Chapter 1
What is Data Engineering?

Data engineering and big data

Data Workflow

- **Data Collection and Storage:** We collect and ingest data from web traffic, surveys, or media consumption. Data is stored in raw format.
- **Data Preparation:** Cleaning data, for instance finding missing or duplicate values, and converting data into a more organized format.
- **Exploration and Visualization:** Explore it, visualize it, build dashboards to track changes, or compare two sets of data.
- **Experimentation and Prediction:** Build predictive models, for example, to forecast stock prices.

Data Engineers are responsible for the first step of the process: ingesting collected data and storing it.

Data Engineer's responsibilities

- Ingest data from different sources
- Optimize databases for analysis
- Remove corrupted data
- Develop, construct, test, and maintain data architectures

What is Big Data?

Big Data can be considered 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.

Mainly contains the forms of data mentioned below:

- Sensors and devices
- Social Media
- Enterprise data
- VoIP (voice communication, multimedia sessions)

Big data is commonly characterized by the five V's.

The five V's of big data

- Volume (how much?)
- Variety (what kind?)
- Velocity (how frequent?)

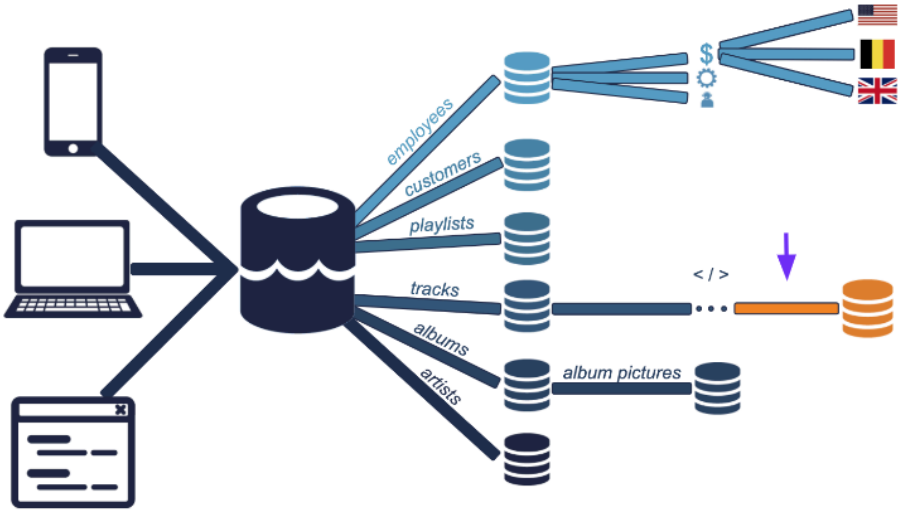
- Veracity (how accurate?)
- Value (how useful?)

Data Pipeline

So to ingest, process, and store data, we need pipelines to automate the flow from one station to the next so that it provides up-to-date, accurate, and relevant data for data scientists to work on.

Let's take the example of Spotify. It has multiple sources from which it extracts data.

- Mobile
- Website
- Desktop App
- Websites Spotify may use internally like their HR Management system for payrolls and compensations etc.



Steps to create a pipeline:

- Data extracted from different places is stored in a Data Lake.
- Different databases are created from the Data Lake.
- Further tables according to the need are created in the databases.
- A database can be cleaned and further stored in a cleaner database as well.

Data pipelines ensure an efficient flow of the data in the organization.

Automate

- Extracting
- Transforming
- Combining
- Validating
- Loading

Reduce

- Human intervention
- Errors
- The time it takes data to flow

ETL and data pipelines

A popular framework for designing data pipelines

- Extract data
- Transform extracted data
- Load transformed data to another database

Data pipelines

- Move data from one system to another
- May follow ETL
- Data may sometimes not be transformed and loaded directly to visualization tools or salesforce

Chapter 2
Storing Data

Data Structures

Structured Data

- Easy to search and organize
- Consistent model, rows, and columns
- Defined types like (text, data, or decimal)
- Can be grouped to form relations
- Stored in relational databases
- About 20% of the data is structured
- Created and queried using SQL

