

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC VÀ KỸ THUẬT THÔNG TIN



UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

XÂY DỰNG CƠ SỞ DỮ LIỆU BÁN HÀNG TRỰC TUYẾN

Sinh viên thực hiện:

Lê Đăng Khoa - 23520740
Phan Trần Văn Khang - 23520708
Trần Tấn Hải - 23520680

Giảng viên:

GV Lý thuyết - Nguyễn Gia Tuấn Anh
GV Trợ giảng - Phạm Nguyễn Phúc Toàn

Thành phố Hồ Chí Minh, tháng 5 năm 2025

Mục lục

1 TỔNG QUAN HỆ THỐNG	1
1.1 Thực trạng lĩnh vực thương mại điện tử	1
1.2 Thực trạng hệ thống	2
1.2.1 Thực trạng hệ thống hiện nay (As-is)	2
1.2.2 Trạng thái hệ thống hướng đến (To-be)	3
1.2.3 Tiểu kết	4
1.3 Đối tượng sử dụng hệ thống	4
1.3.1 Đối tượng có nhu cầu xây dựng hệ thống	4
1.3.2 Người dùng cuối (End user) của hệ thống	4
1.4 Khảo sát các hệ thống tương tự	5
2 CƠ SỞ LÝ THUYẾT	7
2.1 Giới thiệu	7
2.2 MySQL	7
2.2.1 Ưu điểm	7
2.2.2 Nhược điểm	7
2.3 Redis	8
2.3.1 Ưu điểm	8
2.4 Azure	8
2.4.1 Azure Database for MySQL - Flexible Server	8
2.4.2 Azure Cache for Redis	8
3 PHÂN TÍCH HỆ THỐNG - THIẾT KẾ VÀ CÀI ĐẶT CƠ SỞ DỮ LIỆU	11
3.1 Phân tích hệ thống	11
3.1.1 Phân tích tổng quan	11
3.1.2 Phân tích triển khai đối với MySQL	13
3.1.3 Phân tích triển khai đối với Redis	13
3.2 Thiết kế cơ sở dữ liệu	22
3.3 Tạo cơ sở dữ liệu và nhập dữ liệu cho các bảng	22
3.3.1 Tạo bảng dữ liệu và khóa ngoại	22

3.3.2	Tạo và nhập dữ liệu	22
3.3.3	Tạo index	23
4	LẬP TRÌNH CƠ SỞ DỮ LIỆU	30
4.1	Lập trình thủ tục	30
4.1.1	Thủ tục sp_login	30
4.1.2	Thủ tục add_to_cart	31
4.1.3	Thủ tục get_cart	32
4.1.4	Thủ tục Make_product_for_temp_cart	33
4.1.5	Thủ tục CreateUser	36
4.2	Lập trình hàm	38
4.2.1	check_product_inventory	38
4.2.2	calculate_product_discount	38
4.2.3	calculate_order_discount	40
4.2.4	calculate_delivery_discount	41
4.2.5	calculate_loyalty_points	42
4.3	Lập trình Trigger	42
4.3.1	update_order_detail_subtotal	43
4.3.2	update_order_total_after_insert	43
4.3.3	update_order_total_after_update	44
4.3.4	update_order_total_after_delete	45
4.3.5	update_discount_usage	46
4.3.6	update_delivery_discount_usage	46
4.3.7	update_inventory_after_order	47
4.3.8	before_review_insert	48
4.3.9	before_order_status_update	49
4.3.10	update_order_detail_price	51
4.4	Lập trình các chức năng triển khai caching với Redis	51
4.4.1	Lưu trữ phiên đăng nhập	51
4.4.2	Lưu trữ giỏ hàng	53
4.4.3	Lưu trữ sản phẩm thịnh hành	54
4.5	Thực nghiệm Cache	54
4.5.1	So sánh hiệu suất khi login bằng cache và không sử dụng cache	54
4.5.2	Thực nghiệm thủ tục nhiều sản phẩm	56
4.5.3	Thực nghiệm sản phẩm thịnh hành	57
4.6	Lập trình View	57
4.6.1	view_order_full_info	57

4.6.2	view_product_inventory	58
4.6.3	view_reviews	58
4.6.4	view_customer_info	59
4.6.5	view_bill_summary	59
4.6.6	view_user_employee_admin	60
5	Quản Trị Cơ Sở Dữ Liệu	61
5.1	Tạo login, user và role	61
5.2	Phân user vào role	61
5.3	Phân quyền	61
5.4	Backup và Restore cơ sở dữ liệu	65
5.4.1	Backup	65
5.4.2	Restore	66
5.5	Export dữ liệu và quản lý hiệu suất trên nền tảng Cloud	68
5.5.1	Import dữ liệu bảng product:	68
5.5.2	Export dữ liệu bảng user thành file .csv:	69
5.5.3	Export dữ liệu bảng orders theo truy vấn cụ thể:	70
5.5.4	Quản lý hiệu suất cơ sở dữ liệu trên Azure	71
6	TRÌNH BÀY THÔNG TIN	75
6.1	Thiết kế menu	75
6.1.1	Menu chính của hệ thống	75
6.1.2	Các mục chính của Menu	75
6.2	Thiết kế form	76
6.2.1	Form nhập thông tin khách hàng	76
6.2.2	Form thêm/sửa thông tin sản phẩm	77
6.2.3	Form thêm/sửa thông tin nhân viên	77
6.2.4	Form đăng ký	78
6.2.5	Form đăng nhập	79
6.3	Thiết kế báo cáo (Report)	79
6.3.1	Report doanh thu	79
6.3.2	Report khách hàng	79
6.3.3	Report hàng tồn kho	79
6.4	Thiết kế phần trợ giúp (Help)	80
7	KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	82
7.1	Kết quả đạt được	82
7.1.1	Thiết kế và triển khai cơ sở dữ liệu	82

7.1.2	Tối ưu hóa hiệu suất	82
7.1.3	Triển khai lên nền tảng Azure	82
7.2	Ưu điểm và hạn chế	82
7.2.1	Ưu điểm	82
7.2.2	Hạn chế	83
7.3	Hướng phát triển	83
7.3.1	Cải tiến cơ sở dữ liệu	83
7.3.2	Tăng cường hiệu suất và khả năng mở rộng	83
7.3.3	Tăng cường bảo mật và quản lý dữ liệu	84
7.3.4	Triển khai các mô hình AI/ML	84
7.3.5	Triển khai tính năng nâng cao	84
	Tài liệu đính kèm	
	Lời cảm ơn	

Danh sách hình vẽ

1.1	Doanh số thương mại điện tử toàn cầu 2021-2027	1
3.1	Sơ đồ Sequence Diagram tổng quan của hệ thống	11
3.2	Sơ đồ triển khai lưu trữ session đăng nhập	16
3.3	Sơ đồ triển khai lưu trữ giỏ hàng - 1	17
3.4	Sơ đồ triển khai lưu trữ giỏ hàng - 2	18
3.5	Sơ đồ triển khai lưu trữ sản phẩm thị trường	20
3.6	Sơ đồ erd	26
3.7	Lược đồ quan hệ - 1	27
3.8	Lược đồ quan hệ - 2	27
3.9	Lược đồ quan hệ - 3	27
4.1	Dữ liệu ban đầu của giỏ hàng	32
4.2	Dữ liệu của giỏ hàng sau khi thêm product	32
4.3	Số lượng sản phẩm được tự động cập nhật	32
4.4	Lấy đơn hàng có trạng thái processing	35
4.5	Thực hiện complete_payment với đơn hàng 5	35
4.6	Hóa đơn đã hủy	36
4.7	Tạo người dùng thành công.	37
4.8	Truy vấn người dùng mới tạo.	37
4.9	Số lượng tồn kho đủ yêu cầu.	39
4.10	Số lượng tồn kho không đủ yêu cầu.	39
4.11	Giá trị hợp lệ, trả về số tiền mà sản phẩm đó được giảm.	40
4.12	Giá trị không hợp lệ, số tiền được giảm là 0.	40
4.13	Giá trị hợp lệ, trả về số tiền giảm cho đơn hàng.	41
4.14	Giá trị không hợp lệ, đơn hàng không được giảm.	41
4.15	Giá trị hợp lệ, trả về số tiền vận chuyển được giảm.	42
4.16	Giá trị không hợp lệ.	42
4.17	Khách hàng nhận được 1 điểm thành viên tương đương với 10000 VND giá trị đơn hàng.	42
4.18	Giá trị của subtotal sau khi cập nhật.	43

4.19 Giá trị của subtotal sau khi thêm bản ghi.	44
4.20 Tổng tiền trước khi cập nhật.	45
4.21 Tổng tiền sau khi cập nhật.	45
4.22 Tổng tiền trước khi xóa bản ghi.	45
4.23 Tổng tiền sau khi xóa bản ghi.	46
4.24 Dữ liệu mã giảm giá ban đầu.	46
4.25 Kết quả sau khi kiểm tra lại mã giảm giá.	46
4.26 Kiểm tra dữ liệu mã giảm giá khi chưa sử dụng.	47
4.27 Dữ liệu mã giảm giá sau khi đã sử dụng.	47
4.28 Kho trước khi thêm đơn hàng.	48
4.29 Kiểm tra lại tồn kho.	48
4.30 Nếu khách hàng chưa mua sản phẩm thì không thể review.	49
4.31 Dữ liệu có thể được thêm vào bảng review.	49
4.32 Kiểm tra dữ đánh giá đã thêm vào bảng review.	49
4.33 Trạng thái của đơn hàng được sửa thành công.	50
4.34 Không thể chỉnh sửa trạng thái đơn hàng.	50
4.35 Price được thêm vào order_detail	51
4.36 So sánh thời gian thực thi trung bình.	55
4.37 Kết quả dữ liệu bảng order.	56
4.38 Kết quả dữ liệu bảng order_detail.	56
5.1 Dịch vụ thực hiện nhiều Snapshot Backup mỗi ngày.	66
5.2 Đặt tên cho server mới.	67
5.3 Cấu hình tùy chọn mạng.	67
5.4 Bắt đầu khôi phục.	68
5.5 Số lượng discount ban đầu.	69
5.6 Chọn bảng discount để import.	69
5.7 Chọn source là file csv.	70
5.8 Chọn file đầu vào.	70
5.9 Tạo mapping.	71
5.10 Điều chỉnh Data load setting.	71
5.11 Xác nhận import và kiểm tra số lượng bảng dữ liệu.	72
5.12 Chọn format export data.	72
5.13 Kết quả file user.csv.	72
5.14 Dữ liệu bảng export theo truy vấn.	73
5.15 Giám sát truy vấn.	73
5.16 Theo dõi tài nguyên.	73

5.17 Thiết lập cảnh báo.	74
6.1 Menu chính của hệ thống	75
6.2 Các mục chính của Menu.	76
6.3 Form nhập thông tin khách hàng.	76
6.4 Form thêm/sửa thông tin sản phẩm	77
6.5 Form thêm/sửa thông tin nhân viên.	78
6.6 Form đăng ký tài khoản cho khách hàng.	78
6.7 Form đăng nhập tài khoản.	79
6.8 Báo cáo doanh thu.	80
6.9 Báo cáo khách hàng.	80
6.10 Báo cáo hàng tồn kho.	81
6.11 Phân trợ giúp.	81

Danh sách bảng

1.1	Các tiêu chí xây dựng hệ thống bán hàng trực tuyến.	6
3.1	Các mối kết hợp giữa các thực thể trong hệ thống	24
3.2	Danh sách các bảng và thuộc tính trong cơ sở dữ liệu	25
3.3	Danh sách các tập tin dữ liệu mô phỏng	28
3.4	Danh sách các chỉ mục (index) được triển khai trong hệ thống	29
5.1	Phân quyền CRUD theo bảng dữ liệu và vai trò	62
5.2	Số lượng dòng trong các bảng dữ liệu chính	68

Tóm tắt nội dung

Đề tài này trình bày giải pháp xây dựng hệ thống quản lý bán hàng trực tuyến với mục tiêu tối ưu hiệu suất truy vấn và đảm bảo khả năng mở rộng, bảo mật trong môi trường thực tế. Hệ thống được thiết kế trên nền tảng cơ sở dữ liệu quan hệ **MySQL** để lưu trữ dữ liệu giao dịch và thông tin người dùng. Bên cạnh đó, **Redis** được tích hợp nhằm tăng tốc độ truy xuất đối với các dữ liệu tạm thời như phiên đăng nhập, giỏ hàng và sản phẩm thịnh hành thông qua mô hình caching. Cuối cùng, toàn bộ hệ thống được triển khai trên nền tảng **Azure Cloud** nhằm đảm bảo tính sẵn sàng cao, dễ dàng mở rộng và hỗ trợ các chức năng như sao lưu, phục hồi, giám sát hiệu suất. Kết quả thực nghiệm cho thấy hệ thống đạt hiệu năng cao hơn đáng kể khi áp dụng caching bằng Redis, đồng thời đảm bảo tính an toàn và khả năng phục hồi dữ liệu nhờ dịch vụ quản lý của Azure.

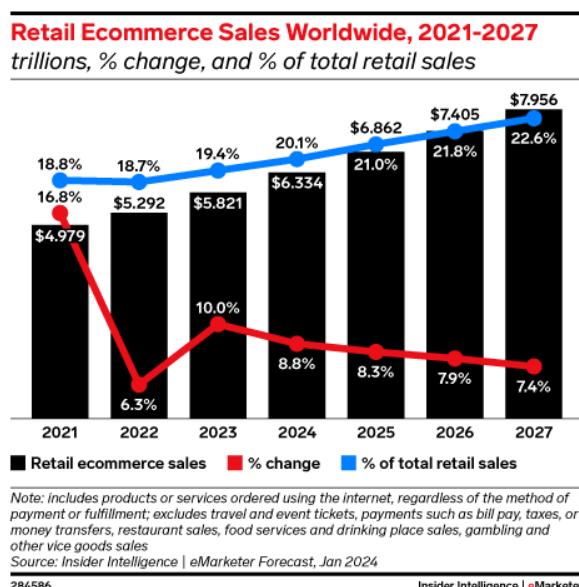
Chương 1 TỔNG QUAN HỆ THỐNG

1.1. Thực trạng lĩnh vực thương mại điện tử

Trong những thập kỷ qua, sự phát triển mạnh mẽ của công nghệ thông tin đã tạo ra những bước tiến đột phá trong mọi lĩnh vực của đời sống, từ giáo dục, y tế đến sản xuất và kinh doanh. Sự ra đời của Internet, trí tuệ nhân tạo, dữ liệu lớn (Big Data) và điện toán đám mây đã mở ra những cơ hội chưa từng có, giúp con người kết nối, xử lý thông tin và tự động hóa nhiều quy trình phức tạp.

Trong bối cảnh đó, thương mại điện tử trở thành một xu hướng tất yếu, tận dụng tối đa những lợi thế mà công nghệ mang lại để tối ưu hóa hoạt động kinh doanh. Không chỉ giúp doanh nghiệp mở rộng thị trường, giảm chi phí vận hành, thương mại điện tử còn tạo ra những trải nghiệm mua sắm thông minh, cá nhân hóa dựa trên dữ liệu người dùng.

Thương mại điện tử toàn cầu đang phát triển mạnh mẽ, với doanh thu đạt 6,33 nghìn tỷ USD vào năm 2024 và dự kiến tăng lên 7,96 nghìn tỷ USD vào năm 2027, theo báo cáo từ eMarketer [1]



Hình 1.1: Doanh số thương mại điện tử toàn cầu 2021-2027

Tốc độ tăng trưởng hàng năm (CAGR) khoảng 7,8% từ 2024 đến 2027, phản ánh xu hướng người tiêu dùng ngày càng ưa chuộng mua sắm trực tuyến, đặc biệt nhờ sự phổ biến của thiết bị di động và Internet.

Hành vi người tiêu dùng: Người tiêu dùng Việt Nam đang chuyển dịch mạnh mẽ sang mua sắm trực tuyến, đặc biệt cho các mặt hàng thiết yếu như thực phẩm và nhu yếu phẩm. Tỷ lệ chi tiêu cho thực phẩm tăng từ 50% lên 54% trong quý III/2024, với xu hướng chuyển từ chợ, tạp hóa sang các nền tảng như Shopee, TikTok Shop để tận dụng ưu đãi và thanh toán thuận tiện [2].

Thách thức và cơ hội: Ngành thương mại điện tử đối mặt với sự cạnh tranh cao giữa các nền tảng lớn như Shopee, TikTok Shop, Lazada, Tiki, Sendo, đòi hỏi doanh nghiệp phải đổi mới và thích ứng với xu hướng. Đầu tư vào cơ sở hạ tầng, đặc biệt là công nghệ và logistics, là cần thiết để đảm bảo phát triển bền vững.

Về quy định, ngành đã đóng góp đáng kể vào ngân sách nhà nước, với doanh thu thuế từ thương mại điện tử tăng 22% so với năm 2023 (khoảng 108.000 tỷ VND từ tháng 1 đến 11/2024), nhưng việc tuân thủ, đặc biệt với nhà cung cấp nước ngoài, là một thách thức. Tuy nhiên, đây cũng là cơ hội để phát triển thị trường minh bạch và bền vững.

Các nền tảng dẫn đầu: Các nền tảng như Shopee, TikTok Shop, Lazada, Tiki, và Sendo đang chiếm ưu thế, với TikTok Shop ghi nhận sự tăng trưởng đáng kể (tăng 15% thị phần trong năm 2024). Những nền tảng này không chỉ cung cấp nền tảng bán hàng mà còn hỗ trợ tiếp thị, logistics, và thanh toán, tạo ra một hệ sinh thái toàn diện.

Tầm nhìn tương lai: Với tốc độ tăng trưởng ấn tượng và các xu hướng công nghệ mới, thương mại điện tử tại Việt Nam đang trên đà trở thành một trong những động lực chính của nền kinh tế số. Đến năm 2025, ngành dự kiến chiếm 10% tổng doanh thu bán lẻ, đóng góp đáng kể vào GDP của nền kinh tế số, và tiếp tục mở rộng với sự tham gia của các doanh nghiệp trong và ngoài nước.

1.2. Thực trạng hệ thống

1.2.1. Thực trạng hệ thống hiện nay (As-is)

Dựa trên các nguồn thông tin gần đây, tình hình công nghệ thông tin (CNTT) trong thương mại điện tử tại Việt Nam năm 2025 có những đặc điểm nổi bật sau:

Ứng dụng công nghệ:

- AI và Machine Learning (ML):** Được sử dụng để cá nhân hóa trải nghiệm người dùng (gợi ý sản phẩm, tìm kiếm thông minh), quản lý kho và tối ưu hóa chuỗi cung ứng. Các nền tảng lớn như Shopee và TikTok Shop đã áp dụng AI để phân tích hành vi khách hàng và dự đoán nhu cầu. Tuy nhiên, chỉ khoảng 20% doanh nghiệp thương mại điện tử vừa và nhỏ (SMEs) ứng dụng AI/ML do chi phí triển khai cao [3].
- Thực tế tăng cường (AR):** Một số nền tảng bắt đầu tích hợp AR để hỗ trợ trải nghiệm xem trước sản phẩm (ví dụ: thử đồ áo, đặt nội thất ảo), nhưng chủ yếu giới hạn trong các danh mục như thời trang và nội thất. Tỷ lệ triển khai còn dưới 10% [4].

-
- **Blockchain:** Ứng dụng vẫn còn sơ khai, chủ yếu dùng để đảm bảo tính minh bạch trong chuỗi cung ứng và xác thực nguồn gốc sản phẩm. Các nền tảng nội địa như Tiki, Sendo chưa áp dụng công nghệ này.
 - **Tìm kiếm bằng giọng nói:** Đã tích hợp trên các nền tảng lớn thông qua trợ lý ảo như Google Assistant, nhưng chưa phổ biến do hạn chế hỗ trợ tiếng Việt và mức độ nhận thức người dùng còn thấp.

Cơ sở hạ tầng CNTT:

- **Điện toán đám mây (Cloud Computing):** Các nền tảng lớn sử dụng dịch vụ đám mây như AWS, Google Cloud để lưu trữ và xử lý dữ liệu, giúp tăng khả năng mở rộng. Tuy nhiên, phần lớn SMEs vẫn sử dụng hạ tầng tại chỗ (on-premise), hạn chế khả năng xử lý dữ liệu lớn và tính linh hoạt.
- **Trung tâm dữ liệu:** Việt Nam hiện có hơn 20 trung tâm dữ liệu, nhưng chỉ đáp ứng khoảng 30% nhu cầu trong nước. Phần lớn các doanh nghiệp vẫn phụ thuộc vào nhà cung cấp quốc tế như AWS, Microsoft [5].

1.2.2. Trạng thái hệ thống hướng đến (To-be)

Dự báo trạng thái lý tưởng mà hệ thống thương mại điện tử tại Việt Nam hướng tới trong tương lai như sau:

Ứng dụng công nghệ:

- 100% nền tảng thương mại điện tử tích hợp AI/ML để cá nhân hóa trải nghiệm người dùng, tối ưu logistics và dự báo nhu cầu thị trường.
- AR trở thành tiêu chuẩn trong các ngành hàng như thời trang, nội thất và mỹ phẩm, với ít nhất 50% nền tảng áp dụng.
- Blockchain được sử dụng rộng rãi để minh bạch hóa chuỗi cung ứng, xác thực sản phẩm và bảo mật giao dịch.
- Tìm kiếm bằng giọng nói hỗ trợ tiếng Việt mượt mà, chiếm khoảng 20% tổng lượt tìm kiếm trên các nền tảng.

Cơ sở hạ tầng CNTT:

- 80% doanh nghiệp, bao gồm cả SMEs, chuyển đổi sang dịch vụ đám mây để tăng khả năng mở rộng và tối ưu chi phí.
- Trung tâm dữ liệu trong nước đáp ứng ít nhất 70% nhu cầu lưu trữ, giảm sự phụ thuộc vào các nhà cung cấp quốc tế.
- Kết nối Internet ổn định trên toàn quốc, bao gồm cả khu vực nông thôn, với tốc độ trung bình đạt 120 Mbps và độ trễ dưới 20 ms.

1.2.3. Tiêu kết

Thực trạng hiện nay cho thấy các doanh nghiệp vừa và nhỏ (SMEs) trong lĩnh vực thương mại điện tử tại Việt Nam đang gặp nhiều hạn chế về cơ sở hạ tầng CNTT, đặc biệt là tỷ lệ ứng dụng AI/ML chỉ chiếm khoảng 20%. Điều này đặt ra yêu cầu cấp thiết trong việc tổ chức, thu thập, xử lý và phân tích dữ liệu một cách hiệu quả để phục vụ cho AI/ML.

Bên cạnh đó, sự phụ thuộc vào hạ tầng tại chỗ làm giảm khả năng mở rộng và hiệu suất hệ thống. Việc chuyển dịch lên nền tảng điện toán đám mây như AWS hoặc Google Cloud là cần thiết để đảm bảo tính linh hoạt, khả năng mở rộng trong các sự kiện cao điểm như lễ hội mua sắm, đồng thời giảm thiểu chi phí vận hành.

Trong bối cảnh cạnh tranh ngày càng khốc liệt, việc tối ưu hóa trải nghiệm người dùng là yếu tố then chốt. Một trong những giải pháp hiệu quả để đạt được điều này chính là sử dụng cơ chế caching. Caching giúp giảm tải hệ thống cơ sở dữ liệu chính, đặc biệt vào những thời điểm cao điểm, góp phần duy trì sự ổn định và hoạt động liên tục cho hệ thống.

Kết luận lý do chọn đề tài: Chính vì những lý do trên, nhóm em quyết định chọn đề tài “*Xây dựng cơ sở dữ liệu quản lý bán hàng trực tuyến kết hợp MySQL và caching bằng Redis triển khai trên nền tảng Cloud Computing*”. Nhóm hi vọng đồ án sẽ đóng góp thiết thực vào việc giải quyết những vấn đề cấp bách của hệ thống thương mại điện tử hiện nay.

1.3. Đối tượng sử dụng hệ thống

1.3.1. Đối tượng có nhu cầu xây dựng hệ thống

Đây là những cá nhân hoặc tổ chức đóng vai trò là chủ sở hữu hệ thống, có nhu cầu đầu tư, phát triển và vận hành hệ thống bán hàng trực tuyến để phục vụ mục tiêu kinh doanh.

- **Doanh nghiệp bán lẻ / cửa hàng kinh doanh:** Muốn chuyển đổi số hoạt động bán hàng, mở rộng kênh phân phối trực tuyến.
- **Cá nhân hoặc nhóm khởi nghiệp thương mại điện tử:** Tìm cách tiếp cận thị trường online và xây dựng thương hiệu nhanh chóng.
- **Các nhà bán lẻ trên các sàn thương mại điện tử lớn như Shopee, Lazada...:** Muốn tối ưu chi phí quảng bá sản phẩm, chi phí đặt sản phẩm trên các nền tảng này và muốn tạo đặc trưng thương hiệu riêng.

1.3.2. Người dùng cuối (End user) của hệ thống

- **Người dùng phổ thông (khách hàng):**

- Truy cập hệ thống để tìm kiếm, xem, so sánh và mua sản phẩm.
- Tạo tài khoản, đăng nhập, đánh giá, quản lý đơn hàng và thanh toán.

-
- Sử dụng các tính năng như giỏ hàng, wishlist, khuyến mãi...

- **Người bán hàng (Seller / Shop Admin):**

- Quản lý gian hàng, thêm mới và cập nhật thông tin sản phẩm.
- Theo dõi và xử lý đơn hàng, hỗ trợ khách hàng.
- Quản lý tồn kho, chương trình khuyến mãi và phản hồi.

- **Quản trị viên hệ thống (Admin):**

- Quản lý toàn bộ hệ thống: người dùng, người bán, đơn hàng, cấu hình nền tảng.
- Phân quyền tài khoản, xử lý tranh chấp hoặc vi phạm.
- Theo dõi báo cáo, doanh thu, thống kê hoạt động.

