**Data Engineering Ecosystem**

**Learning Objectives**

* Describe the elements of a data engineering ecosystem which includes data, data repositories, data integration platforms, data pipelines, languages, and BI and Reporting tools.
* Differentiate between the three structures of data: structured, semi-structured, and unstructured.
* Compare and contrast standard file formats for Data Engineering.
* Describe common sources of data including relational databases; flat files and XML datasets; APIs and web services; web scraping; and data streams and feeds.
* Discuss the characteristics and use of some of the programming, querying, and scripting languages relevant to data professionals.
* Define metadata management and explain its importance.
* Explain what data repositories are and the purpose they serve.
* Describe RDBMSs, list examples of them, and summarize their use cases, advantages, and disadvantages.
* Recall experiences that data professionals have had working with varied data sources and types.
* Describe RDBMSes and NoSQL databases as well as examples of use.
* Define NoSQL databases and list their types.
* Differentiate between RDBMs and NoSQL databases.
* Discuss the characteristics and applications of data warehouses, data marts, and data lakes.
* List essential considerations for choosing a data repository.
* Summarize ETL and ELT process as well as data pipelines.
* Explain the use of Data Integration Platforms and how they relate to data pipelines and the ETL and ELT processes.
* Recognize the data engineering tools, databases, and data repositories with which data professionals work.
* [Optional] Create a Db2 instance on IBM Cloud.
* Summarize what big data is and how it impacts the collection, monitoring, storage, analysis, and reporting of data.
* Discuss the role that Apache Hadoop, Apache Hive, and Apache Spark play in Big Data analytics.
* Recall various data professionals’ viewpoints regarding Big Data’s impact on data engineering.

A data engineer’s ecosystem includes the infrastructure, tools, frameworks, and processes for extracting data from disparate sources, architecting and managing data pipelines for transformation, integration, and storage of data, architecting and managing data repositories, automating and optimizing workflows and flow of data between systems; and developing applications needed through the data engineering workflow.

Data can be categorized as structured, semi-structured, or unstructured.

Data that follows a rigid format and can be organized neatly into rows and columns is **structured data.** e.g: data that you find in a spreadsheet.

**Semi-structured** data is a mix of data that has consistent characteristics and data that doesn’t conform to a rigid structure. For example, emails. An email has a mix of structured data, such as the name of the sender and recipient, but also has the contents of the email, which is unstructured data.

**Unstructured data**—data that is complex, and mostly qualitative information that is impossible to reduce to rows and columns. For example, photos, videos, text files, PDFs, and social media content.

**The type of data drives the kind of data repositorie**s that the data can be collected and stored in, and also the tools that can be used to query or process the data.

There are two main types of data repositories—Transactional and Analytical.

**Transactional systems**, also known as **Online Transaction Processing (or OLTP) systems**, are designed to **store high-volume day-to-day operational data**. Such as online banking transactions, ATM transactions, and airline bookings.

**Analytical systems,** also known as **Online Analytical Processing (OLAP) systems**, are optimized for conducting complex data analytics. These include relational and non-relational databases, data warehouses, data marts, data lakes, and big data stores.

**Data Integration tools** combine data from disparate sources into a unified view, accessed by users to query and manipulate the data.

**Data pipelines**, a set of tools and processes that cover the entire journey of data from source to destination systems.

Data is integrated within a data pipeline using processes such as the **Extract-Transform-and-Load Process or the Extract-Load-and Transform process**

**Type of data**

What is data? Data is unorganized information that is processed to make it meaningful. Generally, data comprises of **facts, observations, perceptions, numbers, characters, symbols, and images that can be interpreted to derive meaning**.

Structured data is objective **facts and numbers** that can be collected, exported, stored, and organized in typical databases. Some of the sources of structured data could include: SQL Databases and Online Transaction Processing (or OLTP) Systems that focus on business transactions Spreadsheets such as Excel and Google Spreadsheets Online forms Sensors such as Global Positioning Systems (or GPS) and Radio Frequency Identification (or RFID) tags; and Network and Web server logs.

Semi-structured data is data that has some organizational properties but lacks a fixed or rigid schema. Semi-structured data cannot be stored in the form of rows and columns as in databases. It contains tags and elements, or metadata, which is used to group data and organize it in a hierarchy. Some of the sources of semi-structured data could include**: E-mails XML and other markup languages Binary executables TCP/IP packets**. Zipped files Integration of data from different sources XML and JSON allow users to define tags and attributes to store data in a hierarchical form and are used widely to store and exchange semi-structured data.

Unstructured data is data that does not have an easily identifiable structure and, therefore, cannot be organized in a mainstream relational database in the form of rows and columns. It does not follow any particular format, sequence, semantics, or rules. Unstructured data can deal with the heterogeneity of sources and has a variety of business intelligence and analytics applications.

Some of the sources of unstructured data could include: **Web pages Social media feeds Images in varied file formats (such as JPEG, GIF, and PNG) Video and Audio files Documents and PDF files PowerPoint presentations Media logs; and Surveys** Unstructured data can be stored in files and documents (such as a Word doc)for manual analysis or in NoSQL databases that have their own analysis tools for examining this type of data.

