UNIT 3

TOPICS:

Large scale data systems Paradigms for distributed data storage
Practical access to example systems ,
Introduction to NoSQL
Amazon Web Services (AWS) provides public data sets in Landsat,
genomics, multimedia.
Layered Framework: Definition of Data Science Framework, Cross
Industry Standard Process for Data Mining (CRISP-DM),
Homogeneous Ontology for Recursive Uniform Schema, The Top
Layers of a Layered Framework, Layered Framework for HighLevel Data **Science and Engineering Business Layer:** Business
Layer, Engineering a Practical Business Layer Utility Layer: Basic
Utility Design, Engineering a Practical Utility Layer
Layer II Three Management Layers: Operational Management
Layer, Processing-Stream Definition and Management, Audit,
Balance, and Control Layer, Balance, Control, Yoke Solution,
Cause-and-Effect, Analysis System, Functional Layer, Data Science
Proces

QUESTIONS:

1. Outline the data science framework
2. What are the different layers of ds framework? Diagrams
3. Explain advantages and disadvantages of different layers with diagram
4. What is the buffer role and drum problem and its phases?
5. Explain functional and non functional requirements
6. What is AWS?
7. What is the importance of aws and why do people prefer it for data sets?
8. Explain different data sets: landsat, genomics and multimedia
9. Explain NOSQL.
10. What are the features of nosql and mongodb?
11. Write code to create schema code and get and crud operations
12. What is ontology? List down advantages and disadvantages and its languages.
13. Describe schema basics of ontology
14. Explain the advantages and disadvantages of using a distributed file system (DFS) for large-scale data storage compared to a traditional centralized storage system.
15. Describe two common paradigms for distributed data storage and provide an example of a system that uses each paradigm.
16. How can data be sharded (partitioned) across different nodes in a distributed storage system? Discuss the benefits and considerations for data sharding.

17. What factors would you consider when choosing between a NoSQL database and a relational database for this purpose?
18. Briefly describe two different types of NoSQL databases and provide an example of a practical application where each type would be preferable.
19. AWS provides public datasets in various domains like Landsat, genomics, and multimedia. Describe how you can leverage these datasets for your data science projects.
20. Give an example of a real-world data science application that could benefit from using one of the publicly available AWS datasets you mentioned.
21. How would you go about ensuring the quality and reliability of data obtained from public datasets?
22. Define the concept of a layered data science framework and explain its benefits for managing complex data science projects.
23. Briefly describe the stages of the Cross-Industry Standard Process for Data Mining (CRISP-DM) and how it relates to a layered data science framework.
24. Compare and contrast homogeneous vs. heterogeneous ontologies in the context of data science. How can ontologies be used within a layered framework?
25. Explain the role of the business layer in a layered framework for data science projects. Provide an example of how a data scientist might collaborate with business stakeholders in this layer.
26. Describe the concept of a practical utility layer within a data science framework. Give some examples of commonly used utilities in this layer.
27. How does a layered framework help manage the operational aspects of data science projects, such as data processing, model training, and monitoring?

LARGE SCALE DATA SYSTEMS

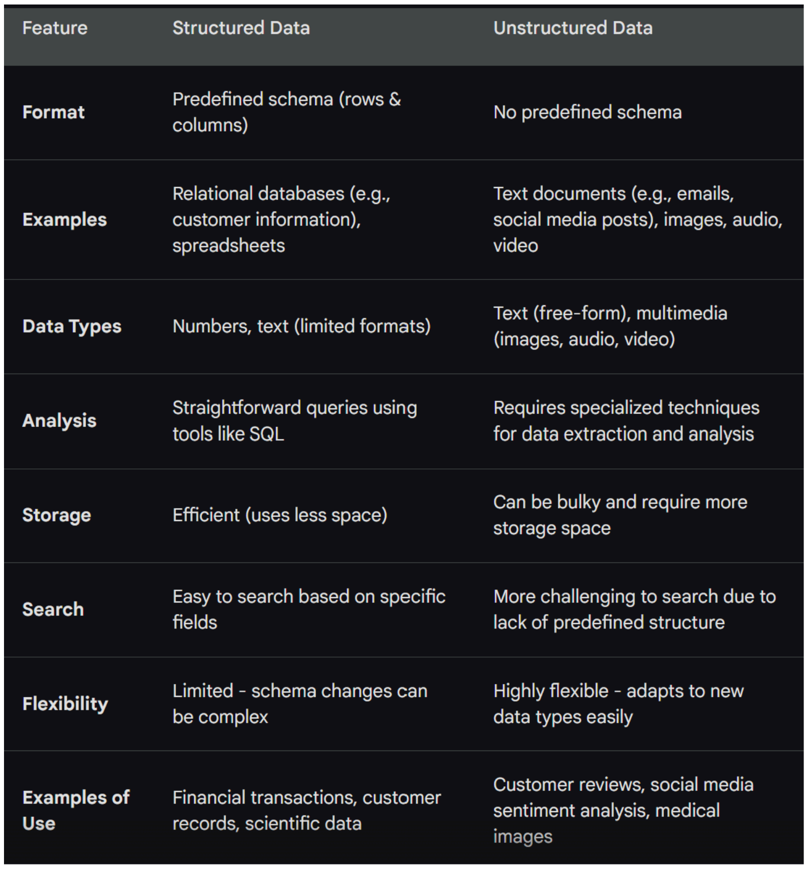
- designed to handle the storage, processing, and analysis of massive datasets => "big data,"
    - sources like social media,
    - sensor networks,
    - scientific instruments,
    - and financial transactions.
- KEY ASPECTS:
    - Distributed Architecture:
        - distribute data across multiple computers (nodes) connected in a network.
        - Gives:
            - Scalability: storage capacity and processing power
            - Fault Tolerance
    - Data Storage Paradigms:
        - DFS - distributed file systems; eg: hadoop
        - NoSQL; eg: mongodb
        - (detail later)
    - Processing and analysis:
        - employ frameworks to manage the processing and analysis of distributed data like
            - Hadoop
            - Spark
    - Advantages
        - Handle massive datasets
        - Gain insights from diverse data
        - Improve decision-making
    - Challenges
        - Complexity
        - Cost
        - Data Security
        -

PARADIGMS OF DIST DATA STORAGE

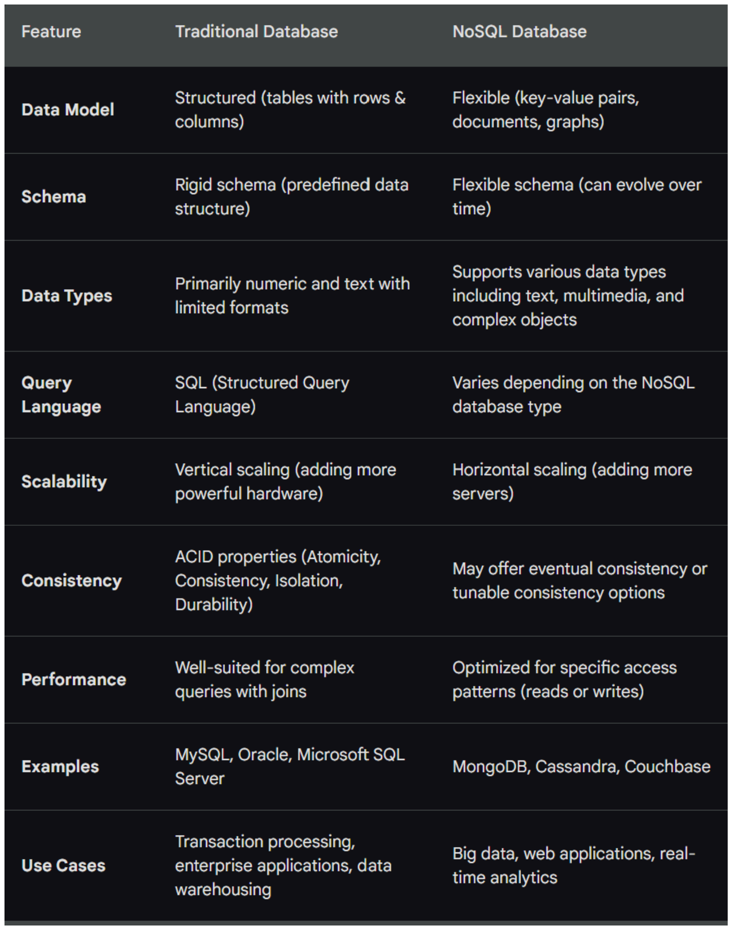
- two main paradigms for storing massive datasets across multiple computers:
    - Distributed File Systems (DFS)
        - giant file cabinet where documents are split into pieces and stored in different drawers across a room.
        - works similarly, dividing data files into smaller chunks and distributing them across various nodes in a network.
        - BENEFITS: Scalability, Fault Tolerance, Parallel Processing
        - CHALLENGES: Complexity, Performance Overhead:
        - EXAMPLES: HADOOP - HDFS , CEPH
        -

- NoSQL Databases.
  - NoSQL databases offer flexible data models that can handle diverse data formats.
  - This makes them ideal for storing and querying semi-structured or unstructured data that's prevalent in big data scenarios
  - BENEFITS: Flexibility, Scalability, Performance
  - CHALLENGES: Data Consistency, Querying Complexity
  - EXAMPLES: MongoDB, DynamoDB

- Choosing the Right Paradigm:
- Use a DFS if:
  - You need to store large files and perform bulk data processing.
  - Data consistency across all nodes is crucial.
- Use a NoSQL database if:
  - You have diverse data structures (semi-structured or unstructured)
  - Performance for frequent reads and writes is a priority.



