**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN Công nghệ thông tin và truyền thông**

**\*\*\*\*\*\*\*\*\*\***

****

**BÁO CÁO THỰC HÀNH**

**ĐỀ TÀI: Trang bán hàng online**

**Giáo viên hướng dẫn : TS. VŨ TUYẾT TRINH**

**Nhóm thực hiện : Nhóm 4**

**Lương Đức Dương 20180057**

**Nguyễn Hoàng Long 20184139**

**Đỗ Tuấn Hoàng 20184107**

**MỤC LỤC**

**Chương 1. Mục đích đề tài**

**Chương 2. Phân tích và thiết kế hệ thống**

* 1. Người dùng hệ thống

2.1.1 Người bán hàng

2.1.2 Người mua hàng

2.1.3 Thiết kế

2.2 Thiết kế bán hàng

2.3 Thiết kế mua hàng

2.4 Chức năng Vip

2.5 Thiết kế lịch sử giao dịch

2.5 Thiết kế cơ sở dữ liệu thực thể liên kết

2.6 Thiết kế cơ sở dữ liệu quan hệ

**Chương 3. Chức năng hệ thống và phân công nhiệm vụ**

3.1 Chức năng người mua

3.2 Chức năng người bán

3.3 Chức năng hệ thống chạy theo lịch

3.4 Phân công nhiệm vụ cá nhân

3.5 Source code

**Chương 4. Truy vấn chính mỗi cá nhân**

4.1 Lương Đức Dương

4.2 Nguyễn Hoàng Long

4.3 Đỗ Tuấn Hoàng

**Chương 5. Nhận xét, đánh giá**

**Chương 1. Tổng quan dự án**

* 1. **Mục đích của dự án**
* Xây dựng hệ thống bán hàng online, ai cũng có thể đăng bán sản phẩm của mình và mua sản phẩm của người khác bán
* Không yêu cầu người bán hàng phải là người bán hàng chuyên nghiệp và phải bán nhiều sản phẩm. Bạn có sản phẩm muốn bán (quần áo mua nhưng không dùng, đồ 2nd, kinh doanh kiếm thêm thu nhập) đều có thể đăng kí trở thành người bán hàng

**Chương 2. Phân tích và thiết kế hệ thống**

* 1. **Người dùng hệ thống**

Hệ thống bao gồm 2 đối tượng người dùng đó là: người bán hàng và người mua hàng

**2.1.1** **Người mua hàng**

- Là người dùng mặc định của hệ thống, mỗi người dùng sau khi đăng kí với hệ thống xong đều có chức năng mua hàng và là người mua hàng

- Khi đăng kí người dùng cần cung cấp: Tên, email, điện thoại, username, địa chỉ. Hệ thống sẽ cung cấp cho người dùng một id cố định để định danh người dùng

**2.1.1.1 Địa chỉ**

Địa chỉ người dùng nhằm mục đích giao hàng thông qua các đơn vị vận chuyển vì vậy cần phải chính xác đến từng khu vực nhỏ và có thể thêm dấu hiệu nhận biết khu vực. Ví dụ như: gần phố đi bộ, gần Lăng Bác,… vì vậy trường địa chỉ giao hàng mặc định sẽ là đoạn text do người dùng tự điền

- Tuy nhiên địa chỉ này cũng quan trọng trong việc tính toán tiền phí giao hàng. Tiền phí giao hàng sẽ thay đổi theo quận, cùng hoặc khác thành phố.

- Từ đó nhóm quyết định chia địa chỉ thành 2 phần:

* Phần địa chỉ chi tiết: để người dùng tự điền
* Phần khu vực: bao gồm quận, thành phố có sẵn và người dùng phải chọn giữa những khu vực có sẵn này

Diagram

Description automatically generated

**2.1.2 Người bán hàng**

- Người bán hàng là người dùng đã đăng kí với hệ thống. Và đăng kí thêm chức năng mua hàng.

- Vì vậy người bán cũng là người mua vì người dùng cứ đăng kí với hệ thống là có chức năng mua hàng. Người bán có đầy đủ các thuộc tính và chức năng của người mua và có thêm các tính năng, thuộc tính khác giành riêng cho người mua hàng.

