

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN



NGUYỄN THANH HÙNG: 20133045

NGUYỄN DUY THÁI: 20133020

Đề tài:

**TÌM HIỂU MODERN DATA STACK
VÀ XÂY DỰNG ỨNG DỤNG
DATA PLATFORM**

**TIỂU LUẬN CHUYÊN NGÀNH
NGÀNH KỸ THUẬT DỮ LIỆU**

GIÁO VIÊN HƯỚNG DẪN

ThS. Nguyễn Văn Thành

KHÓA 2020 – 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN



NGUYỄN THANH HÙNG: 20133045

NGUYỄN DUY THÁI: 20133020

Đề tài:

**TÌM HIỂU MODERN DATA STACK
VÀ XÂY DỰNG ỨNG DỤNG
DATA PLATFORM**

**TIỂU LUẬN CHUYÊN NGÀNH
NGÀNH KỸ THUẬT DỮ LIỆU**

GIÁO VIÊN HƯỚNG DẪN

ThS. Nguyễn Văn Thành

KHÓA 2020 – 2024

PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Họ và tên sinh viên 1: Nguyễn Thanh Hùng

MSSV: 20133045

Họ và tên sinh viên 2: Nguyễn Duy Thái

MSSV: 20133020

Ngành: Kỹ thuật dữ liệu

Tên đề tài: TÌM HIỂU MODERN DATA STACK VÀ XÂY DỰNG ỨNG DỤNG DATA PLATFORM

Họ và tên giáo viên hướng dẫn: Nguyễn Văn Thành

NHẬN XÉT

1. Về nội dung đề tài khối lượng thực hiện:

.....
.....
.....

2. Ưu điểm:

.....
.....
.....

3. Khuyết điểm:

.....
.....
.....

4. Đề nghị có cho bảo vệ hay không ?

5. Đánh giá loại:

6. Điểm:

Tp. Hồ Chí Minh, ngày tháng năm 2023

Giáo viên hướng dẫn

(Ký & ghi rõ họ tên)

PHIẾU NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

Họ và tên sinh viên 1: Nguyễn Thanh Hùng

MSSV: 20133045

Họ và tên sinh viên 2: Nguyễn Duy Thái

MSSV: 20133020

Ngành: Kỹ thuật dữ liệu

Tên đề tài: TÌM HIỂU MODERN DATA STACK VÀ XÂY DỰNG ỨNG DỤNG DATA PLATFORM

Họ và tên giáo viên phản biện: Trần Nhật Quang

NHẬN XÉT

1. Về nội dung đề tài khối lượng thực hiện:

.....
.....
.....

2. Ưu điểm:

.....
.....
.....

3. Khuyết điểm:

.....
.....
.....

4. Đề nghị có cho bảo vệ hay không ?

5. Đánh giá loại:

6. Điểm:

Tp. Hồ Chí Minh, ngày tháng năm 2023

Giáo viên hướng dẫn

(Ký & ghi rõ họ tên)

LỜI CAM ĐOAN

Tiểu luận này là công trình nghiên cứu của chúng em, được thực hiện dưới sự hướng dẫn khoa học của thầy Nguyễn Văn Thành. Các số liệu, những kết luận nghiên cứu và sản phẩm được tạo ra bởi chúng em được trình bày trong khoá luận này là trung thực.

Chúng em xin hoàn toàn chịu trách nhiệm về lời cam đoan này.

Sinh viên thực hiện
(Ký & ghi rõ họ tên)

Nguyễn Thanh Hùng

Sinh viên thực hiện
(Ký & ghi rõ họ tên)

Nguyễn Duy Thái

LỜI CẢM ƠN

Một kỳ thực hiện tiểu luận chuyên ngành đã trôi qua nhưng để lại trong chúng em rất nhiều cảm xúc. Chúng em xin được gửi lời cảm ơn chân thành nhất đến Thầy Nguyễn Văn Thành. Thầy đã hướng dẫn tận tình cho chúng em trong suốt quá trình thực hiện tiểu luận chuyên ngành, Thầy luôn theo dõi tiến độ và giải đáp, chia sẻ giúp chúng em vượt qua những khó khăn. Chúng em rất trân quý sự tâm huyết và trách nhiệm của Thầy trong công việc giảng dạy và truyền đạt kiến thức.

Chúng em cũng xin gửi lời cảm ơn sâu sắc đến Thầy Cô khoa Công nghệ thông tin trường Đại học sư phạm kỹ thuật TP.HCM đã đồng hành và hỗ trợ chúng em trong suốt quá trình học tập và thực hiện tiểu luận. Chúng em xin cảm ơn trường Đại học sư phạm kỹ thuật đã tạo nhiều điều kiện thuận lợi cho hoạt động phục vụ học tập của sinh viên chúng em, đặc biệt là thư viện số với nguồn tri thức vô tận. Chúng em cũng gửi lời cảm ơn chân thành đến các bạn khóa 2020 ngành Kỹ thuật Dữ liệu, cảm ơn những góp ý và chia sẻ quý giá từ tất cả các bạn. Cảm ơn sự động viên từ các bạn để nhóm chúng em có thể giữ vững tinh thần và thực hiện khóa luận đúng tiến độ.

Những giá trị cốt lõi nhà trường, Thầy Cô và bạn bè mang đến, chúng em sẽ luôn ghi nhớ để làm động lực thúc đẩy bản thân phát triển và hoàn thiện hơn nữa. Cuối cùng, chúng em xin cảm ơn đến tác giả của những bài báo khoa học mà chúng em đã tham khảo. Các bài báo này giúp chúng em tiếp thu được nhiều kiến thức mới và quan trọng là hiểu rõ hơn về đề tài đang nghiên cứu. Chúng em cũng nhận thấy bản thân có những khuyết điểm và thiếu sót cần cố gắng cải thiện để tốt hơn, hướng tới mục tiêu lớn trong tương lai.

Xin chân thành cảm ơn!

KẾ HOẠCH THỰC HIỆN

Thời gian	Công việc
Tuần 1 (21/08/2023 đến 27/08/2023)	<ul style="list-style-type: none"> • Chọn, xác định đề tài với giáo viên hướng dẫn. • Tìm hiểu khái niệm, đặc điểm và lợi ích của MDS. • So sánh với các hệ thống dữ liệu truyền thống. • Tìm hiểu các công cụ và nền tảng hỗ trợ MDS. • Tìm hiểu về các tools phổ biến hiện nay trong MDS như: dagster, minio, docker...
Tuần 2 (28/08/2023 đến 03/09/2023)	<ul style="list-style-type: none"> • Tìm hiểu khái niệm, đặc điểm và lợi ích của kiến trúc Data Lakehouse. • So sánh và đánh giá sự khác biệt với các kiến trúc dữ liệu khác như Data Warehouse, Data Lake, Data Mesh. • Tìm hiểu các công nghệ và giải pháp hỗ trợ kiến trúc Lakehouse. • Tìm hiểu docker và công nghệ ảo hóa.
Tuần 3: (04/09/2023 đến 10/09/2023)	<ul style="list-style-type: none"> • Lựa chọn công cụ MDS để thực hành. • Cài đặt và cấu hình công cụ. • Tìm hiểu cách kết nối, thu thập, xử lý và lưu trữ dữ liệu từ các nguồn khác nhau.
Tuần 4: (11/09/2023 đến 17/09/2023)	<ul style="list-style-type: none"> • Tìm hiểu cách tạo, quản lý và truy vấn dữ liệu theo mô hình Lakehouse. • Xây dựng 1 data pipeline đơn giản.
Tuần 5: (18/09/2023 đến 24/09/2023)	<ul style="list-style-type: none"> • Tìm hiểu về domain data main ecommerce. Tìm hiểu về các nguồn dữ liệu, bài toán, yêu cầu và giải pháp liên quan đến domain này. • Thực hiện một dự án nhỏ để minh họa cách ứng dụng MDS và

	kiến trúc Lakehouse vào xây dựng Platform. Chọn một lĩnh vực hoặc một bài toán cụ thể để giải quyết bằng dữ liệu.
Tuần 6: (25/09/2023 đến 01/10/2023)	<ul style="list-style-type: none"> • Tìm hiểu cách xây dựng một Platform với kiến trúc Lakehouse. • Tìm hiểu các ví dụ và best practices của các Platform với kiến trúc Lakehouse đã được triển khai trong thực tế. • Tìm hiểu về kiến trúc Medallion trong Data Lakehouse.
Tuần 7: (02/10/2023 đến 08/10/2023)	<ul style="list-style-type: none"> • Lựa chọn một use case phù hợp để xây dựng một Platform với kiến trúc Lakehouse. • Phân tích yêu cầu, mục tiêu và giải pháp cho use case đó. • Thiết kế một bản vẽ tổng quan cho Platform với kiến trúc Lakehouse cho use case đó.
Tuần 8: (09/10/2023 đến 15/10/2023)	<ul style="list-style-type: none"> • Lựa chọn các công cụ và nền tảng trong MDS để xây dựng Platform với kiến trúc Lakehouse cho use case đã lựa chọn. • Cài đặt và cấu hình các công cụ và nền tảng đó. • Tạo một dự án mới để bắt đầu xây dựng Platform với kiến trúc Lakehouse.
Tuần 9: (16/10/2023 đến 22/10/2023)	<ul style="list-style-type: none"> • Xây dựng data layer cho Platform với kiến trúc Lakehouse. • Kết nối, biến đổi và lưu trữ dữ liệu từ các nguồn dữ liệu khác nhau vào data layer. • Xây dựng các Data Pipeline để tự động hóa quá trình này. • Áp dụng các kỹ thuật và tiêu chuẩn để đảm bảo chất lượng, tính nhất quán và tính an toàn của dữ liệu.
Tuần 10: (23/10/2023 đến 29/10/2023)	<ul style="list-style-type: none"> • Xây dựng analytics layer cho Platform với kiến trúc Lakehouse. • Trích xuất, biến đổi và tải dữ liệu từ data layer vào analytics layer.

Tuần 11: (30/10/2023 đến 05/11/2023)	<ul style="list-style-type: none"> • Xây dựng các data pipeline để tự động hóa quá trình này. • Áp dụng các kỹ thuật và tiêu chuẩn để đảm bảo tính thân thiện, tính tương tác và tính giá trị của ứng dụng.
Tuần 12: (06/11/2023 đến 12/11/2023)	<ul style="list-style-type: none"> • Kiểm thử và đánh giá Platform với kiến trúc Lakehouse.
Tuần 13: (13/11/2023 đến 19/11/2023)	<ul style="list-style-type: none"> • Tối ưu hóa và cải tiến Platform. Dựa trên kết quả kiểm thử và đánh giá, xác định các điểm mạnh, điểm yếu, cơ hội và thách thức của Platform với kiến trúc Lakehouse. Thực hiện các biện pháp để tối ưu hóa và cải tiến các thành phần, quá trình và kết quả của Platform với kiến trúc Lakehouse. • Sử dụng các công cụ phân tích và biểu diễn dữ liệu Metabase, để tạo ra các báo cáo, dashboard và insight từ dữ liệu.
Tuần 14: (20/11/2023 đến 26/11/2023)	<ul style="list-style-type: none"> • Hoàn thiện và báo cáo kết quả. • Viết báo cáo, tài liệu và slide. • Thực hiện demo và thuyết trình thử cho GVHD.
Tuần 15: (27/11/2023 đến 3/12/2023)	<ul style="list-style-type: none"> • Viết báo cáo tổng kết quá trình làm đề tài, kết quả đạt được và hướng phát triển tiếp theo. Thực hiện bảo vệ đề tài trước hội đồng giáo viên và nhận xét của các bạn học. • Chuẩn bị và thuyết trình cho đề tài. Chuẩn bị các tài liệu, thiết bị và kỹ năng cần thiết để thuyết trình cho đề tài. Thuyết trình theo nội dung, thời gian và yêu cầu đã được quy định.

DANH MỤC HÌNH ẢNH

Hình 1.2.1. Chi tiết các thành phần của một Data Platform	17
Hình 1.4.1.1. Logo của Apache Spark	22
Hình 1.4.1.2. Các thành phần chính của Apache Spark	22
Hình 1.4.2.1. Logo của thư viện Polars	23
Hình 1.4.3.1. Logo Minio	24
Hình 1.4.4.1. Logo Delta Lake	25
Hình 1.4.5.1. Logo Hive Metastore	26
Hình 1.4.6.1. Logo Metabase	27
Hình 1.4.7.1. Logo Docker	28
Hình 1.4.7.2. Kiến trúc của Docker	28
Hình 1.4.8.1. Logo Dagster	30
Hình 1.4.8.2. Data Pipeline cơ bản của Dagster	30
Hình 1.4.8.3. Màn hình chạy trên tất cả công việc ở một nơi với chế độ xem dòng thời gian	31
Hình 2.1.1.1. Kiến trúc 2 tầng hiện tại	33
Hình 2.1.1.2. Kiến trúc Data Lakehouse bởi Databricks	34
Hình 2.1.2.1. Kiến trúc Data Warehouse, Data Lake và Data Lakehouse	35
Hình 2.2.1. Kiến trúc Data Lakehouse	37
Hình 3.1.1.1. Kiến trúc tổng quan về hệ thống Data Platform	41
Hình 3.1.2.1. Lược đồ Diagram giữa các bảng trong cơ sở dữ liệu MySQL	44
Hình 3.2.1.1. Extract data từ nguồn vào Data Lake	44
Hình 3.2.1.2. Load data into Bronze Layer	45
Hình 3.2.1.3. Transform Data từ Bronze Layer to Silver Layer	45
Hình 3.2.1. 4. Định dạng file lưu trữ của Delta Lake	46
Hình 3.2.1.5. Vai trò của Delta Lake trong kiến trúc Lakehouse	46
Hình 3.2.1.6. Sử dụng Spark để đọc file định dạng Parquet	47
Hình 3.2.1.7. Cấu trúc của 1 file parquet	47
Hình 3.2.1.8. Bên trong thư mục _delta_log	47
Hình 3.2.1.9. File log lưu trữ nhật ký	48
Hình 3.2.1.10. Table Format khi load lại data	48
Hình 3.2.1.11. Thư mục _delta_log khi load lại data	48
Hình 3.2.2.1. Select tất cả record của bảng clean_customer	49
Hình 3.2.2.2. Update một record trong bảng clean_customer	49
Hình 3.2.2.3. Select xem tất cả có bao nhiêu bản ghi và giá trị duy nhất của từng bản ghi	50
Hình 3.2.2.4. Bảng metadata DBS	50

Hình 3.2.2.5. Bảng metadata SDS.....	50
Hình 3.3.2.1. Bảng thiết kế Data Warehouse	53
Hình 3.4.1. Data Lineage của Bronze Layer	55
Hình 3.4.2. Data Lineage của Silver Layer	55
Hình 3.4.3. Data Lineage của Gold Layer.....	55
Hình 3.5.1. Báo cáo dữ liệu về việc quản lý các đơn hàng	56
Hình 3.5.2. Báo cáo dữ liệu về việc phân tích sản phẩm	57
Hình 3.5.3. Báo cáo dữ liệu về phân tích doanh thu và khách hàng	58

DANH MỤC BẢNG BIỂU

Bảng 1.2.1. Bảng so sánh giữa Traditional Data Platform và Modern Data Platform	15
Bảng 2.1.2.1. Bảng so sánh Data Lake, Data Warehouse và Data Lakehouse.....	36

DANH MỤC VIẾT TẮT

AI	Artificial Intelligence (Trí tuệ nhân tạo)
ACID	Atomicity (tính nguyên tử), Consistency (tính nhất quán), Isolation (tính cô lập), Durability (độ bền vững)
BI	Bussiness Intelligence
DS	Data Science
ETL	Extract, Transform and Load
ELT	Extract, Load and Transform
MDS	Modern Data Stack
ML	Machine Learning

MỤC LỤC

LỜI CẢM ƠN	1
KẾ HOẠCH THỰC HIỆN.....	2
DANH MỤC HÌNH ẢNH.....	5
DANH MỤC BẢNG BIỂU.....	7
DANH MỤC VIẾT TẮT	8
MỤC LỤC.....	9
PHẦN 1: MỞ ĐẦU.....	11
1. Tính cấp thiết của đề tài	11
2. Mục đích của đề tài	11
3. Phương pháp nghiên cứu.....	12
4. Kết quả dự kiến	12
5. Bố cục luận văn.....	12
PHẦN 2: NỘI DUNG	14
CHƯƠNG 1: TỔNG QUAN VỀ MODERN DATA STACK	14
1.1. KHÁI NIỆM VỀ MODERN DATA STACK.....	14
1.2. THÀNH PHẦN CỦA MODERN DATA STACK	14
1.3. SO SÁNH MODERN DATA STACK VÀ TRADITIONAL DATA STACK.....	20
1.4. CÁC CÔNG NGHỆ SỬ DỤNG	22
1.4.1. Apache Spark	22
1.4.2. Polars.....	23
1.4.3. Minio.....	24
1.4.4. Delta Lake	25
1.4.5. Hive Metastore	26
1.4.6. Metabase.....	27
1.4.7. Docker	28
1.4.8. Dagster	30
CHƯƠNG 2: KIẾN TRÚC DATA LAKEHOUSE	32
2.1. TỔNG QUAN VỀ KIẾN TRÚC DATA LAKEHOUSE.....	32
2.1.1. Sự ra đời và khái niệm về Data Lakehouse	32
2.1.2. So sánh Data LakeHouse với Data Lake và Data Warehouse.....	35
2.2. CÁC THÀNH PHẦN CỦA DATA LAKEHOUSE	36
2.3. KIẾN TRÚC MEDALLION ARCHITECTURE.....	39
CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG DATA PLATFORM	41
3.1. KIẾN TRÚC HỆ THỐNG	41
3.1.1. Kiến trúc tổng quan	41

3.1.2. Tập dữ liệu nguồn	42
3.2. QUY TRÌNH HOẠT ĐỘNG CỦA KIẾN TRÚC LAKEHOUSE TRONG ỨNG DỤNG PLATFORM	44
3.2.1. Tổng quan về các quy trình	44
3.2.2. Thực hiện truy vấn và quản lí metadata của dự án	49
3.3. THIẾT KẾ DATA WAREHOUSE	51
3.3.1. Dimensional Data Modeling	51
3.3.2. Bảng thiết kế Data Warehouse	53
3.4. DATA LINEAGE	55
3.5. XÂY DỰNG BẢNG BÁO CÁO	55
PHẦN 3: KẾT LUẬN	59
1. Kết quả đạt được	59
2. Hạn chế	59
3. Hướng phát triển	59
TÀI LIỆU THAM KHẢO	61

PHẦN 1: MỞ ĐẦU

1. Tính cấp thiết của đề tài

Trong thời đại công nghệ số hiện nay, dữ liệu được coi là tài sản quan trọng của các doanh nghiệp, tổ chức và cá nhân. Việc thu thập, lưu trữ, xử lý và phân tích dữ liệu một cách hiệu quả và an toàn là yếu tố then chốt để tạo ra giá trị kinh doanh, cải thiện chất lượng sản phẩm và dịch vụ, hỗ trợ quyết định chiến lược và nâng cao năng lực cạnh tranh. Tuy nhiên, việc xây dựng một hệ thống quản lý dữ liệu truyền thống đòi hỏi nhiều chi phí, thời gian và nhân lực, đồng thời phải đối mặt với nhiều thách thức như khả năng mở rộng, tính linh hoạt, bảo mật và tuân thủ pháp luật.

Trước bối cảnh đó, xuất hiện một xu hướng mới trong lĩnh vực quản lý dữ liệu là Modern Data Stack, hay còn gọi là ngăn xếp dữ liệu hiện đại. MDS là một tập hợp các công cụ và giải pháp được thiết kế để giải quyết các vấn đề của hệ thống quản lý dữ liệu truyền thống, bằng cách tận dụng các tiến bộ của công nghệ đám mây, máy học và trí tuệ nhân tạo. MDS cho phép các doanh nghiệp, tổ chức và cá nhân có thể thu thập, lưu trữ, xử lý và phân tích dữ liệu một cách nhanh chóng, tiết kiệm chi phí, dễ dàng tích hợp và tuân thủ các tiêu chuẩn bảo mật và pháp lý.