**Understanding Different Types of Files**

**Delimited text files** are text files used to store data as text in which each line, or row, has values separated by a delimiter; where a delimiter is a sequence of one or more characters for specifying the boundary between independent entities or values. Most common delimiters are the **comma**, **tab**, **colon**, **vertical bar**, and **space**. E.g: CSV and TSV

**Microsoft Excel Open XML Spreadsheet, or XLSX**, is a Microsoft Excel Open XML file format that falls under the spreadsheet file format created by Microsoft. It can use and save all functions available in Excel and is also known to be one of the more secure file formats as it cannot save malicious code.

**Extensible Markup Language, or XML**, is a markup language with set rules for encoding data. The XML file format is both readable by humans and machines. It is a self-descriptive language designed for sending information over the internet. XML is similar to HTML in some respects, but also has differences—for example, an .XML does not use predefined tags like HTML does. XML is platform independent and programming language independent and therefore simplifies data sharing between various systems.

**Portable Document Format, or PDF**, is a file format developed by Adobe to present documents independent of application software, hardware, and operating systems, which means it can be viewed the same way on any device.

**JavaScript Object Notation, or JSON**, is a text-based open standard designed for transmitting structured data over the web. The file format is a language-independent data format that can be read in any programming language. JSON is easy to use, is compatible with a wide range of browsers, and is considered as one of the best tools for sharing data of any size and type, even audio and video. That is one reason, many APIs and Web Services return data as JSON.

**SOURCES OF DATA**

Relational Databases; Flatfiles and XML Datasets APIs and Web Services; Web Scraping;

**Flat files**, store data in plain text format, with one record or row per line, and each value separated by delimiters such as commas, semi-colons, or tabs. Data in a flat file maps to a single table, unlike relational databases that contain multiple tables. One of the most common flat-file format is CSV in which values are separated by commas.

Spreadsheet files are a special type of flat files, that also organize data in a tabular format–rows and columns. But a spreadsheet can contain multiple worksheets, and each worksheet can map to a different table.

XML files, contain data values that are identified or marked up using tags. While data in flat files is “flat” or maps to a single table, XML files can support more complex data structures, such as hierarchical. Some common uses of XML include data from online surveys, bank statements, and other unstructured data sets.

**API**, many data providers and websites provide APIs, or Application Program Interfaces, and Web Services, which multiple users or applications can interact with and obtain data for processing or analysis.

**Web scraping** is used to extract relevant data from unstructured sources. Also known as screen scraping, web harvesting, and web data extraction, web scraping makes it possible to download specific data from web pages based on defined parameters. Web scrapers can, among other things, extract text, contact information, images, videos, product items, and much more from a website. Some popular uses of web scraping include collecting product details from **retailers**, **manufacturers**, and **eCommerce websites** to provide price comparisons; **generating sales leads through public data sources**; **extracting data from posts and authors** on various forums and communities; and **collecting training and testing datasets for machine learning models** Some of the popular web scraping tools include **BeautifulSoup, Scrapy, Pandas, and Selenium.**

**Data streams** are another widely used source for aggregating constant streams of data flowing from sources such as instruments, IoT devices, and applications, GPS data from cars, computer programs, websites, and social media posts. Some of the data streams and ways in which they can be leveraged include: stock and market tickers for financial trading; retail transaction streams for predicting demand and supply chain management; surveillance and video feeds for threat detection; social media feeds for sentiment analysis; sensor data feeds for monitoring industrial or farming machinery; web click feeds for monitoring web performance and improving design; and real-time flight events for rebooking and rescheduling. Some popular applications used to process data streams include Apache Kafka, Apache Spark Streaming, and Apache Storm. RSS (or Really Simple Syndication) feeds, are another popular data source.

**LANGUAGES FOR DATA PROFESSIONALS**

Query languages, programming languages, and shell scripting.

**Query languages** are designed for accessing and manipulating data in a database; for example,

**Programming languages** are designed for developing applications and controlling application behavior; for example, Python, R, and Java;

**Shell and Scripting languages**, such as Unix/Linux Shell, and PowerShell, are ideal for repetitive and time-consuming operational tasks

**SQL, or Structured Query Language**, is a querying language designed for accessing and manipulating information from, mostly, though not exclusively, relational databases. We can write a set of instructions to perform operations in a database; Create new databases, tables, and views; and Write stored procedures.

Advantages of using SQL:

* SQL is portable and can be used independently of the platform
* It can be used for querying data in a wide variety of databases and data repositories, although each vendor may have some variations and special extensions.
* It has a simple syntax that is similar to the English language Its syntax allows developers to write programs with fewer lines than some of the other programming languages using basic keywords such as select, insert, into, and update.
* It can retrieve large amounts of data quickly and efficiently.
* It runs on an interpreter system, which means code can be executed as soon as it is written, making prototyping quick and easy. SQL is one of the most popular querying languages.

