**Data Engineering Lifecycle**

As the field of data engineering continues to grow, it’s essential to move beyond viewing it as just a collection of technologies. Instead, we should consider it as a comprehensive lifecycle that transforms raw data into valuable insights. This blog post aims to provide an in-depth look at the data engineering lifecycle through each stage from data generation to serving valuable data products.

The goal of this blog post is to summarize the concept of the data engineering lifecycle, introduce its stages, and highlight the major undercurrents that support each phase.

**What Is the Data Engineering Lifecycle?**

The data engineering lifecycle encompasses the entire process of transforming raw data into a useful end product. It involves several stages, each with specific roles and responsibilities. This lifecycle ensures that data is handled efficiently and effectively, from its initial generation to its final consumption.

**Stages Overview**

The data engineering lifecycle is divided into five main stages:

1. **Generation**: Collecting data from various source systems.
2. **Storage**: Safely storing data for future processing and analysis.
3. **Ingestion**: Bringing data into a centralized system.
4. **Transformation**: Converting data into a format that is useful for analysis.
5. **Serving Data**: Providing data to end-users for decision-making and operational purposes.

Press enter or click to view image in full size

A diagram of data engineering lifecycle

AI-generated content may be incorrect.

Data Engineering Lifecycle and it’s undercurrents (Source: Xebia blog and Fundamentals of data engineering book)

**Comparison**

The data engineering lifecycle is a subset of the broader data lifecycle. While the full data lifecycle includes everything from data creation to eventual disposal, the data engineering lifecycle focuses specifically on the stages controlled by data engineers. This distinction is important because it highlights the specific responsibilities and challenges faced by data engineers in the overall data management process.

**Stage 1: Generation**

**Source Systems**

Source systems are the origin points of data. These can range from IoT devices and application message queues to transactional databases. Data engineers interact with these systems to collect raw data, although they typically do not control these systems.

Get Jayant Nehra’s stories in your inbox

Join Medium for free to get updates from this writer.

Subscribe

*Did you know that the average enterprise works with over 400 different data sources?*

**Understanding Source Systems**

Data engineers need a thorough understanding of how source systems operate. This includes knowing how data is generated, its frequency, velocity, and variety. Effective communication with source system owners is also crucial to manage changes that might affect data pipelines.

**Key Considerations**

When evaluating source systems, data engineers must consider:

* **Data Characteristics**: Is the source an application, IoT device, etc.?
* **Data Persistence**: Is the data stored long-term or deleted quickly?
* **Data Generation Rate**: How much data is generated over time?
* **Consistency**: How reliable is the data quality?
* **Error Rates**: Frequency of data errors.
* **Duplicates**: Presence of duplicate data entries.
* **Schema**: The structure of the ingested data.
* **Change Management**: Handling schema changes and data updates.
* **Impact on Performance**: How data reading affects source system performance.

**Examples**

* **Traditional Application Databases**: These databases have been popular since the 1980s and remain common today. They typically involve application servers supported by a relational database management system (RDBMS).
* **IoT Swarms**: A more modern example where fleets of devices send data to a central collection system. This setup is becoming increasingly common with the rise of smart devices and sensors.

A diagram of a cloud computing system

AI-generated content may be incorrect.

An IOT Swarm and Data Warehouse Setup (Source: Snowflake Analytics)

**Stage 2: Storage**

**Importance of Storage**

Data storage is a crucial stage in the data engineering lifecycle. The choice of storage solution can significantly impact the efficiency of data processing, transformation, and serving. It provides the foundation for managing and utilizing data throughout its lifecycle.

