**CÔNG TY CỔ PHẦN MISA JSC**

**---🙠**🕮**🙢---**



**NGÀY HỘI SẢN XUẤT**

**NGHIÊN CỨU GIẢI PHÁP ĐỒNG BỘ DỮ LIỆU REAL-TIME GIỮA SQL VÀ ELASTICSEARCH ĐÁP ỨNG TÌM KIẾM DỮ LIỆU LỚN**

**CHO CÁC SẢN PHẨM CỦA MISA**

BÁO CÁO NGHIÊN CỨU

Nhóm: Dream Chaser From CRM

**Cán bộ hướng dẫn: Nguyễn Quang Hoàng**

**HÀ NỘI, 08-2022**

# **NHẬN XÉT CỦA BAN GIÁM KHẢO**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….……………………………………………………………………………………………………………………………………………………………**

|  |
| --- |
| Hà Nội, ngày …. Tháng …. Năm 2022 |
| Ký tên |
|  |

**Tóm Tắt:** Với sự phát triển mạnh mẽ của công ty cổ phầm MISA JSC, lượng sản phầm của MISA ngày càng phát triển cả về số lượng cũng như chất lượng, cùng với đó là tập dữ liệu khách hàng khổng lồ từ các phần mềm. MISA luôn mong muốn đem lại cho người dung trải nghiệm dịch vụ tốt nhất nhằm đem lại hiệu quả về năng suất cũng như chất lượng công việc lên hàng đầu, luôn đề cao tinh thần phụ sự xã hội. Tuy nhiên, cùng với việc phát triển lớn mạnh của các phần mềm nhà MISA thì dữ liệu từ khách hàng là vô cùng lớn, việc dữ liệu lớn được lưu trữ trên cơ sở dữ liệu quan hệ cho ra một hiệu suất truy vấn không được tốt, MISA cũng đã có những sự cô liên quan đến việc tím kiếm dữ liệu lớn. Các công cụ tìm kiếm ra đời nhằm giải quyết tình trạng này như: Apache Solr, MeiliSearch, Sphinx, Manticore và ElasticSearch là một trong những công cụ tìm kiếm giúp tôi ưu việc tìm kiếm nhanh dữ liệu lớn. Tuy nhiên, việc phát sinh vấn đề đồng bộ dữ liệu giữa cơ sở dữ liệu quan hệ sang cơ sở dữ liệu tìm kiếm là vấn đề lớn nhất, Apache Kafka là một nền tảng theo kiến trúc phân tán cho phép lưu trữ sự kiện và xử lý dữ liệu luồng mã nguồn mở được phát triển bởi Apache Software, với khả năng chịu lỗi cao và là hệ thống nhắn tin nhanh, kafka đang dần được thay thế cho hệ thống nhắn tin truyền thống. Nó được sử dụng cho các hệ thống nhắn tin thông thường trong các ngữ cảnh khác nhau. Trong nghiên cứu này, việc sử dụng ElasticSearch giải quyết bài toán tìm kiếm dữ liệu lớn và quan trọng hơn cả là tìm ra phương pháp đồng bộ dữ liệu real-time giữa cơ sở dữ liệu quan hệ và cơ sở dữ liệu tìm kiếm sẽ được chúng tôi ứng dụng thực tế tại dự án CRM.

**MỤC LỤC**

[NHẬN XÉT CỦA BAN GIÁM KHẢO 3](#_Toc112272886)

[MỤC LỤC 5](#_Toc112272887)

[DANH MỤC HÌNH ẢNH 6](#_Toc112272888)

[DANH MỤC BẢNG BIỂU 7](#_Toc112272889)

[CHƯƠNG 1: GIỚI THIỆU CHUNG 8](#_Toc112272890)

[1.1 Lý do chọn đề tài 8](#_Toc112272891)

[1.2 Giới thiệu 8](#_Toc112272892)

[1.1.1 Bài toán tìm kiếm nhanh với dữ liệu lớn 8](#_Toc112272893)

[1.1.2 Tìm kiếm nhanh trên các công cụ tìm kiếm 8](#_Toc112272894)

[1.1.3 Cơ chế đồng bộ dữ liệu 8](#_Toc112272895)

[1.3 Tiềm năng ứng dụng của bài toán tìm kiếm dữ liệu lớn và đồng bộ dữ liệu real-time tại công ty cổ phần MISA JSC 8](#_Toc112272896)

[1.4 Mục tiêu của nghiên cứu 8](#_Toc112272897)

[1.5 Cấu trúc bài nghiên cứu 8](#_Toc112272898)

[CHƯƠNG 2. ĐỒNG BỘ DỮ LIỆU REAL-TIME GIỮA CSDL GỐC VÀ CSDL TÌM KIẾM 9](#_Toc112272899)

