



**DEPARTEMEN SISTEM INFORMASI**  
**Fak. Teknologi Elektro & Informatika Cerdas**



# **SE235103**

## **MANAJEMEN DATA, INFORMASI, DAN KONTEN**

### **Chapter 07**

### **Introduction to Data Lakehouse**

**Prof. Ir. Arif Djunaidy, M.Sc., Ph.D.**  
arif.djunaidy@its.ac.id  
adjunaidy@gmail.com

# Learning Objectives & Book Reading

- **Learning Objectives** - To Understand:
  1. What is a Data Lake
  2. How does Data Lake help enterprises
  3. How Data Lake works
  4. Differences between Data Lake and Data Warehouse
  5. Approaches to building a Data Lake
  6. Lambda Architecture-driven Data Lake
- **Book reading:** Tomcy John & Pankaj Misra, “*Data Lake for Enterprises*”, Chapter 2 (Comprehensive Concepts of a Data Lake)

# Outline

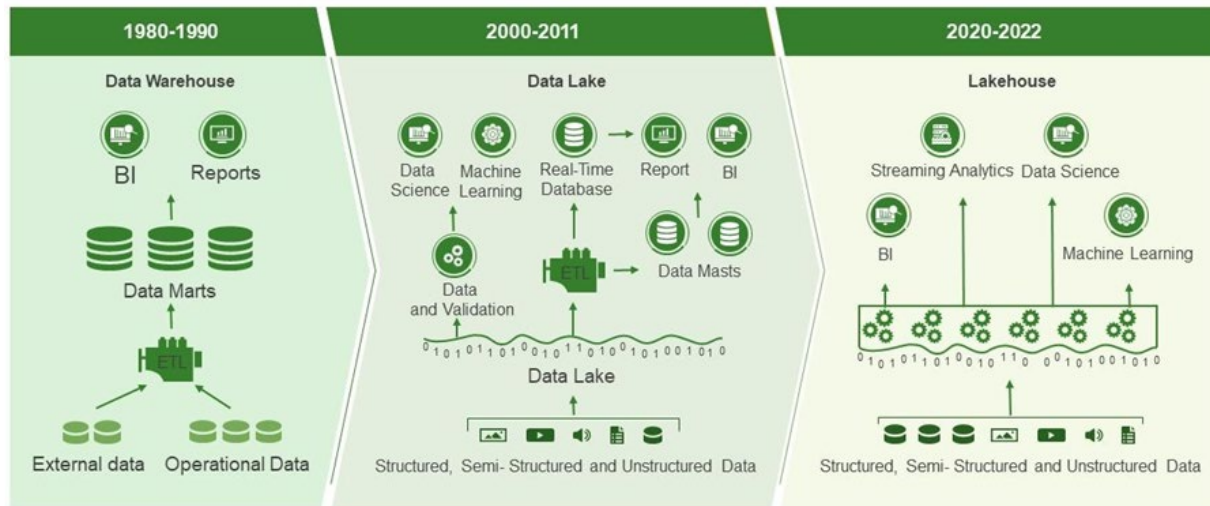
- What is Data Lake?
- Relevance to Enterprises
- How does Data Lake help Enterprises?
- Data Lake Benefits
- How Data Lake Works?
- Differences between Data Lake and Data Warehouse
- Approaches to Building a Data Lake
- Lambda Architecture-driven Data Lake

# What is a Data Lake?

- The term **Data Lake** can be defined as a vast repository of a variety of enterprise-wide, raw information that can be acquired, processed, analyzed, and delivered.
- A Data Lake acquires data from multiple sources in an enterprise in its native form and may also have internal, modeled forms of this same data for various purposes.
  - The information thus handled could be any type of information, from structured or semi-structured data to completely unstructured data.
  - A Data Lake is expected to be able to derive enterprise-relevant meanings and insights from this information using various analysis and machine learning algorithms.





# The Evolution from Data Lakes and Data Warehouses to Data Lakehouses



This diagram shows data storage has evolved over time, with a focus on meeting the growing demands of data volume, variety, and processing.

- **Data Warehouses:** Structured data, strong governance, but high latency and cost.
- **Data Lakes:** Ability to handle all data types, cost-effective, but lacks schema and governance.
- **Data Lakehouses:** Combines the best of both worlds—schema enforcement, ACID compliance, and support for structured and unstructured data.

# Differences between Data Lake & Data Warehouse

Basis of Comparison	 Data Lake	 Data Warehouse
	<b>Vs</b>	
Data Structure	Raw	Structured
Purpose of Data	Still to be determined	Currently in use
Users	Data Scientists	Business Professionals
Accessibility	Highly accessible and updated quickly	Changes are more difficult and expensive to implement

# Differences between Data Lake & Data Warehouse

- Many times, Data Lakes are also perceived as Data Warehouses. Both Data Lake and Data Warehouse have different objectives to be achieved in an enterprise. Some of the key difference are shown here:

Data Lake	Data Warehouse
Captures all types of data and structures, semi-structured and unstructured in their most natural form from source systems	Captures structured information and processes it as it is acquired into a fixed model defined for data warehouse purposes
Possesses enough processing power to process and analyze all kinds of data and have it analyzed for access	Processes structured data into a dimensional or reporting model for advanced reporting and analytics
A Data Lake usually contains more relevant information that has good probability of access and can provide operational needs for an enterprise	A Data Warehouse usually stores and retains data for long term, so that the data can be accessed on demand

# Relevance to Enterprises

- A Data Lake centralizes data, enabling enterprises to tap into previously unexplored capabilities and gain more meaningful business insights.
- With advancements in Data Science and Machine Learning, a Data Lake can optimize operating models and offer specialized capabilities like predictive analysis and recommendations for future growth.
- These hidden capabilities have never been accessible to those who can use them to transform businesses and make better decisions.



