TRƯỜNG ĐẠI HỌC KINH TẾ ĐÀ NẴNG

KHOA THƯƠNG MẠI ĐIỆN TỬ

---

# BÁO CÁO ĐỀ ÁN 2

## XÂY DỰNG PIPELINE CHO DỮ LIỆU CHỨNG KHOÁN REAL TIME CỦA BIG 3 NGÂN HÀNG VCB, CTG, BID TẠI THỊ TRƯỜNG VIỆT NAM

| | | |
|---|---|---|
| **Giáo viên hướng dẫn** | **:** | **ThS. Nguyễn Văn Chức** |
| **Lớp học phần** | **:** | **48K29.1** |
| **Nhóm sinh viên thực hiện** | **:** | **Nhóm 7** |
| **Thành viên** | **:** | **1. Nguyễn Lê Nhật Hồng** |
| | | **2. Nguyễn Minh Thái** |

*Đà Nẵng, tháng 12/2025*

# MỤC LỤC

## THÀNH VIÊN THỰC HIỆN BÁO CÁO

| Họ và tên sinh viên | Mã số sinh viên | Nội dung thực hiện | Đóng góp (%) |
|---|---|---|---|
| Nguyễn Lê Nhật Hồng | 221124029116 | Thu thập, xử lý và lưu trữ dữ liệu | 50% |
| Nguyễn Minh Thái | 221124029140 | Thu thập, lưu trữ và trực quan hóa dữ liệu | 50% |

# I. Bối cảnh nghiên cứu và tổng quan thị trường

## 1. Bối cảnh phát triển thị trường tài chính Việt Nam

Trong những năm gần đây, thị trường tài chính Việt Nam đã ghi nhận sự mở rộng đáng kể nhờ quá trình hội nhập kinh tế, cải cách chính sách và sự gia tăng của dòng vốn đầu tư trong và ngoài nước. Đặc biệt, thị trường chứng khoán đang ngày càng khẳng định vai trò là kênh huy động vốn trung và dài hạn quan trọng cho doanh nghiệp, thể hiện qua sự tăng trưởng về quy mô vốn hóa cũng như số lượng nhà đầu tư tham gia.

Song song với sự phát triển đó, xu hướng chuyển đổi số trong lĩnh vực tài chính trở nên rõ nét hơn. Các công nghệ như dữ liệu lớn (Big Data), phân tích tự động và xử lý thời gian thực đang dần thay thế phương pháp phân tích thủ công truyền thống. Điều này giúp các tổ chức tài chính nâng cao năng lực dự báo, quản trị rủi ro và tối ưu hóa quá trình ra quyết định. Vì vậy, nhu cầu khai thác và quản lý dữ liệu tài chính một cách hệ thống, chính xác và kịp thời đang trở thành yêu cầu cấp thiết trong bối cảnh hiện nay.

## 2. Tầm quan trọng của nhóm ngân hàng trong hệ thống tài chính

Trong cấu trúc tài chính Việt Nam,Khối ngân hàng thương mại giữ vai trò then chốt trong cấu trúc tài chính của Việt Nam, thể hiện qua các yếu tố:

- **Là nguồn cung vốn chủ đạo** cho nền kinh tế, chiếm tỷ trọng lớn trong tổng tín dụng toàn hệ thống.
- **Tác động lan tỏa mạnh** đến thị trường tiền tệ, trái phiếu và chứng khoán.
- **Độ nhạy cao với biến động kinh tế**, khiến nhóm cổ phiếu ngân hàng trở thành chỉ báo phản ánh sức khỏe tài chính quốc gia.
- **Thanh khoản giao dịch lớn**, thường dẫn dắt xu hướng thị trường và ảnh hưởng tới chỉ số VN-Index.

➔ Do đó, việc theo dõi và phân tích diễn biến giá cổ phiếu ngân hàng không chỉ mang ý nghĩa đối với nhà đầu tư mà còn hỗ trợ cơ quan quản lý và các tổ chức tài chính trong việc đánh giá rủi ro hệ thống và ổn định thị trường.

## 3. Khái quát về ba ngân hàng nghiên cứu

Dựa trên phạm vi nghiên cứu, đề tài tập trung vào ba ngân hàng thương mại nhà nước có quy mô lớn và mức độ ảnh hưởng cao trên thị trường chứng khoán Việt Nam:

- **Ngân hàng TMCP Ngoại thương Việt Nam (Vietcombank):** Vietcombank được đánh giá là một trong những ngân hàng có hiệu quả sinh lời tốt nhất trong hệ thống, nhờ sở hữu nền tảng khách hàng đa dạng và chất lượng tài sản ổn định. Với hệ số an toàn vốn cao và tỷ lệ nợ xấu thấp, VCB thường được xem là cổ phiếu mang tính ổn định, phù hợp với chiến lược đầu tư dài hạn.
- **Ngân hàng TMCP Công Thương Việt Nam (VietinBank):** VietinBank nằm trong nhóm ngân hàng có tổng tài sản lớn nhất cả nước và đóng vai trò trọng yếu trong tài trợ các lĩnh vực công nghiệp và thương mại. Giá cổ phiếu CTG thường biến động theo các yếu tố như tăng trưởng tín dụng, chính sách quản lý vốn và tiến độ xử lý nợ xấu, phản ánh rõ tác động của môi trường kinh tế vĩ mô.
- **Ngân hàng TMCP Đầu tư và Phát triển Việt Nam (BIDV):** BIDV là ngân hàng có lịch sử hoạt động lâu đời với mạng lưới rộng khắp và quy mô tín dụng lớn. Sự liên kết chặt chẽ với hoạt động đầu tư công khiến cổ phiếu BID trở thành chỉ báo đáng chú ý đối với chu kỳ tăng trưởng kinh tế. Biến động giá BID thường phản ánh kỳ vọng thị trường về cải thiện chất lượng tài sản và triển khai các chuẩn mực quản trị rủi ro.

## 4. Đặc điểm biến động và xu hướng giao dịch cổ phiếu

Nhìn chung, nhóm cổ phiếu ngân hàng, bao gồm VCB, CTG và BID, thể hiện một số đặc điểm nổi bật:

- Có tính chu kỳ cao, chịu ảnh hưởng mạnh từ tăng trưởng GDP, lãi suất điều hành và hạn mức tín dụng.
- Duy trì thanh khoản lớn và ổn định, thường dẫn dắt giá trị giao dịch toàn thị trường.

- Nhạy cảm với các thông tin vĩ mô như lạm phát, tỷ lệ nợ xấu và yêu cầu vốn theo Basel.
- Thường xuất hiện biến động ngắn hạn đáng kể trong các thời điểm điều chỉnh chính sách hoặc xuất hiện tin tức liên quan đến ngành ngân hàng.

➔ Chính vì vậy, việc xây dựng hệ thống thu thập và phân tích dữ liệu tự động là cần thiết nhằm theo dõi biến động giá theo thời gian thực, hỗ trợ đánh giá xu hướng một cách khách quan và tăng cường hiệu quả ra quyết định dựa trên dữ liệu.

## II. Vai trò của Data Engineering trong phân tích dữ liệu tài chính

Trong bối cảnh dữ liệu tài chính ngày càng đóng vai trò quan trọng trong hoạt động phân tích và dự báo, Data Engineering trở thành một yếu tố không thể thiếu để đảm bảo dữ liệu được thu thập, xử lý và khai thác một cách hiệu quả. Thay vì chỉ tập trung vào phân tích như trước đây, các tổ chức tài chính hiện nay chú trọng xây dựng nền tảng dữ liệu có khả năng vận hành liên tục, ổn định và đáp ứng nhu cầu theo thời gian thực.

### 1. Khung năng lực và nhiệm vụ cốt lõi của Data Engineer

Vai trò cốt lõi của Data Engineer trong hệ thống dữ liệu tài chính bao gồm:

- **Thiết kế kiến trúc và luồng dữ liệu** đảm bảo khả năng mở rộng, tính ổn định và hiệu suất cao. Điều này bao gồm lựa chọn mô hình xử lý dữ liệu tối ưu, xây dựng cấu trúc truyền tải ổn định và đảm bảo hệ thống có thể hoạt động liên tục ngay cả khi dữ liệu tăng đột biến - đặc biệt trong bối cảnh giao dịch tài chính thay đổi theo thời gian thực.
- **Xây dựng pipeline thu thập - xử lý - lưu trữ dữ liệu**, hỗ trợ cả mô hình batch và real-time streaming. Nhờ đó, dữ liệu được đưa vào hệ thống một cách tự động, hạn chế thao tác thủ công và đảm bảo thông tin luôn được cập nhật kịp thời.
- **Đảm bảo chất lượng dữ liệu** bao gồm phát hiện và xử lý dữ liệu thiếu, sai lệch hoặc trùng lặp; thiết lập quy tắc kiểm tra tính hợp lệ; đồng bộ hóa định dạng và

chuẩn hóa thời gian. Trong lĩnh vực tài chính, độ chính xác dữ liệu là yếu tố quyết định vì ảnh hưởng trực tiếp đến kết quả phân tích và đánh giá rủi ro.

- **Tự động hóa quy trình vận hành dữ liệu** giúp hệ thống hoạt động ổn định mà không phụ thuộc nhiều vào can thiệp thủ công. Việc tự động hóa còn giúp giảm thiểu rủi ro sai sót, tối ưu thời gian xử lý và nâng cao khả năng phản hồi khi nhu cầu phân tích gia tăng.
- **Quản trị dữ liệu** bao gồm phân quyền truy cập, bảo mật dữ liệu và theo dõi nguồn gốc thông tin. Những yếu tố này đặc biệt quan trọng trong môi trường ngân hàng, nơi dữ liệu phải tuân thủ nghiêm ngặt các tiêu chuẩn an toàn và quy định quản lý.

Như vậy, Data Engineer không chỉ là vị trí mang tính kỹ thuật mà còn là nền tảng giúp tổ chức xây dựng hệ sinh thái dữ liệu bền vững, đảm bảo dữ liệu tài chính được khai thác một cách nhất quán, chính xác và có giá trị lâu dài.

## 2. Đặc thù dữ liệu tài chính trong môi trường ngân hàng

Dữ liệu tài chính trong lĩnh vực ngân hàng mang tính chất phức tạp và yêu cầu mức độ chính xác cao hơn nhiều so với các loại dữ liệu thông thường. Một số đặc điểm nổi bật có thể kể đến:

- **Tính thời gian thực (real-time sensitivity):** Giá cổ phiếu và khối lượng giao dịch thay đổi liên tục trong suốt phiên giao dịch. Vì vậy, dữ liệu cần được cập nhật gần như ngay lập tức để phản ánh chính xác trạng thái thị trường tại từng thời điểm. Điều này đòi hỏi hệ thống thu thập và xử lý phải hoạt động ổn định với độ trễ thấp.
- **Phụ thuộc thị trường và độ biến động cao:** Dữ liệu cổ phiếu chịu ảnh hưởng trực tiếp từ các yếu tố vĩ mô và thông tin thị trường. Các thay đổi về chính sách tín dụng, lãi suất hay thông tin liên quan đến doanh nghiệp đều có thể khiến giá biến động đáng kể trong thời gian ngắn. Hệ thống dữ liệu cần khả năng thích ứng với biến động này để tránh gián đoạn hoặc sai lệch thông tin.

- **Yêu cầu về độ chính xác và tính toàn vẹn dữ liệu:** Dữ liệu tài chính yêu cầu mức độ chính xác tuyệt đối, vì chỉ một sai lệch nhỏ cũng có thể dẫn đến đánh giá không đúng hoặc rủi ro trong quá trình vận hành. Việc đảm bảo dữ liệu không bị trùng lặp, thiếu hụt hoặc sai lệch theo thời gian là nhiệm vụ trọng tâm của Data Engineering trong môi trường ngân hàng.

## 3. Ứng dụng Data Engineering trong giám sát biến động cổ phiếu

### Tự động hóa thu thập dữ liệu từ nguồn công khai

- Trong hệ thống được xây dựng, dữ liệu giá cổ phiếu của Vietcombank, VietinBank và BIDV được thu thập tự động từ Yahoo Finance thay vì sử dụng phương pháp truy xuất thủ công như trước đây. Quá trình này được thực hiện thông qua cơ chế lập lịch hoặc streaming liên tục, giúp hệ thống duy trì luồng dữ liệu ổn định và cập nhật theo từng thời điểm giao dịch.
- Việc tự động hóa không chỉ đảm bảo dữ liệu được thu thập đầy đủ và nhất quán, mà còn giảm thiểu rủi ro phát sinh từ thao tác con người, chẳng hạn như thiếu sót, nhập sai hoặc chậm trễ trong quá trình cập nhật. Ngoài ra, cơ chế thu thập tự động còn hỗ trợ khả năng mở rộng khi cần theo dõi thêm nhiều mã cổ phiếu hoặc tăng tần suất cập nhật mà không ảnh hưởng đến hiệu suất vận hành.
- Nhờ đó, hệ thống có thể phản ánh sát thực tế biến động thị trường, tạo nền tảng quan trọng cho quá trình xử lý và phân tích dữ liệu tiếp theo.

### Chuẩn hóa dữ liệu phục vụ phân tích định lượng

- Sau khi được thu thập, dữ liệu sẽ trải qua quy trình chuẩn hóa nhằm đảm bảo tính chính xác và đồng nhất trước khi đưa vào xử lý phân tích. Các bước chính bao gồm làm sạch dữ liệu, xử lý giá trị bị thiếu, loại bỏ dữ liệu trùng lặp và quy đổi định dạng thời gian về một chuẩn thống nhất. Đối với dữ liệu tài chính, việc đồng bộ hóa timestamp đặc biệt quan trọng vì chênh lệch múi giờ hoặc sai số thời gian có thể dẫn đến đánh giá sai lệch xu hướng thị trường.

