

TRƯỜNG ĐẠI HỌC BÁCH KHOA
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



HỆ CƠ SỞ DỮ LIỆU (CO2013)

BÀI TẬP LỚN

Thương mại điện tử

Giáo viên hướng dẫn: Dương Huỳnh Anh Đức, CSE-HCMUT

Sinh viên: Nguyễn Tân Phát - 2352888 (CN02)
Vũ Hà Như Ngọc - 2352818 (CN02)
Lê Diệu Quỳnh - 2353036 (CN02)
Lương Đức Huy - 2352384 (CN02)
Nguyễn Ngọc Phát - 2352887 (CN02)

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 10 NĂM 2025



Mục lục

Danh sách Hình ảnh	4
Danh sách Bảng	4
Danh sách thành viên & khối lượng công việc	4
1 Assignment 1	5
1. Phân tích và mô tả yêu cầu dữ liệu cho chủ đề đã chọn	5
1.1. Tìm hiểu ứng dụng/hệ thống tham khảo	5
1.2. Mô tả hệ thống đề xuất	9
1.3. Các ràng buộc ngữ nghĩa không thể hiện trực tiếp trên (E-)ERD	10
2. Mô hình E-ERD	12
3. Ánh xạ lược đồ cơ sở dữ liệu	13
2 Assignment 2	17
2.1 Tạo bảng và dữ liệu mẫu	17
2.1.1 Giới thiệu	17
2.1.2 Mô hình dữ liệu và định hướng thiết kế	17
2.1.3 Hiện thực cơ sở dữ liệu và bảng dữ liệu mẫu	18
2.1.4 Kiểm thử tính đúng đắn của hệ thống	41
2.1.5 Kết luận	42
2.2 Viết các trigger, thủ tục, hàm	42
2.2.1 Thủ tục Insert – Update – Delete và các kiểm tra ràng buộc dữ liệu	42
2.2.1.1 Bảng: User	42
2.2.1.2 Bảng: Buyer	47
2.2.1.3 Bảng: Seller	48
2.2.1.4 Bảng Product	51
2.2.1.5 Order	56
2.2.1.6 OrderLine	60
2.2.1.7 Cart	67
2.2.1.8 CartItem	72
2.2.1.9 Store	78
2.2.1.10 Category	88
2.2.1.11 Product_Variant	94
2.2.1.12 Review	101
2.2.2 Trigger	106
2.2.2.1 Trigger kiểm tra ràng buộc nghiệp vụ	106
2.2.2.2 BẢNG ORDER – Ràng buộc: “Buyer có đơn hàng thì không được xóa Buyer” Ràng buộc nghiệp vụ	106
2.2.2.3 BẢNG ORDERUNIT – Ràng buộc: “Chỉ được tạo Shipment nếu OrderUnit đang ở trạng thái ‘Processing’”	108
2.2.2.4 BẢNG PRODUCT_VARIANT – Ràng buộc: “Không được chuyển biến thể cuối cùng sang sản phẩm khác”	110
2.2.2.5 BẢNG REVIEW – Ràng buộc: “Không được chỉnh sửa đánh giá sau 7 ngày kể từ ngày tạo”	112



2.2.2.6	BẢNG CARTITEM – Ràng buộc: “Không được thêm hoặc cập nhật số lượng vượt quá tồn kho”	114
2.2.2.7	BẢNG SHIPMENT – Ràng buộc: “Chỉ được tạo Shipment nếu OrderUnit đang ở trạng thái Processing”	117
2.2.3	2.2.2 Trigger tính toán thuộc tính dẫn xuất	119
2.2.4	Trigger tính toán thuộc tính dẫn xuất	120
2.2.4.1	A – Hệ thống tính TotalPrice cho Order	120



Danh sách Hình ảnh

1	Giao diện giỏ hàng	6
2	Giao diện trang cá nhân	7
3	Giao diện trang chủ	8
4	Bản vẽ EERD	12
5	User	13
6	Buyer	13
7	Seller	13
8	Store	13
9	Product	13
10	Category	13
11	Order	14
12	PaymentMethod	14
13	Cart	14
14	PaymentTransaction	14
15	ProductVariant	14
16	OrderUnit	15
17	Review	15
18	CartItem	15
19	OrderLine	15
20	Shipment	15
21	ProductCategory	16
22	ProductImage	16
23	ProductVideo	16
24	ReviewImage	16
25	ReviewVideo	16

Danh sách Bảng

1	Danh sách thành viên & khối lượng công việc	4
---	---	---



Danh sách thành viên & khối lượng công việc

STT	Họ Tên	MSSV	Công việc	% Hoàn thành
1	Nguyễn Tân Phát	2352888	Ánh xạ và E-ERD	100%
2	Vũ Hà Như Ngọc	2352818	Mô tả yêu cầu dữ liệu	100%
3	Lê Diệu Quỳnh	2353036	Mô tả yêu cầu dữ liệu	100%
6	Lương Đức Huy	2352384	Ánh xạ và E-ERD	100%
7	Nguyễn Ngọc Phát	2352887	Ánh xạ và E-ERD	100%

Bảng 1: Danh sách thành viên & khối lượng công việc



1 Assignment 1

1. Phân tích và mô tả yêu cầu dữ liệu cho chủ đề đã chọn

1.1. Tìm hiểu ứng dụng/hệ thống tham khảo

Shopee tại thị trường Việt Nam là hệ thống tham khảo chính mà nhóm em lựa chọn. Đó là một nền tảng thương mại điện tử B2C, C2C, nổi tiếng và phổ biến với hàng triệu người dùng. Website chính thức của ứng dụng là <https://shopee.vn>. Các trải nghiệm mang đến cho người dùng ở Shopee là mua sắm trực tuyến từ tìm kiếm sản phẩm, thêm vào giỏ hàng, thanh toán, đến theo dõi quá trình vận chuyển và gửi đánh giá sau mua.

Trong hệ sinh thái Shopee còn có các mô-đun hỗ trợ chuyên biệt. Shopee đã xây dựng nền tảng chuyên biệt cho người bán là Shopee Seller Centre (<https://banhang.shopee.vn>), ngoài ra còn có ứng dụng Shopee dành cho cả người mua và người bán. Về phía nền tảng dành cho người bán thì tại đó, cho phép quản lý cửa hàng, đăng sản phẩm theo danh mục chuẩn hóa, cấu hình vận chuyển, xử lý đơn hàng và triển khai các chương trình khuyến mãi như Flash Sale hoặc tạo voucher. Song song với Shopee Pay (<https://shopeepay.vn>) là ví điện tử tích hợp, hỗ trợ thanh toán nhanh chóng và an toàn, đồng thời cung cấp chức năng quản lý hoàn tiền, nhưng đặc biệt là có chức năng đối soát giao dịch dành cho người bán. Việc tham chiếu 2 module này chỉ để làm rõ thêm nghiệp vụ, đối tượng khảo sát chính vẫn là Shopee.

Luồng nghiệp vụ của Shopee qua khảo sát thực tế đã cho thấy rằng: đối với người mua thì đăng ký tài khoản, duyệt và chọn sản phẩm, thêm vào giỏ hàng, áp mã giảm giá, tiến hành thanh toán bằng nhiều phương thức khác nhau như Shopee Pay, thẻ, COD, theo dõi trạng thái đơn hàng theo từng mốc vận chuyển và gửi đánh giá khi nhận hàng. Đối với người bán thì khởi tạo cửa hàng, niêm yết sản phẩm theo chuẩn dữ liệu và danh mục, quản lý tồn kho, xử lý đơn hàng, đối soát thanh toán và theo dõi hiệu quả kinh doanh thông qua Seller Centre. Ngoài ra, hệ thống logistics tích hợp với nhiều đối tác vận chuyển để cập nhật tình trạng đơn hàng theo từng mốc hành trình đảm bảo thông tin giao nhận được phản ánh kịp thời.

Để minh chứng cho khảo sát, nhóm sẽ tự thao tác và chụp ảnh các màn hình tiêu biểu trên ứng dụng/web như: trang giỏ hàng, trang thanh toán tích hợp Shopee Pay, chi tiết đơn hàng với các mốc vận chuyển, bảng điều khiển Seller Centre (đơn hàng chờ xử lý, Marketing Centre, Account Health), giao diện tạo voucher và trang đánh giá sản phẩm. Các minh chứng này giúp phản ánh chân thực trải nghiệm người dùng và đối chiếu với phân tích nghiệp vụ trong báo cáo.



14:23

57%



Giỏ hàng (30)

Sửa



Mall MITAMALL > Sửa

Bộ Ống Hút Thủy Tinh 84HYDRO Cao Cấp...
Loại hàng đã chọn không còn.

Đổi loại khác

Ông hút thủy tinh Borosilicate cao cấp cho...
Loại hàng đã chọn không còn.

Đổi loại khác

Các sản phẩm bạn có thể chưa cần **Xóa bỏ**

Yêu thích Kho Sỉ Hoàng Dũng > Sửa

Mua thêm 5 sản phẩm để giảm 1% >

COMBO 30 Móc Dán Tường Trọng Suốt, Chị...
10 MÓC - 1 +
đ9.999

Voucher giảm đến đ10k >

Shopee Voucher Chọn hoặc nhập mã >

Bạn chưa chọn sản phẩm ? **Chuyển sang**

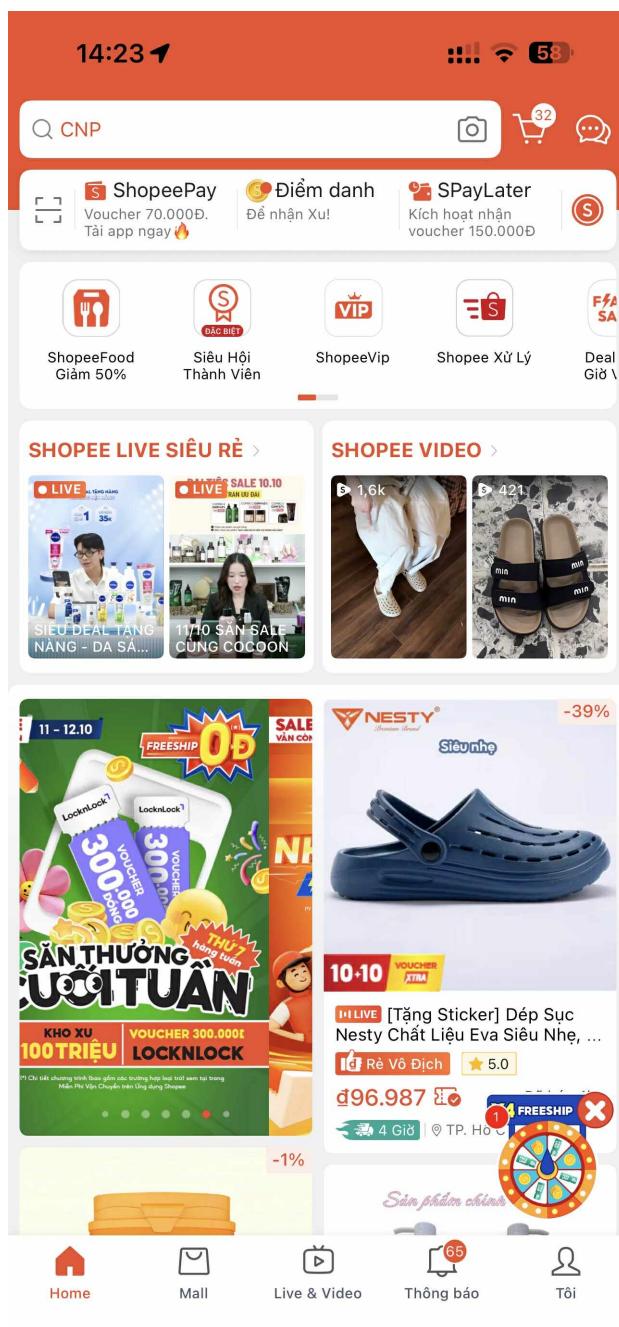
Tất cả **đ0** **Mua hàng (0)**

Hình 1: Giao diện giỏ hàng



The screenshot shows the Shopee mobile application interface. At the top, there is a header bar with the time (14:23), signal strength, battery level (58%), and a 'Bắt đầu bán' (Start selling) button. Below the header is the user profile section for 'dqyn1103' (Silver level), showing 0 people following and 45 people being followed. There is a note to 'Vui lòng chọn Tên của bạn Thiết lập ngay' (Please choose your name and set it up now). Below the profile, there are sections for 'Đơn mua' (Purchase orders) and 'Xem lịch sử mua hàng' (View purchase history). Under 'Đơn mua', there are four categories: 'Chờ xác nhận' (Waiting for confirmation), 'Chờ lấy hàng' (Waiting for pickup), 'Chờ giao hàng' (Waiting for delivery), and 'Đánh giá' (Review). Below these are two promotional banners: one for 'Đơn Nạp điện thoại & Dịch vụ' (Phone recharge & Services) with a 5% discount, and another for 'Đơn ShopeeFood' (ShopeeFood order) with a current promotion. At the bottom, there is a section for '10.10 Đại Tiệc Thương Hiệu' (10.10 Grand Sale) featuring three categories: 'Trang chính' (Main page), 'Hàng Mới Về Sàn' (New products on the platform), and 'Xả Hàng Giá Sốc' (Sudden price drop). Further down, there is a section for 'Tiện ích của tôi' (My conveniences) with icons for ShopeePay, SPayLater, Shopee Xu, and Kho Voucher, along with specific offers like a 300,000đ voucher and a daily Shopee Xu offer. The final section at the bottom is 'Dịch vụ tài chính' (Financial services) with icons for SEasy (New), ShopeePay, and Bảo hiểm của tôi (My insurance), along with offers for a 300,000đ voucher and a mini insurance package. At the very bottom, there are navigation icons for Home, Mall, Live & Video, Thông báo (Notifications), and Tôi (Me).

Hình 2: Giao diện trang cá nhân



Hình 3: Giao diện trang chủ



1.2. Mô tả hệ thống đề xuất

Hệ thống thương mại điện tử Shopee quản lý người dùng, cửa hàng, sản phẩm và các giao dịch mua bán. Cơ sở dữ liệu được mô tả như sau: Trong hệ thống, tất cả tài khoản người dùng được quản lý bằng thực thể Người dùng (USER). Mỗi tài khoản có thông tin cơ bản như số điện thoại, email, vai trò (người mua hoặc người bán), trạng thái tài khoản (chưa xác thực, đã xác thực hay bị khóa) và có một mã định danh duy nhất. Nếu người dùng chọn mua sắm thì sẽ trở thành Người mua (BUYER), còn nếu mở gian hàng thì sẽ là Người bán (SELLER). Hệ thống đảm bảo rằng mỗi tài khoản chỉ có thể mang một trong hai vai trò này.

Dối với người mua, hệ thống gắn cho họ một mã riêng và liên kết duy nhất với một người dùng. Người mua là người đặt đơn hàng và sở hữu Giỏ hàng (CART) của riêng mình, mỗi giỏ hàng có một mã định danh duy nhất. Trong giỏ hàng đó có nhiều Mặt hàng trong giỏ (CART ITEM), mỗi mặt hàng lưu lại mã sản phẩm (SKU) và số lượng mong muốn. Mỗi mặt hàng được gắn với một Biến thể sản phẩm (PRODUCT VARIANT) cụ thể, thể hiện lựa chọn mà người mua đã thêm vào giỏ. Khi người mua tiến hành đặt hàng, hệ thống tạo ra một Đơn hàng (ORDER). Đơn hàng là đơn gốc chứa thông tin như mã, thời điểm tạo, tổng giá trị, chiết khấu và trạng thái thanh toán.

Do một đơn hàng có thể bao gồm các sản phẩm, nên đơn hàng được tách thành nhiều Đơn hàng con (ORDER UNIT), mỗi đơn hàng con tương ứng với một Cửa hàng (STORE) cụ thể. Mỗi đơn hàng con có mã riêng, trạng thái xử lý và phí vận chuyển. Trong từng đơn hàng con, các sản phẩm được liệt kê chi tiết ở mức Dòng đơn hàng (ORDER LINE); mỗi dòng có mã định danh riêng, liên kết với biến thể sản phẩm thông qua mã SKU, kèm theo số lượng, giá đơn vị và chiết khấu áp dụng. Đây là dữ liệu chi tiết nhất của giao dịch mua bán. Khi người bán xác nhận đơn, hệ thống sẽ tạo một Đơn vận chuyển (SHIPMENT) cho đơn hàng con đó. Đơn vận chuyển có mã vận đơn riêng, thông tin đơn vị vận chuyển, trạng thái và lịch trình vận chuyển để theo dõi các mốc như “đã lấy hàng”, “đang giao” hay “đã giao thành công”.

Giao dịch thanh toán (PAYMENT TRANSACTION) lưu toàn bộ thông tin thanh toán của hệ thống. Mỗi giao dịch có số hiệu riêng, thời điểm khởi tạo và được thực hiện bởi người mua. Mỗi người mua có thể tạo nhiều giao dịch thanh toán, và một đơn hàng có thể phát sinh nhiều giao dịch khác nhau (ví dụ: thanh toán lại hoặc hoàn tiền), nhưng mỗi giao dịch chỉ gắn với một đơn hàng duy nhất. Giao dịch thanh toán còn tham chiếu đến Phương thức thanh toán (PAYMENT METHOD) – loại hình thanh toán mà người dùng lựa chọn, bao gồm các thông tin như mã phương thức, loại (ShopeePay, thẻ ngân hàng, COD...) và chi tiết tương ứng. Một người mua có thể lưu nhiều phương thức thanh toán khác nhau, và mỗi phương thức đều có thể được sử dụng trong nhiều giao dịch.

Ở phía người bán, họ được định danh bằng mã riêng và có thông tin xác thực cũng như tài khoản ngân hàng. Mỗi người bán chỉ được phép sở hữu đúng một cửa hàng trên hệ thống. Cửa hàng là nơi đại diện cho người bán, có các thuộc tính như mã cửa hàng, tên, hồ sơ thương hiệu, chính sách đổi trả, trạng thái hoạt động và có thể mang nhãn Mall. Một cửa hàng có thể đăng bán nhiều Sản phẩm (PRODUCT), còn mỗi sản phẩm chỉ thuộc về một cửa hàng.

Các sản phẩm được quản lý trong thực thể sản phẩm. Mỗi sản phẩm có mã sản phẩm duy nhất, tiêu đề, mô tả, hình ảnh, video và trạng thái niêm yết. Một sản phẩm có thể được gắn vào nhiều Danh mục (CATEGORY) khác nhau để người dùng dễ tìm kiếm, và ngược lại, mỗi danh mục có thể bao gồm nhiều sản phẩm. Danh mục được tổ chức theo cấu trúc phân cấp nhiều tầng, giúp việc sắp xếp và tìm kiếm trở nên dễ dàng hơn. Mỗi danh mục có thể có nhiều danh mục con, và mỗi danh mục con sẽ ghi lại mã của danh mục cha thông qua thuộc tính ParentCategoryID.



Một sản phẩm thường có nhiều lựa chọn khác nhau như màu sắc, kích cỡ hoặc phiên bản. Những lựa chọn này được lưu trong thực thể biến thể sản phẩm, mỗi biến thể có mã định danh duy nhất, giá niêm yết, giá khuyến mãi và số lượng tồn kho hiện có.

Khi đơn hàng đã được giao thành công, người mua có thể để lại Đánh giá (REVIEW) cho sản phẩm mình đã mua. Mỗi đánh giá có mã định danh, điểm số, nội dung, hình ảnh, video minh họa và thời điểm tạo. Đánh giá được liên kết với người mua đã viết và sản phẩm được đánh giá. Một người mua có thể viết nhiều đánh giá, và mỗi sản phẩm cũng có thể nhận nhiều đánh giá từ những người mua khác nhau. Nhờ vậy, phần đánh giá giúp người dùng khác tham khảo trước khi mua, đồng thời giúp cửa hàng cải thiện chất lượng sản phẩm của mình.

1.3. Các ràng buộc ngữ nghĩa không thể hiện trực tiếp trên (E-)ERD

Bên cạnh các thực thể và mối quan hệ đã thể hiện trong sơ đồ E-ERD, hệ thống cơ sở dữ liệu còn phải tuân thủ nhiều ràng buộc ngữ nghĩa nhằm đảm bảo tính toàn vẹn dữ liệu và phản ánh chính xác hoạt động thực tế của sàn thương mại điện tử. Những quy tắc này không thể hiện trực tiếp bằng ký hiệu trên sơ đồ, song vẫn được quy định rõ ràng trong quá trình thiết kế hệ thống như sau.

Thứ nhất, mỗi cửa hàng (STORE) chỉ được phép đăng bán sản phẩm (PRODUCT) khi cửa hàng đang ở trạng thái hoạt động (Active). Nếu cửa hàng bị tạm khóa hoặc ngừng hoạt động, mọi thao tác đăng bán sẽ bị vô hiệu.

Thứ hai, người bán (SELLER) chỉ được phép chỉnh sửa thông tin của sản phẩm (PRODUCT) khi sản phẩm đó chưa có đơn hàng (ORDER) nào đang trong quá trình xử lý. Điều này nhằm đảm bảo tính thống nhất giữa dữ liệu sản phẩm và các đơn hàng liên quan.

Thứ ba, giỏ hàng (CART) của người mua (BUYER) sẽ được tự động làm mới sau khi thanh toán thành công. Các mặt hàng trong giỏ (CART ITEM) tương ứng với đơn đã thanh toán sẽ tự động bị xóa khỏi giỏ hàng, giúp người mua dễ dàng quản lý các sản phẩm chưa mua.

Thứ tư, đơn vận chuyển (SHIPMENT) chỉ được khởi tạo khi đơn hàng con (ORDER UNIT) đã được người bán xác nhận. Hệ thống không cho phép tạo vận đơn cho các đơn hàng chưa được xác nhận nhằm tránh thất thoát hàng hóa.

Thứ năm, mỗi đơn vận chuyển (SHIPMENT) đều có thời hạn giao hàng nhất định. Nếu quá X ngày kể từ ngày tạo mà vẫn chưa hoàn tất, hệ thống sẽ tự động đánh dấu đơn đó là thất bại để xử lý hoàn tiền hoặc khiếu nại.

Thứ sáu, người mua (BUYER) không thể hủy đơn hàng (ORDER) sau khi đơn đã được gắn với đơn vận chuyển (SHIPMENT) và đang ở trạng thái đang giao. Việc này nhằm bảo đảm quy trình vận chuyển và tránh rủi ro cho bên bán.

Thứ bảy, một đơn hàng (ORDER) chỉ được chuyển sang trạng thái hoàn tất khi tất cả các đơn hàng con (ORDER UNIT) liên quan đều đã được giao thành công. Nếu còn bất kỳ đơn hàng con nào chưa hoàn tất, toàn bộ đơn sẽ vẫn ở trạng thái chờ xử lý.

Thứ tám, người mua (BUYER) chỉ được phép tạo đánh giá (REVIEW) cho sản phẩm sau khi đơn hàng (ORDER) chứa sản phẩm đó đã được giao thành công. Điều này đảm bảo rằng chỉ những người thực sự đã mua hàng mới có thể để lại đánh giá.

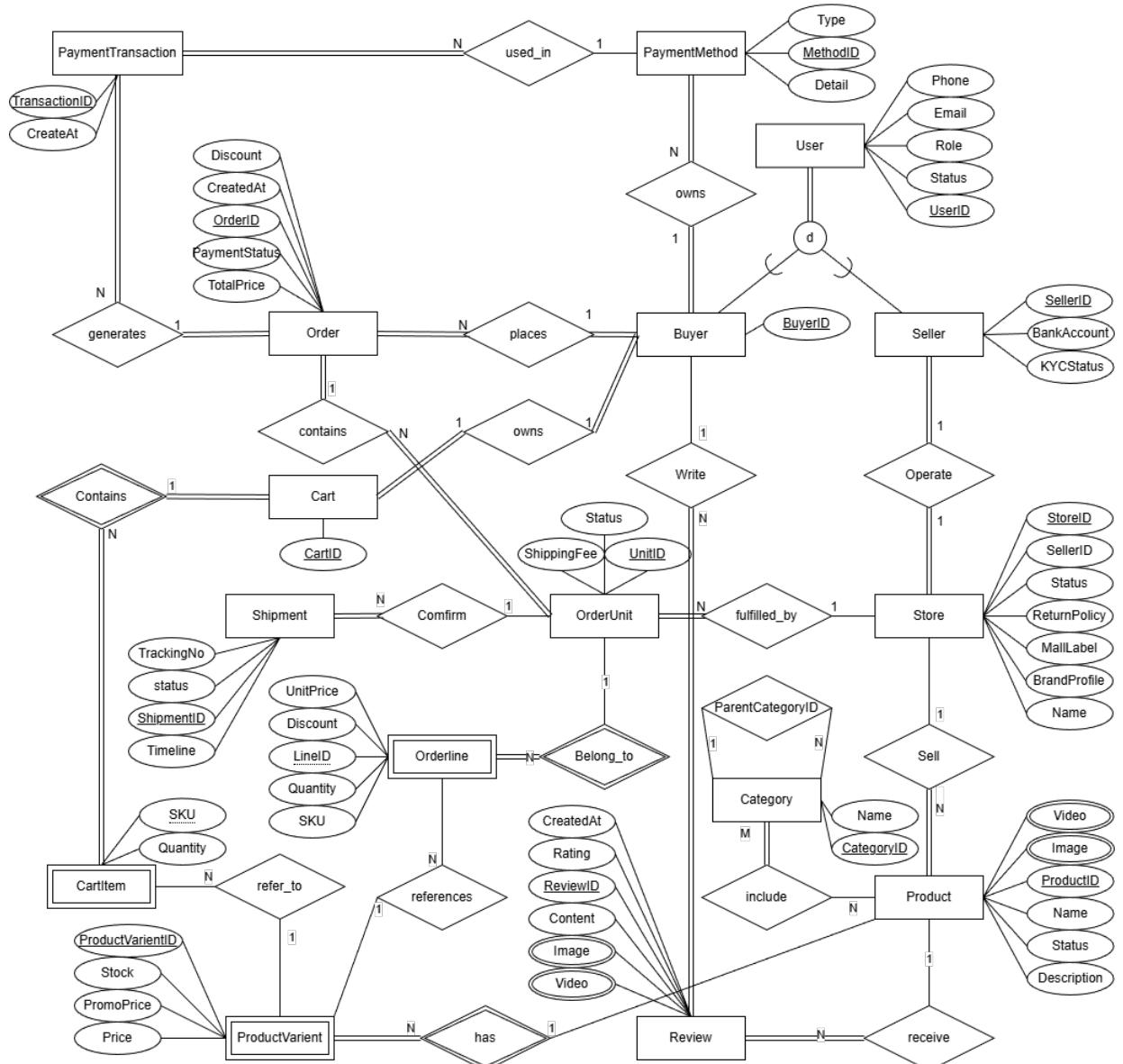
Thứ chín, đánh giá (REVIEW) chỉ được phép chỉnh sửa trong vòng 7 ngày kể từ thời điểm



đăng. Sau thời hạn này, nội dung đánh giá sẽ được khóa để đảm bảo tính minh bạch.

Cuối cùng, điểm đánh giá của đánh giá (REVIEW) chỉ được tính vào uy tín của cửa hàng (STORE) nếu đánh giá đó đến từ đơn hàng thật, tức là không thuộc các đơn bị hủy hoặc hoàn tiền. Nhờ đó, hệ thống duy trì được độ tin cậy và công bằng trong xếp hạng cửa hàng.

2. Mô hình E-ERD



Hình 4: Bản vẽ EERD



3. Ánh xạ lược đồ cơ sở dữ liệu

User	
UserID	Email, Phone, Role, Status
Not null	Email, Phone, Role, Status

Hình 5: User

Buyer	
BuyerID	UserID
FK	UserID to User.UserID
Not null	UserID
	Check total participation of User.UserID in Buyer

Hình 6: Buyer

Seller	
SellerID	UserID, KYCStatus, BankAccount
FK	UserID to User.UserID
Not null	UserID, KYCStatus
	Check total participation of User.UserID in Seller

Hình 7: Seller

Store	
StoreID	SellerID, Brand, BrandProfile, MallLabel, ReturnPolicy, Status
FK	SellerID to Seller.SellerID
Not null	SellerID, Name, Brand, BrandProfile, ReturnPolicy, Status
	Check total participation of Seller.SellerID in Store

Hình 8: Store

Product	
ProductID	StoreID, Status, Title, Description
FK	StoreID to Store.StoreID
Not null	StoreID, Status, Title, Description
	Check total participation of Store.StoreID in Product

Hình 9: Product

Category	
CategoryId	ParentCategoryId, Name
FK	ParentCategoryId to Category.CategoryID
Not null	Name

Hình 10: Category



Order	
OrderID	BuyerID, CreateAt, TotalPrice, Discount, PaymentStatus
FK	BuyerID to Buyer.BuyerID
Not null	BuyerID, CreateAt, TotalPrice, PaymentStatus, CreateAt
	Check total participation of Buyer.BuyerID in Order

Hình 11: Order

PaymentMethod	
MethodID	BuyerID, Type, Detail
FK	BuyerID to Buyer.BuyerID
Not null	BuyerID, Type
	Check total participation of Buyer.BuyerID in PaymentMethod

Hình 12: PaymentMethod

Cart	
CartID	BuyerID
FK	BuyerID to Buyer.BuyerID
Not null	BuyerID

Hình 13: Cart

PaymentTransaction	
TransactionID	OrderID, BuyerID, MethodID, CreatedAt
FK	BuyerID to Buyer.BuyerID
	OrderID to Order.OrderID
	MethodID to PaymentMethod.MethodID
Not null	OrderID, BuyerID, MethodID, CreatedAt
	Check total participation of Order.OrderID in PaymentTransaction

Hình 14: PaymentTransaction

ProductVariant	
ProductVariantID	ProductID, PromotePrice, Stock, Price
FK	ProductID to Product.ProductID
Not null	ProductID, Price, Stock
	Check total participation of product.productID in ProductVariant

Hình 15: ProductVariant



OrderUnit	
UnitID	OrderID, StoreID, ShippingFee, Status
FK	OrderID to Order.OrderID StoreID to Store.StoreID
Not null	OrderID, StoreID, Status
	Check total participation of Store.StoreID in OrderUnit
	Check total participation of Order.OrderID in OrderUnit

Hình 16: OrderUnit

Review	
ReviewID	BuyerID, ProductID, Rating, Content, CreatedAt, Image, Video
FK	BuyerID to Buyer.BuyerID ProductID to Product.ProductID
Not null	BuyerID, ProductID, Rating, Content, CreatedAt
	Check total participation of Product.ProductID in Review

Hình 17: Review

CartItem	
CardID,ProductVarientId	Quantity, SKU
FK	ProductVarientID to ProductVariant.ProductVarientID CardID to Card.CardID
Not null	Quantity,SKU
	Check total participation of ProductVariant.ProductVarientID in CartItem
	Check total participation of Card.CardID in CartItem

Hình 18: CartItem

OrderLine	
LineID	ProductVariantID, UnitID, UnitPrice, Discount, SKU, Quantity
FK	ProductVariantID to ProductVariant.ProductVariantID UnitID to OrderUnit.UnitID
Not null	ProductVariantID, UnitID, Quantity, UnitPrice, SKU
	Check total participation of ProductVariant.ProductVariantID in Orderline
	Check total participation of OrderUnit.UnitID in Orderline

Hình 19: OrderLine

Shipment	
ShipmentID	UnitID, TrackingNo, Status, Timeline
FK	UnitID to OrderUnit.UnitID
Not null	UnitID, TrackingNo, Status, Timeline
	Check total participation of OrderUnit.UnitID in Shipment

Hình 20: Shipment



Product_Category	
	CategoryID,ProductID
FK	CategoryID to Category.CategoryID ProductID to Product.ProductID
	Check total participation of Product.ProductID and Category.CategoryID in Product_Category

Hình 21: ProductCategory

Product_Image	
	ProductID,Image
FK	ProductID to Product.ProductID

Hình 22: ProductImage

Product_Video	
	ProductID,Video
FK	ProductID to Product.ProductID

Hình 23: ProductVideo

Review_Image	
	ReviewID,Image
FK	ReviewID to Review.ReviewID

Hình 24: ReviewImage

Review_Video	
	ReviewID,Video
FK	ReviewID to Review.ReviewID

Hình 25: ReviewVideo



2 Assignment 2

2.1 Tạo bảng và dữ liệu mẫu

2.1.1 Giới thiệu

Trong khuôn khổ Bài tập lớn 2 của học phần Cơ sở dữ liệu, nhóm tiến hành xây dựng và hiện thực hóa cơ sở dữ liệu cho một hệ thống thương mại điện tử mô phỏng theo nền tảng Shopee. Phần 1 của bài tập yêu cầu sinh viên phải hiện thực toàn bộ các bảng dữ liệu đã thiết kế từ BTL1 bằng cú pháp SQL, đồng thời áp dụng đầy đủ các ràng buộc khóa chính, khóa ngoại, ràng buộc dữ liệu và ràng buộc ngữ nghĩa. Bên cạnh đó, đề bài yêu cầu tạo bộ dữ liệu mẫu có ý nghĩa, đảm bảo mỗi bảng có tối thiểu năm bản ghi và dữ liệu phải phù hợp với ngữ cảnh nghiệp vụ.

2.1.2 Mô hình dữ liệu và định hướng thiết kế

CSDL gồm 21 bảng chính, được chia thành 5 nhóm chức năng lớn:

Nhóm Chức Năng	Các Bảng Liên Quan
Quản lý người dùng (User)	User, Buyer, Seller, Payment_Method
Quản lý cửa hàng và danh mục	Store, Category, Product_Category
Quản lý sản phẩm	Product, Product_Variant, Product_Image, Product_Video
Quản lý giỏ hàng và đơn hàng	Cart, CartItem, Order, OrderUnit
Quản lý giao dịch và đánh giá	Order_Line, PAYMENT_TRANSACTION, SHIPMENT, Review, Review_Images, Review_Video

Cấu trúc này mô phỏng gần như đầy đủ hoạt động của hệ thống Shopee thực tế.

Cơ sở dữ liệu được thiết kế hướng tới mô phỏng đầy đủ vòng đời giao dịch của người dùng trong hệ thống thương mại điện tử gồm người bán, người mua, cửa hàng, sản phẩm, giỏ hàng, đơn hàng, vận chuyển và đánh giá sản phẩm. Từ mô hình ERD đã xây dựng ở BTL1, hệ thống được triển khai thành 21 bảng quan hệ có liên kết chặt chẽ với nhau. Cấu trúc được tổ chức theo các nhóm chức năng để đảm bảo tính mạch lạc và dễ mở rộng.

Nhóm “User” đóng vai trò trung tâm, quản lý danh tính và phân loại tài khoản theo hai nhánh Buyer và Seller thông qua mô hình chuyên biệt hóa. Từ đó, hệ thống mở rộng sang các bảng chức năng như phương thức thanh toán, cửa hàng, danh mục sản phẩm, sản phẩm và các thành phần phụ đi kèm. Những bảng bổ trợ như giỏ hàng, dòng đơn hàng, vận đơn, giao dịch thanh toán và đánh giá được thiết kế nhằm mô phỏng chính xác quy trình mua sắm.

Mô hình hướng tới ba tiêu chí cơ bản: tránh dư thừa dữ liệu, đảm bảo toàn vẹn tham chiếu và phản ánh đúng các quy tắc nghiệp vụ (business rules) thông qua các ràng buộc ngữ nghĩa.



2.1.3 Hiệu thực cơ sở dữ liệu và bảng dữ liệu mẫu

Trong phần này, cơ sở dữ liệu được phân tích theo từng bảng. Mỗi bảng được mô tả gồm khóa chính, khóa ngoại liên quan, các ràng buộc dữ liệu và các ràng buộc ngữ nghĩa (trigger) áp dụng trực tiếp lên bảng đó.

```
-- =====
-- TÀNG USER
-- =====

CREATE TABLE `User` (
    UserID INT PRIMARY KEY AUTO_INCREMENT,
    Email VARCHAR(255) NOT NULL UNIQUE,
    Phone VARCHAR(20) NOT NULL UNIQUE,
    `Role` VARCHAR(50) NOT NULL,
    `Status` VARCHAR(50) NOT NULL,
    CONSTRAINT CHK_User_Role CHECK (`Role` IN ('Buyer', 'Seller')),
    CONSTRAINT CHK_User_Status CHECK (`Status` IN ('Active', 'Inactive', 'Banned')),
    CONSTRAINT CHK_User_Phone_Format CHECK (Phone REGEXP '^0[0-9]{9}$'),
    CONSTRAINT CHK_User_Email_Format CHECK (Email REGEXP '^+[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$')
);
```

Bảng User

Bảng User đóng vai trò trung tâm, quản lý danh tính của mọi đối tượng tham gia hệ thống. Khóa chính của bảng là trường UserID tự động tăng. Hai thuộc tính Email và Phone được ràng buộc duy nhất nhằm tránh trùng lặp thông tin tài khoản. Bên cạnh các thuộc tính văn bản cơ bản, bảng còn áp dụng nhiều ràng buộc kiểm tra ví dụ như là Email phải thỏa mãn định dạng email hợp lệ và Phone phải tuân theo cú pháp số điện thoại Việt Nam gồm 10 chữ số bắt đầu bằng số 0. Hai thuộc tính Role và Status được kiểm soát bằng CHECK nhằm đảm bảo chỉ nhận các giá trị được phép.