**Python** is a widely-used open-source, general-purpose, high-level programming language. Its syntax allows programmers to express their concepts in fewer lines of code, as compared to some of the older languages. Because of its focus on simplicity and readability, and a low learning curve, it’s an ideal tool for beginning programmers. It is great for performing high-computational tasks in vast amounts of data, which can otherwise be extremely time-consuming and cumbersome. Python provides libraries like Numpy and Pandas, which eases this task by the use of parallel processing. It has in built functions for almost all of the frequently used concepts. Python supports multiple programming paradigms, such as **object-oriented, imperative, functional, and procedural, making it suitable for a wide variety of use cases**. It runs on Windows and Linux environments and can be ported to multiple platforms. It has widespread community support with plenty of useful analytics libraries available. It has several open-source libraries for data manipulation, data visualization, statistics, and mathematics, to name just a few. Its vast array of libraries and functionalities also include: Pandas for data cleaning and analysis Numpy and Scipy, for statistical analysis Beautifulsoup and Scrapy for web scraping Matplotlib and Seaborn to visually represent data in the form of bar graphs, histogram, and pie-charts OpenCV for image processing

**R** is an open-source programming language and environment for data analysis, data visualization, machine learning, and statistics. Widely used for developing statistical software and performing data analytics, it is especially known for its ability to create **compelling visualizations**, giving it an edge over some of the other languages in this space.

Some of the key benefits of R include the following:

* It is an open-source platform-independent programming language
* It can be paired with many programming languages, including Python
* It is highly extensible, which means developers can continue to add functionalities by defining new functions
* It facilitates the handling of structured as well as unstructured data which means it has a more **comprehensive data capability**
* It has libraries such as Ggplot2 and Plotly that offer aesthetic graphical plots to its users
* You can make reports with the data and scripts embedded in them; also, interactive web apps that allow users to play with the results and the data
* It is dominant among other programming languages for developing statistical tools.

**Java** is an object-oriented, class-based, and platform-independent programming language originally developed by Sun Microsystems. **Java is used in a number of processes all through data analytics, including cleaning data, importing and exporting data, statistical analysis, and data visualization**. In fact, most of the popular frameworks and tools used for big data are typically written in Java, such as Hadoop, Hive, and Spark. It is perfectly suited for speed-critical projects.

**A Unix/Linux Shell** is a computer program written for the UNIX shell. It is a series of UNIX commands written in a plain text file to accomplish a specific task. Writing a shell script is fast and easy. It is most useful for repetitive tasks that may be time-consuming to execute by typing one line at a time. Typical operations performed by shell scripts include: **file manipulation program execution system administration tasks such as disk backups and evaluating system logs installation scripts for complex programs executing routine backups running batches** PowerShell is a cross-platform automation tool and configuration framework by Microsoft that is optimized for working with structured data formats**, such as JSON, CSV, XML, and REST APIs, websites, and office applications**. It consists of a command-line shell and scripting language. PowerShell is object-based, which makes it possible to filter, sort, measure, group, compare, and many more actions on objects as they pass through a data pipeline. It is also a good tool for **data mining, building GUIs, and creating charts, dashboards, and interactive reports**.

**Viewpoint: Working with Varied Data Sources And Types**

Challenge: Version of data, Change of Vendor. You need flexibility, finding the function that works.

DB2 and SQL server import and export data is different. The data that has a lot of characters in It which made it challenging to get a character that could be used as a delimiter. We have to use different separators for different tables.

**METADATA AND METADATA MANAGEMENT**

Metadata is data that provides information about other data.

**Technical metadata:** Technical metadata is metadata that defines the data structures in data repositories or platforms, primarily from a technical perspective.

For example, technical metadata in a data warehouse includes assets such as:

Tables that record information about the tables stored in a database, like:

* each table's name
* the number of columns and rows each table has

A data catalog, which is an inventory of tables that contain information, like:

* the name of each database in the Enterprise data warehouse
* the name of each column present in each database
* the names of every table that each column is contained in
* the type of data that each column contains

The technical metadata for relational databases is typically stored in specialized tables in the database called the **System Catalog**.

**Process metadata** describes the processes that operate behind business systems such as data warehouses, accounting systems, or customer relationship management tools.

Process metadata for such systems includes tracking things like:

* process start and end times
* disk usage
* where data was moved from and to, and
* how many users access the system at any given time

This sort of data is valuable for troubleshooting and optimizing workflows and ad hoc queries.

**Business metadata**, which is information about the data described in readily interpretable ways, such as:

* how the data is acquired
* what the data is measuring or describing
* the connection between the data and other data sources

Business metadata also serves as documentation for the entire data warehouse system.

**Managing metadata** includes developing and administering policies and processes to ensure information can be accessed and integrated from various sources and appropriately shared across the entire enterprise.

Good metadata management has many valuable benefits. Having access to a well-implemented data catalog greatly enhances data discovery, repeatability, and governance, and can also facilitate access to data.

**Data governance** is a data management concept concerning the capability that enables an organization to ensure that high data quality exists throughout the complete lifecycle of the data, and data controls are implemented that support business objectives.