# How does a Data Lake help Enterprises? (1)

- Organizations have been aspiring for a long time to achieve a unified data model that can represent every entity in an enterprise. This has been a challenge due to various reasons, some of which have been listed here:
  - An entity may have multiple representations across the enterprise. Hence there may not exist a single and complete model for an entity.
  - Different enterprise applications may be processing the entities based on specific business objectives, which may or may not align with expected enterprise processes.
  - Different applications may have different access patterns and storage structures for every entity.

# How does a Data Lake help Enterprises? (2)

- Enterprises face challenges in standardizing business processes, service definition, and vocabulary.
- A Data Lake perspective addresses these issues by achieving a unified data model without impacting business applications.
- A Data Lake represents entities fully based on information captured from various systems.
- This provides opportunities for enterprises to handle and manage data effectively, helping them grow and derive business insights to achieve goals.
- Martin Fowler's article provides a summary of key aspects of Data Lake in an enterprise at the following link:  
<https://martinfowler.com/bliki/DataLake.html>

# Data Lake Benefits (1)

- Organizations generate a significant amount of data across their business systems, and as they grow, they need to improve their handling of this data across disparate systems.
- One approach is to have a single domain model that accurately describes their enterprise data, which represents the most significant information for their overall business.
- This data can be managed consistently, with systems classified as data owners and data consumers.
- An owner defines how the data becomes available to other consuming systems.
- Building an enterprise-wide Data Lake is a common way to visualize this model, which helps organizations capture, process, analyze, and serve this data to consuming systems.



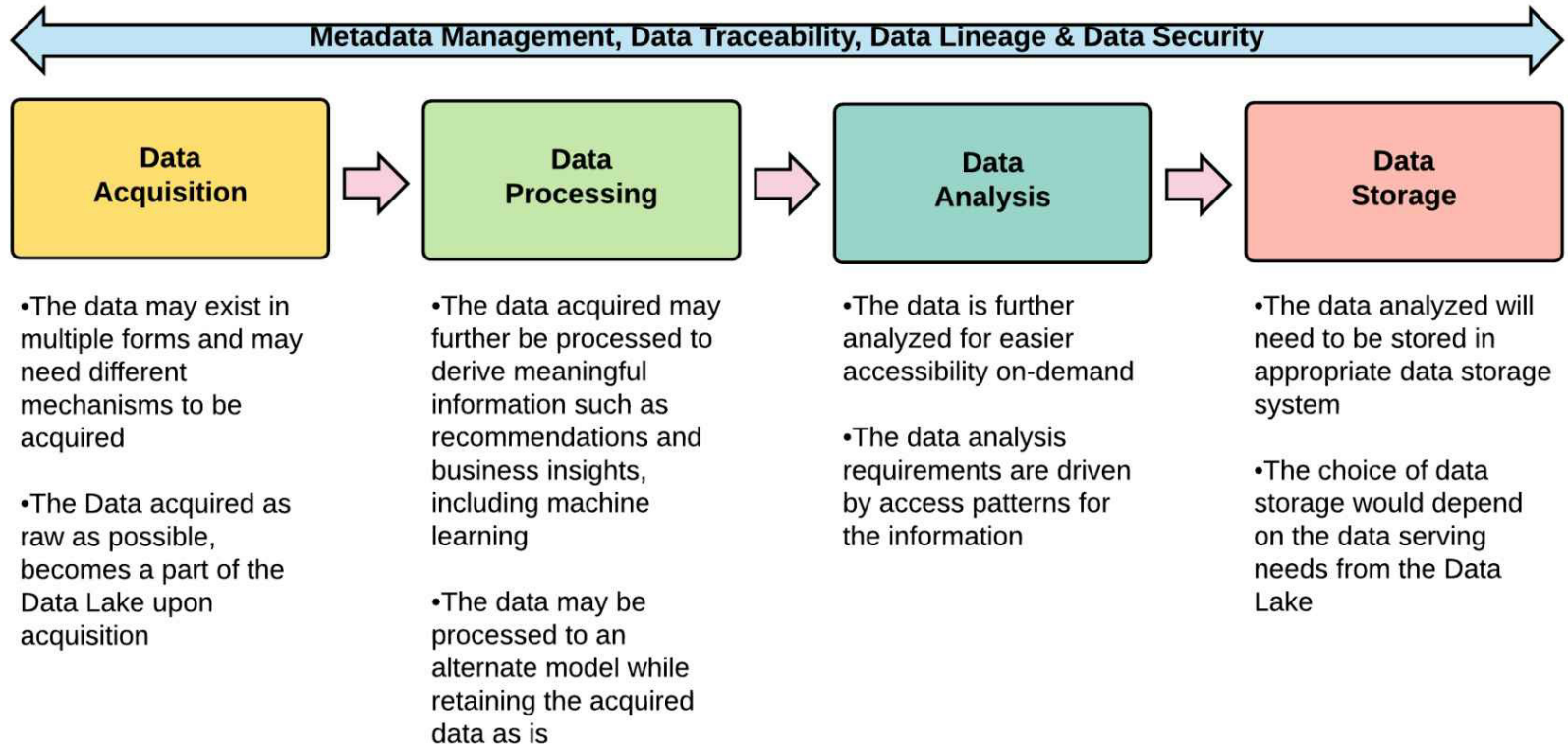
# Data Lake Benefits (2)

