ISSN: 2634 - 8853

Journal of Engineering and Applied Sciences Technology

SCIENTIFIC Research and Community

Review Article

Open Access

Empowering Data Programs: The Five Essential Data Engineering Concepts for Program Managers

Mahesh Deshpande[1]* and Ipsita Nanda[2]

[1]Senior Principal Consultant, Genpact, San Jose, CA, USA

[2]Senior Project Manager, Tech Mahindra, San Jose, CA, USA

ABSTRACT

Data engineering is a crucial aspect of modern organizations, transforming raw data into actionable insights. This article explores five essential data engineering concepts that program managers should understand to effectively lead data-driven initiatives. These concepts include data architecture and storage, data modeling and design, ETL pipelines and processes, big data technologies, and data governance and security. By gaining a solid grasp of these concepts, program managers can better align data infrastructure with business goals, communicate effectively with stakeholders, and ensure the success of data engineering projects. The article delves into each concept, providing definitions, examples, and best practices, empowering program managers to navigate the complex landscape of data engineering with confidence and drive meaningful results for their organizations.

***Corresponding author**

Mahesh Deshpande, Senior Principal Consultant, Genpact, San Jose, CA, USA.

Received: May 06, 2023; **Accepted:** May 10, 2023; **Published:** May 20, 2023

Keywords: Program Management, Data Engineering, Data, TPM Leadership Toolkit, Data Engineering Program Manager

Introduction

In today's business landscape, data serves as the lifeblood for modern organizations, coursing through the veins of analytics and reports to empower leaders in making informed decisions. Amid this data deluge, the role of a data engineering program manager is pivotal, as they play a key role in transforming raw data into actionable insights. As someone deeply immersed in the realm of data engineering program management, we have witnessed firsthand the challenges that arise when dealing with terms like "ETL pipelines," "data fabric," and "cloud infrastructure." This unfamiliarity can create communication gaps, impede project progress, and ultimately hinder the effectiveness of program managers. Navigating the data landscape demands more than just business acumen and communication skills from Program Managers (PMs); it necessitates a solid understanding of the engine powering it all: data engineering.

In simple terms, Data Engineering is both an art and a science, involving the creation of infrastructure that collects, stores, processes, and delivers data for analysis. The architects and builders of this critical infrastructure are the data engineers, responsible for ensuring the accuracy, reliability, and accessibility of the delivered data. This foundation is paramount for the entire data-driven decision-making process. Program managers play a vital role in the success of data engineering initiatives by ensuring that projects are delivered on time, within budget, and to specification. However, effectively managing data engineering projects requires a fundamental understanding of the core concepts and principles that underpin this domain. Program Managers equipped with an understanding of Data Engineering concepts become exceptional in their roles as they can ensure that:

- The data infrastructure better aligns with the business goals prioritized by the leadership.
- The status updates gathered and provided to stakeholders are crisp and non-confusing.
- The communication with the data engineering teams is seamless as they speak the same data lingo.

This article provides an overview of five essential data engineering concepts that can help program managers become effective data engineering managers.

The 5 Essential Data Engineering Concepts for Program Managers
The Data Engineering projects are complex undertakings that require careful planning, execution, and oversight. While program management skills are essential for overseeing any project, data engineering projects demand a deeper understanding of the underlying data engineering concepts. By equipping themselves with a foundational knowledge of data engineering principles and concepts, program managers can make better decisions, collaborate more effectively, manage risks more adeptly, and ultimately, ensure the successful delivery of data-driven initiatives. Program managers who are well-versed in data engineering concepts are better equipped to lead these projects to success. The five essential concepts critical for a data program manager are as follows:

Data Storage and Data Manipulation

Before deep diving into key data storage and complex architecture

concepts, it is imperative to understand the different types of data. Data comes in various types, and understanding these distinctions is crucial for effective management and analysis. Below table summarizes the various data types along with some examples:

Table 1: Different Types of Data

Structured Data	• Highly organized and formatted data with a clear and fixed schema.	Ex: Tables in Relational Databases, Spreadsheets
Unstructured Data	• Data lacking a predefined data model or structure	Ex: Emails, social media posts, multimedia files etc.
Semi Structured Data	• Data that has some level of structure in forms of tags, keys, or elements	Ex: JSON (JavaScript Object Notation), XML (eXtensible Markup Language)
Quantitative Data	• Numerical data that represents measurable quantities and can be subjected to mathematical operations.	Ex: Sales figures, temperature, stock prices
Qualitative Data	• Descriptive, non-numerical data that cannot be measured but can be categorized	Ex: Colors, survey responses, emotions
Time-Series Data	• Data points recorded over regular time intervals, suitable for analyzing trends and patterns over time	Geo-spatial Data
Geo-spatial Data	• Data that includes geographical information or spatial coordinates, enabling analysis based on location	Ex. GPS coordinates, maps, spatial databases
Meta-Data	• Data that provides information about primary data by describing properties, and context of the primary data	Ex: File timestamps, data source information, data format details

Understanding the characteristics and requirements of different types of data is essential for program managers to effectively manage, analyze, and derive meaningful insights from their information resources.

Data Storage Management

Once you are familiar with the basic types of data, a Data Engineering Program Manager should understand how data is stored and accessed. Here's a breakdown of crucial storage concepts:

Relational Databases

Structured data organized in tables with rows and columns, with relationships among tables established through primary and foreign keys [1]. These are widely used in traditional business applications ERP, CRM for maintaining customer records or financial transactions. Examples: MySQL, PostgreSQL, Oracle etc.

NoSQL Databases

A broad category of flexible data structures beyond traditional relational databases designed for large, semi-structured data like sensor readings, social media posts, or website logs [2]. The most common types of NoSQL databases are:

- **Document Stores:** Data is stored in document-like structures (JSON, BSON etc.); ideal for content management systems and e-commerce applications. Examples: MongoDB, Couchbase.
- **Key-Value Stores:** Data is stored as key-value pairs, suitable for caching and storing user sessions. Examples: Redis, Amazon DynamoDB.
- **Wide-Column Stores:** Stores data in tables, rows, and dynamic columns. Great for analyzing large datasets. Examples: Cassandra, HBase.
- **Graph Databases:** Designed for data whose relations are well represented as a graph and has elements interconnected with many relationships. Example: Neo4j, Amazon Neptune

In-Memory Databases

These databases store data in the main memory (RAM) instead of disk, which significantly speeds up data retrieval times. Widely used for Real-time analytics, caching, session storage, and applications where high read and write speeds are critical [3]. Example: Redis, SAP HANA.

Time-Series Databases

Optimized for storing and managing time-stamped data or data that changes over time, such as IoT app monitoring, real-time analytics. Examples: Influx DB, Timescale DB [4].

Vector Database

Vector Database is a powerful concept gaining huge traction in data engineering. While traditional databases store data in rows and columns and excel at storing and retrieving data based on exact matches, Vector databases stores data in multi-dimensional points in a high-dimensional space. Each point (a vector) captures the essence of a piece of data (e.g., an image, text document, or sensor reading) based on its unique characteristics. This unlocks powerful capabilities like:

- **Similarity Search:** Find similar data points even if they have different formats or structures. Perfect for recommending music, identifying fraudulent transactions, or searching for similar images.
- **Real-time Analysis:** Handle massive datasets efficiently for tasks like anomaly detection in sensor data or personalized recommendations in real-time.
- **Machine Learning Integration:** Easily train and deploy machine learning models that rely on vector representations of data.

