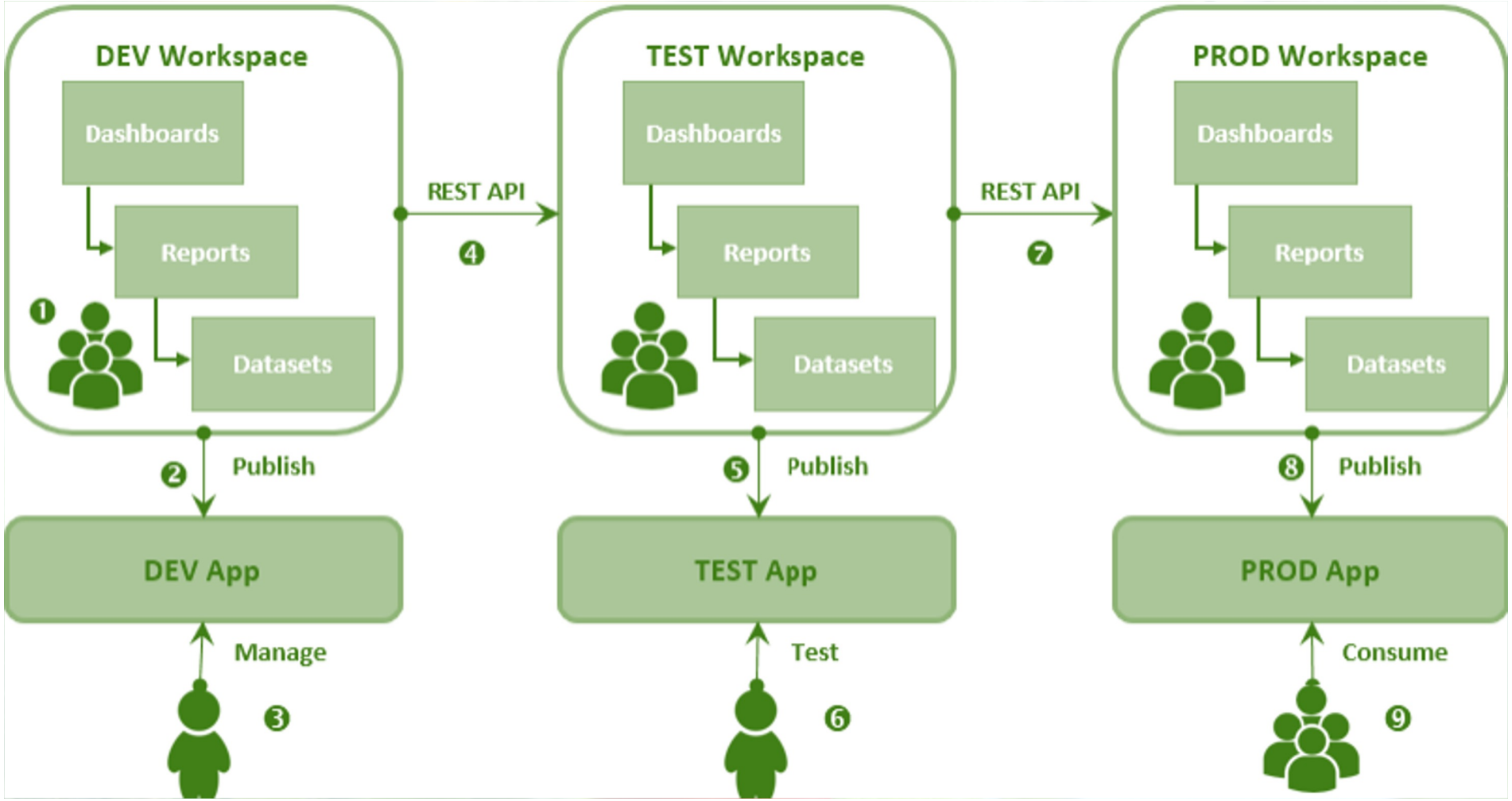



# 12. Data Management

# Fundamental Concepts

**Data** and **knowledge** must be structured so that it can be:

- stored and accessible
- added to
- usefully and efficiently extracted from that store (extract – transform – load)
- operated over by **humans** and **computers** (programs, bots, A.I.)

