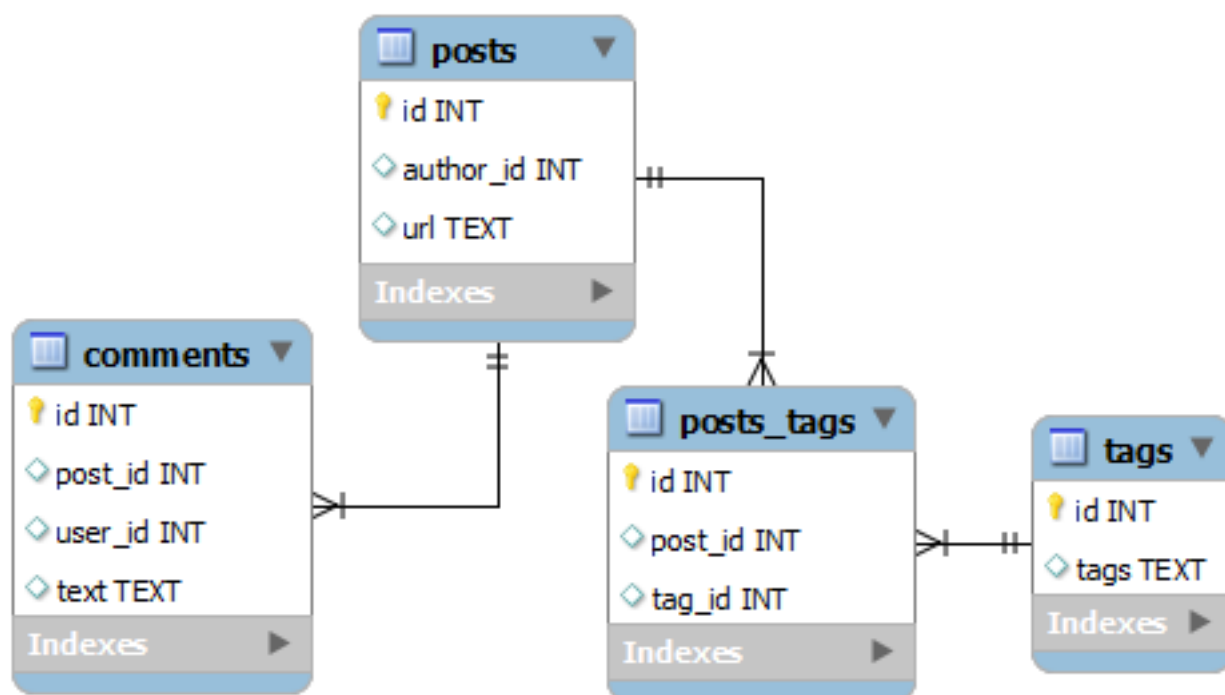


در مطلب قبلی با نوع اول پایگاه‌های داده NoSQL یعنی Key/Value Store آشنا شدیم و در این مطلب به معرفی دسته دوم یعنی Document Database خواهیم پرداخت.

در این نوع پایگاه داده، داده‌ها مانند نوع اول در قالب کلید/مقدار ذخیره می‌شوند و بازگردانی مقادیر نیز دقیقاً مشابه نوع اول یعنی Key/Value Store بر اساس کلید می‌باشد. اما تفاوت این سیستم با نوع اول در دسته‌بندی داده‌های مرتبط با یکدیگر در قالب یک Document می‌باشد. سعی کردم در این مطلب با ذکر مثال مطالب را شفاف‌تر بیان کنم:

به عنوان مثال اگر بخواهیم جداول مربوط به پست‌های یک سیستم CMS را بصورت رابطه‌ای پیاده کنیم، یکی از ساده‌ترین حالات پایه برای پست‌های این سیستم در حالت نرمال به صورت زیر می‌باشد.