[2.1 Cơ chế đồng bộ CDC 9](#_Toc112272900)

[2.2 Đồng bộ theo cơ chế Binlog MySQL 9](#_Toc112272901)

[2.3 Message Queue 10](#_Toc112272902)

[2.4. Các nền tảng và website hiện nay 13](#_Toc112272903)

[2.5 Kết luận 13](#_Toc112272904)

[CHƯƠNG 3. KẾT QUẢ THỰC NGHIỆM 14](#_Toc112272905)

[CHƯƠNG 4. KẾT LUẬN 15](#_Toc112272906)

[TÀI LIỆU THAM KHẢO 16](#_Toc112272907)

# **DANH MỤC HÌNH ẢNH**

**DANH MỤC BẢNG BIỂU**

# **CHƯƠNG 1: ĐẶT VẤN ĐỀ**

## Đặt vấn đề

Hiện tại, với gần 40 sản phẩm khác nhau, cùng nguồn dữ liệu khách hàng lớn, việc dữ liệu lớn được lưu trữ trên cơ sở dữ liệu quan hệ cho ra một hiệu suất truy vấn không được tốt, MISA cũng đã có những sự cố liên quan đến việc tím kiếm dữ liệu lớn. Tìm kiếm nhanh với dữ liệu lớn là một trong những vấn đề nhức nhối tại hầu hết các công ty trong và ngoài nước, mỗi đơn vị, tổ chức đều có cách giải quyết tương đồng hoặc khác nhau, tuy nhiên theo khảo sát tại các dự án tại công ty cổ phần MISA JSC, có tới hơn một nửa trong số đó đang gặp vấn đề về bài toán tìm kiếm dữ liệu lớn.

Theo khảo sát của nhóm có tới 88,2% các dự án (trong 17 dự án được khảo sát tại MISA) đang gặp vấn đề trong giải quyết bài toán tìm kiếm với dữ liệu lớn. Trong đó, có ba dự án đã có cách giải quyết, tuy nhiên những giải pháp đồng bộ dữ liệu giữa cơ sở dữ liệu quan hệ với cơ sở dữ liệu tìm kiếm đều gặp nhiều vấn đề về hiệu năng đồng bộ và hơn thế nữa là ảnh hưởng đến tính đúng đắn của dữ liệu, đây là thách thức rõ nhất, vấn đề này sẽ được tìm hiểu và giải quyết trong bài nghiên cứu này.

## Giới thiệu

Các công cụ tìm kiếm ra đời nhằm giải quyết tình trạng này như: Apache Solr, MeiliSearch, Sphinx, Manticore và ElasticSearch là một trong những công cụ tìm kiếm giúp tôi ưu việc tìm kiếm nhanh dữ liệu lớn.

Tuy nhiên, việc phát sinh vấn đề đồng bộ dữ liệu giữa cơ sở dữ liệu quan hệ sang cơ sở dữ liệu tìm kiếm là vấn đề lớn nhất, qua tìm hiểu sơ bộ cho thấy có rất nhiều phương pháp thủ công lẫn tự động nhằm giải quyết tình trạng này như: lưu dữ liệu đồng thời tại hai cơ sở dữ liệu, sử dụng một woker đồng bộ. Tuy nhiên những phương pháp trên không đem lại hiệu quả cao và tiềm ẩn nhiều rủi ro, trong nghiên cứu này chúng tôi sẽ đưa ra một bức tranh tổng quan về những phương pháp đồng bộ dữ liệu thời gian thực hiện có và đưa ra lời giải tốt nhất.

### **Tìm kiếm nhanh trên các công cụ tìm kiếm**

ElasticSearch đã quá phổ biến tại MISA và trên thế giới, từ những lý do sau:

* Là phần mềm mã nguồn mở, hoàn toàn miễn phí, cộng đồng phát triển lớn.
* Tốc độ nhanh: ElasticSearch có khả năng thực hiện những câu truy vấn phức tập một cách nhanh chóng và cũng có thể lưu lại hầu hết cấu trúc truy vấn vào bộ nhớ đệm để sử dụng cho việc filter kết quả…
* Hỗ trợ Full-text search: với các tính năng như tách từ, tách câu, tạo chỉ mục cho dữ liệu
* Hỗ trợ tìm kiếm mờ, tự động hoàn thành (autocomplete): giúp bạn có thể tìm ra kết quả kể cả khi bạn viết sai chính tả.
* Cung cấp Restful API cho phép xử lý các yêu cầu với các API Restful request.
* Dữ liệu lưu dưới dạng document oriented, free schema nên rất linh hoạt cho những trường hợp dữ liệu thường xuyên thay đổi cấu trúc.
* Khả năng mở rộng và tính sẵn dùng cao: với việc sử dụng mô hình cluster với nhiều node cùng tham gia phục vụ việc sử lý dữ liệu, khi một node chết thì vẫn không ảnh hưởng tới luồng xử lý, ngược lại khi muốn mở rộng ta chỉ cần thêm node mới vào hệ thống.

