Miscellanea

DATA SCIENCE ESSENTIALS

11. Data Engineering

Background

One of the data science challenge: putting large troves of data into formats that can be **read** by algorithms.

Data engineering is related to processing an ever-increasing supply of data.

After processing, data scientists develop **proofs-of-concept**; AI/ML engineers translate these into **deployable models**.

Data/ML engineering have been around a while (software logs); with the rise of **cloud computing**, some argue that expertise in these fields is becoming more sought after than expertise in data analysis (at least, in some circles).

Data Roles (Reprise)

Data Engineers

- receive data from a source
- structure, distribute, and store data into data lakes and warehouses
- create tools and data models which data scientists can use to query the data

ML Engineers

- apply and deploy data models
- bridge gaps between data engineers and data scientists
- take proof-of-concept ideas to large scale

Data Scientists

- receive data procured/provided by DE
- extract value from the data
- build proof-of-concept predictive models
- measure and improve results
- build analytical models

Data Roles

In smaller organizations, data engineering and data science are typically **blended** into the same role.

Larger companies have **dedicated** data engineers on staff, who build **data pipelines** and manage **data warehouses** (populating them with data and creating table schemas to keep track of the stored data).

In general, DE ≠ DS.

Data Pipelines

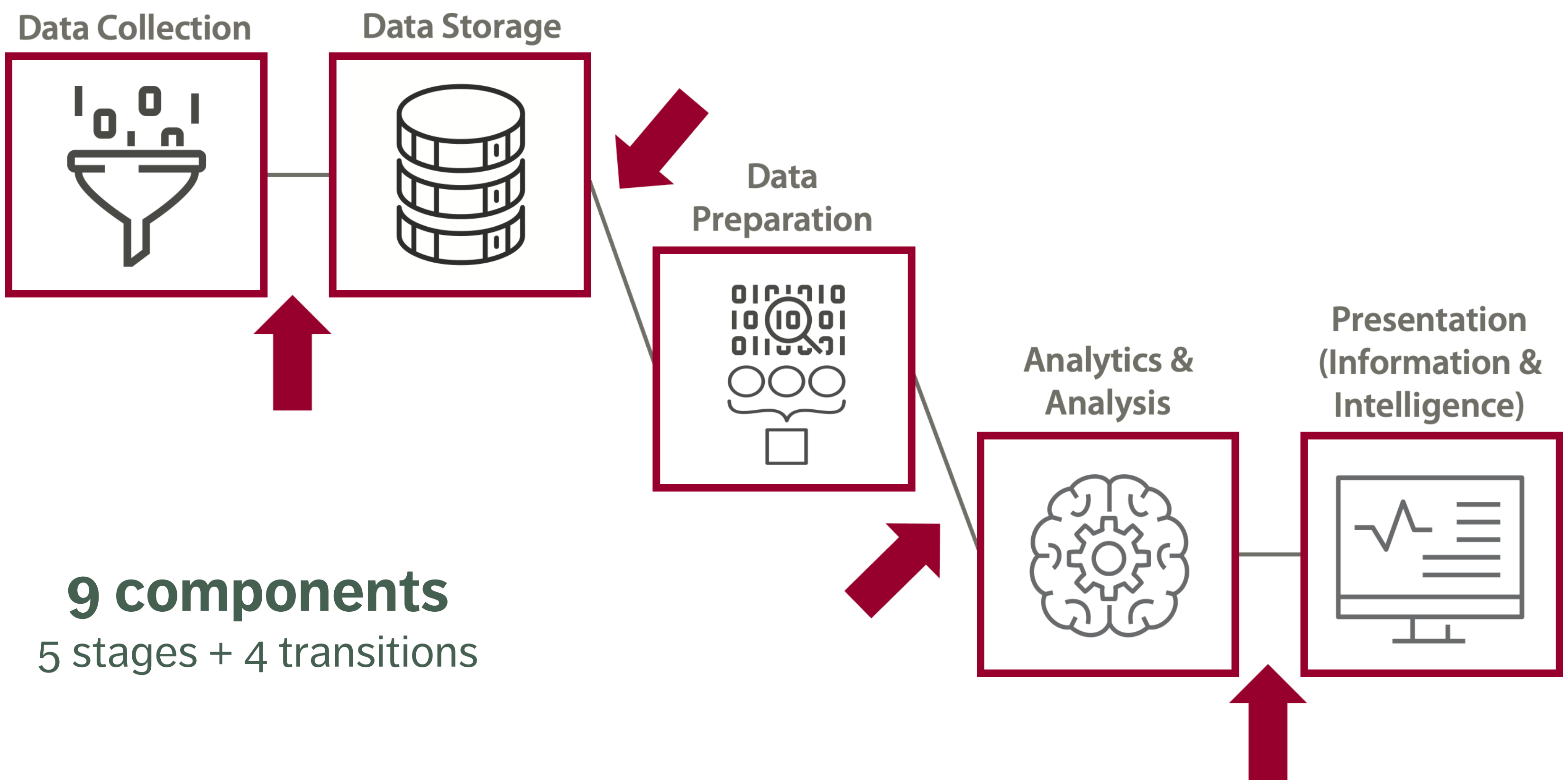
Data engineering

- operations that create **interfaces** and **mechanisms** for the flow and access of information
- setting up **data infrastructure**, preparing it for further analysis by data scientists

Data can arise from many **sources** (and types of sources), and in a variety of formats and size.