The key focus areas of data governance include availability, usability, consistency, data integrity, and data security and includes establishing processes to ensure effective data management throughout the enterprise such as accountability for the adverse effects of poor data quality and ensuring that the data which an enterprise has can be used by the entire organization.

Popular metadata management tools include:

* IBM InfoSphere Information Server
* CA Erwin Data Modeler
* Oracle Warehouse Builder
* SAS Data Integration Server
* Talend Data Fabric
* Alation Data Catalog
* SAP Information Steward
* Microsoft Azure Data Catalog
* IBM Watson Knowledge Catalog
* Oracle Enterprise Metadata Management (OEMM)
* Adaptive Metadata Manager
* Unifi Data Catalog
* data.world
* Informatica Enterprise Data Catalog.

**The data Repositories, Data Pipelines, and Data Integration Platforms**

**Overview of data Repositories**

A data repository is a general term used to refer to data that has been **collected, organized, and isolated** so that it can be used for **business operations or mined for reporting and data analysis.**

**A database** is a collection of **data, or information**, designed for **the input, storage, search and retrieval, and modification of data**. And a Database Management System, or DBMS, is a set of programs that creates and maintains the database.

Several factors influence the choice of database, such as the **data type and structure, querying mechanisms, latency requirements, transaction speeds, and intended use of the data.**

There are two types of databases.

**Relational Databases**: data organized into a tabular format with rows and columns following a well-defined **structure and schema**. However, unlike flat files, RDBMSes are optimized for data operations and querying involving many tables and much larger data volumes.

**Non-relational databases** emerged in response to the volume, diversity, and speed at which data is being generated today, mainly influenced by advances in cloud computing, the Internet of Things, and social media proliferation. Built for speed, flexibility, and scale, non-relational databases made it possible to store data in a **schema-less or free-form** fashion. NoSQL

Non-relational databases are usually big volume and diversify data stored in a **schema-less or free-form** fashion**.**

**Data warehouse** consolidates data through the extract, transform, and load process, also known as the ETL process, into **one comprehensive database** for **analytics and business Intelligence**.

**RDBMS**

A relational database is a collection of data organized into a table structure, where the tables can be linked, or related, based on data common to each.

Relational databases use structured query language, or SQL, for querying data. Relational databases build on the organizational principles of flat files such as spreadsheets, with data organized into rows and columns following a well-defined structure and schema. But that is where the similarity ends. Relational databases, by design, are ideal for the **optimized storage, retrieval, and processing of data for large volumes of data**, unlike spreadsheets that have a limited number of rows and columns. Each table in a relational database has a unique set of rows and columns and relationships can be defined between tables, which minimizes data redundancy. Moreover, you can restrict database fields to **specific data types and values**, which minimizes irregularities and leads to greater consistency and data integrity. Relational databases use SQL for querying data, which gives you the advantage of processing millions of records and retrieving large amounts of data in a matter of seconds.

IBM DB2, Microsoft SQL Server, MySQL, Oracle Database, and PostgreSQL are some of the popular relational databases.

Some of the popular cloud relational databases include **Amazon Relational Database Service (RDS)**, **Google Cloud SQL**, **IBM DB2 on Cloud**, **Oracle Cloud**, and **SQL Azure**.

One of the most significant advantages of the relational database approach is its ability to create meaningful information by **joining tables**. Some of its other advantages include: **Flexibility**: Using SQL, you can add new columns, add new tables, rename relations, and make other changes while the database is running and queries are happening. **Reduced redundancy**: Relational databases minimize data redundancy. For example, the information of a customer appears in a single entry in the customer table, and the transaction table pertaining to the customer stores a link to the customer table. **Ease of backup and disaster recovery**: Relational databases offer easy export and import options, making backup and restore easy. Exports can happen while the database is running, making restore on failure easy.

**ACID-compliance**: ACID stands for **Atomicity, Consistency, Isolation, and Durability**.

And ACID compliance implies that the data in the database remains **accurate and consistent despite failures**, and **database transactions are processed reliably**. Now we’ll look at some use cases for relational databases:

RDBMS does not work well with semi-structured and unstructured data and is, therefore, not suitable **for extensive analytics** on such data. For migration between two RDBMSs, **schemas and type of data** need to be identical between the source and destination tables. Relational databases have a **limit on the length of data fields**, which means if you try to enter more information into a field than it can accommodate, the information will not be stored.

**NoSQL (Not only SQL, Non-SQL)**

NoSQL databases are built for specific data models and have flexible schemas that allow programmers to create and manage modern applications. They do not use a traditional row/column/table database design with fixed schemas, and typically not use the structured query language (or SQL) to query data, although some may support SQL or SQL-like interfaces.

NoSQL allows data to be stored in a **schema-less or free-form fashion**. Any data, structured, semi-structured, or unstructured, can be stored in any record. Based on the model being used for storing data, there are four common types of NoSQL databases:

**Key-value store, Document-based, Column-based, and graph-based**.

**Key-value store:** Data in a key-value database is stored as a collection of **key-value pairs**. The key represents an attribute of the data and is a unique identifier

Key-value stores are great for storing user **session data and user preferences**, making real-time recommendations and targeted advertising, **and in-memory data caching.**

If you want to be able to query the data on specific data values, need relationships between data values, or need to have multiple unique keys, a key-value store may not be the best fit. **Redis, Memcached, and DynamoDB** are some well-known examples in this category.