Sẽ có nhiều thắc mắc giữa ElasticSearch và MongoDB vì chúng khá giống nhau. Mặc dù ElasticSearch và MongoDB đều hỗ trợ cơ chế đánh index cho dữ liệu. Và khi các field của MongoDB được đánh index đầy đủ thì khả năng tìm kiếm của MongoDB cũng không kém cạnh so với ElasticSearch. Nhưng ElasticSearch vẫn có những điểm riêng biệt như:

* Cung cấp Restful API cho phép xử lý các yêu cầu dưới dạng các request API Restful.
* Tự động đánh index khi insert dữ liệu
* Việc đánh index ở mức độ chuyên sâu hơn so với MongoDB, trong khi MongoDB chỉ đánh index ở mức độ từ /words thì ElasticSearch chia nhỏ hơn nên MongoDB sẽ không phù hợp cho trường hợp tìm kiếm mờ (ví dụ bạn có 1 đoạn text ‘tran van b’ khi đánh index trong MongoDB nó sẽ tách ra làm 3 từ là [‘tran’, ‘van’, ‘b’] nếu bạn search từ ‘a’ thì nó sẽ không tìm thấy).

Theo một khảo sát nhanh tại MISA, có hơn 5/17 phiếu khảo sát tại các dự án là sử dụng ElasticSearch. Vì vậy ElasticSearch được xem như là đã phổ biến tại MISA và trong nghiên cứu lần này việc giải quyết bài toán tìm kiếm nhanh bằng công cụ tìm kiếm được coi như đã được giải quyết tại các dự án.

### **Cơ chế đồng bộ dữ liệu**

Bài toán tìm kiếm với dữ liệu lớn chỉ được giải quyết một nửa nếu như không giải quyết bài toán đồng bộ dữ liệu giữa cơ sở dữ liệu quan hệ và cơ sở dữ liệu tìm kiếm, đây là bài toán cốt lõi làm nên thành công của bài toán tìm kiếm nhanh với dữ liệu lớn. Ngoài ra, đặc thù của phần mềm MISA là khách hàng luôn mong muốn tìm kiếm thời gian thực. Tại MISA cũng đã giải quyết bài toán này bằng những phương pháp đơn giản như thêm dữ liệu tại cả hai cơ sở dữ liệu mỗi khi có dữ liệu mới phát sinh, hoặc đồng bộ dữ liệu sau một khoảng thời gian cố định.

Ưu điểm của những phương pháp này là đơn giản và dễ hiểu. Tuy nhiên, tồn tại trong đó là những rủi ro như, dữ liệu không đồng nhất do vấn đề coding, vấn đề hiệu năng khi phải thêm cùng lúc tại hai cơ sử dữ liệu.

Hiện nay, đã có rất nhiều những phương pháp đơn giản và hiệu quả hơn, cùng với đó là nhưng công cụ có sẵn hỗ trợ giải quyết bài toán này, một trong những phương pháp đó là CDC hoặc cơ chế đồng bộ do Apache Kafka, … cung cấp. Mỗi một phương pháp hay công cụ hỗ trợ điều có những điểm mạnh riêng, tuy nhiên để tìm ra một phương pháp hiệu quả nhất và phù hợp nhất với MISA thì cần có sự nghiên cứu và thực nghiệm với những số liệu cụ thể.

## Mục tiêu của nghiên cứu, tiềm năng ứng dụng của bài đồng bộ dữ liệu real-time tại MISA

Số lượng sản phầm của MISA ngày càng phát triển cả về số lượng cũng như chất lượng, cùng với đó là tập dữ liệu khách hàng khổng lồ từ các phần mềm, việc dữ liệu lớn được lưu trữ trên cơ sở dữ liệu quan hệ cho ra một hiệu suất truy vấn không được tốt, MISA cũng đã có những sự cô liên quan đến việc tím kiếm dữ liệu lớn, có thể nói bài toán tìm kiếm với dữ liệu lớn được coi là “gãi đúng chỗ ngứa” của MISA bấy lâu nay, việc hiện thực hóa bài toán lúc này là cần thiết hơn bao giờ hết nhằm đem lại trải nghiệm tốt cho người dùng và nâng cao giá trị sản phẩm MISA.