- Bên cạnh đó, quá trình chuẩn hóa còn có thể bao gồm việc tính toán và bổ sung các chỉ số kỹ thuật như đường trung bình động (moving average), độ biến động (volatility) hoặc thay đổi phần trăm theo phiên. Những giá trị này giúp dữ liệu thô trở nên giàu thông tin hơn và phù hợp với các mô hình phân tích định lượng.
- Nhờ được xử lý theo quy trình chặt chẽ, dữ liệu đầu ra đảm bảo tính nhất quán và có thể được sử dụng trực tiếp cho việc phân tích xu hướng, so sánh biến động giữa các ngân hàng hoặc phục vụ các mô hình dự báo trong những bước triển khai tiếp theo.

**Hỗ trợ ra quyết định thông qua hệ thống trực quan hóa**

- Khi dữ liệu đã được xử lý và lưu trữ ổn định trong cơ sở dữ liệu, các công cụ trực quan hóa như Apache Superset được sử dụng để chuyển đổi dữ liệu thành biểu đồ, bảng thống kê và dashboard theo thời gian thực. Việc trình bày dữ liệu dưới dạng trực quan giúp người dùng dễ dàng theo dõi biến động giá cổ phiếu, nhận diện xu hướng và phát hiện các thay đổi bất thường mà không cần truy vấn thủ công.
- Ngoài khả năng quan sát dữ liệu theo thời gian, hệ thống trực quan còn hỗ trợ so sánh giữa các mã cổ phiếu, phân tích theo giai đoạn và lọc dữ liệu theo nhu cầu cụ thể. Điều này đặc biệt hữu ích đối với nhà đầu tư, bộ phận phân tích và các tổ chức tài chính khi cần đưa ra nhận định nhanh dựa trên thông tin chính xác và cập nhật.
- Thông qua việc kết hợp dữ liệu đã chuẩn hóa với công cụ trực quan hóa, hệ thống đóng vai trò hỗ trợ quá trình ra quyết định một cách chủ động, góp phần giảm thiểu rủi ro và nâng cao hiệu quả đánh giá thị trường.

## III. Phương pháp nghiên cứu và phạm vi triển khai

### 1. Mục tiêu nghiên cứu

Mục tiêu chính của đề tài là xây dựng một hệ thống có thể tự động thu thập, xử lý và lưu trữ dữ liệu giá cổ phiếu của ba ngân hàng Vietcombank, VietinBank và

BIDV từ Yahoo Finance. Hệ thống được thiết kế để hoạt động liên tục, hạn chế can thiệp thủ công và đảm bảo dữ liệu luôn được cập nhật một cách chính xác và kịp thời.

Bên cạnh đó, hệ thống cũng cần hỗ trợ phân tích dữ liệu theo hai dạng: dữ liệu thời gian thực và dữ liệu lịch sử. Luồng dữ liệu thời gian thực giúp theo dõi nhanh sự thay đổi giá trong phiên giao dịch, trong khi dữ liệu lịch sử lại hữu ích để nhìn tổng quan xu hướng dài hạn hoặc phục vụ các bài toán phân tích chuyên sâu hơn.

Cuối cùng, đề tài hướng đến việc tạo ra một giao diện trực quan thông qua Superset. Nhờ đó, người dùng có thể dễ dàng quan sát biến động giá, so sánh giữa các mã cổ phiếu và theo dõi thị trường một cách trực quan thay vì phải xử lý dữ liệu thô. Điều này giúp việc giám sát và ra quyết định trở nên nhanh chóng và hiệu quả hơn.

## 2. Phạm vi nghiên cứu

- Đối tượng nghiên cứu tập trung vào ba mã cổ phiếu ngân hàng có mức độ ảnh hưởng lớn trên thị trường chứng khoán Việt Nam: **VCB, CTG và BID**. Đây là nhóm cổ phiếu đại diện cho các ngân hàng thương mại nhà nước, có quy mô tài sản lớn và đóng vai trò quan trọng đối với hệ thống tài chính.
- Nguồn dữ liệu chính được sử dụng là **Yahoo Finance**, cung cấp cả dữ liệu lịch sử và dữ liệu gần thời gian thực. Yahoo Finance là một trong những nền tảng phổ biến nhất trong việc cung cấp dữ liệu thị trường và được sử dụng rộng rãi trong các dự án phân tích tài chính.

## 3. Phương pháp tiếp cận

Hệ thống được thiết kế dựa trên mô hình kiến trúc hướng sự kiện (event-driven architecture), trong đó mọi dữ liệu mới xuất hiện sẽ tự động kích hoạt các quy trình xử lý. Cụ thể, khi có thông tin giá cổ phiếu mới từ nguồn dữ liệu bên ngoài như Yahoo Finance, hệ thống sẽ ghi nhận sự kiện này và gửi qua Kafka. Spark Structured Streaming sẽ tiếp nhận và xử lý dữ liệu ngay lập tức, đảm bảo thông tin gần như thời gian thực.

Đồng thời, hệ thống áp dụng cơ chế streaming kết hợp micro-batch. Dữ liệu giá cổ phiếu được xử lý liên tục để các dashboard luôn cập nhật các giá trị mới nhất, trong khi các phép tính tổng hợp như giá trung bình, giá cao nhất, giá thấp nhất, độ biến động và khối lượng giao dịch trung bình được thực hiện theo các khoảng thời gian 10 phút. Cách tiếp cận này giúp cân bằng giữa việc cập nhật dữ liệu gần thời gian thực và việc tạo ra các chỉ số tổng hợp dễ sử dụng cho phân tích và trực quan hóa.

Nhờ mô hình này, hệ thống vừa phản ánh kịp thời biến động thị trường, vừa cung cấp dữ liệu tổng hợp đầy đủ để lưu trữ, phân tích và hiển thị trên các công cụ BI như Superset. Tổng thể, quá trình xử lý có thể mô tả theo chuỗi: dữ liệu mới -> sự kiện kích hoạt -> xử lý liên tục -> lưu trữ & tổng hợp, đảm bảo tính nhất quán, độ chính xác và khả năng mở rộng cho các phân tích sâu hơn.

## IV. Nền tảng kỹ thuật và công nghệ sử dụng

### 1. Nguồn dữ liệu

**Yahoo Finance** là một trong những nền tảng cung cấp dữ liệu tài chính được sử dụng phổ biến trên thế giới, đặc biệt trong các dự án nghiên cứu, phân tích thị trường hoặc xây dựng mô hình dữ liệu. Nền tảng này cung cấp đa dạng thông tin liên quan đến thị trường chứng khoán, bao gồm:

- Giá cổ phiếu theo thời gian thực hoặc gần thời gian thực
- Dữ liệu lịch sử với độ dài lên đến vài chục năm
- Khối lượng giao dịch theo ngày
- Dữ liệu giá mở cửa, đóng cửa, cao nhất, thấp nhất
- Các chỉ báo phân tích kỹ thuật cơ bản
- Thông tin doanh nghiệp, báo cáo tổng quan, tin tức cập nhật

**Đặc điểm nổi bật của Yahoo Finance:**

- **Dễ truy cập và hoàn toàn miễn phí:**

Một trong những ưu điểm lớn nhất của Yahoo Finance là khả năng truy cập đơn giản, không yêu cầu đăng ký tài khoản hoặc trả phí. Người dùng có thể lấy dữ liệu trực tiếp thông qua web hoặc thông qua thư viện yfinance mà không gặp các giới hạn chặt chẽ như nhiều API tài chính khác.

- **Cung cấp đầy đủ dữ liệu dạng time-series:**

Yahoo Finance cung cấp dữ liệu theo chuỗi thời gian - loại dữ liệu quan trọng nhất trong phân tích cổ phiếu. Đặc biệt, dữ liệu lịch sử trên Yahoo Finance có thể kéo dài hơn 20 năm, đủ cho các bài toán phân tích xu hướng dài hạn hoặc thử nghiệm mô hình phân tích kỹ thuật.

Với các mã cổ phiếu ngân hàng như VCB, CTG và BID, Yahoo Finance cung cấp đầy đủ dữ liệu cần thiết để quan sát biến động, so sánh giữa các ngân hàng và phục vụ các hệ thống phân tích định lượng.

- **Tính ổn định và cập nhật liên tục:**

Yahoo Finance được nhiều nhà đầu tư cá nhân và các tổ chức tài chính nhỏ sử dụng hằng ngày. Do đó, nền tảng này thường xuyên được cập nhật và duy trì ổn định. Các dữ liệu giá trong phiên có thể được cập nhật theo từng phút, từng giờ hoặc sau khi thị trường đóng cửa.

- **Dễ tích hợp với các công cụ xử lý dữ liệu:**

Thư viện yfinance giúp việc lấy dữ liệu từ Yahoo Finance trở nên cực kỳ đơn giản. Thay vì phải gửi request API dạng JSON phức tạp, người dùng chỉ cần gọi vài dòng mã Python. Dữ liệu trả về ở dạng Pandas DataFrame. Khả năng tích hợp này giúp giảm đáng kể thời gian xây dựng hệ thống so với việc sử dụng API thương mại yêu cầu xử lý phản hồi phức tạp.

- **Không yêu cầu hạ tầng phức tạp để kết nối:**

Một điểm mạnh quan trọng khác là Yahoo Finance không yêu cầu API key, không yêu cầu xác thực, không giới hạn máy chủ gọi dữ liệu. Điều này rất phù hợp với hệ thống cần thu thập dữ liệu tự động qua Kafka vì:

- Producer có thể gọi dữ liệu định kỳ mà không lo bị chặn
- Không cần thiết lập cấu hình xác thực
- Không tốn chi phí duy trì API trả phí
- Pipeline dễ triển khai và đơn giản hóa quy trình vận hành

So với các nguồn dữ liệu tài chính chuyên nghiệp như Bloomberg API, Alpha Vantage hoặc Twelve Data - vốn yêu cầu API key, quản lý hạn mức truy cập và đôi khi phát sinh chi phí - Yahoo Finance mang lại sự linh hoạt và giảm thiểu rủi ro vận hành đối với dự án nghiên cứu.

- **Định dạng dữ liệu chuẩn và dễ mở rộng:**

Yahoo Finance cung cấp dữ liệu theo chuẩn quốc tế, hỗ trợ: CSV, JSON, giao diện web, truy xuất qua thư viện, …

Nhờ đó, hệ thống có thể dễ dàng mở rộng sang: Thêm mã cổ phiếu mới, theo dõi chỉ số VN-Index, phân tích ETF hoặc các nhóm ngành khác, kết hợp với dữ liệu tài chính doanh nghiệp, …

## 2. Ghi nhận dữ liệu

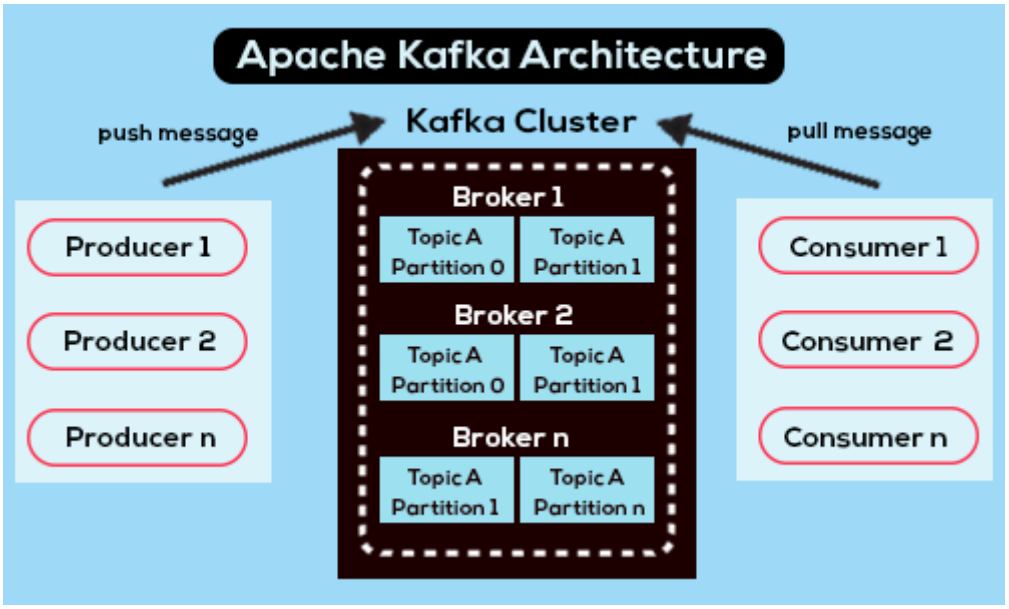
### 2.1. Giới thiệu về Apache Kafka

Apache Kafka là một nền tảng xử lý luồng dữ liệu phân tán được thiết kế để thu thập, truyền tải và lưu trữ dữ liệu với tốc độ cao. Kafka thường được xem như một "hệ thống trung gian truyền tải dữ liệu" cho phép các ứng dụng gửi và nhận dữ liệu theo thời gian thực.

Kafka được sử dụng rộng rãi trong nhiều lĩnh vực như tài chính, viễn thông, thương mại điện tử, và IoT nhờ khả năng xử lý hàng triệu sự kiện mỗi giây với độ trễ

rất thấp. Một số tập đoàn lớn như LinkedIn, Uber, Netflix, Airbnb đều sử dụng Kafka trong hạ tầng dữ liệu của họ.