Vector databases are relatively new compared to traditional databases and are still evolving. They might not be suitable for all data types, especially highly structured data. However, vector databases are a crucial component in the modern data ecosystem, especially with the growing importance of AI and machine learning across various industries [5]. Their ability to efficiently handle complex, high-dimensional data makes them invaluable for applications requiring fast and accurate similarity searches. In an era where data analysis increasingly relies on deep understanding of information, Program Managers can stay ahead of the curve by learning about Vector databases such as *Pinecone* and *Milvus*.

Data Querying and Data Manipulation

Data querying refers to the process of retrieving data from a database or data store based on specific criteria. It involves using query languages to specify the desired data and the conditions for retrieval. Data manipulation, on the other hand, encompasses the tasks of inserting, updating, and deleting data within a database. Data querying and manipulation are essential skills for data engineering program managers as they allow them to extract, transform, and analyze data effectively. SQL (Structured Query Language) and scripting languages such as Python play a crucial role in this process.

SQL Basics

SQL is the standard language used for relational database management systems (RDBMS). It provides a declarative way to interact with databases, allowing users to define the desired result set without specifying the exact steps to retrieve it [6].

Key Concepts of SQL

- **SELECT Statement:** Used to retrieve data from one or more tables based on specified criteria.
- **WHERE Clause:** Filters the result set based on specified conditions.
- **JOIN Operations:** Combines rows from two or more tables based on a related column between them.
- **GROUP BY and HAVING Clauses:** Used for grouping and aggregating data based on specific columns.
- **INSERT, UPDATE, and DELETE Statements:** Used for modifying data within a table.

Scripting Languages - Python

Python is a versatile and widely used programming language in data engineering. It provides a rich set of libraries and tools for data manipulation, analysis, and integration [7].

Key Aspects of Python for Data Engineering

- **Data Processing Libraries:** Python offers powerful libraries like pandas, NumPy, and PySpark for data manipulation, cleansing, and analysis.
- **Database Connectivity:** Python provides libraries like SQLAlchemy and psycopg2 for connecting to various databases and executing SQL queries.
- **Data Integration:** Python can be used to extract data from different sources (e.g., APIs, files, databases), transform it, and load it into target systems.
- **Scripting and Automation:** Python allows data engineering program managers to automate repetitive tasks, such as data pipelines, ETL processes, and data validations.

Basic knowledge of SQL and Python is crucial for data engineering program managers for several reasons:

- **Data Understanding:** SQL allows program managers to explore and understand the structure and content of databases. They can write queries to retrieve specific subsets of data, join tables, and aggregate information to gain insights.
- **Data Validation and Quality Checks:** Program managers can use SQL to perform data validation and quality checks. They can write queries to identify missing values, duplicates, or inconsistencies in the data.
- **Data Transformations:** SQL and Python enable program managers to transform and manipulate data. They can use SQL to filter, sort, aggregate data, while Python can be used for more complex transformations and data cleansing.
- **Data Pipeline Development:** Program managers with SQL and Python skills can actively participate in the development of data pipelines. They can write scripts to extract data from source systems, transform it and load it into target systems.
- **Collaboration With Technical Teams:** Knowledge of SQL and Python allows program managers to effectively communicate with data engineers, analysts, and other technical stakeholders. They can provide precise requirements, review code, and contribute to technical discussions.
- **Performance Optimization:** Understanding SQL and Python enables program managers to identify and optimize slow-performing queries or inefficient data processing scripts.

While data engineering program managers may not be expected to write complex SQL queries or Python scripts daily, having a basic understanding of these technologies empowers them to effectively manage data engineering projects, make informed decisions, and collaborate with technical teams. It's important for program managers to continuously enhance their SQL and Python skills through training, hands-on practice, and staying updated with the latest trends and best practices in data engineering.

Cloud vs. On-Premise Deployment

For a data engineering program, when deciding between cloud and on-prem deployment, it's crucial for data engineering program managers to assess their organization's specific requirements, existing infrastructure, data security needs, budget, and long-term goals. Many a times a hybrid approach, combining both cloud and on-prem components, can also be considered to balance the benefits and address organization's specific needs. Data Engineering Program Managers need to be aware of the following factors, which are critical in deciding between choosing cloud vs. on-prem for deployment:

Scalability and Elasticity

Scaling resources on-prem requires physical hardware upgrades and can be time-consuming and costly, whereas cloud platforms allow for easy scaling of resources up or down based on demand and quick provisioning of additional compute, storage, and network resources as needed, offering high elasticity [8].

Infrastructure Management

With on-prem deployment, the responsibility of managing and maintaining the entire infrastructure stack, including hardware, networking, and security rests with the organization, requiring dedicated IT resources and expertise, whereas in case of cloud deployment, cloud providers handle the underlying infrastructure, including hardware, networking, and maintenance. Cloud allows engineering team to focus on application development instead of infrastructure management.

Cost Structure

On-prem deployment requires upfront capital investment for hardware, software licenses, and infrastructure setup along withs ongoing costs for maintenance, upgrades, and support, whereas cloud deployment follows a pay-as-you-go model, where orgs are billed based on the resources consumed, providing the ability to optimize costs based on usage patterns [9].

Data Security and Compliance

With on-prem deployment, organizations have full control over their data security and can implement stringent security measures tailored to organization's needs, which can be crucial for specific industries such as federal government. Cloud providers also offer

robust security measures and data encryption, which however needs to be carefully assessed first.

Data Transfer and Latency

On-prem eliminates data transfer costs and can provide lower latency for data access within the local network. However, it may limit ability to integrate with cloud-based services [10]. Hybrid model will incur data transfer costs and latency issues. If deployment is completely cloud-based data transfer and latency costs may be minimized.

Data Modeling & Design

As a Data Engineering Program Manager, grasping the core concepts of data modeling and design is crucial for leading successful project as it equips to make informed decisions and communicate effectively.

Data Modeling Basics

Data modeling is the process of creating a visual representation of the structure and relationships within a set of data. It involves defining how data is organized, structured, and related within a system, and depicting it visually. The primary goal of data modeling is to provide a clear and concise blueprint that provides consistent information to database developers, analysts, and stakeholders alike [11]. It's akin to planning a city's layout, ensuring easy access, efficiency, and flexibility.

Key Data Modeling Terminologies

- **Entities:** Real-world objects or concepts represented in the system (e.g., customers, products, orders).
- **Attributes:** Characteristics of entities (e.g., customer name, product price, order date).
- **Relationships:** Connections between entities (e.g., a customer places an order for a product).
- **Primary Key:** A unique identifier for each record in a table to ensure data integrity.
- **Secondary Key:** An attribute in one table that refers to the primary key in another table, used to link tables.
- Normalization: Organizing data to minimize redundancy and ensure data integrity.
- **Denormalization:** Organizing data to optimize query performance.

