

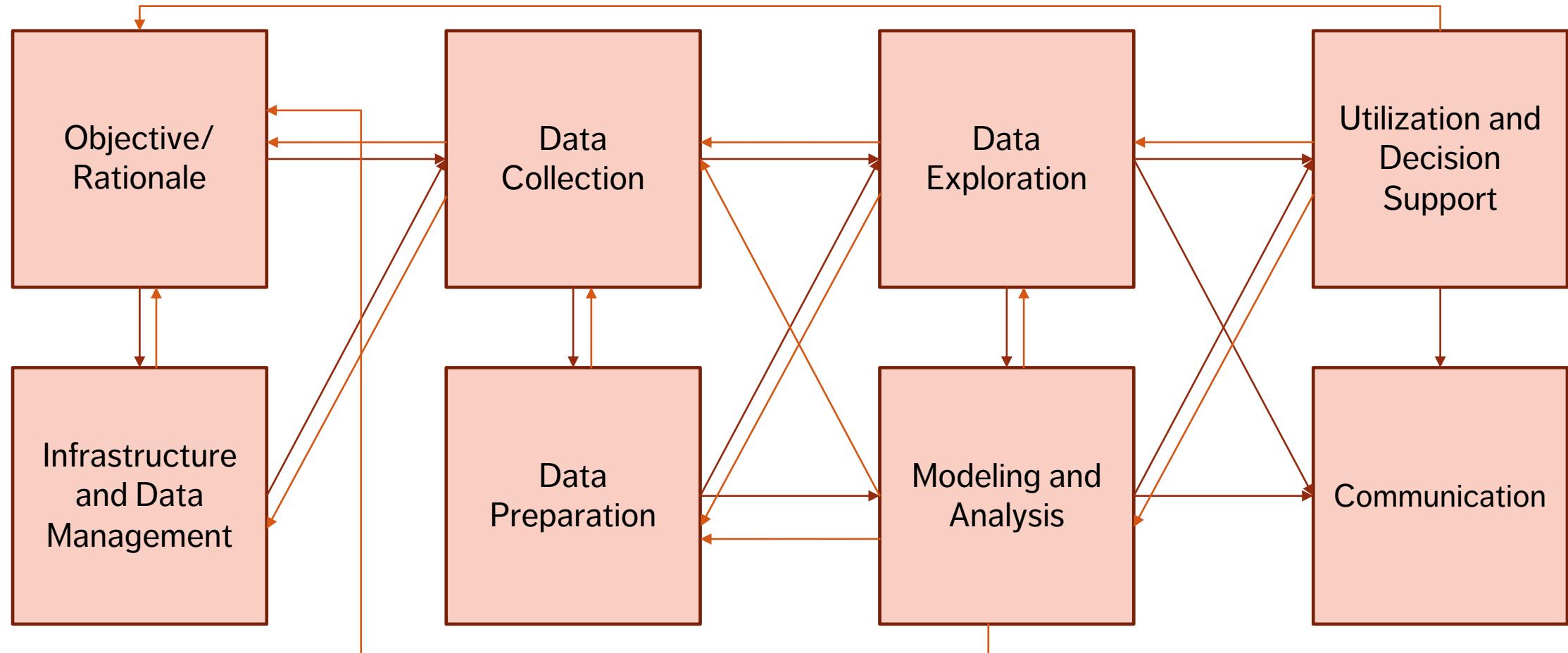


# DATA + INFORMATION ARCHITECTURE

SETTING THE STAGE

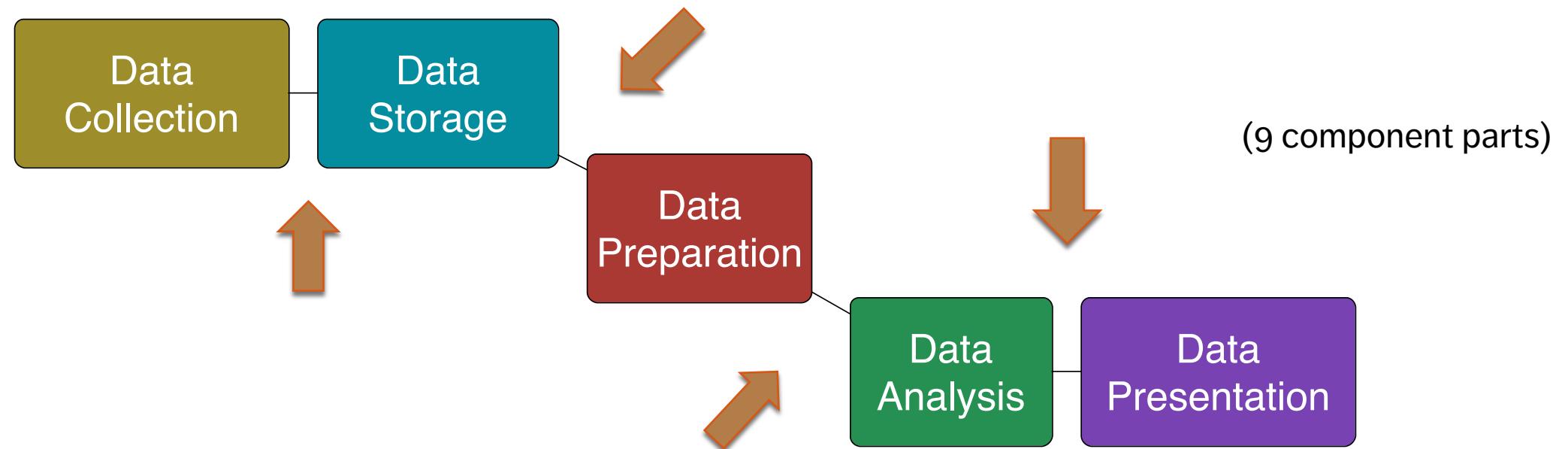


# THE (MESSY) ANALYSIS PROCESS



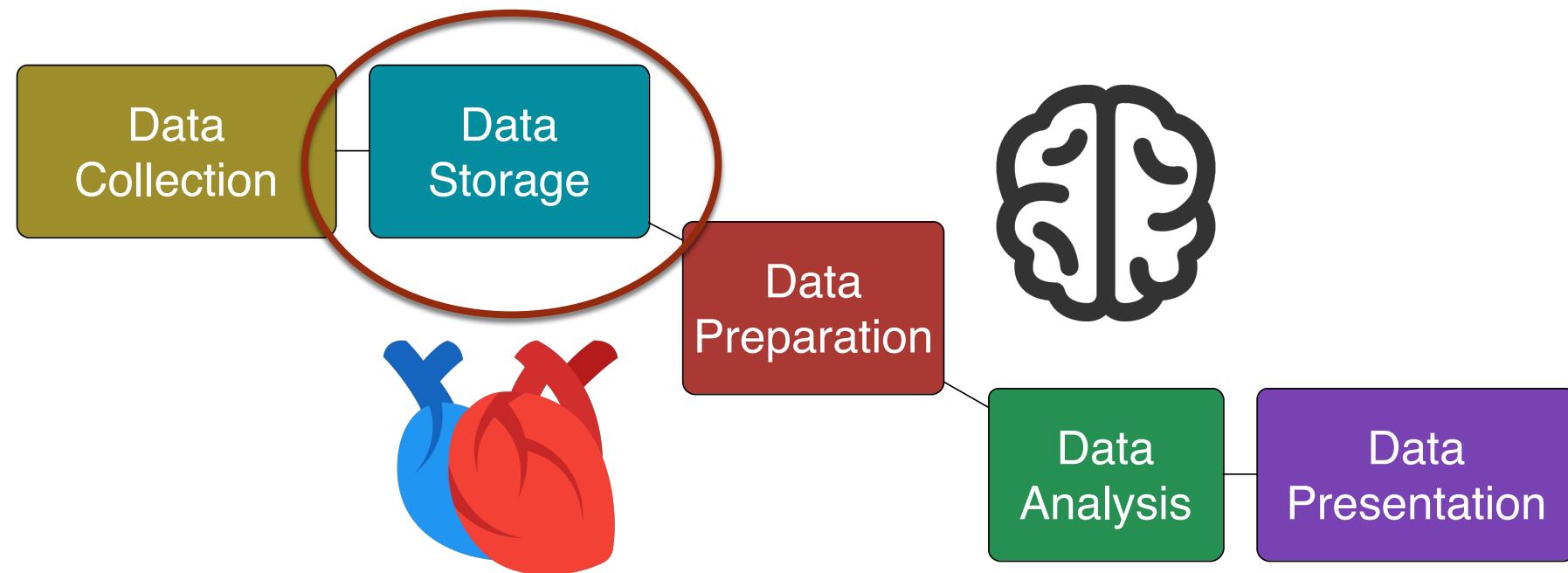
# AN IMPLEMENTED (AUTOMATED) DATA PIPELINE

In the **service delivery context**, you may eventually want one of these...



(as always - beware model 'drift'!)

# AN IMPLEMENTED (AUTOMATED) DATA PIPELINE



# MOTIVATIONS FOR COLLECTION

Three functions, historically:

- Record Keeping (People/Societal Management!)
- Science - New General Knowledge
- Intelligence - Business, Military, Police, Social? Domestic? Personal!



# DIFFERENT DATA CULTURES, DIFFERENT TERMS

## Business Intelligence:

- data warehouse + data mart
- data 'dimension' (= data set)
- hierarchical data (slices)
- data element
- dimension table + fact table

Data

## Science/Statistics:

- experimental data
- trials
- participants
- variables
- correlation

Knowledge

## Record Management:

- information architecture
- file plan
- Information resource
- field
- form
- subject

Information

Data Science

# ENTER COMPUTERS! AND COMPUTER SCIENCE!

Computer science (and information science) has its own theoretical, **fundamental** viewpoint about data, and information.

Computers operate over data in a fundamental sense - 1s and 0s that represent numbers, letters, etc. – **data becomes digital**.

Pragmatically, data is now stored on computers, and is accessible through our world-wide computer network.