### 2.2. Các thành phần chính của Kafka




**Broker**

- Broker là thành phần đóng vai trò là máy chủ của Kafka, chịu trách nhiệm tiếp nhận, lưu trữ và phân phối dữ liệu cho các consumer. Một cụm Kafka có thể bao gồm từ 1 đến hàng chục broker nhằm tăng sức chứa dữ liệu, nâng cao hiệu suất xử lý và đảm bảo tính sẵn sàng khi xảy ra lỗi.
- Mỗi broker có thể lưu trữ nhiều topic và nhiều partition. Trong hệ thống phân tán, Kafka sử dụng cơ chế replication để sao chép dữ liệu giữa các broker, giúp giảm nguy cơ mất dữ liệu khi có lỗi xảy ra.

**Topic**

- Topic là không gian logic để phân loại và lưu trữ dữ liệu theo chủ đề. Mỗi topic có thể được xem như một "kênh dữ liệu", nơi producer ghi dữ liệu vào và consumer đọc dữ liệu ra.
- Đặc điểm của topic:
    - Một topic có thể có nhiều partition.

    - Consumer có thể đọc từ topic theo offset.
    - Dữ liệu trong topic được lưu dưới dạng log tuần tự.
- Topic giúp Kafka tổ chức dữ liệu linh hoạt và dễ mở rộng theo từng nhóm thông tin khác nhau.

**Producer**

- Producer là thành phần gửi dữ liệu vào Kafka. Nhiệm vụ chính của producer là tạo ra các messages và gửi chúng vào topic tương ứng.
- Đặc điểm của producer:
    - Có thể gửi dữ liệu theo từng bản ghi hoặc theo batch.
    - Tự động lựa chọn partition để ghi dữ liệu vào (dựa trên key hoặc vòng tròn phân phối).
    - Hỗ trợ cơ chế retry để đảm bảo message không bị mất khi gặp sự cố.
- Producer hoạt động hoàn toàn độc lập với consumer, nhờ đó hệ thống trở nên linh hoạt và dễ mở rộng.

**Consumer**

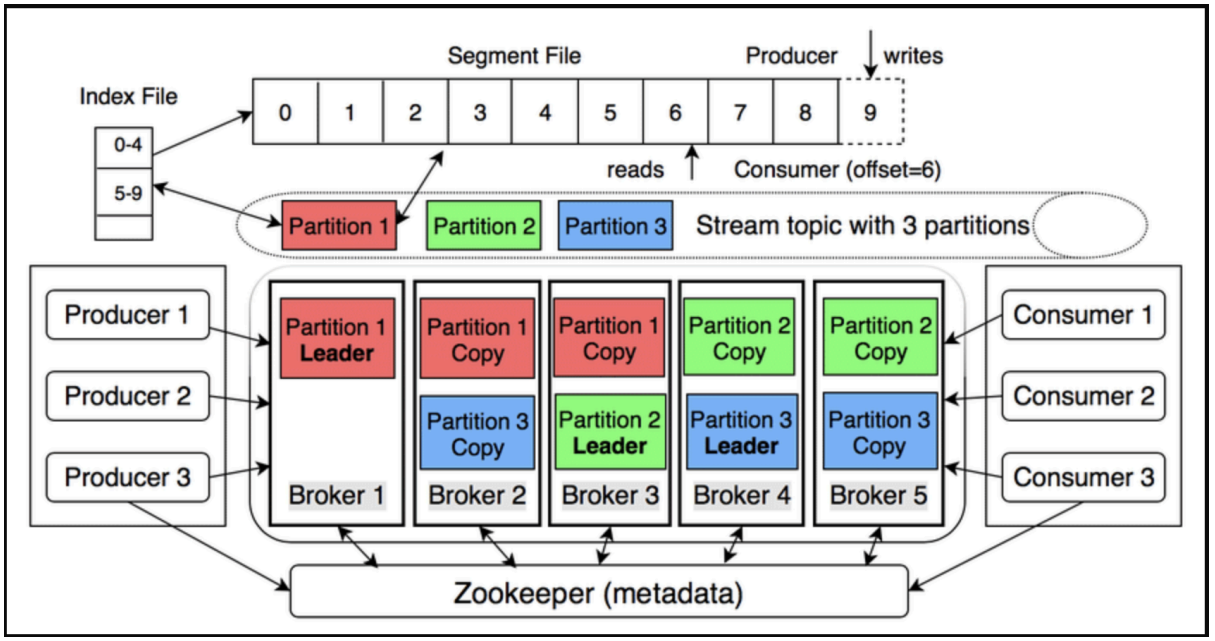
- Consumer là thành phần đọc dữ liệu từ Kafka. Các consumer có thể thuộc nhiều nhóm khác nhau, cho phép phân chia tải và đảm bảo dữ liệu được xử lý song song.
- Đặc điểm của consumer:
    - Lưu lại vị trí đọc dữ liệu thông qua offset.
    - Consumer group giúp nhiều consumer cùng đọc dữ liệu mà không bị trùng lặp.
    - Hoạt động độc lập với producer, giúp tách biệt lớp thu nhận và lớp xử lý dữ liệu.
- Consumer có thể là hệ thống xử lý dữ liệu, một dịch vụ lưu trữ, hoặc công cụ phân tích.

**Partition**

- Partition là đơn vị lưu trữ nhỏ của một topic. Mỗi topic có thể chia thành nhiều partition, và mỗi partition được lưu trên các broker khác nhau để tăng tốc độ và độ an toàn của hệ thống.
- Ưu điểm của partition:
    - Tăng khả năng xử lý song song.
    - Giúp Kafka mở rộng theo chiều ngang.
    - Hỗ trợ cơ chế replication giúp dữ liệu được sao lưu và đảm bảo bền vững.
- Kafka luôn duy trì thứ tự các bản ghi trong từng partition, điều này rất quan trọng đối với các ứng dụng xử lý dữ liệu theo dòng thời gian.

### 2.3. Cách hoạt động của Kafka

Apache Kafka hoạt động dựa trên mô hình publish–subscribe, trong đó dữ liệu được gửi vào hệ thống bởi các thành phần producer và được các consumer đọc ra theo nhu cầu. Cơ chế này giúp Kafka xử lý luồng dữ liệu lớn theo thời gian thực với độ trễ rất thấp.




- **B1: Producer gửi dữ liệu vào Kafka.** Producer là thành phần chịu trách nhiệm tạo ra dữ liệu và gửi dữ liệu đó vào các topic trong Kafka. Producer có thể gửi dữ liệu theo từng bản ghi hoặc theo lô, theo từng khoảng thời gian cố

định. Kafka hỗ trợ cơ chế retry khi gửi thất bại, lựa chọn partition để ghi dữ liệu, đảm bảo rằng dữ liệu được gửi theo đúng thứ tự đối với từng partition. Nhờ đó, dữ liệu luôn được đưa vào hệ thống liên tục và đáng tin cậy.

- **B2: Kafka lưu trữ dữ liệu theo dạng log tuần tự.** Khi nhận dữ liệu từ producer, Kafka lưu các bản ghi vào từng partition của topic. Partition hoạt động như một log tệp tuần tự, trong đó mỗi bản ghi mới sẽ được ghi vào cuối file. Mỗi bản ghi trong partition được gán một vị trí gọi là offset. Offset giúp Kafka xác định thứ tự bản ghi, kiểm soát tiến trình đọc dữ liệu của consumer và cho phép truy xuất lại bất kỳ vị trí nào khi cần. Việc lưu trữ tuần tự giúp Kafka đạt hiệu suất ghi rất cao so với các hệ thống truyền thống.
- **B3: Consumer đọc dữ liệu từ topic theo nhu cầu.** Consumer là thành phần đăng ký đọc dữ liệu từ topic. Consumer có thể đọc liên tục theo thời gian thực, đọc theo nhóm, hoặc đọc lại từ vị trí cũ khi cần. Kafka không điều khiển tiến trình đọc của consumer; thay vào đó, consumer tự quản lý offset của mình. Điều này giúp mỗi nhóm consumer nhận dữ liệu theo kịch bản riêng, tăng tính linh hoạt trong xử lý dữ liệu và hỗ trợ chia tải dễ dàng.
- **B4: Kafka không xóa dữ liệu ngay khi consumer đọc.** Một đặc điểm quan trọng của Kafka là dữ liệu không bị xóa ngay sau khi được tiêu thụ. Dữ liệu được giữ trong topic theo thời gian được cấu hình, gọi là retention time, có thể là: vài giờ, vài ngày hoặc vài tuần tùy hệ thống. Điều này mang lại nhiều lợi ích: Đọc lại dữ liệu khi cần, phục hồi sau sự cố và dễ dàng thêm consumer mới.

### 2.4. Ưu nhược điểm của Kafka:

| Ưu điểm | Nhược điểm |
|---|---|
| Xử lý dữ liệu tốc độ cao, hỗ trợ streaming theo thời gian thực. | Cài đặt và cấu hình phức tạp, đòi hỏi hiểu biết về hệ thống phân tán. |
| Khả năng mở rộng linh hoạt, dễ thêm broker và partition khi lượng dữ liệu tăng. | Yêu cầu tài nguyên lớn, không phù hợp cho hệ thống nhỏ hoặc dữ liệu ít. |

| | |
|---|---|
| Độ tin cậy cao nhờ cơ chế sao chép đảm bảo không mất dữ liệu. | Cần giám sát liên tục, nếu không dễ xảy ra lỗi đầy log, consumer lag. |
| Cho phép đọc lại dữ liệu nhờ lưu trữ theo thời gian. | Không đảm bảo thứ tự toàn bộ topic, chỉ đảm bảo thứ tự trong một partition. |

# 3. Xử lý dữ liệu

## 3.1. Giới thiệu về Apache Spark

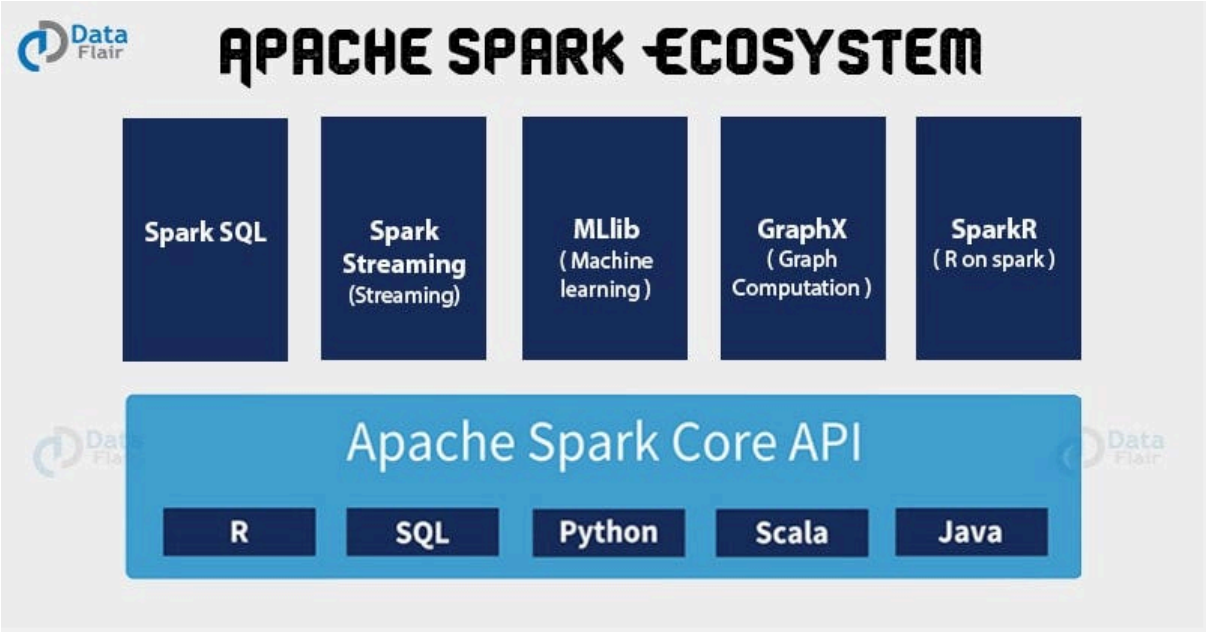
Apache Spark là nền tảng phân tích dữ liệu lớn được thiết kế để xử lý dữ liệu với tốc độ cao và khả năng mở rộng mạnh. Spark hỗ trợ cả hai mô hình xử lý:

- **Batch processing** (xử lý theo lô, dữ liệu lịch sử)
- **Streaming processing** (xử lý liên tục, dữ liệu thời gian thực)

Spark được xây dựng dựa trên bộ nhớ RAM thay vì đọc/ghi liên tục từ ổ đĩa như Hadoop MapReduce, nhờ vậy tốc độ xử lý nhanh hơn gấp hàng chục lần trong nhiều trường hợp.

## 3.2. Các thành phần chính của Spark

Hệ sinh thái Apache Spark được xây dựng từ nhiều module chức năng, cho phép xử lý dữ liệu theo nhiều mục đích khác nhau: phân tích SQL, xử lý streaming, học máy và phân tích đồ thị.




**Spark SQL**

Spark SQL là module được sử dụng phổ biến nhất trong hệ sinh thái Spark, cho phép xử lý dữ liệu có cấu trúc và bán cấu trúc dưới dạng bảng. Thay vì viết các phép biến đổi phức tạp bằng API truyền thống, người dùng có thể truy vấn dữ liệu bằng ngôn ngữ SQL chuẩn, rất quen thuộc trong phân tích dữ liệu.