1.4. Khảo sát các hệ thống tương tự

Từ việc khảo sát các hệ thống liên quan bao gồm:

- Trang Web bán hàng của CellPhoneS.
- Trang Web bán hàng của Điện Máy Xanh.
- Trang Web bán hàng Gear VN.

Nhóm chúng tôi đúc kết được các tiêu chí xây dựng một hệ thống bán hàng trực tuyến như bảng bên dưới.

Bảng 1.1: Các tiêu chí xây dựng hệ thống bán hàng trực tuyến.

STT	Nhóm Chức Năng	Tiêu Chí
1	Người dùng (Khách hàng)	Đăng ký & Đăng nhập
2	Người dùng (Khách hàng)	Quản lý Hồ sơ cá nhân
3	Người dùng (Khách hàng)	Tìm kiếm & Bộ lọc sản phẩm
4	Người dùng (Khách hàng)	Xem Chi tiết Sản phẩm
5	Người dùng (Khách hàng)	Giỏ hàng & Wishlist
6	Người dùng (Khách hàng)	Đặt hàng & Thanh toán
7	Người dùng (Khách hàng)	Theo dõi Đơn hàng
8	Người dùng (Khách hàng)	Đánh giá & Nhận xét
9	Quản trị / Người bán	Quản lý Sản phẩm
10	Quản trị / Người bán	Quản lý Đơn hàng
11	Quản trị / Người bán	Quản lý Người dùng
12	Quản trị / Người bán	Quản lý Thanh toán
13	Quản trị / Người bán	Thống kê & Báo cáo
14	Hỗ trợ	Mã Giảm giá & Khuyến mãi
15	Hỗ trợ	Hệ thống Vận chuyển
16	Hỗ trợ	Hỗ trợ Khách hàng
, 17	Hỗ trợ	Blog & Tin tức
18	Kỹ thuật	Responsive UI
19	Kỹ thuật	API & Tích hợp
20	Kỹ thuật	Tối ưu Hiệu năng & Caching
21	Kỹ thuật	Bảo mật & Phân quyền
22	Kỹ thuật	Logging & Giám sát
23	Hỗ trợ	Tích hợp bên thứ ba

Chương 2 CƠ SỞ LÝ THUYẾT

2.1. Giới thiệu

Hệ thống bán hàng trực tuyến (e-commerce) đòi hỏi các công nghệ mạnh mẽ để xử lý dữ liệu giao dịch, tối ưu hóa hiệu suất, và đảm bảo khả năng mở rộng. Chương này tập trung vào phân tích ưu và nhược điểm của MySQL, Redis, và Azure khi triển khai một hệ thống bán hàng trực tuyến. MySQL là hệ quản trị cơ sở dữ liệu quan hệ (RDBMS), Redis là cơ sở dữ liệu lưu trữ trong bộ nhớ (in-memory data store), và Azure là nền tảng đám mây của Microsoft. Sự kết hợp của ba công nghệ này có thể tạo ra một hệ thống mạnh mẽ, nhưng cũng đi kèm với những thách thức cần được xem xét.

2.2. MySQL

MySQL là một trong những hệ quản trị cơ sở dữ liệu quan hệ phổ biến nhất, được sử dụng rộng rãi trong các ứng dụng thương mại điện tử để lưu trữ dữ liệu có cấu trúc như thông tin sản phẩm, đơn hàng, và khách hàng.

2.2.1. Ưu điểm

- Khả năng lưu trữ dữ liệu bền vững và truy vấn phức tạp:** MySQL được thiết kế để đảm bảo tính toàn vẹn dữ liệu và hỗ trợ các truy vấn SQL phức tạp, rất phù hợp cho việc quản lý dữ liệu giao dịch như đơn hàng, lịch sử mua hàng, và danh mục sản phẩm.
- Cộng đồng hỗ trợ lớn và tài liệu phong phú:** Với hàng triệu người dùng trên toàn cầu, MySQL có một cộng đồng phát triển mạnh mẽ, cung cấp tài liệu chi tiết và các diễn đàn hỗ trợ.
- Tích hợp với Azure:** Azure Database for MySQL flexible server cung cấp một phiên bản quản lý của MySQL, giảm bớt gánh nặng vận hành và tăng cường khả năng tích hợp với các dịch vụ đám mây khác.

2.2.2. Nhược điểm

- Hiệu suất kém trong các tình huống truy cập cao:** MySQL có thể chậm khi xử lý các yêu cầu đọc/ghi liên tục, chẳng hạn như quản lý phiên người dùng hoặc xử lý giao dịch thời gian thực.
- Truy vấn trên chỉ mục thứ cấp tồn thời gian:** Cấu trúc bảng của MySQL khiến các truy vấn trên chỉ mục thứ cấp trở nên chậm, ảnh hưởng đến hiệu suất tìm kiếm sản phẩm hoặc xử lý dữ liệu lớn.

-
- **Khó khăn với dữ liệu không cấu trúc:** MySQL không phù hợp để xử lý dữ liệu không cấu trúc hoặc dữ liệu lớn mà các hệ thống thương mại điện tử hiện đại thường yêu cầu.

2.3. Redis

Redis (Remote Dictionary Server) là một hệ thống lưu trữ dữ liệu mã nguồn mở, hoạt động chủ yếu trong bộ nhớ (in-memory data structure store). Nó được thiết kế để hoạt động như một cơ sở dữ liệu, bộ nhớ đệm (cache), hoặc trung gian tin nhắn (message broker).

2.3.1. Ưu điểm

- **Tốc độ xử lý cực nhanh và độ trễ thấp:** Redis hỗ trợ nhiều cấu trúc dữ liệu như chuỗi (strings), bảng băm (hashes), danh sách (lists), tập hợp (sets), và tập hợp sắp xếp (sorted sets). [6]
- **Ứng dụng trong thương mại điện tử:** Redis được sử dụng để quản lý phiên người dùng, lưu trữ bộ nhớ đệm, cập nhật tồn kho thời gian thực, và cung cấp các tính năng cá nhân hóa như khuyến nghị sản phẩm. [6]

2.4. Azure

Azure là một nền tảng đám mây toàn diện do Microsoft phát triển, cung cấp hơn 200 sản phẩm và dịch vụ để hỗ trợ xây dựng, triển khai và quản lý ứng dụng trên nhiều môi trường.

2.4.1. Azure Database for MySQL - Flexible Server

Azure Database for MySQL - Flexible Server là một dịch vụ cơ sở dữ liệu quan hệ được quản lý hoàn toàn, dựa trên MySQL Community Edition.

Đặc điểm nổi bật:

- **Khả năng sẵn sàng cao:** Hỗ trợ zone-redundant và same-zone HA, với failover tự động.
- **Bảo vệ dữ liệu:** Sao lưu tự động với khả năng khôi phục tại thời điểm cụ thể (point-in-time restore) trong vòng 35 ngày.
- **Hiệu suất và mở rộng:** Đảm bảo hiệu suất ổn định với khả năng mở rộng linh hoạt trong vài giây.

2.4.2. Azure Cache for Redis

Azure Cache for Redis là một dịch vụ lưu trữ dữ liệu trong bộ nhớ được quản lý hoàn toàn, dựa trên Redis mã nguồn mở. Dịch vụ này giúp tăng tốc độ truy cập dữ liệu, cải thiện hiệu suất và khả năng mở rộng của ứng dụng, đặc biệt phù hợp cho các kịch bản cần xử lý hàng triệu yêu cầu mỗi giây với độ trễ dưới mili giây.

Tính năng nổi bật của Azure Cache for Redis

Hiệu suất cao và độ trễ thấp:

- Dữ liệu được lưu trữ trong RAM thay vì ổ đĩa, cho phép xử lý hàng triệu yêu cầu mỗi giây với độ trễ dưới một mili giây. Điều này rất hữu ích cho các tác vụ như hiển thị danh mục sản phẩm, giỏ hàng, hoặc xử lý thanh toán trong thời gian thực.
- Ví dụ, trong hệ thống bán hàng trực tuyến, Azure Cache for Redis có thể lưu trữ tạm thời danh sách sản phẩm được truy cập thường xuyên, giảm tải cho cơ sở dữ liệu chính.

Hỗ trợ cấu trúc dữ liệu phong phú:

- Redis hỗ trợ các kiểu dữ liệu như chuỗi (strings), danh sách (lists), tập hợp (sets), bảng băm (hashes), và tập hợp được sắp xếp (sorted sets). Điều này giúp lưu trữ và quản lý dữ liệu phức tạp như thông tin giỏ hàng, trạng thái phiên người dùng (session state), hoặc xếp hạng sản phẩm.
- Ví dụ: Bạn có thể dùng sorted sets để lưu trữ danh sách sản phẩm bán chạy hoặc hashes để quản lý thông tin chi tiết của đơn hàng.

Khả năng mở rộng linh hoạt:

- Azure Cache for Redis cung cấp các tầng dịch vụ (Basic, Standard, Premium, Enterprise, Enterprise Flash) với khả năng mở rộng theo nhu cầu. Bạn có thể bắt đầu với quy mô nhỏ và tăng dung lượng lưu trữ hoặc hiệu suất khi lưu lượng truy cập tăng, chẳng hạn trong các sự kiện giảm giá lớn.
- Tính năng như Redis Cluster (trong tầng Premium) cho phép phân mảnh dữ liệu trên nhiều node, hỗ trợ khối lượng công việc lớn lên đến 1.2TB.

Tích hợp với các dịch vụ Azure:

- Dịch vụ này bổ sung hoàn hảo cho các cơ sở dữ liệu như Azure SQL Database hoặc Azure Cosmos DB, giúp tăng tốc truy vấn và giảm chi phí vận hành. Ví dụ, kết quả truy vấn sản phẩm từ Cosmos DB có thể được lưu vào cache để truy cập nhanh hơn.
- Hỗ trợ các mô hình như cache-aside, lưu trữ phiên (session store), hoặc hàng đợi tin nhắn (message queue) để quản lý luồng đơn hàng.

Bảo mật và độ tin cậy:

- Hỗ trợ mã hóa SSL/TLS cho dữ liệu truyền tải và tích hợp với Azure Virtual Network để hạn chế truy cập. Điều này đảm bảo an toàn cho dữ liệu nhạy cảm như thông tin khách hàng hoặc giao dịch.

-
- Các tầng Standard và Premium cung cấp cấu hình dự phòng (replication) với hai máy ảo đồng bộ dữ liệu, đảm bảo độ ổn định tối đa. Tầng Premium còn hỗ trợ lưu trữ liên tục (persistence) và sao lưu dữ liệu để phục hồi khi xảy ra sự cố.

Tính năng nâng cao:

- Hỗ trợ các mô-đun như RedisBloom, RediSearch, RedisJSON, và RedisTimeSeries, giúp phân tích dữ liệu, tìm kiếm sản phẩm, hoặc xử lý dữ liệu thời gian thực (như theo dõi lượt xem sản phẩm).
- Tính năng publish/subscribe cho phép định tuyến tin nhắn thời gian thực, hữu ích trong việc cập nhật trạng thái đơn hàng hoặc thông báo khuyến mãi.

Ứng dụng thực tế trong triển khai hệ thống bán hàng trực tuyến

- **Hỗ trợ Caching để Tăng Tốc Hiệu Suất:** Lưu trữ kết quả truy vấn phổ biến, giảm tải cơ sở dữ liệu.
- **Quản lý Session Người Dùng:** Lưu trữ thông tin đăng nhập và trạng thái phiên.
- **Xử lý Giao Dịch và Thanh Toán Thời Gian Thực:** Đảm bảo thông tin giỏ hàng và quá trình thanh toán diễn ra nhanh chóng.
- **Tìm Kiếm và Đề Xuất Sản Phẩm:** Cung cấp kết quả tìm kiếm nhanh và gợi ý sản phẩm cá nhân hóa.

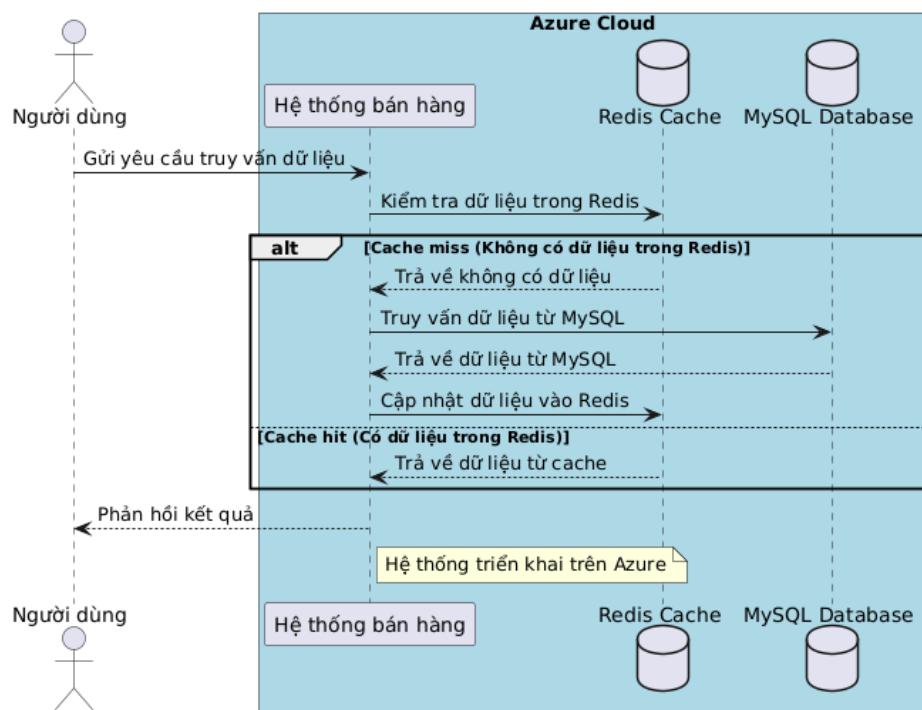
Chương 3 PHÂN TÍCH HỆ THỐNG - THIẾT KẾ VÀ CÀI ĐẶT CƠ SỞ DỮ LIỆU

3.1. Phân tích hệ thống

Do giới hạn về phạm vi môn học và thời gian thực hiện nên nhóm chúng tôi chỉ thực hiện xây dựng cơ sở dữ liệu cho hệ thống bán hàng trực tuyến.

3.1.1. Phân tích tổng quan

Dưới đây là sơ đồ sequence diagram tổng quan của hệ thống:



Hình 3.1: Sơ đồ Sequence Diagram tổng quan của hệ thống

Mô tả sơ đồ UML Sơ đồ UML mô tả luồng tương tác giữa các thành phần trong một hệ thống bán hàng trực tuyến được triển khai trên nền tảng Azure Cloud. Sơ đồ tập trung vào cách hệ thống xử lý yêu cầu truy vấn dữ

liệu từ người dùng, tận dụng Azure Cache for Redis để tăng tốc độ truy cập và Azure Database for MySQL làm cơ sở dữ liệu chính.

Các thành phần chính

- **Người dùng (User)**: Đại diện cho khách hàng hoặc người sử dụng hệ thống bán hàng trực tuyến, gửi yêu cầu truy vấn dữ liệu (ví dụ: xem danh mục sản phẩm, giỏ hàng, hoặc thông tin đơn hàng).
- **Hệ thống bán hàng (System)**: Thành phần trung tâm, chịu trách nhiệm xử lý yêu cầu từ người dùng, tương tác với Redis và MySQL để lấy dữ liệu.
- **Redis Cache (Redis)**: Đại diện cho Azure Cache for Redis, lưu trữ dữ liệu thường xuyên truy cập trong bộ nhớ để giảm độ trễ và tăng hiệu suất.
- **MySQL Database (MySQL)**: Đại diện cho Azure Database for MySQL - Flexible Server, lưu trữ dữ liệu lâu dài của hệ thống (ví dụ: thông tin sản phẩm, đơn hàng, hoặc tài khoản người dùng).
- **Azure Cloud**: Hộp bao quanh các thành phần chính (System, Redis, MySQL), thể hiện rằng hệ thống được triển khai trên nền tảng Azure.

Luồng tương tác

1. Người dùng gửi yêu cầu:

- Người dùng gửi một yêu cầu truy vấn dữ liệu tới Hệ thống bán hàng (ví dụ: yêu cầu xem chi tiết sản phẩm hoặc giỏ hàng).

2. Kiểm tra dữ liệu trong Redis:

- Hệ thống ưu tiên kiểm tra Redis Cache để xem dữ liệu cần thiết đã được lưu trữ hay chưa. Điều này giúp giảm thời gian phản hồi bằng cách tránh truy vấn cơ sở dữ liệu chậm hơn.

3. Xử lý theo hai kịch bản (Cache Hit hoặc Cache Miss):

• Cache Miss (Không có dữ liệu trong Redis):

- Redis trả về thông báo rằng dữ liệu không tồn tại trong cache.
- Hệ thống chuyển sang truy vấn MySQL Database để lấy dữ liệu cần thiết.
- MySQL trả về dữ liệu cho hệ thống.
- Hệ thống cập nhật dữ liệu này vào Redis Cache để sử dụng cho các yêu cầu tương lai, đảm bảo hiệu suất tốt hơn trong lần truy vấn tiếp theo.

• Cache Hit (Có dữ liệu trong Redis):

- Redis trả về dữ liệu trực tiếp từ cache cho hệ thống, loại bỏ nhu cầu truy vấn MySQL. Điều này giúp giảm độ trễ và tăng tốc độ phản hồi.

4. Phản hồi người dùng:

- Sau khi nhận được dữ liệu (từ Redis hoặc MySQL), hệ thống gửi kết quả phản hồi lại cho Người dùng, hoàn thành yêu cầu.

3.1.2. Phân tích triển khai đối với MySQL

Azure Database for MySQL là một cơ sở dữ liệu quan hệ (RDBMS) được thiết kế để lưu trữ và quản lý dữ liệu có cấu trúc với các mối quan hệ phức tạp, phù hợp với vai trò cơ sở dữ liệu chính. Các lý do chính bao gồm:

Tính toàn vẹn dữ liệu (ACID Compliance) MySQL đảm bảo tính Atomicity, Consistency, Isolation, và Durability (ACID) cho các giao dịch, điều quan trọng đối với hệ thống bán hàng trực tuyến. Ví dụ, khi xử lý đơn hàng, MySQL đảm bảo rằng việc trừ tồn kho và ghi nhận thanh toán được thực hiện đồng thời, tránh lỗi dữ liệu (như bán quá số lượng tồn kho).

Hỗ trợ dữ liệu quan hệ Hệ thống bán hàng trực tuyến thường yêu cầu quản lý các bảng liên quan như sản phẩm, đơn hàng, khách hàng, và giao dịch. MySQL cung cấp khả năng liên kết dữ liệu qua các khóa ngoại (foreign keys), hỗ trợ truy vấn phức tạp bằng SQL (như JOIN, GROUP BY), phù hợp để xử lý các yêu cầu như báo cáo doanh thu hoặc phân tích khách hàng.

Lưu trữ lâu dài và bền vững MySQL được thiết kế để lưu trữ dữ liệu lâu dài trên đĩa, với các tính năng như sao lưu tự động, khôi phục tại thời điểm cụ thể (point-in-time restore), và geo-redundant backup. Điều này đảm bảo dữ liệu không bị mất ngay cả trong trường hợp sự cố, không giống như Redis, vốn chủ yếu lưu trữ trong bộ nhớ và có thể mất dữ liệu nếu không cấu hình persistence.

Tương thích và phổ biến MySQL là một trong những cơ sở dữ liệu phổ biến nhất, được hỗ trợ bởi nhiều công cụ và framework (như Django, Laravel, Spring). Azure Database for MySQL tương thích hoàn toàn với MySQL Community Edition, giúp dễ dàng tích hợp và di chuyển từ các hệ thống hiện có.

Bảo mật và tuân thủ Azure Database for MySQL cung cấp mã hóa dữ liệu (AES 256-bit), hỗ trợ TLS 1.2, và tích hợp với Azure Key Vault, đáp ứng các tiêu chuẩn như GDPR và PCI DSS. Điều này rất quan trọng để bảo vệ thông tin nhạy cảm như dữ liệu khách hàng và giao dịch trong thương mại điện tử.

Từ những lý do trên: Chúng tôi quyết định sử dụng MySQL làm cơ sở dữ liệu chính cho hệ thống bán hàng trực tuyến.

3.1.3. Phân tích triển khai đối với Redis

Azure Cache for Redis là một cơ sở dữ liệu trong bộ nhớ (in-memory), được thiết kế để cung cấp truy cập dữ liệu cực nhanh với độ trễ dưới mili giây. Khi kết hợp với MySQL, Redis giải quyết các hạn chế trên bằng cách đóng vai trò là lớp caching.

Nhầm tận dụng những điểm mạnh của Redis trong việc caching:

- **Tăng tốc độ truy cập dữ liệu:** Redis lưu trữ dữ liệu trong RAM, giúp xử lý hàng triệu yêu cầu mỗi giây với độ trễ thấp.
- **Giảm tải cho MySQL:** Bằng cách lưu trữ dữ liệu thường xuyên truy cập (như danh sách sản phẩm bán chạy, thông tin khuyến mãi) trong Redis, hệ thống giảm số lượng truy vấn trực tiếp đến MySQL. Điều này giúp MySQL tập trung vào các tác vụ quan trọng như xử lý giao dịch hoặc cập nhật dữ liệu.
- **Hỗ trợ các cấu trúc dữ liệu linh hoạt:** Redis hỗ trợ nhiều cấu trúc dữ liệu (strings, hashes, lists, sets, sorted sets), phù hợp để lưu trữ dữ liệu tạm thời như giỏ hàng (hashes), danh sách sản phẩm đề xuất (sorted sets), hoặc hàng đợi đơn hàng (lists). Ví dụ, giỏ hàng của người dùng có thể được lưu dưới dạng user:123:cart với TTL (time-to-live) để tự động xóa sau khi không hoạt động.

Chúng tôi quyết định triển khai Azure Cache for Redis để lưu trữ session đăng nhập, giỏ hàng, và các hàng hóa thịnh hành.

Lý do: Do hạn chế về mặt tài nguyên (Chúng tôi chỉ sử dụng tier thấp nhất mà server Cache for redis cung cấp) thêm vào đó là giới hạn về mặt thời gian 2 tháng để thực hiện đồ án.

Thêm vào đó sau khi phân tích chúng tôi nhận thấy:

- **Đối với duy trì phiên đăng nhập:**

- *Về mặt người dùng:* Là một trong những yếu tố góp phần cải thiện trải nghiệm người dùng (ví dụ: trường hợp người dùng chỉ ấn nhầm hoặc mất mạng do vậy phải đăng nhập trở lại bằng tài khoản mật khẩu dù bảo mật có phần cải thiện hơn nhưng trải nghiệm người dùng sẽ đi xuống).
- *Về mặt hệ thống:* Đối với các hệ thống nhỏ và vừa nếu lượng đăng nhập tại một thời điểm đột biến lên rất cao trong trường hợp hệ thống của chúng tôi là do Flash sale chẳng hạn sẽ làm quá tải cơ sở dữ liệu chính MySQL, rủi ro làm chậm hoặc xấu hơn là sập hệ thống.

- **Đối với việc thực hiện caching cho giỏ hàng:**

- *Về mặt người dùng:* Giỏ hàng là nơi người dùng thường xuyên tương tác, đặc biệt là khi người dùng cần mua nhiều sản phẩm. Để mua nhiều sản phẩm người dùng cần phải thêm vào giỏ hàng rồi sau đó mới thực hiện các bước điền thông tin để mua sản phẩm. Nếu mỗi lần phải thực hiện truy vấn MySQL để lấy thông tin sản phẩm sẽ làm tốn khá nhiều thời gian hơn là thông qua Redis (cache), từ đó ảnh hưởng đến trải nghiệm mua hàng.
- *Về mặt hệ thống:* Như đã phân tích ở trên, nếu mỗi lần phiên đăng nhập, người dùng đều thao tác trực tiếp lên bảng dữ liệu Cart ở cơ sở dữ liệu chính, số lần gọi truy vấn MySQL (thêm xóa sửa và lấy thông tin giỏ hàng) sẽ giảm hiệu suất hệ thống. Thay vào đó chúng tôi sẽ load giỏ hàng lên

cache mỗi khi người dùng đăng nhập thành công. Sau đó nếu người dùng đăng xuất hoặc hết phiên đăng nhập chúng tôi sẽ thực hiện đồng bộ giỏ hàng trên MySQL. Và việc thực hiện thao tác mua hàng với các sản phẩm của giỏ hàng cũng được thực hiện dựa trên dữ liệu trên Cache.

- **Đối với việc lưu trữ sản phẩm thịnh hành:**

- *Về mặt người dùng:* Đa phần người dùng sẽ có xu hướng mua hoặc xem các sản phẩm thịnh hành. Do vậy việc đảm bảo tính sẵn sàng của dữ liệu bằng cách sử dụng cache các sản phẩm thịnh hành là cần thiết để người dùng thực hiện xem sản phẩm hoặc thực hiện mua hàng nhanh chóng. Đối với nhân viên việc thực hiện các báo cáo như sản phẩm thịnh hành cũng trở nên dễ dàng hơn thay vì phải thực hiện truy vấn phức tạp.
- *Về mặt hệ thống:* Cũng như 2 chức năng trên đa phần các dữ liệu chúng tôi đưa lên cache cũng nhằm giảm tải cho cơ sở dữ liệu chính. Tránh các hình thức tấn công như DDoS.

