

**TẬP ĐOÀN CÔNG NGHIỆP – VIỄN THÔNG QUÂN ĐỘI**

**BÁO CÁO MINI-PROJECT**

Chủ đề: Xu hướng ứng dụng kiến trúc Data Lakehouse trong việc xây dựng hệ thống kho dữ liệu doanh nghiệp

`

Nguyễn Duy Hưng

[duyhungnguyen2002@gmail.com](mailto:duyhungnguyen2002@gmail.com)

**Chương trình Viettel Digital Talent 2023**

**Lĩnh vực: Software & Data Engineering**

Mentor: Phạm Huy Hoàng

Đơn vị: Ban Công nghệ thông tin

Hà Nội, 06/2023

Lời mở đầu

Trước khi đi vào bài báo cáo, em xin gửi lời cảm ơn đến Ban tổ chức chương trình Viettel Digital Talent 2023 đã tạo ra một môi trường thực tập chuyên nghiệp cho các bạn sinh viên chúng em. Qua chương trình này, em đã có cơ hội để được làm việc, kết nối với các anh chị và các bạn trong ngành vô cùng tài năng, từ đó em được học hỏi, tiếp thu nhiều kiến thức chuyên môn phục vụ cho con đường phát triển sự nghiệp của mình.

Em xin gửi lời cảm ơn chân thành đến anh Phạm Huy Hoàng – mentor cho em trong đề tài lần này. Thông qua đây, em có cơ hội được tiếp xúc với một chủ đề mới mẻ và nhiều tiềm năng trong lĩnh vực Data Engineering em đang theo đuổi, đó là chủ đề về kiến trúc Lakehouse.

Tóm tắt nội dung và đóng góp

**Chủ đề thực hiện nghiên cứu**: Nội dung của bài báo cáo này là tìm hiểu về xu hướng ứng dụng kiến trúc Data Lakehouse trong việc xây dựng hệ thống Kho dữ liệu doanh nghiệp

**Phương pháp thực hiện**: Tìm hiểu thông tin trên Internet, kết hợp tham khảo một số quyển sách (thông tin ở phần Tham khảo)

**Các kết quả đạt được trong mini-project**: Tìm hiểu về tổng quan kiến trúc của Data Lakehouse; các tính năng nổi bật và các công nghệ được sử dụng trong kiến trúc Data Lakehouse; tìm hiểu một số case-study đã ứng dụng thành công kiến trúc Data Lakehouse trong nghiệp vụ.

**Đóng góp của cá nhân trong kết quả chung**: Tìm hiểu về lịch sử hình thành của kiến trúc Data Lakehouse; tham gia tìm hiểu kiến trúc của Data Lakehouse, xác định sự giống và khác nhau giữa các nền tảng Data Lakehouse của các nhà cung cấp; tìm hiểu các công nghệ được sử dụng trong hệ thống sử dụng Data Lakehouse.

**Định hướng phát triển mở rộng**: Có thể xây dựng một hệ thống mô phỏng kiến trúc Data Lakehouse có thể trang bị đầy đủ các tính năng nổi bật của chúng; áp dụng những đặc điểm này vào việc giải quyết các vấn đề thực tại của doanh nghiệp.

**Các kỹ năng, kiến thức đặt được sau khi thực hiện**: Khả năng sử dụng các công cụ tìm kiếm; kỹ năng chọn lọc thông tin; kiến thức tổng quan về mô hình Data Lakehouse.

MỤC LỤC

[CHƯƠNG 1: Lịch sử ra đời của kiến trúc Data Lakehouse 1](#_Toc137117575)

[CHƯƠNG 2: Tổng quan về kiến trúc Data Lakehouse 3](#_Toc137117576)

[2.1 Tầng thu thập dữ liệu (Data ingestion layer) 3](#_Toc137117577)

[2.2 Tầng Data Lake (Data Lake layer) 4](#_Toc137117578)

[2.3 Tầng siêu dữ liệu (Metadata layer) 6](#_Toc137117579)

[2.4 Tầng xử lý dữ liệu (Data processing layer) 6](#_Toc137117580)

[2.5 Tầng phục vụ dữ liệu (Data serving layer) 8](#_Toc137117581)

[2.6 Tầng phân tích và trực quan hoá dữ liệu (Data analytics and visualization layer) 9](#_Toc137117582)

[2.7 Tầng quản trị dữ liệu (Data governance layer) 10](#_Toc137117583)

[2.8 Tầng bảo mật dữ liệu (Data security layer) 11](#_Toc137117584)

[CHƯƠNG 3: Các tính năng nổi bật của Data Lakehouse 13](#_Toc137117585)

[3.1 Hỗ trợ giao dịch ACID (ACID transaction support) 13](#_Toc137117586)

[3.2 Time Travel 14](#_Toc137117587)

[3.3 Thực thi, cải tiến lược đồ và quản trị dữ liệu (Schema Enforcement, Schema Evolution and data gorvenance) 15](#_Toc137117588)

[3.4 Hỗ trợ các dữ liệu phi cấu trúc và theo luồng (Unstructured and streaming data support) 16](#_Toc137117589)

[3.5 Hỗ trợ định dạng mở (Open formats support) 16](#_Toc137117590)

[3.6 Lưu trữ và tính toán tách biệt (Decoupled storage and compute) 17](#_Toc137117591)

[3.7 Hỗ trợ BI (Business Intelligence support) 18](#_Toc137117592)

[CHƯƠNG 4: So sánh Data Lakehouse với Data Lake và Data Warehouse 20](#_Toc137117593)

[CHƯƠNG 5: Các công nghệ dùng trong một hệ thống dữ liệu sử dụng kiến trúc Data Lakehouse 22](#_Toc137117594)

[5.1 Tầng thu thập dữ liệu 23](#_Toc137117595)

[5.2 Tầng lưu trữ dữ liệu 24](#_Toc137117596)

[5.3 Tầng siêu dữ liệu 25](#_Toc137117597)

[5.4 Tầng xử lý dữ liệu 26](#_Toc137117598)

[5.5 Tầng phân tích và trực quan hoá dữ liệu 26](#_Toc137117599)

[5.6 Tầng quản trị dữ liệu 27](#_Toc137117600)

[5.7 Tầng bảo mật dữ liệu 28](#_Toc137117601)

[CHƯƠNG 6: Các case-study tiêu biểu áp dụng thành công kiến trúc Data Lakehouse 29](#_Toc137117602)

[6.1 GRAB và câu chuyện tìm ra mô hình hoàn hảo nhất cho dữ liệu của mình 29](#_Toc137117603)

[6.2 AT&T 30](#_Toc137117604)

[CHƯƠNG 7: Demo một hệ thống Data Lakehouse cơ bản 32](#_Toc137117605)

[7.1 Hỗ trợ giao dịch ACID 33](#_Toc137117606)

[7.2 Thực thi và cải tiến lược đồ 35](#_Toc137117607)

[7.2.1 Thực thi lược đồ (Schema Enforcement) 35](#_Toc137117608)

[7.2.2 Cải tiến lược đồ (Schema Evolution) 36](#_Toc137117609)

[7.3 Audit History 37](#_Toc137117610)

[7.4 Time Travel 38](#_Toc137117611)

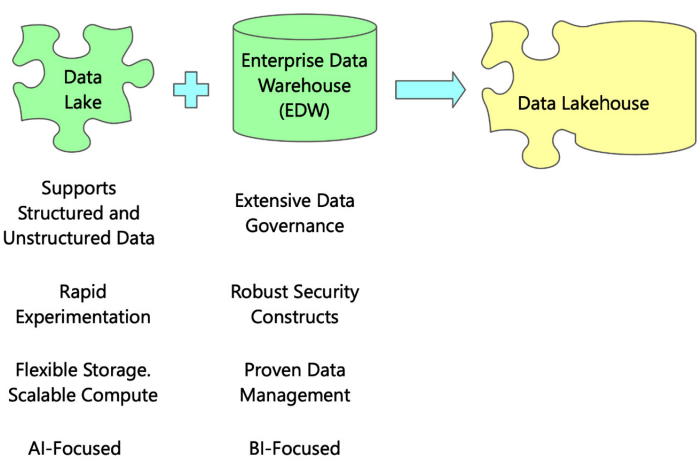
[CHƯƠNG 8: Kết luận 40](#_Toc137117612)

[THAM KHẢO 41](#_Toc137117613)

# Lịch sử ra đời của kiến trúc Data Lakehouse

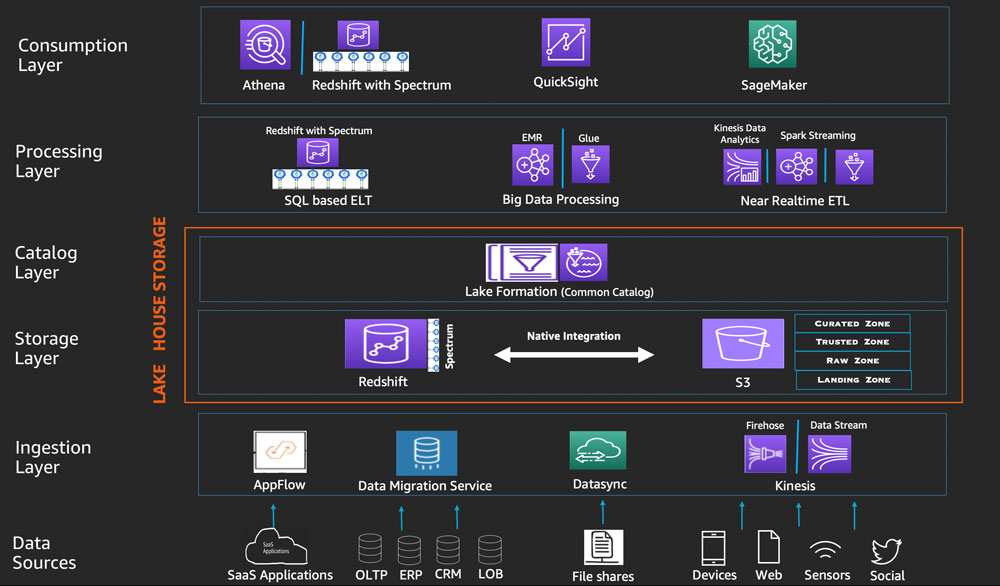
Chúng ta đang sống trong một thế giới mà hầu hết các quyết định được đưa ra trong mọi công việc đều là những quyết định dựa trên dữ liệu. “Dữ liệu là một loại nhiên liệu mới”, Clive Humbly – nhà toán học người Anh, đã phát biểu như vậy để nói về tầm quan trọng của dữ liệu trong cuộc sống hiện nay. Dữ liệu có thể coi như là “huyết mạch” của các doanh nghiệp, và cách họ sử dụng chúng ra sao sẽ quyết định ưu thế cạnh tranh trên thị trường. Do đó, quản lý dữ liệu trở thành một điều tối quan trọng trong thời đại chuyển đổi kỹ thuật số hiện nay. Ngày càng có nhiều doanh nghiệp, tổ chức áp dụng các chương trình chuyển đổi số và dữ liệu chính là thành phần cốt lõi của quá trình này.

Data Warehouse và Data Lake đều là các kiến trúc kho dữ liệu đã phù hợp ở thời điểm mà chúng được tạo ra. Cả hai kiến trúc này đều có những ưu và nhược điểm riêng (việc so sánh chi tiết được đề cập trong phần 3), và để đáp ứng được nhu cầu ngày càng gia tăng cả về chất và lượng trong việc quản lý dữ liệu, một mô hình mới có đầy đủ những điểm mạnh của cả hai kiến trúc cũ là điều được hướng đến. Đó là lý do mà Data Lakehouse ra đời, khi nó được xây dựng để bao hàm những ưu điểm của cả hai kiến trúc Data Warehouse và Data Lake, đồng thời hạn chế ít nhất những yếu điểm của chúng.



Hình 1.1: Minh hoạ về mô hình Data Lakehouse

Về các mốc thời gian, thuật ngữ “lakehouse” lần đầu được nhắc đến vào cuối năm 2017 khi **Snowflake** sử dụng thuật ngữ này trong buổi giới thiệu sản phẩm của mình. Tuy nhiên thời điểm đó tập đoàn đến từ nước Mỹ lại không sử dụng thuật ngữ “lakehouse” cho chiến dịch marketing sản phẩm của họ mà thay vào đó là “data cloud”. Đến tháng 4 năm 2019 **Databricks** mang đến giải pháp ***Delta Lake*** với các tính năng của một Data Warehouse được tích hợp trong mô hình Data Lake. Đến cuối năm đó, **Amazon Web Services** cho ra mắt sản phẩm sử dụng “lakehouse” là thuật ngữ chính mô tả dịch vụ với tên gọi ***Redshift Spectrum***. Bên cạnh đó còn có **Google, Microsoft, IBM**… cũng lần lượt giới thiệu đến công chúng các công cụ được xây dựng từ nền tảng kiến trúc Data Lakehouse để phục vụ nhu cầu của người dùng.



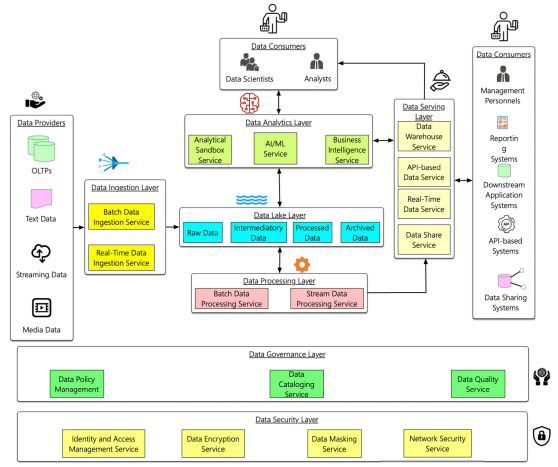
Hình 1.2: Kiến trúc Data Lakehouse của AWS

Như vậy, có thể thấy Data Lakehouse là một kiến trúc còn khá mới, lần đầu được nhắc đến cách đây 5 năm (tính đến thời điểm bài báo cáo này được thực hiện là tháng 5/2023), và mới dần được sử dụng phổ biến trong các tổ chức, doanh nghiệp khoảng hai năm trở lại đây. Data Lakehouse có rất nhiều ưu điểm và còn nhiều tiềm năng để các nhà nghiên cứu và phát triển có thể khám phá. Bên cạnh đó, một kiến trúc mới cũng sẽ có những thách thức đối với các công ty sử dụng chúng và cần áp dụng những giải pháp công nghệ để có thể đem lại kết quả tối ưu. Chúng ta sẽ tìm hiểu kĩ hơn về kiến trúc này ở trong các phần tiếp theo.

# Tổng quan về kiến trúc Data Lakehouse

Một kiến trúc được xây dựng và cân nhắc kĩ lưỡng luôn là nền tảng cho bất kì một hệ thống công nghệ thông tin nào và Data Lakehouse cũng không là ngoại lệ. Trong phần 2 này, chúng ta sẽ tìm hiểu về kiến trúc logic của Data Lakehouse và thảo luận về các thành phần quan trọng của kiến trúc này.

Một kiến trúc logic sẽ cần tập trung vào các thành phần có thể tích hợp với nhau để đáp ứng những yêu cầu chức năng cụ thể cũng như các yêu cầu phi chức năng khác. Với Data Lakehouse thì mỗi nhà cung cấp khác nhau lại thiết kế riêng hệ thống cho sản phẩm của họ, vì thế, có thể nói khó có một khung kiến trúc chung cho Data Lakehouse. Ở trong khuôn khổ bài báo cáo này, ta sẽ tìm hiểu mô hình kiến trúc Data Lakehouse với 8 tầng, trong đó có 7 tầng mô tả như hình dưới đây và một tầng có thể coi là “sự khác biệt” đối với các hệ thống khác đó là tầng siêu dữ liệu.

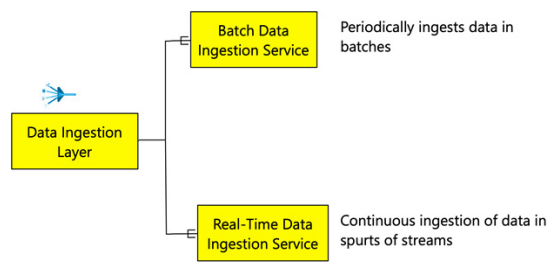


Hình 2.1: Mô hình kiến trúc của Data Lakehouse theo ý tưởng của Pradeep Menon

## Tầng thu thập dữ liệu (Data ingestion layer)

Tầng thu thập dữ liệu được coi như là điểm trung gian giữa các nhà cung cấp dữ liệu bên ngoài và Data Lakehouse. Tầng này chịu trách nhiệm thu thập và đưa dữ liệu lên tầng lưu trữ dữ liệu. Chúng cung cấp khả năng kết nối tới các nguồn dữ liệu cả bên trong lẫn bên ngoài thông qua các giao thức. Sẽ có hai phương thức thu thập dữ liệu, đó là:

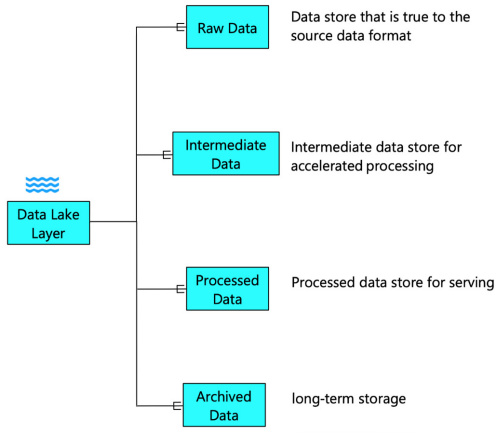
* Batch Data Ingestion: Phương thức thu thập dữ liệu thể hiện rằng dữ liệu sẽ được nhập định kì vào Data Lakehouse. Tần suất thu thập có thể là vài phút cho đến vài ngày, thời gian trên sẽ được lựa chọn phụ thuộc vào nhiều yếu tố như các yêu cầu phi chức năng, khả năng tạo dữ liệu của nguồn cung cấp hay là khả năng đưa dữ liệu và cho phép lấy dữ liệu về của nguồn. Một lưu ý cần có trong khi nhập dữ liệu theo lô là tính khả dụng của hệ thống nguồn để nhập dữ liệu và kích thước của dữ liệu lô được nhập. Cả hai yếu tố này sẽ có tác động đến cách dữ liệu được nhập vào Data Lakehouse.
* Real-Time Data Ingestion: Dịch vụ cho phép dữ liệu được kéo lên Data Lakehouse ngay khi nó vừa được tạo ra. Dữ liệu luồng trực tuyến là một luồng dữ liệu liên tục, từ đó dữ liệu được quan tâm sẽ được định dạng và kéo lên Data Lakehouse để lưu trữ hay xử lý thời gian thực. Quá trình thu thập theo thời gian thực thường bao gồm một dịch vụ hàng đợi như Kafka, dịch vụ này sẽ cho phép nhóm các luồng thời gian thực và lưu trữ tạm thời dưới dạng hàng đợi để được nhập vào. Các dịch vụ streaming cũng được sử dụng để liên tục nắm bắt các thay đổi dữ liệu trong cơ sở dữ liệu thông qua tính năng “change data capture” (CDC). Các cân nhắc liên quan đến thông lượng của dữ liệu truyền phát và các yêu cầu về độ trễ trở nên quan trọng khi thu thập dữ liệu real-time.