Types of Data Models

- **Conceptual Data Model:** helps stakeholders understand domain by represents high-level concepts and relationships between objects. It focuses on what data should be stored and does not concern itself with implementation details. Ex. The below diagram shows a conceptual data model with 3 entities where a customer can place multiple orders and each order contains multiple products.

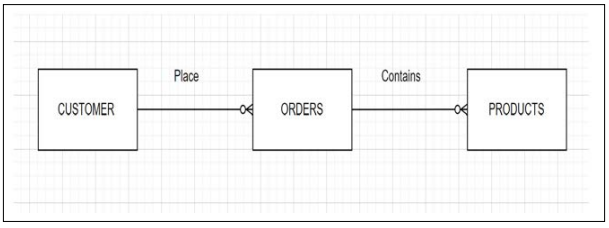

Figure 1: Example of a Conceptual Data Model

- **Logical Data Model:** represents a structure that can be implemented in a database management system (DBMS). It includes entities, attributes, relationships, and contracts as highlighted in the example below.

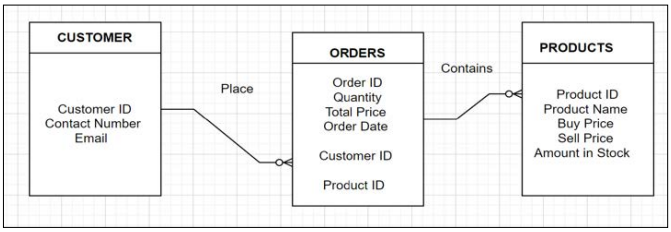

Figure 2: Example of a Logical Data Model

- **Physical Data Model:** provides the physical implementation details of database by specifying technical aspects such as tables, columns, data types, indexes etc. required for database administrators and developers to build, optimize, and maintain the physical database. The example below is a physical data model for the customer, order products scenario.

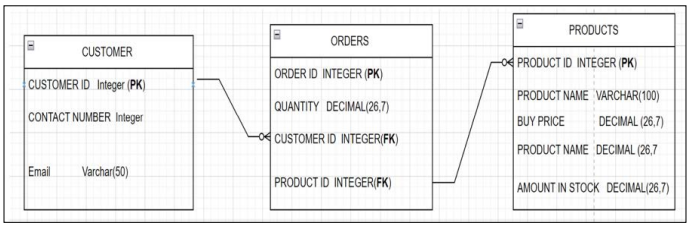

Figure 3: Example of a Physical Data Model

Data Modelling Techniques

Entity-Relationship Diagram (ERD)

Visualizes entities, attributes, and relationships using symbols and lines.

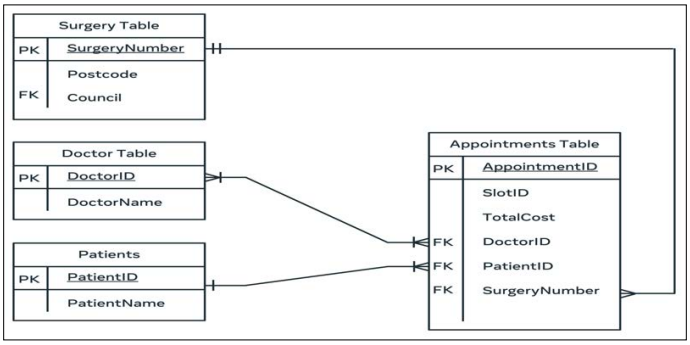

Figure 4: Example of an Entity-Relationship Diagram

Dimensional Modeling

Optimizes data for data warehousing and analytics, focusing on facts and dimensions.

Popular Data Modelling Tools

Erwin Data Modeler, IBM Infosphere Data Architect, MS Visio, SAP PowerDesigner, Oracle SQL Developer Data Modeler, Toad, Lucid Chart:

Knowledge of data modeling concepts helps program managers understand how data modeling choices affect data quality, security, and scalability and help determine the potential modeling complexities impacting project timelines and costs [12].

Data Warehouse, Data Lake and Data Mesh

Data Warehouse, Data Lake, and Data Mesh are all concepts related to storing, managing, and utilizing large amounts of data within an organization. However, they have distinct characteristics and serve different purposes as highlighted below.

Data Warehouse

A data warehouse is a centralized repository that *stores structured, cleaned, and integrated data* from various sources. It is *Designed to support analytics and business intelligence (BI)* by providing a single source of truth for reporting and decision-making. It is typically organized in a *schema-on-write approach*, meaning the data is transformed and structured before being loaded into the warehouse.

Examples: Snowflake, Microsoft Azure Synapse Analytics, Google Big Query, Oracle Exadata, Amazon Redshift.

Data Lake

A data lake is a centralized repository that stores large volumes of *raw, unstructured, and semi-structured data* in its native format. It is Designed to be a flexible and scalable storage solution for big data, allowing organizations to store data from diverse sources without the need for upfront data modeling. It follows a schema-on-read approach, where data is stored in its original format, and the schema is applied only when the data is read or analyzed.

Examples: Apache Hadoop, Amazon S3, Microsoft Azure Data Lake Storage, Google Cloud Storage.

Data Mesh

A Data Mesh is an architectural approach that aims to *decentralize data ownership and management* across an organization. It *advocates for a domain-driven design*, where data is treated as a product and owned by the teams closest to the data (domain teams). Data Mesh *emphasizes self-serve data infrastructure*, enabling domain teams to independently manage their data while adhering to global governance and interoperability standards [13]

Examples: Apache Kafka, Apache Pulsar, Confluent Platform, Amazon MSK, Google Pub/Sub.

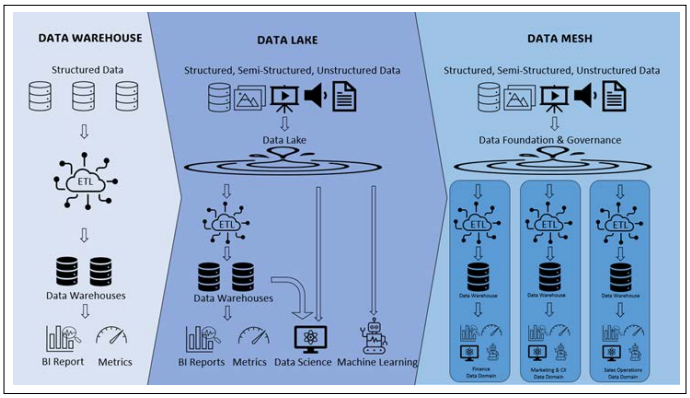

Figure 5: Comparing Data Warehouse, Data Lake & Data Mesh

Understanding the differences and nuances of Data Warehouse, Data Lake, and Data Mesh would help a Data Engineering Program Manager to make establish clear data ownership and foster collaboration between domain teams in a Data Mesh architecture ensuring alignment of business goals with the corresponding data initiatives across the organization [14].

Data Warehouse Design

Data Warehouses are specialized relational databases designed and optimized for analysis and reporting of large volumes of data. Star Schema and Snowflake Schema are two common approaches to designing the logical structure of a data warehouse.

