**Trường Đại học Sư Phạm TPHCM  
Khoa Công nghệ thông tin**

**Báo cáo đồ án cuối kì  
Môn: Cơ sở dữ liệu nâng cao**

****

**Nhóm 17– Đề tài : Apache CouchDB- Quản lí mượn sách**

**Nhóm 8:**

Nguyễn Như Quỳnh – 42.01.104.129

Nguyễn Sơn Lâm **–** 42.01.104.076

Nguyễn Thành Lâm **–** 42.01.104.236

Đỗ Thị Thanh Thi **–** 42.01.104.167

Trần Thị Yến Nhi **–** 42.01.104.099

Contents

[1. Lời mở đầu 3](#_Toc529951268)

[2. Giới thiệu NoSQL 3](#_Toc529951269)

[2.1 Định nghĩa NoSQL 3](#_Toc529951270)

[2.2 Phân loại cơ sở dữ liệu NoSQL 3](#_Toc529951271)

[2.3 Ưu điểm của NoSQL 4](#_Toc529951272)

[3. CouchDB 5](#_Toc529951273)

[3.1 Lịch sử phát triển 5](#_Toc529951274)

[3.2 Định nghĩa 5](#_Toc529951275)

[3.3 Đặc tính của CouchDB 6](#_Toc529951276)

[3.4 Sử dụng CouchDB trong trường hợp nào ? 7](#_Toc529951277)

[3.5 Ưu và nhược điểm của CouchDB 7](#_Toc529951278)

[3.6 Sự khác nhau của CouchDB và các cơ sở dữ liệu NoSQL khác 8](#_Toc529951279)

[3.7 Cách thức truy vấn và xử lý dữ liệu trên CSDL 9](#_Toc529951280)

[3.8 HTTP API 12](#_Toc529951281)

[3.9 Futon 15](#_Toc529951282)

[3.10 Sử dụng MapReduce để chạy câu truy vấn 26](#_Toc529951283)

# Lời mở đầu

Trong vài năm qua, chúng ta đã thấy sự gia tăng của một loại cơ sở dữ liệu mới, đó là cơ sở dữ liệu NoSQL. Sự ra đời của NoSQL đang thách thức sự thống trị của cơ sở dữ liệu quan hệ. Cơ sở dữ liệu quan hệ đã thống trị ngành công nghiệp phần mềm trong một thời gian dài. Cơ sở dữ liệu quan hệ đã cung cấp cơ chế để lưu trữ dữ liệu liên tục, đồng thời kiểm soát, giao dịch, giao diện được chuẩn hóa và được tích hợp vào các hệ thống dữ liệu ứng dụng, báo cáo. Tuy nhiên, ưu thế đó đã không còn tồn tại cho cơ sở dữ liệu quan hệ nữa. CouchDB – một loại cơ sở dữ liệu hướng tài liệu (Document oriented database) của NoSQL đã tạo ra một hướng mới trong việc lưu trữ dữ liệu. NoSQL có những ưu điểm nào? CouchDB được sử dụng như thế nào? Ta hãy cùng tìm hiểu.

# Giới thiệu NoSQL

## Định nghĩa NoSQL

NoSQL có nghĩa là “không chỉ SQL”, ngụ ý rằng khi thiết kế một giải pháp phần mềm hoặc sản phẩm, có nhiều hơn một cơ chế lưu trữ có thể được sử dụng dựa trên các nhu cầu. NoSQL không có một định nghĩa rõ ràng, chúng ta có thể hiểu NoSql là mô hình cơ sở dữ liệu mới có các đặc điểm như sau:

* Không sử dụng các mô hình quan hệ
* Chạy tốt trên các cụm(server)
* Chủ yếu là mã nguồn mở
* Xây dựng cho web thế kỷ 21
* Tối thiểu hóa lược đồ

## Phân loại cơ sở dữ liệu NoSQL

NoSQL được chia làm 4 loại:

**Key – value data stores:** Dữ liệu lưu dưới dạng cặp key – value. Giá trị được truy xuất thông qua key.

* Ví dụ : Redis, Dynomite, Project Voldemort.
* Thường được dùng cho: Content caching Applications
* Ưu điểm: Tìm kiếm rất nhanh
* Nhược điểm: Lưu dữ liệu không theo khuôn dạng (schema) nhất định

**Column-based – Tabular:** Cơ sở dữ liệu tổ chức dưới dạng các bảng. Gần giống với mô hình RDBMS. Tuy nhiên, Chúng lưu dữ liệu bởi các cột chứ không phải bằng các dòng. Nó khá thích hợp với để hiển thị bằng các phần mềm quản lý kho dữ liệu

* Ví dụ : Apache Hbase, Apache Cassandra, Hypertable
* Thường được dùng cho: các hệ phân tán file
* Ưu điểm: Tìm kiếm nhanh, Phân tán dữ liệu tốt
* Nhược điểm: Hỗ trợ được với rất ít phần mềm

D**ocument oriented database:**

Đây là dạng cơ sở dữ liệu được phát triển dựa trên ý tưởng key-value store nơi mà tài liệu (documents) chứa dữ liệu phức tạp hơn. Ở đây, mỗi tài liệu được gán cho một khóa riêng, khóa này được dùng để truy xuất tài liệu. Chúng được thiết kế cho việc lưu trữ, truy xuất và quản lý thông tin hướng tài liệu (document-oriented information), cũng được gọi là semi-structured data (dữ liệu nửa cấu trúc).

* Ví dụ : Apache CouchDB và MongoDB
* Thường được dùng cho: Web applications
* Ưu điểm: Dùng khi dữ liệu nguồn không được mô tả đầy đủ
* Nhược điểm: Hiệu năng truy vấn, Không có cú pháp chuẩn cho câu truy vấn dữ liệu

**Graph-based data-stores:** Những CSDL này áp dụng lý thuyết đồ thị trong khoa học máy tính để lưu trữ và truy xuất dữ liệu. Chúng tập trung vào tính rời rạc giữa các phần dữ liệu. Các phần tử đơn vị dữ liệu được biểu thị như một nút và liên kết với các thành phần khác bằng các cạnh.

* Ví dụ : Neo4j, InfiniteGraph, DEX
* Thường được dùng cho: Social networking, Hệ trợ giúp
* Ưu điểm: Ứng dụng các thuật toán trên đồ thị như Đường đi ngắn nhất, liên thông,…
* Nhược điểm: Phải duyệt nội bộ đồ thị, để trả lời lại các truy vấn. Không dễ để phân tán

## Ưu điểm của NoSQL

**- NoSQL là nguồn mở:**

Các sản phẩm nguồn mở đưa ra cho những người phát triển một vài lợi ích lớn lao, trong đó có lợi ích không có chi phí của chúng. Những lợi ích khác: phần mềm nguồn mở có xu hướng dễ tin cậy hơn, an ninh hơn và nhanh hơn để triển khai so với các lựa chọn thay thế sở hữu độc quyền.

Các hệ quản trị cơ sở dữ liệu (CSDL) NoSQL là Cassandra, CouchDB, Hbase, MongoDB và Redis.

**- Việc mở rộng phạm vi là mềm dẻo**

NoSQL thay thế câu thần chú cũ của các nhà quản trị CSDL về 'mở rộng phạm vi' với một thứ mới: 'mở rộng ra ngoài'. Thay vì bổ sung thêm các máy chủ lớn hơn để điều khiển nhiều tải dữ liệu hơn, thì CSDL NoSQL cho phép một công ty phân tán tải qua nhiều máy chủ khi mà tải gia tăng.

**- Các CSDL NoSQL khác nhau cho những dự án khác nhau**

MongoDB và Redis là những lựa chọn tốt cho việc lưu trữ các dữ liệu thống kê ít được đọc mà lại được viết thường xuyên, như một số đếm truy cập web chẳng hạn.  
Hadoop, một CSDL dạng tự do, phân tán làm tốt công việc lưu trữ các dữ liệu lớn như các con số thống kê thời tiết hoặc công việc phân tích nghiệp vụ.

Memcache, một CSDL nhất thời chóng tàn, tuyệt vời trong lưu trữ các phiên làm việc web, các khóa, và các con số thống kê ngắn hạn.

Cassandra và Riak (các lưu trữ dư thừa, tự động tạo bó cluster) làm tốt trong các môi trường với các ứng dụng có tính sẵn sàng cao, khi thời gian sống tối đa là sống còn.   
**- NoSQL được các hãng lớn sử dụng**

Các công ty như Amazon, BBC, Facebook và Google dựa vào các CSDL NoSQL.   
**- NoSQL phù hợp với công nghệ đám mây**

NoSQL và đám mây là một sự trùng khớp tự nhiên. Các máy chủ ngày nay là không đắt và có thể dễ dàng mở rộng phạm vi được theo yêu cầu có sử dụng một dịch vụ như là Amazon EC2. Giống như tất cả công nghệ đám mây, EC2 dựa vào ảo hóa. Liên kết yếu của ảo hóa là sự thực thi của I/O, với bộ nhớ và CPU các các kết nối mạnh.  
Các CSDL NoSQL hầu hết sử dụng bộ nhớ qua đĩa như là vị trí ghi đầu tiên - vì thế ngăn ngừa được sự thực thi không ổn định của I/O. Và vì NoSQL lưu trữ dữ liệu thường thúc đẩy được tính mở rộng phạm vi theo chiều ngang thông qua việc ngăn chia, chúng có khả năng tận dụng được việc cung cấp mềm dẻo của đám mây.

# CouchDB

## Lịch sử phát triển

Dự án CouchDB được tạo ra vào tháng tư năm 2005 bởi Damien Katz.Vào tháng 2 năm 2008, CouchDB trở thành một dự án Apache. Phiên bản chính thức đầu tiên của CouchDB được phát hành vào tháng 7 năm 2010. Phiên bản 1.2 được phát hành vào tháng tư năm 2012 và phiên bản 1.3 được phát hành vào tháng tư năm 2013.

Phiên bản kế tiếp của CouchDB là 2.1.1 – được phát hành vào tháng 11 năm 2017.

Phiên bản mới nhất của CouchDB là 2.2 .

## Định nghĩa

CouchDB là phần mềm cơ sở dữ liệu mã nguồn mở được phát triển bởi cơ sở phần mềm Apache, với mục tiêu là tạo ra một cơ sở dữ liệu bền vững, chịu lỗi cao, dễ dàng trong việc sử dụng.CouchDB có kiến trúc cơ sở dữ liệu NoSQL hướng tài liệu (document-oriented NoSQL database) và được thực hiện bằng ngôn ngữ Erlang, nó dùng JSON để lưu trữ dữ liệu, dùng JavaScript như là ngôn ngữ truy vấn, dùng MapReduce và HTTP cho API để truy cập tài liệu.

CouchDB là một cơ sở dữ liệu hoàn toàn bao trùm web. Lưu trữ dữ liệu của bạn bằng tài liệu JSON. Truy cập tài liệu bằng lập trình web, qua HTTP. Truy vấn, kết hợp, chuyển đổi tài liệu bằng javascript. CouchDB hoạt động tốt với các ứng dụng web và di động hiện đại. Bạn có thể phân phối dữ liệu hiệu quả bằng cách sử dụng bản sao gia tăng của CouchDB. CouchDB hỗ trợ thiết lập master-master với phát điện xưng đột tự động.

CouchDB được phát hành lần đầu và năm 2005 và sau đó trở thành dự án phần mềm Apache vào năm 2008.