Hình 2.2: Các phương thức thu thập của tầng thu thập dữ liệu

## Tầng Data Lake (Data Lake layer)

Một khi tầng thu thập dữ liệu đã “lấy” data về, chúng sẽ cần được đưa vào các kho lưu trữ, và ở đây các phép chuyển đổi – transformations sẽ được thực hiện theo yêu cầu nghiệp vụ của doanh nghiệp. Ở một số kiến trúc Data Lakehouse hiện tại, tầng này còn có tên gọi là tầng lưu trữ dữ liệu (Storage layer). Tuy nhiên như ta đã đề cập thì Data Lakehouse là sự kết hợp của cả hai kiểu kho dữ liệu truyền thống nên ta sẽ gọi tầng này là tầng Data Lake, bởi đây là tầng thể hiện rõ nhất đặc điểm nổi bật của Data Lake. Ta có thể quan sát trực quan hơn về tầng lưu trữ Data Lake trong hình sau:



Hình 2.3: Một số các kho dữ liệu trong tầng Data Lake

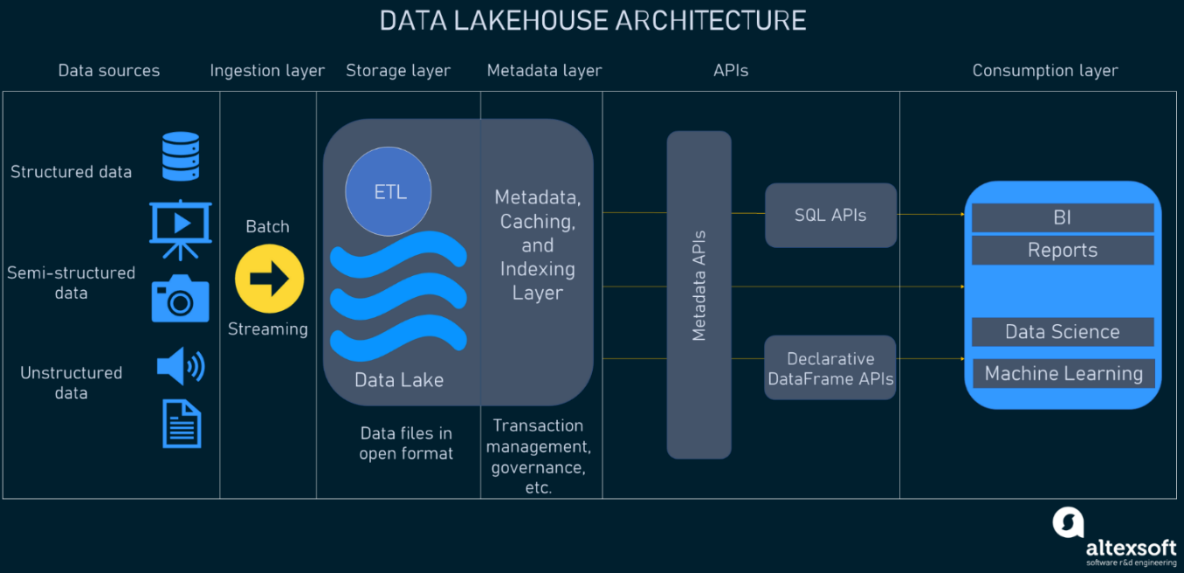
Tầng Data Lake sẽ lưu trữ bốn dạng chính của dữ liệu như hình bên, cụ thể đó là:

* Raw Data: Đây là nơi lưu trữ dữ liệu thô từ các nguồn dữ liệu. Chúng sẽ giữ nguyên bản format, cấu trúc và nội dung của dữ liệu sát nhất với dữ liệu từ các nhà cung cấp.
* Intermediate Data: Khi dữ liệu đi qua kho dữ liệu và được chuyển đổi, các bộ dữ liệu trung gian sẽ được tạo. Các bộ dữ liệu trung gian này có thể là tạm thời hoặc liên tục. Chúng có thể được lưu trữ trong tầng Data Lake và có thể tăng tốc quá trình xử lý dữ liệu. Dữ liệu trung gian cũng làm cho quy trình xử lý dữ liệu không bị khởi động lại hoàn toàn.
* Processed Data: Một khi mà dữ liệu đã được chuyển đổi, ta có thể lưu trữ các bộ dữ liệu này trong Data Lake và chúng sẽ được dùng cho mục đích phân tích hay nghiên cứu. Chúng cũng cho phép các nhân viên khoa học và phân tích dữ liệu tiến hành công việc của mình trong trường hợp không muốn trả chi phí cho tầng phục vụ.
* Archived Data: Dữ liệu được dùng cho mục đích phân tích sẽ được lưu trữ trong một công nghệ lưu trữ tốt hơn, từ đó có thể đảm bảo được thông lượng và khả năng truy cập. Tất nhiên là không phải dữ liệu nào cũng sẽ được dùng để phân tích. Vậy nên những dữ liệu không dùng cho mục đích phân tích, nếu vẫn được yêu cầu phải lưu trữ thì sẽ được lưu tại một kho dữ liệu có chi phí rẻ hơn, và chúng được gọi là dữ liệu được lưu trữ.

## Tầng siêu dữ liệu (Metadata layer)

Đây là tầng thứ ba trong kiến trúc Data Lakehouse của trang web **altexsoft** đưa ra (đường liên kết bài báo ở cuối báo cáo), và như đã đề cập thì đây chính là điều làm nên sự khác biệt của kiến trúc này. Tầng siêu dữ liệu cung cấp các siêu dữ liệu (dữ liệu cung cấp thông tin về các phần dữ liệu khác) cho tất cả các đối tượng trong Data Lakehouse và mang đến cho người dùng giao diện để triển khai các tính năng quản lý như:

* Các giao dịch ACID để đảm bảo rằng các giao dịch đồng thời nhìn thấy một phiên bản nhất quán của cơ sở dữ liệu;
* Ghi vào các tệp bộ đệm từ kho lưu trữ đối tượng đám mây;
* Lập chỉ mục để thêm các chỉ mục cấu trúc dữ liệu, từ đó thực hiện truy vấn nhanh hơn;
* Nhân bản để tạo bản sao của các đối tượng dữ liệu;
* Đánh dấu dữ liệu để lưu các phiên bản cụ thể của dữ liệu, v.v.



Hình 2.4: Tầng siêu dữ liệu trong mô hình kiến trúc Data Lakehouse (nguồn: Altexsoft)

Như đã đề cập trước đó, tầng siêu dữ liệu cũng cho phép áp dụng các kiến trúc lược đồ trong Data Warehouse chẳng hạn như lược đồ hình sao/bông tuyết, tiến hành triển khai quản lý lược đồ cũng như cung cấp chức năng kiểm tra và quản trị dữ liệu trực tiếp như trên Data Lake, cải thiện chất lượng của toàn bộ các luồng dữ liệu.

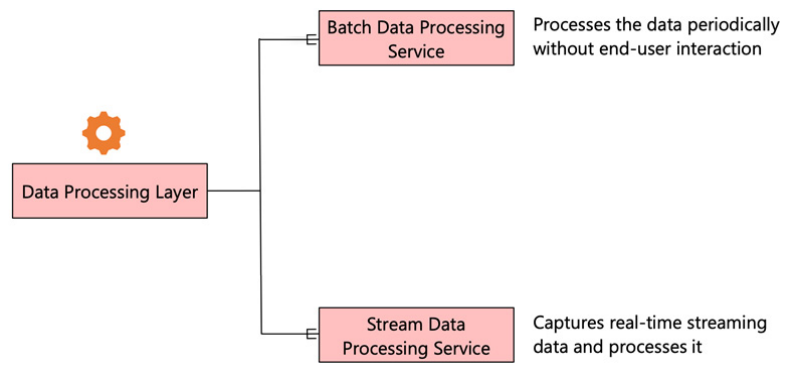
## Tầng xử lý dữ liệu (Data processing layer)

Các thành phần trong tầng xử lý dữ liệu của Kiến trúc Data Lakehouse chịu trách nhiệm chuyển đổi dữ liệu thành trạng thái có thể sử dụng được thông qua quá trình xác thực, làm sạch, chuẩn hóa, chuyển đổi và làm dày dữ liệu.

Tầng xử lý cung cấp thời gian thực hiện nhanh nhất để đưa lên bộ phận khác, bằng cách cung cấp các thành phần được xây dựng có mục đích phù hợp với các đặc điểm của tập dữ liệu (kích thước, định dạng, lược đồ, tốc độ), tác vụ cần xử lý và các bộ công cụ có sẵn như Apache Spark. Tầng xử lý có thể mở rộng quy mô một cách hiệu quả về mặt chi phí để xử lý khối lượng dữ liệu lớn và cung cấp các thành phần để hỗ trợ tập dữ liệu dạng *lược đồ-khi-ghi, lược đồ-khi-đọc*, *tập dữ liệu được phân vùng,*... Tầng xử lý có thể truy cập vào khu vực tập trung dữ liệu của Data Lakehouse, từ đó truy cập tất cả dữ liệu và siêu dữ liệu trong Data Lakehouse. Điều này có những lợi ích sau:

* Hạn chế dư thừa dữ liệu (data redundancies), di chuyển dữ liệu không cần thiết và sao chép mã ETL có thể xảy ra khi xử lý Data Lake và kho dữ liệu riêng biệt
* Giảm thời gian dữ liệu cần xử lý để đưa lên cho bộ phận khác

Tầng xử lý dữ liệu sẽ được mô tả trong hình dưới:

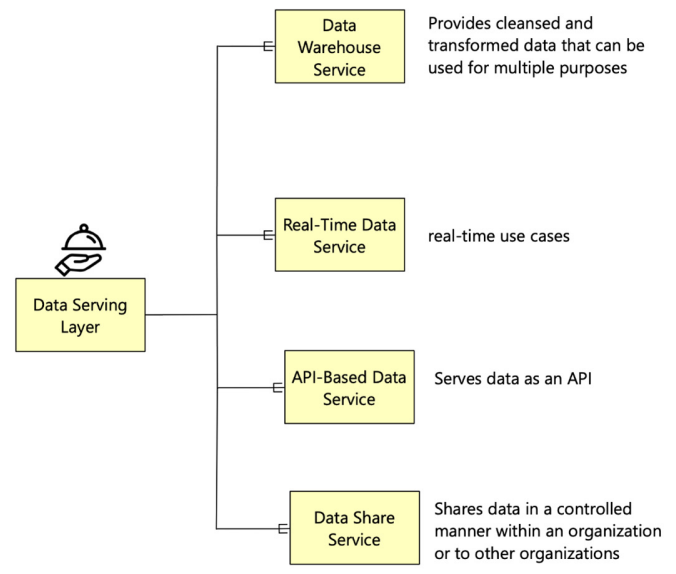


Hình 2.5: Các dịch vụ xử lý dữ liệu

Theo hình trên, ta có thể thấy hai dạng dịch vụ xử lý dữ liệu chính:

* Batch Data Processing Service: Dịch vụ này sẽ xử lý dữ liệu định kì mà không có sự tương tác của người dùng cuối. Dữ liệu đầu tiên sẽ được đưa vào vùng dữ liệu thô (raw data zone), sau đó dịch vụ sẽ tiến hành xử lý và chuyển đổi chúng theo yêu cầu. Điều quan trọng là dịch vụ này có thể mở rộng hay thu nhỏ quy mô xử lý tuỳ theo yêu cầu của doanh nghiệp
* Stream Data Processing Service: Dịch vụ này sẽ xử lý dữ liệu mà không cần chúng phải được đưa vào trong vùng lưu trữ. Tất cả quy trình xử lý sẽ diễn ra trong bộ nhớ, và dữ liệu sẽ được xử lý gần như theo thời gian thực. Một dịch vụ xử lý dữ liệu luồng điển hình sẽ có một hàng chờ thông điệp để “bắt” các luồng dữ liệu (data stream) một cách không liên tục và xếp chúng vào hàng đợi để xử lý thêm. Khi luồng dữ liệu được nhập vào và xử lý, dữ liệu thô sẽ được gửi đến kho Data Lake để lưu trữ dưới dạng một đường dẫn, trong khi đó một đường dẫn khác thực hiện quá trình xử lý theo thời gian thực. Cuối cùng, dữ liệu được chuyển đổi sẽ được đẩy vào tầng lưu trữ dữ liệu để được lưu trữ liên tục.

## Tầng phục vụ dữ liệu (Data serving layer)



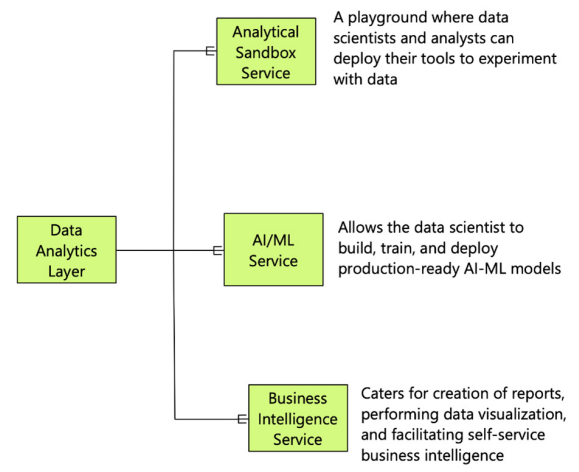
Hình 2.6: Các dạng của phục vụ dữ liệu

Một khi mà dữ liệu đã được xử lý, chúng sẽ được sử dụng để phục vụ các yêu cầu phía sau này của doanh nghiệp. Tuỳ theo yêu cầu riêng của mỗi người trong bộ phận quản lý mà những dữ liệu này có thể được xử lý và phục vụ theo yêu cầu đó. Ta sẽ giải thích dạng dịch vụ ở trong hình như sau:

* Data Warehouse Service: Dịch vụ này cung cấp nguồn dữ liệu đã được làm sạch và chuyển đổi để có thể được sử dụng cho nhiều mục đích khác nhau. Đầu tiên, nó phục vụ như một lớp báo cáo và BI. Thứ hai, nó là một nền tảng để truy vấn dữ liệu cho doanh nghiệp hoặc phân tích dữ liệu. Thứ ba, nó phục vụ như một kho để lưu trữ dữ liệu về lịch sử mà cần sự khả dụng và trực tuyến. Cuối cùng, nó cũng hoạt động như một nguồn dữ liệu được chuyển đổi cho các kho dữ liệu xuôi dòng (downstream) khác có thể đáp ứng các yêu cầu cụ thể của doanh nghiệp.
* Real-time Data Service: Dịch vụ dữ liệu thời gian thực được dùng để phục vụ hầu hết các ứng dụng downstream như các hệ thống di động, hệ thống CRM, công cụ khuyến nghị hay phát hiện gian lận. Dịch vụ này biểu hiện ở trong nhiều định dạng công nghệ khác nhau và nếu được sử dụng đúng cách thì sẽ đem lại nhiều giá trị kinh doanh to lớn cho doanh nghiệp
* API-Based Data Service: Đây là một giao diện cho phép các ứng dụng tương tác với các dịch vụ bên ngoài bằng cách sử dụng một tập các dòng lệnh. Phương pháp này có thể mở rộng để chia sẻ dữ liệu đối với các dịch vụ bên ngoài với sự bảo mật nhất định. Dữ liệu thông qua API sẽ được phục vụ dưới dạng tệp JSON, vì vậy mà cơ sở dữ liệu NoSQL được ưu tiên để sử dụng.
* Data Sharing Service: Dịch vụ này sẽ chia sẻ dữ liệu ở bất kì dạng và kích cỡ nào, từ nhiều nguồn bên trong hay kể cả bên ngoài doanh nghiệp, Dịch vụ chia sẻ dữ liệu cung cấp việc điều khiển theo yêu cầu để chia sẻ dữ liệu và cho phép các chính sách chia sẻ dữ liệu được tạo thêm. Một hệ thống chia sẻ dữ liệu sẽ sử dụng các phép gọi API để tiến hành chia sẻ.