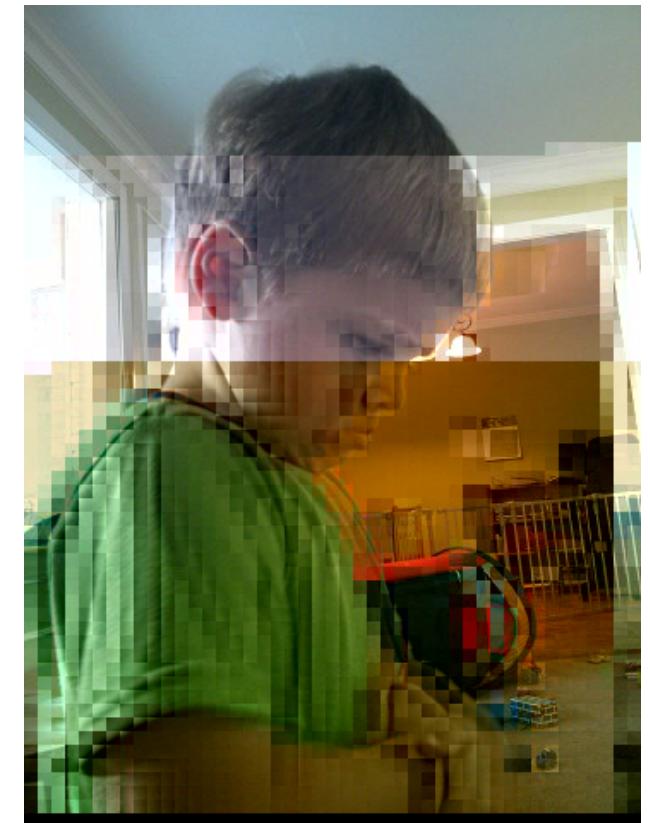
# DATA DECAYS

Data ages over time - it has a shelf life.

We use the phrase "rotten data" or "decaying data":

- **literally** - in the sense that the data storage medium might decay
- **metaphorically** - when the data no longer accurately **represents** the relevant objects and relationships or even when those objects no longer exist in the same way

Your data must be 'fresh', 'current', not 'stale' (context and model dependent!)

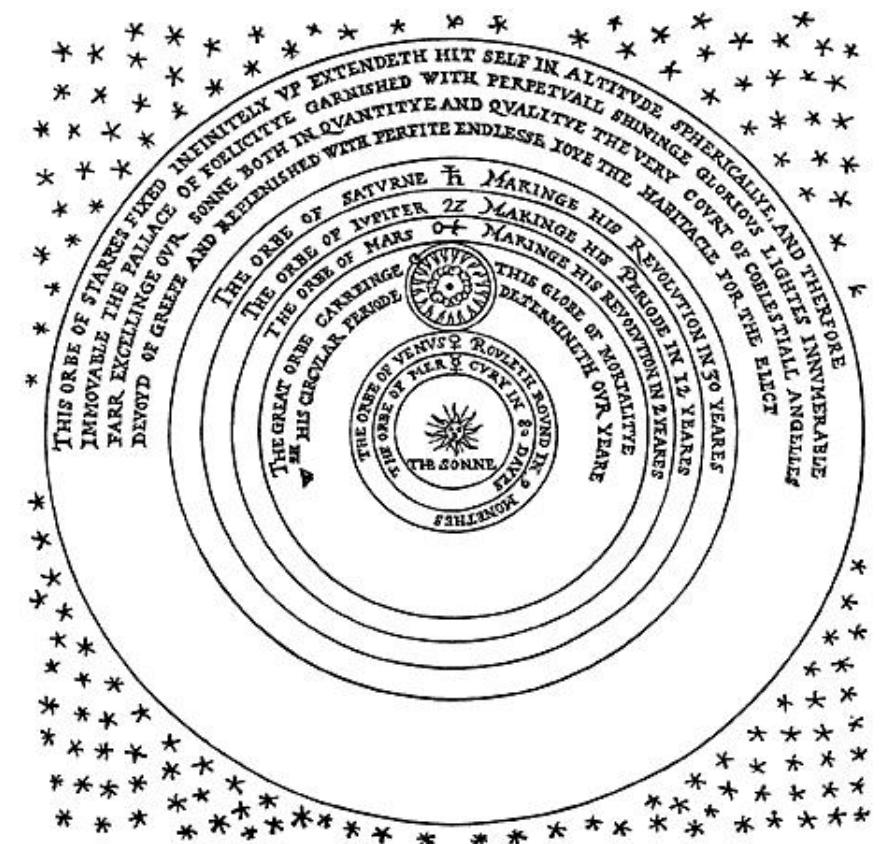


# CORE CONCEPTS

How do we cut across all of the different disciplines that use data?

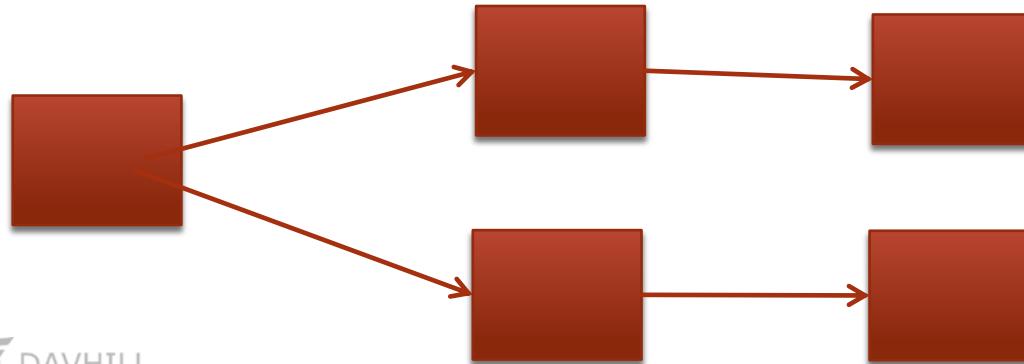
Core (systems) concepts or elements:

- object – attributes (concrete or abstract)
- multiple objects - **relationships** between these objects/attributes
- how these elements change over time



