ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

HỆ THỐNG BÁN HÀNG TRỰC TUYẾN

Môn: QUẢN LÍ CƠ SỞ DỮ LIỆU HIỆN ĐẠI

SINH VIÊN THỰC HIỆN:

Sinh viên 1: 20120229 – Nguyễn Nhật Trường
Sinh viên 2: 20120531 – Lương Văn Nam
Sinh viên 3: 20120586 – Ngô Lê Hưng Thịnh
Sinh viên 4: 20120591 – Lê Đào Thảo Tiên
Sinh viên 5: 20120622 – Châu Nhật Tuyết

GIẢNG VIÊN PHỤ TRÁCH:

Cô: Nguyễn Trần Minh Thư
Cô: Tiết Gia Hồng
Thầy: Phạm Minh Tú

Hồ Chí Minh, ngày 09 tháng 07 năm 2024

I. Danh sách thành viên và bảng đánh giá

Họ tên	MSSV	Công việc	Đánh giá
Nguyễn Nhật Trường	20120229	Thực hiện quy trình đặt vé	
Lương Văn Nam	20120531	Thực hiện quy trình tra cứu vé	
Ngô Lê Hưng Thịnh	20120586	Thực hiện quy trình hủy vé	100 %
Lê Đào Thảo Tiên	20120591	Thực hiện quy trình tìm kiếm chuyến đi	
Châu Nhật Tuyết	20120622	Thực hiện quy trình thanh toán	

II. Phân tích ưu, nhược điểm của MongoDB, Redis, Cassandra và Neo4J

1. MongoDB

Ưu điểm	– Cho phép lập chỉ mục bất kỳ trường nào trong tài liệu – Hiệu suất cao, tốc độ truy vấn (find, update, insert, delete) của MongoDB nhanh hơn hẳn so với các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) (có thể nhanh gấp 100 lần) – Tính sẵn sàng cao – Khả năng mở rộng cao – Ngôn ngữ truy vấn MQL dựa trên tài liệu mạnh mẽ như SQL – Linh hoạt – thêm/xóa trường ít hoặc không ảnh hưởng đến ứng dụng – Dữ liệu không cần đồng nhất về kích thước – Phân phối được – Biểu diễn dữ liệu trong JSON hoặc BSON – Hỗ trợ không gian địa lý (Geospatial) – Tích hợp dễ dàng với BigData Hadoop – Được xây dựng cho cloud. Kiến trúc mở rộng quy mô tự nhiên của nó, được kích hoạt bởi sharding, liên kết tốt với quy mô và sự nhanh nhẹn có được nhờ điện toán đám mây.
Nhược điểm	– Không hỗ trợ phép joins và lưu trữ theo key-value dẫn đến dư thừa dữ liệu (lặp lại key) – Nguy cơ gây mất dữ liệu khi chưa hoàn thành bản lưu (mất khoảng 60s nền tảng này mới thực hiện ghi toàn bộ dữ liệu thay đổi từ Ram vào ổ cứng) – Chỉ lưu được tối đa 16Mb trên 1 tài liệu – Chiếm nhiều bộ nhớ khi sử dụng

2. Redis

Ưu điểm	- Đơn giản: Redis có mô hình dữ liệu đơn giản và dễ sử dụng, dựa trên cấu trúc key-value. Việc lưu trữ, truy xuất và thao tác dữ liệu trở nên dễ dàng và trực quan. - Hiệu suất cao: Redis là một kho lưu trữ dữ liệu trong bộ nhớ, có nghĩa là nó lưu trữ dữ liệu trực tiếp trong RAM, giúp truy cập dữ liệu nhanh hơn nhiều so với các cơ sở dữ liệu dựa trên đĩa. - Tính linh hoạt: Redis hỗ trợ nhiều cấu trúc dữ liệu khác nhau, bao gồm string, list, set, sorted set và hash, cho phép lưu trữ và thao tác nhiều loại dữ liệu một cách hiệu quả. - Khả năng mở rộng: Redis có thể được triển khai trên một máy chủ đơn hoặc phân tán trên nhiều máy chủ để xử lý khối lượng dữ liệu lớn và lưu lượng truy cập cao. - Hỗ trợ nhiều ngôn ngữ: Redis cung cấp giao diện lập trình ứng dụng (API) cho nhiều ngôn ngữ lập trình phổ biến, bao gồm Python, Java, C++, JavaScript, PHP và Ruby, giúp tích hợp dễ dàng vào các ứng dụng. - Độ tin cậy cao: Redis cung cấp nhiều tính năng bảo mật và độ tin cậy, bao gồm sao lưu dữ liệu, replication và failover, giúp đảm bảo tính sẵn sàng và bảo mật dữ liệu.
Nhược điểm	- Lưu trữ dữ liệu giới hạn: Do lưu trữ dữ liệu trong RAM, Redis có dung lượng lưu trữ giới hạn so với các cơ sở dữ liệu dựa trên đĩa. - Không phù hợp cho dữ liệu lớn: Redis không phù hợp cho các ứng dụng cần lưu trữ và xử lý lượng dữ liệu lớn, vì nó có thể ảnh hưởng đến hiệu suất và chi phí. - Thiếu tính ACID: Redis không hỗ trợ đầy đủ tính ACID (Atomicity, Consistency, Isolation, Durability), có nghĩa là dữ liệu có thể bị mất hoặc không nhất quán trong một số trường hợp nhất định. - Khả năng sao lưu hạn chế: Redis cung cấp các tùy chọn sao lưu dữ liệu, nhưng nó không đầy đủ như các cơ sở dữ liệu dựa trên đĩa. - Có thể xảy ra lỗi dữ liệu: Do lưu trữ dữ liệu trong RAM, Redis dễ bị ảnh hưởng bởi lỗi phần mềm hoặc sự cố phần cứng, dẫn đến mất dữ liệu.

3. Cassandra

Ưu điểm	**Khả năng mở rộng tuyến tính**: Cassandra được thiết kế để mở rộng tuyến tính, có thể xử lý hàng tỷ hoặc thậm chí hàng trăm tỷ bản ghi và tải truy vấn lớn mà vẫn giữ được hiệu suất cao. Nó có khả năng chia dữ liệu và phân phối trên nhiều nút trong cụm mạng, giúp cải