Triển khai Lưu trữ session đăng nhập

Mô tả sơ đồ PlantUML

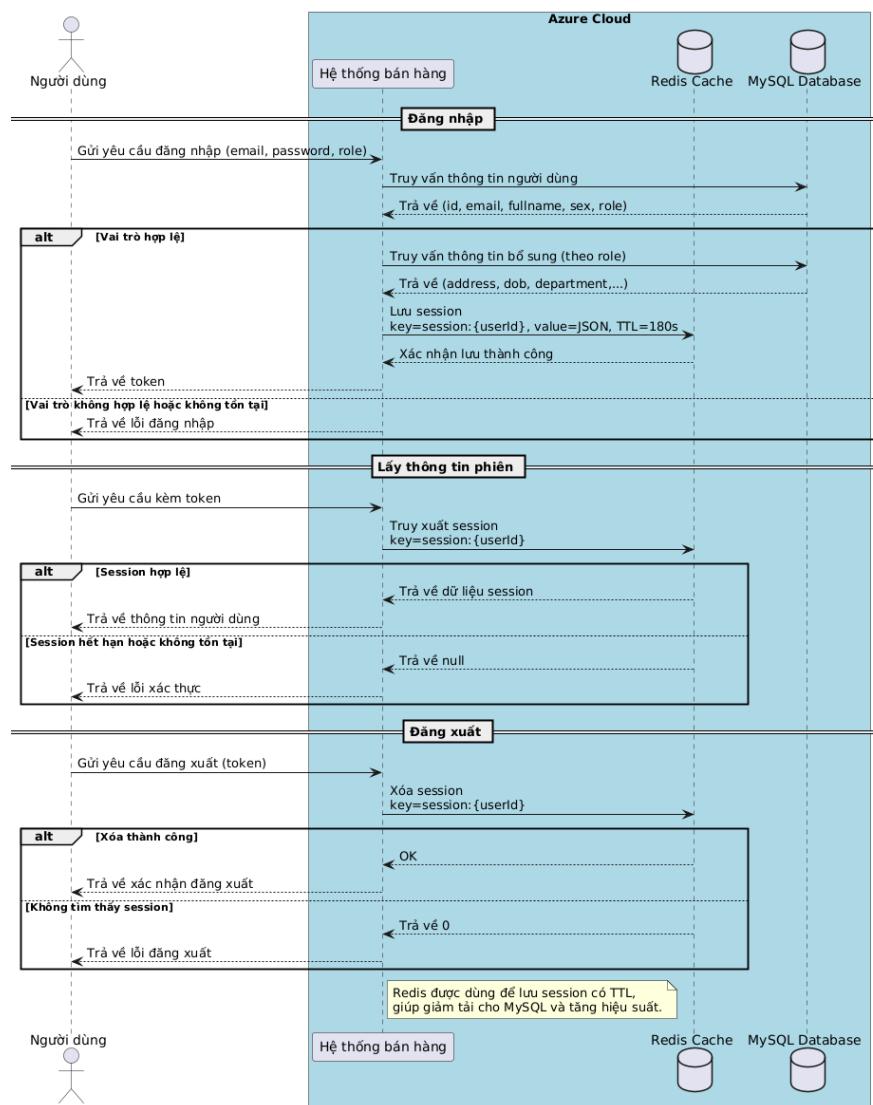
Các thành phần

- **Người dùng (User):** Khách hàng hoặc nhân viên gửi yêu cầu đăng nhập, lấy thông tin session, hoặc đăng xuất.
- **Hệ thống bán hàng (System):** Thành phần trung tâm, xử lý logic đăng nhập, tương tác với Redis và MySQL.
- **Redis Cache (Redis):** Lưu trữ phiên đăng nhập dưới dạng khóa session:<token> với TTL 180 giây.
- **MySQL Database (MySQL):** Lưu trữ thông tin người dùng và thông tin bổ sung theo vai trò (customers, employees, admin).
- **Azure Cloud:** Hộp bao quanh, thể hiện môi trường triển khai trên Azure.

Luồng xử lý

1. Đăng nhập:

- Người dùng gửi yêu cầu đăng nhập với email, password, và role tới hệ thống.
- Hệ thống truy vấn MySQL để xác thực thông tin người dùng (bảng user).
- Nếu tìm thấy người dùng và vai trò khớp:
 - Hệ thống truy vấn bảng tương ứng (customers, employees, hoặc admin) để lấy thông tin bổ sung.



Hình 3.2: Sơ đồ triển khai lưu trữ session đăng nhập

- Hệ thống tạo session trong Redis với khóa session:<token> và dữ liệu JSON, đặt TTL 180 giây.
- Trả về token cho người dùng.
- Nếu vai trò không khớp hoặc không tìm thấy người dùng, trả về lỗi.

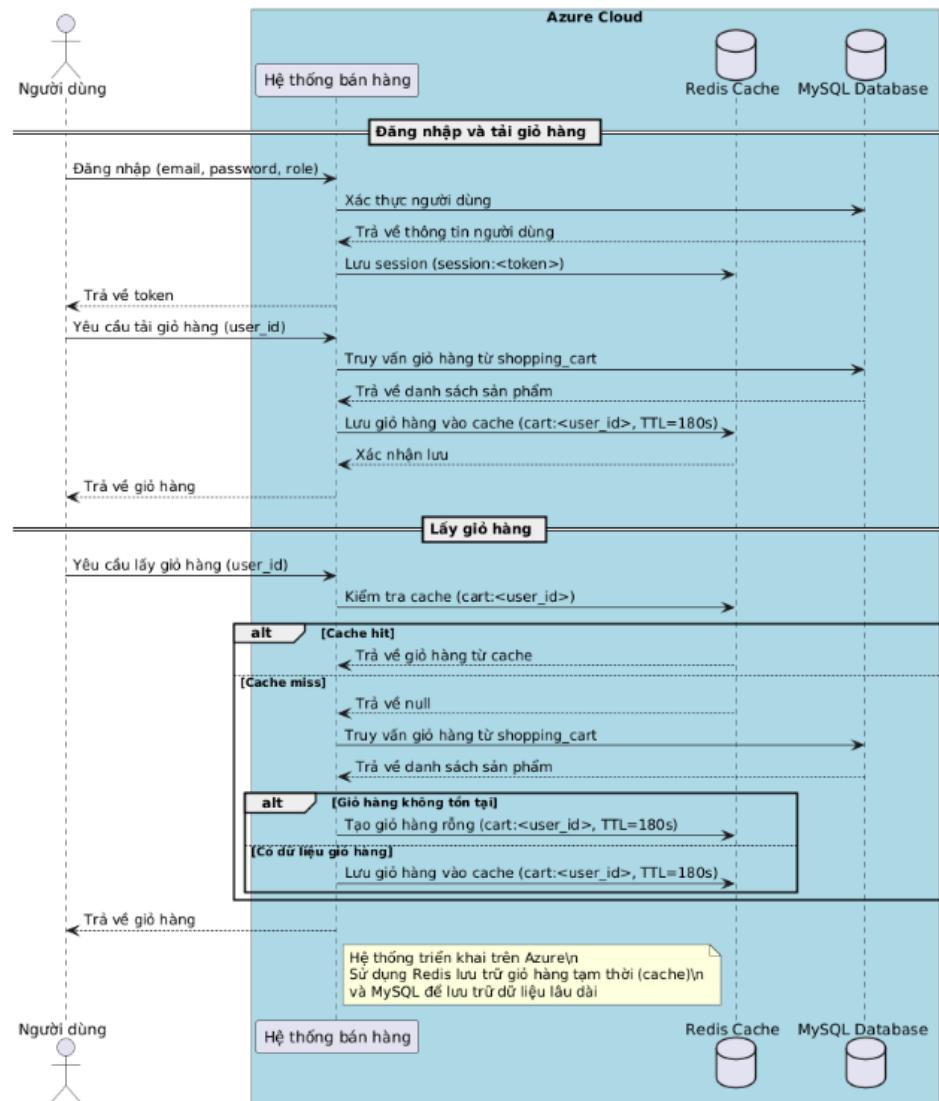
2. Lấy thông tin session:

- Người dùng gửi token tới hệ thống.
- Hệ thống kiểm tra Redis với khóa session:<token>.
- Nếu session tồn tại, trả về thông tin người dùng.
- Nếu không tìm thấy session, trả về lỗi.

3. Đăng xuất:

- Người dùng gửi token để đăng xuất.
- Hệ thống xóa khóa session:<token> khỏi Redis.
- Nếu xóa thành công, trả về thông báo thành công.
- Nếu session không tồn tại, trả về lỗi.

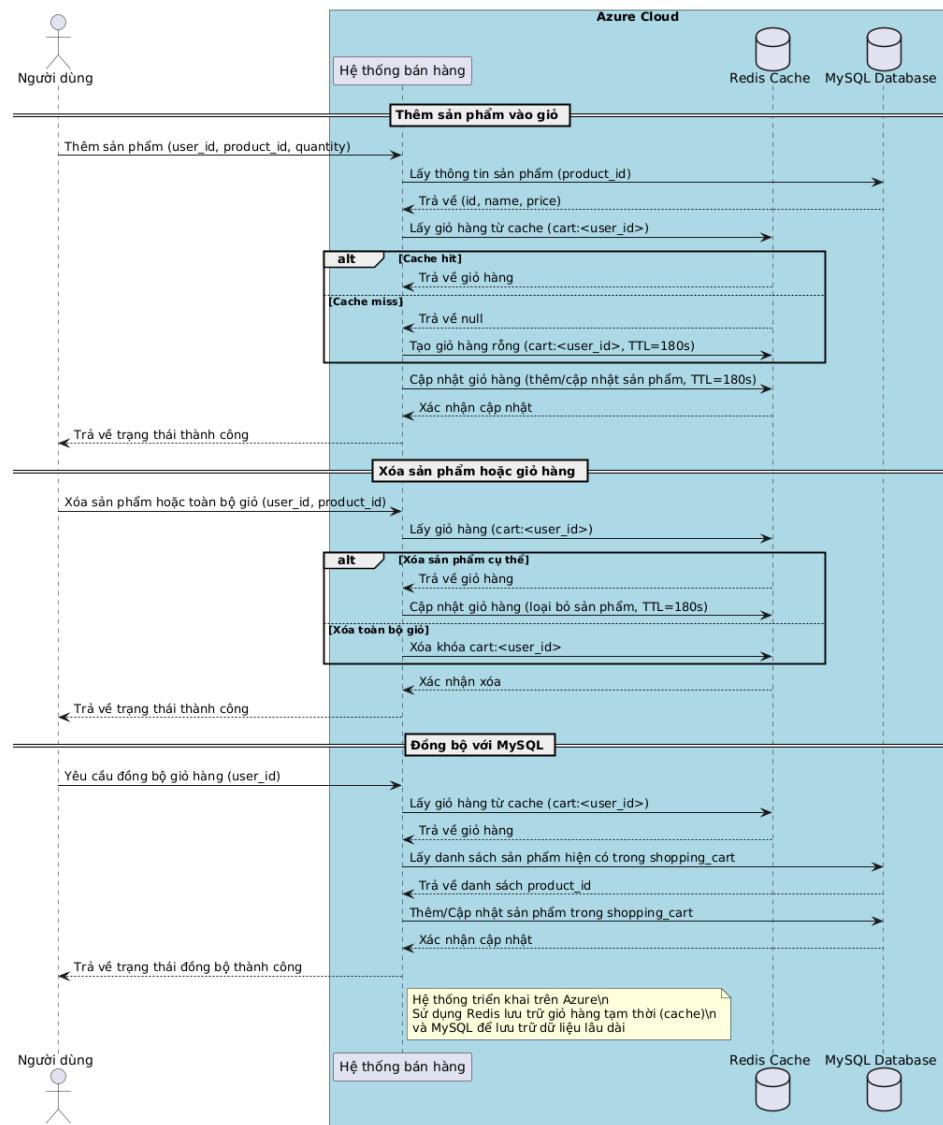
Triển khai lưu trữ giỏ hàng (Hình 3.3 và Hình 3.4)



Hình 3.3: Sơ đồ triển khai lưu trữ giỏ hàng - 1

Các thành phần

- **Người dùng (User):** Khách hàng gửi các yêu cầu như đăng nhập, lấy giỏ hàng, thêm sản phẩm, xóa sản phẩm, hoặc đồng bộ giỏ hàng.



Hình 3.4: Sơ đồ triển khai lưu trữ giỏ hàng - 2

- **Hệ thống bán hàng (System):** Thành phần trung tâm, xử lý logic nghiệp vụ, tương tác với Redis và MySQL.
- **Redis Cache (Redis):** Lưu trữ giỏ hàng tạm thời dưới dạng khóa cart:<user_id> với TTL 180 giây, sử dụng JSON để lưu trữ danh sách sản phẩm.
- **MySQL Database (MySQL):** Lưu trữ dữ liệu lâu dài trong bảng shopping_cart (giỏ hàng) và products (sản phẩm).
- **Azure Cloud:** Hộp bao quanh, thể hiện môi trường triển khai trên Azure.

Luồng xử lý chi tiết

1. Đăng nhập và tải giỏ hàng:

-
- Bước 1: Người dùng gửi yêu cầu đăng nhập với email, password, và role tới hệ thống.
 - Bước 2: Hệ thống truy vấn MySQL để xác thực thông tin người dùng từ bảng user.
 - Bước 3: Nếu đăng nhập thành công, hệ thống lưu session vào Redis với khóa session:<token>.
 - Bước 4: Hệ thống truy vấn bảng shopping_cart trong MySQL để lấy giỏ hàng của người dùng (dựa trên user_id).
 - Bước 5: Nếu giỏ hàng tồn tại, hệ thống lưu dữ liệu vào Redis với khóa cart:<user_id> và TTL 180 giây.
 - Bước 6: Trả về giỏ hàng cho người dùng.

2. Lấy giỏ hàng (get_cart):

- Bước 1: Người dùng yêu cầu lấy giỏ hàng với user_id.
- Bước 2: Hệ thống kiểm tra Redis với khóa cart:<user_id>.
 - *Cache Hit:*
 - Redis trả về dữ liệu giỏ hàng (JSON), hệ thống trả về cho người dùng.
 - *Cache Miss:*
 - Hệ thống truy vấn MySQL để lấy danh sách sản phẩm từ bảng shopping_cart và products.
 - Nếu không có dữ liệu, tạo giỏ hàng rỗng trong Redis (create_cart).
 - Nếu có dữ liệu, lưu vào Redis với TTL 180 giây.
 - Trả về giỏ hàng cho người dùng.

3. Thêm sản phẩm vào giỏ (add_cart):

- Bước 1: Người dùng gửi yêu cầu thêm sản phẩm với user_id, product_id, và quantity.
- Bước 2: Hệ thống truy vấn MySQL để lấy thông tin sản phẩm (ID, tên, giá) từ bảng products.
- Bước 3: Hệ thống kiểm tra Redis để lấy giỏ hàng (cart:<user_id>).
 - Nếu giỏ hàng tồn tại, cập nhật sản phẩm (tăng số lượng nếu đã có, hoặc thêm mới).
 - Nếu không, tạo giỏ hàng rỗng (create_cart) và thêm sản phẩm.
- Bước 4: Cập nhật giỏ hàng vào Redis với TTL 180 giây.
- Bước 5: Trả về trạng thái thành công cho người dùng.

4. Xóa sản phẩm hoặc giỏ hàng (delete_cart):

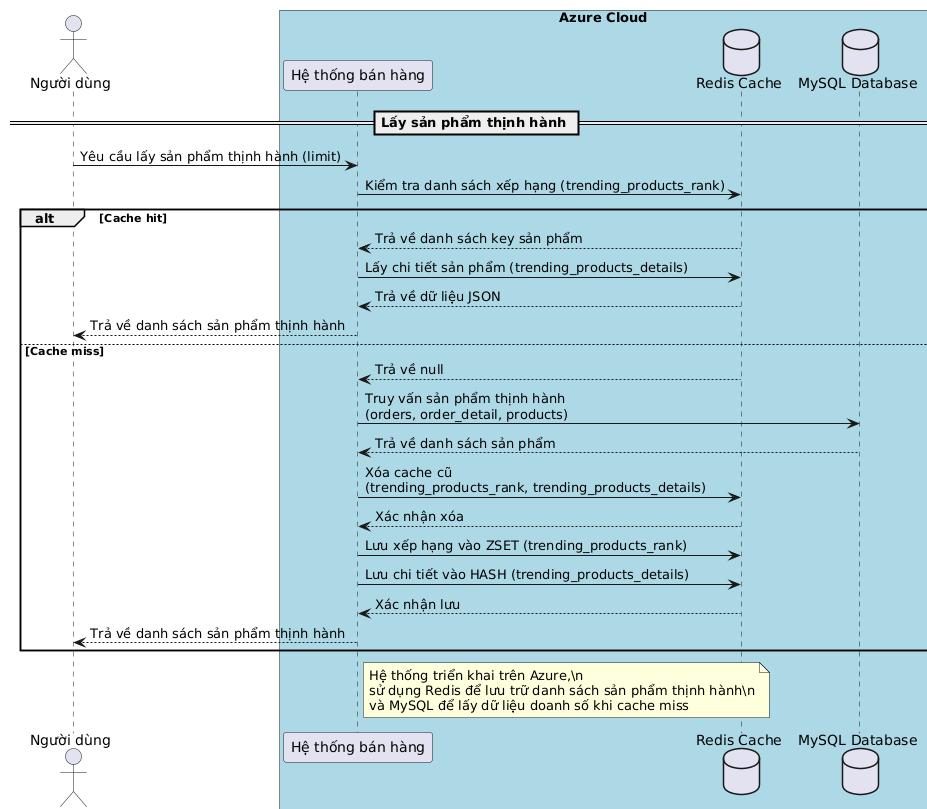
- Bước 1: Người dùng yêu cầu xóa sản phẩm (product_id) hoặc toàn bộ giỏ hàng với user_id.
- *Xóa sản phẩm cụ thể:*
 - Hệ thống lấy giỏ hàng từ Redis.

- Loại bỏ sản phẩm có product_id tương ứng.
 - Cập nhật giỏ hàng vào Redis với TTL 180 giây.
- Xóa toàn bộ giỏ:
 - Hệ thống xóa khóa cart:<user_id> khỏi Redis.
 - Bước 2: Trả về trạng thái thành công hoặc lỗi nếu sản phẩm không tồn tại.

5. Đồng bộ với MySQL (synchronize_mysql):

- Bước 1: Người dùng yêu cầu đồng bộ giỏ hàng (thường khi xác nhận đơn hàng hoặc đăng xuất).
- Bước 2: Hệ thống lấy giỏ hàng từ Redis (cart:<user_id>).
- Bước 3: Hệ thống truy vấn MySQL để lấy danh sách product_id hiện có trong bảng shopping_cart.
- Bước 4: Đối với mỗi sản phẩm trong giỏ hàng Redis:
 - Nếu sản phẩm đã tồn tại, cập nhật số lượng trong MySQL.
 - Nếu sản phẩm chưa có, thêm mới vào bảng shopping_cart.

Triển khai lưu trữ các sản phẩm thịnh hành



Hình 3.5: Sơ đồ triển khai lưu trữ sản phẩm thịnh hành

Mô tả quy trình Sơ đồ PlantUML và mã Python mô tả quy trình quản lý danh sách sản phẩm thịnh hành trong hệ thống bán hàng trực tuyến, với sự kết hợp giữa Azure Cache for Redis và Azure Database for MySQL. Quy trình bao gồm lấy danh sách sản phẩm thịnh hành, cập nhật cache định kỳ.

Các thành phần

- **Người dùng (User):** Khách hàng hoặc hệ thống gửi yêu cầu lấy danh sách sản phẩm thịnh hành hoặc cập nhật doanh số sản phẩm.
- **Hệ thống bán hàng (System):** Thành phần trung tâm, xử lý logic nghiệp vụ, tương tác với Redis và MySQL.
- **Redis Cache (Redis):** Lưu trữ danh sách sản phẩm thịnh hành dưới dạng:
 - *Sorted Set (trending_products_rank):* Xếp hạng sản phẩm dựa trên doanh số (sales_count).
 - *Hash (trending_products_details):* Lưu chi tiết sản phẩm (tên, giá, mô tả, doanh số) dưới dạng JSON.
- **MySQL Database (MySQL):** Lưu trữ dữ liệu lâu dài trong các bảng orders, order_detail, và products để tính toán doanh số.
- **Azure Cloud:** Hộp bao quanh, thể hiện môi trường triển khai trên Azure.

Luồng xử lý chi tiết

1. Lấy sản phẩm thịnh hành (get_trending_products):

- Bước 1: Người dùng yêu cầu lấy danh sách sản phẩm thịnh hành với giới hạn limit (mặc định 10).
- Bước 2: Hệ thống kiểm tra Redis để lấy danh sách key sản phẩm từ Sorted Set (trending_products_rank) bằng lệnh ZREVRANGE.
- *Cache Hit:*
 - Redis trả về danh sách key sản phẩm (ví dụ: product:1, product:2).
 - Hệ thống lấy chi tiết sản phẩm từ Hash (trending_products_details) bằng HGET.
 - Hệ thống thêm product_id và Top (thứ hạng) vào dữ liệu, trả về danh sách sản phẩm cho người dùng.
- *Cache Miss:*
 - Redis trả về null (không có dữ liệu).
 - Hệ thống gọi update_trending_cache để cập nhật cache từ MySQL:
 - * Truy vấn MySQL để lấy danh sách sản phẩm thịnh hành từ bảng orders, order_detail, và products, dựa trên tổng quantity của các đơn hàng có trạng thái received.

-
- * Xóa cache cũ (trending_products_rank và trending_products_details).
 - * Lưu xếp hạng vào Sorted Set (trending_products_rank) với điểm số là sales_count.
 - * Lưu chi tiết sản phẩm vào Hash (trending_products_details) dưới dạng JSON.
 - * Đặt TTL 180 giây (hoặc 3600 giây trong trường hợp cache miss) cho cả hai cấu trúc.
 - Gọi lại get_trending_products để lấy danh sách từ cache mới.

3.2. Thiết kế cơ sở dữ liệu

Đầu tiên, chúng tôi tiến hành xác định các thực thể, thuộc tính của từng thực thể, và các mối quan hệ giữa chúng. Dựa trên kết quả phân tích, chúng tôi tổng hợp thành bảng 3.1. Sau đó chúng tôi xác định các bảng và thuộc tính của chúng trong cơ sở dữ liệu, đồng thời xác định kiểu dữ liệu của các thuộc tính và diễn giải vai trò của các thuộc tính, tất cả được chúng tôi kết luận và ghi lại kết quả trên bảng 3.2. Từ hai bảng này chúng tôi tiến hành vẽ sơ đồ thực thể kết hợp, chính là các hình 3.6, 3.7, 3.8, 3.9. Vì kích thước khổ giấy A4 chúng tôi phải tách sơ đồ bắt đầu từ 3.7 đến 3.9 thành nhiều phần.

3.3. Tạo cơ sở dữ liệu và nhập dữ liệu cho các bảng

Để triển khai việc tạo cơ sở dữ liệu và các bảng hiệu quả, dễ kiểm tra và sửa lỗi, chúng tôi chia làm 3 giai đoạn:

- Giai đoạn 1: Tạo bảng và khóa ngoại.
- Giai đoạn 2: Tạo và nhập dữ liệu.
- Giai đoạn 3: Tạo index.

3.3.1. Tạo bảng dữ liệu và khóa ngoại

Giới thiệu về DrawDB: DrawDB là một công cụ vẽ sơ đồ cơ sở dữ liệu (ERD) mã nguồn mở miễn phí và trực tuyến, cực kỳ đơn giản và tiện dụng – đặc biệt phù hợp với các lập trình viên, sinh viên hoặc kỹ sư dữ liệu muốn chuyển từ mô hình SQL sang sơ đồ trực quan nhanh chóng. Ngoài ra, DrawDB còn có các tính năng như import hoặc export các câu lệnh SQL thành sơ đồ RD tương ứng.

Tận dụng những tính năng trên, nhóm chúng tôi đã chuyển từ ERD sang ED bằng cách sử dụng DrawDB.Thêm vào đó, chúng tôi thêm các ràng buộc toàn vẹn và index. Sau đó xuất thành tập tin SQL để tạo bảng và khóa ngoại.

3.3.2. Tạo và nhập dữ liệu

Do phạm vi môn học không yêu cầu sử dụng dữ liệu thật và thời gian thực hiện không cho phép chúng tôi thu thập và xử lý dữ liệu, nhóm quyết định sử dụng dữ liệu mô phỏng để đưa vào hệ thống.

Giới thiệu về thư viện Faker: Đây là một thư viện Python cực kỳ hữu ích để tạo dữ liệu giả lập (mock data) một cách nhanh chóng và linh hoạt. Faker thường được dùng trong kiểm thử phần mềm, tạo dữ liệu mẫu cho cơ sở dữ liệu, và mô phỏng các hệ thống trước khi có dữ liệu thực.

Tính năng nổi bật của Faker:

- Tạo họ tên, email, số điện thoại, địa chỉ, ngày tháng, văn bản, số liệu thống kê, v.v.
- Hỗ trợ đa ngôn ngữ (bao gồm tiếng Việt).
- Dễ dàng tùy chỉnh định dạng dữ liệu theo nhu cầu.

Bằng cách sử dụng thư viện này, chúng tôi đã tạo ra bộ dữ liệu mô phỏng phù hợp cho việc triển khai và đánh giá hệ thống. [Bảng 3.3](#) là danh sách các tập tin dữ liệu đã được tạo.

Việc nhập dữ liệu được thực hiện thông qua thư viện kết nối với MySQL trên Python thay vì import trực tiếp tập tin dữ liệu vào cơ sở dữ liệu. Điều này giúp kiểm soát và xử lý lỗi dễ dàng hơn, đặc biệt khi các trigger kiểm tra tính đúng đắn của dữ liệu được kích hoạt.

3.3.3. Tạo index

Index làm tăng thời gian nhập dữ liệu, đặc biệt với dữ liệu có kích thước lớn. Ví dụ, nếu bảng `products` có 1 triệu bản ghi và có index trên cột `product_name`, khi thêm một bản ghi mới, hệ thống phải tìm đúng vị trí trong cây chỉ mục (B-Tree) để chèn giá trị mới, dẫn đến tốn thêm thời gian. Vì vậy, việc tạo index được thực hiện cuối cùng để tối ưu thời gian.