# RELATIONSHIPS WITHIN SYSTEMS

- some fundamental relationships:
  - part-whole
  - is-a
  - is-a-type-of
  - cardinality (one-to-one, one-to-many, many-to-many)
- some more object specific relationships:
  - ownership,
  - social relationships
  - becomes,
  - leads-to



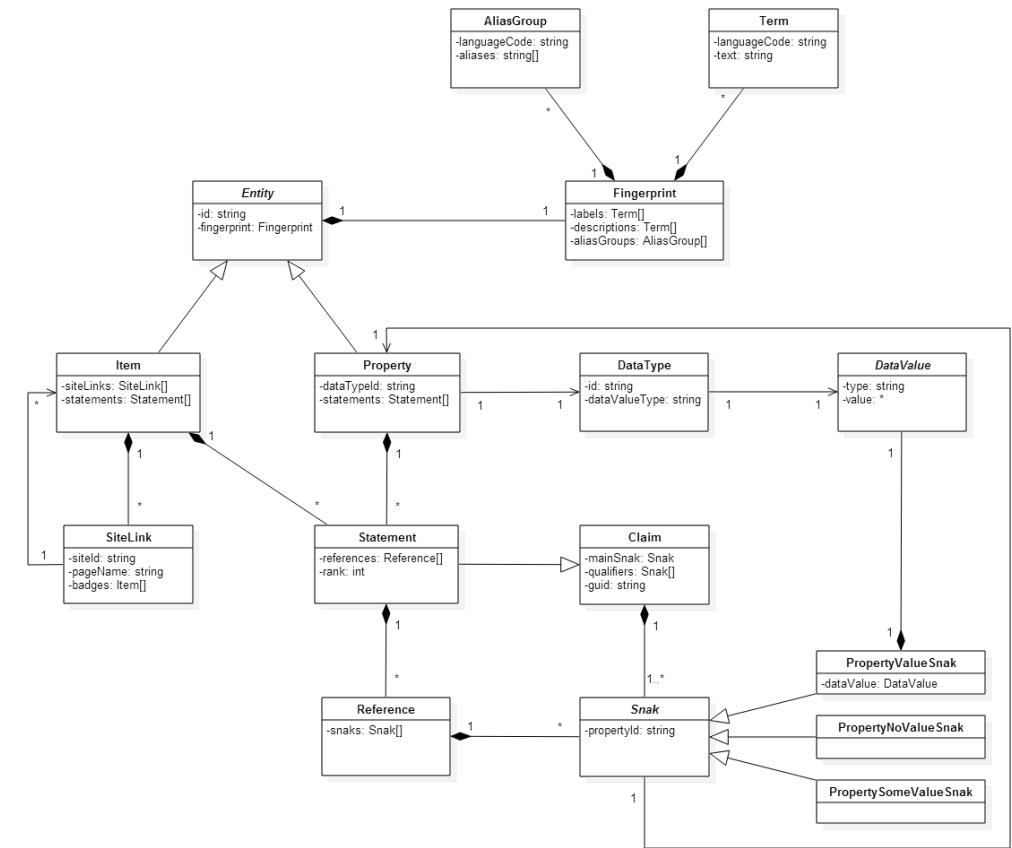
# FORMAL CONCEPTUAL MODELLING LANGUAGES

Conceptual modelling is not an exact science – it's more about making your internal conceptual models **explicit** and **tangible**.

It provides you with an opportunity to examine and explore your ideas and assumptions.

That said, various efforts have been made to formalize conceptual modelling:

- UML – Universal Modelling Language
- Entity Relationship (ER) Models - but they are generally connected to relational databases