Star Schema

Star Schema is a simple and intuitive design that consists of a central fact table surrounded by dimension tables. The fact table contains the main metrics or measures of interest, such as sales amount or quantity, and foreign keys to the dimension tables. Dimension tables store the descriptive attributes related to the facts, such as customer details, product information, or time periods. The star schema is named after its resemblance to a star shape, with the fact table at the center and dimension tables radiating outward.

Example: Consider a retail sales data warehouse with a star schema design with BOOKINGS as the Fact Table and SALES, PRODUCTS, ORDERS, TIME as Dimension Tables. The BOOKINGS table contains the Sales bookings transactions with foreign keys to each of the dimension tables. The dimension tables provide additional details about each of the dimensions as shown in figure 6.

Snowflake Schema

Snowflake Schema is an extension of the Star Schema that normalizes the dimension tables further to reduce redundancy.

In a Snowflake Schema, dimension tables are split into multiple related tables based on their hierarchical relationships or dependencies. The main dimension tables are connected to sub-dimension tables, forming a snowflake-like structure [15].

Snowflake Schema can improve data integrity and reduce data redundancy but may require more complex queries and joins compared to Star Schema.

Example: To illustrate the snowflake schema, let's consider the previous retail sales retail sales data warehouse example with BOOKINGS as the Fact Table and SALES, PRODUCTS, ORDERS, TIME as Dimension Tables and extend it to Snowflake schema where the PRODUCT dimension is normalized into PRODUCTS and CATEGORY and the TIME dimension is normalized into YEAR and QUARTER tables.

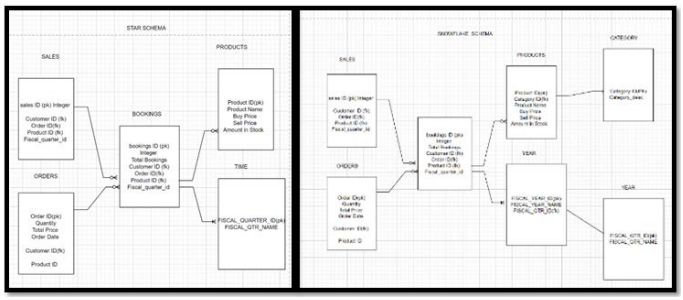

Figure 6: STAR Schema vs. SNOWFLAKE Schema

The choice between Star Schema and Snowflake Schema depends on factors such as data complexity, query performance

requirements, and the level of data redundancy acceptable in the data warehouse. Star Schema is simpler and often preferred for its ease of use and query performance, while Snowflake Schema provides a more normalized structure and can be beneficial when dealing with complex hierarchical relationships or when data integrity is a primary concern.

Data Extraction, Transformation and Loading (ETL) Pipelines and Processes

ETL Pipelines: Understand different Stages of Ingestion, Transformation and Loading

The ETL process is a crucial component of data integration and data warehousing workflows for decades. By developing a solid understanding of ETL processes, Data Program Managers can effectively manage any data integration projects and empower themselves to make informed decisions regarding data architecture and technology choices.

ETL involves three main steps:

- **Extract:** In this step, data is read and retrieved from various sources, such as databases, flat files, APIs etc.
- **Transform:** This step involves cleaning, formatting, restructuring, enriching the data to meet the requirements of the target system or data warehouse. It may include data validation, filtering, sorting, aggregating, joining, and applying business rules or calculations.
- **Load:** In the final step, the transformed data is loaded into the desired target data warehouse or, data mart.

The ETL process is commonly used in scenarios where organizational data needs to be consolidated from multiple sources, transformed to meet specific requirements, and loaded into a centralized repository for reporting, analysis, or further processing [16].

ETL vs. ELT (Extract, Load, Transform)

While the ETL process follows the traditional Extract, Transform, Load sequence, the ELT approach reverses the order of the last two steps. The key difference between ETL and ELT lies in when and where the transformation step occurs. ETL performs transformations before loading the data into the target system, typically in a staging area, while ELT loads the raw data first and then transforms it within the target system.

Table 2: Comparison between ETL and ELT

Comparison Parameter	ETL	ELT
Best Suited	when the transformation logic is complex, and data needs cleansing before loading.	for handling large volumes of data or data with high velocity.
Agility	less Agile, as any change in transformation logic requires entire ETL process to be updated.	more agility, as transformations can be modified without impacting extraction & loading process
Skillset and Tools	requires specialized ETL tools and expertise in transformation logic and data modeling	leverages the processing power and built-in transformation capabilities of the target system

Top ETL Tools

There are various ETL tools available in the market, each with its own strengths and capabilities. Here are top 5 tools:

- **Talend Data Integration:** An open-source ETL tool with a comprehensive set of data integration capabilities, including data profiling, cleansing, and transformation.
- **Informatica PowerCenter:** A widely used commercial ETL tool known for its scalability, performance, and advanced data integration features.
- **AWS Glue:** A serverless data integration service provided by Amazon Web Services (AWS) that simplifies the ETL process for cloud-based data sources and data lakes.
- **Stitch Data Loader:** A cloud based ETL platform that simplifies the process of extracting data from various sources and loading it into data warehouses or data lakes.
- **Apache Nifi:** An open-source, web-based tool for data integration and automation, with a focus on real-time data flows and streaming data processing.

Pipeline Data Processing: Batch vs. Streaming

When it comes to pipeline data processing in the context of ETL workflows, both batch and streaming processing approaches can be used. The choice between batch and streaming ETL processing depends on the specific requirements of the data pipeline, such as the need for real-time insights, the volume and velocity of data, and the complexity of the transformations required.

Batch Data Processing

Batch data processing involves extracting data from source systems, transforming it, and loading it into a target system in batches at scheduled intervals. The data is typically collected over a period, such as hourly, daily, or weekly, and processed together in a single batch. Batch processing is suitable for scenarios where data freshness is not a critical requirement, and the data can be processed periodically. Batch Processing is commonly used in the following scenarios:

- **Data Warehousing:** To populate data warehouses by extracting data from various source systems, transforming it into a consistent format, and loading it into the data warehouse for analysis and reporting.
- **Historical Data Loading:** To load large volumes of historical data into a target system, such as migrating data from legacy systems to a new platform.
- **Periodic Data Synchronization:** To synchronize data between different systems or databases on a regular basis, ensuring data consistency across multiple platforms.

Since data is processed in batches, insights and actions based on the processed data are delayed until the next batch processing cycle. Therefore, Batch Processing may not be suitable for applications that require real-time or near-real-time data processing. For real-time data processing, Stream Processing would be ideal.

Streaming Data Processing

Streaming data processing involves continuously extracting, transforming, and loading data as it arrives in real-time or near-real-time. Data is processed individually or in small batches (micro-batches) as soon as it is generated or received from the source systems allowing for immediate insights and actions based on the processed data.

Streaming Data Processing is commonly used in the following scenarios:

- **Real-time Analytics:** real-time analytics scenarios, such as monitoring and analyzing sensor data, user behavior, or social media feeds in real-time, enabling prompt decision-making and responsiveness [17].
- **Fraud Detection:** Streaming Processing is employed in fraud detection systems to identify and respond to fraudulent activities as they occur, such as in financial transactions or network security.
- **Real-time Data Integration:** Streaming Processing enables real-time data integration between different systems, allowing for seamless data flow and up-to-date information across the organization.