Spark SQL hỗ trợ ba cơ chế quan trọng:

- **DataFrame API:** cho phép xử lý dữ liệu theo dạng bảng với cú pháp gần giống Python Pandas, nhưng ở quy mô lớn và phân tán.
- **Dataset API:** cung cấp kiểu dữ liệu có cấu trúc mạnh, đặc biệt hữu ích khi làm việc với ngôn ngữ Scala và Java.
- **Catalyst Optimizer:** trình tối ưu hóa truy vấn mạnh mẽ của Spark, tự động tối ưu hóa kế hoạch thực thi để tăng hiệu suất xử lý.

**Spark Streaming**

Spark Streaming là module hỗ trợ xử lý dữ liệu thời gian thực, cho phép hệ thống phản ứng nhanh với các luồng dữ liệu liên tục. Trong các phiên bản mới, Spark phát triển Structured Streaming, một công cụ xử lý streaming hiện đại hơn với API thống nhất với DataFrame.

Structured Streaming hoạt động dựa trên hai mô hình:

- Micro-batch: chia dữ liệu streaming thành các lô nhỏ và xử lý liên tục theo chu kỳ.
- Continuous Processing: cho phép xử lý gần như theo từng sự kiện, đáp ứng yêu cầu thời gian thực nghiêm ngặt.

Structured Streaming có khả năng kết nối với nhiều nguồn dữ liệu như Kafka, socket hoặc cloud storage, giúp dễ dàng xây dựng pipeline streaming có độ linh hoạt cao.

**MLlib**

MLlib là thư viện học máy tích hợp sẵn trong Spark, hỗ trợ triển khai các thuật toán học máy trên quy mô dữ liệu lớn. Không giống các thư viện chạy trên một máy đơn lẻ, MLlib được thiết kế để chạy phân tán trên cluster, cho phép xử lý dữ liệu hàng triệu bản ghi một cách hiệu quả.

Một số chức năng chính của MLlib:

- Hỗ trợ nhiều thuật toán học máy như phân loại, hồi quy, phân cụm
- Cung cấp các công cụ tiền xử lý dữ liệu như chuẩn hóa, chuyển đổi đặc trưng, scaling, …
- Xây dựng pipeline học máy, cho phép kết hợp nhiều bước từ feature engineering -> training -> evaluation theo quy trình thống nhất

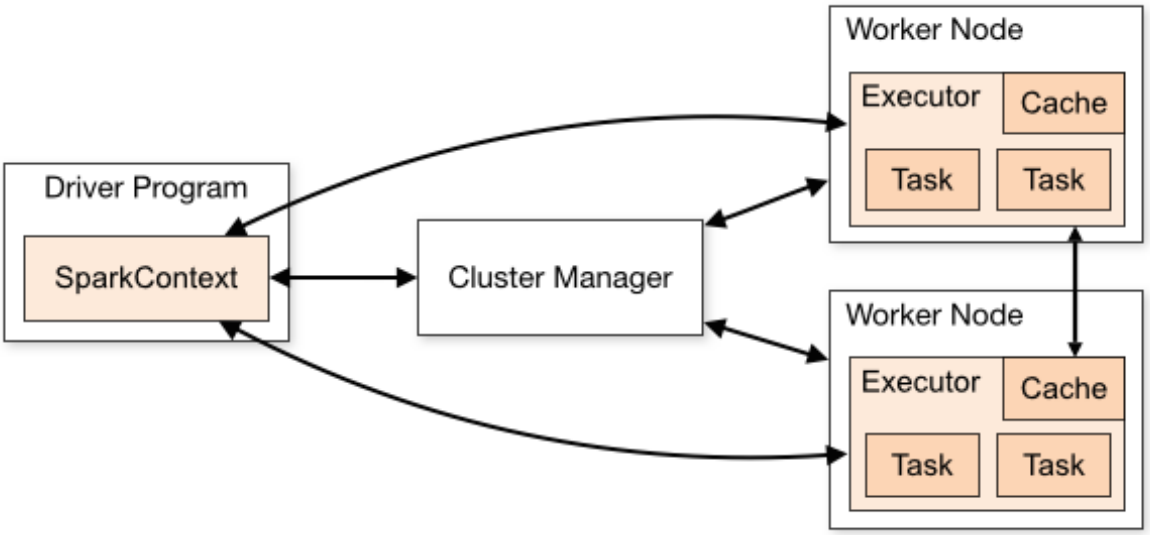
**GraphX**

GraphX là thư viện xử lý đồ thị của Spark, cho phép phân tích các dữ liệu có quan hệ mạng lưới phức tạp. Module này cung cấp mô hình lập trình đồ thị phân tán, hỗ trợ tính toán hiệu quả trên các tập dữ liệu dạng graph lớn.

GraphX hỗ trợ nhiều thuật toán phổ biến: PageRank, Connected Components, Community Detection, Shortest Path, …

### 3.3. Kiến trúc vận hành của Spark

Apache Spark được xây dựng theo kiến trúc phân tán, cho phép xử lý dữ liệu song song trên nhiều node trong cụm. Để thực hiện điều này, Spark sử dụng nhiều thành phần khác nhau phối hợp với nhau nhằm đảm bảo tốc độ, khả năng mở rộng và tính ổn định khi vận hành.




**Driver Program**

Driver Program là trung tâm điều phối của một ứng dụng Spark. Đây là thành phần chịu trách nhiệm:

- Khởi tạo ứng dụng Spark
- Quản lý luồng xử lý chính
- Phân chia nhiệm vụ cho các executor
- Giám sát tiến trình chạy
- Xử lý lỗi, retry và tổng hợp kết quả cuối cùng

Driver lưu trữ thông tin về cách dữ liệu sẽ được xử lý thông qua Directed Acyclic Graph. DAG Scheduler sẽ tối ưu hóa các bước tính toán trước khi gửi tác vụ xuống các executor. Driver đóng vai trò "bộ não" của hệ thống Spark. Nếu driver gặp sự cố, quá trình chạy Spark job sẽ bị gián đoạn.

**Cluster Manager**

Cluster Manager là thành phần chịu trách nhiệm phân bổ tài nguyên cho ứng dụng Spark, bao gồm CPU, RAM và container thực thi. Spark hỗ trợ nhiều loại cluster manager:

- Standalone Cluster: Cluster manager mặc định của Spark, dễ triển khai và phù hợp cho môi trường nhỏ và trung bình.
- YARN (Hadoop): Được sử dụng phổ biến trong hệ sinh thái Hadoop, cho phép Spark chạy chung với các ứng dụng Big Data khác.
- Kubernetes: Ngày càng phổ biến nhờ khả năng mở rộng linh hoạt, dễ triển khai và tự động hóa trên container.

Cluster Manager sẽ quyết định cấp bao nhiêu executor, sử dụng bao nhiêu tài nguyên, gán workload cho worker node nào.

**Executors**

Executors là các tiến trình chạy trên các worker node, thực thi các task được gửi từ driver. Mỗi executor:

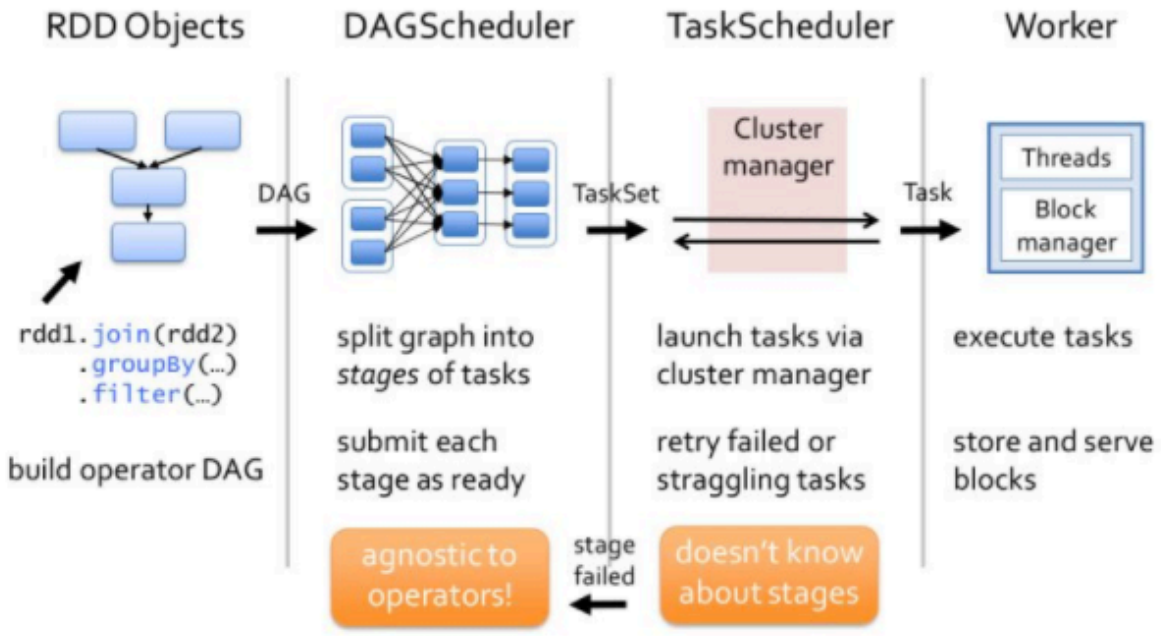
- Xử lý song song một phần dữ liệu
- Lưu cache để tăng tốc độ
- Ghi log và gửi kết quả ngược lại driver

Executors quyết định hiệu suất xử lý của Spark, càng nhiều executor thì khả năng xử lý song song càng cao.

**Structured Streaming Engine**

Là engine xử lý luồng dữ liệu thời gian thực của Spark. Nó cho phép hệ thống đọc dữ liệu từ Kafka hoặc các nguồn streaming, xử lý dữ liệu theo micro-batches hoặc continuous mode, quản lý checkpoint và offset tự động, đảm bảo cơ chế khôi phục khi lỗi. Structured Streaming giúp Spark hoạt động như hệ thống streaming real-time mạnh mẽ.

**Cơ chế hoạt động:**




- **B1: Tạo Logical Plan.** Khi người dùng viết code Spark, Spark không chạy ngay mà xây dựng một biểu diễn logic gọi là Logical Plan. Logical plan mô tả nguồn dữ liệu, các phép biến đổi, các điều kiện lọc, group-by, join, …
- **B2: Tối ưu hóa với Catalyst Optimizer.** Catalyst Optimizer là bộ tối ưu hóa mạnh mẽ của Spark SQL, giúp loại bỏ bước tính dư thừa, tối ưu thứ tự join, phân rã biểu thức phức tạp và chọn chiến lược thực thi hiệu quả nhất. Sau bước này, Spark tạo ra Physical Plan.
- **B3: Chia thành Stages và Tasks:** Spark phân tích Physical Plan và chia thành:
    - Stages: Tập hợp nhiều task có thể chạy song song
    - Tasks: Đơn vị xử lý nhỏ nhất, thực thi trên từng executor

Các stage được chạy theo thứ tự phụ thuộc, còn tasks trong một stage chạy song song.

- **B4: Thực thi song song trên Executors:** Driver gửi tasks xuống Executor, Executor xử lý dữ liệu và trả kết quả.

Một điểm mạnh của Spark là khả năng tự phục hồi khi xảy ra lỗi. Nhờ cơ chế lineage theo dõi nguồn gốc và các bước biến đổi dữ liệu, Spark có thể dễ dàng tính

toán lại các phần dữ liệu bị mất mà không cần lưu trữ toàn bộ kết quả trung gian. Với Structured Streaming, Spark còn hỗ trợ checkpoint và quản lý offset giúp khôi phục chính xác vị trí xử lý trong trường hợp hệ thống gián đoạn.

Nhờ kiến trúc và cơ chế hoạt động như vậy, Spark có thể xử lý nhiều loại dữ liệu - từ batch đến streaming - với tốc độ cao, độ tin cậy tốt và khả năng mở rộng mạnh mẽ.

### 3.4. Ưu và nhược điểm của Apache Spark:

| Ưu điểm | Nhược điểm |
|---|---|
| Tốc độ xử lý cao nhờ kiến trúc in-memory, nhanh hơn nhiều so với MapReduce. | Tiêu tốn nhiều RAM, cần hệ thống có cấu hình mạnh để đạt hiệu năng tối ưu. |
| Hỗ trợ cả batch processing và streaming processing trong cùng một framework. | Cấu hình và tối ưu hiệu năng phức tạp, đòi hỏi kinh nghiệm triển khai hệ phân tán. |
| Khả năng mở rộng mạnh, dễ chạy trên nhiều node và thích hợp với dữ liệu lớn. | Không phù hợp cho workload nhỏ, có thể gây lãng phí tài nguyên. |
| Cung cấp API phong phú: SQL, DataFrame, Machine Learning (MLlib), GraphX. | Học máy trong MLlib chưa mạnh bằng các thư viện chuyên dụng như TensorFlow, Scikit-Learn. |
| Tích hợp tốt với các hệ thống lớn như Kafka, Hadoop, PostgreSQL, Delta Lake,… | Debug trên môi trường phân tán khó khăn hơn so với các ứng dụng chạy đơn lẻ. |

## 4. Lưu trữ và quản trị dữ liệu - PostgreSQL

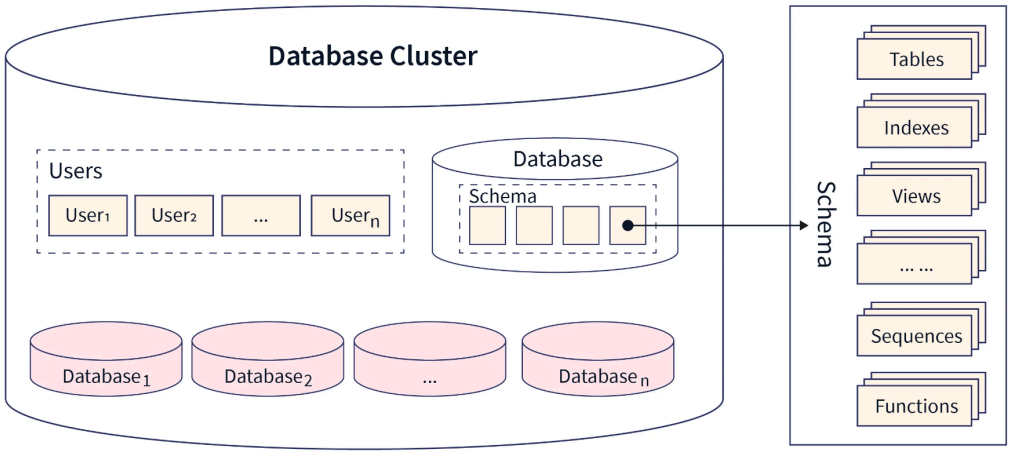
### 4.1. Giới thiệu về PostgreSQL

PostgreSQL là hệ quản trị cơ sở dữ liệu mã nguồn mở mạnh mẽ, được sử dụng rộng rãi trong các hệ thống phân tích dữ liệu và backend hiện đại. Với độ ổn định cao, khả năng xử lý dữ liệu lớn và hỗ trợ nhiều kiểu dữ liệu linh hoạt, PostgreSQL đặc biệt

phù hợp cho các dự án đòi hỏi sự chính xác và an toàn dữ liệu - trong đó có hệ thống lưu trữ dữ liệu giá cổ phiếu theo thời gian thực.