Không giống như cơ sở dữ liệu quan hệ, một cơ sở dữ liệu CouchDB không lưu trữ dữ liệu và quan hệ vào các bảng. Thay vào đó, mỗi cơ sở dữ liệu là một tập hợp các văn bản độc lập. Mỗi văn bản chứa dữ liệu riêng của nó và tự nó bao gồm mô hình của nó (các trường, loại của mỗi trường).

Mỗi một ứng dụng có thể thực thi rất nhiều cơ sở dữ liệu, ví dụ như chúng ta   
dùng một cơ sở dữ liệu để lưu thông tin người dùng điện thoại và cái còn lại là lưu trên   
server.Trên mỗi văn bản(bản ghi) còn bao gồm các thông tin về phiên bản, làm cho việc   
đồng bộ các dữ liệu với nhau khi cơ sởdữ liệu bị mất kết nối một thời gian giữa   
các thiết bị dễ dàng hơn.

CouchDB sử dụng MVCC (multi-Version Concurency Control ) để tránh việc deadlock database trong suốt thời gian ghi.Tức là, trong khi ghi dữ liệu, chúng ta vẫn có thể đọc dữ liệu vìCouchDB sinh ra một bản copy và chúng ta đọc trên bản copy đó.Sau khi ghi xong nó sẽ tiến hành nhập dữ liệu giữa các thiết bị và xóa bản ghi cũ đi.

## Đặc tính của CouchDB

**Document Storage**

CouchDB là một cơ sở dữ liệu NoSQL lưu trữ tài liệu. Những tài liệu là đơn vị chính của dữ liệu, mỗi trường được đặt tên riêng biệt và chứa giá trị của nhiều loại dữ liệu như text, number, boolean, lists, … Trong những tài liệu này, không có giới hạn cho kích cỡ text hoặc bộ đếm phần tử.

CouchDB cung cấp một API được gọi là RESTful HTTP API, được dùng cho việc đọc và cập nhật (add, edit, delete) những tài liệu cơ sở dữ liệu.

Dưới đây làmẫu cấu trúc tài liệu JSON trong CouchDB.

{

"field" : "value",

"field" : "value",

"field" : "value",

}

**ACID Properties (Thuộc tính ACID)**

Việc bố trí tập tin và hệ thống cam kết CouchDB mô tả những nét nổi bật của tất cả thuộc tính ACID. Một khi dữ liệu được thêm vào trong đĩa nó sẽ không bị ghi đè lên. Những việc cập nhật tài liệu (add, edit, delete) theo Atomicity, … Tức là, chúng sẽ được lưu hoàn toàn hoặc không được lưu gì cả. Cơ sở dữ liệu (CSDL) sẽ không có những tài liệu được lưu cục bộ hoặc được chỉnh sửa.

Hầu hết những cập nhật này được xếp theo thứ tự và bất kì số lượng khách hàng nào cũng có thể đọc một tài liệu mà không cần chờ đợi và không bị gián đoạn.

**Compaction (Sự nén dữ liệu)**

Compaction là một toán tử dùng để giúp thêm không gian đĩa cho CSDL bằng cách xóa dữ liệu không được sử dụng. Trong khi thực hiện toán tử nén trên một file nào đó, một file với phần mở rộng .compaction được tạo ra và tất cả dữ liệu hoạt động/ thực được sao chép vào file đó, sau khi quá trình sao chép kết thúc, file cũ sẽ bị xóa. CSDL giữ trạng thái trực tuyến trong suốt quá trình nén và tất cả cập nhật, những bài đọc được cho phép hoàn thành thành công.

**Views**

Dữ liệu trong CouchDB được lưu trữ trong những tài liệu nửa cấu trúc mà linh hoạt với những cấu trúc ngầm riêng, nhưng nó là một mẫu tài liệu đơn giản cho việc lưu trữ và chia sẻ dữ liệu. Nếu chúng ta muốn xem dữ liệu của chúng ta bằng những cách khác nhau, chúng ta cần một cách để lọc, tổ chức và báo cáo dữ liệu nào chưa được phân tích vào các bảng.

Để giải quyết vấn đề này, CouchDB cung cấp một view model. Views là phương pháp tập hợp và báo cáo dữ liệu trong một CSDL, và được xây dựng theo yêu cầu để tập hợp, liên kết và báo cáo những tài liệu CSDL. Vì Views được xây dựng động và không ảnh hưởng đến các tài liệu cơ bản, bạn có thể có bao nhiêu view trình bày dữ liệu giống nhau cũng được.

**HTTP API**

Tất cả thành phần có URL riêng và được trình bày thông qua HTTP. Nó sử dựng phương thức HTTP POST, GET, PUT và DELETE cho 4 toán tử cơ bản CRUD (Create, Read, Update, Delete) trên tất cả tài nguyên.

✪ CouchDB cũng cung cấp một giao diện người dùng (interface) quản trị xây dựng được truy cập qua Web gọi là Futon.

## Sử dụng CouchDB trong trường hợp nào ?

CouchDB sẽ là lựa chọn tốt hơn về cơ sở dữ liệu so với RDBMS ( MySQL,..), ví dụ khả năng mở rộng, thiết kế, thời gian phát triển, độ tin cậy và bảo trì. Dư liệu được lưu trữ dưới dạng tài liệu, chứ không phải các bảng được chuẩn hóa, giao diện REST/ HTTP trực quan.

## Ưu và nhược điểm của CouchDB

**Ưu điểm :**

- CouchDB có một REST API dựa trên HTTP, cái này giúp giao tiếp với cơ sở dữ liệu một cách dễ dàng. Cấu trúc đơn giản của tài nguyên HTTP và các phương thức (GET, PUT, DELETE) thì dễ hiểu và dễ sử dụng.

- Vì chúng ta lưu trữ dữ liệu bằng cấu trúc document-based linh hoạt nên không cần lo lắng về cấu trúc của dữ liệu.

- Người dùng được cung cấp bản vẽ dữ liệu (data mapping) mạnh mẽ - cho phép truy vấn, tổng hợp và lọc thông tin.

- CouchDB cung cấp bản ghi dễ sử dụng, dùng bản ghi này, bạn có thể sao chép, chia sẻ và đồng bộ hóa dữ liệu giữa cơ sở dữ liệu và máy móc.

- CouchDB cho phép bạn truy cập ở dữ liệu của mình ở những nơi bạn cần.

- Lưu trữ dữ liệu của bạn một cách an toàn, trên máy chủ của bạn hoặc bất kỳ nhà cung cấp đám mây hàng đầu nào.

- Giao thức sao chép Couch cho phép luồng dữ liệu của bạn liên tục giữa các cụm máy chủ với điện thoại di động và trình duyệt web, cho phép trải nghiệm người dùng ngoại tuyến hấp dẫn đầu tiên trong khi vẫn duy trì hiệu suất và độ tin cậy cao.

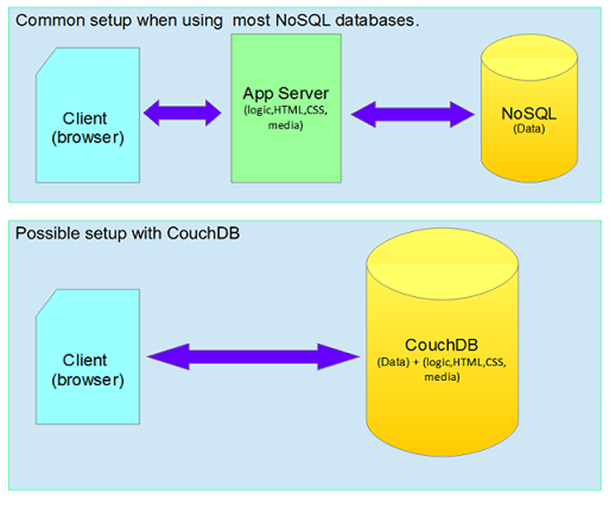
- Có thể thay đổi lược đồ dữ liệu của mình bất cứ lúc nào.

- Dữ liệu sẽ được lập chỉ mục chính xác cho các câu truy vấn, vì vậy sẽ nhận được kết quả theo thời gian không đổi.

**Nhược điểm :**

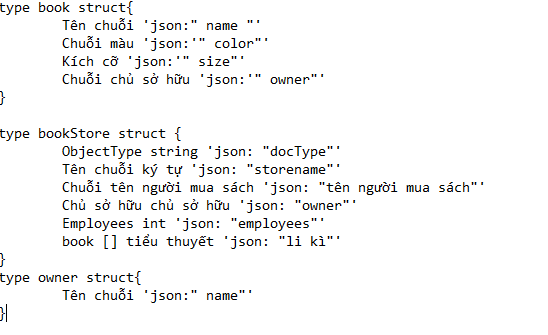
* Phải tạo các khung nhìn cho mỗi câu truy vấn, ví dụ như các câu truy vấn đặc biệt như truy vấn động của where và sort trong một sql không có sẵn.
* Sẽ có dữ liệu thừa, hoặc sẽ kết thúc việc tự thực hiện và sắp xếp logic trên “phía máy khách” (ví dụ: phân loại mối quan hệ trên nhiều trường).

## Sự khác nhau của CouchDB và các cơ sở dữ liệu NoSQL khác

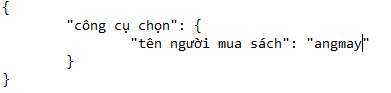


Như bạn biết, nhiều CSDL NoSQL lưu trữ dữ liệu dưới dạng document (tài liệu), CouchDB cũng vậy. Điều khác biệt của CouchDB là nó làm việc như là một webserver cho ứng dụng của bạn. Một máy chủ đơn đóng vai trò như một CSDL và máy chủ ứng dụng. Sử dụng một framework và sự linh hoạt của thiết kế ít sơ đồ của CouchDB, một phạm trù rộng của các ứng dụng web có thể được tạo nhanh chóng và dễ dàng.

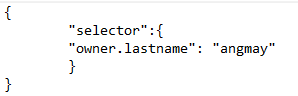
## Cách thức truy vấn và xử lý dữ liệu trên CSDL



* Tìm kiếm cửa hàng sách tiểu thuyết thuộc sở hữu của angmay

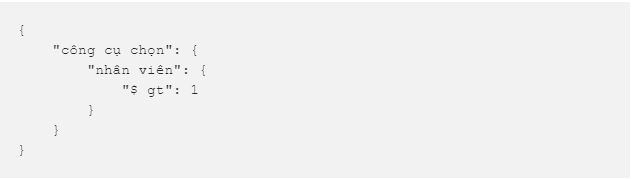


* Truy vấn các subdocuments hoạt động tương tự



* Tìm kiếm các sách li kì có số lượng lớn hơn 1

Cũng tương tự MongoDB, ta cũng có thể sử dụng $gt, $le, $eq,…

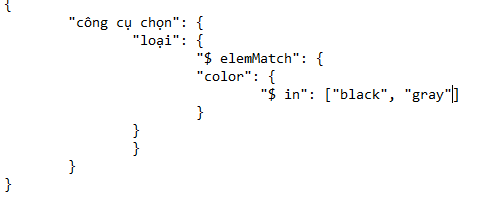


**Truy vấn nâng cao**

* Tìm kiếm một màu cụ thể của sách tiểu thuyết trong cửa hàng

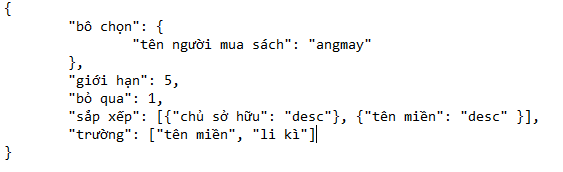


* Tìm kiếm nhiều loại màu của sách tiểu thuyết trong cửa hàng



**Tiện ích nâng cao**

Bên cạnh thuộc tính Selector, CouchDB còn có một vài thuộc tính gọn gàng khác để hỗ trợ truy vấn cho bạn.