	thiện khả năng mở rộng và khả năng chịu lỗi. • **Khả năng xử lý dữ liệu phân tán:** Cassandra có khả năng xử lý dữ liệu phân tán, cho phép lưu trữ dữ liệu trên nhiều máy chủ và nút mạng. Điều này giúp cải thiện độ tin cậy và khả năng chịu lỗi của hệ thống. • **Tính nhất quán linh hoạt**: Cassandra sử dụng mô hình ghi ghi nhất quán linh hoạt (eventual consistency), cho phép cấu hình mức độ nhất quán cho từng truy vấn hoặc 4 cụm dữ liệu. Điều này giúp đáp ứng được các yêu cầu độ nhất quán khác nhau của ứng dụng. • **Tốc độ đọc và ghi cao**: Cassandra là một hệ thống cơ sở dữ liệu phân tán được tối ưu cho việc ghi và đọc dữ liệu với hiệu suất cao. Nó sử dụng cơ chế ghi tới bộ nhớ đệm và cơ chế đọc đọc trước để tăng tốc độ truy cập dữ liệu.
Nhược điểm	**Khó khăn trong việc thiết kế mô hình dữ liệu:** Cassandra không hỗ trợ các truy vấn phức tạp và liên kết giữa các bảng như trong các hệ quản trị cơ sở dữ liệu quan hệ. Điều này đòi hỏi người phát triển phải có kiến thức sâu về mô hình dữ liệu đặc biệt của Cassandra và cách thiết kế cấu trúc bảng hiệu quả. • **Hạn chế trong truy vấn phức tạp:** Cassandra không hỗ trợ các truy vấn phức tạp như các truy vấn liên kết và các phép toán JOIN như trong hệ quản trị cơ sở dữ liệu quan hệ. Điều này có thể làm hạn chế khả năng truy vấn và phân tích dữ liệu phức tạp. • **Chi phí thiết kế và triển khai cao:** Cassandra đòi hỏi kiến thức chuyên sâu về hệ thống phân tán và quản lý cụm để triển khai và vận hành hiệu quả. Điều này có thể tạo ra một ngưỡng học tập và chi phí cao khi sử dụng Cassandra.

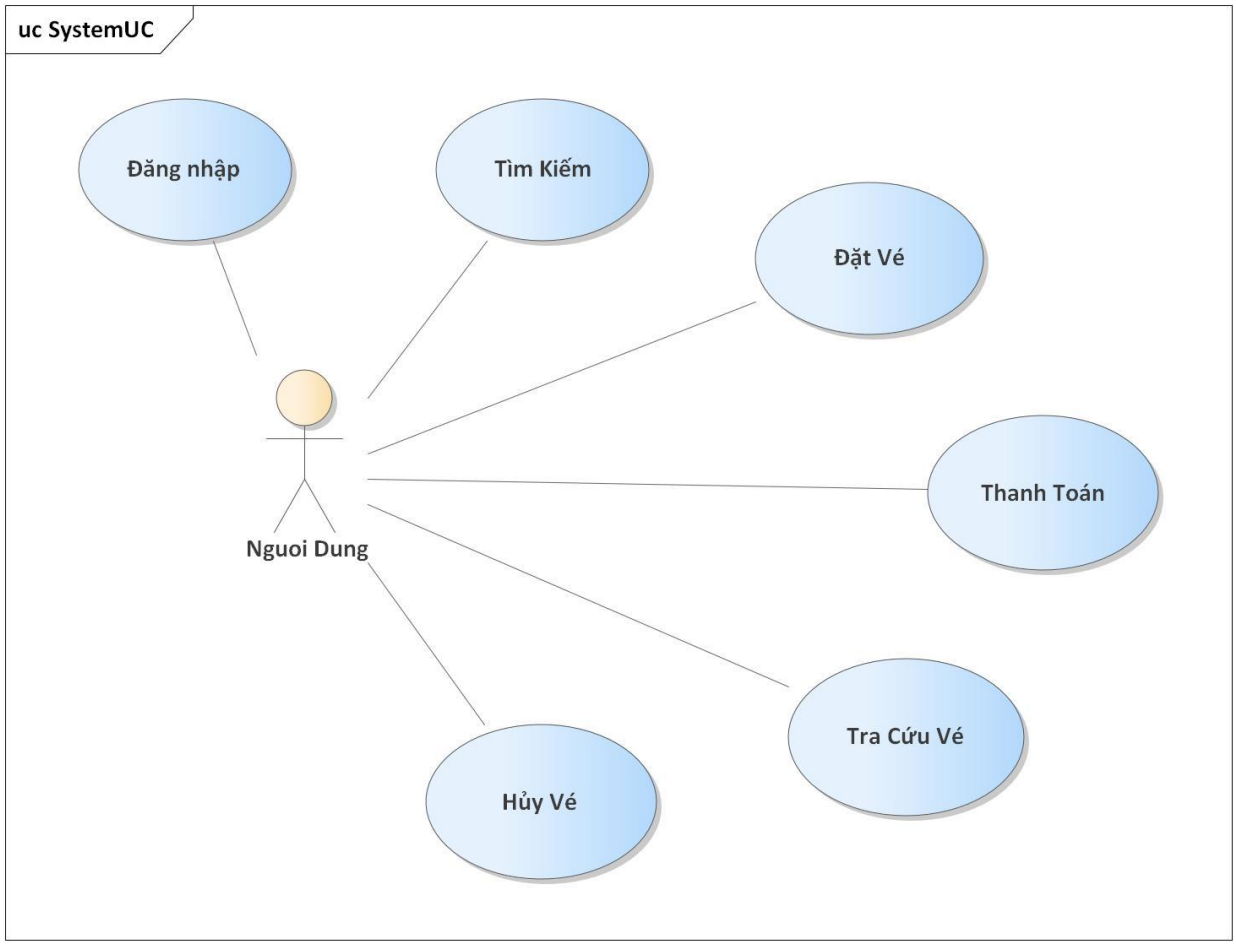
4. Neo4J

Ưu điểm	-*Cấu trúc dữ liệu đồ thị*: Neo4j được thiết kế đặc biệt để lưu trữ và truy vấn dữ liệu theo mô hình đồ thị. Điều này cho phép mô phỏng và biểu diễn quan hệ phức tạp giữa các đối tượng trong dữ liệu một cách hiệu quả. Vì vậy Neo4j rất mạnh mẽ trong việc xử lý các vấn đề liên quan đến mối quan hệ, như mạng xã hội, hệ thống khuyến nghị, phân

	tích mạng lưới… ➔ Là giải pháp tốt nhất để xử lý dữ liệu được kết nối. -*Truy vấn linh hoạt và hiệu suất cao*: Neo4j cung cấp một ngôn ngữ truy vấn tương tự như SQL gọi là Cypher, được tối ưu hóa để thực hiện các truy vấn đồ thị hiệu quả. Cypher cho phép dễ dàng tìm kiếm, trích xuất và phân tích dữ liệu đồ thị một cách linh hoạt. Neo4j cũng sử dụng cơ chế lưu trữ và truy vấn khéo léo giúp cải thiện hiệu suất truy vấn, đặc biệt là khi bạn có nhiều mối quan hệ phức tạp trong dữ liệu. -*Tính nhất quán và đồng bộ:* Neo4j hỗ trợ giao thức ACID để đảm bảo tính nhất quán và đồng bộ trong quá trình ghi và đọc dữ liệu. Điều này rất quan trọng khi làm việc với dữ liệu có mối quan hệ phức tạp và đảm bảo tính chính xác của các thay đổi dữ liệu.
Nhược điểm	-*Khó khăn trong việc thay đổi cấu trúc dữ liệu*: Trong Neo4j, việc thay đổi cấu trúc dữ liệu đồ thị có thể phức tạp và đòi hỏi quá trình cập nhật lớn. Nếu thay đổi cấu trúc dữ liệu đồ thị thường xuyên, có thể gây ra sự bất tiện và tốn nhiều thời gian. -*Giá cả:* Neo4j là một sản phẩm thương mại và có mức phí sử dụng. Điều này có thể là một hạn chế đối với các dự án có nguồn lực hạn chế hoặc các ứng dụng nhỏ. -Không có bảo mật nào được cung cấp ở cấp độ dữ liệu, không có mã hóa dữ liệu.

