بـــه نـــام خـــدا

عنوان مقاله

بررسی پایگاه داده Apache CouchDB

گردآورنده:

سجاد صبح خیـــز

برگرفته از وب سایت:

http://couchdb.apache.org/

فهرست

مقدمه

بطور کلی سیستم های مدیریت دیتابیس مکانیزمی برای ذخیره سازی و بازیابی اطلاعات فراهم می کنند. دو مدل بسیار مهم برای مدیریت دیتابیس عبارتند از: RDBMS و NoSQL

RDBMSها (مانند SQL SERVER، ORACLE، MYSQL) برای مدیریت دیتابیس های رابطه ای می باشند. با توجه به اینکه در عصر حاضر با حجم غول پیکری از داده ها دست و پنجه نرم می کنیم، سازماندهی و ساختارمندکردن داده ها، به ویژه در حجم های بسیار بزرگ به یک مشکل تبدیل شده است. رویکرد ساختاریافتگی RDBMSهایی مانند SQL در مواجهه با داده های بزرگ موجب کند شدن کارایی و تداخل در مقیاس پذیری میگردد. بنابراین NoSQL بعنوان چارچوبی کاملاً متفاوت از پایگاه داده مطرح گردید تا کارایی بالا و پردازشی سریع را در مقیاس بزرگتری از اطلاعات فراهم آورد.

اصطلاح NoSQL نامی عمومی ست که به مجموعه ای از پایگاه های داده اطلاق می شود که از زبان پرس و جوی ساخت یافته SQL یا مدل داده رابطه ای استفاده نمی کنند. گاهی این اصطلاح را مخفف NotOnlySQL می دانند تا تأکید کنند که طرفداران انواع دیتابیس های غیر رابطه ای معتقدند که دیتابیس های سنتی تنها راه موجود برای ذخیره سازی داده نیستند، امّا این به این معنا نیست که به خودی خود انتخاب نادرستی باشند.

دلایل استفاده از NoSQL

- ✓ مقیاس پذیری بالا – که در دیتابیس های رابطه ای یک موضوع پیچیده و گران تلقی میگردد.
- ✓ بدون ساختار بودن – ساختار ثابتی برای ذخیره داده ها تعریف نشده و داده با هر ساختاری را می توان در آنها ذخیره نمود.
- ✓ کارایی بالا
- ✓ استفاده از سخت افزار ارزانتر نسبت به RDBMSها
- ✓ مدیریت و نگهداری آسان تر

چه سازمان هایی از NoSQL استفاده می کنند؟

Google, Facebook, Mozilla, Adobe, Foursquare, LinkedIn, Digg, ...

مقیاس پذیری

مهمترین مزیت دیتابیس های NoSQL قابلیت توسعه ی بالای این نوع دیتابیس ها می باشد. این قابلیت به دیتابیس اجازه میدهد که براحتی گسترش یابد. همچنین مقیاس پذیری در این نوع دیتابیس ها برخلاف دیتابیس های رابطه ای بسیار ارزانتر بوده و براحتی امکانپذیر است. این نوع توسعه در دیتابیس های NoSQL به scale-out یا توسعه افقی مشهور است. در صورتی که توسعه ی دیتابیس های رابطه ای بصورت عمودی بوده و scale-up نامیده می شود چون در اینگونه دیتابیس ها تنها راهکار توسعه این است که منابع سخت افزاری سرور را ارتقاء دهید که در مدیریت داده های کلان بسیار گران و هزینه آور خواهد بود.

مدیریت داده های کلان

حجم محتوای تولید شده در چند سال اخیر نسبت به روزهای ابتدایی پیدایش وب رشد چشمگیری داشته و همچنین روز به روز بر تعداد کاربران اینترنت افزوده می شود. با توجه به اینکه RDBMSها در مواجهه با

داده های کلان با مشکل جدی روبرو هستند، برای مدیریت این حجم داده ها نیاز به سخت افزارهای خاص و پرهزینه دارند. همچنین پرس و جو های SQL بر روی این داده ها بسیار زمان بر خواهد بود. در حالیکه یکی از مهمترین مزیت های NoSQL توانایی مدیریت دیتاها با حجم بسیار بالا می باشد.