## Tầng phân tích và trực quan hoá dữ liệu (Data analytics and visualization layer)

Tầng phân tích dữ liệu của Data Lakehouse chịu trách nhiệm cung cấp các thành phần có hiệu năng cao và khả năng mở rộng linh hoạt, sử dụng giao diện quản lý của Data Lakehouse để truy cập tất cả dữ liệu được lưu trong tầng lưu trữ và tất cả siêu dữ liệu được lưu trong tầng siêu dữ liệu. Nó đem đến sự bình đẳng trong một tổ chức doanh nghiệp khi tham gia thực hiện các phân tích, bằng cách cung cấp các thành phần để tiến hành phân tích bao gồm các truy vấn SQL, phân tích kiểu kho dữ liệu, BI dashboard và học máy.



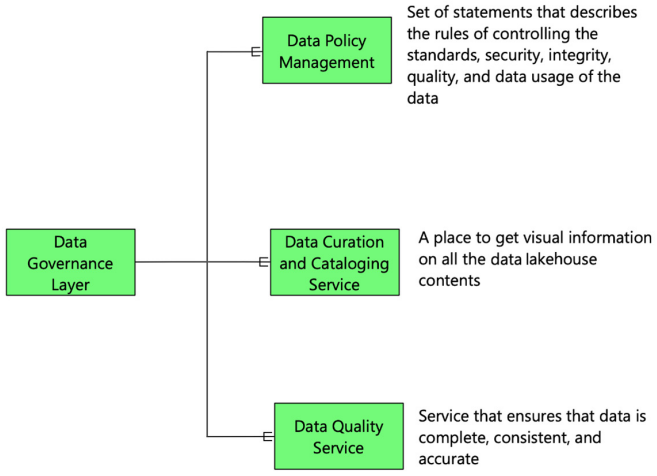
Hình 2.7: Các dịch vụ của tầng phân tích dữ liệu

Theo mô tả của hình bên, tầng phân tích dữ liệu sẽ có ba loại dịch vụ, cụ thể là:

* Analytical Sandbox Service: Đây là nơi mà các nhà khoa học và phân tích dữ liệu sẽ triển khai các công cụ (tools) cho công việc của mình. Chính vì thế chúng cung cấp đầy đủ các công cụ cho việc phân tích dựa trên SQL và phát triển các mô hình học máy.
* Artificial Intelligence and Machine Learning (AI-ML) Service: Dịch vụ này cho phép các chuyên viên khoa học dữ liệu xây dựng, huấn luyện và triển khai các mô hình AI-ML, đồng thời cung cấp các framework để duy trì và giám sát các mô hình này. Tất nhiên là, những dịch vụ này luôn có thể thay đổi phạm vi sử dụng theo nhu cầu, đồng thời có thể triển khai và vận hành mô hình tự động
* Business Intelligence (BI) Service: Các dịch vụ BI ở trong kiến trúc Data Lakehouse tương tự với mô hình kho dữ liệu doanh nghiệp. Ở đây, dịch vụ sẽ tập trung vào việc tạo ra các bảng biểu hay các mô hình trực quan hoá liên quan đến việc vận hành của doanh nghiệp ở hiện tại cũng như trong quá khứ.

## Tầng quản trị dữ liệu (Data governance layer)

Dữ liệu trong Data Lakehouse cũng sẽ cần phải được quản trị một cách hợp lý, bởi nếu dữ liệu không được đảm bảo chất lượng thì kho dữ liệu của chúng ta sẽ trở thành Data Swamp. Data Swamp (tiếng Việt: đầm lầy dữ liệu) là Data Lake nhưng lại lưu trữ dữ liệu không được làm sạch, chứa nhiều lỗi và bị trùng lặp thông tin. Những dữ liệu này này thường thiếu thông tin về siêu dữ liệu nên ít có giá trị sử dụng và tất nhiên, đó là điều mà không doanh nghiệp nào mong muốn. Minh hoạ về tầng quản trị dữ liệu như sau:

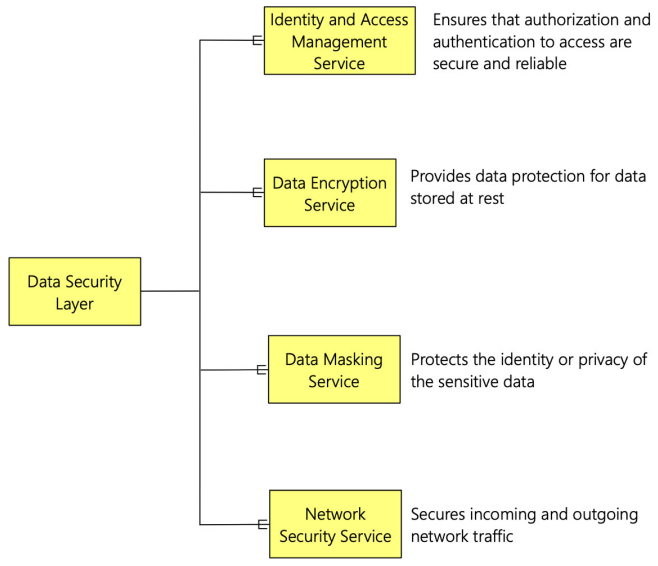


Hình 2.8: Các dịch vụ của tầng quản trị dữ liệu

Tìm hiểu chi tiết về ba dịch vụ của tầng quản trị dữ liệu, cụ thể đó là:

* Data Policy Management: Đây là một danh sách các chính sách hay điều khoản mô tả các quy định về việc quản lý và sử dụng các tiêu chuẩn, bảo mật, tính toàn vẹn và cách sử dụng những dữ liệu ở trong Data Lakehouse
* Data Cataloging and Curation Service: Là một quy trình tổ chức các loại dữ liệu mà có thể dễ dàng định dạng. Dịch vụ này đảm bảo rằng toàn bộ dữ liệu từ các hệ thống nguồn, dữ liệu trong Data Lake và Data Warehouse, các luồng xử lý dữ liệu và dữ liệu đầu ra trích xuất từ Data Lakehouse sẽ được phân loại thành các danh mục hợp lý.
* Data Quality Service: Bất kì dữ liệu được lưu trữ hay thu thập ở trong Data Lakehouse sẽ phải đạt đủ chất lượng theo yêu cầu để xác định độ tin cậy và khả dụng của dữ liệu. Dịch vụ này sẽ đảm bảo cho dữ liệu của chúng ta có là được toàn vẹn, nhất quán và có tính chính xác cao.

## Tầng bảo mật dữ liệu (Data security layer)



Hình 2.9: Các dịch vụ của tầng bảo mật dữ liệu

Việc bảo mật dữ liệu là rất quan trọng và luôn được các doanh nghiệp quan tâm, các dịch vụ của tầng này được minh hoạ trong hình bên trên, bao gồm:

* Identity and Access Management (IAM) Service: Dịch vụ này hoạt động giống như một cái cổng cho việc truy cập vào Data Lakehouse. IAM đảm bảo rằng cho việc uỷ quyền và xác minh để truy cập vào Data Lakehouse được bảo mật và tin cậy. Dịch vụ cung cấp khả năng bảo vệ chống lại các nỗ lực đăng nhập độc hại và bảo vệ thông tin xác thực bằng các biện pháp kiểm soát truy cập dựa trên rủi ro, công cụ bảo vệ danh tính và các tùy chọn xác thực mạnh mẽ—mà không làm gián đoạn năng suất.
* Data Encryption Service: Mã hoá dữ liệu là một phương pháp bảo mật mà thông tin được mã hoá và chỉ có thể được truy cập hoặc giải mã bởi người dùng có khoá giải mã. Phương pháp này rất cần thiết khi mà dữ liệu được lưu trữ trên đám mây. Có nhiều giải thuật khác nhau sẽ được dùng để mã hoá dữ liệu, giúp bảo vệ các dữ liệu nhạy cảm khỏi các cuộc tấn công an ninh mạng.
* Data Masking Service: Nhiều tập hợp con dữ liệu cần được che dấu để bảo vệ danh tính hoặc quyền riêng tư của cá nhân. Loại dữ liệu này bao gồm email, số chứng minh nhân dân, số thẻ tín dụng, v.v. Mặt nạ dữ liệu là một cách để tạo phiên bản dữ liệu khó hiểu nhưng vẫn có thể đọc được. Mục tiêu của nó là để bảo vệ dữ liệu nhạy cảm bằng cách cung cấp chức năng thay thế khi không cần dữ liệu thực tế.
* Network Security Service: Dữ liệu trong Data Lakehouse luôn phải được bảo mật, đảm bảo rằng không có bất kì sự truy cập trái phép nào xảy ra. Kể cả với những luồng dữ liệu giữa các mạng lưới bên ngoài cũng cần phải được đảm bảo tính an toàn và bảo mật, và dịch vụ này sẽ cung cấp các chứng năng như vậy.

Như đã đề cập trong phần giới thiệu ở đầu phần 2, mỗi nhà cung cấp lại có sự thay đổi một chút trong kiến trúc Data Lakehouse của mình. Đối với kiến trúc Data Lakehouse của AWS, tầng phân tích và trực quan hoá dữ liệu được họ sử dụng với tên gọi là tầng tiêu thụ dữ liệu (Consumption layer). Các thành phần trong tầng này hỗ trợ các công việc sau:

* Viết các truy vấn, thực hiện các phép phân tích và công việc học máy để truy cập, kết hợp dữ liệu từ các lược đồ theo chiều (dimensional schemas) của Data Warehouse truyền thống với các bảng lưu trữ của Data Lake (yêu cầu schema-on-read)
* Xử lý các tập dữ liệu lưu trữ trên Data Lake bằng việc sử dụng nhiều định tệp mở khác nhau như **Avro**, **Parquet** hay **ORC**
* Tối ưu hoá hiệu năng và chi phí thông qua việc cắt tỉa phân vùng (partition purning) khi tiến hành đọc các tập dữ liệu phân vùng đang được lưu trữ trong Data Lake.

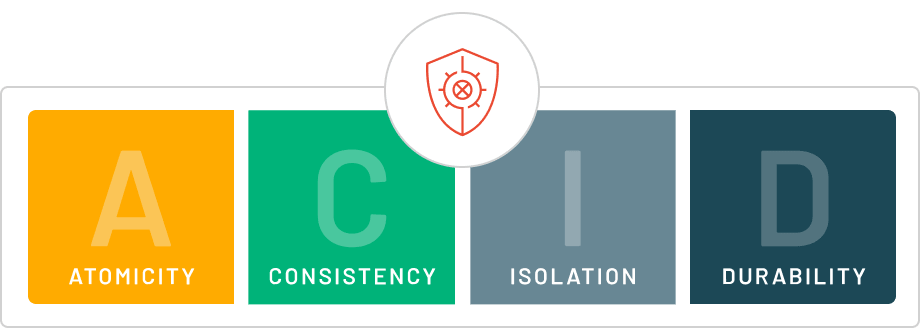
Như vậy, ta đã vừa tìm hiểu tổng quan về cấu trúc và các vai trò của các tầng tham gia trong cấu trúc của Data Lakehouse. Ở một số sản phẩm thương mại trên thị trường, họ thường gộp tầng thu thập với tầng xử lý, tầng lưu trữ với tầng phục vụ để tạo thành kiến trúc 5 tầng. Mục đích của kiến trúc này đó là sẽ giúp mọi người hiểu chi tiết hơn về từng công đoạn sẽ diễn ra trong quy trình làm việc của một Data Lakehouse.

# Các tính năng nổi bật của Data Lakehouse

Như đã đề cập qua trong phần một, Data Lakehouse là sự kết hợp giữa những chức năng tiêu biểu nhất của Data Warehouse và Data Lake. Chúng tận dụng các cấu trúc dữ liệu từ Data Warehouse, kết hợp với sự linh hoạt và chi phí lưu trữ thấp của Data Lake để giúp cho doanh nghiệp có thể lưu trữ và truy cập vào khối dữ liệu lớn một cách nhanh chóng và hiệu quả hơn, đồng thời giúp cho họ giảm thiểu các rủi rỏ liên quan đến chất lượng của dữ liệu. Trong báo cáo này, ta sẽ đi tìm hiểu chi tiết các tính năng nổi bật của Data Lakehouse, những khía cạnh mà kiến trúc này giúp giải quyết vấn đề của người dùng. Các tính năng này được khuyến nghị và đề cập đến trong các sản phẩm sử dụng kiến trúc Data Lakehouse bởi Databricks, IBM và AWS.

## Hỗ trợ giao dịch ACID (ACID transaction support)

Như chúng ta đã biết, ACID là viết tắt của atomicity (tính nguyên tử), consistency (tính nhất quán), isolation (tính cô lập) và durability (tính bền vững), là các đặc điểm quan trọng khi định nghĩa một giao dịch và đảm bảo tính nhất quán và độ tin cậy của dữ liệu. Các giao dịch này thường sẽ chỉ khả dụng ở trong Data Warehouse. tuy nhiên với sự xuất hiện của lakehouse thì chúng cho phép ứng dụng vào trong cả Data Lake. Điều này giúp giải quyết mối lo về chất lượng của dữ liệu sau này, khi có nhiều luồng dữ liệu tham gia vào việc đọc và ghi dữ liệu đồng thời (có sự tương tranh).



Hình 3.1: Tiêu chuẩn ACID

Lấy ví dụ về Delta Lake của Databricks, họ đã đem ACID transactions vào trong Data Lake. Họ cho phép người dùng xem những góc nhìn nhất quán về dữ liệu ngay cả khi dữ liệu mới đang được ghi vào bảng theo thời gian thực, bởi từng hành động này là một giao dịch cô lập và được ghi lại trong một bản lưu thông tin giao dịch. Với việc cài đặt các giao dịch ACID này thì Delta Lake giải quyết một cách hiệu quả những khuyết điểm đã được nói đến trước đây của kiến trúc Lambda: Sự phức tạp, góc nhìn thiếu chính xác về dữ liệu, và khi mà các luồng Lambda bị gián đoạn thì sẽ cần xử lý lại từ đầu. Người dùng có thể thực hiện nhiều giao dịch đồng thời trên dữ liệu của họ và trong trường hợp xảy ra lỗi với nguồn dữ liệu hoặc luồng, Delta Lake sẽ hủy thực hiện giao dịch để đảm bảo dữ liệu được giữ sạch và nguyên vẹn.

Cái hay của các giao dịch ACID là người dùng có thể tin tưởng vào dữ liệu được lưu trữ trong Data Lakehouse. Một chuyên viên phân tích dữ liệu sử dụng các bảng trong Data Lakehouse để thực hiện quá trình ETL trên dữ liệu của mình nhằm chuẩn bị cho dashboard có thể tin tưởng vào thực tế là, các KPI mà họ đang thấy đại diện cho trạng thái thực của dữ liệu. Một kỹ sư học máy sử dụng bảng trong lakehouse để thực hiện các kỹ thuật tính năng có thể tin tưởng 100% rằng tất cả các phép biến đổi và tổng hợp sẽ được thực hiện chính xác như dự định hoặc là hoàn toàn không thực hiện (trong trường hợp đó, thông báo sẽ được gửi đến cho người sử dụng).

## Time Travel

Time travel là một tính năng cho phép người dùng có thể truy cập và truy vấn các phiên bản dữ liệu cũ, cho phép phân tích dữ liệu tại một thời điểm cụ thể trong quá khứ. Điều này có được là do tầng lưu trữ dữ liệu của Data Lakehouse duy trì các bản ghi về sự thay đổi của dữ liệu.

Delta Lake lưu trữ dữ liệu bằng định dạng delta, gồm các tệp định dạng parquet và các tệp ghi nhật kí giao dịch (transaction log) để ghi lại toàn bộ các thay đổi gây ra trên dữ liệu. Một khi dữ liệu được thu thập hay cập nhật, Delta Lake sẽ tạo ra một phiên bản tệp parquet mới trong khi vẫn duy trì các phiên bản cũ trước đó. Quá trình này tạo ra một chuỗi các phiên bản tệp, từ đó tạo nên định nghĩa “Time Travel”. Để có thể thực hiện tính năng này, người dùng có thể truy vấn vào dữ liệu bằng cách sử dụng số phiên bản (version number) hay dấu thời gian (timestamp), thông qua các truy vấn SQL hay các API của Delta Lake.

Ta có thể liệt kế một số ca sử dụng cho tính năng Time Travel như sau:

* Rollbacks and error recovery
* Re-creating analysis, reports, and outputs
* Time-series analytics
* Data auditing and lineage
* A/B testing and experiment analysis
* Regulatory compliance and reporting

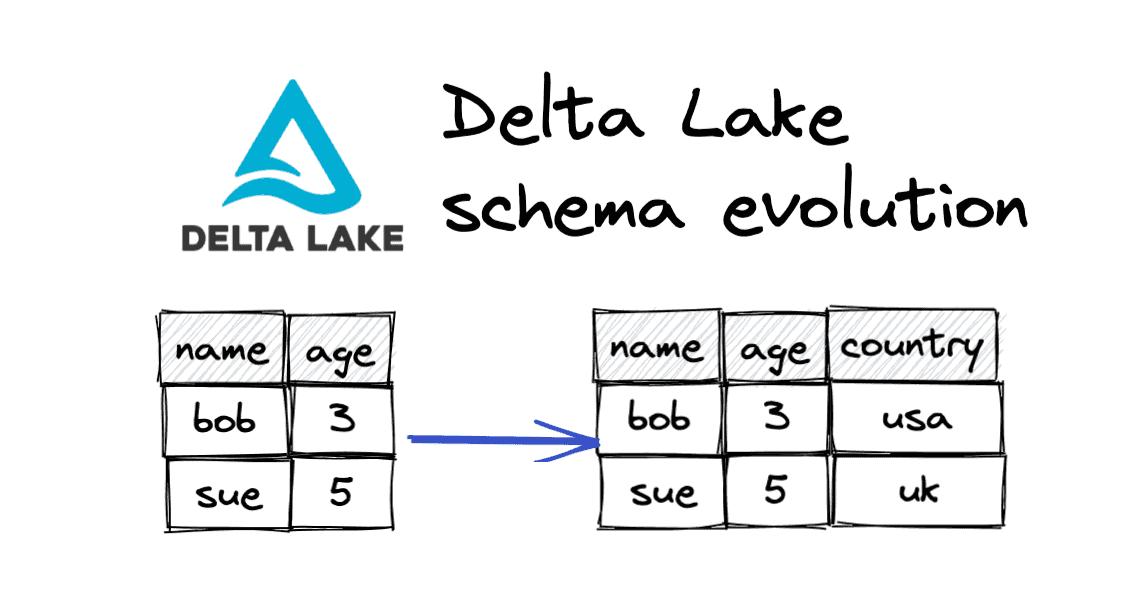
Nhìn chung, Time Travel là một tính năng rất mạnh mẽ và linh hoạt của Data Lakehouse cho phép nâng cao giá trị của việc quản lý và phân tích dữ liệu. Với khả năng có thể quay ngược lại các mốc thời gian và phân tích dữ liệu ở các thời điểm khác nhau, Data Lakehouse sẽ giúp cho doanh nghiệp đưa ra những quyết định đúng đắn hơn, phù hợp với yêu cầu về quy định, duy trì được tính chính xác và toàn vẹn của dữ liệu.