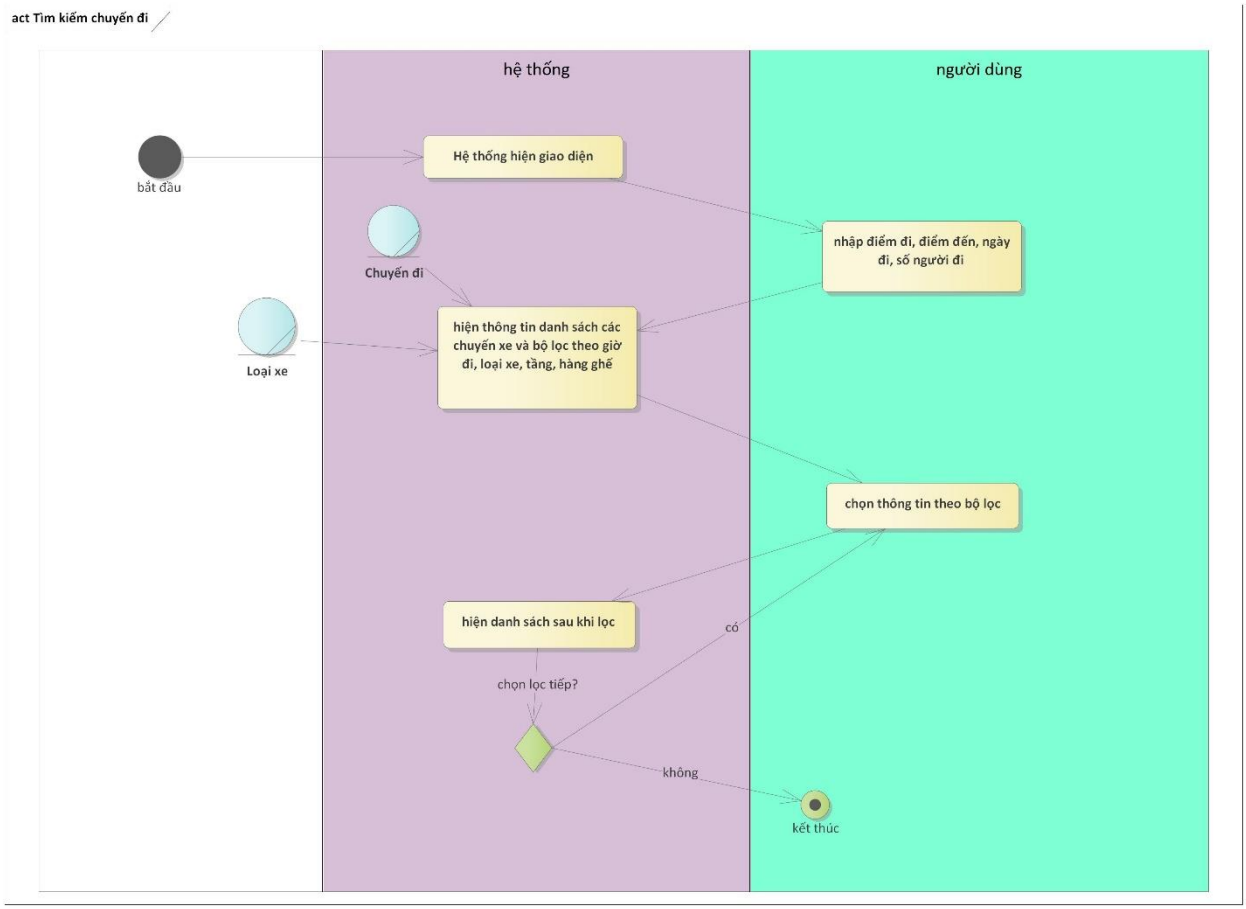
III. Mô tả quy trình nghiệp vụ của hệ thống futabus:

1. Use case hệ thống

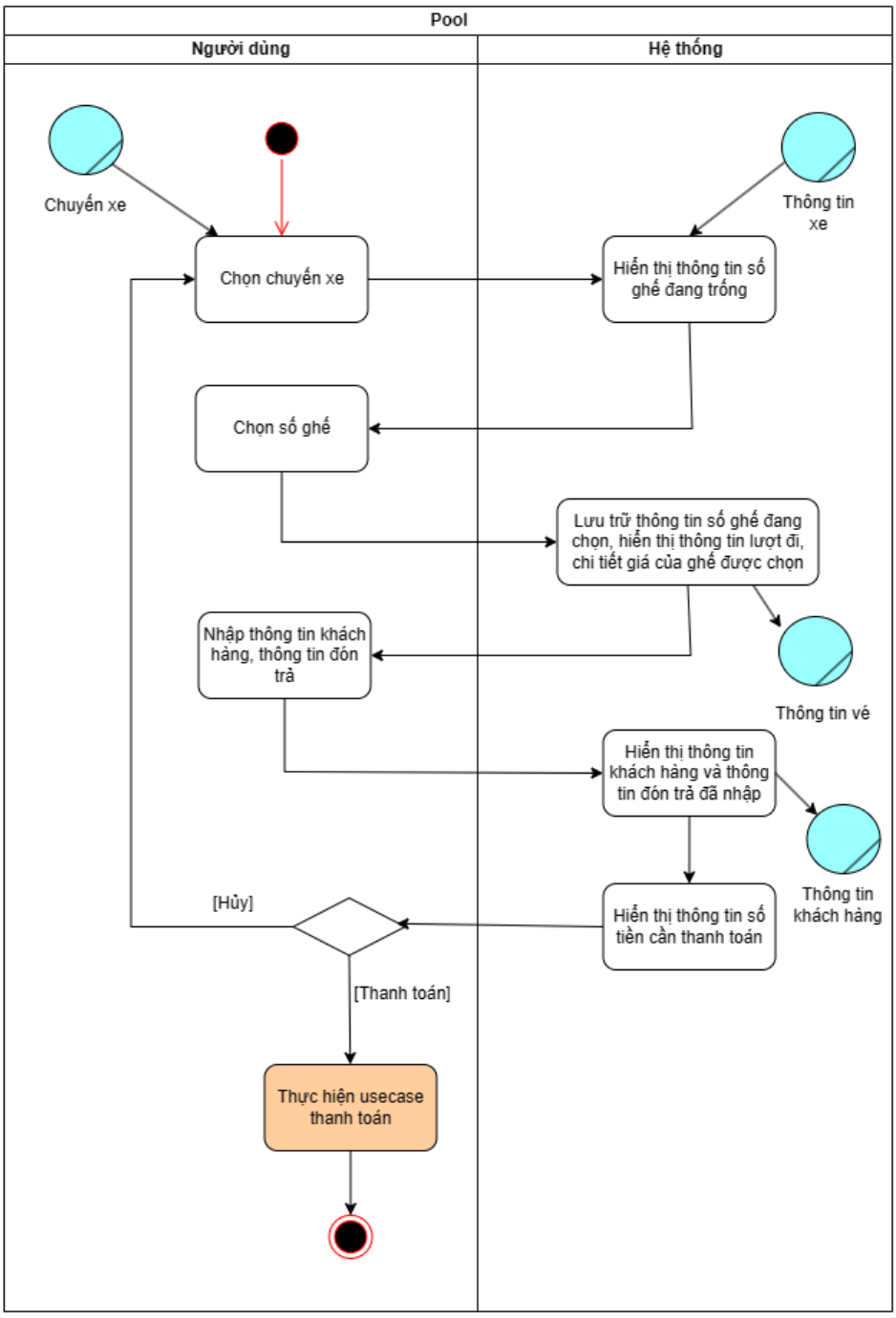


2. Đặc tả use case

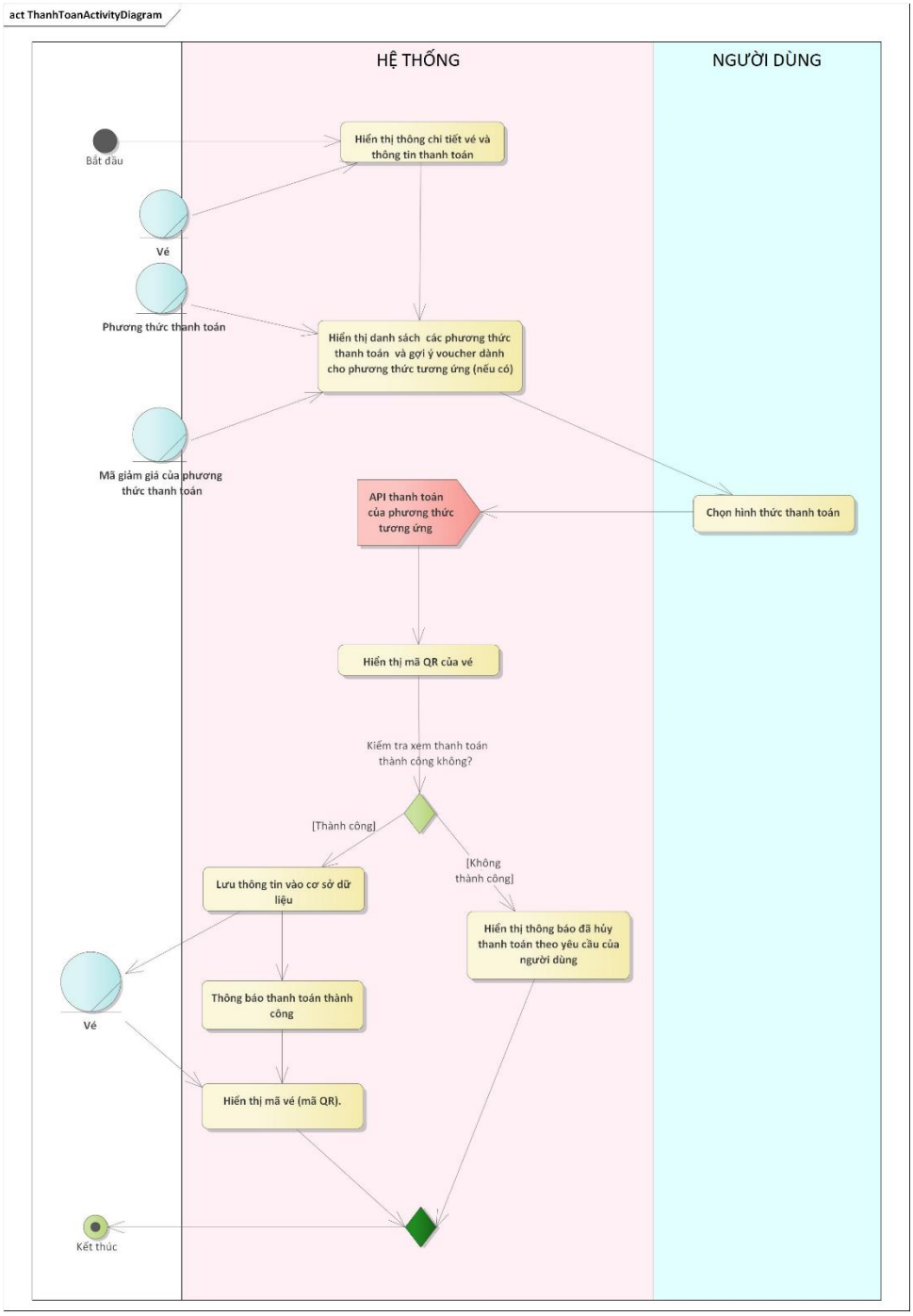
2.1. Tìm kiếm chuyến đi


act Tìm kiếm chuyến đi
hệ thống
người dùng
bắt đầu
Hệ thống hiện giao diện
Chuyến đi
nhập điểm đi, điểm đến, ngày đi, số người đi
Loại xe
hiện thông tin danh sách các chuyến xe và bộ lọc theo giờ đi, loại xe, tầng, hàng ghế
chọn thông tin theo bộ lọc
hiện danh sách sau khi lọc
có
chọn lọc tiếp?
không
kết thúc