بهترین گزینه برای انتخاب نوع دیتابیس

با وجود مزایای مطرح شده نمی توان اظهار داشت که دیتابیس های NoSQL در همه ی موارد نسبت به دیتابیس های رابطه ای برتر هستند و بطور کامل می توانند بعنوان جایگزینی مناسب برای RDBMSها استفاده شوند زیرا بسیاری از موارد (خصوصاً پروژه هایی با مقیاس کوچک) وجود دارد که در آنها دیتابیس های رابطه ای انتخاب بهتری خواهند بود. در واقع NoSQLها در برنامه هایی که ذخیره و بازیابی حجم عظیمی از اطلاعات بسرعت انجام می شود کاربرد بسیار دارند و RDBMSها برای برنامه هایی استفاده می شوند که داده ها ساختارمند بوده و باید پرس و جو های پیچیده روی آنها صورت گیرد. بنابراین تا زمانی که نیاز به استفاده از NoSQLها نباشد بهترین گزینه برای انتخاب، دیتابیس رابطه ای می باشد. در واقع پیشنهاد میگردد زمانی از دیتابیس NoSQL استفاده کنید که:

- ✓ داده های شما بسرعت در حال افزایش هستند و به مقیاس بزرگی می رسند.
- ✓ فیلدهای داده ای ممکن است تغییر کنند. (اضافه و یا کم شوند)
- ✓ مقادیر فیلدها آرایه ای هستند و بدلیل دستیابی به سرعت بالاتر نمی خواهید از پیوند بین جداول استفاده کنید.
- ✓ نگران این هستید که سرور بدلیل درخواست های زیاد از سوی کاربران، از دسترس خارج شود.
- ✓ نیاز دارید که از سیستم های توزیع شده استفاده کنید.

انواع دیتابیس های NoSQL

- ✓ Key-value Stores که پایه ی دیتابیس های NoSQL را تشکیل داده و اهدافی عمومی را دنبال می کنند.
- ✓ Wide column Stores که بیشتر در شرکت های بزرگ اینترنتی مورد استفاده قرار می گیرند.
- ✓ Document Stores یا دیتابیس های سندگرا
- ✓ Graph Database که بیشتر برای ردیابی ارتباطات بین موجودیت ها بکار می روند.

باتوجه به اینکه موضوع مورد بررسی، دیتابیس CouchDB می باشد در خصوص دیتابیس های سندگرا در ادامه توضیحاتی ارائه میگردد.

دیتابیس های سندگرا مجموعه ای از اسناد (document) هستند که به زبان JSON ذخیره می شوند. اساس ذخیره سازی داده ها همانند key-value storeها بوده و اسناد بصورت مجموه ای از کلید/مقدار ها هستند که از طریق کلید می توان به مقدار آن دسترسی داشت. این سند ها ساختار مشخص و از پیش تعریف شده ای ندارند و داده های آن را می توان بصورت جفت کلید/مقدار، جفت کلید/آرایه و حتی بصورت سندهای تودرتو تعریف کرد.


```
{
  "_id": "d6184bf1b414a0d6e459d670fd0002c8",
  "_rev": "8-110322fbcae3c661c4628e4b43cd8666",
  "title": "Learning about CouchDB",
  "author": "Mr.Writer",
  "date": "2016-11-25 22:48:05",
  "body": "A CouchDB server hosts named databases, which store documents. Each document is uniquely named in the database, and
  CouchDB provides a RESTful HTTP API for reading and updating (add, edit, delete) database documents.",
  "tags": ["it","database","nosql","couchdb"],
  "comments": [
    {
      "username": "user1",
      "date": "2016-11-25 23:00:00",
      "body": "thanks for this article"
    },
    {
      "username": "user2",
      "date": "2016-11-28 22:48:05",
      "body": "that's be better..."
    },
    {
      "username": "user3",
      "date": "2016-11-29 11:48:05",
      "body": "very good!!"
    },
    {
      "username": "user4",
      "date": "2016-11-27 12:30:05",
      "body": "very helpful for meeeeeeee"
    },
    {
      "username": "user5",
      "date": "2016-11-30 22:48:05",
      "body": "nice"
    }
  ]
}
```


