Data Management in AI Projects

Data Repositories: Choosing the Right Solution

Learning Objective

By the end of this lesson, learners will be able to:

- Differentiate between Data Warehouses, Data Lakes, and Data Mesh.
- Apply a decision-making framework to choose the right data repository for a given use case.
- Implement hands-on coding exercises to compare data storage formats and simulate federated queries in a Data Mesh architecture.

What is a Data Repository?

A **data repository** is a storage system where data is collected, stored, and managed for future use. The choice of repository impacts:

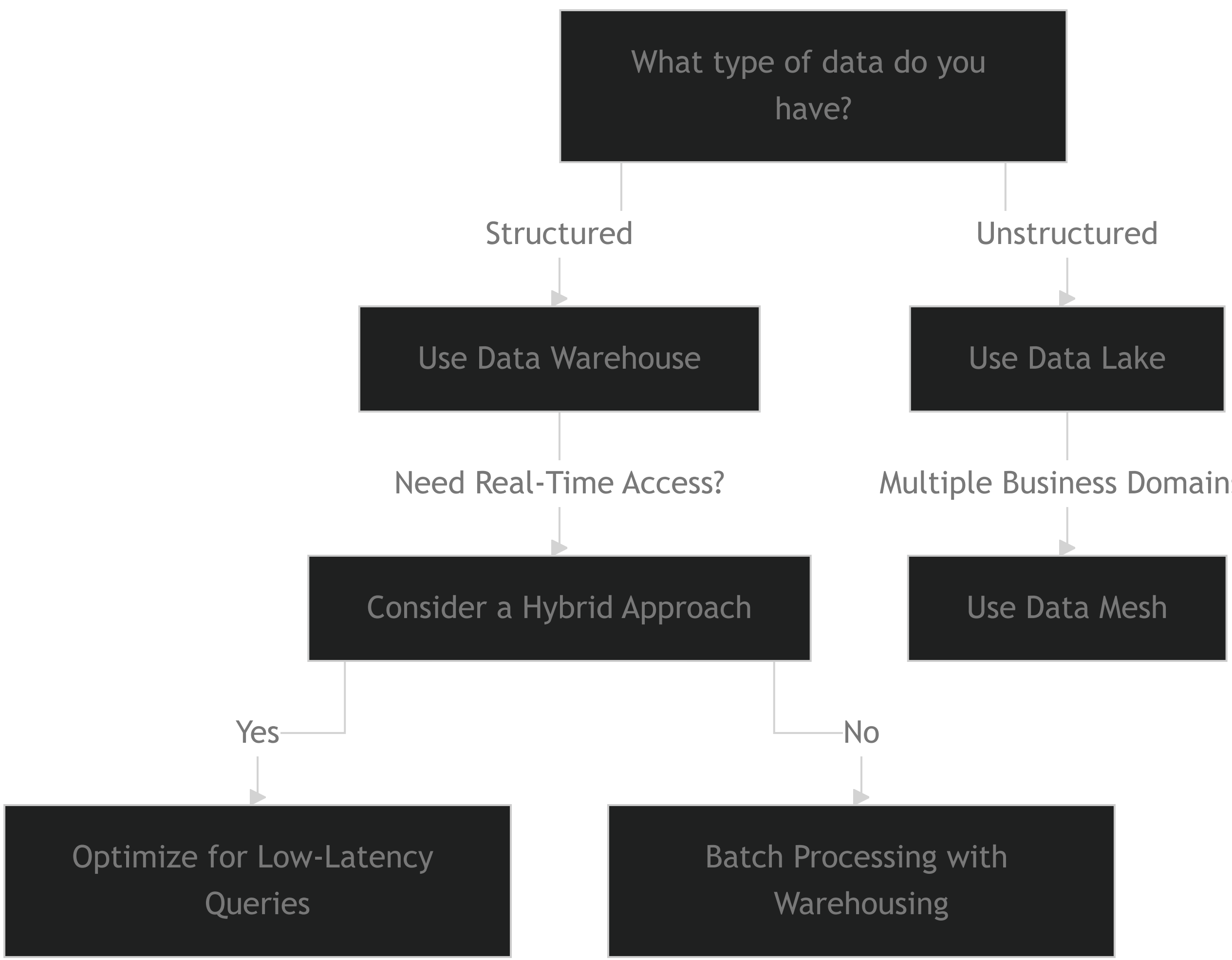
- **Scalability** – How much data can it handle efficiently?
- **Query Performance** – How fast can insights be extracted?
- **Data Structure** – Is the data raw and unstructured or pre-organized?

Types of Data Repositories

Repository Type	Characteristics	Best For
Data Warehouse	Structured, analytics-focused, schema-on-write	Business Intelligence, Historical Reporting
Data Lake	Raw, unstructured/semi-structured, schema-on-read	AI, Machine Learning, Real-time Processing
Data Mesh	Decentralized, domain-driven, federated architecture	Large organizations managing multiple data sources

Decision-Making Framework: How to Choose?