### 4.2. Cấu trúc lưu trữ trong PostgreSQL

PostgreSQL sử dụng một cấu trúc lưu trữ được thiết kế tối ưu cho việc quản lý dữ liệu lớn, đảm bảo hiệu năng truy vấn và tính toàn vẹn của cơ sở dữ liệu. Dữ liệu trên đĩa được tổ chức theo nhiều lớp khác nhau, nhằm giúp hệ thống dễ dàng truy xuất, cập nhật và phục hồi khi cần.




**Tables**

Tables là nơi lưu trữ dữ liệu ở dạng bảng, gồm các dòng và cột. Mỗi bảng trong PostgreSQL được lưu thành một file hoặc nhiều file, không sắp xếp theo bất kỳ thứ tự cụ thể nào trừ khi người dùng áp dụng index hay cơ chế sắp xếp khác.

**Pages**

Pages là đơn vị lưu trữ cơ bản và nhỏ nhất trong PostgreSQL, có kích thước mặc định là 8 KB, chứa nhiều dòng dữ liệu của một bảng hoặc index. Khi PostgreSQL đọc dữ liệu, nó thao tác theo page chứ không đọc từng dòng riêng lẻ giúp tăng tốc độ truy xuất.

**Heap Files**

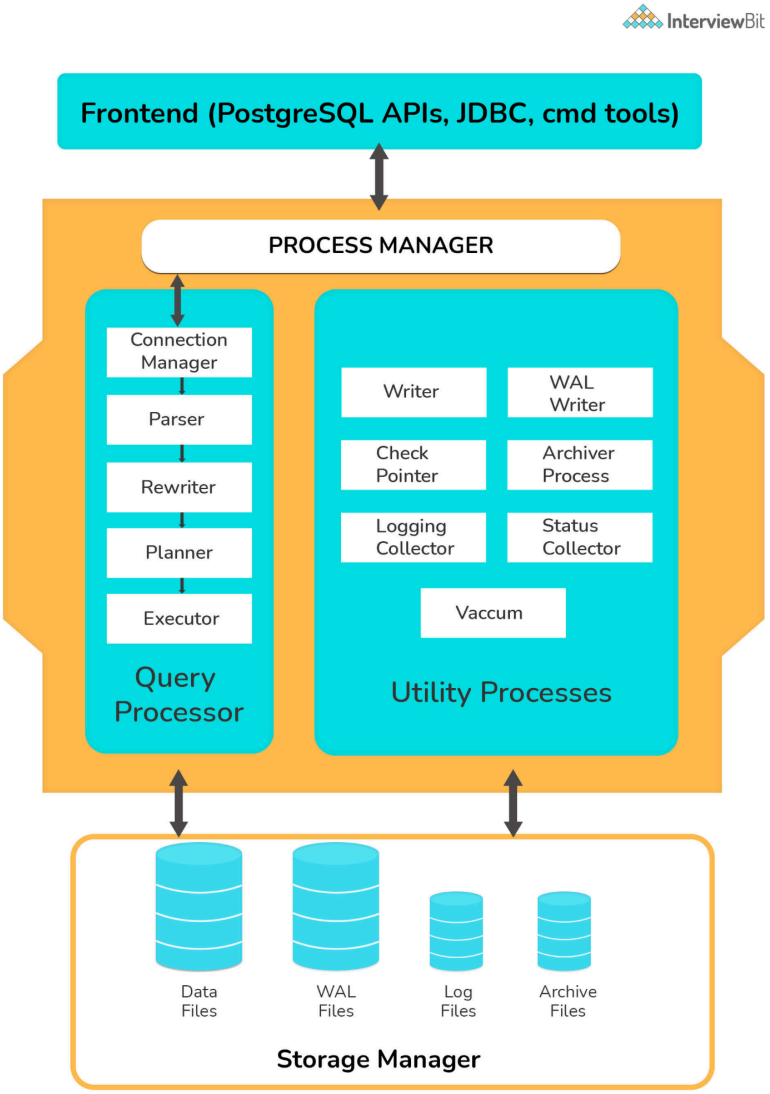
Heap Files là các file lưu trữ dữ liệu bảng theo dạng "heap" - nghĩa là không có thứ tự sắp xếp cố định. Mỗi heap file gồm nhiều page, và mỗi page chứa nhiều tuple. Thiết kế này giúp PostgreSQL dễ dàng chèn dữ liệu mới mà không cần sắp xếp lại toàn bộ bảng, giảm chi phí ghi và hệ thống có thể xử lý lượng lớn giao dịch nhanh chóng.

**Index Files**

Index Files chứa các cấu trúc index như B-Tree, Hash, GIN hoặc GiST. Các Index này giúp tăng tốc truy vấn bằng cách cho phép PostgreSQL định vị nhanh vị trí dữ liệu thay vì quét toàn bộ bảng. Khi bảng lớn dần, index đóng vai trò quan trọng trong việc duy trì hiệu năng truy vấn.

### 4.3. Kiến trúc vận hành của PostgreSQL

Kiến trúc của PostgreSQL được xây dựng theo mô hình client–server, trong đó các thành phần phía server chịu trách nhiệm quản lý dữ liệu, xử lý truy vấn và đảm bảo tính nhất quán, còn phía client đóng vai trò gửi yêu cầu và nhận kết quả. Nhờ kiến trúc rõ ràng và phân tách nhiệm vụ, PostgreSQL có thể hoạt động ổn định trong môi trường nhiều người dùng, đảm bảo hiệu năng và độ tin cậy cao.




**PostgreSQL Server**

PostgreSQL Server chịu trách nhiệm quản lý dữ liệu và xử lý các yêu cầu từ client. Thành phần cốt lõi bao gồm:

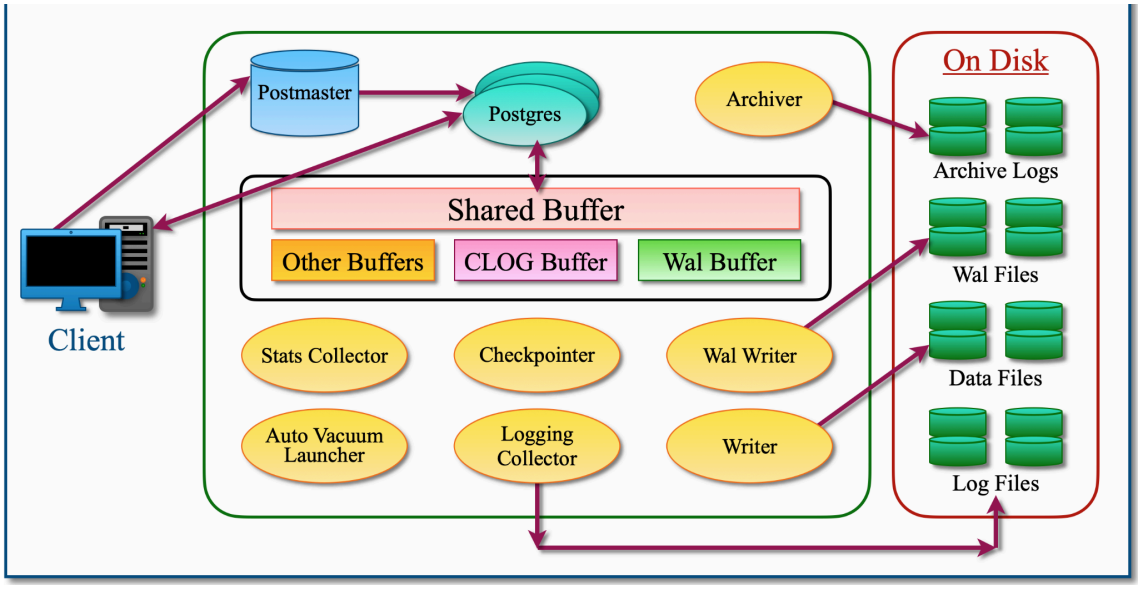
- **Postmaster Process**: Đóng vai trò trung gian điều phối, là tiến trình chính khởi động đầu tiên, chịu trách nhiệm khởi tạo hệ thống, quản lý kết nối và giám sát các tiến trình phụ. Postmaster tiếp nhận kết nối từ phía client, phân bổ tài nguyên và khởi tạo tiến trình backend để xử lý yêu cầu riêng biệt của từng người dùng. Nó cũng theo dõi trạng thái hệ thống, khởi động lại tiến trình con khi cần và duy trì sự ổn định tổng thể của cơ sở dữ liệu.

- **Backend Processes**: Được tạo ra khi có kết nối từ client, xử lý từng phiên làm việc. Mỗi backend là một tiến trình độc lập, đảm trách việc nhận câu lệnh SQL, phân tích, lập kế hoạch và trả kết quả lại cho client, giúp PostgreSQL dễ dàng xử lý các truy vấn song song mà không ảnh hưởng lẫn nhau.
- **Executor và Query Planner**: Chịu trách nhiệm phân tích, tối ưu hóa và thực thi câu truy vấn SQL. Planner xác định thứ tự join, cách lọc dữ liệu và chiến lược truy vấn hiệu quả nhất, còn executor thực thi từng bước theo đúng chiến lược đã chọn và trả kết quả về cho client.
- **Storage Manager**: Chịu trách nhiệm lưu trữ và truy xuất dữ liệu vật lý trên đĩa, tổ chức dữ liệu thành các trang dữ liệu và các buffer cache.
- **Write-Ahead Logging (WAL)**: Là cơ chế đảm bảo an toàn dữ liệu, ghi lại các thay đổi trước khi áp dụng lên data file nhằm bảo vệ dữ liệu khi xảy ra sự cố. Cơ chế này là nền tảng để hỗ trợ các tính năng như phục hồi theo thời gian, replication và đồng bộ dữ liệu giữa nhiều máy chủ.

**PostgreSQL Client**

PostgreSQL Client bao gồm các công cụ hoặc ứng dụng gửi câu truy vấn đến server qua giao thức PostgreSQL. Các client phổ biến gồm psql, PgAdmin hoặc các driver trong ngôn ngữ như Python, Java và Node.js.

**Cơ chế hoạt động:**



- **B1: Phân tích và lập kế hoạch.** Khi nhận một câu truy vấn, PostgreSQL sẽ phân tích cú pháp để hiểu cấu trúc lệnh và kiểm tra tính hợp lệ. Sau đó, hệ thống tạo ra nhiều kế hoạch thực thi dựa trên thống kê của bảng và index, cho biết có bao nhiêu bản ghi, mức độ chọn lọc của dữ liệu và các thông tin hỗ trợ khác.
- **B2: Tối ưu hóa truy vấn.** PostgreSQL so sánh chi phí của từng kế hoạch thực thi dựa trên các yếu tố như số lần truy cập đĩa, kích thước bảng, khả năng sử dụng index và điều kiện lọc trong truy vấn. Kế hoạch có chi phí thấp nhất sẽ được chọn để đảm bảo truy vấn chạy nhanh và hiệu quả.
- **B3: Thực thi truy vấn.** Executor thực thi kế hoạch đã chọn bằng cách đọc dữ liệu từ heap file hoặc index, áp dụng các toán tử như WHERE, JOIN, ORDER BY và GROUP BY. Trong quá trình này, PostgreSQL sử dụng buffer cache để giảm số lần truy cập đĩa, WAL để đảm bảo an toàn dữ liệu và các tiến trình nền như Background Writer và Checkpointer để duy trì ổn định hệ thống.

### 4.4. Ưu và nhược điểm của PostgreSQL

| Ưu điểm | Nhược điểm |
|---|---|
| Tuân thủ chuẩn SQL tốt, hỗ trợ nhiều tính năng nâng cao (trigger, function, stored procedure). | Cấu hình và quản trị tương đối phức tạp, cần kiến thức chuyên sâu để tối ưu. |
| Độ tin cậy cao nhờ cơ chế ACID và WAL, hạn chế mất dữ liệu. | Hiệu suất ghi không bằng các hệ thống NoSQL trong môi trường ghi tải cao. |
| Khả năng mở rộng tốt nhờ partitioning, indexing đa dạng và parallel query. | Việc mở rộng ngang (replication, sharding) còn khó khăn và khá phức tạp. |
| Hỗ trợ nhiều kiểu dữ liệu hiện đại như JSON, XML, arrays, GIS (PostGIS). | Có thể "quá nặng" đối với ứng dụng nhỏ hoặc workload đơn giản. |
| Mã nguồn mở, miễn phí, cộng đồng lớn và cập nhật thường xuyên. | Cần cấu hình tối ưu để tránh tiêu thụ nhiều tài nguyên khi xử lý truy vấn lớn. |

# 5. Trực quan hóa dữ liệu với Superset

## 5.1. Giới thiệu về Superset

Superset là một nền tảng trực quan hóa dữ liệu mã nguồn mở, được phát triển bởi Airbnb, hỗ trợ xây dựng dashboard tương tác, báo cáo và biểu đồ từ nhiều nguồn dữ liệu khác nhau.

