

**GraphQL Và Ưu Điểm Nổi Bật So Với Rest API?**

Như chúng ta đã biết, để client và server giao tiếp được, trao đổi được với nhau, chúng ta thường dùng kỹ thuật gọi là REST API. REST hoạt động chủ yếu dựa vào [giao thức HTTP](https://codelearn.io/sharing/giao-thuc-https-covid19). Các hoạt động cơ bản nêu trên sẽ sử dụng những phương thức HTTP riêng.

* GET: lấy dữ liệu
* POST: tạo mới
* PUT: cập nhật (thay đổi)
* DELETE: Xóa dữ liệu

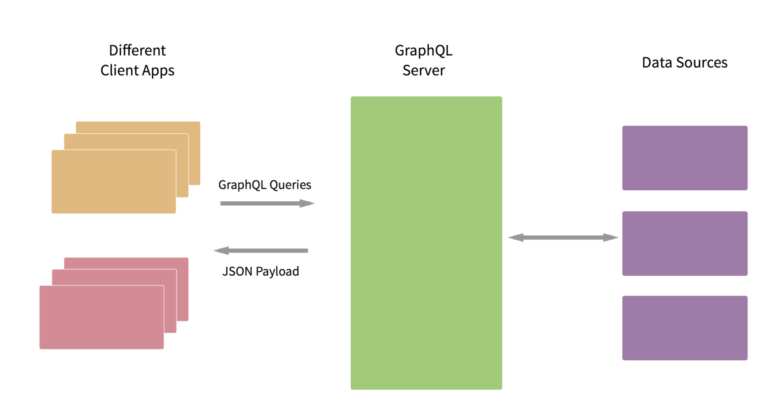
Những phương thức hay hoạt động này thường được gọi là CRUD tương ứng với Create, Read, Update, Delete – Tạo, Đọc, Sửa, Xóa. Mỗi chức năng tương ứng với mỗi phương thức sẽ phải tạo 1 endpoint (kiểu như 1 đường dẫn). REST API được sử dụng rất rộng rãi và nó dường như trở thành 1 tiêu chuẩn trong việc kết nối trao đổi dữ liệu giữa client và server khi phát triển phần mềm.

Dù là thế nào đi nữa thì REST API cũng bộc lộ những hạn chế của nó, nhất là khi đối với những hệ thống lớn. Và các bạn có nghĩ còn có các công nghệ khác kỹ thuật khác giúp ta khác phục những hạn chế của REST API không. Một ví dụ điển hình là **GraphQL**, hôm nay chúng ta cũng nhau tìm hiểu **GraphQL** nhé.

**GraphQL là gì?**

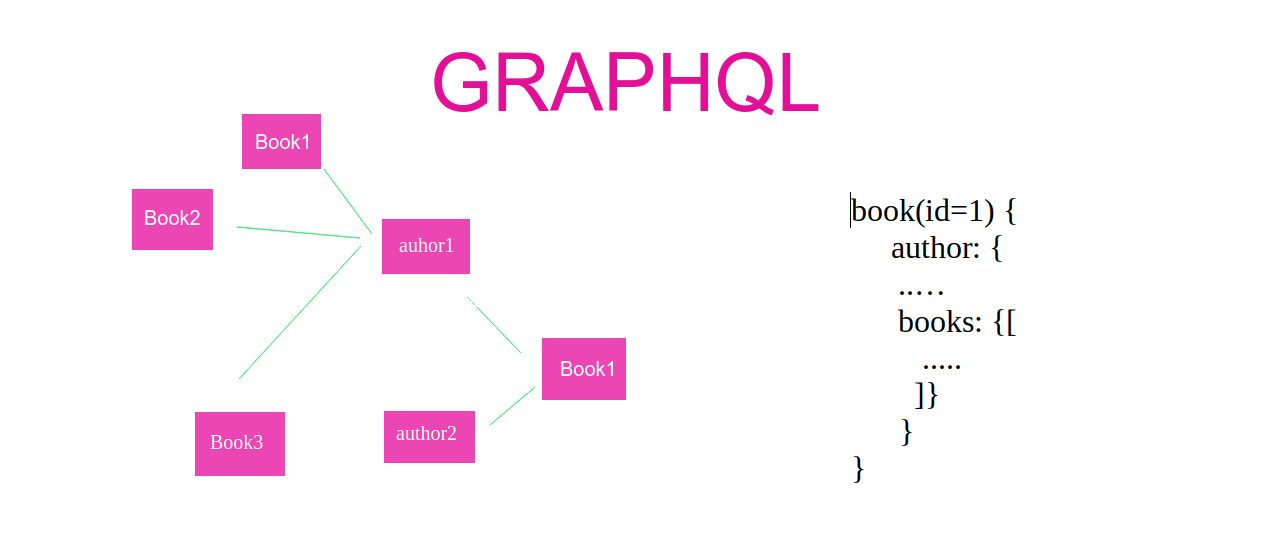
Khi mới nhìn vào tiêu đề chắc nhiều bạn nghĩ đây là một hệ quản trị cơ sở dữ liệu như SQL Server hay My Sql, nhưng không GraphQL không phải là hệ quản trị cơ sở dữ liệu. GraphQL là Graph Query Language do Facebook tạo ra cung cấp giao thức chung cho phép giao tiếp giữa client và server để cập nhật và lấy dữ liệu.

