

DATA LAKEHOUSE FOR SHIPPING DELIVERY MANAGEMENT

Group 13

Tra Trung Tin : 2010702
Dang Lam Tung : 2370506
Nguyen Ngoc Gia Van : 2370700

Under the Guidance of
Dr.Phan Trong Nhan, CSE-HCMUT



Faculty of Computer Science and Engineering
Ho chi Minh University of Technology

Overview

1 Introduction

- Overview of delivery shipping management
- Objective of implementing data lakehouse

2 Prepare and generate data source

- Structured data
- Unstructured data

3 Technology Implementation

- Data lakehouse technologies
 - OCR technology
 - Storage
 - Table Formats
 - Query Engine
 - BI Tool
- Implementation

4 BI and report

- BI and report
- Cancel by store
- Round trip and Single trip
- Mean of duration of shipping service and staff
- The number of delivery staff for each store
- Income by date

Overview of delivery shipping management:

The delivery system will record all things related to the delivery orders. Whenever a delivery order is well prepared by the store, a staff member will call a shipping service, provided by a third party, to deliver that order to the customer's address.

- Shipping mode
- Coordination with Third-Party Shipping Services
- Record shipping information (paper note)

Objective of implementing data lakehouse:

- Unified Data Management
- Scalability and Flexibility
- Data Quality and Consistency
- Advanced Analytics and Insights

Prepare and generate data source

Structured data

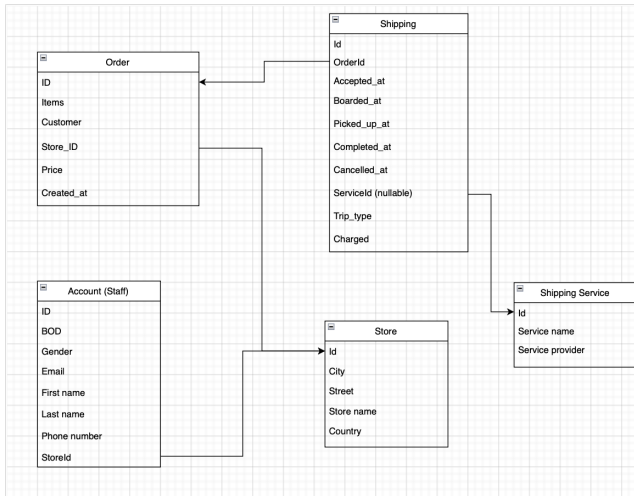


Figure: Relational database

Prepare and generate data source

Unstructured data

Amazon S3

Objects (100) info

Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

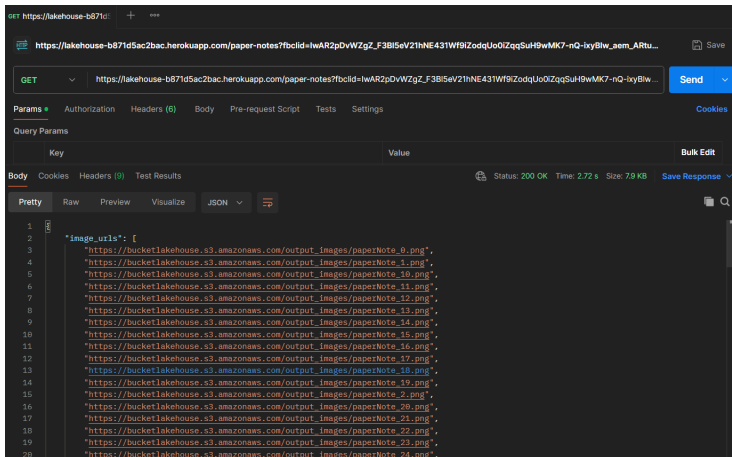
Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	paperNote_31.png	png	(UTC+07:00)	7.4 KB	Standard
<input type="checkbox"/>	paperNote_32.png	png	March 29, 2024, 21:02:14 (UTC+07:00)	7.5 KB	Standard
<input type="checkbox"/>	paperNote_33.png	png	March 29, 2024, 21:02:11 (UTC+07:00)	7.3 KB	Standard
<input type="checkbox"/>	paperNote_34.png	png	March 29, 2024, 21:02:04 (UTC+07:00)	7.3 KB	Standard
<input type="checkbox"/>	paperNote_35.png	png	March 29, 2024, 21:02:02 (UTC+07:00)	7.4 KB	Standard
<input type="checkbox"/>	paperNote_36.png	png	March 29, 2024, 21:01:58 (UTC+07:00)	7.5 KB	Standard
<input type="checkbox"/>	paperNote_37.png	png	March 29, 2024, 21:02:01 (UTC+07:00)	7.3 KB	Standard
<input type="checkbox"/>	paperNote_38.png	png	March 29, 2024, 21:02:21 (UTC+07:00)	7.4 KB	Standard
<input type="checkbox"/>	paperNote_39.png	png	March 29, 2024, 21:02:18 (UTC+07:00)	7.5 KB	Standard