جداول واضح بوده و نیازی به توضیح ندارد، حال نحوه‌ی ذخیره‌سازی داده‌ها در سیستم Document Database برای چنین مثالی را بررسی می‌کنیم:

```

{
  _id: ObjectId('4bf9e8e17cef4644108761bb'),
  Title: 'NoSQL Part3',
  url: 'http://dotnettips.info/yyy/xxxx',
  author: 'hamid samani',
  tags: ['databases', 'mongoDB'],
  comments: [
    {user: 'unknown user',
      text: 'unknown test'
    },
    {user: 'unknown user2',
      text: 'unknown text2'
    }
  ]
}
  
```

```
}
}
}
```

همانگونه که مشاهده می‌کنید نحوه‌ی ذخیره‌سازی داده‌ها بسیار با سیستم رابطه‌ای متفاوت می‌باشد ، با جمع‌بندی تفاوت نحوه‌ی نگهداری داده‌ها در این سیستم و RDBMS و بررسی این سیستم نکات اصلی به شرح زیر می‌باشند:

۱- فرمت ذخیره سازی داده‌ها مشابه فرمت JSON می‌باشد.

۲- به مجموعه داده‌های مرتبط به یکدیگر Document گفته می‌شود.

۳- در این سیستم JOIN ها وجود ندارند و داده‌های مرتبط کنار یکدیگر قرار می‌گیرند ، و یا به تعریف دقیق‌تر داده‌ها در یک داکيومنت اصلی Embed می‌شوند .

به عنوان مثال در اینجا مقدار comment ها برابر با آرایه‌ای از Document ها می‌باشد.

۴- مقادیر می‌توانند بصورت آرایه نیز در نظر گرفته شوند.

۵- در سیستم‌های RDBMS در صورتی که بخواهیم از وجود JOIN ها صرف‌نظر کنیم. به عدم توانایی در نرمال‌سازی بخواهیم خورد که یکی از معایب عدم نرمال‌سازی وجود مقادیر Null در جداول می‌باشد؛ اما در این سیستم به دلیل Schema free بودن می‌توان ساختارهای متفاوت برای Document ها در نظر گرفت.

به عنوان مثال برای یک پست می‌توان مقدار n کامنت تعریف کرد و برای پست دیگر هیچ کامنتی تعریف نکرد.

۶- در این سیستم اصولاً نیازی به تعریف ساختار از قبل موجود نمی‌باشد و به محض اعلان دستور قرار دادن داده‌ها در پایگاه داده ساختار متناسب ایجاد می‌شود.

با مقایسه دستورات CRUD در هر دو نوع پایگاه داده با نحوه‌ی کوئری گرفتن از Document Database آشنا می‌شویم:

در SQL برای ایجاد جدول خواهیم داشت:

```
CREATE TABLE posts (
  id INT NOT NULL
    AUTO_INCREMENT,
  author_id INT NOT NULL,
  url VARCHAR(50),
  PRIMARY KEY (id)
)
```

دستور فوق در Document Database معادل است با:

با قرار دادن مقدار نوع // db.posts.insert({id: "256" , author\_id:"546",url:"http://example.com/xxx"}) ساختار مشخص می‌شود

در SQL جهت خواندن خواهیم داشت:

```
SELECT * from posts  
WHERE author_id > 100
```

و معادل آن برابر است با:

```
db.posts.find({author_id:{$gt:"1000"}})
```

در SQL جهت بروزرسانی داریم:

```
UPDATE posts  
SET author_id= "123"
```

که معادل است با:

```
db.posts.update({ $set: { author_id: "123" } })
```

در SQL جهت حذف خواهیم داشت:

```
DELETE FROM posts  
WHERE author_id= "654"
```

که معادل است با:

```
db.posts.remove( { author_id: "654" } )
```

همانگونه که مشاهده می‌فرمایید نوشتن کوئری برای این پایگاه داده ساده بوده و زبان آن نیز بر پایه جاوا اسکریپت می‌باشد که برای اکثر برنامه‌نویسان قابل درک است.