Khi client yêu cầu dữ liệu lên server, cấu trúc dữ liệu trả về không khô cứng mà nó sẽ thay đổi theo ngữ cảnh tùy theo cách mà client yêu cầu cần những dữ liệu gì. Những format của dữ liệu sẽ được mô tả sẵn ở server. Vì vậy khi yêu cầu dữ liệu lên server chúng ta chỉ cần duy nhât 1 endpoint.



**Tại sao lại đặt tên là GraphQL ??**

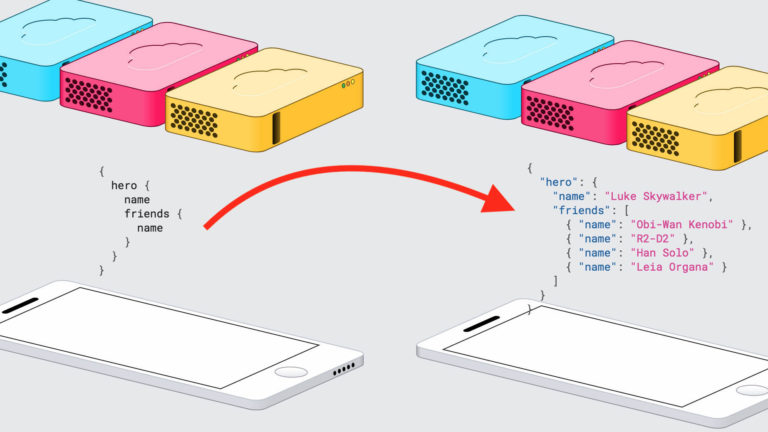
Khi các bạn định nghĩa mô tả dữ liệu ở trên server thì các mô tả đó có liên quan với nhau (ví dụ như khóa ngoại trong các CSDL như My Sql,..) tạo nên một sơ đồ các query có liên quan với nhau. Tóm lại nếu bạn vẽ ra các mối quan hệ đó bạn sẽ thấy nó trong giống sơ đồ. Đó là theo ý kiến của mình còn ý kiến của các bạn như nào thì cùng nhau góp ý nha.



**Tại sao nên sử dụng GraphQL?**

Đối với những server lớn, REST API để lộ khuyến điểm thiếu linh hoạt do dựa trên tài nguyên cố định. Vì vậy chúng ta phải cần có số lượng lớn endpoint, trong số đó có rất nhiều endpoint dư thừa và cũng phải kể đến việc quản lý chúng là điều không hề dễ. Một ví dụ đơn giản đó là Facebook, vì vậy Facebook đành phải nghĩ ra giải pháp giải quyết các vấn đề kể trên. Và Facebook đã chọn giải pháp chỉ sử dụng 1 endpoint duy nhất với khả năng tiếp nhận các query từ phía client và chỉ trả về những data tùy theo client cần. Nói đơn giản hơn, đây là truy vấn hướng client, cấu trúc dữ liệu không khô cứng 1 khuôn mẫu từ server (REST API) mà thay đổi theo từng ngữ cảnh sao cho hiệu quả nhất đối với client.