## Cấu trúc bài nghiên cứu

Dựa trên mục tiêu cụ thể đã trình bày, bài nghiên cứu sẽ chia sẻ 2 phần:

Chương 2: Đồng bộ dữ liệu real-time giữa CSDL gốc và CSDL tìm kiếm

Chương 3: Case Study của Tiki và Alibaba

**CHƯƠNG 2. ĐỒNG BỘ DỮ LIỆU REAL-TIME GIỮA CSDL GỐC VÀ CSDL TÌM KIẾM**

**2.1 Vấn đề đặt ra**

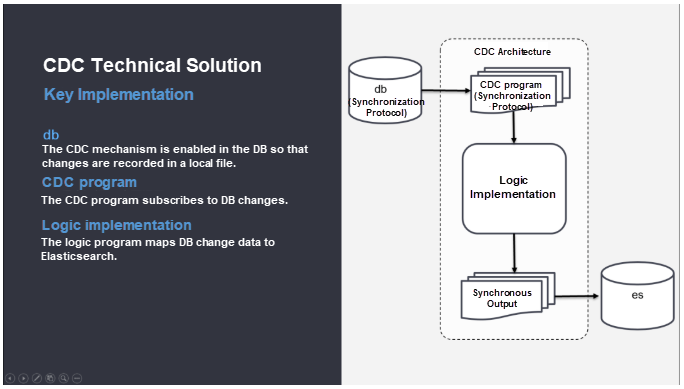
Trong nội dung bài toán này, dựa vào tình hình thực tế của Khối sản xuất, nhóm xin sử dụng hệ cơ sở dữ liệu quan hệ MySQL để tiến hành nghiên cứu.

Việc đồng bộ dữ liệu giữa CSDL gốc và ElasticSearch đòi hỏi phải giải quyết hai vấn đề:

* Đồng bộ dữ liệu đúng và nhanh: Mất bao lâu để đồng bộ hóa dữ liệu từ MySQL đến ElasticSearch? Dưới 1 giây. Thời gian chờ tối đa có thể chấp nhận được đối với hệ thống ứng dụng là bao nhiêu? Dưới 5 giây
* Đồng bộ dữ liệu đủ: Khi dữ liệu DB nguồn thường xuyên bị thay đổi, làm cách nào để có thể đảm bảo tính nhất quán dữ liệu giữa ElasticSearch và DB nguồn? Cần xây dựng cơ chế trace message và phục hồi dữ liệu đồng bộ bị thiếu.

Về mặt kỹ thuật, có nhiều cách để đồng bộ hóa dữ liệu DB với ElasticSearch:

* Lưu dữ liệu đồng bộ: ElasticSearch được cập nhật đồng bộ khi DB được cập nhật. Giải pháp kỹ thuật này là giải pháp đơn giản nhất, nhưng nó phải đối mặt với nhiều vấn đề nhất, bao gồm xung đột dữ liệu, ghi đè dữ liệu và mất dữ liệu. Vì vậy ta có thể khẳng định là phương pháp này không phù hợp để giải quyết bài toán đặt ra.
* Lưu dữ liệu không đồng bộ: Khi DB thay đổi, dữ liệu thay đổi sẽ được lưu vào 1 bộ đệm nào đó, sau đó sử dụng worker đặt lịch để quét bộ đệm, đọc dữ liệu và update sang ElasticSearch. Nhược điểm của phương án này là tốc độ đồng bộ không cao, không real-time, nếu trường hợp có quá nhiều transaction thì worker sẽ đồng bộ không kịp gây ra độ trễ lớn. Điều này đã được chứng minh thực tế tại dự án AMISCRM (MISA), dự án vẫn ghi nhận các lỗi gây ra do đồng bộ thiếu bản ghi giữa MySQL và ElasticSearch.
* Change Data Capture (CDC): Thay đổi dữ liệu được ghi lại từ DB bằng cơ chế đọc file BinLog của hệ cơ sở dữ liệu quan hệ, được đẩy sang chương trình trung gian và đồng bộ đến ElasticSearch. Dựa trên cơ chế CDC, dữ liệu tha đổi được trả về chính xác với tốc độ cực nhanh để đáp ứng các truy vấn. Dưới đây là mô hình CDC cơ bản. Nhóm nhận thấy đây là cách thức phù hợp để giải quyết bài toán đồng bộ real-time



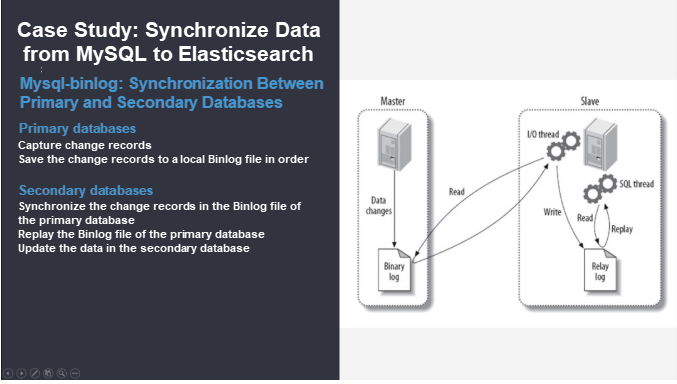
## 2.2 Cơ chế Change Data Capture (CDC)