Ngoài các ràng buộc dữ liệu, bảng User còn liên quan đến ràng buộc ngữ nghĩa. Khi người dùng chỉnh sửa Role thì sẽ ngăn không cho chuyển người dùng sang Buyer hoặc Seller nếu họ đã tồn tại trong bảng chuyên biệt hóa tương ứng.



	User ID	Email	Phone	Role	Status
▶	1	john.seller@example.com	0912345678	Seller	Active
	2	jane.seller@example.com	0923456789	Seller	Active
	3	mike.seller@example.com	0934567890	Seller	Active
	4	lisa.seller@example.com	0945678901	Seller	Inactive
	5	david.seller@example.com	0956789012	Seller	Active
	6	buyer.anna@example.com	0967890123	Buyer	Active
	7	buyer.bob@example.com	0978901234	Buyer	Active
	8	buyer.charlie@example.com	0989012345	Buyer	Banned
	9	buyer.diana@example.com	0990123456	Buyer	Active
	10	buyer.edward@example.com	0901234567	Buyer	Active

Dữ liệu mẫu bảng User

Bảng User được khởi tạo với mười tài khoản, trong đó năm tài khoản thuộc vai trò người bán (Seller) và năm tài khoản thuộc vai trò người mua (Buyer). Các tài khoản này phân bố nhiều trạng thái khác nhau như Active, Inactive hoặc Banned để phục vụ việc kiểm thử luồng nghiệp vụ liên quan đến quyền truy cập và hạn chế thao tác. Dữ liệu email và số điện thoại được tạo theo đúng định dạng hợp lệ và không trùng lặp, phù hợp với yêu cầu UNIQUE và REGEXP. Việc phân tách rõ ràng hai nhóm Buyer và Seller hỗ trợ cho bước chèn dữ liệu vào bảng Buyer và Seller ở phần tiếp theo.

```
-- =====
-- TẢNG BUYER
-- =====
CREATE TABLE Buyer (
    BuyerID INT PRIMARY KEY AUTO_INCREMENT,
    UserID INT NOT NULL UNIQUE,
    CONSTRAINT FK_Buyer_User FOREIGN KEY (UserID) REFERENCES `User` (UserID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

Bảng Buyer

Buyer là bảng chuyên biệt hóa từ User. Thuộc tính BuyerID là khóa chính và UserID đồng thời đóng vai trò khóa ngoại trỏ sang bảng User. Để đảm bảo một người dùng không thể vừa là Buyer



vì là Seller, thì sẽ kiểm tra rằng UserID được thêm mới phải có Role = 'Buyer' trong bảng User và không được tồn tại trong bảng Seller. Bảng Buyer không có thêm thuộc tính mở rộng, toàn bộ đặc điểm của người mua được quản lý thông qua các bảng chức năng như Payment Method, Cart, Order, Review.

```
-- ======  
-- BẢNG SELLER  
-- ======  
CREATE TABLE Seller (  
    SellerID INT PRIMARY KEY AUTO_INCREMENT,  
    UserID INT NOT NULL UNIQUE,  
    KYCStatus VARCHAR(50) NOT NULL,  
    BankAccount VARCHAR(100),  
    CONSTRAINT FK_Seller_User FOREIGN KEY (UserID) REFERENCES `User` (UserID)  
        ON DELETE CASCADE  
        ON UPDATE CASCADE,  
    CONSTRAINT CHK_Seller_KYCStatus CHECK (KYCStatus IN ('Pending', 'Approved', 'Rejected'))  
);
```

Bảng Seller

Tương tự Buyer, bảng Seller là nhánh còn lại của chuyên biệt hóa từ User. SellerID là khóa chính, và UserID là khóa ngoại tham chiếu bảng User với ràng buộc duy nhất. Bảng có thêm hai thuộc tính là KYCStatus và BankAccount. KYCStatus được kiểm soát bằng CHECK, chỉ nhận các giá trị Pending, Approved hoặc Rejected.

Yêu cầu phải bảo đảm UserID được thêm vào phải có Role 'Seller' và đặc biệt không được xuất hiện trong bảng Buyer, từ đó duy trì tính toàn vẹn của phân cấp.

	BuyerID	UserID
▶	1	6
	2	7
	3	8
	4	9
	5	10
*	NULL	NULL

	SellerID	UserID	KYCStatus	BankAccount
▶	1	1	Approved	111-222-333
	2	2	Approved	444-555-666
	3	3	Pending	777-888-999
	4	4	Approved	123-456-789
	5	5	Rejected	NULL

Dữ liệu mẫu của Buyer và Seller

Từ mươi tài khoản trong bảng User, năm tài khoản có Role = 'Buyer' được ánh xạ sang bảng Buyer thông qua UserID. Bảng Seller cũng được khởi tạo dựa trên năm tài khoản còn lại, mỗi bản ghi gắn liền với trạng thái KYC khác nhau như Approved, Pending hoặc Rejected. Việc cung cấp dữ liệu KYC đa dạng cho phép kiểm thử các luồng xử lý liên quan đến xác minh tài khoản người bán. Đồng thời, việc chèn dữ liệu đảm bảo không có tài khoản nào vừa xuất hiện ở Buyer vừa xuất hiện ở Seller, phản ánh đúng ràng buộc ngữ nghĩa đã nêu.



```
-- =====
-- TẢ BÀNG PAYMENT_METHOD
-- =====

› CREATE TABLE PAYMENT_METHOD (
    MethodID INT PRIMARY KEY AUTO_INCREMENT,
    BuyerID INT NOT NULL,
    `Type` VARCHAR(50) NOT NULL,
    Detail VARCHAR(255),
    CONSTRAINT FK_PaymentMethod_Buyer FOREIGN KEY (BuyerID) REFERENCES Buyer(BuyerID)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT CHK_PM_Type CHECK (`Type` IN ('Credit Card', 'E-Wallet', 'COD')),
    CONSTRAINT UQ_PM UNIQUE (BuyerID, `Type`, Detail)
);

```

Bảng Payment Method

Bảng này mô tả các phương thức thanh toán của từng Buyer. MethodID đóng vai trò khóa chính. BuyerID là khóa ngoại tham chiếu bảng Buyer. Để tránh trường hợp một Buyer lưu trùng phương thức thanh toán, nhóm sử dụng ràng buộc UNIQUE trên ba thuộc tính BuyerID, Type và Detail. Kiểu thanh toán được kiểm soát chính xác bằng CHECK.

Bảng này còn chịu một ràng buộc ngữ nghĩa quan trọng: mỗi Buyer phải có ít nhất một phương thức thanh toán. Hai trigger BD_PaymentMethod_Total và BU_PaymentMethod_Total đảm bảo khi xóa hoặc thay đổi BuyerID của một phương thức thanh toán, hệ thống sẽ từ chối nếu đó là phương thức cuối cùng của Buyer.



```
-- =====
-- TẢ BẢNG PAYMENT_METHOD
-- =====
› CREATE TABLE PAYMENT_METHOD (
    MethodID INT PRIMARY KEY AUTO_INCREMENT,
    BuyerID INT NOT NULL,
    `Type` VARCHAR(50) NOT NULL,
    Detail VARCHAR(255),
    CONSTRAINT FK_PaymentMethod_Buyer FOREIGN KEY (BuyerID) REFERENCES Buyer(BuyerID)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT CHK_PM_Type CHECK (`Type` IN ('Credit Card', 'E-Wallet', 'COD')),
    CONSTRAINT UQ_PM UNIQUE (BuyerID, `Type`, Detail)
    );

```

Dữ liệu mẫu của bảng Payment method

Mỗi Buyer được gán từ một đến hai phương thức thanh toán, bao gồm Credit Card, E-Wallet hoặc COD. Dữ liệu được thiết kế sao cho không trùng lặp kết hợp giữa BuyerID – Type – Detail, phù hợp với ràng buộc UNIQUE đang áp dụng trên bảng. Một số Buyer chỉ có một phương thức thanh toán nhằm kiểm thử trigger ngăn chặn việc xóa phương thức cuối cùng. Ngoài ra, các chuỗi mô tả thẻ hoặc ví điện tử được ghi đúng định dạng và liên quan tới số điện thoại hoặc thông tin danh tính của Buyer.



```
-- ======  
-- BẢNG STORE  
-- ======  
  
CREATE TABLE Store (  
    StoreID INT AUTO_INCREMENT PRIMARY KEY,  
    SellerID INT NOT NULL,  
    Name VARCHAR(255) NOT NULL,  
    Brand VARCHAR(255) NOT NULL,  
    BrandProfile TEXT NOT NULL,  
    MallLabel BOOLEAN,  
    ReturnPolicy TEXT NOT NULL,  
    Status VARCHAR(50) NOT NULL CHECK (Status IN ('Active', 'Inactive')),  
    FOREIGN KEY (SellerID) REFERENCES Seller(SellerID)  
);
```

Bảng Store

Store quản lý các cửa hàng của người bán. StoreID là khóa chính và SellerID là khóa ngoại, liên kết mỗi cửa hàng với đúng một người bán. Các thuộc tính Name, Brand, BrandProfile và ReturnPolicy đều bắt buộc nhằm đảm bảo thông tin cửa hàng đầy đủ. Thuộc tính Status được kiểm soát bằng CHECK để nhận giá trị Active hoặc Inactive. Không có trigger gắn trực tiếp cho Store nhưng nó đóng vai trò trung gian trong liên kết đến Product.

StoreID	SellerID	Name	Brand	BrandProfile	MallLabel	ReturnPolicy	Status
1	1	John Electronics Store	TechPro	TechPro là thương hiệu chuyên về thiết bị điện tử và phụ kiện công nghệ cao với hơn 10 năm kinh nghiệm trong ngành. Chúng tôi cam kết mang đến sản phẩm chất lượng với giá cả cạnh tranh.	1	Cho phép trả hàng trong vòng 30 ngày với điều kiện sản phẩm còn nguyên vẹn, không trầy xước.	Active
2	1	John Fashion Hub	StyleUp	StyleUp mang đến những xu hướng thời trang mới nhất cho giới trẻ, luôn cập nhật các mẫu mã hot trend. Chất liệu vải cao cấp, thoáng mát.	0	Cho phép trả hàng trong vòng 7 ngày, áo quần chưa giặt và còn tem mác.	Active
3	2	Jane Beauty Paradise	Glamour	Glamour cung cấp mỹ phẩm và sản phẩm chăm sóc da chất lượng cao, chiết xuất từ thiên nhiên, an toàn cho mọi loại da. Được kiểm nghiệm da liễu.	1	Không cho phép trả hàng với sản phẩm đã mở seal. Được đổi trả nếu sản phẩm lỗi.	Active
4	3	Mike Sports World	ActiveLife	ActiveLife chuyên về dụng cụ thể thao và đồ tập luyện chính hãng. Sản phẩm đạt tiêu chuẩn quốc tế, phù hợp cho cả người mới và vận động viên chuyên nghiệp.	0	Cho phép trả hàng trong vòng 14 ngày nếu sản phẩm bị lỗi kỹ thuật.	Active
5	4	Lisa Home Collection	CozyHome	CozyHome mang đến các sản phẩm trang trí nhà cửa độc đáo, thiết kế hiện đại và tinh tế. Nâng tầm không gian sống của bạn.	1	Cho phép trả hàng trong vòng 30 ngày, sản phẩm phải còn nguyên đồng gói.	Inactive
6	5	David Bookstore	ReadMore	ReadMore cung cấp sách và văn phòng phẩm với giá cả hợp lý. Sách được nhập trực tiếp từ các nhà	0	Không cho phép trả hàng với sách đã sử dụng hoặc có dấu hiệu đọc. Chỉ đổi trả nếu sách bị lỗi in.	Active

Dữ liệu mẫu của bảng Store

Bảng Store được khởi tạo với sáu cửa hàng, mỗi cửa hàng thuộc về một Seller. Các cửa hàng này khác nhau về ngành hàng (điện tử, thời trang, mỹ phẩm, thể thao, gia dụng, sách) và mang các thương hiệu cụ thể như TechPro, StyleUp hoặc Glamour. Thông tin mô tả thương hiệu và chính sách đổi trả được xây dựng chi tiết nhằm mô phỏng nội dung thực tế trong hệ thống thương mại.



diện tử. Trạng thái cửa hàng được phân bổ giữa Active và Inactive nhằm hỗ trợ kiểm thử thao tác tạo và hiển thị sản phẩm.

```
-- =====
-- BẢNG CATEGORY
-- =====

CREATE TABLE Category (
    CategoryID INT AUTO_INCREMENT PRIMARY KEY,
    ParentCategoryID INT,
    Name VARCHAR(255) NOT NULL CHECK (CHAR_LENGTH(Name) >= 3),
    FOREIGN KEY (ParentCategoryID) REFERENCES Category(CategoryID)
);
```

Bảng Category

Category là bảng phân cấp danh mục. CategoryID là khóa chính. Thuộc tính ParentCategoryID tham chiếu đến CategoryID trong cùng bảng, thực hiện quan hệ cây. Tên danh mục được kiểm soát độ dài tối thiểu bằng CHECK (tối thiểu 3 ký tự). Thì ràng buộc sẽ từ chối xóa Category nếu còn tồn tại sản phẩm liên kết trong bảng Product_Category nhằm duy trì tính toàn vẹn.

	CategoryID	ParentCategoryID	Name
▶	1	NULL	Điện tử
	2	NULL	Thời trang
	3	NULL	Gia dụng
	4	1	Điện thoại
	5	2	Phụ kiện thời trang

Dữ liệu mẫu của bảng Category

Bảng Category được tạo với năm danh mục, bao gồm các danh mục cấp cha như Điện tử, Thời trang và Gia dụng, cùng hai danh mục con là Điện thoại và Phụ kiện thời trang. Cấu trúc phân cấp đúng với mô hình nhiều cấp đã xây dựng ở BTL1. Dữ liệu này hỗ trợ cho bảng Product_Category trong việc gán sản phẩm vào danh mục tương ứng.



```
-- =====
-- BẢNG PRODUCT
-- =====
• CREATE TABLE Product (
    ProductID INT AUTO_INCREMENT PRIMARY KEY,
    StoreID INT NOT NULL,
    Status VARCHAR(50) NOT NULL COMMENT 'Trạng thái (ví dụ: Available, Out of Stock)',
    Title VARCHAR(255) NOT NULL COMMENT 'Tiêu đề sản phẩm',
    Description TEXT NOT NULL COMMENT 'Mô tả sản phẩm',
    FOREIGN KEY (StoreID) REFERENCES Store(StoreID)
);
```

Bảng Product

Product quản lý thông tin chung của sản phẩm. ProductID là khóa chính, StoreID là khóa ngoại tham chiếu Store. Trạng thái sản phẩm và mô tả chứa các thông tin hiển thị trên trang sản phẩm. Sản phẩm liên quan đến rất nhiều bảng khác: product_variant, product_image, product_video, Product_Category và OrderLine.

Một ràng buộc ngữ nghĩa quan trọng liên quan đến bảng này là không cho phép xóa một sản phẩm nếu còn liên kết với danh mục trong Product_Category.

	ProductID	StoreID	Status	Title	Description
▶	1	1	Available	Điện thoại Smartphone XYZ Pro	Điện thoại thông minh với camera 48MP, pin 50...
	2	1	Available	Tai nghe Bluetooth không dây chống ồn	Tai nghe Bluetooth công nghệ chống ồn chủ độ...
	3	1	Out of Stock	Laptop Gaming Pro RTX 4060	Laptop chuyên game với card đồ họa RTX 4060 ...
	4	2	Available	Áo thun nam form rộng cotton	Áo thun cotton 100% thoáng mát, form rộng th...
	5	2	Available	Quần jeans nữ ống rộng cao cấp	Quần jeans chất liệu denim co giãn 4 chiều, thiế...
	6	3	Available	Serum dưỡng ẩm ban đêm hoa hồng	Serum chiết xuất từ tinh dầu hoa hồng giúp dướ...
	7	3	Available	Son kem lì không trôi 20 màu	Son kem lì với 20 màu sắc đa dạng từ nude đến ...
	8	4	Available	Dây nhảy thể dục chuyên nghiệp bằng thép	Dây nhảy bằng thép không gỉ, tay cầm bọc cao ...
	9	4	Available	Bóng đá size 5 chính hãng FIFA	Bóng đá da tổng hợp cao cấp, đạt tiêu chuẩn FI...
	10	5	Available	Đèn ngủ LED điều chỉnh độ sáng 7 màu	Đèn ngủ với 7 màu RGB thay đổi linh hoạt, điều ...
	11	6	Available	Sách "Nghệ thuật giao tiếp và thuyết phục"	Cuốn sách về kỹ năng giao tiếp và xây dựng m...
*	12	6	Available	Bút bi cao cấp 0.5mm không lem	Bút bi với mực ra đều, ngồi 0.5mm cho nét chữ t...
	HULL	HULL	HULL	HULL	HULL

Dữ liệu mẫu bảng Product

Bảng Product chứa mười hai sản phẩm đến từ nhiều cửa hàng. Các sản phẩm được lựa chọn từ nhiều ngành hàng nhằm bảo đảm khả năng kiểm thử đa dạng như điện thoại, laptop, áo thun, quần jeans, mỹ phẩm, phụ kiện thể thao, bóng đá, đèn ngủ, sách và bút. Mỗi sản phẩm có đầy đủ tiêu đề, mô tả chi tiết và trạng thái sản phẩm (Available hoặc Out of Stock). Việc chọn sản phẩm đa dạng cho phép kiểm thử đầy đủ các ràng buộc khóa ngoại với Store, Category và các bảng phụ.



```
-- =====
-- 8 BẢNG PRODUCT_VARIANT
-- =====

CREATE TABLE product_variant (
    ProductVariantID INT PRIMARY KEY AUTO_INCREMENT,
    ProductID INT NOT NULL,
    PromotePrice DECIMAL(10, 2) COMMENT 'Giá khuyến mãi',
    Stock INT NOT NULL COMMENT 'Số lượng tồn kho',
    Price DECIMAL(10, 2) NOT NULL COMMENT 'Giá gốc',
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
        ON DELETE CASCADE
);
```

Bảng Product_Variant

Product_variant lưu biến thể của sản phẩm (màu sắc, kích thước...). ProductVariantID là khóa chính và ProductID là khóa ngoại. Các thuộc tính giá gốc, giá khuyến mãi và tồn kho được áp dụng CHECK bảo đảm giá trị hợp lệ. Quan trọng hơn, trigger trg_prevent_last_variant_update đảm bảo rằng một sản phẩm luôn phải có ít nhất một biến thể. Trigger sẽ từ chối việc thay đổi ProductID của biến thể nếu biến thể đó là biến thể cuối cùng của sản phẩm cũ.



	ProductVariantID	ProductID	PromotePrice	Stock	Price
▶	1	1	7990000.00	25	8990000.00
	2	1	9490000.00	15	10990000.00
	3	1	11490000.00	10	12990000.00
	4	2	1290000.00	50	1590000.00
	5	2	1290000.00	45	1590000.00
	6	2	1390000.00	30	1590000.00
	7	3	NULL	5	32900000.00
	8	4	149000.00	100	199000.00
	9	4	149000.00	80	199000.00
	10	4	149000.00	70	199000.00
	11	4	149000.00	60	199000.00
	12	5	389000.00	40	489000.00
	13	5	389000.00	35	489000.00
	14	5	389000.00	30	489000.00
	15	5	389000.00	25	489000.00
	16	6	345000.00	60	425000.00
	17	6	589000.00	40	699000.00
	18	7	125000.00	150	165000.00
	19	7	125000.00	120	165000.00
	20	7	125000.00	100	165000.00
	21	7	125000.00	80	165000.00
	22	8	89000.00	120	119000.00
	23	8	119000.00	80	149000.00
	24	9	279000.00	50	349000.00
	25	9	279000.00	45	349000.00
	26	10	199000.00	70	249000.00
	27	10	199000.00	65	249000.00
	28	11	89000.00	150	119000.00
	29	11	129000.00	100	159000.00
	30	12	27000.00	300	35000.00

Dữ liệu mẫu bảng Product_Variant



Mỗi sản phẩm được gán ít nhất một biến thể, nhiều sản phẩm có từ ba đến bốn biến thể khác nhau. Dữ liệu biến thể bao gồm giá gốc, giá khuyến mãi và số lượng tồn kho, tất cả đều tuân theo ràng buộc CHECK. Các biến thể được mô tả theo thực tế, chẳng hạn các dòng điện thoại có các phiên bản bộ nhớ 128GB, 256GB hoặc 512GB, còn quần áo có nhiều size. Một sản phẩm chỉ có một biến thể được dùng để kiểm thử trigger ngăn không cho di chuyển biến thể cuối cùng sang sản phẩm khác.

```
-- =====
-- 9 BẢNG PRODUCT_IMAGE
-- =====

CREATE TABLE product_image (
    ProductID INT NOT NULL,
    Image VARCHAR(255) NOT NULL COMMENT 'Đường dẫn (URL) đến hình ảnh',
    PRIMARY KEY (ProductID, Image),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
        ON DELETE CASCADE
);

-- =====
-- 10 BẢNG PRODUCT_VIDEO
-- =====

CREATE TABLE product_video (
    ProductID INT NOT NULL,
    Video VARCHAR(255) NOT NULL COMMENT 'Đường dẫn (URL) đến video',
    PRIMARY KEY (ProductID, Video),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
        ON DELETE CASCADE
);
```

Bảng product_image và product_video

Hai bảng này lưu lần lượt URL ảnh và video của sản phẩm, mỗi bảng sử dụng khóa kép làm khóa chính: (ProductID, Image/Video). ProductID đóng vai trò khóa ngoại. Không có ràng buộc ngữ nghĩa bổ sung vì các bảng này chỉ lưu dữ liệu tĩnh.



ProductID	Image
1	https://example.com/images/phone_xyz_pro_b...
1	https://example.com/images/phone_xyz_pro_b...
1	https://example.com/images/phone_xyz_pro_fr...
1	https://example.com/images/phone_xyz_pro_si...
2	https://example.com/images/headphone_black...
2	https://example.com/images/headphone_black...
2	https://example.com/images/headphone_white...
3	https://example.com/images/laptop_gaming_do...
3	https://example.com/images/laptop_gaming_op...
4	https://example.com/images/tshirt_black_back...
4	https://example.com/images/tshirt_black_front...
4	https://example.com/images/tshirt_white_front...
5	https://example.com/images/jeans_black_front...
5	https://example.com/images/jeans_blue_back.jpg
5	https://example.com/images/jeans_blue_front.jpg
6	https://example.com/images/serum_rose_front...
6	https://example.com/images/serum_rose_ingre...
7	https://example.com/images/lipstick_nude_swa...
7	https://example.com/images/lipstick_pink_swat...
7	https://example.com/images/lipstick_red_swatc...
8	https://example.com/images/rimowa_premium_lugg...

Dữ liệu mẫu bảng product_image



	ProductID	Video
▶	1	https://example.com/videos/phone_xyz_pro_c...
	1	https://example.com/videos/phone_xyz_pro_re...
	2	https://example.com/videos/headphone_noise_...
	3	https://example.com/videos/laptop_gaming_pe...
	5	https://example.com/videos/jeans_fitting_guid...
	6	https://example.com/videos/serum_application...
	10	https://example.com/videos/lamp_features_de...
*	NULL	NULL

Dữ liệu mẫu bảng product_video

Hai bảng này lưu thông tin hình ảnh và video của sản phẩm thông qua URL. Mỗi sản phẩm có từ hai đến bốn hình ảnh nhằm mô phỏng việc hiển thị sản phẩm đầy đủ trên giao diện người dùng. Video sản phẩm được thêm vào cho một số sản phẩm chính như điện thoại, tai nghe, laptop hoặc mỹ phẩm. Dữ liệu được chuẩn hóa theo cấu trúc “[https://example.com/...](https://example.com/)”, bảo đảm không vi phạm ràng buộc khóa kép.

```
-- =====
-- 11 BẢNG PRODUCT_CATEGORY
-- =====

CREATE TABLE Product_Category (
    ProductID INT NOT NULL,
    CategoryID INT NOT NULL,
    PRIMARY KEY (ProductID, CategoryID),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID),
    FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID)
);
```

Bảng ProductCategory

Product_Category mô tả quan hệ nhiều-nhiều giữa Product và Category. Khóa chính là cặp ProductID – CategoryID. Có 2 ràng buộc nhằm đảm bảo rằng Product và Category phải tồn tại trước khi gán, đồng thời không được xóa khi còn đang liên kết.



The screenshot shows a database result grid titled "Result Grid". The grid has two columns: "ProductID" and "CategoryID". The data consists of 13 rows, each containing a ProductID and its corresponding CategoryID. The rows are numbered 2 through 12, and the last row is marked with a NULL value for both columns. The grid includes navigation buttons for first, previous, next, and last pages, as well as a "Filter Row" button.

	ProductID	CategoryID
▶	2	1
	3	1
	11	1
	4	2
	5	2
	6	3
	8	3
	9	3
	10	3
	1	4
	7	5
	12	5
*	NULL	NULL

Dữ liệu mẫu bảng ProductCategory

Các sản phẩm được liên kết với danh mục phù hợp. Ví dụ, điện thoại được gán vào danh mục Điện thoại (CategoryID = 4), quần jeans vào Thời trang và son môi vào Phụ kiện thời trang. Việc gán danh mục cho từng sản phẩm phục vụ kiểm thử trigger ngăn xóa Product hoặc Category khi còn liên kết, đồng thời giúp kiểm thử truy vấn lọc sản phẩm theo danh mục.



```
56      -- =====
37 • CREATE TABLE Cart (
38     CartID INT PRIMARY KEY AUTO_INCREMENT,
39     BuyerID INT NOT NULL,
40     FOREIGN KEY (BuyerID) REFERENCES Buyer(BuyerID)
41         ON DELETE CASCADE
42         ON UPDATE CASCADE
43 );
44
45      -- =====
46      -- BẢNG CART_ITEM
47      -- =====
48 • CREATE TABLE CartItem (
49     CartID INT NOT NULL,
50     ProductVariantID INT NOT NULL,
51     SKU VARCHAR(100) NOT NULL,
52     Quantity INT NOT NULL CHECK (Quantity > 0),
53     PRIMARY KEY (CartID, ProductVariantID),
54     FOREIGN KEY (CartID) REFERENCES Cart(CartID)
55         ON DELETE CASCADE,
56     FOREIGN KEY (ProductVariantID) REFERENCES product_variant(ProductVariantID)
57 );
58
```

Bảng Cart và CartItem

Cart lưu giỏ hàng theo BuyerID. CartItem lưu từng mặt hàng trong giỏ. Khóa chính của Cart là CartID, còn CartItem dùng khóa kép (CartID, ProductVariantID). Quantity được kiểm soát bằng CHECK > 0.



	CartID	BuyerID
▶	1	1
	2	2
	3	3
	4	4
	5	5
*	HU LL	HU LL

Dữ liệu mẫu bảng Cart

	CartID	ProductVariantID	SKU	Quantity
▶	1	1	SKU-1	2
	1	2	SKU-2	1
	2	3	SKU-3	3
	2	4	SKU-4	1
	3	5	SKU-5	2
	3	6	SKU-6	1
*	HU LL	HU LL	HU LL	HU LL

Dữ liệu mẫu bảng CartItem

Bảng Cart khởi tạo ba giỏ hàng thuộc ba người mua khác nhau. Các giỏ hàng này chứa các CartItem liên quan đến nhiều biến thể sản phẩm khác nhau. Dữ liệu Quantity luôn lớn hơn 0 theo ràng buộc CHECK và SKU được sinh mô phỏng theo định dạng "SKU-xxx". Bố trí giỏ hàng theo nhiều Buyer cho phép kiểm thử trigger ngăn xóa giỏ hàng khi vẫn còn CartItem.



```
-- =====
-- BẢNG ORDER
-- =====

CREATE TABLE `Order` (
    OrderID INT PRIMARY KEY AUTO_INCREMENT,
    BuyerID INT NOT NULL,
    CreatedAt DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    TotalPrice DECIMAL(10,2) NOT NULL,
    Discount DECIMAL(10,2),
    PaymentStatus VARCHAR(50) NOT NULL CHECK (PaymentStatus IN ('Pending','Paid','Failed','Refunded')),
    FOREIGN KEY (BuyerID) REFERENCES Buyer(BuyerID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

;
```

Bảng Order

Order quản lý đơn hàng của Buyer. OrderID là khóa chính, BuyerID là khóa ngoại. Thuộc tính PaymentStatus được kiểm soát bằng CHECK. Đơn hàng này liên kết đến OrderUnit, OrderLine và PAYMENT_TRANSACTION.

	OrderID	BuyerID	CreatedAt	TotalPrice	Discount	PaymentStatus
▶	1	1	2025-11-21 21:29:01	150000.00	0.00	Pending
	2	2	2025-11-21 21:29:01	250000.00	0.00	Pending
	3	1	2025-11-21 21:29:01	50000.00	0.00	Pending
	4	3	2025-11-21 21:29:01	450000.00	0.00	Pending
✳	5	2	2025-11-21 21:29:01	75000.00	0.00	Pending
	HULL	HULL	HULL	HULL	HULL	HULL

Dữ liệu mẫu bảng Order

Năm đơn hàng được tạo ra thuộc về ba Buyer khác nhau. Các đơn hàng được khởi tạo với tổng giá trị và trạng thái thanh toán “Pending”. Dữ liệu thời gian được chèn bằng NOW() để mô phỏng đơn hàng mới. Số lượng đơn hàng cho mỗi Buyer đủ để kiểm thử trigger không cho phép xóa Buyer khi vẫn còn đơn hàng tồn tại.



```
5 •  -- =====
5 -- BẢNG ORDER_UNIT
7 -- =====
3 Ⓜ CREATE TABLE OrderUnit (
4     UnitID INT PRIMARY KEY AUTO_INCREMENT,
5     OrderID INT NOT NULL,
6     StoreID INT NOT NULL,
7     ShippingFee DECIMAL(10,2),
8     Status VARCHAR(50) NOT NULL CHECK (Status IN ('Processing','Shipping','Completed','Cancelled')),
9     FOREIGN KEY (OrderID) REFERENCES `Order` (OrderID)
10    ON DELETE CASCADE,
11    FOREIGN KEY (StoreID) REFERENCES Store(StoreID)
12 );
13
```

Bảng OrderUnit

OrderUnit là đơn vị chia nhỏ của Order theo từng cửa hàng. UnitID là khóa chính, OrderID và StoreID là khóa ngoại. Trạng thái của Unit được kiểm soát qua CHECK với các trạng thái Processing, Shipping, Completed và Cancelled. Bảng này đóng vai trò quan trọng vì là điểm liên kết giữa đơn hàng, quá trình giao hàng và sản phẩm.

```
-- Tạo bảng ORDER_LINE
Ⓜ CREATE TABLE ORDER_LINE (
    LineID INT PRIMARY KEY AUTO_INCREMENT,
    UnitID INT NOT NULL,
    ProductVariantID INT NOT NULL,
    UnitPrice DECIMAL(10, 2) NOT NULL,
    Discount DECIMAL(5, 2) DEFAULT 0,
    SKU VARCHAR(100),
    Quantity INT NOT NULL,
    FOREIGN KEY (UnitID) REFERENCES OrderUnit(UnitID),
    FOREIGN KEY (ProductVariantID) REFERENCES product_variant(ProductVariantID),
    CHECK (Quantity > 0),
    CHECK (UnitPrice >= 0),
    CHECK (Discount >= 0 AND Discount <= 100)
);
```

Bảng OrderLine



ORDER_LINE mô tả chi tiết từng sản phẩm trong Unit. LineID là khóa chính, UnitID và ProductVariantID là khóa ngoại. Các thuộc tính UnitPrice, Discount và Quantity đều có ràng buộc kiểm tra giá trị nhằm bảo đảm tính chính xác trong tính toán tài chính.

	UnitID	OrderID	StoreID	ShippingFee	Status
▶	1	1	1	30000.00	Completed
	2	2	2	25000.00	Shipping
	3	3	1	20000.00	Shipping
	4	4	3	35000.00	Completed
●	5	5	6	15000.00	Cancelled
	NULL	NULL	NULL	NULL	NULL

Dữ liệu mẫu bảng OrderUnit

	LineID	UnitID	ProductVariantID	UnitPrice	Discount	SKU	Quantity
▶	1	1	1	150000.00	10.00	SKU001	1
	2	2	2	250000.00	0.00	SKU002	1
	3	3	3	50000.00	5.00	SKU003	2
	4	4	4	450000.00	15.00	SKU004	1
●	5	5	1	150000.00	0.00	SKU001	1
	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Dữ liệu mẫu bảng OrderLine

Mỗi đơn hàng được chia thành các OrderUnit theo từng cửa hàng khác nhau. Các Unit có trạng thái Processing, Shipping, Completed hoặc Cancelled đúng với luồng xử lý thực tế. Các mặt hàng trong từng Unit được mô tả qua bảng ORDER_LINE, bao gồm giá đơn vị, discount và số lượng, đảm bảo tính đầy đủ trong kiểm thử tính tổng tiền đơn hàng.



```
-- Tạo bảng PAYMENT_TRANSACTION
▶ CREATE TABLE PAYMENT_TRANSACTION (
    TransactionID INT PRIMARY KEY AUTO_INCREMENT,
    OrderID INT NOT NULL,
    BuyerID INT NOT NULL,
    MethodID INT NOT NULL,
    CreatedAt DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (OrderID) REFERENCES `Order` (OrderID),
    FOREIGN KEY (BuyerID) REFERENCES Buyer(BuyerID),
    FOREIGN KEY (MethodID) REFERENCES PAYMENT_METHOD(MethodID)
);
```

Bảng PaymentTransaction

Bảng PAYMENT_TRANSACTION ghi nhận giao dịch thanh toán của từng đơn hàng. TransactionID là khóa chính, còn OrderID, BuyerID và MethodID lần lượt là khóa ngoại tham chiếu đúng các bảng Order, Buyer và PAYMENT_METHOD. Trường CreatedAt tự động lưu thời điểm giao dịch. Mặc dù không có trigger riêng, ba khóa ngoại bảo đảm giao dịch chỉ được tạo khi đơn hàng, người mua và phương thức thanh toán đều hợp lệ.

	TransactionID	OrderID	BuyerID	MethodID	CreatedAt
▶	1	1	1	1	2025-11-21 21:29:02
	2	2	2	2	2025-11-21 21:29:02
	3	3	1	1	2025-11-21 21:29:02
	4	4	3	3	2025-11-21 21:29:02
	5	5	2	1	2025-11-21 21:29:02
*	HULL	HULL	HULL	HULL	HULL

Dữ liệu mẫu bảng PaymentTransaction

Bảng PAYMENT_TRANSACTION đóng vai trò ghi nhận thông tin thanh toán cho từng đơn hàng. Mỗi bản ghi xác định phương thức thanh toán mà Buyer đã sử dụng khi tạo đơn, thời điểm thanh toán và liên kết trực tiếp với OrderID, BuyerID và MethodID. Trong bộ dữ liệu mẫu, năm giao dịch thanh toán được tạo ra để bao phủ đầy đủ các tình huống nghiệp vụ khác nhau. Các giao dịch này được gán cho năm đơn hàng tương ứng và sử dụng nhiều phương thức thanh toán khác nhau như thẻ tín dụng, ví điện tử hoặc COD, từ đó hỗ trợ kiểm thử đầy đủ luồng xử



lý thanh toán.

Việc chèn dữ liệu mẫu đảm bảo ba điều kiện quan trọng của Payment Transaction: thứ nhất, BuyerID phải đúng với Buyer đã tạo đơn hàng tương ứng trong bảng Order; thứ hai, phương thức thanh toán (MethodID) phải thuộc về Buyer đó theo đúng ràng buộc khóa ngoại; và thứ ba, mỗi giao dịch đều được tạo với thời điểm thanh toán mặc định CURRENT_TIMESTAMP, phản ánh quá trình thanh toán diễn ra ngay khi đơn hàng được khởi tạo. Nhờ vậy, dữ liệu mẫu giúp hệ thống có thể kiểm thử đồng thời tính toàn vẹn tham chiếu và các quy tắc nghiệp vụ liên quan đến xử lý thanh toán.