- Consistent knowledge of this central model can help the organisations with the following:
  - Data Governance and Lineage
  - Applying machine learning and artificial intelligence to derive business intelligence
  - Predictive Analysis, such as a domain-specific recommendation engine
  - Information traceability and consistency
  - Historical Analysis to derive dimensional data
  - A centralized data source for all enterprise data results in data services primarily optimized for data delivery
  - Helping organizations take more informed decisions for future growth

# How Data Lake Works? (1)

- In order to realize the benefits of a Data Lake, it is important to know how a Data Lake may be expected to work and what components architecturally may help to build a fully functional Data Lake.
- Before we pounce on the architectural details, let us understand the life cycle of data in the context of a Data Lake.
- At a high level, the life cycle of a data lake may be summarized as shown in the following figure.

# How Data Lake Works? (2)



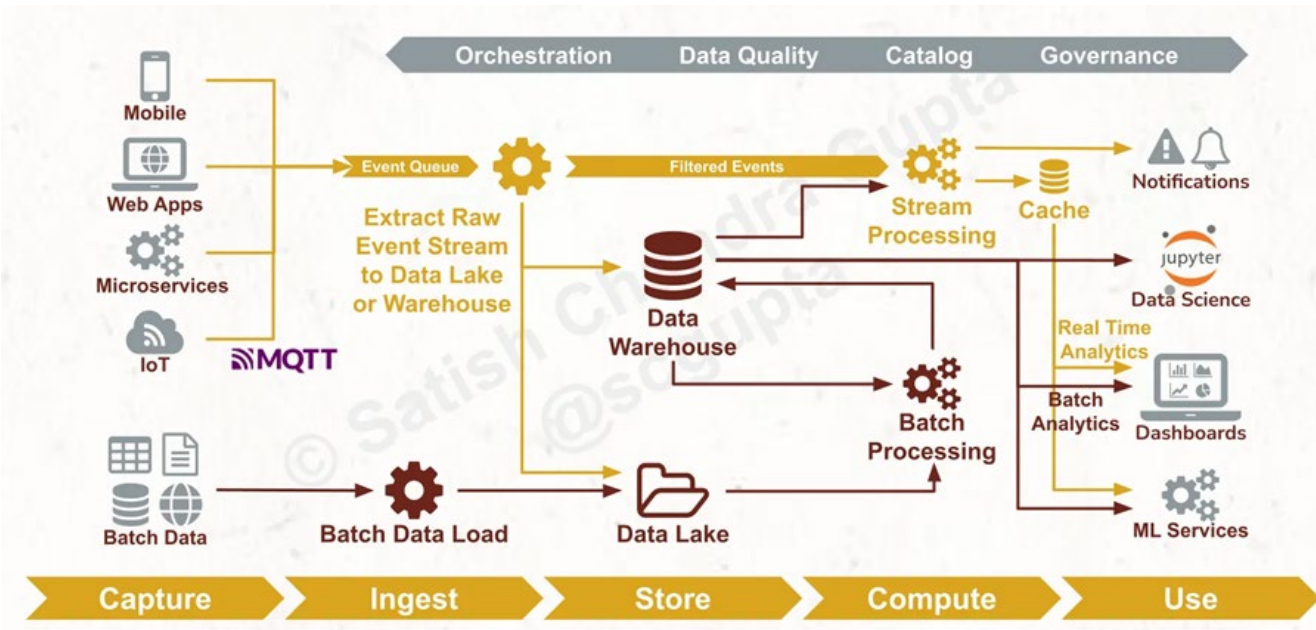
## *Data Lake life cycle*

# How Data Lake Works? (3)

- Data Lakes are systems that store and process data in various stages, including batch and near-real-time processes.
- The choice of processing and analysis depends on the amount of data needed and the specific use cases.
- The choice of storage depends on data accessibility requirements, such as supporting SQL interfaces or allowing data views.
- Data as a service is a more recent requirement, allowing service-based integration with systems that consume data services.
- Data Lakes capture and manage metadata, including data traceability, lineage, and security aspects based on data sensitivity throughout its life cycle.
- These aspects help provide visibility to the data analytics pipeline and simplify error tracing.