**Document-based**: Document databases store each record and its associated data within a single document. They enable flexible indexing, powerful ad hoc queries, and analytics over collections of documents. Document databases are preferable for **eCommerce platforms, medical records storage, CRM platforms, and analytics platforms.** However, if you’re looking to **run complex search queries and multi-operation transactions**, a document-based database may not be the best option for you. **MongoDB, DocumentDB, CouchDB, and Cloudant** are some of the popular document-based databases

**Column-based**: Column-based models store data in cells grouped as columns of data instead of rows. A logical grouping of columns, that is, columns that are usually accessed together, is called a **column family**. For example, a customer’s name and profile information will most likely be accessed together but not their purchase history. So, customer name and profile information data can be grouped into a column family.

Since column databases store all cells corresponding to a column as a continuous disk entry, **accessing and searching the data becomes very fast**. Column databases can be great for systems that require **heavy write requests, storing time-series data, weather data, and IoT data**. But if you need to **use complex queries or change your querying patterns frequently**, this may not be the best option for you. The most popular column databases are **Cassandra and HBase**.

**Graph-based**: Graph-based databases use a graphical model to represent and store data. They are particularly useful for visualizing, analyzing, and finding connections between different pieces of data. The circles are **nodes**, and they contain the data. The arrows represent relationships. Graph databases are an excellent choice for working with **connected data**, which is data that **contains lots of interconnected relationships**. Graph databases are great for **social networks, real-time product recommendations, network diagrams, fraud detection, and access management**. But if you want to process high volumes of transactions, it may not be the best choice for you, because graph databases are not optimized for **large-volume analytics queries**. **Neo4J and CosmosDB** are some of the more popular graph databases

The primary advantage of NoSQL is its ability to handle large volumes of structured, semi-structured, and unstructured data. Some of its other advantages include: The ability to run as a distributed system across multiple data centers, which enables them to take advantage of cloud computing infrastructure; An efficient and cost-effective scale-out architecture that provides additional capacity and performance with the addition of new nodes; and Simpler design, better control over availability, and improved scalability that enables you to be more agile, more flexible, and to iterate more quickly.

**key differences between relational and non-relational databases:**

**RDBMS** schemas rigidly define how all data inserted into the database must be typed and composed, whereas **NoSQL** databases can be schema-agnostic, allowing unstructured and semi-structured data to be stored and manipulated. Maintaining high-end, commercial **relational database management systems** is expensive whereas **NoSQL** databases are specifically designed for low-cost commodity hardware **Relational databases**, unlike most NoSQL, support ACID-compliance, which ensures reliability of transactions and crash recovery. **RDBMS** is a mature and well-documented technology, which means the risks are more or less perceivable as compared to **NoSQL**, which is a relatively newer technology.

we will learn about some of the characteristics and applications of data warehouses, data marts, and data lakes.

**A data warehouse** is a central repository of data integrated from multiple sources. Data warehouses serve as the single source of truth—storing current and historical data that has been cleansed, conformed, and categorized.

When data gets loaded into the data warehouse, it is already modeled and structured for a specific purpose, meaning it's **analysis-ready**. Traditionally, data warehouses are known to store relational data from transactional systems and operational databases such as CRM, ERP, HR, and Finance applications. But with the emergence of NoSQL technologies and new data sources, non-relational data repositories are also being used for data warehousing.

Typically, a data warehouse has a three-tier architecture: **The bottom tier** of the architecture includes the **database servers**, which could be relational, non-relational, or both, that extract data from different sources. **The middle tier** of the architecture consists of the **OLAP Server**, a category of software that allows users to **process and analyze** information coming from multiple database servers. And **the topmost tier** of the architecture includes **the client front-end layer**. This tier includes all the tools and applications used for **querying, reporting, and analyzing data**.

In response to the rapid data growth and today's sophisticated analytics tools, data warehouses that once resided in on-premise data centers are moving to the cloud. Compared to their on-premise versions, some of the benefits offered by cloud-based data warehouses include: **Lower costs, Limitless storage and compute capabilities, Scale on a pay-as-you-go basis; and Faster disaster recovery.**

Some of the popularly used data warehouses include Teradata Enterprise Data Warehouse platform, Oracle Exadata, IBM Db2 Warehouse on Cloud, IBM Netezza Performance Server, Amazon RedShift, BigQuery by Google Cloudera's Enterprise Data Hub, and Snowflake Cloud Data Warehouse.

**A data mart** is a sub-section of the data warehouse, built specifically for a particular business function, purpose, or community of users.

There are three basic types of data marts—dependent, independent, and hybrid data marts.

**Dependent data marts** are a sub-section of an **enterprise data warehouse**. Since a dependent data mart offers analytical capabilities for a restricted area of the data warehouse, it also provides isolated security and isolated performance.