## 5.2. Kiến trúc Superset

Superset là nền tảng trực quan hóa dữ liệu, bao gồm các thành phần chính:

- **Giao diện người dùng (Frontend / UI)**: là phần mà người dùng nhìn thấy và tương tác trực tiếp. Đây là nơi dashboard và biểu đồ được hiển thị, và người dùng có thể thao tác để phân tích dữ liệu. Cụ thể:
    - Hiển thị dashboard và biểu đồ:
        + Mọi chart, bảng dữ liệu, biểu đồ tổng quan đều xuất hiện tại giao diện này.
        + Người dùng có thể quan sát các xu hướng dữ liệu, so sánh giá cổ phiếu, khối lượng giao dịch, v.v.
    - Tương tác trực quan:
        + Superset hỗ trợ kéo-thả (drag-and-drop) để tạo và chỉnh sửa biểu đồ.
        + Người dùng không cần viết code hay biết SQL vẫn có thể tạo chart, thêm bộ lọc và thay đổi layout dashboard.
    - Tùy chỉnh và phản hồi trực tiếp:
        + Khi thay đổi filter hoặc metric, biểu đồ cập nhật ngay lập tức.
        + Giúp người dùng thử nhiều góc nhìn khác nhau mà không cần thao tác phức tạp.
- **Máy chủ (Backend / Server)**: là phần xử lý trung tâm, chịu trách nhiệm "nói chuyện" với cơ sở dữ liệu và gửi kết quả lên giao diện cho người dùng. Cụ thể:
    - Xử lý truy vấn dữ liệu:

        + Khi người dùng tạo biểu đồ hoặc dashboard, máy chủ nhận yêu cầu từ giao diện (frontend) và gửi truy vấn đến cơ sở dữ liệu (ví dụ PostgreSQL).
        + Sau khi cơ sở dữ liệu trả kết quả, máy chủ sẽ xử lý dữ liệu này (tính toán, lọc, tổng hợp nếu cần) trước khi gửi về giao diện hiển thị.
        + Ví dụ: nếu muốn vẽ biểu đồ giá trung bình cổ phiếu theo ngày, máy chủ sẽ gửi truy vấn lấy dữ liệu từng ngày, tính trung bình và trả kết quả dưới dạng bảng dữ liệu cho frontend hiển thị.
    - Hỗ trợ API REST:
        + Superset cung cấp API để các hệ thống khác có thể gửi yêu cầu tự động, ví dụ: cập nhật dữ liệu mới, lấy báo cáo, hoặc tạo dashboard tự động mà không cần thao tác thủ công.
        + Ví dụ: một script Python có thể gọi API để lấy dữ liệu khối lượng giao dịch hàng ngày từ dashboard Superset, phục vụ cho báo cáo tự động.
- **Cơ sở dữ liệu metadata**: là nơi Superset lưu thông tin về cách hệ thống được tổ chức, chứ không phải dữ liệu gốc như giá cổ phiếu. Cụ thể:
    - Lưu trữ thông tin quan trọng:
        + Dashboard, biểu đồ (chart), người dùng và quyền truy cập.
        + Các thiết lập hiển thị, bộ lọc, và cấu hình chart cũng được lưu tại đây.
        + Ví dụ: khi bạn tạo một biểu đồ hiển thị giá trung bình cổ phiếu theo ngày, metadata database sẽ lưu lại biểu đồ này, các trục (metric và dimension), và quyền ai được xem.
    - Đảm bảo ổn định và mở rộng:
        + Superset thường dùng PostgreSQL làm backend cho metadata database.
        + PostgreSQL giúp hệ thống hoạt động ổn định, lưu trữ an toàn và có thể mở rộng khi số lượng dashboard, chart hoặc người dùng tăng lên.

        + Ví dụ: khi một công ty có hàng trăm nhân viên sử dụng Superset, metadata database đảm bảo mọi thiết lập dashboard và quyền truy cập vẫn chính xác, ngay cả khi thêm chart mới hoặc người dùng mới.

- **Bộ nhớ đệm (Cache Layer, tùy chọn)**: là một lớp phụ trợ giúp **tăng tốc độ truy vấn và giảm tải cho cơ sở dữ liệu**, đặc biệt quan trọng khi dữ liệu lớn hoặc dashboard có nhiều biểu đồ:
    - Giảm thời gian chờ:
        + Khi nhiều người dùng cùng xem một dashboard, bộ nhớ đệm lưu lại kết quả truy vấn gần nhất, nên không phải gửi lại truy vấn tới cơ sở dữ liệu mỗi lần.
        + Ví dụ: nếu dashboard hiển thị khối lượng giao dịch 30 ngày gần nhất, bộ nhớ đệm có thể lưu kết quả này và phục vụ cho các lượt truy cập tiếp theo chỉ trong vài giây, thay vì phải tính toán lại từ database.
    - Giảm tải cho cơ sở dữ liệu chính:
        + Khi dashboard phức tạp hoặc dữ liệu lớn, việc gửi truy vấn trực tiếp đến database nhiều lần sẽ làm chậm hệ thống.
        + Bộ nhớ đệm giúp giảm số lượng truy vấn trực tiếp, duy trì hiệu năng ổn định cho toàn hệ thống.
        + Ví dụ: nếu có 100 người cùng xem dashboard "Giá cổ phiếu theo ngày" cùng lúc, bộ nhớ đệm sẽ trả kết quả nhanh chóng cho tất cả, thay vì database phải tính toán 100 lần.

### 5.3. Kết nối nguồn dữ liệu và truy vấn

Superset có thể kết nối trực tiếp đến PostgreSQL hoặc nhiều loại cơ sở dữ liệu khác thông qua SQLAlchemy. Điều này giúp người dùng truy vấn dữ liệu và xây dựng biểu đồ mà không cần biết nhiều về lập trình. Các bước kết nối cơ bản bao gồm:

- Khai báo kết nối đến cơ sở dữ liệu: Người dùng nhập thông tin như địa chỉ server, database name, username, password.

- Chọn schema và bảng dữ liệu: Schema giống như "thư mục" chứa các bảng dữ liệu. Người dùng sẽ tiến hành tạo các dataset bằng các bảng liên quan đến mục tiêu phân tích.

### 5.4. Tạo biểu đồ và dashboard

- **Biểu đồ (Chart):** Hỗ trợ nhiều loại: line, bar, area, scatter, heatmap, treemap, v.v. Một số cấu hinh cơ bản:
    - Metric: giá trị cần đo lường (ví dụ: giá trung bình, tổng khối lượng).
    - Dimension: trục phân loại (ví dụ: ngày, mã cổ phiếu).
    - Filter: bộ lọc dữ liệu để hiển thị các phân đoạn quan tâm.
- **Dashboard:**
    - Kết hợp nhiều chart thành một trang tổng quan giúp phân tích toàn diện.
    - Hỗ trợ global filter, tức là bộ lọc áp dụng cho tất cả các chart trên dashboard.
    - Layout kéo-thả, dễ tùy chỉnh.
    - Có thể chia sẻ hoặc nhúng vào website khác, thuận tiện cho báo cáo nhóm hoặc trình bày.

### 5.5. Cơ chế vận hành

Quy trình Superset vận hành từ dữ liệu đến biểu đồ có thể hiểu theo các bước sau:

- Người dùng chọn nguồn dữ liệu → Superset gửi truy vấn SQL tới PostgreSQL.
- PostgreSQL trả dữ liệu → Superset xử lý dữ liệu: tổng hợp, lọc, biến đổi.
- Frontend hiển thị chart/dashboard → người dùng có thể tương tác: lọc dữ liệu, drill-down để xem chi tiết.
- Bộ nhớ đệm (cache, nếu sử dụng) → giảm truy vấn lặp lại, tăng tốc độ hiển thị và giảm tải cho cơ sở dữ liệu.

### 5.6. Ưu và nhược điểm

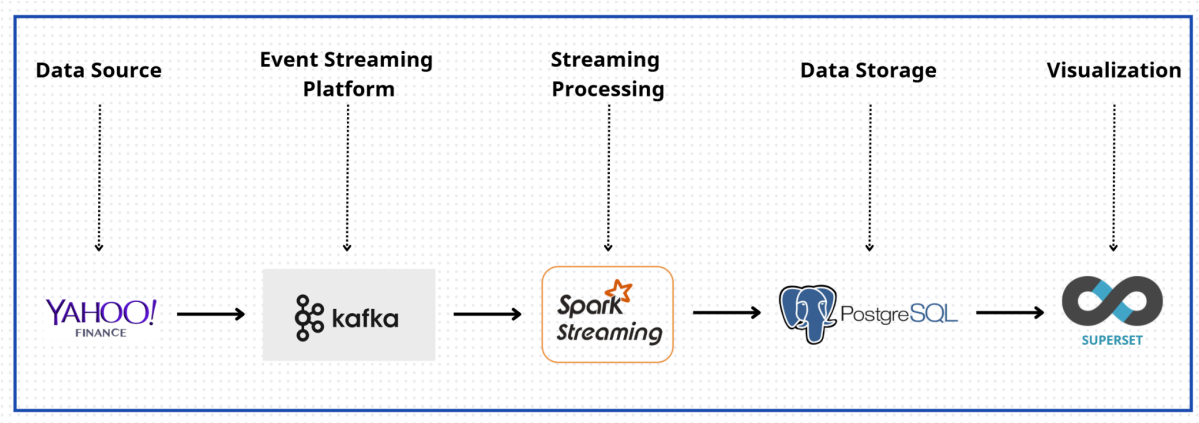
- **Ưu điểm:**

- Dễ trực quan hóa dữ liệu phức tạp mà không cần nhiều kiến thức lập trình.
- Tích hợp với nhiều loại cơ sở dữ liệu, không chỉ PostgreSQL.
- Dashboard tương tác: filter linh hoạt, kéo-thả, drill-down dữ liệu.
- Hỗ trợ phân quyền và bảo mật: có thể quản lý ai được xem hoặc chỉnh sửa dashboard.

- **Nhược điểm:**
  - Cấu hình ban đầu hơi phức tạp với người mới làm quen.
  - Dashboard phức tạp có thể chậm nếu dữ liệu quá lớn.
  - Cần kiến thức SQL để khai thác tối đa tính năng truy vấn và tạo biểu đồ nâng cao.

## V. Thiết kế kiến trúc hệ thống Data Pipeline

### 1. Mô hình kiến trúc tổng thể




Mô tả sơ lược về quy trình chính của pipeline:

- Bước 1: Dữ liệu được thu thập từ Yahoo Finance.
- Bước 2: Dữ liệu được gửi đến Kafka.
- Bước 3: Spark Streaming đọc dữ liệu từ Kafka và xử lý.
- Bước 4: Kết quả được lưu vào PostgreSQL.
- Bước 5: Kết nối dữ liệu từ PostgreSQL đến Superset và từ dữ liệu được kết nối đó sẽ sử dụng Superset để trực quan hóa.

## 2. Luồng xử lý dữ liệu chi tiết

### 2.1. Khởi tạo và thu nhận dữ liệu từ Yahoo Finance vào Kafka

Đây là bước đầu tiên và nền tảng trong kiến trúc Data Pipeline. Trong đề tài này, dữ liệu được thu thập từ Yahoo Finance, thông qua việc sử dụng Python để tự động hóa quá trình lấy dữ liệu. Dữ liệu tập trung vào các ngân hàng thương mại tại Việt Nam, cụ thể là Vietcombank, VietinBank và BIDV. Lý do chọn ba ngân hàng này là do đây là các ngân hàng quốc doanh, được thành lập bằng 100% vốn nhà nước, đặc biệt là nhóm "Big 4" bao gồm Agribank, Vietcombank, VietinBank, BIDV. Trong số này, Agribank hiện không có mã cổ phiếu niêm yết trên sàn, vì vậy dữ liệu chứng khoán chỉ có thể thu thập từ ba ngân hàng còn lại.

Các thông tin được lấy bao gồm: Tên ngân hàng (symbol), Mã chứng khoán (ticker), Giá hiện tại (current price), Giá đóng cửa trước đó (previous close), Giá mở cửa (open), Giá cao nhất trong ngày (day high), Giá thấp nhất trong ngày (day low), Khối lượng giao dịch (volume), Vốn hóa thị trường (market cap), Hệ số P/E (PE ratio) và Thời gian lấy dữ liệu (timestamp UTC).

```
data = {
    "symbol": symbol,
    "ticker": ticker,
    "current_price": info.get("currentPrice"),
    "previous_close": info.get("previousClose"),
    "open": info.get("open"),
    "day_high": info.get("dayHigh"),
    "day_low": info.get("dayLow"),
    "volume": info.get("volume"),
    "market_cap": info.get("marketCap"),
    "pe_ratio": info.get("trailingPE"),
    "time": datetime.now(pytz.utc).isoformat()
}
```

Quá trình thu thập dữ liệu trong trong pipeline này diễn ra theo các bước:

- Khởi tạo Kafka Producer: Tạo kết nối đến Kafka server, đảm bảo khả năng gửi dữ liệu từ Python lên topic tương ứng.
- Lấy dữ liệu từ Yahoo Finance: Sử dụng thư viện yfinance để truy xuất thông tin chứng khoán theo ticker của từng ngân hàng.
- Tạo payload JSON: Gom tất cả thông tin quan trọng của từng công ty thành một đối tượng JSON, bao gồm cả timestamp thời gian thực.
- Gửi dữ liệu vào Kafka: Dữ liệu JSON được producer gửi vào Kafka topic, đảm bảo dữ liệu có thể được xử lý tiếp theo bởi các consumer downstream trong pipeline.
- Lặp định kỳ: Quá trình thu thập được lập lại định kỳ, ở đây là mỗi 60 giây, để đảm bảo dữ liệu luôn cập nhật theo thời gian thực.