Transforming this into a process that data scientists can use and from which they can derive meaning is known as **building a data pipeline**.

Data Pipelines



9 components

5 stages + 4 transitions

Data Pipelines

Main data engineering challenge:

- building a pipeline that **runs in (close to) real-time whenever it is requested**
- so that users get **up-to-date information** from the source with **minimal delays**

Working pipeline proof-of-concept solutions are passed on to ML engineers for **deployment** and **production**. Some of the work surrounding this includes:

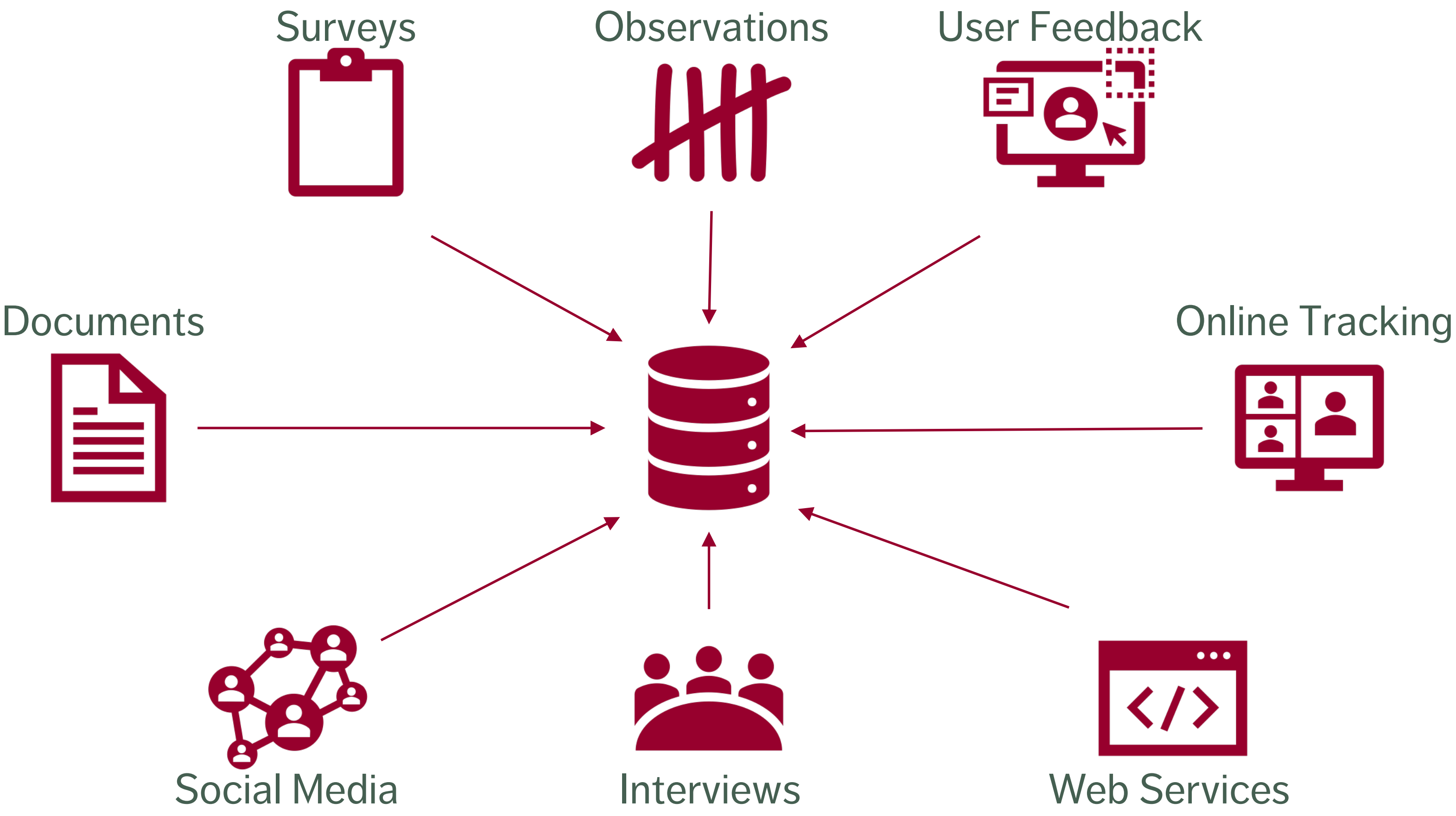
- data quality checks
- optimizing query performance
- creating a continuous integration/continuous delivery ecosystem around model changes
- ingesting data from various sources into the data model
- carrying machine learning and data science techniques to distributed systems.