```
-- Tạo bảng SHIPMENT
CREATE TABLE SHIPMENT (
    ShipmentID INT PRIMARY KEY AUTO_INCREMENT,
    UnitID INT NOT NULL,
    TrackingNo VARCHAR(100) UNIQUE,
    Status VARCHAR(50) NOT NULL,
    Timeline TEXT,
    CreatedAt DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    EstimatedDelivery DATE,
    FOREIGN KEY (UnitID) REFERENCES OrderUnit(UnitID),
    CHECK (Status IN ('Preparing', 'On the way', 'Delivered', 'Failed', 'Cancelled'))
);
```

Bảng Shipment

SHIPMENT lưu thông tin vận chuyển. ShipmentID là khóa chính, UnitID là khóa ngoại. Trạng thái giao hàng được kiểm soát bởi CHECK. Ràng buộc nhằm bảo đảm chỉ được tạo vận đơn cho Unit đang ở trạng thái Processing. Event SQL được dùng để tự động đánh dấu các lô hàng “On the way” quá 15 ngày thành “Failed”.

	ShipmentID	UnitID	TrackingNo	Status	Timeline	CreatedAt	EstimatedDelivery
▶	1	1	TRACK001	Delivered	{"created": "2025-10-26", "delivered": "2025-10-26"}	2025-11-21 21:29:01	2025-10-29
▶	2	2	TRACK002	On the way	{"created": "2025-10-27"}	2025-11-21 21:29:01	2025-10-30
▶	3	3	TRACK003	On the way	{"created": "2025-10-28"}	2025-11-21 21:29:01	2025-11-01
▶	4	4	TRACK004	Delivered	{"created": "2025-10-20", "delivered": "2025-10-20"}	2025-11-21 21:29:01	2025-10-24
▶	5	5	TRACK005	Cancelled	{"created": "2025-10-29"}	2025-11-21 21:29:01	2025-11-02
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Dữ liệu mẫu bảng Shipment

SHIPMENT được khởi tạo nhằm mô phỏng nhiều tình huống giao hàng như Delivered, On the way hoặc Cancelled. Mỗi UnitID có một vận đơn tương ứng với thời điểm dự kiến giao hàng. Timeline được lưu ở dạng JSON mô phỏng quá trình vận chuyển. Dữ liệu này phục vụ kiểm thử trigger không cho tạo vận đơn nếu Unit chưa ở trạng thái Processing, và kiểm thử event tự động đánh dấu các đơn bị treo quá 15 ngày.



```
-- Tạo bảng REVIEW
CREATE TABLE REVIEW (
    ReviewID INT PRIMARY KEY AUTO_INCREMENT,
    BuyerID INT NOT NULL,
    ProductID INT NOT NULL,
    Rating INT NOT NULL,
    Content TEXT,
    CreatedAt DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (BuyerID) REFERENCES Buyer(BuyerID),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID),
    CHECK (Rating >= 1 AND Rating <= 5)
);

-- Tạo bảng REVIEW_IMAGES & REVIEW_VIDEO
CREATE TABLE REVIEW_IMAGES (
    ReviewImageID INT PRIMARY KEY AUTO_INCREMENT,
    ReviewID INT NOT NULL,
    ImageURL VARCHAR(255) NOT NULL,
    FOREIGN KEY (ReviewID) REFERENCES REVIEW(ReviewID)
);

CREATE TABLE REVIEW_VIDEO (
    ReviewVideoID INT PRIMARY KEY AUTO_INCREMENT,
    ReviewID INT NOT NULL,
    VideoURL VARCHAR(255) NOT NULL,
    FOREIGN KEY (ReviewID) REFERENCES REVIEW(ReviewID)
);
```

Bảng REVIEW, REVIEW_IMAGES và REVIEW_VIDEO

REVIEW chứa đánh giá của Buyer về Product. ReviewID là khóa chính, BuyerID và ProductID



là khóa ngoại. Rating phải nằm trong khoảng từ 1 đến 5. Ngoài ra sẽ có ràng buộc nhằm xác minh rằng người mua chỉ được đánh giá sản phẩm sau khi đơn hàng hoàn tất, phản ánh đúng logic của hệ thống thương mại. Ràng buộc ngữ nghĩa ở đây cấm chỉnh sửa đánh giá sau 7 ngày.

Hai bảng REVIEW_IMAGES và REVIEW_VIDEO lưu URL ảnh và video của đánh giá, mỗi bảng dùng khóa chính tự tăng và khóa ngoại tham chiếu Review.

	ReviewID	BuyerID	ProductID	Rating	Content	CreatedAt
▶	1	1	1	5	Sản phẩm tuyệt vời, chất lượng tốt.	2025-11-01 10:00:00
*	NULL	NULL	NULL	NULL	NULL	NULL

Dữ liệu mẫu bảng REVIEW

	ReviewImageID	ReviewID	ImageURL
▶	1	1	http://example.com/image1.jpg
	2	1	http://example.com/image2.jpg
	3	1	http://example.com/image3.jpg
	4	1	http://example.com/image4.jpg
	5	1	http://example.com/image5.jpg
*	NULL	NULL	NULL

Dữ liệu mẫu bảng REVIEW_IMAGES

	ReviewVideoID	ReviewID	VideoURL
▶	1	1	http://example.com/video1.mp4
	2	1	http://example.com/video2.mp4
	3	1	http://example.com/video3.mp4
	4	1	http://example.com/video4.mp4
	5	1	http://example.com/video5.mp4
*	NULL	NULL	NULL

Dữ liệu mẫu bảng REVIEW_VIDEO

Một đánh giá hợp lệ được tạo cho Buyer đã nhận hàng thành công. Bảng REVIEW_IMAGES và REVIEW_VIDEO lưu lại tối đa năm hình ảnh và video minh họa cho bài đánh giá. Dữ liệu này nhằm kiểm chứng trigger chỉ cho phép đánh giá sau khi đơn hàng hoàn tất, đồng thời ngăn chặn chỉnh sửa bài đánh giá sau bảy ngày.

2.1.4 Kiểm thử tính đúng đắn của hệ thống

Sau khi hoàn tất quá trình tạo bảng và chèn dữ liệu, hệ thống được kiểm tra bằng các truy vấn thống kê nhằm đảm bảo mỗi bảng đều có dữ liệu và các ràng buộc hoạt động đúng. Nhiều tình



huống kiểm thử lỗi được mô phỏng, chẳng hạn như thử xóa phương thức thanh toán cuối cùng, di chuyển biến thể duy nhất của sản phẩm hoặc tạo đánh giá khi chưa nhận hàng. Kết quả cho thấy hệ thống từ chối các thao tác vi phạm đúng như kỳ vọng, chứng minh tính chính xác của các ràng buộc được xây dựng.

Ngoài ra, dữ liệu trong các bảng chức năng như Order, OrderUnit và Shipment được đối chiếu để đảm bảo sự nhất quán giữa trạng thái giao hàng và trạng thái đơn hàng. Việc sử dụng Event SQL để tự động cập nhật trạng thái vận đơn sau 15 ngày cũng giúp mô phỏng tốt quy trình xử lý đơn thực tế.

2.1.5 Kết luận

Phần 1 của bài tập lớn đã hoàn thành đầy đủ và chính xác yêu cầu của đề bài. Toàn bộ cơ sở dữ liệu đã được hiện thực hóa dựa trên mô hình đã thiết kế ở BTL1, bao gồm hệ thống bảng phong phú với đầy đủ ràng buộc khóa chính, khóa ngoại, ràng buộc dữ liệu và ràng buộc ngữ nghĩa. Việc kết hợp các cơ chế CHECK, UNIQUE, REGEXP cùng với Trigger giúp đảm bảo rằng dữ liệu luôn nhất quán và phản ánh đúng logic nghiệp vụ của một nền tảng thương mại điện tử.

Bộ dữ liệu mẫu được xây dựng công phu, đa dạng, đáp ứng yêu cầu tối thiểu về số dòng và hỗ trợ hữu hiệu cho công tác thử nghiệm. Hệ thống sau khi kiểm thử cho thấy vận hành ổn định, các ràng buộc ngữ nghĩa hoạt động chính xác, sẵn sàng cho các phần tiếp theo của bài tập lớn như xây dựng truy vấn và tầng ứng dụng.

2.2 Viết các trigger, thủ tục, hàm

2.2.1 Thủ tục Insert – Update – Delete và các kiểm tra ràng buộc dữ liệu

2.2.1.1 Bảng: User a. Thủ tục Insert



```
1 USE BTLL2_SHOPEE;
2 SET GLOBAL log_bin_trust_function_creators = 1;
3 DELIMITER $$ 
4 DROP PROCEDURE IF EXISTS sp_User_Insert$$
5 CREATE PROCEDURE sp_User_Insert(
6     IN p_Email VARCHAR(255),
7     IN p_Phone VARCHAR(20),
8     IN p_Role VARCHAR(50),
9     IN p_Status VARCHAR(50)
10    )
11 BEGIN
12     IF NOT (p_Email REGEXP '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$') THEN
13         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Email không đúng định dạng.';
14     ELSEIF NOT (p_Phone REGEXP '^0[0-9]{9}$') THEN
15         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Số điện thoại phải có 10 chữ số và bắt đầu bằng 0.';
16     ELSEIF EXISTS (SELECT 1 FROM `User` WHERE Email = p_Email) THEN
17         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Email này đã được sử dụng.';
18     ELSEIF EXISTS (SELECT 1 FROM `User` WHERE Phone = p_Phone) THEN
19         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Số điện thoại này đã được sử dụng.';
20     ELSEIF p_Role NOT IN ('Buyer', 'Seller') THEN
21         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Vai trò (Role) phải là Buyer hoặc Seller.';
22     ELSEIF p_Status NOT IN ('Active', 'Inactive', 'Banned') THEN
23         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Trạng thái (Status) phải là Active, Inactive, hoặc Banned.';
24     END IF;
25     INSERT INTO `User` (Email, Phone, `Role`, `Status`)
26     VALUES (p_Email, p_Phone, p_Role, p_Status);
27 END$$
```

Mô tả: Thủ tục sp_User_Insert được xây dựng nhằm mục đích khởi tạo một tài khoản người dùng chung. Đây là thực thể cha, làm cơ sở để chuyên biệt hóa thành người mua (buyer) hoặc người bán (seller).

Các bước kiểm tra ràng buộc:

- Kiểm tra định dạng: Đảm bảo email tuân thủ đúng định dạng (example@domain.com) và SDT tuân thủ đúng định dạng (10 số, bắt đầu bằng 0).
- Kiểm tra tính duy nhất : Đảm bảo email không được trùng lặp với bất kỳ tài khoản nào khác đã có trong hệ thống và SDT không được trùng lặp.
- Kiểm tra miền giá trị : Vai trò phải là 'Buyer' hoặc 'Seller' và trạng thái (status) phải thuộc danh sách hợp lệ ('Active', 'Inactive', 'Banned').

Khi sử dụng email và SDT hợp lệ thì sẽ insert vào bảng như sau:



The screenshot shows the MySQL Workbench interface. The SQL tab contains the following code:

```
1 • USE BTL2_SHOPEE;
2 • CALL sp_User_Insert('user.moi@example.com', '0988776655', 'Buyer', 'Active');
3 • SELECT * FROM User WHERE Email = 'user.moi@example.com';
```

The Results tab displays a single row in the Result Grid:

UserID	Email	Phone	Role	Status
12	user.moi@example.com	0988776655	Buyer	Active
*	NULL	NULL	NULL	NULL

Khi sử dụng email đã tồn tại trong hệ thống sẽ báo ra lỗi:

The screenshot shows the MySQL Workbench interface. The SQL tab contains the following code:

```
1 • USE BTL2_SHOPEE;
2 • CALL sp_User_Insert('john.seller@example.com', '0999888777', 'Buyer', 'Active');
```

The Output tab shows the error message:

Action Output	Message
1 21:55:37 CALL sp_User_Insert('john.seller@example.com', '0999888777', 'Buyer', 'Active')	Error Code: 1644. Lỗi: Email này đã được sử dụng.

b. Thủ tục Update



```
CREATE PROCEDURE sp_User_Update(
    IN p UserID INT,
    IN p_Email VARCHAR(255),
    IN p Phone VARCHAR(20),
    IN p_Status VARCHAR(50)
)
BEGIN
    IF NOT EXISTS (SELECT 1 FROM `User` WHERE UserID = p.UserID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: UserID không tồn tại.';
    ELSEIF NOT (p.Email REGEXP '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$') THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Email không đúng định dạng.';
    ELSEIF NOT (p.Phone REGEXP '^0[0-9]{9}$') THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Số điện thoại phải có 10 chữ số và bắt đầu bằng 0.';
    ELSEIF EXISTS (SELECT 1 FROM `User` WHERE Email = p.Email AND UserID != p.UserID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Email này đã được sử dụng bởi tài khoản khác.';
    ELSEIF EXISTS (SELECT 1 FROM `User` WHERE Phone = p.Phone AND UserID != p.UserID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Số điện thoại này đã được sử dụng bởi tài khoản khác.';
    ELSEIF p_Status NOT IN ('Active', 'Inactive', 'Banned') THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Trạng thái (Status) phải là Active, Inactive, hoặc Banned.';
    END IF;
    -- 3. Cập nhật
    UPDATE `User`
    SET
        Email = p.Email,
        Phone = p.Phone,
        `Status` = p_Status
    WHERE UserID = p.UserID;
END$$
```

Mô tả: Thủ tục sp_User_Update cho phép cập nhật các thông tin cơ bản của một tài khoản User đã tồn tại, như thông tin liên lạc hoặc trạng thái.

Thủ tục đảm bảo tính nhất quán của dữ liệu thông qua các bước kiểm tra:

- Kiểm tra tham chiếu: Xác thực UserID (Mã người dùng) cần sửa phải thực sự tồn tại trong hệ thống.
- Kiểm tra tính duy nhất (Uniqueness): Khi cập nhật Email hoặc Phone, thủ tục kiểm tra để đảm bảo giá trị mới không bị trùng với một người dùng khác.
- Kiểm tra định dạng và miền giá trị: Tương tự như khi thêm mới, Email, Phone và Status phải tuân thủ đúng định dạng và miền giá trị cho phép.



```
1 • USE BTL2_SHOPEE;
2 • CALL sp_User_Update(1, 'john.seller@example.com', '0912345678', 'Inactive');
3 • SELECT * FROM User WHERE UserID = 1;
```

	UserID	Email	Phone	Role	Status
▶	1	john.seller@example.com	0912345678	Seller	Inactive
*	NULL	NULL	NULL	NULL	NULL

c. Thủ tục Delete

```
• DROP PROCEDURE IF EXISTS sp_User_Delete$$
• CREATE PROCEDURE sp_User_Delete(
    IN pUserID INT
)
BEGIN
    -- 1. Khi nào KHÔNG được xóa
    IF NOT EXISTS (SELECT 1 FROM `User` WHERE UserID = pUserID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: UserID không tồn tại.';
    END IF;
    -- 2. Mục đích: Xóa tài khoản người dùng khỏi hệ thống.
    DELETE FROM `User` WHERE UserID = pUserID;
END$$
```

Mô tả: thủ tục sp_User_Delete chịu trách nhiệm xóa hoàn toàn một tài khoản người dùng ra khỏi hệ thống khi UserID đưa ra yêu cầu.

Mục đích của việc xóa User:

- Xóa User trong hệ thống Shopee dùng để loại bỏ một tài khoản không còn hoạt động, tài khoản bị khóa vĩnh viễn, hoặc tài khoản tạo nhầm.
- Khi User bị xóa, toàn bộ thông tin phụ thuộc như Buyer, Seller cũng bị xóa theo cơ chế ON DELETE CASCADE, giúp hệ thống giữ sạch dữ liệu thừa.



The screenshot shows a MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it, a SQL editor window contains the following code:

```
1 • USE BTM2_SHOPEE;
2 • CALL sp_User_Delete(10);
3 • SELECT * FROM User WHERE UserID = 10;
```

Below the SQL editor is a "Result Grid" pane. It has a header row with columns: UserID, Email, Phone, Role, and Status. A single row of data is shown, with all values set to NULL.

Kiểm tra ràng buộc: luôn được phép xóa khi UserID tồn tại và chỉ chặn khi UserID cung cấp không tồn tại. Khi User bị xóa, các bản ghi con liên quan cũng sẽ tự động bị xóa theo. Hơn nữa, việc xóa Buyer sẽ kích hoạt cascade để xóa PAYMENT_METHOD, và trigger sẽ đảm bảo việc này thành công ngay cả khi Buyer chỉ có một phương thức thanh toán.

2.2.1.2 Bảng: Buyer a. Thủ tục Insert

The screenshot shows a MySQL Workbench interface with a large block of SQL code for a stored procedure named sp_Buyer_Insert. The code includes logic to check if the provided UserID exists in the User table, if it has a role other than 'Buyer', or if it exists in the Seller table. If any of these conditions are true, it signals an error message. Otherwise, it inserts the UserID into the Buyer table.

```
76 • CREATE PROCEDURE sp_Buyer_Insert(
77     IN pUserID INT
78 )
79 BEGIN
80     DECLARE v_Role VARCHAR(50);
81     IF NOT EXISTS (SELECT 1 FROM `User` WHERE UserID = pUserID) THEN
82         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: UserID không tồn tại.';
83     END IF;
84     SELECT `Role` INTO v_Role FROM `User` WHERE UserID = pUserID;
85     IF v_Role != 'Buyer' THEN
86         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: UserID này không có vai trò (Role) là Buyer.';
87     END IF;
88     IF EXISTS (SELECT 1 FROM Seller WHERE UserID = pUserID) THEN
89         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: UserID này đã tồn tại trong bảng Seller. Không thể vừa là Buyer vừa là Seller.';
90     END IF;
91     IF EXISTS (SELECT 1 FROM Buyer WHERE UserID = pUserID) THEN
92         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: UserID này đã là Buyer rồi.';
93     END IF;
94     INSERT INTO Buyer (UserID) VALUES (pUserID);
95 END$$
96
```

Mô tả: Thủ tục sp_Buyer_Insert dùng để "chuyên biệt hóa" một tài khoản User đã tồn tại thành một Buyer (Người mua). Thủ tục này không tạo User mới mà chỉ liên kết với một User có sẵn.

Các bước kiểm tra ràng buộc:

- Kiểm tra tham chiếu: Dảm bảo UserID phải tồn tại trong bảng User.
- Kiểm tra vai trò (Role): UserID được cung cấp phải có role là 'Buyer' trong bảng User.
- Kiểm tra chuyên biệt hóa (Specialization): Dảm bảo UserID này chưa tồn tại trong bảng Seller. Một User không thể vừa là Buyer vừa là Seller.



- Kiểm tra tính duy nhất (Uniqueness): Đảm bảo UserID này chưa được thêm vào bảng Buyer trước đó.

b. Thủ tục Delete

```
-- XÓA BUYER (sp_Buyer_Delete)
DROP PROCEDURE IF EXISTS sp_Buyer_Delete$$
CREATE PROCEDURE sp_Buyer_Delete(
    IN p_BuyerID INT
)
BEGIN
    IF NOT EXISTS (SELECT 1 FROM Buyer WHERE BuyerID = p_BuyerID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: BuyerID không tồn tại.';
    END IF;
    DELETE FROM Buyer WHERE BuyerID = p_BuyerID;
END$$
```

Mô tả: thủ tục sp_Buyer_Delete dùng để xóa hồ sơ Buyer của một người dùng mà không xóa tài khoản User gốc.

Mục đích xóa Buyer: khi người dùng chuyển vai trò, ngừng sử dụng tài khoản mua hàng hoặc cần xóa dữ liệu lập trình kiểm thử, việc xóa Buyer giúp đảm bảo hệ thống không tồn tại dữ liệu dư thừa không còn được sử dụng.

Kiểm tra ràng buộc: Hệ thống không cho phép xóa Buyer nếu BuyerID không tồn tại trong bảng Buyer. Đây là điều kiện duy nhất chặn việc xóa hồ sơ Buyer, nhằm đảm bảo việc xóa chỉ áp dụng cho các bản ghi hợp lệ. Khi Buyer bị xóa, tất cả các PAYMENT_METHOD liên kết với BuyerID đó sẽ tự động bị xóa theo.

2.2.1.3 Bảng: Seller a. Thủ tục Insert



```
1 DROP PROCEDURE IF EXISTS sp_Seller_Insert$$
2
3 CREATE PROCEDURE sp_Seller_Insert(
4     IN p.UserID INT,
5     IN p.KYCStatus VARCHAR(50),
6     IN p.BankAccount VARCHAR(100)
7 )
8 BEGIN
9     DECLARE v_Role VARCHAR(50);
10
11     IF NOT EXISTS (SELECT 1 FROM `User` WHERE UserID = p.UserID) THEN
12         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: UserID không tồn tại.';
13     END IF;
14
15     SELECT `Role` INTO v_Role FROM `User` WHERE UserID = p.UserID;
16
17     IF v_Role != 'Seller' THEN
18         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: UserID này không có vai trò (Role) là Seller.';
19     END IF;
20
21     IF EXISTS (SELECT 1 FROM Buyer WHERE UserID = p.UserID) THEN
22         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: UserID này đã tồn tại trong bảng Buyer. Không thể vừa là Buyer vừa là Seller.';
23     END IF;
24
25     IF EXISTS (SELECT 1 FROM Seller WHERE UserID = p.UserID) THEN
26         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: UserID này đã là Seller rồi.';
27     END IF;
28
29     IF p.KYCStatus NOT IN ('Pending', 'Approved', 'Rejected') THEN
30         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: KYCstatus phải là Pending, Approved, hoặc Rejected.';
31     END IF;
32
33     INSERT INTO Seller (UserID, KYCStatus, BankAccount)
34     VALUES (p.UserID, p.KYCStatus, p.BankAccount);
35
36 END$$
```

Mô tả: Thủ tục sp_Seller_Insert dùng để "chuyên biệt hóa" một tài khoản User thành Seller và ghi nhận thông tin định danh. Tương tự như Buyer, thủ tục này thực thi các kiểm tra chuyên biệt hóa nghiêm ngặt.

Các bước kiểm tra ràng buộc:

- Kiểm tra tham chiếu: Dảm bảo UserID phải tồn tại trong bảng User.
- Kiểm tra vai trò (Role): UserID được cung cấp phải có Role là 'Seller' trong bảng User.
- Kiểm tra chuyên biệt hóa (Specialization): Dảm bảo UserID này chưa tồn tại trong bảng Buyer.
- Kiểm tra tính duy nhất (Uniqueness): Dảm bảo UserID này chưa được thêm vào bảng Seller trước đó.
- Kiểm tra miền giá trị (Domain): Trạng thái định danh phải là 'Pending', 'Approved', hoặc 'Rejected'.

Thủ tục Update



```
CREATE PROCEDURE sp_Seller_Update(
    IN p_SellerID INT,
    IN p_KYCStatus VARCHAR(50),
    IN p_BankAccount VARCHAR(100)
)
BEGIN
    IF NOT EXISTS (SELECT 1 FROM Seller WHERE SellerID = p_SellerID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: SellerID không tồn tại.'; -- 2. Validate KYCStatus
    ELSEIF p_KYCStatus NOT IN ('Pending', 'Approved', 'Rejected') THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: KYCStatus phải là Pending, Approved, hoặc Rejected.';
    END IF;
    UPDATE Seller
    SET
        KYCStatus = p_KYCStatus,
        BankAccount = p_BankAccount
    WHERE SellerID = p_SellerID;
END$$
```

Mô tả: Thủ tục sp_Seller_Update cho phép cập nhật thông tin của Seller, chủ yếu phục vụ cho nghiệp vụ duyệt định danh của Admin hoặc khi người bán thay đổi tài khoản ngân hàng. Kiểm tra tham chiếu: Xác thực SellerID cần sửa phải tồn tại.

Kiểm tra miền giá trị: Đảm bảo KYCStatus mới vẫn nằm trong miền giá trị cho phép ('Pending', 'Approved', 'Rejected').

c. Thủ tục Delete

```
DROP PROCEDURE IF EXISTS sp_Seller_Delete$$
CREATE PROCEDURE sp_Seller_Delete(
    IN p_SellerID INT
)
BEGIN
    IF NOT EXISTS (SELECT 1 FROM Seller WHERE SellerID = p_SellerID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: SellerID không tồn tại.';
    END IF;
    IF EXISTS (SELECT 1 FROM Store WHERE SellerID = p_SellerID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Không thể xóa Seller vì vẫn còn cửa hàng (Store) liên kết.';
    END IF;
    DELETE FROM Seller WHERE SellerID = p_SellerID;
END$$
```

Mô tả: Thủ tục sp_Seller_Delete dùng để xóa hồ sơ Seller của một người dùng mà không xóa tài khoản User gốc.

Mục đích xóa Seller:

- Xóa Seller khi người bán ngừng kinh doanh, yêu cầu đóng tài khoản hoặc khi cần dọn dữ liệu test.
- Điều này giúp đảm bảo hệ thống không giữ lại các cửa hàng hoặc sản phẩm không còn hoạt động.



Kiểm tra ràng buộc: thủ tục sẽ bị chặn nếu Seller này vẫn còn sở hữu Store và nếu SellerID không tồn tại trong hệ thống. Người dùng phải xóa Store trước (vì Store là bảng con của Seller theo CSDL gốc của bạn). Chỉ được xóa khi SellerID tồn tại và không còn Store nào liên kết.

2.2.1.4 Bảng Product a. Thủ tục Insert

```
CREATE PROCEDURE sp_Insert_Product(
    IN p_StoreID INT,
    IN p_Title VARCHAR(255),
    IN p_Description TEXT,
    IN p_Status VARCHAR(50)
)

BEGIN
    -- Validate 1: Kiểm tra Cửa hàng (Store) có tồn tại không
    IF NOT EXISTS (SELECT 1 FROM Store WHERE StoreID = p_StoreID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Mã cửa hàng (StoreID) không tồn tại.';
    END IF;

    -- Validate 2: Tiêu đề sản phẩm không được để trống hoặc quá ngắn
    IF p_Title IS NULL OR CHAR_LENGTH(TRIM(p_Title)) < 5 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Tên sản phẩm phải có ít nhất 5 ký tự.';
    END IF;

    -- Validate 3: Trạng thái sản phẩm phải hợp lệ
    IF p_Status NOT IN ('Available', 'Out of Stock', 'Inactive', 'Banned') THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Trạng thái sản phẩm không hợp lệ.';
    END IF;

    -- Thực hiện INSERT
    INSERT INTO Product (StoreID, Title, Description, Status)
    VALUES (p_StoreID, p_Title, p_Description, p_Status);

    -- Trả về thông báo và ID vừa tạo (để tiện thêm biến thể sau này)
    SELECT CONCAT('Thêm sản phẩm thành công! ID mới là: ', LAST_INSERT_ID()) AS Message;
END$$
```

Mô tả: Thủ tục sp_Insert_Product được xây dựng nhằm mục đích khởi tạo một hồ sơ sản phẩm mới (sản phẩm cha) vào hệ thống, làm cơ sở để thêm các biến thể sau này.

Các bước kiểm tra ràng buộc:

- Kiểm tra tham chiếu: Đảm bảo StoreID (Mã cửa hàng) phải tồn tại trong bảng Store, xác định rõ sản phẩm này thuộc về người bán nào.
- Kiểm tra tính hợp lệ của dữ liệu: Tên sản phẩm (Title) không được phép để trống và bắt buộc phải có độ dài tối thiểu 5 ký tự để đảm bảo tên gọi rõ nghĩa, tránh tình trạng spam dữ liệu rác.
- Kiểm tra miền giá trị: Trạng thái sản phẩm (Status) bắt buộc phải thuộc danh sách quy định của hệ thống (gồm: 'Available', 'Out of Stock', 'Inactive', 'Banned'), ngăn chặn việc nhập sai



chính tả hoặc nhập trạng thái không hỗ trợ
Kiểm tra:

```
1 • CALL sp_Insert_Product(1, 'Laptop Dell XPS Test', 'Laptop mỏng nhẹ', 'Available');  
2 • SELECT * FROM btl2_shopee.product;
```

ProductID	StoreID	Status	Title	Description
3	1	Out of Stock	Laptop Gaming Pro RTX 4060	Laptop chuyên game với card đồ họa RTX 4060 ...
4	2	Available	Áo thun nam form rộng cotton	Áo thun cotton 100% thoáng mát, form rộng th...
5	2	Available	Quần jeans nữ ống rộng cao cấp	Quần jeans chất liệu denim co giãn 4 chiều, thiế...
6	3	Available	Serum dưỡng ẩm ban đêm hoa hồng	Serum chiết xuất từ tinh dầu hoa hồng giúp dướ...
7	3	Available	Son kem lì không trôi 20 màu	Son kem lì với 20 màu sắc đa dạng từ nude đến ...
8	4	Available	Dây nhảy thể dục chuyên nghiệp bằng thép	Dây nhảy bằng thép không gỉ, tay cầm bọc cao ...
9	4	Available	Bóng đá size 5 chính hãng FIFA	Bóng đá da tổng hợp cao cấp, đạt tiêu chuẩn FI...
10	5	Available	Đèn ngủ LED điều chỉnh độ sáng 7 màu	Đèn ngủ với 7 màu RGB thay đổi linh hoạt, điều ...
11	6	Available	Sách "Nghệ thuật giao tiếp và thuyết phục"	Cuốn sách về kỹ năng giao tiếp và xây dựng m...
12	6	Available	Bút bi cao cấp 0.5mm không lem	Bút bi với mực ra đều, ngòi 0.5mm cho nét chữ t...
13	1	Available	Laptop Dell XPS Test	Laptop mỏng nhẹ

Nhận xét: Sau khi thêm thành công đã tự động cập nhật vô bảng
Thủ tục Update



```
CREATE PROCEDURE sp_Update_Product(
    IN p_ProductID INT,
    IN p_Title VARCHAR(255),
    IN p_Description TEXT,
    IN p_Status VARCHAR(50)
)
BEGIN
    -- Validate 1: Kiểm tra ProductID tồn tại
    IF NOT EXISTS (SELECT 1 FROM Product WHERE ProductID = p_ProductID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Mã sản phẩm không tồn tại.';
    END IF;

    -- Validate 2: Tên sản phẩm
    IF p_Title IS NULL OR CHAR_LENGTH(TRIM(p_Title)) < 5 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Tên sản phẩm phải có ít nhất 5 ký tự.';
    END IF;

    -- Validate 3: Trạng thái
    IF p_Status NOT IN ('Available', 'Out of Stock', 'Inactive', 'Banned') THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Trạng thái sản phẩm không hợp lệ.';
    END IF;

    -- Thực hiện UPDATE
    UPDATE Product
    SET Title = p_Title,
        Description = p_Description,
        Status = p_Status
    WHERE ProductID = p_ProductID;

    SELECT 'Cập nhật thông tin sản phẩm thành công!' AS Message;
END$$
```

Mô tả: Thủ tục sp_Update_Product cho phép cập nhật các thông tin chung của sản phẩm như tên gọi, mô tả chi tiết và trạng thái kinh doanh. Thủ tục đảm bảo tính nhất quán của dữ liệu thông qua các chốt chặn kiểm tra:

- Kiểm tra tham chiếu: Xác thực mã sản phẩm (ProductID) cần sửa phải thực sự tồn tại trong hệ thống.
- Kiểm tra logic cập nhật: Tương tự như khi thêm mới, tên sản phẩm sau khi sửa cũng phải đảm bảo độ dài tối thiểu và trạng thái mới phải nằm trong miền giá trị cho phép.
- Bảo toàn quan hệ sở hữu: Thủ tục chỉ cho phép sửa thông tin mô tả, không cho phép thay đổi StoreID, đảm bảo một sản phẩm khi đã tạo ra thì gắn liền với cửa hàng đó, không thể chuyển nhượng trái phép sang cửa hàng khác.

Kiểm tra:



```
1 • CALL sp_Update_Product(13, 'Laptop Dell XPS 2025', 'Laptop mỏng nhẹ', 'Out of Stock');
2 • SELECT * FROM bt12_shopee.product;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

ProductID	StoreID	Status	Title	Description
8	4	Available	Dây nhảy thể dục chuyên nghiệp bằng thép	Dây nhảy bằng thép không gỉ, tay cầm bọc cao ...
9	4	Available	Bóng đá size 5 chính hãng FIFA	Bóng đá da tổng hợp cao cấp, đạt tiêu chuẩn FI...
10	5	Available	Đèn ngủ LED điều chỉnh độ sáng 7 màu	Đèn ngủ với 7 màu RGB thay đổi linh hoạt, điều ...
11	6	Available	Sách "Nghệ thuật giao tiếp và thuyết phục"	Cuốn sách về kỹ năng giao tiếp và xây dựng m...
12	6	Available	Bút bi cao cấp 0.5mm không lem	Bút bi với mực ra đều, ngòi 0.5mm cho nét chữ t...
13	1	Out of Stock	Laptop Dell XPS 2025	Laptop mỏng nhẹ
*	HULL	HULL	HULL	HULL

product_variant 17 product 23 × Apply

Output

Action Output
Time Action Message
231 23:01:42 CALL sp_Update_Product(13, 'Laptop Dell XPS 2025', 'Laptop mỏng nhẹ', 'Out of Stock') 1 row(s) returned
232 23:01:45 SELECT * FROM bt12_shopee.product LIMIT 0, 100 13 row(s) returned

Nhận xét: Cập nhật lại trạng thái sản phẩm thành công

Thủ tục Delete



```
CREATE PROCEDURE sp_Delete_Product(IN p_ProductID INT)
BEGIN
    DECLARE v_HasOrder INT;

    -- Validate 1: Kiểm tra ProductID tồn tại
    IF NOT EXISTS (SELECT 1 FROM Product WHERE ProductID = p_ProductID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Mã sản phẩm không tồn tại.';
    END IF;

    -- Validate 2: Kiểm tra xem sản phẩm này đã từng BÁN được chưa?
    -- (Logic: Tìm tất cả biến thể của sản phẩm này, xem có cái nào nằm trong Order_Line không)
    SELECT COUNT(*) INTO v_HasOrder
    FROM ORDER_LINE ol
    JOIN product_variant pv ON ol.ProductVariantID = pv.ProductVariantID
    WHERE pv.ProductID = p_ProductID;

    IF v_HasOrder > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Lỗi: Sản phẩm này đã có lịch sử giao dịch (đã bán), KHÔNG ĐƯỢC XÓA. Hãy chuyển trạng t
    END IF;

    -- Lưu ý: Do bảng product_variant có ON DELETE CASCADE (thường thiết kế vậy),
    -- nên khi xóa Product thì Variant tự mất. Nhưng ta phải chặn nếu đã có đơn hàng.

    -- Thực hiện DELETE
    DELETE FROM Product WHERE ProductID = p_ProductID;

    SELECT 'Xóa sản phẩm thành công!' AS Message;
END$$
```

Mô tả: Thủ tục sp_Delete_Product chịu trách nhiệm loại bỏ hoàn toàn một sản phẩm khỏi danh sách bán hàng. Tuy nhiên, để bảo vệ tính toàn vẹn của dữ liệu lịch sử và báo cáo tài chính, thủ tục áp dụng quy tắc nghiệp vụ nghiêm ngặt sau:

- Kiểm tra ràng buộc lịch sử giao dịch: Trước khi xóa, hệ thống thực hiện truy vấn quét toàn bộ các đơn hàng (ORDER_LINE) liên quan đến tất cả các biến thể của sản phẩm này. Nếu phát hiện sản phẩm đã từng được bán (tồn tại trong lịch sử đơn hàng), thủ tục sẽ CHẶN XÓA và thông báo lỗi.

- Mục đích: Cơ chế này giúp ngăn chặn việc xóa nhầm các sản phẩm đã có doanh thu, gây sai lệch dữ liệu đơn hàng cũ của khách hàng. (Trong trường hợp này, hệ thống khuyến nghị người dùng nên chuyển trạng thái sang Inactive thay vì xóa vĩnh viễn).

- Cơ chế tự động (Cascade): Nếu sản phẩm chưa từng phát sinh giao dịch và thỏa mãn điều kiện xóa, hệ thống sẽ xóa sản phẩm cha, đồng thời các dữ liệu con liên quan (Biến thể, Hình ảnh, Video) cũng sẽ được tự động dọn dẹp theo.

Mục đích xóa Product:

- Xóa Product dùng cho trường hợp người bán ngừng cung cấp mặt hàng, sản phẩm không hợp pháp hoặc dữ liệu test được tạo nhầm.



-Xóa Product giúp hệ thống giữ sạch danh mục và giảm rủi ro hiển thị nhầm sản phẩm đã ngừng kinh doanh.