# What Makes a Data Lakehouse Unique?

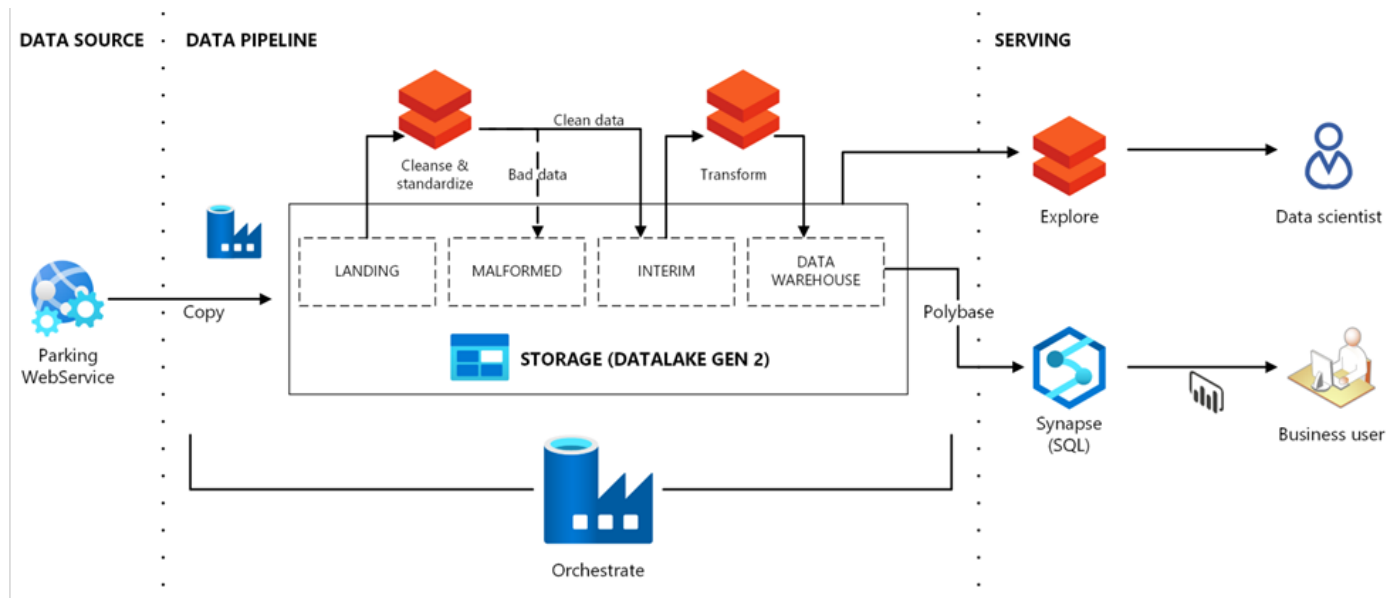


- Key features:

- **ACID Transactions:** Ensures data reliability and consistency.
- **Unified Storage Layer:** Supports both structured and unstructured data.
- **Real-time and Batch Processing:** Seamless integration.
- **Enhanced Metadata Management:** Facilitates better data governance.



# Modern Architecture of a Data Lakehouse



- Data Lakehouse architecture is designed for flexibility and scalability
- Key Components:
  - **Ingestion Layer:** Processes and ingests data from multiple sources.
  - **Storage Layer:** Centralized storage with support for structured and unstructured data.
  - **Processing Layer:** Performs data cleaning, transformation, and analytics.
  - **Serving Layer:** Delivers data for BI tools, dashboards, and applications.

# Approach to Building a Data Lake (1)

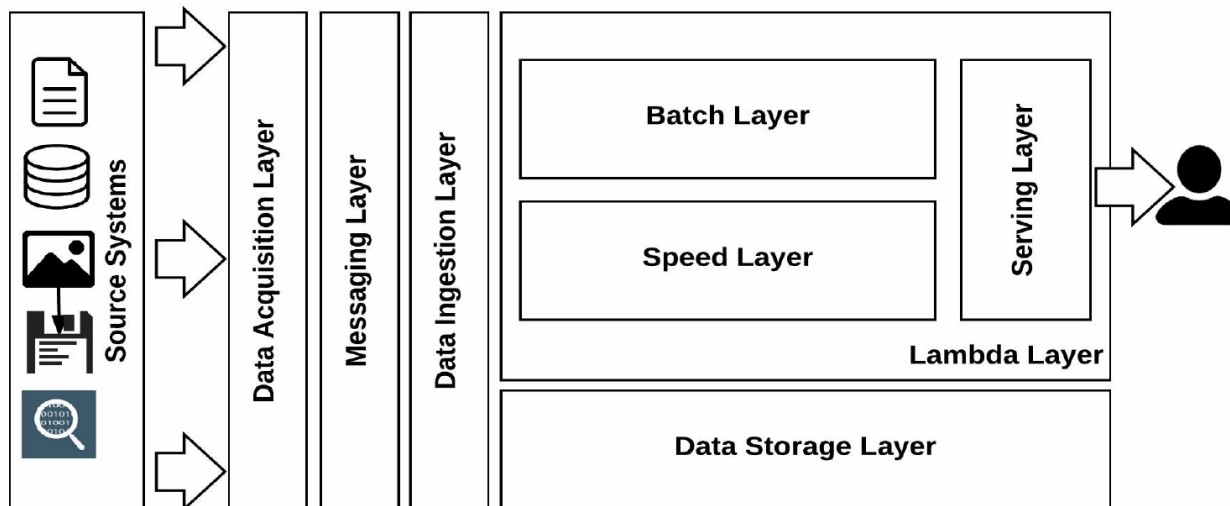
- Different organizations would prefer to build the data lake in different ways, depending on where the organisation is in terms of the business, processes, and systems.
- A simple data lake may be as good as defining a central data source, and all systems may use this central data source for all the data needs.
- Though this approach may be simple and look very lucrative, it may not be a very practical way for the following reasons:
  - This approach would be feasible only if the organizations are building their information systems from scratch
  - This approach does not solve the problems of existing systems
  - Even if organization decides to build the data lake with this approach, there is a lack of clarity of responsibility and separation of concerns
  - Such systems often try to do everything in a single shot, but eventually lose out with increasing demand of data transactions, analysis, and processing