Data Pipelines

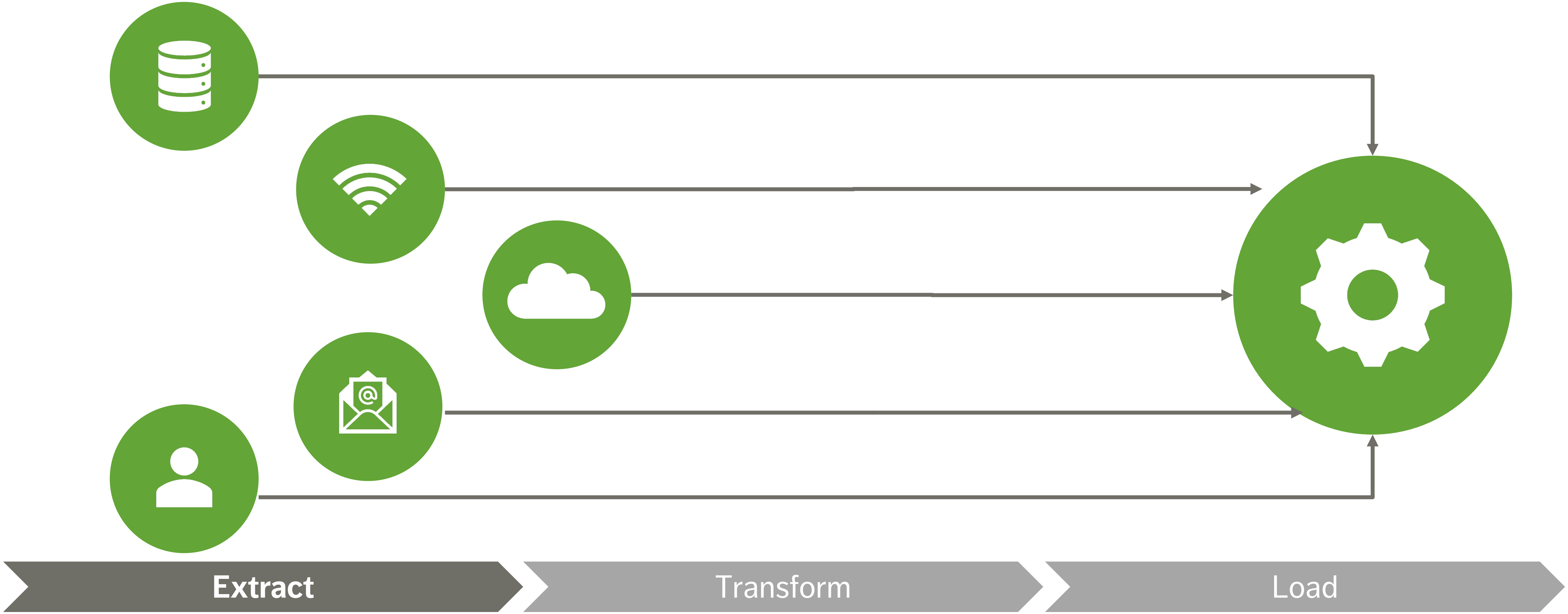
Common **themes** (operations/framework/tasks/sources) for pipeline steps :

- **data collection:** applications, mobile apps, microservices, Internet of Things (IoT) devices, websites, instrumentation, logging, sensors, external data, user generated content, etc.
- **data storage:** Master Data Management (MDM), warehouse, data lake, etc.
- **data integration/preparation:** ETL, stream data integration, etc.
- **data analysis:** machine learning, predictive analytics, A/B testing, experiments, artificial intelligence (AI), deep learning, etc.
- **delivery and presentations:** dashboards, reports, microservices, push notifications, email, SMS, etc.