Kiểm tra:

```
1 • CALL sp_Delete_Product(13);
2 • SELECT * FROM bt12_shopee.product;
```

ProductID	StoreID	Status	Title	Description
7	3	Available	Son kem lì không trôi 20 màu	Son kem lì với 20 màu sắc đa dạng từ nude đến ...
8	4	Available	Dây nhảy thể dục chuyên nghiệp bằng thép	Dây nhảy bằng thép không gỉ, tay cầm bọc cao ...
9	4	Available	Bóng đá size 5 chính hãng FIFA	Bóng đá da tổng hợp cao cấp, đạt tiêu chuẩn FI...
10	5	Available	Đèn ngủ LED điều chỉnh độ sáng 7 màu	Đèn ngủ với 7 màu RGB thay đổi linh hoạt, điều ...
11	6	Available	Sách "Nghệ thuật giao tiếp và thuyết phục"	Cuốn sách về kỹ năng giao tiếp và xây dựng m...
12	6	Available	Bút bi cao cấp 0.5mm không lem	Bút bi với mực ra đều, ngòi 0.5mm cho nét chữ t...

product 28 x

Output

#	Time	Action	Message
236	23:04:12	CALL sp_Delete_Product(13)	1 row(s) returned
237	23:04:16	SELECT * FROM bt12_shopee.product LIMIT 0, 100	12 row(s) returned

Nhận xét: Xóa sản phẩm Product ID 13 thành công.

2.2.1.5 Order a. Insert



```
-- Insert Order --  
  
DELIMITER $$  
CREATE PROCEDURE InsertOrder(  
    IN p_BuyerID INT,  
    IN p_TotalPrice DECIMAL(10,2),  
    IN p_Discount DECIMAL(10,2),  
    IN p_PaymentStatus VARCHAR(50)  
)  
BEGIN  
    DECLARE v_exists INT;  
  
    -- Validate 1: Buyer phải tồn tại  
    SELECT COUNT(*) INTO v_exists FROM Buyer WHERE BuyerID = p_BuyerID;  
    IF v_exists = 0 THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: BuyerID không tồn tại.';  
    END IF;  
  
    -- Validate 2: PaymentStatus hợp lệ  
    IF p_PaymentStatus NOT IN ('Pending','Paid','Failed','Refunded') THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: PaymentStatus không hợp lệ.';  
    END IF;  
  
    -- Validate 3: Giá hợp lệ  
    IF p_TotalPrice < 0 THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: TotalPrice không thể âm.';  
    END IF;  
  
    -- Insert  
    INSERT INTO `Order` (BuyerID, TotalPrice, Discount, PaymentStatus)  
    VALUES (p_BuyerID, p_TotalPrice, p_Discount, p_PaymentStatus);  
END$$  
DELIMITER ;
```

Mô tả: Thủ tục InsertOrder được thiết kế để tạo mới một đơn hàng (Order) cho Buyer khi người dùng tiến hành đặt hàng.

Các bước kiểm tra ràng buộc:

- Kiểm tra tham chiếu: BuyerID phải tồn tại → đảm bảo đơn hàng thuộc về người mua hợp lệ.
- Kiểm tra giá trị: TotalPrice phải ≥ 0 , Discount có thể null nhưng nếu có phải hợp lệ
- Kiểm tra miền giá trị: PaymentStatus phải nằm trong danh sách quy định ('Pending', 'Paid', 'Failed', 'Refunded').

Kiểm tra:



```
31 • CALL InsertOrderUnit(1, 1, 30000, 'Processing');  
32 • SELECT * FROM OrderUnit WHERE OrderID = 1;  
33  
34  
35
```

100% ⏪ | 1:31 |

Action Output ⏪

	Time	Action
✓ 30	16:25:28	CALL InsertOrderUnit(1, 1, 30000, 'Processing')

b. Thủ tục Update



```
-- Update Order --
DELIMITER $$

CREATE PROCEDURE UpdateOrderStatus(
    IN p_OrderID INT,
    IN p_PaymentStatus VARCHAR(50)
)

BEGIN
    DECLARE v_exists INT;

    -- Validate 1: Order phải tồn tại
    SELECT COUNT(*) INTO v_exists FROM `Order` WHERE OrderID = p_OrderID;
    IF v_exists = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Lỗi: OrderID không tồn tại.';
    END IF;

    -- Validate 2: PaymentStatus hợp lệ
    IF p_PaymentStatus NOT IN ('Pending','Paid','Failed','Refunded') THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Lỗi: PaymentStatus không hợp lệ.';
    END IF;

    -- Update
    UPDATE `Order`
    SET PaymentStatus = p_PaymentStatus
    WHERE OrderID = p_OrderID;
END$$
DELIMITER ;
```

Mô tả: Thủ tục UpdateOrderStatus dùng để thay đổi trạng thái thanh toán của một đơn hàng. Các kiểm tra được thực hiện: OrderID phải có trong hệ thống và PaymentStatus phải nằm trong danh sách hợp lệ.

Kiểm tra:

The screenshot shows the MySQL Workbench interface with the following details:

- Code area:

```
34 • CALL UpdateOrderUnitStatus(1, 'Shipping');
35 • SELECT * FROM OrderUnit WHERE UnitID = 1;
36
37
```
- Output area:

Action	Time	Action	Response
38	21:10:38	CALL UpdateOrderUnitStatus(1, 'Shipping')	1 row(s) affected

Thủ tục Delete Mô Tả: Thủ tục DeleteOrder được dùng để xóa đơn hàng khỏi hệ thống. Tuy nhiên, việc xóa chỉ được phép khi đơn hàng không còn phụ thuộc.

-Kiểm tra tồn tại: OrderID phải có trong bảng Order

-Kiểm tra ràng buộc toàn vẹn: Không được xóa Order nếu tồn tại OrderUnit liên quan → tránh xóa đơn hàng trong khi đơn vị xử lý vẫn hoạt động.

Mục đích xóa:

-Đơn tạo nhầm, đơn test, hoặc đơn bị hủy trước khi phân chia sang các cửa hàng. -Giúp hệ thống giảm dữ liệu rác và giữ tính sạch trong các bảng vận hành.

Kiểm tra:

```
37 • CALL DeleteOrder(3);
38 • SELECT * FROM `Order`;
39
100% ◄ 1:38
```

Result Grid Filter Rows: Search Edit: Export/Import:

OrderID	BuyerID	CreatedAt	TotalPrice	Discount	PaymentStatus
1	1	2025-11-17 15:14:40	150000.00	0.00	Pending
2	2	2025-11-17 15:14:40	250000.00	0.00	Pending
3	1	2025-11-17 15:14:40	500000.00	0.00	Pending
4	3	2025-11-17 15:14:40	450000.00	0.00	Pending
5	2	2025-11-17 15:14:40	75000.00	0.00	Pending
NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL

2.2.1.6 OrderLine Thủ tục Insert



```
-- ORDERLINE ADD --
DELIMITER $$

CREATE PROCEDURE `sp_AddOrderLine`(
    IN p_UnitID INT,
    IN p_ProductVariantID INT,
    IN p_Quantity INT,
    IN p_UnitPrice DECIMAL(10, 2),
    IN p_Discount DECIMAL(5, 2)
)
BEGIN
    -- Khai báo các biến cần thiết
    DECLARE v_order_unit_status VARCHAR(50);
    DECLARE v_stock INT;
    DECLARE v_OrderID INT;
    DECLARE v_error_message VARCHAR(255);

    -- Validate 1: Kiểm tra sự tồn tại và trạng thái của OrderUnit
    SELECT ou.Status, ou.OrderID INTO v_order_unit_status, v_OrderID
    FROM OrderUnit ou WHERE ou.UnitID = p_UnitID;

    IF v_order_unit_status IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Đơn hàng con (UnitID) không tồn tại.';
    END IF;

    IF v_order_unit_status != 'Processing' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Chỉ có thể thêm sản phẩm khi đơn hàng con ở trạng thái "Processing".';
    END IF;

    -- Validate 2: Kiểm tra sự tồn tại và số lượng tồn kho của sản phẩm
    SELECT Stock INTO v_stock FROM product_variant WHERE ProductVariantID = p_ProductVariantID;
    IF v_stock IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Biến thể sản phẩm (ProductVariantID) không tồn tại.';
    END IF;

    IF p_Quantity <= 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Số lượng sản phẩm phải lớn hơn 0.';
    END IF;

```



```
IF v_stock < p_Quantity THEN
    SET v_error_message = CONCAT('Lỗi: Không đủ hàng. Chỉ còn ', v_stock, ' sản phẩm trong kho.');
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_error_message;
END IF;

-- Thực hiện INSERT với định dạng SKU đã được cập nhật
INSERT INTO ORDER_LINE(UnitID, ProductVariantID, Quantity, UnitPrice, Discount, SKU)
VALUES (p_UnitID, p_ProductVariantID, p_Quantity, p_UnitPrice, p_Discount, CONCAT('SKU', LPAD(p_ProductVariantID, 3, '0')));

-- Tác vụ phụ: Cập nhật tồn kho và tổng tiền
UPDATE product_variant SET Stock = Stock - p_Quantity WHERE ProductVariantID = p_ProductVariantID;
-- CALL sp_RecalculateOrderTotal(v_OrderID); -- Bạn cần tạo thủ tục này

-- Trả về thông báo
SELECT 'Thêm dòng đơn hàng thành công!' AS Message;
END$$

DELIMITER ;
```

Mô tả: Thủ tục sp_AddOrderLine được xây dựng nhằm mục đích thêm một sản phẩm cụ thể vào một đơn hàng con (OrderUnit) đang trong quá trình xử lý.

Các bước kiểm tra ràng buộc:

- Kiểm tra tham chiếu và trạng thái: Xác thực UnitID phải tồn tại trong bảng OrderUnit và đơn hàng con đó phải đang ở trạng thái "Processing". Điều này ngăn chặn việc thay đổi nội dung đơn hàng khi nó đã được chuyển sang giai đoạn vận chuyển.
- Kiểm tra tồn kho: Đảm bảo ProductVariantID tồn tại và số lượng sản phẩm (Quantity) thêm vào không vượt quá số lượng tồn kho (Stock) hiện có, tránh tình trạng bán hàng vượt quá khả năng cung cấp.
- Cập nhật tự động: Sau khi INSERT thành công, thủ tục sẽ tự động thực hiện hai tác vụ quan trọng: (1) Trừ số lượng tồn kho tương ứng trong bảng product_variant; và (2) Gọi một thủ tục khác để tính toán lại tổng giá trị (TotalPrice) của đơn hàng cha (Order).

Kiểm tra:

```
1 • CALL sp_AddOrderLine(1, 4, 1, 199000, 0);
2 • SELECT * FROM BTL2_shopee.order_line WHERE UnitID = 1;
```

	LineID	UnitID	ProductVariantID	UnitPrice	Discount	SKU	Quantity
▶	1	1	1	150000.00	10.00	SKU001	1
	6	1	4	199000.00	0.00	SKU004	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Nhận xét: Sau khi thực thi, một dòng mới đã được thêm thành công vào ORDER_LINE cho UnitID = 1. Tồn kho của ProductVariantID = 4 và TotalPrice của OrderID tương ứng đã được tự động cập nhật.

Thủ tục Update



```
-- ORDERLINE UPDATE --
DELIMITER $$

CREATE PROCEDURE `sp_UpdateOrderLine`(
    IN p_LineID INT,
    IN p_NewQuantity INT
)
BEGIN
    DECLARE v_order_unit_status VARCHAR(50);
    DECLARE v_stock INT;
    DECLARE v_ProductVariantID INT;
    DECLARE v_OldQuantity INT;
    DECLARE v_quantity_diff INT;
    DECLARE v_OrderID INT;
    DECLARE v_error_message VARCHAR(255);

    -- Validate 1: Kiểm tra sự tồn tại của LineID và trạng thái của OrderUnit
    SELECT ol.Quantity, ol.ProductVariantID, ou.Status, ou.OrderID
    INTO v_OldQuantity, v_ProductVariantID, v_order_unit_status, v_OrderID
    FROM ORDER_LINE ol JOIN OrderUnit ou ON ol.UnitID = ou.UnitID WHERE ol.LineID = p_LineID;

    IF v_OldQuantity IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Đòng đơn hàng (LineID) không tồn tại.';
    END IF;

    IF v_order_unit_status != 'Processing' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Chỉ có thể sửa khi đơn hàng con ở trạng thái "Processing".';
    END IF;

    -- Validate 2: Kiểm tra số lượng mới và tồn kho nếu tăng số lượng
    IF p_NewQuantity <= 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Số lượng mới phải lớn hơn 0.';
    END IF;

    SET v_quantity_diff = p_NewQuantity - v_OldQuantity;
    IF v_quantity_diff > 0 THEN
        SELECT Stock INTO v_stock FROM product_variant WHERE ProductVariantID = v_ProductVariantID;
        IF v_stock < v_quantity_diff THEN
            SET v_error_message = CONCAT('Lỗi: Không đủ hàng để tăng số lượng. Chỉ còn ', v_stock, ' sản phẩm.');
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_error_message;
        END IF;
    END IF;

    -- Thực hiện UPDATE
    UPDATE ORDER_LINE SET Quantity = p_NewQuantity WHERE LineID = p_LineID;

    -- Tác vụ phụ: Cập nhật lại tồn kho và tổng tiền
    UPDATE product_variant SET Stock = Stock - v_quantity_diff WHERE ProductVariantID = v_ProductVariantID;
    CALL sp_RecalculateOrderTotal(v_OrderID);

    SELECT 'Cập nhật số lượng thành công!' AS Message;
END$$
DELIMITER ;
```



```
6      -- ORDERLINE UPDATE --
7 •  CALL sp_UpdateOrderLine(6, 3);
8 •  SELECT * FROM btl2_shopee.order_line WHERE LineID = 6;
```

	Message
▶	Cập nhật số lượng thành công!

Mô tả: Thủ tục sp_UpdateOrderLine cho phép điều chỉnh số lượng của một sản phẩm trong một dòng đơn hàng đã tồn tại. Để đảm bảo tính nhất quán và chính xác của dữ liệu, thủ tục thực thi các chốt chặn kiểm tra nghiêm ngặt trước khi cập nhật:

-Kiểm tra tham chiếu và trạng thái: Xác thực mã dòng đơn hàng (LineID) phải tồn tại trong hệ thống. Quan trọng hơn, thủ tục kiểm tra và chỉ cho phép sửa đổi khi đơn hàng con (OrderUnit) liên quan đang ở trạng thái "Processing", ngăn chặn mọi thay đổi sau khi đơn hàng đã được xác nhận xử lý.

-Kiểm tra logic cập nhật: Thủ tục yêu cầu số lượng mới phải là số dương. Trong trường hợp tăng số lượng, hệ thống sẽ tính toán lượng chênh lệch và kiểm tra với số lượng tồn kho (Stock) hiện tại của sản phẩm để đảm bảo không xảy ra tình trạng bán hàng vượt kho.

-Cập nhật tự động: Sau khi lệnh UPDATE được thực thi thành công, thủ tục sẽ tự động điều chỉnh lại số lượng tồn kho trong bảng product_variant và gọi một thủ tục phụ (sp_RecalculateOrderTotal) để tính toán lại tổng giá trị của đơn hàng cha, đảm bảo dữ liệu luôn được đồng bộ

Kiểm tra:



```
-- ORDERLINE DELETE --
DELIMITER $$

CREATE PROCEDURE `sp_DeleteOrderLine`(
    IN p_LineID INT
)
BEGIN
    DECLARE v_order_unit_status VARCHAR(50);
    DECLARE v_ProductVariantID INT;
    DECLARE v_Quantity INT;
    DECLARE v_UnitID INT;
    DECLARE v_OrderID INT;
    DECLARE v_remaining_lines INT;

    -- Validate 1: Kiểm tra sự tồn tại của LineID và trạng thái của OrderUnit
    SELECT ol.Quantity, ol.ProductVariantID, ou.Status, ou.UnitID, ou.OrderID
    INTO v_Quantity, v_ProductVariantID, v_order_unit_status, v_UnitID, v_OrderID
    FROM ORDER_LINE ol JOIN OrderUnit ou ON ol.UnitID = ou.UnitID WHERE ol.LineID = p_LineID;

    IF v_Quantity IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Dòng đơn hàng (LineID) không tồn tại.';
    END IF;

    IF v_order_unit_status != 'Processing' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Chỉ có thể xóa khi đơn hàng con ở trạng thái "Processing".';
    END IF;

    -- Thực hiện DELETE
    DELETE FROM ORDER_LINE WHERE LineID = p_LineID;

    -- Tác vụ phụ 1: Hoàn trả tồn kho
    UPDATE product_variant SET Stock = Stock + v_Quantity WHERE ProductVariantID = v_ProductVariantID;

    -- Tác vụ phụ 2: Kiểm tra nếu OrderUnit rỗng thì xóa luôn OrderUnit
    SELECT COUNT(*) INTO v_remaining_lines FROM ORDER_LINE WHERE UnitID = v_UnitID;
    IF v_remaining_lines = 0 THEN
        DELETE FROM OrderUnit WHERE UnitID = v_UnitID;
    END IF;

    -- Tác vụ phụ 3: Cập nhật lại tổng tiền
    CALL sp_RecalculateOrderTotal(v_OrderID);

    SELECT 'Xóa dòng đơn hàng thành công!' AS Message;
END$$
DELIMITER ;
```

Nhân xét: Lệnh gọi thủ tục thực thi thành công. Cột Quantity của LineID = 6 đã được cập nhật chính xác từ 1 lên 3. Số lượng tồn kho của sản phẩm tương ứng và tổng giá trị đơn hàng cha cũng đã được hệ thống tự động điều chỉnh lại.

Thủ tục Delete



```
10      -- ORDERLINE DELETE --
11 •  CALL sp_DeleteOrderLine(6);
12 •  SELECT * FROM btl2_shopee.order_line WHERE LineID = 6;
```

	LineID	UnitID	ProductVariantID	UnitPrice	Discount	SKU	Quantity
▶	6	1	4	199000.00	0.00	SKU004	3
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
232 20:25:45 CREATE PROCEDURE 'sp_DeleteOrderLine' ( IN p_LineID INT ) BEGIN   DECLARE v_o... 0 row(s) affected
233 20:26:32 SELECT * FROM btl2_shopee.order_line WHERE LineID = 6 LIMIT 0, 1000           1 row(s) returned
```

Mô tả: Thủ tục sp_DeleteOrderLine thực hiện việc loại bỏ hoàn toàn một dòng sản phẩm (Order Line) khỏi một đơn hàng con (OrderUnit). Việc xóa chỉ được phép khi đơn hàng con vẫn còn trong giai đoạn xử lý (Processing). Thủ tục đồng thời thực hiện các tác vụ tự động như hoàn trả tồn kho, xóa OrderUnit nếu rỗng và cập nhật lại tổng tiền của đơn hàng gốc.

-Mục đích xóa:

+Mục đích của thao tác xóa OrderLine là cho phép người bán hoặc quản trị viên chỉnh sửa các sai sót trong giai đoạn xử lý đơn hàng, chẳng hạn như thêm nhầm sản phẩm, nhập sai số lượng, hoặc khách hàng thay đổi lựa chọn trước khi đơn hàng được xác nhận.

+Cơ chế này đảm bảo rằng việc chỉnh sửa chỉ diễn ra khi đơn hàng chưa chuyển sang các trạng thái cao hơn (Shipping, Completed), nhằm bảo vệ lịch sử giao dịch.

+Các thay đổi sau khi đơn đã xử lý phải tuân theo các nghiệp vụ riêng như trả hàng hoặc hoàn tiền, thay vì xóa dòng đơn hàng trực tiếp.

Kiểm tra: -Chỉ được xóa nếu OrderUnit đang ở trạng thái Processing. Nếu không, thủ tục chặn xóa để bảo toàn tính toàn vẹn dữ liệu. Khi xóa thành công, số lượng sản phẩm trong OrderLine được cộng trả lại vào tồn kho của biến thể sản phẩm tương ứng.

-Nếu OrderUnit không còn dòng sản phẩm nào sau khi xóa, thủ tục tự động xóa OrderUnit để tránh tồn tại dữ liệu rỗng.

-Sau khi xóa, hệ thống tính toán và cập nhật lại TotalPrice của đơn hàng gốc nhằm bảo đảm thông tin luôn chính xác.



```
-- INSERT CART --
DELIMITER $$

CREATE PROCEDURE InsertCart(
    IN p_BuyerID INT
)
BEGIN
    DECLARE v_exists INT;

    -- Validate 1: Buyer phải tồn tại
    SELECT COUNT(*) INTO v_exists FROM Buyer WHERE BuyerID = p_BuyerID;
    IF v_exists = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Lỗi: BuyerID không tồn tại.';
    END IF;

    -- Validate 2: Buyer chỉ được phép có 1 Cart
    SELECT COUNT(*) INTO v_exists FROM Cart WHERE BuyerID = p_BuyerID;
    IF v_exists > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Lỗi: Buyer này đã có Cart.';
    END IF;

    -- Thực hiện INSERT
    INSERT INTO Cart(BuyerID)
    VALUES (p_BuyerID);
END$$
DELIMITER ;
```

Nhân xét:Lệnh goi thủ tục thực thi thành công. Dòng dữ liệu có LineID = 6 đã bị xóa hoàn toàn khỏi bảng ORDER_LINE. Số lượng tồn kho của sản phẩm đã được hoàn trả và tổng giá trị đơn hàng được tính toán lại chính xác.

2.2.1.7 Cart Thủ tục Insert



```
1 • CALL InsertCart(4);
2 • SELECT * FROM Cart WHERE BuyerID=4;
3
```

100% 31:2

Result Grid Filter Rows: Search Edit: Export/Import:

CartID	BuyerID
4	4
HULL	HULL

Mô tả: Thủ tục InsertCart được thiết kế nhằm mục đích tạo mới một giỏ hàng (Cart) trong hệ thống cho một người mua (Buyer). -Các bước kiểm tra ràng buộc: +Kiểm tra tham chiếu : BuyerID phải nằm trong bảng Buyer -> Giúp đảm bảo giỏ hàng chỉ được tạo cho người dùng hợp lệ Kiểm tra:



```
DELIMITER $$  
CREATE PROCEDURE UpdateCart(  
    IN p_CartID INT,  
    IN p_NewBuyerID INT  
)  
BEGIN  
    DECLARE v_exists INT;  
  
    -- Validate 1: Cart phải tồn tại  
    SELECT COUNT(*) INTO v_exists FROM Cart WHERE CartID = p_CartID;  
    IF v_exists = 0 THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: CartID không tồn tại.';  
    END IF;  
  
    -- Validate 2: Buyer mới phải tồn tại  
    SELECT COUNT(*) INTO v_exists FROM Buyer WHERE BuyerID = p_NewBuyerID;  
    IF v_exists = 0 THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'BuyerID mới không tồn tại.';  
    END IF;  
  
    -- Validate 3: Buyer mới không được có cart khác  
    SELECT COUNT(*) INTO v_exists  
    FROM Cart  
    WHERE BuyerID = p_NewBuyerID AND CartID <> p_CartID;  
    IF v_exists > 0 THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: Buyer này đã có Cart khác.';  
    END IF;  
  
    -- Thực hiện UPDATE  
    UPDATE Cart  
    SET BuyerID = p_NewBuyerID  
    WHERE CartID = p_CartID;  
END$$  
DELIMITER ;
```

Thủ tục Update



```
3 • CALL UpdateCart(1, 4);
4 • SELECT * FROM Cart WHERE CartID = 1;
5
6
```

VI 1:5

Result Grid Filter Rows: Search Edit: Export/Import:

CartID	BuyerID
1	1
NULL	NULL

Mô tả: Thủ tục UpdateCart được thiết kế để thay đổi Buyer sở hữu một Cart.

Trước khi cập nhật, thủ tục thực hiện các bước kiểm tra nhằm đảm bảo tính thống nhất dữ liệu.

Điều kiện để validate là:

- Cart phải tồn tại
- Buyer mới phải tồn tại
- Buyer mới không được có cart khác
- Khi hợp lệ hết tất cả mới được Update

Kiểm tra:



```
-- DELETE CART --
DELIMITER $$

CREATE PROCEDURE DeleteCart(
    IN p_CartID INT
)
BEGIN
    DECLARE v_exists INT;

    -- Validate 1: Cart phải tồn tại
    SELECT COUNT(*) INTO v_exists FROM Cart WHERE CartID = p_CartID;
    IF v_exists = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Lỗi: CartID không tồn tại.';
    END IF;

    -- Validate 2: Không được xóa nếu còn CartItem
    SELECT COUNT(*) INTO v_exists FROM CartItem WHERE CartID = p_CartID;
    IF v_exists > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Lỗi: Không thể xóa Cart vì còn CartItem.';
    END IF;

    -- Thực hiện DELETE
    DELETE FROM Cart WHERE CartID = p_CartID;
END$$
DELIMITER ;
```

Thủ tục Delete

```
5 •    CALL DeleteCart(2);
6 •    SELECT * FROM Cart WHERE CartID = 2;
7
100%  37:4
```

Result Grid Filter Rows: Search Edit: Export/Import

CartID	BuyerID
2	2
NULL	NULL

Mô tả: Thủ tục DeleteCart được sử dụng để xóa toàn bộ một giỏ hàng (Cart) khỏi hệ thống. Trước khi xóa, hệ thống kiểm tra sự tồn tại của Cart và đảm bảo rằng giỏ hàng không chứa bất kỳ CartItem nào.

Mục đích: Việc xóa Cart nhằm dọn dẹp dữ liệu khi giỏ hàng không còn được sử dụng, giỏ hàng bị tạo nhầm hoặc giỏ hàng trả về trống sau khi người dùng hủy phiên đăng nhập hoặc xóa tài khoản. Điều này giúp cơ sở dữ liệu tránh lưu các giỏ hàng rác, cải thiện hiệu suất và tính sạch của dữ liệu.

2.2.1.8 CartItem Thủ tục Insert



```
• └ CREATE PROCEDURE InsertCartItem(
    IN p_CartID INT,
    IN p_ProductVariantID INT,
    IN p_SKU VARCHAR(100),
    IN p_Quantity INT
)
└ BEGIN
    DECLARE v_exists INT;

    -- Validate 1: Cart phải tồn tại
    SELECT COUNT(*) INTO v_exists FROM Cart WHERE CartID = p_CartID;
    IF v_exists = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Lỗi: CartID không tồn tại.';
    END IF;

    -- Validate 2: ProductVariant phải tồn tại
    SELECT COUNT(*) INTO v_exists
    FROM product_variant
    WHERE ProductVariantID = p_ProductVariantID;
    IF v_exists = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Lỗi: ProductVariantID không tồn tại.';
    END IF;

    -- Validate 3: Quantity hợp lệ
    IF p_Quantity <= 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Lỗi: Quantity phải > 0.';
    END IF;
```



```
-- Validate 4: CartItem không được trùng
SELECT COUNT(*) INTO v_exists
FROM CartItem
WHERE CartID = p_CartID
    AND ProductVariantID = p_ProductVariantID;
IF v_exists > 0 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Lỗi: Sản phẩm này đã tồn tại trong Cart.';
END IF;

-- Insert dữ liệu
INSERT INTO CartItem(CartID, ProductVariantID, SKU, Quantity)
VALUES (p_CartID, p_ProductVariantID, p_SKU, p_Quantity);
END$$
DELIMITER ;
```

Mô tả: Thủ tục InsertCartItem được thiết kế nhằm mục đích thêm một sản phẩm (variant) vào giỏ hàng của người dùng.

Các bước kiểm tra ràng buộc:

Kiểm tra tham chiếu: CartID phải tồn tại trong bảng Cart và ProductVariantID phải tồn tại trong bảng product_variant

Kiểm tra dữ liệu: Quantity phải lớn hơn 0 để đảm bảo giá trị hợp lệ Kiểm tra:



```
7 • CALL InsertCartItem(1, 2, 'SKU-TEST', 3);
8 • SELECT * FROM CartItem WHERE CartID = 1;
9
10
11
```

00% 0 | 1:7

Action Output

Time Action Response

Thủ tục Update



```
-- Update CartItem --  
  
DELIMITER $$  
CREATE PROCEDURE UpdateCartItem(  
    IN p_CartID INT,  
    IN p_ProductVariantID INT,  
    IN p_Quantity INT  
)  
BEGIN  
    DECLARE v_exists INT;  
  
    -- Validate 1: CartItem phải tồn tại  
    SELECT COUNT(*) INTO v_exists  
    FROM CartItem  
    WHERE CartID = p_CartID  
        AND ProductVariantID = p_ProductVariantID;  
    IF v_exists = 0 THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: CartItem không tồn tại.';  
    END IF;  
  
    -- Validate 2: Quantity hợp lệ  
    IF p_Quantity <= 0 THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: Quantity phải > 0.';  
    END IF;  
  
    -- Update  
    UPDATE CartItem  
    SET Quantity = p_Quantity  
    WHERE CartID = p_CartID AND ProductVariantID = p_ProductVariantID;  
END$$  
DELIMITER ;
```

Mô tả: Thủ tục UpdateCartItem được xây dựng để điều chỉnh số lượng sản phẩm trong giỏ hàng.
Trước khi thực hiện UPDATE, thủ tục kiểm tra :

Kiểm tra tồn tại: CartItem (CartID + ProductVariantID) phải tồn tại.

Kiểm tra dữ liệu: Quantity phải lớn hơn 0 -> đảm bảo số lượng hợp lệ

Kiểm tra:



```
27 • CALL UpdateCartItem(1, 2, 10);
28 • SELECT * FROM CartItem WHERE CartID = 1;
29
100% 41:28 |
```

Result Grid Filter Rows: Search Edit: Export/Import:

CartID	ProductVariantID	SKU	Quantity
1	1	SKU-1	2
1	2	SKU-2	1
NULL	NULL	NULL	NULL

Thủ tục Delete

```
-- Delete CartItem --

DELIMITER $$

CREATE PROCEDURE DeleteCartItem(
    IN p_CartID INT,
    IN p_ProductVariantID INT
)
BEGIN
    DECLARE v_exists INT;

    -- Validate 1: CartItem phải tồn tại
    SELECT COUNT(*) INTO v_exists
    FROM CartItem
    WHERE CartID = p_CartID AND ProductVariantID = p_ProductVariantID;
    IF v_exists = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Lỗi: CartItem không tồn tại.';
    END IF;

    -- Delete
    DELETE FROM CartItem
    WHERE CartID = p_CartID AND ProductVariantID = p_ProductVariantID;
END$$
DELIMITER ;
```

Mô tả: Thủ tục DeleteCartItem thực hiện xóa một sản phẩm (variant) ra khỏi giỏ hàng của người dùng dựa trên cặp (CartID, ProductVariantID).

Mục đích: Giúp người dùng chỉnh sửa giỏ hàng theo nhu cầu thực tế, loại bỏ sản phẩm không muốn mua, tránh tính tiền sai hoặc tránh lưu trữ sản phẩm lỗi thời trong giỏ hàng.

Kiểm tra ràng buộc: chỉ cho phép xóa khi CartItem tồn tại đúng với cặp CartID – ProductVariantID; hệ thống sẽ tự động chặn nếu CartItem không tồn tại để tránh thao tác xóa sai và bảo toàn tính nhất quán dữ liệu.

Kiểm tra:

```
29 • CALL DeleteCartItem(1, 2);
30 • SELECT * FROM CartItem WHERE CartID = 1;
31
100% 4:130 |
```

Result Grid Filter Rows: Search Edit: Export/Import:

CartID	ProductVariant...	SKU	Quantity
1	1	SKU-1	2
1	2	SKU-2	1
NULL	NULL	NULL	NULL

2.2.1.9 Store Thủ tục Insert



```
DELIMITER $$  
CREATE PROCEDURE InsertStore(  
    IN p_SellerID INT,  
    IN p_Name VARCHAR(255),  
    IN p_Brand VARCHAR(255),  
    IN p_BrandProfile TEXT,  
    IN p_MallLabel BOOLEAN,  
    IN p_ReturnPolicy TEXT,  
    IN p_Status VARCHAR(50)  
)  
BEGIN  
    -- 1. Kiểm tra SellerID có tồn tại không  
    IF NOT EXISTS (SELECT 1 FROM Seller WHERE SellerID = p_SellerID) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: SellerID không tồn tại.';  
    END IF;  
    -- 2. Kiểm tra NOT NULL  
    IF p_Name IS NULL OR p_Brand IS NULL OR p_BrandProfile IS NULL  
        OR p_ReturnPolicy IS NULL OR p_Status IS NULL THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: Các trường của Store không được NULL.';  
    END IF;  
    -- 3. Kiểm tra Status hợp lệ  
    IF p_Status NOT IN ('Active', 'Inactive') THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: Status phải là Active hoặc Inactive.';  
    END IF;  
    -- 4. Thực hiện Insert  
    INSERT INTO Store(SellerID, Name, Brand, BrandProfile, MallLabel, ReturnPolicy, Status)  
    VALUES(p_SellerID, p_Name, p_Brand, p_BrandProfile, p_MallLabel, p_ReturnPolicy, p_Status);  
END$$  
DELIMITER ;
```

Mô tả : Thủ tục InsertStore được xây dựng nhằm mục đích thêm mới một cửa hàng vào hệ thống, liên kết trực tiếp với một người bán (Seller).

Các bước kiểm tra ràng buộc:

Kiểm tra tham chiếu: Xác nhận SellerID phải tồn tại trong bảng Seller để đảm bảo cửa hàng được gắn đúng với người bán hợp lệ.

Kiểm tra tính hợp lệ dữ liệu: Các trường thông tin bắt buộc như Name, Brand, BrandProfile, ReturnPolicy, Status đều không được NULL nhằm tránh dữ liệu thiếu hoàn chỉnh.

Kiểm tra miền giá trị: Trạng thái cửa hàng (Status) chỉ được phép nhận giá trị 'Active' hoặc 'Inactive', ngăn chặn nhập sai chính tả hoặc sai quy định.

Sau khi vượt qua toàn bộ kiểm tra, cửa hàng sẽ được thêm mới vào hệ thống.



Kiểm tra:

The screenshot shows a MySQL Workbench interface. The SQL editor pane contains the following code:

```
1 • USE BTL2_SHOPEE;
2 • CALL InsertStore(
3     1,
4     'Test Store 14',
5     'Brand',
6     'Profile test',
7     TRUE,
8     'Cho đổi trả 14 ngày',
9     'Inactive'
10 );
11 • SELECT * FROM Store ORDER BY StoreID DESC LIMIT 5;
```

The Result Grid pane displays the following data:

StoreID	SellerID	Name	Brand	BrandProfile	MallLabel	ReturnPolicy	Status
7	1	Test Store 14	Brand	Profile test	1	Cho đổi trả 14 ngày	Inactive
6	5	David Bookstore	ReadMore	ReadMore cung cấp sách và văn phòng phẩm v...	0	Không cho phép trả hàng với sách đã sử dụng h...	Active
5	4	Lisa Home Collection	CozyHome	CozyHome mang đến các sản phẩm trang trí nh...	1	Cho phép trả hàng trong vòng 30 ngày, sản ph...	Inactive
4	3	Mike Sports World	ActiveLife	ActiveLife chuyên về dụng cụ thể thao và đồ t...	0	Cho phép trả hàng trong vòng 14 ngày nếu sản...	Active
3	2	Jane Beauty Paradise	Glamour	Glamour cung cấp mỹ phẩm và sản phẩm chăm ...	1	Không cho phép trả hàng với sản phẩm đã mở s...	Active
HULL	HULL		HULL	HULL	HULL	HULL	HULL

Nhận xét: Dữ liệu cửa hàng được thêm mới và hiển thị đúng trong bảng Store, đảm bảo toàn bộ ràng buộc được tuân thủ.

Thủ tục Update



