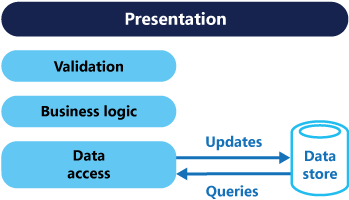
CQRS Design Pattern

Hiện tại mình đang muốn triển khai một app bán hàng với một số tính năng như thêm mới sản phẩm và tìm kiếm sản phẩm.

Trong chức năng tìm kiếm mình có tìm kiếm theo tên, tìm kiếm theo loại, tìm kiếm theo hãng, …

Với ý tưởng trên mình đã xây dựng được mô hình như sau

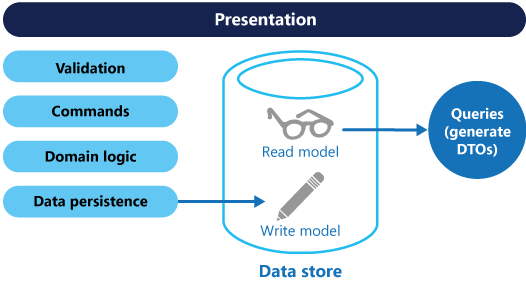


Nhưng mình lại thấy khi mình thực hiện truy vấn thì kết quả trả về chứa rất nhiều thông tin không cần thiết.

Vì mình đang dùng chung model của Thêm mới cho việc tìm kiếm sản phẩm.

Sau đó mình mới tìm kiếm được giải pháp là thay vì dùng chung model cho việc thêm mới và tìm kiếm thì mình tách biệt thành hai model khác nhau

Một model dành riêng cho việc thêm mới và một model dành riêng cho việc truy vấn (DTO)



Một thời gian sau khi triển khai app thì số luợng sản phẩm cũng tăng theo thời gian, đồng thời số lượng khách hàng thực hiện tìm kiếm sản phẩm trên app cũng tăng lên.

Với một lượng lớn khách hàng thực hiện tìm kiếm như thế thì mình cảm thấy tốc độ truy vấn để trả về thông tin cho khách hàng chậm đi đáng kể, và mỗi lần mình thực hiện thêm mới sản phẩm cũng ảnh hưởng trực tiếp đến việc tìm kiếm của khách hàng.

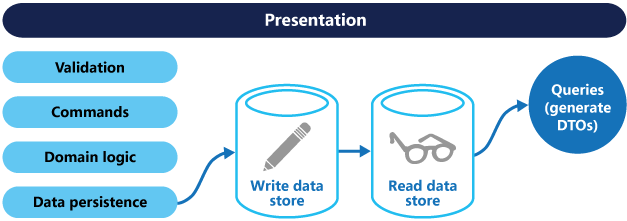
Sau một hồi suy nghĩ mình cảm thấy việc dùng chung cơ sở dữ liệu vừa để thực hiện thay đổi vừa để thực hiện truy vấn dữ liệu sẽ ảnh hưởng lớn đến hiệu suất của ứng dụng.

Và để đáp ứng được vời nhu cầu tìm kiếm càng lớn của người dùng thì việc sử dụng database SQL không còn phù hợp nữa.

Vì thế mình đã tách biệt thành hai cơ sở dữ liệu. Một cho việc thay đổi và một cho việc tìm kiếm.

Với việc tách biệt thế này ở cơ sở dữ liệu cho viêc thêm mới mình vẫn có thể dùng cơ sở dữ liệu dạng quan hệ (Write database) và dùng redis hay mongodb hoặc là elasticsearch để tối ưu hóa cho việc tìm kiếm (Read database).

Đồng thời mình cũng có thể tối ưu cấu trúc dữ liệu lưu ở Read database dành cho những mục đích khác nhau.



Sau một thời gian tìm hiểu thì mình thấy có một Design Pattern cũng đang làm tương tự đó là CQRS Design Pattern.

Vậy thì CQRS Design Pattern là gì?

CQRS viết tắt của Command Query Responsibility Segregation. Đây là một mô hình kiến trúc phân tách các hoạt động đọc dữ liệu (query) và sửa đổi dữ liệu (command) thành các thành phần riêng biệt.

Và CQRS khuyến khích sự phân tách rõ ràng giữa việc đọc và việc ghi của ứng dụng.

Với cấu trúc trên khi triển khai CQRS sẽ giúp ứng dụng có thể tối đa performance, scalability và security hơn.

CQRS gồm 2 thành phần chính là Command và Query.

Command đại diện cho các hoạt động ghi dữ liệu, bao gồm tạo mới, cập nhật hoặc xóa dữ liệu.

Query đại diện cho các hoạt động đọc dữ liệu, như truy vấn thông tin từ cơ sở dữ liệu.

Command và Query có thể được xử lý bởi các lớp hoặc module riêng biệt, tối ưu hóa cho mục đích cụ thể.

Khi áp dụng CQRS cần chú ý một số cái như sau:

Commands nên dựa trên nhiệm vụ, thay vì tập trung vào dữ liệu.

Commands có thể được đặt trên một hàng đợi để xử lý không đồng bộ, thay vì được xử lý đồng bộ.

Query không bao giờ sửa đổi cơ sở dữ liệu. Query trả về DTO không gói gọn bất kỳ domain nào.