Data Collection



ETL – Extract



ETL – Transform

Changing Structure

Altering Data Types

Aggregating Data

Cleaning

Joining

Grouping

Extract

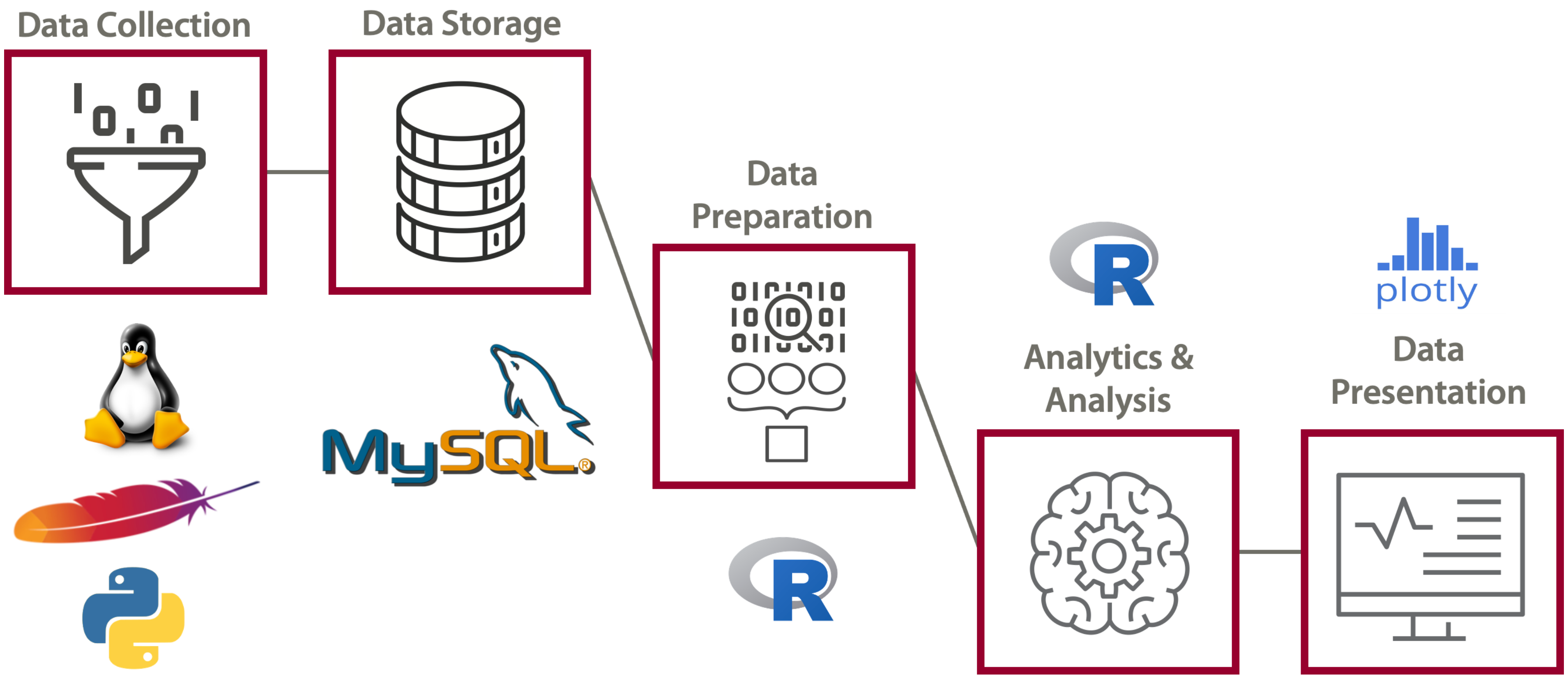
Transform

Load

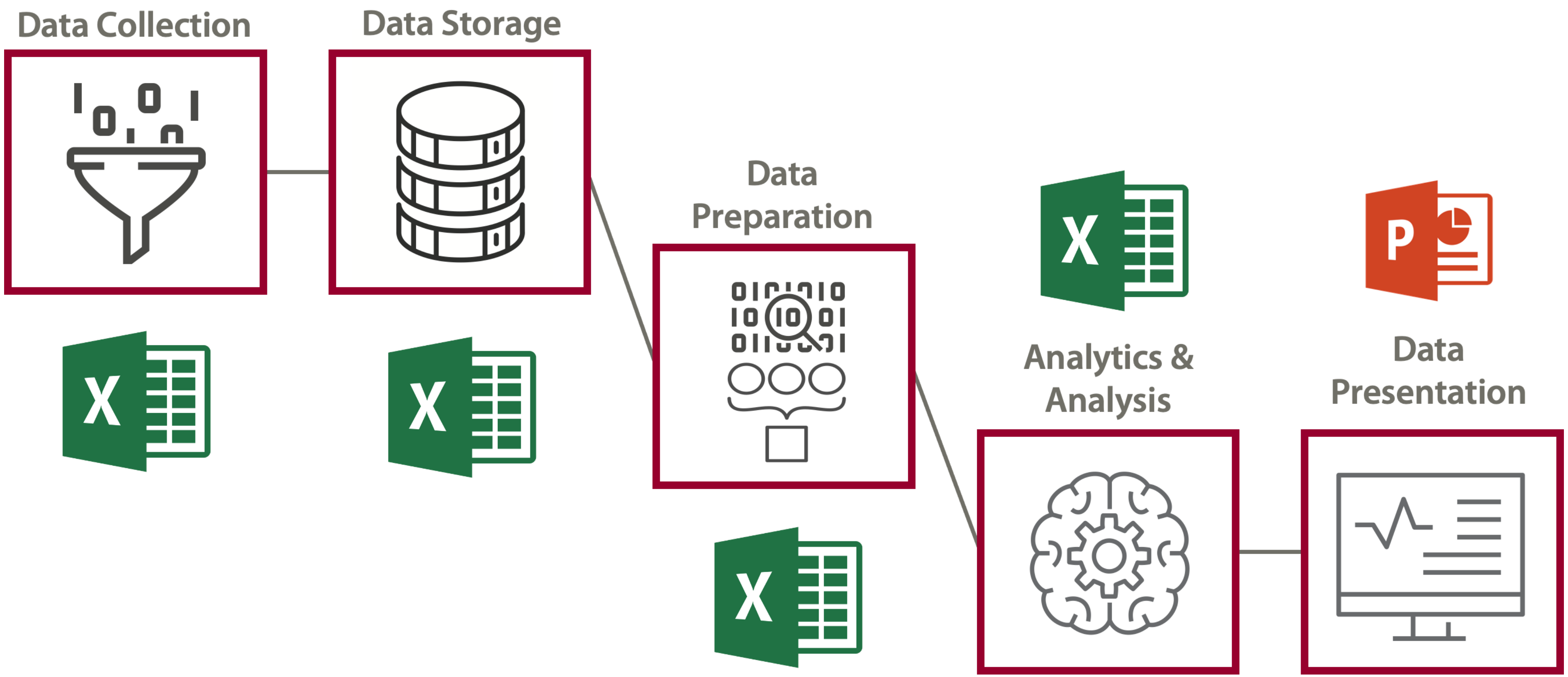
ETL – Load

Extract → Transform → **Load**

Data Pipeline: Open Source



Data Pipeline: GoC (?)



Data Pipeline Tools

Pipelines let users split large tasks into a series of smaller sequential steps, which can help **optimize** each step.

If using TensorFlow for the analysis component of a DL pipeline which consists of a single large script, then **everything** from data collection to presentation has to be done with TensorFlow; may not be optimal.