Feature	Structured Data	Unstructured Data
Format	Predefined schema (rows & columns)	No predefined schema
Examples	Relational databases (e.g., customer information), spreadsheets	Text documents (e.g., emails, social media posts), images, audio, video
Data Types	Numbers, text (limited formats)	Text (free-form), multimedia (images, audio, video)
Analysis	Straightforward queries using tools like SQL	Requires specialized techniques for data extraction and analysis
Storage	Efficient (uses less space)	Can be bulky and require more storage space
Search	Easy to search based on specific fields	More challenging to search due to lack of predefined structure
Flexibility	Limited - schema changes can be complex	Highly flexible - adapts to new data types easily
Examples of Use	Financial transactions, customer records, scientific data	Customer reviews, social media sentiment analysis, medical images



Feature	Traditional Database	NoSQL Database
Data Model	Structured (tables with rows & columns)	Flexible (key-value pairs, documents, graphs)
Schema	Rigid schema (predefined data structure)	Flexible schema (can evolve over time)
Data Types	Primarily numeric and text with limited formats	Supports various data types including text, multimedia, and complex objects
Query Language	SQL (Structured Query Language)	Varies depending on the NoSQL database type
Scalability	Vertical scaling (adding more powerful hardware)	Horizontal scaling (adding more servers)
Consistency	ACID properties (Atomicity, Consistency, Isolation, Durability)	May offer eventual consistency or tunable consistency options
Performance	Well-suited for complex queries with joins	Optimized for specific access patterns (reads or writes)
Examples	MySQL, Oracle, Microsoft SQL Server	MongoDB, Cassandra, Couchbase
Use Cases	Transaction processing, enterprise applications, data warehousing	Big data, web applications, real-time analytics

Practical access to example systems

"Practical access to example systems" can refer to a few different things in the context of data science. Here are two interpretations:

1. Accessing Public Datasets:

Many organizations and cloud providers offer publicly available datasets that can be used for data science projects.

- Identify Relevant Datasets: Search for datasets related to your area of interest. Popular sources include:
    - ■ Kaggle: A platform dedicated to data science competitions and datasets covering various domains.
    - ■ UCI Machine Learning Repository: A well-known repository with a wide range of datasets for machine learning tasks.
    - ■ AWS Public Datasets: Amazon Web Services offers a collection of public datasets in various areas like Landsat (satellite imagery), genomics, and multimedia.
- Choose the Right Format
- Download or Access the Data
- Explore and Preprocess the Data

2. Working with Sample Code and Tutorials:

Many online resources provide sample code and tutorials that demonstrate working with specific data science libraries or frameworks.

- Find Relevant Tutorials: Search for tutorials related to the data science tools you're learning (e.g., Python libraries, machine learning frameworks). Popular platforms include:
    - Kaggle Learn: Offers interactive tutorials and notebooks for learning data science concepts and tools.
    - Scikit-learn Tutorials: The official documentation for the popular machine learning library provides tutorials and examples.
    - Coursera/Udacity/edX: Online courses often include sample code and datasets for practicing data science skills.
- Set Up Your Environment:
- Download or Access the Code
- Run and Understand the Code

NOSQL

- NoSQL (Not Only SQL) databases
    - alternative to traditional relational databases for storing and managing data.
- Unlike relational databases with their rigid table structures,
    - NoSQL databases provide more flexibility in handling diverse data formats.
- NoSQL databases are designed to handle large and complex datasets that may not fit neatly into a predefined schema of rows and columns (like relational databases). They offer:
    - Flexible Data Models: NoSQL databases come in various flavors, each with its own data model. Common models include:
        - Key-Value Pairs: Simple structure where data is stored as key-value pairs (like a dictionary).
        - Document Stores: Documents with flexible schemas can hold various data types within a single unit.
        - Graph Databases: Store data as nodes (entities) connected by edges (relationships).
    - Scalability: NoSQL databases excel at horizontal scaling. You can add more servers to the cluster for increased storage capacity and processing power as your data grows.
    - Performance: Optimized for specific access patterns like frequent reads or writes, leading to faster data retrieval and updates.
- Advantages of NoSQL:
    - Flexibility, Scalability, Performance, Cost-Effectiveness, Development Simplicity
- Disadvantages of NoSQL:
    - Data Consistency, Querying Complexity,Limited Schema Enforcement, Data Backup and Recovery
- The best choice depends on your data and application needs. Consider these factors:
    - Data Structure: If your data is well-defined and requires complex queries, a traditional database might be better.
    - Scalability: If you anticipate significant data growth, NoSQL's horizontal scalability is a plus.
    - Performance: If specific access patterns are crucial, consider NoSQL's optimized performance.
    - Consistency Requirements: If strong data consistency is essential, a traditional database with ACID properties might be preferred.

DIFFERENT TYPES OF NOSQL DATABASES

1. Document Stores:
    - Document stores are a type of NoSQL database that uses a flexible schema to store data in self-contained units called documents.
    - These documents can be in various formats like JSON, XML, or even custom formats.
    - They offer flexibility in the structure and type of data each document can hold.
    - Practical Application:
        - Consider a content management system (CMS) for a website.
        - Each webpage might be stored as a document in a document store.
        - The document could contain various elements like text content, images, metadata (author, creation date), and comments.
        - This flexible schema allows for easy addition of new data fields (e.g., SEO tags) without altering the entire structure.

2. Key-Value Stores:
    - Key-value stores are the simplest type of NoSQL database.
    - They function like giant hash tables, storing data as key-value pairs.
    - The key acts as a unique identifier for retrieving the associated value.
    - This approach offers fast retrieval for known keys but lacks the ability to perform complex queries across multiple key-value pairs.
    - Practical Application:
        - A good example is a shopping cart in an e-commerce application.
        - Each product added to the cart can be stored as a key-value pair.
        - The product ID serves as the key, and the value could be the quantity chosen by the user.
        - This simple structure allows for fast addition and retrieval of items in the cart without needing complex relational queries.

MONGODB

- Leading NoSQL database known for
    - its flexibility, scalability, and performance in handling large and complex datasets.
- utilizes a document-oriented data model,
    - strong choice for storing and managing semi-structured and unstructured
- KEY FEATURES:
    - Document-Oriented Model:
        - stores data in flexible JSON-like documents.
        - document represents a complete entity (e.g., a customer record) and can contain various data types like text, numbers, arrays, and embedded documents (nested structures).
        - storing diverse data formats without rigid upfront design.

    - Scalability:
        - MongoDB offers horizontal scalability.
        - can add more servers (shards) to the cluster to distribute data and processing load as your data volume grows.
        - efficient handling of large datasets and high traffic applications.
    - Performance:
        - optimized for various data access patterns (reads, writes, updates) and offers good performance for both simple and complex queries.
        - Indexing specific fields within documents further enhances query speed.
    - Rich Query Language:
        - provides a powerful query language (MQL) that allows for querying data based on specific criteria and performing aggregations.
    - Replication:
        - can be replicated across multiple servers for enhanced fault tolerance and disaster recovery.
    - Authentication and Authorization:
        - Security features allow for user access control and data protection.
    - Integration with Various Tools:
        - Integrates seamlessly with various programming languages and data analytics tools.
- Advantages of MongoDB:
    - Flexibility, Scalability, Performance, Ease of Use, Large Community and Support
- Disadvantages of MongoDB:
    - ACID Compliance, Schema Enforcement, Queries on Non-Indexed Fields
- Use Cases of MongoDB:
    - Web Applications, Real-Time Analytics, Content Management Systems (CMS), Big Data Analytics: Plays a role in managing and querying large, semi-structured datasets in big data projects.