**Key Considerations**

When selecting a storage solution, data engineers must evaluate several key factors to ensure optimal performance and scalability:

* **Compatibility**: Ensure the storage solution supports the required write and read speeds for your architecture.
* **Performance Bottlenecks**: Assess if the storage system could create bottlenecks for downstream processes.
* **Understanding of Technology**: Utilize the storage system optimally without committing antipatterns (e.g., high-rate random access updates in an object storage system).
* **Scalability**: Consider the system’s ability to handle future scale in terms of storage capacity, read/write operation rates, etc.
* **Data Retrieval**: Ensure downstream users can retrieve data within required service-level agreements (SLAs).
* **Metadata Management**: Capture metadata about schema evolution, data flows, and data lineage to enhance data utility.
* **Query Capabilities**: Determine if the storage solution supports complex query patterns or is purely for storage.
* **Schema Handling**: Understand if the storage system is schema-agnostic, flexible schema, or enforced schema.
* **Data Governance**: Track master data, golden records, data quality, and data lineage for regulatory compliance and data sovereignty.

**Data Access Frequency**

Data access frequency determines the “temperature” of your data, influencing the choice of storage solution:

* **Hot Data**: Frequently accessed data, needing fast retrieval, stored in high-speed storage solutions.
* **Lukewarm Data**: Accessed occasionally, stored in moderately fast storage solutions.
* **Cold Data**: Rarely accessed, suitable for archival storage solutions with lower retrieval costs but higher storage efficiency.

**Storage Solutions**

There are various storage solutions available, each suited to different data access patterns and use cases:

* **Cloud Data Warehouses**: Offer scalable storage with integrated processing capabilities (e.g., Amazon Redshift, Google BigQuery).
* **Data Lakes**: Allow storage of large volumes of raw data in its native format, supporting a wide range of data types (e.g., AWS S3, Azure Data Lake Storage).
* **Object Storage**: Suitable for storing large amounts of unstructured data, often with built-in query capabilities (e.g., Amazon S3, Google Cloud Storage).

**Stage 3: Ingestion**

**Ingestion Process**

Data ingestion is the process of gathering and moving data from various source systems into a centralized data repository. This stage is critical for ensuring that data is available for further processing and analysis.

**Key Considerations**

When architecting or building a data ingestion system, data engineers need to consider:

* **Use Cases**: Identify the purpose of the ingested data and potential reuse scenarios.
* **Reliability**: Ensure the source systems and ingestion processes are reliable and data is available when needed.
* **Data Destination**: Determine where the ingested data will be stored and how it will be accessed.
* **Access Frequency**: Understand how often the data will be accessed.
* **Data Volume**: Assess the typical volume of incoming data.
* **Data Format**: Ensure downstream systems can handle the ingested data format.
* **Data Quality**: Evaluate the quality of source data and its suitability for immediate use.
* **Transformation Needs**: Determine if data needs to be transformed before reaching its destination, including potential in-flight transformations.

**Batch vs. Streaming**

Data ingestion can be done in batch mode or as a continuous stream:

* **Batch Ingestion**: Processes large chunks of data at scheduled intervals. Suitable for use cases where real-time processing is not required.

Example*: A company processes its sales data every night to generate daily sales reports.*

* **Streaming Ingestion**: Provides data to downstream systems in near real-time. Ideal for scenarios requiring immediate data availability.

Example*: A financial institution processes transaction data in real-time to detect and prevent fraud.*

**Push vs. Pull Models**

* **Push Model**: Source systems write data directly to the target system. Useful when immediate data transfer is required.

Example*: IoT devices sending sensor data directly to a cloud storage system.*

* **Pull Model**: Target system retrieves data from source systems. Suitable for periodic data collection.

Example*: A data warehouse periodically pulling data from an external API.*

**Stage 4: Transformation**

**Transformation Process**

Data transformation involves converting raw data into a format suitable for analysis and consumption. This stage is crucial for adding value to data and making it useful for downstream processes.

**Key Considerations**

When planning data transformations, data engineers must consider:

* **Business Value**: Assess the cost and ROI of the transformation processes.
* **Simplicity**: Ensure transformations are as simple and self-contained as possible.
* **Business Rules**: Incorporate business logic to ensure data transformations support organizational needs.