تاکنون توسط شرکت‌های مختلف پیاده‌سازی‌های مختلفی از این سیستم انجام شده است که از مهم‌ترین و پر استفاده‌ترین آنها می‌توان به موارد زیر اشاره کرد:

[MongoDB](#)

[CouchDB](#)

[RavenDB](#)

## نظرات خوانندگان

نویسنده: سعید یزدانی  
تاریخ: ۱۹:۵۷ ۱۳۹۱/۱۱/۲۹

با تشکر از مطلب زیباتون  
یک سوال داشتم ایا این روش اونقدر به بلوغ رسیده که بشه در پروژه‌ها روش حساب کرد . یا اینکه فعلا از همون روش قبلی استفاده کنیم  
سوال دیگر من هم این هست که به نظر شما در nosql آینده ایی دیده میشه ؟  
با تشکر

نویسنده: سعید یزدانی  
تاریخ: ۱۹:۵۹ ۱۳۹۱/۱۱/۲۹

اگر هم امکان داره refrence ی در این زمینه هست link بدید

نویسنده: حمید سامانی  
تاریخ: ۲۱:۳ ۱۳۹۱/۱۱/۲۹

در رابطه با سوال اولتون عارضم که در حال حاضر همه‌ی شرکت‌های بزرگ و فعال در این صنعت مثل گوگل ، فیس ب و ک توئیترو از این شیوه استفاده می‌کنند ، در حالت کلی این مبحث یک تکنولوژی خاص نیست که مصرفی باشه و بعد از مدتی تاریخش بگذره ، یک Movement و یا یک نگرش کلی در تعریف عامه از مجموعه‌ای از راه حل‌ها به منظور رفع مشکلات RDBMS در پردازش داده‌های بزرگ (BigData) ، داده‌ها در حوزه‌ی وب هم که رشدی نمایی دارند.  
در رابطه با سوال دوم هم بستگی به خود فرد و یا شرکت مربوطه داره ، در حوزه‌ی نرم‌افزارهای داخلی به دلیل پایین‌تر بودن حجم داده‌ها الزامی در استفاده از این روش‌ها نیست. (استفاده و یا عدم استفاده مستقیما به نوع نرم‌افزار و ساختار آن بستگی دارد)

نویسنده: حمید سامانی  
تاریخ: ۲۱:۵ ۱۳۹۱/۱۱/۲۹

از [اینجا](#) که شما شروع کنید به همه جا لینک می‌شوید .)

نویسنده: سعید یزدانی  
تاریخ: ۲۱:۲۴ ۱۳۹۱/۱۱/۲۹

ممنون بابت جواب کاملتون

نویسنده: توحید عزیزی  
تاریخ: ۳:۵۲ ۱۳۹۱/۱۱/۳۰

سلام

سپاسگزارم از موضوع جالبی که انتخاب کرده اید و مطالب خوبی که می‌نویسید.  
آیا امکان دارد که در مورد هر کدام از انواع دیتابیس نوسیکوئل، مثالهای بیشتری بزنید.  
از یک سرویس رایگان برای نوشتن مثال‌ها می‌تواند استفاده کرد که برای همه در دسترس باشد، مثل: [cloudant.com](http://cloudant.com)  
با تشکر

نویسنده: Meysam Navaei  
تاریخ: ۹:۱۶ ۱۳۹۱/۱۱/۳۰

سلام

تو حید این سایت که معرفی کردی خیلی جالب بود. می خاستم بینم محدودیت حجمی در استفاده ازش وجود داره یا نه نامحدود. ریسک محسوب نمیشه ی پروژه بزرگ داشته باشی و بخای دیتابیس رو از سرویس این سایت استفاده کنی؟ منظورم اینکه از این سایتها نباشه که یهو محدودیت ایجاد بکنه و یا پولی بشه و...

نویسنده: حمید سامانی  
تاریخ: ۱۶:۱۵ ۱۳۹۱/۱۱/۳۰

سلام

سعی می کنم مثال های بیشتری را در مطالب آتی بگنجانم.  
(با سپاس)

نویسنده: تو حید عزیزی  
تاریخ: ۹:۳ ۱۳۹۱/۱۲/۰۳

سلام.

نسخه رایگانش محدودیت داره: فکر کنم 2000 کوئری در روز.  
اگر می خواهید پروژه ی بزرگ روش ببرید، باید از نسخه های تجاریش استفاده کنید. البته من خودم تستش نکرده ام هنوز.