Streaming processing systems can be complex to design, implement, and maintain due to the real-time nature of data processing and need for fault-tolerant and scalable architectures. Ensuring data consistency and handling out-of-order or delayed data can also be challenging in streaming scenarios, especially when dealing with multiple data sources.

Kappa and Lambda Architectures

Kappa architecture is a stream-based data processing architecture that treats all data as a stream. It eliminates the need for separate batch and streaming processing layers by using a single stream processing engine for both real-time and historical data processing.

Lambda architecture combines both batch and streaming processing approaches. It consists of three layers:

1. the batch layer for processing historical data,
2. the speed layer for real-time processing of incoming data, and
3. the serving layer for querying and exposing the results.

Kappa architecture simplifies the data processing architecture by using a single stream processing engine, while Lambda architecture provides a hybrid approach that combines batch and streaming processing for comprehensive data processing and analysis. Ultimately, for Data Engineering Program Managers understanding the characteristics, use cases, and limitations of batch and streaming ETL processing, along with the concepts of Kappa and Lambda architectures, helps in designing and implementing data pipelines that align with the specific needs of the organization.

Pipelines Automation & Orchestration: Illustration through Apache Airflow

The process of automating and managing the execution of ETL workflows ensuring smooth and efficient data processing involves defining dependencies, scheduling tasks, handling failures, and monitoring of the overall pipeline.

Apache Airflow is an open-source platform for programmatically authoring, scheduling, and monitoring workflows. It provides a way to define ETL pipelines as code, allowing for version control, collaboration, and easy maintenance [18]. Let's consider an example where we have an ETL pipeline that extracts data from a database, performs transformations, and loads the transformed data into a data warehouse. ETL Pipeline can be automated and orchestrated using Airflow following the below steps:

Define the DAG (Directed Acyclic Graph)

In Airflow, workflows are defined as DAGs, which represent a collection of tasks and their dependencies. Each task corresponds to a specific operation. Dependencies between tasks are defined to ensure the proper execution order.

Define the Tasks

Each task in the DAG represents a specific operation in the ETL pipeline. For a simple ETL pipeline, we have three tasks: extract_data, transform_data, and load_data. Each task is defined using an operator, such as the PythonOperator, which allows executing Python functions.

Set Dependencies

Dependencies between tasks are defined using the >> operator in Airflow. For example, extract_task can be set as a dependency for transform_task, and transform_task can be set as a dependency for load_task. This ensures that the tasks are executed in the correct order: extraction, transformation, and loading.

Schedule the Pipeline

Airflow allows scheduling the pipeline to run at specific intervals or based on certain conditions. For example, if the pipeline needs to be run daily, the schedule_interval parameter can be set to timedelta (days=1)

Monitor and Manage the Pipeline

Airflow provides a web-based user interface for monitoring and managing the pipeline execution. It allows tracking the status of tasks, handling failures, and visualizing the pipeline's progress. Airflow also provides features like email notifications, logging, and error handling to ensure the pipeline runs smoothly.

Airflow's modular architecture, extensive set of operators and hooks, and the ability to define workflows as code make it a powerful tool for ETL pipeline automation and orchestration. By using Airflow for ETL pipeline automation and orchestration, Data Engineering Program Managers can enable efficient and reliable data processing, reducing manual intervention and promote code reusability, version control, and collaboration among team members.

Big Data Technologies

Big Data refers to extremely large and complex datasets that are difficult to process and analyze using traditional data processing tools and techniques. Big Data is characterized by the "*5 V's*": *Volume* (large amounts of data), *Velocity* (high speed of data generation and processing), *Variety* (diverse types and sources of data), *Veracity* (data accuracy and reliability), and *Value* (extracting meaningful insights from the data). Hadoop and Spark are two popular frameworks used for Big Data processing.

Hadoop & Spark

Hadoop is an open-source framework designed for distributed storage and processing of large datasets across clusters of computers. It consists of two main components: Hadoop Distributed File System (HDFS) for storage and *MapReduce* for processing. Hadoop allows for fault-tolerant and scalable data processing by distributing data and computations across multiple nodes in a cluster. It is suitable for batch processing and can handle structured, semi-structured, and unstructured data [19]. Hadoop ecosystem includes various tools and libraries like Hive, Pig, HBase, and Mahout for different Big Data tasks.

Key Benefits of Hadoop

- **Scalability:** Hadoop can scale horizontally by adding more nodes to the cluster, allowing it to handle massive datasets.
- **Fault Tolerance:** Hadoop ensures data reliability by replicating data across multiple nodes, making it resilient to hardware failures.
- **Cost-Effective:** Hadoop runs on commodity hardware,

making it cost-effective compared to traditional data processing.
- **Flexibility:** Hadoop can handle various types of data, including structured, semi-structured, and unstructured data.

Spark is an open-source distributed computing framework designed for fast and efficient data processing. It provides a unified engine for batch processing, real-time streaming, machine learning, and graph processing. Spark uses *in-memory* computation and a directed acyclic graph (*DAG*) execution engine, which enables it to process data much faster than Hadoop's MapReduce [20]. It offers APIs in multiple programming languages, including Scala, Java, Python, and R. Spark ecosystem includes libraries like Spark SQL, Spark Streaming, MLlib (machine learning), and GraphX (graph processing).

Key Benefits of Spark
- **Speed:** Spark's in-memory computation and optimized execution engine make it significantly faster than Hadoop for many data processing tasks.
- **Ease of Use:** Spark provides a simple and expressive API, making it easier to write and maintain complex data processing workflows.
- **Real-Time Processing:** Spark Streaming enables real-time data processing, allowing for low-latency applications.
- **Machine Learning:** MLlib, Spark's machine learning library, offers a wide range of algorithms for data analysis and predictive modeling.
- **Integration:** Spark can seamlessly integrate with various data sources, including Hadoop, databases, and cloud systems

Both Hadoop and Spark have their strengths and are widely used in the Big Data ecosystem. Hadoop is known for its scalability and fault tolerance, while Spark excels in fast data processing, real-time streaming, and machine learning. The differences are highlighted in the below table:

Table 3: Hadoop vs. Spark

Feature	Hadoop	Spark
Processing Model	Batch processing using MapReduce	Batch processing, real-time streaming, machine learning, graph processing.
Speed	Slower due to disk I/O and MapReduce overhead	Faster due to in-memory computation and optimized execution engine
Ease of Use	Requires writing MapReduce jobs, which can be complex	Provides a simple and expressive API, making it easier to use
Machine Learning	Mahout library for machine learning, but limited compared to Spark	MLlib offers a wide range of machine learning algorithms
Ecosystem	Larger ecosystem with various tools and libraries	Growing ecosystem with a focus on data processing and analytics

The Data Engineering Program Managers should decide between choosing Hadoop and Spark, depending on the specific requirements of their Big Data project, considering the type of ecosystem, processing needs, and desired performance.