Figure: Paper notes store in S3 bucket

Prepare and generate data source

Unstructured data



The screenshot shows a REST client interface with a GET request to the URL: `https://lakehouse-b871d5ac2bac.herokuapp.com/paper-notes?fbclid=IwAR2pDvWZgZ_F3BI5eV21hNE431Wf9IZodqUo0iZqqSuH9wMK7-nQ-ixyBlw_aem_Artu...`. The response status is 200 OK, and the body is displayed in a JSON format. The JSON contains a single key, `image_urls`, which maps to an array of 25 image URLs. Each URL follows the pattern: `https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_0.png` through `paperNote_24.png`.

```
1 {
2   "image_urls": [
3     "https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_0.png",
4     "https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_1.png",
5     "https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_10.png",
6     "https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_11.png",
7     "https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_12.png",
8     "https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_13.png",
9     "https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_14.png",
10    "https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_15.png",
11    "https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_16.png",
12    "https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_17.png",
13    "https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_18.png",
14    "https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_19.png",
15    "https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_2.png",
16    "https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_20.png",
17    "https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_21.png",
18    "https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_22.png",
19    "https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_23.png",
20    "https://bucketlakehouse.s3.amazonaws.com/output_images/paperNote_24.png",
```

Figure: Paper notes URL

OCR Technology

To process unstructured data in the form of papernote, we choose the PaddleOCR package, which contain the whole OCR pipeline for easier detection of the data. After extracted the data in the form of text string from the OCR, we also use FuzzyMatching to fix any wrong character (if any).

```
Date time details:  
Accepted at: 2023-12-12 12:35 PM  
Completed at: 2023-12-12 01:35 PM  
Boarded at: 2023-12-12 03:10 PM  
Picked up at: 2023-12-12 04:09 PM
```

```
Customer details:  
Name: Kimberly Nichols  
Address: 02898 Hardin Islands  
Phone: 434-778-5842x9825  
Email: anthony97@example.net
```

```
Order details:  
Name: job girl  
Price: $17.72  
Trip type: Single
```

```
Staff details:  
Name: Emily Cobb  
Phone: 608-913-5087x000  
Email: valerie81@example.org
```

Figure: Example papernote

OCR Pipeline

Data after processed by the OCR pipeline will be add to the ETL pipeline to push into the warehouse part. The structure of the papernotes are the same as the database for easier process data.

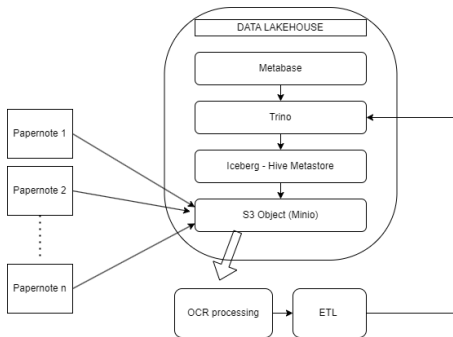
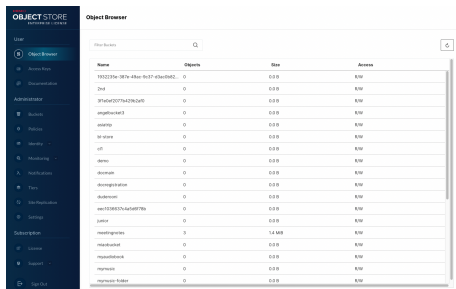


Figure: OCR pipeline

Data lakehouse technologies

For the storage layer, Minio has been selected, an open-source, S3-compliant object storage solution.