Các tiêu chí đánh giá để lập index:

- **Tần suất sử dụng trong truy vấn:** Các cột được sử dụng thường xuyên trong các điều kiện lọc (WHERE), sắp xếp (ORDER BY), kết hợp (JOIN), hoặc nhóm (GROUP BY) nên được lập chỉ mục.
- **Độ chọn lọc của cột (Cardinality):** Cột có độ chọn lọc cao (nhiều giá trị duy nhất) sẽ mang lại lợi ích lớn khi lập chỉ mục.
- **Vai trò của cột trong khóa ngoại (Foreign Key):** Các cột là khóa ngoại thường được sử dụng trong các phép JOIN hoặc kiểm tra ràng buộc toàn vẹn, nên được lập chỉ mục.
- **Loại truy vấn đặc biệt:** Một số loại truy vấn đặc biệt yêu cầu chỉ mục đặc biệt, như tìm kiếm toàn văn (FULLTEXT) hoặc truy vấn không gian (Spatial).

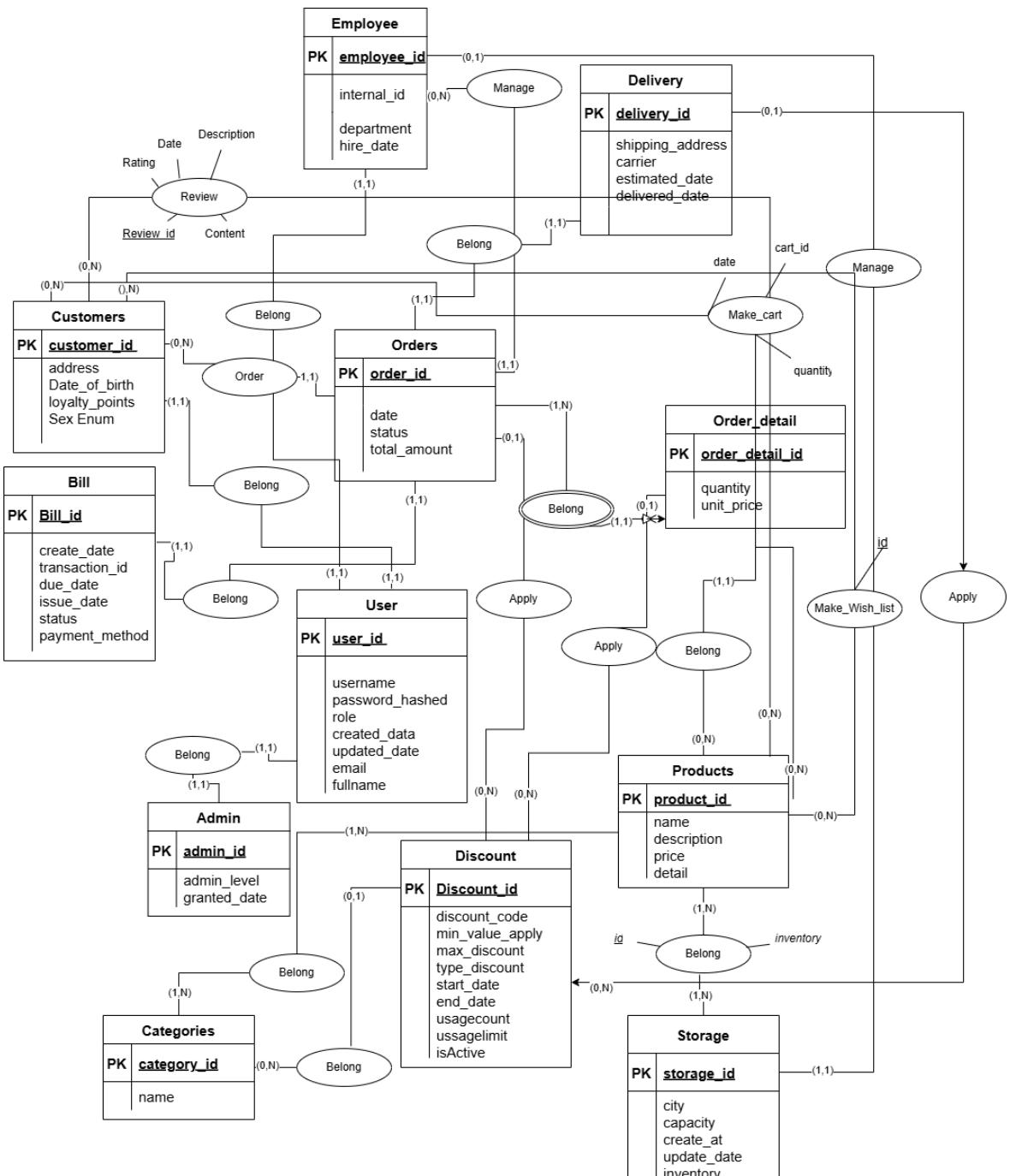
Sau khi xác định được các indexes (chỉ mục) cần thiết, chúng tôi xác định các bảng, cột, loại indexes và mục đích triển khai qua [bảng 3.4](#) cuối cùng là triển khai thực tế trên cơ sở dữ liệu.

Bảng 3.1: Các mối kết hợp giữa các thực thể trong hệ thống

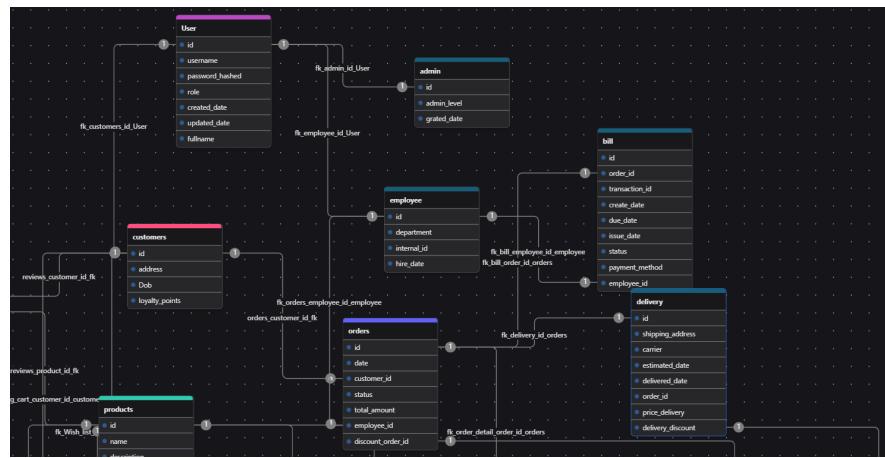
Mối kết hợp	Thực thể A	Thực thể B	Kiểu quan hệ	Mô tả
Đặt hàng	CUSTOMER	ORDERS	(0:N) - (1,1)	Một khách hàng có thể có nhiều đơn hàng hoặc không có đơn hàng nào, một đơn hàng chỉ thuộc một khách hàng
Thuộc	ORDERS	ORDERDETAIL	(1:N) - (1,1)	Một đơn hàng có nhiều chi tiết đơn hàng, một chi tiết đơn hàng chỉ thuộc một đơn hàng
Thuộc	PRODUCT	ORDERDETAIL	(0:N) - (1:1)	Một sản phẩm xuất hiện trong nhiều chi tiết đơn hàng hoặc không xuất hiện ở chi tiết đơn hàng nào, một chi tiết chỉ có 1 sản phẩm
Thuộc	ORDERS	BILLS	(1:1) - (1:1)	Mỗi đơn hàng có duy nhất 1 hóa đơn, 1 hóa đơn thuộc duy nhất 1 đơn hàng
Thuộc	ORDERS	DELIVERY	(1:1) - (1:1)	Mỗi đơn hàng chỉ có 1 giao hàng và 1 giao hàng chỉ thuộc 1 đơn hàng
Thuộc	CATEGORIES	DISCOUNT	(0:N) - (0:1)	1 loại sản phẩm có thể không có hoặc có nhiều mã giảm giá, 1 mã giảm giá có thể thuộc 1 hoặc không thuộc loại sản phẩm nào
Áp dụng	ORDERS	DISCOUNT	(0:1) - (0:N)	Mỗi đơn hàng có thể áp dụng 1 giảm giá hoặc không áp dụng giảm giá nào, 1 giảm giá có thể áp dụng cho nhiều đơn hàng
Áp dụng	ORDERDETAIL	DISCOUNT	(0:1) - (0:N)	Mỗi chi tiết đơn hàng có thể áp dụng nhiều nhất 1 mã giảm giá hoặc không áp dụng, 1 mã giảm giá có thể áp dụng cho nhiều chi tiết hoặc không có áp dụng cho chi tiết nào
Áp dụng	DELIVERY	DISCOUNT	(0:1) - (0:N)	Mỗi giao hàng có thể áp dụng nhiều nhất 1 mã giảm giá hoặc không áp dụng, 1 mã giảm giá có thể áp dụng cho nhiều giao hàng hoặc không có áp dụng cho giao hàng nào
Thuộc	PRODUCT	CATEGORIES	(1:N) - (1:N)	Một sản phẩm phải thuộc duy nhất 1 loại sản phẩm, 1 loại sản phẩm có thể bao gồm nhiều sản phẩm
Thuộc	STORAGE	PRODUCT	(1:N) - (1:N)	Một sản phẩm có thể có ở nhiều kho và ngược lại
Đánh giá	PRODUCT	CUSTOMER	(0:N) - (0:N)	Một sản phẩm có nhiều đánh giá, 1 khách hàng có thể đánh giá nhiều sản phẩm
Quản lý	EMPLOYEE	ORDERS	(0:N) - (1:1)	Một nhân viên có thể xử lý nhiều đơn hàng, 1 đơn hàng phải được duyệt bởi 1 nhân viên
Quản lý	EMPLOYEE	STORAGE	(0:1) - (1:1)	1 Nhân viên quản lý 1 kho hoặc không quản lý kho nào và 1 kho được quản lý duy nhất 1 nhân viên
Thêm sản phẩm yêu thích	CUSTOMER	PRODUCT	(0:N) - (0:N)	1 khách hàng có nhiều sản phẩm yêu thích hoặc không có sản phẩm yêu thích nào, 1 sản phẩm có nhiều khách hàng yêu thích hoặc không có trong danh sách yêu thích của ai
Thêm giỏ hàng	CUSTOMER	PRODUCT	(0:N) - (0:N)	1 khách hàng có nhiều sản phẩm trong giỏ hoặc không có sản phẩm nào, 1 sản phẩm có thể nằm trong giỏ của nhiều khách hàng hoặc không nằm trong giỏ hàng nào
Thuộc	USER	CUSTOMER	(1:1) - (1:1)	1 user vai trò customer ứng với 1 khách hàng
Thuộc	USER	ADMIN	(1:1) - (1:1)	1 user vai trò admin ứng với 1 quản trị viên
Thuộc	USER	EMPLOYEE	(1:1) - (1:1)	1 user vai trò employee ứng với 1 nhân viên

Bảng 3.2: Danh sách các bảng và thuộc tính trong cơ sở dữ liệu

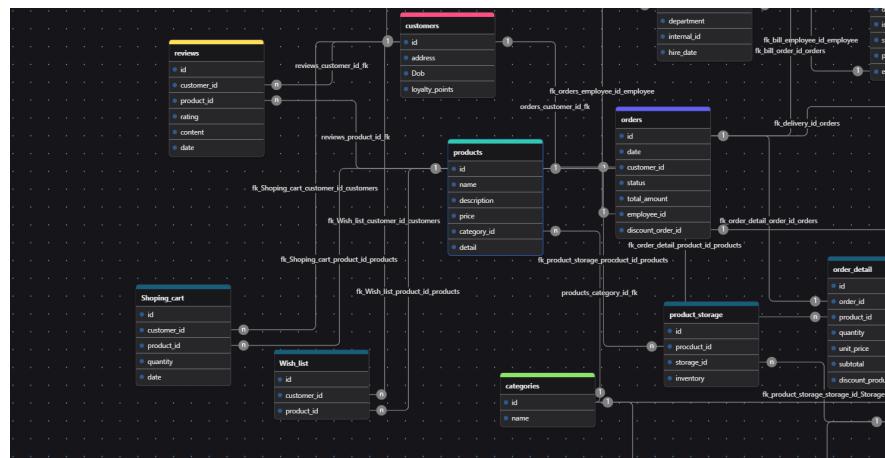
Quan hệ	Thuộc tính	Kiểu dữ liệu	Điễn giải
products	id	INT NOT NULL AUTO_INCREMENT UNIQUE	Khóa chính của bảng sản phẩm
products	name	VARCHAR(255)	Tên sản phẩm
products	description	TEXT(65535)	Mô tả chi tiết sản phẩm
products	price	DOUBLE	Giá sản phẩm
products	category_id	INT	Mã danh mục của sản phẩm
products	detail	JSON	Thông số cụ thể của sản phẩm
categories	id	INT NOT NULL AUTO_INCREMENT UNIQUE	Khóa chính của bảng danh mục
categories	name	VARCHAR(255)	Tên danh mục
orders	id	INT NOT NULL AUTO_INCREMENT UNIQUE	Khóa chính của bảng đơn hàng
orders	date	DATETIME	Ngày đặt hàng
orders	customer_id	INT	Mã khách hàng
orders	status	ENUM('delivered', 'received', 'processing')	Trạng thái đơn hàng
orders	total_amount	DECIMAL(15,2)	Tổng giá trị đơn hàng
orders	employee_id	INT	Nhân viên xử lý đơn hàng
orders	discount_order_id	INT	Mã giảm giá áp dụng cho đơn hàng
reviews	id	INT NOT NULL AUTO_INCREMENT UNIQUE	Khóa chính của bảng đánh giá
reviews	customer_id	INT	Mã khách hàng đánh giá
reviews	product_id	INT	Mã sản phẩm được đánh giá
reviews	rating	SMALLINT	Xếp hạng đánh giá
reviews	content	TEXT(65535)	Nội dung đánh giá
reviews	date	DATETIME	Ngày đánh giá
customers	id	INT NOT NULL UNIQUE	Khóa chính của bảng khách hàng
customers	address	VARCHAR(255)	Địa chỉ khách hàng
customers	Dob	DATE	Ngày sinh khách hàng
customers	loyalty_points	BIGINT	Điểm thành viên
customers	Sex	ENUM('Female', 'Male', 'Other')	Giới tính
order_detail	id	INT NOT NULL AUTO_INCREMENT UNIQUE	Khóa chính của bảng chi tiết đơn hàng
order_detail	order_id	INT	Mã đơn hàng
order_detail	product_id	INT	Mã sản phẩm
order_detail	quantity	INT	Số lượng sản phẩm
order_detail	unit_price	DECIMAL(15,2)	Giá sản phẩm tại thời điểm mua
order_detail	discount_product_id	INT	Mã giảm giá áp dụng cho sản phẩm
User	id	INT NOT NULL AUTO_INCREMENT UNIQUE	Khóa chính của bảng người dùng
User	username	VARCHAR(30)	Tên đăng nhập
User	password_hashed	VARCHAR(60)	Mật khẩu đã mã hóa
User	role	ENUM('customer', 'employee', 'admin')	Vai trò người dùng
User	created_date	DATETIME	Ngày tạo tài khoản
User	updated_date	DATETIME	Ngày cập nhật tài khoản
User	email	VARCHAR(30)	Email user
User	fullname	VARCHAR(30)	Họ và tên người dùng



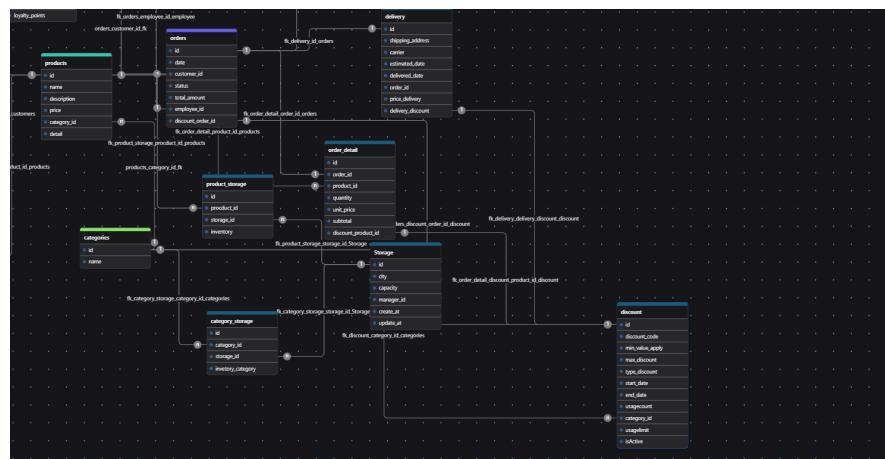
Hình 3.6: Sơ đồ erd



Hình 3.7: Lược đồ quan hệ - 1



Hình 3.8: Lược đồ quan hệ - 2



Hình 3.9: Lược đồ quan hệ - 3

Bảng 3.3: Danh sách các tập tin dữ liệu mô phỏng

Tên File	Kích thước
users.csv	10.000 user
customers.csv	9.000 customer
employees.csv	900 employee
admins.csv	100 admin
products.csv	20 product
categories.csv	8 category
orders.csv	9.000 order
order_details.csv	50.000 order_detail
review_generated.csv	6.000 reviews
carts.csv	50.000 cart
wishlists.csv	20.000 wish_list
bills.csv	9.000 bill
deliveries.csv	9.000 delivery
discounts.csv	1.000 discount

Bảng 3.4: Danh sách các chỉ mục (index) được triển khai trong hệ thống

Tên Index	Bảng	Cột	Loại Index	Mục đích
products_index_category_id	products	category_id	B-Tree	Tăng tốc độ truy vấn lọc hoặc tìm kiếm sản phẩm theo danh mục (category_id), thường dùng trong danh sách sản phẩm theo danh mục.
orders_index_customer_id	orders	customer_id	B-Tree	Tăng tốc độ truy vấn lịch sử đơn hàng của khách hàng hoặc báo cáo đơn hàng theo khách hàng.
orders_index_date	orders	date	B-Tree	Tăng tốc độ truy vấn đơn hàng theo ngày hoặc phạm vi thời gian, dùng trong báo cáo doanh thu hoặc phân tích xu hướng.
orders_index_employee	orders	employee_id	B-Tree	Tăng tốc độ truy vấn đơn hàng được xử lý bởi nhân viên cụ thể, dùng trong quản lý hiệu suất nhân viên.
reviews_index_product_id	reviews	product_id	B-Tree	Tăng tốc độ truy vấn đánh giá của một sản phẩm, thường dùng để hiển thị đánh giá trên trang chi tiết sản phẩm.
order_detail_index_order_id	order_detail	order_id	B-Tree	Tăng tốc độ truy vấn chi tiết đơn hàng dựa trên order_id, dùng khi hiển thị thông tin chi tiết của một đơn hàng.
order_detail_index_product_id	order_detail	product_id	B-Tree	Tăng tốc độ truy vấn các đơn hàng chứa một sản phẩm cụ thể, dùng trong phân tích sản phẩm hoặc kiểm tra lịch sử bán hàng.
User_index_0	User	username	B-Tree	Tăng tốc độ truy vấn hoặc xác thực người dùng dựa trên username, thường dùng trong đăng nhập hoặc tìm kiếm người dùng.
Wish_list_index_composite_customer_product	Wish_list	customer_id, product_id	B-Tree (Composite)	Tăng tốc độ truy vấn danh sách yêu thích của khách hàng hoặc kiểm tra sản phẩm cụ thể trong danh sách yêu thích.
Shopping_cart_index_composite_cus_prod	Shopping_cart	customer_id, product_id	B-Tree (Composite)	Tăng tốc độ truy vấn giờ hàng của khách hàng hoặc kiểm tra sản phẩm cụ thể trong giờ hàng, dùng trong quản lý giờ hàng.
product_storage_index_product_id	product_storage	product_id	B-Tree	Tăng tốc độ truy vấn thông tin kho của một sản phẩm, dùng trong quản lý tồn kho hoặc kiểm tra trạng thái kho.
product_storage_index_storage_id	product_storage	storage_id	B-Tree	Tăng tốc độ truy vấn tất cả sản phẩm trong một kho cụ thể, dùng trong quản lý kho hoặc báo cáo tồn kho.
bill_index_order_id	bill	order_id	B-Tree	Tăng tốc độ truy vấn hóa đơn liên quan đến một đơn hàng, dùng khi hiển thị thông tin hóa đơn hoặc kiểm tra thanh toán.
delivery_index_0	delivery	order_id	B-Tree	Tăng tốc độ truy vấn thông tin giao hàng của một đơn hàng, dùng trong theo dõi trạng thái giao hàng.
FULLTEXT (name, description)	products	name, description	FULLTEXT	Hỗ trợ tìm kiếm toàn văn (full-text search) trên tên và mô tả sản phẩm, dùng trong chức năng tìm kiếm sản phẩm theo từ khóa.

Chương 4 LẬP TRÌNH CƠ SỞ DỮ LIỆU

4.1. Lập trình thủ tục

4.1.1. Thủ tục sp_login

Mô tả: Xử lý quá trình đăng nhập của người dùng bằng cách xác thực tên người dùng hoặc email và mật khẩu, sau đó trả về thông tin người dùng và thông tin bổ sung dựa trên vai trò.

Tham số:

- **Đầu vào:**

- p_username_or_email (VARCHAR): Tên người dùng hoặc email để đăng nhập.
- p_password (VARCHAR): Mật khẩu đã mã hóa.
- p_role (VARCHAR): Vai trò người dùng (customer, employee, admin).

- **Đầu ra:**

- p_user_id (INT): ID của người dùng.
- p_email (VARCHAR): Email của người dùng.
- p_fullname (VARCHAR): Họ tên của người dùng.
- p_sex (VARCHAR): Giới tính của người dùng.
- p_extra_info (JSON): Thông tin bổ sung dựa trên vai trò.

Hoạt động:

1. Tìm kiếm người dùng trong bảng user dựa trên p_username_or_email và p_password.
2. Nếu không tìm thấy, trả về lỗi "Invalid credentials".
3. Kiểm tra vai trò người dùng và cảnh báo nếu vai trò yêu cầu không khớp.
4. Lấy thông tin bổ sung từ bảng tương ứng (customers, employee, hoặc admin) dựa trên vai trò và lưu vào p_extra_info.
5. Trả về thông tin người dùng và thông tin bổ sung.

Thực hiện kiểm thử thủ tục:

- Đăng nhập thành công:

```
|| CALL sp_login('mullinsmichelle@example.org',
  'd8faeff5e40bcce793759a7514a663ba4fb735866b6a9c370f0d589d47ebddf2',
  'customer', @user_id, @email, @fullname, @sex, @extra_info);
```

Kết quả:

```
|| Test Case 1: Successful login as customer| 1|mullinsmichelle@example.org|
  Benjamin Gomez|Female|{"dob": "1955-01-08", "address": "50067 Deborah
  Parkways Apt. 404, Wendymouth, NM 84562"}|
```

- Đăng nhập thất bại:

```
|| CALL sp_login('Email_sai',
  'd8faeff5e40bcce793759a7514a663ba4fb735866b6a9c370f0d589d47ebddf2',
  'customer', @user_id, @email, @fullname, @sex, @extra_info);
```

Kết quả:

```
|| SQL Error [1644] [45000]: Invalid username/email or password
```

4.1.2. Thủ tục add_to_cart

Mô tả: Thêm sản phẩm vào giỏ hàng của khách hàng hoặc cập nhật số lượng nếu sản phẩm đã có trong giỏ.

Tham số:

- Đầu vào:

- p_customer_id (INT): ID của khách hàng.
- p_product_id (INT): ID của sản phẩm.
- p_quantity (INT): Số lượng sản phẩm muốn thêm.

Hoạt động:

1. Kiểm tra xem sản phẩm đã có trong giỏ hàng (Shopping_cart) của khách hàng chưa.
2. Nếu sản phẩm đã tồn tại, cập nhật số lượng bằng cách cộng thêm p_quantity.
3. Nếu sản phẩm chưa có, thêm một bản ghi mới vào bảng Shopping_cart với thông tin customer_id, product_id, và quantity.

	id	customer_id	product_id	quantity	date
1	50,005	1	1	40	2025-04-13 14:04:46
2	50,001	1	8	9	2025-04-13 11:06:05
3	50,004	1	14	91	2025-04-13 11:06:05

Hình 4.1: Dữ liệu ban đầu của giỏ hàng

Thực hiện kiểm thử thủ tục: Dữ liệu ban đầu của giỏ hàng được thể hiện qua hình 4.1. Sau khi cập nhật giỏ hàng trở thành hình 4.2. Tiếp tục thêm sản phẩm product_id = 2 với số lượng 5 thì giỏ hàng sẽ tự động cập nhật dữ liệu quantity trước đó (hình 4.3).

- Thêm sản phẩm mới vào giỏ hàng:

```
|| INSERT INTO Shopping_cart (customer_id, product_id, quantity) VALUES (1,
||                      2, 3);
```

- Cập nhật số lượng sản phẩm:

```
|| UPDATE Shopping_cart SET quantity = quantity + 5 WHERE customer_id = 1 AND
||                      product_id = 2;
```

	id	customer_id	product_id	quantity	date
1	50,005	1	1	40	2025-04-13 14:04:46
2	50,006	1	2	3	2025-04-30 02:54:18
3	50,001	1	8	9	2025-04-13 11:06:05
4	50,004	1	14	91	2025-04-13 11:06:05

Hình 4.2: Dữ liệu của giỏ hàng sau khi thêm product

	id	customer_id	product_id	quantity	date
1	50,005	1	1	40	2025-04-13 14:04:46
2	50,006	1	2	8	2025-04-30 02:56:30
3	50,001	1	8	9	2025-04-13 11:06:05
4	50,004	1	14	91	2025-04-13 11:06:05

Hình 4.3: Số lượng sản phẩm được tự động cập nhật

4.1.3. Thủ tục get_cart

Mô tả: Lấy danh sách các sản phẩm trong giỏ hàng của khách hàng dưới dạng JSON.

