* RavenDB lưu trữ dữ liệu không theo một lược đồ có định, nó có lược đồ tùy ý tùy biến. Nhưng điều đó không có nghĩa rằng chúng ta không nên dành nhiều thời gian để xem xét làm thế nào để thiết kế các document để đảm bảo rằng chúng ta có thể truy cập tất cả dữ liệu chúng ta cần để phục vụ các yêu cầu của người dùng một cách hiệu quả, đáng tin cậy và chi phí bảo trì ít nhất có thể.
* Lỗi điển hình nhất mà chúng ta mắc phải là cố gắng thiết kế mô hình dữ liệu của document database giống với cách chúng ta thiết kế mô hình dữ liệu trong cơ sở dữ liệu quan hệ.
* RavenDB lưu trữ dữ liệu không quan hệ. Cố gắng thiết kế theo mô hình quan hệ thì chúng ta sẽ có được nhiều kết quả tốt. Nhưng chúng ta sẽ đạt được kết quả vô cùng to lớn nếu sử dụng những điểm mạnh của cớ sở dữ liệu hướng document như là RavenDB.

**Document is not flat (không như nhau)**

* Documents, không giống như một dòng (row) trong RDBMS, document thì không giống nhau. Chúng ta không bị giới hạn chỉ lưu keys và values. Thay vào đó, chúng ta có thể lưu một đối tượng phức tạp như là một document. Nó có thể bao gốm arrays, dictionaries và trees. Không giống như mô hình dữ liệu quan hệ, nơi mà một dòng chỉ có thể chứa dữ liệu đơn giản và những cấu trúc dữ liệu phức tạp hơn cần được lưu trữ như là mối quan hệ, chúng ta không khó để lấy dữ liệu trong RavenDB.
* Xem ví dụ dưới đây:



* Trong cơ sở dữ liệu quan hệ, chúng ta cần không ít hơn 4 table để hiển thị dữ liệu trong một trang đơn (Posts, Comments, Tags, RelatedPosts)
* Sử dụng RavenDB, chúng ta lưu trữ tất cả thông tin chúng ta cần để làm việc như một document với cấu trúc bên dưới:



* Cấu trúc này cho phép chúng ta lấy mọi thông tin chúng ta cần để hiện thị lên trang chỉ trong một yêu cầu.

**Raven is not relation (phi quan hệ)**

* Khi bắt đầu dùng RavenDB , chúng ta sẽ gặp những vấn đề khi chúng ta cố gắng sử dụng các khái niệm của cơ sở dữ liệu quan hệ. Vấn đề chính đó là Raven không quan hệ. Tuy nhiên, nó thực sự nhiều hơn thế, đây chỉ là một lý do tại sao RavenDB không quan hệ.
* Raven xem mỗi document như một thực thể độc lập. Bằng cách làm như vậy, nó có thể tối ưu hóa cách các documents được lưu trữ và quản lý. Hơn nữa, một trong điểm tuyệt vời của Raven là Raven lưu trữ một lượng lớn dữ liệu (quá nhiều dữ liệu để lưu trữ trên một máy tính duy nhất)
* Raven hỗ trợ sharding, do đó không cần phải lưu trữ một nhóm dữ liệu liên hệ với nhau. Mỗi document là độc lập và có thể được lưu trữ trên bất kỳ shard nào của hệ thống.
* Một khía cạnh khác về bản chất không quan hệ của Raven là những documents được mong đợi sẽ có đầy đủ thông tin trong chính bản thân nó. Chắc chắn là bạn có thể lưu trữ tham chiếu đến các document khác, nhưng nếu bạn cần tham khảo document khác để hiểu document hiện tại có ý nghĩa gì thì có thể là bạn đã sử dụng RavenDB không đúng cách.
* Với Raven, chúng ta được khuyến khích nhóm tất cả thông tin chúng ta cần vào trong một document duy nhất. Nhìn vào ví dụ trên, trong cơ sở dữ liệu quan hệ, chúng ta sẽ có một bảng liên kết cho RelatedPosts chứa id của các bài viết liên kết. Nếu chúng ta muốn lấy Title của related posts, chúng ta cần phải join vào bảng bài viết một lần nữa. Chúng ta có thể làm điều đó trong Raven, nhưng đó không phải là phương pháp được đề xuất. Thay vào đó, như trong ví dụ trên, chúng ta nên bao gồm tất cả các thông tin cần thiết vào trong document. Sử dụng phương pháp này, chúng ta có thể hiển thị trang với chỉ một yêu cầu, dẫn đến hiệu suất tổng thể tốt hơn nhiều.