# Approach to Building a Data Lake (2)

- Building a data lake involves examining an organization's information systems, classifying data ownership, and defining a unified enterprise model.
- This approach offers flexibility, control, and clear data definition, while also having independent mechanisms for data capture, processing, and analysis.

# Lambda Architecture-driven Data Lake

- Data processing can be classified into batch and real-time, but merging batch and real-time data can be challenging.
- The Lambda Architecture pattern addresses this problem by providing scalable, performant distributed computing on large data sets.
- It ensures consistent data with required processing in batch and near real-time, enabling scale-out architecture across various data load profiles in an enterprise with low latency expectations.



Layers in a Data Lake

# Data Ingestion Layer:

## Ingest for Processing & Storage

- A fast ingestion layer is one of the key layers in the Lambda Architecture pattern. This layer needs to control how fast data can be delivered into the working models of the Lambda Architecture. Some of the key specifications of this layer are:
  - It must be highly scalable with on-demand scalability to be able to scale based on varying load conditions
  - It must be fault tolerant with both fail-safety (recovery) as well as failover (resiliency)
  - This layer must be able to support multi-thread and multi-event execution
  - This layer must be able to quickly transform the acquired data structure into the target data formats as needed by the processing layers of the Lambda Architecture
  - This layer must ensure that all of the data delivered is in its purest form for further processing

# Batch Layer:

## Batch Processing of Ingestion Data (1)

- The batch processing layer of a Lambda Architecture is expected to process the ingested data in batches so as to ensure optimum utilization of system resources; at the same time, long-running operations may be applied to the data to ensure high quality of data output, which is also known as modeled data.
- The conversion of raw data to modeled data is the primary responsibility of this layer, wherein the modeled data is the data model that can be served by the serving layers of the Lambda Architecture.

# Batch Layer:

## Batch Processing of Ingestion Data (2)

- The primary specifications for this layer can be defined as follows:
  - The batch layer must be able to apply data cleaning, data processing, and data modeling algorithms on the raw data ingested
  - It must have mechanisms in place to replay/rerun the batches for recovery purposes
  - The batch layer must be able to support machine learning and data science based processing on the raw ingested data to produce high quality of modeled data
  - This layer may also have to perform some other operations to improve the quality of the overall modeled data by de-duplication, detecting erroneous data, and providing a view of the data lineage

# Speed Layer: Near Real Time Data (1)

- This layer is expected to perform near-real-time processing on the data received from the ingestion layer.
- Since the processing is expected to be in near real time, such data processing will need to be quick, fast, and efficient, with support and design for high-concurrency scenarios and an eventually consistent outcome.
- A lot of factors play a role in making this layer fast, which will be discussed in detail later in this book.



# Speed Layer: Near Real Time Data (2)

- Broadly, the specifications for such a layer can be summarized as follows:
  - MusMust be able to produce a data model relevant to near-real-time processing needs. All long-running processes must be delegated to batch mode.
  - Must be supported by fast access and storage layers so as to have no backlog/pile-up of events to be processed.
  - Must be decoupled like the batch process from the ingestion layer.
  - Must produce output model in a way that it can be merged with the batchprocessed dataset to provide enriched enterprise data.
  - Must support fast operation on very specific data streams ingested.

# Data Storage Layer: Store All Data (1)

- The data storage layer is crucial in the Lambda Architecture pattern, as it determines the overall solution's reactivity to incoming events/data streams.
- According to the theory of connected systems, a system's speed is as fast as the slowest system in the chain.
- In the Lambda Architecture, there are two active operations: batch processing and near-real-time processing.
- For batch processing, a Hadoop storage layer may suffice, while for near-real-time processing, an indexed data storage is required.

Batch Mode	Near-real-time processing
Serial read and serial write operation	Quick lookups and quick writes
Hadoop storage layer (yes)	Hadoop storage layer (no)

# Data Storage Layer: Store All Data (2)

- Typical specifications for a storage layer in a Lambda Architecture can be summarized as given here:
  - Must support both serial as well as random operations
  - Must be tiered based on the usage pattern with appropriate data solutions
  - Must be able to handle large volumes of data for both batch as well as near-real-time processing
  - Must be flexible and scalable for multiple data structure storage

# Serving Layer: Data Delivery & Exports (1)

- The Lambda Architecture emphasizes the importance of data delivery to the consuming application.
- Data can be delivered through services, such as Data Services, or exports, such as messages, files, or dumps.
- The primary focus is to deliver data in the desired form, enforced as a data contract.
- During data delivery operations, it is crucial to merge batch data with near-real-time processing data, as both streams hold key organizational information.
- The data serving/delivery layer must ensure consistency and adhering to an agreed contract with the consuming application.