```
DELIMITER $$  
CREATE PROCEDURE UpdateStore(  
    IN p_StoreID INT,  
    IN p_Name VARCHAR(255),  
    IN p_Brand VARCHAR(255),  
    IN p_BrandProfile TEXT,  
    IN p_MallLabel BOOLEAN,  
    IN p_ReturnPolicy TEXT,  
    IN p_Status VARCHAR(50)  
)  
BEGIN  
    -- 1. Kiểm tra StoreID tồn tại  
    IF NOT EXISTS (SELECT 1 FROM Store WHERE StoreID = p_StoreID) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: StoreID không tồn tại.';  
    END IF;  
    -- 2. Kiểm tra ràng buộc NOT NULL  
    IF p_Name IS NULL OR p_Brand IS NULL OR p_BrandProfile IS NULL  
        OR p_ReturnPolicy IS NULL OR p_Status IS NULL THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: Các trường của Store không được NULL.';  
    END IF;  
    -- 3. Kiểm tra Status hợp lệ  
    IF p_Status NOT IN ('Active', 'Inactive') THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: Status phải là Active hoặc Inactive.';  
    END IF;  
    -- 4. Update  
    UPDATE Store  
    SET Name = p_Name,  
        Brand = p_Brand,  
        BrandProfile = p_BrandProfile,  
        MallLabel = p_MallLabel,  
        ReturnPolicy = p_ReturnPolicy,  
        Status = p_Status  
    WHERE StoreID = p_StoreID;  
END$$  
DELIMITER ;
```



Mô tả: Thủ tục UpdateStore cho phép người dùng cập nhật thông tin của một cửa hàng đang tồn tại. Để đảm bảo tính nhất quán dữ liệu, thủ tục áp dụng các cơ chế kiểm tra sau:

Kiểm tra sự tồn tại: StoreID cần cập nhật phải thực sự tồn tại trong hệ thống.

Kiểm tra dữ liệu bắt buộc: Các trường Name, Brand, BrandProfile, ReturnPolicy và Status phải có giá trị đầy đủ, tránh tình trạng ghi đè dữ liệu bằng NULL.

Kiểm tra miền giá trị: Trạng thái cửa hàng được cập nhật phải thuộc danh sách cho phép ('Active', 'Inactive').

Thủ tục đảm bảo chỉ cập nhật thông tin mô tả, không làm thay đổi SellerID đã gắn liền với cửa hàng. Kiểm tra:

Trước khi test

The screenshot shows a database interface with the following content:

```
1 • USE BTL2_SHOPEE;
2 • CALL InsertStore(
3     1,
4     'Test Store 33',
5     'BrandTest',
6     'Profile test',
7     TRUE,
8     'Cho đổi trả 7 ngày',
9     'Active'
10 );
11 • SELECT * FROM Store ORDER BY StoreID DESC LIMIT 7;
```

Result Grid:

StoreID	SellerID	Name	Brand	BrandProfile	MailLabel	ReturnPolicy	Status
20	1	Test Store 33	BrandTest	Profile test	1	Cho đổi trả 7 ngày	Active
6	5	David Bookstore	ReadMore	ReadMore cung cấp sách và văn phòng phẩm v...	0	Không cho phép trả hàng với sách đã sử dụng h...	Active
5	4	Lisa Home Collection	CozyHome	CozyHome mang đến các sản phẩm trang trí nh...	1	Cho phép trả hàng trong vòng 30 ngày, sản ph...	Inactive
4	3	Mike Sports World	ActiveLife	ActiveLife chuyên về dụng cụ thể thao và đồ tâ...	0	Cho phép trả hàng trong vòng 14 ngày nếu sản...	Active
3	2	Jane Beauty Paradise	Glamour	Glamour cung cấp mỹ phẩm và sản phẩm chăm ...	1	Không cho phép trả hàng với sản phẩm đã mở s...	Active
2	1	John Fashion Hub	StyleUp	StyleUp mang đến những xu hướng thời trang ...	0	Cho phép trả hàng trong vòng 7 ngày, áo quần...	Active
1	1	John Electronics Store	TechPro	TechPro là thương hiệu chuyên về thiết bị điện t...	1	Cho phép trả hàng trong vòng 30 ngày với điều...	Active
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL

Sau khi test



The screenshot shows a MySQL Workbench interface. At the top, there is a toolbar with various icons. Below the toolbar, the SQL editor contains the following code:

```
1 • USE BTL2_SHOPEE;
2 • SELECT StoreID INTO @TestStoreID FROM Store ORDER BY StoreID DESC LIMIT 1;
3
4 • CALL UpdateStore(
    @TestStoreID,
    'Updated Test Store',
    'UpdatedBrand',
    'Updated Profile',
    FALSE,
    'Đổi trả 14 ngày',
    'Inactive'
);
13 • SELECT * FROM Store ORDER BY StoreID DESC LIMIT 7;
```

Below the SQL editor is a "Result Grid" table with the following data:

StoreID	SellerID	Name	Brand	BrandProfile	MailLabel	ReturnPolicy	Status
20	1	Updated Test Store	UpdatedBrand	Updated Profile	0	Đổi trả 14 ngày	Inactive
6	5	David Bookstore	ReadMore	ReadMore cung cấp sách và văn phòng phẩm v...	0	Không cho phép trả hàng với sách đã sử dụng h...	Active
5	4	Lisa Home Collection	CozyHome	CozyHome mang đến các sản phẩm trang trí nh...	1	Cho phép trả hàng trong vòng 30 ngày, sản ph...	Inactive
4	3	Mike Sports World	ActiveLife	ActiveLife chuyên về dụng cụ thể thao và đồ tă...	0	Cho phép trả hàng trong vòng 14 ngày nếu sản...	Active
3	2	Jane Beauty Paradise	Glamour	Glamour cung cấp mỹ phẩm và sản phẩm chăm ...	1	Không cho phép trả hàng với sản phẩm đã mở s...	Active
2	1	John Fashion Hub	StyleUp	StyleUp mang đến những xu hướng thời trang ...	0	Cho phép trả hàng trong vòng 7 ngày, áo quần...	Active
1	1	John Electronics Store	TechPro	TechPro là thương hiệu chuyên về thiết bị điện t...	1	Cho phép trả hàng trong vòng 30 ngày với điều...	Active
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL

Nhận xét: Thông tin cửa hàng được cập nhật thành công sau khi vượt qua các ràng buộc.

Thủ tục Delete



```
DELIMITER $$  
CREATE PROCEDURE DeleteStore(IN p_StoreID INT)  
BEGIN  
    -- 1. Kiểm tra StoreID  
    IF NOT EXISTS (SELECT 1 FROM Store WHERE StoreID = p_StoreID) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: StoreID không tồn tại.';  
    END IF;  
  
    -- 2. Không được xóa nếu còn Product thuộc store này  
    IF EXISTS (SELECT 1 FROM Product WHERE StoreID = p_StoreID) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: Store còn Product nên không thể xóa.';  
    END IF;  
  
    -- 3. Delete  
    DELETE FROM Store WHERE StoreID = p_StoreID;  
END$$  
DELIMITER ;
```

Mô tả: Thủ tục DeleteStore được dùng để xóa một cửa hàng khi cửa hàng không còn sản phẩm nào đang được quản lý trong hệ thống.

Mục đích: Dùng để loại bỏ các cửa hàng đã ngừng kinh doanh, cửa hàng bị khóa hoặc cửa hàng tạo nhầm; giúp danh sách cửa hàng rõ ràng, đảm bảo dữ liệu gọn và đúng thực tế.

Kiểm tra ràng buộc: chỉ cho phép xóa khi StoreID tồn tại và cửa hàng không còn bất kỳ sản phẩm nào trong bảng Product; hệ thống sẽ tự động chặn nếu StoreID không tồn tại hoặc nếu cửa hàng vẫn còn sản phẩm, nhằm bảo toàn dữ liệu liên kết.

Kiểm tra:

Trường hợp nếu Store không còn product:

Trước khi xóa:



```
USE BTL2_SHOPEE;
-- Delete
CALL DeleteStore(9);
SELECT * FROM Store ORDER BY StoreID DESC LIMIT 5;
```

StoreID	SellerID	Name	Brand	BrandProfile	MallLabel	ReturnPolicy	Status
9	1	Store Xâa	Brand X	Profile X	1	Policy X	Active
7	1	Test Store 14	Brand	Profile test	1	Cho đổi trả 14 ngày	Inactive
6	5	David Bookstore	ReadMore	ReadMore cung cấp sách và văn phòng phẩm v...	0	Không cho phép trả hàng với sách đã sử dụng h...	Active
5	4	Lisa Home Collection	CozyHome	CozyHome mang đến các sản phẩm trang trí nh...	1	Cho phép trả hàng trong vòng 30 ngày, sản ph...	Inactive
4	3	Mike Sports World	ActiveLife	ActiveLife chuyên về dụng cụ thể thao và đồ tă...	0	Cho phép trả hàng trong vòng 14 ngày nếu sản...	Active
HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

Sau khi xóa:

```
USE BTL2_SHOPEE;
-- Delete
CALL DeleteStore(9);
SELECT * FROM Store ORDER BY StoreID DESC LIMIT 5;
```

StoreID	SellerID	Name	Brand	BrandProfile	MallLabel	ReturnPolicy	Status
7	1	Test Store 14	Brand	Profile test	1	Cho đổi trả 14 ngày	Inactive
6	5	David Bookstore	ReadMore	ReadMore cung cấp sách và văn phòng phẩm v...	0	Không cho phép trả hàng với sách đã sử dụng h...	Active
5	4	Lisa Home Collection	CozyHome	CozyHome mang đến các sản phẩm trang trí nh...	1	Cho phép trả hàng trong vòng 30 ngày, sản ph...	Inactive
4	3	Mike Sports World	ActiveLife	ActiveLife chuyên về dụng cụ thể thao và đồ tă...	0	Cho phép trả hàng trong vòng 14 ngày nếu sản...	Active
3	2	Jane Beauty Paradise	Glamour	Glamour cung cấp mỹ phẩm và sản phẩm chăm ...	1	Không cho phép trả hàng với sản phẩm đã mở s...	Active
HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

Store 44 x

Action Output

#	Time	Action	Message
178	14:34:08	CALL DeleteStore(10)	1 row(s) affected
179	14:34:08	SELECT * FROM Store ORDER BY StoreID DESC LIMIT 5	5 row(s) returned
180	14:35:11	USE BTL2_SHOPEE	0 row(s) affected
181	14:35:11	CALL DeleteStore(9)	1 row(s) affected
182	14:35:11	SELECT * FROM Store ORDER BY StoreID DESC LIMIT 5	5 row(s) returned

Trường hợp nếu Store còn product:



```
File Edit View Insert Cell Tools Help
Limit to 1000 rows | Star | Undo | Redo | Search | Print | Copy | Paste | Clear | Save As | Open | Import | Export | Refresh | Help | About | Exit

1 • USE BTL2_SHOPEE;
2 -- Delete
3 • SELECT * FROM Product WHERE StoreID = 5;
4 • CALL DeleteStore(7);
5 • SELECT * FROM Store ORDER BY StoreID DESC LIMIT 7;
6
7
8
```

Result Grid					Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
	ProductID	StoreID	Status	Title	Description			
▶	10	5	Available	Đèn ngủ LED điều chỉnh độ sáng 7 màu	Đèn ngủ với 7 màu RGB thay đổi linh hoạt, điều ...	HULL	HULL	HULL
*								

Trước khi xóa:



```
1 • USE BTL2_SHOPEE;
2 -- Delete
3 • SELECT * FROM Product WHERE StoreID = 5;
4 • CALL DeleteStore(5);
5 • SELECT * FROM Store ORDER BY StoreID DESC LIMIT 7;
6
7
8
```

Result Grid								
StoreID	SellerID	Name	Brand	BrandProfile	MailLabel	ReturnPolicy	Status	
6	5	David Bookstore	ReadMore	ReadMore cung cấp sách và văn phòng phẩm v...	0	Không cho phép trả hàng với sách đã sử dụng h...	Active	
5	4	Lisa Home Collection	CozyHome	CozyHome mang đến các sản phẩm trang trí nh...	1	Cho phép trả hàng trong vòng 30 ngày, sản ph...	Inactive	
4	3	Mike Sports World	ActiveLife	ActiveLife chuyên về dụng cụ thể thao và đồ tâ...	0	Cho phép trả hàng trong vòng 14 ngày nếu sản...	Active	
3	2	Jane Beauty Paradise	Glamour	Glamour cung cấp mỹ phẩm và sản phẩm chăm ...	1	Không cho phép trả hàng với sản phẩm đã mở s...	Active	
2	1	John Fashion Hub	StyleUp	StyleUp mang đến những xu hướng thời trang ...	0	Cho phép trả hàng trong vòng 7 ngày, áo quần...	Active	
1	1	John Electronics Store	TechPro	TechPro là thương hiệu chuyên về thiết bị điện t...	1	Cho phép trả hàng trong vòng 30 ngày với điều...	Active	
HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL
Product 56		Store 57	X					

Sau khi xóa:



```
1 • USE BTL2_SHOPEE;
2   -- Delete
3 • SELECT * FROM Product WHERE StoreID = 5;
4 • CALL DeleteStore(5);
5 • SELECT * FROM Store ORDER BY StoreID DESC LIMIT 7;
6
7
8
```

ProductID	StoreID	Status	Title	Description
10	5	Available	Đèn ngủ LED điều chỉnh độ sáng 7 màu	Đèn ngủ với 7 màu RGB thay đổi linh hoạt, điều ...
*	HULL	HULL	HULL	HULL

Action Output:

#	Time	Action	Message
203	14:50:01	SELECT * FROM Product WHERE StoreID = 5 LIMIT 0, 1000	1 row(s) returned
204	14:50:01	CALL DeleteStore(7)	1 row(s) affected
205	14:50:01	SELECT * FROM Store ORDER BY StoreID DESC LIMIT 7	6 row(s) returned
206	21:17:29	USE BTL2_SHOPEE	0 row(s) affected
207	21:17:29	SELECT * FROM Product WHERE StoreID = 5 LIMIT 0, 1000	1 row(s) returned
208	21:17:29	CALL DeleteStore(5)	Error Code: 1644. Lỗi: Store còn Product nên không thể xóa.

Nhận xét: Cửa hàng được xóa đúng quy tắc. Trường hợp còn Product đã bị hệ thống từ chối và thông báo đúng lỗi.

2.2.1.10 Category Thủ tục Insert



```
DELIMITER $$  
CREATE PROCEDURE InsertCategory(  
    IN p_ParentID INT,  
    IN p_Name VARCHAR(255)  
)  
BEGIN  
    -- 1. Name không được NULL  
    IF p_Name IS NULL THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: Category.Name không được NULL.';  
    END IF;  
    -- 2. ParentCategoryID phải tồn tại nếu không NULL  
    IF p_ParentID IS NOT NULL AND  
        NOT EXISTS (SELECT 1 FROM Category WHERE CategoryID = p_ParentID) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: ParentCategoryID không tồn tại.';  
    END IF;  
    -- 3. Insert  
    INSERT INTO Category(ParentCategoryID, Name)  
    VALUES(p_ParentID, p_Name);  
END$$  
DELIMITER ;
```

Mô tả Thủ tục InsertCategory được thiết kế để thêm mới một danh mục vào hệ thống.
Các bước thực hiện ràng buộc:

Kiểm tra tính bắt buộc: Tên danh mục (Name) không được phép NULL.

Kiểm tra tham chiếu cha – con: Nếu ParentCategoryID được cung cấp, ID này phải tồn tại trong bảng Category nhằm đảm bảo cấu trúc cây danh mục hợp lệ. Sau khi đáp ứng đầy đủ yêu cầu, danh mục được thêm vào hệ thống.

Kiểm tra



```
1 • USE BTL2_SHOPEE;
2 • CALL InsertCategory(NULL, 'Category Test Insert');
3 • CALL InsertCategory(9999, 'Fail Category');
4 • SELECT * FROM Category ORDER BY CategoryID DESC LIMIT 7;
5
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

CategoryID	ParentCategoryID	Name
8	NULL	Category Test Insert
5	2	Phụ kiện thời trang
4	1	Điện thoại
3	NULL	Gia dụng
2	NULL	Thời trang
1	NULL	Điện tử
*	NULL	NULL

Category 69 x

Output:

Action Output	#	Time	Action	Message
1 22:07:28 USE BTL2_SHOPEE	1	22:07:28	USE BTL2_SHOPEE	0 row(s) affected
2 22:07:28 CALL InsertCategory(NULL, 'Category Test Insert')	2	22:07:28	CALL InsertCategory(NULL, 'Category Test Insert')	1 row(s) affected
3 22:07:28 CALL InsertCategory(9999, 'Fail Category')	3	22:07:28	CALL InsertCategory(9999, 'Fail Category')	Error Code: 1644. Lỗi: ParentCategoryID không tồn tại.
4 22:07:32 SELECT * FROM Category ORDER BY CategoryID DESC LIMIT 7	4	22:07:32	SELECT * FROM Category ORDER BY CategoryID DESC LIMIT 7	6 row(s) returned

Nhận xét: Danh mục mới được tạo thành công và xuất hiện trong cây phân cấp.

Thủ tục Update



```
DELIMITER $$  
CREATE PROCEDURE UpdateCategory(  
    IN p_CategoryID INT,  
    IN p_ParentID INT,  
    IN p_Name VARCHAR(255)  
)  
BEGIN  
    -- 1. Kiểm tra CategoryID  
    IF NOT EXISTS (SELECT 1 FROM Category WHERE CategoryID = p_CategoryID) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: CategoryID không tồn tại.';  
    END IF;  
    -- 2. Name không được NULL  
    IF p_Name IS NULL THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: Name không được NULL.';  
    END IF;  
    -- 3. ParentCategoryID phải tồn tại nếu không NULL  
    IF p_ParentID IS NOT NULL AND p_ParentID <> p_CategoryID AND  
        NOT EXISTS (SELECT 1 FROM Category WHERE CategoryID = p_ParentID) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: ParentCategoryID không hợp lệ.';  
    END IF;  
    -- 4. Update  
    UPDATE Category  
    SET ParentCategoryID = p_ParentID,  
        Name = p_Name  
    WHERE CategoryID = p_CategoryID;  
END$$  
DELIMITER ;
```

Mô tả: Thủ tục UpdateCategory hỗ trợ cập nhật tên và danh mục cha của một Category hiện có. Trước khi cập nhật, thủ tục chạy các bước kiểm tra:

Kiểm tra sự tồn tại: CategoryID phải có trong bảng Category.

Kiểm tra not-null: Tên mới của danh mục (Name) không được NULL.

Kiểm tra quan hệ cha hợp lệ: Nếu cập nhật ParentCategoryID, ID này phải tồn tại và không được trùng với chính CategoryID để đảm bảo không tạo vòng lặp.

Kiểm tra



```
1 • USE BTL2_SHOPEE;
2 • CALL UpdateCategory(8, NULL, 'Category Test Updated');
3 • SELECT * FROM Category ORDER BY CategoryID DESC LIMIT 7;
4
```

Result Grid		
CategoryID	ParentCategoryID	Name
8	NULL	Category Test Updated
5	2	Phụ kiện thời trang
4	1	Điện thoại
3	NULL	Gia dụng
2	NULL	Thời trang
1	NULL	Điện tử
*	NULL	NULL

Nhận xét: Danh mục được cập nhật thành công và cấu trúc phân cấp vẫn hợp lệ.

Thủ tục Delete



```
DELIMITER $$  
CREATE PROCEDURE DeleteCategory(IN p_CategoryID INT)  
BEGIN  
    -- 1. Kiểm tra CategoryID  
    IF NOT EXISTS (SELECT 1 FROM Category WHERE CategoryID = p_CategoryID) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: CategoryID không tồn tại.';  
    END IF;  
    -- 2. Không được xóa nếu còn liên kết Product_Category  
    IF EXISTS (SELECT 1 FROM Product_Category WHERE CategoryID = p_CategoryID) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Lỗi: Category đang được sử dụng trong Product_Category.';  
    END IF;  
    DELETE FROM Category WHERE CategoryID = p_CategoryID;  
END$$  
DELIMITER ;
```

Mô tả: Thủ tục DeleteCategory xóa một danh mục khỏi hệ thống khi danh mục đó không được sử dụng bởi bất kỳ sản phẩm nào.

Mục đích: Dảm bảo cấu trúc phân loại sản phẩm luôn rõ ràng, loại bỏ các danh mục bị sai, lỗi, trùng hoặc không còn sử dụng. Điều này giúp khách hàng duyệt danh mục chính xác hơn và giúp hệ thống gọn nhẹ.

Kiểm tra ràng buộc: chỉ cho phép xóa khi CategoryID tồn tại và danh mục không còn xuất hiện trong bảng Product_Category; hệ thống sẽ tự động chặn nếu CategoryID không tồn tại hoặc nếu danh mục vẫn đang liên kết với sản phẩm, nhằm đảm bảo không tạo ra liên kết rỗng trong dữ liệu.

Kiểm tra:



```
1 • USE BTL2_SHOPEE;
2 • CALL DeleteCategory(8);
3 • SELECT * FROM Category ORDER BY CategoryID DESC LIMIT 7;
4
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	CategoryID	ParentCategoryID	Name
▶	5	2	Phụ kiện thời trang
	4	1	Điện thoại
	3	NULL	Gia dụng
	2	NULL	Thời trang
	1	NULL	Điện tử
*	NULL	NULL	NULL

Category 71 ×

Output:

Action Output	#	Time	Action	Message
✓ 8 22:10:04 USE BTL2_SHOPEE				0 row(s) affected
✗ 9 22:10:04 CALL DeleteCategory(8, NULL, 'Category Test Updated')				Error Code: 1318. Incorrect
✓ 10 22:10:13 USE BTL2_SHOPEE				0 row(s) affected
✓ 11 22:10:13 CALL DeleteCategory(8)				1 row(s) affected
✓ 12 22:10:13 SELECT * FROM Category ORDER BY CategoryID DESC LIMIT 7				5 row(s) returned

Nhận xét: Danh mục chỉ bị xóa khi không còn bất kỳ sản phẩm nào thuộc danh mục đó.

2.2.1.11 Product_Variant Thủ tục Insert



```
CREATE PROCEDURE sp_Insert_ProductVariant(
    IN p_ProductID INT,
    IN p_Price DECIMAL(10, 2),
    IN p_PromotePrice DECIMAL(10, 2),
    IN p_Stock INT
)
BEGIN
    -- Validate 1: Kiểm tra ProductID có tồn tại không
    IF NOT EXISTS (SELECT 1 FROM Product WHERE ProductID = p_ProductID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Sản phẩm (ProductID) không tồn tại.';
    END IF;

    -- Validate 2: Giá gốc phải dương
    IF p_Price <= 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Giá gốc sản phẩm phải lớn hơn 0.';
    END IF;

    -- Validate 3: Tồn kho không âm
    IF p_Stock < 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Số lượng tồn kho không được là số âm.';
    END IF;

    -- Validate 4: Logic Giả Khuyến Mãi (PromotePrice < Price)
    IF p_PromotePrice IS NOT NULL THEN
        IF p_PromotePrice >= p_Price THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Giá khuyến mãi phải nhỏ hơn giá gốc.';
        END IF;
        IF p_PromotePrice <= 0 THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Giá khuyến mãi phải lớn hơn 0.';
        END IF;
    END IF;

    -- Thực hiện INSERT
    INSERT INTO product_variant (ProductID, Price, PromotePrice, Stock)
    VALUES (p_ProductID, p_Price, p_PromotePrice, p_Stock);

    SELECT 'Thêm biến thể thành công!' AS Message;
END$$
```

Mô tả: Thủ tục sp_Insert_ProductVariant được xây dựng nhằm mục đích thêm mới một biến thể sản phẩm (như màu sắc, kích cỡ, cấu hình) vào hệ thống.

Các bước kiểm tra ràng buộc:

Kiểm tra tham chiếu: Đảm bảo ProductID phải tồn tại trong bảng Product.

Kiểm tra miền giá trị: Giá gốc (Price) phải lớn hơn 0 và số lượng tồn kho (Stock) không được là số âm. Kiểm tra logic nghiệp vụ: Nếu có giá khuyến mãi (PromotePrice), giá này bắt buộc phải



nhỏ hơn giá gốc để đảm bảo tính hợp lý trong kinh doanh.

Kiểm tra :

```
1 • CALL sp_Insert_ProductVariant(1, 5000000, 4500000, 100);
2 • SELECT * FROM bt12_shopee.product_variant;
```

Result Grid					
	ProductVariantID	ProductID	PromotePrice	Stock	Price
	24	9	279000.00	50	349000.00
	25	9	279000.00	45	349000.00
	27	10	199000.00	65	249000.00
	29	11	129000.00	100	159000.00
▶	30	12	27000.00	200	35000.00
	31	1	4500000.00	100	5000000.00

Nhận xét : Thêm sản phẩm mẫu khác thành công. Thủ tục Update



```
CREATE PROCEDURE sp_Update_ProductVariant(
    IN p_ProductVariantID INT,
    IN p_Price DECIMAL(10, 2),
    IN p_PromotePrice DECIMAL(10, 2),
    IN p_Stock INT
)
BEGIN
    -- Validate 1: Kiểm tra ID biến thể có tồn tại không
    IF NOT EXISTS (SELECT 1 FROM product_variant WHERE ProductVariantID = p_ProductVariantID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Mã biến thể (ProductVariantID) không tồn tại.';
    END IF;

    -- Validate 2: Giá gốc
    IF p_Price <= 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Giá gốc sản phẩm phải lớn hơn 0.';
    END IF;

    -- Validate 3: Tồn kho
    IF p_Stock IS NULL THEN SET p_Stock = 0; END IF;

    -- Validate 4: Logic Giá Khuyến Mãi
    IF p_PromotePrice IS NOT NULL THEN
        IF p_PromotePrice >= p_Price THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Giá khuyến mãi phải nhỏ hơn giá gốc.';
        END IF;
        IF p_PromotePrice <= 0 THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Giá khuyến mãi phải lớn hơn 0.';
        END IF;
    END IF;

    -- Thực hiện UPDATE
    UPDATE product_variant
    SET Price = p_Price, PromotePrice = p_PromotePrice, Stock = p_Stock
    WHERE ProductVariantID = p_ProductVariantID;

    SELECT 'Cập nhật thành công!' AS Message;
END$$
```



Mô tả: Thủ tục sp_Update_ProductVariant cho phép cập nhật thông tin của một biến thể sản phẩm đã tồn tại (như thay đổi giá bán hoặc cập nhật số lượng tồn kho). Tương tự như thủ tục thêm mới, thủ tục cập nhật cũng thực hiện nghiêm ngặt các bước kiểm tra tính hợp lệ của dữ liệu trước khi ghi vào cơ sở dữ liệu:

Kiểm tra sự tồn tại: Đảm bảo mã biến thể (ProductVariantID) cần sửa phải thực sự tồn tại trong hệ thống.

Kiểm tra logic giá: Giá gốc mới phải là số dương. Nếu có cập nhật giá khuyến mãi, hệ thống sẽ kiểm tra để đảm bảo giá khuyến mãi luôn thấp hơn giá gốc.

Kiểm tra tồn kho: Đảm bảo số lượng tồn kho cập nhật không được là số âm, giữ cho dữ liệu kho hàng luôn chính xác thực tế.

Kiểm tra

```
1 • CALL sp_Update_ProductVariant(7, 4400000, NULL, 80);  
2 • SELECT * FROM bt12_shopee.product_variant;
```

Result Grid Filter Rows: <input type="text"/> Edit: Export/Import					
	ProductVariantID	ProductID	PromotePrice	Stock	Price
4	2	1290000.00	50	1590000.00	
5	2	1290000.00	45	1590000.00	
6	2	1390000.00	30	1590000.00	
7	3	NULL	80	4400000.00	
8	4	149000.00	100	199000.00	

Nhận xét : Sau khi thực lệnh update chuyển Promote Price về null, bảng đã được cập nhật thành công
Nhận xét : Sau khi thực lệnh update chuyển Promote Price về null, bảng đã được cập nhật thành công.

Thủ tục Delete



```
CREATE PROCEDURE sp_Delete_ProductVariant(IN p_ProductVariantID INT)
BEGIN
    DECLARE v_ProductID INT;
    DECLARE v_VariantCount INT;
    DECLARE v_IsInOrder INT;

    -- Lấy ProductID của biến thể định xóa
    SELECT ProductID INTO v_ProductID FROM product_variant WHERE ProductVariantID = p_ProductVariantID;

    -- Validate 1: Kiểm tra ID tồn tại
    IF v_ProductID IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Mã biến thể không tồn tại.';
    END IF;

    -- Validate 2: Không xóa biến thể cuối cùng
    SELECT COUNT(*) INTO v_VariantCount FROM product_variant WHERE ProductID = v_ProductID;
    IF v_VariantCount <= 1 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Không thể xóa biến thể cuối cùng. Sản phẩm phải có ít nhất 1 biến thể.';
    END IF;

    -- Validate 3: Không xóa biến thể đã có trong đơn hàng (Order Line)
    SELECT COUNT(*) INTO v_IsInOrder FROM ORDER_LINE WHERE ProductVariantID = p_ProductVariantID;
    IF v_IsInOrder > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Biến thể đã có trong đơn hàng, không thể xóa.';
    END IF;

    -- Thực hiện DELETE
    DELETE FROM product_variant WHERE ProductVariantID = p_ProductVariantID;

    SELECT 'Xóa thành công!' AS Message;
END$$
```

Mô tả: Thủ tục sp_Delete_ProductVariant xóa một biến thể sản phẩm khi biến thể đó không phải biến thể cuối cùng và chưa từng xuất hiện trong các dòng đơn hàng (ORDER_LINE).

Mục đích: Dùng để xóa các biến thể lỗi, biến thể nhập sai cấu hình (giá, stock), hoặc biến thể đã ngừng kinh doanh. Đồng thời bảo đảm rằng sản phẩm vẫn còn ít nhất một biến thể, tránh trường hợp sản phẩm bị “rỗng” thông tin.

Kiểm tra ràng buộc: chỉ cho phép xóa khi Product Variant ID tồn tại, sản phẩm cha vẫn còn hơn một biến thể, và biến thể chưa từng xuất hiện trong ORDER_LINE; hệ thống sẽ tự động chặn nếu Product Variant ID không tồn tại, nếu đây là biến thể cuối cùng của sản phẩm, hoặc nếu biến thể đã từng được dùng trong đơn hàng, nhằm bảo toàn lịch sử và tính toàn vẹn dữ liệu.

Kiểm tra:

Trước khi xóa



	ProductVariantID	ProductID	PromotePrice	Stock	Price
	24	9	279000.00	50	349000.00
	25	9	279000.00	45	349000.00
	27	10	199000.00	65	249000.00
	28	11	89000.00	150	119000.00
	29	11	129000.00	100	159000.00
▶	30	12	27000.00	200	35000.00

Sau khi xóa

```
1 • CALL sp_Delete_ProductVariant(28);
2 • SELECT * FROM bt12_shopee.product_variant;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	ProductVariantID	ProductID	PromotePrice	Stock	Price
	23	8	119000.00	80	149000.00
	24	9	279000.00	50	349000.00
	25	9	279000.00	45	349000.00
	27	10	199000.00	65	249000.00
	29	11	129000.00	100	159000.00
▶	30	12	27000.00	200	35000.00
*	NULL	NULL	NULL	NULL	NULL

product_variant 15 ×

Output:

Action Output

#	Time	Action	Message
✓	198	21:46:35 CALL sp_Delete_ProductVariant(28)	1 row(s) returned
✓	199	21:46:38 SELECT * FROM bt12_shopee.product_variant LIMIT 0, 100	27 row(s) returned

Nhận xét: Sau khi product _varitant ID :28, trong bảng đã không còn xuất hiện



2.2.1.12 Review Thủ tục Insert

```
-- REVIEW INSERT --
DELIMITER $$

CREATE PROCEDURE `sp_AddReview`(
    IN p_BuyerID INT,
    IN p_ProductID INT,
    IN p_Rating INT,
    IN p_Content TEXT
)
BEGIN
    DECLARE v_purchase_verified INT DEFAULT 0;
    DECLARE v_review_exists INT DEFAULT 0;

    -- Validate 1: Kiểm tra xem người mua đã đánh giá sản phẩm này chưa
    SELECT COUNT(*) INTO v_review_exists FROM REVIEW WHERE BuyerID = p_BuyerID AND ProductID = p_ProductID;
    IF v_review_exists > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Bạn đã đánh giá sản phẩm này rồi.';
    END IF;

    -- Validate 2: Xác thực người mua đã thực sự mua và nhận thành công sản phẩm này
    SELECT COUNT(*) INTO v_purchase_verified
    FROM `Order` o
    JOIN OrderUnit ou ON o.OrderID = ou.OrderID
    JOIN ORDER_LINE ol ON ou.UnitID = ol.UnitID
    JOIN product_variant pv ON ol.ProductVariantID = pv.ProductVariantID
    WHERE o.BuyerID = p_BuyerID
        AND pv.ProductID = p_ProductID
        AND ou.Status = 'Completed';

    IF v_purchase_verified = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Bạn chỉ có thể đánh giá sản phẩm sau khi đã mua và nhận hàng thành công.';
    END IF;

    -- Validate 3: Kiểm tra điểm đánh giá hợp lệ
    IF p_Rating NOT BETWEEN 1 AND 5 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Điểm đánh giá (Rating) phải từ 1 đến 5.';
    END IF;

    -- Thực hiện INSERT
    INSERT INTO REVIEW (BuyerID, ProductID, Rating, Content, CreatedAt)
    VALUES (p_BuyerID, p_ProductID, p_Rating, p_Content, NOW());

    SELECT CONCAT('Thêm đánh giá thành công! ID mới là: ', LAST_INSERT_ID()) AS Message;
END$$
DELIMITER ;
```

Mô tả: Thủ tục sp_AddReview được xây dựng để cho phép người dùng đăng tải một đánh giá mới cho một sản phẩm.

Các bước kiểm tra ràng buộc:

Xác thực giao dịch mua hàng: Đây là ràng buộc quan trọng nhất. Trước khi cho phép INSERT, thủ tục sẽ truy vấn qua các bảng Order, OrderUnit, và ORDER_LINE để xác thực rằng người mua (BuyerID) đã thực sự đặt mua sản phẩm (ProductID) và đơn hàng đó đã ở trạng thái "Completed" (Đã giao thành công). Cơ chế này ngăn chặn tuyệt đối các đánh giá giả mạo hoặc spam từ những người chưa từng trải nghiệm sản phẩm.



Kiểm tra đánh giá trùng lặp: Thủ tục kiểm tra để đảm bảo một người mua chỉ có thể đánh giá một sản phẩm một lần duy nhất, tránh tình trạng một người dùng gửi nhiều đánh giá cho cùng một sản phẩm. Kiểm tra miền giá trị: Điểm đánh giá (Rating) được kiểm tra để đảm bảo nó nằm trong khoảng hợp lệ từ 1 đến 5.

Kiểm tra:

```
-- REVIEW INSERT --
-- Test Validate trả lỗi 1644
CALL sp_AddReview(1, 1, 5, 'Điện thoại tốt, pin trâu, chụp ảnh đẹp!');
SELECT * FROM btl2_shopee.review WHERE BuyerID = 1 AND ProductID = 1;
```

200 23:33:58 CALL sp_AddReview(4, 4, 5, 'Điện thoại tốt, pin trâu, chụp ảnh đẹp!')

Error Code: 1644. Lỗi: Bạn chỉ có thể đánh giá sản phẩm sau khi đã mua và nhận hàng thành công.

Nhận xét: Xử lý trường hợp review khi đã có review trước đó **Thủ tục Update**



```
-- REVIEW UPDATE --  
  
DELIMITER $$  
CREATE PROCEDURE `sp_UpdateReview`(  
    IN p_ReviewID INT,  
    IN p_BuyerID INT, -- Cần ID người mua để xác thực quyền  
    IN p_NewRating INT,  
    IN p_NewContent TEXT  
)  
BEGIN  
    DECLARE v_original_buyer_id INT;  
    DECLARE v_created_at DATETIME;  
  
    -- Validate 1: Kiểm tra sự tồn tại của Review  
    SELECT BuyerID, CreatedAt INTO v_original_buyer_id, v_created_at  
    FROM REVIEW WHERE ReviewID = p_ReviewID;  
  
    IF v_original_buyer_id IS NULL THEN  
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Đánh giá (ReviewID) không tồn tại.';  
    END IF;  
  
    -- Validate 2: Kiểm tra quyền sở hữu  
    IF v_original_buyer_id != p_BuyerID THEN  
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Bạn không có quyền sửa đánh giá này.';  
    END IF;  
  
    -- Validate 3: Kiểm tra giới hạn thời gian sửa (7 ngày)  
    IF v_created_at < NOW() - INTERVAL 7 DAY THEN  
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Không thể chỉnh sửa đánh giá sau 7 ngày.';  
    END IF;  
  
    -- Validate 4: Kiểm tra điểm đánh giá mới hợp lệ  
    IF p_NewRating NOT BETWEEN 1 AND 5 THEN  
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Điểm đánh giá (Rating) mới phải từ 1 đến 5.';  
    END IF;  
  
    -- Thực hiện UPDATE  
    UPDATE REVIEW  
    SET  
        Rating = p_NewRating,  
        Content = p_NewContent  
    WHERE ReviewID = p_ReviewID;  
  
    SELECT 'Cập nhật đánh giá thành công!' AS Message;  
END$$
```

Mô tả: Thủ tục sp_UpdateReview cho phép người dùng chỉnh sửa nội dung và điểm đánh giá mà họ đã đăng. Để đảm bảo tính công bằng và an ninh, thủ tục áp dụng các kiểm tra: Kiểm tra sự tồn tại và quyền sở hữu: Thủ tục xác thực ReviewID phải tồn tại. Quan trọng hơn, nó kiểm tra để đảm bảo rằng người dùng đang thực hiện hành động (p_BuyerID) chính là người đã tạo ra bài đánh giá gốc. Điều này ngăn chặn người dùng này sửa đổi đánh giá của người dùng



khác.