## Thực thi, cải tiến lược đồ và quản trị dữ liệu (Schema Enforcement, Schema Evolution and data gorvenance)

Việc thực thi lược đồ (schema) trong một Data Lakehouse đề cập đến việc áp dụng một cấu trúc hay lược đồ cho dữ liệu được lưu trữ trong Data Lake. Data Lakehouse kết hợp các lợi ích của Data Lake, cho phép lưu trữ dữ liệu đa dạng và thô, với những lợi ích của Data Warehouse như việc áp dụng lược đồ và cung cấp dữ liệu có cấu trúc (chẳng hạn như lược đồ hình sao hay lược đồ bông tuyết).

Việc thực thi lược đồ cũng sẽ đi kèm với tính năng quản trị dữ liệu giúp đảm bảo chất lượng dữ liệu, tính nhất quán và khả năng tương tác trong Data Lakehouse. Những lược đồ này cung cấp một định dạng chuẩn để tổ chức và truy vấn dữ liệu, giúp người dùng dễ dàng hiểu và phân tích dữ liệu.

Bên cạnh đó, kiến trúc Data Lakehouse còn có tính năng **schema evolution**, cho phép người dùng có thể dễ dàng thay đổi dữ liệu, thay đổi cấu trúc của bảng và lược đồ sao cho phù hợp với nhu cầu theo thời gian, mà vẫn đảm bảo tính toàn vẹn của dữ liệu hiện có. Thông thường nhất, tính năng được sử dụng khi thực hiện thao tác nối hoặc ghi đè để tự động điều chỉnh lược đồ, bao gồm việc thêm một hay nhiều cột mới.



Hình 3.2: Minh hoạ về tính năng schema evolution trong Delta Lake

Bằng việc thực thi và cải tiến lược đồ trong một Data Lakehouse, doanh nghiệp có thể đạt được những lợi ích như cải thiện quản trị dữ liệu, đơn giản hóa việc tích hợp dữ liệu và tương tác, tìm kiếm dữ liệu tốt hơn, nâng cao khả năng phân tích và báo cáo dữ liệu.

Thực thi lược đồ có thể được thực hiện bằng cách sử dụng các công cụ và framework khác nhau như **Apache Hive**, **Apache Hudi** hoặc **Delta Lake**. Các công cụ này cung cấp các cơ chế để định nghĩa và thực thi lược đồ trên lớp lưu trữ Data Lake. Các công cụ này cho phép tiến hóa schema, đảm bảo rằng schema có thể được thay đổi theo thời gian để phù hợp với yêu cầu kinh doanh trong khi vẫn duy trì tính tương thích ngược với dữ liệu hiện có.

## Hỗ trợ các dữ liệu phi cấu trúc và theo luồng (Unstructured and streaming data support)

Dữ liệu phi cấu trúc đề cập đến dữ liệu không có cấu trúc hay lược đồ được xác định trước, chẳng hạn như tài liệu văn bản, hình ảnh, video và bài đăng trên mạng xã hội. Data Lakehouse có thể lưu trữ dữ liệu phi cấu trúc ở dạng thô, cho phép phân tích linh hoạt và dựa trên *lược đồ-khi-đọc*. Người dùng có thể áp dụng lược đồ và cấu trúc cho dữ liệu khi truy vấn hoặc phân tích dữ liệu, mang lại sự linh hoạt trong việc khám phá và phân tích dữ liệu.

Dữ liệu streaming là dữ liệu đề cập đến dữ liệu được tạo và xử lý liên tục trong thời gian thực, chẳng hạn như dữ liệu cảm biến, tệp ghi nhật ký và nguồn cấp dữ liệu mạng xã hội. Data Lakehouse có thể tích hợp với các framework và công nghệ phát trực tuyến, chẳng hạn như **Apache Kafka** hoặc **Apache Flink**, để nhập, xử lý và lưu trữ dữ liệu phát trực tuyến. Điều này cho phép phân tích và giám sát theo thời gian thực, cho phép các tổ chức thu được thông tin chi tiết từ dữ liệu khi dữ liệu đến.

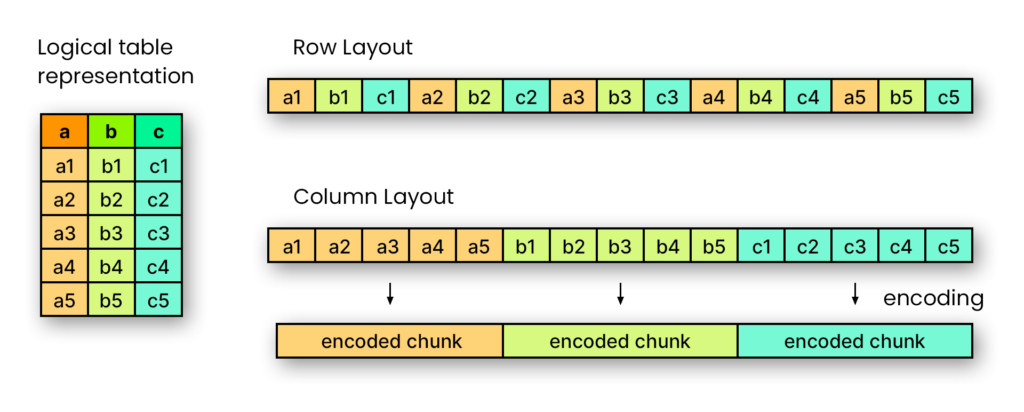
Bằng cách hỗ trợ cả dữ liệu phi cấu trúc và dữ liệu truyền trực tuyến, Data Lakehouse cung cấp một nền tảng thống nhất để lưu trữ, xử lý và phân tích các loại dữ liệu khác nhau, từ dữ liệu có cấu trúc cho đến phi cấu trúc, cho phép các tổ chức tận dụng toàn bộ tiềm năng từ dữ liệu của họ.

## Hỗ trợ định dạng mở (Open formats support)

Kiến trúc Lakehouse hỗ trợ các định dạng mở, là các định dạng tệp được chuẩn hóa, không độc quyền nhằm thúc đẩy khả năng tương tác và tương thích giữa các hệ thống và công cụ khác nhau. Các định dạng mở được ưu tiên trong kiến trúc Data Lakehouse để đảm bảo tính di động của dữ liệu, tuổi thọ và tính linh hoạt trong xử lý và phân tích dữ liệu.

Một số định dạng mở thường được sử dụng trong kiến trúc này bao gồm:

1. **Apache Parquet**: là định dạng lưu trữ dạng cột được tối ưu hóa cao cho hiệu suất và nén truy vấn. Nó cung cấp các hoạt động đọc và ghi hiệu quả trên các tập dữ liệu lớn và được hỗ trợ rộng rãi bởi các công cụ và framework phân tích khác nhau.
2. **Apache Avro**: Avro là một hệ thống tuần tự hóa dữ liệu dựa trên hàng cung cấp định dạng nhị phân. Nó hỗ trợ quá trình phát triển lược đồ, cho phép các lược đồ dữ liệu phát triển theo thời gian mà đảm bảo khả năng tương thích.
3. **Apache ORC**: ORC (Cột hàng được tối ưu hóa) là một định dạng lưu trữ dạng cột khác cung cấp khả năng nén hiệu quả và cải thiện hiệu suất truy vấn. Nó được tối ưu hóa để xử lý dữ liệu quy mô lớn và hỗ trợ quá trình phát triển lược đồ.



Hình 3.3: Minh hoạ kiến trúc của Apache Parquet

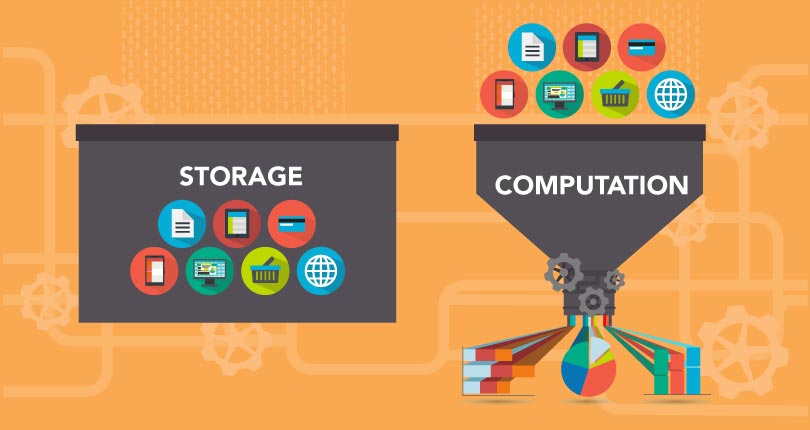
Bằng cách tận dụng các định dạng dữ liệu mở, lakehouses thúc đẩy khả năng tương tác dữ liệu, đơn giản hóa việc tích hợp và xử lý dữ liệu cũng như cung cấp sự linh hoạt trong việc lựa chọn các công cụ và framework phân tích. Các định dạng này cũng đảm bảo rằng dữ liệu vẫn có thể truy cập và sử dụng được trong thời gian dài, ngay cả khi các công nghệ và công cụ liên tục cập nhật và phát triển. Ví dụ với **Parquet**, họ cung cấp các API với đa dạng các công cụ và nền tảng hỗ trợ, bao gồm học máy và các thư viện **Python/R**, giúp cho người dùng dễ dàng truy cập trực tiếp vào dữ liệu.

## Lưu trữ và tính toán tách biệt (Decoupled storage and compute)

Giống như một số Data Warehouse sử dụng lưu trữ đám mây thì kiến trúc lakehouse phân chia các cụm riêng biệt cho hai mục đích lưu trữ và tính toán. Điều này cho phép khả năng thay đổi phạm vi hoạt động cũng như phát triển độc lập, tuỳ theo nhu cầu của doanh nghiệp.

Như đã đề cập, tầng lưu trữ nhận trách nhiệm lưu trữ và quản lý dữ liệu thô ở định dạng ban đầu. Tầng này thường sử dụng hệ thống tệp phân tán hoặc hệ thống lưu trữ đối tượng, chẳng hạn như Hệ thống tệp phân tán **Hadoop (HDFS)** hay **Amazon S3**. Lưu trữ dữ liệu ở dạng thô cho phép sự linh hoạt và nhanh chóng trong việc khám phá và phân tích dữ liệu, vì nó cho phép lược đồ khi đọc thay vì lược đồ khi ghi.

Đối với tầng tính toán, đây là tên gọi khác của tầng xử lý dữ liệu, sẽ chịu trách nhiệm xử lý và phân tích dữ liệu được lưu trữ trong tầng lưu trữ. Nó bao gồm nhiều công cụ phân tích, công cụ truy vấn và framework xử lý khác nhau như **Apache Spark**, **Presto** hoặc **Apache Flink**. Các công cụ điện toán này có thể được cung cấp độc lập và thay đổi được quy mô theo chiều ngang dựa trên các yêu cầu về khối lượng công việc, cho phép khả năng xử lý linh hoạt và hiệu quả.



Hình 3.4: Minh hoạ về lưu trữ và tính toán

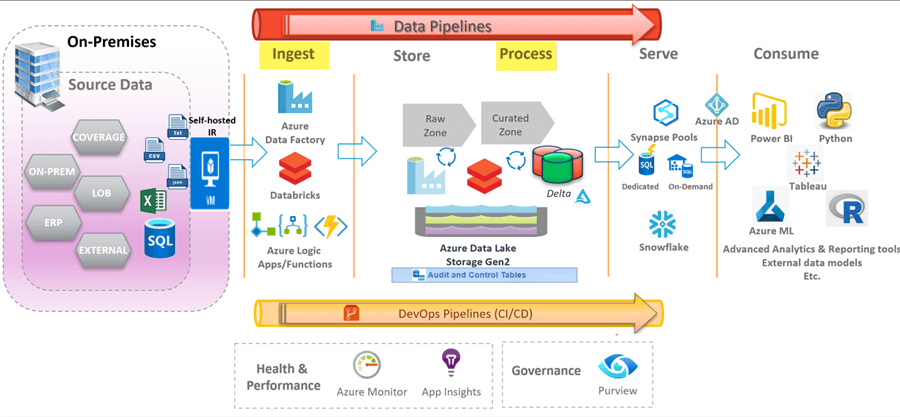
Việc tách rời lưu trữ và tính toán mang lại một số lợi ích:

* Khả năng mở rộng: Với các tầng điện toán và lưu trữ riêng biệt, các doanh nghiệp có thể mở rộng quy mô từng lớp một cách độc lập để đáp ứng nhu cầu thay đổi. Điều này cho phép họ phân bổ tài nguyên một cách tối ưu và tránh việc cung cấp tài nguyên quá mức hoặc dưới mức.
* Khả năng truy cập dữ liệu: Việc tách biệt lưu trữ và tính toán cho phép nhiều công cụ tính toán truy cập và xử lý đồng thời cùng một dữ liệu. Điều này tăng cường khả năng truy cập dữ liệu và cho phép các nhóm và ứng dụng khác nhau tận dụng kho dữ liệu được chia sẻ một cách hiệu quả.
* Hiệu quả về chi phí: Bằng cách tách rời hai tầng này, người dùng có thể tối ưu hóa chi phí bằng cách sử dụng các giải pháp lưu trữ tiết kiệm chi phí để lưu giữ dữ liệu lâu dài và chỉ tận dụng tài nguyên điện toán khi cần. Tính linh hoạt này giúp giảm chi phí cơ sở hạ tầng.
* Linh hoạt và luôn đổi mới: Kiến trúc tách rời cho phép người dùng thử nghiệm các công cụ khác nhau mà không ảnh hưởng đến tầng lưu trữ bên dưới. Nó cho phép áp dụng các công nghệ và framework mới khi chúng xuất hiện, thúc đẩy sự đổi mới trong xử lý và phân tích dữ liệu.

## Hỗ trợ BI (Business Intelligence support)

Các ứng dụng BI trong Data Lakehouse được sử dụng trực tiếp dữ liệu trong Data Lake, điều này giúp loại bỏ nhu cầu cần có những bản sao chép theo đúng quy cách yêu cầu của Data Warehouse.

Tính năng này cho phép người dùng hiểu rõ hơn về dữ liệu của họ một cách dễ dàng và hiệu quả hơn, cải thiện khả năng ra quyết định của họ. Ngoài ra, Nền tảng Data Lakehouse cũng cung cấp các tính năng bảo mật mạnh mẽ, cho phép người dùng kiểm soát quyền truy cập vào dữ liệu của họ, đảm bảo rằng chỉ những người được ủy quyền mới có thể truy cập và thao tác dữ liệu của họ.



Hình 3.5: Một số công cụ BI được tích hợp trong kiến trúc của Azure Data Lakehouse (tầng Consume)

Sự tích hợp này cho phép các tổ chức thực hiện các nhiệm vụ báo cáo và phân tích dữ liệu một cách dễ dàng mà không phải lo lắng về sự phức tạp tiềm ẩn của kho dữ liệu. Nền tảng này cung cấp một nguồn dữ liệu chính xác duy nhất, cho phép các nhóm dữ liệu cung cấp và cập nhật thông tin chi tiết cho người dùng doanh nghiệp. Ngoài ra, nền tảng này cũng hỗ trợ cộng tác giữa các nhóm dữ liệu và người dùng doanh nghiệp, cho phép họ làm việc cùng nhau để xây dựng và chia sẻ thông tin chi tiết.

# So sánh Data Lakehouse với Data Lake và Data Warehouse

Như đã đề cập ở phần 1, kiến trúc Data Lakehouse ra đời để giải quyết những yếu điểm còn tồn tại của hai kiến trúc cũ Data Warehouse và Data Lake, các chuyên gia dữ liệu đã nghiên cứu để tạo ra một công nghệ tích hợp những ưu điểm của các kiến trúc cũ.

