**CƠ SỞ DỮ LIỆU QUẢN LÝ SÀN THƯƠNG MẠI ĐIỆN TỬ**

1. **Mô tả hệ thống**

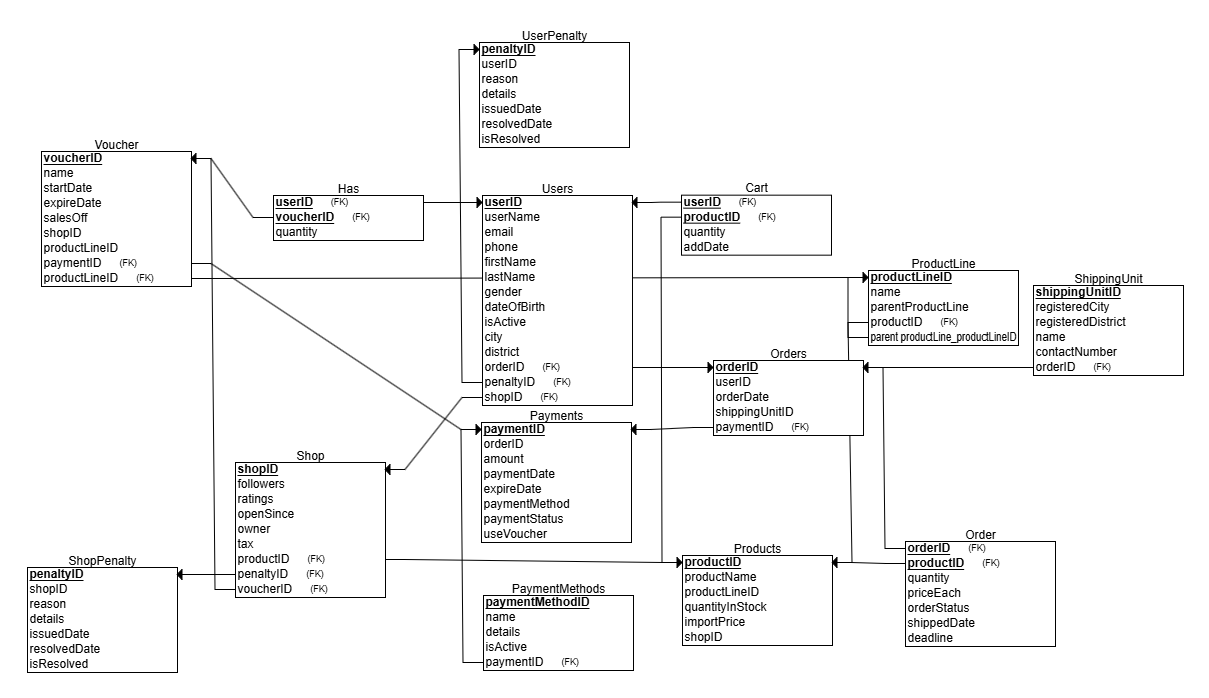
* Đây là hệ thống quản lý sàn thương mại điện tử.
* Các thành phần của cơ sở dữ liệu:
  + users(userID, username, email, phone, firstName, lastName, gender, dateOfBirth, isActive, city, district)
  + shops(shopID, numberOfFollowers, ratings, openSince, shopOwner, taxRate)
  + products(productID, productName, productLineID, quantityInStock, sellPrice, shopID)
  + productLines(productLineID, name, description)
  + payments(payemntID, orderID, amount, paymentDate, expireDate, paymentMethodID, paymentStatus, voucherID)
  + paymentMethods(paymentMethodID, name, details, isActive)
  + vouchers(voucherID, name, salesOff, shopID, productLineID, minimumOrderValue, maximumDiscount)
  + voucherStorage(voucherID, userID, quantity, acquiredDate, expireDate)
  + orders(orderID, userID, orderDate, shippingUnitID)
  + orderDetails(orderID, productID, orderStatus, shippedDate, deadline, quantity)
  + itemsInCart(userID, productID, quantity, isAvailable)
  + userPenalty(penaltyID, userID, reason, details, issuedDate, resolvedDate, isResolved)
  + shopPenalty(penaltyID, shopID, reason, details, issuedDate, resolvedDate, isResolved)
  + shippingUnits(shippingUnitID, registeredCity, registeredDistrict, name, contactNumber)
* Chức năng của các bảng trong cơ sở dữ liệu
  + **users**: Quản lý thông tin người dùng, bao gồm thông tin cá nhân, trạng thái hoạt động, và địa chỉ.
  + **shops**: Lưu thông tin cửa hàng, chủ sở hữu, và các thuộc tính liên quan như đánh giá, số lượng người theo dõi, ngày mở cửa.
  + **products**: Quản lý sản phẩm, bao gồm thông tin chi tiết về giá, số lượng trong kho, và liên kết với cửa hàng.
  + **productLines**: Lưu thông tin về các danh mục sản phẩm, dùng để nhóm các sản phẩm tương tự.
  + **payments**: Theo dõi các khoản thanh toán cho đơn hàng, bao gồm chi tiết phương thức thanh toán, trạng thái, và voucher sử dụng.
  + **paymentMethods**: Lưu thông tin về các phương thức thanh toán, như tên, chi tiết, và trạng thái hoạt động.