Ràng buộc về thời gian: Một quy tắc nghiệp vụ quan trọng được áp dụng là giới hạn thời gian cho phép chỉnh sửa. Thủ tục sẽ CHẶN việc cập nhật nếu bài đánh giá đã được đăng quá 7 ngày. Cơ chế này đảm bảo tính nhất quán của các đánh giá, tránh việc người dùng thay đổi nhận xét sau một thời gian dài sử dụng vì những lý do không còn liên quan đến trải nghiệm ban đầu.

Kiểm tra miền giá trị: Điểm đánh giá mới (p_NewRating) cũng được kiểm tra để đảm bảo nằm trong khoảng hợp lệ từ 1 đến 5.

Kiểm tra:

```
-- REVIEW INSERT --
DELETE FROM REVIEW WHERE BuyerID = 3 AND ProductID = 4;

-- BƯỚC 2: Chuẩn bị dữ liệu bằng chính thủ tục INSERT
-- Thủ tục `sp_AddReview` sẽ tự kiểm tra mọi điều kiện, đảm bảo REVIEW được tạo ra là hợp lệ.
-- (BuyerID=3, ProductID=4 là cặp hợp lệ và chưa có review)
CALL sp_AddReview(3, 4, 3, 'Nội dung ban đầu.');

-- Lấy ID của review vừa được tạo một cách chính xác.
SET @review_id_to_update = (SELECT ReviewID FROM review WHERE BuyerID = 3 AND ProductID = 4);

-- BƯỚC 3: Gọi thủ tục UPDATE và kiểm tra
-- Bây giờ, chúng ta chắc chắn đang cập nhật một review hợp lệ, do chính Buyer 3 sở hữu.
CALL sp_UpdateReview(@review_id_to_update, 3, 5, 'Nội dung đã được cập nhật thành công!');

-- Kiểm tra kết quả cuối cùng.
SELECT * FROM bt12_shopee.review WHERE ReviewID = @review_id_to_update;
```

123	23:54:32	DELETE FROM REVIEW WHERE BuyerID = 3 AND ProductID = 4	0 row(s) affected
124	23:54:38	DELETE FROM REVIEW WHERE BuyerID = 3 AND ProductID = 4	0 row(s) affected
125	23:54:42	SET @review_id_to_update = (SELECT ReviewID FROM review WHERE BuyerID = 3 AND ProductID = 4)	0 row(s) affected

Thủ tục Delete



```
-- REVIEW DELETE --
DELIMITER $$

CREATE PROCEDURE `sp_DeleteReview`(
    IN p_ReviewID INT,
    IN p_BuyerID INT -- Cần ID người mua để xác thực quyền
)
BEGIN
    DECLARE v_original_buyer_id INT;

    -- Validate 1: Kiểm tra sự tồn tại của Review và quyền sở hữu
    SELECT BuyerID INTO v_original_buyer_id
    FROM REVIEW WHERE ReviewID = p_ReviewID;

    IF v_original_buyer_id IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Đánh giá (ReviewID) không tồn tại.';
    END IF;

    IF v_original_buyer_id != p_BuyerID THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: Bạn không có quyền xóa đánh giá này.';
    END IF;

    -- Tác vụ phụ: Xóa các dữ liệu liên quan trước để tránh lỗi khóa ngoại
    -- (Giả sử các bảng REVIEW_IMAGES và REVIEW_VIDEO có ON DELETE CASCADE thì không cần bước này)
    DELETE FROM REVIEW_IMAGES WHERE ReviewID = p_ReviewID;
    DELETE FROM REVIEW_VIDEO WHERE ReviewID = p_ReviewID;

    -- Thực hiện DELETE
    DELETE FROM REVIEW WHERE ReviewID = p_ReviewID;

    SELECT 'Xóa đánh giá thành công!' AS Message;
END$$
DELIMITER ;
```

Mô tả: Thủ tục sp_DeleteReview xóa một bài đánh giá cùng toàn bộ media của bài đánh giá khi người xóa là chính chủ bài viết.

Mục đích: Cho phép người dùng kiểm soát nội dung họ đã tạo, xóa các đánh giá đăng nhầm, sai thông tin hoặc không còn muốn công khai. Điều này giúp bảo vệ quyền riêng tư và đảm bảo thông tin phản hồi trên hệ thống luôn chính xác.

Kiểm tra ràng buộc: chỉ cho phép xóa khi ReviewID tồn tại và BuyerID của người thực hiện thao tác trùng với BuyerID của người đã tạo review; hệ thống sẽ tự động chặn nếu ReviewID không tồn tại hoặc nếu người thao tác không phải chủ sở hữu review, nhằm ngăn chặn việc xóa trái phép. Trước khi xóa bản ghi REVIEW, thủ tục sẽ xóa toàn bộ REVIEW_IMAGES và REVIEW_VIDEO liên quan, đảm bảo không còn media mồ côi trong cơ sở dữ liệu.

Kiểm tra:



```
-- REVIEW DELETE --
-- Bước 1: Chuẩn bị dữ liệu - Tạo một bài đánh giá mới để có đối tượng để xóa.
-- (BuyerID=3, ProductID=4 là cặp hợp lệ và chưa có review)
CALL sp_AddReview(3, 4, 1, 'Nội dung này sẽ bị xóa.');

-- Lấy ID của review vừa tạo để thực hiện xóa.
SET @review_id_to_delete = (SELECT ReviewID FROM review WHERE BuyerID = 3 AND ProductID = 4);

-- Bước 2: Gọi thủ tục DELETE.
-- BuyerID = 3 đang xóa đánh giá của chính mình.
CALL sp_DeleteReview(@review_id_to_delete, 3);

-- Bước 3: Kiểm tra kết quả.
SELECT * FROM bt12_shopee.review WHERE ReviewID = @review_id_to_delete;
```

Result Grid						
	ReviewID	BuyerID	ProductID	Rating	Content	CreatedAt
*	NULL	NULL	NULL	NULL	NULL	NULL

Nhận xét: Lệnh SELECT cuối cùng trả về một bảng rỗng (không có dòng dữ liệu nào). Điều này chứng tỏ bản ghi đánh giá đã được xóa thành công khỏi hệ thống, xác nhận thủ tục DELETE hoạt động chính xác.

2.2.2 Trigger

2.2.2.1 Trigger kiểm tra ràng buộc nghiệp vụ

2.2.2.2 BẢNG ORDER – Ràng buộc: “Buyer có đơn hàng thì không được xóa Buyer” **Ràng buộc nghiệp vụ** Trong một hệ thống thương mại điện tử, thông tin lịch sử giao dịch của người mua (Buyer) có giá trị quan trọng đối với vận hành hệ thống, đối soát, phục vụ khách hàng, và lý do pháp lý. Vì vậy, khi một Buyer đã từng tạo đơn hàng, hệ thống không được phép xóa Buyer, nhằm đảm bảo tính toàn vẹn của dữ liệu và tránh thất thoát lịch sử mua bán.

Ràng buộc này không thể được kiểm soát bằng các cơ chế ràng buộc có sẵn như CHECK, UNIQUE hay FOREIGN KEY vì:

- **FOREIGN KEY:** chỉ có thể ngăn xóa Buyer nếu cấu hình ON DELETE RESTRICT. Tuy nhiên hệ thống hiện đang sử dụng ON DELETE CASCADE, nghĩa là nếu xóa Buyer thì toàn bộ Order liên quan cũng sẽ bị xóa theo → điều này vi phạm nghiệp vụ vì làm mất lịch sử giao dịch.
- **CHECK:** không thể truy vấn sang bảng khác để kiểm tra sự tồn tại đơn hàng.



- **UNIQUE / NOT NULL:** không liên quan, không thể dùng để kiểm soát ràng buộc dạng liên bảng.

Do vậy, **trigger** là cơ chế duy nhất cho phép hệ thống chủ động từ chối thao tác xóa Buyer khi Buyer vẫn còn các Order liên quan. **Các thao tác DML có thể gây vi phạm** Thao tác có nguy cơ trực tiếp vi phạm ràng buộc nghiệp vụ này là **DELETE** trên bảng Buyer. Khi hệ thống tiếp nhận yêu cầu xóa một Buyer, CSDL sẽ thực thi các ràng buộc toàn vẹn dựa trên cấu hình khóa ngoại. Tuy nhiên, trong mô hình hiện tại, khóa ngoại giữa Order và Buyer được thiết lập với **ON DELETE CASCADE**, nghĩa là:

- Khi BuyerID bị xóa, hệ thống coi Buyer đó không còn tồn tại trong hệ thống.
- Do cơ chế **CASCADE**, tất cả các Order liên quan cũng sẽ bị xóa tự động.
- Kết quả là toàn bộ lịch sử giao dịch của người mua này, bao gồm các thông tin về đơn hàng, giá trị thanh toán, vận đơn, đánh giá sản phẩm,... đều bị mất hoàn toàn.

Điều này mang đến nghiêm trọng với yêu cầu nghiệp vụ cốt lõi của một hệ thống thương mại điện tử, vốn cần duy trì lịch sử giao dịch đầy đủ vì:

- Là cơ sở để giải quyết khiếu nại, bảo hành, hoàn tiền.
- Cung cấp dữ liệu phân tích hành vi mua hàng và thống kê doanh thu.
- Đáp ứng các yêu cầu lưu trữ dữ liệu theo quy định pháp lý.
- Bảo đảm tính minh bạch và tin cậy của nền tảng.

Chính vì vậy, mặc dù ràng buộc khóa ngoại có thể xử lý xóa lan truyền về mặt kỹ thuật, hệ thống tuyệt đối không được phép để thao tác **DELETE Buyer** làm mất dữ liệu giao dịch. Do không thể thay đổi cấu trúc Foreign Key trong phạm vi bài tập, và cũng không thể sử dụng CHECK hoặc UNIQUE (vì không hỗ trợ truy vấn liên bảng), trigger trở thành cơ chế duy nhất cho phép hệ thống:

- Kiểm tra sự tồn tại của Order trước khi xóa.
- Từ chối thao tác DELETE nếu Buyer còn đơn hàng liên quan.

Nhờ trigger, mọi nỗ lực xóa Buyer đang sở hữu các Order đều bị ngăn chặn từ sớm, bảo toàn hoàn toàn lịch sử giao dịch của hệ thống.

Trigger kiểm tra ràng buộc Trigger trong file code:

```
DELIMITER $$  
CREATE TRIGGER TRG_Check_Buyer_Order_Total  
BEFORE DELETE ON Buyer  
FOR EACH ROW  
BEGIN  
    IF EXISTS (SELECT 1 FROM `Order` WHERE BuyerID = OLD.BuyerID) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Không thể xóa Buyer vì còn Order liên quan.';  
    END IF;  
END$$  
DELIMITER ;
```



Giải thích

- Trigger được kích hoạt trước khi thực hiện thao tác xóa Buyer, cho phép hệ thống kiểm tra trạng thái ràng buộc trước khi dữ liệu bị tác động. Trong quá trình này, trigger truy vấn bảng Order để xác định liệu Buyer đang bị xóa có tồn tại bất kỳ đơn hàng nào hay không.
 - Nếu BuyerID xuất hiện trong bảng Order, điều đó có nghĩa là người mua đã phát sinh giao dịch và có lịch sử mua hàng.
 - Khi đó, trigger lập tức từ chối thao tác xóa, ngăn chặn việc xóa dữ liệu gây mất toàn bộ thông tin đơn hàng liên quan.
- Cơ chế này đảm bảo rằng lịch sử giao dịch của khách hàng luôn được bảo tồn, dữ liệu trong hệ thống không bị mâu thuẫn, và các yêu cầu nghiệp vụ về lưu trữ thông tin mua bán được tuân thủ đầy đủ.

Kiểm tra

```
9
10 •  SELECT BuyerID, COUNT(*) AS TotalOrders
11   FROM `Order`
12  GROUP BY BuyerID;
13 •  DELETE FROM Buyer WHERE BuyerID = 2;
14

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
BuyerID | TotalOrders |
1 | 2 |
2 | 2 |
3 | 1 |

Result 3 ×
Output: 11:41:32 SELECT BuyerID, COUNT(*) AS TotalOrders FROM `Order` GROUP BY BuyerID LIMIT 0, 1000
Action Output | Message |
1 | 11:41:32 SELECT BuyerID, COUNT(*) AS TotalOrders FROM `Order` GROUP BY BuyerID LIMIT 0, 1000 | 3 row(s) returned
2 | 11:41:32 DELETE FROM Buyer WHERE BuyerID = 2 | Error Code: 1644. Không thể xóa Buyer vì còn Order liên quan.
```

2.2.2.3 BẢNG ORDERUNIT – Ràng buộc: “Chỉ được tạo Shipment nếu OrderUnit đang ở trạng thái ‘Processing’” Ràng buộc nghiệp vụ

Trong quy trình xử lý đơn hàng của một hệ thống thương mại điện tử, mỗi đơn hàng con (OrderUnit) phải tuân theo chu trình trạng thái chặt chẽ: Processing → Shipping → Completed hoặc Cancelled.

Theo yêu cầu nghiệp vụ, việc tạo vận đơn (Shipment) chỉ được phép xảy ra khi OrderUnit vẫn đang ở trạng thái “Processing” – đây là giai đoạn đơn hàng đang được cửa hàng chuẩn bị, xác nhận và đóng gói.

Khi OrderUnit đã chuyển sang các trạng thái khác như Shipping, Completed hoặc Cancelled, hệ thống tuyệt đối không được phép tạo vận đơn mới, nhằm tránh tạo ra các bản ghi vận chuyển sai, trùng, hoặc không phù hợp với tiến trình xử lý thực tế của đơn hàng.

Ràng buộc này không thể được triển khai bằng CHECK (vì CHECK không truy vấn bảng khác), và cũng không thể dùng FOREIGN KEY (vì ràng buộc này phụ thuộc vào giá trị động của một thuộc tính trong OrderUnit). Do vậy, trigger trở thành cơ chế bắt buộc để đảm bảo ràng buộc được kiểm tra đúng thời điểm.



Các thao tác DML có thể gây vi phạm

Thao tác có khả năng dẫn đến vi phạm ràng buộc là: INSERT trên bảng Shipment.

Khi người dùng thực hiện thao tác thêm mới một Shipment, hệ thống sẽ phải kiểm tra trạng thái hiện tại của OrderUnit tương ứng. Nếu OrderUnit đã chuyển sang các trạng thái Shipping, Completed hoặc Cancelled, việc tạo thêm vận đơn là hoàn toàn không hợp lệ. Trong các trường hợp này, vận đơn mới sẽ phá vỡ logic xử lý đơn hàng, gây nhầm lẫn trong quá trình vận chuyển và làm sai lệch lịch sử nghiệp vụ.

Vì vậy, cần có trigger để ngăn ngừa ngay tại thời điểm INSERT, nhằm đảm bảo rằng không có Shipment nào được tạo ra cho những OrderUnit đã vượt qua giai đoạn xử lý (Processing).

Trigger kiểm tra ràng buộc

Trigger trong file code:

```
DROP TRIGGER IF EXISTS before_shipment_insert;
DELIMITER $$ 
CREATE TRIGGER before_shipment_insert BEFORE INSERT ON SHIPMENT FOR EACH ROW
BEGIN
    DECLARE order_unit_status VARCHAR(50);
    SELECT Status INTO order_unit_status FROM orderunit WHERE UnitID = NEW.UnitID;
    IF order_unit_status != 'Processing' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi Logic: Chỉ có thể tạo vận đơn khi đơn hàng đang ở trạng thái "Processing".';
    END IF;
END$$
DELIMITER ;
```

Giải thích Trigger này được kích hoạt trước khi hệ thống chèn một bản ghi Shipment mới, cho phép kiểm tra trạng thái OrderUnit ngay tại thời điểm phát sinh thao tác. Nếu trạng thái của OrderUnit khác “Processing”, trigger sẽ lập tức chặn thao tác INSERT và phát sinh thông báo lỗi nghiệp vụ.

Cơ chế này đảm bảo rằng:

- Không phát sinh Shipment ở các trạng thái không phù hợp.
- Chu trình xử lý đơn hàng không bị phá vỡ hoặc trùng lặp.
- Thông tin vận chuyển luôn phản ánh đúng tiến trình xử lý thực tế.
- Toàn bộ hệ thống duy trì được tính chính xác, minh bạch và toàn vẹn dữ liệu.

Kiểm tra



```
21 •  SELECT UnitID, Status
22   FROM OrderUnit;
23
24 •  INSERT INTO Shipment (UnitID, TrackingNo, Status)
25   VALUES (8, CONCAT('SHIP-OK-', UNIX_TIMESTAMP()), 'Preparing');
26
27 •  INSERT INTO Shipment (UnitID, TrackingNo, Status)
28   VALUES (4, CONCAT('SHIP-ERR-', UNIX_TIMESTAMP()), 'Preparing')
29
```

Result Grid | Filter Rows: | Edit | Export/Import: | Wrap Cell Content: |

UnitID	Status
4	Completed
5	Canceled
6	Processing
7	Processing
8	Processing

OrderUnit 11 | X

Output: Action Output

#	Time	Action	Message
1	11:54:44	SELECT UnitID, Status FROM OrderUnit LIMIT 0, 1000	10 row(s) returned
2	11:54:44	INSERT INTO Shipment (UnitID, TrackingNo, Status) VALUES (8, CONCAT('SHIP-OK-', UNIX_TIMESTAMP()), 'Preparing')	1 row(s) affected
3	11:54:44	INSERT INTO Shipment (UnitID, TrackingNo, Status) VALUES (4, CONCAT('SHIP-ERR-', UNIX_TIMESTAMP()), 'Preparing')	Error Code: 1644. Lỗi Logic: Chỉ có thể tạo và duy trì một biến thể duy nhất cho một sản phẩm.

2.2.2.4 BẢNG PRODUCT_VARIANT – Ràng buộc: “Không được chuyển biến thể cuối cùng sang sản phẩm khác” Ràng buộc nghiệp vụ

Trong hệ thống thương mại điện tử, mỗi sản phẩm (Product) phải có ít nhất một biến thể (ProductVariant) để có thể tồn tại và hiển thị trên gian hàng. Mỗi biến thể đại diện cho một lựa chọn cụ thể về màu sắc, kích cỡ, dung lượng..., vì vậy việc đảm bảo một sản phẩm luôn có tối thiểu một biến thể là điều kiện thiết yếu.

Nếu hệ thống cho phép chuyển biến thể cuối cùng của một sản phẩm sang sản phẩm khác (tức là thay đổi ProductID của biến thể), sản phẩm ban đầu sẽ không còn bất cứ biến thể nào, khiến sản phẩm trở thành một thực thể rỗng, vi phạm logic nghiệp vụ và gây lỗi trong quá trình hiển thị cũng như mua bán.

Ràng buộc này:

- Không thể được kiểm tra bằng CHECK, vì CHECK không thể truy vấn số lượng biến thể của sản phẩm (không thể COUNT từ một bảng khác).
- Không thể dùng FOREIGN KEY, vì khóa ngoại chỉ đảm bảo tham chiếu hợp lệ, không kiểm soát số lượng biến thể tối thiểu.

→ Do đó, trigger là cơ chế duy nhất để kiểm soát nghiệp vụ này.

Các thao tác DML có thể gây vi phạm

Thao tác có khả năng dẫn đến vi phạm ràng buộc là UPDATE trên bảng Product_Variant.

Cụ thể, khi người dùng thực hiện cập nhật để thay đổi ProductID của một biến thể sang thuộc về một sản phẩm khác, hệ thống cần phải kiểm tra xem biến thể đó có phải là biến thể duy nhất còn lại của sản phẩm cũ hay không.

Nếu biến thể đang cập nhật là biến thể cuối cùng của sản phẩm ban đầu, thì việc di chuyển biến thể sang một sản phẩm khác sẽ khiến sản phẩm cũ không còn bất kỳ biến thể nào, dẫn đến tình trạng sản phẩm “rỗng”. Đây là lỗi nghiêm trọng trong nghiệp vụ vì một sản phẩm trong hệ thống thương mại điện tử luôn bắt buộc phải có ít nhất một biến thể hợp lệ để tồn tại và hiển thị.

Do đó, thao tác UPDATE làm thay đổi ProductID của biến thể cuối cùng chính là thao tác có nguy cơ trực tiếp tạo ra lỗi nghiệp vụ, và cần được hệ thống kiểm soát bằng trigger.



Trigger kiểm tra ràng buộc

Trigger trong code:

```
CREATE TRIGGER trg_prevent_last_variant_update
BEFORE UPDATE ON product_variant
FOR EACH ROW
BEGIN
    DECLARE old_product_variant_count INT;

    -- Chỉ kiểm tra khi ProductID thay đổi
    IF OLD.ProductID != NEW.ProductID THEN
        SELECT COUNT(*) INTO old_product_variant_count
        FROM product_variant
        WHERE ProductID = OLD.ProductID
        AND ProductVariantID != OLD.ProductVariantID;

        IF old_product_variant_count = 0 THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Lỗi: Không thể di chuyển biến thể cuối cùng sang sản phẩm khác. Mỗi sản phẩm phải có ít nhất 1 biến thể.';
        END IF;
    END IF;
END$$
```

Giải thích

Trigger này được kích hoạt trước khi hệ thống thực thi câu lệnh UPDATE trên bảng **ProductVariant**. Mục tiêu của trigger là kiểm tra xem biến thể đang được cập nhật có phải là biến thể cuối cùng của sản phẩm cũ hay không, nhằm ngăn ngừa tình trạng sản phẩm không còn bất kỳ biến thể nào — một lỗi nghiêm trọng trong nghiệp vụ quản lý sản phẩm.

Cơ chế xử lý được triển khai theo các bước sau:

- **Bước 1:** Xác định việc thay đổi ProductID. Trigger trước tiên kiểm tra xem thao tác UPDATE có làm thay đổi ProductID hay không.
 - Nếu ProductID giữ nguyên, việc cập nhật không ảnh hưởng đến số lượng biến thể, nên trigger cho phép thực thi.
 - Nếu ProductID thay đổi, hệ thống phải kiểm tra số lượng biến thể còn lại của sản phẩm cũ.
- **Bước 2:** Dếm số biến thể còn lại của sản phẩm ban đầu. Trigger thực hiện truy vấn để đếm tất cả biến thể thuộc cùng ProductID cũ, ngoại trừ biến thể đang được cập nhật.
- **Bước 3:** Phát hiện tình huống vi phạm.
 - Nếu số lượng biến thể còn lại lớn hơn 0, sản phẩm vẫn còn biến thể khác và thao tác cập nhật là hợp lệ.
 - Nếu kết quả bằng 0, có nghĩa là biến thể đó là biến thể cuối cùng của sản phẩm cũ.
- **Bước 4:** Từ chối thao tác nếu vi phạm ràng buộc. Khi phát hiện đây là biến thể cuối cùng, trigger lập tức chặn thao tác UPDATE và phát sinh lỗi nghiệp vụ, ngăn không cho chuyển biến thể này sang sản phẩm khác.

Nhờ cơ chế kiểm soát này:

- Hệ thống không bao giờ để sản phẩm rơi vào trạng thái không có biến thể.
- Mỗi sản phẩm luôn đảm bảo có ít nhất một variant hợp lệ, đúng theo yêu cầu nghiệp vụ.
- Các sai sót trong quá trình cập nhật được ngăn chặn ngay trước khi dữ liệu được ghi vào CSDL, giúp duy trì tính toàn vẹn và ổn định của dữ liệu sản phẩm.



Kiểm tra

```
38 *  SELECT ProductID, COUNT(*) AS VariantCount
39   FROM product_variant
40  GROUP BY ProductID
41  HAVING VariantCount = 1
42  -- 2. Lấy VariantID tương ứng để test
43 *  SELECT ProductVariantID, ProductID, Price, Stock
44   FROM product_variant
45  WHERE ProductID = 3;
46  -- 3. Chuyển biến thể cuối cùng sang sản phẩm khác + Lỗi
47 *  UPDATE product_variant
48    SET ProductID = 1      -- ProductID khác
49  WHERE ProductVariantID = 7; -- VariantID duy nhất
50  -- 4. Chuyển biến thể hợp lệ (sản phẩm có nhiều variant)
51 *  UPDATE product_variant
52    SET ProductID = 5
53  WHERE ProductVariantID = 15; -- variant KHÔNG phải cuối cùng
54
```

Output

Action	Time	Action	Message
①	1 12:19:29	UPDATE product_variant SET ProductID = 1 -- ProductID khác WHERE ProductVariantID = 7	Error Code: 1644. Lỗi: Không thể di chuyển biến thể cuối cùng sang sản phẩm khác. Mỗi sản phẩm p...
②	2 12:19:35	UPDATE product_variant SET ProductID = 5 WHERE ProductVariantID = 15	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0

2.2.2.5 BẢNG REVIEW – Ràng buộc: “Không được chỉnh sửa đánh giá sau 7 ngày kể từ ngày tạo” Ràng buộc nghiệp vụ

Trong các nền tảng thương mại điện tử, đánh giá sản phẩm (Review) là một phần quan trọng quyết định trải nghiệm người dùng và mức độ tin cậy của sản phẩm. Mỗi đánh giá, một khi đã được người dùng đăng tải, cần được duy trì tính khách quan và phản ánh đúng trải nghiệm của người mua tại thời điểm giao dịch.

Do đó, hệ thống thường quy định một giới hạn thời gian cho phép chỉnh sửa đánh giá. Trong mô hình này, người dùng chỉ được phép chỉnh sửa Review trong vòng 7 ngày kể từ thời điểm tạo (CreatedAt). Sau 7 ngày, đánh giá được xem là đã ổn định và không được phép thay đổi nữa.

Ràng buộc này không thể áp dụng bằng CHECK hoặc FOREIGN KEY, vì:

- CHECK: không thể truy vấn dữ liệu cũ (OLD.CreatedAt) để tính số ngày đã trôi qua.
- FOREIGN KEY: không liên quan, vì không phải quan hệ giữa các bảng.
- Ứng dụng (frontend/back-end) có thể kiểm tra, nhưng không đáng tin vì có thể bị bỏ qua hoặc chỉnh sửa thủ công.

Vì vậy, trigger là phương thức duy nhất giúp đảm bảo dữ liệu luôn tuân thủ đúng nghiệp vụ, chặn trực tiếp mọi UPDATE không hợp lệ ngay tại tầng CSDL.

Các thao tác DML có khả năng vi phạm

- Chỉ duy nhất thao tác UPDATE Review gây vi phạm ràng buộc.
- Nếu thời điểm UPDATE cách CreatedAt hơn 7 ngày, người dùng đang cố sửa một đánh giá đã quá hạn cho phép → trái với nghiệp vụ hệ thống.
- Không có thao tác INSERT hoặc DELETE nào gây ra lỗi trong ràng buộc này.

Trigger kiểm tra ràng buộc



```
DELIMITER $$  
CREATE TRIGGER before_review_update BEFORE UPDATE ON REVIEW FOR EACH ROW  
BEGIN  
IF OLD.CreatedAt < NOW() - INTERVAL 7 DAY THEN  
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Không thể chỉnh sửa đánh giá sau 7 ngày.';  
END IF;  
END$$  
DELIMITER ;
```

Giải thích

Trigger được kích hoạt trước khi hệ thống thực hiện câu lệnh UPDATE, giúp đảm bảo rằng mọi thay đổi đối với nội dung review đều được kiểm tra nghiêm ngặt theo đúng nghiệp vụ.

Khi người dùng cố cập nhật đánh giá, trigger sẽ:

- Lấy thời điểm tạo ban đầu (OLD.CreatedAt).
- Tính khoảng cách giữa thời điểm hiện tại và CreatedAt.
- Nếu số ngày vượt quá 7, trigger lập tức báo lỗi và chặn thao tác.

Cơ chế này đảm bảo:

- Tính khách quan của đánh giá không bị thay đổi theo ý muốn sau thời gian dài.
- Lịch sử phản hồi của người mua được bảo toàn.
- Tính minh bạch của hệ thống thương mại điện tử được duy trì.

Kiểm tra



The screenshot shows a MySQL Workbench interface with two tabs: 'TEST CASE 1: CHO PHÉP SỬA REVIEW TRONG VÒNG 7 NGÀY' and 'TEST CASE 2: KHÔNG CHO PHÉP SỬA REVIEW SAU 7 NGÀY'. The code in both tabs is identical, demonstrating how to insert a review, update its content, and then attempt to update it again after 7 days.

```
-- TEST CASE 1: CHO PHÉP SỬA REVIEW TRONG VÒNG 7 NGÀY
-- Tạo review mới (CreatedAt = NOW() - 2 DAY -> hợp lệ)
18 • INSERT INTO REVIEW (BuyerID, ProductID, Rating, Content, CreatedAt)
VALUES (1, 1, 5, 'Review được tạo 2 ngày trước - hợp lệ', NOW() - INTERVAL 2 DAY);
-- Lấy ID review vừa tạo
21 • SET @review_ok := LAST_INSERT_ID();
-- Test sửa review -> phải THÀNH CÔNG
22 • UPDATE REVIEW
SET Content = 'Sửa trong 7 ngày - Kỳ vọng: THÀNH CÔNG'
WHERE ReviewID = @review_ok;
-- TEST CASE 2: KHÔNG CHO PHÉP SỬA REVIEW SAU 7 NGÀY
-- Tạo review mới (CreatedAt = NOW() - 10 DAY -> quá hạn)
30 • INSERT INTO REVIEW (BuyerID, ProductID, Rating, Content, CreatedAt)
VALUES (1, 1, 4, 'Review được tạo 10 ngày trước - quá hạn', NOW() - INTERVAL 10 DAY);
-- Lấy ID review vừa tạo
33 • SET @review_fail := LAST_INSERT_ID();
-- Test sửa review -> phải BỊ CHẶN bởi trigger
34 • UPDATE REVIEW
SET Content = 'Sửa sau 7 ngày - Kỳ vọng: BẢO LỘI'
WHERE ReviewID = @review_fail;
--
```

The 'Output' tab shows the execution results:

Time	Action	Message
106 21:54:40	USE BTL2_SHOPEE	0 row(s) affected
107 21:54:40	INSERT INTO REVIEW (BuyerID, ProductID, Rating, Content, CreatedAt) VALUES (1, 1, 5, 'Review được tạo...', NOW() - INTERVAL 2 DAY)	1 row(s) affected
108 21:54:40	SET @review_ok := LAST_INSERT_ID()	0 row(s) affected
109 21:54:40	UPDATE REVIEW SET Content = 'Sửa trong 7 ngày - Kỳ vọng: THÀNH CÔNG' WHERE ReviewID = @review...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
110 21:54:40	INSERT INTO REVIEW (BuyerID, ProductID, Rating, Content, CreatedAt) VALUES (1, 1, 4, 'Review được tạo...', NOW() - INTERVAL 10 DAY)	1 row(s) affected
111 21:54:40	SET @review_fail := LAST_INSERT_ID()	0 row(s) affected
112 21:54:40	UPDATE REVIEW SET Content = 'Sửa sau 7 ngày - Kỳ vọng: BẢO LỘI' WHERE ReviewID = @review_fail	Error Code: 1644: Không thể chỉnh sửa đánh giá sau 7 ngày

2.2.2.6 BẢNG CARTITEM – Ràng buộc: “Không được thêm hoặc cập nhật số lượng vượt quá tồn kho” Ràng buộc nghiệp vụ

Trong giỏ hàng, mỗi CartItem lưu số lượng người dùng muốn mua. Để đảm bảo tính chính xác khi đặt hàng và tránh lỗi trong bước thanh toán, số lượng này không được phép vượt quá số lượng tồn kho thực tế của biến thể sản phẩm (ProductVariant.Stock).

Ràng buộc này không thể dùng CHECK, vì CHECK không thể truy vấn bảng ProductVariant. Cũng không thể dùng khóa ngoại vì khóa ngoại chỉ quản lý tính tồn tại của ProductVariantID, không kiểm soát giá trị Stock.

Do đó, trigger là công cụ duy nhất để:

- Kiểm tra tồn kho tại thời điểm thêm/cập nhật giỏ hàng.
- Ngăn chặn người dùng nhập số lượng vượt quá giá trị thật.
- Duy trì tính toàn vẹn dữ liệu trong suốt quy trình mua hàng.

Các thao tác DML gây vi phạm

Hai thao tác có thể làm số lượng vượt tồn kho:

- INSERT CartItem → Khi thêm mới sản phẩm vào giỏ.
- UPDATE CartItem → Khi tăng số lượng sản phẩm đã có.

Cả hai đều có nguy cơ gây sai lệch so với tồn kho thực tế.

Trigger kiểm tra ràng buộc



```
/* ----- BEFORE INSERT: kiểm tra tồn kho ----- */
CREATE TRIGGER trg_cartitem_before_insert
BEFORE INSERT ON CartItem
FOR EACH ROW
BEGIN
    DECLARE available_stock INT;

    SELECT Stock INTO available_stock
    FROM product_variant
    WHERE ProductVariantID = NEW.ProductVariantID;

    IF available_stock IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Lỗi: ProductVariantID không tồn tại.';
    END IF;

    IF NEW.Quantity > available_stock THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Số lượng vượt quá tồn kho.';
    END IF;
END$$
```



```
/* ----- BEFORE UPDATE: kiểm tra tồn kho ----- */
CREATE TRIGGER trg_cartitem_before_update
BEFORE UPDATE ON CartItem
FOR EACH ROW
BEGIN
    DECLARE available_stock INT;

    SELECT Stock INTO available_stock
    FROM product_variant
    WHERE ProductVariantID = NEW.ProductVariantID;

    IF available_stock IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Lỗi: ProductVariantID không tồn tại.';
    END IF;

    IF NEW.Quantity > available_stock THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Không thể cập nhật số lượng vượt tồn kho.';
    END IF;
END$$
```

Giải thích

Hai trigger này đảm bảo mọi thao tác thay đổi số lượng sản phẩm trong giỏ hàng đều được kiểm soát.

Trigger hoạt động theo quy trình:

- Lấy tồn kho thật từ bảng ProductVariant.
- So sánh với số lượng người dùng yêu cầu (NEW.Quantity).
- Nếu vượt tồn kho → chặn thao tác ngay lập tức.

Cơ chế này giúp:

- Ngăn tình trạng giỏ hàng bị sai lệch so với tồn kho thật.
- Dảm bảo quá trình tạo đơn hàng (Order) không gặp lỗi.
- Giữ cho hệ thống vận hành ổn định và tránh tranh chấp với khách hàng.

Kiểm tra



```
-- =====
-- ↗ TEST 1: INSERT HỢP LỆ (KHÔNG LỖI)
-- =====
-- Quantity = 5 < Stock 10 → Hợp lệ
● INSERT INTO CartItem (CartID, ProductVariantID, SKU, Quantity)
VALUES (@TestCartID, @TestVariantID, 'SKU-TEST-1', 5);
-- =====
-- ↗ TEST 2: INSERT LỖI - VƯỢT TỒN KHO
-- =====
-- Quantity = 20 > Stock 10 → Trigger chặn
● INSERT INTO CartItem (CartID, ProductVariantID, SKU, Quantity)
VALUES (@TestCartID, @TestVariantID, 'SKU-TEST-2', 20);
-- "Số lượng vượt quá tồn kho."
-- =====
-- ↗ TEST 3: UPDATE HỢP LỆ (KHÔNG LỖI)
-- =====
-- Tăng lên Quantity = 8 < Stock 10 → OK
● UPDATE CartItem
SET Quantity = 8
WHERE CartID = @TestCartID AND ProductVariantID = @TestVariantID;

33 -- =====
34 -- ↗ TEST 4: UPDATE LỖI - VƯỢT TỒN KHO
35 -- =====
36 -- Update Quantity = 15 > Stock 10 → Trigger chặn
37 ● UPDATE CartItem
38 SET Quantity = 15
39 WHERE CartID = @TestCartID AND ProductVariantID = @TestVariantID;
40 -- Không thể cập nhật số lượng vượt tồn kho.
41 -- =====
42 -- ↗ TEST 5: UPDATE ProductVariantID SANG ID KHÔNG TỒN TẠI
43 -- =====
44 ● UPDATE CartItem
45 SET ProductVariantID = 999999
46 WHERE CartID = @TestCartID AND ProductVariantID = @TestVariantID;
47 -- Lỗi: ProductVariantID không tồn tại.
48
```

Output		
	Action	Time
109	INSERT INTO CartItem (CartID, ProductVariantID, SKU, Quantity) VALUES (@TestCartID, @TestVariantID, 'SKU-TEST-1', 5)	01:51:40
110	INSERT INTO CartItem (CartID, ProductVariantID, SKU, Quantity) VALUES (@TestCartID, @TestVariantID, 'SKU-TEST-2', 20)	01:51:40
111	UPDATE CartItem SET Quantity = 8 WHERE CartID = @TestCartID AND ProductVariantID = @TestVariantID	01:51:55
112	UPDATE CartItem SET Quantity = 15 WHERE CartID = @TestCartID AND ProductVariantID = @TestVariantID	01:52:01
113	UPDATE CartItem SET ProductVariantID = 999999 WHERE CartID = @TestCartID AND ProductVariantID = @TestVariantID	01:52:05

2.2.2.7 BẢNG SHIPMENT – Ràng buộc: “Chỉ được tạo Shipment nếu OrderUnit đang ở trạng thái Processing” Ràng buộc nghiệp vụ



Mỗi Shipment tương ứng với một OrderUnit. Trong hệ thống thương mại điện tử, quy trình trạng thái của OrderUnit phải được tuân thủ chặt chẽ:

- Processing (đang chuẩn bị hàng) → Mới được phép tạo vận đơn (Shipment).
- Shipping / Completed / Cancelled → Không được tạo thêm Shipment.