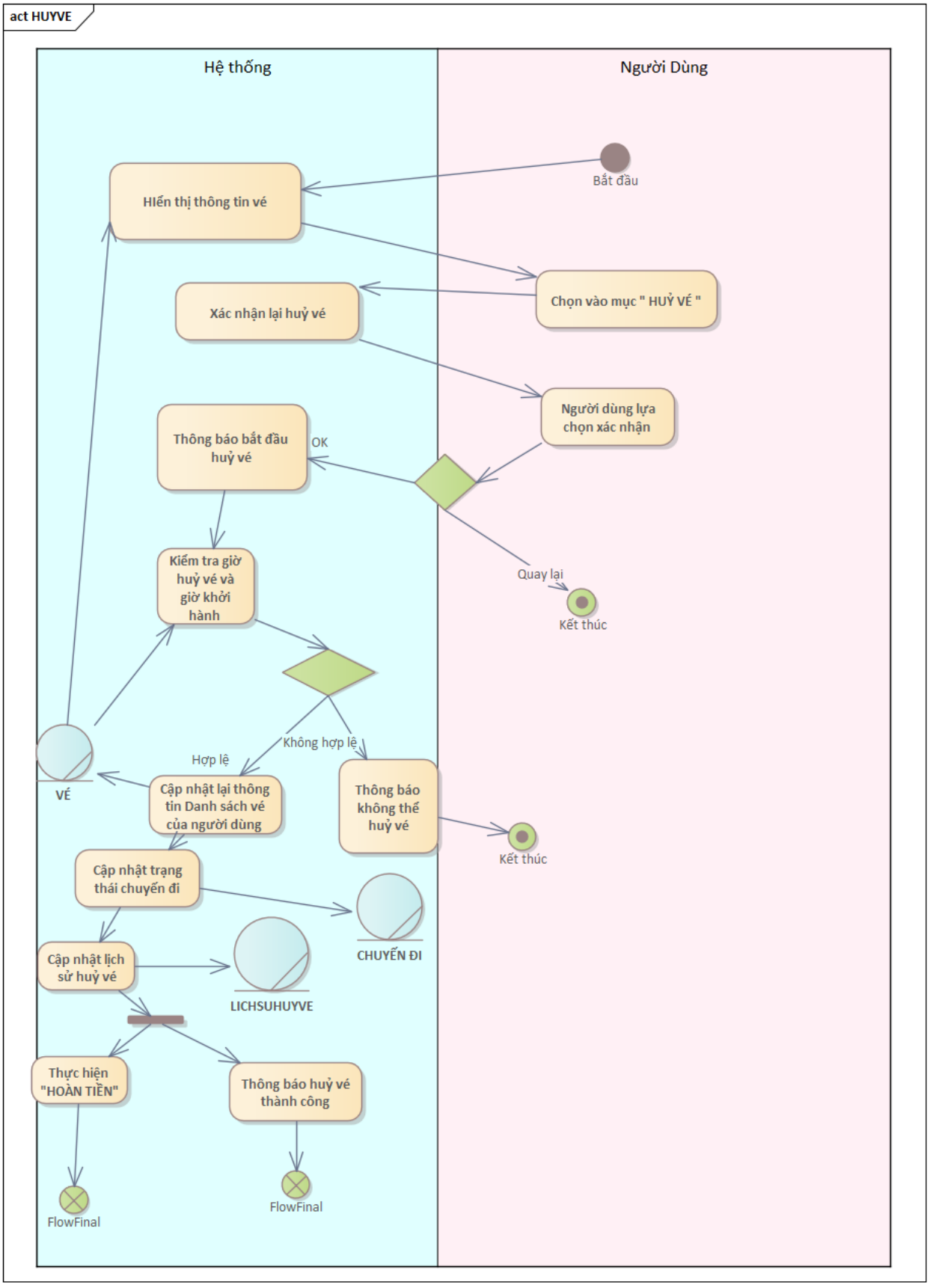
2.2. Đặt vé



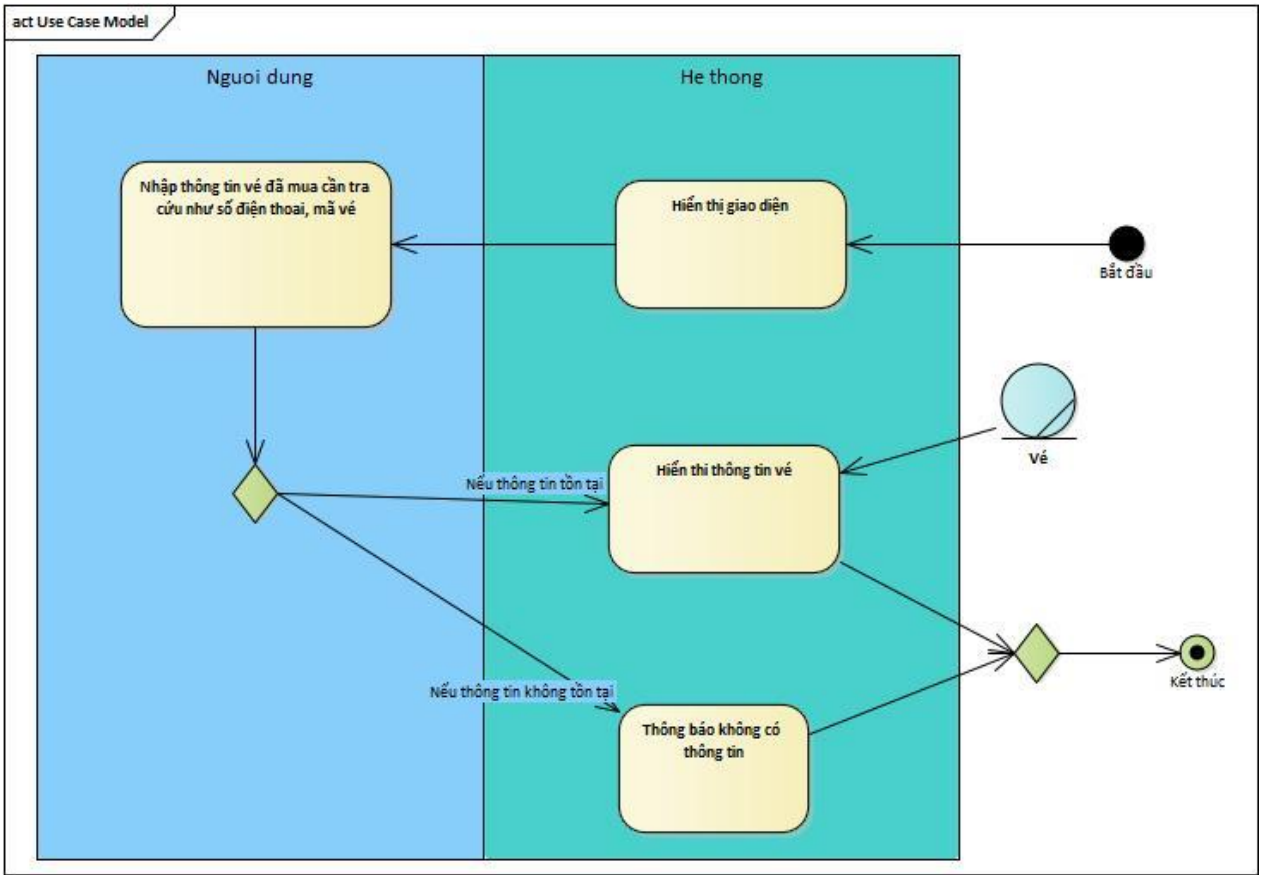
2.3. Thanh toán


act ThanhToanActivityDiagram
HỆ THỐNG
NGƯỜI DÙNG
Bắt đầu
Hiển thị thông chi tiết vé và thông tin thanh toán
Vé
Phương thức thanh toán
Hiển thị danh sách các phương thức thanh toán và gợi ý voucher dành cho phương thức tương ứng (nếu có)
Mã giảm giá của phương thức thanh toán
API thanh toán của phương thức tương ứng
Chọn hình thức thanh toán
Hiển thị mã QR của vé
Kiểm tra xem thanh toán thành công không?
[Thành công]
[Không thành công]
Lưu thông tin vào cơ sở dữ liệu
Hiển thị thông báo đã hủy thanh toán theo yêu cầu của người dùng
Vé
Thông báo thanh toán thành công
Hiển thị mã vé (mã QR).
Kết thúc

2.4. Hủy vé


act HUYVE
Hệ thống
Người Dùng
Bắt đầu
HIển thị thông tin vé
Chọn vào mục " HUỶ VÉ "
Xác nhận lại huỷ vé
Người dùng lựa chọn xác nhận
Thông báo bắt đầu huỷ vé
OK
Quay lại
Kết thúc
Kiểm tra giờ huỷ vé và giờ khởi hành
Không hợp lệ
Hợp lệ
VÉ
Cập nhật lại thông tin Danh sách vé của người dùng
Thông báo không thể huỷ vé
Kết thúc
Cập nhật trạng thái chuyến đi
CHUYẾN ĐI
Cập nhật lịch sử huỷ vé
LICHSUHUYVE
Thực hiện "HOÀN TIỀN"
Thông báo huỷ vé thành công
FlowFinal
FlowFinal