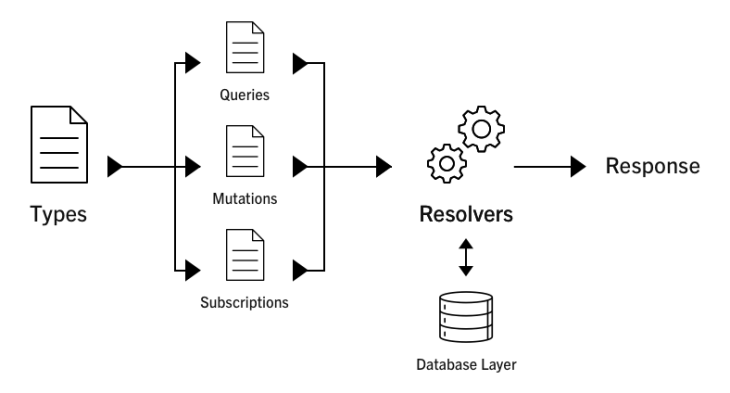
Ngoài ra, GraphQL hoàn toàn có thể create, update, delete, nhưng với cấu trúc sáng sủa và cấu trúc phân tầng nên lại càng thuận lợi cho lập trình viên phía client. Nó tương thích tốt với các cơ sở dữ liệu phổ biến hiện nay. GraphQL còn cung cấp websocket giúp việc client có thể lắng nghe dữ liệu từ server dễ dàng hơn cũng như phát triền các ứng dụng realtime trở nên không khó.



**Cấu trúc của GraphQL.**

GraphQL chia thành 3 phần chính :

* Query:  Các câu lệnh lấy dữ liệu (tương tự method GET trong REST API)
* Mutation: Các câu lệnh để thêm/sửa dữ liệu (tuơng tự method POST/PUT/DELETE trong REST API)
* Subscription: Câu lệnh dùng để lắng nghe sự kiên trên server, khi dữ liệu trên server được thay đổi nó cũng thay đổi theo nhằm cung cấp dữ liệu cho client một cách realtime nhất có thể. Ở điểm này REST API không có cửa để đấu lại với GraphQL.



**Ví dụ về GraphQL**

Hiện tại, mình đã dựng sẵn 1 server sử dụng GraphQL và định nghĩa sẵn các mô tả về dữ liệu để demo trước cho các bạn xem. Mình sẽ có 1 bài hướng dẫn về dựng server sử dụng GraphQL.

Trong hình đang sử dụng trình PlayGround của GraphQL để kiểm tra trước các chức năng, ngoài ra chúng ta cũng có thể sử dụng Postman.

Hình dưới mình sử dụng chức năng Mutation để thực hiện hàm createAuthor (cái này được mô tả trước ở trên server) với đầu vào gồm 3 trường: **FirstName**, **LastName** và **dbo** với mục đích tạo mới một tác giả. Và mong muốn của mình là sau khi tạo server sẽ trả lại cho mình 4 trường: **id**, **FirstName**, **LastName** và **dbo** của tác giả mình vừa tạo.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ví dụ về chức năng Query thực hiện hàm authors nhằm lấy danh sách các tác giả đã được tạo. Với mỗi tác giả mình mong muốn lấy 4 trường đó là **id**, **FirstName**, **LastName** và **dbo**. Kết quả server trả về dữ liệu cho mình với hình bên dưới.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Cũng là câu lệnh Query ở trên nhưng bây giờ mình chỉ muốn mỗi tác giả chỉ lấy 2 trường là **FirstName** và **LastName** thì mình làm như hình bên dưới.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Thật tuyệt phải không các bạn, tránh được việc dư thừa dữ liệu khi không cần cũng như chẳng cần thêm 1 endpoint nào khác.