Data pipeline tools select the best framework/language for each pipeline component/task:

- Luigi (Spotify)
- Airflow (AirBnB)
- scikit-learn
- pandas/tidyverse
- etc.

Data Engineering Tools

It is unlikely that one data engineer could achieve mastery over all possible data engineering tools, but teams might get a lot of **coverage:**

- **analytical databases** (Big Query, Redshift, Synapse, etc.)
- **ETL** (Spark, Databricks, DataFlow, DataPrep, etc.)
- **scalable compute engines** (GKE, AKS, EC2, DataProc, etc.)
- **process orchestration** (AirFlow/Cloud Composer, Bat, Azure Data Factory, etc.)
- **platform deployment and scaling** (Terraform, custom tools, etc.)
- **visualization tools** (Power BI, Tableau, Google Data Studio, D3.js, ggplot2, etc.)
- **programming** (tidyverse, numpy, pandas, matplotlib, scikit-learn, scipy, Spark, Scala, Java, SQL, T-SQL, H-SQL, PL/SQL, etc.)

What is Data Governance?

Data governance encompasses:

- **people**
- **processes**
- **information technology**

It is required to create a **consistent** and **proper** handling of an organization's data across the enterprise.

It provides the foundation, strategy, and structure to ensure that data is managed as an **asset** and transformed into **meaningful** information.

Data Governance

Goals:

- create self-service data culture
- establish internal rules for data use
- implement compliance requirements
- improve internal and external comms
- increase value of data
- reduce costs
- continually manage risks
- ensure continued existence

Suggested Reading

Data Engineering

Data Understanding, Data Analysis, Data Science

Volume 2: Fundamentals of Data Insight

17. Data Engineering and Data Management

    17.1 Background and Context

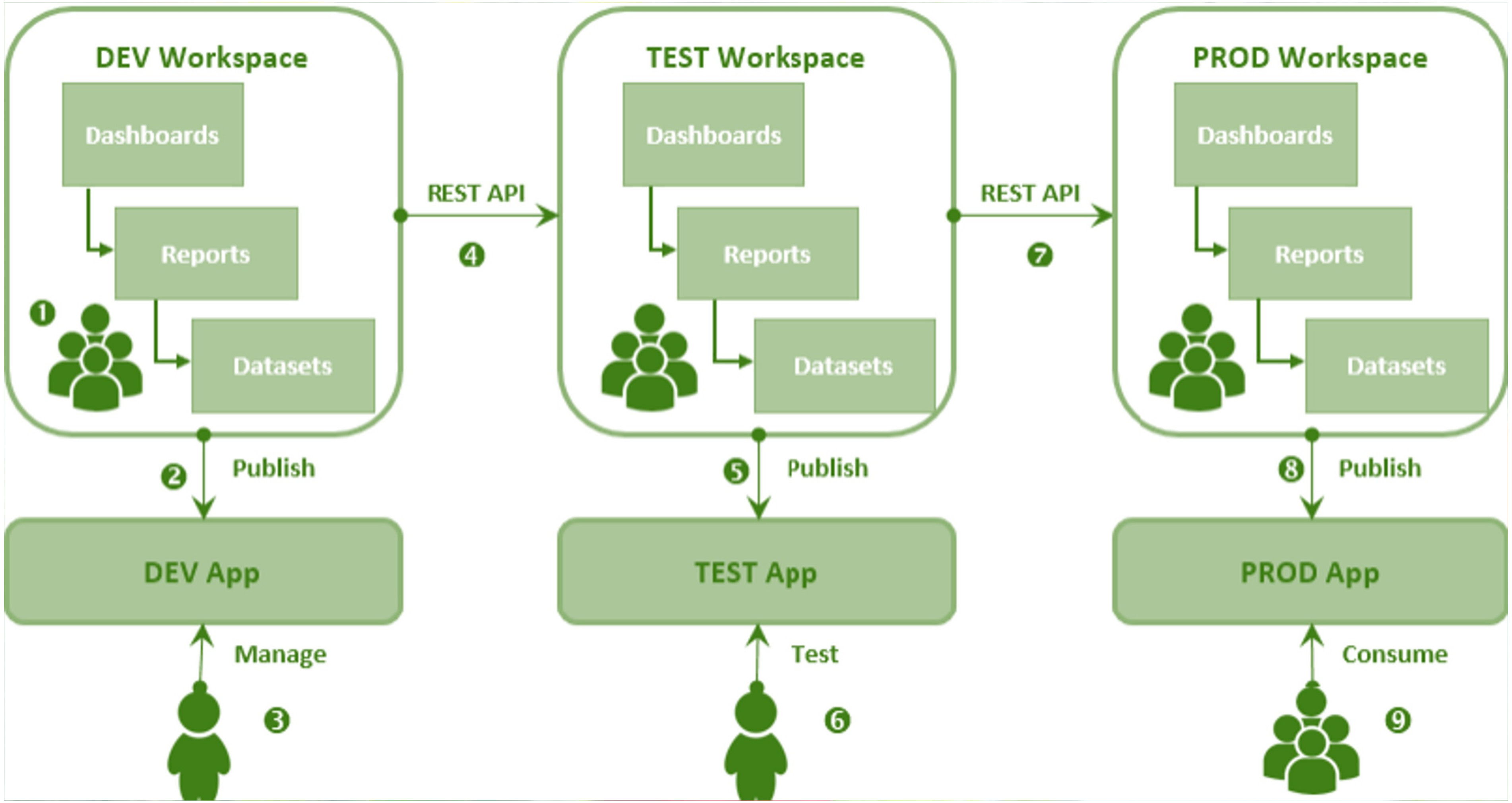
    17.2 Data Engineering

    - Data Pipelines
    - Automatic Deployment and Operations
    - Scheduled Pipelines and Workflows
    - Data Engineering Tools

Exercises

Data Engineering

1. What does your (or your organization's) data science pipeline look like? Could it be improved?
2. Identify instances where you have had issues due to data availability, usability, consistency, integrity, quality, security, or trustworthiness.
3. Complete any of the previous exercises you have not had the chance to finish.