Table 1: `customer_info`

customer_id	name	age	gender	city
1	John	35	Male	New York
2	Emily	28	Female	Los Angeles
3	Michael	40	Male	Chicago
4	Sophia	33	Female	Houston

Table 2: `purchase_history`

transaction_id	customer_id	product_id	purchase_date	amount
101	1	201	2023-01-15	50.00
102	2	202	2023-02-20	75.00
103	1	203	2023-03-10	30.00
104	3	204	2023-04-05	100.00

Because the table above is structured we can easily relate this table to other structured data. For example, these two tables contain information about customers' demographics and their purchase history. We can join them together when need to extract data using the transaction_id.

Semi-structured data

- Relatively easy to search and organize
- Consistent model, less-rigid implementation: different observations have different sizes
- Different types
- Can be grouped, but need more work
- NoSQL databases: JSON, XML, YAML

Unstructured Data

- Does not follow a model, can’t be contained in rows and columns
- Difficult to search and organize
- Usually text, sound, pictures, or videos
- Usually stored in data lakes, can appear in data warehouses or databases
- Most of the data is unstructured
- Can be extremely valuable

SQL databases

SQL

- Structured Query Language
- Industry standard for Relational Database Management Systems (RDBMS)
- Allows you to access many records at once, and group, filter or aggregate them

- Close to written English, easy to write and understand
- Data engineers use SQL to create and maintain databases
- Data Scientists use SQL to query (request information from) databases

Database schema

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

Several implementations

- SQLite
- MySQL
- PostgreSQL
- Oracle SQL
- SQL Server

Data warehouses and data lakes

Data lake

- Stores all the raw data
- Can be petabytes (1 million GBs)
- Stores all data structures
- Cost-effective (since it does not enforce any model to store data)
- Difficult to analyze (maybe some deep learning models and that's it)
- Requires to up-to-date catalog
- Used by data scientists
- Big data, real-time analytics

Data warehouse

- Specific data for specific use (eg: user and subscription type)
- Relatively small
- Stores mainly structured data
- More costly to update
- Optimized for data analysis
- Also used by data analysts and business analysts
- Ad-hoc, read-only queries

Data catalog for data lakes

- What is the source of this data?
- Where is this data used?
- Who is the owner of the data?
- How often is this data updated?
- Good practice in terms of data governance
- Ensures reproducibility (in case anything happens or someone wants to reproduce an analysis from the very beginning)
- No catalog → data swamp

Database vs. data warehouse

- Database:
  - General term
  - Loosely defined as organized data stored and accessed on a computer
- A data warehouse is a type of database

Chapter 3
Moving and processing data

Processing data

When Spotify extracts data from different sources like mobile applications, desktop applications, or websites and stores it in a data lake, when it creates a database from data lakes, when it creates tables from those databases or removes corrupted files and creates a cleaner database, it is all a part of processing data.

Data processing: converting **raw** data into **meaningful** information.

Why do we process data?

- Remove unwanted data (No long-term need for testing feature data)
- Optimize memory, process, and network costs (can't afford to store and steam files this big)
- Convert data from one type to another (at Spotify let's say artists upload data in .wav or .flac format which is high quality and incurs higher network costs so they are converted to .ogg, a lighter format with slightly lower sound quality)
- Organize data (reorganize data from the data lake to data warehouses)
- To fit into a schema/structure (employee table example)
- Increases productivity

How do data engineers process data?

- Data manipulation, cleaning, and tidying tasks
  - That can be automated
  - That will always need to be done (like rejecting corrupt song files and deciding what happens with missing metadata like what to do when the genre of the song is missing)
- Store data in a sanely structured database
- Create views on top of the database tables
  - views are the output of the stored query on the data

    - For example, artists and albums tables are in separate tables in a database but people often work on them together so data engineers create a view for easy access to the combined result
- Optimizing the performance of the database (indexing)

Apache Spark is a popular tool for this.