### 2.2. Xử lý và chuẩn hóa dữ liệu bằng Spark Streaming

Sau khi dữ liệu từ Yahoo Finance được thu thập và đưa vào Kafka dưới dạng các topic, bước tiếp theo trong pipeline là Data Transformation, cụ thể là xử lý và chuẩn hóa dữ liệu bằng Spark Streaming. Mục tiêu của bước này là biến dữ liệu dòng thô thành dạng có cấu trúc, sạch sẽ và sẵn sàng cho các phân tích hoặc lưu trữ lâu dài.

- **Kết nối Spark với Kafka:** Dữ liệu từ Kafka là dữ liệu thời gian thực (streaming), được lưu dưới dạng key-value:
    - Key: Không sử dụng trong dự án này.
    - Value: Chứa thông tin chính, được lưu dưới dạng JSON string, bao gồm các trường như: symbol, ticket, current_price, volume, pe_ratio, time.

Spark Streaming được sử dụng để **đọc trực tiếp từ topic Kafka**, tạo thành DataFrame streaming (df_raw) và chuẩn bị cho các bước xử lý tiếp theo.

- **Giải mã JSON và chuyển đổi kiểu dữ liệu:**

Dữ liệu JSON được parse thành DataFrame có cấu trúc (structured DataFrame) bằng hàm from_json, dựa trên schema được định nghĩa trước. Sau đó, cột thời gian time được chuyển thành kiểu timestamp (event_time) bằng to_timestamp, giúp Spark có thể thực hiện các phép toán theo cửa sổ thời gian.

Các trường quan trọng trong schema bao gồm:

+ symbol: Mã cổ phiếu (VCB, CTG, BID).
+ ticker: Mã giao dịch trên sàn (VCB.VN, CTG.VN, BID.VN).
+ current_price: Giá cổ phiếu hiện tại.
+ volume: Khối lượng giao dịch.
+ pe_ratio: Hệ số P/E.
+ time: Thời gian lấy dữ liệu.

```
# ======================
# 5️⃣ Parse JSON + chuyển kiểu thời gian
# ======================
df_parsed = df_raw.select(
    from_json(col("value").cast("string"), schema).alias("data")
).select("data.*")

df_parsed = df_parsed.withColumn("event_time", to_timestamp("time"))
```

- **Chuẩn hóa dữ liệu:** giúp đảm bảo dữ liệu đồng nhất và dễ phân tích.
    - Chuyển đổi timestamp: Dữ liệu thời gian từ JSON được chuẩn hóa sang định dạng timestamp.
    - Lọc và chọn trường quan trọng: Chỉ giữ các cột cần thiết cho phân tích: symbol, ticker, current_price, volume, pe_ratio, event_time.
    - Tạo cột bổ sung: Trong quá trình streaming, Spark tự động thêm cột current_time (current_timestamp()), cho phép theo dõi thời gian xử lý và đo lường độ trễ của pipeline.

```
# ======================
# 4️⃣ Schema JSON
# ======================
schema = StructType() \
    .add("symbol", StringType()) \
    .add("ticker", StringType()) \
    .add("current_price", DoubleType()) \
    .add("volume", DoubleType()) \
    .add("pe_ratio", DoubleType()) \
    .add("time", StringType())

# ======================
# 5️⃣ Parse JSON + chuyển kiểu thời gian
# ======================
df_parsed = df_raw.select(
    from_json(col("value").cast("string"), schema).alias("data")
).select("data.*")

df_parsed = df_parsed.withColumn("event_time", to_timestamp("time"))
```

- **Tính toán các chỉ số theo cửa sổ thời gian:** Spark Streaming thực hiện aggregation theo cửa sổ 10 phút.
    - Nhóm dữ liệu theo cửa sổ (window(event_time, "10 minutes")) và symbol, ticker.
    - Các chỉ số tính toán:
        + avg_price: Giá trung bình trong 10 phút.
        + max_price / min_price: Giá cao nhất / thấp nhất.
        + volatility: Độ lệch chuẩn của giá.
        + avg_volume: Khối lượng trung bình.
        + volume_volatility: Độ lệch chuẩn của khối lượng.
        + avg_pe: Giá trị trung bình hệ số P/E.
        + record_count: Số lượng bản ghi.

```
# ======================
# 6️⃣ Gộp dữ liệu theo cửa sổ 10 phút
# ======================
agg_df = df_parsed.groupBy(
    window(col("event_time"), "10 minutes"),
    col("symbol"), col("ticker")
).agg(
    avg("current_price").alias("avg_price"),
    max_("current_price").alias("max_price"),
    min_("current_price").alias("min_price"),
    stddev("current_price").alias("volatility"),
    avg("volume").alias("avg_volume"),
    stddev("volume").alias("volume_volatility"),
    avg("pe_ratio").alias("avg_pe"),
    count("*").alias("record_count")
)
```

- **Ghi dữ liệu đã xử lý vào PostgreSQL:**

Dữ liệu sau khi được xử lý và chuẩn hóa được lưu vào cơ sở dữ liệu PostgreSQL (stock_aggregates) bằng foreachBatch:

  - Mỗi batch của Spark Streaming được ghi riêng lẻ, đảm bảo tính liên tục.
  - Spark tự động kiểm tra batch có dữ liệu hay không trước khi ghi.
  - Dữ liệu ghi vào PostgreSQL bao gồm các trường quan trọng: symbol, ticker, window_start, window_end, avg_price, price_change_pct, max_price, min_price, volatility, avg_volume, volume_volatility, avg_pe, record_count.

### 2.3. Lưu trữ tối ưu hóa phân tích trong PostgreSQL

Sau khi dữ liệu được xử lý và tổng hợp trong Spark Streaming, kết quả được lưu vào PostgreSQL trong bảng stock_aggregates. Mục tiêu là đảm bảo dữ liệu sẵn sàng cho phân tích, dễ truy vấn và tối ưu hiệu suất.

Các thông tin lưu trữ bao gồm: symbol, ticker, window_start, window_end, avg_price, max_price, min_price, price_change_pct, avg_volume, volume_volatility, avg_pe, volatility và record_count.

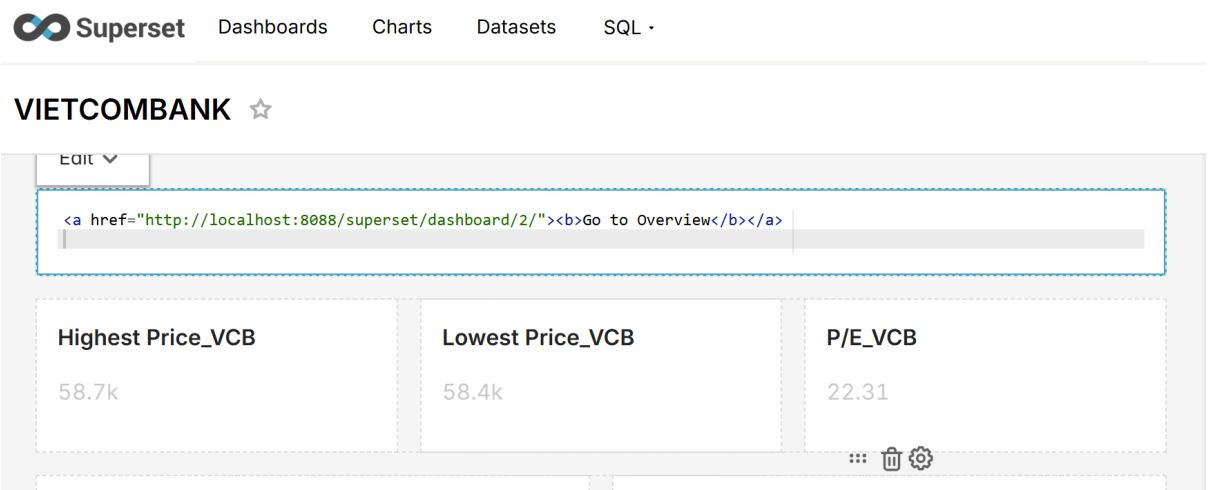
Dữ liệu được ghi theo batch từ Spark Streaming, giúp giảm tải cho cơ sở dữ liệu và đảm bảo tính liên tục của luồng dữ liệu. Việc chuẩn hóa tên cột và kiểu dữ liệu cũng giúp tối ưu truy vấn SQL, đồng thời hỗ trợ các phân tích theo thời gian hoặc theo ngân hàng một cách hiệu quả.

### 2.4. Trực quan hóa dữ liệu với Superset

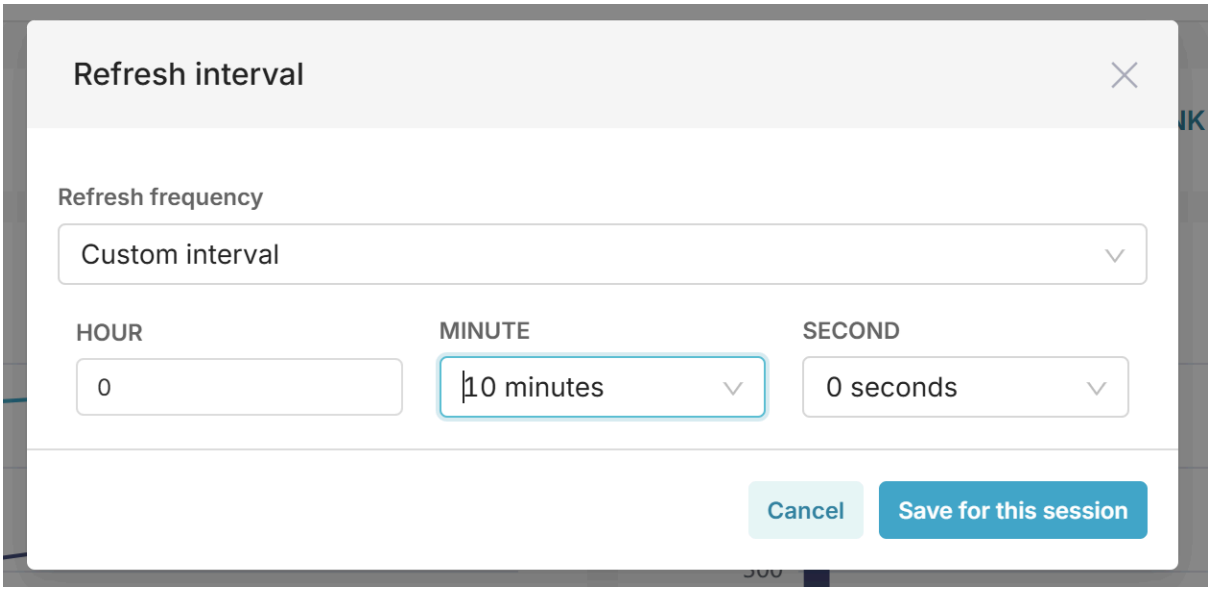
Mục tiêu của phần trực quan hóa là hiển thị dữ liệu ngân hàng đã được xử lý một cách trực quan, giúp người dùng theo dõi biến động giá, khối lượng giao dịch và các chỉ số tài chính quan trọng gần thời gian thực. Superset được kết nối trực tiếp tới cơ sở dữ liệu PostgreSQL, sử dụng bảng stock_aggregates làm nguồn dữ liệu. Nhờ đó, các dashboard luôn được cập nhật dữ liệu mới nhất từ Spark Streaming.

Để tăng tính tương tác, các dashboard được bổ sung Markdown component làm nút bấm điều hướng: Overview có nút dẫn đến từng dashboard ngân hàng, trong khi mỗi dashboard ngân hàng có nút quay về Overview. Dashboard cũng được cấu hình tự động làm mới dữ liệu sau mỗi 10 phút, đảm bảo các biểu đồ luôn hiển thị thông tin cập nhật, phục vụ phân tích kịp thời. Nhờ các tính năng này, người dùng có thể dễ dàng phân tích dữ liệu ở nhiều cấp độ, từ tổng quan cho đến chi tiết từng ngân hàng, so sánh các chỉ số quan trọng và theo dõi sự biến động theo thời gian.

- Minh họa về tạo markdown để chuyển dashboard:




- Minh họa về refresh dashboard 10 phút một lần:




# VI. Kết quả triển khai, thử nghiệm và đánh giá

## 1. Kết quả thu thập dữ liệu thực tế

Hệ thống được triển khai mô phỏng theo điều kiện giao dịch thực tế của thị trường chứng khoán Việt Nam, với khung giờ giao dịch từ 9h30 đến 14h30. Trong mỗi phiên giao dịch:

- Hệ thống cập nhật dữ liệu giá cổ phiếu mỗi 60 giây (1 phút/lần).

- Mỗi lần thu thập ghi nhận các trường cơ bản như: giá mở cửa, giá cao nhất, giá thấp nhất, giá đóng cửa phiên 1 phút, khối lượng và thời gian.