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.)

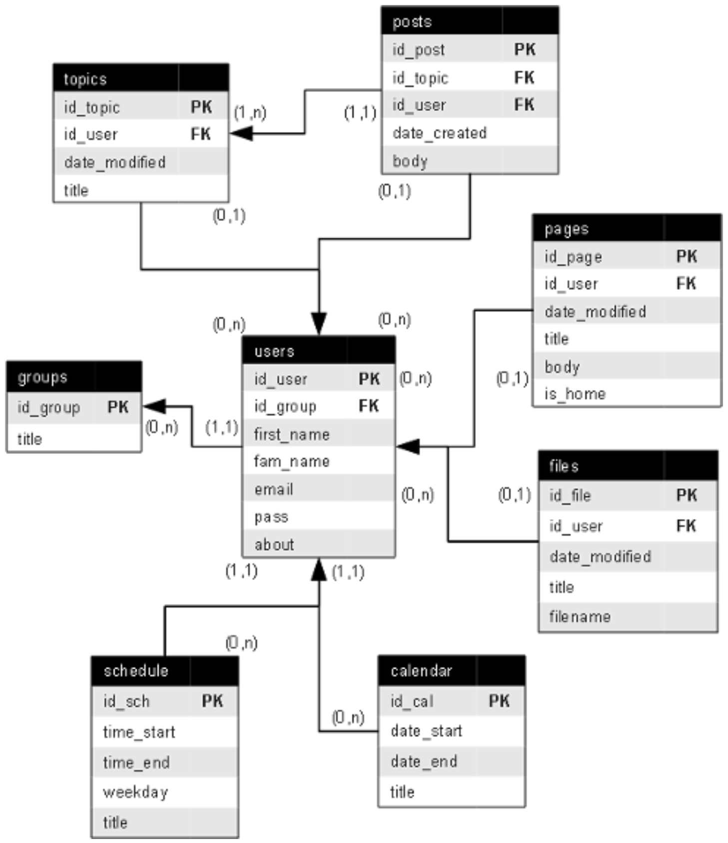
Data Modeling

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

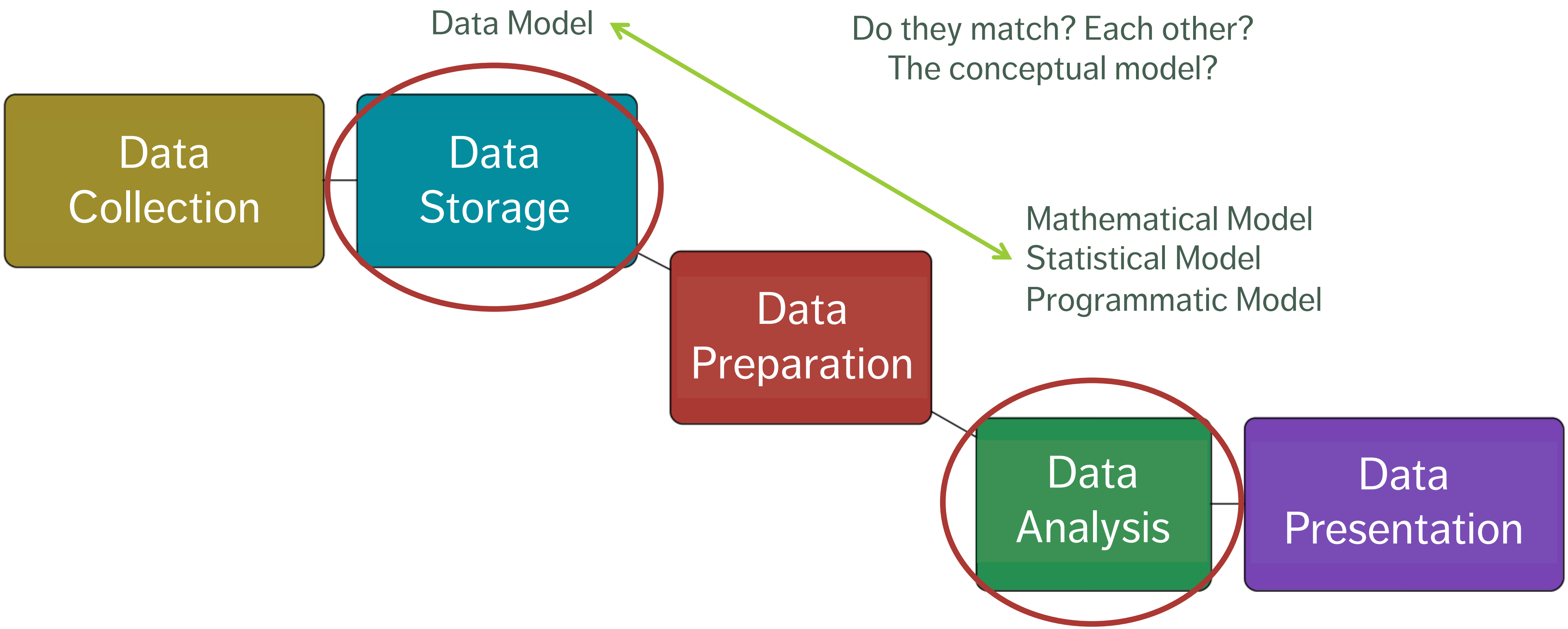
This is half-way between a **conceptual model** and a **database implementation**.