**Truy vấn kết quả xem**

Khi truy vấn kết quả xem, bạn có thể sử dụng các đối số truy vấn được hiển thị trong bảng bên dưới để giới hạn và thao tác truy vấn của bạn. Một số tham số truy vấn chính là:

**Giảm dần**

Theo mặc định, CouchDB sắp xếp các khóa và xuất kết quả theo thứ tự tăng dần (tức là, A-Z). Bằng cách chỉ định giảm dần = true, kết quả xem sẽ được xuất theo thứ tự giảm dần (ví dụ: Z-A). Một tạo phẩm của quá trình này là nếu bạn đang tìm kiếm một loạt các mục thì bạn cũng phải đảo ngược các giá trị của khóa khởi động và khóa kết thúc, vì thứ tự CouchDB sẽ tiếp cận đã thay đổi. Ví dụ, câu truy vấn:

/recipes/\_design/simple/\_view/bytitle?startkey=Lasagne&endkey=Pasta

Nên viết như sau:

/recipes/\_design/simple/\_view/bytitle? descending=true&endkey=Lasagne&startkey=Pasta

**Giới hạn**

Giới hạn đầu ra số hàng được chỉ định. Ví dụ: để giới hạn đầu ra thành 10 hàng:

/recipes/\_design/simple/\_view/bytitle?limit=10

**Bỏ qua**

Bỏ qua số hàng được chỉ định trước khi bắt đầu đầu ra. Ví dụ:

/recipes/\_design/simple/\_view/bytitle?skip=100

CouchDB sẽ xử lý 100 hàng đầu ra khung nhìn, và sau đó chỉ bắt đầu xuất các giá trị hàng trên hàng 101.

Bạn có thể sử dụng điều này kết hợp với tham số giới hạn để phân trang thông qua đầu ra, nhưng đây là một hoạt động tương đối tốn kém, vì CouchDB vẫn thực sự truy cập thông tin hàng.

**Thông tin cũ**

Cho phép xem cũ. Bởi vì CouchDB chỉ xây dựng lại chỉ mục xem; khi xem được truy cập, nó có thể có nghĩa là có sự chậm trễ giữa yêu cầu thông tin xem và đầu ra được tạo ra, vì chỉ mục khung nhìn phải được cập nhật. Nếu bạn không lo lắng về việc cập nhật chỉ mục xem (hoặc bao gồm bất kỳ bản ghi nào) kể từ lần cuối cùng chế độ xem được cập nhật, bạn có thể sử dụng stale = true để sử dụng chỉ mục chế độ xem hiện tại và không cập nhật dữ liệu chế độ xem. Điều này nhanh hơn, nhưng có thể xuất ra thông tin cũ vì nó sẽ không bao gồm bất kỳ cập nhật, xóa và chèn tài liệu gần đây nào kể từ lần xem được cập nhật lần cuối.

**Danh sách**

Danh sách là để xem những gì hiển thị là tài liệu cá nhân. Một cách rất đơn giản, chúng thực hiện chính xác cùng một chức năng. Chúng xuất kết quả từ mỗi hàng được tạo ra bởi một khung nhìn và chuyển nó thành bất kỳ định dạng nào bạn muốn. Như với một chương trình, bạn có thể sử dụng nó để xuất ra dưới dạng HTML, XML, định dạng JSON, thậm chí là CSV nếu đó là những gì bạn cần. Không giống như một chương trình, danh sách phải xử lý thông tin phức tạp hơn một chút dưới dạng một hoặc nhiều hàng thông tin được tạo, thay vì chỉ một tài liệu. Những điều cơ bản về chức năng danh sách tương tự như chức năng hiển thị và xem mà chúng ta đã thấy. Bạn có thể chỉ định nhiều hơn một hàm danh sách trong một tài liệu thiết kế. Tuy nhiên, bạn phải chỉ định danh sách và chế độ xem mà bạn đang sử dụng để xuất thông tin trong cùng một tài liệu thiết kế. Định nghĩa hàm cho một danh sách như sau:

function(head, req) {}

Đối số đầu chứa thông tin cốt lõi về chế độ xem đang được cung cấp cho hàm danh sách, cụ thể là thông tin ngoài dữ liệu hàng (nghĩa là tổng số hàng và số lượng bỏ qua hoặc bù trừ, hàng):

{

total\_rows:2667,

offset:0

}

## HTTP API

Sử dụng tiêu đề yêu cầu HTTP, bạn có thể giao tiếp với CouchDB. Thông qua những yêu cầu này, chúng ta có thể lấy dữ liệu từ cơ sở dữ liệu, lưu trữ dữ liệu trong cơ sở dữ liệu dưới dạng văn bản, và chúng ta có thể xem cũng như định dạng tài liệu được lưu trữ trong một cơ sở dữ liệu.

1. **Định đạng HTTP Request**

Trong khi giao tiếp với cơ sở dữ liệu, chúng ta sẽ sử dụng các yêu cầu định dạng khác nhau như get, head, post, put, delete, và copy. Đối với tất cả các toán tử trong CouchDB, các dữ liệu đầu vào và các cấu trúc dữ liệu đầu ra sẽ được định dạng thành đối tượng JavaScript Object Notation (JSON).

Sau đây là các yêu cầu định dạng khác nhau của giao thức HTTP được sử dụng để giao tiếp với CouchDB.

* GET - Định dạng này được sử dụng để lấy được một sản phẩm cụ thể. Để có được các mục khác nhau, bạn phải gửi mẫu url cụ thể. Trong CouchDB, sử dụng yêu cầu GET này, chúng ta có thể có được các phần tử tĩnh, tài liệu cơ sở dữ liệu, cấu hình và thông tin thống kê dưới dạng tài liệu JSON (trong hầu hết các trường hợp).
* HEAD - Các phương thức HEAD được sử dụng để có được các HTTP header của một yêu cầu GET mà không có nội dung hồi đáp.
* POST – yêu cầu Post được sử dụng để tải dữ liệu lên. Trong CouchDB sử dụng yêu cầu POST, bạn có thể thiết lập các giá trị, tài liệu tải lên, cài đặt giá trị tài liệu, và cũng có thể bắt đầu các lệnh quản lý nhất định.
* PUT - Sử dụng yêu cầu PUT, bạn có thể tạo mới các đối tượng, cơ sở dữ liệu, tài liệu, views và các tài liệu thiết kế.
* DELETE - Sử dụng yêu cầu DELETE, bạn có thể xóa documetn, views và các tài liệu thiết kế.
* COPY - Sử dụng phương pháp COPY, bạn có thể sao chép văn bản và các đối tượng.

1. **HTTP Request Headers**

Tiêu đề HTTP cần được cung cấp để có thể lấy được định dạng phù hợp và mã hóa. Trong khi gửi yêu cầu đến máy chủ CouchDB, bạn có thể gửi các tiêu đề yêu cầu Http cùng với yêu cầu. Sau đây là các tiêu đề yêu cầu Http khác nhau.

Content-type - Header này được sử dụng để xác định các kiểu nội dung của dữ liệu mà chúng ta cung cấp cho các máy chủ cùng với yêu cầu. Chủ yếu là các loại nội dung chúng ta gửi cùng với yêu cầu sẽ được định dạng MIME hoặc JSON (application / json). Sử dụng kiểu nội dung trên một yêu cầu là rất được khuyến khích.

* Accept - tiêu đề này được sử dụng để xác định máy chủ, danh sách các kiểu dữ liệu mà khách hàng có thể hiểu được, vì vậy mà các máy chủ sẽ gửi phản hồi của mình bằng những kiểu dữ liệu đó. Nói chung ở đây, bạn có thể gửi danh sách các kiểu dữ liệu MIME mà khách hàng chấp nhận, cách nhau bằng dấu hai chấm.
* Mặc dù, sử dụng Accept trong các truy vấn của CouchDB là không cần thiết, nhưng được khuyến khích để đảm bảo rằng dữ liệu trả về có thể được xử lý bởi các khách hàng.

1. **Tiêu đề khản hồi ( Reponses Headers)**

Đây là những tiêu đề của các phản hồi gửi bởi máy chủ. Những tiêu đề này cung cấp thông tin về nội dung gửi bởi máy chủ phản hồi.

**Content-type** - Tiêu đề này quy định các kiểu dữ liệu MIME được trả về bởi máy chủ. Đối với hầu hết các yêu cầu, các loại MIME được trả lại là text / plain.

**Cache-control** - Tiêu đề này gợi ý cho khách hàng cách sử dụng thông tin được gửi bởi máy chủ. CouchDB chủ yếu trả về những giá trị phải xác nhận lại, điều này chỉ ra rằng các thông tin cần được xác nhận lại nếu có thể.

**Content-length** - Tiêu đề này trả về độ dài của các nội dung được gửi bởi máy chủ, bằng đơn vị bytes.

**Etag** - Tiêu đề này được sử dụng để hiển thị các sửa đổi cho văn bản, hay một view.

Status Codes (Các mã trạng thái)

Dưới đây là bảng mã trạng thái được gửi bởi http header và mô tả của nó

|  |  |
| --- | --- |
| **200 – OK** | Status này sẽ hiện ra khi một yêu cầu được thực hiện thành công. |
| **201 – Created** | Status này sẽ hiện ra khi Status này sẽ hiện ra khi một tài liệu được tạo. |
| **202 – Accepted** | Status này sẽ hiện ra khi một yêu cầu được chấp nhận. |
| **404 − Not Found** | Status này sẽ hiện ra khi máy chủ không thể tìm được nội dung yêu cầu. |
| **405 − Resource Not Allowed** | Status này sẽ hiện ra khi loại yêu cầu HTTP được dùng không khả dụng. |
| **409 – Conflict** | Status này sẽ hiện ra bất cứ khi nào có sự xung đột trong việc cập nhật. |
| **415 − Bad Content Type** | Status này chỉ ra rằng loại nội dung được yêu cầu không được hỗ trợ bởi máy chủ. |
| **500 − Internal Server Error** | Status này sẽ hiện ra bất cứ khi nào dữ liệu được gửi trong yêu cầu không khả dụng. |

HTTP URL Paths

Có những đường dẫn url nhất định mà khi sử dụng chúng, chúng ta có thể tương tác với CSDL một cách trực tiếp. Sau đây là bảng của những đường dẫn url như vậy.

|  |  |
| --- | --- |
| **URL** | **Operation** |
| **PUT /db** | URL này được dùng để tạo một CSDL mới |
| **GET /db** | URL này được dùng để lấy thông tin về các CSDL hiện có. |
| **PUT /db/document** | URL này được dùng để tạo một document hoặc cập một document có sẵn. |
| **GET /db/document** | URL này được dùng để lấy tài liệu |
| **DELETE /db/document** | URL này được dùng để xóa tài liệu nhất định từ CSDL tương ứng. |
| **GET /db/\_design/design-doc** | URL này được dùng để lấy định nghĩa của một tài liệu thiết kế |
| **GET /db/\_design/designdoc/\_view/view-name** | URL này được dùng để truy cập view, view-name từ tài liệu thiết kế từ CSDL nhất định. |

## Futon

Futon là interface dựa trên web, là hệ quản trị được tích hợp sẵn trong CouchDB. Nó cung cấp một giao diện đồ họa đơn giản, dùng giao diện này, bạn có thể tương tác với CouchDB. Nó là một giao diện đơn giản và nó truy cập đầy đủ tới tất cả đặc tính của CouchDB. Sau đây là danh sách của những đặc tính này:

**CSDL**

Tạo CSDL

Xóa CSDL

**Tài liệu**

Tạo tài liệu

Cập nhật tài liệu

Chỉnh sửa tài liệu

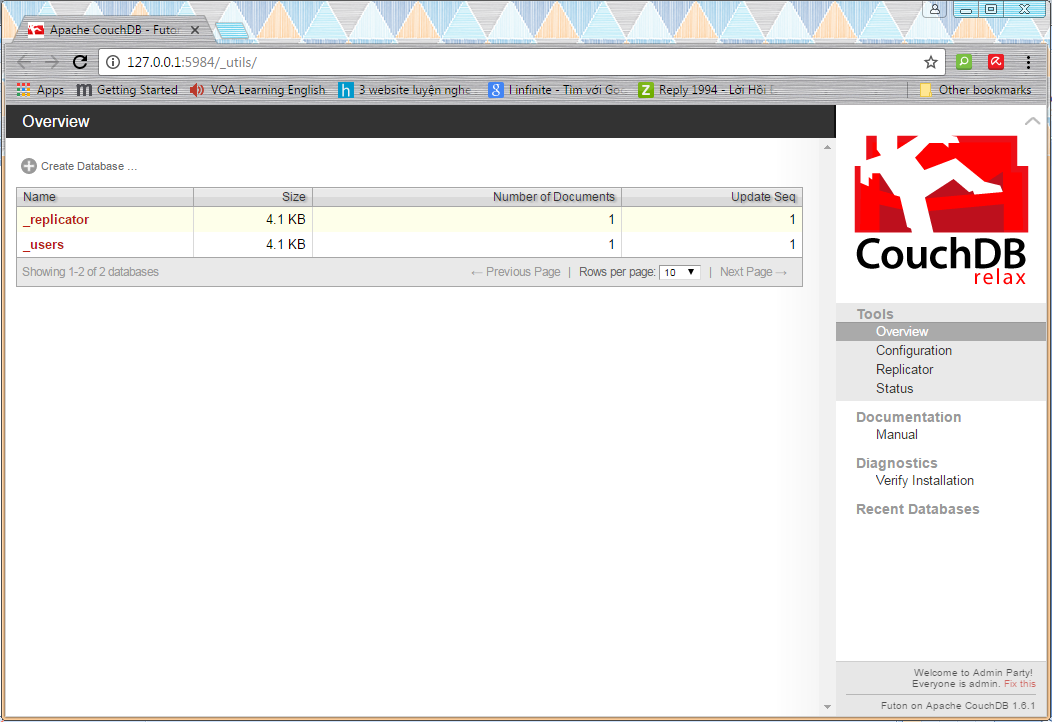
Xóa tài liệu

**a. Khởi động Futon**

Đầu tiên, bạn phải chắc rằng CouchDB đang chạy và sau đó mở url sau:

http://127.0.0.1:5984/\_utils/

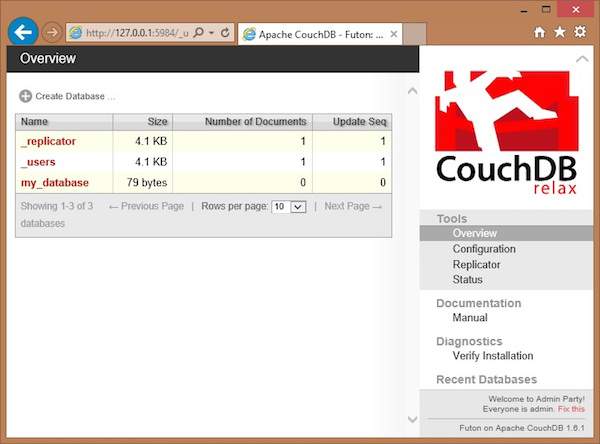
Nếu bạn mở url này, Trang chủ Futon sẽ hiện ra như sau:



* Ở phía bên tay phải của trang này, bạn có thể thấy danh sách của tất cả các CSDL củaCouchDB. Trong ví dụ này, chúng ta có một CSDL tên là **my\_database**, bên cạnh là cơ sở dữ liệu hệ thống **\_replicator** và **\_user**.
* Phía bên phải bạn có thể thấy những mục sau:
* **Tools** – Trong mục này bạn có thể tìm mục **Configuration** để cấu hình CouchDB, **Replicator** để lặp lại hoạt động, và **Status** để xác định trạng thái của CouchDB và những cấu hình được thực hiện gần đây trên CouchDB
* **Documentation** – Mục này chứa tài liệu hoàn chỉnh cho phiên bản gần đây của CouchDB
* **Diagnostic** – Dưới phần này bạn có thể xác định các cài đặt của CouchDB.
* **Recent Database** – Dưới phần này bạn có thể tìm tên của những CSDL được thêm vào gần đây.

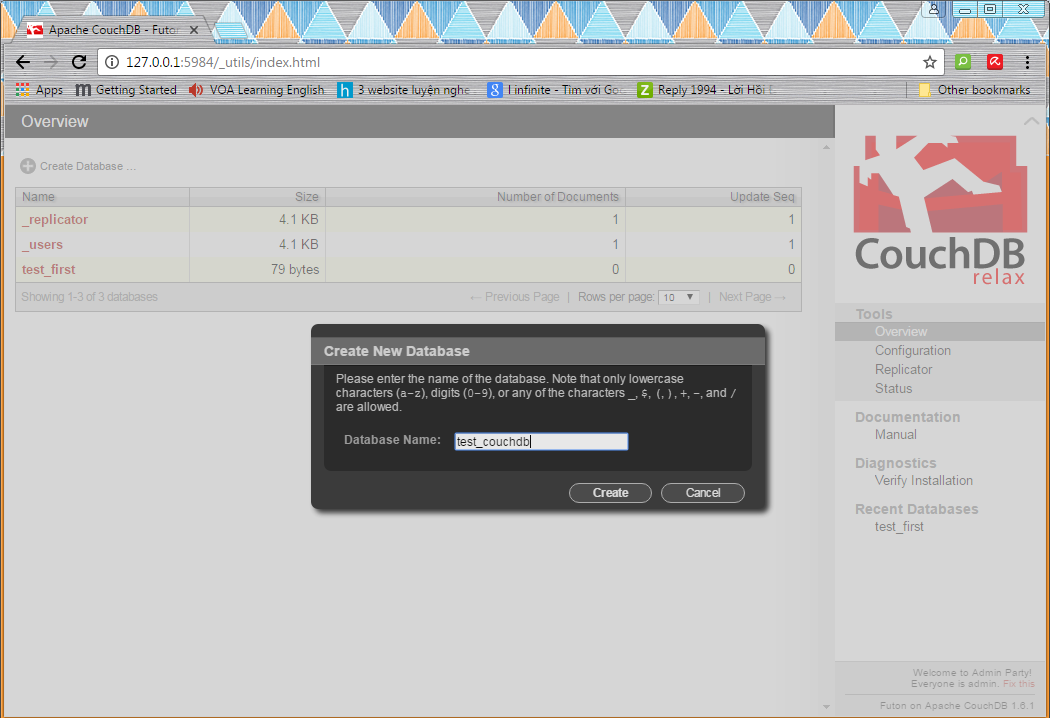
**b. Dùng Futon để tạo một CSDL**

Để tạo một CSDL, hãy mở <http://127.0.0.1:5984/_utils/>. Bạn sẽ thấy được trang Overview/index của CouchDB như hình dưới:



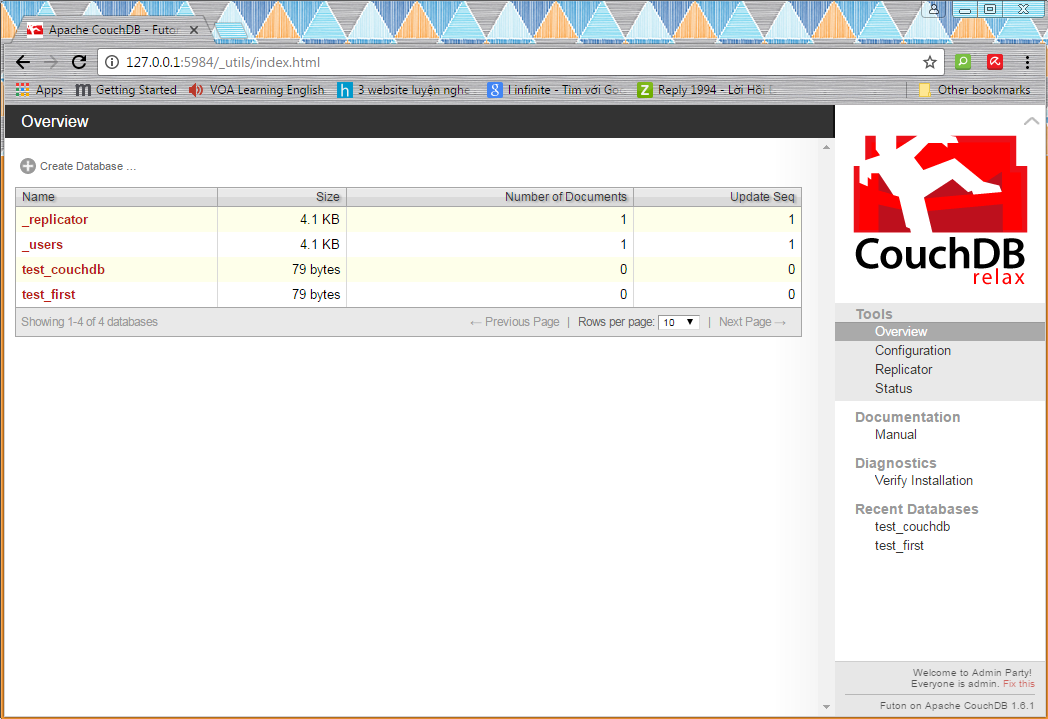
Trong trang này, bạn có thể thấy danh sách các CSDL trong CouchDB, một nút tùy chọn Create Database ở bên tay trái.