شکل (۱) – نمونه ای از یک سند در دیتابیس سندگرا

معرفی CouchDB

در سال ۲۰۰۵ Damien Katz پروژه ای در خصوص یک پایگاه داده جدید مطرح کرد و نام آن را CouchDb نهاد و در سال ۲۰۰۸ امتیاز این پروژه رسماً توسط Apache خریداری و بعنوان یک دیتابیس NoSQL به بازار عرضه شد. CouchDB یک دیتابیس سندگرا ست که بطور کامل پذیرای بستر وب بوده و برای وب سایت ها و برنامه های تحت وب می باشد و داده ها در آن با فرمت JOSN ذخیره می شوند. این دیتابیس به زبان Erlang نوشته شده و در تمام سیستم عامل ها اعم از OSX، Linux و Windows

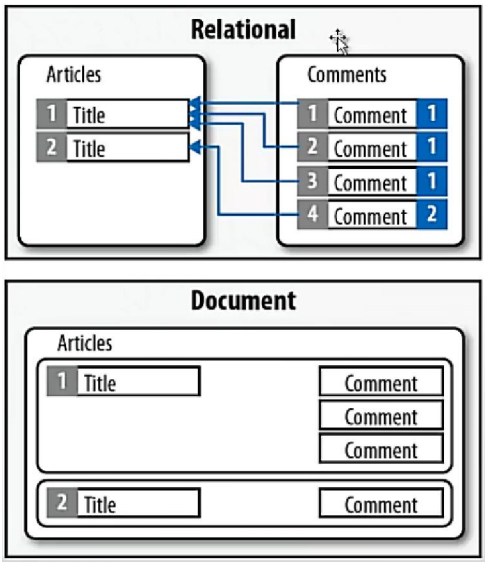
قابل اجرا می باشد. در CouchDB از پروتکل های HTTP برای دسترسی و بروزرسانی اسناد، استفاده شده و برای کار با داده ها از زبان JavaScript بهره می برد.

مزیت های CouchDB

- ✓ سرعت بالا
- ✓ کاربری آسان
- ✓ عدم وجود پیوند بین اسناد (تعریف اسناد بصورت self-contained)
- ✓ استفاده از HTTP RESful API برای خواندن و بروزرسانی اسناد
- ✓ هسته اصلی ساده و در عین حال قدرتمند
- ✓ مدیریت راحت ترافیک کاربران
- ✓ Replication

ذخیره اسناد

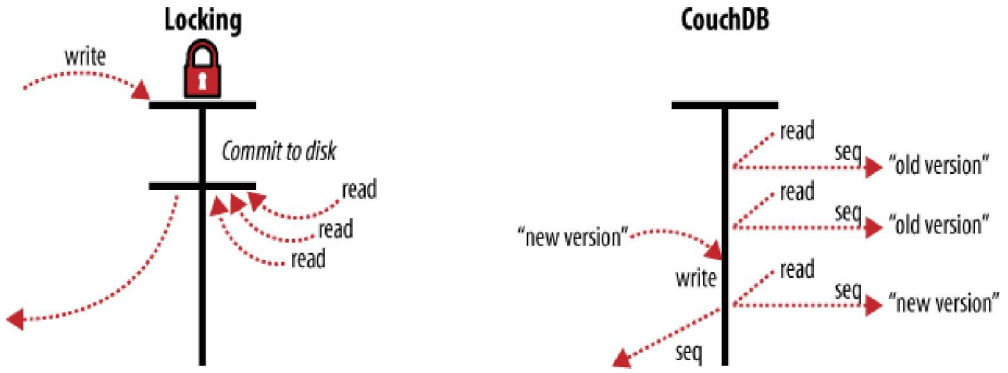
سندها واحد اصلی اطلاعات در CouchDB هستند که از تعداد نامحدودی فیلد (کلید) و فایل پیوست تشکیل شده اند. دیتاها در داخل این سندها ذخیره می شوند و هر سند در دیتابیس یک نام منحصربفرد دارد، همچنین عنوان هر فیلد (کلید) نیز در هر سند باید یکتا باشند. هیچ محدودیتی برای تعداد کلیدهای تعریف شده برای سندها وجود ندارد. مقادیر کلیدها نیز می تواند از هر نوعی باشد (متن، عدد، boolean، لیست و ...). همانگونه در شکل (۲) نشان داده شده برخلاف دیتابیس های رابطه ای اسناد موجود در CouchDB هیچگونه وابستگی به دیگر اسنادِ موجود در دیتابیس ندارند. همچنین از طرح (schema)ی خاص و از پیش تعریف شده ای پیروی نمی کنند. این دیتابیس در برخورد با درخواست های همزمان بجای استفاده از تکنیک قفل گذاری، از روش چند نسخه ای (MVCC) بهره می برد. با استفاده از این تکنیک بروزرسانی اسناد بصورت سریالی انجام می گیرد و به هیچگونه قفل گذاری نیاز نیست.



