Understanding Data Engineering

DataCamp

Content

- **What is data engineering?**
  - Data engineering and big data
  - Data engineers vs. data scientists
  - The data pipeline
- **Storing data**
  - Data structures
  - SQL databases
  - Data warehouses and data lakes
- **Moving and processing data**
  - Processing data
  - Scheduling data
  - Parallel computing
  - Cloud computing

What is data engineering?

Data Workflow:

There are four general steps through which data flows within an organization:

1. **Data collection & storage**: Collect and Ingest data, from web traffic, surveys, or media consumption for example.
2. **Preparation**: prepare it, which includes "cleaning data", for instance finding missing or duplicate values, and converting data into a more organized format.
3. **Exploration & Visualization**: explore, visualize, and build dashboards to track changes or compare two data sets.
4. **Experimentation & Prediction**: ready to run experiments, like evaluate which article title gets the most hits, or to build predictive models, for example, to forecast stock prices.

Data Engineers:

Data engineers are responsible for the first step of the process: ingesting collected data and storing it.
They are responsible for laying the groundwork for data analysts, data scientists and machine learning engineers.
If the data is scattered around, corrupted, and difficult to access, there's not much to prepare, explore, or experiment with.
And that's exactly why you need a Data engineer: their job is to **deliver the correct data, in the right form, to the right people, as efficiently as possible**.

They:

- Ingest data from different sources.
- Optimize the databases for analysis.
- Manage data corruption.

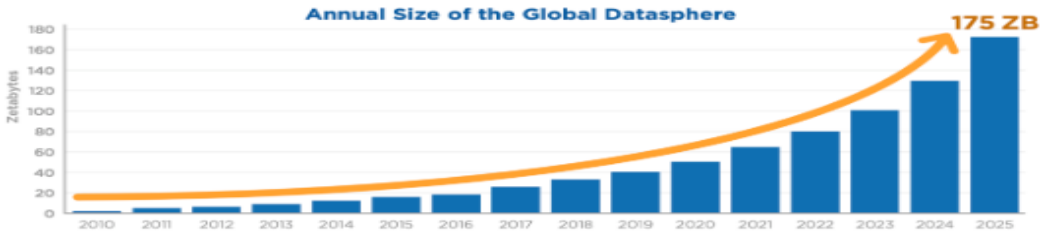
Data engineers develop, construct, test, and maintain architectures such as databases and large-scale processing systems to process and handle massive amounts of data.

Big Data & Data Engineers:

With the advent of big data, the demand for data engineers has increased.

Big data can be defined as data so large you have to think about how to deal with its size, because it's difficult to process using traditional data management methods.

This graph helps make sense of the growth of big data.



The Five Vs:

Big data is commonly characterized by five Vs:

1. **Volume**: the quantity of data points.
2. **Variety**: type and nature of the data (text, image, video, audio).
3. **Velocity**: how fast the data is generated and processed.
4. **Veracity**: how trustworthy the sources are.
5. **Value**: how actionable the data is.

Data engineers need to take all of this into consideration.

---------->>>

Data engineers vs. data scientists:

Data Engineers role is to ingest and store the data so it's easily accessible and ready to be analyzed.
Data scientist intervene on the rest of the workflow: they prepare the data according to their analysis needs, explore it, build insightful visualizations, and then run experiments or build predictive models.

Data engineers lay the groundwork that makes data science activity possible.

Data Engineer	Data Scientist
• Ingest and store data • Data engineers ensure that databases are optimized for analysis (correct table structure, information easy to retrieve) • Build Data pipeline	• Exploit this stored data • Access the databases to exploit the data it contains, For example, can analyze data without too much preparation work. • Use pipeline outputs

Based on the above, it's no surprise that data engineers are **software experts**, while data scientists are **analytics experts**. In general, **Data Engineers** uses languages like **software-oriented Python** or **Java**, and **SQL** to create, update and transform databases, while **Data scientist** uses **analytics-oriented Python** or **R**, and **SQL** to query - or, in other words, request information from - databases.