Với kho dữ liệu thì đây là một công nghệ lưu trữ dữ liệu lớn có tuổi đời lâu nhất, tuy nhiên chi phí để sử dụng Data Warehouse là tương đối đắt đỏ và sẽ gặp khó khăn nếu như doanh nghiệp muốn sử dụng dữ liệu phi cấu trúc và luồng trực tuyến. Trong khi đó Data Lake có thể lưu trữ nhiều dạng dữ liệu khác nhau với chi phí rẻ hơn, phục vụ cho công việc học máy và khoa học dữ liệu. Dù vậy thì, chúng lại gặp yếu điểm trong tính năng giao dịch ACID (đã đề cập ở phần 3) nên sẽ khó đảm bảo tính nhất quán và tin cậy của dữ liệu. Data Lakehouse đã được thiết kế để có thể kết hợp những ưu điểm kể trên. Bảng so sánh dưới đây giúp cho chúng ta có cái nhìn chi tiết sự khác nhau giữa ba hệ thống lưu dữ dữ liệu này:

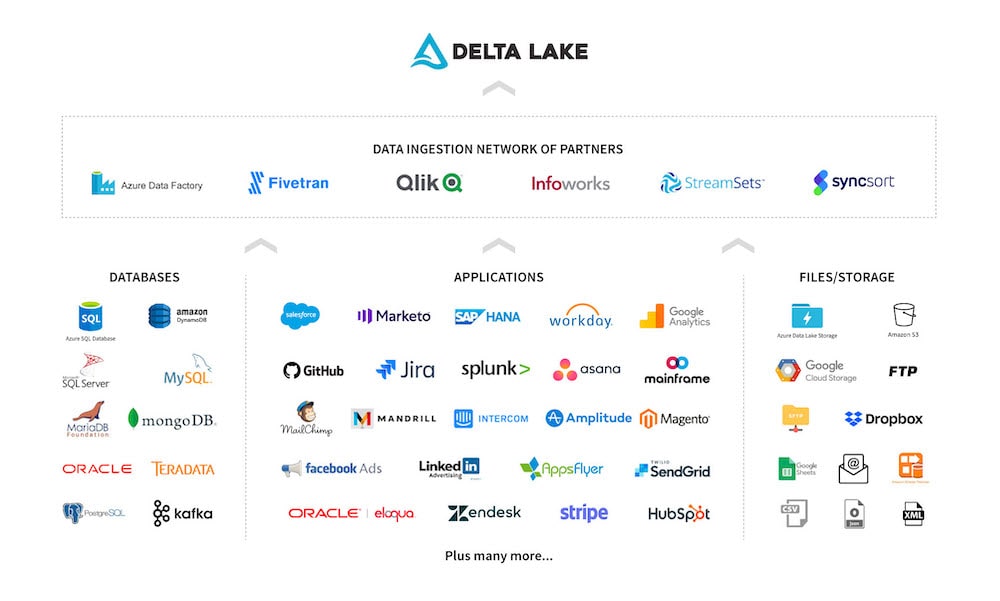
|  |  |  |  |
| --- | --- | --- | --- |
|  | Data Warehouse | Data Lake | Data Lakehouse |
| Định dạng dữ liệu | Đóng, dịnh dạng độc quyền | Định dạng mở | Định dạng mở |
| Kiểu dữ liệu lưu trữ | Dữ liệu có cấu trúc | Tất cả các dữ liệu từ có cấu trúc, bán cấu trúc, phi cấu trúc | Tất cả các dữ liệu từ có cấu trúc, bán cấu trúc, phi cấu trúc cho đến dữ liệu luồng trực tuyến |
| Chất lượng dữ liệu | Dữ liệu có độ chính xác cao, đáng tin cậy, tính bảo mật cao | Dữ liệu thô, chất lượng thấp | Dữ liệu thô và có cấu trúc, chất lượng và tính bảo mật cao |
| Cách thức truy cập  dữ liệu | SQL | Sử dụng APIs cho việc truy cập trực tiếp vào tệp thông qua SQL, R, Python… | Sử dụng APIs cho việc truy cập trực tiếp vào tệp thông qua SQL, R, Python… |
| Quy trình xử lý | ETL | ELT | Cả hai quy trình ETL và ELT |
| Chi phí và khả năng mở rộng | Đắt đỏ nếu tăng theo cấp số nhân và tốn nhiều thời gian | Giá thành hợp lý, tốc độ nhanh và linh hoạt | Chi phí hợp lý, dễ dàng sử dụng, tốc độ nhanh và linh hoạt |
| Tuân thủ giao dịch ACID | Ghi lại dữ liệu trong phiên ACID để đảm bảo tính toàn vẹn cao nhất | Không hỗ trợ | Tuân thủ nghiêm ngặt ACID, đảm bảo tính nhất quản của dữ liệu khi được đọc và ghi bởi nhiều nguồn |
| Quản trị và bảo mật | Bảo mật chi tiết và quản trị cấp hàng/cột cho bảng | Quản trị kém, bảo mật cần được áp dụng cho các tệp | Bảo mật chi tiết và quản trị cấp hàng/cột cho bảng |
| Hiệu suất xử lý | Cao | Thấp | Cao |
| Các ca  sử dụng  hỗ trợ | Tối ưu cho việc phân tích dữ liệu là xử lý use-cases BI | Thích hợp cho khối công việc học máy và trí tuệ nhân tạo | Được thiết kế để đảm nhận tất cả công việc, từ BI, SQL, phân tích dữ liệu và học máy |

Nhìn chung, kiến trúc Data Lakehouse sẽ mang đến cho người sử dụng những giải pháp về các vấn đề còn tồn đọng của các kiến trúc trong thị trường kho dữ liệu. Khả năng độc nhất này của lakehouse để quản lý dữ liệu trong môi trường mở; kết hợp tất cả các định dạng, kiểu dữ liệu khác nhau từ toàn bộ các nguồn; đồng thời đi cùng với sự chuyên tâm sâu cho mảng khoa học dữ liệu của Data Lake và quy trình phân tích người dùng cuối của Data Warehouse đã đem đến những giá trị đáng kinh ngạc cho các doanh nghiệp.

# Các công nghệ dùng trong một hệ thống dữ liệu sử dụng kiến trúc Data Lakehouse

Trong phần 5, ta sẽ đi tìm hiểu trong hệ thống sử dụng kiến trúc Data Lakehouse để lưu trữ và xử lý dữ liệu sẽ có những công nghệ, công cụ gì. Thông thường một hệ thống này của mỗi doanh nghiệp có kiến trúc tương đối phức tạp, có nhiều thành phần (như mô tả ở phần 2), và một hệ thống có thể xem như một hệ sinh thái lưu trữ và xử lý dữ liệu, đi kèm với đó là danh sách khổng lồ các công cụ hỗ trợ cho từng pha trong hệ thống đó.

Có một thực tế là, mỗi nhà cung cấp lại có những hướng dẫn xây dựng kiến trúc lakehouse khác nhau, ví dụ Databricks cung cấp giải pháp gồm 5 bước để doanh nghiệp xây dựng Data Lakehouse của riêng mình. Tuy nhiên chúng đều bắt nguồn từ việc lựa chọn các công cụ cho từng thành phần của kiến trúc, và mỗi kiến trúc trên thị trường lại có những hệ sinh thái các công cụ riêng. Ta có thể xem những mạng lưới các ứng dụng có mặt trong hai kiến trúc Data Lakehouse của Delta Lake và Snowflake dưới đây:



Hình 5.1: Hệ sinh thái của Delta Lake năm 2020



Hình 5.2: Hệ sinh thái của của Snowflake năm 2023

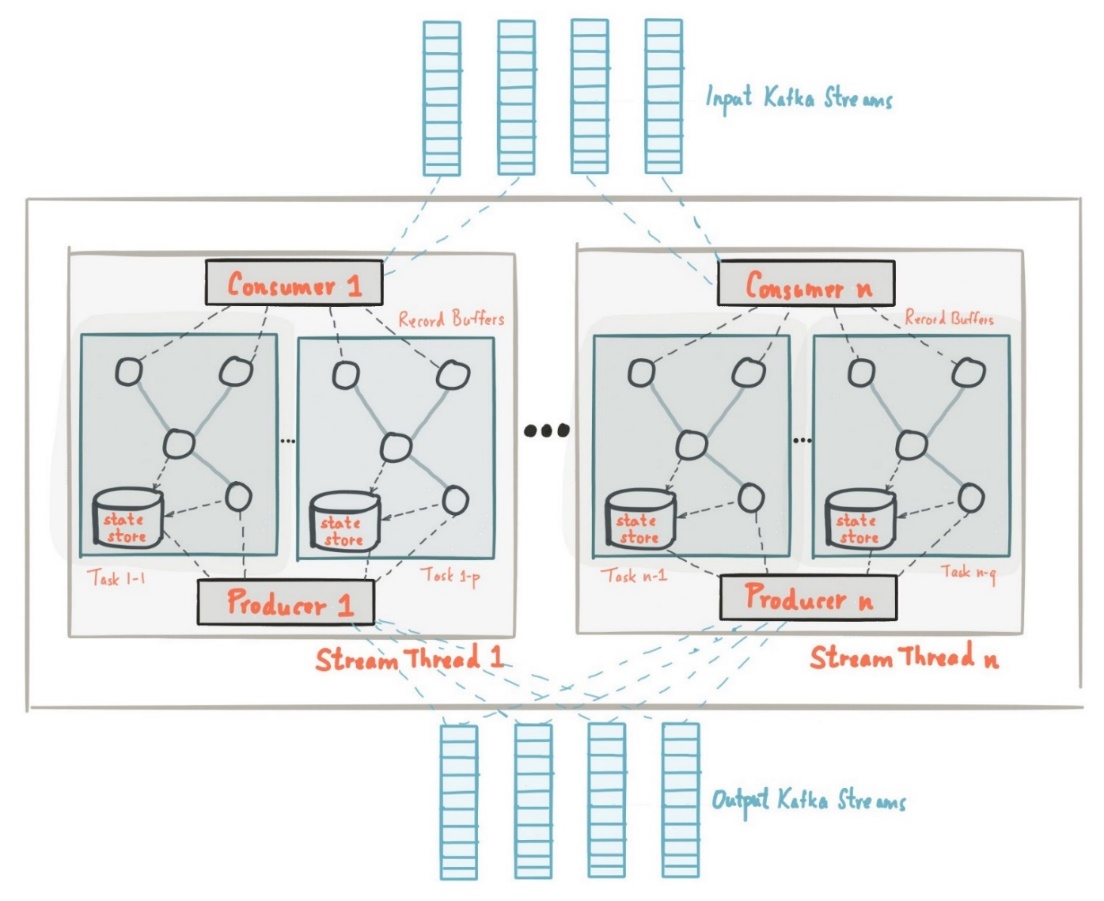
Sau đây chúng ta sẽ tìm hiểu xem ở mỗi tầng công vụ trong kiến trúc của Data Lakehouse sẽ sử dụng công nghệ và công cụ gì cho công việc hàng ngày.

## Tầng thu thập dữ liệu

Tầng đầu tiên trong hệ thống, như đã phân tích, đảm nhận việc kéo dữ liệu từ các nguồn và đem chúng đến với tầng lưu trữ. Và để hợp nhất quy trình xử lý dữ liệu theo lô và theo luồng thì doanh nghiệp có thể sử dụng các giao thức khác nhau, từ đó kết nối với các nguồn cả trong lẫn ngoài Data Lakehouse, chẳng hạn như:

* Hệ quản trị cơ sở dữ liệu quan hệ
* Cơ sở dữ liệu NoSQL
* Ứng dụng CRM
* Ứng dụng SaaS
* Cảm biến IoT
* Mạng xã hội
* Chia sẻ tệp
* Trang web…

Các thành phần sử dụng trong bước này có thể bao gồm **Amazon Data Migration Service (Amazon DMS)** được dùng để thêm dữ liệu RDBMSs và cơ sở dữ liệu NoSQL, trong khi đó ta có thể sử dụng **Apache Kafka** cho luồng dữ liệu trực tuyến.



Hình 5.3: Kiến trúc luồng dữ liệu trực tuyến trong Apache Kafka

Hiện nay có một số công cụ chuyên dụng cho quá trình thu thập dữ liệu tuỳ thuộc theo kiến trúc và yêu cầu mà doanh nghiệp lựa chọn, ngoài các tên kể trên, còn có các công cụ như **Datavid Rover**, **Improvado**, **Google BigQuery**, **Hevo**, **Apache NiFi**, **Wavefront**…

## Tầng lưu trữ dữ liệu

Thiết kế lakehouse hướng đến việc cho phép lưu trữ tất cả loại dữ liệu với chi phí thấp, ví dụ như **AWS S3**. Các công cụ khách từ đó có thể đọc những đối tượng trực tiếp từ kho, sử dụng định dạng tệp mở. Các lược đồ của của tập dữ liệu có cấu trúc và phi cấu trúc được giữ lại ở tầng siêu dữ liệu cho các thành phần để ứng dụng nó vào dữ liệu trong khi đọc chúng.

Data Lakehouse là kiến trúc phù hợp nhất cho dịch vụ lưu trữ đám mây để phân chia việc tính toán và lưu trữ, tất nhiên là có thể cài đặt tại chỗ, tiêu biểu và phổ biến nhất là nền tảng **hệ thống tệp phân tán của Hadoop – HDFS**.



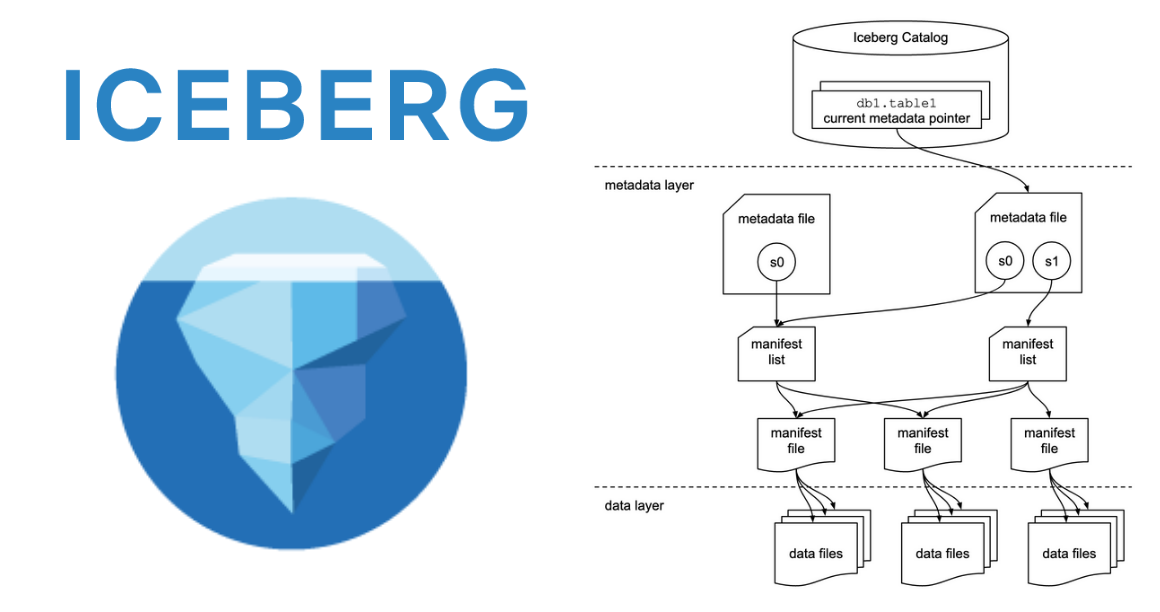
Hình 5.4: Mô tả kiến trúc của HDFS

## Tầng siêu dữ liệu

Tầng siêu dữ liệu giúp áp dụng các kiến trúc lược đồ của Data Warehouse như lược đồ hình sao, bông tuyết; triển khai quản trị lược đồ và cung cấp chức năng kiếm toán và quản trị dữ liệu trực tiếp trên Data Lake, giúp cải thiện chất lượng của toàn bộ các luồng dữ liệu.

Quản lý lược đồ bao gồm thực thi và cải tiến lược đồ (đã đề cập ở mục 3.2). Schema enforcement cho phép người dùng điều khiển tính toàn vẹn và chất lượng dữ liệu. Schema evolution cho phép sự thay đổi của lược đồ hiện tại của các bảng phù hợp với dữ liệu động.

Các hệ thống như **Delta Lake** hay **Apache Iceberg** luôn sẵn sàng thực thi quy trình quản lý dữ liệu và tối ưu hiệu suất theo hướng này.



Hình 5.5: Kiến trúc của Apache Iceberg

## Tầng xử lý dữ liệu

Tầng API (đối với kiến trúc của Databricks) hay tầng xử lý dữ liệu (đối với AWS) cho phép người dùng cuối có thể tiến hành xử lý các tác vụ nhanh hơn và có những phân tích chuyên sâu hơn. Các API siêu dữ liệu giúp thông hiểu những dữ liệu gì sẽ được yêu cầu cho một ứng dụng cụ thể và cách để lấy được chúng.

Trong khi đó, tầng xử lý dữ liệu cung cấp các công cụ giúp tìm ra tập dữ liệu có đặc điểm phù hợp nhất (về size, định dạng, lược đồ, tốc độ) trong thời gian ngắn nhất. Vì có thể truy cập vào không gian lưu trữ dữ liệu của lakehouse và các danh mục tương tự nên tất nhiên ta có thể truy cập cập toàn bộ dữ liệu và siêu dữ liệu trong Data Lakehouse. Tầng xử lý có khả năng mở rộng để xử lý khối lượng dữ liệu lớn với chi phí thấp, đồng thời cung cấp các thành phần hỗ trợ các công đoạn xử lý dữ liệu. Hiện nay, các doanh nghiệp đang sử dụng **SQL**, **Apache Spark**, **Rapid Miner**, **Apache Hive** cho quy trình xử lý dữ liệu.



Hình 5.6: Các công cụ phổ biến cho quá trình xử lý dữ liệu

Trong trường hợp phải dùng đến các thư viện học máy, chúng ta có thể sử dụng **TensorFlow** và **Spark Mlib** để có thể đọc các định dạng tệp như **Parquet** và truy vấn tầng siêu dữ liệu trực tiếp.

## Tầng phân tích và trực quan hoá dữ liệu

Tầng phân tích và trực quan hoá dữ liệu có nhiều công cụ và ứng dụng khác nhau như **Power BI**, **Tableau**, **Looker**... Với kiến trúc lakehouse, các ứng dụng khách có quyền truy cập vào tất cả dữ liệu được lưu trữ trong Data Lake và tất cả siêu dữ liệu. Tất cả người dùng trong một tổ chức có thể sử dụng lakehouse để thực hiện các nhiệm vụ phân tích bao gồm bảng điều khiển BI về kinh doanh, trực quan hóa dữ liệu, truy vấn SQL và công việc học máy.