شکل (۲)

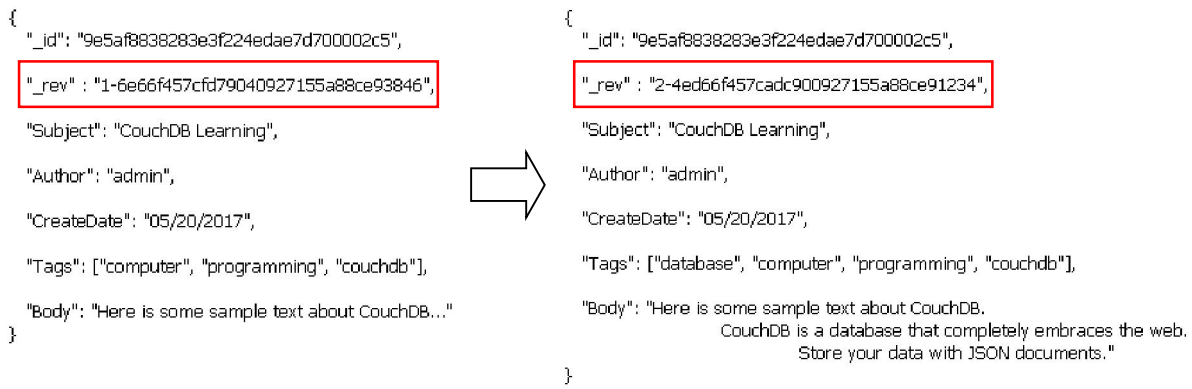
خاصیت ACID

در CouchDB تمام ویژگی های ACID پشتیبانی می شود. هیچگونه قفل گذاری صورت نمی گیرد و از خاصیت چند نسخه ای برای کنترل همروندی استفاده می شود. وقتی یک درخواست خواندن از طرف کاربر ارسال می شود همیشه آخرین تصویر از دیتابیس در لحظه ی شروع درخواست به کاربر ارائه می گردد.



استفاده از MVCC باعث می شود که CouchDB در هر زمان با نهایت سرعت اجرا شده و پاسخگوی درخواست های کاربران باشد. (حتی در صورت وجود درخواست های موازی پی در پی). یعنی کاربران می توانند در هر لحظه دیتاها را بخوانند بدون اینکه هیچ اختلالی در سیستم بوجود آید. در این روش در هنگام بروزرسانی اسناد، هیچ سندی از بین نمی رود بلکه یک نسخه ی جدید از آن ایجاد می شود.