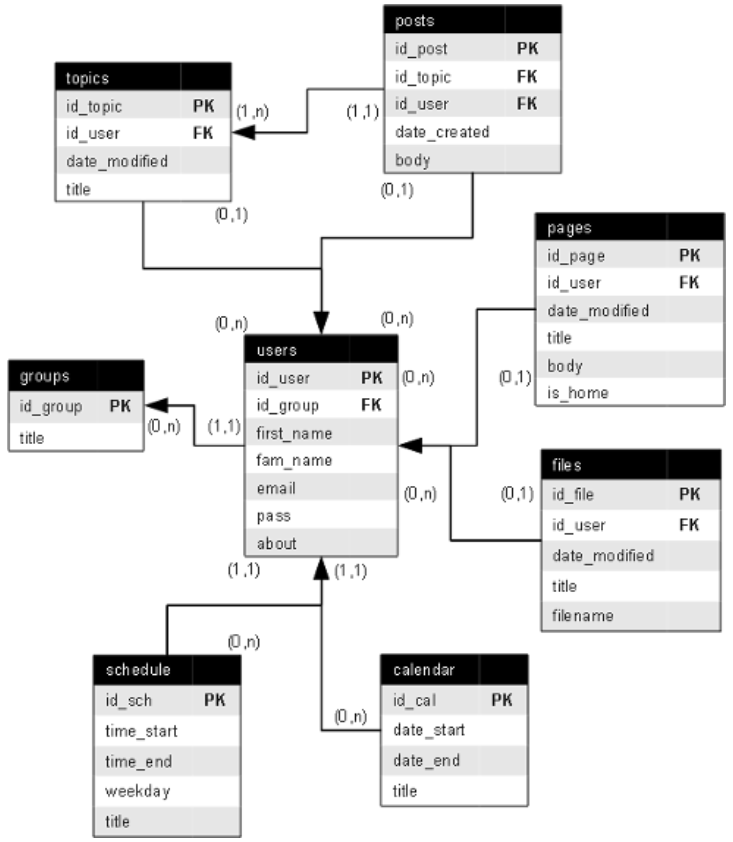
# Data Modeling

Data models are **abstract/logical** descriptions of a system, using terms that are implementable as the structure of a type of data management software.

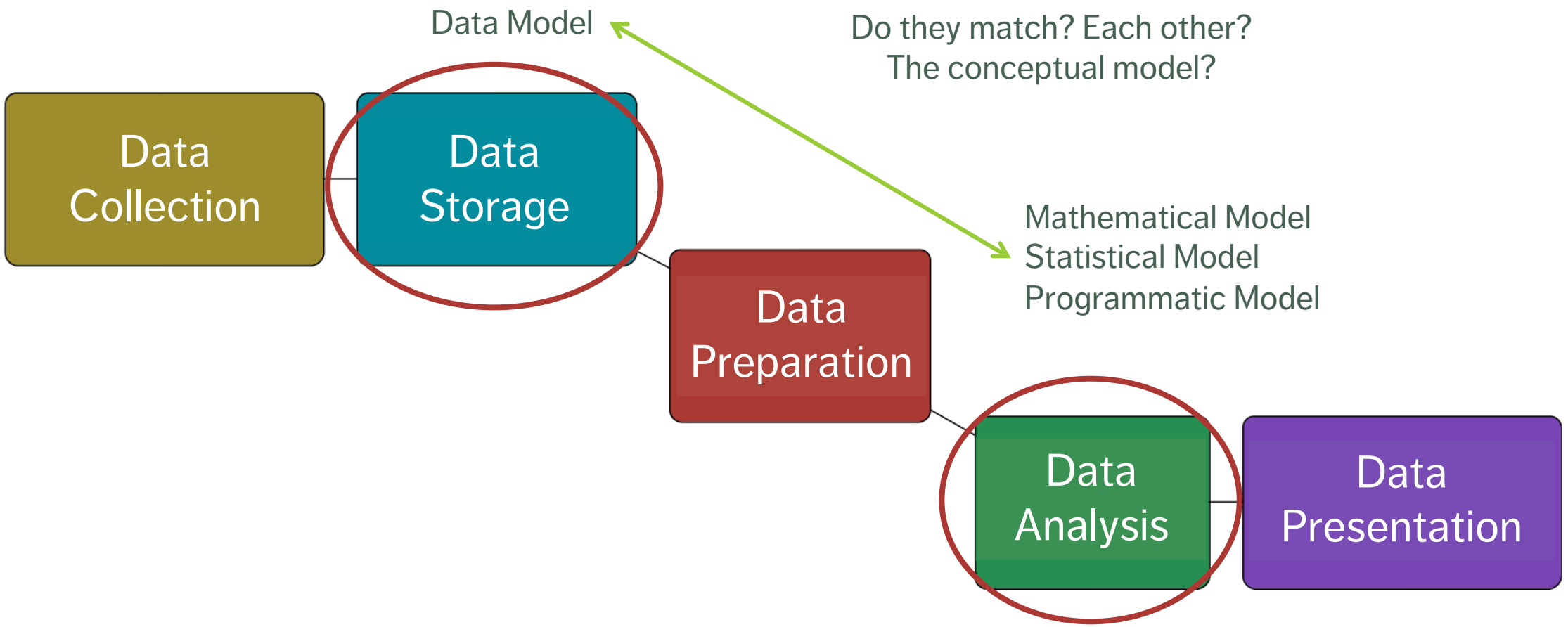
This is half-way between a **conceptual model** and a **database implementation**.