**2.1.3 Thiết kế**

**2.1.3.1** Chia người bán và người mua thành 2 đối tượng riêng biệt

Có 2 cách mà nhóm đã nghĩ đến đó là:

1. Người bán, người mua là 2 đối tượng riêng biệt không liên quan gì đến nhau

2. Người bán, người mua là 2 đối tượng được thừa kế từ một đối tượng chung đó là người dùng

- Về cách đầu tiên, vì người bán và người mua sẽ có rất nhiều thuộc tính giống nhau. Yêu cầu của hệ thống của nhóm đặt ra đó là người bán thì chắc chắn là người mua. Như vậy đối với người bán trong hệ thống, các thuộc tính của họ sẽ bị lặp lại 2 lần. Và trong trường hợp người bán sửa thông tin của mình thì thông tin ở bên người mua tương ứng cũng phải sửa theo.

- Cách thứ 2 sẽ loại bỏ được các khó khăn của cách 1. Tuy nhiên để người mua cũng có thể có các chức năng của người bán, sẽ phải phân quyền cho người mua.

=> Từ đó nhóm quyết định sử dụng cách sau đây cho việc thiết kế cơ sở dữ liệu

**2.1.3.2** Người bán là một đối tượng kế thừa từ người mua

- Cách này giảm dư thừa dữ liệu và không cần phân quyền cho người bán các chức năng của người mua.

- Hệ thống không yêu cầu phân biệt rạch ròi 2 đối tượng này

Diagram

Description automatically generated

* Các thuộc tính riêng của seller, nhóm chưa nghĩ ra.
  1. **Bán hàng**

Người bán hàng đăng sản phẩm lên bán. Sau đây là những cách thiết kế nhóm nghĩ đến

* + 1. **Người bán hàng bán các sản phẩm thông qua liên kết supply, sản phẩm là một đối tượng của hệ thống**

Diagram

Description automatically generated

* Vì một sản phẩm có thể bán bởi nhiều người dùng, chỉ khác về giá và số lượng. Để sản phẩm là một đối tượng trong hệ thống sẽ tránh được việc lặp đi lặp lại thuộc tính của những sản phẩm giống nhau bán bởi người bán khác nhau
* Tuy nhiên với cách này người bán sẽ gặp khó khăn trong việc quản lý những sản phẩm mình đang bán. Khó khăn trong việc áp dụng chương trình gì đó lên toàn bộ sản phẩm của mình như (giảm giá, khuyến mại, …)
  + 1. **Người bán quản lý một shop riêng, shop bán các sản phẩm**
* Shop là một đối tượng trong hệ thống, có các thuộc tính:
  + id\_shop chính là id của người bán mà quản lý shop. Vì mỗi người bán chỉ sở hữu một shop duy nhất vì vậy có thể lấy luôn id\_seller làm khoá
  + Tên shop: do người dùng tự đặt
  + Ngày mở shop
  + Địa chỉ: Vì mua hàng online nên những shop ở gần, người mua sẽ nhận được hàng sớm hơn.
    - Địa chỉ của shop không thể lấy luôn địa chỉ giao hàng của người bán vì người bán có thể đặt shop ở vị trí khác
    - Địa chỉ của shop chỉ cần chi tiết đến quận huyện
  + Vip: một chức năng đặc biệt dành cho người bán sẽ được nói ở phần sau
* Thiết kế thực thể shop

Diagram

Description automatically generated

* Thiết kế bán hàng

Diagram

Description automatically generated

**Đây là thiết kế bán hàng nhóm sử dụng trong hệ thống**

* 1. **Mua hàng**

Người mua hàng mua các sản phẩm được bán trên hệ thống. Để tiện lợi cho người mua hàng, mỗi người mua hàng sẽ có các giỏ hàng riêng, các sản phẩm cùng một người mua ở cùng một shop sẽ tính trong cùng một giỏ hàng.

* Thiết kế giỏ hàng ban đầu của nhóm:

Diagram

Description automatically generated

* Tuy nhiên thiết kế như thế này sẽ không thể chắc chắn sản phẩm ở trong giỏ hàng có đúng là được shop đó cung cấp hay không ? Hay shop đó có cung cấp sản phẩm này hay không
* Nhóm quyết định Supply là thực thể có id\_supply. Thiết kế giỏ hàng sau khi thực thể hoá Supply:

Diagram

Description automatically generated

* 1. **Vip**

Vip là một chức năng đặc biệt dành riêng cho người bán và chỉ người bán mới có thể đăng kí Vip. Vip không phải là vĩnh viễn mà có thời hạn. Hết hạn, người dùng phải gia hạn nếu không sẽ bị huỷ

* + 1. Chức năng VIP

Đưa sản phẩm người bán lên đầu trang trong quá trình tìm kiếm

* + 1. Thiết kế VIP
* **C1**: **Vip là một thuộc tính của người bán (isVip : true hoặc false)**

Với cách này sẽ khó khăn khi thực hiện yêu cầu của chức năng.