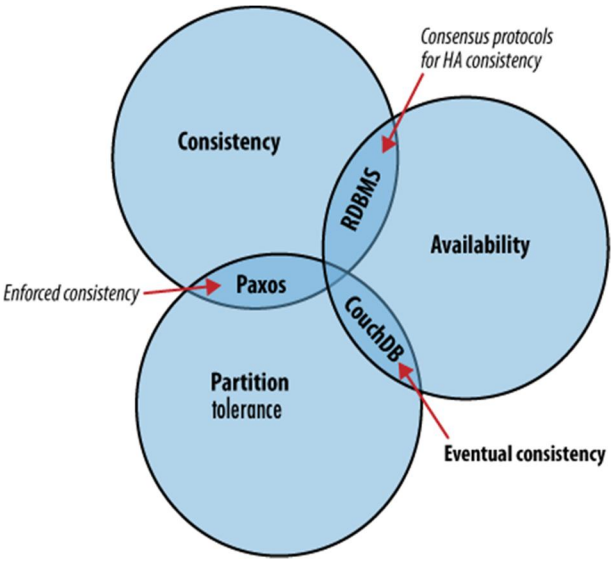
همانگونه در شکل های ۳ و ۴ نشان داده شده بعد از ایجاد سند، بطور خودکار یک فیلد با عنوان _rev به فیلدهای سند افزوده می شود که نشان دهنده ی شماره ی نسخه سند می باشد. بعد از اولین بروزرسانی، فیلد _rev خودبخود مقدار جدید می گیرد. عدد ۲ درج شده در ابتدای مقدار فیلد _rev بیانگر این امر است که نسخه نمایش داده شده، دومین نسخه از سند مربوطه می باشد.



شکل (۳)- بعد از ایجاد سند

شکل (۴)- بعد از اولین بروزرسانی

تئوری CAP



با توجه به رشد روزافزون اطلاعات و افزایش دسترسی کاربران به اینترنت وجود سیستم های توزیع شده برای مدیریت این اطلاعات احساس گردید. با افزودن هر گره در این سیستم ها می توان بازدهی و توانایی آنها را بالا برد که به این نوع گسترش ، توسعه افقی(scale-out) گفته می شود. تئوری CAP برای رفع پیچیدگی اینگونه شبکه ها مطرح شده است.

مواردی که در این تئوری مطرح می شود عبارتند از:

Consistency: همه ی کلاینت ها در هر لحظه حتی در صورت بروزرسانی های همزمان، داده های یکسان را مشاهده می کنند.

Availability: همه ی کلاینت ها در هر لحظه می توانند به داده های دیتابیس دسترسی داشته باشند.

Partition Tolerance: اگر بخشی از شبکه دچار مشکل شد، بقیه سیستم به کار خود ادامه خواهد داد. اشکال در بخشی از شبکه نباید منجر به توقف کار کل سیستم شود.

براساس این تئوری مسئله ای که در دیتابیس CouchDB ارجحیت دارد قابلیت در دسترس بودن سیستم است (برخلاف RDBMSها که سازگاری را در الویت قرار می دهند). یعنی در دسترس بودن و سرعت بیشتر الویت بالاتری نسبت به ثبات و دوام اطلاعات دارد. به همه ی درخواست ها پاسخ داده می شود و حتی در صورتیکه بروزرسانی هم ممکن نباشد، سیستم به کار خود ادامه خواهد داد و در نهایت به ثبات اطلاعات (Eventual Consistency) خواهد رسید.

View

داده ها در CouchDB بصورت سندهای بدون ساختار ذخیره می شوند که هیچ ارتباطی با یکدیگر ندارند. در نتیجه باید روشی وجود داشته باشد که بتوان این داده ها را برای کاربر قابل نمایش ساخت. در واقع از viewها برای سازماندهی، فیلتر گذاری و گزارشگیری از داده هایی استفاده می کنیم که اصولاً بصورت جدول نیستند. همانند دیتابیس های رابطه ای viewها هیچ تأثیری روی اسناد ندارند و فقط جهت نمایش اسناد بکار می روند. viewها به زبان JavaScript و در نگارش Map/Reduce نوشته می شوند و در داخل یک سند مخصوص طراحی به نام _design ذخیره می شوند.

```
"customers": {
        "map": "function(doc) {if(doc.type=='customer') emit(doc.name, null)}",
        "reduce": "_count"
    }
```