The data itself is about **instances** – the model is about the **object types**.

Another option to consider: **ontologies**.



# Automated Data Pipeline



[Representation of Langerhans cells in the *Cell Ontology*, A. M. Masci *et al.*]

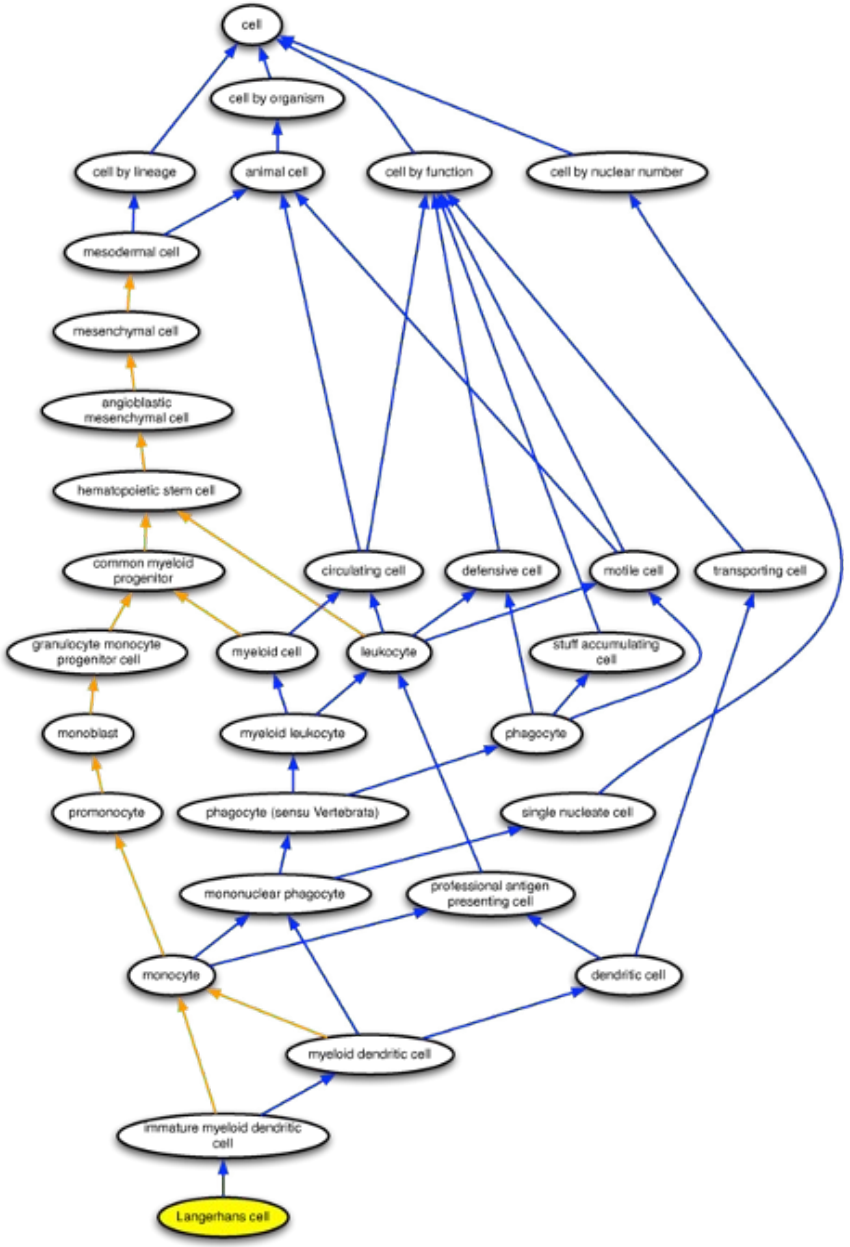
# Contextual Metadata

Something gets lost when we move from conceptual models to either a data or a knowledge model.

One way of keeping the context is to provide rich **metadata** – data **about** the data.

Metadata is crucial when it comes to carrying out strategies for working across datasets.