  + **vouchers**: Quản lý mã giảm giá, bao gồm thông tin khuyến mãi và các điều kiện áp dụng.
  + **voucherStorage**: Theo dõi việc lưu trữ và sử dụng mã giảm giá của người dùng.
  + **orders**: Quản lý các đơn đặt hàng, liên kết người dùng với đơn hàng và đơn vị vận chuyển.
  + **orderDetails**: Chi tiết sản phẩm trong từng đơn hàng, bao gồm trạng thái giao hàng và số lượng.
  + **itemsInCart**: Lưu trữ thông tin giỏ hàng của người dùng, bao gồm sản phẩm và số lượng.
  + **userPenalty**: Ghi nhận các hình phạt hoặc vi phạm của người dùng.
  + **shopPenalty**: Ghi nhận các vi phạm hoặc hình phạt áp dụng cho cửa hàng.
  + **shippingUnits**: Quản lý thông tin đơn vị vận chuyển, như địa chỉ đăng ký và thông tin liên hệ.
* Work flow của hệ thống:
  + Mỗi người mua có 1 giỏ hàng. Giỏ hàng có thể chứa nhiều sản phẩm được chọn.
  + Khi 1 sản phẩm được đặt mua, nếu sản phẩm được chọn từ giỏ hàng, sản phẩm sẽ được xóa khỏi giỏ hàng và thêm vào danh sách đặt hàng; nếu sản phẩm được mua trực tiếp không qua giỏ hàng, sản phẩm sẽ được thêm trực tiếp vào danh sách đặt hàng.
  + Khi người mua xác nhận 1 đơn hàng, hệ thống sẽ tổng hợp thông tin các sản phẩm trong danh sách đặt hàng (tên sản phẩm, số lượng,..) thành một danh sách đặt hàng chi tiết, tham chiếu tới danh sách đặt hàng ban đầu thông qua mã đơn hàng.
  + Mỗi sản phẩm có 1 mã sản phẩm. Cửa hàng phải cung cấp thông tin về sản phẩm của mình vào hệ thống.
  + Khi đơn đặt hàng được xác nhận bởi hệ thống, sản phẩm sẽ được bàn giao cho đơn vị vận chuyển.
  + Đơn vị vận chuyển tìm shipper phù hợp (cùng khu vực với khách hàng) để giao hàng.
  + Người mua có thể thực hiện thanh toán ngay khi đặt hàng hoặc khi nhận được hàng.
  + Người mua có thể sử dụng voucher để giảm giá và nhận ưu đãi.
  + Voucher có thể chỉ áp dụng cho một cửa hàng và chỉ có hiệu lực với các sản sản phẩm của cửa hàng đó; có thể chỉ áp dụng với một số loại sản phẩm nhất định; hoặc có thể áp dụng với mọi sản phẩm của mọi cửa hàng.