در نگارش Map/Reduce تابع map یک سند را بعنوان ورودی می گیرد و لیستی از جفت کلید/مقدارها را بعنوان خروجی بر می گرداند و به یک مقدار قابل نمایش تبدیل می کند. تابع reduce آرایه ای از کلید/مقدار ها را می گیرد و یک مقدار را بعنوان خروجی نمایش می دهد. این تابع حتماً باید بعد از تابع map فراخوانی شود. از reduce برای گروه بندی نتایج استفاده می شود. بکار بردن این تابع کاملاً اختیاری بوده و ممکن است اصلاً فراخوانی نشود. برای درک بهتر این توابع در ادامه چند مثال از این نوع نگارش بیان میگردد.

```
function(doc){
        if(doc.date && doc.title){
                emit(doc.date, doc.title);
        }
}
```

Key	Value
"2009/01/15 15:52:20"	"Hello World"
"2009/02/30 18:09:10"	"Breaking Bad"
"2012/07/27 09:18:20"	"Banking"

```
function(doc){
        if(doc.type=="customer"){
                emit(doc._id, doc.fullname);
        }
}
```

Key	Value
"979634e52d3b95f05d"	"Adam Norton"
"6c453fbfece66e2f994"	"Robert Watson"
"d3b95f05dbb2e578f0"	"Tim Cook"

Viewهای موقت

علاوه بر Viewهایی که در دیتابیس ذخیره می شوند نوع دیگری View وجود دارد که بصورت فایل ذخیره نمی شود ولی از طریق یک درخواست POST قابل دسترسی است و به Temporary View شهرت دارند.

فشرده سازی و پاکسازی

یکی از ویژگی هایی که در دیتابیس CouchDB تعبیه شده فشرده سازی (Compaction) و پاکسازی (Cleanup) می باشد. فشرده سازی به معنای آزادسازی فضای حافظه با حذف بخش های بلا استفاده در دیتابیس می باشد. با استفاده از این روش تمامی اسنادِ حذف شده و نسخه های پیشین از کلیه اسناد و viewها حذف شده و فضای اشغال شده توسط آنها آزاد میگردد.

در دیتابیس CouchDB زمانی که viewها بروزرسانی می شوند شاخص های قبلی آنها روی حافظه باقی می ماند و به همین دلیل از خاصیت Cleanup برای حذف شاخص های منقضی شده ی viewها استفاده می شود.

امنیت و اعتبارسنجی

بعنوان یک توسعه دهنده ی دیتابیس ما باید بدانیم که چه کسانی به دیتابیس دسترسی دارند و می توانند دیتاها را ویرایش کنند. در حالت پیشفرض بعد از نصب اولیه ی CouchDB کلیه کاربرانی که به دیتابیس دسترسی دارند نقش مدیر (admin) را دارند. admin می تواند کاربرانی را با نقش مدیر یا کاربر عادی تعریف کند. علاوه بر تعریف کاربرانِ مختلف با دسترسی های گوناگون، با استفاده از توابع موجود در JavaScript می توان روی اسناد اعتبارسنجی انجام داد تا ورودی هایی که کاربران هنگام بروزرسانی اسناد وارد می کنند بررسی شود.

توابع اعتبارسنجی (Validate_doc_update) سه نوع پارامتر بعنوان ورودی دریافت می کنند:

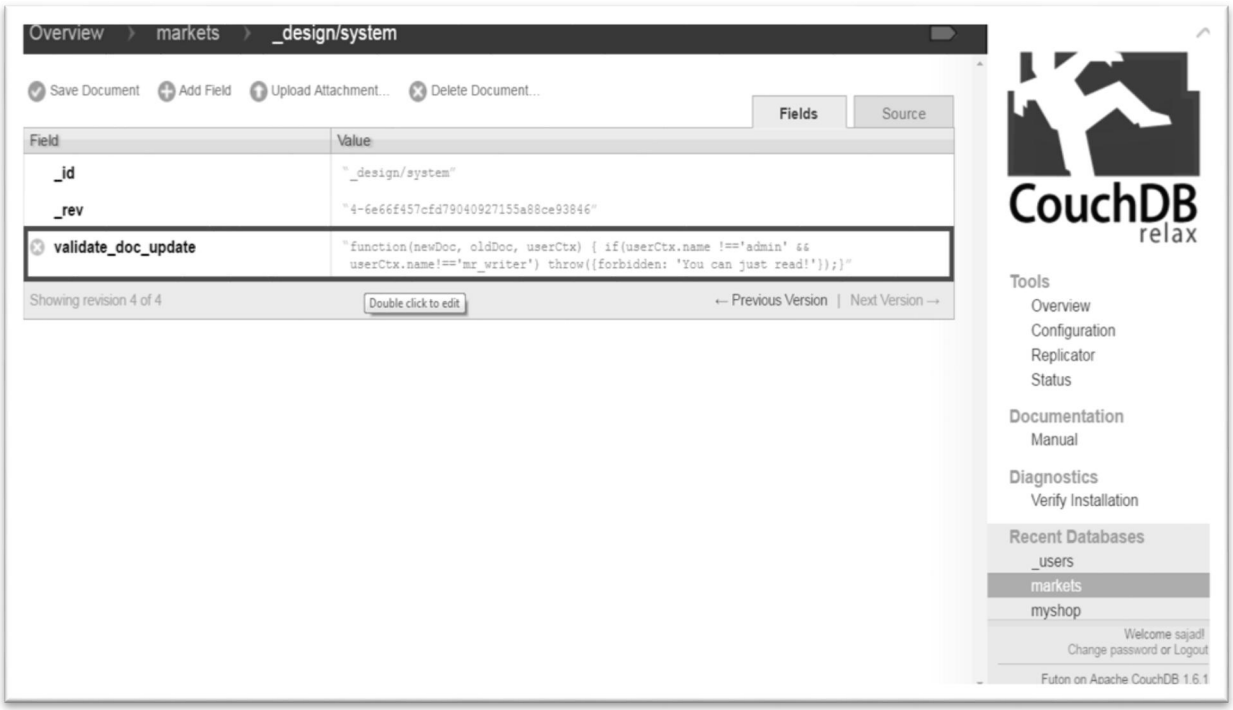
newDoc: نسخه بروزرسانی شده ی سند

oldDoc: نسخه حال حاضر سند

userCtx: اطلاعات مربوط به کاربر موردنظر

خروجی تابع بیانگر این است که سند موردنظر می بایست بروز رسانی گردد یا خیر؟!

در شکل زیر نمونه ای از یک تابع اعتبارسنجی که در محیط Futon تعریف شده را مشاهده می نمایید:



خروجی تابع بالا بیان می کند که فقط کاربرانی به نام admin و mr_writer اجازه ی بروزرسانی اسناد در دیتابیس را دارند.

Replication

CouchDB به کاربران خود اجازه می دهد که به داده های دیتابیس حتی در زمانی که به اینترنت متصل نیستند دسترسی داشته باشند و بتوانند تغییرات موردنیاز را روی دیتابیس اعمال کنند. این تغییرات را می توان به محض اتصال به اینترنت بصورت دوطرفه انتقال داد. این قابلیت باعث می شود که تمامی اسناد و viewها و حتی کلیه توابعی که برای امنیت و اعتبارسنجی نوشته شده اند در تمامی سرورها توزیع شده و برای همه قابل دسترس باشند. عملیات Replication بصورت تدریجی انجام می شود یعنی فقط تغییراتی اعمال می گردد که پس از آخرین عملیات Replication صورت گرفته است (برای مثال فقط فیلدهایی که دچار تغییر شده اند بروزرسانی می گردند) اگر به هر دلیلی این عملیات با مشکل مواجه شود در اجرای بعدیِ عملیات، همگام سازی از همان سندی که در هنگام بروز مشکل رها شده بود آغاز می گردد. این همگام سازی بین سرورها در مکان های فیزیکی مختلف می تواند باشد.

پایان