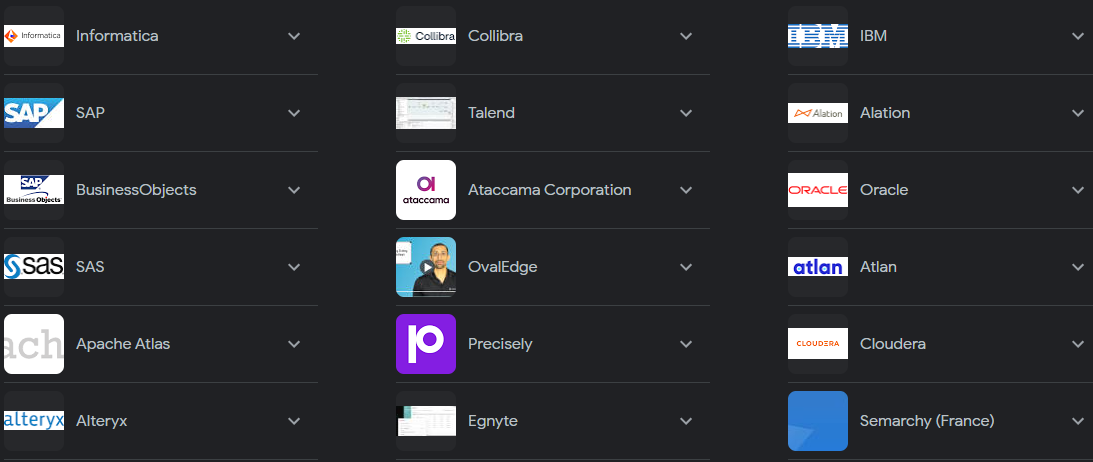
Hình 5.7: Các công cụ BI phổ biến được sử dụng trong kiến trúc Lakehouse

Với AWS, họ tạo ra một hệ sinh thái riêng gồm các phần mềm phục vụ cho từng mục đích của tầng này, ví dụ:

* Khi muốn truy vấn tương tác thông qua SQL, ta có thể sử dụng **Amazon Athena**
* Sử dụng Data Warehouse và phân tích theo đợt, ta có thể sử dụng **Amazon Redshift**
* Khi dùng cho các công việc BI, **Amazon QuickSight** là giải pháp được nhắc đến
* Phân tích có dự đoán và công việc học máy, lựa chọn mà Amazon đưa đến đó là **Amazon SageMaker**

## Tầng quản trị dữ liệu

Quản trị dữ liệu là quy trình tạo ra các tiêu chuẩn bên trong, bên cạnh những chính sách để kiểm soát những ai có thể truy cập dữ liệu và cách dữ liệu được dùng trong vận hành kinh doanh và phân tích. Một chương trình quản trị dữ liệu đủ tốt sẽ đảm bảo dữ liệu có được sự nhất quán, chất lượng tốt, khả dụng và phù hợp với những chính sách, quy định đã được đưa ra.



Hình 5.8: Danh sách các công cụ quản trị dữ liệu phổ biến

Có thể kể đến một số công cụ được các công ty lớn trên thế giới sử dụng hiện nay, như **Alation Data Catalog**, **Ataccama**, **Apache Atlas**, **Collibra**, **SAP Master Data Governance**, **Informatica**,…

## Tầng bảo mật dữ liệu

Các công cụ bảo mật dữ liệu cung cấp khả năng bảo vệ chuyên biệt cho dữ liệu bên cạnh các sản phẩm bảo mật mạng hiện có. Về lý thuyết, dữ liệu đã được bảo vệ chống lại hoạt động độc hại bằng tường lửa và hệ thống phát hiện xâm nhập. Tuy nhiên, dữ liệu trong Data Lakehouse yêu cầu được bảo vệ riêng để chống lại các mối đe dọa dành bởi vì dữ liệu phải trở nên dễ tiếp cận hơn đối với phạm vi người dùng rộng hơn. Các công cụ cũng sẽ chuyên biệt để bảo vệ tập hợp các tập dữ liệu được triển khai tại chỗ, trên đám mây hoặc trong các môi trường kết hợp.

Các công cụ nổi tiếng có thể kể đến như **IBM Security Guardium**, **dbWatch**, **Netwrick Auditor**…

Tổng quan, ta đã vừa tìm hiểu xem để xây dựng nên kiến trúc Data Lakehouse của riêng mình, các doanh nghiệp, công ty đang có xu hướng sử dụng những công cụ, giải pháp như thế nào. Với sự phát triển bủng nổ của dữ liệu lớn nói riêng và công nghệ thông tin nói chung, các phần mềm hỗ trợ cho hệ thống dữ liệu của doanh nghiệp ngành càng tăng cả về lượng và chất. Hầu hết các nghiệp vụ của doanh nghiệp, dù ở lĩnh vực nào cũng đã có những công cụ chuyên biệt cho lĩnh vực đó. Ngoài ra, những nhà cung cấp giải pháp Data Lakehouse như AWS, Databricks hay Snowflake đều đã xây dựng sẵn, những doanh nghiệp này có vô số công cụ trong hệ sinh thái của họ, và tuỳ theo nhu cầu nghiệp vụ từ công ty muốn áp dụng thì họ sẽ đem đến giải pháp tốt nhất.

# Các case-study tiêu biểu áp dụng thành công kiến trúc Data Lakehouse

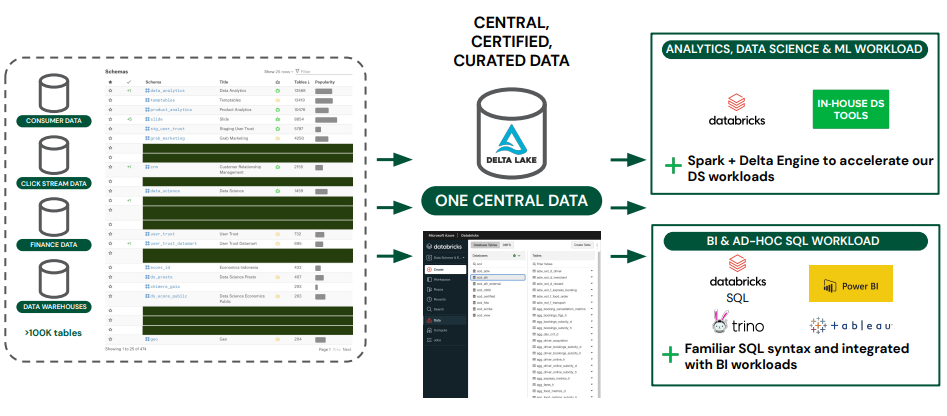
Sau khi đã tìm hiểu tổng quan về kiến trúc, những tính năng nổi bật cũng như các công nghệ được sử dụng trong Data Lakehouse, ta đã có một cái nhìn trực quan hơn về một kiến trúc mới này. Và để có thể kiểm chứng được những ưu điểm và sự cần thiết của chúng trong công việc quản trị dữ liệu, ta sẽ tìm hiểu một số case-study của những công ty, tập đoàn đã, đang thành công trong việc sử dụng mô hình kiến trúc Data Lakehouse để phục vụ cho nghiệp vụ của họ.

## GRAB và câu chuyện tìm ra mô hình hoàn hảo nhất cho dữ liệu của mình

Trong sự kiện Data + AI SUMMIT 2022, trưởng nhóm phân tích khoa học dữ liệu của GRAB – ông Zulfikar Lazuardi Maulana đã chia sẻ về quá trình lựa chọn mô hình kiến trúc Data Lakehouse để xây dựng một hệ thống dữ liệu cho công ty của mình.

“Sẽ là một điều chẳng hay ho gì nếu mà ta phải lựa chọn giữa Data Lake và Data Warehouse”, vị này chia sẻ và tin tưởng rằng GRAB đã tận dụng được những ưu điểm của cả hai kiểu mô hình này để xây dựng nên một nền tảng dữ liệu tập trung.

Được mệnh danh là một dữ liệu trung tâm của Grab, hay gọi tắt là OCD (One central data), nền tảng này được xây dựng dựa trên sản phẩm của Databricks là Delta Lake và được thiết kế để kết hợp tính linh hoạt và độ tin cậy của Data Lake cũng như khả năng BI của Data Warehouse.



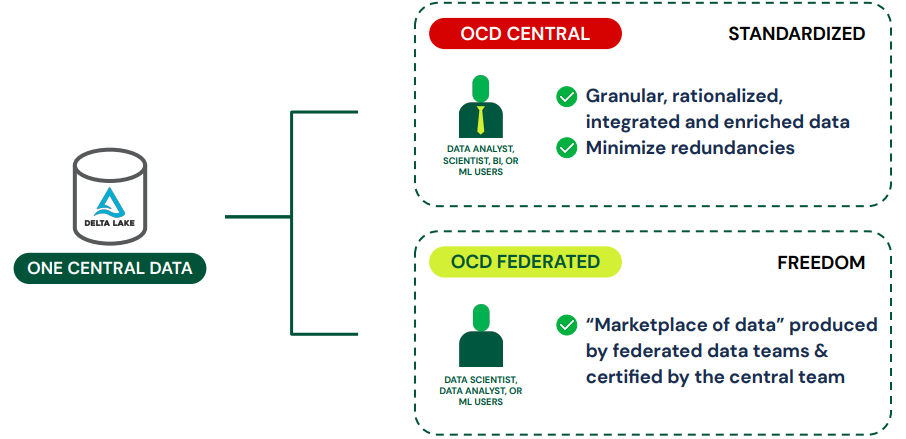
Hình 6.1: Minh hoạ kiến trúc OCD hay Analytics Lakehouse của GRAB

Giải pháp được cân nhắc sau khi công ty bắt đầu tiến hành hoạt động với một Data Lake nhưng gặp khó khăn về quá trình chuẩn hóa cũng như các trường hợp phải sử dụng BI cho mục đích phân tích. Sau đó, họ quyết định sử dụng Data Warehouse để giải quyết vấn đề, nhưng việc này chỉ dẫn đến các khối dữ liệu được lưu trữ (data siloes), khiến việc phục vụ 25 triệu người dùng giao dịch hàng tháng và hơn 9 triệu đối tác đã đăng ký trở nên khó khăn hơn.

Lazuardi cho biết: "Một hệ thống dữ liệu tập trung đã được xây dựng để tận dụng tất cả các khả năng của Data Lake và Data Warehouse. Bằng cách sử dụng OCD, chúng tôi có thể hỗ trợ đầy đủ tất cả các ca sử dụng cho mục đích phân tích, cho khoa học dữ liệu và thậm chí cho BI".

"Đối với khía cạnh khoa học dữ liệu, chúng tôi có thể xây dựng các mô hình với đầy đủ lợi ích của Spark và Delta Engine, còn đối với mảng BI, nó sẽ đem lại sự quen thuộc và hiệu suất được đảm bảo của Data Warehouse."

Tạo nên OCD là hai thành phần chính: OCD central và OCD federated. Lazuardi cho biết OCD central đã được thiết kế để "hoạt động như một nguồn thông tin duy nhất cho nhiều cá nhân", trong khi OCD federated đã được thiết kế để cung cấp cho hơn 50 nhóm dữ liệu trong công ty một "thị trường" dữ liệu để tạo ra các bộ dữ liệu.



Hình 6.2: Hai thành phần chính trong nền tảng OCD của GRAB

Đồng thời, Lazuardi chỉ ra rằng OCD đã giúp nhóm dữ liệu tăng tốc quá trình ETL, đồng thời loại bỏ mọi công việc thủ công liên quan đến các bộ dữ liệu tích hợp.

Một số trường hợp sử dụng cụ thể hơn mà OCD đã hỗ trợ bao gồm dự đoán giá trị trọn đời của khách hàng, hỗ trợ nhóm tiếp thị đánh giá và tăng cường tương tác với khách hàng thông qua quá trình cá nhân hóa trong tiếp thị.

## AT&T

AT&T đang nỗ lực hết sức để cung cấp cho 182 triệu khách hàng của mình có thể thông tin liên lạc một các an toàn, tin cậy và không gặp trở ngại. Với 10 triệu giao dịch diễn ra mỗi giây, việc vượt qua những hình thức lừa đảo, từ cuộc gọi tự động, tin nhắn đến các hành vi trộm cắp thông tin đòi hỏi cần phải khai thác sức mạnh của dữ liệu và AI để ngăn chặn các cuộc tấn công trước khi chúng xảy ra. Nhưng kiến trúc tại chỗ của AT&T đã tạo ra sự phức tạp về cơ sở hạ tầng và không mang lại sự đổi mới cần thiết cho trải nghiệm của khách hàng. Bằng cách hiện đại hóa cơ sở hạ tầng dữ liệu của họ trên đám mây với Databricks Lakehouse, AT&T đã có thể triển khai cách tiếp cận thống nhất đối với dữ liệu và AI. Hiện họ có hơn 100 mẫu ML đang được sản xuất để có thể bảo vệ khách hàng và doanh nghiệp của họ được tốt hơn, ngoài ra AT&T còn sử dụng dữ liệu thời gian thực, cảnh báo tự động và các khuyến nghị để trang bị cho nhân viên trong các hoạt động công việc hàng ngày. Sử dụng một hệ thống như này giúp cho công ty có thể làm việc với mọi dạng dữ liệu được sinh ra, sau đó tiến hành chuẩn hoá, chạy các mô hình học máy và có thể đưa ra các cảnh báo gian lận theo thời gian thực. Với một Lakehouse hợp nhất cho dữ liệu và AI, AT&T đã giảm đáng kể tình trạng gian lận và có một lộ trình mạnh mẽ để cung cấp nhiều giải pháp dựa trên dữ liệu hơn, giúp dân chủ hóa AI trong toàn doanh nghiệp.

“Databricks Lakehouse đã tạo nên một nền tảng hợp nhất kết hợp dữ liệu và AI để cung cấp các giải pháp dự đoán giúp bảo vệ khách hàng và doanh nghiệp của chúng tôi bằng cách ngăn chặn gian lận trước khi nó xảy ra” - Kate Hopkins, phó chủ tịch của AT&T chia sẻ. Kể từ khi loại bỏ hệ thống gian lận dựa trên quy tắc và tạo ra các mô hình ML để phát hiện gian lận tự động, theo thời gian thực, AT&T đã làm giảm đến 80% các vụ gian lận có thể xảy ra. Hopkins cho biết: “Giờ đây, khả năng phát hiện các gian lận của chúng tôi diễn ra theo thời gian thực, chúng tôi có thể đánh lừa những kẻ lừa đảo và đi trước những nỗ lực của chúng”.

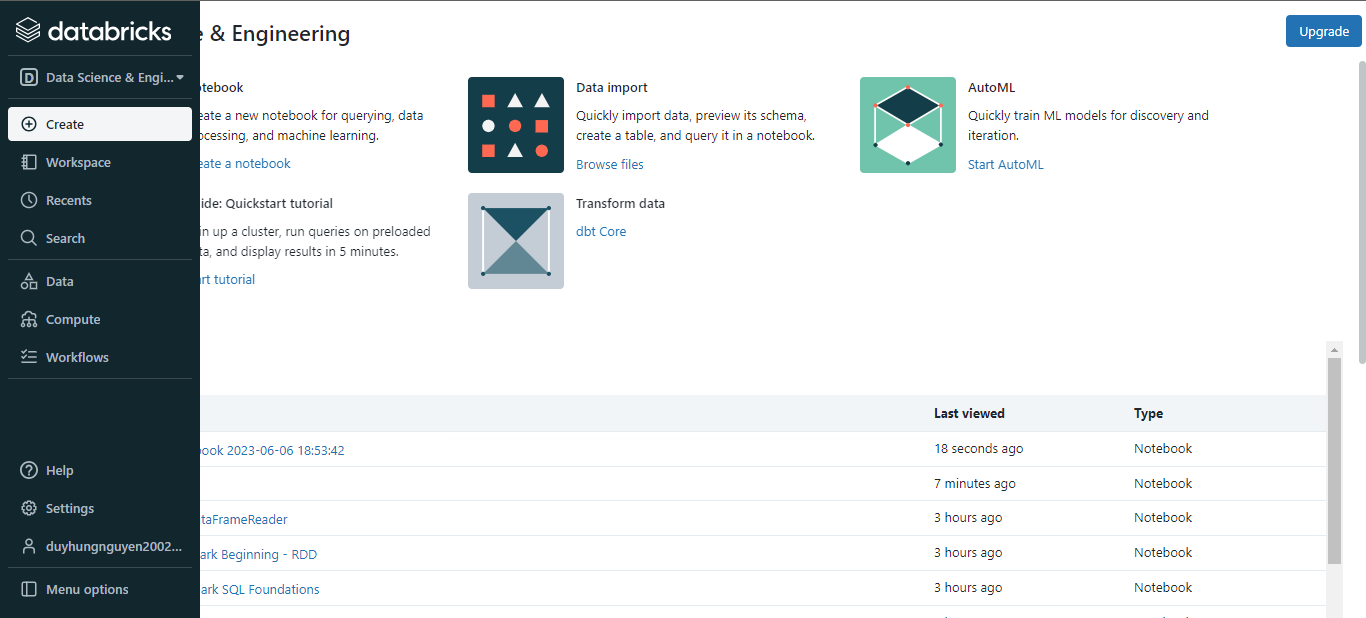
Phát hiện gian lận chỉ là một ví dụ về cách AT&T có thể tạo ra sự thay đổi với khả năng mở rộng và truy cập dữ liệu một cách dân chủ trên nền tảng Databricks Lakehouse. Trong tương lai, AT&T sẽ tiếp tục tăng cường áp dụng cho các trường hợp sử dụng mang lại lợi ích như việc điều phối, độ tín nhiệm của dịch vụ, chất lượng phủ sóng và doanh số bán hàng. Mục tiêu của họ là loại bỏ hoàn toàn các Data Lake tại chỗ vào năm 2023.