**II. ER Model**

**A diagram of a network

Description automatically generated**

**III. Relational Model and Functional Dependencies**

****

**users**(userID, username, email, phone, firstName, lastName, gender, dateOfBirth, isActive, city, district)

F = {userID -> (username, email, phone, firstName, lastName, gender, dateOfBirth, isActive, city, district),

username -> (userID, email, phone, firstName, lastName, gender, dateOfBirth, isActive, city, district)}

**shops**(shopID, numberOfFollowers, ratings, openSince, shopOwner, taxRate)

F = {shopID -> (numberOfFollowers, ratings, openSince, shopOwner, taxRate)}

**products**(productID, productName, productLineID, quantityInStock, sellPrice, shopID)

F = {productID -> (productName, productLineID, quantityInStock, sellPrice, shopID)}

**productLines**(productLineID, name, description)

F = {productLineID -> (name, description)}

**payments**(payemntID, orderID, amount, paymentDate, expireDate, paymentMethodID, paymentStatus, voucherID)

F = {paymentID -> (orderID, orderID, amount, paymentDate, expireDate, paymentMethodID, paymentStatus, voucherID)}

**paymentMethods**(paymentMethodID, name, details, isActive)

F = {paymentMethodID -> ( name, details, isActive)}

**vouchers**(voucherID, name, salesOff, shopID, productLineID, minimumOrderValue, maximumDiscount)

F = {voucherID -> (name, salesOff, shopID, productLineID, minimumOrderValue, maximumDiscount)}

**voucherStorage**(voucherID, userID, quantity, acquiredDate, expireDate)

F = {(voucherID, userID) -> (quantity, acquiredDate, expireDate)}

**orders**(orderID, userID, orderDate, shippingUnitID)

F = {orderID -> (userID, orderDate, shippingUnitID)}

**orderDetails**(orderID, productID, orderStatus, shippedDate, deadline, quantity)

F = {(orderID, productID) -> (quantity, orderStatus, shippedDate, deadline)}

**itemsInCart**(userID, productID, quantity, isAvailable)

F = {(userID, productID) -> (quantity, isAvailable)}

**userPenalty**(penaltyID, userID, reason, details, issuedDate, resolvedDate, isResolved)

F = {penaltyID -> (userID, reason, details, issuedDate, resolvedDate, isResolved)}

**shopPenalty**(penaltyID, shopID, reason, details, issuedDate, resolvedDate, isResolved)

F = {penaltyID -> (shopID, reason, details, issuedDate, resolvedDate, isResolved)}

**shippingUnits**(shippingUnitID, registeredCity, registeredDistrict, name, contactNumber)

F = {shippingUnitID -> (registeredCity, registeredDistrict, name, contactNumber)}

**IV. Constraints**

Sau khi tạo bảng, việc tiếp theo em làm là thêm các ràng buộc vào cơ sở dữ liệu. Ngoài các khóa ngoại được thêm vào các bảng để thể hiện mối quan hệ giữa chúng, em cũng thêm những ràng buộc sau:

* **Default Constraint:**
  + Thêm default constraint vào thuộc tính paymentStatus của bảng payments là ‘Pending’, để khi không khai báo dữ liệu cho thuộc tính này, thì trạng thái thanh toán luôn là ‘Pending’, tức đang thực hiện.
  + Thêm default constraint vào thuộc tính isResolved của bảng userPenalty và shopPenalty, để đặt mặc định tình trạng xử lý những hình phạt của shop và người dùng là đang xử lý.
  + Thêm default constraint vào thuộc tính orderStatus của bảng orderDetails là ‘Processing’, tức mặc định tình trạng đơn hàng chi tiết là đang xử lý nếu không khai báo cụ thể.
* **Index Constraint:**
  + Thêm index vào các thuộc tính productID, productLineID, shippingUnitID, voucherID, userID nhằm giúp việc tìm kiếm trong các bảng lần lượt là products, productLines, shippingUnits, vouchers, users nhanh chóng hơn.
  + Các thuộc tính của các bảng đó được sử dụng để tìm kiếm nhiều, đồng thời cũng ít dữ liệu mới thêm vào hơn so với các bảng khác, vì vậy việc thêm index constraint giúp cơ sở dữ liệu hoạt động tối ưu hơn.
* **Check Constraint:**
  + Thêm check constraint vào thuộc tính ratings của bảng shops để giới hạn ratings của shops chỉ từ 0 đến 5, tức cửa hàng chỉ được đánh giá trên thang 5 sao, không được vượt quá.
  + Thêm check constraint vào thuộc tính salesOff của bảng vouchers để giới hạn mức giảm giá của vouchers chỉ được từ 0% đến 100%, không thể vượt ngoài mức này.