Đề tài "Tìm hiểu Modern Data Stack và ứng dụng xây dựng Data Platform" nhằm mục đích nghiên cứu về khái niệm, thành phần, ưu điểm và nhược điểm của MDS, cũng như các ví dụ về việc áp dụng MDS để xây dựng data platform cho các doanh nghiệp, tổ chức và cá nhân. Đề tài có tính cấp thiết và động lực cao vì MDS là một xu hướng mới và tiềm năng trong lĩnh vực quản lý dữ liệu, có thể mang lại nhiều lợi ích cho người sử dụng. Đồng thời, đề tài cũng góp phần nâng cao kiến thức và kỹ năng của người nghiên cứu về MDS, data platform và các công cụ liên quan.

2. Mục đích của đề tài

Mục tiêu: Xây dựng một ứng dụng Data Platform dựa trên Modern Data Stack.

Nhiệm vụ:

- Tìm hiểu về MDS và công nghệ sử dụng.
- Tìm hiểu kiến trúc Data Lakehouse, so sánh với Data Lake, Data Warehouse.
- Áp dụng MDS vào việc xây dựng Data Platform dựa trên kiến trúc Lakehouse.
- Tạo ra các báo cáo, report từ dữ liệu thực nghiệm.

3. Phương pháp nghiên cứu

Tìm hiểu các tài liệu: Bắt đầu bằng việc tìm kiếm tài liệu, sách, báo, bài viết liên quan đến MDS, kiến trúc Data Lakehouse và cách chúng hoạt động, từ đó xây dựng nền tảng kiến thức vững chắc để xây dựng một ứng dụng Data Platform hoàn chỉnh.

Lựa chọn công nghệ và công cụ phù hợp: Dựa trên yêu cầu đã xác định, chọn các công nghệ và công cụ phù hợp để xây dựng ứng dụng Data Platform. Các công nghệ và công cụ này có thể bao gồm hệ thống lưu trữ dữ liệu, công cụ trích xuất dữ liệu, công cụ xử lý dữ liệu, công cụ phân tích dữ liệu và công cụ trực quan hóa dữ liệu. Đảm bảo tính tương thích và tính mở rộng của các công nghệ và công cụ được lựa chọn.

Thiết kế kiến trúc ứng dụng Data Platform: Dựa trên MDS và công nghệ được chọn, thiết kế kiến trúc tổng quan của ứng dụng Data Platform. Xác định các thành phần, tầng và giao tiếp giữa chúng trong kiến trúc tổng thể. Thiết kế phải đảm bảo tính linh hoạt, mở rộng và tương thích với các yêu cầu cụ thể của doanh nghiệp.

Triển khai và thử nghiệm ứng dụng Data Platform: Xây dựng và triển khai ứng dụng Data Platform dựa trên kiến trúc đã thiết kế. Thử nghiệm các chức năng và tính năng của ứng dụng, xác định sự hoạt động và hiệu suất của nó. Đảm bảo rằng ứng dụng đáp ứng được yêu cầu và mục tiêu đã đặt ra.

Đánh giá và tối ưu hóa: Đánh giá sự hoạt động, hiệu suất và hiệu quả của ứng dụng. Xác định các khía cạnh cần cải thiện và tối ưu hóa để tăng cường khả năng quản lý và sử dụng dữ liệu. Điều chỉnh và cải thiện ứng dụng dựa trên các kết quả đánh giá.

4. Kết quả dự kiến

Hiểu rõ về MDS và các công nghệ để xây dựng một Data Platform.

Xây dựng một ứng dụng Data Platform hoàn chỉnh để có thể đáp ứng nhu cầu của một doanh nghiệp cụ thể.

5. Bố cục luận văn

Bộ cục của luận văn bao gồm:

Phần 1: MỞ ĐẦU

Phần 2: NỘI DUNG. Phần này gồm 3 chương

- CHƯƠNG 1: TỔNG QUAN VỀ MODERN DATA STACK
- CHƯƠNG 2: KIẾN TRÚC DATA LAKEHOUSE

- CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG DATA PLATFORM

Phần 3: KẾT LUẬN

TÀI LIỆU THAM KHẢO

PHẦN 2: NỘI DUNG

CHƯƠNG 1: TỔNG QUAN VỀ MODERN DATA STACK

1.1. KHÁI NIỆM VỀ MODERN DATA STACK

Modern Data Stack là tập hợp các công cụ và công nghệ được sử dụng để thu thập, lưu trữ, xử lý và phân tích dữ liệu được hiểu theo cách có thể mở rộng, hiệu quả và tiết kiệm chi phí. Thuật ngữ “stack” trong điện toán có nghĩa là một nhóm công nghệ làm việc cùng nhau để đạt được mục tiêu chung. Các kỹ sư phần mềm sử dụng nhóm công nghệ - sự kết hợp của ngôn ngữ lập trình, framework, thư viện, v.v. để xây dựng các sản phẩm và dịch vụ cho nhiều mục đích khác nhau. Ngược lại, data stack tập trung vào dữ liệu, nó giúp doanh nghiệp quản lý dữ liệu và tập trung tận dụng tối đa dữ liệu đó.

MDS có thể giúp công ty quản lý và phân tích dữ liệu này một cách hiệu quả bằng cách sử dụng kho dữ liệu dựa trên đám mây như Snowflake, các công cụ tích hợp dữ liệu như Stitch và phần mềm trực quan hóa dữ liệu như Power BI. MDS cũng có thể hữu ích với khả năng xử lý mở rộng khi khối lượng dữ liệu tăng lên mà không cần xây dựng lại toàn bộ cơ sở hạ tầng. Sau khi hiểu rõ hơn về dữ liệu, công ty có thể đưa ra các đề xuất, cải thiện sự hài lòng của khách hàng và cuối cùng là thúc đẩy tăng trưởng kinh doanh.

1.2. THÀNH PHẦN CỦA MODERN DATA STACK

Để hiểu lợi ích của từng công cụ MDS cụ thể và đưa ra lựa chọn công cụ tốt, trước tiên cần hiểu các thành phần riêng lẻ của một Data Platform và khả năng chung của các công cụ phục vụ từng thành phần đó.

Đối với Traditional Data Platform, nó thường tuân theo cách tiếp cận có cấu trúc và phi tập trung. Chúng chủ yếu bao gồm các cơ sở dữ liệu (MySQL, Oracle Database, SQL Server) và kho dữ liệu. Các cấu trúc này thường được thiết kế để xử lý dữ liệu có cấu trúc và thường được đặt tại chỗ (on-premises).

Với Modern Data Platform, nó đã phát triển để khắc phục những hạn chế trên và hỗ trợ nhu cầu đa dạng của môi trường giàu dữ liệu ngày nay. Họ có khả năng xử lý khối lượng dữ liệu khổng lồ, xử lý dữ liệu trong thời gian thực và quản lý cả dữ liệu có cấu trúc và không cấu trúc. Hơn nữa, họ thường tận dụng các công nghệ đám mây để có khả năng mở rộng, tính linh hoạt và tiết kiệm chi phí.

Dưới đây là bảng so sánh giữa Traditional Data Platform và Modern Data Platform:

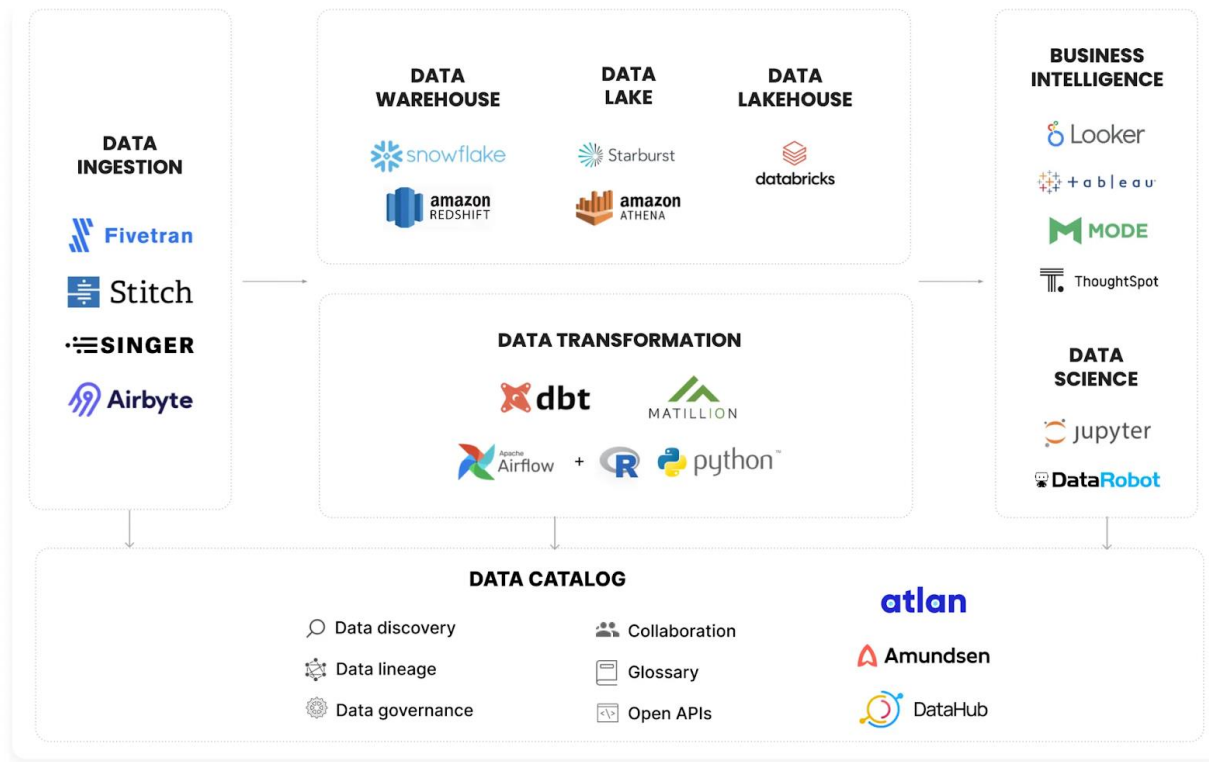
Bảng 1.2.1. Bảng so sánh giữa Traditional Data Platform và Modern Data Platform

Thuộc tính	Traditional Data Platform	Modern Data Platform
Kiểu dữ liệu	Chủ yếu dạng có cấu trúc	Có cấu trúc, bán cấu trúc và không cấu trúc
Khối lượng dữ liệu	Bị giới hạn	Xử lý khối lượng dữ liệu rất lớn (big data)
Xử lý dữ liệu	Xử lý hàng loạt (batch)	Xử lý hàng loạt (batch) và xử lý theo thời gian thực (real-time)
Lưu trữ dữ liệu	Cơ sở dữ liệu quan hệ, kho dữ liệu (Data Warehouse)	Kết hợp cơ sở dữ liệu quan hệ, NoSQL, Data Lake và Data Warehouse
Hạ tầng	On-premises	Dựa trên đám mây, khả năng mở rộng và linh hoạt
Phân tích dữ liệu	Hỗ trợ các công cụ phân tích và BI truyền thống	Hỗ trợ nhiều công cụ phân tích, bao gồm phân tích nâng cao và khả năng AI/ML

Tính linh hoạt	Dữ liệu phải phù hợp với các lược đồ được xác định trước	Lược đồ linh hoạt , đặc biệt là trong hồ dữ liệu và cơ sở dữ liệu NoSQL
Quản trị dữ liệu	Khả năng quản trị dữ liệu cơ bản	Các tính năng bảo mật và quản trị dữ liệu nâng cao, thường được tích hợp sẵn hoặc tích hợp

Các thành phần của một Modern Data Platform bao gồm:

- Data Collection and Tracking
- Data Ingestion
- Data Transformation
- Data Storage
- Metrics layer
- BI Tools
- Reserve ETL
- Orchestration
- Data Management, Quality, and Governance



Hình 1.2.1. Chi tiết các thành phần của một Data Platform

Chi tiết về các thành phần của Data Platform:

- **Data Collection and Tracking:**

Là quá trình thu thập và theo dõi thông tin và dữ liệu liên quan đến hành vi, hoạt động hoặc thuộc tính của người dùng, khách hàng từ các ứng dụng khách (mobile, web, thiết bị IoT) và dữ liệu giao dịch từ các dịch vụ backend.

Tuy nhiên, việc thu thập và theo dõi dữ liệu cần tuân thủ các quy định về bảo mật và quyền riêng tư, như GDPR (General Data Protection Regulation) tại Liên minh châu Âu hoặc CCPA (California Consumer Privacy Act) tại California, Hoa Kỳ. Do đó, các tổ chức cần chú ý đảm bảo rằng việc thu thập và xử lý dữ liệu được thực hiện theo quy định pháp luật và có sự đồng ý của người dùng.

- **Data Ingestion:**

Là quá trình trích xuất và tải dữ liệu nguyên bản từ nguồn dữ liệu chính đến một kho dữ liệu trung tâm (Data Warehouse) hoặc hồ dữ liệu (Data Lake).

Quá trình này bao gồm việc thu thập dữ liệu từ các nguồn khác nhau, chuyển đổi định dạng dữ liệu (nếu cần thiết), và lưu trữ dữ liệu trong một cấu trúc và định dạng chuẩn để sử dụng cho mục đích phân tích và xử lý dữ liệu sau này. Data Ingestion thường được

thực hiện thông qua các công cụ và quy trình tự động để đảm bảo tính nhất quán và hiệu quả trong việc thu thập và nhập dữ liệu.

Các công cụ MDS phổ biến phục vụ quá trình này bao gồm: Apache Kafka, Apache Nifi, Airbyte,...

- **Data Transformation:**

Để có giá trị, dữ liệu được lưu trữ trong kho dữ liệu phải trải qua quá trình chuyển đổi. Nó liên quan đến việc chuyển đổi dữ liệu thô thành các mô hình thân thiện với người dùng mà các nhà phân tích hoặc nhà khoa học dữ liệu có thể truy vấn để rút ra thông tin chi tiết, xây dựng trang tổng quan hoặc thậm chí là mô hình ML.

Các công cụ như dbt và Dataform thường được sử dụng để chuyển đổi và lập mô hình dữ liệu trong MDS. Tất cả các công cụ này cho phép chuyển đổi dữ liệu thô hoặc được xử lý nhẹ được lưu trữ trong kho dữ liệu thành các mô hình thân thiện với người dùng. Chúng đơn giản hóa việc chuyển đổi dữ liệu bằng cách giảm sự dư thừa, cho phép thiết kế mô hình dữ liệu nhất quán và thúc đẩy khả năng tái sử dụng và kiểm tra mã.

- **Data Storage:**

Data Lake ngày xưa là một lựa chọn ưa thích vì chúng có thể lưu trữ dữ liệu thô ở các định dạng khác nhau, như dữ liệu phi cấu trúc và bán cấu trúc, trong khi kho dữ liệu (Data Warehouse) được sử dụng để lưu trữ dữ liệu có cấu trúc. Nhưng sự khác biệt này đã bị xóa nhòa trong thời đại kho dữ liệu trên đám mây.

Kho dữ liệu đám mây đóng vai trò là thành phần trung tâm của MDS và chúng cung cấp khả năng phân tích mạnh mẽ cho phép truy vấn trực tiếp dữ liệu được lưu trữ trong đó. Một số giải pháp phổ biến bao gồm Snowflake, BigQuery của Google và Amazon Redshift. Ngoài ra, Databricks với Delta Lake, là một lựa chọn thường được sử dụng trong MDS.

Ngoài ra, Databricks đang phát triển khái niệm Lakehouse để kết hợp tính năng quản lý dữ liệu của kho dữ liệu với tính linh hoạt và chi phí thấp của hồ dữ liệu. Từ đó tạo ra một mô hình quản lý dữ liệu được cải tiến và hiệu quả hơn.

- **Metrics Layer (Headless BI):**

Metrics Layer nằm giữa mô hình dữ liệu và các công cụ BI. Nó cung cấp một giao diện lập trình ứng dụng (API) chuyển đổi yêu cầu tính toán số liệu thành các truy vấn SQL và thực thi chúng trên kho dữ liệu.

Metrics Layer giúp đạt được sự báo cáo nhất quán, đặc biệt là trong các tổ chức lớn nơi định nghĩa số liệu và logic tính toán thường khác nhau giữa các bộ phận khác nhau.

- **BI và DS tools:**

BI và DA liên quan đến việc sử dụng dữ liệu để hiểu rõ hơn, giúp đưa ra quyết định kinh doanh. Điều này bao gồm việc tạo báo cáo và trực quan hóa để giúp các bên liên quan hiểu được xu hướng và xác định các lĩnh vực cần cải thiện. Các công cụ thường được sử dụng trong lĩnh vực này bao gồm Tableau, Looker và Power BI.

DS liên quan đến việc sử dụng dữ liệu để xây dựng các mô hình dự đoán nhằm tự động hóa việc ra quyết định hoặc cung cấp thông tin cho các phân tích phức tạp hơn. Lĩnh vực này đòi hỏi sự hiểu biết sâu sắc về thống kê và thuật toán học máy, cũng như khả năng lập trình bằng các ngôn ngữ như Python hoặc R. Các công cụ phổ biến cho khoa học dữ liệu và học máy bao gồm Jupyter Notebooks, công cụ DataRobot autoML, Scikit-learn và TensorFlow.

- **Reverse ETL:**

Reverse ETL là quá trình di chuyển dữ liệu đã được chuyển đổi từ kho dữ liệu (Data Warehouse) đến các hệ thống phụ trợ như hoạt động, tài chính, marketing, quản lý quan hệ khách hàng (CRM), bán hàng và ngay cả trở lại sản phẩm để hỗ trợ quyết định hoạt động.

Các công cụ Reverse ETL tương tự như các công cụ nhập dữ liệu MDS, trừ việc hướng dữ liệu được đảo ngược (từ kho dữ liệu đến các hệ thống phụ trợ).

- **Orchestration (Workflow engine):**

Hệ thống quản lý (Orchestration systems) là những hệ thống cần thiết để chạy các đường ống dữ liệu theo lịch trình, yêu cầu/hoàn trả tài nguyên cơ sở hạ tầng theo yêu cầu, phản ứng với các lỗi và quản lý các phụ thuộc giữa các đường ống dữ liệu từ một giao diện chung.

Về phần này, Apache Airflow là một lựa chọn nền tảng nguồn mở phổ biến được sử dụng để điều phối dữ liệu trên toàn bộ đường dẫn dữ liệu. Nó cung cấp giao diện thân thiện với người dùng để lên lịch các công việc hàng loạt, xâu chuỗi các nhiệm vụ lại với nhau và theo dõi tiến trình của luồng công việc dữ liệu.

- **Data Management, Quality, and Governance:**

Quản lý, chất lượng và quản trị dữ liệu là thuật ngữ bao gồm quản trị chất lượng dữ liệu, nguồn gốc dữ liệu, khám phá dữ liệu, đăng ký, bảo mật thông tin và quyền riêng tư dữ liệu bằng cách hiệu quả thu thập và sử dụng siêu dữ liệu (metadata).

Các công cụ quản trị dữ liệu MDS tập trung vào việc tạo điều kiện cho mức độ minh bạch, cộng tác và dân chủ dữ liệu cao.

