order>
  ....
</orders>

```

This is entirely possible in XML. The author of the document format decides.

SWINBURNE  
UNIVERSITY  
OF TECHNOLOGY

In case you wondered why we call XML documents semi-structured, not structured, here is why. There is no requirement for you to subdivide the document into its smallest pieces. You can have elements whose values are several paragraphs long. While this seems to make little sense in this case of customer orders, you can imagine it would suit a Twitter feed. No two tweets ever have the same structure. As the author of the document, you have to decide which is the most usable format.

So Billy the cat, who is despondent about us having done a U-turn towards unstructured data, is actually more negative about it than necessary..

$\pi$

## Unstructured: Still needed

How many people use Safari to view your web site?

```
180.76.15.31 - [09/Jun/2015:17:12:08 -0700] "GET /Archive/ HTTP/1.1" 200 1798 "-" "Mozilla/5.0 (compatible; Baiduspider/2.0; http://www.baidu.com/search/spider.html)"
"www.redug.com" 50.118.159.140 - [09/Jun/2015:17:17:45 -0700] "GET /logs/access.log HTTP/1.1" 200 178 "http://redug.com/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3)
AppleWebKit/536.29.13 (KHTML, like Gecko) Version/6.0 Safari/536.29.13" "redug.com" 61.152.102.40 - [09/Jun/2015:17:51:0700] "GET /logs/access.log HTTP/1.1" 200 304
"redug.com" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3) AppleWebKit/536.29.13 (KHTML, like Gecko) Version/6.0 Safari/536.29.13" "redug.com" 220.181.108.115 - [
09/Jun/2015:17:17:51-0700] "GET /old_socialview.htm HTTP/1.1" 200 4518 "-" "Mozilla/5.0 (compatible; YandexBot/3.0; http://yandex.com/bot)" "redug.com"
104.209.130.212 - [09/Jun/2015:17:18:15 -0700] "GET /paper0004J00409ragWar.htm HTTP/1.1" 200 2865 "http://redug.com/" "Mozilla/5.0 (Windows NT 6.1; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.116 Safari/537.36" "redug.com" 77.247.181.162 - [09/Jun/2015:17:21:05-0700] "GET /logs/ HTTP/1.1" 200 50141
"http://tophamerson.com/" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.124 Yabrowser/14.10.2062.12061 Safari/537.36"
"redug.com" 100.43.81.131 - [09/Jun/2015:17:22:48-0700] "GET /robots.txt HTTP/1.1" 200 37 "-" "Mozilla/5.0 (compatible; YandexBot/3.0; http://yandex.com/bot)" "redug.com"
23.225.30.164 - [09/Jun/2015:17:28:44-0700] "GET /logs/access.log HTTP/1.1" 200 560 "http://redug.com/" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/40.0.2214.85 Safari/537.36" "redug.com" 117.146.116.19 - [09/Jun/2015:17:27:49-0700] "GET /logs/ HTTP/1.1" 200 50141 "http://www.hvdeesgpmos.com/" "Mozilla/5.0
(Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.71 Safari/537.36" "redug.com" 147.143.2.221 - [09/Jun/2015:17:32:05-0700] "GET /logs/ HTTP/1.1" 200
50141 "http://www.hamsterpomox.com/" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.71 Safari/537.36" "redug.com" 178.19.108.220 - [
09/Jun/2015:17:33:04-0700] "GET /logs/access.log HTTP/1.1" 200 669 "http://redug.com/" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/40.0.2214.85 Safari/537.36" "redug.com" 86.106.16.217 - [09/Jun/2015:17:45:28-0700] "GET /logs/access.log HTTP/1.1" 200 2485 "http://redug.com/" "Mozilla/5.0 (Windows NT
6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.85 Safari/537.36" "redug.com" 23.229.30.164 - [09/Jun/2015:17:48:07-0700] "GET /logs/access.log
HTTP/1.1" 200 716 "http://redug.com/" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.85 Safari/537.36" "redug.com"
204.12.216.18 - [09/Jun/2015:17:52:48-0700] "GET /Archive/195022AnsSect.htm HTTP/1.1" 200 9028 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) " "redug.com"
123.125.71.46 - [09/Jun/2015:17:53:53-0700] "GET /sitemap.xml HTTP/1.1" 200 188 "-" "Mozilla/5.0 (compatible; Baiduspider/2.0; http://www.baidu.com/search/spider.html)" "redug.com"