**V. Triggers**

Trigger (kích hoạt) là một đối tượng trong cơ sở dữ liệu, tự động thực thi một đoạn mã lệnh (thường là SQL) khi xảy ra một sự kiện cụ thể trên một bảng hoặc khung nhìn (view). Trigger thường được sử dụng để duy trì tính toàn vẹn dữ liệu, thực hiện kiểm tra tự động, hoặc thực hiện các hành động bổ sung khi dữ liệu được thay đổi. Nhóm em đã cài một số triggers cho cơ sở dữ liệu:

* Trigger check\_product\_stock của bảng itemsInCarts: Trước khi thêm một hàng dữ liệu mới vào bảng itemsInCarts, tức là một người dùng thêm sản phẩm vào giỏ hàng của họ, kiểm tra xem số lượng sản phẩm có thỏa mãn điều kiện nhỏ hơn hoặc bằng số lượng hàng trong kho không, nếu không thì sẽ báo lỗi vào không thực hiện việc thêm dữ liệu.
* Trigger check\_items\_in\_carts của bảng products: Sau khi cập nhật dữ liệu bảng products, kiểm tra xem số lượng sản phẩm trong giỏ hàng của người dùng có còn thỏa mãn điều kiện nhỏ hơn hoặc bằng số hàng trong kho không, nếu không sẽ đặt thuộc tính isAvailable của bảng itemsInCart thành ‘No’.
* Trigger check\_valid\_order của bảng orderDetails: trước khi thêm dữ liệu vào bảng orderDetails, tức khi người dùng đặt hàng, thì số lượng sản phẩm được đặt không vượt quá số lượng hàng trong kho, nếu không sẽ báo lỗi.
* Trigger update\_quantity\_in\_stock của bảng orderDetails: Khi một order được tạo thành công, thì trigger sẽ cập nhật lại số lượng hàng trong kho.

**VI. Procedures**

Procedure applyVoucher được thiết kế để xử lý việc áp dụng mã giảm giá (voucher) vào một giao dịch thanh toán cụ thể. Nó đảm bảo rằng mã giảm giá được sử dụng hợp lệ, người dùng có quyền sử dụng mã giảm giá và các điều kiện sử dụng được đáp ứng

Luồng hoạt động

1. Bắt đầu giao dịch:
   * Procedure bắt đầu một giao dịch (START TRANSACTION) để đảm bảo tính nhất quán dữ liệu. Nếu bất kỳ bước nào thất bại, giao dịch sẽ được hoàn tác (ROLLBACK).
2. Kiểm tra tính hợp lệ của mã giảm giá:
   * Gọi hàm is\_valid\_voucher để kiểm tra xem mã giảm giá có thể áp dụng cho đơn hàng liên quan đến thanh toán không.
   * Nếu mã giảm giá không hợp lệ, Procedure dừng lại và báo lỗi: *"Invalid voucher, try another voucher!"*.
3. Kiểm tra quyền sở hữu mã giảm giá:
   * Gọi hàm has\_voucher để xác minh xem người dùng có mã giảm giá trong kho voucher của họ hay không.
   * Nếu không có, Procedure báo lỗi: *"The user does not have the voucher, try another voucher!"*.
4. Lấy thông tin người dùng và thanh toán:
   * Gọi hàm payments\_user để xác định người dùng liên quan đến giao dịch thanh toán.
   * Nếu không tìm thấy người dùng, Procedure dừng lại và báo lỗi: *"User not found for this payment"*.
   * Lấy số tiền của giao dịch từ bảng payments.
5. Lấy thông tin mã giảm giá:
   * Truy vấn chi tiết mã giảm giá từ bảng vouchers, bao gồm tỷ lệ giảm giá (salesOff), mức giảm giá tối đa (maximumDiscount), và giá trị đơn hàng tối thiểu (minimumOrderValue).
6. Tính giá trị giảm giá:
   * Tính giá trị giảm giá (salesOffValue) dựa trên tỷ lệ giảm giá và số tiền thanh toán.
   * Đảm bảo rằng giá trị giảm giá không vượt quá mức giảm giá tối đa (maximumDiscount).
7. Kiểm tra điều kiện áp dụng:
   * Nếu số tiền thanh toán nhỏ hơn giá trị tối thiểu (minimumOrderValue), Procedure dừng lại và báo lỗi: *"Invalid payment amount to apply voucher, try another voucher!"*.
8. Cập nhật giao dịch và kho voucher:
   * Cập nhật giao dịch trong bảng payments với mã giảm giá và số tiền sau khi trừ giảm giá.
   * Giảm số lượng mã giảm giá trong bảng voucherStorage của người dùng.
9. Kết thúc giao dịch:
   * Nếu tất cả các bước trên thành công, Procedure hoàn tất giao dịch với lệnh COMMIT.