Bây giờ click vào đường link Create database. Bạn có thể thấy một popup window Create New Databases yêu cầu điền tên CSDL mới. Chọn bất kì tên nào theo các tiêu chí được đề cập. Ở đây chúng ta tạo một CSDL mới với tên test\_couchdb. Click nút create như hình dưới đây



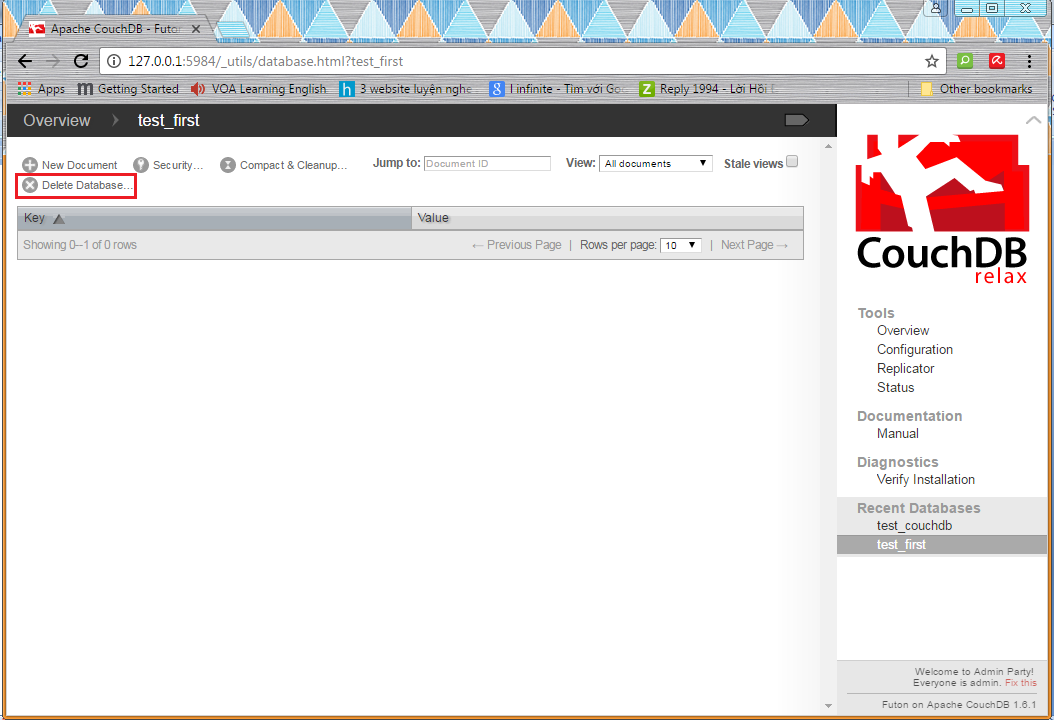
**c. Xóa một cơ sở dữ liệu với Futon**

Để xóa một CSDL, mở đường dẫn **http://127.0.0.1:5984/\_utils/**  nơi bạn sẽ thấy được trang Overview/index của CouchDB như hình dưới

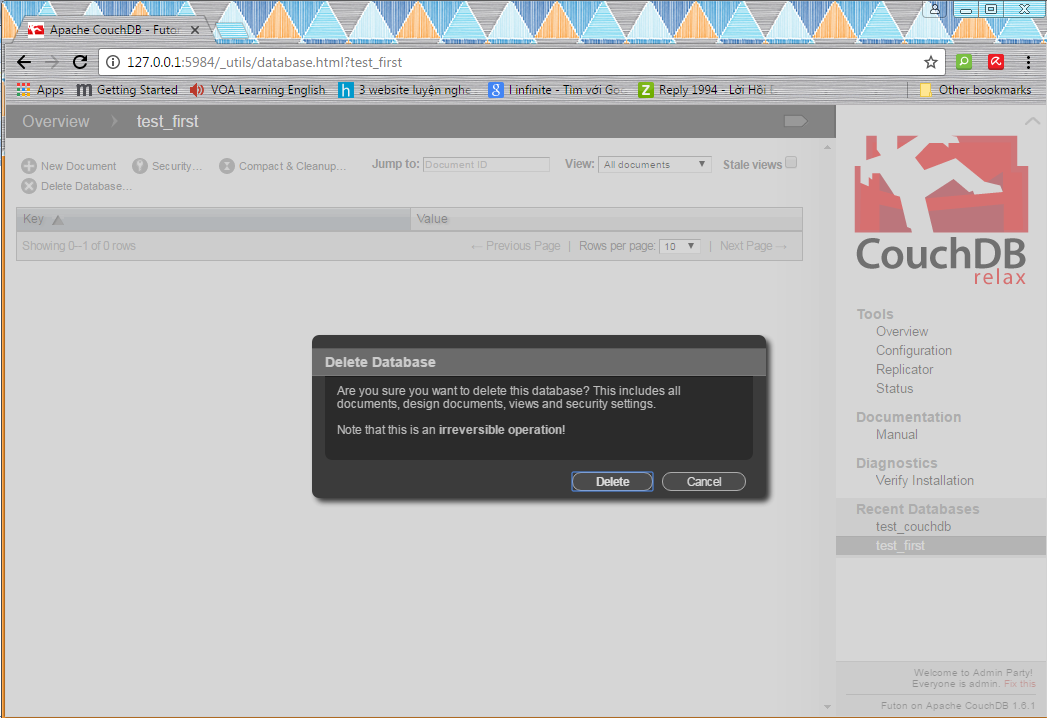


Ở đây bạn có thể có vài CSDL. Chúng ta sẽ xóa CSDL tên test\_first. Để xóa một CSDL, chọn một CSDL từ danh sách các CSDL, và click vào nó, nó sẽ dẫn tới trang tổng quan

của CSDL mới được chọn. Ở đó, bạn có thể thấy nhiều loại toán tử trên CSDL như hình dưới đây.

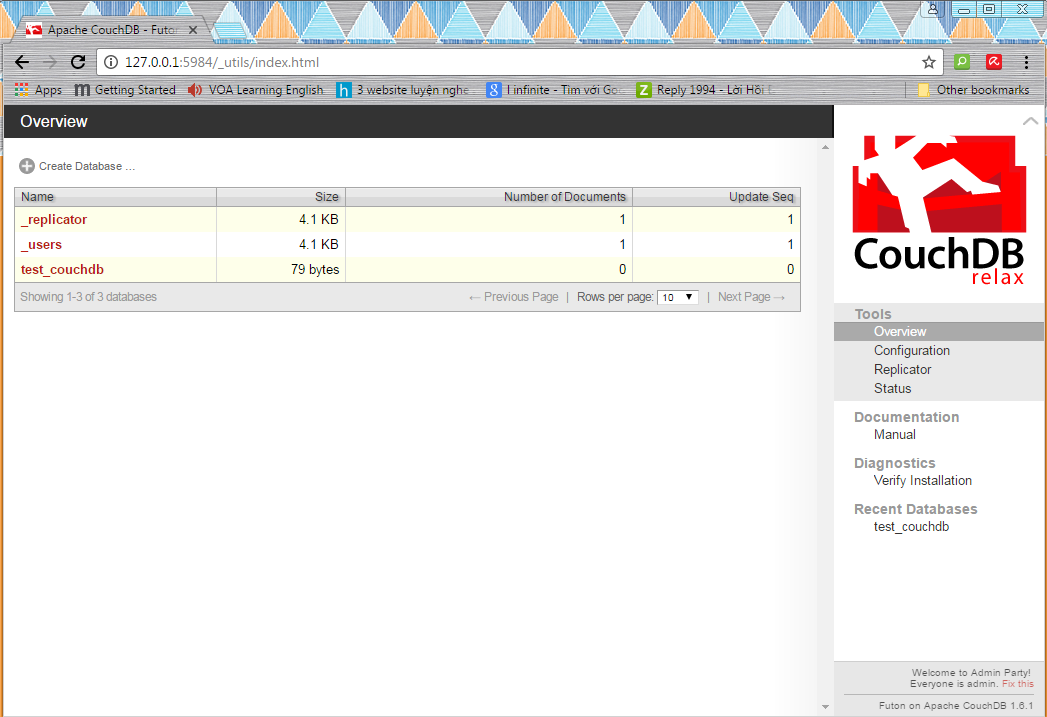


Trong số các loại toán tử bạn có thể tìm thấy tùy chọn Delete Database. Khi click vào nó, một trang popup window sẽ hiện ra, hỏi bạn đã chắc chắn hay chưa. Click delete để xóa CSDL bạn đã chọn.

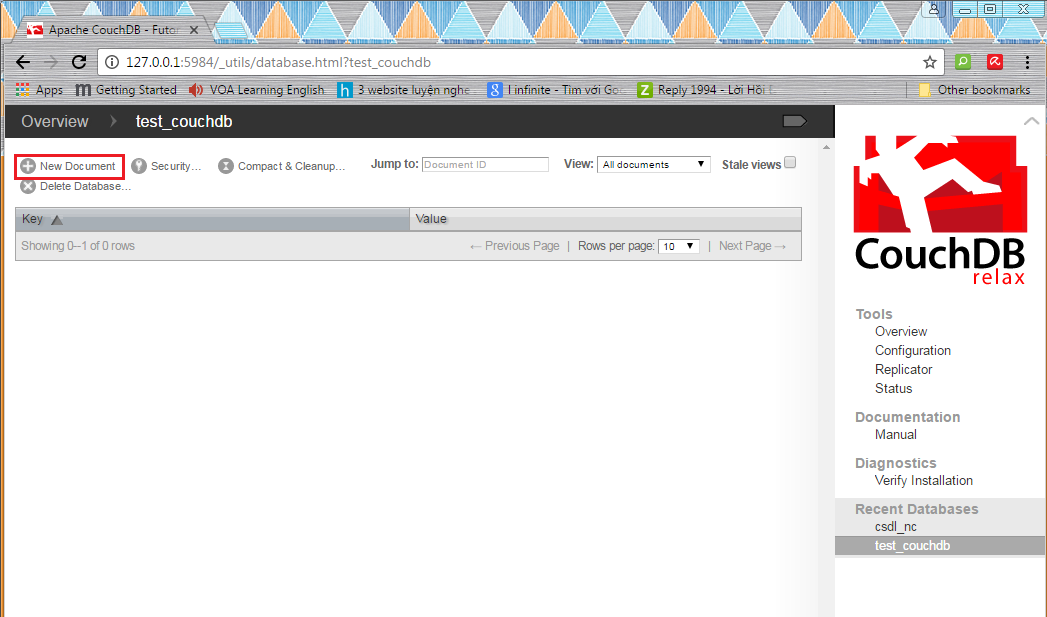


**d. Tạo một tài liệu (document) với Futon**

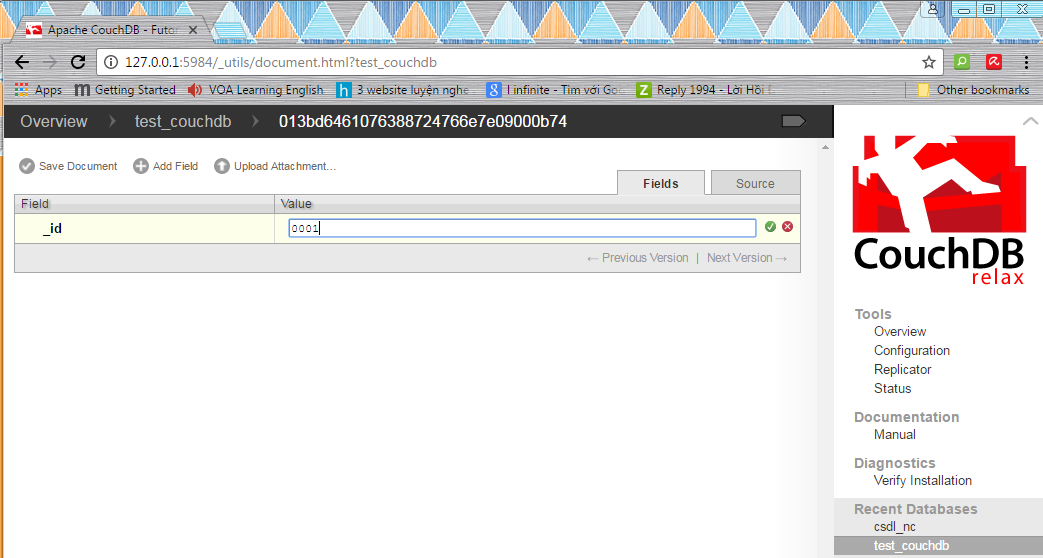
Để tạo một document, hãy mở đường dẫn **http://127.0.0.1:5984/\_utils/**  để mở trang Overview/index của CouchDB như hình dưới.



Chọn CSDL mà bạn muốn thêm document. Mở trang tổng quan của CSDL và chọn New Document như hình dưới.



Khi bạn chọn tùy chọn New Document, CouchDB tạo một document của CSDL mới, đặt cho nó một id mới. Bạn có thể sửa giá trị của id và đặt giá trị của riêng bạn dưới dạng chuỗi (string). Trong ví dụ sau, chúng ta tạo một document mới với một id là 001.