The screenshot shows the MinIO Object Browser interface. On the left is a dark sidebar with navigation options: Object Browser (selected), Access Keys, Documentation, Administrator, Buckets, Policies, Identity, Monitoring, Notifications, Tiers, Site Replication, Settings, Subscriptions, Logs, and Support. The main area is titled 'Object Browser' and contains a search bar and a table of objects.

Name	Objects	Size	Access
1902235e-087e-48ae-9c37-43ec5b402...	0	0.0 B	R/W
2nd	0	0.0 B	R/W
3f1af6af2077b429b24d5	0	0.0 B	R/W
angelflower3	0	0.0 B	R/W
anilap	0	0.0 B	R/W
bi-vare	0	0.0 B	R/W
git	0	0.0 B	R/W
demo	0	0.0 B	R/W
download	0	0.0 B	R/W
decomposition	0	0.0 B	R/W
download	0	0.0 B	R/W
elec1218527a7a09f709	0	0.0 B	R/W
junior	0	0.0 B	R/W
meetingnotes	3	1.4 MB	R/W
mindcloud	0	0.0 B	R/W
myanbuback	0	0.0 B	R/W
mypraxis	0	0.0 B	R/W
mypraxis-folder	0	0.0 B	R/W

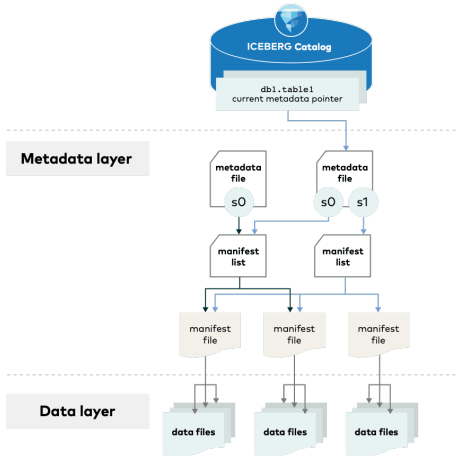
Figure: MinIO dashboard

Key characteristics of MinIO:

- S3 Compatibility
- High Performance Optimization
- Distributed Architecture
- Implementation of Erasure Coding
- Data Lifecycle Management
- Encryption Capabilities
- Event Handling and Webhooks

Table Formats

Table formats are instrumental in organizing data files and bringing database-like features to the data lake, a key distinction between a Data Lake and a Lakehouse. Apache Iceberg has been chosen for this project due to several advantages over Hudi and Delta tables. Iceberg offers faster insert and update operations, requires less storage, and facilitates easy partition and schema evolution across multiple query engines.



Component of Iceberg table format

- **Snapshot metadata file:** contains metadata about the table like the table schema, the partition specification as well as a path to the manifest list.
 - **Manifest list:** contains an entry for each manifest file associated with the snapshot. Each entry includes a path to the manifest file and some metadata about the file, including partition stats and data file counts. These stats can be used to avoid reading manifests that aren't required for an operation.
 - **Manifest file:** contains a list of paths to related data files. Each entry for a data file includes some metadata about the file, such as per-column upper and lower bounds which can be used to prune files during query planning.
 - **Data file:** the physical data file, written in formats like Parquet, ORC, Avro etc.
- decorative separator

Benefit of Iceberg

- Using the snapshot pattern means that Iceberg can guarantee isolated reads and writes. Readers will always see a consistent version of the data (i.e. no 'dirty reads') without the need to lock the table. Writers work in isolation, not affecting the live table and will perform a metadata swap only when the write is complete, making the changes in one atomic commit. Use of snapshots also enables time-travel operations as users can perform various operations on different versions of the table by specifying the snapshot to use.
- **Huge performance benefits to using Iceberg.** Instead of listing $O(n)$ partitions in a table during job planning, Iceberg performs an $O(1)$ RPC to read the snapshot. The file pruning and predicate pushdown can also be distributed to jobs so the Hive metastore is no longer a bottleneck. This also removes the barriers to using finer-grained partitioning. The file pruning available due to the statistics stored for each data file also speeds up query planning significantly.