Đồng bộ dữ liệu sử dụng cơ chế Binlog MySQL Binlog là một cơ chế chức năng được cung cấp bởi MySQL. BinLog là một tệp nhị phân trên đĩa, chứa tất cả các sự kiện thay đổi nội dung hoặc cấu trúc của cơ sở dữ liệu MySQL. Tất cả các thay đổi sẽ được lưu thêm vào (append only) vào một file log được đánh số thứ tự. Mỗi thay đổi được đặc trưng bởi hai tham số: file log được ghi và vị trí offset trên file log. Có 3 loại MySQL Binlog:

* Statement Based: lưu trữ các câu lệnh làm thay đổi dữ liệu
* Row based: lưu trữ các dữ liệu thay đổi
* Mixed: lưu trữ cả statement và data thay đổi của từng row

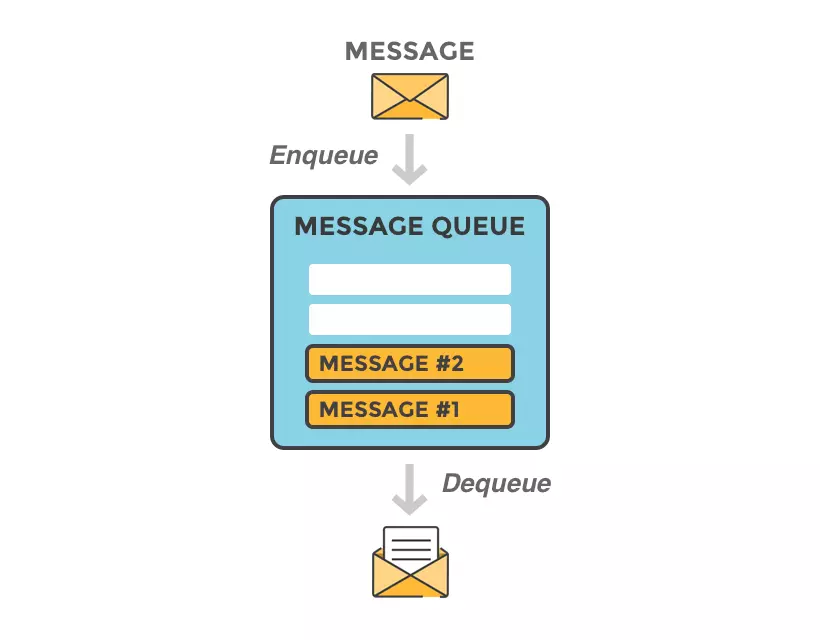
MySQL hỗ trợ cơ chế replicate binary log từ master sang slave. Nó sẽ streamming tất cả các thay đổi của database từ master tới slave. Toàn bộ binary log sẽ được replicate realtime tới slave theo các bước:

* Cơ sở dữ liệu chính ghi dữ liệu vào tệp binlog khi cơ chế BinLog được bật.
* Cơ sở dữ liệu phụ lấy dữ liệu binlog từ cơ sở dữ liệu chính, phát lại dữ liệu binlog và cập nhật dữ liệu trong cơ sở dữ liệu phụ.



## 2.3 Message Queue

Định nghĩa: Message queue là một kiến trúc cung cấp giao tiếp không đồng bộ. Ý nghĩa của queue ở đây chính là 1 hàng đợi chứa message chờ để được xử lý tuần tự theo cơ chế vào trước thì ra trước (FIFO - First In First Out).

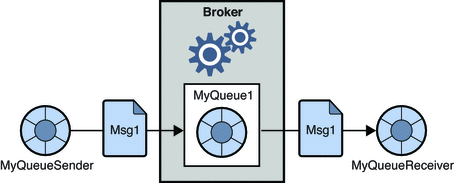
Một message là các dữ liệu cần vận chuyển giữa người gửi và người nhận. Vậy có thể hiểu đơn giản, message queue giống như một hòm thư email của chúng ta. Email có lẽ là ví dụ tốt nhất về giao tiếp không đồng bộ. Khi một email được gửi đi, người gửi tiếp tục xử lý những thứ khác mà không cần phản hồi ngay lập tức từ người nhận. Cách xử lý tin nhắn này tách người gửi khỏi người nhận để họ không cần phải tương tác với hàng đợi tin nhắn cùng một lúc.