**Independent data marts** are created from sources other than an enterprise data warehouse, such as internal operational systems or external data.

**Hybrid data marts** combine inputs from data warehouses, operational systems, and external systems. The difference also lies in how data is extracted from the source systems, the transformations that need to be applied, and how the data is transported into the mart.

**A Data Lake** is a data repository that can store large amounts of structured, semi-structured, and unstructured data in their native format.

A data lake is a reference architecture that is independent of technology. Data lakes combine a variety of technologies that come together to **facilitate agile data exploration for analysts and data scientists**. Data lakes can be deployed using **Cloud Object Storage, such as Amazon S3**, **or large-scale distributed systems such as Apache Hadoop, used for processing Big Data**. They can also be deployed on different relational database management systems, as well as NoSQL data repositories that can store very large amounts of data.

Data lakes offer a number of benefits, such as:

The ability to store all types of data – unstructured data such as documents, emails, PDFs, semi-structured data such as JSON, XML, CSV, and logs, as well as structured data from relational databases

The agility to **scale based on storage capacity** – growing from terabytes to petabytes of data.

Saving time in defining structures, schemas, and transformations since data is imported in its original format

And The ability to repurpose data in several different ways and wide-ranging use cases. This is extremely beneficial as it is hard for businesses to foresee all the different ways in which you could potentially leverage their data in the future. Some of the vendors that provide technologies, platforms, and reference architectures for data lakes include Amazon, Cloudera, Google, IBM, Informatica, Microsoft, Oracle, SAS, Snowflake, Teradata, and Zaloni

**Data Lakehouses Expalined**

Developers took a step back and said, hey, let's take the best of both data lakes and data warehouses and combine them into a new technology called the data lakehouse.

So we get the **flexibility and we get the cost-effectiveness** of a data lake, and we get the **performance and structure** of a data warehouse. So we'll talk more specifically about the architecture of a data lake house in a future video. But from a value point of view, the **lake house** lets us store data from the exploding number of new sources in a low-cost way and then leverages built-in data management and governance layers to allow us to power both business intelligence and high performance machine learning workloads quickly.

**Viewpoints: Considerations for choice of data repository**

Look at the use cases, What data and type of data will be used for? Structured, unstructured or semi structured information. Schema of the data? What is the performance requirements? Do you need data at rest or streaming data. Data encryption needs? The volume of data and whether you need a big data system?

Storage requirements. Frequency of data access, frequent updates, Keep in vault for long time. Standards set by your organization on the databases and database repositories that can be used. Purpose of data repository: Transactional, Analytical, Archival, Data warehouse. Compatibility of the data repository with the existing ecosystem of programming languages, tools and processes. Security features of the data repository. Scalability.

**ETL, ELT, AND DATA PIPELINES**

ETL(Extract, Transform, Load) process, is an automated process in which you gather raw data from identified sources, extract the information that aligns with your reporting and analysis needs, clean, standardize, and transform that data into a format that is usable in the context of your organization; and load it into a data repository.

**Data Extraction** could be through:

**Batch processing:** Data is moved in large chunks from the source to the target system at scheduled intervals. Tools used are **Stitch and Blendo**

**Stream processing:** Data is pulled in real-time from the source and transformed while is it in transit and before it is loaded into the data repository. Tools used are **Apache Samza, Apache Storm and Apache Kafka**.

**Data Transform:** Transform involves the execution of rules and functions that convert raw data into data that can be used for analysis.

For example making date formats and units of measurement consistent across all source data removing duplicate data filtering out data that you do not need enriching data, for example, splitting full name to first, middle, and last names establishing key relationships across tables applying business rules and data validations.

**Data Load:** Load is the step where processed data is transported to a destination system or data repository. It could be:

* Initial loading, that is, populating all the data in the repository;
* Incremental loading, that is, applying ongoing updates and modifications as needed periodically.
* Full refresh: Erasing a data table and reloading fresh data.

**Load verification**—which includes data checks for:

* missing or null values,
* server performance,
* and monitoring load failures. It is important to keep an eye on load failures.

Some of the popular ETL tools available include **IBM Infosphere Information Server, AWS Glue, Improvado, Skyvia, HEVO, and Informatica PowerCenter.**

In the ELT process, extracted data is first loaded into the target system, and transformations are applied in the target system. The destination system for an ELT pipeline is most likely a data lake, though it can also be a data warehouse. ELT is a relatively new technology powered by cloud technologies.

The ELT Process:

* Helps process large sets of unstructured and non-relational data.
* It is ideal for data lakes where transformations on the data are applied once the raw data is loaded into the data lake.

Advantages:

* Shortens the cycle between extraction and delivery.
* Allows you to ingest volumes of raw data as immediately as the data becomes available
* Affords greater flexibility to analysts and data scientists for **exploratory data analytics**
* ELT transforms only the data that is required for a particular analysis so it can be leveraged for multiple use cases.
* Is more suited to work with Big Data.