Query Engine

Trino serves as the query engine, enabling the creation and querying of tables based on flat files via Iceberg. It supports query federation, allowing for the joining of data from multiple sources. Trino is deployed in a Docker container and operates as a massively parallel processing database query engine, scaling vertically instead of increasing the processing power of a single node. Its connector-based architecture translates SQL statements into API calls, returning collections of rows in columnar format (pages) for processing by worker nodes based on the query or execution plan.

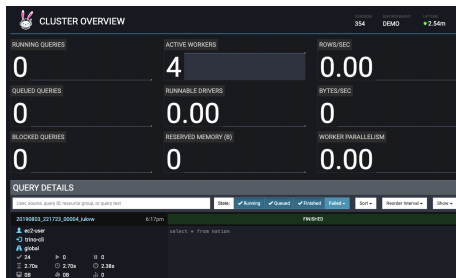


Figure: Trino Dashboard

BI Tool

Metabase has been added and pre-configured as a BI tool for direct interaction with the Lakehouse. Metabase serves as a robust open-source solution for data visualization and exploration, offering users the ability to seamlessly connect to various data sources, construct interactive dashboards, and derive insights with efficiency. Designed with accessibility in mind, Metabase caters to individuals across different proficiency levels, providing a user-friendly platform for data interaction.

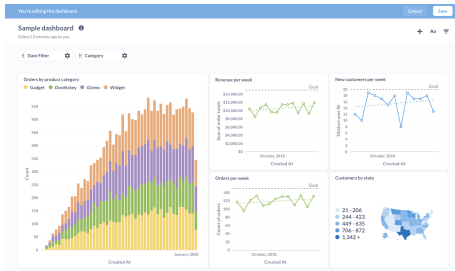


Figure: Metabase example dashboard

Key features of Metabase include:

- Simple Query Building
- Interactive Dashboard Creation
- Support for Multiple Data Sources
- Facilitated Data Exploration
- Intuitive Interface
- Collaborative Data Sharing