Write code to create schema code and get and crud operations
Python with PyMongo: Creating Schema, CRUD Operations in MongoDB

1. Connect to MongoDB:
```
# Replace with your connection string and database name
connection_string = "mongodb://localhost:27017/"
database_name = "your_database_name"
from pymongo import MongoClient
client = MongoClient(connection_string)
db = client[database_name]
```

2. Define a Sample Document Structure (Schema):
```
# Sample document structure
document_structure = {
    "name": str,
    "age": int,
    "city": str,
    "hobbies": list
}
```

3. Create a Collection (Similar to a Table in Relational Databases):
```
# Get a reference to the collection (if it doesn't exist, it will be created)
collection = db["your_collection_name"]
```

4. Perform CRUD Operations:
a) Create (Insert) a Document:
Python
```
new_document = {
    "name": "John Doe",
    "age": 30,
    "city": "New York",
    "hobbies": ["reading", "coding"]
}

# Insert the document into the collection
result = collection.insert_one(new_document)
# Get the inserted document's unique identifier (_id)
document_id = result.inserted_id
print(f"Document inserted successfully with ID: {document_id}")
```

b) Read (Find) a Document:

```
# Find a document by its ID
document = collection.find_one({"_id": document_id})
if document:
    print(f"Found document: {document}")
else:
    print(f"No document found with ID: {document_id}")

# Find all documents (without any filter)
all_documents = collection.find()
# You can iterate through the cursor to access each document
for doc in all_documents:
    print(doc)
```

c) Update a Document:

```
# Update a document by its ID
update_result = collection.update_one(
    {"_id": document_id}, {"$set": {"age": 35}}  # Update age field
)

if update_result.matched_count > 0:
    print(f"{update_result.matched_count} document(s) updated successfully.")
else:
    print(f"No document found with ID: {document_id}")
```

d) Delete a Document:

```
# Delete a document by its ID
delete_result = collection.delete_one({"_id": document_id})

if delete_result.deleted_count > 0:
    print(f"{delete_result.deleted_count} document(s) deleted successfully.")
else:
    print(f"No document found with ID: {document_id}")
```

Use code with caution.

Remember to replace placeholders:

AWS

- giant digital toolbox – that's Amazon Web Services (AWS).
- provides various cloud computing services like storage, databases, computing power, and more.
- Used data storage, processing, and analysis on a large scale.
- PUBLIC DATASETS:
    - giant collections of digital information freely available for anyone to use.
    - can be text, images, audio, video, or even scientific data like genetic information.
- AWS DATASETS:
    - 3 main
        - Landsat:
            - Satellite images of Earth's surface, helpful for studying things like deforestation, agriculture, and urban planning.
        - Genomics:
            - Data related to genes and DNA, useful for research in personalized medicine, drug discovery, and understanding evolution
        - Multimedia:
            - Images, audio, and video datasets used to train and improve artificial intelligence applications like self-driving cars, social media analysis, and medical image analysis.
    - BENEFITS:
        - Free and Accessible, Large and Diverse, Cloud-Based, Collaboration
    - How to Use AWS Public Datasets:
        - Find the Data: Browse the AWS Public Data Registry to discover datasets relevant to your project
        - Access the Data: Many datasets are downloadable, or you can access them directly through AWS services like storage and machine learning platforms.
        - Analyze the Data: Utilize AWS tools and resources to analyze and gain insights from the data.

LANDSAT

- provides access to a vast archive of satellite imagery captured by the Landsat program, a joint effort by **NASA and the U.S. Geological Survey (USGS).**
- Data Coverage:
  - Landsat imagery spans decades, offering historical and ongoing observations of Earth's surface. Users can access data for various locations across the globe.
- Applications: Landsat imagery finds applications in diverse fields like:
  - **Environmental Monitoring:** Track deforestation, assess land cover changes, monitor natural disasters like floods and wildfires.
  - **Agriculture:** Analyze crop health, monitor irrigation practices, and predict crop yields.
  - **Urban Planning:** Study urban growth patterns, track infrastructure development, and assess environmental impact of urbanization.
- Benefits of using Landsat on AWS:
  - Accessibility, Scalability,
  - Ready-to-Use : GeoTIFF (COG) format, optimized for fast access and analysis within cloud environments

GENOMICS.

- AWS provides access to various public datasets related to genomics research.
- These datasets can include:
  - Genome Sequences:
    - Complete genetic makeup of various organisms.
  - Gene Expression Data:
    - Shows the activity levels of different genes in a cell or tissue.
  - Variant Data:
    - Identifies genetic variations between individuals or populations.
- Applications: Genomics datasets fuel advancements in healthcare and biological research:
  - **Personalized Medicine:** Analyze genetic data to develop targeted therapies and predict disease risk.
  - **Drug Discovery:** Identify potential drug targets based on gene function and expression patterns.
  - **Evolutionary Biology:** Study genetic variations across species to understand evolution and adaptation.
- Benefits of using Genomics Datasets on AWS:
  - Collaboration, Cost-Effectiveness, High-Performance Computing

MULTIMEDIA

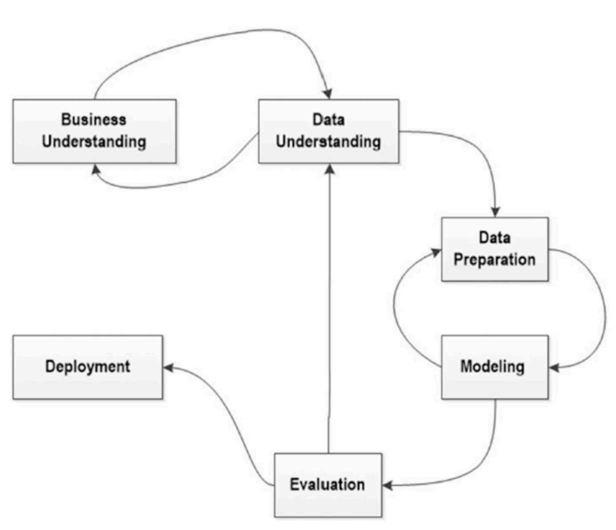
- ecompasses various types of multimedia data, including:
    - **Images:** Large collections of images for tasks like object recognition and image classification.
    - **Audio:** Datasets containing audio recordings for applications like speech recognition and music information retrieval.
    - **Video:** Extensive video datasets useful for tasks like video analysis, action recognition, and anomaly detection.
- Applications: artificial intelligence and computer vision:
    - **Self-Driving Cars:** Train algorithms to recognize objects and navigate roads by analyzing large image and video datasets.
    - **Social Media Content Analysis:** Analyze social media images and videos to extract insights into user behavior and trends.
    - **Medical Image Analysis:** Train algorithms to detect diseases like cancer in medical images (X-rays, CT scans)
- Benefits of using Multimedia Datasets on AWS:
    - Training Data, Benchmarking, Innovation

LAYERED FRAMEWORK

- DEFINITION: A Data Science Framework is a structured approach to handling data science projects. It provides a systematic way to break down the complex process of extracting knowledge and insights from data into manageable steps.
- COMMON PHASES
    - **Business Understanding:** Defining the business problem or question you're trying to solve with data.
    - **Data Acquisition:** Gathering relevant data from various sources like databases, APIs, or sensors.
    - **Data Cleaning and Preprocessing:** Preparing the data for analysis by handling missing values, inconsistencies, and formatting issues.
    - **Exploratory Data Analysis (EDA):** Gaining initial insights from the data through visualization and statistical techniques.
    - **Communication and Reporting:** Communicating the findings and insights gained from the analysis to stakeholders.
- POPULAR FRAMEWORKS:
    - CRISP-DM (Cross-Industry Standard Process for Data Mining):Structured approach and process-oriented methodology.
    - TDD (Test-Driven Development): A development practice also applicable to data science, where you write tests before writing code, promoting robust and well-defined analysis steps.
    - Kaggle Learn: An online platform from Kaggle offering structured learning paths and notebooks for various data science concepts and projects.

CROSS INDUSTRY STANDARD PROCESS FOR DATA MINING (CRISP-DM)

- CRISP-DM (Cross-Industry Standard Process for Data Mining) is a widely used methodology that provides a structured approach for data mining projects. It outlines a six-phase lifecycle that helps data scientists and analysts navigate the process of extracting knowledge and insights from data.