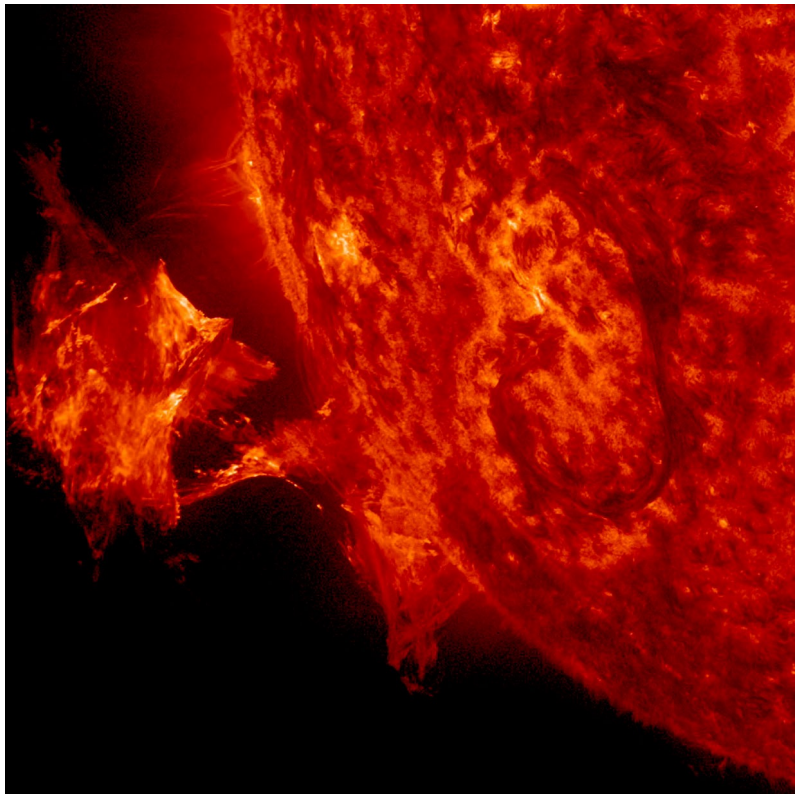
Ontologies can also play a role here.



# Structured/Unstructured Data

A major motivator for new developments in database types and other data storing strategies is the increasing availability of **unstructured** data and '**blob**' data:

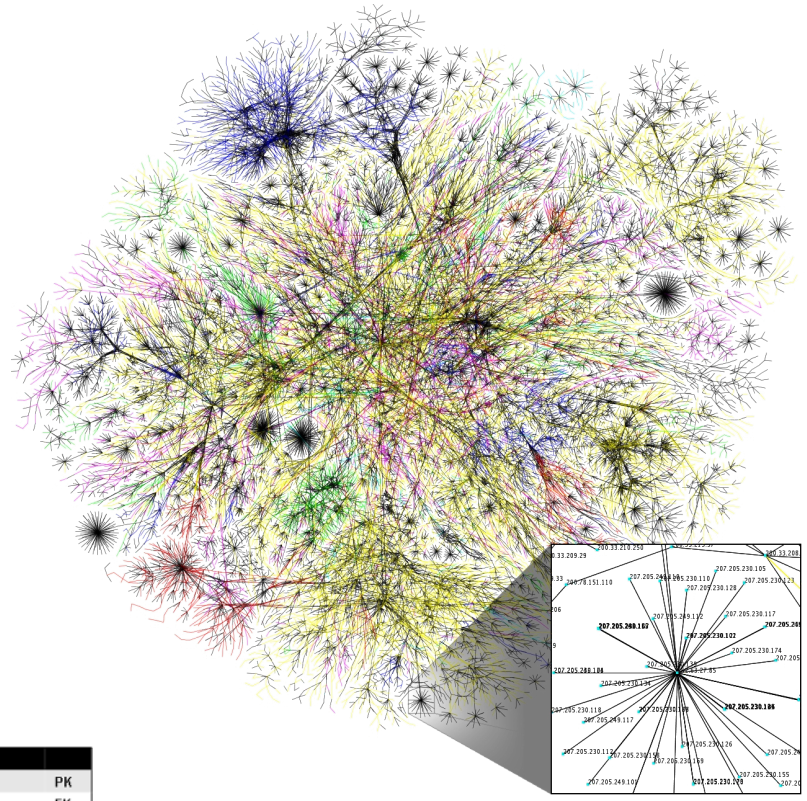
- **structured data:** labeled, organized, discrete structure is constrained and pre-defined
- **unstructured data:** not organized, no specific pre-defined structure data model (text)
- **blob data: B**inary **L**arge **Ob**ject (blob) – images, audio, multi-media