# Serving Layer: Data Delivery & Exports (2)

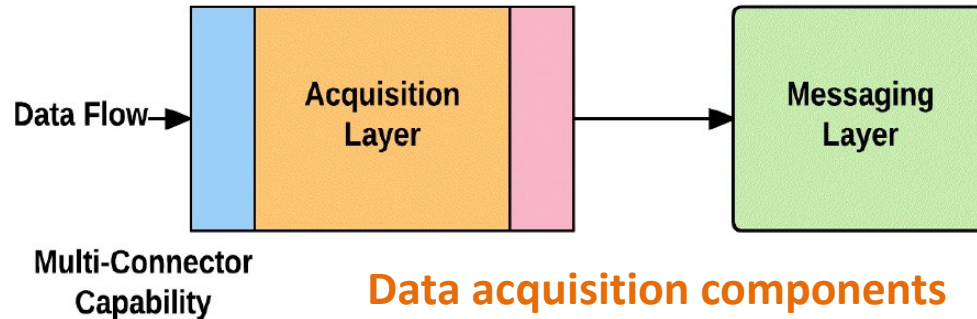
- It must support multiple mechanisms to Overall, high-level specifications for the data serving/delivery layer can be summarized as follows:
  - Serve data to the consuming application
  - For every mechanism supported for serving the data, there should be adherence to a contract in agreement with the consuming application
  - It must support merged views of both batch-processed and near real time processed data
  - It must be scalable and responsive to the consuming application

# Data Acquisition Layer:

## Get Data from Source Systems (1)

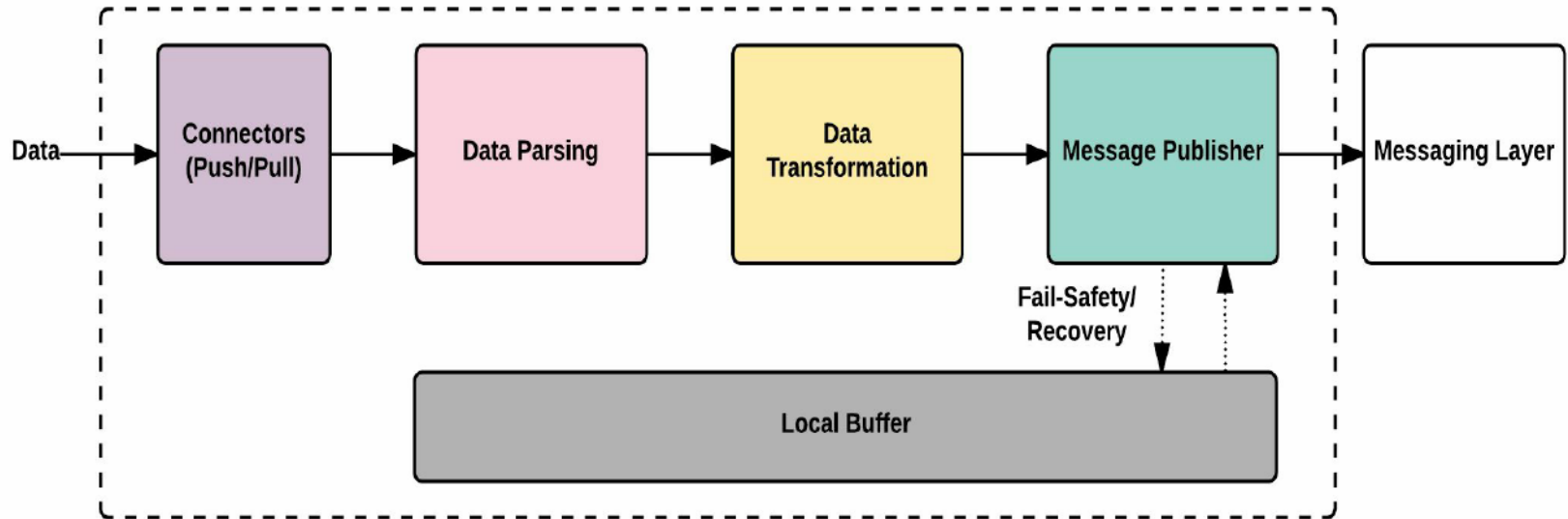
- Organizations have various forms of data, including structured, semi-structured, and unstructured data.
- Structured data includes relational databases, XML/JSON data, and messages across systems.
- Semi-structured data includes emails, chats, documents, and images.
- Unstructured data includes raw texts, audio, and video.
- Schemas are essential for translating data into meaningful information, but they cannot be defined for semi-structured or unstructured data.
- The acquisition layer's role is to convert data into messages for further processing in a Data Lake.
- It must be flexible and have a fast connect mechanism to push translated data messages into the lake.

# Data Acquisition Layer: Get Data from Source Systems (2)



- A data acquisition layer is a component of a data system that uses multi-connector components to transfer data from various sources to a specific destination, such as the messaging layer in Data Lakes.
- The layer performs limited transformations to minimize latency and converts the data into a message or event for posting to the messaging layer.
- In case the messaging layer is unavailable due to network outages or downtime, the data acquisition layer must support fail-safety and fail-over mechanisms.
- This includes local and persistent buffering of messages, allowing them to be recovered when the messaging layer is available again.