- [illegible]

- PHASES:
  - Business Understanding:
    - Defines the business goals and objectives of the data science project.
    - businesses use a decision model or process mapping tool that is based on the Decision Model and Notation (DMN) standard.
      - Offers businesses a modeling notation for describing decision management flows and business rules.
  - Data Understanding:
    - Explores and analyzes the available data to understand its characteristics and quality.
  - Data Preparation:
    - Cleans, transforms, and prepares the data for modeling.
    - ensures you have all the data required for your data science.
  - Modeling:
    - Selects, trains, and evaluates different data science models to extract insights from the data.
    - It returns to the data preparation phase in a cyclical order until the processes achieve success.
  - Evaluation:
    - Assesses the performance of the chosen model and its suitability for the business needs.
    - If this fails, the process returns to the data understanding phase, to improve the delivery.

        - Deployment:
            - Integrates the model into production for real-world use and monitoring.

Homogeneous Ontology for Recursive Uniform Schema

- internal data format structure that enables the framework to reduce the permutations of transformations required by the framework.
- use of HORUS methodology results in a hub-and-spoke data transformation approach.
- External data formats are converted to HORUS format, and then a HORUS format is transformed into any other external format
- take native raw data and then transform it first to a single format
    - only one format for text files, one format for JSON or XML, one format for images and video.
    - framework's only requirements are a data-format-to-HORUS and HURUS to-data-format converter
- Hub-and-Spoke Transformation Approach:
    - Imagine a hub (HORUS) connected to various spokes (external data formats).
    - Data flows from any external format to the central hub (HORUS) for processing.
    - Then, the processed data can be converted back to any desired external format.
- Benefits of the Hub-and-Spoke Approach:
    - Reduced Transformation Scripts, Efficiency,Scalability

ONTOLOGY

In data science, ontologies offer a structured way to represent the knowledge and concepts within a specific domain. Here's a quick summary:
Benefits:

- Improved Data Understanding: Ontologies clarify the meaning and relationships between data elements, making them easier to interpret and analyze.
- Enhanced Feature Engineering: By defining relevant concepts and properties, ontologies can guide the creation of meaningful features for machine learning models.
- Machine Learning Explainability: Ontologies can help explain the reasoning behind machine learning models, especially those using techniques like rule-based learning.
- Standardized Data Integration: Sharing an ontology across projects or teams ensures consistent data representation and facilitates integration from diverse sources.

Challenges:

- Development Effort: Creating a comprehensive ontology can be time-consuming and require domain expertise.
- Maintaining Consistency: Keeping the ontology up-to-date with evolving knowledge and data formats can be challenging.
- Complexity: Complex ontologies might become difficult to understand and manage for non-experts.

Schema: The Blueprint

Imagine an ontology schema as a blueprint for your data. It defines the essential components that structure your knowledge representation within a specific domain. Here are the key elements:

- Classes: These are the fundamental building blocks, representing the types of entities you'll encounter in your data (e.g., "Customer," "Product," "Transaction").
- Instances: These are specific occurrences of classes (e.g., John Doe is an instance of the "Customer" class).
- Properties: These describe the characteristics of classes and instances (e.g., "age" property for "Customer" class, "price" property for "Product" class).
- Relationships: These define how classes and instances are connected (e.g., "purchased" relationship between "Customer" and "Product").

Schema Languages:

Just like database schemas have specific languages (e.g., SQL), ontologies use specialized languages for defining their structure. Here are some common ones:

- OWL (Web Ontology Language): A powerful language offering various features for defining complex relationships and constraints within the schema.
- RDF (Resource Description Framework): A flexible language often used as a foundation for other ontology languages. It allows representing data in a graph-based structure, where entities and their properties are connected through relationships.

Basic Schema Example:

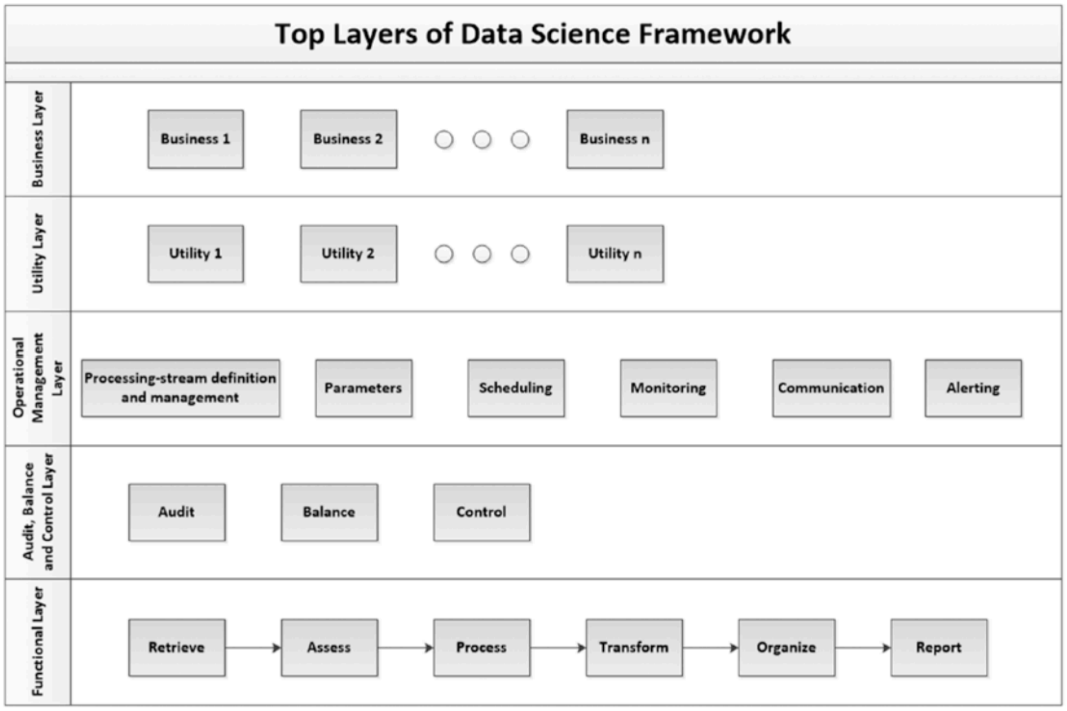
Imagine an ontology schema for a movie recommendation system:

- Classes: Movie, Genre, Director, Actor
- Instances: "The Godfather" (Movie), "Action" (Genre), "Francis Ford Coppola" (Director), "Marlon Brando" (Actor)
- Properties: releaseDate (Movie), genre (Movie), director (Movie), actors (Movie)
- Relationships: directedBy (Movie, Director), actedIn (Movie, Actor)

Benefits of a Well-Defined Schema:

- Clarity and Consistency: A clear schema ensures everyone understands the meaning and relationships within the data.
- Improved Data Quality: Defining properties and constraints helps maintain data consistency and identify potential errors.
- Interoperability: A well-defined schema facilitates sharing and integrating data from different sources that follow the same ontology.

TOP LAYERS OF DATA SCIENCE FRAMEWORK



- **Center of Excellence (COE) for Data Science:**
  - The top layers in the framework (Figure 3-1, not shown) are designed to support the creation of a COE for data science.
  - A COE fosters a collaborative environment where data scientists can share knowledge and best practices.
  - This collaborative approach is crucial for teams working on data science projects.
- **LAYERS: (TOP TO DOWN) & their basics**
  - BUSINESS
    - Source of Requirements: This layer is the starting point, providing business information and requirements for data science projects.
    - Essential Materials: It includes details like organizational structure charts, business process descriptions, subject matter experts (SMEs), project plans, budgets, and data standards.
    - Importance:
      - Clear communication of business needs is crucial for successful data science projects.