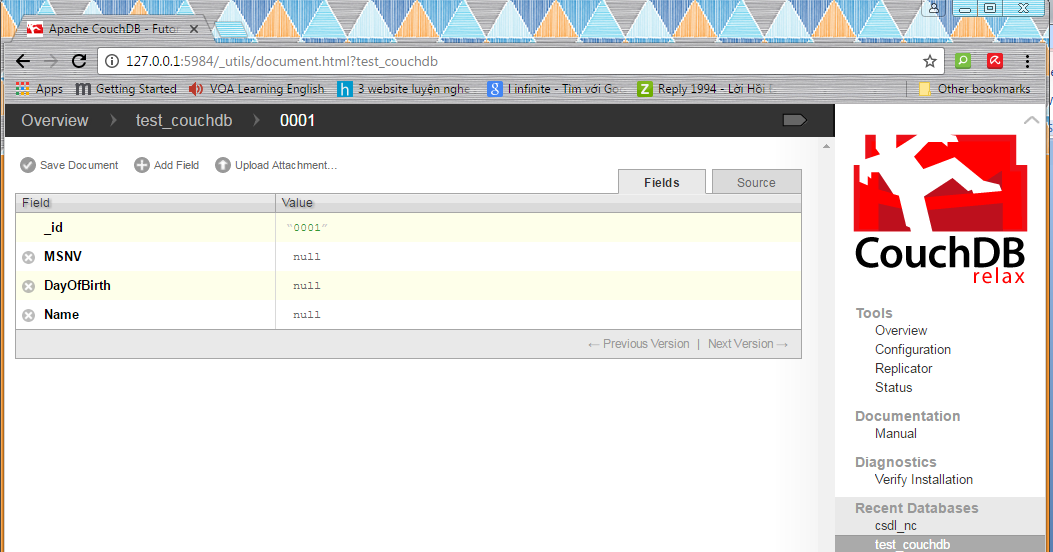


Trong trang này, bạn có thể thấy ba lựa chọn – save Document, Add Field và Upload Attachment.

* + 1. **Thêm trường vào document**

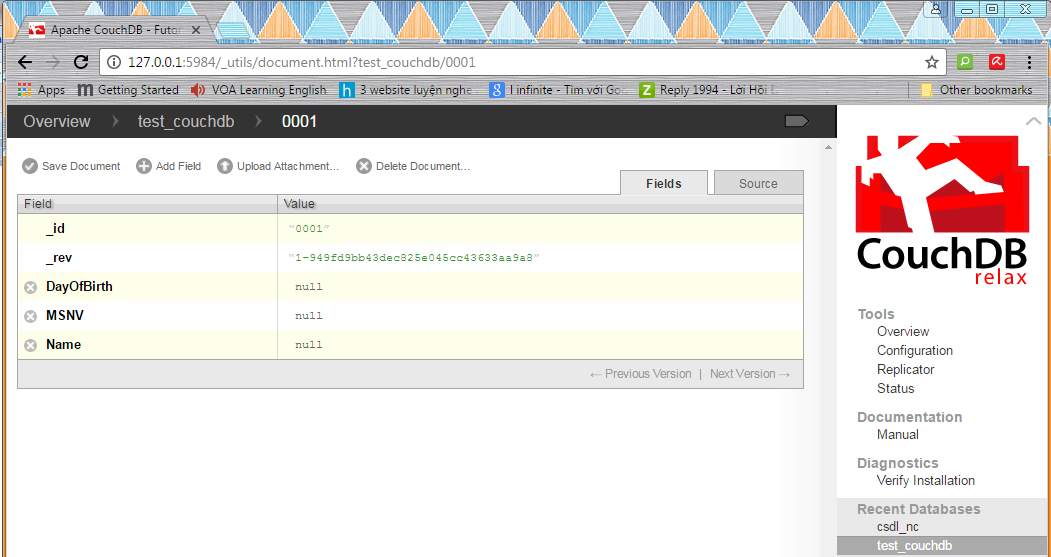
Để thêm trường (field) vào document, click lựa chọn Add Field. Sau khi tạo một CSDL, bạn có thể thêm 1 field vào nó bằng lựa chọn này. Click vào nó, bạn sẽ thấy một cặp text box, tên là Field, value. Bạn có thể chỉnh sửa những giá trị này bằng cách click đúp vào chúng. Chỉnh sửa những giá trị này và nhập cặp Field-Value bạn muốn. Click nút màu xanh lá để lưu những giá trị này.

Trong ví dụ này, chúng tôi đã tạo ba trường MSNV, DayOfBirth và Name của nhân viên



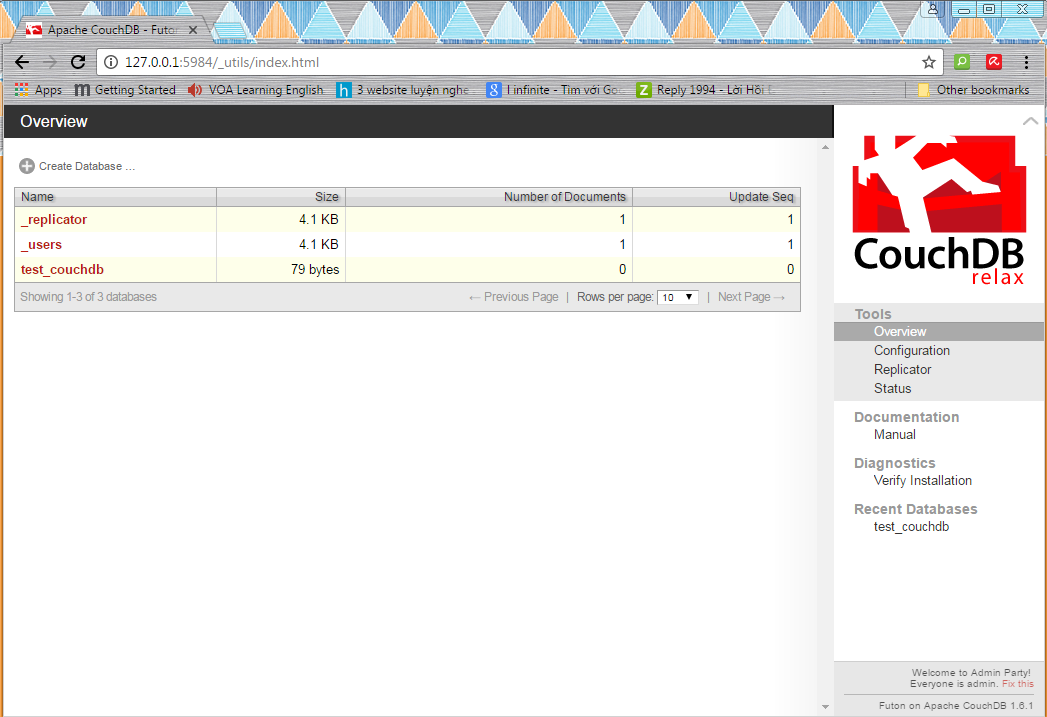
* + 1. **Lưu Document**

**Bạn có thể lưu những thay đổi vào document bằng cách click vào lựa chọn này. Sau khi lưu, một id mới id\_rev sẽ được tự động tạo ra như hình dưới.**

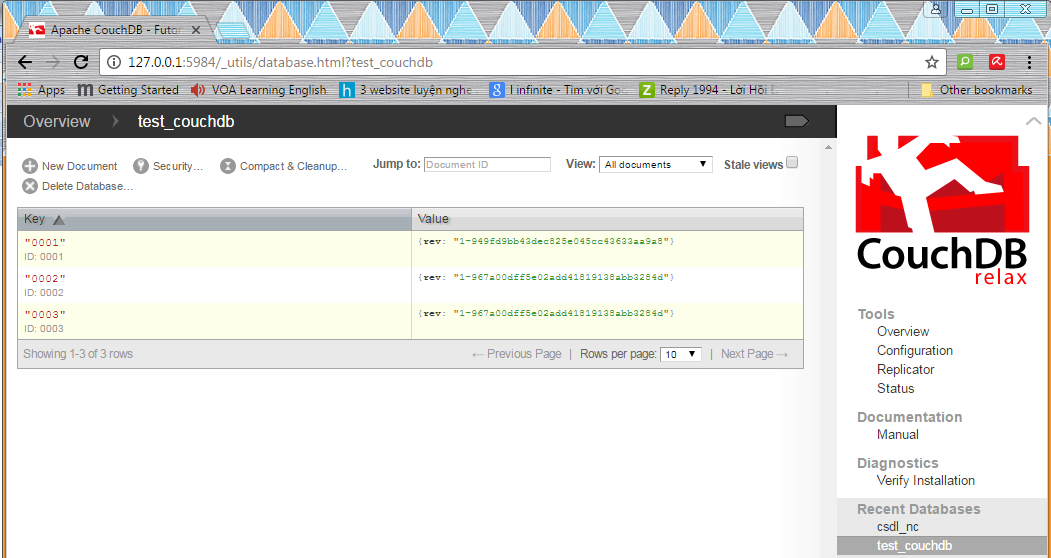
****

**i. Dùng Futon cập nhật document**

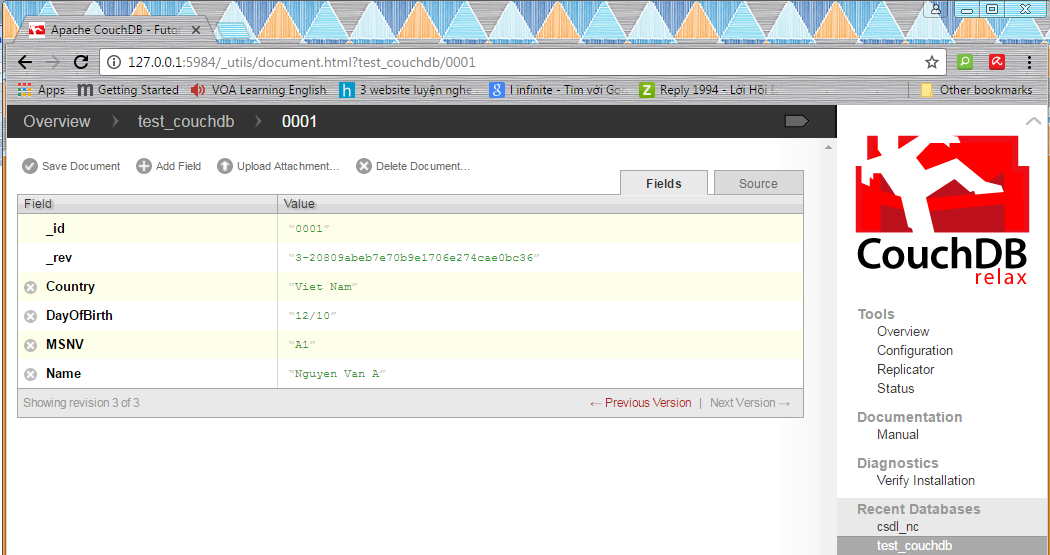
Để cập nhật một document, hãy mở đường dẫn **http://127.0.0.1:5984/\_utils/**  để mở trang Overview/index của CouchDB như hình dưới.