Cloud-Based Big Data Solutions

Managed Big Data services, such as Amazon EMR, Google Cloud Dataproc, and Azure HDInsight, are cloud-based offerings that simplify the deployment, management, and scaling of Hadoop and Spark clusters. These services are designed to abstract away the complexities of setting up and maintaining the underlying infrastructure, allowing Data Engineering Program Managers to drive the data processing and analytics tasks.

Amazon EMR (Elastic MapReduce)

Amazon EMR is a fully managed Big Data platform that makes it easy to process and analyze vast amounts of data using. Hadoop and Spark frameworks. It provides a scalable and flexible environment for running distributed data processing jobs, interactive querying, and machine learning workloads.EMR takes care of provisioning and configuring the underlying EC2 instances, installing and managing Hadoop and Spark software, and handling cluster scaling and fault tolerance. It integrates seamlessly with other AWS services, such as Amazon S3 for data storage, Amazon Kinesis for real-time data streaming, and Amazon Redshift for data warehousing. EMR supports a wide range of Hadoop and Spark ecosystem tools, including Hive, Pig, HBase, Presto, and Zeppelin, enabling diverse data processing and analytics use cases.

Google Cloud Dataproc

Google Cloud Dataproc is a fully managed Hadoop and Spark service that allows for fast and cost-effective processing of large datasets. It enables the creation of Hadoop and Spark clusters in a matter of seconds, with automatic configuration and management of the underlying infrastructure. Dataproc integrates with other Google Cloud services, such as Google Cloud Storage for data storage, BigQuery for data warehousing, and Pub/Sub for real-time data ingestion. It provides a familiar Hadoop and Spark ecosystem experience, supporting tools like Hive, Pig, and Jupyter Notebooks for data processing and analysis. Dataproc offers flexibility in terms of cluster sizing, configuration, and scaling, allowing Data Engineering Program Managers to optimize performance and costs based on workload requirements.

Azure HDInsight

Azure HDInsight is a fully managed, open-source analytics service for processing large datasets using Hadoop and Spark clusters. HDInsight integrates with Azure storage services, such as Azure Blob Storage and Azure Data Lake Storage, for seamless data storage and access. It provides a user-friendly interface for cluster creation, management, and monitoring, along with integration with Azure Active Directory for secure access control. HDInsight offers enterprise-grade security features, such as network isolation, encryption, and integration with Azure Virtual Networks, ensuring data protection and compliance. It supports a wide range of Hadoop and Spark components, including Hive, Pig, Spark SQL, and Spark Streaming, enabling diverse data processing scenarios.

Key Benefits of Managed Hadoop and Spark Services
- Simplified Deployment and Management: It eliminate the need for manual setup, configuration, and management of the underlying infrastructure. Data Engineering Program Managers can quickly provision clusters and focus on writing and executing data processing jobs, rather than worrying about the underlying complexities.
- **Scalability and Flexibility:** It allows for easy scaling of Hadoop and Spark clusters based on data processing requirements. Data Engineering Program Managers can

dynamically adjust the number and size of nodes in the cluster to handle varying workloads, ensuring optimal performance and cost efficiency.

- **Integration with Cloud Ecosystem:** It helps seamlessly integrate with other cloud services, such as storage, data warehousing, and real-time data ingestion, enabling setting up of end-to-end data processing pipelines and leveraging the full potential of the cloud ecosystem.
- **Cost Optimization:** Managed services provide cost-effective options for running Hadoop and Spark clusters, with the ability to pay only for the resources used. Program Managers can optimize costs by leveraging features like autoscaling, spot instances, and transient clusters, which automatically adjust resources based on workload demands.
- **Focused on Data Processing and Analytics:** By offloading the infrastructure management to the cloud provider, Data Engineering Program Managers can dedicate more time and resources to data processing and analytics. This allows for faster time-to-market, improved data-driven decision-making, and accelerated innovation within the organization.

Managed Hadoop and Spark services like Amazon EMR, Google Cloud Dataproc, and Azure HDInsight empower Data Engineering Program Managers to harness the power of Big Data processing and analytics in a simplified and cost-effective manner. By leveraging these services, program managers can focus on delivering value through data-driven initiatives, while the cloud provider takes care of the underlying infrastructure and management complexities.

NoSQL Databases for Big Data

NoSQL databases are non-relational databases designed to handle large volumes of unstructured, semi-structured, and structured data. Unlike traditional relational SQL databases, NoSQL databases provide flexible schemas, horizontal scalability, and high performance for handling Big Data workloads. They are particularly useful in scenarios where data is rapidly changing, unstructured, and requires real-time processing. The two most popular NoSQL databases are Cassandra and MongoDB

Apache Cassandra

Cassandra is a highly scalable, distributed NoSQL database designed to handle large amounts of structured data across multiple commodity servers. It provides high availability and fault tolerance through its distributed architecture, with no single point of failure. Cassandra uses a wide-column data model, where data is stored in tables with rows and columns, but the *columns can vary per row*. It offers linear scalability, allowing seamless addition of new nodes to the cluster to handle increased data volume and throughput. Cassandra *excels in handling time-series data*, such as sensor data, logs, and metrics, making it suitable for IoT, real-time analytics, and monitoring use cases [21].

Key Benefits of Cassandra

- **Real-Time Data Ingestion and Processing:** Cassandra can handle high-velocity data streams and perform real-time analytics on the ingested data.
- **Scalable Event Logging and Monitoring:** Cassandra's distributed architecture makes it suitable for storing and analyzing large volumes of event logs and metrics.
- **Real-Time Recommendation Engines:** Cassandra can store user profiles, preferences, and interaction data to power real-time recommendation systems.

Key Limitations of Cassandra

- **Limited Support for Complex Queries:** Cassandra's query language (CQL) has limited support for complex joins and aggregations compared to SQL databases.
- **Eventual Consistency:** Cassandra prioritizes availability and partition tolerance over strong consistency, which means there may be a slight delay in data consistency across nodes.
- **Lack of ACID Transactions:** Cassandra does not support full ACID (Atomicity, Consistency, Isolation, Durability) transactions, which can be a limitation for certain use cases requiring strict data consistency.

MongoDB

MongoDB is a document-oriented NoSQL database that stores data in flexible, JSON-like documents called BSON (Binary JSON). It provides a dynamic schema, allowing for easy modification and evolution of data structures without requiring predefined schemas. MongoDB supports rich queries, indexing, and aggregation, enabling complex data retrieval and analysis. It *offers horizontal scalability* through sharding, distributing data across multiple servers to handle large data volumes and high read/write throughput. MongoDB provides *automatic failover and data replication*, ensuring high availability and data durability [22].

Key Benefits of MongoDB

- **Content Management Systems:** MongoDB's flexible document model makes it suitable for storing and managing unstructured content, such as articles, videos, and user-generated content.
- **Real-Time Analytics:** MongoDB's aggregation framework and indexing capabilities enable real-time analytics on large datasets.
- **Mobile and Web Applications:** MongoDB's flexible schema and scalability make it a good fit for developing mobile and web applications with evolving data models.