        - Standardizing data helps maintain consistency and quality.
- UTILITY
    - Reusable Components: This layer stores reusable code snippets and utilities commonly used in data science projects.
    - Benefits:
        - Saves time by avoiding repetitive coding.
        - Ensures consistency and quality across projects.
        - Promotes collaboration by sharing reusable tools.
- OPERATIONAL MANAGEMENT LAYER
    - Production Pipeline Design: This layer focuses on designing and managing data science pipelines for production environments.
    - Key Components:
        - Processing stream definition and management
        - Parameters for processing tasks
        - Scheduling and monitoring of pipelines
        - Communication and alerting mechanisms
    - Importance:
        - Streamlines data processing for production use.
        - Enables monitoring and control of ongoing processes.
- AUDIT BALANCE CONTROL LAYER
    - Monitoring and Control Center: This layer oversees the data science ecosystem, ensuring smooth operation.
    - Sub-layers:
        - Audit: Tracks process execution, aiding improvement and troubleshooting.
        - Balance: Manages processing capacity, scaling up or down as needed (e.g., cloud computing).
        - Control: Manages the execution of active data science processes. It can initiate recovery or clean-up actions in case of errors.
    - Benefits:
        - Provides insights into process performance and efficiency.
        - Enables proactive management and optimization of resources.
        - Ensures data science pipelines run smoothly and reliably.
    -
- FUNCTIONAL LAYER
    - Core Programming Layer: This layer is responsible for the core data science functionality.
    - Components:
        - Data models: Define the structure and representation of data.
        - Processing algorithms: Perform data analysis, transformation, and modeling tasks.

        - Infrastructure provisioning: Manages the hardware and software resources needed for data science tasks.
        - six-step processing approach (Retrieve, Assess, Process, Transform, Organize, Report)

BUSINESS LAYER

- Bridge Between Business and Data Science:
    - This layer translates non-technical business needs and desires into actionable data science requirements.
- Success Requires Collaboration:
    - Effective data science projects involve collaboration between business experts, analysts, data scientists, and others, not just data science skills.
- Identifying True Requirements:
    - Data scientists have a responsibility to ensure they understand the true business needs, avoiding "bad science" due to poor requirement gathering.
- FUNCTIONAL REQUIREMENTS:
    - record the detailed criteria that must be followed to realize the business's aspirations from its real-world environment when interacting with the data science ecosystem.
        - Described as will of the business
    - MoSCoW method - prioritization technique

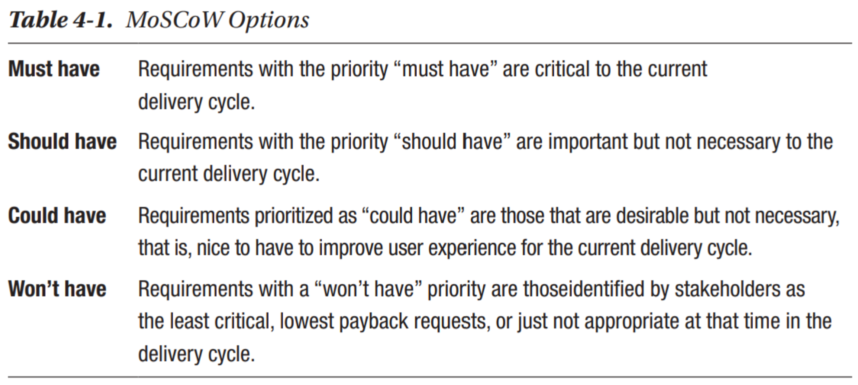
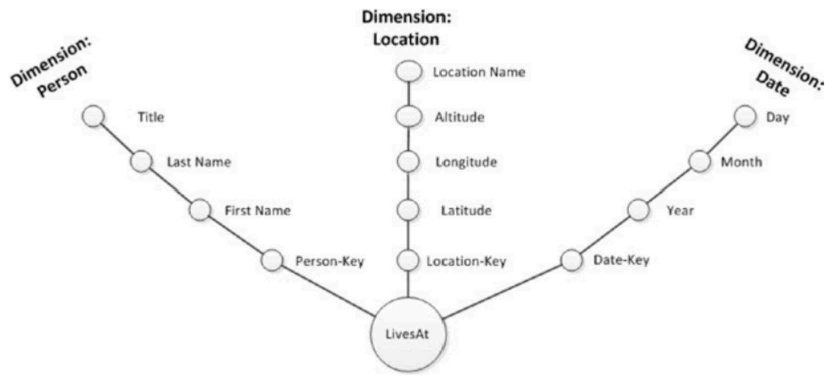


***Table 4-1.** MoSCoW Options*

Must have	Requirements with the priority "must have" are critical to the current delivery cycle.
Should have	Requirements with the priority "should have" are important but not necessary to the current delivery cycle.
Could have	Requirements prioritized as "could have" are those that are desirable but not necessary, that is, nice to have to improve user experience for the current delivery cycle.
Won't have	Requirements with a "won't have" priority are thoseidentified by stakeholders as the least critical, lowest payback requests, or just not appropriate at that time in the delivery cycle.

    -
    - General vs. Specific Requirements:
        - General requirements use a format like "As a [user role], I want [goal] so that [business value] is achieved."
        - Specific requirements for data science involve techniques like data mapping matrices and sun models.
    - DATA MAPPING MATRIX
        - Tracks Data Availability: This matrix documents all data items available within the data sources used for the project.
        - Importance of Updating: It's crucial to keep this matrix up-to-date as you progress through the data science workflow.

- SUN MODELS
  - Requirement Mapping for Non-Technical Users:
    - This technique helps document requirements in a way that business users can understand the analysis intent, while also providing a foundation for technical modeling.
  - Example: A sun model can represent facts (e.g., "LivesAt") and their relationships with dimensions (e.g., "person," "location," "date").
  - 

- Slowly Changing Dimensions (SCDs):
  - Purpose: Manage data in dimensions that change over time.
  - Types of SCD:
    - SCD Type 1 (Only Update): Stores only the latest value for a dimension (e.g., most recent address).
    - SCD Type 2 (Keeps Complete History): Maintains a history of changes:
      - Versioning: Tracks changes with version numbers.
      - Flagging: Latest version is flagged as "1", others as "0".
      - Effective Date: Records the valid period for each data point.
    - SCD Type 3 (Transition Dimension): Records only the most recent transition (e.g., previous and current address).
    - SCD Type 4 (Fast-Growing Dimension): Designed for dimensions with very frequent changes (explained in Chapter 9).

- NON FUNCTIONAL REQUIREMENTS:
  - Non-functional requirements define the criteria for evaluating a data science system's overall operation, ensuring it meets business needs.
  - Accessibility:
    - Enables users with disabilities to access and benefit from the system using assistive technologies like screen readers or color-blind friendly palettes.
  - Audit and Control:
    - Audit: Tracks system usage and identifies violations of data and processing rules.

        - Control: Ensures authorized users access the system according to predefined roles (Role-Based Access Control - RBAC).
    - Compliance:
        - Regulations often mandate audit and control requirements for data privacy and processing.
    - Availability:
        - The ratio of a system's uptime to its downtime.
        - Specifying Requirements:
            - Define the expected uptime as a percentage over a specific period (e.g., weekly reports must be available 100% of the time between 6AM and 10AM every Monday).
            - Consider time zones for geographically distributed users.
            - Identify Single Points of Failure (SPOFs) that can impact availability.
    - Backup Requirements:
        - Regularly archive the data lake, code, libraries, algorithms, and models to restore the system to a known good state in case of data loss or corruption.
        - can be effectively restored to minimize downtime during recovery.
    - Capacity:
        - The ability to load, process, and store specific data volumes.
        - Current and Forecast:
            - Track current usage and forecast future needs as data science models can become complex and require additional processing power.
        - Recording Requirements:
            - Specify the component, its capacity percentage, number of users, data size, and timeframe (e.g., the data hard drive will provide 95% capacity for 1000 users, each with 10MB of data during a 10-minute timeframe).
        - Disaster Recovery and Scaling: Consider cloud-based on-demand capacity for emergencies or peak usage periods.
    - Concurrency:
        - The ability to maintain performance under multiple simultaneous loads.
        - Recording Requirements:
            - Specify the component, the number of concurrent users, and the predefined scripts they will run (e.g., the memory will support 100 users running a sorting algorithm of 1000 records simultaneously).