* + Vì khi muốn biết một sản phẩm do shop này cung cấp có phải Vip hay không?
* Ta phải xem shop này do ai quản lý rồi xem người đó có phải là Vip hay không.
* Việc check xem supply\_id đó có phải là vip hay không đã tương đối phức tạp. Chưa kể đến chức năng của Vip đưa những sản phẩm người bán Vip lên đầu.
* **C2**: **Vip là một thuộc tính của supply**

Với cách này, với mỗi id\_supply, sẽ có thêm thuộc tính isVip (bằng true hoặc false).

* Từ đó ta biết được ngay sản phẩm này do shop kia cung cấp có phải là Vip hay không.
  + Tuy nhiên vì Vip có thời hạn và có đăng kí mới. Vì vậy, khi một người dùng đăng kí Vip mới hoặc một người dùng huỷ bỏ chức năng hoặc không gia hạn
* Sẽ phải sửa thuộc tính isVip của từng sản phẩm do người đó bán gây tốn kém tài nguyên hệ thống
* **C3**: **Vip là một thuộc tính của Shop** (**Nhóm sử dụng cách này**)
  + Với cách này, muốn biết supply này có phải là vip hay không? Chỉ cần kiếm tra xem shop có phải là Vip hay không
  + Khi thêm mới Vip hoặc gia hạn Vip chỉ cần sửa một thuộc tính isVip của shop ấy (true hoặc false)
    1. Thực hiện chức năng

Đưa sản phẩm Vip lên đầu. Làm sao để đưa các sản phẩm do người dùng Vip cung cấp lên đầu trong quá trình tìm kiếm ?

* Cách đầu tiên nhóm nghĩ ra đó là tìm trước các supply do người dùng Vip cung cấp, sau đó tìm những supply còn lại rồi hợp 2 kết quả vào.
* Phải tìm 2 lần làm tốn kém tài nguyên hệ thống cộng thêm phép hợp 2 bảng. Chưa chắc 2 bảng sau khi hợp những sản phẩm vip đã ở đầu
* Cách sau đó nhóm sử dụng đó là: chuyển thuộc tính isVip từ boolean thành int với giá trị:
  + 1 tương ứng với true
  + 0 tương ứng với false
* Khi tìm sản phẩm chỉ cần sắp xếp theo thuộc tính isVip theo DESC thì những sản phẩm Vip sẽ tự khắc lên đầu tiên
* Lợi ích nếu sau này muốn mở rộng chức năng Vip thành superVip hay megaVip (vip càng cao thì sản phẩm càng lên đầu). Chỉ cần quy định SuperVip tương ứng với 2, megaVip tương ứng với 3 là được

Diagram

Description automatically generated

* 1. **Lịch sử giao dịch**

Lịch sử giao dịch sẽ lưu lại lịch sử mua hàng và bán hàng của các Shop:

* Người bán có thể xem lịch sử bán hàng và mua hàng của mình
* Người mua có thể xem lịch sử mua hàng của mình
  + 1. Các thuộc tính của giao dịch:
* Id giao dịch
* Id product
* Id Shop
* Id người mua
* Quantity
* Ngày mua
* Tổng tiền phải trả