# METHODOLOGICAL MODELS

Conceptual Model



Mathematical Model

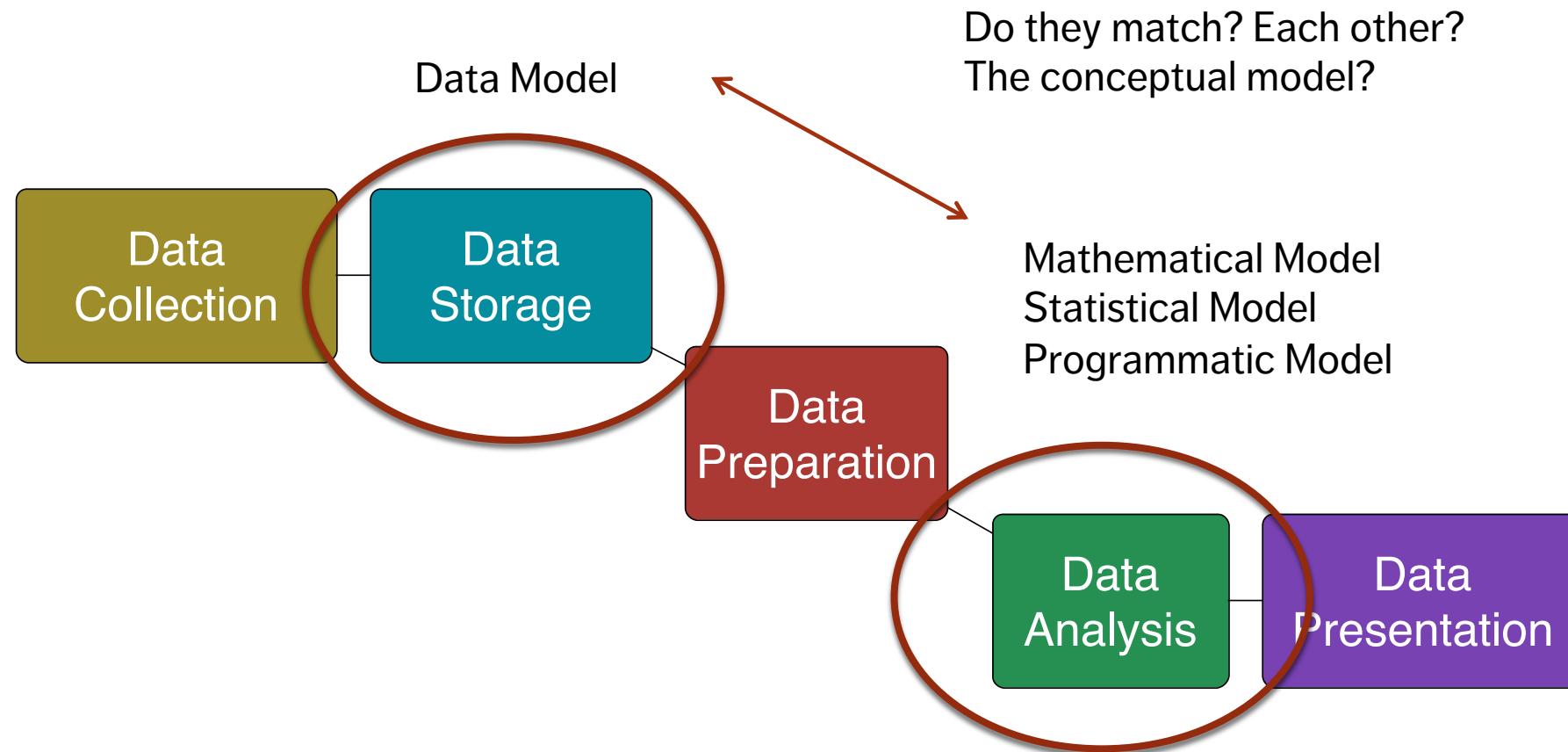
Statistical Model

Programmatic (Computer) Model

**Ontology (Knowledge Model)**  
(Simulation)

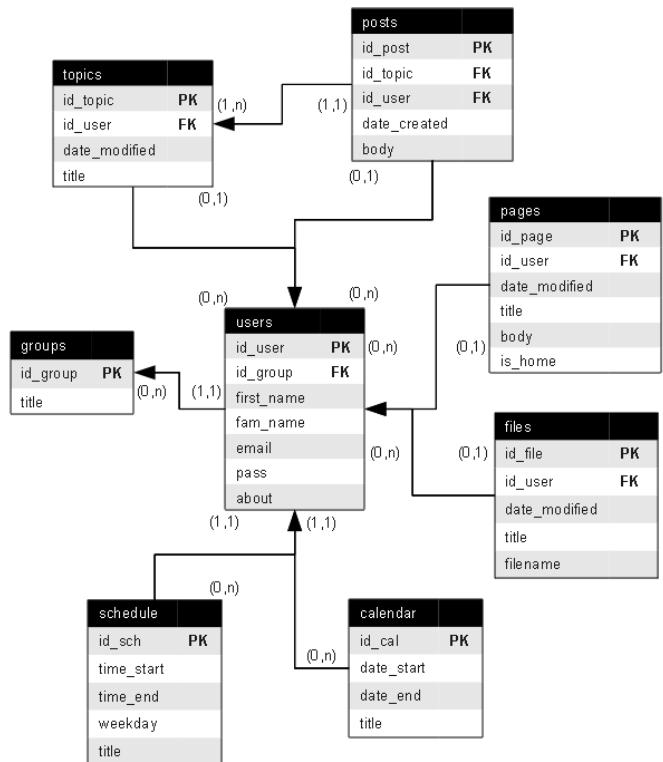
Data Model

# AN IMPLEMENTED (AUTOMATED) DATA PIPELINE



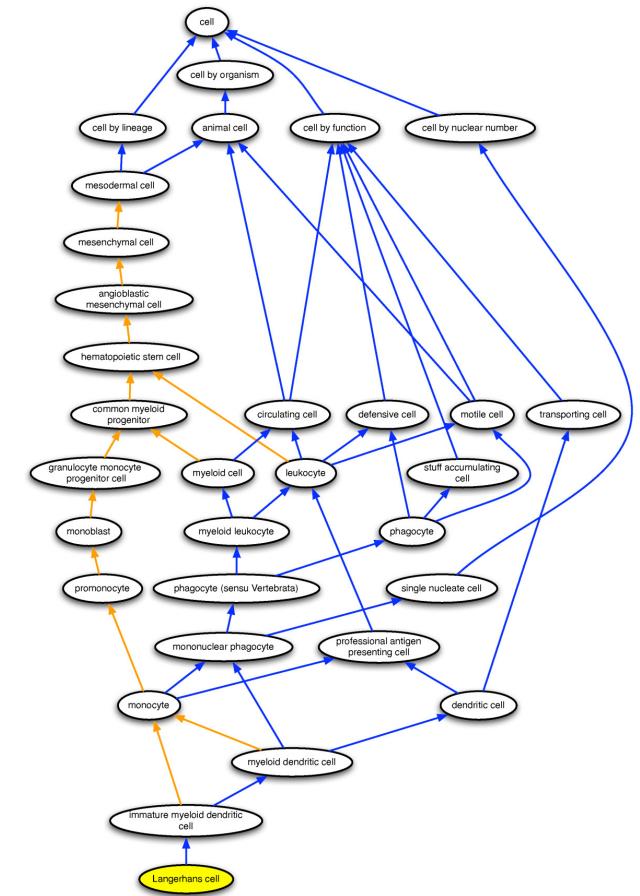
# WHAT IS DATA MODELLING

- A data model is an abstract (logical) description of a system, constructed in terms that can then be implemented as the structure of a type of data management software.
- You might argue that it is half way between a conceptual model and a database implementation.
- The data itself is about **instances** – the model is about the **object types**.
- There is another option that is worth considering on this front – **ontologies**.