Scheduling

- This can apply to any task listed in the data processing
- Scheduling is the glue of your system
- Holds each piece and organizes how they work together
- Runs tasks in a specific order and resolves all dependencies

Manual, time, and sensor scheduling

- **Manually** (for example if an employee is moving from Belgium to the USA we can raise a request to change the location manually). There are some downsides to it as we likely want the pipeline to be automated.
- **Time** (automatically runs at a specific time like updating the employee table at 6 AM so let's say if an employee was added yesterday the table will get updated at 6 in the morning)
- **Sensor** (task is executed when the condition is met. Let's say we update the employee table only when a new employee is added but this requires sensors always listening to see if something is added and requires more resources which might not be worth it.)

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

Data can be ingested in:

- Batches
    - Group records at intervals
    - Often cheaper
    - For example, songs uploaded by artists may be batched and sent together to the databases every 10 minutes or the employee table can be batched together and updated every morning at 6 AM and the revenue table used by the finance department can be updated overnight as well.
- Streams
    - Send individual records through the pipeline right away
    - For example, if a user subscribes to a service, they want to be able to use it right away like Netflix or Spotify

    - Another example is online vs offline listening. If a user listens to the song online we 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 for them to save it.
- Real time: There is another called real time which is commonly used in fraud detection but since streaming is usually real time we would consider them as same here.

Apache Airflow is a popular tool for this.

Parallel computing / Parallel processing

- Basis of modern data processing tools
- Necessary:
    - Mainly because of memory concerns
    - Also for processing power
- How it works:
    - Split tasks up into several smaller subtasks
    - Distribute these subtasks over several computers

Example:

Let's say you're running a music merchandise shop and need to get a batch of 1000 t-shirts folded.

Your senior sales assistant folds 100 shirts in 15 minutes. Junior sales assistants typically take 30 minutes.

If just one sales assistant can work at a time, it's obvious you'd have to choose the quickest one to finish the job.

However, if you can split the batch into 250 shirts each, having 4 junior employees working in parallel is faster. They will finish in 1 hour and 15 minutes.

Benefits and risks of parallel computing

- Employees = processing units
- Advantages
    - Extra processing power
    - Reduced memory footprint
- Disadvantages
    - Moving data incurs a cost
    - Communication time (the time it takes to split the tasks and then merge the results back again is time-consuming and would require much communication which might not be good if the benefits of parallel computing are minimal for that task)

In Spotify, the company might use parallel computing to convert songs from lossless format to .ogg. It prevents us from having to load all the new songs in one computer, and to benefit from extra processing power to run the conversion scripts.

Cloud Computing

Cloud computing for data processing

Servers on premises

- Bought
- Need space
- Electrical and maintenance cost
- Enough Power for peak moments
- Processing power unused at quieter times

Servers on the cloud

- Rented
- Don’t need space
- Use just the resources we need
- When we need them
- The closer to the user the better

Cloud computing for data storage

- Database reliability: data replication (In case there is fire we can replicate data in another geographical location)
- The risk with sensitive data (if your company manipulates sensitive or confidential data, there is a risk associated with someone else hosting it, and government surveillance)

Cloud providers

With the advent of big data, companies specializing in these kinds of issues, called cloud providers, were born.

The three big players, in decreasing order of market share, are:

1. Amazon Web Services
2. Microsoft Azure
3. Google Cloud

For file storage:

- AWS has **AWS S3**
- Microsoft Azure has **Azure Blob Storage**
- Google Cloud has **Google Cloud Storage**

For computation:

- AWS has **AWS EC2**
- Microsoft Azure has an **Azure Virtual Machine**
- Google Cloud has a Google Compute Engine

For Databases:

- AWS has **AWS RDS**
- Microsoft Azure has an **Azure SQL Database**
- Google has a **Google Cloud SQL**

Let’s say Spotify uses AWS so, they would use:

- AWS S3 to store cover albums
- AWS EC2 to process songs
- AWS RDS to store employee information

Multicloud

It is not necessary to use all three services from a single cloud provider.

Pros:

- Reducing reliance on a single vendor
- Cost-efficiencies
- Local laws requiring certain data to be physically present within the country
- Mitigating against disasters

Cons:

- Cloud providers try to lock in consumers
- Incompatibility
- Managing security and governance is harder