Data pipeline

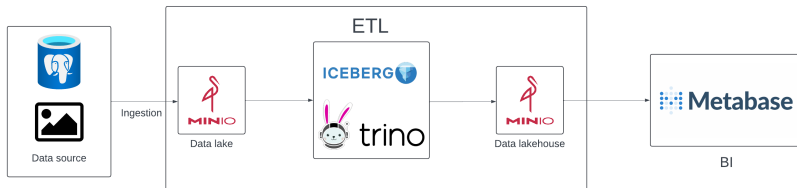


Figure: Data pipeline

Dashboard

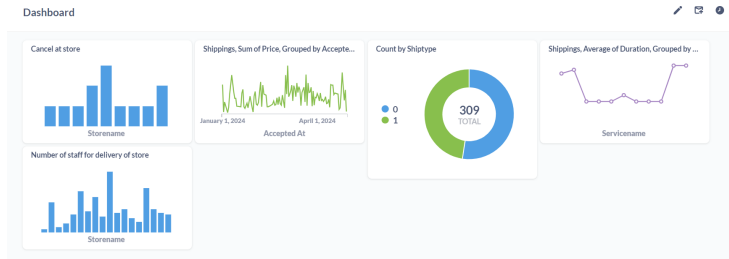


Figure: Cancelled by store

Cancel by store

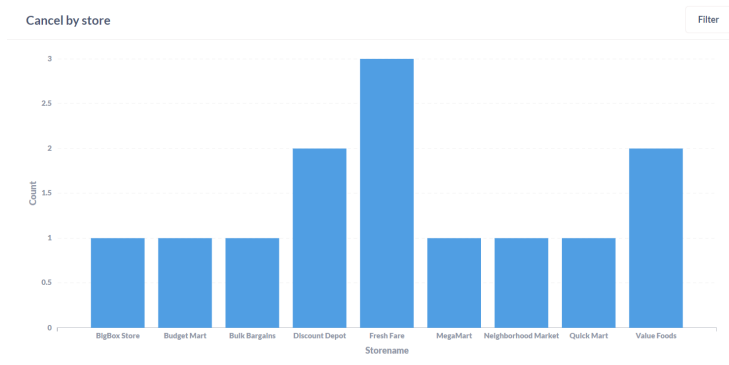


Figure: Cancelled by store

Round trip and Single trip

Count by Shiptype

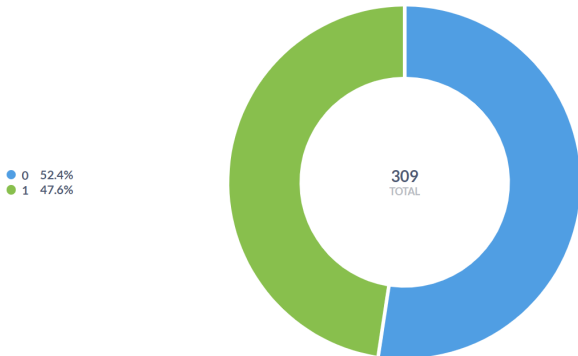


Figure: Round trip and Single trip

Mean of duration of shipping service and staff

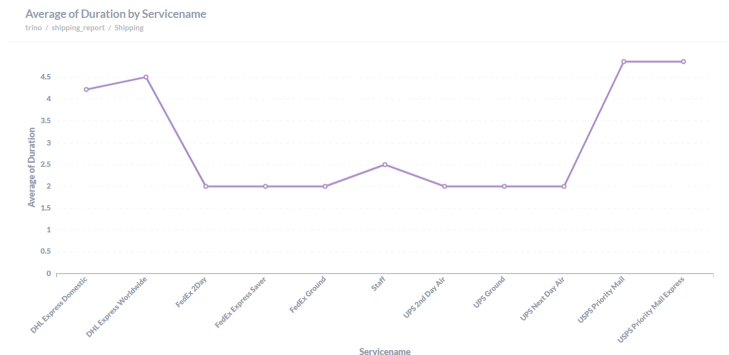


Figure: Duration of shipping

The number of delivery staff for each store

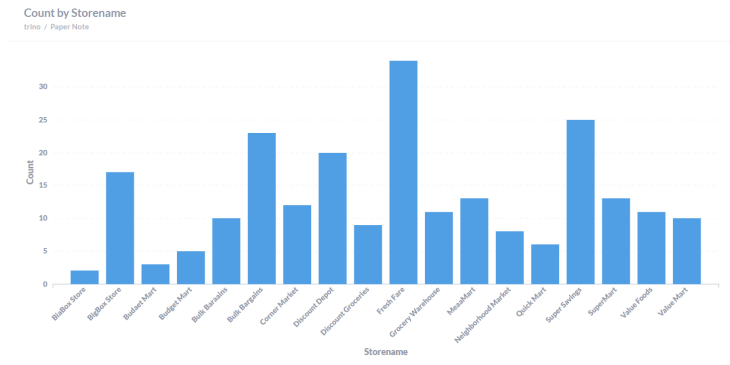


Figure: Count staff by store

Income by date

Sum of Price by Accepted At: Day

trino / shipping_report / Shipping

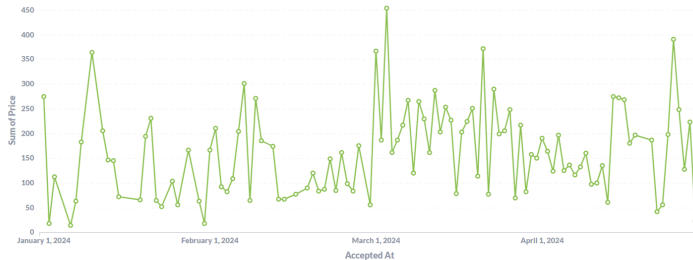


Figure: Income by date

The End