**Entities và Aggregates**

* Khi suy nghĩ về Raven lưu trữ các entities, chúng ta cần xem xét 2 điểm trước đó. Phương pháp được đề xuất là theo mẫu Aggregates từ cuốn sách Domain Driven Design. Một Aggregate Root chứa một vài thực thể, các loại giá trị, và kiểm soát tất cả truy cập các đối tượng được chứa trong nó. Những tham chiếu từ bên ngoài chỉ có thể tham chiếu đến Aggregate Root , không bao giờ được tham chiếu đến các đối tượng con của nó.
* Khi bạn áp dụng cách suy nghĩ này vào trong document database, mối tương quan giữa Aggregatre Root (trong DDD) và document trong Raven thì tự nhiên và dễ dàng làm theo. Một Aggregrate Root, và tất cả objects được nó giữ trong, là một document trong RavenDB.
* Điều này cũng giải quyết gọn gàng một vấn đề phổ biến với Aggregates: đi qua các đường đi từ Aggregates đến các đối tượng chúng ta cần một thao tác cụ thể, điều này tốn kém về số lượng lời gọi đến database. Trong Raven, nạp dữ liệu cho thực thể Aggragate chỉ với một lời gọi và kết hợp một document với đối tượng đầy đủ Aggregate Root thì ít tốn chi phí.
* Thay đổi sang Aggregate cũng dễ dàng hơn để kiểm soát, khi sử dụng RDBMS, thật khó khăn để đảm bảo rằng các yêu cầu đồng thời sẽ không vi phạm các business rules. Vấn đề là 2 yêu cầu riêng biệt có thể tác động đến 2 phần khác nhau của Aggregate, mỗi phần thì hợp lệ trong riêng nó, nhưng kết quả tổng hợp thì lại không hợp lệ. Điều này dẫn đến sử dụng  [coarse grained locks](http://martinfowler.com/eaaCatalog/coarseGrainedLock.html), cái mà khó thực hiện được khi sử dụng OR/Ms điển hình.
* Vì Raven xem toàn bộ Aggregate như một document duy nhất, vấn đề thường không tồn tại. Chúng ta có thể sử dụng hỗ trợ đồng thời trong Raven để xác định xem Aggregate hay con của nó có thay đổi gì không. Chúng ta có thể nạp lại dữ liệu đã được thay đổi của Aggregate và thử lại các transaction.

**Associations management**

* Aggregate Root có thể chứa tất cả con của nó, nhưng thậm chí Aggregates không thể tồn tại trong cô lập. Ví dụ:



* Aggregate Root cho Order sẽ chứa OrderLines, nhưng một OrderLine sẽ không chứa một Product. Thay vào đó, nó chứa tham chiếu đến Product Id.
* Raven Client API sẽ không cố gắng giải quyết những mối liên quan (associations). Đây là cố ý và do thiết kế. Thay vào đó, cách sử dụng mong muốn là giữ giá trị của khóa đối tượng được liên kết, và chỉ load các liên kết này khi thực sự cần.
* Lý do rất đơn giản: nhà thiết kế muốn một chút khó khăn khi tham chiếu dữ liệu trong những document khác. Trong OR/M chúng ta thường sử dụng: orderLine.Product.Name, nó sẽ load thực thể Product. Điều này làm chúng ta cảm giác như khi sử dụng mô hình quan hệ, nhưng thực sự thì Raven không quan hệ(non relation).