Key Limitations of MongoDB

- **Lack of Strong Consistency:** MongoDB prioritizes availability and partition tolerance, which means there may be a slight delay in data consistency across replicas.
- **Limited Support for Transactions:** While MongoDB supports multi-document ACID transactions starting from version 4.0, it may not be as mature as traditional SQL databases in terms of transactional capabilities.
- **Memory Usage:** MongoDB's in-memory working set can consume a significant amount of RAM, requiring careful capacity planning and monitoring.

From a Data Engineering Program Manager's perspective, understanding the strengths and limitations of NoSQL databases like Cassandra and MongoDB is crucial for making informed decisions about data storage and processing architectures. NoSQL databases offer scalability, flexibility, and high performance for handling Big Data workloads, but they may not be suitable for all use cases, especially those requiring strong consistency, complex transactions, or strict ACID properties.

When considering NoSQL databases, Data Engineering Program Managers should evaluate factors such as data structure, scalability requirements, query complexity, consistency needs, and integration with existing data processing workflows. It's essential to align the choice of NoSQL database with the specific requirements of the Big Data initiative and ensure that the selected database can effectively support the desired use cases while meeting

performance, scalability, and reliability expectations.

Additionally, Data Engineering Program Managers should consider the skills and expertise of the team, as well as the availability of resources and support for the selected NoSQL database. Proper training, documentation, and best practices should be established to ensure the effective utilization and management of NoSQL databases within the organization's Big Data ecosystem.

Data Governance and Security

A Data Governance Framework involves a set of policies, procedures, roles, and responsibilities that ensure the proper management, security, quality, and use of an organization's data assets. It provides a structured approach to defining, implementing, and maintaining data governance practices throughout the data lifecycle. For a Data Engineering Program Manager, understanding and implementing a Data Governance Framework is crucial for ensuring data integrity, compliance, and effective data management [23]. The top 3 Areas of Data Governance are given below:

Data Audit and Quality Management

Data Audit and Quality Management are critical components of a comprehensive data governance framework. They involve processes, practices, and tools to ensure the accuracy, completeness, consistency, and reliability of an organization's data assets. Data Engineering Program Managers need to be well-versed in the below aspects of Data Audit and Quality Management to ensure the integrity and usability of data for downstream applications and decision-making.

- **Data Quality Assessment:** Conduct regular data quality assessments to identify data quality issues, such as missing values, duplicates, inconsistencies, and use data profiling techniques to analyze data patterns and distributions.
- **Data Quality Rules and Standards:** Establish data validation rules to ensure data conforms to defined formats, ranges, and create data quality scorecards or dashboards to measure and monitor data quality metrics against defined standards.
- **Data Cleansing and Transformation:** Implement and automate data cleansing processes to identify and rectify data quality issues, such as removing duplicates, correcting inconsistencies, and standardizing formats.
- **Data Lineage and Traceability:** Implement data lineage tracking that documents the flow of data from source to target systems, including any transformations applied, and enable traceability of data quality issues to their root causes by mapping data lineage and dependencies.
- **Data Validation and Testing:** Develop and execute data validation tests to verify the accuracy, completeness, and consistency of data at various stages of the data pipeline. Implement data quality checks and controls at all points.
- **Data Quality Monitoring and Alerting:** Set up data quality monitoring dashboards and alerts to proactively identify and address data quality anomalies. Use statistical process control techniques to monitor data quality metrics over time
- **Data Quality Reporting and Communication:** Generate regular data quality reports to communicate data quality findings and actions to relevant stakeholders, including business users, data stewards, and executive sponsors.

Data Engineering Program Managers should work closely with data governance teams, data stewards, and business stakeholders to establish and maintain robust Data Audit and Quality Management practices and should continuously assess and optimize data quality processes to meet evolving business needs.

Version Control

Version control is a critical aspect of data governance that ensures the integrity, traceability, and reproducibility of data assets and related artifacts. It involves tracking and managing changes to data, code, and documentation over time. GitHub, a popular web-based platform built on top of Git, provides a powerful version control system that can be leveraged for data governance purposes [24]. GitHub allows for version control in data governance through:

- **Data and Code Versioning:** GitHub allows to store and version data files, such as CSV, JSON, or Parquet files, along with the code that processes and analyzes the files. Each version of a data file is tracked through commits, which are snapshots of the repository at a specific point in time. With GitHub, changes made to data files can be tracked including who made the changes, when they were made, and what specifically was modified.
- **Data Pipeline Versioning:** GitHub can be used to version control data pipeline code, including scripts, configuration files, and workflow definitions. By storing data pipeline code in GitHub repositories, a single source of truth for data pipelines can be maintained ensuring transparency and reliability.
- **Documentation and Metadata Management:** GitHub provides features such as readme files and data dictionaries for storing and versioning documentation and metadata related to the data assets.
- **Collaboration and Access Control:** GitHub facilitates collaboration among team members through features like pull requests, code reviews, and issue tracking. Access control can be managed through GitHub's user roles and permissions, ensuring that only authorized individuals can view, modify, or contribute to specific data repositories.
- **Branching and Merging:** GitHub's branching and merging capabilities allow for parallel development and experimentation without affecting the main data or code repository. Branches can be created to work on new features, bug fixes, or data updates independently. This enables a controlled approach to making changes to data pipelines.
- **Auditing and Traceability:** GitHub provides a detailed audit trail of all activities and changes made to data and code repositories. Each commit in GitHub includes information about the author, timestamp, and a description of the changes made, allowing for traceability and accountability.
- **Integration with Data Governance Tools:** GitHub can be integrated with other data governance tools and platforms to extend its capabilities. For example, you can integrate GitHub with data cataloging tools to automatically extract and synchronize metadata from data files stored in GitHub repositories.

Implementing version control with GitHub as part of the data governance strategy helps Data Engineering Program Managers in ensuring the integrity, reliability, and transparency of data assets and provides a structured approach to managing changes, collaborating with team members, and maintaining a single source of truth for the data ecosystem.

Data Security, Access Control, and Compliance

Data Security, Access Control, and Compliance are the final critical component of a comprehensive Data Governance framework. They

focus on protecting sensitive data, ensuring authorized access, and adhering to legal and regulatory requirements. As a Data Engineering Program Manager, understanding these concepts is crucial to safeguarding data assets and maintaining the trust of stakeholders.

Data Security

Data security involves protecting data from unauthorized access, theft, misuse, and damage. It includes implementing technical, administrative, and physical controls to ensure the confidentiality, integrity, and availability of data.

Key aspects of Data Security Include:

- **Encryption:** Encrypting sensitive data both at rest and in transit to protect it from unauthorized access.
- **Access Controls:** Implementing stringent access controls to ensure that only authorized individuals can access specific data assets.
- **Network Security:** Securing the network infrastructure to prevent unauthorized intrusions and data breaches.
- **Data Backup and Recovery:** Regularly backing up data and having robust disaster recovery plans to protect against data loss.

Access Control

Access control involves managing and restricting access to data assets based on the principle of least privilege. It ensures that individuals can only access the data they need to perform their job functions and prevents unauthorized access.