Nếu hệ thống cho phép tạo vận đơn khi OrderUnit không nằm trong trạng thái Processing, điều này sẽ gây:

- Sai lệch về quy trình vận hành.
- Phát sinh vận đơn dư thừa, sai dữ liệu.
- Gây lỗi hậu vận (COD, giao nhận, hủy đơn, hoàn tiền...).

Ràng buộc này:

- Không dùng CHECK được (không truy vấn bảng OrderUnit).
 - Không dùng FOREIGN KEY được (không kiểm soát trạng thái động).
- Trigger là lựa chọn bắt buộc.

Các thao tác DML gây vi phạm

- Chỉ thao tác INSERT Shipment gây nguy cơ vi phạm nghiệp vụ.
- Khi thêm Shipment mới, hệ thống bắt buộc phải kiểm tra trạng thái hiện tại của OrderUnit tương ứng.

Trigger kiểm tra ràng buộc

```
DELIMITER $$  
CREATE TRIGGER before_shipment_insert BEFORE INSERT ON SHIPMENT FOR EACH ROW  
BEGIN  
    DECLARE order_unit_status VARCHAR(50);  
    SELECT Status INTO order_unit_status FROM orderunit WHERE UnitID = NEW.UnitID;  
    IF order_unit_status != 'Processing' THEN  
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi Logic: Chỉ có thể tạo vận đơn khi đơn hàng đang ở trạng thái "Processing".';  
    END IF;  
END$$  
DELIMITER ;
```

Giải thích

Trigger này được kích hoạt ngay trước khi hệ thống thêm một vận đơn mới. Quy trình hoạt động:

- Lấy trạng thái hiện tại của OrderUnit.
- So sánh với trạng thái hợp lệ duy nhất ("Processing").
- Nếu khác → hệ thống lập tức từ chối thao tác INSERT.

Trigger đảm bảo:



- Không có Shipment nào được tạo sai quy trình.
- Dữ liệu vận hành logistics luôn chính xác.
- Tránh được lỗi nghiêm trọng trong hệ thống giao hàng và đối soát.

Kiểm tra

```
40 -- =====
41 -- TEST TRƯỜNG HỢP ĐÚNG (Không báo lỗi)
42 --
43 * INSERT INTO SHIPMENT (UnitID, TrackingNo, Status, Timeline, EstimatedDelivery)
44   VALUES (@ValidUnit, CONCAT('TRACK-', RAND()), 'Preparing', '{"created":"now"}', CURDATE() + INTERVAL 3 DAY);
45 -- Kiểm tra shipment được thêm
46 * SELECT * FROM SHIPMENT WHERE UnitID = @ValidUnit;
47 --
48 -- TEST TRƯỜNG HỢP LỖI (Trigger phải chặn)
49 --
50 * INSERT INTO SHIPMENT (UnitID, TrackingNo, Status, Timeline, EstimatedDelivery)
51   VALUES (@InvalidUnit, CONCAT('TRACK-', RAND()), 'Preparing', '{"created":"now"}', CURDATE() + INTERVAL 3 DAY);
52
```

Action Output		
#	Time	Action
144	01/44/52	SELECT -- Test 1: Thành công vì Unit ở trạng thái Processing -- AS TEST LIMIT 0, 1000
145	01/44/52	INSERT INTO SHIPMENT (UnitID, TrackingNo, Status, Timeline, EstimatedDelivery) VALUES (@ValidUnit, C...
146	01/44/52	SELECT * FROM SHIPMENT WHERE UnitID = @ValidUnit LIMIT 0, 1000
147	01/44/52	SELECT -- Test 2: Lỗi vì Unit ở trạng thái Shipping -- AS TEST LIMIT 0, 1000
148	01/44/52	INSERT INTO SHIPMENT (UnitID, TrackingNo, Status, Timeline, EstimatedDelivery) VALUES (@InvalidUnit, ... Error Code: 1644. Lỗi Logic: Chỉ có thể tạo vận đơn khi đơn hàng đang ở trạng thái "Processing".

2.2.3 2.2.2 Trigger tính toán thuộc tính dãn xuất



2.2.4 Trigger tính toán thuộc tính dẫn xuất

2.2.4.1 A – Hệ thống tính TotalPrice cho Order 1. Giới thiệu

Hệ thống được thiết kế để tự động tính toán và cập nhật thuộc tính dẫn xuất TotalPrice (tổng tiền đơn hàng) trong cơ sở dữ liệu, đảm bảo tính chính xác và nhất quán dữ liệu.

2. Thuộc tính dẫn xuất được chọn

Hệ thống sử dụng kết hợp Stored Procedure và Trigger theo mô hình “Helper & Listeners”.
A. Stored Procedure: sp_UpdateOrderTotal (Helper)

```
-- REVIEW DELETE --
-- Bước 1: Chuẩn bị dữ liệu - Tạo một bài đánh giá mới để có đối tượng để xóa.
-- (BuyerID=3, ProductID=4 là cặp hợp lệ và chưa có review)
CALL sp_AddReview(3, 4, 1, 'Nội dung này sẽ bị xóa.');

-- Lấy ID của review vừa tạo để thực hiện xóa.
SET @review_id_to_delete = (SELECT ReviewID FROM review WHERE BuyerID = 3 AND ProductID = 4);

-- Bước 2: Gọi thủ tục DELETE.
-- BuyerID = 3 đang xóa đánh giá của chính mình.
CALL sp_DeleteReview(@review_id_to_delete, 3);

-- Bước 3: Kiểm tra kết quả.
SELECT * FROM btl2_shopee.review WHERE ReviewID = @review_id_to_delete;
```

Result Grid						
	ReviewID	BuyerID	ProductID	Rating	Content	CreatedAt
*	NULL	NULL	NULL	NULL	NULL	NULL

Module tính toán trung tâm

Đây là module tính toán trung tâm, đóng vai trò như một hàm tiện ích được tái sử dụng nhiều lần.

Đầu vào: p_OrderID (Mã đơn hàng cần tính lại).

Nguyên tắc tính toán:

- **TotalPrice** được tính dựa trên: Tổng tiền sản phẩm có trong ORDER_LINE và tổng phí vận chuyển lưu trong OrderUnit.
- Công thức tổng quát:

$$\text{TotalPrice} = \text{Tổng tiền sản phẩm} + \text{Tổng phí vận chuyển}$$



- Trong đó:

$$\text{Tổng tiền sản phẩm} = \sum(Quantity \times UnitPrice \times (1 - \frac{Discount}{100}))$$

$$\text{Tổng phí vận chuyển} = \sum(ShippingFee)$$

3. Các thao tác DML ảnh hưởng đến thuộc tính dẫn xuất

3.1. Trên bảng ORDER_LINE

- INSERT: Thêm sản phẩm vào đơn hàng → thay đổi tổng tiền sản phẩm.

```
-- -----
-- Trigger sau khi THÊM sản phẩm vào chi tiết đơn
DROP TRIGGER IF EXISTS trg_OrderLine_AfterInsert $$;
CREATE TRIGGER trg_OrderLine_AfterInsert
AFTER INSERT ON ORDER_LINE
FOR EACH ROW
BEGIN
    DECLARE v_OrderID INT;
    -- Lấy OrderID từ UnitID của dòng vừa thêm
    SELECT OrderID INTO v_OrderID FROM OrderUnit WHERE UnitID = NEW.UnitID;

    -- Gọi procedure tính lại tiền
    CALL sp_UpdateOrderTotal(v_OrderID);
END $$;
```

- UPDATE: Sửa số lượng, đơn giá, discount → thay đổi tổng tiền sản phẩm.

```
-- Trigger sau khi SỬA sản phẩm trong chi tiết đơn
DROP TRIGGER IF EXISTS trg_OrderLine_AfterUpdate $$;
CREATE TRIGGER trg_OrderLine_AfterUpdate
AFTER UPDATE ON ORDER_LINE
FOR EACH ROW
BEGIN
    DECLARE v_OrderID INT;
    -- Lấy OrderID (giả sử UnitID không đổi, nếu UnitID đổi cần update cả đơn cũ)
    SELECT OrderID INTO v_OrderID FROM OrderUnit WHERE UnitID = NEW.UnitID;

    CALL sp_UpdateOrderTotal(v_OrderID);

    -- Trường hợp đặc biệt: Nếu chuyển line sang Unit của Order khác (hiếm gặp nhưng cần đề phòng)
    IF OLD.UnitID <> NEW.UnitID THEN
        SELECT OrderID INTO v_OrderID FROM OrderUnit WHERE UnitID = OLD.UnitID;
        CALL sp_UpdateOrderTotal(v_OrderID);
    END IF;
END $$;
```

- DELETE: Xóa sản phẩm khỏi đơn hàng → thay đổi tổng tiền sản phẩm.



```
-- Trigger sau khi XÓA sản phẩm khỏi chi tiết đơn
DROP TRIGGER IF EXISTS trg_OrderLine_AfterDelete $$

CREATE TRIGGER trg_OrderLine_AfterDelete
AFTER DELETE ON ORDER_LINE
FOR EACH ROW
BEGIN
    DECLARE v_OrderID INT;
    SELECT OrderID INTO v_OrderID FROM OrderUnit WHERE UnitID = OLD.UnitID;

    CALL sp_UpdateOrderTotal(v_OrderID);
END $$
```

3.2. Trên bảng OrderUnit

- UPDATE: Thay đổi ShippingFee hoặc OrderID → thay đổi phí vận chuyển.

```
-- 3. TRIGGER TRÊN BẢNG ORDER_UNIT (Xử lý thay đổi phí ship)
-- -----
' DROP TRIGGER IF EXISTS trg_OrderUnit_AfterUpdate $$

' CREATE TRIGGER trg_OrderUnit_AfterUpdate
AFTER UPDATE ON OrderUnit
FOR EACH ROW
BEGIN
    -- Chi tính lại nếu phí ship thay đổi hoặc chuyển đơn sang Order khác
    IF OLD.ShippingFee <> NEW.ShippingFee OR OLD.OrderID <> NEW.OrderID THEN
        CALL sp_UpdateOrderTotal(NEW.OrderID);

        -- Nếu đổi OrderID (hiếm), update cả Order cũ
        IF OLD.OrderID <> NEW.OrderID THEN
            CALL sp_UpdateOrderTotal(OLD.OrderID);
        END IF;
    END IF;
END $$

DELIMITER ;
```

Kiểm tra

Giá tiền hiện tại OrderID = 1 là **165000**.

```
17    -- Mua thêm 1 sản phẩm (ProductVariantID=2) giá 159k
18 •  INSERT INTO ORDER_LINE (UnitID, ProductVariantID, UnitPrice, Discount, SKU, Quantity)
19     VALUES (1, 2, 159000, 0, 'TEST-ADD', 1);
20
21    -- Kết quả: TotalPrice phải TĂNG thêm 159,000
22 •  SELECT OrderID, TotalPrice AS '2_Sau_Khi_Them_Hang' FROM `Order` WHERE OrderID = 1;
```

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
OrderID	2_Sau_Khi_Them_Hang				
1	324000.00				



Sau khi thêm 1 sản phẩm với giá tiền **159**, lúc này TotalPrice sẽ tăng lên là **324000**.

```
-- Trigger sau khi XÓA sản phẩm khỏi chi tiết đơn
DROP TRIGGER IF EXISTS trg_OrderLine_AfterDelete $$

CREATE TRIGGER trg_OrderLine_AfterDelete
AFTER DELETE ON ORDER_LINE
FOR EACH ROW
BEGIN
    DECLARE v_OrderID INT;
    SELECT OrderID INTO v_OrderID FROM OrderUnit WHERE UnitID = OLD.UnitID;

    CALL sp_UpdateOrderTotal(v_OrderID);
END $$
```

Bên cạnh đó, tiền ship có thể tăng lên hoặc giảm tùy thuộc vào thời gian mua sắm.

Trong trường hợp giá ship sau khi thêm đơn hàng buyer được giảm thêm **20.000**, lúc này TotalPrice sẽ được cập nhật.

The screenshot shows a MySQL Workbench interface. The SQL editor contains the following code:

```
28 • UPDATE OrderUnit
29     SET ShippingFee = ShippingFee - 20000
30     WHERE UnitID = 1;
31
32     -- Kết quả: TotalPrice phải giảm thêm 20,000 so với Bước 2
33 • SELECT OrderID, TotalPrice AS '3_Sau_Khi_Thay_Doi_Ship' FROM `Order` WHERE OrderID = 1;
```

The Result Grid shows the following data:

OrderID	3_Sau_Khi_Tang_Ship
1	304000.00
NULL	NULL

B – Hệ thống tính TotalCartValue cho Cart

Giới thiệu

Hệ thống được thiết kế để tự động tính toán và cập nhật thuộc tính dãy xuất UpdateCartTotal (Cập nhật giá trị đơn hàng hiện tại có trong giỏ) trong cơ sở dữ liệu, đảm bảo tính chính xác và nhất quán dữ liệu.

Thuộc tính dãy xuất được chọn

Thủ tục nhận vào CartID của giỏ hàng cần cập nhật. Nó tính tổng giá trị giỏ hàng bằng cách duyệt qua tất cả các mục (CartItem) thuộc giỏ đó, nhân số lượng của từng sản phẩm với giá tương ứng từ bảng ProductVariant, và cộng dồn kết quả. Nếu giỏ hàng không có sản phẩm nào, tổng giá trị sẽ được đặt bằng 0. Cuối cùng, thủ tục cập nhật trường TotalCartValue trong



bảng Cart với tổng giá trị vừa tính được.

```
DROP PROCEDURE IF EXISTS UpdateCartTotal;
DELIMITER $$

CREATE PROCEDURE UpdateCartTotal(IN p_CartID INT)
BEGIN
    UPDATE Cart
    SET TotalCartValue = (
        SELECT COALESCE(SUM(ci.Quantity * pv.Price), 0)
        FROM CartItem ci
        JOIN product_variant pv
            ON ci.ProductVariantID = pv.ProductVariantID
        WHERE ci.CartID = p_CartID
    )
    WHERE CartID = p_CartID;
END$$
DELIMITER ;
```

Công thức tính:

$$TotalCartValue = \sum(CartItem.Quantity \times ProductVariant.Price)$$

Các thao tác DML ảnh hưởng

- INSERT: Thêm sản phẩm mới vào giỏ hàng → thay đổi tổng giá trị giỏ.

```
DROP TRIGGER IF EXISTS trg_cartitem_after_insert;
DELIMITER $$

CREATE TRIGGER trg_cartitem_after_insert
AFTER INSERT ON CartItem
FOR EACH ROW
BEGIN
    CALL UpdateCartTotal(NEW.CartID);
END$$
DELIMITER ;
```



CartID	TotalCartValue
1	28970000.00
NULL	NULL

- UPDATE: Thay đổi số lượng hoặc giá sản phẩm trong giỏ → thay đổi tổng giá trị giỏ.

```
UPDATE CartItem
SET Quantity = 5
WHERE CartID = 1 AND ProductVariantID = 3;

SELECT 'After UPDATE' AS Step;
SELECT CartID, TotalCartValue FROM Cart WHERE CartID = 1;
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```
88 • UPDATE CartItem
89     SET Quantity = 5
90     WHERE CartID = 1 AND ProductVariantID = 3;
91
92 • SELECT 'After UPDATE' AS Step;
93 • SELECT CartID, TotalCartValue FROM Cart WHERE CartID = 1;
```

The results grid shows the following data:

CartID	TotalCartValue
1	28970000.00
NULL	NULL

The status bar at the bottom indicates "Result 31" and "Cart 32". Below the results grid, the "Action Output" section displays the log entries:

Action	Time	Details
483	01:32:54	UPDATE CartItem SET Quantity = 5 WHERE CartID = 1 AND ProductVariantID = 3
484	01:32:54	SELECT 'After UPDATE' AS Step LIMIT 0, 1000
485	01:32:54	SELECT CartID, TotalCartValue FROM Cart WHERE CartID = 1 LIMIT 0, 1000

- DELETE: Xóa sản phẩm khỏi giỏ hàng → thay đổi tổng giá trị giỏ.

```
DELETE FROM CartItem
WHERE CartID = 1 AND ProductVariantID = 3;

SELECT 'After DELETE' AS Step;
SELECT CartID, TotalCartValue FROM Cart WHERE CartID = 1;
```



CartID	TotalCartValue
1	28970000.00
2	0.00
3	0.00

2.3 Thủ tục truy vấn dữ liệu

a. Thủ tục: sp_Cart_GetDetailsByBuyer

```
DROP PROCEDURE IF EXISTS sp_Cart_GetDetailsByBuyer$$
CREATE PROCEDURE sp_Cart_GetDetailsByBuyer(IN p_BuyerID INT)
BEGIN
    -- Kiểm tra Buyer có tồn tại không (tùy chọn, nhưng tốt cho logic)
    IF NOT EXISTS (SELECT 1 FROM Buyer WHERE BuyerID = p_BuyerID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lỗi: BuyerID không tồn tại.';
    END IF;

    SELECT
        st.Name AS StoreName,          -- Tên cửa hàng
        p.Title AS ProductName,        -- Tên sản phẩm
        pv.Price AS UnitPrice,         -- Giá gốc biển thẻ
        ci.Quantity,                  -- Số lượng trong giỏ
        (pv.Price * ci.Quantity) AS TotalItemPrice, -- Thành tiền tạm tính
        ci.SKU                         -- Mã SKU
    FROM Buyer b
    JOIN Cart c ON b.BuyerID = c.BuyerID
    JOIN CartItem ci ON c.CartID = ci.CartID
    JOIN product_variant pv ON ci.ProductVariantID = pv.ProductVariantID
    JOIN Product p ON pv.ProductID = p.ProductID
    JOIN Store st ON p.StoreID = st.StoreID
    WHERE b.BuyerID = p_BuyerID
    ORDER BY st.Name ASC, p.Title ASC;   -- Sắp xếp theo tên Shop rồi đến tên SP
END$$
```

Mục đích:

Thủ tục này được sử dụng để hiển thị chi tiết toàn bộ giỏ hàng của một người mua (Buyer) cụ thể. Nó giúp người dùng xem lại các sản phẩm họ đã chọn, giá tiền từng món, số lượng và tổng tiền tạm tính cho từng dòng sản phẩm, đồng thời biết được sản phẩm đó thuộc cửa hàng (Store) nào.

Tham số đầu vào:

p_BuyerID (INT): Mã định danh của người mua hàng cần xem giỏ.

Giải thích logic xử lý:



- **Kiểm tra điều kiện (Validation):** Đầu tiên, hệ thống kiểm tra xem p_BuyerID có tồn tại trong bảng Buyer hay không. Nếu không, trả về lỗi để đảm bảo tính toàn vẹn dữ liệu.
- **Truy vấn dữ liệu (Query):** Sử dụng câu lệnh SELECT kết hợp với mệnh đề JOIN để liên kết dữ liệu từ 6 bảng khác nhau:
 - Buyer → Cart: Tìm giỏ hàng của người dùng.
 - Cart → CartItem: Lấy danh sách các sản phẩm trong giỏ.
 - CartItem → ProductVariant: Lấy thông tin giá (Price) và mã SKU cụ thể của biến thể.
 - ProductVariant → Product: Lấy tên gốc của sản phẩm (Title).
 - Product → Store: Lấy tên cửa hàng (Name) bán sản phẩm đó.

- **Tính toán cột dẫn xuất:** Cột TotalItemPrice được tính toán trực tiếp trong câu truy vấn bằng công thức:

$$\text{TotalItemPrice} = \text{Price} \times \text{Quantity}$$

- **Lọc dữ liệu (Filtering):** Mệnh đề WHERE b.BuyerID = p_BuyerID đảm bảo chỉ lấy dữ liệu của đúng người dùng được yêu cầu.
- **Sắp xếp (Sorting):** Mệnh đề ORDER BY st.Name ASC, p.Title ASC giúp hiển thị danh sách gọn gàng: gom các sản phẩm cùng Shop lại với nhau (xếp theo tên Shop), sau đó xếp theo tên sản phẩm.

b. Thủ tục: sp_Cart_AnalyzeStoreTotal

```
▶ DROP PROCEDURE IF EXISTS sp_Cart_AnalyzeStoreTotal$$
▶ CREATE PROCEDURE sp_Cart_AnalyzeStoreTotal(
    IN p_BuyerID INT,
    IN p_MinTotal DECIMAL(10,2)
)
BEGIN
    SELECT
        st.Name AS StoreName,
        COUNT(ci.ProductVariantID) AS TotalItems,          -- Tổng số loại sản phẩm mua tại shop này
        SUM(ci.Quantity) AS TotalQuantity,                 -- Tổng số lượng hàng
        SUM(pv.Price * ci.Quantity) AS StoreSubTotal      -- Tổng tiền phải trả cho shop này
    FROM Cart c
    JOIN CartItem ci ON c.CartID = ci.CartID
    JOIN product_variant pv ON ci.ProductVariantID = pv.ProductVariantID
    JOIN Product p ON pv.ProductID = p.ProductID
    JOIN Store st ON p.StoreID = st.StoreID
    WHERE c.BuyerID = p_BuyerID
    GROUP BY st.StoreID, st.Name
    HAVING StoreSubTotal >= p_MinTotal                -- Chỉ hiện Shop có đơn > p_MinTotal
    ORDER BY StoreSubTotal DESC;                         -- Shop nào mua nhiều tiền nhất xếp trên
```

Mục đích:

Thủ tục này phục vụ nhu cầu thống kê và phân tích. Nó tính toán tổng số tiền mà người mua dự kiến sẽ trả cho từng Cửa hàng (Shop). Điều này hỗ trợ các chức năng như kiểm tra điều kiện áp mã giảm giá của Shop hoặc đơn giản là để người dùng quản lý chi tiêu.

Tham số đầu vào:



- p_BuyerID (INT): Mã người mua hàng.
- p_MinTotal (DECIMAL): Mức tổng tiền tối thiểu. Chỉ hiển thị những Shop mà người dùng đã mua vượt quá số tiền này.

Giải thích logic xử lý:

- **Liên kết bảng (Joining):** Tương tự như thủ tục trên, ta thực hiện JOIN qua các bảng Cart, CartItem, Product, Store để lấy được mối quan hệ giữa sản phẩm trong giỏ và cửa hàng bán nó.
- **Hàm tổng hợp (Aggregate Functions):**
 - COUNT(ci.ProductVariantID): Đếm xem người dùng mua bao nhiêu loại sản phẩm khác nhau từ shop này.
 - SUM(ci.Quantity): Tính tổng số lượng hàng hóa mua từ shop.
 - SUM(pv.Price * ci.Quantity): Tính tổng thành tiền (SubTotal) của shop đó.
- **Gom nhóm (Grouping):** Mệnh đề GROUP BY st.StoreID, st.Name là bắt buộc để các hàm tổng hợp ở trên tính toán dữ liệu riêng biệt cho từng cửa hàng.
- **Điều kiện trên nhóm (Having):** Mệnh đề HAVING StoreSubTotal >= p_MinTotal được sử dụng thay vì WHERE để lọc dữ liệu dựa trên kết quả tính toán tổng tiền (Aggregate). Nó loại bỏ các Shop mà tổng giá trị đơn hàng nhỏ hơn mức p_MinTotal.
- **Sắp xếp (Sorting):** ORDER BY StoreSubTotal DESC đưa các Shop mà người dùng phải trả nhiều tiền nhất lên đầu danh sách.

Kiểm tra

Tình huống 1: Buyer có ID = 1 muốn xem mình đang chọn mua những gì.

```
1 • CALL sp_Cart_GetDetailsByBuyer(2);
2
```

StoreName	ProductName	UnitPrice	Quantity	TotalItemPrice	SKU
John Electronics Store	Điện thoại Iphone 17ProMax	12990000.00	3	38970000.00	SKU-3
John Electronics Store	Tai nghe Bluetooth không dây chống ồn	1590000.00	1	1590000.00	SKU-4

Nhận xét:

Bảng kết quả sẽ hiển thị:

- Tên cửa hàng
- Tên sản phẩm
- Số lượng
- Tổng tiền của từng dòng sản phẩm đối với BuyerID = 2



Tình huống 2:

Người mua có BuyerID = 2 muốn xem tổng tiền phải trả cho từng Shop, nhưng chỉ quan tâm đến các Shop mà tổng tiền lớn hơn **500.000 VNĐ** (ví dụ để gom đơn freeship).

```
1 • CALL sp_Cart_AnalyzeStoreTotal(2, 500000);  
2
```

Result Grid				
	StoreName	TotalItems	TotalQuantity	StoreSubTotal
▶	John Electronics Store	2	4	40560000.00

c. Xây dựng Stored Procedures (Thủ tục lưu trữ)

Thủ tục 1: sp_SearchProductByCategoryAndPrice



```
-- =====
-- 1. TẠO THỦ TỤC TÌM KIẾM
-- =====
DELIMITER $$

DROP PROCEDURE IF EXISTS sp_SearchProductByCategoryAndPrice $$

CREATE PROCEDURE sp_SearchProductByCategoryAndPrice(
    IN p_CategoryName VARCHAR(255),
    IN p_MinPrice DECIMAL(10,2),
    IN p_MaxPrice DECIMAL(10,2)
)
BEGIN
    SELECT
        p.Title AS TenSanPham,
        c.Name AS TenDanhMuc,
        s.Name AS TenShop,
        pv.Price AS GiaTien,
        pv.Stock AS TonKho
    FROM Product p
    JOIN Product_Category pc ON p.ProductID = pc.ProductID
    JOIN Category c ON pc.CategoryID = c.CategoryID
    JOIN Store s ON p.StoreID = s.StoreID
    JOIN product_variant pv ON p.ProductID = pv.ProductID
    WHERE
        c.Name LIKE CONCAT('%', p_CategoryName, '%')
        AND pv.Price BETWEEN p_MinPrice AND p_MaxPrice
    ORDER BY pv.Price ASC;
END $$

DELIMITER ;
```

Mục đích:

Hỗ trợ chức năng tìm kiếm sản phẩm nâng cao trên hệ thống. Thủ tục này cho phép người dùng lọc sản phẩm dựa trên tên danh mục (có thể tìm gần đúng) và nằm trong một khoảng ngân sách (giá) nhất định.

Tham số đầu vào:

- p_CategoryName (VARCHAR): Tên danh mục cần tìm kiếm.
- p_MinPrice (DECIMAL): Mức giá sàn (thấp nhất).
- p_MaxPrice (DECIMAL): Mức giá trần (cao nhất).

Giải thích logic xử lý:

- **Liên kết bảng (Joining):** Thực hiện JOIN qua 5 bảng gồm Product, Product_Category, Category, Store, và ProductVariant để lấy đầy đủ thông tin hiển thị: Tên sản phẩm, Tên danh mục, Tên cửa hàng bán, Giá tiền và Tồn kho.
- **Điều kiện lọc (Filtering):**
 - Sử dụng LIKE CONCAT(...) với tham số p_CategoryName để tìm kiếm tương đối (ví dụ: nhập "Điện" ra "Điện tử", "Điện thoại").



- Sử dụng toán tử BETWEEN để lọc các sản phẩm có giá nằm trong khoảng từ p_MinPrice đến p_MaxPrice.
- **Sắp xếp (Sorting):** ORDER BY pv.Price ASC giúp hiển thị các sản phẩm có giá rẻ nhất lên đầu danh sách.

Kiểm tra:

Tình huống: Người dùng muốn tìm các sản phẩm thuộc nhóm "Điện" (Điện tử/Điện thoại) có giá từ 1 triệu đến 50 triệu VND.

Nhận xét: Kết quả trả về danh sách 7 sản phẩm thỏa mãn, bao gồm tai nghe, điện thoại và laptop, được sắp xếp giá tăng dần từ 1.590.000 đến 32.900.000.

-- Test Procedure 1

```
CALL sp_SearchProductByCategoryAndPrice('Điện', 1000000, 50000000);
```

	TenSanPham	TenDanhMuc	TenShop	GiaTien	TonKho
▶	Tai nghe Bluetooth không dây chống ồn	Điện tử	John Electronics Store	1590000.00	50
	Tai nghe Bluetooth không dây chống ồn	Điện tử	John Electronics Store	1590000.00	45
	Tai nghe Bluetooth không dây chống ồn	Điện tử	John Electronics Store	1590000.00	30
	Điện thoại Smartphone XYZ Pro	Điện thoại	John Electronics Store	8990000.00	25
	Điện thoại Smartphone XYZ Pro	Điện thoại	John Electronics Store	10990000.00	15
	Điện thoại Smartphone XYZ Pro	Điện thoại	John Electronics Store	12990000.00	10
	Laptop Gaming Pro RTX 4060	Điện tử	John Electronics Store	32900000.00	5

Thủ tục 2: sp_GetTopStoresByStockValue



```
-- =====
-- 2. TẠO THỦ TỤC THỐNG KÊ SHOP
-- =====

DELIMITER $$

DROP PROCEDURE IF EXISTS sp_GetTopStoresByStockValue $$

CREATE PROCEDURE sp_GetTopStoresByStockValue(
    IN p_MinTotalStock INT
)
BEGIN
    SELECT
        s.StoreID,
        s.Name AS TenCuaHang,
        COUNT(p.ProductID) AS SoLuongSanPham,
        SUM(pv.Stock) AS TongTonKho,
        SUM(pv.Stock * pv.Price) AS TongGiaTriHangHoa
    FROM Store s
    JOIN Product p ON s.StoreID = p.StoreID
    JOIN product_variant pv ON p.ProductID = pv.ProductID
    WHERE s.Status = 'Active'
    GROUP BY s.StoreID, s.Name
    HAVING SUM(pv.Stock) >= p_MinTotalStock
    ORDER BY TongGiaTriHangHoa DESC;
END $$

DELIMITER ;
```

Mục đích:

Phục vụ nhu cầu quản trị và thống kê hệ thống. Thủ tục này giúp xác định các Cửa hàng (Shop) có quy mô lớn dựa trên tổng lượng hàng tồn kho và tính toán tổng giá trị tài sản hàng hóa của họ.

Tham số đầu vào:

- p_MinTotalStock (INT): Nguồn số lượng tồn kho tối thiểu. Chỉ hiển thị các Shop có tổng số lượng sản phẩm trong kho lớn hơn hoặc bằng con số này.

Giải thích logic xử lý:

- **Liên kết bảng (Joining):** Kết nối bảng Store, Product và ProductVariant để truy xuất thông tin kho hàng của từng sản phẩm thuộc cửa hàng.
- **Điều kiện lọc (Filtering):** WHERE s.Status = 'Active' đảm bảo chỉ thống kê các cửa hàng đang hoạt động bình thường.



- **Hàm tổng hợp (Aggregate Functions):**

- COUNT(p.ProductID): Đếm tổng đầu mục sản phẩm của shop.
- SUM(pv.Stock): Tính tổng số lượng hàng tồn kho.
- SUM(pv.Stock * pv.Price): Tính tổng giá trị hàng hóa ước tính (Số lượng × Đơn giá).

- **Gom nhóm (Grouping):** GROUP BY s.StoreID, s.Name để gom dữ liệu theo từng cửa hàng riêng biệt.
- **Điều kiện trên nhóm (Having):** HAVING SUM(pv.Stock) >= p_MinTotalStock để lọc kết quả sau khi gom nhóm, loại bỏ các Shop quy mô nhỏ.
- **Sắp xếp (Sorting):** ORDER BY TongGiaTriHangHoa DESC đưa các Shop có giá trị tài sản lớn nhất lên đầu bảng xếp hạng.

Kiểm tra:

Tình huống: Quản trị viên muốn lọc ra danh sách các Shop có tổng lượng hàng tồn kho trên 50 sản phẩm để đánh giá năng lực cung ứng.

Nhận xét: Kết quả hiển thị 5 cửa hàng thỏa mãn điều kiện, trong đó "John Electronics Store" dẫn đầu với tổng tồn kho 180 và giá trị hàng hóa cao nhất.

-- Test Procedure 2

CALL sp_GetTopStoresByStockValue(50);

	StoreID	TenCuaHang	SoluongSanPham	TongTonKho	TongGiaTriHangHoa
▶	1	John Electronics Store	7	180	882750000.00
	3	Jane Beauty Paradise	6	550	127710000.00
	2	John Fashion Hub	8	440	125260000.00
	6	David Bookstore	6	1280	69800000.00
	4	Mike Sports World	4	295	59355000.00

2.4. Hàm xử lý dữ liệu

Xây dựng Functions (Hàm người dùng)

Mục tiêu:

Xử lý logic nghiệp vụ tính toán và kiểm tra tính hợp lệ của dữ liệu trong giao dịch, sử dụng các cấu trúc điều khiển nâng cao.

a. **Hàm fn_CalculateCartTotal**



```
-- =====
-- 3. TẠO HÀM TÍNH TỔNG TIỀN
-- =====

DELIMITER $$

DROP FUNCTION IF EXISTS fn_CalculateCartTotal $$
CREATE FUNCTION fn_CalculateCartTotal(p_CartID INT)
RETURNS DECIMAL(15,2)
DETERMINISTIC
READS SQL DATA

BEGIN
    DECLARE v_TotalAmount DECIMAL(15,2) DEFAULT 0;
    DECLARE v_Quantity INT;
    DECLARE v_Price DECIMAL(10,2);
    DECLARE v_Done BOOLEAN DEFAULT FALSE;

    DECLARE cur_CartItem CURSOR FOR
        SELECT ci.Quantity, pv.Price
        FROM CartItem ci
        JOIN product_variant pv ON ci.ProductVariantID = pv.ProductVariantID
        WHERE ci.CartID = p_CartID;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_Done = TRUE;

    IF NOT EXISTS (SELECT 1 FROM Cart WHERE CartID = p_CartID) THEN
        RETURN -1;
    END IF;

    OPEN cur_CartItem;
    read_loop: LOOP
        FETCH cur_CartItem INTO v_Quantity, v_Price;
        IF v_Done THEN LEAVE read_loop; END IF;
        SET v_TotalAmount = v_TotalAmount + (v_Quantity * v_Price);
    END LOOP;
    CLOSE cur_CartItem;

    RETURN v_TotalAmount;
END $$

DELIMITER ;
```

Mục đích:

Tính toán tổng số tiền (Grand Total) của một giỏ hàng bất kỳ. Hàm này được sử dụng khi hiển thị giỏ hàng hoặc bước thanh toán cuối cùng.

Tham số đầu vào:



p_CartID (INT): Mã định danh của giỏ hàng cần tính tiền.

Giải thích logic xử lý:

- **Kiểm tra dữ liệu (Validation):** Sử dụng IF NOT EXISTS để kiểm tra CartID có tồn tại trong hệ thống hay không. Nếu không, trả về -1 để báo lỗi.
- **Con trỏ (Cursor):** Khai báo con trỏ cur_CartItem để truy vấn danh sách các mặt hàng trong giỏ (CartItem kết hợp ProductVariant).
- **Vòng lặp (Looping):** Sử dụng cấu trúc LOOP để duyệt qua từng dòng dữ liệu mà con trỏ lấy được. Tại mỗi vòng lặp, thực hiện cộng dồn giá trị:

$$v_TotalAmount = v_TotalAmount + (Quantity \times UnitPrice)$$

- **Kết quả:** Trả về tổng tiền kiểu DECIMAL sau khi duyệt hết danh sách.

Kiểm tra:

Tình huống: Hệ thống cần hiển thị tổng tiền cho Giỏ hàng số 1.