Tham số:

- **Đầu vào:**

- customer_id_param (INT): ID của khách hàng.

- **Đầu ra:**

- list_item (JSON): Danh sách các sản phẩm trong giỏ hàng, bao gồm thông tin như ID sản phẩm, tên, giá, số lượng, và tổng giá trị.

Hoạt động:

1. Truy vấn bảng shopping_cart và products để lấy thông tin các sản phẩm trong giỏ hàng của khách hàng.
2. Tạo một mảng JSON chứa thông tin chi tiết của từng sản phẩm (ID, tên, giá, số lượng, tổng giá).
3. Gán kết quả vào list_item.

Thực hiện kiểm thử thủ tục:

```
CALL get_cart(1, @list_item);
SELECT @list_item;
```

Kết quả:

```
[{"name": "Ao so mi nam", "price": 350000.00, "total": 14000000.00, "quantity": 40, "product_id": 1},
 {"name": "Ao thun nu", "price": 200000.00, "total": 1600000.00, "quantity": 8, "product_id": 2}]
```

4.1.4. Thủ tục Make_product_for_temp_cart

Mô tả: Tạo thông tin sản phẩm cho giỏ hàng tạm (trên tầng cache) và kiểm tra tồn kho.

Tham số:

- **Đầu vào:**

- product_id_param (INT): ID của sản phẩm.
 - quantity_param (INT): Số lượng sản phẩm yêu cầu.

- **Đầu ra:**

- list_items (JSON): Thông tin sản phẩm (ID, tên, giá, số lượng, tổng giá) hoặc thông báo lỗi nếu không đủ tồn kho hoặc không tìm thấy sản phẩm.

Hoạt động:

-
- Lấy số lượng tồn kho từ bảng product_storage dựa trên product_id_param.
 - Nếu không tìm thấy sản phẩm, trả về lỗi "Product not found".
 - Nếu số lượng yêu cầu vượt quá tồn kho, trả về lỗi "Not enough inventory".
 - Nếu đủ tồn kho, trả về thông tin sản phẩm dưới dạng JSON.

Thực hiện kiểm thử:

- **Trường hợp thành công:**

```
|| CALL Make_product_for_temp_cart(1, 15, @list_items);
|| SELECT 'Test_temp_cart' AS Test_Case, @list_items AS List_Items;
```

Kết quả:

```
|| Test_temp_cart [{"name": "Ao so mi nam", "price": 350000.00, "total": 5250000.00,
|| "quantity": 15, "product_id": 1}]
```

- **Trường hợp thất bại (vượt quá tồn kho):**

```
|| CALL Make_product_for_temp_cart(1, 10000, @list_items);
|| SELECT 'Test_temp_cart' AS Test_Case, @list_items AS List_Items;
```

Kết quả:

```
|| Test_temp_cart {"error": "Not enough inventory"}
```

4.1.5 Thủ tục complete_payment

Mô tả: Xử lý thanh toán cho một đơn hàng, cập nhật trạng thái hóa đơn và đơn hàng.

Tham số:

- **Đầu vào:**

- p_order_id (INT): ID của đơn hàng.

- **Đầu ra:**

- p_error_message (VARCHAR): Thông báo lỗi nếu có (NULL nếu thành công).

Hoạt động:

1. Kiểm tra xem hóa đơn tương ứng với p_order_id có tồn tại trong bảng bill không.
2. Nếu không tìm thấy, trả về lỗi "Không tìm thấy hóa đơn".

-
3. Nếu hóa đơn đã được thanh toán (paid), trả về lỗi "Hóa đơn đã được thanh toán trước đó".
 4. Nếu hóa đơn bị hủy (canceled) hoặc đã hoàn tiền (refunded), trả về lỗi "Không thể thanh toán".
 5. Cập nhật trạng thái hóa đơn thành paid và ghi lại ngày thanh toán.
 6. Cập nhật trạng thái đơn hàng trong bảng orders thành received.
 7. Trả về p_error_message là NULL nếu thành công.

Kiểm thử:

- Lấy đơn hàng có trạng thái processing:

```
|| SELECT * FROM orders WHERE orders.status = 'processing' LIMIT 2;
```

0	123 id	0 date	123 customer_id	A2 status	123 total_amount	123 employee_id	123 discount_order_id
1	2	2024-09-28 09:24:50	7,306	processing	46,799,940.34	760	[NULL]
2	5	2025-04-01 12:26:07	4,500	processing	2,377,499,861.98	7,761	574

Hình 4.4: Lấy đơn hàng có trạng thái processing

- Thực hiện complete_payment với đơn hàng ID = 5:

```
|| CALL complete_payment(5, @msg);
SELECT @msg;
SELECT * FROM orders WHERE orders.id = 5;
```

0	123 id	0 date	123 customer_id	A2 status	123 total_amount	123 employee_id	123 discount_order_id
1	5	2025-04-01 12:26:07	4,500	received	2,377,499,861.98	7,761	574

Hình 4.5: Thực hiện complete_payment với đơn hàng 5

Kết quả: Cập nhật thành công, đơn hàng chuyển sang trạng thái received.

- Thực hiện complete_payment với đơn hàng ID = 2:

```
|| CALL complete_payment(2, @msg);
SELECT @msg;
```

Kết quả: Không thể thực hiện, lý do hóa đơn đã bị canceled. (Hình 4.6)

	HoaDon	VanChuyen
1	canceled	[NULL]

Hình 4.6: Hóa đơn đã hủy

4.1.5. Thủ tục CreateUser

Mô tả: Tạo một người dùng mới với vai trò cụ thể (Customer, Employee, hoặc Admin) và lưu thông tin vào các bảng tương ứng.

Tham số:

- **Đầu vào:**

- p_username, p_password, p_role, p_fullname, p_sex, p_email, p_address, p_dob, p_department, p_internal_id, p_hire_date, p_admin_level, p_granted_date.

Hoạt động:

1. Kiểm tra các trường bắt buộc và đảm bảo username chưa tồn tại.
2. Kiểm tra dữ liệu cần thiết theo từng vai trò.
3. Bắt đầu giao dịch:

- Thêm vào bảng user, lấy ID vừa thêm.
- Thêm thông tin vào bảng customers, employee hoặc admin tùy theo vai trò.
- Nếu lỗi thì rollback, nếu thành công thì commit.

Kiểm thử:

Tạo 1 user với vai trò khách hàng:

```
CALL CreateUser(
    'customer123',
    'hashed_password',
    'Customer',
    'Nguyen Van A',
    'Male',
    'customer@example.com',
    '123 Duong ABC',
    '1990-01-01',
```

```

    NULL,
    NULL,
    NULL,
    NULL,
    NULL
);

```

CALL CreateUser('customer123', hashed_password)		
	New User ID	Status Message
1	10,001	Tạo người dùng Customer thành công

Hình 4.7: Tạo người dùng thành công.

Kiểm tra kết quả:

```

SELECT * FROM `user` u JOIN customers c ON c.id = u.id WHERE u.id = 10001;

```

Kết quả: Người dùng mới được tạo thành công, dữ liệu được lưu vào bảng user và customers.

CALL GetUserDetails(10001)									
ID	ID	Username	password_hashed	Role	Created Date	Updated Date	Fullname	Sex	Email
1	10,001	customer123	hashed_password	Customer	2025-04-30 03:33:57	2025-04-30 03:33:57	Nguyễn Văn A	Male	customer@example.com

Hình 4.8: Truy vấn người dùng mới tạo.

4.1.7 Thủ tục CreateOrder

Mô tả: Tạo một đơn hàng mới cho khách hàng với danh sách sản phẩm được cung cấp dưới dạng JSON.

Tham số:

- Đầu vào:**

- p_customer_id (INT): ID của khách hàng.
- p_employee_id (INT): ID của nhân viên xử lý đơn hàng.
- p_status (ENUM): Trạng thái đơn hàng (delivered, received, processing).
- p_discount_order_id (INT): ID của chương trình giảm giá (nếu có).
- p_products (JSON): Danh sách sản phẩm với product_id và quantity.

Hoạt động:

1. Bắt đầu giao dịch:

-
2. Thêm đơn hàng mới vào bảng orders.
 3. Lấy ID đơn hàng vừa tạo.
 4. Duyệt qua danh sách sản phẩm trong p_products:
 - Trích xuất product_id, quantity, thêm vào order_detail.
 5. Nếu lỗi thì rollback và trả về lỗi.
 6. Nếu thành công thì commit và trả về ID đơn hàng mới.

Lưu ý: Việc kiểm thử hàm tạo đơn hàng sẽ được thực hiện ở phần mua nhiều sản phẩm thông qua cache.

4.2. Lập trình hàm

Các hàm được sử dụng để thực hiện các phép tính hoặc kiểm tra logic, trả về một giá trị duy nhất. Dưới đây là danh sách các hàm và mô tả chức năng của chúng:

4.2.1. check_product_inventory

Mô tả: Kiểm tra xem số lượng tồn kho của một sản phẩm có đủ để đáp ứng yêu cầu đặt hàng hay không.

Tham số:

- product_id_param (INT) : ID của sản phẩm.
- quantity_param (INT) : Số lượng cần kiểm tra.

Kết quả trả về: BOOLEAN (TRUE nếu tồn kho đủ, FALSE nếu không đủ).

Logic:

- Tính tổng số lượng tồn kho từ bảng product_storage cho sản phẩm được chỉ định.
- So sánh tổng tồn kho với số lượng yêu cầu.

Test case:

```
||| SELECT check_product_inventory(1, 5) AS is_inventory_sufficient;  
||| SELECT check_product_inventory(1, 10000000) AS is_inventory_sufficient;
```

4.2.2. calculate_product_discount

Mô tả: Tính giá trị giảm giá cho một sản phẩm dựa trên mã giảm giá.

Tham số:

1	123 is_inventory_sufficient	1
---	-----------------------------	---

Hình 4.9: Số lượng tồn kho đủ yêu cầu.

1	123 is_inventory_sufficient	0
---	-----------------------------	---

Hình 4.10: Số lượng tồn kho không đủ yêu cầu.

- product_id_param (INT) : ID của sản phẩm.
- price_param (DECIMAL) : Giá đơn vị của sản phẩm.
- discount_id_param (INT) : ID của mã giảm giá.
- quantity_param (INT) : Số lượng sản phẩm.

Kết quả trả về: DECIMAL (10, 2) – Số tiền giảm giá.

Logic:

- Nếu discount_id_param là NULL, trả về 0.
- Truy xuất category_id của sản phẩm từ bảng products.
- Truy xuất thông tin từ bảng discount với điều kiện:
 - isActive = TRUE
 - CURRENT_TIMESTAMP nằm trong khoảng hiệu lực
 - usagecount < usagelimit
- Kiểm tra:
 - Loại là 'Discount on categories'
 - Danh mục khớp với sản phẩm
 - Giá trị sản phẩm ($price \times quantity \geq min_value_apply$)
- Nếu hợp lệ, trả về max_discount.

Test case:

```
-- Test 1 valid discount --
SELECT calculate_product_discount(3, 467.58, 6, 3) AS product_discount;
SELECT * FROM discount WHERE type_discount = 'Discount on categories';
SELECT * FROM products WHERE category_id = 2;
-- Test 2 invalid discount --
SELECT calculate_product_discount(3, 10, 6, 3) AS product_discount;
```

SELECT calculate_product_discount(3, 40);	
	product_discount
1	41.51

Hình 4.11: Giá trị hợp lệ, trả về số tiền mà sản phẩm đó được giảm.

SELECT calculate_product_discount(3, 40);	
	product_discount
1	0

Hình 4.12: Giá trị không hợp lệ, số tiền được giảm là 0.

4.2.3. calculate_order_discount

Mô tả: Tính giá trị giảm giá cho toàn bộ đơn hàng.

Tham số:

- total_param (DECIMAL): Tổng giá trị đơn hàng.
- discount_id_param (INT): ID mã giảm giá.

Kết quả trả về: DECIMAL (15, 2) – Số tiền giảm giá.

Logic:

- Nếu discount_id_param là NULL, trả về 0.
- Truy xuất mã giảm giá với điều kiện hợp lệ.
- Nếu loại là 'Discount on order' và total_param ≥ min_value_apply, trả về max_discount.

Test case:

```
SELECT * FROM discount WHERE type_discount = 'Discount on order';
-- Test 1 valid case --
SELECT calculate_order_discount(1000, 2);
```

```

|| -- Test 2 invalid case --
|| SELECT calculate_order_discount(10, 2);

```

	123 product_discount	▼
1	0	

Hình 4.13: Giá trị hợp lệ, trả về số tiền giảm cho đơn hàng.

	123 calculate_order_discount(10, 2)	▼
1	0	

Hình 4.14: Giá trị không hợp lệ, đơn hàng không được giảm.

4.2.4. calculate_delivery_discount

Mô tả: Tính giá trị giảm giá cho phí vận chuyển.

Tham số:

- price_delivery_param (DECIMAL) : Phí vận chuyển.
- discount_id_param (INT) : ID mã giảm giá.

Kết quả trả về: DECIMAL (10, 2) – Số tiền giảm giá.

Logic:

- Nếu discount_id_param là NULL, trả về 0.
- Truy xuất mã giảm giá hợp lệ.
- Nếu loại là 'Discount on delivery' và phí vận chuyển \geq min_value_apply, trả về max_discount.

Test case:

```

SELECT * FROM discount WHERE type_discount = 'Discount on delivery';
-- Valid case --
SELECT calculate_delivery_discount(10000, 4) AS product_discount;
-- Invalid case --
SELECT calculate_delivery_discount(10, 4) AS product_discount;

```

SELECT calculate_delivery_discount(10)	
	delivery_discount
1	21.98

Hình 4.15: Giá trị hợp lệ, trả về số tiền vận chuyển được giảm.

SELECT calculate_delivery_discount(10)	
	delivery_discount
1	0

Hình 4.16: Giá trị không hợp lệ.

4.2.5. calculate_loyalty_points

Mô tả: Tính điểm tích lũy cho khách hàng dựa trên tổng giá trị đơn hàng.

Tham số:

- total_amount_param (DECIMAL) : Tổng giá trị đơn hàng.

Kết quả trả về: INT – Số điểm tích lũy.

Logic:

- Cứ mỗi 10.000 VND được 1 điểm, làm tròn xuống.

Test case:

```
|| SELECT calculate_loyalty_points(10000) AS loyal_point;
```

+-----+-----+	
	loyal_point
1	1

Hình 4.17: Khách hàng nhận được 1 điểm thành viên tương đương với 10000 VND giá trị đơn hàng.

4.3. Lập trình Trigger

Trigger được sử dụng để tự động thực hiện các hành động khi có sự thay đổi trong cơ sở dữ liệu (INSERT, UPDATE, DELETE). Dưới đây là danh sách các trigger và chức năng của chúng.

Để thực hiện kiểm thử các trigger, nhóm chúng tôi đã tạo ra một cơ sở dữ liệu test_quanlybanhang mới để việc kiểm thử được thực hiện dễ dàng và bảo đảm toàn vẹn dữ liệu cho cơ sở dữ liệu chính.

Việc tạo cơ sở dữ liệu để kiểm tra được lưu trong file Tao_CSDL_Test.sql.

4.3.1. update_order_detail_subtotal

Mô tả: Tự động tính và cập nhật giá trị subtotal trong bảng order_detail trước khi thêm một bản ghi mới.

Sự kiện: BEFORE INSERT trên bảng order_detail.

Logic:

- Gọi hàm calculate_product_discount để tính giảm giá cho sản phẩm.
- Tính subtotal = (đơn giá - giảm giá) * số lượng.

Kiểm thử trigger:

Listing 4.1: Test không có giảm giá

```
|| INSERT INTO order_detail (order_id, product_id, quantity)
|| VALUES (1, 2, 1);
```

Listing 4.2: Test có giảm giá

```
|| INSERT INTO order_detail (order_id, product_id, quantity, discount_product_id)
|| VALUES (1, 3, 2, 3);
```

Listing 4.3: Kiểm tra kết quả

```
|| SELECT od.id, od.product_id, od.quantity, od.unit_price,
||        od.discount_product_id, od.subtotal
|| FROM order_detail od
|| WHERE od.order_id = 1;
```

+-----+ <th>+-----+<th>+-----+<th>+-----+<th>+-----+<th>+-----+</th></th></th></th></th>	+-----+ <th>+-----+<th>+-----+<th>+-----+<th>+-----+</th></th></th></th>	+-----+ <th>+-----+<th>+-----+<th>+-----+</th></th></th>	+-----+ <th>+-----+<th>+-----+</th></th>	+-----+ <th>+-----+</th>	+-----+
id <th> product_id <th> quantity <th> unit_price <th> discount_product_id <th> subtotal </th></th></th></th></th>	product_id <th> quantity <th> unit_price <th> discount_product_id <th> subtotal </th></th></th></th>	quantity <th> unit_price <th> discount_product_id <th> subtotal </th></th></th>	unit_price <th> discount_product_id <th> subtotal </th></th>	discount_product_id <th> subtotal </th>	subtotal
1	1	2	999.99	NULL	NULL
4	2	1	1999.99	NULL	1999.99
5	3	2	249.99	3	499.98

Hình 4.18: Giá trị của subtotal sau khi cập nhật.

4.3.2. update_order_total_after_insert

Mô tả: Tự động cập nhật tổng tiền đơn hàng trong bảng orders sau khi thêm một dòng mới vào bảng order_detail.

Sự kiện: AFTER INSERT trên bảng order_detail.

Logic:

-
- Tính tổng subtotal của các dòng chi tiết thuộc đơn hàng.
 - Trừ giảm giá theo đơn hàng nếu có và cập nhật vào total_amount.

Kiểm thử:

Listing 4.4: Tạo đơn hàng

```
|| INSERT INTO orders (id, date, customer_id, status, total_amount,
|| discount_order_id)
|| VALUES (2, NOW(), 2, 'processing', 0, 1);
```

Listing 4.5: Thêm chi tiết đơn hàng

```
|| INSERT INTO order_detail (order_id, product_id, quantity)
|| VALUES (2, 1, 1), (2, 2, 1);
```

Listing 4.6: Kiểm tra tổng tiền

```
|| SELECT id, total_amount FROM orders WHERE id = 2;
```

id	total_amount	sum_subtotal
2	2949.98	2999.98

Hình 4.19: Giá trị của subtotal sau khi thêm bản ghi.

4.3.3. update_order_total_after_update

Mô tả: Cập nhật tổng tiền đơn hàng khi có sự thay đổi trong bảng order_detail.

Sự kiện: AFTER UPDATE trên bảng order_detail.

Logic: Tương tự như trigger update_order_total_after_insert.

Kiểm thử trigger:

- **Bước 1:** Kiểm tra giá trị tổng tiền đơn hàng trước khi cập nhật:

```
||     SELECT total_amount FROM orders WHERE id = 1;
```

- **Bước 2:** Cập nhật số lượng sản phẩm trong chi tiết đơn hàng:

```
||     UPDATE order_detail SET quantity = 2 WHERE order_id = 2 AND product_id
||                           = 2;
```

id	total_amount	sum_subtotal
1	3499.96	3499.96

Hình 4.20: Tổng tiền trước khi cập nhật.

- **Bước 3:** Kiểm tra kết quả sau khi cập nhật:

```
||   SELECT * FROM orders;
```

id	date_yee_id	discount_order_id	customer_id	status	total_amount
1	2025-04-25 09:09:22	NULL	1	processing	5499.95
2	2025-04-25 09:59:21	NULL	1	processing	2949.98

Hình 4.21: Tổng tiền sau khi cập nhật.

4.3.4. update_order_total_after_delete

Mô tả: Cập nhật tổng tiền đơn hàng khi một bản ghi trong order_detail bị xóa.

Sự kiện: AFTER DELETE trên bảng order_detail.

Logic: Tương tự như trigger update_order_total_after_insert.

Kiểm thử trigger:

- **Bước 1:** Kiểm tra giá trị tổng tiền đơn hàng trước khi xóa:

```
||   SELECT total_amount FROM orders WHERE id = 1;
```

```
mysql> SELECT total_amount FROM orders WHERE id = 1;
+-----+
| total_amount |
+-----+
|      5499.95 |
+-----+
1 row in set (0.00 sec)
```

Hình 4.22: Tổng tiền trước khi xóa bản ghi.

- **Bước 2:** Xóa một chi tiết đơn hàng:

```
||   DELETE FROM order_detail WHERE order_id = 1 AND product_id = 2;
```

- **Bước 3:** Kiểm tra lại giá trị bảng đơn hàng và chi tiết đơn hàng:

```
||   SELECT * FROM order_detail;
||   SELECT id, total_amount FROM orders WHERE id = 2;
```

	123 id	123 total_amount
1	2	2,949.98

Hình 4.23: Tổng tiền sau khi xóa bản ghi.

4.3.5. update_discount_usage

Mô tả: Tăng số lần sử dụng mã giảm giá khi thêm một đơn hàng mới.

Sự kiện: AFTER INSERT trên bảng orders.

Logic:

- Nếu đơn hàng sử dụng mã giảm giá, tăng cột usagecount trong bảng discount thêm 1.

Kiểm thử trigger:

Listing 4.7: Kiểm tra dữ liệu mã giảm giá ban đầu

```
|| SELECT id, discount_code, usagecount FROM discount WHERE id = 1;
```

	123 id	AZ discount_code	123 usagecount
1	1	ORDER10	0

Hình 4.24: Dữ liệu mã giảm giá ban đầu.

Listing 4.8: Thêm 1 đơn hàng mới sử dụng mã giảm giá

```
|| INSERT INTO orders (id, date, customer_id, status, total_amount,
    discount_order_id)
VALUES (3, NOW(), 1, 'processing', 500, 1);
```

Listing 4.9: Kiểm tra lại kết quả

```
|| SELECT id, discount_code, usagecount FROM discount WHERE id = 1;
```

	123 id	AZ discount_code	123 usagecount
1	1	ORDER10	1

Hình 4.25: Kết quả sau khi kiểm tra lại mã giảm giá.

4.3.6. update_delivery_discount_usage

Mô tả: Tăng số lần sử dụng mã giảm giá khi thêm một bản ghi vào bảng delivery.

Sự kiện: AFTER INSERT trên bảng delivery.

Logic:

- Nếu vận chuyển sử dụng mã giảm giá, tăng cột usagecount trong bảng discount thêm 1.

Kiểm thử trigger:

Listing 4.10: Kiểm tra dữ liệu mã giảm giá ban đầu

```
|| SELECT id, discount_code, usagecount FROM discount WHERE id = 2;
```

Grid	123 id	A-Z discount_code	123 usagecount
	1	2 SHIP20	0

Hình 4.26: Kiểm tra dữ liệu mã giảm giá khi chưa sử dụng.

Listing 4.11: Thêm dữ liệu vận chuyển sử dụng mã giảm giá

```
|| INSERT INTO delivery (shipping_address, carrier, estimated_date, order_id,
    price_delivery, delivery_discount)
VALUES ('123 ABC Street, Ha Noi', 'GHTK', DATE_ADD(CURRENT_DATE, INTERVAL 3
    DAY), 3, 50.00, 2);
```

Listing 4.12: Kiểm tra lại kết quả

```
|| SELECT id, discount_code, usagecount FROM discount WHERE id = 2;
```

Grid	123 id	A-Z discount_code	123 usagecount
	1	2 SHIP20	1

Hình 4.27: Dữ liệu mã giảm giá sau khi đã sử dụng.

4.3.7. update_inventory_after_order

Mô tả: Cập nhật số lượng tồn kho sau khi thêm một bản ghi vào order_detail.

Sự kiện: AFTER INSERT trên bảng order_detail.

Logic:

- Lặp qua các kho (product_storage) theo thứ tự tồn kho giảm dần.
- Trừ số lượng tồn kho cho đến khi đáp ứng đủ số lượng yêu cầu hoặc không còn kho nào có hàng.

Kiểm thử trigger:

Listing 4.13: Kiểm tra kho trước khi thêm đơn hàng

```
|| SELECT id, product_id, storage_id, inventory FROM product_storage WHERE
|| product_id = 1;
```

	id	product_id	storage_id	inventory
1	1	1	1	100
2	2	1	2	150

Hình 4.28: Kho trước khi thêm đơn hàng.

Listing 4.14: Thêm 1 đơn hàng mới

```
|| INSERT INTO orders (id, date, customer_id, status, total_amount)
|| VALUES (4, NOW(), 2, 'processing', 0);
```

Listing 4.15: Thêm chi tiết đơn hàng

```
|| INSERT INTO order_detail (order_id, product_id, quantity)
|| VALUES (4, 1, 120);
```

Listing 4.16: Kiểm tra lại tồn kho

```
|| SELECT id, product_id, storage_id, inventory FROM product_storage WHERE
|| product_id = 1;
```

	id	product_id	storage_id	inventory
1	1	1	1	0
2	2	1	2	10

Hình 4.29: Kiểm tra lại tồn kho.

4.3.8. before_review_insert

Mô tả: Tự động kiểm tra khách hàng đã mua sản phẩm mới được đưa ra đánh giá.

Sự kiện: BEFORE INSERT trên bảng review.

Logic:

- Kết hợp dữ liệu New.customer_id và New.product_id với bảng order và order_detail để kiểm tra khách hàng đã mua sản phẩm này hay chưa.

Kiểm thử trigger:

Listing 4.17: Tạo 1 user role khách hàng mới

```

    INSERT INTO `User` (username, password_hashed, role, fullname, sex, email)
VALUES ('user5', '$2a$15$somehashedpassword', 'Customer', 'Nguyen Van X',
       'Male', 'X@gmail.com');

    INSERT INTO customers (id, address, Dob, loyalty_points)
VALUES (6, 'Ha Noi', '1999-01-15', 100);

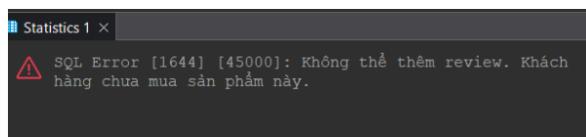
```

Listing 4.18: Thêm dữ liệu vào bảng review (sai điều kiện)

```

    INSERT INTO reviews (customer_id, product_id, rating, content)
VALUES (6, 1, 5, 'good');

```



Hình 4.30: Nếu khách hàng chưa mua sản phẩm thì không thể review.