Diagram

Description automatically generated

* 1. **Tổng hợp thiết kế CSDL thực thể liên kết**

Diagram

Description automatically generated

**2.7** **Thiết kế cơ sở dữ liệu quan hệ**

Diagram

Description automatically generated

**Chương 3. Chức năng hệ thống và phân công nhiệm vụ**

* 1. **Đối với người mua hàng**
* Chức năng tìm kiếm sản phẩm, Shop, theo các thuộc tính:
  + Tên
  + Khu vực
  + Giá
  + Hãng sản xuất
  + Năm sản xuất
  + Loại sản phẩm
  + Tìm kiếm tích hợp các thuộc tính
* Chức năng thêm hàng vào giỏ và thanh toán
  + Người mua hàng thêm hàng vào giỏ và tính tiền sau cùng
  + Mỗi người dùng có thể có nhiều giỏ hàng
  + Mỗi giỏ hàng có thể chứa nhiều sản phẩm nhưng chỉ được cung cấp bởi 1 shop
* Chức năng thanh toán giỏ hàng
* Chức năng tra cứu lịch sử mua hàng
* Chức năng đăng kí mở cửa hàng:
  + Bất kì người dùng nào cũng có thể đăng kí với hệ thống để trở thành người bán hàng
  1. **Đối với người bán hàng**
* Chức năng đổi tên Shop, đóng Shop
* Chức năng đăng bán sản phẩm:
  + Người dùng khi đăng bán sản phẩm, hệ thống sẽ gợi ý những sản phẩm đã có sẵn trong hệ thống
  + Nếu sản phẩm người dùng cần thêm chưa có sẵn. Người dùng sẽ phải tự thêm các thuộc tính cho sản phẩm của mình:
    - Tên
    - Loại
    - Nhà sản xuất
    - Năm phát hành
  + Người dùng có thể không thêm hết các trường thuộc tính nhưng nó sẽ gây bất lợi cho chính sản phẩm của họ trong quá trình tìm kiếm
* Chức năng truy vấn tất cả sản phẩm người dùng đang bán
* Chức năng thay đổi thông tin hoặc xoá sản phẩm đăng bán:
  + Người dùng có thể đổi về giá và số lượng sản phẩm bán
  + Nếu người dùng muốn thay đổi thông tin về sản phẩm ví dụ như: năm phát hành, hãng sản xuất, … thì phải xoá sản phẩm cũ đi và thay thế bằng sản phẩm mới
* Chức năng đăng kí Vip và gia hạn Vip: Người bán đăng kí Vip thông qua một giao diện đăng kí
  1. **Chức năng hệ thống chạy theo lịch**
* Tự động tăng, giảm số lượng sản phẩm khi có người mua thêm hoặc bớt hàng khỏi giỏ.
* Tự động check Vip sắp hết thời hạn và yêu cầu gia hạn. Xoá Vip của các tài khoản hết hạn, thay đổi trường isVip của Shop tương ứng
* Xoá các sản phẩm không được Shop nào bán
  1. **Phân công nhiệm vụ cá nhân**

### **Lương Đức Dương:**

## Tìm gian hàng theo khu vực, tên

## Tìm kiếm tích hợp (tích hợp tất cả các đặc điểm để tìm kiếm)

## Thêm hàng vào giỏ, loại bỏ hàng khỏi giỏ

## Tạo trigger để tự động trừ và cộng vào số lượng hàng trong supply

* Tìm tất cả sản phẩm shop đang bán

## Thay đổi thông tin sản phẩm của shop đang bán

## Tạo trigger để tự thay đổi trong isVip của shop khi thêm hoặc xoá bản ghi trong bảng Vip

## Tìm những sản phẩm không được shop nào bán và xoá

### **Nguyễn Hoàng Long**

* + Tìm theo loại hàng (điện thoại, ….)
  + Tìm theo giá thấp -> cao
  + Tìm theo giá trong khoảng chỉ định
  + Xoá sản phẩm đang bán
  + Thêm sản phẩm bán
  + Đăng kí vip, Gia hạn vip
  + Chức năng thanh toán (lấy sản phẩm trong cart, đưa ra thanh toán, gộp theo từng shop)

## Check vip (Nếu tài khoản đăng kí vip quá hạn thì xoá khỏi bản Vip)

### **Đỗ Tuấn Hoàng**

* + Tìm theo tên sản phẩm
  + Tìm theo năm sản xuất
  + Tìm theo hãng sản xuất (Apple, Samsung)
  + Tìm sản phẩm top mua hàng
  + Tìm theo giá cao -> thấp
  + Thay đổi tên gian hàng
  + Mở gian hàng
  + Đóng gian hàng
  1. Source Code

Link đến source code của project:

[GitHub](https://github.com/DuongLD-140800/Project_CSDL_20201.git)

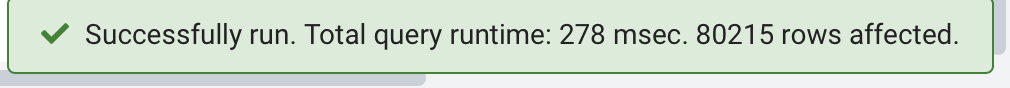
**Chương 4: Truy vấn chính mỗi cá nhân**

**4.1 Lương Đức Dương**

Câu truy vấn chính trong phần công việc của em đó là: Tìm kiếm tích hợp theo tất cả các thuộc tính.