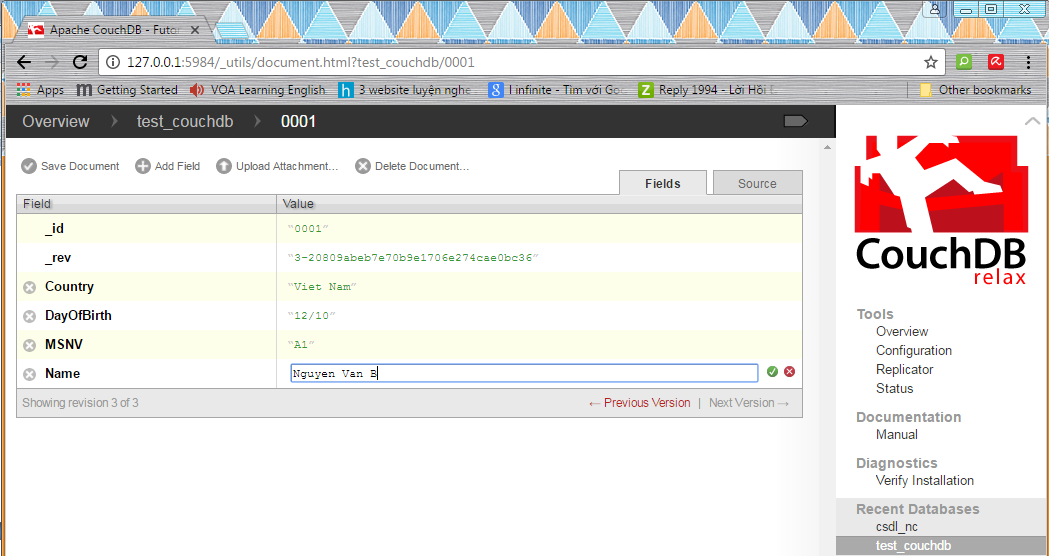
Chọn CSDL có document cần cập nhật và click vào nó. Ở đây chúng tôi cập nhật một document trong CSDL tên test\_couchdb. Bạn sẽ thấy danh sách các document trong các CSDL như hình dưới.



Chọn một document mà bạn muốn cập nhật và click vào nó. Bạn sẽ thấy được nội dung của document như hình dưới

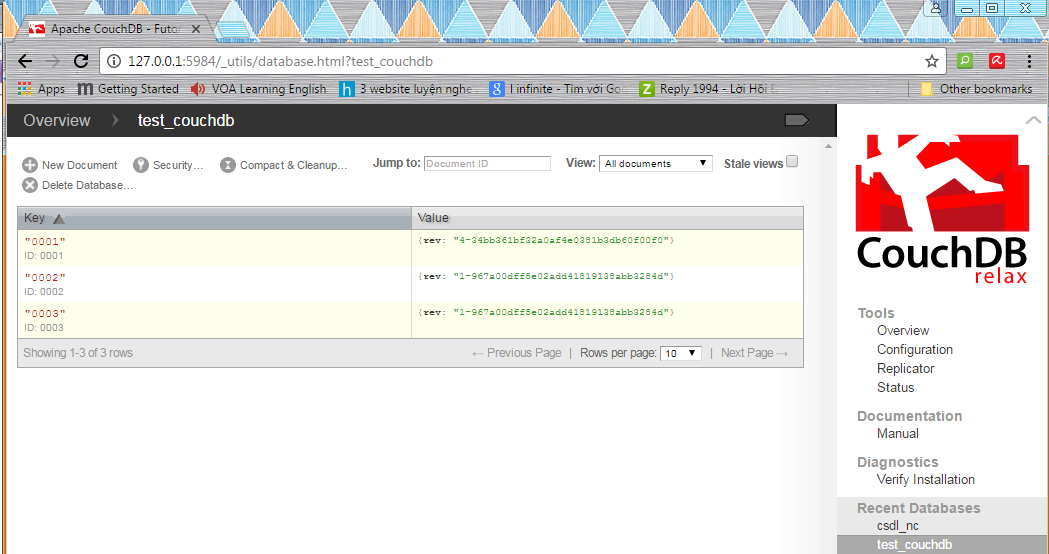


Ở đây, để cập nhật tên từ Nguyễn Văn A sang Nguyễn Văn B, double click vào text box, chỉnh sửa trường, và click nút màu xanh lá để lưu những thay đổi như hình dưới.

****

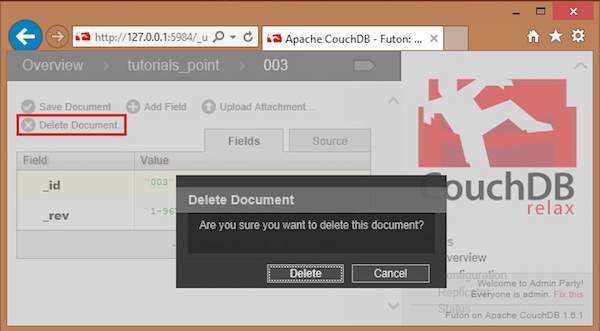
**j. Dùng Futon để xóa một document**

Đầu tiên, xác định các document trong CSDL. Hình sau là hình chụp màn hình của CSDL tên test\_couchdb.



Ở đây bạn có thể thấy, CSDL bao gồm 3 document. Để xóa bất kì cái nào, làm như sau:

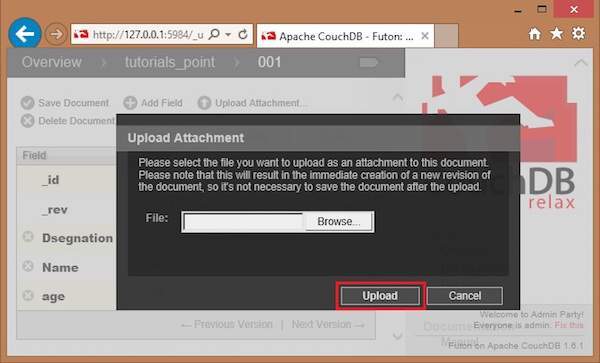
* Click vào document, bạn sẽ thấy một trang cho thấy nội dung của document đã chọn trong dạng những cặp field-value.
* Trang này cũng chứa bốn lựa chọn tên là Save Document, Add Field, Upload Attachment, Delete Document.
* Chọn lựa chọn Delete Document.
* Bạn sẽ thấy một hộp hội thoại thông báo là “Bạn có chắc bạn muốn xóa document này không?” Click vào delete để xóa document



**k. Dùng Futon để đính kèm file**

Tải file đính kèm

Dùng tùy chọn này, bạn có thể tải một file đính kèm mới như là một file, hình hoặc document, vào CSDL. Để làm vậy, click vào nút Upload Attachment. Một hộp hội thoại sẽ xuất hiện nơi mà bạn có thể chọn file để tải lên, Chọn file và click vào nút Upload.



File được tải lên sẽ được hiển thị dưới trường under\_attachments. Sau đó bạn có thể xem file bằng việc click vào nó.

## 3.10 Sử dụng MapReduce để chạy câu truy vấn

Các CSDL quan hệ truyền thống cho phép bạn chạy bất kì câu truy vấn nào miễn là dữ liệu của bạn được cấu trúc đúng. Ngược lại, CouchDB dùng các hàm map và reduce được định nghĩa lại theo một loại là MapReduce. Các chức năng này cung cấp sự linh hoạt cao vì chúng có thể tương thích với nhiều thể loại trong cấu trúc document, và các index vì mỗi document có thể được tính vào một cách độc lập và song song. Sự kết hợp của một map và một chức năng reduce được gọi là một view trong thuật ngữ CouchDB.

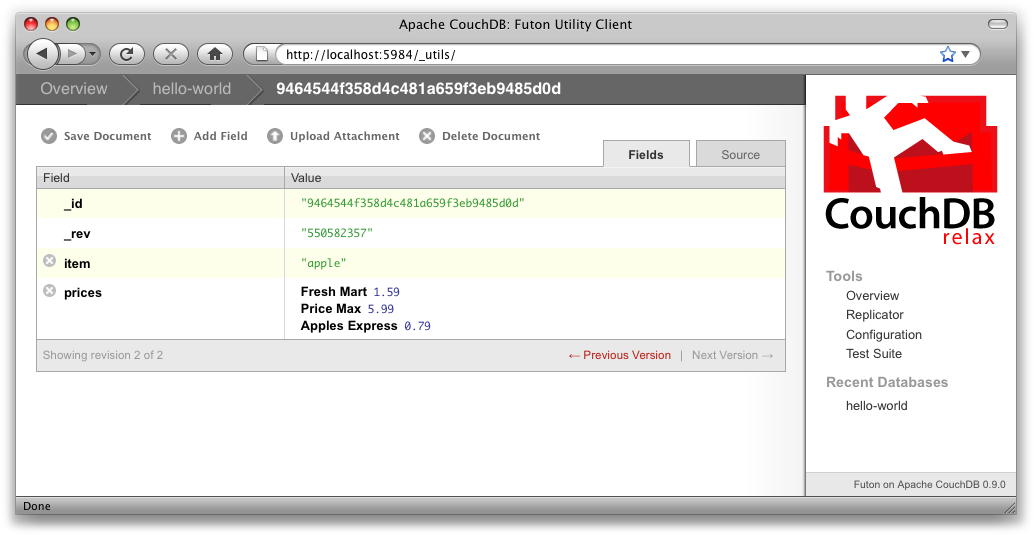
Các hàm map được gọi một lần với mỗi hàm số document. Hàm này có thể chọn duyệt qua tất cả document cùng một lúc hoặc emit một hoặc nhiều dòng view như các cặp key/value. Các hàm map có thể không phụ thuộc vào bất kì thông tin nào ngoài document. Tính độc lập này là cái mà cho phép các view CouchDB được sản sinh song song.

CouchDB views được lưu trữ như những dòng được sắp xếp bởi key. Điều này làm cho việc lấy dữ liệu từ một phạm vi của các key hiệu quả ngay cả khi có hàng ngàn hay hàng triệu dòng. Khi viết các hàm CouchDB map, mục tiêu chính của bạn là xây dựng một index mà lưu trữ dữ liệu liên quan dưới những key lân cận.