1.3. SO SÁNH MODERN DATA STACK VÀ TRADITIONAL DATA STACK

Traditional Data Stack là các giải pháp tại chỗ dựa trên cơ sở hạ tầng phần cứng và phần mềm do chính tổ chức quản lý. Thường dựa trên kiến trúc nguyên khối, Traditional Data Stack rất phức tạp và đòi hỏi phải đầu tư đáng kể vào cơ sở hạ tầng công nghệ thông tin và nhân sự. Ngoài ra, chúng có thể không dễ dàng tích hợp vào môi trường dựa trên đám mây, khiến chúng kém linh hoạt và kém khả năng mở rộng hơn so với MDS.

Dưới đây là một số sự khác biệt giữa Traditional Data Stack và MDS:

- **Khả năng mở rộng:**

Traditional Data Stack thường sử dụng cơ sở hạ tầng tại chỗ và các giải pháp phần mềm độc quyền, hạn chế khả năng mở rộng do hạn chế về phần cứng và kiến trúc cứng nhắc. Việc mở rộng quy mô đòi hỏi phải đầu tư đáng kể và thay đổi cơ sở hạ tầng.

Ngược lại, MDS tận dụng cơ sở hạ tầng dựa trên đám mây, phần mềm nguồn mở và các công nghệ có thể mở rộng khác để xử lý khối lượng dữ liệu lớn và tích hợp dễ dàng hơn với cơ sở hạ tầng hiện có.

- **Dữ liệu đa dạng:**

Traditional Data Stack: chủ yếu tập trung vào dữ liệu có cấu trúc, thường đến từ cơ sở dữ liệu quan hệ và hệ thống doanh nghiệp có cấu trúc. Ví dụ về dữ liệu có cấu trúc bao gồm thông tin khách hàng, giao dịch bán hàng và dữ liệu hàng tồn kho.

Ngược lại, MDS được xây dựng để xử lý nhiều loại dữ liệu khác nhau, bao gồm dữ liệu phi cấu trúc và bán cấu trúc như bài đăng trên mạng xã hội, dữ liệu cảm biến, nhật ký và nội dung đa phương tiện. Họ kết hợp data lakes và cơ sở dữ liệu NoSQL để chứa các định dạng dữ liệu đa dạng. Họ cũng sử dụng các khung điện toán phân tán, data lakes và cơ sở dữ liệu NoSQL để cung cấp giải pháp có thể mở rộng và tiết kiệm chi phí để lưu trữ các loại dữ liệu đa dạng.

- **Xử lý thời gian thực:**

Traditional Data Stack thường hoạt động theo mô hình xử lý hàng loạt, trong đó dữ liệu được xử lý theo khoảng thời gian định kỳ. Việc xử lý dữ liệu theo thời gian thực hoặc gần thời gian thực là một thách thức khó đạt được.

Mặt khác, MDS bao gồm khả năng xử lý dữ liệu theo luồng và thời gian thực. Họ tận dụng các công nghệ như Apache Kafka, Apache Flink hoặc Apache Spark Streaming để xử lý dữ liệu khi dữ liệu đến. Xử lý theo thời gian thực mang lại lợi ích đáng kể trong kho dữ liệu hiện đại, giúp doanh nghiệp đưa ra quyết định kịp thời, hiểu rõ hơn, nâng cao trải nghiệm của khách hàng và cho phép giám sát liên tục trên các ứng dụng dựa trên dữ liệu.

- **Chi phí và bảo trì:**

Traditional Data Stack yêu cầu đầu tư trả trước đáng kể vào phần cứng, giấy phép phần mềm và bảo trì. Các tổ chức chịu trách nhiệm quản lý cơ sở hạ tầng, đảm bảo tính sẵn sàng cao và thực hiện cập nhật và bảo trì thường xuyên.

Một trong những khác biệt lớn nhất và lợi thế có tác động lớn nhất của MDS là khả năng cung cấp cách tiếp cận linh hoạt và hiệu quả hơn về mặt chi phí. Các giải pháp dựa trên đám mây cung cấp mô hình định giá theo mức sử dụng, loại bỏ nhu cầu đầu tư lớn vào phần cứng. Nhà cung cấp đám mây đảm nhiệm việc quản lý, cập nhật và bảo trì cơ sở hạ tầng, cho phép các tổ chức tập trung vào phân tích dữ liệu và hiểu biết sâu sắc.

- **Tính linh hoạt và nhanh nhẹn:**

Traditional Data Stack thường cứng nhắc và ít thích ứng với nhu cầu kinh doanh thay đổi. Việc nâng cấp hoặc giới thiệu các công nghệ mới có thể tốn thời gian và phức tạp.

MDS ưu tiên tính linh hoạt và nhanh nhẹn. Chúng cho phép các tổ chức dễ dàng tích hợp các nguồn dữ liệu mới, thử nghiệm các công cụ và công nghệ khác nhau, đồng thời nhanh chóng mở rộng quy mô hoặc xoay vòng dựa trên các yêu cầu ngày càng phát triển. MDS thường sử dụng kiến trúc mô-đun cho phép tích hợp và hoán đổi các thành phần dễ dàng, do đó các công ty có thể áp dụng các giải pháp tốt nhất cho từng thành phần của ngăn xếp dữ liệu.

1.4. CÁC CÔNG NGHỆ SỬ DỤNG

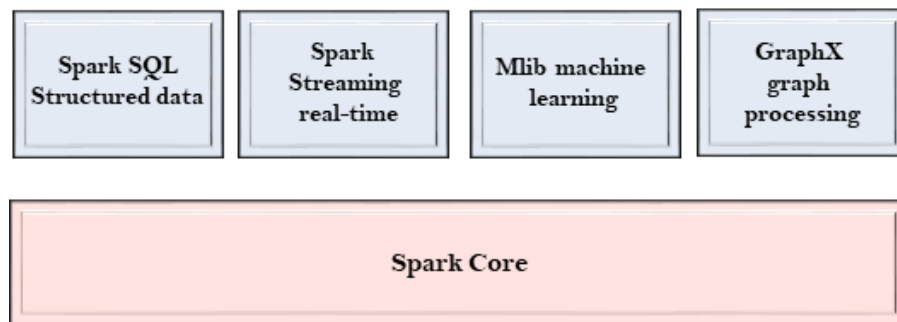
1.4.1. Apache Spark



Hình 1.4.1.1. Logo của Apache Spark

Apache Spark là một framework mã nguồn mở tính toán cụm, được phát triển sơ khởi vào năm 2009 bởi AMPLab. Tốc độ xử lý của Spark có được do việc tính toán được thực hiện cùng lúc trên nhiều máy khác nhau. Đồng thời việc tính toán được thực hiện ở bộ nhớ trong hay thực hiện hoàn toàn trên RAM.

Thành phần chính của Apache Spark:



Hình 1.4.1.2. Các thành phần chính của Apache Spark

Apache Spark gồm có 5 thành phần chính : Spark Core, Spark Streaming, Spark SQL, MLlib và GraphX, trong đó:

- **Spark Core:** Là nền tảng cho các thành phần còn lại, các thành phần khác muốn chạy thì phải thông qua Spark Core.
- **Spark SQL:** Để thực hiện các thao tác trên DataFrames bằng ngôn ngữ Scala, Java hoặc Python và nó cũng hỗ trợ cả ngôn ngữ SQL với giao diện command-line và ODBC/JDBC server.
- **Spark Streaming:** Được sử dụng để thực hiện việc phân tích stream.
- **MLlib:** Là một nền tảng học máy phân tán bên trên Spark do kiến trúc phân tán dựa trên bộ nhớ.
- **GraphX:** Là nền tảng xử lý đồ thị dựa trên Spark.

Ưu điểm:

- Tốc độ xử lý nhanh.
- Khả năng phân tích, xử lý dữ liệu thời gian thực với quy mô lớn.
- Tương thích cao với nhiều ngôn ngữ lập trình và hệ điều hành.

Nhược điểm:

- Yêu cầu tài nguyên phần cứng.
- Học và triển khai phức tạp với những người mới nghiên cứu.
- Không phù hợp cho các tác vụ nhỏ.

1.4.2. Polars



Hình 1.4.2.1. Logo của thư viện Polars

Polars là thư viện DataFrame được viết hoàn toàn bằng Rust và được xây dựng để trao quyền cho các nhà phát triển Python với framework hiệu quả và có thể mở rộng để xử lý dữ liệu và được coi là giải pháp thay thế cho thư viện Pandas rất phổ biến. Nó cung cấp một loạt các chức năng hỗ trợ các tác vụ phân tích và thao tác dữ liệu khác nhau.

Một số tính năng và lợi thế chính của việc sử dụng Polars bao gồm:

- Hiệu suất cao.
- Xử lý dữ liệu đa luồng.
- Cú pháp dễ sử dụng.
- Hỗ trợ dữ liệu đa dạng.

1.4.3. Minio



Hình 1.4.3.1. Logo Minio

MinIO là một hệ thống lưu trữ đối tượng mã nguồn mở và dựa trên đám mây. Nó được thiết kế để cung cấp dịch vụ lưu trữ đám mây hiệu suất cao và có khả năng mở rộng. MinIO triển khai giao thức lưu trữ đối tượng phổ biến như Amazon S3, cho phép các ứng dụng và dịch vụ web lưu trữ và truy xuất dữ liệu theo cách linh hoạt và hiệu quả.

Cấu trúc của MinIO:

- **Buckets:** MinIO tổ chức dữ liệu vào các đối tượng gọi là "buckets". Mỗi bucket là một không gian lưu trữ độc lập trong MinIO, tương tự như các thùng trong hệ thống lưu trữ đám mây khác như Amazon S3. Ta có thể tạo, xóa và quản lý các bucket trong MinIO.
- **Objects:** Mỗi bucket chứa một tập hợp các đối tượng. Đối tượng là các file hoặc dữ liệu được lưu trữ trong MinIO. Mỗi đối tượng có một tên duy nhất và được xác định bởi đường dẫn của nó trong bucket.
- **Object Metadata:** Mỗi đối tượng trong MinIO có một bộ siêu dữ liệu (metadata) liên quan đến nó. Siêu dữ liệu đối tượng chứa thông tin như tên file, kích thước, ngày tạo và loại nội dung. Điều này giúp quản lý và tìm kiếm dữ liệu trong MinIO.
- **Access Keys:** MinIO sử dụng cặp khóa truy cập (access key) để xác thực và phân quyền người dùng. Mỗi khóa truy cập gồm một access key ID và một secret key. Access key ID là một chuỗi định danh duy nhất và secret key là một chuỗi bí mật được sử dụng để xác thực yêu cầu truy cập vào MinIO.
- **Endpoints:** MinIO sử dụng các điểm cuối (endpoints) để định vị và truy cập vào các phiên bản MinIO Server. Mỗi phiên bản MinIO Server có một endpoint duy nhất, được xác định bằng URL hoặc địa chỉ IP.

- **Policies:** MinIO hỗ trợ chính sách để kiểm soát truy cập vào dữ liệu. Bằng cách sử dụng các chính sách, ta có thể xác định quyền truy cập và phân quyền cho các người dùng và nhóm người dùng trong MinIO.

1.4.4. Delta Lake



Hình 1.4.4.1. Logo Delta Lake

Delta Lake là một hệ thống quản lý Data Lake mã nguồn mở cung cấp các giao dịch ACID, lập phiên bản dữ liệu và khả năng phát triển lược đồ trên các khung dữ liệu lớn hiện có. Delta Lake được phát triển bởi Databricks, người tạo ra Apache Spark và hiện nó là một dự án nguồn mở thuộc Linux Foundation's Delta Lake Project.

Delta Lake cho phép các tổ chức quản lý và xử lý hiệu quả khối lượng dữ liệu lớn trong môi trường phân tán đồng thời cung cấp tính nhất quán trong giao dịch và tính toàn vẹn dữ liệu. Delta Lake được xây dựng dựa trên khung Apache Spark, có nghĩa là nó có thể được sử dụng với các công cụ và quy trình làm việc dữ liệu lớn hiện có, đồng thời có thể tích hợp liền mạch với các ứng dụng dựa trên Spark khác. Một số tính năng và lợi ích chính của Delta Lake bao gồm:

- **ACID Transactions:** Delta Lake hỗ trợ các giao dịch ACID để đảm bảo tính toàn vẹn và đáng tin cậy của dữ liệu trong quá trình xử lý. Đây là các tính năng quan trọng của hệ thống quản lý cơ sở dữ liệu (DBMS) để đảm bảo tính toàn vẹn và đáng tin cậy của dữ liệu trong quá trình thực hiện các giao dịch.
- **Schema Evolution:** Delta Lake bảo vệ một schema nghiêm ngặt cho dữ liệu, đảm bảo tính nhất quán và đáng tin cậy của cấu trúc dữ liệu. Điều này đảm bảo rằng bất kỳ thay đổi nào đối với schema (cấu trúc dữ liệu) đều phải có cơ chế rõ ràng và quản lý một cách kiểm soát.

- **Data Versioning:** Delta Lake cung cấp hỗ trợ lập phiên bản dữ liệu, cho phép người dùng theo dõi các thay đổi của dữ liệu theo thời gian. Tính năng này cho phép người dùng truy xuất các phiên bản dữ liệu trước đó, so sánh các phiên bản dữ liệu khác nhau và kiểm tra các thay đổi của dữ liệu.
- **Tối ưu hóa hiệu suất:** Delta Lake cung cấp nhiều tính năng và cơ chế tối ưu hóa để cải thiện hiệu suất và hiệu quả trong việc xử lý dữ liệu.
- **Tích hợp batch và stream:** Delta Lake hỗ trợ cả xử lý dữ liệu batch và dữ liệu luồng, cung cấp một giải pháp thống nhất cho các khối công việc thời gian thực và batch.

1.4.5. Hive Metastore



Hình 1.4.5.1. Logo Hive Metastore

Hive Metastore là một phần của hệ thống Hive, một công cụ phân tích dữ liệu mã nguồn mở dựa trên Apache Hadoop. Hive Metastore là một metadata repository cho Hive, nơi lưu trữ thông tin về các bảng, cột, vị trí lưu trữ và các thông tin liên quan khác về các bảng và cơ sở dữ liệu được tạo bằng Hive.

Khi ta tạo một bảng trong Hive, thông tin về cấu trúc bảng, kiểu dữ liệu và vị trí lưu trữ được lưu trữ trong Hive Metastore. Hive Metastore cung cấp một giao diện để truy xuất và quản lý metadata này. Nó lưu trữ thông tin về các bảng, phân vùng, chỉ mục và các thuộc tính khác của cơ sở dữ liệu Hive.

1.4.6. Metabase



Hình 1.4.6.1. Logo Metabase

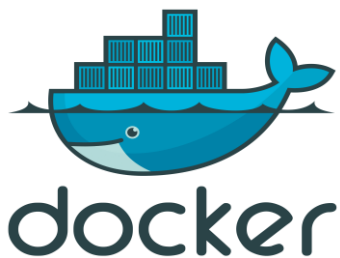
Metabase là công cụ mã nguồn mở hỗ trợ phân tích dữ liệu và chia sẻ chúng một cách dễ dàng. Metabase cho phép chúng ta đặt câu hỏi về dữ liệu và hiển thị câu trả lời dưới nhiều dạng biểu đồ khác nhau, thuận tiện cho việc đọc và xử lý thông tin.

Metabase có giao diện thân thiện, dễ sử dụng với cả những người dùng không có nhiều kiến thức về SQL, và có cả những tính năng cho những user có kiến thức tốt về SQL.

Các tính năng nổi bật của Metabase:

- Hỗ trợ nhiều datasource khác nhau như MySQL, Postgres, Mongo, SQL Server, AWS Redshift, Google BigQuery, Druid, H2, SQLite, Oracle, Crate, Google Analytics, Vertica, Spark, Presto, Snowflake.
- Hỗ trợ xuất dữ liệu đa dạng các biểu đồ: Number, Smart number, Progress bar, Gauge, Table, Line chart, Bar chart, Line + bar chart, Row chart, Area chart, Scatter Plot or bubble chart, Pie/donut chart, Funnel, Map.
- Theo dõi dữ liệu thời gian thực.
- Hỗ trợ tạo query bằng giao diện, kéo thả.
- Cung cấp quyền kiểm soát truy cập rất chi tiết và đầy đủ, bao gồm LGPD (Luật bảo vệ dữ liệu chung) và GDPR (Quy định chung về bảo vệ dữ liệu).
- Vì là công cụ mã nguồn mở nên miễn phí và dễ cài đặt.

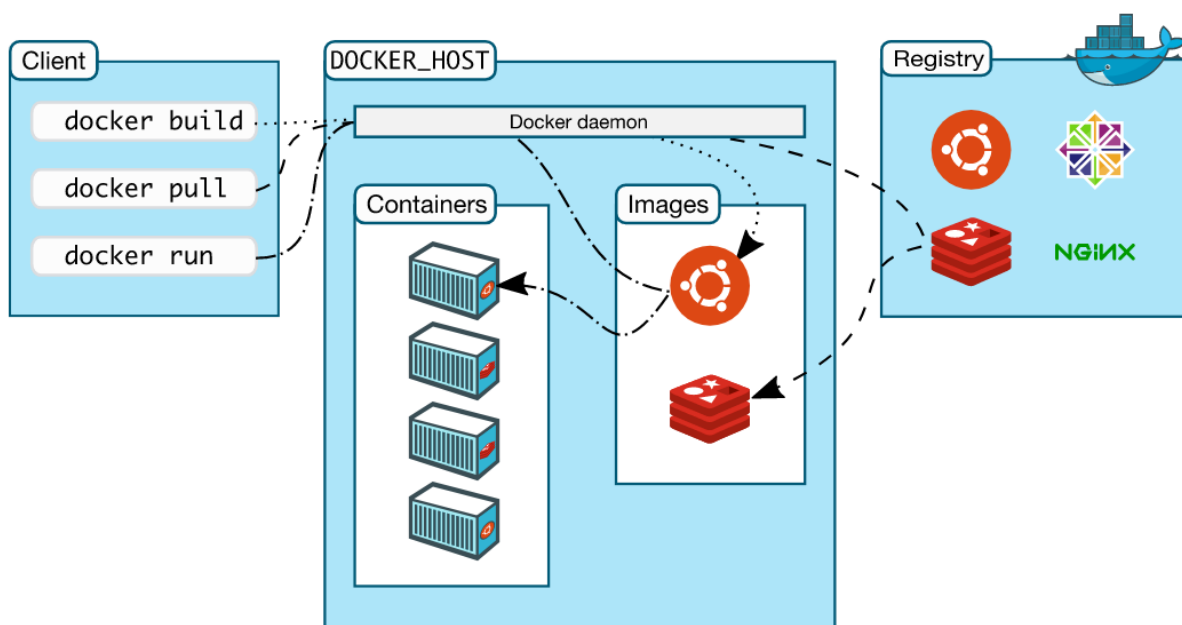
1.4.7. Docker



Hình 1.4.7.1. Logo Docker

Docker là nền tảng phần mềm cho phép chúng ta dựng, kiểm thử và triển khai ứng dụng một cách nhanh chóng. Docker đóng gói phần mềm vào các đơn vị tiêu chuẩn hóa được gọi là container có mọi thứ mà phần mềm cần để chạy, trong đó có thư viện, công cụ hệ thống, mã code và thời gian chạy. Bằng cách sử dụng Docker, ta có thể nhanh chóng triển khai và thay đổi quy mô ứng dụng vào bất kỳ môi trường nào và biết chắc rằng mã code của ta sẽ chạy được.