Key Aspects of Access Control Include:

- **Role-Based Access Control (RBAC):** Defining user roles and permissions based on job functions and granting access accordingly.
- **Authentication and Authorization:** Implementing strong authentication mechanisms (e.g., multi-factor authentication) and authorization processes to verify user identities and permissions.
- **Auditing and Monitoring:** Regularly auditing and monitoring user access to detect and investigate any suspicious activities or access violations.
- **Data Masking and Anonymization:** Applying data masking or anonymization techniques to protect sensitive data while allowing authorized access for specific purposes.

Compliance

Compliance refers to adhering to legal, regulatory, and industry-specific requirements related to data protection and privacy. It involves understanding and implementing the necessary controls and processes to meet compliance obligations.

Key Compliance Frameworks and Regulations Include:

- **GDPR (General Data Protection Regulation):** A European Union regulation that sets strict requirements for the collection, processing, and protection of personal data of members of the European Union [25].
- **SOX (Sarbanes-Oxley Act):** A U.S. federal law that establishes requirements for financial reporting and internal controls. It includes provisions related to data accuracy, integrity, and retention.
- **HIPAA (Health Insurance Portability and Accountability Act):** A U.S. law that sets standards for protecting sensitive patient health information. It enables appropriate strict access controls for healthcare data.
- **PCI DSS (Payment Card Industry Data Security Standard):** A set of security standards for organizations that handle credit card data. It mandates specific security controls and practices to protect cardholder information.

By prioritizing Data Security, Access Control, and Compliance as part of the Data Governance framework, Data Engineering Program Managers can protect sensitive data, maintain the trust of stakeholders, and avoid costly legal and reputational consequences. It is essential to collaborate closely with cross-functional teams and stakeholders to implement robust security measures, access controls, and compliance processes throughout the data lifecycle.

Conclusion

In today's data-driven world, the role of data engineering program managers is increasingly critical. By understanding and applying the seven essential data engineering concepts discussed in this article, program managers can effectively lead and manage data engineering initiatives. From data architecture and storage to data governance and security, each concept plays a vital role in ensuring the success of data-driven projects.

Program managers who are well-versed in these concepts can make informed decisions, communicate effectively with stakeholders, and collaborate seamlessly with data engineering teams. They can ensure that data infrastructure aligns with business objectives, data quality is maintained, and data security and compliance requirements are met. Moreover, staying current with the latest trends and best practices in data engineering is crucial for program managers. As big data technologies continue to evolve, program managers must adapt and leverage new tools and techniques to optimize data processing, storage, and analysis.

By combining a strong understanding of data engineering concepts with effective project management skills, program managers can drive innovation, improve decision-making, and create tangible business value through data-driven initiatives. They can help their organizations harness the power of data, gain competitive advantages, and achieve strategic goals. In conclusion, data engineering program managers play a pivotal role in bridging the gap between technical expertise and business objectives. By mastering the essential data engineering concepts and applying them effectively, program managers can lead their organizations towards a data-driven future, unlocking the full potential of data and driving meaningful insights and outcomes.

References

1. Elmasri Ramez A, Shamkant B Navathe (2017) Database System Concepts and Architecture. Fundamentals of Database Systems, Pearson, Boston i Pozostałe https://asolanki.co.in/wp-content/uploads/2019/02/Fundamentals_of_Database_Systems_6th_Edition-1.pdf.
2. Sadalage Pramod J, Martin Fowler (2012) NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley Professional https://bigdata-ir.com/wp-content/uploads/2017/04/NoSQL-Distilled.pdf.
3. Zhang H, Chen G, Ooi BC, Tan KL, Zhang M (2015) In-memory big data management and processing: A survey. IEEE Transactions on Knowledge and Data Engineering 27: 1920-1948.
4. Jensen SK, Pedersen TB, Thomsen C (2017) Time series management systems: A survey. IEEE Transactions on Knowledge and Data Engineering 29: 2581-2600.
5. Johnson J, Douze M, Jégou H (2021) Billion-scale similarity

search with GPUs. IEEE Transactions on Big Data 7: 535-547.
6. Beaulieu A (2020) Learning SQL: Generate, Manipulate, and Retrieve Data. O'Reilly Media https://www.oreilly.com/library/view/learning-sql-3rd/9781492057604/.
7. Wes McKinney (2017) Python for Data Analysis. Data Wrangling with Pandas, NumPy, and IPython https://www.oreilly.com/library/view/python-for-data/9781491957653/.
8. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, et al. (2010) A view of cloud computing. Communications of the ACM 53: 50-58.
9. Kilcioglu C, Rao JM (2016) Competition on price and quality in cloud computing. In Proceedings of the 25th International Conference on World Wide Web 1123-1132.
10. Data Transfer Service. AWS Datasync – AWS https://aws.amazon.com/datasync/.
11. Hoberman Steve (20017) Data Modeling Made Simple: A Practical Guide for Business & IT Professionals. Technics Publications https://technicspub.com/data-modeling-made-simple/.
12. Database Design Tool. Lucidchart https://www.lucidchart.com/pages/examples/database-design-tool.
13. Dehghani Zhamak (2019) How to Move beyond a Monolithic Data Lake to a Distributed Data Mesh. Martinfowler.Com https://martinfowler.com/articles/data-monolith-to-mesh.html.
14. Jeffrey Richman (2023) Top 6 Data Mesh Tools and Companies. Estuary https://estuary.dev/data-mesh-tools/#databricks.
15. Levene Mark, George Loizou (2003) Why Is the Snowflake Schema A Good Data Warehouse Design?. Information Systems 28: 225-240.
16. Kimball Ralph, Joe Caserta (2009) The Data Warehouse ETL Toolkit Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. Wiley https://nibmehub.com/opac-service/pdf/read/The%20Data%20Warehouse%20ETL%20Toolkit%20_%20Practical%20Techniques%20for%20Extracting-%20Cleaning-.pdf.
17. Reeve April (2013) Data warehousing with real-time updates. Managing Data in Motion 113-117.
18. Apache Airflow. Documentation https://airflow.apache.org/docs/.
19. White Tom (2015) Hadoop: The Definitive Guide. 4th Edition. O'Reilly Media https://www.oreilly.com/library/view/hadoop-the-definitive/9781491901687/.
20. Chambers Bill, Matei Zaharia (2018) Spark: The Definitive Guide: Big Data Processing Made Simple. O'Reilly https://www.oreilly.com/library/view/spark-the-definitive/9781491912201/copyright-page01.html.
21. Carpenter Jeff, Eben Hewitt (2020) Cassandra: The Definitive Guide, 3rd Edition. O'Reilly Media, Inc https://www.oreilly.com/library/view/cassandra-the-definitive/9781098115159/.
22. Chodorow Kristina (2013) MongoDB the Definitive Guide 2nd Edition https://www.oreilly.com/library/view/mongodb-the-definitive/9781449344795/.
23. Ladley John (2020) Data Governance How to Design, Deploy, and Sustain an Effective Data Governance Program. Academic Press https://www.everand.com/book/239466636/Data-Governance-How-to-Design-Deploy-and-Sustain-an-Effective-Data-Governance-Program.
24. (2020) GitHub.Com Help Documentation. GitHub Docs https://docs.github.com/en.
25. (2022) What Is GDPR, the EU's New Data Protection Law?. GDPR.Eu https://gdpr.eu/what-is-gdpr/?cn-reloaded=1.