# Demo một hệ thống Data Lakehouse cơ bản

Chúng ta đã tìm hiểu tổng quan các nội dung về kiến trúc Data Lakehouse, ở chương này ta sẽ đi vào thực tế việc xây dựng và sử dụng một hệ thống cơ bản của mô hình này. Trong khuôn khổ các bài báo cáo thì phần demo hệ thống sẽ chỉ đi vào một số tính năng nổi bật của kiến trúc đã được đề cập tới trong chương 3.

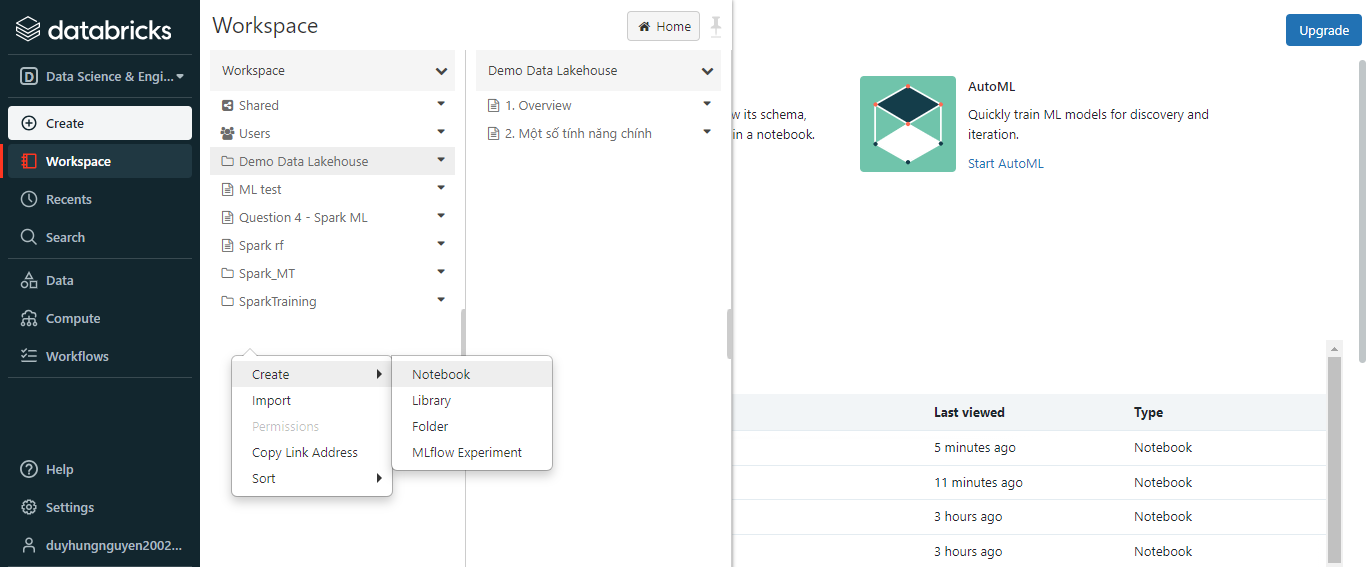
Trong phần demo này, ta sẽ sử dụng giải pháp **Delta Lake** trên giao diện của **databricks** là Databricks Lakehouse Platform, xây dựng cụm tính toán và lưu trữ dữ liệu, sau đó tiến hành một số bước xử lý dữ liệu để minh hoạ tính năng của Data Lakehouse. Ta sẽ cùng xem qua một số hình ảnh minh hoạ cho demo.

Sau khi đăng kí và đăng nhập tài khoản cộng đồng của databricks, ta sẽ có giao diện quản lý hệ thống của databricks như sau:



Hình 7.1: Giao diện của Databricks Lakehouse Platform

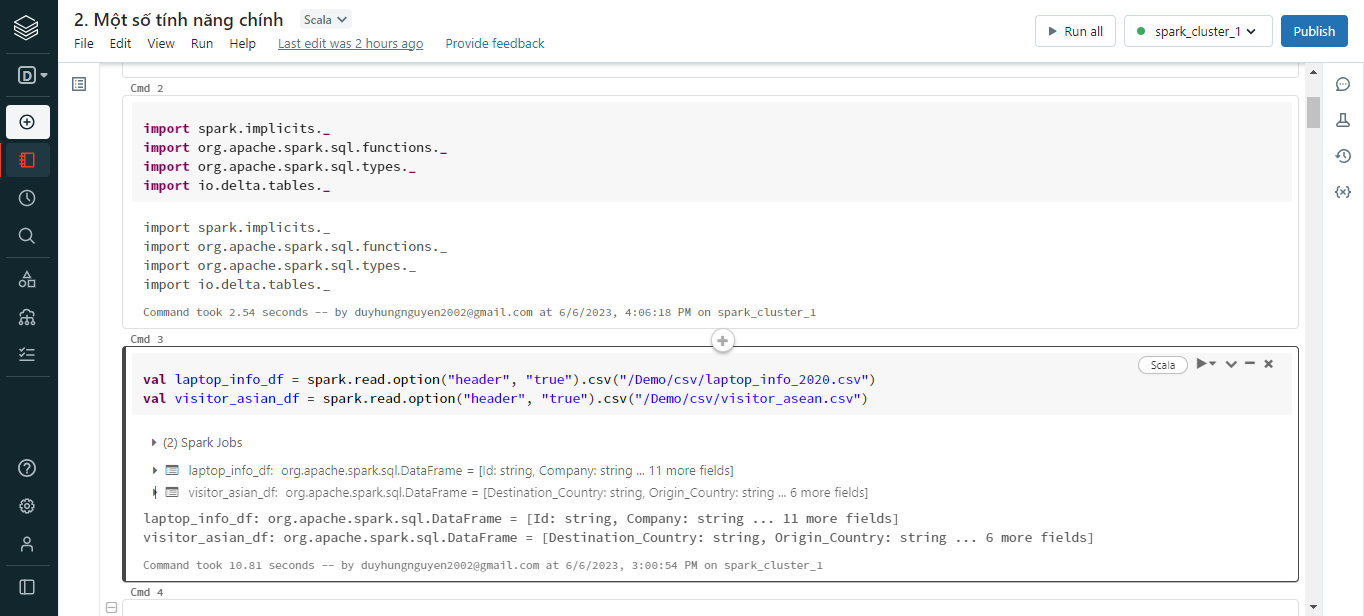
Tiếp theo, ta tiến hành tạo một Notebook mới trong phần **Workspace** (bấm chuột phải, chọn “Create”, sau đó chọn “Notebook”)



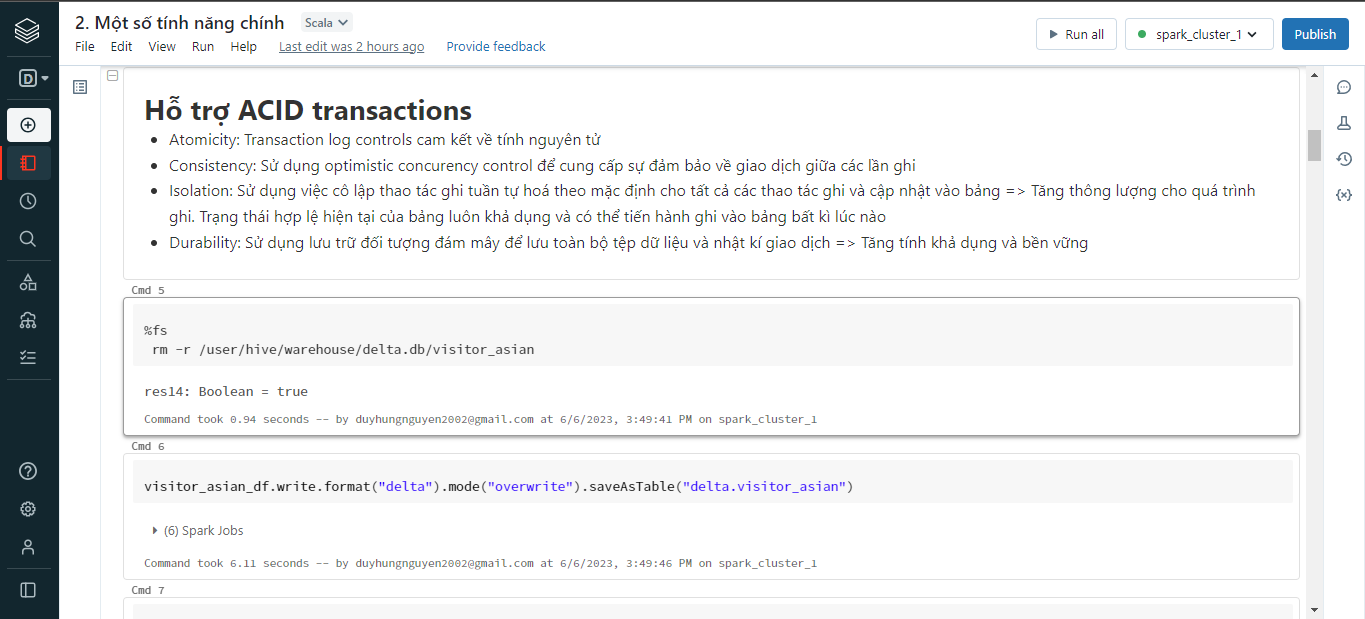
Hình 7.2: Hướng dẫn tạo notebook để tiến hành demo hệ thống

Ở đây, phần demo của em gồm hai notebook, nằm trong folder “Demo Data Lakehouse”. Trong notebook “1. Tổng quan”, em tiến hành một số thao tác cơ bản với cụm tính toán, bao gồm việc thêm, sửa, xoá thư mục; đọc, ghi tệp và thực hiện truy vấn đơn giản. Đối với lựa chọn xây dựng như trên, ta có một cụm tính toán sử dụng Apache Spark và Scala. Vì đây chỉ là phiên bản cộng đồng nên ta chỉ sử dụng được một vài tính năng nhất định, tuy nhiên ta vẫn có thể sử dụng các công nghệ khác kết nối với cụm này để có thể phục vụ cho công việc. Nếu sử dụng phiên bản dành cho doanh nghiệp, ta sẽ được hỗ trợ tối đa, cả về hệ sinh thái lẫn khả năng tính toán và lưu trữ.

Phần demo chính về các tính năng nằm ở notebook “2. Một số tính năng chính”. Để phục vụ cho việc minh hoạ các tính năng, em có sử dựng hai tệp dữ liệu về thông tin laptop năm 2020 và số liệu khách du lịch đến các nước Đông Nam Á (nguồn được ghi trong phần Tham khảo).



## Hỗ trợ giao dịch ACID

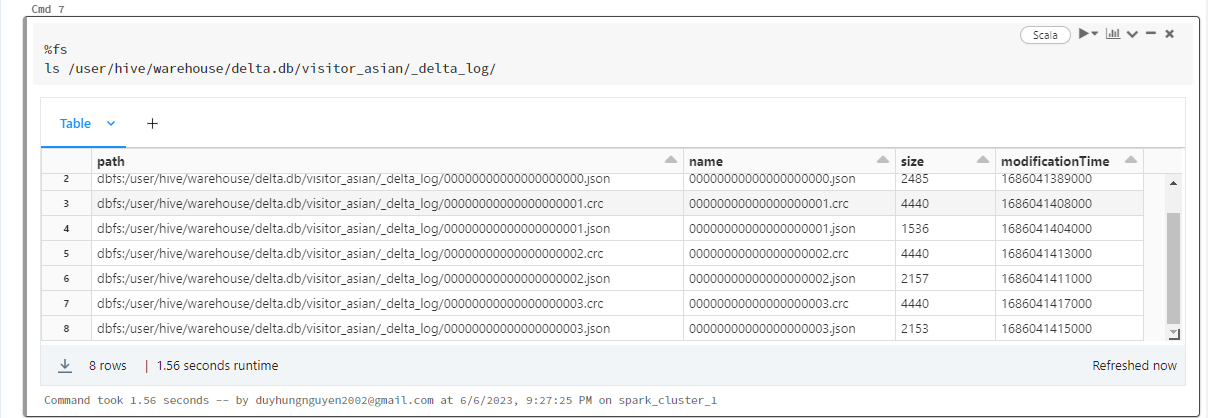


Đối với tính năng này, ta sẽ sử dụng định bảng delta để minh hoạ. Ta sẽ tiến hành ghi dataframe và lưu dưới dạng bảng trong metastore.

Sau khi đã ghi thành công, ta sẽ tiến hành các hành động DML (Data Manipulation Language operation) bao gồm thêm bản ghi, cập nhật bản ghi và xoá bản ghi để chứng minh tính chất ACID của giao dịch trong Data Lakehouse.

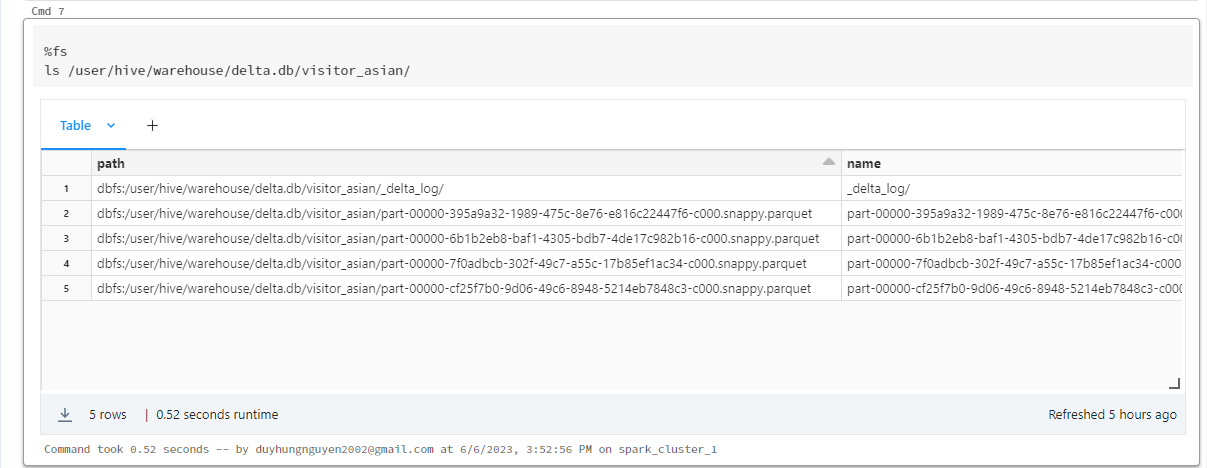


Hình 7.3: DML Operations trong Data Lakehouse



Hình 7.4: Danh sách các tệp lưu thông tin về siêu dữ liệu và nội dung của giao dịch

Nếu như có thao tác ghi hay cập nhật nào bị lỗi, nó sẽ không gây ảnh hưởng đến phiên bản hiện tại của tệp Parquet, trong khi vẫn lưu lại những phiên bản cũ. Những định dạng tệp này bao gồm các tệp CRC và JSON dùng để lưu trữ siêu dữ liệu, các thông tin liên quan đến nội dung chỉnh sửa. Tất cả đều được lưu trên kho lưu trữ đám mây, giúp đảm bảo tính khả dụng và bền vững.



Hình 7.5: Các phiên bản thay đổi của dữ liệu được ghi lại thành từng tệp định dạng Parquet

Ban đầu khi tiến hành ghi dữ liệu từ dataframe, ta chỉ có một tệp Parquet, tuy nhiên sau khi tiến hành các bước cập nhật dữ liệu trên bảng, ta thấy đã có thêm ba tệp Parquet mới. Điều này có thể giải thích là vì, các thao tác đọc và ghi được tiến hành độc lập với nhau theo tuần tự, điều này để đảm bảo tính nhất quán và cô lập của giao dịch, từ đó mà thông lượng cho thao tác ghi sẽ lớn hơn.

## Thực thi và cải tiến lược đồ

### Thực thi lược đồ (Schema Enforcement)

Đối với Data Lake mà điển hình khi ta sử dụng định dạng Parquet, ta sẽ không có tính năng này và nếu muốn đạt được hiệu quá thì sẽ phải tiến hành thiết lập thủ công. Tất nhiên không phải lúc nào ta cũng nhớ được điều đó, và với Data Lakehouse thì tính năng đã được xây dựng bên trong kiến trúc.

Lấy ví dụ cho tính năng này, ta sẽ tạo hai dataframe có cấu trúc khác nhau và cùng được ghi vào cùng một thư mục theo hình thức “append” như hình sau:



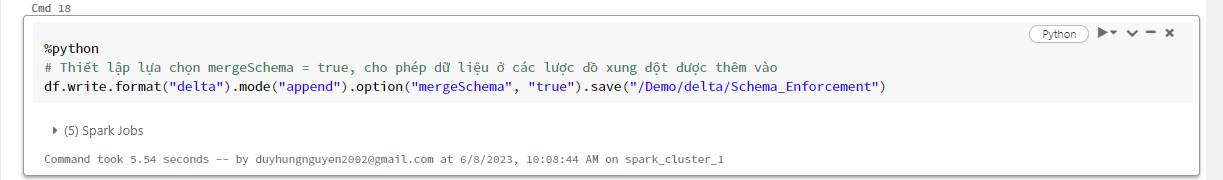
Hình 7.6: Ví dụ về tính năng thực thi lược đồ

Như ta có thể thấy đã có lỗi xảy ra khi tiến hành ghi hai dataframe có cấu trúc bảng khác nhau dưới định dạng delta. Đây chính là tính năng thực thi lược đồ, giải thích dễ hiểu là khi ta đã tiến hành ghi một dataframe vào một thư mực dưới định dạng delta, nếu như có một dataframe khác cũng được ghi vào thư mục đó mà có cấu trúc bảng khác thì Delta Lake sẽ từ chối việc ghi dữ liệu vào.