180.180.98.50 - [09/Jun/2015:18:03:18-0700] "GET /logs/access_150224.log HTTP/1.1" 200 40272 "http://redug.com/" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/40.0.1847.116 Safari/537.36" "redug.com" 180.180.98.50 - [09/Jun/2015:18:03:21-0700] "GET /wp-
login.php?action=register HTTP/1.1" 404 88 "http://redug.com/wp-login.php?action=register" "Opera/9.80 (Windows NT 6.2; Win64; x64) Presto/2.12.388 Version/12.17" "redug.com"
180.76.15.6 - [09/Jun/2015:18:03:50-0700] "GET /sitemap/DBA HTTP/1.1" 200 344 "-" "Mozilla/5.0 (compatible; Baiduspider/2.0; http://www.baidu.com/search/spider.html)"
"www.redug.com" 94.231.176.150 - [09/Jun/2015:18:09:37-0700] "GET /logs/access_130930.log HTTP/1.1" 404 89 "http://achatslimex.aircus.com/" "Mozilla/5.0 (Windows NT
6.1; en-US; rv:1.9.2.22 Gecko/20110902 Firefox/3.2.22" "redug.com" 94.231.176.150 - [09/Jun/2015:18:09:37-0700] "GET /logs/access_130930.log HTTP/1.1" 404 73
"http://achatslimex.aircus.com/" "Opera/9.80 (Windows NT 6.0; U; ru) Presto/2.10.289 Version/12.02" "redug.com" 94.231.176.150 - [09/Jun/2015:18:09:58-0700] "GET
/access_130930.log HTTP/1.1" 404 73 "http://achatslimex.aircus.com/" "Opera/9.80 (Windows NT 6.0; U; ru) Presto/2.10.289 Version/12.02" "redug.com" 94.231.176.150 - [
09/Jun/2015:18:09:59-0700] "GET /logs/access_130930.log HTTP/1.1" 404 73 "http://achatslimex.aircus.com/" "Mozilla/5.0 (Windows NT 6.1; rv:17.0) Gecko/17.0 Firefox/17.0" "redug.com"
94.231.176.150 - [09/Jun/2015:18:09:59-0700] "GET /access_130930.log HTTP/1.1" 404 73 "http://achatslimex.aircus.com/" "Mozilla/5.0 (Windows NT 6.1; rv:17.0) Gecko/17.0
Firefox/17.0" "redug.com" 94.231.176.150 - [09/Jun/2015:18:10:00-0700] "GET /logs/access_130930.log HTTP/1.1" 404 73 "http://achatslimex.aircus.com/" "Opera/9.80 (Windows NT 6.1;
WOW64; U; Edition Next; Edition Yx; ru) Presto/2.11.310 Version/12.50" "redug.com" 94.231.176.150 - [09/Jun/2015:18:10:01-0700] "GET /access_130930.log
http://achatslimex.aircus.com/" "Opera/9.80 (Windows NT 6.1; WOW64; U; Edition Next; Edition Yx; ru) Presto/2.11.310 Version/12.50" "redug.com"
```



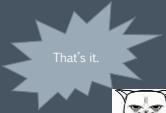
Apache access log

Dealing with unstructured data is still a practical problem many people face. The most prominent example is log files. Many applications write logs, and the log files only have to be read when something goes wrong. Hopefully this only happens in a small percentage of cases, meaning that most log files are never looked at and probably overwritten after a while. So it makes no sense to try to format them neatly.

In the event when we need to read them, we need good tools to help us with this. There are many text processing tools, both visual and command line. As an IT professional, you have to be familiar with at least one of them.

The example shows a log file of an instance of an Apache web server. It logs all the HTTP requests it gets, what IP address they come from, what operating systems these computers run and what browser type. If you are optimising a web page, you may be interested in what browsers people use. You can't find this information in a hundred page log file – you have to use the right tools.

## Summary



- › The world needs to store more and more data.
- › Data can be stored in a more or less structured way, depending on how precisely the individual pieces of data are defined.
- › Data redundancy can be a problem because repetition adds to the volume, but most importantly, because there can be inconsistencies.
- › Relational databases are an example of structured data formats.
- › Unstructured data is still needed for some purposes. You have to know how to handle it efficiently.

Here are the most important points discussed in this module. You may want to stop the recording to have a read through them. When you are ready, start the quiz about this module.