2.5. Tra cứu vé


act Use Case Model
Nguoi dung
He thong
Nhập thông tin vé đã mua cần tra cứu như số điện thoai, mã vé
Hiển thị giao diện
Bắt đầu
Vé
Hiển thi thông tin vé
Nếu thông tin tồn tại
Nếu thông tin không tồn tại
Thông báo không có thông tin
Kết thúc

IV. Các chức năng của hệ thống

TÊN UC	Thông tin lưu trữ	Nhu cầu truy vấn	Nhu cầu hiệu suất truy vấn	Khả năng mở rộng
Tìm kiếm chuyến đi	Thông tin chuyến đi (điểm đi, điểm đến, thời gian đi, thời gian đến, chính sách, lịch trình), thông tin loại xe, số chỗ ngồi còn trống, thông tin giá	Đọc lên thông tin chuyến đi, thông tin loại xe, chỗ ngồi, giá cả	Trễ tối đa 5s	Theo khảo sát: Có 94 điểm đến Chỉ riêng điểm đi Hồ Chí Minh đến các điểm đến còn lại khoảng gần 3000 chuyến/ngày →Dữ liệu mở rộng nhiều
Đặt vé	- Thông tin khách hàng (Họ và tên, Số điện thoại, Email) - Thông tin điểm đón trả (điểm trung chuyển, bến xe), điểm đến - Thông tin xe: Loại xe, biển số - Thông tin vé: Số ghế, ngày khởi hành, giờ, giá vé.	Đọc thông tin xe, vé Ghi xuống thông tin khách hàng	Trễ tối đa 5s	Dữ liệu mở rộng nhiều
Thanh toán	Thông tin thanh toán cho vé (mã giao dịch, số tiền thanh toán) Thông tin các phương thức thanh toán được hỗ trợ và mã khuyến mãi dành riêng cho phương thức thanh toán đó	Đọc lên danh sách phương thức thanh toán và mã khuyến mãi ứng với từng phương thức đó. Lưu thông tin giao dịch của từng vé xe.	Trễ tối đa 15s	Mở rộng ứng với số vé đã đặt
Hủy vé	Thông tin vé, lịch sử hủy vé	Đọc thông tin vé lên. Cập nhật thông tin trạng thái vé đã hủy Thêm thông tin hủy vé. Thêm thông tin giao dịch hoàn tiền	Tần suất sử dụng có thể cao. Trễ tối đa 5s cho hủy vé	Tốc độ mở rộng không cố định, cần mở rộng ứng với số lượng vé đã đặt
Tra cứu vé	Thông tin vé	Đọc thông tin vé lên.	Trễ tối đa 5s	Tốc độ mở rộng ứng với số lượng khách hàng đặt vé và có nhu cầu tra cứu thông tin.

V. Phân tích sử dụng cơ sở dữ liệu

1. Tìm kiếm

Dùng neo4j để lưu trữ thông tin chuyến đi, và các điểm đến, vì mỗi điểm có mối quan hệ với nhau nên dùng neo4j, một dạng csdl đồ thị rất hiệu quả trong việc quản lý các mối quan hệ phức tạp là một đặc trưng mà các csdl nosql khác không có, truy vấn nhanh chóng, dễ mở rộng mối quan hệ, đặc biệt lượng dữ liệu chuyến đi là rất lớn.
Và các cơ sở dữ liệu NoSQL còn lại không phù hợp cho các yêu cầu trên.

2. Đặt vé

Đặt vé thường phải xử lý hàng ngàn yêu cầu thêm mới dữ liệu mỗi ngày khi người dùng đặt vé xe. Yêu cầu này có thể tăng đột biến trong các mùa lễ tết. Còn đối với thông tin về vé thì việc cập nhật này có thể xảy ra khi có sự thay đổi về giá vé. Do đó sự cập nhật dữ liệu diễn ra thực sự không đến mức thường xuyên.
Với cấu trúc lưu trữ dữ liệu đặt vé thì ta có thể thấy đa số là loại cấu trúc phi dữ liệu. Tuy nhiên thì chúng ta có thể quản lý nó một cách có cấu trúc và có thể thay đổi cấu trúc một cách dễ dàng.
Đối sánh giữa các mô hình dữ liệu và sản phẩm tương ứng:

Redis:

• Redis là một cơ sở dữ liệu key-value store, tập trung vào hiệu suất và khả năng truy vấn nhanh.
• Tuy nhiên, với chức năng đặt vé vì cần lưu trữ dữ liệu có cấu trúc phức tạp hơn, chẳng hạn thông tin về vé, ghế, khách hàng, giá cả, và địa điểm/ ngày giờ khởi hành.
• Chỉ tập trung vào việc xuất dữ liệu cơ bản, không hỗ trợ tốt tính toán và truy vấn phức tạp.

MongoDB:

• MongoDB là một cơ sở dữ liệu document, do đó tập trung vào lưu trữ dữ liệu phi cấu và linh hoạt thay đổi về cấu trúc. Ví dụ như về vé, ghế, khách hàng, giá cả, và địa điểm/ ngày giờ khởi hành thuộc tính liên quan theo cấu trúc phức tạp mà không cần tuân theo một mô hình cố định.
• Hỗ trợ khá tốt trong việc tính toán như là tính tổng tiền đặt phòng.
• Thực hiện tốt các truy vấn phức tạp bằng các toán tử.

Cassandra:

• Cassandra là một cơ sở dữ liệu cột gia đình (column family), phân phối và mở rộng tốt.
• Dữ liệu cần được lưu trữ không quá là rộng lớn, hơn thế nữa chức năng đặt phòng/chỗ ở lại thường xuyên có nhu cầu tính toán như là tổng tiền vé.

Neo4j:

• Neo4j là một cơ sở dữ liệu đồ thị, phù hợp cho các trường hợp có nhiều lớp quan hệ phức tạp.
• Tuy nhiên, đặt vé có tính chất chủ yếu là lưu trữ thông tin về vé và các thuộc tính liên quan, không yêu cầu mô hình đồ thị quan hệ phức tạp.

Dựa trên các đặc điểm trên, MongoDB được xem là lựa chọn hợp lý nhất cho chức năng đặt vé. MongoDB hỗ trợ lưu trữ dữ liệu có cấu trúc linh hoạt, khả năng truy vấn phức tạp, và khả năng mở rộng ngang, đồng thời cung cấp hiệu suất tốt cho các tác vụ truy vấn và ghi dữ liệu.
Do đó đối với dữ liệu trong tính năng đặt vé, ta thấy sử dụng MongoDB là hợp lý nhất.