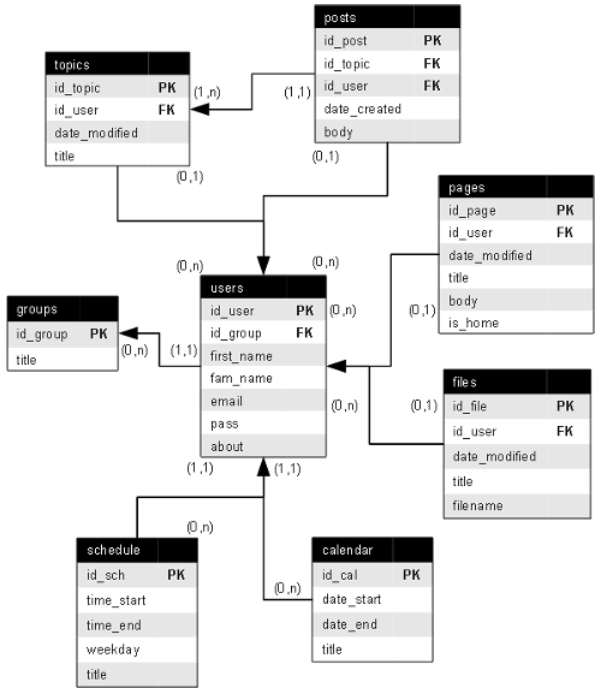


# Data Modeling

Different options are currently popular in terms of fundamental **data** and **knowledge** modeling or structuring strategies:

- key-value pairs (e.g., JSON)
- triples (e.g., RDF)
- graph databases
- relational databases
- spreadsheets





# Stores and Databases

## Relational Database:

- widely supported, well understood, works well for many types of systems and use cases, difficult to change once implemented, doesn't deal with relationships well

## Key-Value Stores:

- can take any sort of data, no need to know much about its structure in advance, missing values don’t take up space, can get messy, difficult to find specific data

## Graph Databases:

- fast and intuitive for heavily relation-based data, might be the only option in this case as traditional databases may slow to a crawl, probably overkill in other cases, not yet widely supported

# Flat Files and Spreadsheets

## Pros:

- very efficient if collecting data only once, about one particular type of object
- some types of analysis require all the data in one place
- easy to read into analysis software and do operations over the entire dataset

## Cons:

- very hard to manage data integrity if continually collecting data
- not ideal for system data involving multiples types of objects and relationships
- can be very difficult to carry out data querying operations

# Tools and Buzzwords

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

# Data Model Implementation

To implement your data/knowledge model, one needs access to **data storage and management software**.

This can be a challenge for individuals: such software usually runs on **servers**.

Servers are good because they allows multiple users to access a single database **simultaneously**, from different client programs, but it makes it difficult to "play" with the data.

This is where **SQLite** comes into play.

# Data Management Software

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

It's essentially a **human** – **data** interface.

Through this interface, users can:

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

# Names / Terminology

**Previously:**

- database
- data warehouse
- data marts
- database management system
- (SQL)

**Now:**

- data lake
- data pool
- data swamp?
- data graveyard?
- (NoSQL)

Increasingly: distinction between data **store** and data **management software**.

# From Data Model to Implementation

Once the (logical) data mode is **completed**

1. **instantiate the model** in chosen software (e.g., create tables in MySQL)
2. **load the data**
3. **query the data:**
   - traditional relational databases use **Structured Query Language** (SQL)
   - others use different query languages (AQL, semantic engines, etc.) or rely on bespoke computer programs (e.g., written in R, Python)

# Database Management

Once data has been collected, it must also be **managed**.

Fundamentally, this means that the database must be **maintained**, so that the data is

- **accurate,**
- **precise,**
- **consistent**
- **complete**

Don’t let your data lake turn into a data swamp!

# Cloud Service Provider

1. Store **large** amounts of data
2. Run expensive and advanced processes with **click of a button**
3. **Flexible** and **scalable**
4. Enable **low-code** data wrangling

# Cloud vs. On-Premise

**Cloud**

- hands-off
- pay-as-you-go model
- questionable data ownership

**On-Premise (On-Prem)**

- self-maintained
- all costs absorbed
- fully-controlled security

# Suggested Reading

Data Management

*Data Understanding, Data Analysis, Data Science*

**Data Science Basics**

Getting Insight From Data

- Structuring and Organizing Data

**Data Engineering and Management**

Data Management

- Databases
- Data Modeling
- Data Storage

Reporting and Deployment

- Reports and Products
- Cloud and On-Premise Architecture

# Exercises

Data Management

1. Does your organization have data? If so, is it hosted on-premise or on the cloud? How is it accessed? Structured?
2. Complete any of the previous exercises you have not had the chance to finish.