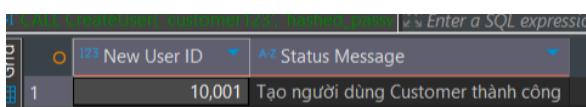
Listing 4.19: Thêm dữ liệu vào bảng review (dúng điều kiện)

```

    UPDATE orders SET status = 'received' WHERE id = 1;

    INSERT INTO reviews (customer_id, product_id, rating, content)
VALUES (1, 1, 5, 'good');

```



Hình 4.31: Dữ liệu có thể được thêm vào bảng review.

Listing 4.20: Kiểm tra kết quả

```

    SELECT * FROM reviews;

```

	id	username	password_hashed	role	created_date	updated_date	fullname	sex	email
exec	10001	customer123	Hashed_password	Customer	2023-04-10 03:31:57	2023-04-10 03:31:57	Nguyen Van A	Male	customer@example.com

Hình 4.32: Kiểm tra dữ đánh giá đã thêm vào bảng review.

4.3.9. before_order_status_update

Mô tả: Tự động kiểm tra điều kiện trước khi điều chỉnh trạng thái đơn hàng.

Sự kiện: BEFORE UPDATE trên trường dữ liệu status của bảng orders.

Logic:

- **Kích hoạt điều kiện:** Trigger chỉ kích hoạt kiểm tra khi trạng thái đơn hàng được cập nhật thành 'delivered' hoặc 'received' và khi trạng thái mới khác trạng thái cũ.

- **Kiểm tra hóa đơn:**

- Trigger kiểm tra xem đơn hàng có hóa đơn liên quan không.
- Nếu có, kiểm tra trạng thái hóa đơn có phải là 'paid' không.

- **Kiểm tra giao hàng:**

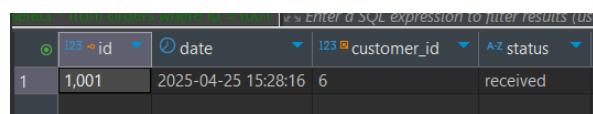
- Trigger kiểm tra xem đơn hàng có thông tin giao hàng không.
- Nếu có, kiểm tra xem trạng thái giao hàng có phải là 'delivered' không (dựa vào việc delivered_date có giá trị).

- **Phản hồi lỗi:** Nếu bất kỳ điều kiện nào không đáp ứng, trigger sẽ phát ra lỗi với thông báo chi tiết về vấn đề gặp phải.

Kiểm thử trigger:

Listing 4.21: Trường hợp đúng điều kiện trigger

```
|| UPDATE orders SET status = 'received' WHERE id = 1001;
|| SELECT * FROM orders WHERE id = 1001;
```

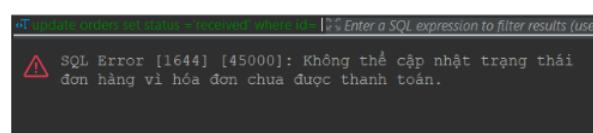


	id	date	customer_id	status
1	1,001	2025-04-25 15:28:16	6	received

Hình 4.33: Trạng thái của đơn hàng được sửa thành công.

Listing 4.22: Trường hợp sai điều kiện trigger

```
|| UPDATE orders SET status = 'received' WHERE id = 1004;
```



SQL Error [1644] [45000]: Không thể cập nhật trạng thái đơn hàng vì hóa đơn chưa được thanh toán.

Hình 4.34: Không thể chỉnh sửa trạng thái đơn hàng.

4.3.10. update_order_detail_price

Mô tả: Tự động tìm và thêm giá trị sản phẩm vào price của order_detail.

Sự kiện: AFTER INSERT trên bảng order_detail.

Logic:

- Kết nối với bảng product để lấy giá sản phẩm tại thời điểm đó.

Kiểm thử:

Listing 4.23: Tạo 1 đơn hàng mới

```
|| INSERT INTO orders (id, date, customer_id, status, total_amount)
|| VALUES (1, NOW(), 1, 'processing', 0);
```

Listing 4.24: Tạo 1 order_detail tương ứng với đơn hàng vừa tạo

```
|| INSERT INTO order_detail (order_id, product_id, quantity)
|| VALUES (1, 1, 2);
```

Listing 4.25: Kiểm tra kết quả

```
|| SELECT od.id, od.product_id, p.price, od.unit_price
|| FROM order_detail od
|| JOIN products p ON od.product_id = p.id
|| WHERE od.order_id = 1;
```

id	product_id	price	unit_price
1	1	999.99	999.99

1 row in set (0.00 sec)

Hình 4.35: Price được thêm vào order_detail

4.4. Lập trình các chức năng triển khai caching với Redis

4.4.1. Lưu trữ phiên đăng nhập

Các hàm triển khai cho chức năng lưu trữ phiên đăng nhập:

- `create_session(user_id, email, fullname, sex, role, address=None, dob=None, department=None, internal_id=None, admin_level=None)`

Mô tả: Tạo một phiên đăng nhập mới trong Redis sau khi đăng nhập thành công. Sinh một token duy nhất (UUID4) và lưu thông tin người dùng dưới dạng JSON với TTL là 180 giây.

Logic:

-
1. Sinh token bằng UUID4.
 2. Tạo dictionary `session_data` với thông tin cơ bản người dùng.
 3. Thêm thông tin bổ sung tùy theo vai trò: customer, employee hoặc admin.
 4. Lưu vào Redis với khóa `session:<token>` và TTL 180 giây.
 5. Trả về token hoặc `None` nếu lỗi.

- **get_user_from_token(token)**

Mô tả: Lấy thông tin người dùng từ Redis thông qua token.

Logic:

1. Kiểm tra tính hợp lệ của token.
2. Lấy dữ liệu từ Redis khóa `session:<token>`.
3. Nếu có dữ liệu, giải mã JSON và trả về dictionary.
4. Nếu lỗi hoặc không tìm thấy, trả về `None`.

- **login(username=None, email=None, password="", role="customer")**

Mô tả: Xác thực thông tin đăng nhập từ MySQL và tạo phiên trong Redis nếu thành công.

Logic:

1. Kiểm tra username/email được cung cấp.
2. Truy vấn MySQL xác thực người dùng.
3. Kiểm tra vai trò đúng.
4. Lấy thông tin bổ sung từ bảng tương ứng.
5. Gọi `create_session()` và trả về token.

- **logout(token)**

Mô tả: Xoá phiên đăng nhập khỏi Redis thông qua token.

Logic:

1. Kiểm tra token hợp lệ.
2. Xoá khóa `session:<token>` khỏi Redis.
3. Trả về trạng thái thành công hoặc lỗi.

- **benchmark_login_vs_cache(email, password, role="customer")**

Mô tả: So sánh thời gian xử lý giữa truy vấn từ MySQL và lấy từ Redis.

4.4.2. Lưu trữ giỏ hàng

- **get_cart(user_id)**

Mô tả: Lấy giỏ hàng từ Redis hoặc MySQL theo mô hình cache-aside.

Logic:

1. Kiểm tra Redis với khóa `cart:<user_id>`.
2. Nếu có, trả về dữ liệu.
3. Nếu không, truy vấn MySQL, lưu Redis với TTL 180 giây rồi trả về.
4. Nếu giỏ hàng trống, gọi `create_cart`.

- **create_cart(user_id)**

Mô tả: Tạo giỏ hàng rỗng trong Redis.

Logic: Tạo dictionary gồm `user_id`, danh sách sản phẩm rỗng, thời gian cập nhật và lưu vào Redis với TTL 180 giây.

- **add_cart(user_id, product_id, quantity=1)**

Mô tả: Thêm sản phẩm vào giỏ hàng trong Redis, cập nhật số lượng nếu sản phẩm đã có.

Logic:

1. Truy vấn MySQL để lấy thông tin sản phẩm.
2. Lấy hoặc tạo giỏ hàng từ Redis.
3. Cập nhật hoặc thêm sản phẩm, lưu lại với TTL 180 giây.

- **delete_cart(user_id, product_id=None)**

Mô tả: Xoá sản phẩm cụ thể hoặc toàn bộ giỏ hàng khỏi Redis.

Logic:

- Nếu có `product_id`, xoá sản phẩm khỏi giỏ và cập nhật Redis.
- Nếu không, xoá toàn bộ khóa `cart:<user_id>`.

- **synchronize_mysql(user_id)**

Mô tả: Đồng bộ giỏ hàng từ Redis sang MySQL để lưu lâu dài.

Logic:

1. Lấy giỏ hàng từ Redis.
2. Kiểm tra và cập nhật vào bảng `shopping_cart`.

- **load_cart_to_redis_after_login(user_id)**

Mô tả: Tải giỏ hàng từ MySQL vào Redis khi người dùng đăng nhập.

Logic: Truy vấn giỏ hàng từ MySQL, lưu vào Redis với TTL 180 giây.

4.4.3. Lưu trữ sản phẩm thịnh hành

- **get_trending_products_from_db(mysql_conn)**

Mô tả: Truy vấn sản phẩm thịnh hành từ MySQL dựa trên số lượng bán ra.

Logic:

1. Kết hợp bảng orders, order_detail, và products.
2. Tính tổng số lượng bán (sales_count), sắp xếp giảm dần, giới hạn 100 sản phẩm.

- **update_trending_cache(redis_client, mysql_conn, update_interval=180)**

Mô tả: Cập nhật cache sản phẩm thịnh hành trong Redis.

Logic:

1. Gọi get_trending_products_from_db.
2. Xoá cache cũ: trending_products_rank, trending_products_details.
3. Lưu vào:
 - Hash: trending_products_details, khóa product:<product_id>, giá trị JSON.
 - Sorted Set: trending_products_rank, với điểm là sales_count.
4. Đặt TTL cho cả hai.

- **get_trending_products(redis_client, mysql_conn, limit=10)**

Mô tả: Lấy danh sách sản phẩm thịnh hành từ Redis hoặc MySQL nếu cache miss.

Logic:

1. Lấy danh sách sản phẩm từ Sorted Set.
2. Nếu có:
 - Lấy chi tiết từ Hash.
 - Trả về danh sách sản phẩm.
3. Nếu cache miss:
 - Gọi update_trending_cache.
 - Gọi lại get_trending_products.

4.5. Thực nghiệm Cache

4.5.1. So sánh hiệu suất khi login bằng cache và không sử dụng cache

Kịch bản thực nghiệm:

1. Chọn 100 user từ cơ sở dữ liệu.

Để thực hiện bước này, chúng tôi viết hàm load_test_data (sample_size=10) với chức năng:

- Truy xuất dữ liệu người dùng từ cơ sở dữ liệu với các trường email, password_hashed và role.
- Xử lý khi không có dữ liệu bằng truy vấn thay thế.
- Kiểm tra định dạng dữ liệu trả về và chuyển đổi nếu cần.
- Ghi log và xử lý lỗi đầy đủ, đảm bảo luôn trả về danh sách người dùng (có thể rỗng).

2. **Thực hiện đăng nhập lần đầu với tất cả user đã chọn từ MySQL.**

3. **Chia 100 user thành các nhóm theo tỷ lệ (20%, 50%, 80%, 100%).**

Các nhóm này sẽ đăng nhập lại bằng token thay vì truy vấn MySQL, tức là sử dụng cơ chế Caching.

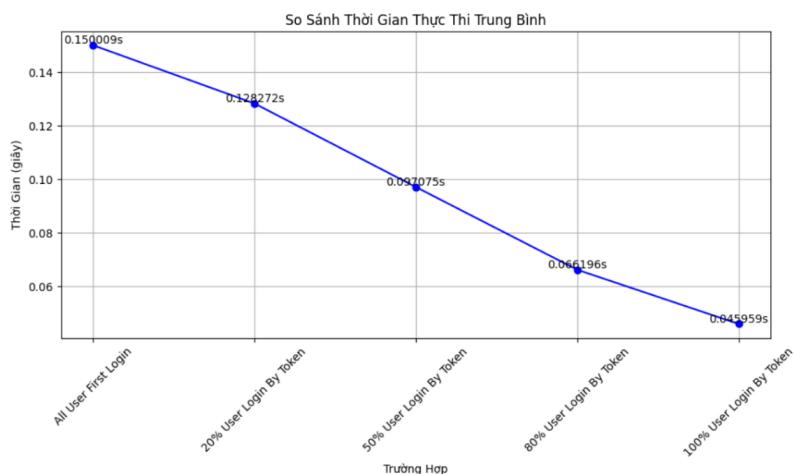
4. **Hàm `benchmark_login_vs_cache(email, password, role, token)` thực hiện đo thời gian:**

- Nếu có email và password: Đăng nhập, đo thời gian và lưu token.
- Nếu có token: Truy xuất thông tin người dùng từ Redis cache, đo thời gian.
- Tính toán tốc độ cải thiện ($\text{speedup} = \text{login_time} / \text{cache_time}$) nếu có đủ dữ liệu.
- Trả về kết quả dưới dạng dictionary: thời gian, token, dữ liệu người dùng, tốc độ cải thiện.

5. **Vẽ biểu đồ so sánh tốc độ của từng tỷ lệ và phân tích kết quả.**

Kết quả Benchmark rút ra được từ 4.36:

- Thời gian đăng nhập trung bình (MySQL): **0.150009 giây**
- Thời gian truy xuất cache trung bình (Redis): **0.045959 giây**
- Tốc độ cải thiện trung bình: **3.26 lần**



Hình 4.36: So sánh thời gian thực thi trung bình.

4.5.2. Thực nghiệm thủ tục nhiều sản phẩm

Các bước thực hiện:

1. Đăng nhập một user.
2. Tự động tải giỏ hàng từ MySQL lên Redis cache.
3. Thêm sản phẩm vào giỏ hàng trên Redis:
 - Thêm sản phẩm #12 - số lượng: 2
 - Thêm sản phẩm #5 - số lượng: 16
4. Thực hiện mua hàng dựa trên thông tin giỏ hàng lưu trên cache.
5. Kiểm tra bảng orders hình 4.37 và order_detail hình 4.38 trong cơ sở dữ liệu chính.

select * from orders where id = 9024 ↵ Enter a SQL expression to filter results (use Ctrl+Space)						
Grid	#	id	date	customer_id	status	total_amount
	1	9,024	2025-04-29 15:36:17	1	processing	0 [NULL] [NULL]

Hình 4.37: Kết quả dữ liệu bảng order.

select * from order_detail where order_id = 9024 ↵ Enter a SQL expression to filter results (use Ctrl+Space)						
Grid	#	id	order_id	product_id	quantity	unit_price
	1	50,006	9,024	12	2	50,000 [NULL] 100,000
	2	50,007	9,024	5	16	1,200,000 [NULL] 19,200,000

Hình 4.38: Kết quả dữ liệu bảng order_detail.

6. Xóa toàn bộ giỏ hàng khi người dùng logout.

Ví dụ giỏ hàng sau khi load từ cache:

```
{  
    "user_id": 1,  
    "items": [  
        {"name": "Ao so mi nam", "price": 350000.0, "total": 14000000.0, "quantity":  
            40, "product_id": 1},  
        {"name": "Dong ho nam Casio", "price": 2500000.0, "total": 22500000.0,  
            "quantity": 9, "product_id": 8},  
        {"name": "Vot cau long Yonex", "price": 1800000.0, "total": 163800000.0,  
            "quantity": 91, "product_id": 14}  
    ],  
    "last_updated": "2025-04-29 22:33:39.011058"  
}
```

Ghi log hệ thống:

- Đăng nhập thành công: user_id = 1
- Load giỏ hàng từ MySQL -> Redis
- Thêm sản phẩm vào Redis cache
- Xóa session và giỏ hàng khi logout

4.5.3. Thực nghiệm sản phẩm thịnh hành

Các bước thực hiện:

1. Gọi thủ tục get_trending_products_from_db(mysql_conn) để lấy dữ liệu từ MySQL.
2. Lưu dữ liệu vào Redis và truy xuất top 5 sản phẩm thịnh hành bằng get_trending_products(redis_client, mysql_conn).

Kết quả dưới dạng JSON (thời điểm cập nhật: 2025-04-29 22:55:19.953303):

```
{'name': 'Kinh ram chong UV', 'price': 500000.0, 'description': 'Kinh ram thoii  
trang chong tia UV', 'sales_count': 15920, 'Top': 1, 'product_id': 19}  
{'name': 'Day chuyen bac nu', 'price': 750000.0, 'description': 'Day chuyen bac  
S925 tinh te', 'sales_count': 15455, 'Top': 2, 'product_id': 7}  
{'name': 'Bong da Adidas', 'price': 600000.0, 'description': 'Bong da tieu  
chuan FIFA Adidas', 'sales_count': 15428, 'Top': 3, 'product_id': 13}  
{'name': 'May xay sinh to', 'price': 1200000.0, 'description': 'May xay sinh to  
Philips cong suat lon', 'sales_count': 15307, 'Top': 4, 'product_id': 17}  
{'name': 'Sach lap trinh Python', 'price': 300000.0, 'description': 'Sach huong  
dan lap trinh Python tu co ban den nang cao', 'sales_count': 15252, 'Top':  
5, 'product_id': 11}
```

4.6. Lập trình View

4.6.1. view_order_full_info

Mô tả chức năng: Hiển thị thông tin đầy đủ về từng đơn hàng, bao gồm khách hàng, nhân viên xử lý, sản phẩm, số lượng, giá, và mã giảm giá áp dụng cho từng sản phẩm trong đơn.

```
CREATE VIEW view_order_full_info AS  
SELECT  
    o.id AS order_id,  
    o.date AS order_date,
```

```

    o.status,
    o.total_amount,
    c.id AS customer_id,
    c.address,
    u.fullname AS employee_name,
    p.name AS product_name,
    od.quantity,
    od.unit_price,
    d.discount_code AS discount_applied
FROM orders o
JOIN customers c ON o.customer_id = c.id
LEFT JOIN user u ON o.employee_id = u.id
JOIN order_detail od ON o.id = od.order_id
JOIN products p ON od.product_id = p.id
LEFT JOIN discount d ON od.discount_product_id = d.id;

```

4.6.2. view_product_inventory

Mô tả chức năng: Hiển thị số lượng tồn kho của từng sản phẩm tại từng kho hàng, giúp kiểm soát và quản lý hàng hóa theo địa điểm.

```

CREATE VIEW view_product_inventory AS
SELECT
    ps.product_id,
    p.name AS product_name,
    ps.storage_id,
    s.city AS warehouse_city,
    ps.inventory
FROM product_storage ps
JOIN products p ON ps.product_id = p.id
JOIN storage s ON ps.storage_id = s.id;

```

4.6.3. view_reviews

Mô tả chức năng: Hiển thị đánh giá sản phẩm từ khách hàng, giúp theo dõi chất lượng sản phẩm và phản hồi người dùng.

```

CREATE VIEW view_reviews AS
SELECT
    r.product_id,
    p.name AS product_name,

```

```

    r.customer_id,
    r.rating,
    r.content,
    r.date
FROM reviews r
JOIN products p ON r.product_id = p.id;

```

4.6.4. view_customer_info

Mô tả chức năng: Cung cấp thông tin hồ sơ khách hàng phục vụ cho thống kê, chăm sóc khách hàng, hoặc phân tích dữ liệu người dùng.

```

CREATE VIEW view_customer_info AS
SELECT
    c.id AS customer_id,
    u.fullname,
    u.email,
    c.address,
    u.sex,
    c.Dob AS date_of_birth,
    u.created_date,
    c.loyalty_points
FROM customers c
JOIN user u ON c.id = u.id
WHERE u.role = 'customer';

```

4.6.5. view_bill_summary

Mô tả chức năng: Hiển thị thông tin tổng hợp về hóa đơn, gồm mã hóa đơn, đơn hàng liên quan, trạng thái, phương thức thanh toán, nhân viên xử lý và tổng tiền hóa đơn.

```

CREATE VIEW view_bill_summary AS
SELECT
    b.id AS bill_id,
    b.order_id,
    b.transaction_id,
    b.create_date,
    b.status,
    b.payment_method,
    u.fullname AS employee_name,
    SUM(od.unit_price * od.quantity) AS total_bill_amount

```

```
||| FROM bill b
||| JOIN orders o ON b.order_id = o.id
||| LEFT JOIN user u ON o.employee_id = u.id
||| JOIN order_detail od ON o.id = od.order_id
||| GROUP BY b.id, b.order_id, b.transaction_id, b.create_date, b.status,
|||          b.payment_method, u.fullname;
```

4.6.6. view_user_employee_admin

Mô tả chức năng: Hiển thị thông tin tổng hợp về nhân viên và admin của hệ thống, phục vụ quản lý nhân sự, phân quyền và giám sát hoạt động.

```
||| CREATE VIEW view_user_employee_admin AS
||| SELECT
|||         u.id AS user_id,
|||         u.username,
|||         u.fullname,
|||         u.role,
|||         u.created_date,
|||         u.updated_date,
|||         e.department,
|||         e.internal_id,
|||         e.hire_date,
|||         a.admin_level,
|||         a.granted_date
|||     FROM user u
|||     LEFT JOIN employee e ON u.id = e.id
|||     LEFT JOIN admin a ON u.id = a.id
|||     WHERE u.role IN ('employee', 'admin');
```

Chương 5 Quản Trị Cơ Sở Dữ Liệu

Giới thiệu

Công tác quản trị cơ sở dữ liệu đóng vai trò then chốt trong việc đảm bảo an toàn, hiệu suất và khả năng phục hồi cho hệ thống bán hàng trực tuyến. Chương này trình bày các thao tác cơ bản trong quản trị CSDL bao gồm tạo người dùng, phân quyền, sao lưu – phục hồi dữ liệu và import/export dữ liệu trong môi trường MySQL.

5.1. Tạo login, user và role

Đầu tiên chúng tôi tiến hành tạo 3 role cơ bản trong cơ sở dữ liệu là user, employee, admin bằng lệnh CREATE ROLE:

```
||| CREATE ROLE 'Customer';
||| CREATE ROLE 'Employee';
||| CREATE ROLE 'Admin';
```

Trong MySQL, ta có thể tạo login, user bằng lệnh CREATE USER:

```
||| CREATE USER 'test_customer'@'localhost' IDENTIFIED BY 'password_customer';
||| CREATE USER 'test_employee'@'localhost' IDENTIFIED BY 'password_employee';
||| CREATE USER 'test_admin'@'localhost' IDENTIFIED BY 'password_admin';
```

5.2. Phân user vào role

Sau khi tạo user, ta tiến hành phân vào một trong ba vai trò cơ bản:

```
||| GRANT 'Customer' TO 'test_customer'@'localhost';
||| GRANT 'Employee' TO 'test_employee'@'localhost';
||| GRANT 'Admin' TO 'test_admin'@'localhost';
```

5.3. Phân quyền

Ta phân quyền cho từng vai trò đối với mỗi bảng. Ví dụ:

- **Customer:** chỉ đọc (READ) sản phẩm, danh mục.

- **Employee:** toàn quyền (CRUD) trên bảng sản phẩm, đơn hàng, lưu kho.
- **Admin:** toàn quyền trên tất cả các bảng.

Bảng 5.1: Phân quyền CRUD theo bảng dữ liệu và vai trò

Bảng dữ liệu	Vai trò	Tạo	Xem	Sửa	Xoá	Ghi chú
products	Khách hàng	-	+	-	-	Khách hàng có thể xem sản phẩm.
	Nhân viên	+	+	+	+	Nhân viên (bán hàng, kho) có thể quản lý sản phẩm.
	Quản trị viên	+	+	+	+	Quản trị viên có toàn quyền với sản phẩm.
categories	Khách hàng	-	+	-	-	Khách hàng có thể xem danh mục.
	Nhân viên	+	+	+	+	Nhân viên (bán hàng) có thể quản lý danh mục.
	Quản trị viên	+	+	+	+	Quản trị viên có toàn quyền với danh mục.
orders	Khách hàng	+	+	-	-	Khách hàng có thể tạo và xem đơn hàng của mình.
	Nhân viên	-	+	+	-	Nhân viên (bán hàng, hỗ trợ) có thể cập nhật trạng thái đơn hàng.
	Quản trị viên	+	+	+	+	Quản trị viên có toàn quyền với đơn hàng.
reviews	Khách hàng	+	+	+	+	Khách hàng có thể quản lý đánh giá của mình.
	Nhân viên	-	+	-	+	Nhân viên (hỗ trợ) có thể xem và xoá đánh giá nếu cần.
	Quản trị viên	+	+	+	+	Quản trị viên có toàn quyền với đánh giá.
customers	Khách hàng	-	+	+	-	Khách hàng có thể xem và cập nhật thông tin cá nhân.
	Nhân viên	-	+	+	-	Nhân viên (hỗ trợ) có thể cập nhật thông tin khách hàng.

Bảng dữ liệu	Vai trò	Tạo	Xem	Sửa	Xoá	Ghi chú
	Quản trị viên	+	+	+	+	Quản trị viên có toàn quyền với dữ liệu khách hàng.
order_detail	Khách hàng	+	+	-	-	Khách hàng có thể thêm sản phẩm và xem chi tiết đơn hàng.
	Nhân viên	-	+	+	+	Nhân viên (kho) có thể quản lý chi tiết đơn hàng.
	Quản trị viên	+	+	+	+	Quản trị viên có toàn quyền với chi tiết đơn hàng.
user	Khách hàng	+	+	+	+	Khách hàng có thể tạo và quản lý tài khoản của mình.
	Nhân viên	-	+	-	-	Nhân viên có thể xem dữ liệu tài khoản (hỗ trợ).
	Quản trị viên	+	+	+	+	Quản trị viên có toàn quyền với tài khoản người dùng.
admin	Khách hàng	-	-	-	-	Không có quyền truy cập.
	Nhân viên	-	-	-	-	Không có quyền truy cập.
	Quản trị viên	+	+	+	+	Quản trị viên quản lý vai trò và cấp quyền.
employee	Khách hàng	-	-	-	-	Không có quyền truy cập.
	Nhân viên	-	+	-	-	Nhân viên có thể xem dữ liệu nhân viên của mình.
	Quản trị viên	+	+	+	+	Quản trị viên có toàn quyền với dữ liệu nhân viên.
wish_list	Khách hàng	+	+	+	+	Khách hàng có thể quản lý danh sách yêu thích.
	Nhân viên	-	+	-	-	Nhân viên (bán hàng) có thể xem danh sách yêu thích.
	Quản trị viên	+	+	+	+	Quản trị viên có toàn quyền với danh sách yêu thích.

