**Phần thực hiện truy vấn: Lương Đức Dương**

1. **Tìm theo khu vực: Tìm theo khu vực cần kết hợp 2 bảng: shops và location. Vì vậy có các cách kết hợp như sau:**
   * 1. Kết hợp 2 bảng trước rồi mới tìm xem shop nào phù hợp
     2. Chiếu lên bảng location trước rồi mới kết hợp với shops
     3. Chiếu lên bảng location để ra những id\_location phù hợp sau đó đối chiếu trên bảng shops

* Tuy nhiên sau khi sử dụng **explain(câu truy vấn)**, ta thấy được ở cả 3 cách postgres đều có chiến thuật thực hiện câu truy vấn như nhau. Chạy thực tế cũng cho thấy thời gian của cả 3 cách là tương đương. Chọn 1 trong 3 cách đều được

1. **Tìm theo tên:**
   * 1. Để quá trình tìm kiếm diễn ra nhanh hơn, em đã thêm index vào cột shops.name\_shop
     2. Người dùng có thể không nhớ hết tên shop. Vì vậy sẽ kiểm tra và đưa ra các shop có tên có substring trùng với từ khoá của người dùng
     3. Có 3 cách:
        1. Position(str\_input in name\_shop) > 0
        2. Name\_shop LIKE ‘%’ || str\_name || ‘%’
        3. Name\_shop LIKE str\_name || ‘%’

* Tuy nhiên sau khi explain câu lệnh, ta thấy được cách 1, 2 postgres sử dụng seq search do đó không tận dụng được index ta đã tạo. Vì vậy chọn cách 3

1. **Tìm kiếm tích hợp:**
   * 1. Tìm kiếm tích hợp tất cả các thuộc tính
     2. Đơn giản nhất sẽ là hợp 3 bảng shops, supply, products lại với nhau rồi tìm kiếm theo điều kiện đề bài ra. Tuy nhiên số lượng bản ghi của bảng supply và products là rất lớn vì vậy cách này không khả thi
     3. Ý tưởng ban đầu là lấy đầu ra của tìm kiếm theo từng thuộc tính rồi join các bảng với nhau. Vì đầu ra của mỗi thuộc tính sau khi tìm kiếm là nhỏ hơn rất nhiều so với đầu vào. Qua đó tiết kiệm chi phí ghép nối bảng đi rất nhiều
     4. Tuy nhiên sau khi explain câu truy vấn. Em thấy được server postgres chạy trên máy em chỉ sử dụng 1 nhân máy tính tương đương với 2 luồng xử lý. Vì vậy tại một thời điểm server chỉ có thể xử lý đồng thời 2 hàm ứng với đầu ra của 2 thuộc tính. Với số lượng thuộc tính lớn như vậy, cách này là không khả thi
     5. Phải làm sao để số bảng join cùng thời điểm là lớn hơn hoặc bằng 2, càng gần 2 càng tốt. Nhận thấy các thuộc tính hầu hết chỉ tìm trên 1 bảng riêng lẻ. Làm thêm 3 hàm tìm kiếm với 3 bảng (shops, supply, products) .Các thuộc tính tìm kiếm được chia theo các bảng. Khi thực hiện việc ghép nối, đẩy hàm tìm kiếm trên bảng supply, products lên trước.
2. **Thêm hàng vào giỏ**
   * 1. Check xem sản phẩm có tồn tại hay không
     2. Check số lượng sản phẩm có đủ hay không
     3. Tìm kiếm sản phẩm này trong giỏ hàng
        1. Nếu chưa có: chỉ việc thêm vào cart
        2. Nếu đã có rồi: số lượng sản phẩm mới phải bằng tổng số lượng sản phẩm cũ cộng với số lượng sản phẩm thêm vào
3. **Xoá hàng khỏi giỏ**
   * 1. Kiểm tra xem sản phẩm này còn được người bán đăng bán không (có thể sảy ra trường hợp người dùng chọn sản phẩm vào giỏ, sau đó người bán không bán sản phẩm đó nữa. Nhưng sản phẩm đó vẫn trong giỏ hàng người dùng)
   * Nếu không còn bán: xoá tất cả số lượng sản phẩm ra khỏi giỏ, thông báo với người dùng
     1. Kiểm tra xem sản phẩm có tồn tại trong cart không, nếu tồn tại thì số lượng có đủ để xoá hay không
     2. Nếu thoả mãn tiến hành xoá sản phẩm:
        1. Nếu số lượng sản phẩm xoá bằng với số lượng trong giỏ. Xoá sản phẩm đó trong giỏ
        2. Nếu số lượng sản phẩm xoá nhỏ hơn với số lượng trong giỏ. Trừ đi số lượng sản phẩm trong giỏ
4. **Tạo trigger để tự thay đổi bảng supply khi có người dùng muốn thêm hàng vào giỏ hay xoá hàng khỏi giỏ**
   * 1. Tạo 2 trigger function increseQuantity() và decreseQuantity() có trách nghiệm tự động cộng trừ trong bảng supply
     2. Tạo 2 trigger ứng với 2 function trên
5. **Thay đổi thông tin của sản phẩm đang bán**
   * 1. Hàm tìm các sản phẩm người dùng đang bán
        1. Check xem người dùng có phải seller hay không
        2. Nếu phải trả về danh sách sản phẩm người dùng bán
     2. Thay đổi thông tin sản phẩm đang bán
        1. Check xem người đó có đang bán sản phẩm này không. Nếu không hiện ra thông báo
        2. Nếu có. Xoá bản ghi cũ trong bảng supply đồng thời thêm bản ghi mới
6. **Check vip**
   * 1. Tạo index start\_date\_idx để đẩy nhanh tìm kiếm
     2. Có 2 cách check:
        1. Start\_date + use\_time < now()
        2. Start\_date < now () – use\_time
   * Tuy nhiên cách 1 không tận dụng được index đã tạo vì vậy dùng cách 2
     1. Chưa tìm ra cách lập lịch cho câu truy vấn này chạy hàng ngày
7. **Tạo trigger để tự thay đổi trong isVip của shop**
   * 1. Tạo 2 trigger function deleteVip() và addVip() chức năng tương ứng của mỗi function là update thuộc tính isVip của shop bằng 0 và 1
     2. Tạo 2 trigger tương ứng với 2 function trên với event là xoá 1 shop khỏi bảng Vip và thêm 1 shop vào bảng Vip
8. **Xoá các sản phẩm không được shop nào bày bán**
   * 1. Chiếu lên bảng supply để tìm ra các sản phẩm đang được bày bán kiểm tra trên bảng products bằng toán tử NOT IN
     2. Sử dụng NOT IN sẽ không tận dụng được index đã tạo. Tuy nhiên câu truy vấn này không được sử dụng thường xuyên