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.*]

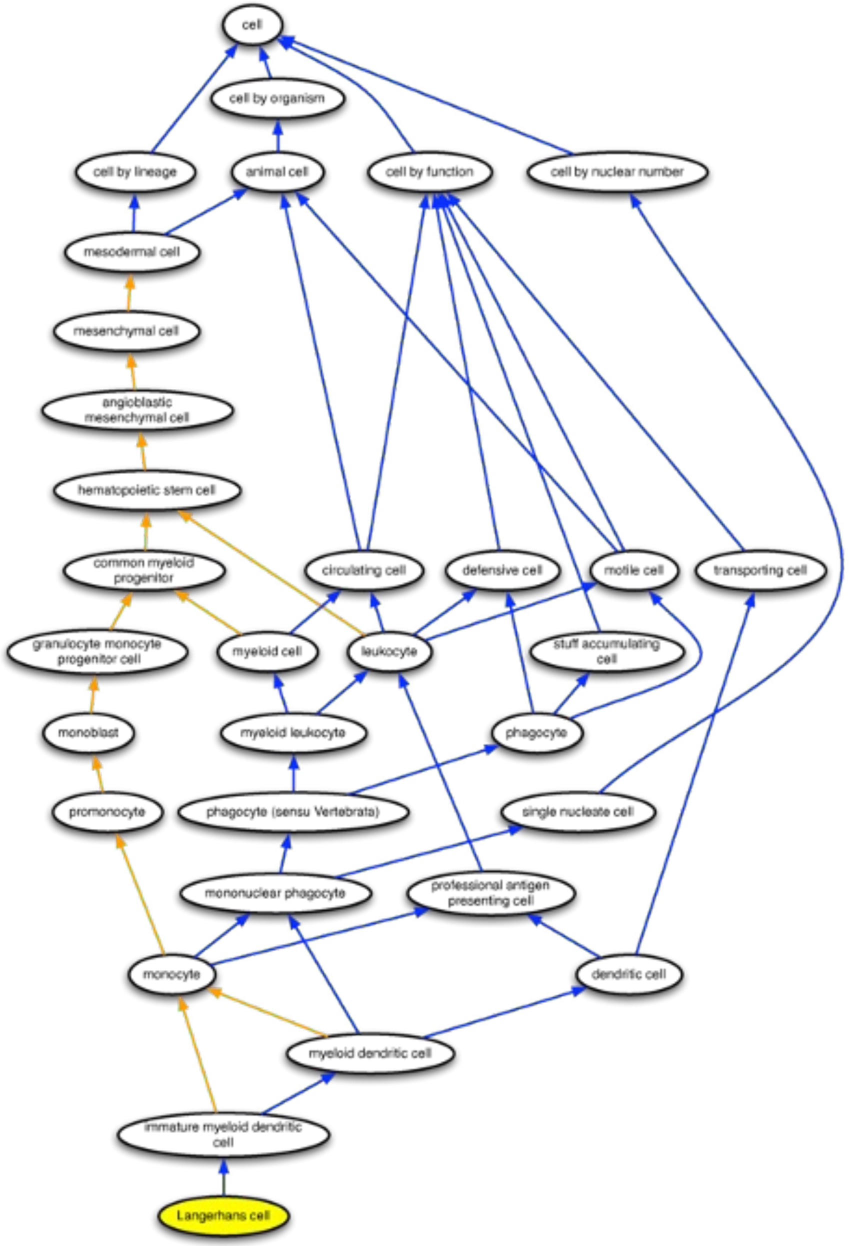
Contextual Metadata

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.