Nhận xét: Hàm trả về giá trị **28,970,000.00**, khớp với tổng giá trị các mặt hàng đang có trong giỏ của người dùng này.

```
-- Test Function 1
SELECT fn_CalculateCartTotal(1) AS TongTienGioHang;
```

	TongTienGioHang
▶	28970000.00

b. Hàm fn_ValidateCartStock



```
-- =====
-- 4. TẠO HÀM CHECK TỒN KHO
-- =====
DELIMITER $$

DROP FUNCTION IF EXISTS fn_ValidateCartStock $$
CREATE FUNCTION fn_ValidateCartStock(p_CartID INT)
RETURNS BOOLEAN
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE v_IsValid BOOLEAN DEFAULT TRUE;
    DECLARE v_BuyQty INT;
    DECLARE v_StockQty INT;
    DECLARE v_Done BOOLEAN DEFAULT FALSE;

    DECLARE cur_CheckStock CURSOR FOR
        SELECT ci.Quantity, pv.Stock
        FROM CartItem ci
        JOIN product_variant pv ON ci.ProductVariantID = pv.ProductVariantID
        WHERE ci.CartID = p_CartID;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_Done = TRUE;

    IF NOT EXISTS (SELECT 1 FROM Cart WHERE CartID = p_CartID) THEN
        RETURN FALSE;
    END IF;

    OPEN cur_CheckStock;
    check_loop: LOOP
        FETCH cur_CheckStock INTO v_BuyQty, v_StockQty;
        IF v_Done THEN LEAVE check_loop; END IF;

        IF v_BuyQty > v_StockQty THEN
            SET v_IsValid = FALSE;
            LEAVE check_loop;
        END IF;
    END LOOP;
    CLOSE cur_CheckStock;

    RETURN v_IsValid;
END $$
DELIMITER ;
```

Mục đích:

Kiểm tra tính hợp lệ của giỏ hàng trước khi tiến hành tạo đơn hàng (Checkout). Đảm bảo rằng khách hàng không thể đặt mua số lượng lớn hơn số lượng thực tế còn trong kho.



Tham số đầu vào:

p_CartID (INT): Mã định danh của giỏ hàng cần kiểm tra.

Giải thích logic xử lý:

- **Con trỏ (Cursor):** Sử dụng con trỏ cur_CheckStock để lấy cặp dữ liệu {Số lượng mua, Tồn kho thực tế} của từng sản phẩm trong giỏ.
- **Vòng lặp và Điều kiện (Loop & IF):** Duyệt qua từng sản phẩm. Trong mỗi lần lặp, sử dụng câu lệnh IF:
 - Nếu v_BuyQty > v_StockQty (Mua nhiều hơn tồn): Gán biến kết quả v_IsValid = FALSE và sử dụng lệnh LEAVE để thoát vòng lặp ngay lập tức (cơ chế *Fail-fast*).
- **Kết quả:** Hàm trả về TRUE (1) nếu tất cả sản phẩm đều đủ hàng, và FALSE (0) nếu có ít nhất một sản phẩm không đủ.

Kiểm tra:

Tình huống: Người dùng nhấn nút “Thanh toán” cho Giỏ hàng số 1. Hệ thống gọi hàm để kiểm tra tồn kho.

Nhận xét: Kết quả trả về “Hợp lệ” (tương ứng với giá trị 1), nghĩa là tất cả các món hàng trong giỏ số 1 đều có số lượng mua nhỏ hơn hoặc bằng số lượng tồn kho hiện có.

```
-- Test Function 2
SELECT CartID, IF(fn_ValidateCartStock(CartID), 'Hợp lệ', 'Hết hàng') AS TrangThai FROM Cart WHERE CartID = 1;
```

	CartID	TrangThai
▶	1	Hợp lệ

c. Hàm fn_TotalRevenueByStore



```
DROP FUNCTION IF EXISTS fn_TotalRevenueByStore //
CREATE FUNCTION fn_TotalRevenueByStore(p_StoreID INT)
RETURNS DECIMAL(18,2)
DETERMINISTIC
READS SQL DATA
BEGIN
    -- 1. Khai báo biến cục bộ để lưu trữ dữ liệu tạm thời
    DECLARE done INT DEFAULT 0;          -- Cờ đánh dấu khi con trỏ duyệt hết dòng
    DECLARE v_quantity INT;             -- Biến lưu số lượng sản phẩm của từng dòng
    DECLARE v_unitPrice DECIMAL(10,2);   -- Biến lưu đơn giá tại thời điểm mua
    DECLARE totalRevenue DECIMAL(18,2) DEFAULT 0; -- Biến tích lũy tổng doanh thu

    -- 2. Khai báo CON TRỎ (CURSOR)
    -- Nhiệm vụ: Duyệt qua từng dòng trong bảng ORDER_LINE khớp với StoreID
    DECLARE cur CURSOR FOR
        SELECT ol.Quantity, ol.UnitPrice
        FROM ORDER_LINE ol
        JOIN OrderUnit ou ON ol.UnitID = ou.UnitID
        WHERE ou.StoreID = p_StoreID
        AND ou.Status = 'Completed'; -- Quan trọng: Chỉ tính đơn đã hoàn thành

    -- 3. Khai báo HANDLER để xử lý khi con trỏ duyệt hết dữ liệu
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    -- 4. Kiểm tra tham số đầu vào (Validation)
    IF p_StoreID IS NULL OR p_StoreID <= 0 THEN
        RETURN 0;
    END IF;

    -- Kiểm tra sự tồn tại của Store trong database
    IF NOT EXISTS (SELECT 1 FROM Store WHERE StoreID = p_StoreID) THEN
        RETURN -1; -- Trả về -1 để báo hiệu mã lỗi: Store không tồn tại
    END IF;

    -- 5. Mở con trỏ và bắt đầu vòng lặp tính toán
    OPEN cur;
    read_loop: LOOP
        -- Lấy dữ liệu từng dòng gán vào biến
        FETCH cur INTO v_quantity, v_unitPrice;

        -- Nếu đã duyệt hết danh sách thì thoát vòng lặp
        IF done = 1 THEN LEAVE read_loop; END IF;

        -- Tính toán: Cộng dồn (Số lượng * Đơn giá) vào tổng doanh thu
        SET totalRevenue = totalRevenue + (v_quantity * v_unitPrice);
    END LOOP;
    CLOSE cur;

    -- 6. Trả về kết quả cuối cùng
    RETURN totalRevenue;
END//
```

Mục đích:



Tính tổng doanh thu thực tế của một cửa hàng dựa trên các đơn hàng đã giao thành công.

Tham số đầu vào:

p_StoreID: Mã cửa hàng cần tính.

Giải thích thuật toán:

- **Validation (Kiểm tra dữ liệu):** Hàm kiểm tra p_StoreID có hợp lệ không (khác NULL, > 0) và có tồn tại trong bảng Store hay không. Nếu Store không tồn tại, hàm trả về -1 để báo lỗi.
- **Sử dụng Con trỏ (Cursor):** Khai báo con trỏ cur truy vấn vào bảng ORDER_LINE kết hợp với OrderUnit.
- **Điều kiện lọc (WHERE):** Chỉ lấy dữ liệu thuộc đúng StoreID và quan trọng là trạng thái Status = 'Completed'.
- **Vòng lặp tính toán (Loop):** Con trỏ duyệt qua từng dòng sản phẩm bán ra.
- **Hàm thực hiện phép tính:**

$$TngTin = TngTin + (SLng \times \text{EnGiLcMua})$$

- **Lưu ý:** Việc sử dụng giá trong ORDER_LINE (giá lúc mua) thay vì bảng ProductVariant (giá hiện tại) đảm bảo tính chính xác của doanh thu lịch sử.

Kết quả:

Hàm trả về con số tổng doanh thu dạng DECIMAL.

d. **Hàm fn_TotalSoldByVariant**

```
DROP FUNCTION IF EXISTS fn_TotalSoldByVariant //  
CREATE FUNCTION fn_TotalSoldByVariant(p_ProductVariantID INT)  
RETURNS INT  
DETERMINISTIC  
READS SQL DATA  
BEGIN  
    -- 1. Khai báo biến  
    DECLARE done INT DEFAULT 0;  
    DECLARE v_qty INT;  
    DECLARE totalQty INT DEFAULT 0;  
  
    -- 2. Khai báo CON TRỎ (CURSOR)  
    -- Nhiệm vụ: Lấy cột Quantity từ bảng ORDER_LINE của biến thể tương ứng  
    DECLARE cur CURSOR FOR  
        SELECT Quantity  
        FROM ORDER_LINE  
        WHERE ProductVariantID = p_ProductVariantID;  
  
    -- 3. Khai báo HANDLER  
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
```



```
-- 4. Kiểm tra tham số đầu vào
) IF p_ProductVariantID IS NULL OR p_ProductVariantID <= 0 THEN
    RETURN 0;
END IF;

-- Kiểm tra biến thể có tồn tại trong bảng product_variant không
) IF NOT EXISTS (SELECT 1 FROM product_variant WHERE ProductVariantID = p_ProductVariantID) THEN
    RETURN -1; -- Trả về -1 để báo lỗi: Biến thể không tồn tại
END IF;

-- 5. Mở con trỏ và vòng lặp
OPEN cur;
) read_loop: LOOP
    FETCH cur INTO v_qty;

    IF done = 1 THEN LEAVE read_loop; END IF;

    -- Cộng dồn số lượng
    SET totalQty = totalQty + v_qty;
END LOOP;
CLOSE cur;

-- 6. Trả về kết quả
RETURN totalQty;
END//
```

DELIMITER ;

Mục đích:

Thống kê tổng số lượng sản phẩm của một biến thể cụ thể đã được bán ra (dùng để phân tích độ hot của sản phẩm).

Tham số đầu vào:

p_ProductVariantID: Mã biến thể.

Giải thích thuật toán:

- Validation:** Kiểm tra tham số đầu vào và xác thực sự tồn tại của biến thể trong bảng ProductVariant. Trả về -1 nếu biến thể không tồn tại.
- Sử dụng Con trỏ (Cursor):** Khai báo con trỏ cur truy vấn vào bảng ORDER_LINE.
- Dữ liệu lấy ra:** Toàn bộ cột Quantity của các dòng có ProductVariantID tương ứng.
- Vòng lặp (Loop):** Duyệt qua từng dòng lệnh đặt hàng.
- Cộng dồn:** Biến totalQty được cộng dồn với giá trị Quantity của từng dòng.
- Kết quả:** Trả về số nguyên INT thể hiện tổng số lượng đã bán.

Kiểm tra:

- Kiểm tra hàm fn_TotalRevenueByStore:**



- Giả lập tình huống:
 - * Gán Status = 'Completed' cho đơn hàng thuộc Store 1 (để tính doanh thu).
 - * Gán Status = 'Shipping' cho đơn hàng thuộc Store 2 (để kiểm tra logic loại bỏ đơn chưa hoàn thành).
- Gán chi tiết đơn hàng (OrderLine):
 - * Xóa dữ liệu cũ của UnitID = 1.
 - * Thêm mới 2 dòng dữ liệu:
 - Sản phẩm A: Số lượng 2 × Đơn giá 100,000 = 200,000.
 - Sản phẩm B: Số lượng 1 × Đơn giá 200,000 = 200,000.
- **Tổng doanh thu kỳ vọng:** 400,000.

```
UPDATE OrderUnit SET Status = 'Completed' WHERE UnitID = 1;

-- 2. Set trạng thái đơn hàng của Store 2 thành 'Shipping'
-- (Để kiểm chứng hàm doanh thu sẽ KHÔNG tính đơn này -> Kết quả phải là 0)
UPDATE OrderUnit SET Status = 'Shipping' WHERE UnitID = 2;

-- 3. Reset và chèn dữ liệu chi tiết vào ORDER_LINE cho UnitID 1 (Store 1)
-- Mục tiêu: Tạo ra phép tính đơn giản: (2 * 100k) + (1 * 200k) = 400k
DELETE FROM ORDER_LINE WHERE UnitID = 1;

INSERT INTO ORDER_LINE (UnitID, ProductVariantID, SKU, Quantity, UnitPrice, Discount) VALUES
(1, 1, 'TEST-SKU-1', 2, 100000, 0), -- Mua 2 cái Variant 1, giá 100k
(1, 2, 'TEST-SKU-2', 1, 200000, 0); -- Mua 1 cái Variant 2, giá 200k

-----  

-- BUỘC 2: THỰC THI KIỂM THỬ (EXECUTION)  

-----  

SELECT '--- TEST HÀM 1: TÍNH DOANH THU STORE (fn_TotalRevenueByStore) ---' AS Check_Point;
```

The screenshot shows the results of a SQL query in the Results pane of SSMS. The query is:

```
39     StoreID,
40     Name AS Ten_Cua_Hang,
41     fn_TotalRevenueByStore(1) AS Ket_Qua_Thuc_Te,
42     400000.00 AS Ket_Qua_Mong_Doi
43   FROM Store WHERE StoreID = 1;
```

The result grid displays one row of data:

Kich_Ban	StoreID	Ten_Cua_Hang	Ket_Qua_Thuc_Te	Ket_Qua_Mong_Doi
Store 1 (Valid - Completed Order)	1	John Electronics Store	400000.00	400000.00

- Kiểm tra hàm fn_TotalSoldByVariant:



```
--  
63  
64 •  SELECT '--- TEST HÀM 2: TÍNH TỔNG SỐ LƯỢNG BÁN (fn_TotalSoldByVariant) ---' AS Check_Point;  
65  
66     -- Test Case 2.1: Variant Hợp lệ (Variant ID 1)  
67     -- Kỳ vọng: >= 2 (Vì ở bước chuẩn bị đã insert 2 cái, cộng với dữ liệu cũ nếu có)  
68 •  SELECT  
69         'Variant 1 (Valid)' AS Kich_Ban,  
70         ProductVariantID,  
71         fn_TotalSoldByVariant(1) AS Tong_So_Luong_Ban_Ra  
72     FROM product_variant WHERE ProductVariantID = 1;  
73
```

Result Grid | Filter Rows: | export: | wrap

	Kich_Ban	ProductVariantID	Tong_So_Luong_Ban_Ra
▶	Variant 1 (Valid)	1	3



3. Hiện thực ứng dụng

Nhóm đã xây dựng ứng dụng web Shopmie (mô phỏng Shopee) để minh họa việc kết nối và thao tác với cơ sở dữ liệu MySQL. Ứng dụng bao gồm hai phân hệ chính: Kênh Người Bán (Quản lý dữ liệu) và Cửa Hàng (Hiển thị và tương tác người dùng).

3.1. Chức năng quản lý dữ liệu (Thêm/Xóa/Sửa)

Giao diện "Kênh Người Bán" được hiện thực hóa để cho phép người dùng (với vai trò Seller) thao tác trực tiếp với bảng Product và các bảng liên quan (product_image, product_video, product_variant) trong cơ sở dữ liệu. Tại màn hình chính, danh sách sản phẩm được hiển thị dưới dạng bảng, cung cấp cái nhìn tổng quan về: Tên sản phẩm, Giá gốc, Giá khuyến mãi, Tồn kho và Trạng thái (Available, Out of Stock, v.v.).

The screenshot shows the Shopmie Admin Panel interface. At the top, there are navigation links for 'Kênh Người Bán' and 'Cửa Hàng', and user status indicators for 'Thông báo' and 'Tài khoản'. Below the header is a search bar with placeholder text 'Tim sản phẩm, thương hiệu...'. A red button labeled '+ Thêm Sản Phẩm' is located on the right.

The main area is titled 'Kênh Người Bán' and displays a table of products. The table has columns: SẢN PHẨM, GIÁ GỐC, GIÁ KM, KHO, TRẠNG THÁI, and THAO TÁC. The 'TRẠNG THÁI' column contains status buttons like 'Available' (green), 'Out of Stock' (grey), and 'Banned' (red). The 'THAO TÁC' column contains 'Sửa' (blue) and 'Xóa' (red) buttons. Each row represents a product with its ID, name, price, discounted price, stock level, and status.

SẢN PHẨM	GIÁ GỐC	GIÁ KM	KHO	TRẠNG THÁI	THAO TÁC
CPU Intel Core i9 14900KF ID: 21	đ18.999.000	đ12.499.000	10	Available	Sửa Xóa
CPU Intel Core Ultra 9 285K ID: 20	đ20.999.000	đ16.399.000	8	Available	Sửa Xóa
Asus ROG Zephyrus GT16 ID: 19	đ67.990.000	đ62.990.000	5	Available	Sửa Xóa
MSI Stealth 18 HX AI ID: 18	đ10.999.900	đ9.699.000	2	Available	Sửa Xóa
Asus ZenBook A14 ID: 13	đ29.990.000	đ25.999.000	10	Available	Sửa Xóa
ASUS TUF Gaming A14 (AMD Ryzen AI 9 HX 370 32GB 1TB SSD NVIDIA RTX 4060 144Hz)	đ32.900.000	-	5	Out of Stock	Sửa Xóa
Tai nghe Bluetooth AirPod Pro ID: 2	đ3.590.000	đ3.290.000	50	Available	Sửa Xóa
Điện thoại iPhone 17 Pro Max ID: 1	đ38.990.000	đ37.990.000	25	Available	Sửa Xóa

At the bottom of the page, there are four sections: CHĂM SÓC KHÁCH HÀNG, VỀ SHOPMIE, THANH TOÁN, and THEO DÕI CHỦNG TỐI. Each section contains links to various website pages.



Hình 1: Hình ảnh kênh người bán

Chức năng Thêm mới (Create): Khi người dùng nhấn nút "Thêm Sản Phẩm", một cửa sổ (Modal) sẽ hiện ra. Hệ thống yêu cầu người dùng nhập đầy đủ các thông tin như: Hình ảnh/Video (URL), Tên sản phẩm, Danh mục, Giá, Kho và Mô tả.

The screenshot shows the Shopmie platform's product management interface. On the left, there's a sidebar with navigation links like 'Kênh Người Bán' and 'Cửa hàng'. The main area displays a list of products with columns for 'SẢN PHẨM' (Product), 'THÁI' (Status), and 'THAO TÁC' (Actions). A modal window titled 'Thêm mới sản phẩm' (Add Product) is open in the center. It includes fields for 'Hình ảnh (URL - Mỗi dòng 1 link)' (Image URL), 'Video (URL - Mỗi dòng 1 link)' (Video URL), 'Tên sản phẩm (Title)' (Product Title), 'Danh mục (Category)' (Category), 'Trạng thái (Status)' (Status), 'Giá gốc (Price)' (Original Price), 'Giá KM (Promote)' (Promotion Price), 'Kho (Stock)' (Stock), and 'Mô tả (Description)' (Description). At the bottom of the modal are 'Hủy' (Cancel) and 'Lưu' (Save) buttons. Below the modal, there's a footer section with links for 'CHĂM SÓC KHÁCH HÀNG', 'VỀ SHOPMIE', 'THANH TOÁN', and 'THEO DÕI CHÚNG TÔI', along with payment method icons and social media links.

Hình 2: Thêm dữ liệu hàng hóa Smart Tivi LG Evo Oled 4K 55 inch

Chức năng Cập nhật (Update): Khi nhấn nút "Sửa" trên một dòng sản phẩm, hệ thống sẽ gọi API lấy chi tiết dữ liệu hiện tại từ Database và điền vào Form (Pre-fill). Người dùng có thể chỉnh sửa lại giá bán, cập nhật số lượng tồn kho hoặc thay đổi thông tin mô tả.



The screenshot shows the Shopmie platform's product management interface. A modal window titled "Cập nhật sản phẩm" (Update Product) is open over a list of products. The modal contains fields for updating product details like title, category, price, and description. The product being updated is a "Smart Tivi LG Evo Oled 4K 55 inch 2025". The modal includes sections for "Hình ảnh (URL - Mỗi dòng 1 link)" (Image URLs), "Video (URL - Mỗi dòng 1 link)" (Video URLs), "Tên sản phẩm (Title)" (Product Name), "Danh mục (Category)" (Category), "Trạng thái (Status)" (Status), "Giá gốc (Price)" (Original Price), "Giá KM (Promote)" (Promotional Price), "Kho (Stock)" (Stock), and "Mô tả (Description)" (Description). At the bottom of the modal are "Hủy" (Cancel) and "LƯU" (Save) buttons. The background shows a grid of other products including CPU Intel Core i9, GPU ASUS ROG Zephyrus G16, and a MSI Stealth 18 HX AI laptop.

Hình 3: Chỉnh sửa giá và tồn kho của Smart Tivi LG Evo Oled 4K 55 inch

Chức năng Xóa (Delete) và Xử lý lỗi logic: Để đảm bảo an toàn dữ liệu, khi người dùng chọn "Xóa", hệ thống sẽ hiển thị một thông báo xác nhận (Confirm Dialog) để tránh thao tác nhầm lẫn. Xử lý Logic: Trước khi thực hiện lệnh DELETE trong CSDL, Backend sẽ kiểm tra các ràng buộc khóa ngoại (Foreign Key). Nếu sản phẩm đã nằm trong một đơn hàng (ORDER_LINE), hệ thống sẽ chặn việc xóa và thông báo lỗi phù hợp để bảo toàn lịch sử giao dịch. Nếu sản phẩm chưa có phát sinh giao dịch, việc xóa sẽ được thực thi thành công.



The screenshot shows the Shopmie admin dashboard for managing products. A central modal dialog box is displayed with the message: "localhost:5173 cho biết" and "Bạn có chắc chắn muốn xóa "Smart Tivi LG Evo Oled 4K 55 inch 2025" không?". Below the modal are two buttons: "OK" and "Hủy". The main table lists various products with columns for Product Name, Price, Stock Level, Status, and Actions (Edit, Delete). One product, "Smart Tivi LG Evo Oled 4K 55 inch 2025", is marked as "Available". At the bottom of the page, there are footer sections for Customer Support, About Shopmie, Payment Methods, and Social Media links.

SẢN PHẨM	GIÁ GỐC	GIÁ KM	KHO	TRẠNG THÁI	THAO TÁC
Smart Tivi LG Evo Oled 4K 55 inch 2025 ID: 22	đ42.900.000	đ27.999.999	7	Available	Sửa Xóa
CPU Intel Core i9 14900KF ID: 21	đ18.999.000	đ12.499.000	10	Available	Sửa Xóa
CPU Intel Core Ultra 9 285K ID: 20	đ20.999.000	đ16.399.000	8	Available	Sửa Xóa
Asus ROG Zephyrus G16 ID: 19	đ67.990.000	đ62.990.000	5	Available	Sửa Xóa
MSI Stealth 18 HX AI ID: 18	đ10.999.900	đ9.699.000	2	Available	Sửa Xóa
Asus ZenBook A14 ID: 13	đ29.990.000	đ25.999.000	10	Available	Sửa Xóa
ASUS TUF Gaming A14 (AMD Ryzen AI 9 HX 370 32GB 1T... ID: 3	đ32.900.000	-	5	Out of Stock	Sửa Xóa
Tai nghe Bluetooth AirPod Pro ID: 2	đ3.590.000	đ3.290.000	50	Available	Sửa Xóa
Điện thoại iPhone 17 Pro Max ID: 1	đ38.990.000	đ37.990.000	25	Available	Sửa Xóa

CHĂM SÓC KHÁCH HÀNG
Trung Tâm Trợ Giúp
Shopmie Blog
Hướng Dẫn Mua Hàng

VỀ SHOPMIE
Giới Thiệu Về Shopmie
Tuyển Dụng
Điều Khoản Shopmie

THANH TOÁN
VISA MasterCard COD

ĐƠN VỊ VẬN CHUYỂN

THEO DÕI CHỦNG TÔI
Facebook
Instagram
LinkedIn

Hình 4: Xác nhận trước khi xóa sản phẩm

Sau khi xóa thành công, danh sách sẽ tự động được tải lại (Re-fetch) để cập nhật giao diện mới nhất mà không cần tải lại trang.



The screenshot shows a list of products from a seller named "Kênh Người Bán". The products include:

SẢN PHẨM	GIÁ GỐC	GIÁ KM	KHO	TRẠNG THÁI	THAO TÁC
CPU Intel Core i9 14900KF Idx: 21	đ18.999.000	đ12.499.000	10	Available	Sửa Xóa
CPU Intel Core Ultra 9 285K Idx: 20	đ20.999.000	đ16.399.000	8	Available	Sửa Xóa
Asus ROG Zephyrus G16 Idx: 19	đ67.990.000	đ62.990.000	5	Available	Sửa Xóa
MSI Stealth 18 HX AI Idx: 18	đ10.999.900	đ9.699.000	2	Available	Sửa Xóa
Asus ZenBook A14 Idx: 13	đ29.990.000	đ25.999.000	10	Available	Sửa Xóa
ASUS TUF Gaming A14 (AMD Ryzen AI 9 HX 370 32GB 1T... Idx: 3	đ32.900.000	-	5	Out of Stock	Sửa Xóa
Tai nghe Bluetooth AirPod Pro Idx: 2	đ3.590.000	đ3.290.000	50	Available	Sửa Xóa
Điện thoại iPhone 17 Pro Max Idx: 1	đ38.990.000	đ37.990.000	25	Available	Sửa Xóa

At the bottom, there are sections for customer support, shop information, payment methods, delivery partners, and social media links.

Hình 5: Sau khi xóa dữ liệu thành công

3.2. Giao diện hiển thị, tìm kiếm và tương tác người dùng

Phần giao diện "Cửa hàng" (Storefront) được xây dựng để minh họa cho việc gọi các Thủ tục lưu trữ (Stored Procedures) đã viết ở câu 2.3 (ví dụ: sp_SearchProductByCategoryAndPrice). Giao diện này tập trung vào trải nghiệm người dùng với bố cục lưới (Grid layout) trực quan. Màn hình chính hiển thị danh sách sản phẩm với đầy đủ thông tin cần thiết cho người mua: Hình ảnh, Tên, Giá bán, Nhãn giảm giá (Discount Badge) và Số lượng đã bán.



The screenshot shows a search results page on the Shopmie website. At the top, there's a red header bar with links for 'Kênh Người Bán' and 'Cửa hàng', and icons for 'Thông báo' and 'Tài khoản'. Below the header is a search bar with placeholder text 'Tim sản phẩm, thương hiệu...'. The main content area has a white background with a grid of product cards. Each card includes a small image of the product, its name, and a price tag. Some cards also show a discount percentage. The products listed include:

- Tai nghe Bluetooth AirPods Pro (8% Giảm) - Giá: ₫3,290,000
- Điện thoại iPhone 17 Pro Max (3% Giảm) - Giá: ₫37,990,000
- CPU Intel Core i9 1400KF (34% Giảm) - Giá: ₫12,499,000
- CPU Intel Core Ultra 9 285K (22% Giảm) - Giá: ₫16,399,000
- Asus ROG Zephyrus G16 (2024) (7% Giảm) - Giá: ₫62,990,000
- MSI Stealth 18 HX AI (12% Giảm) - Giá: ₫9,699,000
- ASUS Zenbook A14 UX3401TA-QD51W (13% Giảm) - Giá: ₫25,999,000
- ASUS TUF Gaming A14 (AMD Ryzen AI 9 16X 3.7) | 32GB | 11... - Giá: ₫32,900,000

Below the products, there are four sections: 'CHĂM SÓC KHÁCH HÀNG', 'VỀ SHOPMIE', 'THANH TOÁN', and 'THEO DÕI CHÚNG TÔI'. Each section contains several links related to customer service, payment methods, and social media links.

Hình 6: Giao diện cửa hàng hiển thị các sản phẩm

Chức năng Tìm kiếm và Lọc (Search & Filter): Hệ thống cung cấp thanh điều hướng theo danh mục (Tabs). Khi người dùng chọn một danh mục (ví dụ: "Điện thoại"), ứng dụng sẽ gửi tham số CategoryName xuống Database thông qua thủ tục lưu trữ để lọc ra chính xác các sản phẩm thuộc nhóm đó.



CHĂM SÓC KHÁCH HÀNG

Trung Tâm Trợ Giúp
Shopmie Blog
Hướng Dẫn Mua Hàng
Hướng Dẫn Bán Hàng
Thanh Toán
Vận Chuyển
Trả Hàng & Hoàn Tiền

VỀ SHOPMIE

Giới Thiệu về Shopmie
Tuyển Dụng
Điều Khoản Shopmie
Chính Sách Bảo Mật
Chính Hàng
Kính Người Bán
Flash Sales

THANH TOÁN

Visa | MasterCard | COD

DON VI VẬN CHUYỂN

SGP | GHN | GHN | ViettelPost

THEO DÕI CHUNG TÔI

Facebook
Instagram
LinkedIn

© 2025 Shopmie. Tất cả các quyền được bảo lưu.
Chuyển nhượng Khu vực Việt Nam

Hình 7: Lọc theo danh mục sản phẩm Điện thoại

Chức năng Sắp xếp (Sorting): Người dùng có thể sắp xếp danh sách sản phẩm theo các tiêu chí: Phổ biến, Mới nhất, Bán chạy và Giá (Thấp đến Cao / Cao đến Thấp). Việc này giúp người dùng dễ dàng tìm kiếm sản phẩm phù hợp với nhu cầu và ngân sách.



The screenshot shows a search results page on the Shopmie website. At the top, there are navigation links for 'Kênh Người Bán' and 'Cửa hàng', and icons for 'Thông báo' and 'Tài khoản'. The main header features the 'Shopmie' logo and a search bar with the placeholder 'Tim sản phẩm, thương hiệu...'. Below the header is a 'DANH MỤC SẢN PHẨM' section with tabs for 'Tất cả' (selected), 'Điện tử', 'Thời trang', 'Gia dụng', 'Điện thoại', and 'Phụ kiện'. A sorting dropdown menu shows 'Sắp xếp theo' with options 'Phổ biến', 'Mới nhất', 'Bán chạy', and 'Giá: Cao đến Thấp' (selected). The main content area displays a grid of products:

Product	Description	Original Price	Sale Price	Stock Status
Asus ROG Zephyrus G16 (2024)	Ultrathin Gaming Laptop RTX 4060 12GB 2TB NVMe SSD 32GB LPDDR5 RAM	đ67.999.000	đ62.990.000	08 bản 0
Điện thoại iPhone 17 Pro Max	Smartphone 17 Pro Max 256GB 5G	đ38.990.000	đ37.990.000	08 bản 1
ASUS TUF Gaming A14 (AMD Ryzen AI 9 HX 370 32GB 1TB)	Gaming Laptop AMD Ryzen AI 9 HX 370 32GB 1TB NVMe SSD RTX 4060 12GB	đ32.900.000	đ32.900.000	08 bản 0
Asus ZenBook A14	Ultrathin Laptop Intel Core Ultra 9 280K 14.5 inch FHD 16GB RAM 512GB NVMe SSD	đ25.999.000	đ25.999.000	08 bản 0
CPU Intel Core Ultra 9 280K	Processor Intel Core Ultra 9 280K 18MB Cache 2.8GHz - 4.8GHz	đ16.399.000	đ16.399.000	08 bản 0
CPU Intel Core i9 14900KF	Processor Intel Core i9 14900KF 24MB Cache 3.5GHz - 5.8GHz	đ12.499.000	đ12.499.000	08 bản 0
MSI Stealth 18 HX AI	Gaming Laptop Intel Core i9 14900KF 18.4 inch FHD 32GB RAM 1TB NVMe SSD	đ99.999.000	đ9.699.000	08 bản 0
Tai nghe Bluetooth AirPod Pro	Bluetooth Earphones AirPod Pro White	đ5.990.000	đ3.290.000	08 bản 1

Below the products, there are sections for 'CHĂM SÓC KHÁCH HÀNG', 'VỀ SHOPMIE', 'THANH TOÁN', and 'THEO DÕI CHÚNG TÔI'. The footer contains copyright information: © 2025 Shopmie. Tất cả các quyền được bảo lưu. Quốc gia & Khu vực: Việt Nam.

Hình 8: Sắp xếp giá từ cao đến thấp

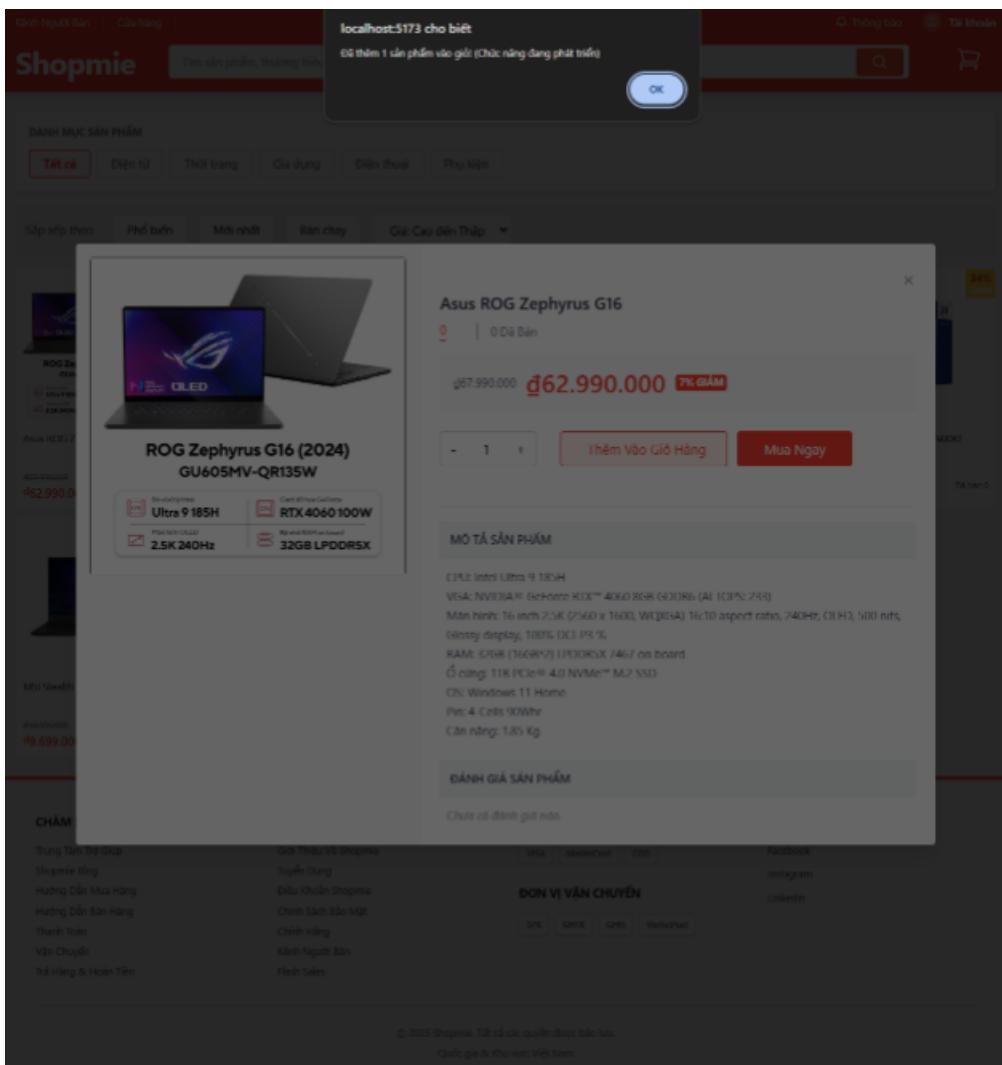
Chi tiết sản phẩm và Validate logic mua hàng: Khi nhấp vào một sản phẩm, Modal chi tiết sẽ hiện ra với đầy đủ mô tả và thông số kỹ thuật. Tại đây, hệ thống hiển thị số lượng tồn kho thực tế.



The screenshot shows a product detail page for the **Asus ROG Zephyrus G16 (2024) GU605MV-QR135W**. The main image displays the laptop from two angles. Below the image, key specifications are listed: Intel Ultra 9 108H, NVIDIA GeForce RTX 4060 8GB GDDR6, 2.5K 240Hz display, and 32GB LPDDR5X RAM. The price is shown as **đ62.990.000** with a **7% GIẢM** discount. A red button labeled **Mua Ngay** (Buy Now) is prominent. The page also includes sections for **MÔ TẢ SẢN PHẨM** (Product Description) and **DÀNH GIÁ SẢN PHẨM** (Special Offer). At the bottom, there's a footer with links to various services like Trung Tâm Hỗ Trợ Khách Hàng, Tuyển Dụng, and Chính Sách Bảo Mật, along with payment method icons (VISA, MasterCard, COD) and social media links (Facebook, Instagram, LinkedIn).

Hình 9: Hiển thị thông tin chi tiết khi chọn sản phẩm

Chức năng Thêm vào giỏ hàng: Khi người dùng nhấn "Thêm Vào Giỏ Hàng", hệ thống sẽ thực hiện kiểm tra logic (sử dụng Hàm 2.4 `fn_ValidateCartStock`).



Hình 10: Chức năng thêm vào giỏ hàng