# ONTOLOGY – KNOWLEDGE MODEL

- A structured, machine-readable collection of **facts** about a domain.
- Motivated by a desire to create increasingly machine readable but still conceptually sophisticated data.
- you could describe in a slightly tongue in cheek manner as 'a data model on steroids'.
- An attempt to get closer to the level of detail of a full conceptual model.



## METADATA TO PROVIDE CONTEXT

- We lose something when we move from our conceptual model to a specific type of model – e.g. the data or knowledge model.
- One way of keeping the context is to provide (hopefully rich) metadata – data **about** our data!
- Metadata is crucial when it comes to carrying out strategies for working across datasets.
- Ontologies can also play a role here!

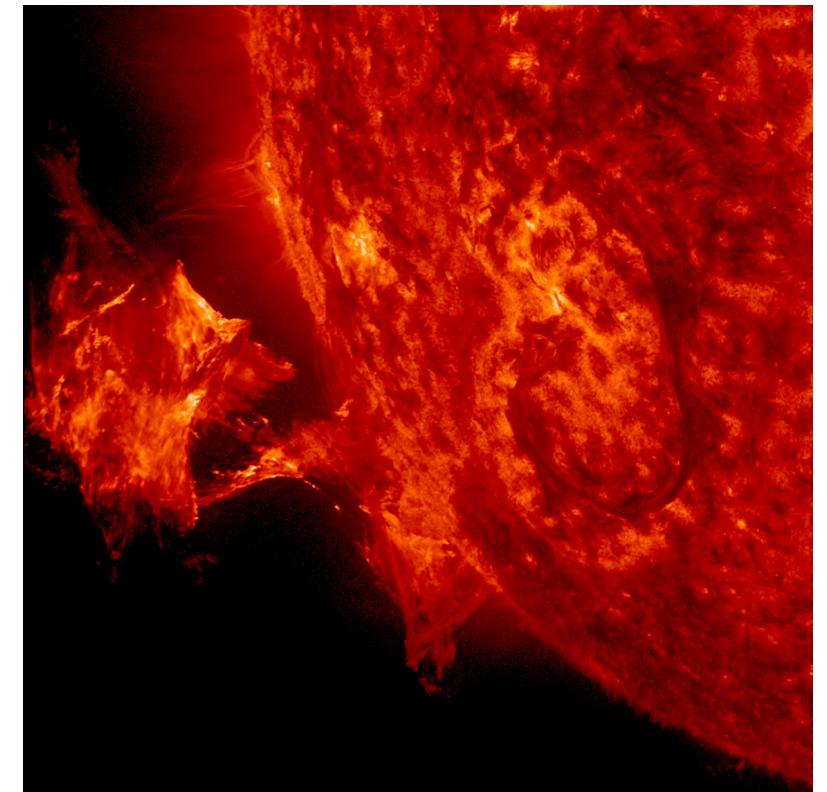
# STRUCTURED VS UNSTRUCTURED DATA

A major motivator for some of the new developments in types of databases and other data storing strategies is the increasing availability of **unstructured** data and also '**blob**' data.

**Structured Data:** labelled, organized, discrete, structure is constrained and pre-defined

**Unstructured Data:** not organized, no specific pre-defined structure data model – e.g. text in a document

**Blob Data:** Binary Large Object (BLOB) – images, audio, multi-media



# WHAT IS DATA MODELLING

We're going to look at four different options that are currently popular in terms of fundamental **data and knowledge** modelling or structuring strategies:

- key-value pairs (e.g. JSON)
- triples (e.g. resource description framework (RDF))
- graph databases
- relational databases

NoSQL

# KEY - VALUE STORES AND TRIPLE STORES

These are relatively unstructured ways to store data:

- **Key Value:** all data is simply stored as a giant list of keys and values, where the key is a name or label (possibly of an object) and the value is a value associated with this.
- **Triple:** Data is stored as subject – predicate – object

## EXAMPLES

**apple type - apple colour**

Granny Smith - green

Red Delicious – red

**person - shoe size**

Jen Schellinck - women's size 7

Colin Henein - men's size 10

**word - definition**

**url – webpage**

**report name – report (document file)**

Triples add a verb to the mix:

**Person - is - age**

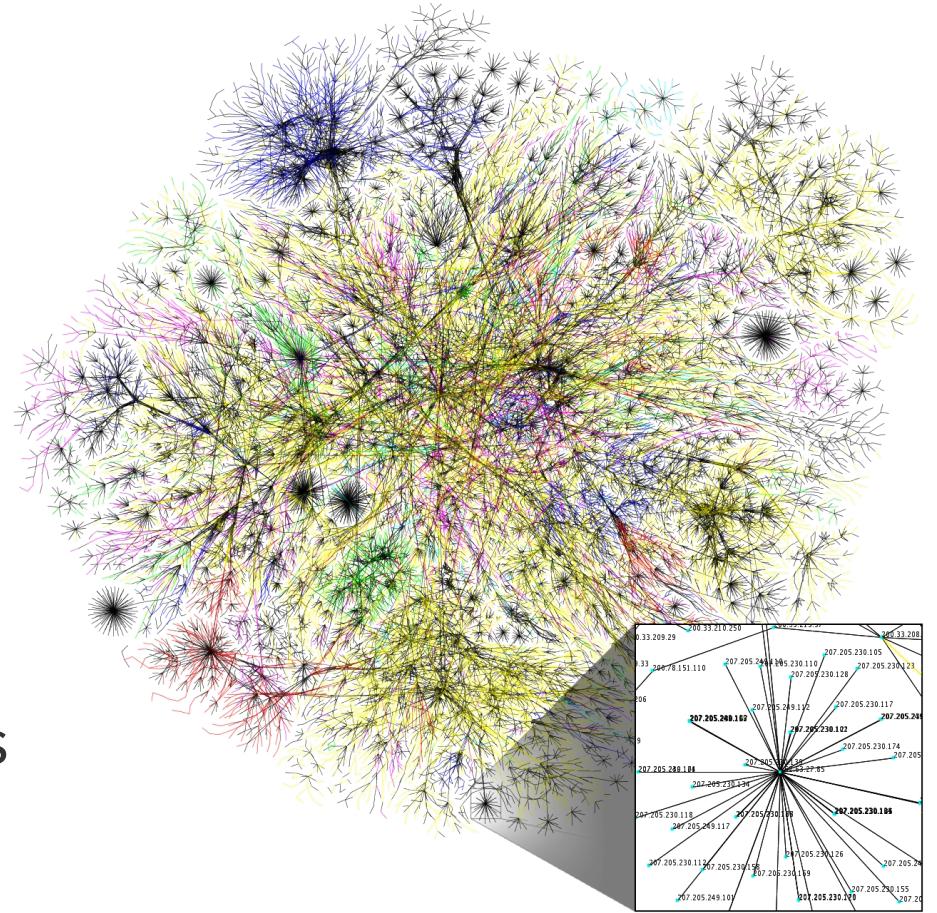
**Object - is-colour - colour**

# GRAPH DATABASES

Emphasis on the **relationships** between different types of objects, rather than between an object and the properties of that object.

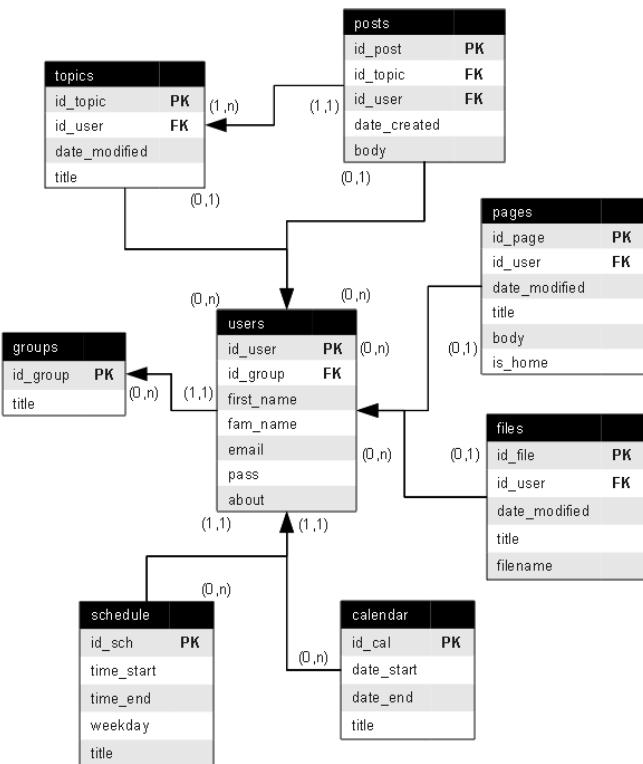
## The data model:

- objects represented by nodes
  - relationships between these objects represented by edges
  - objects can have a relationship with other objects of that same type - person is-a-sibling-of person



# RELATIONAL DATABASES

- Data stored in a series of tables
- Broadly speaking, each table represents an object and some properties related to this object.
- Special columns in tables connect object instances across table.



## PROS, CONS AND USE CASES OF EACH

**Relational Database:** widely supported, well understood, works well for many types of systems and use cases. But difficult to change once implemented, doesn't deal with relationships well (despite the name).

**Key-Value Stores:** can take any sort of data, don't need to know much about its structure advance, if you have many missing values these won't take up space. But can be messy and mysterious, difficult to find data.

**Graph Databases:** fast and intuitive if you are using heavily relation-based data, might be the only option if your data is like this because traditional databases may slow to a crawl. But probably overkill/too specialized if your data is not like this, not yet widely supported.

# A NOTE ON FLAT FILES AND SPREADSHEETS

What about keeping your data in a single giant table (spreadsheet)? Or multiple spreadsheets? How bad can it be?

Wayne Eckerson coined the term ‘spreadmart’ (as opposed to data mart) to describe a situation with many (ad hoc) spreadsheets as a data strategy.