<https://cloudant.com/#home-pricing>

نویسنده: masi  
تاریخ: ۱:۵۶ ۱۳۹۲/۰۲/۱۹

سلام ، واقعا مطالب خوبی بود هر جا رو گشتم کاملتر و جامع تر از همه بودید ، موضوع پروژه ی من روی این موضوعه ، ای کاش میشد در مورد موضوع زیر صحبت کنید. key value store

نویسنده: جواد زبیدی  
تاریخ: ۳:۳۳ ۱۳۹۲/۰۵/۱۶

سلام تشکر از مطلب بسیار مفیدتون .

می خواستم بدونم که کدام یک از روش ها بیشتر امتحان خودش رو توی داده های زیاد پس داده . و بشه راحت تر باهاش کار کرد .  
روش

Document store

Key value

روش هایی دیگری رو هم توی سایت دیدم اگر امکان داره مزایا و معایب هر کدوم رو توضیح دهید ممنون.

نویسنده: دادخواه  
تاریخ: ۱۲:۱۰ ۱۳۹۲/۰۶/۰۷

سلام

تشکر از مطالب خوبتون

اما چند تا سوال دارم.

1- از این سه تا پایگاه داده که در اخر نوشتید فکر کنم فقط MongoDB مجانی باشه. درسته؟

2- آیا دستورات در همه این پایگاه داده ها به همین صورت است؟

3- آیا همه سرورها و هاست ها از این پایگاه داده ها مانند MS SQL پشتیبانی می کنند و یا سرورهای خاص را باید پیدا کرد؟  
تشکر

نویسنده: محسن خان

تاریخ: ۱۳۹۲/۰۶/۰۷ ۱۲:۲۵

اگر مطالب [مقدماتی تر رو](#) مطالعه می کردید، می دید که اصلا هدف از بانک اطلاعاتی NoSQL این نیست که باهش سایت معمولی درست کنند اون هم روی سرور اجاره ای با 100 مگ فضا. هدفش توزیع شده بودن در سرورهایی متعدد و یا با پراکندگی جغرافیایی بالا است.

نتیجه گیری؟ ابزار زده نباشید. اول مفاهیم رو مطالعه کنید. اول تئوری کار مهمه.

نویسنده: saremi

تاریخ: ۱۳۹۲/۱۱/۲۴ ۱۶:۴۴

با سلام؛ میخواستم در مورد UNQL، CQL، HQL، map reduce و... بپرسم. توی همون سایتی که لینکش رو دادین اینها جزو انواع query method هستند. من دقیق نمیفهمم الان ما دستورات مشابه sql رو معادلش رو با java script نوشتیم. در مورد تفاوت اینها و استفاده شون اگر میشه کمی توضیح بدین لطفا. دقیقا توی انواع مختلف پایگاه داده با چه زبانی کوئری نویسی می شه؟ با تشکر

نویسنده: محسن خان

تاریخ: ۱۳۹۲/۱۱/۲۴ ۱۶:۵۲

در مورد تفاوت اینها در مطلب [مروری بر مفاهیم مقدماتی NoSQL](#) بیشتر توضیح داده شده. برچسب [NoSQL](#) را بهتر است دنبال کنید.

نویسنده: وحید نصیری

تاریخ: ۱۳۹۲/۱۱/۲۴ ۱۷:۴۹

در مورد MongoDB یک کتابچه ی فارسی 90 صفحه ای [موجود است](#) .

نویسنده: salam

تاریخ: ۱۳۹۳/۰۵/۰۹ ۱۱:۱۶

سلام

در قسمت دوم این مطلب اومده که "در حالت کلی پایگاه های داده NoSQL به ۴ دسته تقسیم می شوند " دسته 3 و 4 را توضیح نمی دین؟

نویسنده: وحید نصیری

تاریخ: ۱۳۹۳/۰۵/۰۹ ۱۱:۲۶

برای دنبال کردن مطالب هم خانواده در این سایت، در ذیل هر مطلب یک سری گروه یا برچسب تعریف شده اند. برای مثال اگر برچسب [NoSQL](#) را دنبال کنید، در مطالب دیگری پاسخ خود را خواهید یافت.