### Cải tiến lược đồ (Schema Evolution)

Tính năng này đề cập đến khả năng thích ứng với các lược đồ khác nhau theo thời gian, thông thường sẽ là cập nhật nội dung hay là thêm các cột bổ sung vào bảng. Nếu như tính năng thực thi lược đồ giúp cho chúng ta kiểm tra sự xung đột giữa các lược đồ của các dataframe, nhằm tránh gây ra việc gộp các lược đồ không mong muốn thì với Schema Evolution, ta sẽ có giải pháp khác phục vấn đề **AnalystException** trong hình 7.4.

Đầu tiên, ta có thể bỏ qua tính năng Schema Enforcement bằng cách thiết lập lựa chọn **mergeSchema** thành **true** để có thể ghi vào bảng delta dữ liệu của một lược đồ không khớp với lược đồ ban đầu:



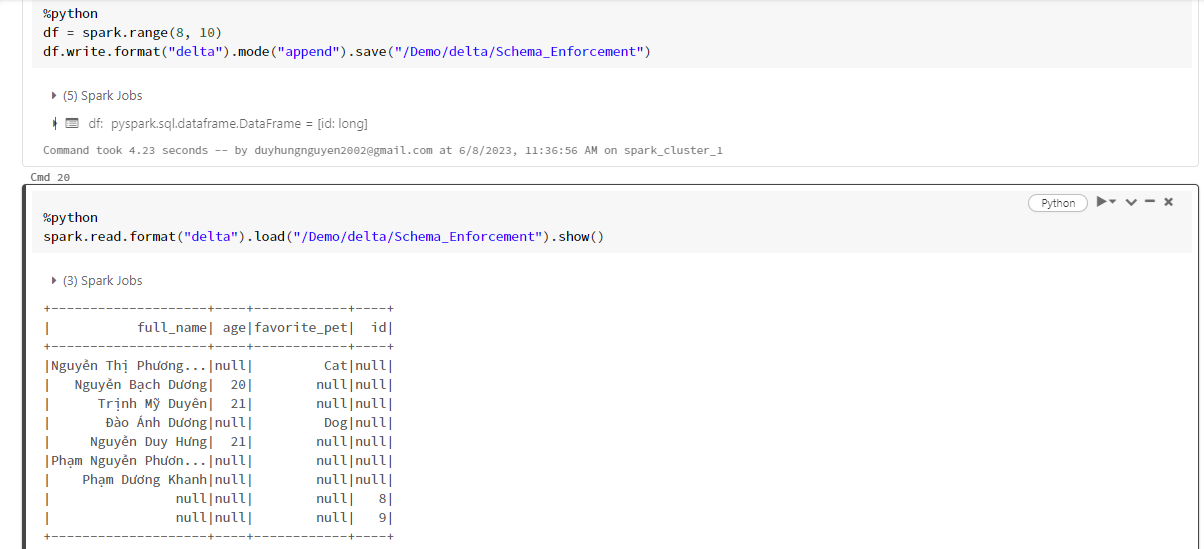
Và đây là nội dung trên bảng sau khi ta tiến hành thiết lập:



Như ta có thể thấy, hai dataframe có lược đồ khác nhau đã được gộp lại, và bảng delta hiện đã có ba cột trong khi trước đó chỉ có hai cột. Bên cạnh đó, ta hoàn toàn có thể cho phép mặc định tính năng này được thực hiện bằng cách thiết lập cài đặt **autoMerge** thành **true**:



Thiết lập này cho phép ta tránh việc phải cài đặt mergeSchema mỗi lần ta thêm dữ liệu và bảng, đồng thời cho phép thêm những dataframe có số lượng cột ít hơn so với bảng delta hiện tại.

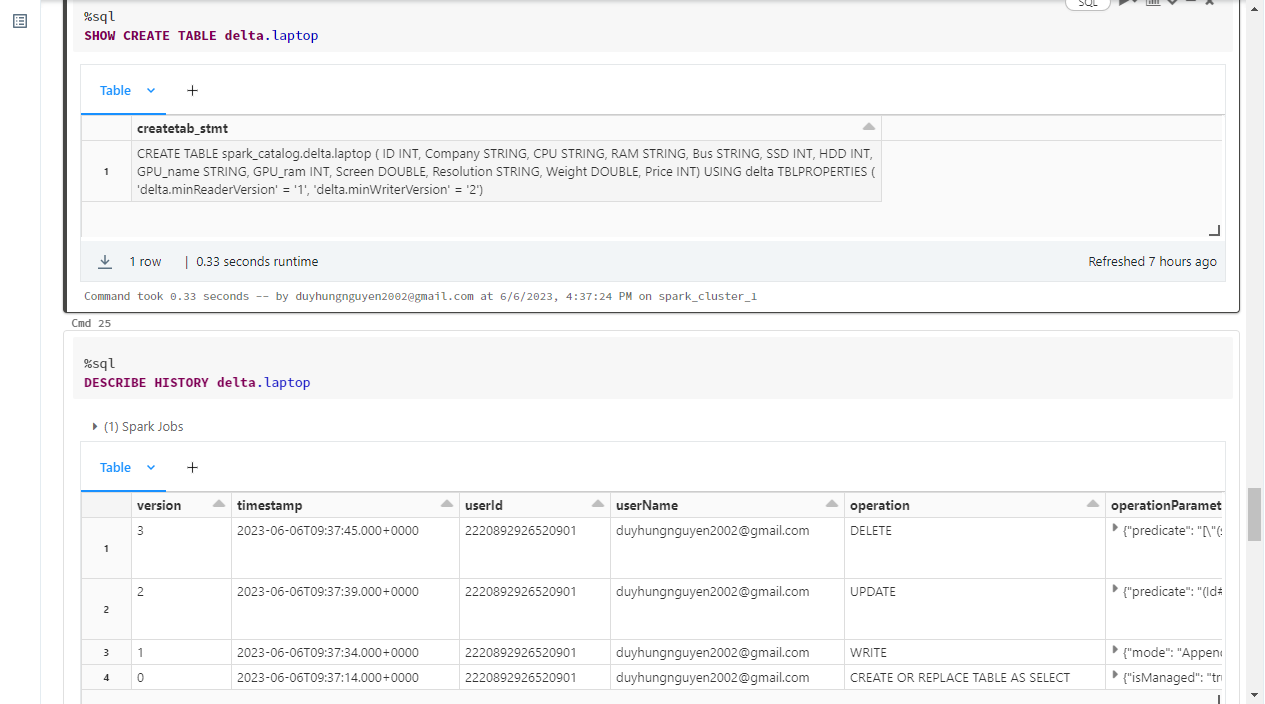


Hình 7.7: Schema Evolution cho phép thêm các dataframe khác nhau hoàn toàn về lược đồ

Kể cả với việc ta gộp một dataframe có các cột hoàn toàn khác với lược đồ trong bảng delta thì Schema Evolution cho phép điều đó mà không bị chồng chất dữ liệu. Một lý do khác mà giúp cho tính năng này hữu ích là khi cần phải thay đổi lược đồ của bảng mà không phải ghi lại toàn dữ liệu thì Schema Evolution cho phép điều đó.

## Audit History

Đây là một tính năng được chính Delta Lake công bố, với tính năng này ta có thể xem thông tin về việc tạo ra bảng, cấu hình cấu trúc của bảng, xem các thông tin về siêu dữ liệu. Bên cạnh đó ta còn có thể xem các mô tả về lịch sử của bảng, các lần sửa đổi, bao gồm phiên bản, thời gian, thông tin người tiến hành, loại thao tác… Như trong ảnh, ta có thể xem đang thông tin về bảng laptop, cũng như xem được các thông tin về siêu dữ liệu liên quan đến bảng ở mỗi lần thao tác dữ liệu trên bảng:



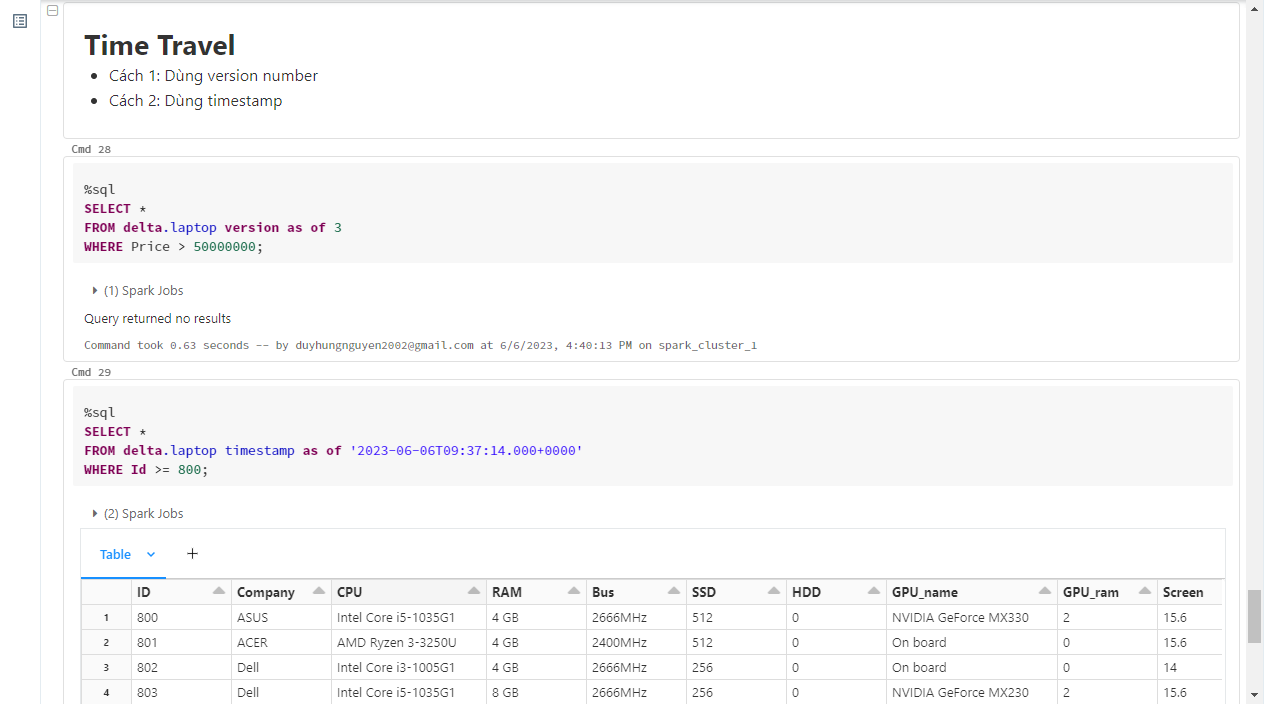
Hình 7.8: Tính năng Audit History trên Delta Lake

## Time Travel

Như đã đề cập ở chương 3, thông qua tính năng này ta có thể truy cập, truy vấn và phân tích dữ liệu ở các phiên bản cũ hơn, tại thời điểm bất kì trong quá khứ đã được ghi lại. Trong phần demo, ta tiếp tục tiến hành ba thao tác với bảng laptop như sau:



Giống với phần demo cho tính năng giao dịch ACID, khi ta tiến hành thao tác như trong hình thì sẽ tạo ra ba phiên bản mới của dataframe ứng với một thao tác. Ta có thể truy cập đến ba phiên bản này cũng như có thể quay lại (rollback) với phiên bản ban đầu bằng việc sử dụng số phiên bản (version number) hoặc là bằng dấu thời gian (timestamp):



Hình 7.9: Thực hiện tính năng Time Travel trong Data Lakehouse

Trên đây là toàn bộ phần demo hệ thống Data Lakehouse cơ bản, gồm tổng quan và một số tính năng nổi bật của kiến trúc này. Bên cạnh các tính năng được thể hiện rõ trong quá trình demo, ta còn có thể kể đến một số tính năng được thể hiện xuyên suốt đó là khả năng hỗ trợ các định dạng mở, khả năng lưu trữ và tính toán tách biệt. Có hai tính năng chưa được đề cập đến trong phần demo đó là khả năng hỗ trợ các dữ liệu phi cấu trúc và theo luồng, cùng với cả hỗ trợ BI. Trong thời gian tới, khi có nhiều điều kiện và thời gian hơn, em sẽ xây dựng một hệ thống Data Lakehouse hoàn thiện hơn, thể hiện rõ kiến trúc theo tầng và minh hoạ cụ thể hơn các tính năng của mô hình này. Phần demo vừa rồi, hy vọng có thể giúp mọi người có thể hình dung được dễ hiểu hơn về kiến trúc Data Lakehouse.

# Kết luận

Như vậy, ta đã đi qua bài báo cáo tìm hiểu về Data Lakehouse, xu hướng sử dụng kiến trúc này trong các hệ thống kho dữ liệu của doanh nghiệp. Trong khuôn khổ của bài báo cáo này, ta đã đi qua và cùng nhau phân tích 6 chương. Chương 1 mở đầu với lịch sử hình thành của Data Lakehouse, những thách thức và cơ hội đối với kiến trúc này trong việc giải quyết các nghiệp vụ của doanh nghiệp. Chương 2 là góc nhìn chi tiết về các tầng trong mô hình kiến trúc Data Lakehouse, ta đã tìm hiểu vai trò, chức năng của từng tầng trong mô hình đó đối với bài toán dữ liệu của doanh nghiệp. Đến với chương 3, ta đã tìm hiểu về những đặc điểm, tính năng nổi bật của Data Lakehouse giúp hỗ trợ tổ chức trong việc quản trị nguồn dữ liệu của mình và tận dụng tiềm năng từ dữ liệu hiện có. Việc so sánh Data Lakehouse với hai mô trình kho dữ liệu truyền thống Data Lake và Data Warehouse được tiến hành ở chương 4, sự so sánh này giúp cho người đọc và tổ chức có thể tìm ra kiến trúc hợp lý nhất đối với dữ liệu hiện có của họ. Tiếp theo, chúng ta tìm hiểu các công nghệ và công cụ được sử dụng phổ biến trong mô hình kiến trúc Data Lakehouse ở chương 5, qua đây ta có thể thấy được hệ sinh thái của dữ liệu lớn đang phát triển rất nhanh và hỗ trợ được hầu hết các nghiệp vụ khác nhau ở đa dạng ngành nghề. Sang đến chương 6, ta đã tìm hiểu về hai case-study đã ứng dụng thành công kiến trúc Data Lakehouse trong việc quản trị nền tảng dữ liệu của họ đó là GRAB và The Home Depot. Và cuối cùng, ta đã thông qua một demo hệ thống Data Lakehouse cơ bản để có thêm những góc nhìn về mô hình này, những tính năng cũng như các thành phần xây dựng nên hệ thống. [1]

Thông qua bài báo cáo này, người đọc đã có cái nhìn tổng quan về những khía cạnh quan trọng nhất của mô hình kiến trúc Data Lakehouse. Từ đó, người đọc sẽ được trang bị những kiến thức cơ bản về quá trình tạo ra một mô hình Data Lakehouse cho riêng [2] mình. Như đã đề cập, kiến trúc này mới chỉ được nhắc đến và đầu tư nghiên cứu trong vòng 3-4 năm trở lại đây. Tuy nhiên với sự phát triển quá nhanh của công nghệ, thì việc thông hiểu ở mức cơ bản về mô hình này là rất cần thiết đối với mọi doanh nghiệp trong tương lai, khi xu hướng đưa ra quyết định dựa trên dữ liệu đang chiếm đa số trong toàn bộ công việc diễn ra quanh chúng ta.

# THAM KHẢO

|  |  |
| --- | --- |
| [1] | O. E. KHENG, "Visitors to ASEAN countries," 2020. [Online]. Available: https://www.kaggle.com/datasets/ongengkheng/visitors-to-asean-countries-by-origin-countries. |
| [2] | SHEEPB, "Laptop Prices in Viet Nam 2020," 2020. [Online]. Available: https://www.kaggle.com/datasets/sheepb/laptop-prices-in-viet-nam-2020. |
| [3] | Bill Inmon, Mary Levins and Ranjeet Srivastava, Building the Data Lakehouse, Technics Publication, 2021. |
| [4] | P. Meno, Data Lakehouse in Action, Birmingham-Mumbai: Packt Publishing Ltd., 2022. |
| [5] | Praful Kava, Changbin Gong, "Build a Lake House Architecture on AWS," October 2022. [Online]. Available: https://aws.amazon.com/blogs/big-data/build-a-lake-house-architecture-on-aws/. |
| [6] | "Data Lakehouse: Concept, Key Features, and Architecture Layers," 10 November 2021. [Online]. Available: https://www.altexsoft.com/blog/data-lakehouse/. |
| [7] | Bill Inmon, Mary Levins, "Evolution to the Data Lakehouse," 19 May 2021. [Online]. Available: https://www.databricks.com/blog/2021/05/19/evolution-to-the-data-lakehouse.html. |
| [8] | J. Summan, "Delta Lake Architecture: A Bridge Between Data Lakes & Data Warehouses," 7 February 2023. [Online]. Available: https://streamsets.com/blog/delta-lake-architecture-bridge-between-data-lakes-data-warehouses/. |
| [9] | A. Chanthadavong, "zdnet," 17 November 2021. [Online]. Available: https://www.zdnet.com/article/grab-finds-sweet-spot-of-data-lake-and-data-warehouse-in-its-centralised-data-platform/. |
| [10] | Databricks, "Protecting 182 million wireless customers from harm," [Online]. Available: https://www.databricks.com/customers/att. |
| [11] | databricks, "What are ACID guarantees on Databricks?," 1 June 2023. [Online]. Available: https://docs.databricks.com/lakehouse/acid.html. |