Kiến trúc cơ bản của message queue rất đơn giản, bao gồm các thành phần như sau:

* Message: Thông tin được gửi (có thể là text, binary hoặc JSON)
* Producer: Service tạo ra thông tin, đưa thông tin vào message queue.
* Message Queue: Nơi chứa những message này, cho phép producer và consumer có thể trao đổi với nhau
* Consumer: Service nhận message từ message queue và xử lý
* Một service có thể vừa làm producer, vừa làm consumer

Một số message queue phổ biển hiện nay: Kafka, RamitMQ, Pulsar, ZeroMQ, IronMQ, RocketMQ, ..

Tại sao nên sử dụng Message queue.

* Ưu điểm:
  + Dễ scaling hệ thống: Vào giờ cao điểm, nhiều truy vấn, ta có thể tăng số lượng consumer lên để xử lý được nhiều messege hơn. Khi không cần ta có thể giảm lại.
  + Phân tán hệ thống: Giúp phân tách hệ thống thành nhiều service nhỏ hơn, mỗi service chỉ xử lý 1 chức năng nhất định theo cấu trúc [microservice](https://lcdung.top/phat-trien-phan-mem-theo-kien-truc-microservice/).
  + Đảm bảo duration/recovery: Do message đã được lưu trong queue, khi 1 service đang xử lý nhưng bị crash, ta không lo bị mất data vì có thể lấy message từ trong queue ra và retry. Trong 1 hệ thống có nhiều consumer, nếu vài consume crash cũng không làm crash cả hệ thống
  + Hỗ trợ rate limit, batch process: Trong trường hợp khả năng xử lý của hệ thống có hạn (chỉ có thể xử lý 100 lượt release/s) mà phải xử lý 10000 lượt release. Với message queue, ta có thể lấy từng lượt release chưa xử lý trong queue ra xử lý từ từ, không sợ bị mất.
* Nhược điểm:
  + Làm phức tạp hệ thống
  + Phải có message format: Từ 2 phía producer và consumer cần phải thống nhất format message để có thể gửi và nhận message.
  + Monitor Queue là cần thiết: Bạn cần phải theo dõi queue của mình để đảm bảo queue không quá nhiều hay đầy queue.
  + Khó xử lý đồng bộ: Không phải lúc nào message queue cũng là lựa chọn hàng đầu khi chúng ta xây dựng hệ thống. Sẽ có nhiều trường hợp hệ thống bắt buộc cần phải xử lý đồng bộ giữa các service, khi đó ta sẽ cần lựa chọn những cơ chế phù hợp hơn như Remote Procedure Invocation (RPI).
* Các loại message queue.
  + Point-to-point: Message queue có thể là kiểu point-to-point, tức là khi đó ta chỉ có một hàng đợi và một consumer duy nhất dể xử lý các tin nhắn trong hàng đợi:
  + Diagram

    Description automatically generatedPublisher-Subscriber. message queue có thể sử dụng định dạng Publisher-Subscriber, trong đó publisher (nhà sản xuất) gửi tin nhắn đến hàng đợi (trong trường hợp này được gọi là Topic) và tất cả subscriber (người đăng ký) vào cùng 1 Topic đều sẽ nhận được tin nhắn trong Topic đó:

## 2.4. Các tool triển khai cơ chế Change Data Capture (CDC)

Dựa trên tìm hiểu các công nghệ trên thế giới, các recommend của cộng đồng, nhóm đưa ra 3 nền tảng để phục vụ việc đọc binlog từ MySQL để push sự thay đổi dữ liệu.

***Debezium***

Debezium là một nền tảng mã nguồn mở để thu thập dữ liệu thay đổi. Bắt đầu nó, trỏ nó vào cơ sở dữ liệu và nó có thể bắt đầu phản hồi tất cả các thay đổi như insert, update, delete xảy ra trên database. Tốc độ của Debezium nhanh, vì vậy các ứng dụng của bạn có thể phản hồi nhanh chóng và không bao giờ bỏ lỡ một event nào.

Debezium là 1 tool để triển khai CDC. Nguyên lý hoạt động của nó là trích xuất những thay đổi trong log của database để xử lý

Hỗ trợ nhiều hệ quản trị cơ sở dữ liệu như SQL Server, MySQL, PostgreSQL…

***Airbyte***

Airbyte là nền tảng mã nguồn mở giúp chúng ta có thể sao chép dữ liệu sang 1 nền tảng khác 1 cách dễ dàng

Hỗ trợ nhiều hệ quản trị cơ sở dữ liệu như SQL Server, MySQL, PostgreSQL….

***Canal***

Canal là một hệ thống đồng bộ hóa dữ liệu dựa trên binary log của MySQL. Canal được sử dụng rộng rãi trong công ty Alibaba (bao gồm trang thương mại điện tử Taobao) để cải thiện độ trễ của data pipeline. Canal chỉ hỗ trợ DB Source là MySQL.