Date	Con	Lab	LDs	SNP	UKIP	Greens	Con av	Lab av	LD av	SNP av	UKIP av	Green av
15 September 2017	41	41	5	4	5	3	40.7	41.4	6.8	3.3	4	2.7
15 September 2017	39	38	8	3	6	4	40.7	41.7	7	3.2	3.8	2.6
13 September 2017	41	42										
10 September 2017	42	42										
1 September 2017	38	43										
Date	Con	Lab	LDs	SNP	UKIP	Greens	Con av	Lab av	LD av	SNP av	UKIP av	Green av
15 September 2017	41	41	5	4	5	3	40.7	41.4	6.8	3.3	4	2.7
15 September 2017	39	38	8	3	6	4	40.7	41.7	7	3.2	3.8	2.6
13 September 2017	41	42	7	4	3	2	40.9	42.2	6.8	3.3	3.5	2.4
10 September 2017	42	42	7	3	4	3	40.9	42.2	7	3.2	3.5	2.4
1 September 2017	38	43	7	3	1	4	40.9	42.3	7	3.2	3.4	2.3
31 August 2017	41	42	6	4	4	2	41	42.1	7.1	3.2	3.9	2
22 August 2017	42	42	7	2	3	3	41	42.2	7	3.1	4	2
22 August 2017	41	42	8	4	4	1	40.8	42.5	7	3.3	3.9	1.8
18 August 2017	40	43	6	4	4	2	40.5	42.9	6.8	3.3	3.9	1.8
11 August 2017	42	39	7	2	6	3	40.6	42.9	6.9	3.2	3.8	1.8
1 August 2017	41	44	7	3	3	2	40.5	43	6.9	3.2	3.4	1.7
19 July 2017	41	43	6	4	3	2	40.3	43.1	6.7	3.2	3.6	1.7
18 July 2017	41	42	9	3	3	2	40.3	43.4	6.7	3.1	3.5	1.6
16 July 2017	42	43	7	3	3	2	40.3	43.6	6.4	3.1	3.4	1.5
15 July 2017	39	41	8	3	6	1	40.0	43.8	6.4	3.1	3.4	1.6
14 July 2017	41	43	5	3	5	2	40.5	43.8	6.4	3.1	3.0	1.7
11 July 2017	40	45	7	4	2	1	40.4	43.9	6.5	3.1	2.8	1.6
6 July 2017	38	46	6	4	4	1	40.4	43.8	6.5	3.0	2.9	1.7
3 July 2017	41	43	7	3	3	2	40.8	43.4	6.5	2.9	2.7	1.8
30 June 2017	41	40	7	2	2	2	40.8	43.5	6.4	2.9	2.7	1.8
29 June 2017	39	45	5	3	5	2	40.7	44.2	6.3	3.0	2.8	1.7
30 June 2017	41	40	7	2	2	2	40.8	43.5	6.4	2.9	2.7	1.8
29 June 2017	39	45	5	3	5	2	40.7	44.2	6.3	3.0	2.8	1.7

## A NOTE ON FLAT FILES AND SPREADSHEETS

- **Pros** - very efficient if you are only collecting data once, about one particular type of object (e.g. scientific studies!), some types of analysis require you to have all the data in one place, so you are going to need to generate a flat file anyway. Easy to read into analysis software (e.g. R) and do operations over the entire dataset
- **Cons** - very hard to manage data integrity over the long term if you are continually collecting and working with the data. Doesn't work well if you are dealing with data on a system involving many different types of objects and their relationships. Can be very difficult to carry out data querying operations

# SOME TOOLS AND BUZZWORDS

- MongoDB, ArangoDB
- Document store
- JSON, YAML
- API, GraphQL
- Linked Data
- Semantic Web
- Web Ontology Language (OWL)
- Protégé

# IMPLEMENTING YOUR MODEL

- To implement your data/knowledge model, you need to have access to **data storage and management software**.
- Access might be challenging for you, the individual, because traditionally such software runs on servers.
- Servers are good because they allows many people to access a single database at the same time, from different client programs. However, it makes it hard to 'play' with a database.
- SQLite to the rescue!

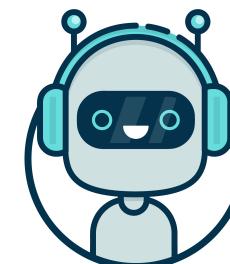
# ROLE OF DATA MANAGEMENT SOFTWARE

Data management software provides humans with an easy way to interact with their data.

It's essentially a human – data – interface.

Through this interface you can:

- add data to your data collection
- extract subsets of data from your collection based on certain criteria
- delete or edit data in your collection



# A DATA WHAT??

Previously:

- Database
- Data warehouse
- Data Marts
- Database Management System
- (SQL)

Now:

- Data Lake
- Data Pool
- Data swamp?
- Data graveyard?
- (NoSQL)

Increasingly more of a distinction between the data store and the data management software

# FROM DATA MODEL TO IMPLEMENTATION

What do we do once we've completed a (logical) data model?

- instantiate the model in your chosen software (e.g. create tables in MySQL)
- load the data
- **query** the data:
  - Traditional relational databases use Structured Query Language (SQL)
  - Other types of databases use entirely different query languages (AQL, semantic engines, etc.) or rely on bespoke computer programs (e.g. written in R, Python)

# DATA COLLECTION MANAGEMENT

Once you've created a data collection, you must also manage it! What does this mean? Fundamentally, it means maintaining data in the database that is:

- accurate,
- precise,
- consistent
- complete

Don't let your data lake turn into a data swamp...