3. Thanh toán

Thanh toán thường phải xử lý hàng ngàn yêu cầu thêm mới dữ liệu mỗi ngày khi người dùng đặt vé xe và thực hiện hành động thanh toán để hoàn tất việc mua vé xe. Yêu cầu này có thể tăng đột biến trong các mùa lễ tết.
Đối sánh giữa các mô hình dữ liệu và sản phẩm tương ứng:

Redis:

• Redis là một cơ sở dữ liệu key-value store, tập trung vào hiệu suất và khả năng truy vấn nhanh.
• Tuy nhiên, với chức năng thanh toán vì cần lưu trữ dữ liệu có cấu trúc phức tạp hơn, chẳng hạn thông tin về vé, ghế, khách hàng, giá cả, và địa điểm/ ngày giờ khởi hành.
• Chỉ tập trung vào việc xuất dữ liệu cơ bản, không hỗ trợ tốt tính toán và truy vấn phức tạp.

MongoDB:

• MongoDB là một cơ sở dữ liệu document, do đó tập trung vào lưu trữ dữ liệu phi cấu và linh hoạt thay đổi về cấu trúc. Ví dụ như về vé, ghế, khách hàng, giá cả, và địa điểm/ ngày giờ khởi hành thuộc tính liên quan theo cấu trúc phức tạp mà không cần tuân theo một mô hình cố định.
• Hỗ trợ khá tốt trong việc tính toán như là tính tổng tiền đặt phòng.
• Thực hiện tốt các truy vấn phức tạp bằng các toán tử.

Cassandra:

• Cassandra là một cơ sở dữ liệu cột gia đình (column family), phân phối và mở rộng tốt.
• Dữ liệu cần được lưu trữ không quá là rộng lớn, hơn thế nữa chức năng thanh toán lại thường xuyên có nhu cầu tính toán như là tổng tiền vé.

Neo4j:

• Neo4j là một cơ sở dữ liệu đồ thị, phù hợp cho các trường hợp có nhiều lớp quan hệ phức tạp.
• Tuy nhiên, thanh toán có tính chất chủ yếu là lưu trữ thông tin về vé và các thuộc tính liên quan, không yêu cầu mô hình đồ thị quan hệ phức tạp.
Dựa trên các đặc điểm trên, MongoDB được xem là lựa chọn hợp lý nhất cho chức năng thanh toán. MongoDB hỗ trợ lưu trữ dữ liệu có cấu trúc linh hoạt, khả năng truy vấn phức tạp, và khả năng mở rộng ngang, đồng thời cung cấp hiệu suất tốt cho các tác vụ truy vấn và ghi dữ liệu.
Do đó đối với dữ liệu trong tính năng thanh toán, ta thấy sử dụng MongoDB là hợp lý nhất.

4. Tra cứu vé

Tra cứu vé thường phải xử lý hàng ngàn yêu cầu tra cứu dữ liệu mỗi ngày. Yêu cầu này có thể tăng đột biến trong các mùa lễ tết.
Với cấu trúc lưu trữ dữ liệu tra cứu vé thì ta có thể thấy đa là loại cấu trúc phi dữ liệu. Tuy nhiên thì chúng ta có thể quản lý nó một cách có cấu trúc và có thể thay đổi cấu trúc một cách dễ dàng. Hơn nữa, việc tra cứu vé sẽ cần thao tác dữ liệu nhanh chóng và hiệu quả. Điều này rất quan trọng khi phải xử lý hàng ngàn yêu cầu từ người dùng hàng ngày.
Đối sánh giữa các mô hình dữ liệu và sản phẩm tương ứng:

Redis:

• Redis là một cơ sở dữ liệu key-value store, tập trung vào hiệu suất và khả năng truy vấn nhanh.
• Tuy nhiên, với chức năng tra cứu vé cần lưu trữ dữ liệu có cấu trúc phức tạp hơn, chẳng hạn thông tin về vé, ngày giờ/ địa điểm khởi hành, ngày giờ/ địa điểm đến dự kiến, giá cả, tình trạng thanh toán.
• Chỉ tập trung vào việc xuất dữ liệu cơ bản, không hỗ trợ tốt tính toán và truy vấn phức tạp.

MongoDB:

• MongoDB là một cơ sở dữ liệu document, do đó tập trung vào lưu trữ dữ liệu phi cấu và linh hoạt thay đổi về cấu trúc. Ví dụ như về vé, ngày giờ/ địa điểm khởi hành, ngày giờ/ địa điểm đến dự kiến, giá cả, tình trạng thanh toán và các thuộc tính liên quan theo cấu trúc phức tạp mà không cần tuân theo một mô hình cố định.

Cassandra:

• Cassandra là một cơ sở dữ liệu cột gia đình (column family), phân phối và mở rộng tốt.
• Dữ liệu cần được lưu trữ không quá là rộng lớn

Neo4j:

• Neo4j là một cơ sở dữ liệu đồ thị, phù hợp cho các trường hợp có nhiều lớp quan hệ phức tạp.
• Tuy nhiên, tra cứu vé có tính chất chủ yếu là lưu trữ thông tin về vé và các thuộc tính liên quan, không yêu cầu mô hình đồ thị quan hệ phức tạp.

Dựa trên các đặc điểm trên, MongoDB được xem là lựa chọn hợp lý nhất cho chức năng tra cứu vé. MongoDB hỗ trợ lưu trữ dữ liệu có cấu trúc linh hoạt, khả năng truy vấn phức tạp, và khả năng mở rộng ngang, đồng thời cung cấp hiệu suất tốt cho các tác vụ truy vấn.
Do đó đối với dữ liệu trong tính năng tra cứu vé, ta thấy sử dụng MongoDB là hợp lý nhất.

5. Hủy vé

Trong hệ thống đặt vé xe thường phải xử lý hàng ngàn yêu cầu cập nhật dữ liệu mỗi ngày khi người dùng có nhu cầu huỷ vé. Yêu cầu này có thể tăng đột biến trong một trường hợp đặc biệt như thời tiết, dịp lễ hội.
Việc cập nhật dữ liệu khi khách hang thực hiện hành động huỷ vé không chỉ cập nhật lại trang thái vé của khách hang mà còn là việc cập nhật lại trạng thái của ghế ngồi mà khách

hang đã huỷ lựa chọn, lưu lại lịch sử huỷ vé trong hệ thống để quản lí việc theo dõi tình trạng hệ thống, thông tin về hoàn trả tiền vé cho khách hàng.
Với cấu trúc lưu trữ dữ liệu chuyến đi, vé xe, ghế ngồi, lịch sử huỷ vé thì ta có thể thấy đa là loại cấu trúc phi dữ liệu. Tuy nhiên thì chúng ta có thể quản lý nó một cách có cấu trúc và có thể thay đổi cấu trúc một cách dễ dàng. Hơn nữa, việc thực hiện huỷ vé ở sẽ cần thao tác dữ liệu nhanh chóng và hiệu quả, đồng thời tương tác với dữ liệu như là xử lý các yêu cầu thêm, cập nhật và xóa dữ liệu như trên đã đề cập cũng cần được cung cấp hiệu suất đọc/ghi tốt. Điều này rất quan trọng khi phải xử lý hàng ngàn yêu cầu từ người dùng hàng ngày.
So sánh sự hiệu quả giữa các mô hình dữ liệu và sản phẩm tương ứng:

Redis

- Redis là một cơ sở dữ liệu key-value store, tập trung vào hiệu suất và khả năng truy vấn nhanh.
- Tuy nhiên, với chức năng huỷ vé vì cần lưu trữ dữ liệu có cấu trúc phức tạp hơn, chẳng hạn thông tin về chuyến đi, vé của khách hàng, số ghế thường lưu ở dạng mảng, …
- Chỉ tập trung vào việc xuất dữ liệu cơ bản, không hỗ trợ tốt tính toán và truy vấn phức tạp, không quá tối ưu cho việc tính toán hoàn tiền.