Nhóm đưa ra so sánh giữa các công cụ này theo bảng sau:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Debezium | Airbyte | Canal |
|  |  |  |  |
|  |  |  |  |
| Kết nối không giới hạn cơ sở dữ liệu | Có | Có | Có |
| Trả phí | Miễn phí | Có phí | Miễn phí |

## 2.5 Các vấn đề cần lưu ý khi triển khai

**2.6 Kết luận**

# **CHƯƠNG 3. CASE STUDY CỦA TIKI VÀ ALIBABA**

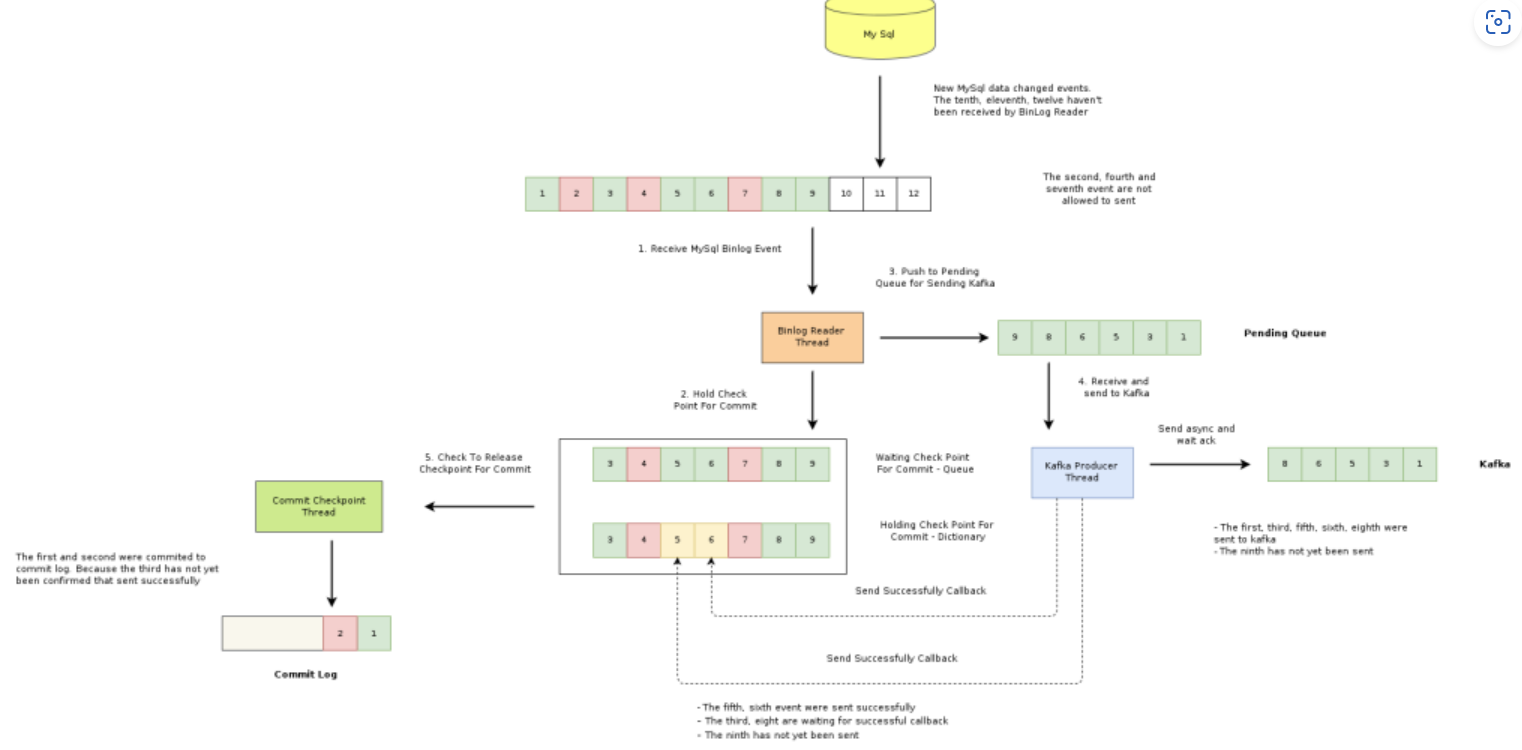
## 4.1. Cách Tiki xử lý vấn đề tích hợp giữa hệ thống lưu trữ và hệ thống tìm kiếm ElasticSearch

Tiki là một startup rất thành công của Việt Nam, sở hữu Tiki.vn hiện đang là trang thương mại điện tử lọt top 2 tại Việt Nam và top 6 tại khu vực Đông Nam Á, phục vụ hàng chục triệu giao dịch thương mại điện tử mỗi tháng, do đó dữ liệu của Tiki rất lớn. Qua các tìm hiểu, thử nghiệm, Tiki đã đưa ra một phương thức giải quyết bài toán đồng bộ dữ liệu giữa MySQL và ElasticSearch phục vụ tìm kiếm nhanh.