**4.1.1** Khó khăn khi thực hiện câu truy vấn

* Khi thực hiện tìm kiếm theo tất cả các thuộc tính phải tiến hành nối 4 bảng: locations, shops, supply, products
* Tuy nhiên kích thước của các bảng rất lớn, chỉ việc nối các bảng lại với nhau đã tốn rất nhiều tài nguyên hệ thống chứ chưa nói đến sau khi nối xong phải tìm kiếm theo các điều kiện cho trước
* Đây là thời gian server cần để nối 4 bảng lại với nhau



* Số lượng thuộc tính cần tìm kiếm rất nhiều

**4.1.2** Các cách thực hiện

**C1.** Vì thời gian dùng để nối bảng rất lớn cho nên phải làm số lượng bản ghi của mỗi bảng nhỏ đi trước khi thực hiện phép nối.

* + Để làm cho số lượng bản ghi nhỏ đi trước khi nối, nhóm đã có sẵn các câu lệnh để tìm kiếm theo từng thuộc tính đơn lẻ
* Chỉ cần nối các kết quả tìm kiếm đơn lẻ lại với nhau sẽ được kết quả cuối cùng
  + Tuy nhiên sau khi thực hiện theo cách trên, thời gian thực hiện truy vấn thậm chí còn lớn gần gấp đôi phép nối 4 bảng
  + Lý do:
    1. Số lượng thuộc tính cần tìm kiếm rất nhiều

=> số lượng hàm tìm kiếm rất nhiều

* + 1. Server nhóm sử dụng chỉ có 2 nhân xử lý. Vì vậy tại cùng một thời điểm chỉ có tối đa 2 hàm được chạy đồng thời

* + Vì vậy thời gian để ghép nối bảng không nhiều nhưng thời gian chạy các hàm tìm kiếm đơn lẻ lại chiếm rất nhiều tài nguyên của hệ thống

**C2.** Vì server chỉ có 2 nhân sử lý vì vậy thay vì phải chạy rất nhiều hàm tìm kiếm đơn lẻ thì số hàm tìm kiếm phải rút về gần 2 nhất. Em quyết định sẽ tạo ra 3 hàm tìm theo từng bảng:

* + Shops join với locations: vì số lượng bản ghi của 2 bảng này nhỏ (locations và shops chỉ ở cỡ vài trăm bản ghi). Vì vậy thời gian nối trực tiếp 2 bảng cũng không lớn hơn nhiều so với việc tìm trên từng bảng rồi mới nối. Đồng thời giảm số lượng hàm con phải thực hiện từ 4 => 3
  + Supply
  + Products

Cách này thoả mãn:

* + Số lượng hàm con phải thực hiện gần với số nhân xử lý của server nhất
  + Tài nguyên dùng để ghép nối bảng đã giảm đi đáng kể so với việc nối trực tiếp 3 bảng

=> Em sử dụng cách này trong hệ thống

* + 1. Thực hiện phương án đề ra

**4.1.3.1** Hàm tìm kiếm trên bảng shops và locations

Graphical user interface, text

Description automatically generated

* Khi thực hiện kiểm tra điều kiện, em đều cho phép đầu vào bằng rỗng. Nhằm trường hợp người dùng không điền đủ tất cả thuộc tính thì vẫn có thể tìm được
* Vì tìm kiếm trên thuộc tính city, district, name\_shop. Em đã thêm 3 index tương ứng với mỗi thuộc tính

A picture containing text

Description automatically generated

* Khi người dùng nhập tên shop để tìm kiếm, người dùng chưa chắc nhớ chính xác tên shop mà chỉ nhớ một số kí tự vì vậy có hai cách tìm:
  + 
  + 
  + Tuy nhiên cách đầu tiên không tận dụng được index đã tạo vì vậy em sử dụng cách 2 cho hàm này.