Trước khi chúng ta có thể chạy một ví dụ view MapReduce, chúng ta sẽ cần vài dữ liệu đã dùng. Chúng ta sẽ tạo các document chứa giá của nhiều sản phẩm ở nhiều cửa hàng. Hãy tạo document cho táo, cam và chuối. (Cho phép CouchDB sinh ra các trường \_id và \_rev). Dùng Futon để tạo các document mà có cấu trúc JSON như sau:

{  
 "\_id": "00a271787f89c0ef2e10e88a0c0001f4",  
 "\_rev": "1-2628a75ac8c3abfffc8f6e30c9949fd6",  
 "item": "apple",  
 "prices": {  
 "Fresh Mart": 1.59,  
 "Price Max": 5.99,  
 "Apples Express": 0.79  
 }  
}

Document khi đã được nhập vào Futon trông giống như hình dưới đây:

[[](http://guide.couchdb.org/draft/tour.html#figure/6)](http://guide.couchdb.org/draft/tour.html" \l "figure/6)

*[Figure 6. An example document with apple prices in Futon](http://guide.couchdb.org/draft/tour.html" \l "figure/6)*

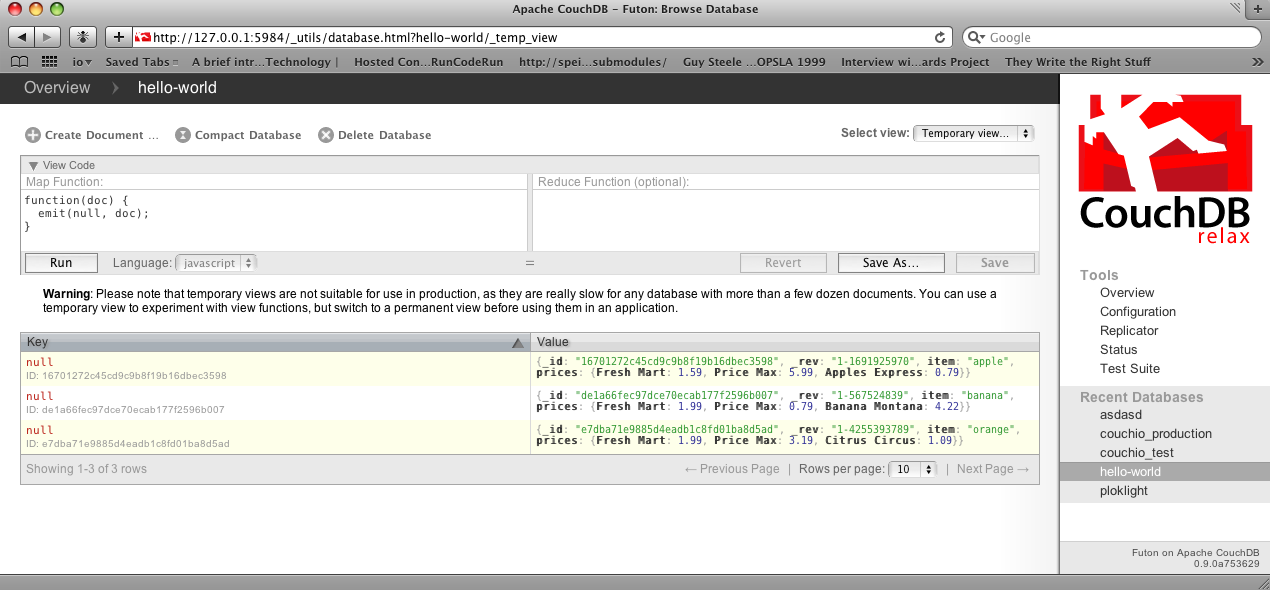
Xong, bây giờ hãy tạo document cho orange

{  
 "\_id": "00a271787f89c0ef2e10e88a0c0003f0",  
 "\_rev": "1-e9680c5d9a688b4ff8dd68549e8e072c",  
 "item": "orange",  
 "prices": {  
 "Fresh Mart": 1.99,  
 "Price Max": 3.19,  
 "Citrus Circus": 1.09  
 }  
}

Và cuối cùng, document cho banana

{  
 "\_id": "00a271787f89c0ef2e10e88a0c00048b",  
 "\_rev": "1-60e25d93dc12884676d037400a6fa189",  
 "item": "banana",  
 "prices": {  
 "Fresh Mart": 1.99,  
 "Price Max": 0.79,  
 "Banana Montana": 4.22  
 }  
}

Tượng tưởng chúng ta đang tìm món tráng miệng cho một bữa ăn, nhưng khách hàng rất nhạy cảm về giá cả. Để tìm giá thấp nhất, chúng ta sẽ tạo cái view đầu tiên, nơi cho thấy mỗi loại trái cây được sắp xếp theo giá. Click hello-world để quay lại trang tổng quan, rồi ở mục “select view” chọn “Temporary view” để tạo một view mới. Kết quả sẽ trông giống như hình dưới

[[](http://guide.couchdb.org/draft/tour.html#figure/7)](http://guide.couchdb.org/draft/tour.html" \l "figure/7)

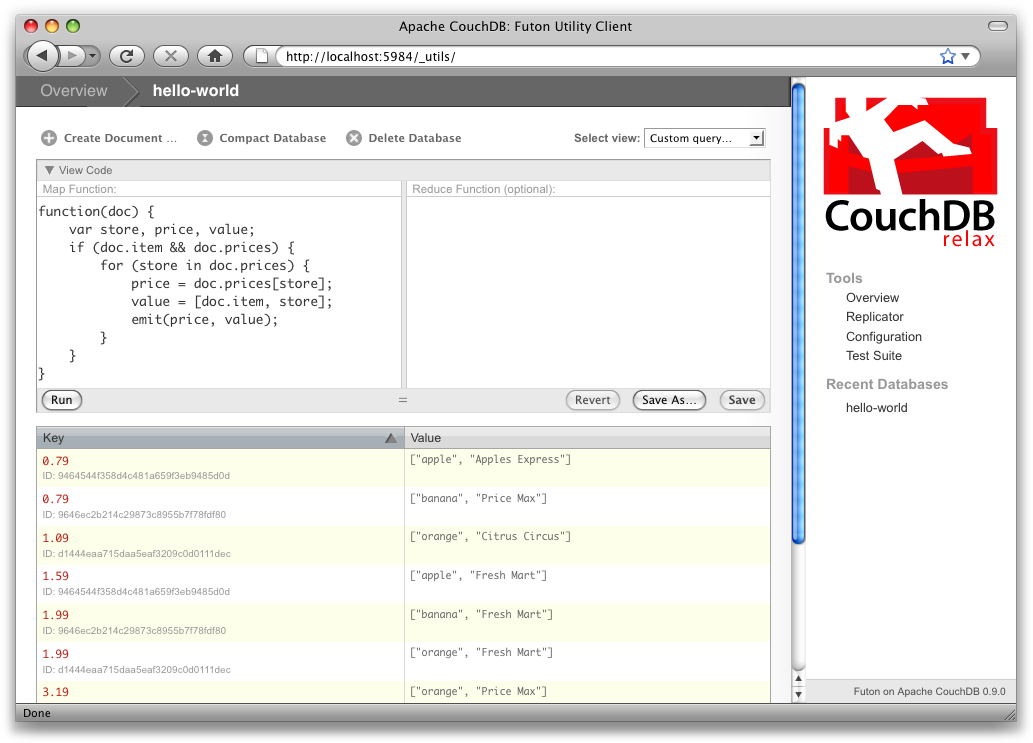
Chỉnh sửa hàm map ở bên trái như sau:

function(doc) {  
 var shop, price, value;  
 if (doc.item && doc.prices) {  
 for (shop in doc.prices) {  
 price = doc.prices[shop];  
 value = [doc.item, shop];  
 emit(price, value);  
 }  
 }  
}

Đây là hàm JavaScript mà CouchDB chạy cho mỗi document khi nó chạy view. Hiện tại chúng ta sẽ để hàm reduce trống.

Click “Run” và bạn sẽ thấy những dòng kết quả như trong hình dưới, với nhiều loại sản phẩm được sắp xếp theo giá cả. Hàm map này thậm chí có thể hữu ích hơn nếu nó gom nhóm các thành phần theo loại để mà tất cả giá cho chuối ở bên cạnh nhau trong cài đặt cuối cùng.

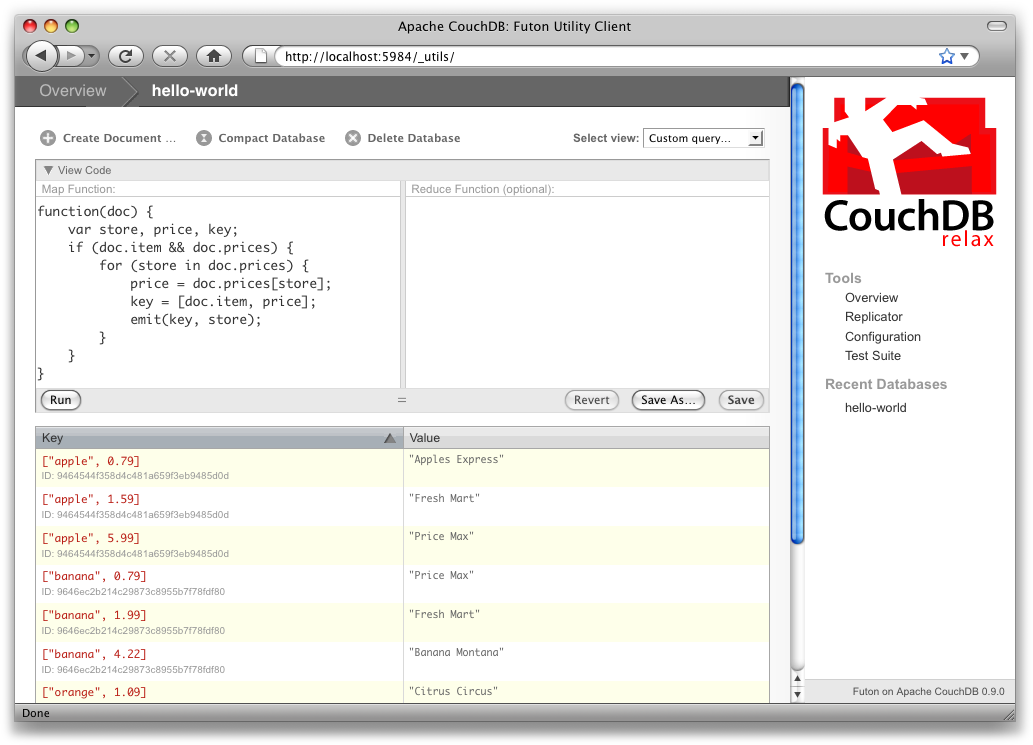
Hệ thống sắp xếp key của CouchDB cho phép bất kì đối tượng JSON hợp lệ nào làm key. Trong trường hợp này, chúng ta sẽ xuất ra (emit) một mảng của [item, price] để mà CouchDB gom nhóm bằng loại sản phẩm và giá cả.

[[](http://guide.couchdb.org/draft/tour.html#figure/8)](http://guide.couchdb.org/draft/tour.html" \l "figure/8)

Hãy cấu hình hàm view như thế này:

function(doc) {  
 var shop, price, key;  
 if (doc.item && doc.prices) {  
 for (shop in doc.prices) {  
 price = doc.prices[shop];  
 key = [doc.item, price];  
 emit(key, shop);  
 }  
 }  
}

Ở đây, đầu tiên chúng ta kiểm tra xem document có những trường chúng ta muốn sử dụng hay không, bỏ qua document hello world đầu tiên chúng ta đã tạo. Kết quả của câu truy vấn này trông giống như hình dưới đây:

[[](http://guide.couchdb.org/draft/tour.html#figure/9)](http://guide.couchdb.org/draft/tour.html" \l "figure/9)

Một khi chúng ta biết chúng ta đã có một document với một loại sản phẩm và một vài mức giá, chúng ta duyệt qua các mức giá của sản phẩm và in (emit) các cặp key/value. Key là một mảng sản phẩm và giá, và các dạng đơn giản cho index được sắp xếp của CouchDB. Trong trường hợp này, giá trị là tên của cửa hàng nơi sản phẩm có thể được tìm thấy cho các mức giá trong danh sách.

Các dòng view được sắp xếp bởi các key của chúng – trong vd này, đầu tiên là bằng sản phẩm, sau đó là bằng mức giá. Phương thức của việc sắp xếp phức tạp này là mấu chốt trong việc tạo ra những index hữu ích với CouchDB.

*MapReduce có thể là một thách thức, đặc biệt là nếu bạn đã làm việc với CSDL quan hệ nhiều năm. Điều quan trọng bạn cần nhớ là hàm map mang đến cho bạn một cơ hội để sắp xếp dữ liệu dùng bất kì key nào bạn chọn, và thiết kế của CouchDB tập trung vào việc cung cấp sự truy cập đến dữ liệu nhanh, hiệu quả trong một phạm vi của các key.*

**Tài liệu tham khảo**

<http://docs.couchdb.org/en/stable/intro/overview.html#document-storage>

<https://www.tutorialspoint.com/couchdb/couchdb_introduction.htm>

<http://www.allitebooks.com/datebases/>