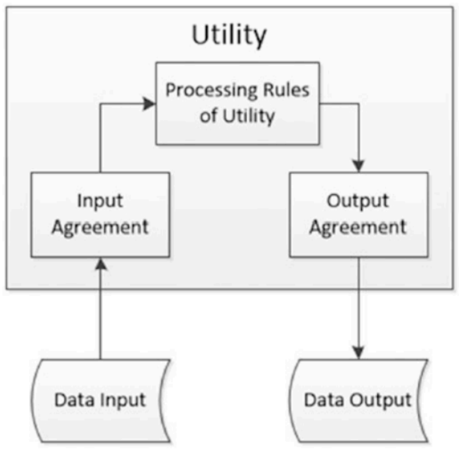
    - High Concurrency vs. High Capacity:
        - High concurrency efficiently handles a subset of users.
        - High capacity supports a larger user base but may require adding more processing resources.
    - Throughput Capacity:
        - The number of transactions the system can handle at peak times.
    - Storage Requirements:
        - Memory:
            - Data persisted in memory for effective processing.
            - More memory generally improves performance.
        - Disk:
            - Short-term Storage: Use fast solid-state drives (SSDs) for processing needs.
            - Long-term Storage: Plan for larger, slower storage using techniques like clustered storage for scalability and redundancy.
            - Warning: Avoid sharing storage or network resources with transactional systems, as data science workloads can consume significant resources.
        - GPU Storage:
            - Leverage GPUs for parallel processing by storing data in GPU memory.
            - Benefits: Significant speed improvements for tasks suited for parallel processing.
    - Year-on-Year Growth:
        - Forecast storage requirements to handle increasing data volumes.
        - Recording Requirements:
            - Specify the component, additional data storage needed, and the timeframe (e.g., component C requires additional 10 MB of storage capacity every year).
- Configuration Management
    - a process to ensure a system meets its requirements throughout its life cycle. It involves things like maintaining consistent performance and making sure the system can be updated.
    - Things to take note of:
        - Deployment, Documentation, Disaster Recovery (DR), Efficiency (Resource Consumption), Effectiveness (Resulting Performance), Extensibility, Failure Management, Fault Tolerance, Latency, Interoperability, Maintainability, Modifiability, Network Topology, Privacy, Quality, Recovery/Recoverability, Reliability, Resilience, Resource Constraints, Reusability, Scalability,Security,Privacy, Physical Access,Testability, Controllability, Isolation Ability. Understandability, Automatability

- **engineering a practical business layer**
    - Structure:
        - Use a directory structure like ./VKHCG/05-DS/5000-BL for clear organization and version control compatibility.
    - Requirements Management:
        - Record each requirement with version control in a separate file (e.g., 000001-01.txt).
        - Implement a numbering scheme (e.g., 000001-00 to 000999-99) to accommodate up to a million requirements with versioning.
    - Requirements Registry:
        - Maintain a central registry file (e.g., 0100-Requirements-Registry.txt) to search for specific requirements easily.
        - Include columns for requirement number, MoSCoW priority, a brief description, creation and last version dates, and status (In-Development, In-Production, Retired).
    - Traceability Matrix:
        - Create a matrix (e.g., 0200-Traceability-Matrix.txt) linking each requirement to the corresponding data science process it supports.
        - This ensures comprehensive control and simplifies changes by understanding dependencies.
    - Benefits:
        - Clear organization and version control for requirements.
        - Easy searching and tracking of requirements.
        - Ensures all requirements are addressed by data science processes.
        - Simplifies future modifications by understanding connections between requirements and processes.

UTILITY LAYER

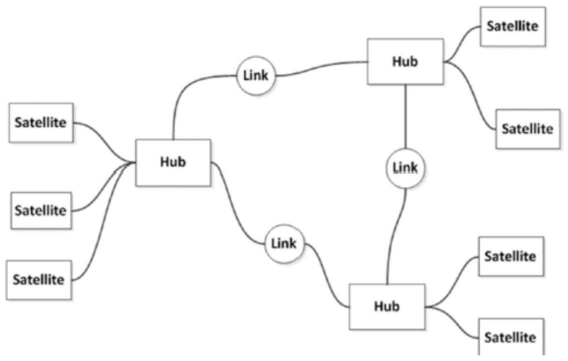
- The utility layer is used to store repeatable practical methods of data science
- Benefits:
    - Prevents the use of outdated or duplicate algorithms.
    - Promotes consistency and stability across projects.
    - Enables collaboration by providing a common set of tools for data scientists and engineers.
- Record Keeping:
    - Maintain all versions of utility source code for future reference.
    - Keep proof and credentials for high-quality, industry-accepted algorithms (demonstrating validity).
- Compliance:
    - Utilities need to comply with regulations like GDPR (Europe), which mandates:
        - Valid consent for data processing. (Testing for consent will be covered later)

        - Transparency about data collection and processing. (Audit trails discussed in Chapter 6)
        - Up-to-date and accurate personal data. (Data processing techniques in Chapter 8)
        - Right to data erasure (removal of personal information - covered later)
        - Approval for data transfer between service providers.
    - Non-compliance with GDPR can result in significant fines.
- Right to be Forgotten: GDPR allows individuals to request removal of their data from all systems. Careful implementation is crucial to avoid legal issues (discussed later).
- Automated Decision Making: The use of automated processing for decisions requires careful consideration and might be subject to regulations (compliance is easier with approved algorithms).
- BASIC UTILITY DESIGN:
    - basic design pattern for creating reusable data science utilities:
    - Structure:
    - The proposed design follows a three-stage approach:
        - Load Data as per input agreement
            - This stage focuses on loading data according to a pre-defined input agreement.
            - specifies the format, location, and any specific requirements for the data being ingested.
        - Apply Processing Rules of utility:
            - the actual processing logic specific to the utility is applied to the loaded data.
        - Save Data as per output agreement:
            - processed data is saved according to a pre-defined output agreement.
            - defines the format, location, and any specific requirements for the resulting data.



- three types of utilities
    - Data processing utilities
        - These utilities are pre-defined functions that perform specific data transformation tasks within your project. They act as building blocks for complex data processing workflows.
        - Benefits of Data Processing Utilities:
            - Standardization, Efficiency, Scalability
        - TYPES:
            - Retrieve Utilities: These utilities handle data retrieval from various sources (e.g., text files, XML, databases) and convert them into a unified HORUS format (defined by the author). Examples include:
                - Text-Delimited to HORUS: Imports comma-separated value (CSV) files.
                - XML to HORUS: Imports XML data.
                - JSON to HORUS: Imports JSON data.
                - Database to HORUS: Imports data from databases (e.g., SQLite).
            - Expert Utilities: These are more advanced utilities for specific data types:
                - Picture to HORUS: Extracts features from images (e.g., object identification).
                - Video to HORUS: Processes videos in two stages:
                    - Movie to Frames: Converts video into individual frames.
                    - Frames to HORUS: Extracts features from each frame.
                - Audio to HORUS: Extracts features from audio files (e.g., channels).
            - ASSESS UTILITIES:
                - Ensure data quality imported via the "Retrieve" step meets the project's standards.
                - Improve data for better processing success in later stages.
                - Types:
                    - Feature Engineering:
                        - Enhance or extract data features to improve characteristic extraction during analysis.

- Fixers Utilities:
  - Address specific data quality issues.
- Adders:
  - enrich your data by finding or calculating new information based on existing entries.
  - Lookups: Translate codes or IDs to full names (e.g., country code to country name).
  - Zone Assignment: Add new data points based on rules or tests (e.g., "in the black" for positive bank balance).



ocess:
  - utilities for building a specific data structure called a "data vault".
  - Components:
  - Hubs: Central entities (e.g., Time, Person, Object, Location, Event)
  - Satellites: Detailed data associated with hubs
  - Links: Relationships between hubs

- DATA VAULT UTILITIES:
  - Data Vault: A specialized data storage method created by Dan Linstedt. It emphasizes detail, historical tracking, and unique linking between normalized tables.
  - Hybrid Approach: Combines the strengths of two common data models:
    - 3rd Normal Form (3NF): Ensures minimal data redundancy.
    - Star Schema: Simplifies querying data for analysis.
  - Components and Utilities:
    - Hubs: Central entities like Time, Person, Object, Location, and Event. Hub utilities ensure the integrity of data in these hubs.
    - Satellites: Detailed data associated with hubs. Satellite utilities ensure the integrity of these satellite tables and their connection to hubs.
    - Links: Relationships between hubs. Link utilities ensure the integrity of these links and their connection to the relevant hubs.

- TRANSFORM UTILITIES
    - contain all the processing chains for building the data warehouse from the results of your practical data science
    - TYPES:
        - Dimensions
            - Ensure the integrity and consistency of the dimensions (descriptive categories) in your data warehouse.
        - Fact:
            - Maintain the integrity of both the dimensions and the facts (numerical measures) in the data warehouse.
- DATA SCIENCE UTILITIES
    - several data science–specific utilities that are required for you to achieve success in the data processing ecosystem.
    - TYPES:
        - Data binning/bucketing: Grouping continuous values into smaller ranges to reduce noise and improve processing efficiency.
        - Data averaging: Summarizing data by calculating averages, which can significantly reduce data volume.
        - Outlier detection: Identifying and handling data points that deviate significantly from the norm.
- ORGANIZE UTILITIES
    - all the processing chains for building the data marts.
    - The organize utilities are mostly used to create data marts against the data science results stored in the data warehouse dimensions and facts
- REPORT UTILITIES
    - all the processing chains for building virtualization and reporting of the actionable knowledge.
    - The report utilities are mostly used to create data virtualization against the data science results stored in the data marts.