Kiến trúc của Docker: Docker sử dụng kiến trúc client – server. Docker client sử dụng một REST để có thể giao tiếp với Docker daemon, tiến trình này thực hiện công việc tạo, chạy và phân phối các Docker container. Docker client và daemon có thể chạy trên cùng một hệ thống hoặc có thể kết nối Docker client với Docker daemon từ xa. Kiến trúc của Docker sẽ bao gồm:



Hình 1.4.7.2. Kiến trúc của Docker

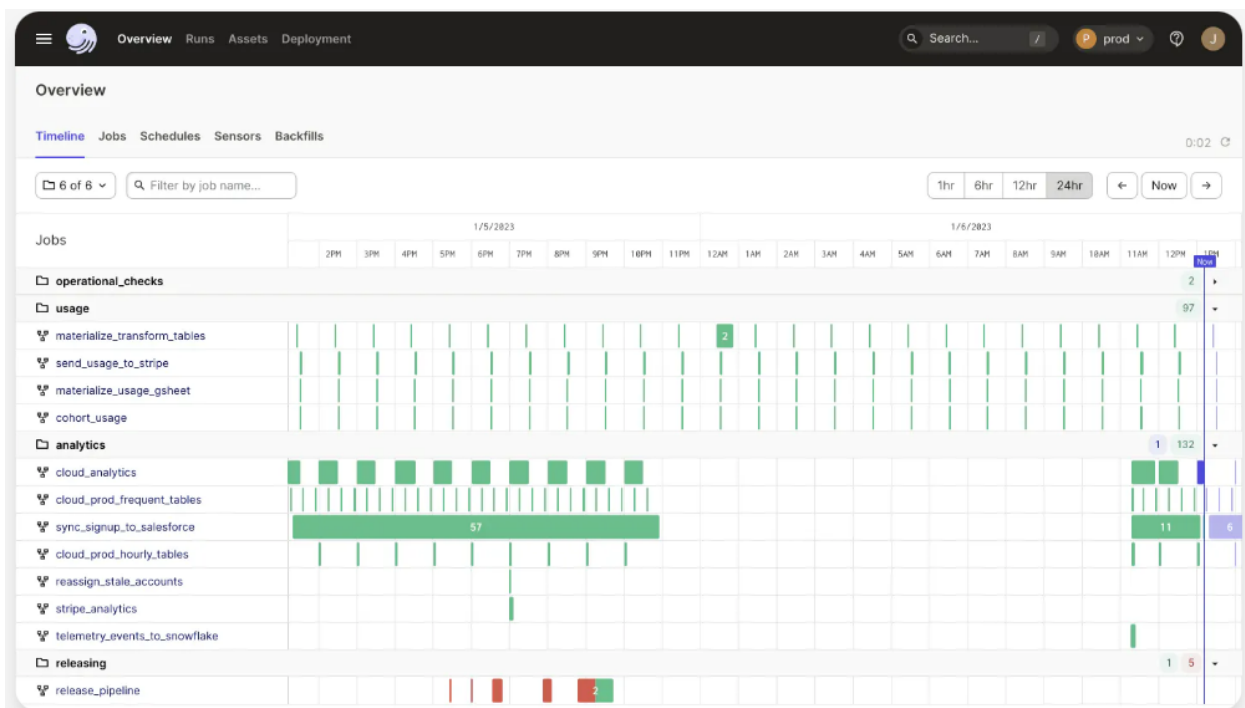
- **Docker daemon:** Lắng nghe các yêu cầu của người dùng thông qua Docker API và quản lý các đối tượng Docker như image, container, network và volume.
- **Docker client:** Là công cụ giúp người dùng giao tiếp với Docker host thông qua command.
- **Docker registry:** Là nơi lưu trữ các Docker image. Docker Hub là nơi lưu trữ Docker image công khai (public registry) mà bất kỳ ai cũng có thể sử dụng và Docker được định cấu hình mặc định để tìm image trên Docker Hub. Ngoài ra người dùng có thể cấu hình các registry riêng tư khác để lưu trữ Docker image.
- **Docker image:** Image sẽ được sử dụng để đóng gói các ứng dụng và các thành phần đi kèm của ứng dụng, được lưu trữ ở server hoặc trên registry
- **Docker container:** Container là 1 “runable instance” của image. Bạn có thể khởi tạo, dừng hoặc xóa container bằng cách sử dụng Docker API hoặc CLI.
- **Docker volume:** Volume được thiết kế để làm nơi lưu trữ các dữ liệu độc lập với vòng đời của container.
- **Docker networking:** cho phép kết nối các container lại với nhau. Kết nối này có thể trên 1 host hoặc nhiều host.

1.4.8. Dagster



Hình 1.4.8.1. Logo Dagster

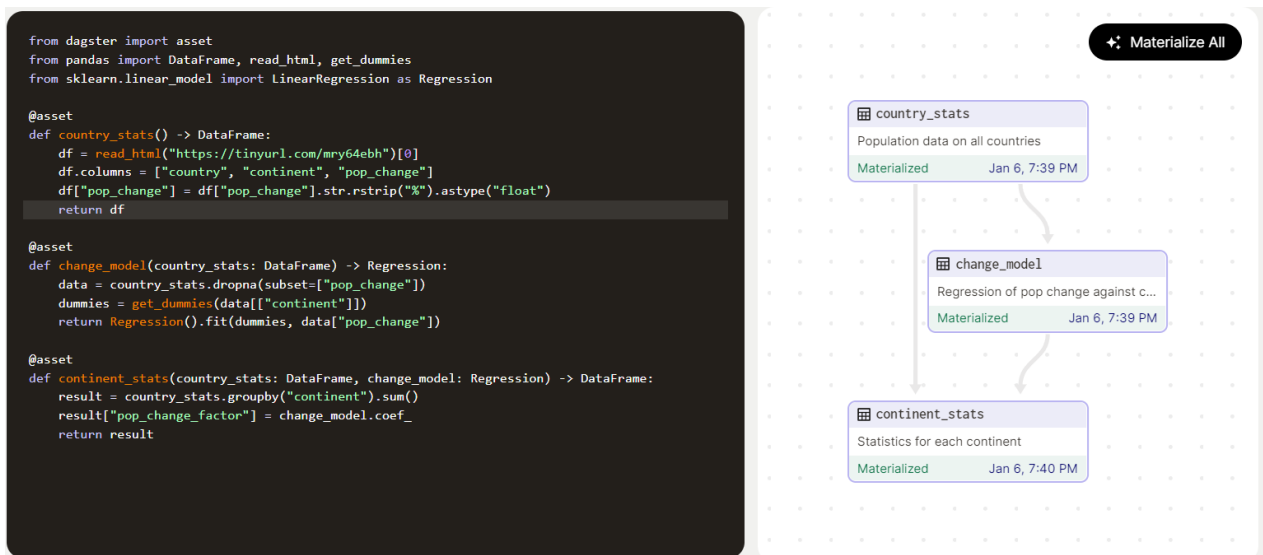
Dagster là một công cụ mã nguồn mở hỗ trợ Orchestrate Task (quản lý, tổ chức, điều phối và kiểm soát các tác vụ và công việc). Các công cụ trên hỗ trợ xây dựng Data Pipeline.



Hình 1.4.8.2. Data Pipeline cơ bản của Dagster

Dagster được thiết kế để phát triển và duy trì các data assets (ví dụ như Dataframe tables, các data sets, ML models, ...). Các data assets được sử dụng thông qua các function. Ta khai báo các function và các data assets mà các function đó tạo ra hoặc cập nhật. Sau đó sử dụng Dagster để chạy các function đúng thời điểm và đảm bảo các assets luôn được cập nhật.

Dagster được sử dụng trong hầu như mọi giai đoạn của vòng đời data development như local development, unit tests, integration tests, staging environments cho đến giai đoạn đưa lên production



Hình 1.4.8.3. Màn hình chạy trên tất cả công việc ở một nơi với chế độ xem dòng thời gian

CHƯƠNG 2: KIẾN TRÚC DATA LAKEHOUSE

2.1. TỔNG QUAN VỀ KIẾN TRÚC DATA LAKEHOUSE

2.1.1. Sự ra đời và khái niệm về Data Lakehouse

Data Lakehouse là một thuật ngữ tương đối mới trong kiến trúc dữ liệu lớn và đã phát triển nhanh chóng trong những năm gần đây. Nó kết hợp những ưu điểm tốt nhất của cả hai thế giới: khả năng mở rộng và tính linh hoạt của Data Lake cũng như độ tin cậy và hiệu suất của Data Warehouse.

Data Lake, được giới thiệu lần đầu tiên vào đầu những năm 2010, cung cấp một kho lưu trữ tập trung để lưu trữ một lượng lớn dữ liệu thô, phi cấu trúc.

Mặt khác, Data Warehouse đã tồn tại lâu hơn và được thiết kế để lưu trữ dữ liệu có cấu trúc nhằm truy vấn và phân tích nhanh chóng và hiệu quả. Tuy nhiên, kho dữ liệu có thể tồn kém và phức tạp để thiết lập và chúng thường yêu cầu chuyển đổi và làm sạch dữ liệu rộng rãi trước khi dữ liệu có thể được tải và phân tích.

Modern Data Platform đã đi một chặng đường dài trong việc cố gắng tạo ra kiến trúc dữ liệu khả thi. Ban đầu, nó bắt đầu bằng việc tạo Data Lake và sau đó trích xuất Data Warehouse để báo cáo, nghĩa là có hai kho lưu trữ dữ liệu. Các tổ chức sớm nhận ra rằng kiến trúc này không khả thi, tồn kém khi duy trì hai kho lưu trữ và không phù hợp với nhu cầu xử lý dữ liệu hiện đại. Hơn nữa, nó mang đến gánh nặng thực hiện quản lý và quản trị dữ liệu trên hai kho lưu trữ riêng biệt. Điều này làm tăng thêm chi phí, độ phức tạp.

Data Lakehouse được tạo ra để giải quyết những thách thức này và cung cấp giải pháp tiết kiệm chi phí hơn và có khả năng mở rộng hơn để quản lý dữ liệu lớn.

Với lượng dữ liệu do các doanh nghiệp tạo ra ngày càng tăng và nhu cầu xử lý dữ liệu nhanh chóng và hiệu quả, nhu cầu về Data Lakehouse đã tăng lên đáng kể. Do đó, nhiều công ty đã áp dụng phương pháp mới này, phương pháp này đã phát triển thành kho lưu trữ trung tâm cho tất cả các loại dữ liệu trong một tổ chức.



Hình 2.1.1.1. Kiến trúc 2 tầng hiện tại

Data Warehouse:

Kho dữ liệu đã trở thành công nghệ tiên tiến cho khối lượng công việc phân tích kể từ những năm 1980 với sự xuất hiện của mô hình chiều của Ralph Kimball. Chúng được xây dựng để tổng hợp dữ liệu có cấu trúc từ các nguồn khác nhau và chạy các truy vấn phân tích trên chúng để tạo ra thông tin chi tiết.

Cho đến khi xuất hiện đám mây Data Warehouse như Aws Redshift, hầu hết các kho dữ liệu đều có bộ nhớ và điện toán được kết hợp chặt chẽ với nhau. Điều này làm cho việc mở rộng quy mô không linh hoạt vì bạn không thể mở rộng quy mô lưu trữ một cách độc lập từ máy tính và ngược lại.

Tuy nhiên, nhìn chung, Kho dữ liệu khá đắt tiền và không hỗ trợ dữ liệu phi cấu trúc. Hơn nữa, chúng không phù hợp với khối lượng công việc phân tích nâng cao. Kho dữ liệu cũng là một hệ thống bị khóa duy nhất, có nghĩa là dữ liệu của ta chỉ có thể được truy cập bằng công cụ tính toán của kho.

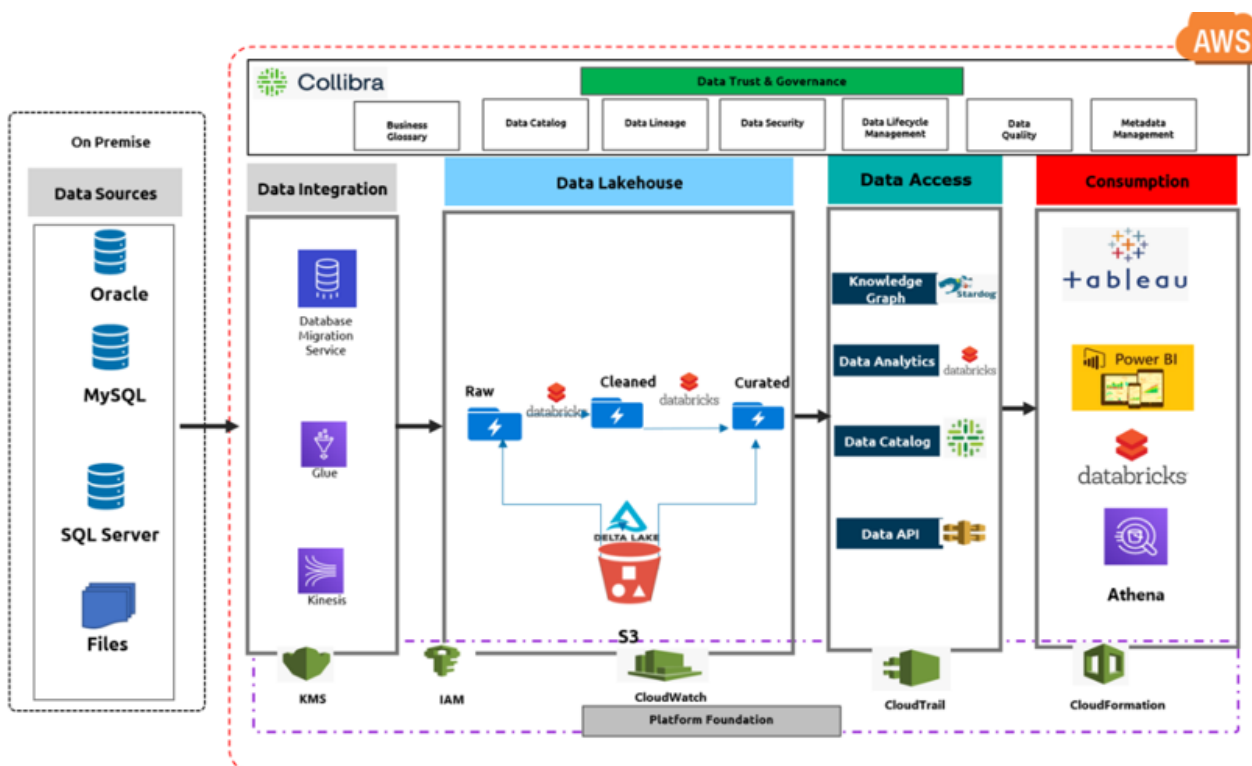
Data Lake:

Những nhược điểm này đã được Data Lake giải quyết. Họ cung cấp dung lượng lưu trữ rẻ và tính linh hoạt với định dạng dữ liệu và loại tệp. Ngoài ra, chúng có thể được truy cập bởi nhiều công cụ tính toán khác nhau.

Các bộ dữ liệu riêng lẻ trong Data Lake thường được tổ chức dưới dạng tập hợp các

tệp trong cấu trúc thư mục, thường có nhiều tệp trong một thư mục biểu thị một bảng duy nhất. Lợi ích của phương pháp này là dữ liệu rất dễ tiếp cận và linh hoạt.

Giải pháp mới Data Lakehouse:



Hình 2.1.1.2. Kiến trúc Data Lakehouse bởi Databricks

Data Lakehouse vẫn sử dụng bộ lưu trữ linh hoạt và ít chi phí của Data Lake, nhưng họ bổ sung thêm một lớp cấu trúc và quản trị vào dữ liệu. Họ sử dụng các định dạng bảng mở như Apache Iceberg để cung cấp khả năng thực thi lược đồ, tổ chức tệp và các tính năng giống cơ sở dữ liệu truyền thống khác.

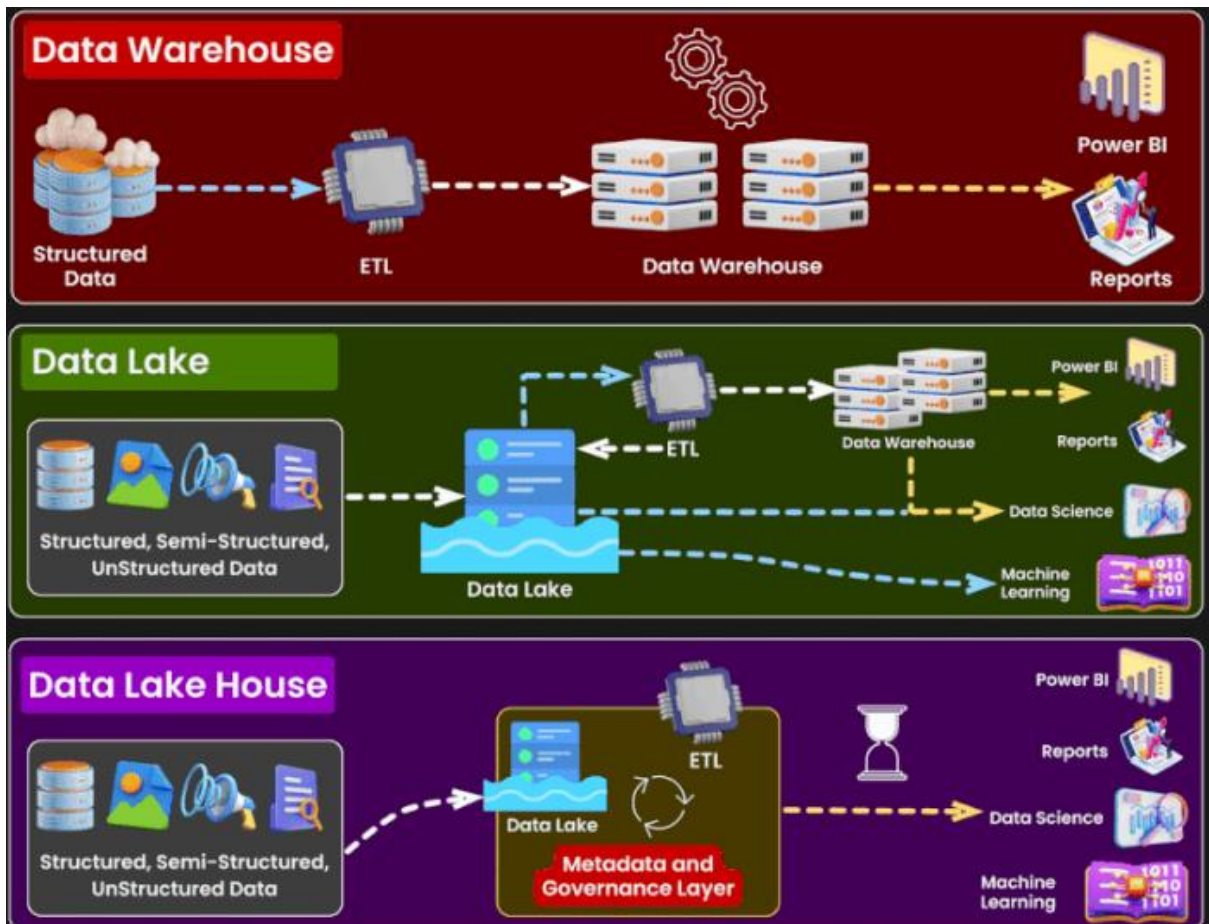
Data Lakehouse cung cấp giải pháp toàn diện hơn cho các tổ chức dựa trên dữ liệu hiện đại. Nó cân bằng khả năng mở rộng và tính linh hoạt của Data Lake với hiệu suất và cấu trúc của Data Warehouse.

Điều này cho phép Data Lakehouse sử dụng các công cụ và kỹ thuật dựa trên SQL truyền thống để truy vấn và phân tích dữ liệu, giúp cả nhà phân tích kinh doanh và nhà khoa học dữ liệu làm việc với dữ liệu và tìm hiểu thông tin chi tiết dễ dàng hơn.

Bằng cách kết hợp các khía cạnh tốt nhất của Data Lake và Data Warehouse, Data Lakehouse cung cấp cách lưu trữ và quản lý dữ liệu vừa linh hoạt vừa có cấu trúc. Điều này giúp các nhóm cộng tác, tìm hiểu thông tin chi tiết và đưa ra quyết định sáng suốt dễ dàng.

hơn.. Độ trễ truy vấn thấp và độ tin cậy cao cho BI và phân tích nâng cao.