## مروری بر MongoDB

MongoDB یک پایگاه داده سند-گرا (Document-Oriented) و مستقل از سکو است که کارایی بالا، دسترسی پذیری بالا و مقیاس پذیری آسانی را فراهم می‌کند. MongoDB بر اساس مفهوم مجموعه (Collection) و سند (Document) کار می‌کند.

### پایگاه داده

پایگاه داده یک نگهدارنده‌ی فیزیکی برای مجموعه‌ها است. هر پایگاه داده مجموعه‌ای از فایل‌های خود را روی فایل سیستم دارد. یک سرور MongoDB معمولاً چندین پایگاه داده دارد.

### مجموعه

مجموعه یک گروه از سندهای MongoDB است. مجموعه معادل جدول در پایگاه داده‌های رابطه‌ای (Relational Database) است. یک مجموعه داخل یک پایگاه داده وجود دارد. مجموعه‌ها به شمای (Schema) تاکید ندارند. سندهای داخل مجموعه می‌توانند فیلدهای مختلفی داشته باشند. معمولاً همه‌ی سندهای داخل یک مجموعه، شبیه یا مربوط به یک هدف هستند.

### سند

یک سند مجموعه‌ای از جفت‌های کلید-مقدار (Key-Value Pairs) است. سند، شمای پویا دارد؛ یعنی سندها در مجموعه‌های مشابه نیازی به ساختار یا فیلدهای مشابه ندارند و فیلدهای مشترک در سند ممکن است نوع داده‌های متفاوتی را نگهداری کنند. جدول زیر مقایسه اصطلاحات پایگاه داده‌های رابطه‌ای و MongoDB را نمایش می‌دهد:

MongoDB	پایگاه داده رابطه‌ای
پایگاه داده	پایگاه داده
مجموعه	جدول
سند	سطر
فیلد	ستون
سند توکار	ملحق کردن (Join)
کلید اصلی (کلید پیش فرض id_ توسط MongoDB فراهم شده)	کلید اصلی
پایگاه داده نسخه سرور و کلاینت	
Mongod	Mysqld/Oracle
mongo	mysql/sqlplus

### مثالی از سند

در جدول زیر ساختار سند یک وبلاگ آمده است که جفت‌های کلید-مقدار بسادگی با کاما از هم جدا شده اند.

```
{
  _id: ObjectId(7df78ad8902c)
  title: 'MongoDB Overview',
  description: 'MongoDB is no sql database',
  by: 'tutorials point',
  url: 'http://www.tutorialspoint.com',
  tags: ['mongodb', 'database', 'NoSQL'],
  likes: 100,
  comments: [
    {
      user: 'user1',
      message: 'My first comment',
      dateCreated: new Date(2011,1,20,2,15),
      like: 0
    },
    {
      user: 'user2',
      message: 'My second comments',
      dateCreated: new Date(2011,1,25,7,45),
      like: 5
    }
  ]
}
```

`_id` یک 12بایتی هگزادسیمال است که یکتایی هر سند را اطمینان می‌دهد. شما می‌توانید یک `_id` را هنگام درج سند بسازید. اگر اینکار را نکنید، MongoDB یک شناسه‌ی یکتا را برای هر سند تهیه می‌کند. از این 12بایت، 4بایت اول آن مربوط به برچسب زمان جاری است، 3بایت بعدی برای شناسه‌ی ماشین، 2بایت بعدی برای شناسه‌ی پروسه MongoDB سرور و 3بایت باقیمانده یک مقدار صعودی ساده است.



## مزایای MongoDB

هر پایگاه داده رابطه‌ای، یک طراحی شمای معمول داشته و تعدادی جدول و رابطه‌های بین این جدول‌ها را نشان می‌دهد؛ درحالی‌که مفهوم رابطه در MongoDB وجود ندارد.