Bảng dữ liệu	Vai trò	Tạo	Xem	Sửa	Xoá	Ghi chú
shopping_cart	Khách hàng	+	+	+	+	Khách hàng có thể quản lý giỏ hàng.
	Nhân viên	-	+	-	-	Nhân viên (hỗ trợ) có thể xem giỏ hàng khách.
	Quản trị viên	+	+	+	+	Quản trị viên có toàn quyền với giỏ hàng.
storage	Khách hàng	-	-	-	-	Không có quyền truy cập.
	Nhân viên	+	+	+	+	Nhân viên (kho) có thể quản lý kho hàng.
	Quản trị viên	+	+	+	+	Quản trị viên có toàn quyền với kho hàng.
product_storage	Khách hàng	-	+	-	-	Khách hàng có thể xem tồn kho sản phẩm.
	Nhân viên	+	+	+	+	Nhân viên (kho) có thể quản lý tồn kho.
	Quản trị viên	+	+	+	+	Quản trị viên có toàn quyền với tồn kho sản phẩm.
bill	Khách hàng	-	+	-	-	Khách hàng có thể xem hóa đơn của mình.
	Nhân viên	+	+	+	+	Nhân viên (tài chính) có thể quản lý hóa đơn.
	Quản trị viên	+	+	+	+	Quản trị viên có toàn quyền với hóa đơn.
delivery	Khách hàng	-	+	-	-	Khách hàng có thể xem thông tin giao hàng.
	Nhân viên	+	+	+	+	Nhân viên (kho, hỗ trợ) có thể quản lý giao hàng.
	Quản trị viên	+	+	+	+	Quản trị viên có toàn quyền với giao hàng.

Bảng dữ liệu	Vai trò	Tạo	Xem	Sửa	Xoá	Ghi chú
discount	Khách hàng	-	+	-	-	Khách hàng có thể xem khuyến mãi.
	Nhân viên	+	+	+	+	Nhân viên (bán hàng) có thể quản lý khuyến mãi.
	Quản trị viên	+	+	+	+	Quản trị viên có toàn quyền với khuyến mãi.

5.4. Backup và Restore cơ sở dữ liệu

Mục đích: Backup và restore là các thành phần thiết yếu trong chiến lược đảm bảo tính liên tục kinh doanh (business continuity), giúp bảo vệ dữ liệu khỏi xóa nhầm, hỏng hóc hoặc mất mát do sự cố.

Cơ chế tự động: Azure Database for MySQL Flexible Server tự động tạo và lưu trữ các bản sao lưu một cách an toàn trong *local redundant storage* hoặc *geo-redundant storage* (tùy cấu hình).

Loại sao lưu:

- Snapshot backups:** Sao lưu toàn bộ dữ liệu dưới dạng ảnh chụp nhanh, thực hiện một lần mỗi ngày.
- Transaction log backups:** Sao lưu binlogs mỗi 5 phút, cho phép khôi phục đến một thời điểm cụ thể (Point-in-time restore - PITR).

5.4.1. Backup

Tần suất và lịch trình:

- Snapshot backup:** Tự động thực hiện hàng ngày, bắt đầu ngay sau khi server được tạo.
- Transaction log backup:** Ghi lại thay đổi mỗi 5 phút. Nếu thất bại do tải cao, hệ thống sẽ thử lại mỗi 20 phút cho đến khi thành công.

Thời gian lưu trữ:

- Mặc định: 7 ngày.
- Có thể tùy chỉnh từ 1–35 ngày.
- Long-term retention (LTR): Lưu trữ lên đến 10 năm (dạng file, hiện tại là bản preview).

Sao lưu theo yêu cầu (On-demand Backup):

- Người dùng có thể kích hoạt thủ công.
- Tối đa 50 bản sao lưu on-demand.

- Dùng cho khôi phục nhanh hơn (giảm thời gian khôi phục đến 90%).

Bảo mật:

- Backup được mã hóa bằng AES 256-bit khi lưu trữ.
- Không thể xuất bản sao lưu ra ngoài, chỉ sử dụng nội bộ trong Azure.

Lưu trữ backup:

- Locally redundant: Mặc định với server không có HA.
- Zone-redundant: Mặc định với server có HA theo vùng.
- Geo-redundant: Tuỳ chọn khi tạo server, cho phép khôi phục ở vùng khác.

Xử lý tải cao: Nếu binlog tăng nhanh do tải cao, hệ thống có thể tạo thêm nhiều snapshot backup mỗi ngày.

Name	Status	Completion time (UTC)	Retained until (UTC)	Type	Actions
Automated backup #7	Completed	2025-04-29 10:30:03.291	2025-05-06 10:30:03.291	Automatic	Fast restore
Automated backup #6	Completed	2025-04-28 10:30:03.415	2025-05-05 10:30:03.415	Automatic	Fast restore
Automated backup #5	Completed	2025-04-27 10:30:03.310	2025-05-04 10:30:03.310	Automatic	Fast restore
Automated backup #4	Completed	2025-04-26 10:30:03.329	2025-05-03 10:30:03.329	Automatic	Fast restore
Automated backup #3	Completed	2025-04-25 10:30:03.359	2025-05-02 10:30:03.359	Automatic	Fast restore
Automated backup #2	Completed	2025-04-24 10:30:03.270	2025-05-01 10:30:03.270	Automatic	Fast restore
Automated backup #1	Completed	2025-04-23 10:30:03.225	2025-04-30 10:30:03.225	Automatic	Fast restore

Hình 5.1: Dịch vụ thực hiện nhiều Snapshot Backup mỗi ngày.

5.4.2. Restore

Các loại khôi phục:

- **Point-in-Time Restore (PITR):** Khôi phục server đến một thời điểm cụ thể trong khoảng thời gian lưu trữ (1-35 ngày). Phù hợp để khôi phục dữ liệu bị xóa nhầm hoặc hỏng do lỗi ứng dụng.
- **Fastest Restore Point:** Dựa vào bản snapshot gần nhất để rút ngắn thời gian khôi phục. Sử dụng bản full backup (snapshot) gần nhất để giảm thời gian khôi phục (lên đến 90% nhanh hơn PITR). Chỉ áp dụng cho các bản sao lưu theo nhu cầu hoặc snapshot tự động.
- **Geo-redundant Restore:** Khôi phục ở vùng địa lý khác (nếu có cấu hình Geo-redundant storage). Chỉ hỗ trợ khôi phục đến thời điểm gần nhất (Latest UTC Now).

- **Restore as Files (LTR Preview):** Khôi phục dạng file vào Azure Blob Storage.

Khôi phục cơ sở dữ liệu quanlybanhangstructuyen sang server mới restorequanybanhangstructuyen:

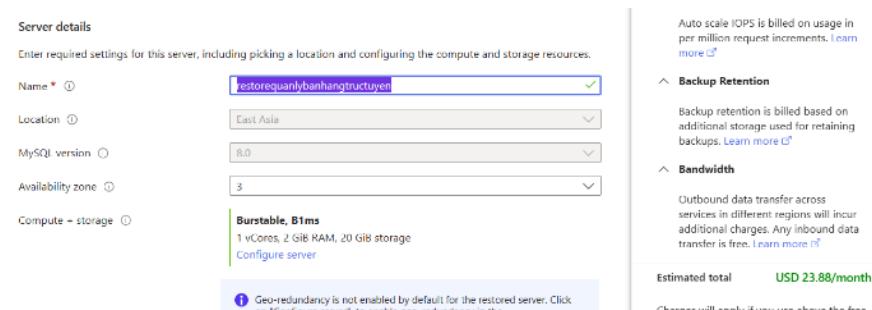
1. Truy cập trang **Overview** của server trên Azure Portal.

2. Chọn **Restore** trên thanh công cụ.

3. Chọn hình thức khôi phục:

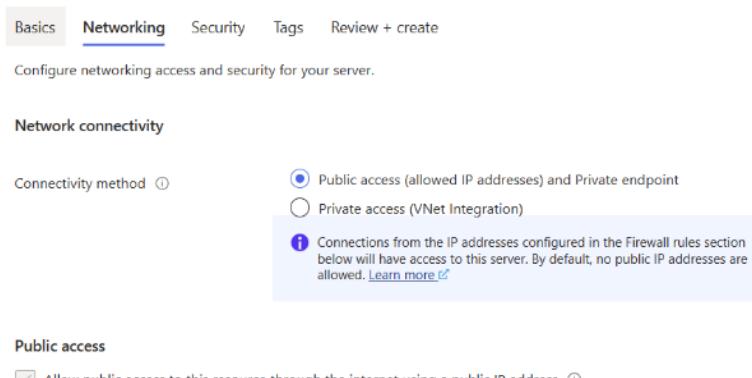
- **Latest restore point**
- **Custom restore point**
- **Fastest restore point**
- **Geo-redundant restore**

4. Cung cấp tên server mới: `restorequanybanhangstructuyen` (khôi phục luôn tạo server mới, không ghi đè server hiện tại).



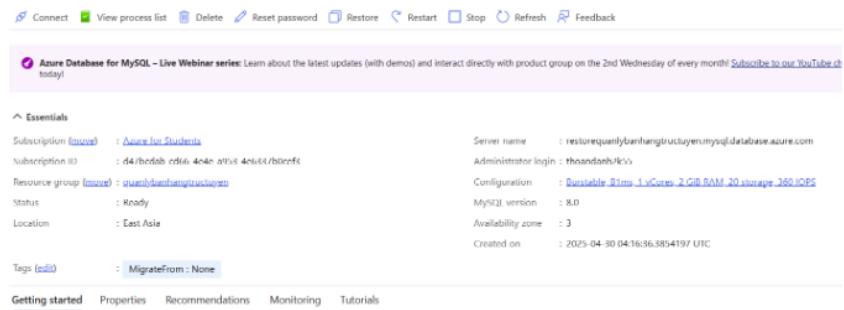
Hình 5.2: Đặt tên cho server mới.

5. Cấu hình tùy chọn mạng (nếu cần, ví dụ: VNet integration).



Hình 5.3: Cấu hình tùy chọn mạng.

6. Chọn **Review + Create** để bắt đầu khôi phục.



Hình 5.4: Bắt đầu khôi phục.

Kết quả kiểm tra bảng:[ht]

```
||| SELECT table_name, table_rows
  ||| FROM information_schema.tables
  ||| WHERE table_schema = 'quanlybanhangtructuyen';
```

Bảng 5.2: Số lượng dòng trong các bảng dữ liệu chính

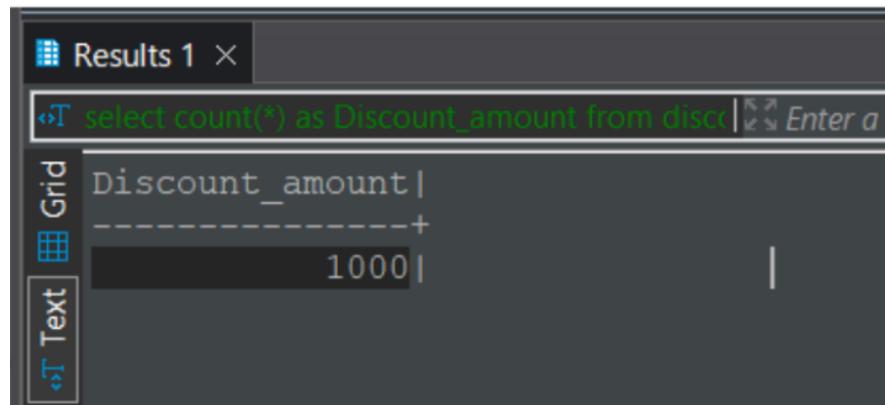
Tên bảng	Số dòng
admin	100
bill	8859
categories	8
customers	8836
delivery	8824
discount	1000
employee	900
order_detail	50159
orders	9244
product_storage	20
products	20
reviews	6079
shopping_cart	44456
storage	6
user	9922
wish_list	19989

5.5. Export dữ liệu và quản lý hiệu suất trên nền tảng Cloud

5.5.1. Import dữ liệu bảng product:

- Mở DBeaver và kết nối đến cơ sở dữ liệu.

-
2. Kiểm tra số lượng discount ban đầu. (Hình 5.5)

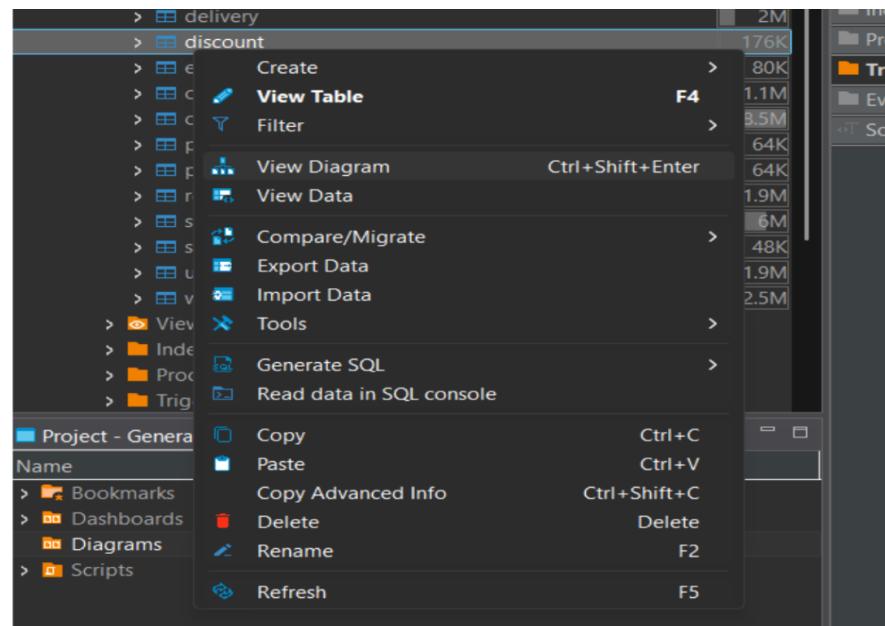


```
Results 1 ×
select count(*) as Discount_amount from discount
|-----+
| 1000 |
```

The screenshot shows the 'Results 1' tab in DBeaver. A SQL query is entered in the text input field: 'select count(*) as Discount_amount from discount'. The result is displayed in a grid with one row, showing 'Discount_amount' as 1000.

Hình 5.5: Số lượng discount ban đầu.

3. Chọn bảng dữ liệu import (ở đây là bảng discount). (Hình 5.6)



Hình 5.6: Chọn bảng discount để import.

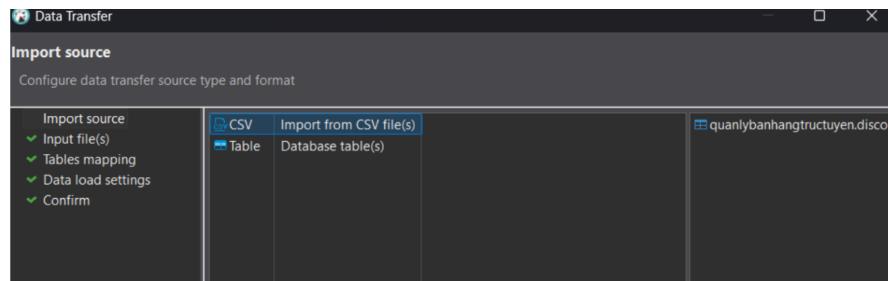
4. Chọn source là csv file. (Hình 5.7)

5. Chọn file đầu vào, tạo mapping, điều chỉnh Data load setting. (Hình 5.8, 5.9, 5.10)

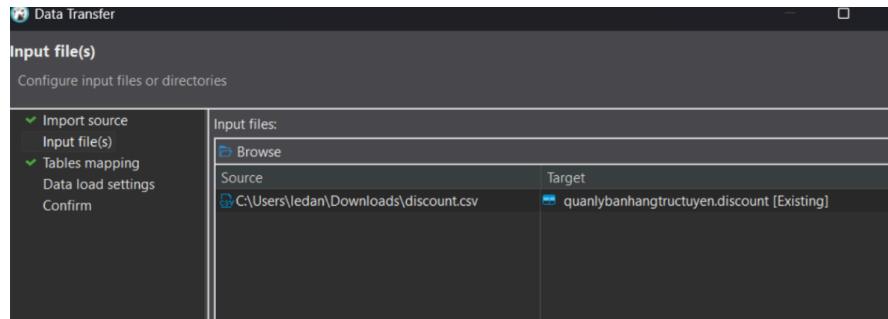
6. Xác nhận import và kiểm tra số lượng bảng dữ liệu discount. (Hình 5.11)

5.5.2. Export dữ liệu bảng user thành file .csv:

1. Mở DBeaver và kết nối đến cơ sở dữ liệu.



Hình 5.7: Chọn source là file csv.



Hình 5.8: Chọn file đầu vào.

2. Tìm bảng user → click đúp để hiển thị dữ liệu.
3. Click chuột phải trên bảng dữ liệu → chọn **Export Data**.
4. Chọn định dạng: **CSV**.
5. Thiết lập nơi lưu file, delimiter, encoding, dòng tiêu đề...
6. Nhấn **Next** → **Finish**.

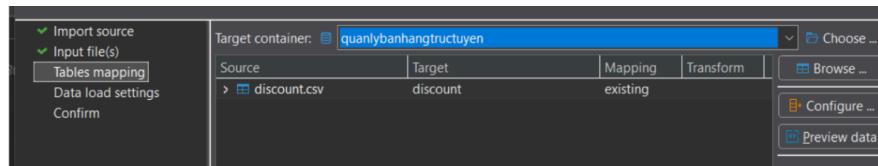
Kết quả: Xuất file user.csv chứa toàn bộ dữ liệu từ bảng.

5.5.3. Export dữ liệu bảng orders theo truy vấn cụ thể:

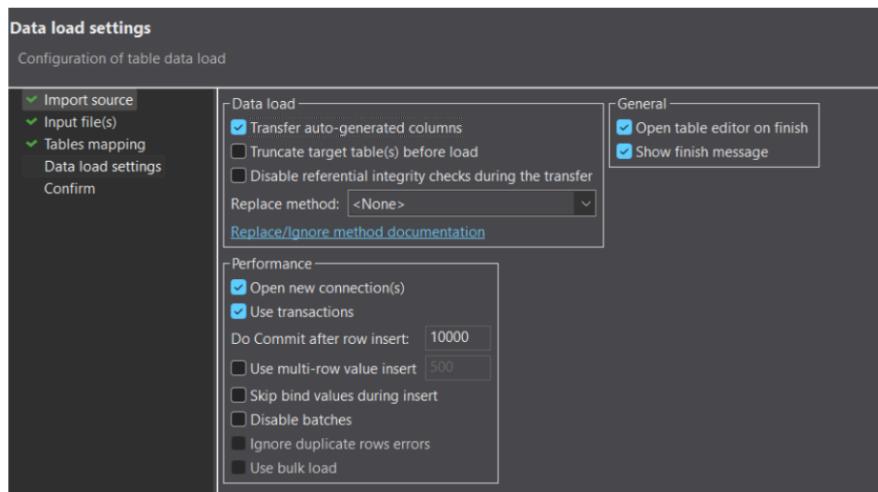
1. Mở SQL Editor, nhập truy vấn:


```
|| select * from orders where status = 'received';
```
2. Chạy truy vấn và chuyển sang tab kết quả.
3. Click chuột phải → **Export Data** → chọn định dạng .txt hoặc .csv.

Kết quả: File chứa dữ liệu đơn hàng có trạng thái received.



Hình 5.9: Tạo mapping.



Hình 5.10: Điều chỉnh Data load setting.

5.5.4. Quản lý hiệu suất cơ sở dữ liệu trên Azure

Giám sát truy vấn:

- Thiết lập Server logs để ghi nhận các truy vấn chậm (Slow query logs) và lỗi (Error logs).

Theo dõi tài nguyên:

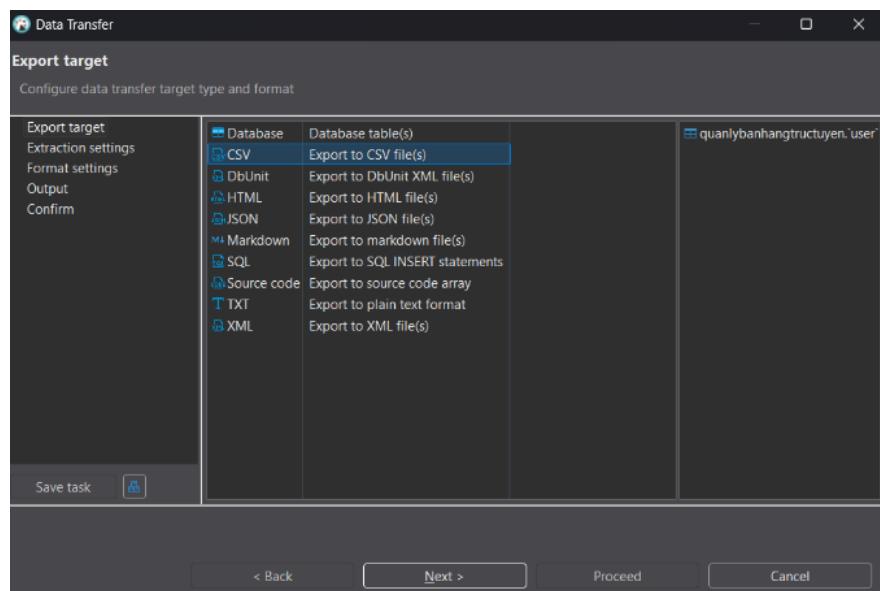
- Giám sát CPU, RAM, IOPS.
- Thiết lập cảnh báo khi tài nguyên vượt ngưỡng.

Tạo cảnh báo CPU_Alerts:

1. Truy cập Azure Monitor → chọn Alerts.
2. Nhấn + Create > Alert Rule.
3. Chọn Resource (VD: CSDL cần theo dõi).
4. Chọn Condition:
 - CPU percentage > 80% for 5 minutes
5. Chọn Action Group để gửi email/SMS.
6. Nhập tên cảnh báo: CPU_Alerts, mô tả và nhấn **Create Alert Rule**.

Discount_amount
1005

Hình 5.11: Xác nhận import và kiểm tra số lượng bảng dữ liệu.



Hình 5.12: Chọn format export data.

user_202504301213.csv	
Tệp	Chỉnh sửa
["id", "username", "password_hashed", "role", "created_date", "updated_date", "fullname", "sex", "email", "1,rushkimberly,d8faeff5e40bcce793759a7514a663ba4fb735866b6a9c370f0d589d47ebddf2,custom", "2023-10-06 08:46:17", "2025-02-27 11:11:22", "Benjamin Gomez, Female, mullinsmichelle@example.org", "2,shuber,"995b2ad8108ff3685baadacede41500ac83e0883dcc23f6f3583b83e7caedc3", "customer", "2024-08-05 04:00:26", "2025-02-07 07:26:55", "Larry Farmer, Female, cheryl97@example.com", "3,jorgemccullough,"195aa2da00847d5688c8c26bb3a887e80d9b56d3d2b7a948a725343ef80565d3", "customer", "2024-10-23 19:51:21", "2025-02-24 16:15:35", "Edward Berry, Female, michelle39@example.net", "4,gloria21,a7b42928fdf5c9f5815bf71da9e731f55b883cce915aea51b374eb00026c2a4,custom", "2024-02-14 20:38:08", "2024-12-27 21:46:42", "Karen Lee, Female, pmiles@example.org", "5,zcrawford,"911fb33db618c465b1715bc309d37505aa22fc68447b28aaf0c8ebd0315572ce", "customer", "2022-10-15 14:05:27", "2024-03-18 00:56:29", "Holly"]	

Hình 5.13: Kết quả file user.csv.

id	date	customer_id	status	total_amount	employee_id	discount_order_id
1	2025-01-22 19:32:55	75	received	934,299,892.21	6,705	
3	2024-08-10 00:20:29	1,246	received	1,423,159,784.61	4,288	
4	2024-03-05 12:48:32	1,739	received	1,941,859,800.34	8,928	
10	2025-01-12 22:22:01	8,703	received	484,019,769.97	2,582	
27	2023-10-05 12:12:34	2,855	received	338,099,971.73	9,603	
44	2023-04-30 02:42:53	7,176	received	2,146,999,917.01	8,885	
48	2025-01-25 01:22:32	1,807	received	507,249,816.64	660	
57	2023-10-01 04:54:19	7,079	received	1,108,949,854.57	5,145	
59	2024-08-11 09:39:24	2,043	received	144,149,854.16	3,897	
75	2024-06-21 21:35:29	6,307	received	1,837,139,856.99	5,463	
82	2025-03-16 14:41:14	4,173	received	866,699,898.19	7,815	
94	2025-02-26 11:26:11	3,931	received	9,449,982.91	2,362	
101	2023-05-08 21:29:09	7,159	received	237,869,871.35	6,541	
106	2025-03-23 13:44:36	3,598	received	486,989,664.75	299	126
107	2025-01-22 23:36:10	3,561	received	384,449,798.96	4,288	
114	2023-02-06 12:15:45	9,143	received	1,298,249,747.2	4,901	

Hình 5.14: Dữ liệu bảng export theo truy vấn.