# Data Acquisition Layer: Get Data from Source Systems (3)



**Data Acquisition Component Design**



# Messaging Layer:

## Guaranteed Data Delivery (1)

- The messaging layer forms the Message Oriented Middleware (MOM) for data lake architecture, ensuring guaranteed delivery of messages.
- Persistent messages are stored on a storage drive, which should be suitable for the number and size of messages.
- Spindle disks are suitable for serial access, but SSD may provide better IO rates for large-scale applications with millions of messages streamed per second.
- The messaging layer can enqueue and dequeue messages, as most messaging frameworks provide mechanisms for managing publishing and consumption.
- Each framework provides its own libraries to connect to its resources.

# Messaging Layer: Guaranteed Data Delivery (2)

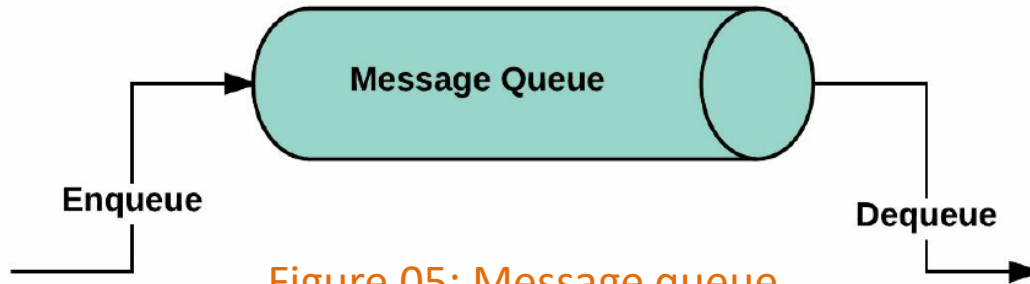
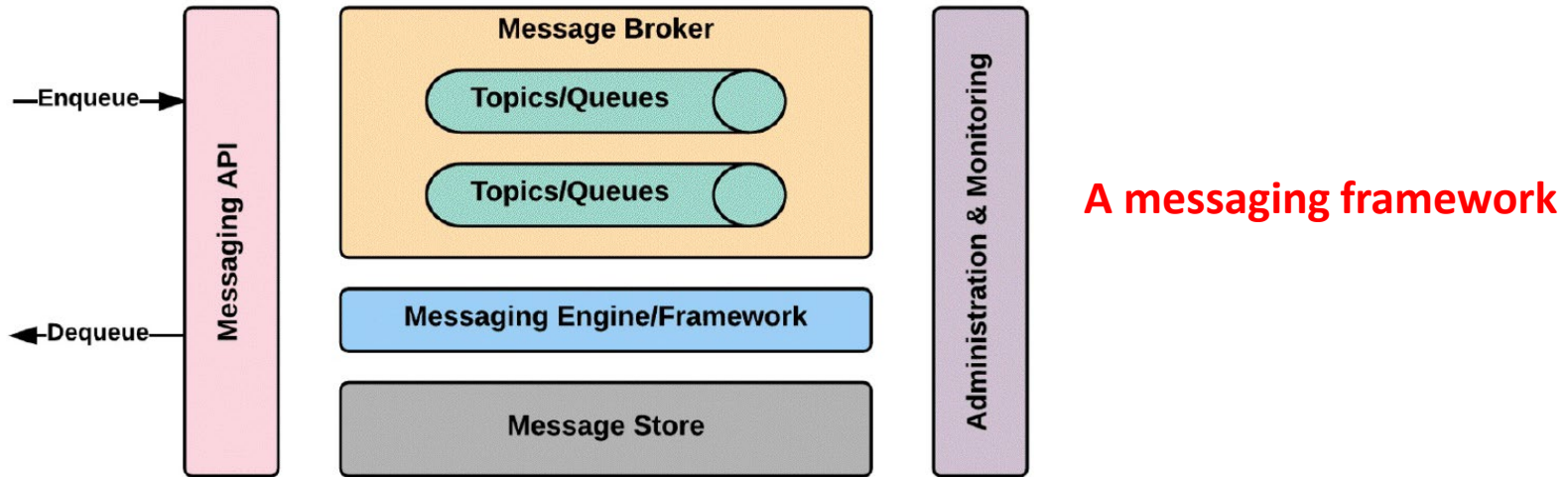


Figure 05: Message queue

- Any message-oriented middleware generally supports two types of communication with queue and topic messaging structures. They are as follows:
  - Queues are mostly used for point-to-point communication, with every message consumed only once by one of the consumers
  - Topics are mostly used for publish/subscribe mechanisms, wherein a message is published once but is consumed by multiple subscribers (consumers).
  - Hence a message is consumed multiple times, once by every consumer. Internally, topics are based on queues; however, these internal queues are managed differently by the messaging engine to provide a publish/subscribe mechanism.

# Messaging Layer: Guaranteed Data Delivery (3)



- Both queues and topics can be configured to be non-persistent or persistent.
- For the purpose of guaranteed delivery, it is imperative to have persistent queues such that messages are never lost.
- At a high level, the message-oriented middleware can be abstracted with components such as message broker, message store, and queues/topics with a messaging framework/engine.