**Data Pipeline**: data pipeline is a broader term that

* Encompasses the entire journey of moving data from one system to another, of which ETL and ELT may be subsets.
* Can be used for batch processing, for streaming data, and a combination of batch and streaming data.
* Supports both long-running batch queries and smaller interactive queries.
* Typically loads data into a data lake but can also load data into a variety of target destinations, including other applications and visualization tools.

There are a number of data pipeline solutions available, most popular among them being **Apache Beam, AirFlow, and DataFlow**.

**DATA INTEGRATION PLATFORMS**

Gartner defines data integration as a discipline comprising the practices, architectural techniques, and tools that allow organizations to ingest, transform, combine, and provision data across various data types.

**Data integration** has several usage scenarios, such as

* Data consistency across applications
* Master data management
* Data sharing between enterprises
* And data migration and consolidation.

In the field of analytics and data science, data integration includes:

* Accessing, queueing, or extracting data from operational systems
* Transforming and merging extracted data either logically or physically
* Data quality and governance
* And delivering data through an integrated approach for analytics purposes.

For example, to make customer data available for analytics, you would need to extract individual customers' information from operational systems such as sales, marketing, and finance. You would then need to provide a unified view of the combined data so that your users can access, query, and manipulate this data from a single interface to derive statistics, analytics, and visualizations.

How does a data integration platform relate to ETL and data pipelines?

While data integration combines disparate data into a unified view of the data, a data pipeline covers the entire data movement journey from source to destination systems. In that sense, you use a data pipeline to perform data integration, while ETL is a process within data integration.

There is no one approach to data integration. However, Modern data integration solutions typically support the following capabilities:

* An extensive catalog of **pre-built connectors and adopters** that help you connect and build integration flows with a wide variety of data sources such as databases, flat files, social media data, APIs, CRM and ERP applications.
* Open-source architecture that provides greater flexibility and avoids vendor lock-in.
* Optimization for both batch processing of large-scale data and continuous data streams, or both.
* Integration with Big Data sources.
* Additional functionalities. For example, specific demands around data quality and governance, compliance, and security.
* Portability, which ensures that as businesses move to cloud models, they should be able to run their data integration platforms anywhere. And data integration tools are able to work natively in a single cloud, multi-cloud, or hybrid cloud environment.

|  |
| --- |
| **IBM** offers a host of data integration tools targeting a range of enterprise integration scenarios, such as Information Server for IBM, Cloud Pak for Data, IBM Cloud Pak for Integration, IBM Data Replication, IBM Data Virtualization Manager, IBM InfoSphere Information Server on Cloud, and IBM InfoSphere DataStage  **Talend's** data integration tools include Talend Data Fabric, Talend Cloud, Talend Data Catalog, Talend Data Management, Talend Big Data, Talend Data Services, and Talend Open Studio. SAP, Oracle, Denodo, SAS, Microsoft, Qlik, and TIBCO are some of the other vendors that offer data integration tools and platforms.  Examples of open-source frameworks include **Dell Boomi, Jitterbit, and SnapLogic**.  **Cloud-based Integration Platform as a Service**, or iPaaS, as a hosted service via virtual private cloud or hybrid cloud. Such as the Adeptia Integration Suite, Google Cloud's Cooperation 534, IBM's Application Integration Suite on Cloud, and Informatica's Integration Cloud. |

**VIEWPOINTS: TOOLS, DATABASES AND DATA REPOSITORIES OF CHOICE**

Open sources: RDBMS databases such as MySQL, NoSQL databases such as MongoDB, Cassandra, Graph Databases like Neo4J, python, Apache Airflow for data pipelines, Spark for Big Data Processing. Kafka for handling streaming data.

Talend for ETL, Beauriful soup and Scrappy for web scrapping and variety of cloud storage.

SQL Server data repository, SQL server Integration services, OKADA for data integration, AWS redshift Data warehouses.

Jenkins.

**BIG DATA PLATFORMS**

Big Data refers to the dynamic, large and disparate volumes of data being created by people, tools, and machines. It requires new, innovative, and scalable technology to collect, host, and analytically process the vast amount of data gathered in order to derive real-time business insights that relate to consumers, risk, profit, performance, productivity management, and enhanced shareholder value.

**Velocity, volume, variety, veracity, and value**. These are the V's of Big Data.

**Velocity** is the speed at which data accumulates. Data is being generated extremely fast, in a process that never stops. Near or real-time streaming, local, and cloud-based technologies can process information very quickly.

**Volume** is the scale of the data, or the increase in the amount of data stored. **Drivers of volume** are the increase in data sources, higher resolution sensors, and scalable infrastructure.

**Variety** is the diversity of the data. Structured data fits neatly into rows and columns, in relational databases while unstructured data is not organized in a pre-defined way, like Tweets, blog posts, pictures, numbers, and videos. Variety also reflects that data comes from different sources, machines, people, and processes, both internal and external to organizations. **Drivers** are mobile technologies, social media, wearable technologies, geo technologies, video, and many, many more.

**Veracity** is the quality and origin of data, and its conformity to facts and accuracy. Attributes include consistency, completeness, integrity, and ambiguity. Drivers include cost and the need for traceability.