2.1.2. So sánh Data LakeHouse với Data Lake và Data Warehouse



Hình 2.1.2.1. Kiến trúc Data Warehouse, Data Lake và Data Lakehouse

Data Lakehouse kết hợp các ưu điểm của Data Lake và Data Warehouse. Data Lakehouse giúp khắc phục một số hạn chế của Data Lake như việc thiếu metadata, khó khăn trong việc truy vấn và xử lý dữ liệu, cũng như của Data Warehouse hạn chế về tính mở rộng và linh hoạt. Nó cung cấp một kiến trúc linh hoạt cho việc lưu trữ và xử lý dữ liệu phức tạp, đồng thời hỗ trợ các khía cạnh quản lý dữ liệu như quản lý phiên bản, metadata và tính toàn vẹn dữ liệu.

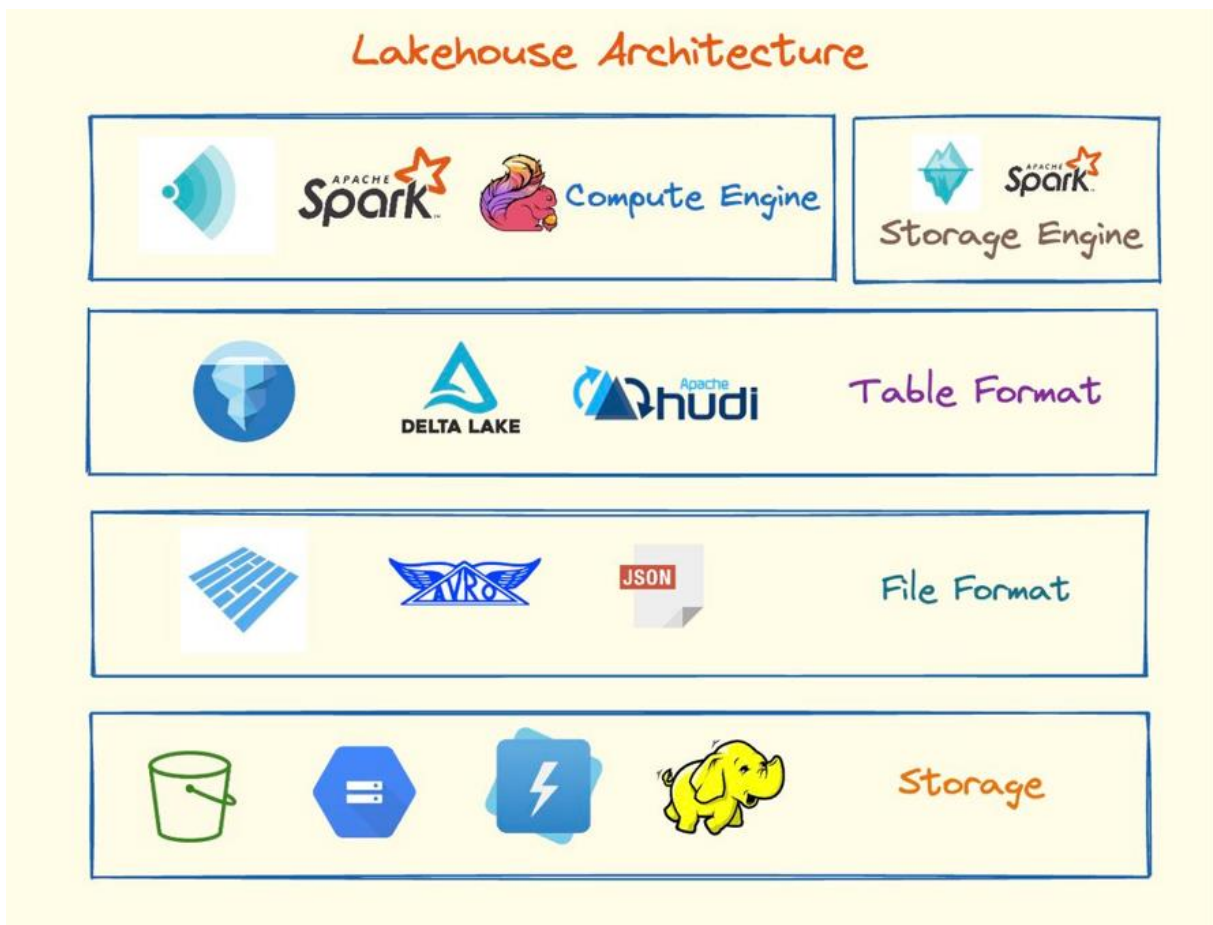
Dưới đây là bảng so sánh giữa Data Lake, Data Warehouse và Data Lakehouse:

Bảng 2.1.2.1. Bảng so sánh Data Lake, Data Warehouse và Data Lakehouse

	Data Lake	Data Warehouse	Data Lakehouse
Mục đích của dữ liệu	ML và AI	Analytic và BI	Có thể sử dụng cho ML/AI và nhu cầu Analytics/BI
Loại dữ liệu	Không có cấu trúc	Có cấu trúc	Không có cấu trúc và có cấu trúc
Người dùng	Các nhà khoa học và kỹ sư dữ liệu	Chuyên gia về kinh doanh	Chuyên gia về kinh doanh và kỹ sư dữ liệu
Chất lượng dữ liệu	Dữ liệu thô, chất lượng thấp và không đáng tin cậy	Dữ liệu được tuyển chọn, đáng tin cậy	Dữ liệu thô và được tuyển chọn, chất lượng cao với khả năng quản lý dữ liệu được tích hợp sẵn
ACID	Không tuân thủ ACID: cập nhật và xóa rất phức tạp	Tuân thủ ACID: đảm bảo mức toàn vẹn cao nhất	Tuân thủ ACID để đảm bảo tính nhất quán khi nhiều nguồn đọc/ghi dữ liệu đồng thời
Lưu trữ	Tiết kiệm chi phí, nhanh chóng và linh hoạt	Tốn kém và mất thời gian	Tiết kiệm chi phí, nhanh chóng và linh hoạt

2.2. CÁC THÀNH PHẦN CỦA DATA LAKEHOUSE

Data Lakehouse yêu cầu một bộ thành phần mạnh mẽ để quản lý và phân tích dữ liệu một cách hiệu quả. Việc triển khai hiệu quả các thành phần này đảm bảo dữ liệu được an toàn, tuân thủ các quy định và có thể truy cập được đối với người dùng được ủy quyền.



Hình 2.2.1. Kiến trúc Data Lakehouse

Các thành phần chính của một Data Lakehouse:

- **Computer Engine:**

Là một thành phần phần mềm cho phép người dùng tương tác với dữ liệu được lưu trữ trong Data Lakehouse. Công cụ truy vấn cho phép người dùng viết truy vấn SQL hoặc sử dụng các ngôn ngữ lập trình khác để truy cập, thao tác và phân tích dữ liệu. Một công cụ truy vấn tốt là điều cần thiết để đảm bảo rằng người dùng có thể tương tác với dữ liệu một cách hiệu quả.

- **File Format:**

Định dạng file parquet là một định dạng file cột với các tối ưu hóa giúp tăng tốc độ truy vấn. Nó là một định dạng file hiệu quả hơn so với CSV hay JSON. Parquet có thể lưu trữ dữ liệu theo các cột liên kế nhau, thay vì các hàng liên kế nhau. Điều này giúp giảm dung lượng lưu trữ, tăng khả năng nén và lọc dữ liệu, và hỗ trợ các kiểu dữ liệu phức tạp. Data Lakehouse thường sử dụng định dạng file parquet làm định dạng lưu trữ vì nó cho

phép lưu trữ nhiều loại dữ liệu khác nhau dưới dạng các đối tượng trong các kho lưu trữ đối tượng rẻ tiền, chẳng hạn như AWS S3. Parquet cũng tương thích với nhiều công cụ AI có thể sử dụng DataFrame để truy xuất dữ liệu lưu trữ đối tượng thô. Vì vậy, parquet là một lựa chọn phù hợp cho Data Lakehouse, một kiến trúc lưu trữ dữ liệu mới kết hợp khả năng thích ứng của các hồ dữ liệu với việc quản lý dữ liệu của các kho dữ liệu.

Parquet là một định dạng lưu trữ cột, nghĩa là nó lưu trữ dữ liệu theo từng cột riêng biệt, chứ không phải theo từng hàng như các định dạng khác. Điều này có nhiều lợi ích, chẳng hạn như:

Giảm dung lượng lưu trữ: Do dữ liệu cùng kiểu dữ liệu và cùng ý nghĩa được lưu trữ gần nhau, nên có thể áp dụng các phương pháp nén hiệu quả hơn.

Tăng tốc độ truy vấn: Khi truy vấn dữ liệu, chỉ cần đọc những cột cần thiết, không cần phải đọc toàn bộ hàng. Điều này giúp giảm thời gian I/O và băng thông mạng.

Hỗ trợ kiểu dữ liệu phức tạp: Parquet có thể lưu trữ các kiểu dữ liệu phức tạp như mảng, cấu trúc, map, v.v. mà không cần phải chuyển đổi sang các định dạng đơn giản hơn.

- **Table Format:**

Một trong những quyết định quan trọng nhất khi tạo Data Lakehouse là định dạng bảng mà ta sử dụng làm nền tảng cho nó, định dạng bảng là một thành phần quan trọng trong kiến trúc Data Lakehouse, nó quyết định việc bảng được lưu trữ như thế nào. Nhờ có Table Format mà dữ liệu được lưu trữ dưới dạng file trên Data Lake được ánh xạ thành các bảng như trong cơ sở dữ liệu, cho chúng ta khả năng tương tác với các định dạng lưu trữ trên Data Lake như là trên database. Đến thời điểm hiện tại, chỉ có ba công cụ hỗ trợ định dạng lưu trữ cho Data Lakehouse là Delta Lake, Apache Iceberg, Apache Hudi, tất cả đều cho phép tận dụng Cloud object storage để lưu trữ dữ liệu. Cung cấp ACID đầy đủ như một Data Warehouse thực thụ, bên cạnh đó còn hỗ trợ lưu trữ dữ liệu bán cấu trúc và không cấu trúc.

- **Storage Layer:**

Trong kiến trúc Data Lakehouse, nơi lưu trữ dữ liệu và công cụ tính toán được tách biệt với nhau, vậy nên để đảm bảo tính sẵn sàng cao của dữ liệu, người ta thường sử dụng các Cloud Object Storage làm nơi lưu trữ cho Lakehouse.

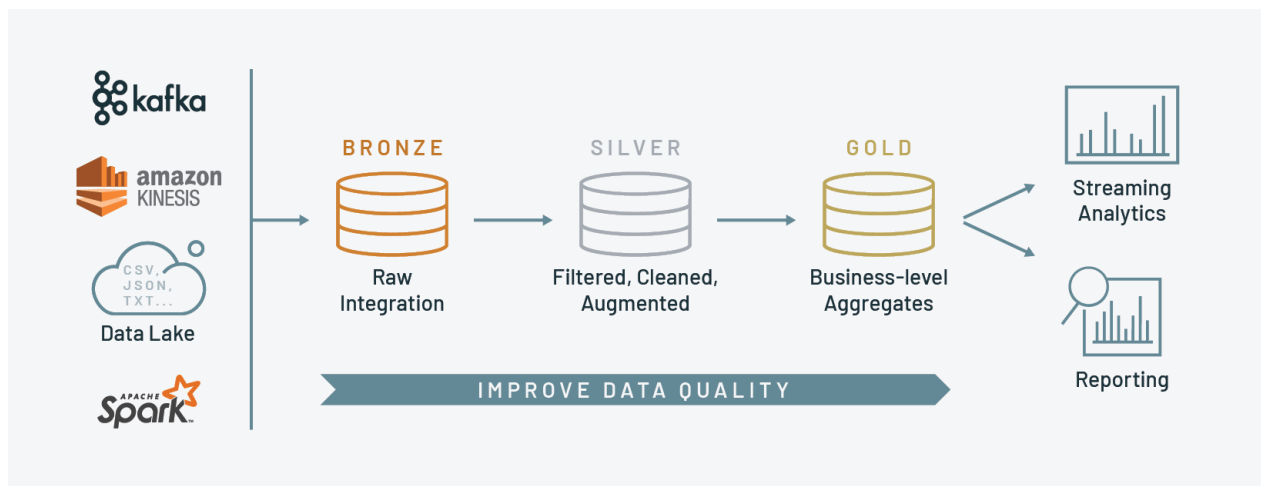
Hiện tại, các nhà cung cấp điện toán đám mây lớn có thể kể tới là Amazon Web Services, Google Cloud Platform, Microsoft Azure. Mỗi nhà cung cấp này đều có các Cloud Object Storage của riêng họ, vậy nên chúng ta có thể tận dụng các Cloud Object Storage này để làm nơi lưu trữ dữ liệu cho Data Lakehouse.

- **Metadata:**

Lớp siêu dữ liệu, lưu trữ siêu dữ liệu hoặc tất cả thông tin của các đối tượng dữ liệu trong lớp lưu trữ dữ liệu (Data Lake). Nó cũng có các tính năng quản lý dữ liệu như giao dịch ACID, bộ nhớ đệm, lập chỉ mục, lập phiên bản dữ liệu và sao chép. Lớp siêu dữ liệu có thể được xem như một danh mục siêu dữ liệu thống nhất. Lớp này cho phép các chức năng quản lý dữ liệu, kiểm toán và quản lý lược đồ.

2.3. KIẾN TRÚC MEDALLION ARCHITECTURE

Medallion architecture là một mẫu thiết kế dữ liệu được sử dụng để tổ chức dữ liệu một cách hợp lý trong data Lakehouse, với mục tiêu cải thiện dần dần cấu trúc và chất lượng của dữ liệu khi nó chảy qua từng lớp của kiến trúc (Bronze – Silver - Gold). Kiến trúc medallion đôi khi còn được gọi là kiến trúc "đa bước nhảy".



Hình 2.3.1. Kiến trúc Medallion Architecture

Chi tiết cấu trúc của từng lớp:

- **Bronze layer (dữ liệu thô):**

Lớp Bronze là nơi tiếp nhận tất cả dữ liệu từ các hệ thống nguồn bên ngoài. Các cấu trúc bảng trong lớp này tương ứng với các cấu trúc bảng hệ thống nguồn "nguyên trạng", cùng với bất kỳ cột siêu dữ liệu bổ sung nào ghi lại ngày/giờ tải, ID của tiến trình, v.v.

Trọng tâm trong lớp này là thu thập dữ liệu thay đổi nhanh và khả năng cung cấp kho lưu trữ lịch sử của nguồn, xử lý lại nếu cần mà không cần đọc lại dữ liệu từ hệ thống nguồn.

- **Silver layer (dữ liệu được làm sạch):**

Trong lớp Silver của Lakehouse, dữ liệu từ lớp Bronze được khòp, hợp nhất, tuân thủ và làm sạch vừa đủ để lớp Silver có thể cung cấp "Chế độ xem doanh nghiệp" về tất cả các thực thể kinh doanh chính, khái niệm và giao dịch.

Lớp Silver đưa dữ liệu từ các nguồn khác nhau vào chế độ xem doanh nghiệp và cho phép phân tích tự phục vụ cho báo cáo đặc biệt, phân tích nâng cao và ML. Nó đóng vai trò là nguồn để các nhà phân tích bộ phận, kỹ sư dữ liệu và nhà khoa học dữ liệu tiếp tục tạo các dự án và phân tích nhằm giải đáp các vấn đề kinh doanh thông qua các dự án dữ liệu cấp phòng ban và doanh nghiệp trong lớp Gold.

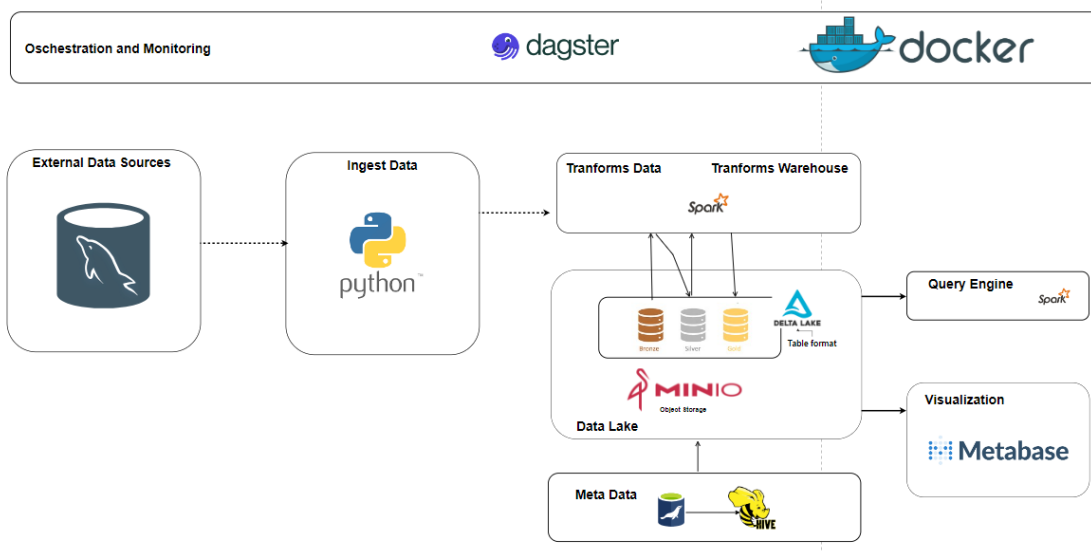
- **Gold layer (dữ liệu hướng doanh nghiệp):**

Dữ liệu trong lớp Gold của Lakehouse thường được sắp xếp trong cơ sở dữ liệu "dành riêng cho dự án" sẵn sàng sử dụng. Lớp Gold dùng để báo cáo và sử dụng nhiều mô hình dữ liệu không chuẩn hóa và tối ưu hóa việc đọc với ít kết nối hơn. Lớp chuyển đổi dữ liệu cuối cùng và quy tắc chất lượng dữ liệu được áp dụng ở đây. Lớp trình bày cuối cùng của các dự án như phân tích khách hàng, phân tích chất lượng sản phẩm, phân tích hàng tồn kho, phân khúc khách hàng, đề xuất sản phẩm, phân tích bán hàng, v.v. phù hợp với lớp này. Nhiều mô hình dữ liệu dựa trên lược đồ ngôi sao theo phong cách Kimball hoặc Data Mart theo phong cách Inmon phù hợp với lớp Gold này của Lakehouse.

CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG DATA PLATFORM

3.1. KIẾN TRÚC HỆ THỐNG

3.1.1. Kiến trúc tổng quan



Hình 3.1.1.1. Kiến trúc tổng quan về hệ thống Data Platform

Chi tiết về các thành phần và công nghệ chính trong kiến trúc trên:

- **Data source:** Dữ liệu thô được lưu trữ trong MySQL, MySQL là một hệ thống quản trị cơ sở dữ liệu mã nguồn mở (gọi tắt là RDBMS) hoạt động theo mô hình client-server. Với RDBMS là viết tắt của Relational Database Management System. MySQL được tích hợp apache, PHP. MySQL quản lý dữ liệu thông qua các cơ sở dữ liệu. Mỗi cơ sở dữ liệu có thể có nhiều bảng quan hệ chứa dữ liệu. MySQL cũng có cùng một cách truy xuất và mã lệnh tương tự với ngôn ngữ SQL.
- **Ingestion:** Quá trình tải dữ liệu từ nguồn (MySQL) vào lớp Bronze trong Data Lakehouse sử dụng thư viện của Python là Polars. Python được sử dụng để xây dựng các công cụ và kịch bản để trích xuất và tiền xử lý dữ liệu từ MySQL.
- **Transforms Data:** Dữ liệu thô từ lớp Bronze được chuyển đổi và tinh chế trong các lớp dữ liệu Silver và Gold. Quá trình này được thực hiện bằng các sử dụng Spark, một công cụ có tốc độ xử lý nhanh do việc tính toán được thực hiện trên nhiều máy khác nhau. Đồng thời việc tính toán được thực hiện ở bộ nhớ trong (in-memories) hay thực hiện hoàn toàn trên RAM.