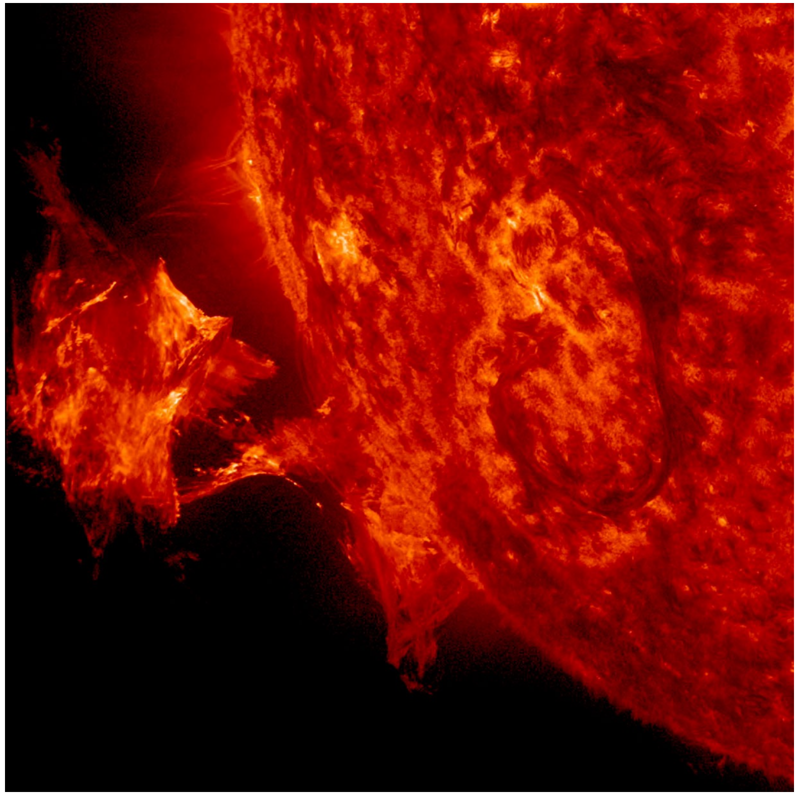
Ontologies can also play a role here.



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:

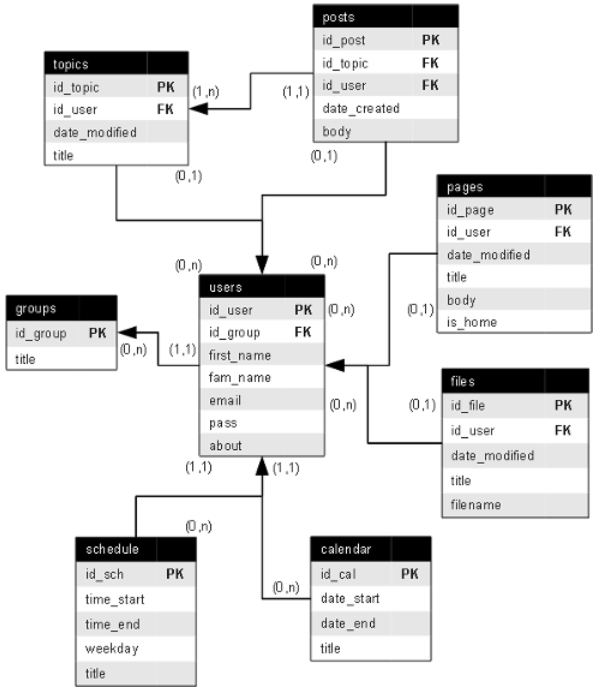
- **structured data:** labeled, organized, discrete structure is constrained and pre-defined
- **unstructured data:** not organized, no specific pre-defined structure data model (text)
- **blob data: B**inary **L**arge **Ob**ject (blob) – images, audio, multi-media

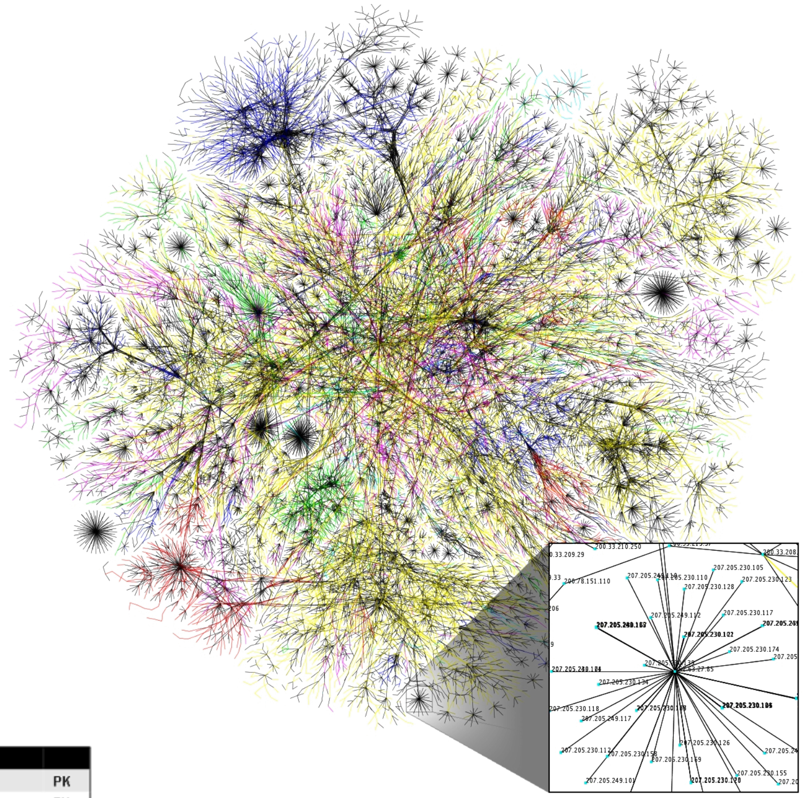


Data Modeling

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	On-Premise (On-Prem)
hands-off	self-maintained
pay-as-you-go model	all costs absorbed
questionable data ownership	fully-controlled security

Suggested Reading

Data Management

Data Understanding, Data Analysis, Data Science
Volume 2: Fundamentals of Data Insight

14. Data Science Basics

14.5 Getting Insight From Data

- Structuring and Organizing Data

17. Data Engineering and Data Management

17.3 Data Management

- Databases
- Data Modeling
- Data Storage

17.4 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.