Như vậy, trong một phiên giao dịch kéo dài khoảng 5 giờ (300 phút), mỗi mã cổ phiếu sẽ có tối đa khoảng 300 bản ghi. Với ba mã cổ phiếu được theo dõi, tổng số bản ghi cho một phiên vào tối đa khoảng 900 dòng dữ liệu. Kịch bản thử nghiệm này đủ để đánh giá khả năng hoạt động ổn định của pipeline trong điều kiện dữ liệu cập nhật liên tục theo thời gian.

## 2. Kết quả xử lý & hiệu năng

Dữ liệu sau khi được thu thập được đưa vào hệ thống xử lý với cơ chế cửa sổ thời gian 10 phút:

- Spark Structured Streaming gộp dữ liệu theo cửa sổ 10 phút: 9h30 - 9h40, 9h40 - 9h50, …
- Với mỗi cửa sổ, hệ thống thực hiện các thao tác chuẩn hóa thời gian, tính toán các chỉ số tổng hợp (giá mở đầu cửa sổ, giá đóng cuối cửa sổ, giá cao nhất/thấp nhất trong cửa sổ, khối lượng giao dịch 10 phút), loại bỏ bản ghi lỗi.

Thời gian xử lý mỗi cửa sổ trung bình khoảng 20 giây, tính từ khi cửa sổ 10 phút kết thúc đến khi dữ liệu đã được xử lý và ghi xong vào cơ sở dữ liệu. Trong quá trình thử nghiệm, hệ thống không xuất hiện backlog hoặc kẹt batch, không ghi nhận lỗi mất bản ghi và giữ được trạng thái ổn định trong suốt cả phiên giao dịch. Điều này cho thấy pipeline có thể xử lý đều đặn các cửa sổ dữ liệu 10 phút mà không bị quá tải.

## 3. Đánh giá độ chính xác và độ trễ

Về độ chính xác, khi đối chiếu dữ liệu lại với nguồn gốc (Yahoo Finance) tại cùng thời điểm thu thập. Kết quả so sánh cho thấy:

- Các trường giá (open, high, low, close) trùng khớp với nguồn
- Timestamp ghi nhận đúng theo từng phút
- Không phát hiện trường hợp bị lệch thời gian hoặc sai số do định dạng

➔ Nhờ việc lấy dữ liệu định kỳ 60 giây một lần từ cùng một nguồn, độ sai lệch gần như không đáng kể, đủ độ tin cậy cho các bài toán phân tích biến động theo thời gian.

Về độ trễ, do hệ thống sử dụng cửa sổ 10 phút và trigger xử lý trung bình 20 giây, có thể ước tính:

- Kết quả tổng hợp cho một cửa sổ sẽ sẵn sàng khoảng 20 giây sau khi cửa sổ kết thúc, tức khoảng 9h40’20.
- Như vậy, độ trễ phân tích của hệ thống vào khoảng ~20 giây so với thời điểm kết thúc cửa sổ.
- Nếu xét trung bình trên toàn bộ dữ liệu nằm trong cửa sổ 10 phút, độ trễ hiệu dụng so với thời điểm phát sinh dữ liệu rơi vào khoảng 5–10 phút, tùy vị trí điểm dữ liệu trong cửa sổ.

➔ Đối với các bài toán giám sát xu hướng và phân tích tổng hợp theo khung 10 phút, mức độ trễ này được xem là chấp nhận được và vẫn đảm bảo ý nghĩa gần thời gian thực.

## 4. Khả năng mở rộng trong tương lai

Tương lai, hệ thống có thể được mở rộng theo các hướng sau:

- **Mở rộng theo chiều dữ liệu**: Tăng số lượng mã cổ phiếu theo dõi, mở rộng sang các nhóm ngành khác hoặc chỉ số thị trường mà không cần thay đổi cấu trúc pipeline.
- **Mở rộng theo tài nguyên**: Tăng số executor/worker của Spark, tăng partition trong Kafka, tối ưu partition trong cơ sở dữ liệu để xử lý tốt hơn khối lượng dữ liệu lớn hơn.
- **Mở rộng chức năng phân tích**: Thay vì chỉ tổng hợp theo cửa sổ 10 phút, có thể bổ sung thêm các khung thời gian khác (5 phút, 30 phút, 1 giờ).

## VII. Kết luận

Đề tài đã xây dựng thành công một hệ thống Data Pipeline hoàn chỉnh phục vụ cho việc thu thập, xử lý và lưu trữ dữ liệu giá cổ phiếu của ba ngân hàng VCB, CTG và BID. Hệ thống hoạt động ổn định trong suốt thời gian thử nghiệm.

Kết quả cho thấy pipeline đảm bảo độ chính xác gần như tuyệt đối, độ trễ thấp, và khả năng vận hành liên tục mà không bị gián đoạn. Đồng thời, dữ liệu được lưu trữ có cấu trúc, dễ truy vấn, và có thể trực quan hóa ngay lập tức.

Hệ thống không chỉ đáp ứng các mục tiêu đặt ra mà còn tạo nền tảng cho những nghiên cứu nâng cao hơn trong tương lai, đặc biệt là dự báo giá cổ phiếu, tạo cảnh báo tự động hoặc phân tích rủi ro dựa trên dữ liệu thời gian thực.

### 1. Giá trị ứng dụng

Hệ thống mang lại nhiều giá trị thực tế cho công tác phân tích tài chính và các hoạt động liên quan đến thị trường chứng khoán:

- **Hỗ trợ nhà đầu tư giám sát biến động thị trường**

Pipeline cho phép cập nhật dữ liệu cổ phiếu ngân hàng liên tục theo từng phút và tổng hợp theo khung 10 phút, giúp nhà đầu tư nắm bắt được xu hướng giá trong phiên giao dịch mà không cần theo dõi thủ công.

- **Cung cấp dữ liệu lịch sử phục vụ phân tích chuyên sâu**

Việc lưu trữ tập trung giúp dễ dàng truy xuất dữ liệu nhằm phân tích biến động dài hạn, tính toán các chỉ báo kỹ thuật, đánh giá rủi ro, xây dựng mô hình dự báo giá. Điều này mang lại lợi ích cho các chuyên viên phân tích, công ty chứng khoán cũng như các bộ phận quản lý danh mục.

- **Tăng tính tự động hóa và chuẩn hóa thông tin**

Hệ thống thu thập – xử lý – lưu trữ hoàn toàn tự động, giảm sự phụ thuộc vào thao tác thủ công, tránh sai lệch dữ liệu, tiết kiệm thời gian và đảm bảo dòng dữ liệu nhất quán và tin cậy.

- **Nâng cao khả năng hỗ trợ ra quyết định tài chính**

Thông qua Superset, dữ liệu được trình bày trực quan dưới dạng biểu đồ, bảng, chỉ số tổng hợp, giúp người dùng theo dõi diễn biến giá dễ dàng, so sánh xu hướng giữa các ngân hàng và đưa ra quyết định dựa trên dữ liệu thay vì cảm tính. Hệ thống phù hợp cho các nhà đầu tư cá nhân, chuyên viên phân tích, và các tổ chức tài chính cần báo cáo theo thời gian thực.

## 2. Hạn chế

Mặc dù đáp ứng được mục tiêu nghiên cứu, hệ thống vẫn còn một số hạn chế:

- **Chỉ sử dụng dữ liệu từ một nguồn duy nhất**, chưa phản ánh đầy đủ các biến động thực của thị trường.
- **Chưa xử lý các yếu tố bất thường của thị trường chứng khoán:** dư mua - dư bán, biên độ giá, sự kiện doanh nghiệp, tin tức.
- **Độ trễ phụ thuộc vào kiến trúc xử lý theo cửa sổ.** Với cửa sổ 10 phút, hệ thống chỉ phản ánh xu hướng sau khi kết thúc mỗi cửa sổ, chưa phù hợp cho phân tích theo từng giây hoặc mức độ siêu thời gian thực.
- **Hệ thống mới dừng ở mức thu thập - tổng hợp.** Chưa tích hợp các mô hình dự báo nâng cao hoặc công cụ cảnh báo tự động.

## 3. Bài học rút ra

Trong quá trình thiết kế, triển khai và vận hành hệ thống thu thập - xử lý - lưu trữ dữ liệu giá cổ phiếu, nhóm đã rút ra một số bài học quan trọng. Những bài học này không chỉ hữu ích cho việc cải thiện pipeline hiện tại, mà còn có giá trị tham khảo cho các dự án Data Engineering tương tự trong tương lai.

- **Thiết kế pipeline cần được lên kế hoạch cẩn hành:** Việc lựa chọn chu kỳ thu thập dữ liệu, độ dài cửa sổ xử lý và phương pháp tổng hợp có ảnh hưởng trực tiếp đến độ trễ và sự ổn định của hệ thống. Một thiết kế hợp lý ngay từ đầu giúp pipeline vận hành trơn tru, tránh phải điều chỉnh nhiều khi triển khai thực tế.
- **Chất lượng dữ liệu đầu vào quyết định chất lượng đầu ra:** Nếu dữ liệu lấy từ nguồn không chính xác hoặc không ổn định, mọi bước xử lý phía sau đều bị ảnh hưởng. Vì vậy cần chọn nguồn dữ liệu đáng tin cậy, đồng thời có cơ chế kiểm tra - đối chiếu để đảm bảo dữ liệu luôn nhất quán.
- **Giám sát hệ thống là yếu tố bắt buộc:** Kafka, Spark và PostgreSQL đều là các thành phần vận hành liên tục, nên cần được theo dõi thường xuyên để phát hiện sớm lỗi, nghẽn hoặc backlog. Việc giám sát tốt giúp duy trì độ ổn định của pipeline và đảm bảo dữ liệu luôn được cập nhật đúng thời gian.
- **Trực quan hóa mang lại góc nhìn rõ ràng hơn về dữ liệu:** Dashboard giúp nhận diện xu hướng, biến động và các điểm bất thường nhanh chóng. Nhờ trực quan hóa, nhóm thực hiện có thể đánh giá hiệu quả pipeline, điều chỉnh tham số và tối ưu quá trình xử lý một cách trực tiếp và dễ dàng hơn.

# TÀI LIỆU THAM KHẢO


1. **VietnamNet (2018)** 'The big five in Vietnam's banking system'. Available at: https://vietnamnet.vn/en/the-big-five-in-vietnams-banking-system-E210359.html (Accessed: 6 December 2025).
2. **200Lab (n.d.)** 'Kafka là gì?'. Available at: https://200lab.io/blog/kafka-la-gi/ (Accessed: 6 December 2025).
3. **Viblo (n.d.)** 'Kafka là gì?'. Available at: https://viblo.asia/p/kafka-la-gi-gDVK2Q7A5Lj (Accessed: 6 December 2025).
4. **Confluent (n.d.)** 'Introduction to Apache Kafka'. Available at: https://docs.confluent.io/kafka/introduction.html (Accessed: 6 December 2025).
5. **Viblo (n.d.)** 'Spark Streaming trong Apache Spark'. Available at: https://viblo.asia/p/spark-streaming-trong-apache-spark-y37Ld1m2Vov (Accessed: 6 December 2025).
6. **200Lab (n.d.)** 'Apache Spark là gì?'. Available at: https://200lab.io/blog/apache-spark-la-gi/ (Accessed: 6 December 2025).
7. **Viblo (n.d.)** 'Spark Streaming với Kafka'. Available at: https://viblo.asia/p/spark-streaming-voi-kafka-Ny0VG7n5VPA (Accessed: 6 December 2025).
8. **Apache Spark (n.d.)** 'Spark Streaming + Kafka Integration'. Available at: https://spark.apache.org/docs/latest/streaming-kafka-integration.html (Accessed: 6 December 2025).
9. **Atekco (n.d.)** 'Superset – Công cụ mạnh và miễn phí cho người làm dữ liệu'. Available at: https://atekco.io/1638847875880-superset-cong-cu-manh-va-mien-phi-cho-nguoi-lam-du-lieu/ (Accessed: 6 December 2025).
10. **Viblo (n.d.)** 'Superset là gì và xây dựng custom viz plugin đầu tiên'. Available at: https://viblo.asia/p/superset-la-gi-va-building-custom-viz-plugins-dau-tien-tren-superset-MkNLrbaoLgA (Accessed: 6 December 2025).

11. **PostgreSQL (n.d.)** ‘Documentation – Storage’. Available at: https://www.postgresql.org/docs/current/storage.html (Accessed: 6 December 2025).
12. **PostgreSQL (n.d.)** ‘Documentation – Indexes’. Available at: https://www.postgresql.org/docs/current/indexes.html (Accessed: 6 December 2025).
13. **PostgreSQL (n.d.)** ‘Documentation – Tablespaces’. Available at: https://www.postgresql.org/docs/current/manage-ag-tablespaces.html (Accessed: 6 December 2025).
14. **PostgreSQL Wiki (n.d.)** ‘Performance optimization’. Available at: https://wiki.postgresql.org/wiki/Performance_Optimization (Accessed: 6 December 2025).
15. **Docker (n.d.)** ‘Docker Compose – Official documentation’. Available at: https://docs.docker.com/compose/ (Accessed: 6 December 2025).
16. **Confluent (n.d.)** ‘Kafka Platform — Docker installation guide’. Available at: https://docs.confluent.io/platform/current/installation/docker/index.html (Accessed: 6 December 2025).
17. **Docker Hub (n.d.)** ‘PostgreSQL – Official Docker image’. Available at: https://hub.docker.com/_/postgres (Accessed: 6 December 2025).
18. **Apache Superset (n.d.)** ‘Installation using Docker Compose’. Available at: https://superset.apache.org/docs/installation/docker-compose/ (Accessed: 6 December 2025).