Use the following framework to decide the best data repository based on **data type, use case, and scalability needs**.



Key considerations when working with clients:

- Does the client need real-time analytics, or is batch processing sufficient?
- How does the organization handle **data governance and security** across teams?
- What are the **storage and query performance trade-offs** based on business needs?

Comparing Data Storage Formats

Scenario: Choosing Between a Warehouse & Lake

Your client, an e-commerce company, wants to store **customer transactions** for analytics and machine learning. Should they use a **Data Warehouse (CSV)** or **Data Lake (Parquet)?**

Code Implementation: Storing Data in Different Formats

Copy

```
import pandas as pd

# Sample transaction data
data = {
    "customer_id": [101, 102, 103],
    "purchase_amount": [250, 80, 150],
    "purchase_date": pd.to_datetime(["2024-01-10", "2024-02-15", "2024-03-05"])
}
df = pd.DataFrame(data)

# Store in CSV (Data Warehouse Format)
df.to_csv("transactions_warehouse.csv", index=False)

# Store in Parquet (Data Lake Format)
df.to_parquet("transactions_lake.parquet", index=False)
```

Follow-Up Discussion:

- **Query Performance**: CSV is human-readable but slower for large-scale queries. Parquet is optimized for analytics and storage efficiency.
- **Storage Costs**: Parquet files take up less space and load faster for ML pipelines.

- **Use Case Fit**: If the company needs **structured reporting**, go with a Warehouse. If they want **AI-driven personalization**, go with a Data Lake.

Key considerations when working with clients:

- Does the organization require a **single source of truth** (warehouse) or a **flexible storage format** (lake)?
- How will **data governance** be enforced in a decentralized setup?
- What are the **long-term scalability needs** for AI-driven analytics?

4. Hands-On: Simulating a Data Mesh with Federated Queries

Scenario: Federated Data Across Multiple Domains

Your client, a multinational retail company, has separate data teams for **Marketing, Sales, and Logistics**. Each team owns its data but wants to integrate it for AI-driven analytics.

Step 1: Creating Domain-Specific Data Files

Copy

```
import pandas as pd

# Marketing domain dataset
marketing_data = pd.DataFrame({
    "customer_id": [101, 102, 103],
    "campaign_clicks": [5, 2, 7],
    "ad_spend": [120, 80, 150]
})
marketing_data.to_parquet("marketing.parquet", index=False)

# Sales domain dataset
sales_data = pd.DataFrame({
    "customer_id": [101, 102, 104],
    "purchase_amount": [250, 80, 300],
    "purchase_date": pd.to_datetime(["2024-01-10", "2024-02-15", "2024-03-05"])
})
sales_data.to_parquet("sales.parquet", index=False)

# Logistics domain dataset
```

```
logistics_data = pd.DataFrame({
    "customer_id": [101, 103, 105],
    "delivery_time_days": [2, 5, 3],
    "return_status": [False, True, False]
})
logistics_data.to_parquet("logistics.parquet", index=False)
```

Step 2: Federated Query Across Domains

Copy

```
# Load all domain datasets
marketing_df = pd.read_parquet("marketing.parquet")
sales_df = pd.read_parquet("sales.parquet")
logistics_df = pd.read_parquet("logistics.parquet")

# Merge datasets using 'customer_id' as a key
merged_df = marketing_df \
    .merge(sales_df, on="customer_id", how="outer") \
    .merge(logistics_df, on="customer_id", how="outer")

# Display the combined federated dataset
print(merged_df)
```

Key considerations when working with clients:

- How do different teams ensure **consistent data governance** in a federated system?
- Should the client implement **APIs or centralized indexing** for better data discoverability?
- What security policies prevent unauthorized access to domain-specific data?

Case Study: Helping a Client Select the Right Repository

Client Scenario:

A **multinational retail company** wants to integrate customer behavior data across online and physical stores. They need:

- **Fast reporting on sales trends** for executives.

- **AI-driven product recommendations** based on purchase patterns.
- **Data governance across multiple teams & locations.**

What Would You Recommend?

Discuss in teams:

- Would a **Data Warehouse, Data Lake, or Data Mesh** be the best fit?
- What trade-offs does the company face in **speed, scalability, and data management?**
- How would you advise the client on **cost-effective and scalable storage solutions?**

Recap & Takeaways

✅ **Data Warehouses** are great for structured data and analytics-heavy use cases.
✅ **Data Lakes** handle unstructured data efficiently, making them ideal for AI & ML applications.
✅ **Data Mesh** is best for decentralized, multi-domain enterprise data architectures.
✅ Choosing the right repository **depends on business goals, scalability needs, and governance models.**

< Previous

© 2025 General Assembly
Attributions