Về cơ bản, Tiki cũng sử dụng cơ chế đọc file BinaryLog của MySQL để bắt các thay đổi của cơ sở dữ liệu quan hệ, lợi dụng cơ chế replication binlog, tạo background worker đóng vai trò là slave để nhận các thay đổi từ master database MySQL. Bằng giải pháp như vậy, worker sẽ nhận được các dữ liệu thay đổi real-time rồi gửi đến các hệ thống đích khác.

Tiki sử dụng hai cơ chế trên để xây dựng background worker xác định các thay đổi gửi vào Message Queue là Kafka, sau đó xử lý thêm dữ liệu vào ElasticSearch.

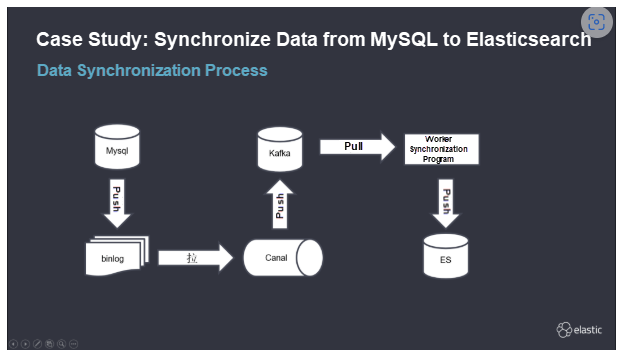


Việc áp dụng cơ chế binaray log, Tiki đưa ra các vấn đề cần giải quyết:

* Quản lý checkpoint: worker có thể tắt đột ngột, nếu khởi động lại làm thế nào để worker biết vị trí offset cuối cùng của file log để quay lại đọc tiếp ?
* MySQL ghi log tất cả các thay đổi, worker chỉ lọc ra các thay đổi nhất định gửi đi. Làm thế nào để đảm bảo thứ tự commit checkpoint chính xác với thự tự các checkpoint đọc được?
* Làm thế nào để đảm bảo hiệu năng của worker. Database Tiki có thể ghi nhận tới 10 nghìn thay đổi một giây. Tất cả các thay đổi được streamming real time cho worker. Nếu worker không xử lý tốt sẽ dẫn tới độ trễ lớn.
* Các cơ chế retry thế nào để đảm bảo tính chính xác và ổn định tuyệt đối của worker. Một ngày database tiki ghi nhận tầm 50 triệu lần thay đổi. Bất cứ sự cố nào khi xử lý của worker đều có thể dẫn tới sự không ổn định và rất khó để truy vết, khắc phục.

## 4.2. Cách Alibaba giải quyết bài toán đồng bộ dữ liệu real-time giữa MySQL và ElasticSearch

Alibaba là một công ty công nghệ lớn của Trung Quốc và thế giới, với lượng người dùng của riêng đất nước tỷ dân, lượng dữ liệu là rất lớn. Alibaba cũng đã gặp phải thách thức về đồng bộ dữ liệu real-time từ DB MySQL của họ sang ElasticSearch. Dưới đây mô tả tổng quan về mô hình giải pháp mà Alibaba đưa ra để giải quyết bài toán:



1. MySQL đẩy dữ liệu vào tệp Binlog cục bộ.
2. Canal lấy dữ liệu từ tệp Binlog, và sau đó đẩy dữ liệu đến Kafka.
3. Chương trình đồng bộ hóa Worker lấy dữ liệu từ Kafka, xử lý dữ liệu và sau đó đẩy dữ liệu đến ElasticSearch.

Alibaba cũng đưa ra các thách thức mà họ gặp phải khi triển khai mô hình trên, đó là tăng sự phức tạp của hệ thống, khi triển khai một nút trong mô hình trên các môi trường khác nhau có thể xảy ra một số vấn đề sau:

* DB được cập nhật hàng loạt. Điều này dẫn đến tắc nghẽn hiệu suất trong các nút kỹ thuật tiếp theo.
* Độ phức tạp của dữ liệu quan hệ. Do ElasticSearch là cơ sở dữ liệu không quan hệ, nên để ánh xạ từ cơ sở dữ liệu quan hệ, sẽ phải trải phẳng bảng quan hệ ra, càng nhiều quan hệ thì việc trải phẳng càng phức tạp.
* Hạn chế đối với kiểu dữ liệu nâng cao của ElasticSearch.