Các kiểm tra và bảo vệ

* Tính hợp lệ của mã giảm giá: Đảm bảo rằng mã giảm giá được áp dụng đúng cho đơn hàng liên quan.
* Quyền sở hữu mã giảm giá: Đảm bảo người dùng có quyền sử dụng mã giảm giá.
* Giá trị đơn hàng tối thiểu: Đảm bảo số tiền thanh toán đáp ứng điều kiện mã giảm giá.
* Giá trị giảm giá tối đa: Đảm bảo số tiền giảm giá không vượt quá giới hạn.

**VII. Transaction**

Transaction cấp mã giảm giá cho người dùng mới.

Transaction này được thiết kế để:

1. Thêm một người dùng mới vào hệ thống.
2. Kiểm tra và thêm mã giảm giá dành cho người dùng mới (nếu chưa tồn tại trong hệ thống).
3. Cấp mã giảm giá cho người dùng vừa tạo và lưu trữ trong bảng voucherStorage.

Luồng hoạt động

1. Bắt đầu transaction:
   * Transaction bắt đầu với lệnh START TRANSACTION để đảm bảo các thao tác được thực hiện một cách nhất quán. Nếu có lỗi xảy ra ở bất kỳ bước nào, transaction sẽ được hoàn tác (ROLLBACK).
2. Thêm người dùng mới:
   * Thêm một bản ghi vào bảng users với thông tin chi tiết của người dùng mới:
     + Username: hoangsonbandon.
     + Email: hs.hoangson15062005@gmail.com.
     + Ngày sinh: 2005-06-15.
     + Địa chỉ: Hà Đông, Hà Nội.
     + Trạng thái hoạt động là "Yes".
3. Kiểm tra và thêm mã giảm giá:
   * Kiểm tra xem mã giảm giá "New Comer Voucher" đã tồn tại hay chưa:
     + Nếu chưa tồn tại, thêm một bản ghi mới vào bảng vouchers:
       - Giảm giá: 50% (tương ứng với trường salesOff).
       - Giá trị đơn hàng tối thiểu: 100,000 VNĐ.
       - Mức giảm giá tối đa: 1,000,000 VNĐ.
     + Nếu đã tồn tại, bỏ qua bước này.
4. Lưu trữ mã giảm giá trong voucherStorage:
   * Thêm thông tin mã giảm giá cho người dùng mới vào bảng voucherStorage:
     + Số lượng mã giảm giá: 10.
     + Ngày nhận mã: Ngày hiện tại (CURDATE()).
     + Ngày hết hạn: Sau 30 ngày kể từ ngày nhận mã (DATE\_ADD(CURDATE(), INTERVAL 30 DAY)).
   * Liên kết được thực hiện dựa trên:
     + voucherID của mã "New Comer Voucher" trong bảng vouchers.
     + userID của người dùng mới từ bảng users.
5. Kết thúc transaction:
   * Thay vì hoàn tất (COMMIT), transaction bị hoàn tác (ROLLBACK) để hủy bỏ tất cả các thay đổi.

**VIII. Queries**

* Truy vấn phương thức thanh toán của các giao dịch: Liệt kê chi tiết các giao dịch thanh toán và phương thức thanh toán được sử dụng. Kết quả hiển thị toàn bộ thông tin từ bảng payments và paymentMethods, giúp phân tích cách người dùng thanh toán
* Truy vấn mã giảm giá được sử dụng cho giao dịch: Liệt kê thông tin giao dịch kèm theo mã giảm giá (nếu có) được sử dụng. Nếu giao dịch có mã giảm giá, thông tin chi tiết từ bảng vouchers sẽ được hiển thị. Nếu không có mã giảm giá, kết quả sẽ là NULL cho các cột từ vouchers.
* Truy vấn người dụng bị phạt: Tìm danh sách chi tiết của các người dùng đã bị phạt. Hiển thị thông tin từ bảng users của những người dùng có mã phạt trong bảng userPenalty.
* Truy vấn sản phẩm bán chạy nhất của mỗi cửa hàng: Tìm sản phẩm bán chạy nhất (dựa trên tổng số lượng đặt hàng) của từng cửa hàng. Hiển thị tên cửa hàng, tên sản phẩm, và tổng số lượng bán ra của sản phẩm bán chạy nhất trong từng cửa hàng. Sử dụng MAX(totalQuantity) để xác định sản phẩm có doanh số cao nhất trong mỗi cửa hàng.
* Truy vấn số lần bị phạt của từng người dùng: Đếm số lần bị phạt của từng người dùng. Trả về userID và tổng số lần bị phạt (totalPenalty) của từng người trong bảng userPenalty.