- Maintenance utilities
    - Data science solutions require ongoing maintenance, just like any other system.
    - TYPES:
        - Backup & Restore: Standard database backup and restore functionalities to ensure data safety.
        - Data Integrity Checks: Utilities verify the structure and allocation of data objects across the system for accurate processing.
        - History Cleanup: Archive and remove old entries in historical data tables (considering privacy regulations like "right-to-be-forgotten").
        - Maintenance Cleanup: Remove temporary files and artifacts generated during maintenance tasks.
        - Notification Utilities: Alert operations teams about the system's status for proactive maintenance.
        - Data Structure Rebuild: Rebuild database tables and views to ensure everything functions as designed.
        - Reorganize Indexing: Optimize indexing in databases as data volume and complexity grow.
        - Shrink/Move Data: Reduce storage footprint of databases and logs for efficient operation.
        - Solution Statistics: Track and record information about data science artifacts for better management.
- Processing utilities
    - Processing utilities manage the workflow of your data science projects.
        - Scheduling Utilities: these utilities define when specific tasks should run in your data science workflow.
        - Backlog Utilities: Act as a queue, accepting new processing requests to be included in future cycles.
        - To-Do Utilities: Select a subset of requests from the backlog based on priority and dependencies (parent-child relationships) for processing in the next cycle.
        - Doing Utilities: Execute the actual processing tasks for the chosen requests in the current cycle.
        - Done Utilities: Verify that the completed requests have been processed successfully.
        - Monitoring Utilities: Continuously monitor the entire system to ensure everything is functioning as expected.

ENGINEERING A PRACTICAL UTILITY LAYER

This section explains how to design and organize a utility layer for your data science projects.

The Utility Layer:

- Stores reusable code modules (utilities) that various parts of your data science workflow can access.
- Improves efficiency and reduces redundancy by sharing common functionalities.

Sub-layers for better organization:

The book suggests creating three sub-layers to manage utilities effectively:

1. Maintenance Utilities (MU):
    - Directory: ./VKHCG/05-DS/4000-UL/0100-MU (as mentioned in Chapter 2)
    - Purpose: Stores utilities for maintaining the data science environment (e.g., backup/restore, data integrity checks).
    - Registry: Maintain a registry with documentation for each utility, explaining its function and requirements.
2. Data Utilities (DU):
    - Directory: ./VKHCG/05-DS/4000-UL/0200-DU (as mentioned in Chapter 2)
    - Purpose: Stores utilities for data manipulation and processing (e.g., data cleaning, transformation).
    - Registry: Similar to MU, keep a registry with documentation for each data utility.
3. Processing Utilities (PU):
    - Directory: ./VKHCG/05-DS/4000-UL/0300-PU (as mentioned in Chapter 2)
    - Purpose: Stores utilities for managing workflow execution (e.g., scheduling, task execution).
    - Registry: Maintain a registry with documentation for each processing utility.

Important Considerations:

- Ensure the utility layer aligns with your company's existing processing environment and agile principles.
- Modifications to utilities can impact ongoing projects, so exercise caution during updates.

Benefits:

- Improved organization and discoverability of reusable code.
- Reduced redundancy and development time.
- Easier maintenance and collaboration within the data science team.

Layer II Three Management Layers:
three important layers for managing a large-scale data science system:

1. Operational Management Layer: This layer focuses on the day-to-day operations of your data science system. It ensures things are running smoothly, efficiently, and consistently.
2. Audit and Control Balance Layer: This layer is all about governance and compliance. It makes sure your data science system is used ethically, responsibly, and adheres to any regulations.
3. Functional Layer: This layer deals with the core functionalities of your data science system.

OPERATIONAL MANAGEMENT LAYER

- Operations management is one of the areas inside the ecosystem responsible for
    a. designing and controlling the process chains of a data science environment.
- center for complete processing capability in the data science ecosystem.
- stores
    a. what you want to process
    b. along with every processing schedule and workflow for the entire ecosystem.
- This area enables us to see an integrated view of the entire ecosystem.
- It reports the status each and every processing in the ecosystem.
- This is where we plan our data science processing pipelines.
- following in the operations management layer:
    a. Definition and Management of Data Processing stream
    b. Eco system Parameters
    c. Overall Process Scheduling
    d. Overall Process Monitoring
    e. Overall Communication
    f. Overall Alerting

Defn and mgmt of data processing stream

- The processing-stream definitions are the building block of the data
- science environment.
- This section of the ecosystem stores all currently active processing
- scripts.
- Management
    - refers to Definition management,
    - it describes the workflow of the scripts throughout the ecosystem, it manages the correct execution order according to the workflow designed by the data scientist.

Eco system Parameters:

- The processing parameters are stored in this section; here it is made sure that a single location is made available for all the system parameters.
- In any production system, for every existing customer, all the parameters can be placed together in a single location and then calls could be made to this location every time the parameters are needed.
- Two ways to maintain a central location for all parameters are:
  - 1. Having a text file which we can import into every processing script.
  - 2. A standard parameter setup script that defines a parameter database which we can import into every processing script.

Overall process scheduling

- Along with other things the scheduling plan is stored in this section, it
- enables a centralized control and visibility of the complete scheduling
- plan for the entire system.
- One of the scheduling methods is a Drum-Buffer-Rope method.
- STEPS:
  - Retrieve
  - Assess
  - Process
  - Transform
  - Organize
  - Report

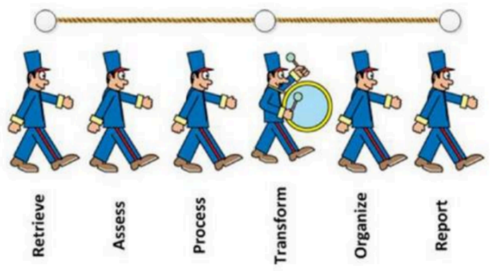


Figure 3.1 : Original Drum-Buffer-Rope use

- The Drum-buffer-rope Method:
  - It is a standard practice to identify the slowest process among all.
  - Once identified it is then used to control the speed of the complete pipeline
  - this is done by tying or binding the remaining processes of the pipeline to this process.
  - The method implies that the "drum" is placed at the slow part of the pipeline, to give the processing pace,
  - the "rope" is attached to all the processes from beginning to end of the pipeline, this makes sure that no processing is done that is not attached to the drum.
  - This approach ensures that all the processes in the pipeline complete more efficiently, as no process is entering or leaving the process pipeline without been recorded by the drum's beat.
  -

Process Monitoring:

- The central monitoring process makes sure that there is a single unified view of the complete system.
- We should always ensure that the monitoring of our data science is being done from a single point.
- With no central monitoring running different data science processes on the same ecosystem will make managing a difficult task.

Overall Communication:

- The Operations management handles all communication from the system, it makes sure that any activities that are happening are communicated to the system.
- To make sure that we have all our data science processes trackedwe may use a complex communication process.

Over all Alerting

- The alerting section of the Operations management layeruses communications to inform the correct status of the complete system to the correct person, at the correct time.

AUDIT BALANCE CONTROL LAYER

- Any process currently under executing is controlled by the audit, balance, and control layer.
- It is this layer that has the engine that makes sure that every processing request is completed by the ecosystem according to the plan.
- This is the only area where you can observe which processes are is currently running within your data scientist environment.
- It records the following information:
  - Process-execution statistics
  - Balancing and controls
  - Rejects- and error-handling
  - Fault codes management
- AUDIT SUBLAYER:
  - An audit refers to an examination of the ecosystem that is systematic and independent
  - This sublayer records which processes are running at any given specific point within the ecosystem.
  - This information collected is used to better understand and plan future improvements to the processing to be done
  - The audit sublayer isn't just a single entity; it comprises a series of specialized tools called watchers. Each watcher focuses on a specific type of information.