- **Data Lakehouse:** Trong kiến trúc Lakehouse, Delta Lake được sử dụng làm định dạng bảng lưu trữ dữ liệu. Minio được sử dụng để lưu trữ dữ liệu dưới dạng object. Delta Lake là một công cụ mã nguồn mở xây dựng trên Apache Parquet và Apache Arrow, cung cấp tính năng ACID cho việc lưu trữ và quản lý dữ liệu. Hive Metastore được sử dụng để lưu trữ metadata cho các bảng dữ liệu, trong khi Apache Spark được sử dụng làm công cụ tính toán để truy vấn và xử lý dữ liệu.
- **Visualization:** Metabase là công cụ được sử dụng để trực quan hóa và khám phá dữ liệu. Metabase cung cấp giao diện đồ họa đơn giản và dễ sử dụng cho việc tạo các báo cáo, biểu đồ và truy vấn dữ liệu. Với Metabase, người dùng có thể khám phá dữ liệu một cách tương tác và tùy chỉnh theo nhu cầu của mình.
- **Orchestration:** Dagster thường được sử dụng trong vai trò của một công cụ orchestration trong quy trình xử lý dữ liệu. Dagster thường đảm nhận nhiệm vụ điều phối các công việc và quá trình xử lý dữ liệu, đồng bộ hóa chúng, và đảm bảo rằng chúng được thực hiện theo đúng thứ tự và có thể theo dõi được. Dagster không chỉ giúp quản lý các tasks mà còn cung cấp cơ sở hạ tầng cho việc theo dõi, đảm bảo chất lượng dữ liệu, và quản lý metadata.
- **Monitoring:** Docker là nền tảng phần mềm cho phép chúng dựng, kiểm thử và triển khai ứng dụng một cách nhanh chóng. Docker đóng gói phần mềm vào các đơn vị tiêu chuẩn hóa được gọi là container có mọi thứ mà phần mềm cần để chạy, trong đó có thư viện, công cụ hệ thống, mã code và thời gian chạy.

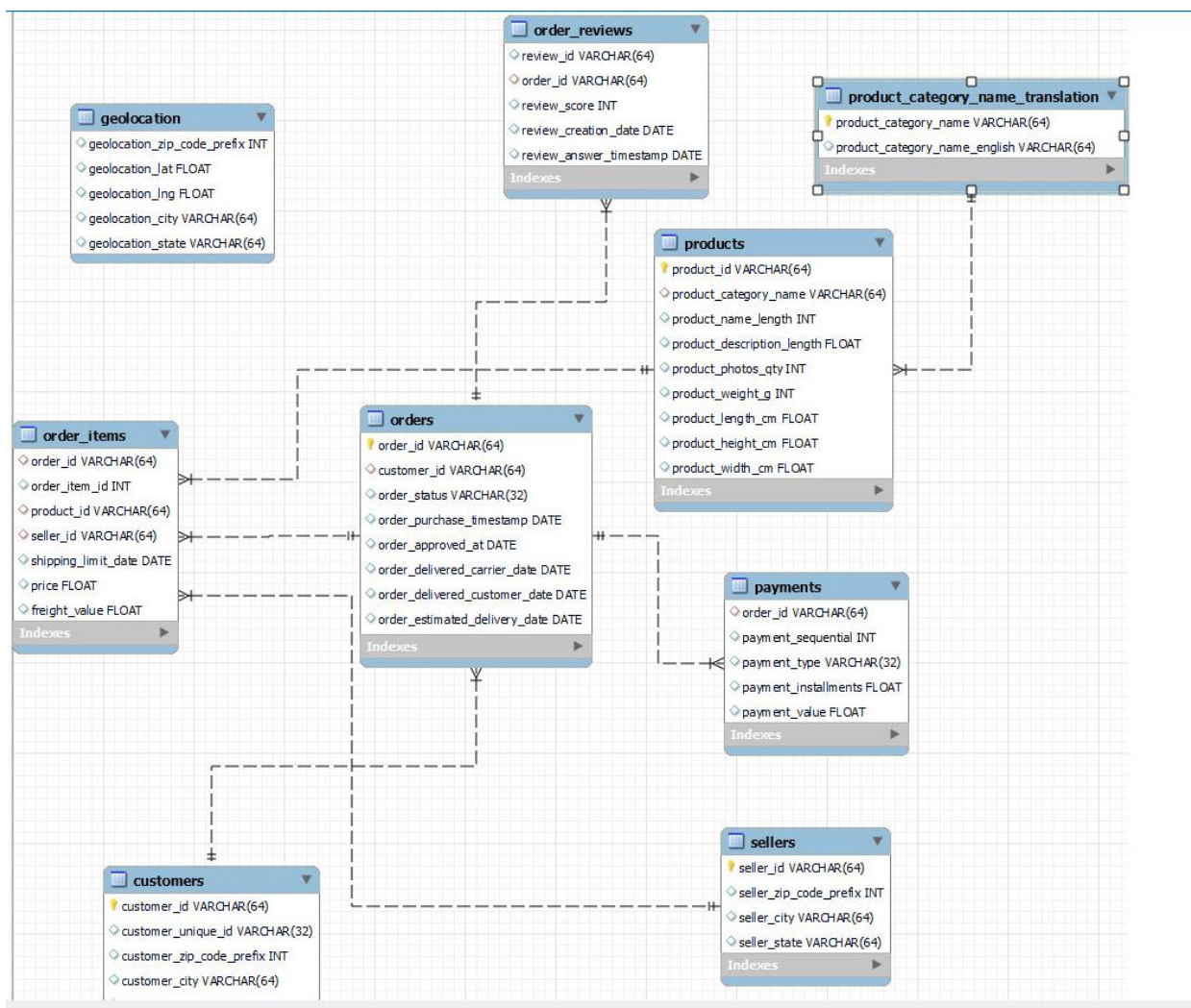
3.1.2. Tập dữ liệu nguồn

Trong dự án này, nhóm đã chọn tập dữ liệu có tên “**Brazilian E-Commerce Public Dataset by Olist**” được lấy trên trang kaggle. Đây là tập dữ liệu công khai về thương mại điện tử của Brazil về các đơn đặt hàng được thực hiện tại Olist Store. Tập dữ liệu có thông tin về 100 nghìn đơn đặt hàng từ năm 2016 đến năm 2018 được thực hiện tại nhiều thị trường ở Brazil. Các tính năng của nó cho phép xem đơn đặt hàng từ nhiều khía cạnh: từ trạng thái đơn hàng, giá cả, thanh toán và hiệu suất vận chuyển đến vị trí của khách hàng, thuộc tính sản phẩm và cuối cùng là đánh giá do khách hàng viết.

Tập dữ liệu bao gồm 9 bảng:

- `olist_customers_dataset`: Bảng dữ liệu này có thông tin về khách hàng và vị trí của khách hàng. Sử dụng nó để xác định các khách hàng duy nhất trong tập dữ liệu đơn hàng và tìm địa điểm giao đơn hàng.
- `olist_geolocation_dataset`: Bảng dữ liệu này có thông tin về mã zip của Brazil và tọa độ, vĩ độ của nó.
- `olist_order_items_dataset`: Bảng dữ liệu này bao gồm dữ liệu về các mặt hàng được mua trong mỗi đơn hàng.
- `olist_order_payments_dataset`: Bảng dữ liệu này bao gồm dữ liệu về các tùy chọn thanh toán đơn hàng.
- `olist_order_reviews_dataset`: Bảng dữ liệu này bao gồm dữ liệu về các đánh giá của khách hàng đối với các đơn hàng.
- `olist_orders_dataset`: Bảng là tập dữ liệu cốt lõi. Mọi thông tin cần thiết đều được ánh xạ tới từng đơn hàng trong này.
- `olist_products_dataset`: Bảng dữ liệu này bao gồm dữ liệu về các sản phẩm được bán bởi Olist.
- `olist_sellers_dataset`: Bảng dữ liệu này bao gồm dữ liệu về người bán đã thực hiện các đơn đặt hàng được thực hiện tại Olist.
- `product_category_name_translation`: Bảng dữ liệu này bao gồm tên của các sản phẩm được dịch sang tiếng Anh.

Lược đồ Diagram giữa các bảng:

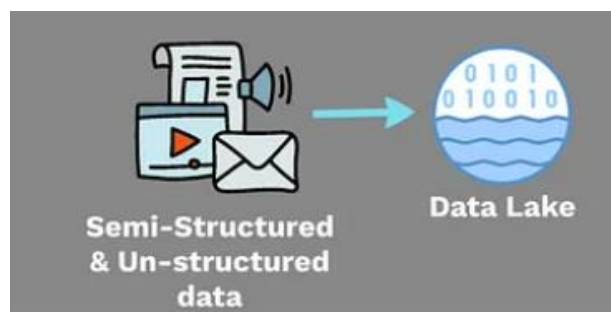


Hình 3.1.2.1. Lược đồ Diagram giữa các bảng trong cơ sở dữ liệu MySQL

3.2. QUY TRÌNH HOẠT ĐỘNG CỦA KIẾN TRÚC LAKEHOUSE TRONG ỨNG DỤNG PLATFORM

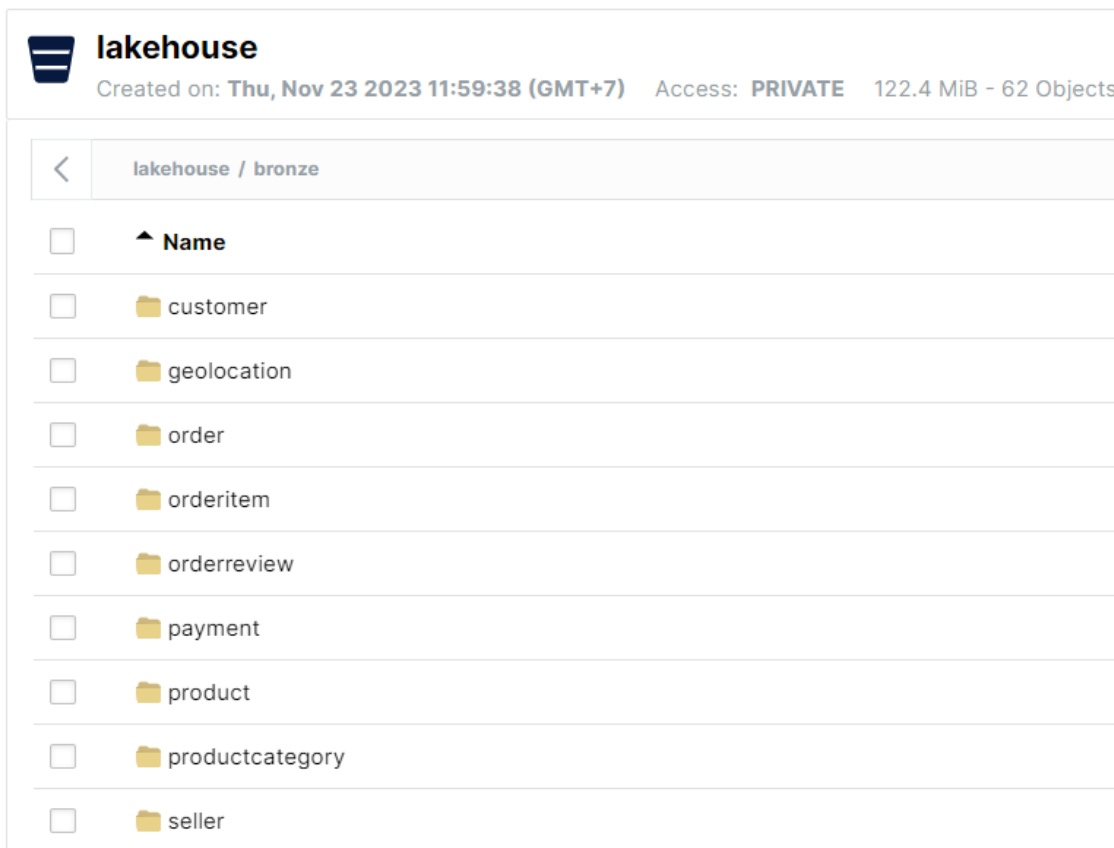
3.2.1. Tổng quan về các quy trình

- **Bronze Layer:** đại diện cho một bản sao gần giống của nguồn cho mục đích kiểm tra và truy xuất nguồn gốc.



Hình 3.2.1.1. Extract data từ nguồn vào Data Lake

Sử dụng thư viện mới Polars của python connect với database mysql để extract data từ nguồn để đẩy vào Data Lake.



Hình 3.2.1.2. Load data into Bronze Layer

Load data đã extract, load vào lớp Bronze để lưu raw data. Ở đây data được lưu dưới dạng file Parquet. Một định dạng lưu trữ data olap theo cột.

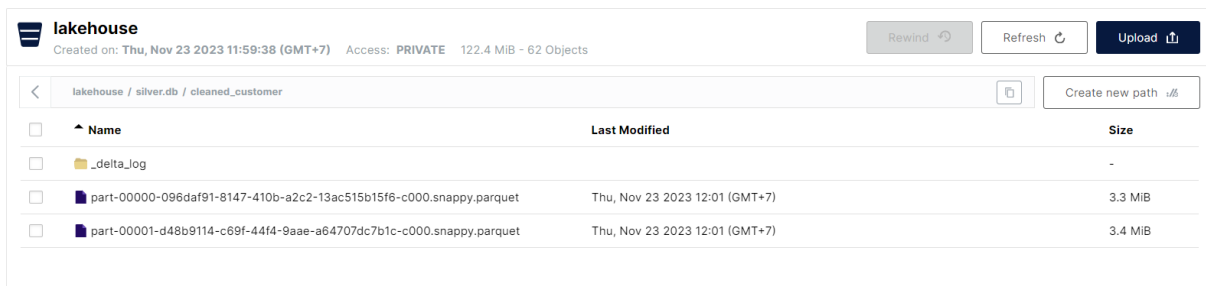
- **Silver Layer:** chứa dữ liệu đã được tinh chỉnh thường được yêu cầu cho các trường hợp sử dụng ML. Dữ liệu này vẫn gần với dữ liệu gốc và chứa một số tập hợp nhưng không được cấu trúc để hỗ trợ các trường hợp sử dụng BI như Data Warehouse.



Hình 3.2.1.3. Transform Data từ Bronze Layer to Silver Layer

Data được đẩy sang lớp Silver để transform như: làm sạch, biến đổi, sắp xếp, xóa bỏ trùng lặp, tổng hợp.

Data sau khi đã làm sạch thì được lưu xuống định dạng file Delta Lake như sau:



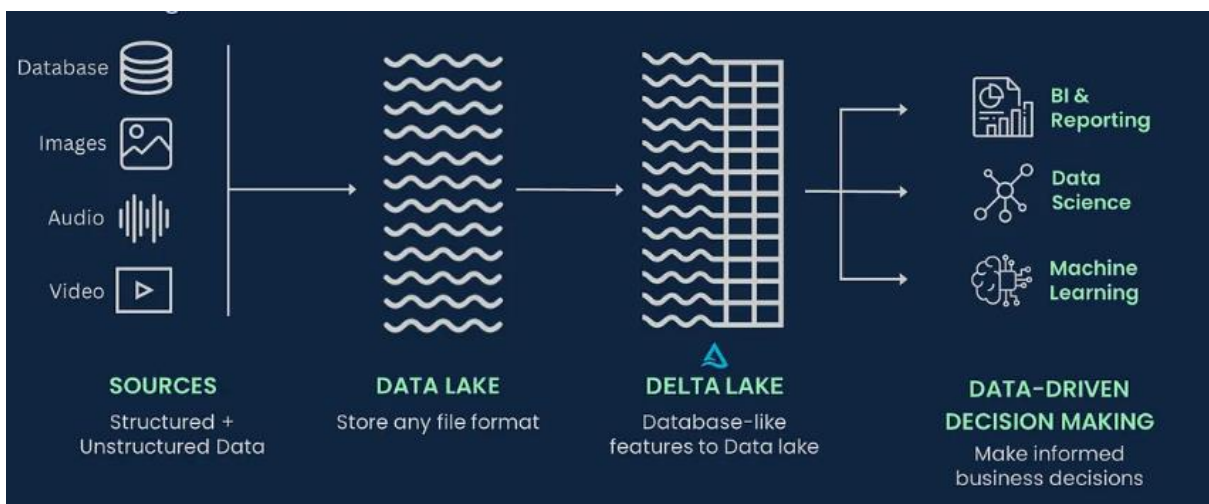
The screenshot shows the Lakehouse interface with the following details:

- lakehouse** header: Created on: Thu, Nov 23 2023 11:59:38 (GMT+7), Access: PRIVATE, 122.4 MiB - 62 Objects.
- Navigation: Rewind, Refresh, Upload buttons.
- Path: lakehouse / silver.db / cleaned_customer.
- Table with columns: Name, Last Modified, Size.

<input type="checkbox"/>	Name	Last Modified	Size
<input type="checkbox"/>	_delta_log		-
<input type="checkbox"/>	part-00000-096daf91-8147-410b-a2c2-13ac515b15f6-c000.snappy.parquet	Thu, Nov 23 2023 12:01 (GMT+7)	3.3 MiB
<input type="checkbox"/>	part-00001-d48b9114-c69f-44f4-9aae-a64707dc7b1c-c000.snappy.parquet	Thu, Nov 23 2023 12:01 (GMT+7)	3.4 MiB

Hình 3.2.1. 4. Định dạng file lưu trữ của Delta Lake

Delta Lake còn gọi là table format, linh hồn của kiến trúc lakehouse:



Hình 3.2.1.5. Vai trò của Delta Lake trong kiến trúc Lakehouse

```
t = silver_cleaned_customer"
df = spark.read.table(t)

df.show(10)
```

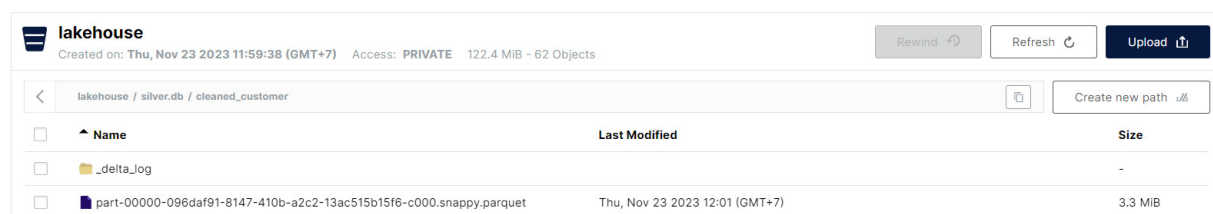
	customer_id	customer_unique_id	customer_rip_code_prefix	customer_city	customer_state
0054556ea954a76ad...	3cc8e80baa86a7bef...	95680	canela	RS	
009b0d84996868a85...	63df52c362d4b7183...	13076	campinas	SP	
00ab3f0ac737c0b...	d4d7e1cbf735af1f46...	6226	osasco	SP	
02625456239ab29f0...	570eb70ff97166b85...	31939	belo horizonte	MG	
028514f8b6e8c2ad...	be4d46b2c26abc028...	35200	aimores	MG	
033225f6250f8ebf7...	07b9a3d7d17b53a55...	83702	aracaria	PR	
03514f9ba6258fad...	3a4f5d794447ef719...	75115	anapolis	GO	
04bf1a1d175b07ee...	6afbf81bd0c7c75be...	13295	itupeva	SP	
05f49d46685d544c...	d9a1aaab532b2af5b...	51250	recife	PE	
067f45ab00b8417ea...	26b0aa1978f09039f...	57072	maceio	AL	

```
get_x.show(df.columns[0:18].names)
```

Hiệu đơn giản là dữ liệu vẫn được lưu trữ dưới dạng các file parquet trên Data Lake, nhưng Delta Lake đã ánh xạ các file data này thành dạng table và có schema.