Data Pipeline:

Companies ingest data from many different sources, which needs to be processed and stored in various ways. To handle that, we need data pipelines that **efficiently automate the flow from one station to the next, so that data scientists can use up-to-date, accurate, relevant data.**

Data pipelines ensure the data flows efficiently through the organization.

They automate:

- extracting
- transforming
- combining
- validating
- loading data

To

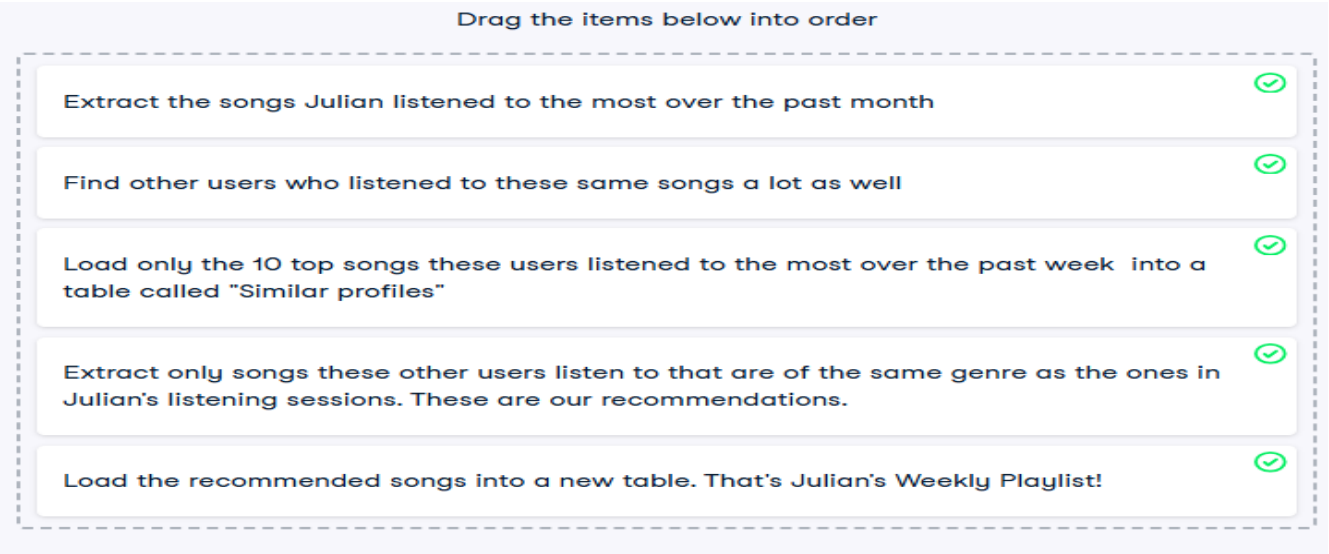
- Reduce human intervention and errors
- Decrease the time it takes for data to flow through the organization.

ETL:

It's a popular framework **for designing data pipelines**. It breaks up the flow of data into three sequential steps: first **E for extracting** the data, then **T for transforming** the data, and finally, **L for loading** this transformed data to a new database. The key here is that data is processed before it's stored.

In general, **data pipelines move data from one system to another**. They may follow ETL, but not all the time. For instance, the data may not be transformed, and routed directly to applications like visualization tools or Sales force.

Example:



Storing data

Data structure:

- **Structured data**
  - Structured data is easy to search and organize.
  - Data is entered following a rigid structure, like a spreadsheet where there are set columns.
  - Each column takes values of a certain type, like text, data, or decimal.
  - It makes it easy to form relations; hence it's organized in what is called a relational database.
  - About 20% of the data is structured.
  - SQL, which stands for Structured Query Language, is used to query such data.