# Exploring the Data Ingestion Layer (1)

- The data ingestion layer consumes messages from the messaging layer and performs necessary transformations to ingest them in the lambda layer, ensuring consistent consumption and no loss of messages.
- It requires multiple consumers/threads for parallel consumption, stateless, and fast streaming capabilities.
- The rate of message consumption must be above or equal to the ingestion rates, preventing latency and ensuring near-real-time processing capability.
- The layer must support fast consumption for recovery from pile-ups and maintain near-real-time processing.
- Message consumers deliver messages to the lambda layer for further processing, with internal components similar to the data acquisition layer.

# Exploring the Data Ingestion Layer (2)

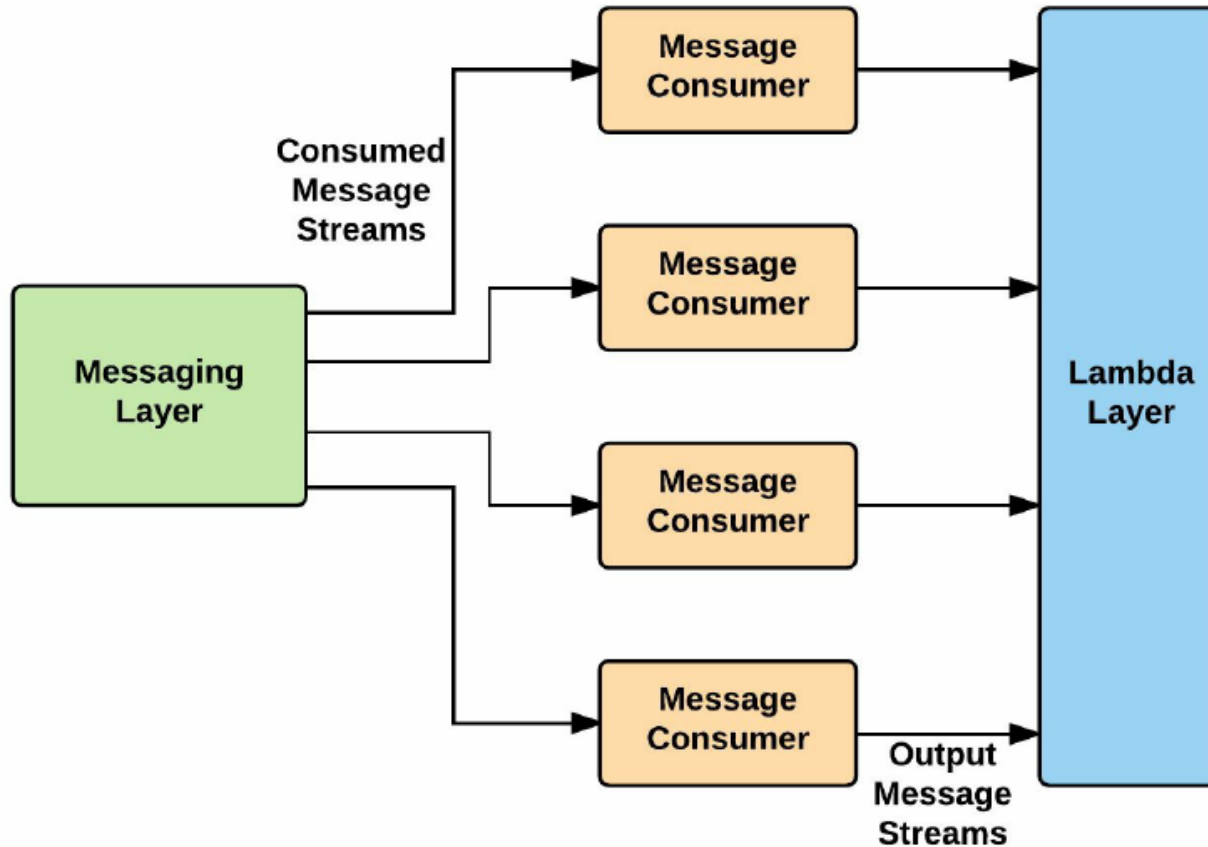


Figure 07: Message consumers

# Challenges and Best Practices for Implementing Data Lakehouses

## ○ Common Challenges:

- **Data Governance:** Managing data quality and compliance.
- **Latency Issues:** Ensuring minimal lag in data availability.
- **Data Security:** Protecting sensitive data in an integrated environment.

## ○ Best Practices:

- **Layered Data Governance:** Implement policies for data lineage, access control, and data cataloging.
- **Scalable Infrastructure:** Use cloud-native tools to ensure scalability.
- **Consistent Data Quality Management:** Regular monitoring and data profiling.

# Future Trends in Data Lakehouses

## ○ Emerging Technologies:

- **AI and Machine Learning Integration:** Automating data pipelines and enhancing decision-making.
- **Real-time Analytics:** Leveraging in-memory processing to achieve faster insights.
- **Cloud and Hybrid Cloud Solutions:** Increasing reliance on scalable cloud infrastructure.

## ○ Predictions:

- Data Lakehouses will be a standard in data architectures due to their flexibility.
- Innovations in data orchestration tools and platforms will enhance their capabilities.



**End of Chapter 07**