**Transformation Methods**

* **Data Cleaning**: Correcting or removing inaccurate records.

Example*: Removing duplicate entries from a customer database.*

* **Schema Transformation**: Changing the data schema to a more usable structure.

Example*: Converting a nested JSON structure into a flat table format.*

* **Normalization**: Organizing data to reduce redundancy.

Example*: Breaking down a large customer table into separate tables for customers, orders, and products.*

* **Aggregation**: Summarizing data for reporting purposes.

Example*: Calculating the total sales per month from daily transaction records.*

* **Featurization**: Extracting features for machine learning models.

Example*: Creating features such as average transaction value and purchase frequency from raw sales data.*

**Examples**

* **Basic Transformations**: Converting data types, removing null values.

Example*: Changing string data representing dates into actual date formats.*

* **Complex Transformations**: Joining tables, applying business logic, creating derived metrics.

Example*: Combining customer data with transaction data to calculate customer lifetime value.*

* **Real-World Business Logic**: Applying accounting rules, sales calculations, etc.

Example*: Applying tax calculations to sales data based on regional tax rules.*

**Stage 5: Serving Data**

**Purpose of Serving Data**

Serving data involves providing processed data to end-users for decision-making, operational purposes, and advanced analytics. This is where the value of data engineering efforts becomes evident.

**Types of Data Serving**

**Analytics**

* **Business Intelligence (BI)**: Using data to describe the business’s past and current state.

Example*: Generating quarterly financial reports to assess company performance.*

* **Operational Analytics**: Real-time dashboards and metrics for operational decision-making.

Example*: Monitoring website traffic in real-time to identify and address performance issues.*

* **Embedded Analytics**: Providing analytics within other applications for enhanced functionality.

Example*: Embedding sales analytics directly into a CRM platform for sales teams.*

**Machine Learning**

* **Model Training**: Using transformed data to train ML models.

Example*: Training a predictive model on historical sales data to forecast future sales.*

* **Real-Time Predictions**: Deploying models to make instant predictions based on new data.

Example*: Using a recommendation engine to suggest products to users in real-time based on their browsing history.*

**Reverse ETL**

* **Feeding Data Back**: Pushing processed data back into source systems for operational use.

Example*: Updating customer profiles in a CRM system with the latest purchase history and interaction data.*

**Key Considerations**

* **Data Quality**: Ensuring data is clean, accurate, and ready for use.
* **Security**: Implementing robust access controls to protect data.
* **Performance**: Ensuring data can be served efficiently to meet user needs.

**Examples**

* **Dashboards and Reports**: Creating visualizations for business insights.

Example*: A sales dashboard showing key metrics like total sales, average order value, and sales by region.*

* **ML Models**: Deploying models for predictive analytics.

Example*: A churn prediction model identifying customers at risk of leaving.*

* **CRM Systems**: Updating customer relationship management systems with the latest data insights.

Example*: Automatically updating a customer’s profile with recent purchase history and engagement scores.*

**Major Undercurrents Across the Data Engineering Lifecycle**

The major undercurrents that support every aspect of the data engineering lifecycle are essential for ensuring the efficiency, security, and scalability of data processes. These undercurrents include security, data management, DataOps, data architecture, orchestration, and software engineering.

**Security**

Security is paramount in data engineering. Ensuring that data is protected from unauthorized access and breaches is critical to maintaining the integrity and confidentiality of information.

* **Data and Access Security**: Implementing robust security measures to protect data at rest and in transit.

Example*: Using encryption to secure data stored in databases and transmitted over networks.*

* **Principle of Least Privilege**: Granting users and systems the minimal level of access necessary to perform their tasks.

Example*: Configuring database access controls so that only authorized users can read or write specific data.*

* **Compliance and Regulations**: Adhering to data protection laws and regulations such as GDPR, CCPA, and HIPAA.