- **Semi-structured**

  - Semi-structured data resembles structured data, but allows more freedom.
  - It's therefore relatively easy to organize, and pretty structured, but allows more flexibility.
  - It also has different types and can be grouped to form relations, although this is not as straightforward as with structured data - you have to pay for that flexibility at some point.
  - Semi-structured data is stored in NoSQL databases (as opposed to SQL) and usually leverages the JSON, XML or YAML file formats.

Example:

```
{
  {"user_1645156":
     "last_name": "Lacroix",
     "first_name: "Hadrien",
     "favorite_artists": ["Fools in Deed", "Gojira", "Pain", "Nanowar of Steel"]},
  {"user_5913764":
     "last_name": "Billen",
     "first_name: "Sara",
     "favorite_artists": ["Tamino", "Taylor Swift"]},
  {"user_8436791":
     "last_name": "Sulmont",
     "first_name: "Lis",
     "favorite_artists": ["Arctic Monkeys", "Rihanna", "Nina Simone"]},
  ...
}
```

Here is an example of a **JSON file** storing the favorite artists of each Spotflix user.
As you can see, the model is consistent: each user id contains the user's last and first name, and their favorite artists. **However, the number of favorite artists may differ**: I have four, Sara has two and Lis has three favorite artists. **Relational databases don't allow that kind of flexibility, but semi-structured formats let you do it.**

- **Unstructured data**

  - Unstructured data is data that does not follow a model and can't be contained in a rows and columns format.
  - This makes it difficult to search and organize.

- It's usually text, sound, pictures or videos.
- It's usually stored in data lakes, although it can also appear in data warehouses or databases.
- Most of the data around us is unstructured.
- Unstructured data can be extremely valuable, but because it's hard to search and organize, this value could not be extracted until recently, with the advent of machine learning and artificial intelligence.

SQL databases:

- **SQL :**
  - SQL stands for Structured Query Language.
  - It's the preferred language to query RDBMS or Relational Database Management System - basically systems that gather several tables, where all tables are related to each other.
  - SQL has two main advantages:
    - It allows you to access many records at once, and group, filter or aggregate them.
    - It's also very close to English, which makes it easy to write and understand.
  - Data engineers use SQL to create and maintain databases, while data scientists use SQL to query databases.

Example on SQL for Data Engineer:

```
CREATE TABLE employees (
    employee_id INT,
    first_name VARCHAR(255),
    last_name VARCHAR(255),
    role VARCHAR(255),
    team VARCHAR(255),
    full_time BOOLEAN,
    office VARCHAR(255)
);
```

Example on SQL for Data Scientist:

```
SELECT first_name, last_name

FROM employees

WHERE role LIKE '%Data%'
```

Database schema:

- Databases are made of many tables.
- The database schema governs how tables are related.

Several implementations :

There are several implementations of SQL. They differ, but they are pretty similar. Switching from one to the other is like switching from a QWERTY keyboard to an AZERTY one, or switching from British English to American English. **A few things change, but most things stay the same**.

Data lakes and data warehouses:

- The data lake is where all the collected raw data gets stored, just as it was uploaded from the different sources. It's unprocessed and messy.
- While **the data lake stores all the data**, the **data warehouse stores specific data** for a specific use.
- A **data lake** can store any kind of data, whether it's **structured, semi-structured or unstructured**.
- This means that it does not enforce any model on the way to store the data. This makes it **cost-effective**.
- **Data warehouses** enforce a **structured** format, which makes them **more costly** to manipulate. However, this **lack** of structure also means it's **very difficult to analyze**.
- Some big data analytics using deep learning can be implemented to discover hidden patterns and trends, but that's about it, and should probably be last resort.
- The data warehouse, on the other hand, is optimized for analytics to drive business decisions.
- Because no model is enforced in data lakes and any structure can be stored, it is necessary to keep a data catalog up to date.
- Data lakes are used by data scientists for real-time analytics on big data, while data warehouses are used by analysts for ad-hoc, read-only queries like aggregation and summarization.