**4.1.3.2** Hàm tìm kiếm trên bảng products và hàm tìm kiếm trên bảng supply

* Hàm trên bảng supply

Graphical user interface, text

Description automatically generated

* Hàm trên bảng products

Text

Description automatically generated

* Tạo index trên supply(price) và products(name), products(manufacturer), products(type).
* Không tạo index trên thuộc tính thời gian sản xuất bởi vì sử dụng hàm “date\_part” sẽ không tận dụng được index đã tạo và em cũng chưa nghĩ ra cách để kiểm tra năm mà có thể tận dụng được index

**4.1.3.3** Ghép các hàm tìm kiếm lại và order theo giá nếu đầu vào yêu cầu

Graphical user interface, text, application

Description automatically generated

* Để thực hiện chức năng Vip đó là đưa các sản phẩm Vip lên đầu, chỉ cần sắp xếp theo thuộc tính is\_vip theo thứ tự giảm dần
* Thời gian thực hiện truy vấn: shop ở ‘Hà Nội’ có tên ‘Shop1’, mức giá nhỏ hơn 10 triệu, với điều kiện giá cao lên trước và sản phẩm có tên ‘Iphone’ là:

Graphical user interface, text, application, table, email

Description automatically generated with medium confidence

* Vì server có 2 nhân nên để các hàm tìm kiếm trên 2 bảng với số bản ghi lớn: supply, products lên trước trong mệnh đề FROM sẽ giảm được thời gian tìm kiếm

Text

Description automatically generated

* Thời gian tìm kiếm sau khi đảo vị trí các hàm con với điều kiện tìm kiếm tương tự là:

Graphical user interface, text, application

Description automatically generated

* Vì dữ liệu test được tạo ra bằng script vì vậy tên của các sản phẩm và shop sẽ không đa dạng và tương đối giống nhau.Vì giá trị của các trường thuộc tính tương đối giống nhau nên khi index được tạo, cây btree sẽ rất sâu chứ không rộng
* Index được tạo ra chưa thể hiện được ưu việt của nó. Thời gian thực hiện hàm khi đã tạo index và chưa tạo index hơn kém nhau không đáng kể
  + Trong thực tế, tên của sản phẩm và shop sẽ đa dạng hơn, cây btree được tạo cho index sẽ rộng về bề ngang. Từ đó rút ngắn được thời gian tìm kiếm
    1. Kết luận
  + Thời gian thực hiện tích hợp tìm kiếm đã giảm được 4 lần và tiềm cận thời gian tìm kiếm chỉ trên 1 thuộc tính đơn lẻ
    1. Thời gian ghép nối 4 bảng: 273 ms
    2. Thời gian tìm kiếm tích hợp: 66 ms
    3. Thời gian tìm kiếm chỉ trên 1 thuộc tính đơn lẻ: 56 ms
  + Hàm cho phép tìm kiếm khi người dùng không nhập đủ các trường điều kiện tìm kiếm
  1. **Nguyễn Hoàng Long**

Tìm theo loại sản phẩm:

* + Tìm sản phẩm sẽ cho ra kết quả là thông tin sản phẩm, cửa hàng bán, số lượng bán, giá bán (với điều kiện các shop có đăng ký vip sẽ được ưu tiên đưa lên đầu danh sách tìm kiếm) nên ta sẽ kết hợp 3 bảng products, shops và supply. Ta có thể có nhiều cách ghép bảng:
* Ghép 3 bảng lại với nhau rồi tìm kiếm theo điều kiện. Tuy nhiên với số lượng bản ghi lớn thì cách này không thực sự hiệu quả.
* Ghép bảng products và bảng supply lại với nhau, tìm kiếm theo điều kiện, sau đó ghép với bảng shop đã được sắp xếp (cho các shop là vip lên đầu kết quả tìm kiếm). Cách này đã giúp thời gian tìm kiếm nhanh hơn.
* Tìm kiếm theo điều kiện trên bảng products, sau đó ghép với bảng supply và bảng shops đã sắp xếp. khi chạy thử truy vấn cho thấy cách này đem lại hiệu quả tốt nhất.
* Để giảm thời gian tìm kiếm thêm nữa, em đã sử dụng index cho cột type của bảng products, thời gian tìm kiếm đã giảm đi đôi chút.
* Trong trường hợp người dùng nhập không đúng với tên loại sản phẩm, để tăng độ chính xác trong kết quả tìm kiếm, em đã đưa từ khóa tìm kiếm và các giá trị trong thuộc tính type về cùng 1 format:
* Đưa các chuỗi kí tự về dạng chuỗi kí tự thường.
  + Sau đó em tạo 1 hàm chuẩn hóa chuỗi để loại bỏ các dấu cách trong chuỗi.
  + So sánh các chuỗi tương tự với điều kiện type LIKE ‘%s’ || \_type || ‘%s’.