Cách mà table format hoạt động như ghi data và ghi lại giao dịch:

Cấu trúc của 1 file parquet cũng như là 1 table sau khi được ghi xuống dưới dạng delta table.



lakehouse

Created on: Thu, Nov 23 2023 11:59:38 (GMT+7) Access: PRIVATE 122.4 MiB - 62 Objects

Rewind ↶

Refresh ↺

Upload 📁

< lakehouse / silver.db / cleaned_customer / _delta_log

🗑️

Create new path ➦

<input type="checkbox"/>	Name	Last Modified	Size
<input type="checkbox"/>	000000000000000000000000.json	Thu, Nov 23 2023 12:01 (GMT+7)	2.7 KiB

47

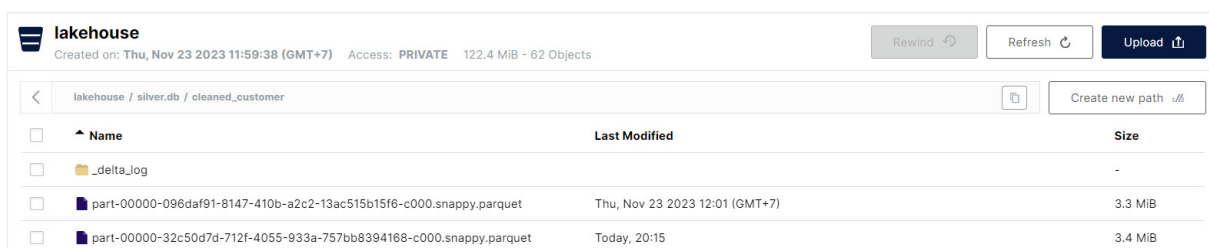
Tập Parquet chứa dữ liệu thực tế và tập JSON lưu trữ nhật ký, chẳng hạn như lịch sử phiên bản hoặc loại hoạt động nào đã được thực hiện (update, insert) và tất cả các nhật ký khác.

Log file mẫu như sau:



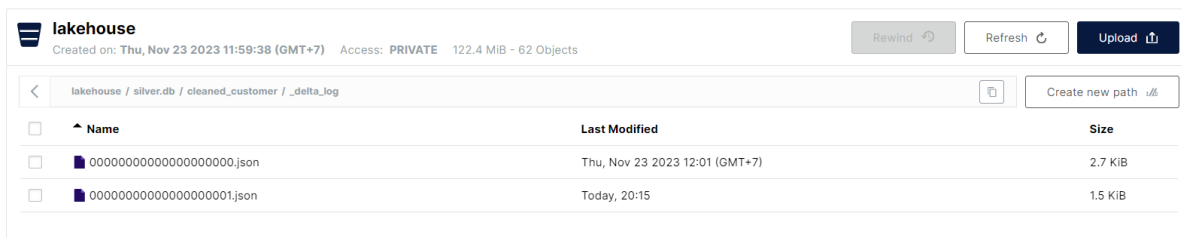
Hình 3.2.1.9. File log lưu trữ nhật ký

Nếu như chúng ta load lại data định dạng format trong cùng 1 bảng nó sẽ như sau:



Hình 3.2.1.10. Table Format khi load lại data

Nó sẽ tạo thêm một tệp parquet và một tệp JSON. Nó sẽ tải dữ liệu vào tệp Parquet và tất cả siêu dữ liệu vào tệp JSON.



Hình 3.2.1.11. Thư mục delta log khi load lại data

- **Gold Layer:** rất gần với biểu diễn thực tế và kích thước của Data Warehouse, đồng thời chứa các tập hợp giúp sử dụng lớp này để phân tích và báo cáo BI dễ dàng hơn.

Tại Gold Layer: ở đây data được transform theo yêu cầu Bussiness và kiến trúc Data Warehouse.

Dữ liệu Data Warehouse sau khi được transform được lưu lại dưới dạng file parquet ở lớp Gold dùng để truy suất, phân tích và phục vụ các tác vụ BI chuyên sâu.

Sau khi Dữ Liệu được ELT xuyên suốt và liên tục từ Bronze- Silver- Gold thì dữ

liệu đã sẵn sàng kích hoạt trên tất cả các lớp và sẵn sàng phục vụ để phân tích quản trị, ML, BI...

3.2.2. Thực hiện truy vấn và quản lý metadata của dự án

Việc tích hợp Delta Lake, công cụ truy vấn SQL và danh mục dữ liệu vào Data Lakehouse mang lại một số lợi thế. Nó cho phép thực hiện các hoạt động hợp nhất, cập nhật và xóa trên các tệp dữ liệu được lưu trữ trong Data Lake. Hơn nữa, Lakehouse có thể xử lý liền mạch cả dữ liệu hàng loạt và dữ liệu truyền phát, loại bỏ các kho dữ liệu và thúc đẩy sự hợp tác trong toàn tổ chức.

- **Query Engine:** Khi các bảng đã được ghi dưới dạng Table Format được thiết lập, ta có thể thực thi các truy vấn SQL bao gồm các lệnh DML và DDL như UPDATE TABLE như bên dưới. Nếu không có Delta Lake, ta gần như bị giới hạn ở bản sao dữ liệu chỉ đọc mà không hỗ trợ cập nhật, xóa và chèn.

Các đoạn mã sau thực hiện Query trên các table trên Data Lake:

```
spark.sql('SELECT * FROM silver.cleaned_customer').limit(5).show()
```

customer_id	customer_unique_id	customer_zip_code_prefix	customer_city	customer_state
0054556ea954a76ad...	3cc8e80baa86a7bef...	95680	canela	RS
009bd844996868ab5...	63df52c362d4b7183...	13076	campinas	SP
00abf30c1a93c7c8b...	d43e7cbf7354f1f46...	6226	osasco	SP
02625456293ab29f0...	570eb70ff97166b85...	31930	belo horizonte	MG
028514f8be6e8c2ad...	be4d46b2c26abc028...	35200	aimores	MG

Hình 3.2.2.1. Select tất cả record của bảng clean_customer

```
spark.sql("UPDATE silver.cleaned_customer SET customer_state = 'SP' WHERE customer_id = '0054556ea954a76ad6f9c4ba79d34a98'")
```

```
DataFrame[num_affected_rows: bigint]
```

Hình 3.2.2.2. Update một record trong bảng clean_customer


```
spark.sql("SELECT count(*) FROM silver.cleaned_customer").show()
```

```
+-----+
|count(1)|
+-----+
|   99441|
+-----+
```

```
spark.sql("SELECT DISTINCT(customer_id) FROM silver.cleaned_customer").show(5,False)
```

```
+-----+
|customer_id|
+-----+
|36a1aa63bf2ebcd4911e026092700610|
|512f27d822abe6af95d86529e73724a6|
|384fbbcdcf45c174ca6407d4ade90112|
|4632eb5a8f175f6fe020520ae0c678f3|
|174cf4e5e95b5a49bac9cee9ef6cef70|
+-----+
only showing top 5 rows
```

Hình 3.2.2.3. Select xem tất cả có bao nhiêu bản ghi và giá trị duy nhất của từng bản ghi

- **Metadata:** Lakehouse cung cấp cái nhìn toàn diện về dữ liệu được lưu trữ trong Data Lake, tạo điều kiện cho sự cộng tác và quản lý dữ liệu tốt hơn trong toàn tổ chức.

ID	DB_ID	DESC	DB_LOCATION_URI	NAME	OWNER_NAME	OWNER_TYPE	CTGL_NAME
1		Default Hive database	file:/user/hive/warehouse	default	public	ROLE	hive
2	6		s3a://lakehouse/silver.db	silver	root	USER	hive
3	7		s3a://lakehouse/gold.db	gold	root	USER	hive
4	8		s3a://lakehouse/platinum.db	platinum	root	USER	hive

Hình 3.2.2.4. Bảng metadata DBS

Bảng trên cho biết nơi lưu trữ Data Lake cũng như user đã tạo, tên và nơi lưu trữ database.

ID	SD_ID	CD_ID	INPUT_FORMAT	IS_COMPRESSED	IS_STOREDASUBDIRECTORIES	LOCATION	NUM_BUCKETS	OUTPUT_FORMAT
1	1	1	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/silver.db/cleaned_customer	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF
2	2	2	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/silver.db/cleaned_geolocation	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF
3	3	3	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/silver.db/date	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF
4	4	4	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/silver.db/cleaned_order	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF
5	5	5	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/silver.db/cleaned_order_item	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF
6	6	6	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/silver.db/cleaned_payment	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF
7	7	7	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/silver.db/cleaned_order_review	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF
8	8	8	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/silver.db/cleaned_product	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF
9	9	9	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/silver.db/cleaned_seller	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF
10	10	10	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/silver.db/cleaned_product_category	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF
11	11	11	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/gold.db/dim_customer	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF
12	12	12	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/gold.db/dim_date	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF
13	13	13	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/gold.db/dim_geolocation	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF
14	14	14	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/gold.db/dim_order	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF
15	15	15	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/gold.db/dim_product	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF
16	16	16	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/gold.db/dim_seller	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF
17	17	17	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/gold.db/dim_review	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF
18	18	18	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/gold.db/fact_table	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF
19	19	19	org.apache.hadoop.mapred.SequenceFileInputFormat	0	0	s3a://lakehouse/platinum.db/cube_sale	-1	org.apache.hadoop.hive ql.io.HiveSequenceFileOutputF

Hình 3.2.2.5. Bảng metadata SDS

Bảng trên cho biết vị trí lưu các table trên Data Lake, output format.

3.3. THIẾT KẾ DATA WAREHOUSE

3.3.1. Dimensional Data Modeling

Dimensional Data Modeling là một trong những kỹ thuật mô hình hóa dữ liệu được sử dụng trong thiết kế kho dữ liệu. Khái niệm Dimensional Data Modeling được phát triển bởi Ralph Kimball, bao gồm các fact và dimension. Vì mục tiêu chính của mô hình này là cải thiện việc truy xuất dữ liệu nên nó được tối ưu hóa cho Select Operation. Ưu điểm của việc sử dụng mô hình này là chúng ta có thể lưu trữ dữ liệu theo cách dễ dàng lưu trữ và truy xuất dữ liệu hơn sau khi được lưu trữ trong kho dữ liệu. Dimensional Model là mô hình dữ liệu được nhiều hệ thống OLAP sử dụng.

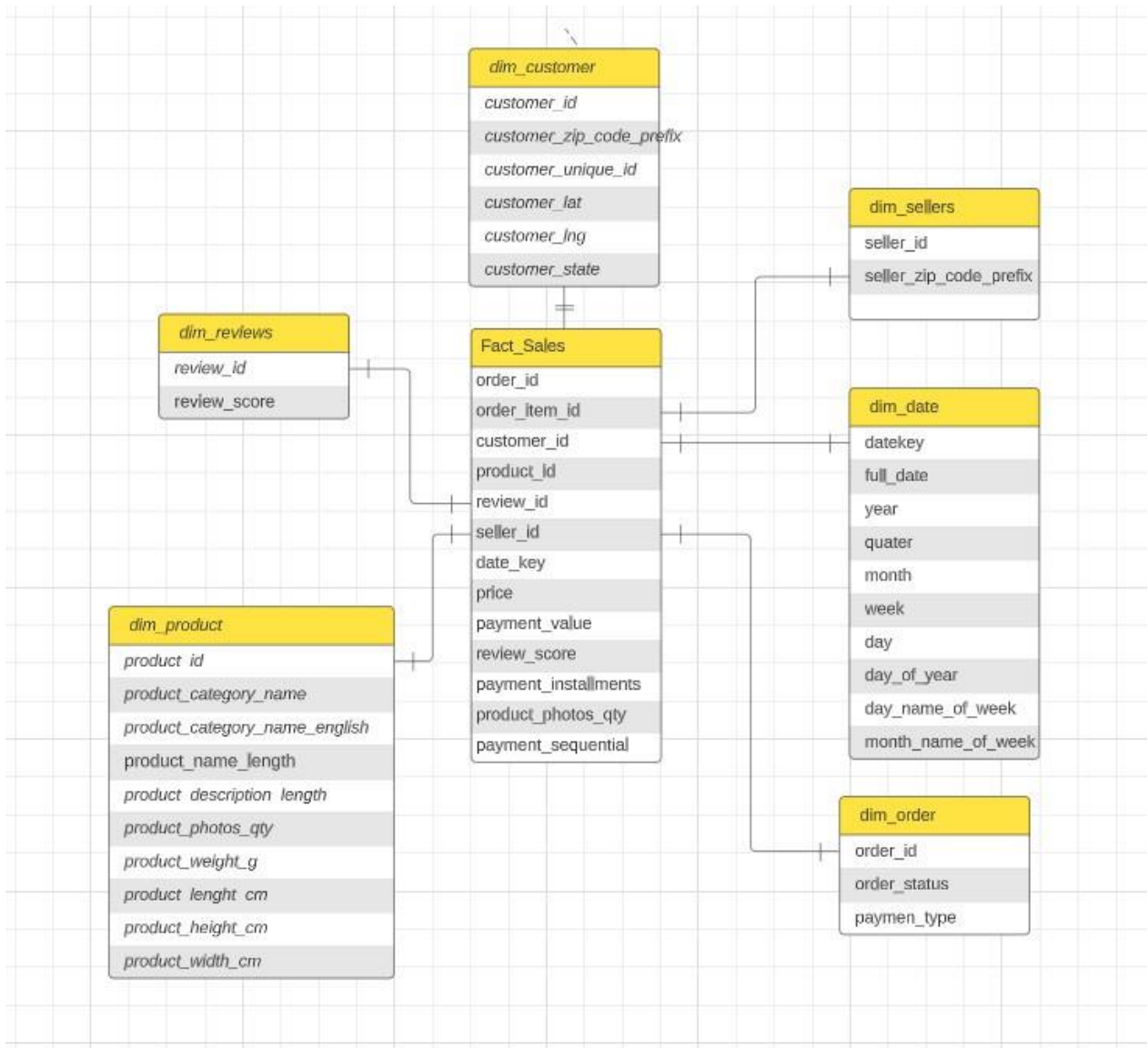
Các thành phần của Dimensional Data Model:

- **Facts:** là các yếu tố dữ liệu có thể đo lường được, đại diện cho các số liệu kinh doanh được quan tâm. Ví dụ: trong kho dữ liệu bán hàng, facts có thể bao gồm doanh thu bán hàng, số lượng đơn vị đã bán và tỷ suất lợi nhuận. Mỗi fact được liên kết với một hoặc nhiều chiều, tạo ra mối quan hệ giữa fact và dữ liệu mô tả.
- **Dimension:** Dimension là các thành phần dữ liệu mô tả được sử dụng để phân loại dữ liệu. Ví dụ: trong kho dữ liệu bán hàng, các dimension có thể bao gồm sản phẩm, khách hàng, thời gian và địa điểm. Mỗi dimension được tạo thành từ một tập hợp các thuộc tính mô tả dimension. Ví dụ: dimension sản phẩm có thể bao gồm các thuộc tính như tên sản phẩm, danh mục sản phẩm và giá sản phẩm.
- **Attributes:** Đặc điểm của dimension trong mô hình hóa dữ liệu được gọi là đặc điểm. Chúng được sử dụng để lọc, tìm kiếm dữ kiện, v.v. Đối với dimension vị trí, các thuộc tính có thể là bang, quốc gia, mã zip, v.v.
- **Fact table:** Trong Dimensional data model, bảng fact là bảng trung tâm chứa các thước đo hoặc số liệu quan tâm, được bao quanh bởi các bảng dimension mô tả các thuộc tính của độ đo. Các bảng dimensions có liên quan đến bảng fact thông qua các mối quan hệ khóa ngoại.
- **Dimension table:** Các dimensions của fact được đề cập trong bảng dimension và về cơ bản chúng được nối với nhau bằng khóa ngoại. Bảng dimension đơn giản là các bảng không chuẩn hóa. Các dimensions có thể có một hoặc nhiều mối quan hệ.

Các bước để tạo Dimensional Data Model:

- **Bước 1: Xác định mục tiêu kinh doanh:** Bước đầu tiên là xác định mục tiêu kinh doanh. Bán hàng, nhân sự, tiếp thị, v.v. là một số ví dụ về nhu cầu của tổ chức. Vì đây là bước quan trọng nhất của Dimensional Data Modeling nên việc lựa chọn mục tiêu kinh doanh cũng phụ thuộc vào chất lượng dữ liệu có sẵn cho quy trình đó.
- **Bước 2: Xác định mức độ chi tiết:** Mức độ chi tiết là mức thông tin thấp nhất được lưu trữ trong bảng. Mức độ chi tiết của các vấn đề kinh doanh và giải pháp của nó được mô tả bởi Grain.
- **Bước 3: Xác định Dimensions và thuộc tính của chúng:** Dimensions phân loại và mô tả các facts và độ đo của kho dữ liệu theo cách hỗ trợ các câu trả lời có ý nghĩa cho các câu hỏi kinh doanh. Kho dữ liệu tổ chức các thuộc tính mô tả dưới dạng cột trong bảng thứ nguyên. Ví dụ: date dimension có thể chứa dữ liệu như năm, tháng và ngày trong tuần.
- **Bước 4: Xác định fact:** Dữ liệu có thể đo lường được lưu giữ trong bảng fact. Hầu hết các hàng của bảng fact là các giá trị số như giá hoặc chi phí trên mỗi đơn vị, v.v.
- **Bước-5: Xây dựng lược đồ schema:** Một lược đồ là một cấu trúc cơ sở dữ liệu. Có hai sơ đồ phổ biến: lược đồ sao và lược đồ bông tuyết.

3.3.2. Bảng thiết kế Data Warehouse



Hình 3.3.2.1. Bảng thiết kế Data Warehouse

Trong mô hình trên, nhóm xây dựng các bảng dim dựa trên các nghiệp vụ từ dữ liệu, ở đây, từng dim sẽ được lấy được ở những nơi khác nhau, từng dim sẽ đại diện cho 1 tập dataset trên tập data mẫu. Các bảng dim bao gồm:

- **dim_product**: bảng dim này thể hiện cho đối tượng là sản phẩm, có key là product_id, thể hiện sự duy nhất của mỗi dòng dữ liệu. Các thuộc tính liên quan đến sản phẩm đó bao gồm: tên sản phẩm, mô tả sản phẩm và các kích thước liên quan đến sản phẩm đó.
- **dim_review**: bảng dim này thể hiện sự đánh giá giữa người dùng với sản phẩm

mà họ mua. Mỗi sản phẩm sẽ có 1 điểm số do người dùng tự đánh giá.

- `dim_customer`: bảng dim này thể hiện cho đối tượng là khách hàng. Bao gồm các thông tin liên quan đến khách hàng như mã code của khu vực, vị trí tọa độ cũng như tiểu bang mà khách hàng đó ở.
- `dim_sellers`: bảng dim này thể hiện cho đối tượng là người bán hàng, người đã hoàn thành các đơn cho khách hàng.
- `dim_order`: bảng dim này thể hiện cho các đơn hàng, bao gồm tình trạng của đơn hàng (đang chuẩn bị, đang vận chuyển, ...) và phương thức thanh toán của đơn hàng đó.
- `dim_date`: bảng dim này thể hiện đối tượng thời gian.

Tiếp theo là bảng fact, nhóm thiết kế bảng `Fact_Sales` dùng để lưu trữ thông tin chính xác về chi tiết các giao dịch bán hàng, chi tiết các thuộc tính trong bảng fact:

"`order_id`": Mã đơn hàng, định danh duy nhất cho mỗi đơn hàng.