Data Catalog:

- A data catalog is a source of truth that compensates for the lack of structure in a data lake.
- Among other things, it keeps track of where the data comes from, how it is used, who is responsible for maintaining it, and how often it gets updated.
- It's good practice in terms of data governance (managing the availability, usability, integrity and security of the data), and guarantees the reproducibility of the processes in case anything unexpected happens. Or if someone wants to reproduce an analysis from the very beginning, starting with the ingestion of the data.
- Because of the very flexible way data lakes store data, a data catalog is necessary to prevent the data lake becoming a data swamp.
- It's good practice to have a data catalog referencing any data that moves through your organization, so that we don't have to rely on tribal knowledge, which makes us autonomous, and makes working with the data more scalable.
- We can go from finding data to preparing it without having to rely on a human source of information every time we have a question.

Data catalog helps ensure that everyone in the organization can access and understand the data they need, without having to rely on specific individuals. This makes the organization more autonomous and scalable, as it can grow and adapt more easily.

Database:

Database is a very general term that can be loosely defined as organized data stored and accessed on a computer. It's a general term and a data warehouse is a type of database.

Moving and processing data

Processing data:

In a nutshell, data processing consists in converting raw data into meaningful information.

Precisely, **why do we need to process data?**

- Well, there may be some data that we don't need at all (**Removing unnecessary data**).
- Storing and processing data is not free, so we want to **optimize our memory, process and network costs**.
- Some data may come in a type, but would be easier to use in another (**Convert data from one type to other**).
- We want to move and **organize data so it is easier for analysts**.
- You may want your data to **fit a certain schema or structure**.
- Data processing also **increases productivity**. At Organizations, we automate all the data preparation steps we can, so that when it arrives to data scientists, they can analyze it almost immediately.

How data engineers process data?

In terms of data processing, data engineers have different responsibilities.

- They perform **data manipulation, cleaning, and tidying tasks** that can be automated, and that will always need to be done, regardless of the analysis anyone wants to do with them.
- They also ensure that the **data is stored in a sanely structured database**, and **create views** on top of the database tables for easy access by analysts.
  - **Views** are **the output of a stored query on the data**. For example, artist data and album data should be stored in separate tables in the database, but people will often want to work on these things together. That means data engineers need to create a view in the database combining both tables.
- Data engineers also **optimize the performance of databases**, for example by indexing the data so it's easier to retrieve.

Data Processing Tools:

Scheduling Data:

Scheduling is the **glue of a data engineering system**.
It **holds** each small piece and **organizes** how they work together, by **running tasks in a specific order and resolving all dependencies correctly**.

There are different ways to glue things together. For example, we can run tasks **manually**. If an employee is moving from the United States to Belgium, and therefore changing offices, someone can request an immediate update and we can update the table right away ourselves.
However, there are **downsides with human dependencies**. Ideally, we'd like our pipeline to be **automated** as much as possible.

Automation **is when you set some tasks to execute at a specific time or condition (sensor scheduling)**.

Manual and **automated** systems can also work together: if a user manually upgrades their subscription tier on the app, automated tasks need to propagate this information to other parts of the system, to unlock new features and update billing information.

Batches and streams:

How the data is ingested?

- Data can be ingested in **batches,** which means it's sent by groups at specific intervals.
  Batch processing is often cheaper because you can schedule it when resources aren't being used elsewhere, typically overnight.

- The data can also be **streamed**, which means individual data records are sent through the pipeline as soon as they are updated.

- An example of **batch vs. stream** processing would be **offline Vs online** listening. If a user listens online, Spotflix can stream parts of the song one after the other. If the user wants to save the song to listen offline, we need to batch all parts of the song together so they can save it.

- There's a third option called **real-time**, used for example in **fraud detection**, but for the sake of simplification and because streaming is almost always real-time, we will consider them to be the same for now.