- Good indicators for audit purposes:
    - Built-in Logging:
        - Centralized location for all relevant log entries. Crucial for analysis and issue handling.
    - Debug Watcher:
        - This level of logging is the maximum worded logging level. Indicates unnecessary debugging and resource usage, indicating that the tool is using some precise processing cycles to perform the necessary low-level debugging
    - Information Watchers:
        - Log information that is beneficial to therunning and management of a system. These logs be piped to the central Audit, Balance, andControl data store of the ecosystem
    - Warning Watcher:
        - Used for exceptions that are handled or other important log events. This means that the issue was handled by the tool and also took corrective action for recovery.
    - Error Watcher:
        - Logs all unhandled exceptions in the data science tool.
    - Fatal Watcher:
        - Fatal is a state reserved for special exceptions or conditions for which it is mandatory that the event causing this state be identified immediately.
    - Basic Logging:
        - Records everything occurring during process execution.
    - Process Tracking:
        - Monitors hardware-related parameters for performance analysis.
    - Data Provenance:
        - Tracks data transformations for reproducibility and source history.
    - Data Lineage:
        - Records every change to individual data values for historical tracking.

- BALANCE SUBLAYER
    - The balance sublayer has the responsibility to make sure that the data science environment is balanced between
        - the available processing capability
        - against the required processing capability
        - or has the ability to upgrade processing capability during periods of extreme processing.
    - In such cases the on-demand processing capability of a cloud environment becomes highly desirable

- CONTROL SUBLAYER
  - The execution of the current active data science processes is controlled by the control sublayer.
  - The control elements of the control sublayer are a combination of:
    - the control element available in the Data Science Technology Stack's tools and
    - a custom interface to control the overarching work.
  - When processing pipeline encounters an error,
    - the control sublayer attempts a recovery as per our pre specified requirements
    - else if recovery does not work out it will schedule a cleanup utility
  - The cause-and-effect analysis system is the core data source for the distributed control system in the ecosystem

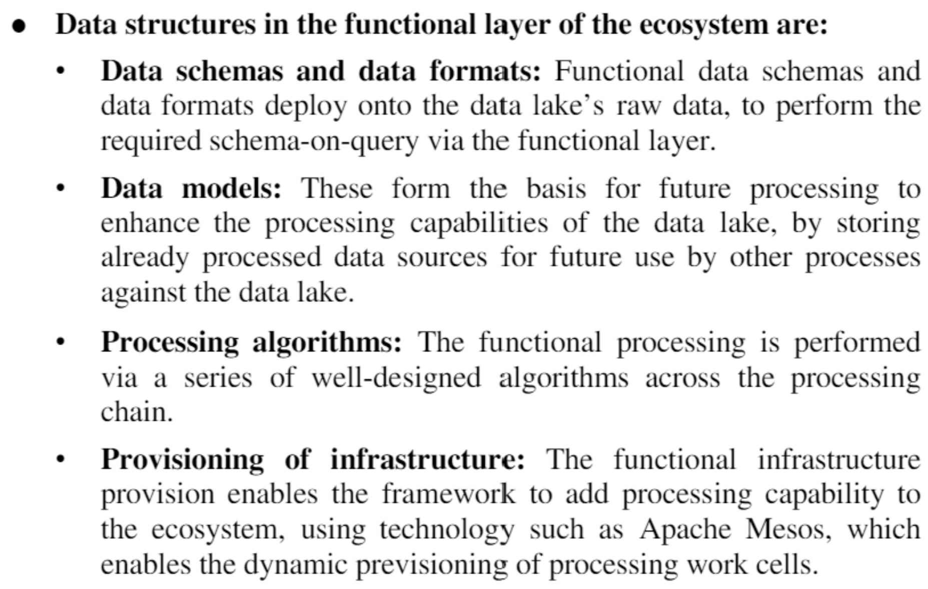
YOKE SOLUTION

- The yoke solution is a custom design
  - Apache Kafka is developed as
    - an open source stream processing platform.
    - Its function is to deliver a platform that is unified,
    - has high-throughput and low-latency for handling real-time data feeds.
  - Kafka provides a publish-subscribe solution that can handle all activity-stream data and processing.
    - The Kafka environment enables you to send messages
      - between producers and consumers that
      - enable you to transfer control between different parts of your ecosystem
      - while ensuring a stable process.
- PRODUCER:
  - The producer is the part of the system that generates the requests for data science processing, by creating structures messages for each type of data science process it requires.
  - The producer is the end point of the pipeline that loads messages into Kafka.
- CONSUMER:
  - The consumer is the part of the process that takes in messages and organizes them for processing by the data science tools.
  - The consumer is the end point of the pipeline that offloads the messages from Kafka.
- DIRECTED ACYLCIC GRAPH SCHED:
  - This solution uses a combination of graph theory and publish-subscribe stream data processing to enable scheduling.
  - You can use the Python NetworkX library to resolve any conflicts, by simply formulating the graph into a specific point before or after you

    - send or receive messages via Kafka.
    - That way, you ensure an effective and an efficient processing pipeline
- Cause-and-Effect Analysis System
    - The cause-and-effect analysis system is the part of the ecosystem that Collects all the logs, schedules, and other ecosystem-related information
    - Enables data scientists to evaluate the quality of their system.

DATA SCIENCE PROCESS:

- Following are the five fundamental data science process steps.
    - Begin process by asking a What if question
    - Attempt to guess at a probably potential pattern
    - Create a hypothesis by putting together observations
    - Verify the hypothesis using real-world evidence
    - Promptly and regularly collaborate with subject matter experts and
    - customers as and when you gain insights
- **Begin process by asking a What if question:**
    - Decide what you want to know, even if it is only the subset of the data lake you want to use for your data science, which is a good start.
- **Create a hypothesis by putting together observations:**
    - Use your experience or insights to guess a pattern you want to discover, to uncover additional insights from the data you already have
- **Gather Observations and Use Them to Produce a Hypothesis:**
    - A hypothesis, it is a proposed explanation, prepared on the basis of limited evidence, as a starting point for further investigation.
- **Verify the hypothesis using real-world evidence:**
    - Now, we verify our hypothesis by comparing it with real-world evidence
- **Promptly and regularly collaborate with subject matter experts and customers as and when you gain insights:**
    - Things that are communicated with experts may include technical aspects like workflows or more specifically data formats & data schemas.



- **Data structures in the functional layer of the ecosystem are:**
  - **Data schemas and data formats:** Functional data schemas and data formats deploy onto the data lake's raw data, to perform the required schema-on-query via the functional layer.
  - **Data models:** These form the basis for future processing to enhance the processing capabilities of the data lake, by storing already processed data sources for future use by other processes against the data lake.
  - **Processing algorithms:** The functional processing is performed via a series of well-designed algorithms across the processing chain.
  - **Provisioning of infrastructure:** The functional infrastructure provision enables the framework to add processing capability to the ecosystem, using technology such as Apache Mesos, which enables the dynamic previsioning of processing work cells.

FUNCTIONAL LAYER

- Largest and most essential layer
- Programming and modelling
- Must have processing elements in this
- "The functional layer is responsible for transforming raw unstructured data from the data lake into actionable business data, following the business strategy and requirements"
- How?
  - Retrieve
    - The Retrieve super step is a practical method for importing a data lake consisting of different external data sources completely into the processing ecosystem.
    - supports the interaction between external data sources and the factory
  - Assess
    - supports the data quality clean-up in the factory
    - Data Quality:
      - refers to the condition of a set of qualitative or quantitative variables
      - multidimensional measurement of the acceptability of specific data sets
    - If error:
      - Accept
      - Reject
      - Correct
      - Default value

- Process
    - converts data into data vault
    - The Process superstep adapts the assess results of the retrieve versions of the data sources into a highly structured data vault that will form the basic data structure for the rest of the data science steps.
    - This data vault involves the formulation of a standard data amalgamation format across a range of projects.
- Transform
    - converts data vault via sun modeling into dimensional modeling to form a data warehouse.
    - The Transform Superstep allow us to take data from data vault and answer the questions raised by the investigation.
    - It takes standard data science techniques and methods to attain insight and knowledge about the data that then can be transformed into actionable decisions. These results can be explained to non-data scientist.
- Organize
    - sub-divides the data warehouse into data marts
    - takes the complete data warehouse you built at the end of the Transform superstep and subsections it into business-specific data marts.
    - A data mart is the access layer of the data warehouse environment built to expose data to the users.
    - The data mart is a subset of the data warehouse and is generally oriented to a specific business group.
- Report
    - Virtualization
    - The Report superstep is the step in the ecosystem that enhances the data science findings with the art of storytelling and data visualization.
    - Appropriate Visualization
        - Eliminate Clutter
        - Draw Attention Where You Want It
        - Telling a Story (Freytag’s Pyramid)
        - Exposition
        - Rising Action
        - Climax
        - Falling Action
        - Resolution