"`order_item_id`": Mã hàng hóa trong đơn hàng, định danh duy nhất cho mỗi mặt hàng trong đơn hàng.

"`customer_id`": Mã khách hàng, định danh duy nhất cho mỗi khách hàng.

"`product_id`": Mã sản phẩm, định danh duy nhất cho mỗi sản phẩm được bán.

"`review_id`": Mã đánh giá, định danh duy nhất cho mỗi đánh giá về sản phẩm.

"`seller_id`": Mã người bán, định danh duy nhất cho mỗi người bán.

"`date_key`": Khóa ngày, đại diện cho ngày giao dịch diễn ra.

"`price`": Giá sản phẩm.

"`freight_value`": Giá vận chuyển.

"`payment_value`": Giá trị thanh toán của đơn hàng.

"`payment_installments`": Số lần trả góp trong thanh toán.

"`payment_sequential`": Thứ tự thanh toán trong đơn hàng.

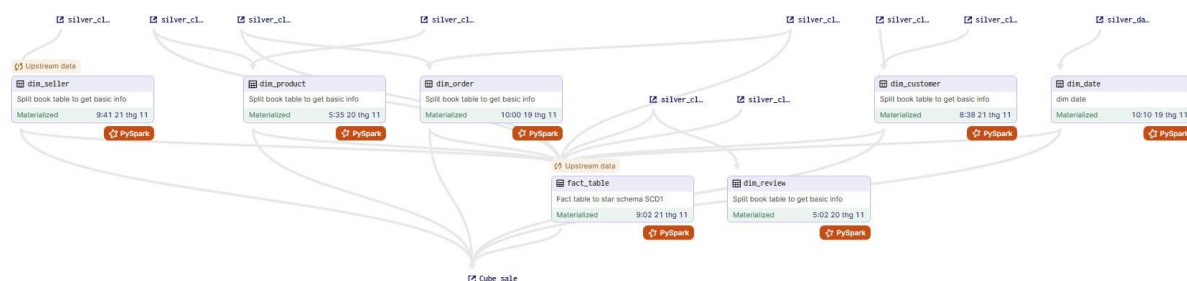
3.4. DATA LINEAGE

Chi tiết Bronze Layer:



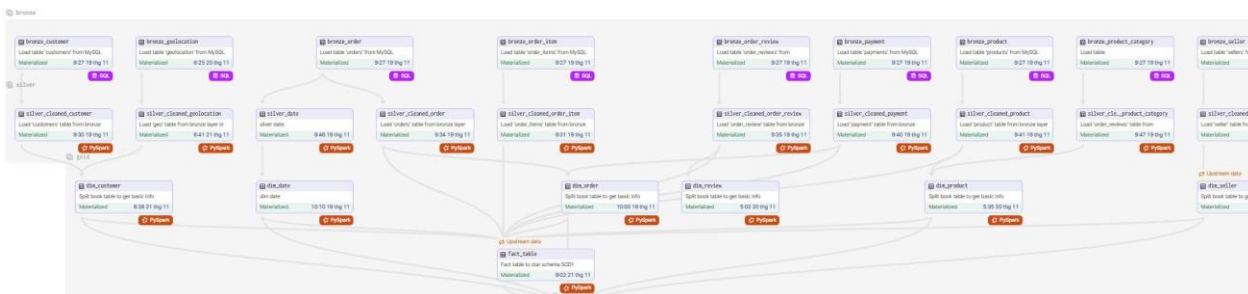
Hình 3.4.1. Data Lineage của Bronze Layer

Chi tiết Silver Layer:



Hình 3.4.2. Data Lineage của Silver Layer

Chi tiết Gold Layer:

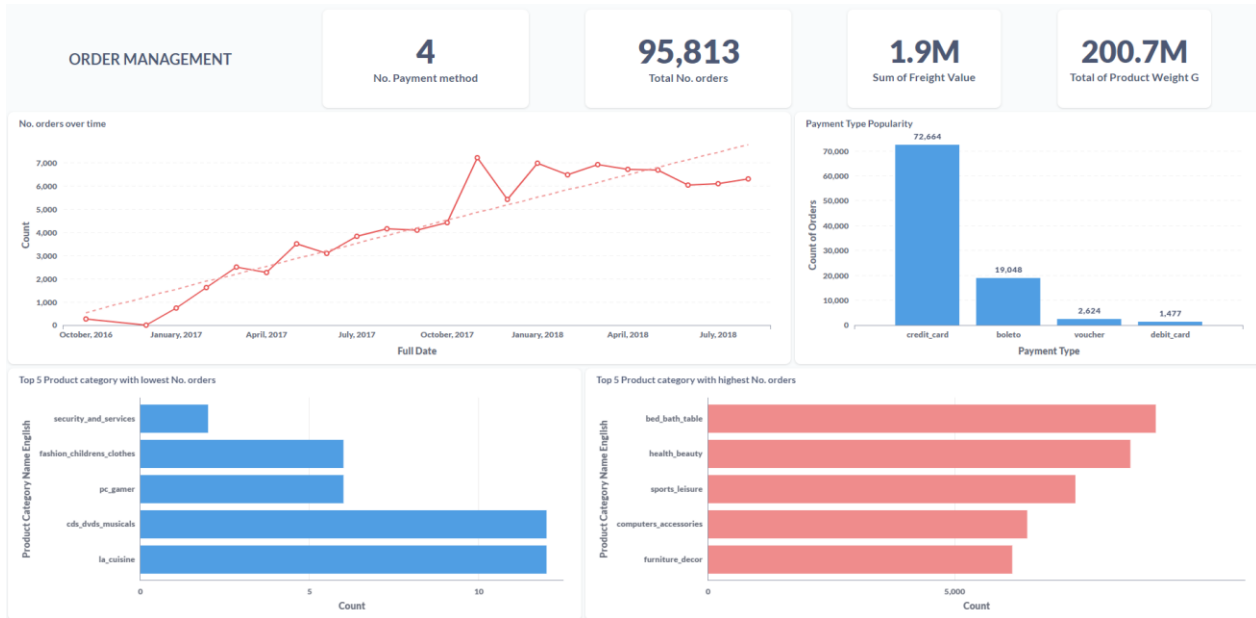


Hình 3.4.3. Data Lineage của Gold Layer

3.5. XÂY DỰNG BẢNG BÁO CÁO

Từ Gold Layer, ta có thể dùng Metabase để có thể trực quan hóa data thành những biểu đồ để thể hiện thông tin một cách rõ ràng hơn. Ở đây, nhóm đã cũng cấp 3 dashboard chính biểu diễn thông tin dữ liệu từ Gold Layer bao gồm: Order Management (Quản lý đơn hàng), Product Analysis (Phân tích sản phẩm) và Customer and Seller (Khách hàng và nhân viên bán hàng).

Order Management Dashboard:



Hình 3.5.1. Báo cáo dữ liệu về việc quản lý các đơn hàng

Từ hình 3.5.1, ta có thể thống kê được các đơn hàng đã giao từ năm 2016 – 2018. Có tổng cộng 95.813 đơn hàng đã được giao, tổng số tiền vận chuyển cho các đơn hàng đạt gần 2 triệu \$ và khối lượng hàng hóa được vận chuyển là 200 tấn.

Chi tiết một số biểu đồ trong dashboard trên:

- Biểu đồ “*No. orders over time*” thể hiện số đơn hàng đã được vận chuyển qua từng tháng theo từng năm. Ta thấy, các đơn hàng có xu hướng tăng và tổng số đơn hàng cao nhất thuộc tháng 11 năm 2017 với khoảng 7200 đơn hàng.
- Biểu đồ “*Payment Type Popularity*” thể hiện số lượt thanh toán theo các phương thức với mỗi đơn hàng. Chủ yếu khách hàng thanh toán bằng phương thức sử dụng thẻ credit (Thẻ tín dụng).
- Biểu đồ “*Top 5 Product category with lowest No. orders*” và “*Top 5 Product category with highest No. orders*” cho biết các sản phẩm nào được order nhiều nhất và sản phẩm nào được order ít nhất, từ đó công ty có thể đưa ra các chiến lược quảng cáo hoặc quan tâm đến sản phẩm đó nhiều hơn.

Product Analysis:



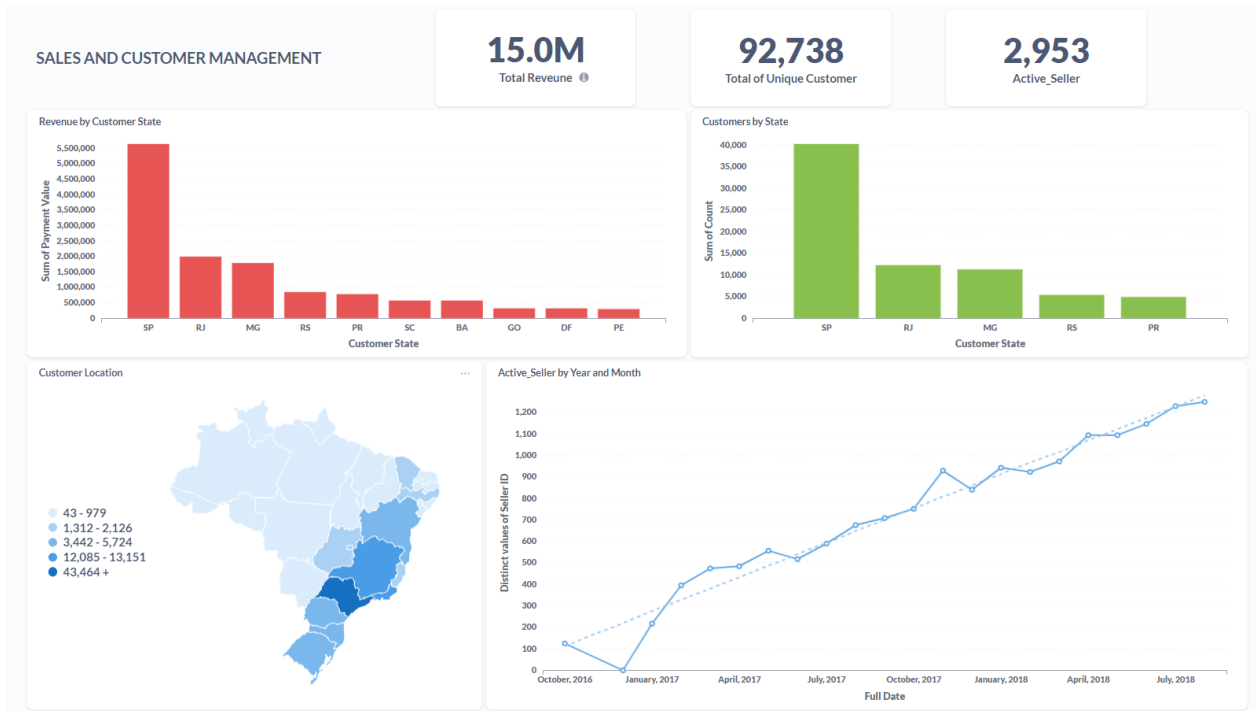
Hình 3.5.2. Báo cáo dữ liệu về việc phân tích sản phẩm

Từ hình 3.5.2, ta có thể xác định được các sản phẩm bán chạy, dự đoán được nhu cầu của khách hàng và đánh giá hiệu suất sản phẩm. Tổng quan từ năm 2016-2018 công ty có 73 sản phẩm tất cả và tổng số lượng sản phẩm bán ra là 102.500.

Chi tiết một số biểu đồ trong dashboard trên:

- Biểu đồ “*No. Sold product over time*” thể hiện số sản phẩm bán ra từ năm 2016-2018, số lượng có xu hướng tăng qua từng tháng của năm.
- Biểu đồ “*Top 10 Product Categories with Highest Revenue*” cho biết top 10 loại sản phẩm có doanh thu cao nhất.
- Biểu đồ “*Top 10 Product Categories with Highest selling quantity*” và “*Top 10 Product Categories with Lowest selling quantity*” cho biết top 10 sản phẩm bán ra với số lượng nhiều nhất và top 10 sản phẩm bán ra với số lượng ít nhất.

Sale and Customer Management:



Hình 3.5.3. Báo cáo dữ liệu về phân tích doanh thu và khách hàng

Từ hình 3.5.3, ta có thể tính được doanh thu và hiểu rõ hơn về khách hàng của mình. Doanh thu trong 3 năm đạt khoảng 15 triệu \$, tổng số khách hàng đạt khoảng 93.000 và tổng số nhân viên đã làm việc khoảng 3.000 người.

Chi tiết một số biểu đồ trong dashboard trên:

- Biểu đồ “*Revenue by Customer state*” thể hiện doanh thu theo từng khu vực, chi tiết là các bang của Brazil. Bang Sau Paulo đạt doanh thu cao nhất khoảng 5.500\$.
- Biểu đồ “*Customer by State*” thống kê tổng số khách hàng phân bố theo từng tiểu bang.
- Biểu đồ “*Customer Location*” trực quan tổng số khách hàng trên một map. Mức độ phân bố đông đúc hay thưa thớt dựa vào màu sắc. Càng đậm thì khu vực đó khách hàng càng nhiều.
- Biểu đồ “*Active_Seller by Year and Month*” thống kê số nhân viên thực sự làm trong mỗi tháng qua mỗi năm. Với số lượng đơn hàng càng tăng kéo theo việc nhân viên cũng tăng theo.

PHẦN 3: KẾT LUẬN

1. Kết quả đạt được

Nhóm đã nghiên cứu về các khái niệm cơ bản, các thành phần chính và các lợi ích của Modern Data Stack. Nhóm cũng đã thực hiện một ví dụ minh họa về cách áp dụng Modern Data Stack vào việc xây dựng một nền tảng phân tích dữ liệu cho một doanh nghiệp bán hàng trực tuyến. Nhóm đã sử dụng các công cụ như Spark, Minio, Delta Lake, Metabase Dagster và Docker để thiết kế và triển khai một pipeline dữ liệu từ nguồn đến đầu ra. Nhóm đã đánh giá kết quả của đề tài qua các tiêu chí như tính khả thi, tính hiệu quả, tính mở rộng và tính linh hoạt của giải pháp.

Kết quả của đề tài cho thấy Modern Data Stack là một xu hướng mới trong lĩnh vực phân tích dữ liệu, mang lại nhiều lợi thế cho các doanh nghiệp về khả năng thu thập, xử lý, phân tích và trực quan hóa dữ liệu một cách nhanh chóng, đơn giản và chi phí thấp. Đề tài cũng góp phần nâng cao kiến thức và kỹ năng của nhóm về các công cụ và công nghệ mới trong Modern Data Stack. Nhóm hy vọng đề tài sẽ có ích cho những ai quan tâm đến chủ đề này và mong muốn áp dụng Modern Data Stack vào việc xây dựng Platform cho doanh nghiệp của mình.

2. Hạn chế

Đề tài chỉ tập trung vào một số công cụ và kỹ thuật phổ biến trong Modern Data Stack, chưa đưa ra được những so sánh và đánh giá toàn diện về các lựa chọn khác nhau.

Đề tài chưa khai thác được những tiềm năng và thách thức của Modern Data Stack trong bối cảnh dữ liệu ngày càng phong phú và đa dạng, cũng như những yêu cầu cao về tính an toàn, bảo mật và hiệu năng của platform.

Đề tài chưa áp dụng tới các kiến trúc phổ biến hơn như delta, kappa để minh họa luồng dữ liệu streaming.

Chỉ mới áp dụng mô hình DW cơ bản, chưa đi sâu vào kiến trúc để ghi lại dữ liệu lịch sử

3. Hướng phát triển

Nếu có thời gian thêm để nghiên cứu, nhóm sẽ:

Áp dụng Continuous Integration/Continuous Deployment (Tích hợp liên tục/ Triển khai liên tục) vào nền tảng.

Nghiên cứu về những xu hướng và thách thức mới của Modern Data Stack trong bối cảnh dữ liệu ngày càng phong phú và đa dạng hơn, như Big Data, Streaming Data, Unstructured Data, Semi-structured Data, etc. Tìm ra những giải pháp sáng tạo và hiệu quả để khai thác và tận dụng tối đa giá trị của dữ liệu.

Áp dụng Modern Data Stack để xây dựng các giải pháp data-driven cho các doanh nghiệp và tổ chức, giúp họ tận dụng tối đa lợi thế của dữ liệu trong việc ra quyết định, cải thiện hiệu quả và nâng cao khả năng cạnh tranh.

Nghiên cứu và phát triển các công cụ và phương pháp mới để quản lý, xử lý, phân tích và trực quan hóa dữ liệu trên nền tảng Data Lakehouse, với sự kết hợp hài hòa giữa Data Warehouse và Data Lake, đảm bảo tính linh hoạt, an toàn và hiệu năng cao.

Khai thác và ứng dụng các kỹ thuật Machine Learning và Deep Learning trên Data Lakehouse để giải quyết các bài toán phức tạp và thách thức trong lĩnh vực nhận diện hình ảnh, xử lý ngôn ngữ tự nhiên, dự báo thị trường, phát hiện gian lận, khuyến nghị sản phẩm và dịch vụ, v.v.

Triển khai tự động hóa quy trình ML với Mlflow.

TÀI LIỆU THAM KHẢO

[1] Atlan, 19-9-2023, *What Is Modern Data Stack: History, Components, Platforms, and the Future*.

<https://atlan.com/modern-data-stack-101/>

[2] Atlan, 28-7-2023, *What is a Data Platform? Understanding its Components, Tools, and Evolution*.

<https://atlan.com/what-is-a-data-platform/#traditional-data-platform-vs-modern-data-platform-their-evolution-and-what-makes-them-different>

[3] Dremio, *Data Lakehouse Architecture*.

<https://www.dremio.com/topics/data-lakehouse/architecture/#:~:text=Introduction%20to%20Data%20Lakehouse%20Architecture&text=The%20platform%20consists%20of%20multiple,%2C%20flexibility%2C%20and%20economic%20efficiency>

[4] Databricks, *Medallion Architecture*.

<https://www.databricks.com/glossary/medallion-architecture>

[5] Developer's HOME, *Delta Lake: An Introduction to a High-Performance Data Management System*.

<https://developershome.blog/2023/05/01/delta-lake-an-introduction-to-a-high-performance-data-management-system/>

[6] Viblo, *Docker là gì ? Kiến thức cơ bản về Docker*

<https://viblo.asia/p/docker-la-gi-kien-thuc-co-ban-ve-docker-maGK7qeelj2>

[7] AI design, *Giới thiệu về Metabase – công cụ hỗ trợ phân tích cơ sở dữ liệu*

<https://aithietke.com/gioi-thieu-ve-metabase-cong-cu-ho-tro-phan-tich-co-so-du-lieu/>

[8] Geeksforgeeks, *Dimensional Data Modelling*

<https://www.geeksforgeeks.org/dimensional-data-modeling/>

[9] Michael Armbrust, *Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics*, *Conference on Innovative Data Systems Research (CIDR)*, 2021.

https://www.cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf

- [10] Medium, 30-6-2021, *Thinking Data: The Modern Data Stack* Piyush Kharbanda Vertex Ventures
<https://medium0.com/vertexventures/thinking-data-the-modern-data-stack-d7d59e81e8c6>
- [11] Michael Armbrust, *Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores*
https://people.eecs.berkeley.edu/~matei/papers/2020/vldb_delta_lake.pdf
- [12] Victor Chua, *Data Lakehouses with Databricks — A Modern Approach to Data Management*
<https://medium0.com/@chua.vctr/data-lakehouses-with-databricks-a-modern-approach-to-data-management-eca54e4a624e>
- [13] lelouvincx's blog, 16-4-2023, *Goodreads ETL pipelines*
https://lelouvincx.github.io/projects/fde2-goodreads-elt-pipeline/?fbclid=IwAR0yXQY32KxInOC4Pf00t0nk6YQs-jDn5OWq89X0RetlrGs_JzcZJHnD03M