Tools for Scheduling: Apache Airflow or luigi

Parallel computing:

- Parallel computing forms the basis of almost all modern data processing tools.
- It is important mainly for memory concerns, but also for processing power.
- When big data processing tools perform a processing task, they split it up into several smaller subtasks. These subtasks are then distributed over several computers.

Benefits and risks of parallel computing

Advantages:

- One benefit of having multiple processing units is the **extra processing power itself**.
- Another benefit of parallel computing for big data relates to **memory**. Instead of needing to load all of the data in one computer's memory, you can **partition the data and load the subsets into memory of different computers**. That means the memory footprint per computer is relatively small.

Disadvantages:

- Moving data **incurs a cost**.
- Splitting a task into subtasks and merging the results of the subtasks back into one final result requires some communication between processes, which takes some additional time.

So if the **gains** of splitting into subtasks are **minimal**, it **may not be worth taking that risk**.

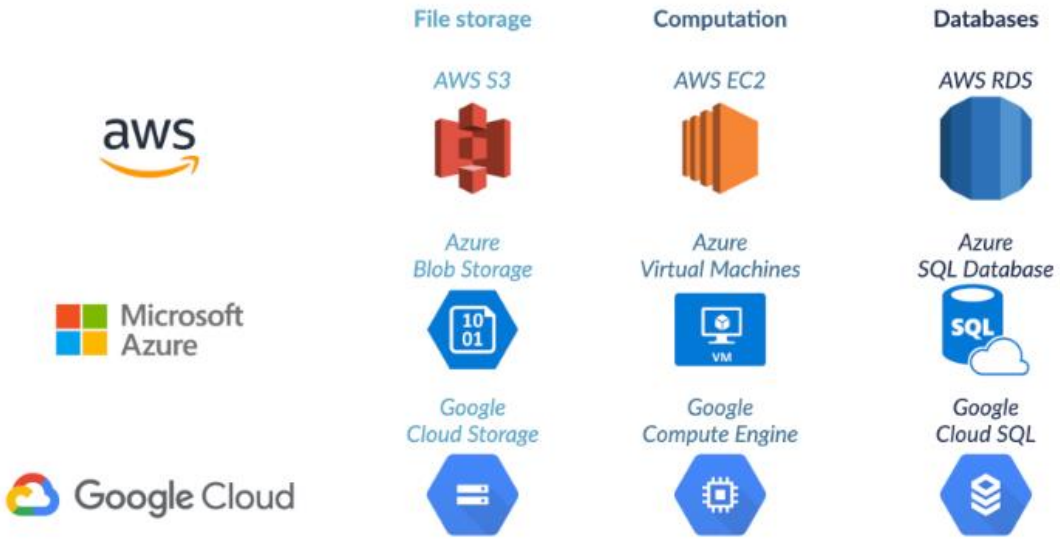
Cloud computing for data processing:

- Companies can process data in their own data center, often on premises. We can imagine racks of servers, ready to be used, that the company has to buy. We also need a room to store them, and if we move offices, we have to transport servers without losing service. The electrical bill and maintenance would be at the company's cost. Moreover, data processing tasks can be more or less intense, and don't happen continuously. Companies would need to provide enough processing power for peak moments, and at quieter times, much of the processing power would remain unused. It's avoiding this waste of resources **that makes cloud computing so appealing**.

- In the cloud,

    - We rent servers, and the rent is cheap.
    - We don't need a room to store them, and we use the resources we need, at the time we need them.
    - Many companies moved to the cloud as a way of cost optimization.
    - Also, the closer the server is to the user, the less latency they will experience when using our application.
    - To serve a global customer base, we need servers all over the world.

The three big players, in decreasing order of market share, are Amazon Web Services (AWS), Microsoft Azure, and Google Cloud.



Multicloud:

- You don't need to take all your cloud services from the same provider though. You can use several ones: it's called **multicloud**.
- It has some advantages, like reducing reliance a single vendor.
- It also optimizes costs, and might be necessary because of local laws.
- It also allows you to militate against disasters.

Thanks!