Example*: Ensuring data anonymization and pseudonymization to protect personal identifiable information (PII).*

**Data Management**

Effective data management practices ensure that data is accurate, accessible, and usable throughout its lifecycle.

* **Data Governance**: Establishing policies and procedures to manage data quality, integrity, and security.

Example*: Implementing a data governance framework to define data ownership, stewardship, and accountability.*

* **Metadata Management**: Capturing and managing metadata to improve data discoverability and usability.

Example*: Using data catalogs to document data lineage, definitions, and relationships.*

* **Data Quality Management**: Ensuring that data is accurate, complete, and timely.

Example*: Implementing data validation checks and automated data quality monitoring to detect and correct errors.*

**DataOps**

DataOps applies DevOps practices to data management, emphasizing collaboration, automation, and continuous improvement.

* **Automation**: Using CI/CD pipelines to automate data workflows and deployments.

Example*: Implementing automated testing and deployment processes for data pipelines to ensure reliability and consistency.*

* **Monitoring and Observability**: Continuously monitoring data workflows and systems to detect and address issues promptly.

Example*: Using monitoring tools to track data pipeline performance, detect anomalies, and alert data engineers to potential problems.*

* **Incident Response**: Establishing processes to quickly respond to and resolve data incidents.

Example*: Developing incident response playbooks and conducting regular drills to ensure the team is prepared to handle data breaches or system failures.*

**Data Architecture**

A well-designed data architecture supports scalability, flexibility, and efficiency in data processing and storage.

* **Scalability**: Designing systems that can handle increasing volumes of data and users.

Example*: Using distributed data storage solutions such as Apache Cassandra or Amazon DynamoDB to scale horizontally.*

* **Flexibility**: Building systems that can adapt to changing business requirements and technologies.

Example*: Implementing a microservices architecture to enable modular and flexible data processing components.*

* **Cost Optimization**: Balancing performance and cost in data infrastructure.

Example*: Leveraging cloud-based storage and compute resources to scale on-demand while managing costs.*

**Orchestration**

Orchestration tools coordinate and manage the execution of data workflows, ensuring that data processes run smoothly and efficiently.

* **Workflow Management**: Automating the scheduling and execution of data workflows.

Example*: Using Apache Airflow to define and manage complex data workflows with dependencies and error handling.*

* **Data Dependency Management**: Ensuring that data workflows execute in the correct order based on data dependencies.

Example*: Configuring data pipelines to trigger downstream processes only when upstream data is available and validated.*

* **Error Handling and Recovery**: Implementing robust error handling and recovery mechanisms.

Example*: Setting up retry policies and alerting for failed tasks to minimize the impact of errors on data workflows.*

**Software Engineering**

Strong software engineering practices are fundamental to building reliable and maintainable data systems.

* **Core Data Processing Code**: Writing efficient and maintainable code for data processing tasks.

Example*: Using Apache Spark for scalable data processing with well-structured and documented code.*

* **Infrastructure as Code (IaC)**: Managing infrastructure using code for consistency and repeatability.

Example*: Using Terraform or AWS CloudFormation to define and deploy cloud infrastructure programmatically.*

* **Testing and Quality Assurance**: Implementing rigorous testing and quality assurance practices for data systems.

Example*: Writing unit tests, integration tests, and end-to-end tests to ensure the correctness and reliability of data pipelines.*

**Recap**

The data engineering lifecycle is a comprehensive framework that guides data engineers through the process of transforming raw data into valuable insights. The lifecycle comprises five main stages:

1. **Generation**: Collecting data from various source systems.
2. **Storage**: Safely storing data for future processing and analysis.
3. **Ingestion**: Bringing data into a centralized system.
4. **Transformation**: Converting data into a format that is useful for analysis.
5. **Serving Data**: Providing data to end-users for decision-making and operational purposes.

These stages are supported by essential undercurrents such as security, data management, DataOps, data architecture, orchestration, and software engineering, which ensure the efficiency, security, and scalability of data processes.