Với cách này, kết quả tìm kiếm đã trở nên chính xác hơn. Tuy nhiên nếu như số lượng bản ghi lớn, việc chuẩn hóa chuỗi trở nên rất mất thời gian. Trong thực tế khi em sử dụng, thời gian tìm kiếm tăng lên 3-4 lần. vì vậy em sẽ bỏ qua bước chuẩn hóa chuỗi.

* 1. **Đỗ Tuấn Hoàng**

Tìm theo tên sản phẩm

* Hàm tìm theo tên sản phẩm sẽ trả về tên shop bán từ bảng shops, số lượng và giá tiền của sản phẩm từ bảng supply, loại sản phẩm, hãng sản xuất và năm sản xuất từ bảng products (Với điều kiện các shop có đăng kí vip sẽ được ưu tiên đưa lên đầu trong danh sách tìm kiếm), nên ta sẽ kết hợp 3 bảng shops, products và supply. Ta có thể có nhiều cách ghép bảng:
  1. Ghép 3 bảng với nhau rồi tìm kiếm theo điều kiện.
  2. Ghép bảng products và bảng supply, sau đó tìm kiếm theo điều kiện (tên sản phẩm) và tiến hành ghép với bảng shops đã được sắp xếp (đưa shop đăng kí vip lên đầu kết quả tìm kiếm).
  3. Tìm kiếm theo điều kiện (tên sản phẩm) trên bảng products đồng thời tiến hành phép chiếu những trường cần trả về từ products (loại sản phẩm, hãng sản xuất, năm sản xuất), sau đó ghép với bảng kết quả của phép chiếu những trường cần trả về từ bảng supply (số lượng và giá tiền sản phẩm) sau đó thực hiện phép nối với bảng shops và sắp xếp đưa các shop đã đăng kí vip lên đầu
     + Nhận xét:
       - Cách thứ nhất: Thực hiện phép ghép giữa các bảng có số lượng bản ghi lớn sẽ làm tăng thời gian chạy và độ phức tạp của thuật toán.
       - Cách thứ hai: Thực hiện ghép bảng bảng products và bảng supply, sau đó tìm kiếm theo điều kiện (tên sản phẩm) đã làm giảm số lượng bản ghi của bảng mới được tạo (do thực hiện phép chọn) nên khi ghép với bảng shops độ phức tạp và thời gian chạy của thuật toán đã giảm. Tuy nhiên đây chưa phải là cách tối ưu nhất.
       - Cách thứ ba: Bằng việc áp dụng các chiến lược tối ưu tổng quát đó là đẩy phép chọn và chiếu lên thực hiện trước đồng thời nhóm dãy các phép chiếu rồi mới thực hiện ghép bảng. Cách thứ 3 đã cho ra kết quả tốt nhất.

**Chương 5: Nhận xét đánh giá project**

**5.1** Đối với nhiệm vụ phân tích và thiết kế hệ thống

* + Phân tích kĩ yêu cầu hệ thống
  + Xây dựng cơ sở dữ liệu phù hợp với yêu cầu hệ thống đặt ra

**5.2** Đối với triển khai chức năng

* Các chức năng nhóm phát triển đáp ứng đầy đủ yêu cầu chức năng của hệ thống đặt ra
* Có sự đầu tư tối ưu câu truy vấn đồng thời chỉ ra bằng lý thuyết và thực tiễn tính đúng đắn của tối ưu

**5.3** Điểm còn thiếu xót

* Các hàm, truy vấn được dùng để dọn dẹp cơ sở dữ liệu
  + Xoá các sản phẩm không được shop nào cung cấp
  + Xoá các người dùng hết hạn Vip
  + …

Theo yêu cầu của nhóm ban đầu là sẽ được chạy theo lịch (hàng ngày, hàng tuần)

* Tuy nhiên postgres không hỗ trợ một cách chính thức chạy theo lịch. Có một project đã được tạo ra để bổ sung chức năng này cho postgres:
  + [Pg\_cron](https://github.com/citusdata/pg_cron)
* Nhưng project này không tương thích với mọi nền tảng hệ điều hành. Vì vậy nhóm đã không áp dụng cho hệ thống của mình.