Save Discard Download Feedback

Configure the service to capture MySQL server logs and upgrade logs, so that you can download them. [Learn more](#)

Server logs

Enable

Select logs to enable.

Slow query logs Error logs (Preview)

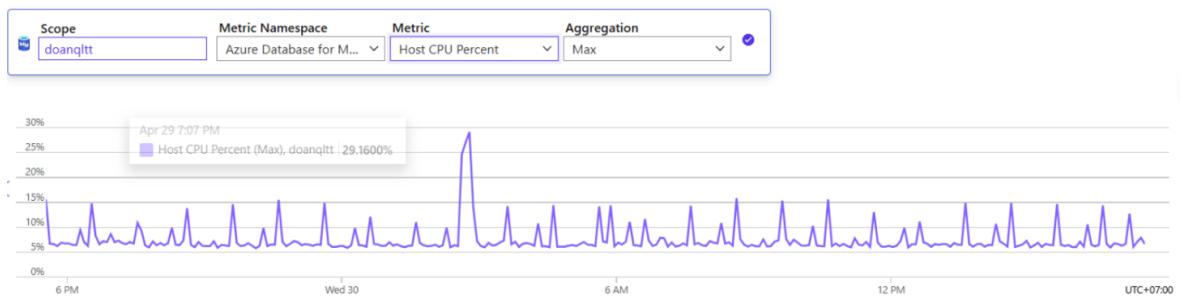
Search for files with names that contain Time range

All

Slow query logs Error logs

Name	Last update time	Size	Actions
No logs with matching filtering conditions were found.			

Hình 5.15: Giám sát truy vấn.



Hình 5.16: Theo dõi tài nguyên.

Scope **Condition** Actions Details Tags Review + create

Configure when the alert rule should trigger by selecting a signal and defining its logic.

Signal name * ⓘ **Host CPU Percent** ▼

[See all signals](#)

Alert logic

Threshold type ⓘ Static Dynamic

Aggregation type ⓘ Maximum ▼

Value is ⓘ Greater than ▼

Threshold * ⓘ **80** % ✓

When to evaluate

Check every ⓘ **1 minute** ▼

Lookback period ⓘ **5 minutes** ▼

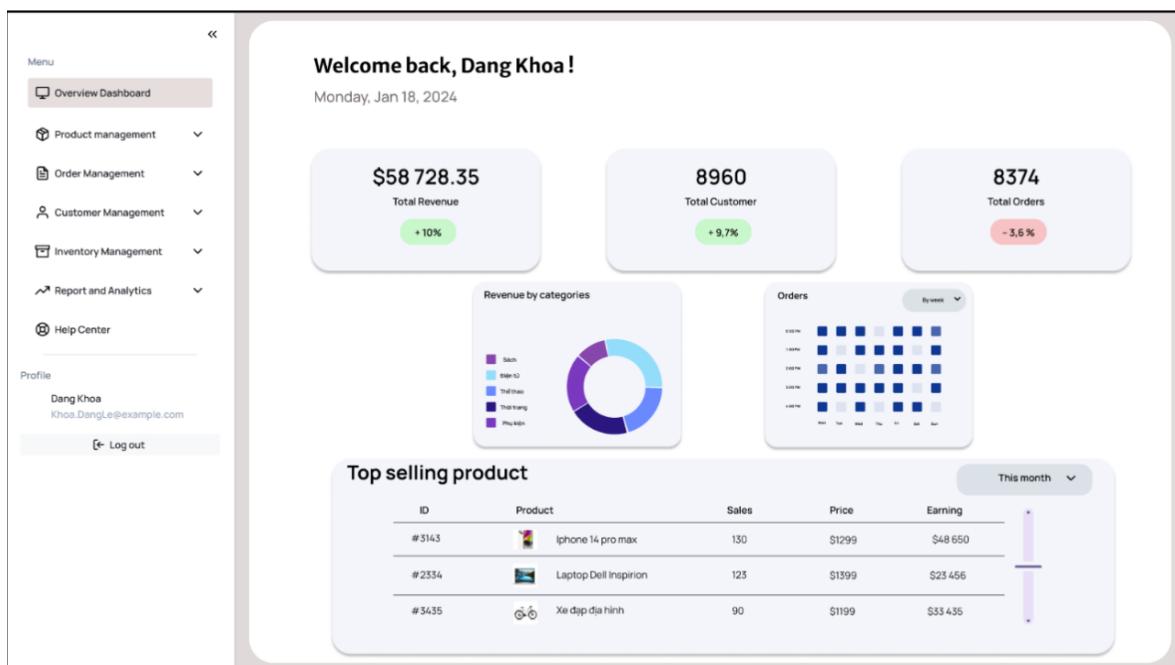
Hình 5.17: Thiết lập cảnh báo.

Chương 6 TRÌNH BÀY THÔNG TIN

6.1. Thiết kế menu

6.1.1. Menu chính của hệ thống

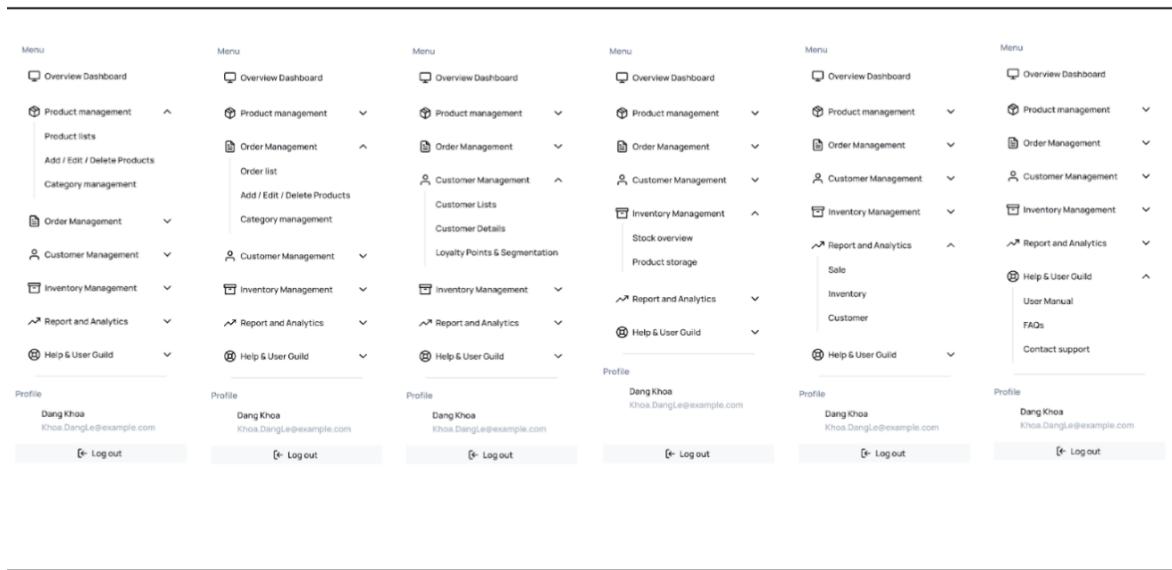
Menu chính của hệ thống được thiết kế giúp người dùng dễ dàng điều hướng đến các chức năng chính.



Hình 6.1: Menu chính của hệ thống

6.1.2. Các mục chính của Menu

- Tổng quan dashboard
- Quản lý sản phẩm
- Quản lý đặt hàng
- Quản lý khách hàng
- Quản lý kho hàng



Hình 6.2: Các mục chính của Menu.

- Báo cáo và phân tích
- Trợ giúp

6.2. Thiết kế form

6.2.1. Form nhập thông tin khách hàng

Let's buy everything you need

You can reach us anytime via @@@@@

Full name
Your name

Email
you@company.com

Gender
 Male
 Female

Date of Birth
DD / MM / YY

Address
eg: 50067 Deborah Parkways Apt. 404, Wendymouth, NM 84562

Discard Saved

Hình 6.3: Form nhập thông tin khách hàng.

Form nhập thông tin khách hàng hình 6.3 bao gồm các trường:

- Họ tên
- Ngày sinh

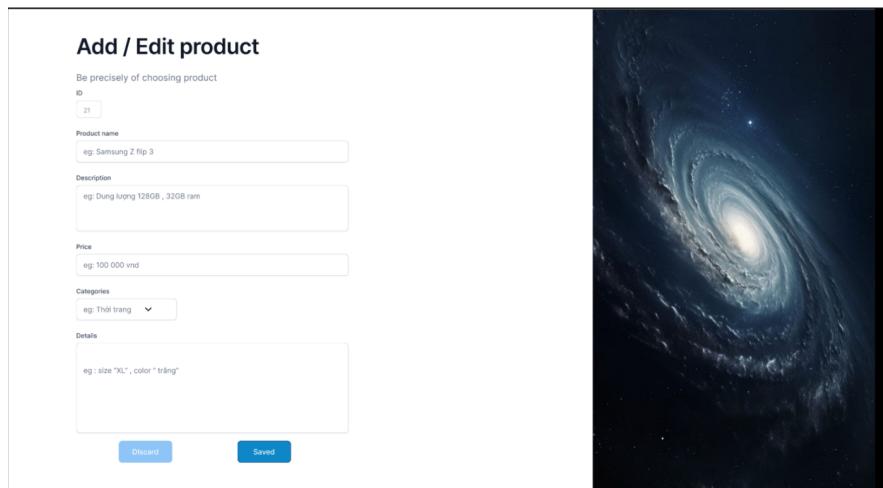
-
- Giới tính

- Địa chỉ

- Số điện thoại

- Email

6.2.2. Form thêm/sửa thông tin sản phẩm



The screenshot shows a user interface for managing product information. On the left, there's a white rectangular form with a title 'Add / Edit product'. Inside the form, there are several input fields: 'ID' (containing '21'), 'Product name' (with placeholder 'eg: Samsung Z flip 3'), 'Description' (with placeholder 'eg: Dung lượng 128GB, 32GB ram'), 'Price' (with placeholder 'eg: 100 000 vnd'), 'Categories' (with placeholder 'eg: Thời trang'), and 'Details' (with placeholder 'eg: size "XL", color "trắng"'). At the bottom of the form are two blue buttons: 'Discard' and 'Saved'. To the right of the form, there is a large, high-resolution image of a spiral galaxy against a dark background.

Hình 6.4: Form thêm sửa thông tin sản phẩm

Form thêm hoặc chỉnh sửa sản phẩm bao gồm:

- Tên sản phẩm
- Loại sản phẩm
- Giá bán
- Số lượng trong kho
- Mô tả sản phẩm
- Hình ảnh sản phẩm

6.2.3. Form thêm/sửa thông tin nhân viên

Form thêm hoặc cập nhật thông tin nhân viên:

- Họ tên nhân viên
- Ngày sinh

Hình 6.5: Form thêm/sửa thông tin nhân viên.

- Giới tính
- Chức vụ
- Email
- Địa chỉ

6.2.4. Form đăng ký

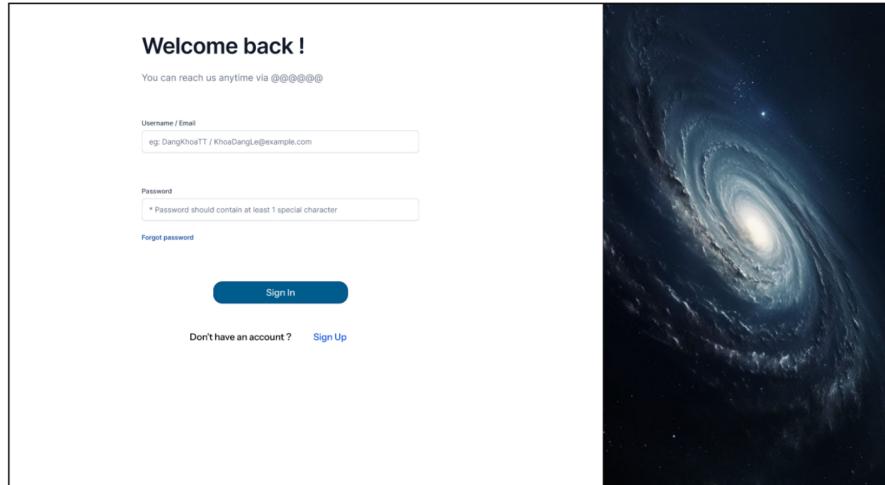
Hình 6.6: Form đăng ký tài khoản cho khách hàng.

Form đăng ký tài khoản dành cho khách hàng bao gồm:

- Họ tên
- Email

-
- Mật khẩu
 - Nhập lại mật khẩu

6.2.5. Form đăng nhập



The image shows a login interface. At the top, it says "Welcome back!". Below that, a note says "You can reach us anytime via ☎️✉️". There are two input fields: one for "Username / Email" with placeholder text "eg: DangKhoaTT / KhoaDangLe@example.com" and another for "Password" with a note "Password should contain at least 1 special character". Below the password field is a "Forgot password?" link. A large blue "Sign In" button is centered below the fields. At the bottom, there are links for "Don't have an account ?" and "Sign Up". The background of the form is a dark image of a spiral galaxy.

Hình 6.7: Form đăng nhập tài khoản.

Form đăng nhập tài khoản yêu cầu:

- Email Tên đăng nhập
- Mật khẩu

6.3. Thiết kế báo cáo (Report)

6.3.1. Report doanh thu

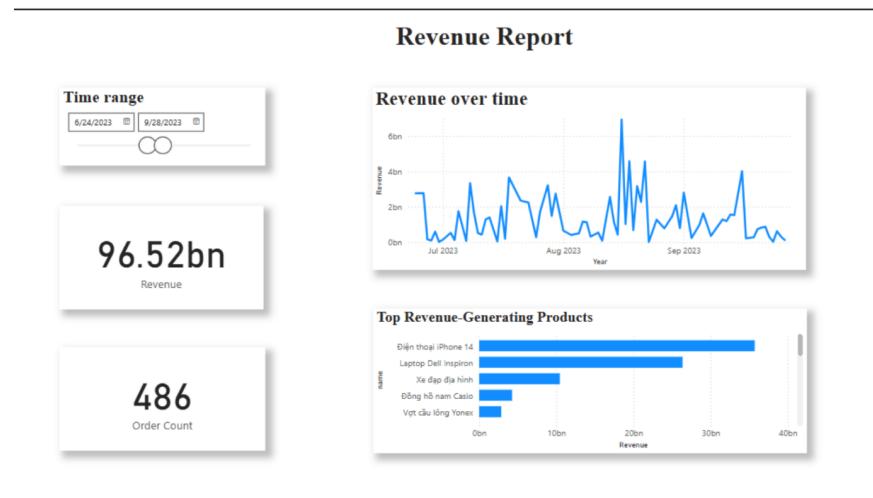
Báo cáo doanh thu hình 6.8 thể hiện tổng doanh thu theo từng ngày, tháng hoặc khoảng thời gian cụ thể. Báo cáo này có thể lọc theo thời gian, trạng thái đơn hàng, và hiển thị dưới dạng bảng hoặc biểu đồ. Ngoài ra còn cho biết mặt hàng có doanh thu cao nhất

6.3.2. Report khách hàng

Báo cáo khách hàng 6.9 thống kê số lượng khách hàng mới, khách hàng thân thiết, tổng số đơn hàng mỗi khách hàng đã thực hiện và tổng giá trị giao dịch.

6.3.3. Report hàng tồn kho

Báo cáo hàng tồn kho hình 6.10 liệt kê danh sách các sản phẩm, số lượng tồn, cảnh báo nếu tồn kho thấp dưới ngưỡng quy định. Báo cáo giúp kiểm soát hàng hóa hiệu quả.



Hình 6.8: Báo cáo doanh thu.

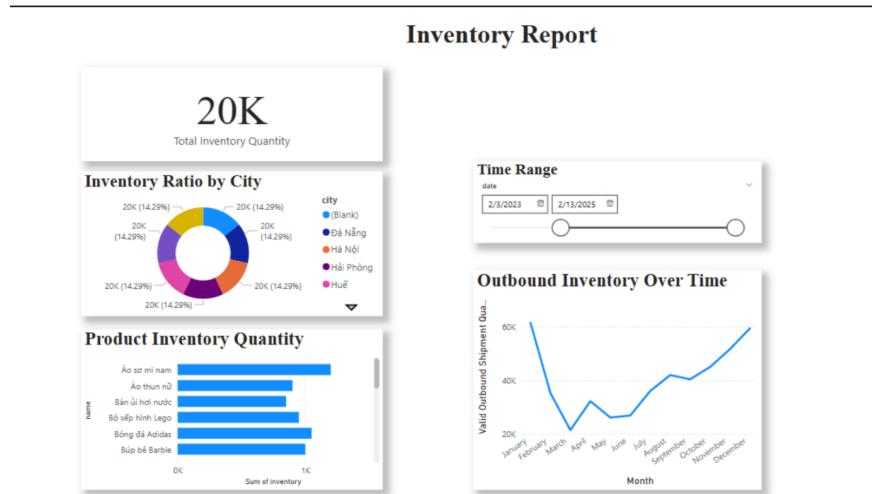


Hình 6.9: Báo cáo khách hàng.

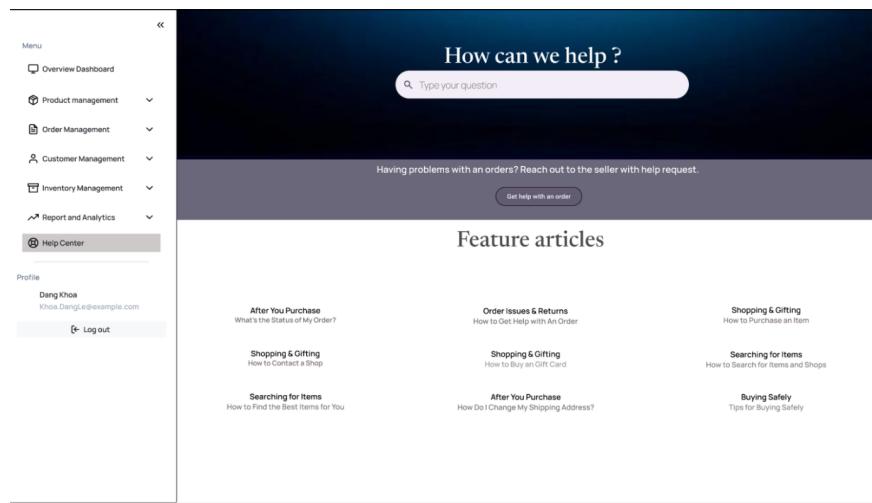
6.4. Thiết kế phần trợ giúp (Help)

Chức năng trợ giúp hình 6.11 được tích hợp trong hệ thống để hỗ trợ người dùng:

- Câu hỏi thường gặp (FAQ)
- Hướng dẫn sử dụng chức năng



Hình 6.10: Báo cáo hàng tồn kho.



Hình 6.11: Phản trợ giúp.

Chương 7 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

7.1. Kết quả đạt được

7.1.1. Thiết kế và triển khai cơ sở dữ liệu

- Đã xây dựng thành công mô hình cơ sở dữ liệu quan hệ MySQL đầy đủ cho hệ thống bán hàng trực tuyến.
- Thiết kế các bảng và mối quan hệ phù hợp cho việc quản lý sản phẩm, người dùng, đơn hàng, thanh toán và các thực thể khác.
- Tối ưu lược đồ lưu trữ của cơ sở dữ liệu. Ví dụ: phân chia bảng `user` thành các bảng nhỏ theo vai trò (`customer`, `employee`, `admin`).

7.1.2. Tối ưu hóa hiệu suất

- Triển khai thành công giải pháp caching với Redis để tăng tốc truy vấn cho các dữ liệu thường xuyên truy cập.
- Xây dựng các chỉ mục (indexes) phù hợp trên các trường thường dùng để tìm kiếm và lọc.
- Tối ưu hóa các câu truy vấn phức tạp để đảm bảo thời gian phản hồi nhanh.

7.1.3. Triển khai lên nền tảng Azure

- Cấu hình và triển khai thành công dịch vụ *Azure Database for MySQL Flexible Server* để lưu trữ dữ liệu nhanh.
- Cấu hình *Azure Cache for Redis* làm giải pháp caching.
- Thiết lập các kết nối an toàn giữa các thành phần cơ sở dữ liệu.

7.2. Ưu điểm và hạn chế

7.2.1. Ưu điểm

- Thiết kế cơ sở dữ liệu tối ưu:** Mô hình dữ liệu được thiết kế tuân theo các nguyên tắc chuẩn hóa, đảm bảo tính toàn vẹn và hiệu quả.

-
- **Hiệu suất truy vấn cao:** Kết hợp giữa MySQL được tối ưu và Redis caching giúp tăng tốc độ truy xuất dữ liệu.
 - **Khả năng mở rộng:** Cấu trúc cơ sở dữ liệu trên Azure cho phép dễ dàng mở rộng theo nhu cầu mà không cần thay đổi lớn về thiết kế.
 - **Bảo mật dữ liệu:** Áp dụng các biện pháp bảo mật cơ sở dữ liệu như mã hóa dữ liệu và phân quyền chi tiết.
 - **Sẵn sàng cao:** Tận dụng các tính năng backup và restore của *Azure Database for MySQL* đảm bảo tính sẵn sàng của dữ liệu.

7.2.2. Hạn chế

- **Chi phí cao:** Việc sử dụng dịch vụ cơ sở dữ liệu quản lý trên Azure có chi phí cao hơn so với giải pháp tự quản lý.
- **Phụ thuộc vào nền tảng Azure:** Thiết kế hiện tại có sự phụ thuộc vào các dịch vụ đặc trưng của Azure.
- **Chiến lược caching chưa toàn diện:** Một số trường hợp caching phức tạp hoặc dữ liệu ít sử dụng chưa được xác định và xử lý tối ưu.

7.3. Hướng phát triển

7.3.1. Cải tiến cơ sở dữ liệu

- Triển khai phân vùng dữ liệu: Áp dụng kỹ thuật phân vùng (*sharding*) cho các bảng lớn như lịch sử đơn hàng, thông tin sản phẩm.
- Tích hợp giải pháp NoSQL: Bổ sung MongoDB hoặc Azure Cosmos DB để lưu trữ hiệu quả dữ liệu phi cấu trúc như thông tin chi tiết sản phẩm, đánh giá người dùng.
- Tối ưu hóa chiến lược caching: Xây dựng chiến lược caching nhiều tầng, kết hợp caching ở nhiều cấp độ (memory cache, Redis, CDN).
- Triển khai thêm cơ sở dữ liệu Media nhằm lưu trữ video, hình ảnh sản phẩm.

7.3.2. Tăng cường hiệu suất và khả năng mở rộng

- Triển khai *Read Replicas* để phân tải các truy vấn đọc.
- Cấu hình tự động mở rộng (*auto-scaling*) cho cơ sở dữ liệu.
- Tối ưu truy vấn nâng cao bằng các công cụ phân tích hiệu suất.

7.3.3. Tăng cường bảo mật và quản lý dữ liệu

- Triển khai mã hóa dữ liệu toàn diện: Áp dụng mã hóa cho dữ liệu nhạy cảm cả khi lưu trữ và truyền tải.
- Xây dựng chiến lược sao lưu và phục hồi tiên tiến: Triển khai sao lưu theo lịch trình và khả năng phục hồi đến một thời điểm cụ thể (*point-in-time recovery*).
- Giám sát và cảnh báo thông minh: Triển khai các giải pháp giám sát hiệu suất cơ sở dữ liệu và cảnh báo dựa trên Azure Monitor.

7.3.4. Triển khai các mô hình AI/ML

- Tận dụng Redis for AI để triển khai chatbots hỗ trợ mua hàng.
- Khai thác dữ liệu lớn tích hợp các công cụ ETL: Xây dựng quy trình ETL (*Extract, Transform, Load*) sử dụng Azure Data Factory để chuyển đổi dữ liệu giữa các hệ thống. Xây dựng mô hình tăng tính cá nhân hóa.

7.3.5. Triển khai tính năng nâng cao

- Triển khai AR để tối ưu hóa trải nghiệm mua hàng.
- Triển khai Elasticsearch và vector search để cải thiện khả năng tìm kiếm.

Tài liệu đính kèm.

Video thực nghiệm chức năng: [drive](#)

Lời cảm ơn

Cảm ơn thầy đã dành thời gian đọc báo cáo của nhóm. Chúng tôi rất mong nhận được sự chỉ bảo và đóng góp ý kiến từ thầy để có thể bổ sung, hoàn thiện hơn về kiến thức cũng như kỹ năng thực hành trong môn học. Sự hướng dẫn tận tình của thầy là động lực giúp nhóm hoàn thành đề tài này.

Tài liệu Tiếng Việt

- [2] Q. N., “Thương mại điện tử Việt Nam tiếp tục bứt phá, hướng đến mục tiêu 25 tỷ USD vào 2025,” *Nhip Sóng Kinh Tế Việt Nam & Thế Giới*, **december** 2024. url: <https://vneconomy.vn/thuong-mai-dien-tu-viet-nam-tiep-tuc-but-pha-huong-den-muc-tieu-25-ty-usd-vao-2025.htm>.
- [4] H. P. Q., *Xu hướng thương mại điện tử tại Việt Nam giai đoạn 2024 – 2025*, Magenest - One-Stop Digital Transformation Solution, **february** 2025. url: <https://magenest.com/vi/xu-huong-thuong-mai-dien-tu/>.

Tài liệu Tiếng Anh

- [1] eMarketer, *Worldwide Retail Ecommerce Forecast 2024*, 2024. url: <https://www.emarketer.com/content/worldwide-retail-ecommerce-forecast-2024>.
- [3] PCMI, *Payments and Commerce Market Intelligence*, Payments and Commerce Market Intelligence - Global Payments Market Research and Insights, february 2024. url: <https://paymentscmi.com/>.
- [5] International Trade Administration, *Trade.gov - Your Source for Global Market Information*, 2025. url: <https://www.trade.gov/>.
- [6] Redis, *Redis: In-Memory Data Structure Store*, 2025. url: <https://redis.io/>.