**Value** is our ability and need to turn data into value. Value isn't just profit. It may have medical or social benefits, as well as customer, employee, or personal satisfaction. The main reason that people invest time to understand Big Data is to derive value from it.

Velocity: Every 60 seconds, hours of footage are uploaded to YouTube which is generating data. Think about how quickly data accumulates over hours, days, and years. Volume: The world population is approximately seven billion people and the vast majority are now using digital devices; mobile phones, desktop and laptop computers, wearable devices, and so on. These devices all generate, capture, and store data -- approximately 2.5 quintillion bytes every day. That's the equivalent of 10 million Blu-ray DVD's. Variety: Let's think about the different types of data; text, pictures, film, sound, health data from wearable devices, and many different types of data from devices connected to the Internet of Things. Veracity: 80% of data is considered to be unstructured and we must devise ways to produce reliable and accurate insights.

The scale of the data being collected means that it’s not feasible to use conventional data analysis tools. However, alternative tools that leverage distributed computing power can overcome this problem. Tools such **as Apache Spark, Hadoop** and its ecosystem provide ways to extract, load, analyze, and process the data across distributed compute resources, providing new insights and knowledge.

**Big Data Processing Tools: Hadoop, HDFS, Hive, and Spark**

three open source technologies and the role they play in big data analytics — ApacheHadoop, Apache Hive, and Apache Spark.

**Hadoop** is a collection of tools that provides distributed storage and processing of big data. **Hive** is a data warehouse for data query and analysis built on top of Hadoop. **Spark** is a distributed data analytics framework designed to perform complex data analytics in real-time.

Hadoop, a java-based open-source framework, allows distributed storage and processing of large datasets across clusters of computers. In Hadoop distributed system, a node is a single computer, and a collection of nodes forms a cluster. Hadoop can scale up from a single node to any number of nodes, each offering local storage and computation. Hadoop provides **a reliable, scalable, and cost-effective** solution for storing data with no format requirements. Using Hadoop, you can: Incorporate emerging data formats, such as streaming audio, video, social media sentiment, and clickstream data, along with structured, semi-structured, and unstructured data not traditionally used in a data warehouse. Provide real-time, self-service access for all stakeholders. Optimize and streamline costs in your enterprise data warehouse by consolidating data across the organization and **moving “cold” data**, that is, data that is not in frequent use, to a Hadoop-based system. One of the four main components of Hadoop **is Hadoop Distributed File System, or HDFS**, which is a storage system for big data that runs on multiple commodity hardware connected through a network. **HDFS** provides scalable and reliable big data storage by partitioning files over multiple nodes. It splits large files across multiple computers, allowing parallel access to them. Computations can, therefore, run in parallel on each node where data is stored. It also replicates file blocks on different nodes to prevent data loss, making **it fault-tolerant**. Let’s understand this through an example. Consider a file that includes phone numbers for everyone in the United States; the numbers for people with last name starting with A might be stored on server 1, B on server 2, and so on. With Hadoop, pieces of this phonebook would be stored across the cluster. To reconstruct the entire phonebook, your program would need the blocks from every server in the cluster. HDFS also replicates these smaller pieces onto two additional servers by default, ensuring availability when a server fails, In addition to higher availability, this offers multiple benefits. It allows the **Hadoop cluster** to break up work into smaller chunks and run those jobs on all servers in the cluster for better scalability. Finally, you gain the benefit of **data locality**, which is the process of moving the computation closer to the node on which the data resides.

Some of the other benefits that come from using HDFS include: Fast recovery from hardware failures, because HDFS is built to detect faults and automatically recover. Access to streaming data, because HDFS supports high data throughput rates. Accommodation of large data sets, because HDFS can scale to hundreds of nodes, or computers, in a single cluster. Portability, because HDFS is portable across multiple hardware platforms and compatible with a variety of underlying operating systems.

**Hive** is an open-source data warehouse software for reading, writing, and managing large data set files that are stored directly in either HDFS or other data storage systems such as Apache HBase. Hadoop is intended for long sequential scans and, because Hive is based on Hadoop, queries have very high latency—which means Hive **is less appropriate for applications that need very fast response times**. Also, Hive is read-based, and **therefore not suitable for transaction processing that typically involves a high percentage of write operations**. Hive is better suited for data warehousing tasks such as **ETL, reporting, and data analysis and includes tools that enable easy access to data via SQL.**

**Spark**, a general-purpose data processing engine designed to extract and process large volumes of data for a wide range of applications, including Interactive Analytics, Streams Processing, Machine Learning, Data Integration, and ETL. It takes advantage of in-memory processing to significantly increase the speed of computations and spilling to disk only when memory is constrained. Spark has interfaces for major programming languages, including Java, Scala, Python, R, and SQL. It can run using its standalone clustering technology as well as on top of other infrastructures such as Hadoop. And it can access data in a large variety of data sources, including HDFS and Hive, **making it highly versatile**. The ability to process streaming data fast and perform complex analytics in real-time is the key use case for Apache Spark.

**Viewpoints: Impact of Big data on Data Engineering.**