Cassandra

- Cassandra là một cơ sở dữ liệu cột gia đình (column family), phân phối và mở rộng tốt.
- Dữ liệu cần được lưu trữ không quá là rộng lớn, hơn thế nữa chức năng huỷ vé lại thường xuyên phải tính toán số tiền hoàn trả.
- Có thể sẽ giảm hiệu suất cho hệ thống khi thực hiện câu truy vấn phức tạp như tìm một số chuyến đi bị huỷ nhiều nhất, ngày có chuyến đi bị huỷ nhiều nhất.

Neo4J

- Neo4j là một cơ sở dữ liệu đồ thị, phù hợp cho các trường hợp có nhiều lớp quan hệ phức tạp.
- Tuy nhiên, thông tin lưu trữ lịch sử huỷ vé có tính chất chủ yếu là lưu trữ thông tin về vé khách hàng, chuyến đi và các thuộc tính liên quan, không yêu cầu mô hình đồ thị quan hệ phức tạp.

MongoDB

- MongoDB là một cơ sở dữ liệu document, do đó tập trung vào lưu trữ dữ liệu phi cấu và linh hoạt thay đổi về cấu trúc. Ví dụ như về chuyến đi, vé, trạng thái, hoàn tiền và các thuộc tính liên quan theo cấu trúc phức tạp mà không cần tuân theo một mô hình cố định.
- Hỗ trợ khá tốt trong việc tính toán như là tính chi phí hoàn trả, index theo ngày, chuyến đi
- Thực hiện tốt các truy vấn phức tạp bằng các toán tử.

Kết luận: Sử dụng MongoDB cho Huỷ vé nhằm lưu trữ thông tin lịch sử huỷ vé.

VI. Mô tả dữ liệu

MongoDB

Ve: Lưu thông tin các vé xe được khách hành đặt mua.

```
{
  "MaVe": 76835,
  "MaChuyenDi": 3,
  "MaKH": 1,
  "MaTX": 7,
  "MaXe": 6,
  "SoGhe": [
    "A05",
    "A06"
  ],
  "GiaVe": 220000,
  "DiaDiemKhoiHanh": "Bến Xe Mỹ Đình",
  "DiaDiemDen": "Bến Xe Hải Phòng",
  "ThoiGian": {
    "NgayKhoiHanh": "2024-08-06",
    "GioKhoiHanh": "11:00",
    "NgayDenDuKien": "2024-08-06",
    "GioDenDuKien": "17:00"
  },
  "ThanhToan": {
    "MaGiaoDich": 320401947,
    "SoTien": "440000",
    "PhuongThucThanhToan": "MoMo"
  }
}
```


LichSuHuyVe: Lưu thông tin các vé được khách hàng mua rồi hủy sau đó.


```
{
  "_id": {
    "$oid": "668bf54f70d596d883e8fa69"
  },
  "ID": 129404,
  "MaVe": 6811,
  "MaChuyenDi": 1,
  "MaKH": 1,
  "SoGhe": [
    "A05",
    "A21"
  ],
  "NgayHuy": "07/08/2024",
  "HoanTien": 144559,
  "TrangThai": "Đã hoàn tiền"
}
```


TaiKhoan: Lưu thông tin tài khoản của khách hàng.


```
{
  "userID": 1,
  "username": "user1",
  "password": "password1",
  "email": "user1@gmail.com"
}
```

KhachHang: Lưu thông tin cá nhân của khách hàng.


```
{
  "cid": 1,
  "userID": 1,
  "hoTen": "Jane Johnson",
  "sdt": "0917000001",
  "email": "user1@gmail.com",
  "diachi": "Q 2, HCM"
}
```


Ghe: Lưu thông tin trạng thái (đã bán hoặc còn trống) của từng ghế có trong các chuyến đi.


```
{
  "MaChuyenDi": 11,
  "DSGhe": [
    {
      "SoGhe": "A01",
      "TrangThai": "Đã Bán"
    },
    ...
    {
      "SoGhe": "A22",
      "TrangThai": "Còn Trống"
    },
    {
      "SoGhe": "B01",
      "TrangThai": "Đã Bán"
    },
    ...
    {
      "SoGhe": "B22",
      "TrangThai": "Đã Bán"
    }
  ]
}
```


PhuongThucThanhToan: Lưu thông tin các phương thức thanh toán và mã giảm giá tương ứng của phương thức đó.

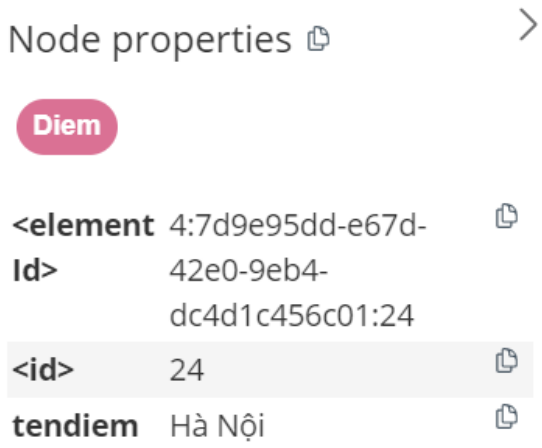

```
{
  "TenPhuongThuc": "MoMo",
  "MoTaMaGiamGia": "Hiện MoMo không có mã giảm giá",
  "ChietKhau": 0
}
```

ThongTinTaiXe: Lưu thông tin cá nhân của các tài xế trong doanh nghiệp.


```
{
  "userID": 1,
  "hoTen": "Trần Minh Anh",
  "sdt": "0967536251",
  "email": "tma@gmail.com",
  "diachi": "Q.2, TP.HCM"
}
```


Neo4j

Node **Diem**: Lưu thông tin các điểm nằm ở hai đầu chuyến đi nào đó.



Mối quan hệ **chuyendi**: Lưu thông tin chi tiết của các chuyến đi.

```
{
  "identity": 17,
  "type": "CHUYENDI",
  "properties": {
    "dauben": "Bến Xe Quận 5",
    "cuoiben": "Bến Xe Quãng Ngãi",
    "trangthai": "chưa chạy",
    "MaXe": 3,
    "thoigianhanhtrinh": "12:00",
    "loaixe": "Giường",
    "quangduong": 320,
    "giodi": "05:00",
    "giave": 220000,
    "sochuyen": 122,
    "ngaydi": "2024-08-10",
    "ngayden": "2024-08-10",
    "MaTX": 2,
    "gioden": "17:00",
    "soghetrong": 44,
    "luuy": "Cần hỗ trợ thêm thông tin vui lòng liên hệ hotline
      19006067."
  }
}
```