### مزایای MongoDB نسبت به پایگاه داده رابطه‌ای

- بدون شمای (Schema less): در واقع MongoDB یک پایگاه داده سند-گراست که یک مجموعه از سندهای متفاوت را نگهداری می‌کند. تعداد فیلدها، محتوا و اندازه یک سند می‌تواند متفاوت از بقیه سندها باشد.
- ساختار یک شیء واحد واضح است
- عدم وجود Joinهای پیچیده
- قابلیت کوئری عمیق. MongoDB با استفاده از زبان کوئری سند-گرا از کوئری‌های پویا بر روی سندها پشتیبانی می‌کند و تقریباً مانند SQL قدرتمند است.
- میزان سازی (Tuning)
- سهولت مقیاس پذیری: MongoDB آسان است برای مقیاس پذیری
- از حافظه داخلی برای مرتب سازی مجموعه کاری استفاده می‌کند؛ جهت امکان دسترسی سریع به داده

### چرا باید از MongoDB استفاده کنیم

- ذخیره سازی سند-گرا: داده بصورت سندهایی از JSON ذخیره می‌شوند
- ایندکس گذاری روی هر خاصیت
- رونوشت (Replication) و دسترس پذیری بالا
- Sharding خودکار
- کوئری‌های غنی
- بروز رسانی‌های درجا
- پشتیبانی حرفه ای توسط MongoDB

### کجا باید از MongoDB استفاده کنیم

- داده‌های عظیم (Big Data)
- سیستم‌های مدیریت محتوا و تحویل
- زیرساخت‌های اجتماعی و موبایل
- مدیریت داده کاربر

عنوان:	MongoDB #3
نویسنده:	جوادی
تاریخ:	۹:۴۰ ۱۳۹۳/۱۰/۳۰
آدرس:	<a href="http://www.dotnettips.info">www.dotnettips.info</a>
گروه‌ها:	NoSQL, Database, mongodb

## محیط MongoDB

### نصب MongoDB در ویندوز

برای نصب MongoDB در ویندوز، اول باید آخرین نسخه MongoDB را از آدرس <http://www.mongodb.org/downloads> دریافت کنید. مطمئن شوید که نسخه‌ی صحیحی از MongoDB را نسبت به معماری ویندوزتان دریافت کرده‌اید. برای پیدا کردن معماری ویندوز، پنجره‌ی Command Prompt را باز کنید و دستور زیر را اجرا کنید:

```
C:\>wmic os get osarchitecture
OSArchitecture
64-bit
C:\>
```

نسخه‌های 32 بیتی MongoDB فقط پایگاه داده‌های کوچکتر از 2 گیگابایت را پشتیبانی می‌کنند و صرفاً برای تست و ارزیابی مناسب هستند. اکنون فایل دریافتی را نصب کنید. MongoDB یک پوشه داده، برای ذخیره فایل‌هایش نیاز دارد. مسیر پیش فرض پوشه داده c:\data\db است؛ بنابراین نیاز دارید این پوشه را بسازید. شما می‌توانید یک مسیر دیگر را نیز برای مسیر داده تنظیم کنید. برای انجام این کار، Command Prompt را در پوشه bin (در مسیر نصب شده MongoDB) باز کنید و دستور زیر را اجرا کنید: (فرض کنید MongoDB در مسیر D:\set up\mongodb نصب شده است)

```
D:\set up\mongodb\bin>mongod.exe --dbpath "d:\set up\mongodb\data"
```

بعد از اجرای دستور، پیام "waiting for connections" در کنسول نمایش داده می‌شود که نشان دهنده‌ی این است که پروسه Mongod.exe با موفقیت اجرا شده است. حالا برای اجرای MongoDB یک Command Prompt دیگر نیاز دارید تا دستور زیر را اجرا کنید:

```
D:\set up\mongodb\bin>mongo.exe
MongoDB shell version: 2.6.6
connecting to: test

>db.test.save( { a: 1 } )
>db.test.find()
{ "_id" : ObjectId(5879b0f65a56a454), "a" : 1 }
>
```

این دستور نشان خواهد داد که MongoDB نصب و با موفقیت اجرا شده است. برای اجرای MongoDB در دفعات بعدی نیز همین 2 مرحله را تکرار کنید (تعیین مسیر پوشه داده و اجرای Mongo.exe در یک Command Prompt دیگر).

### نصب MongoDB در اوبونتو

دستور زیر را برای وارد کردن کلید عمومی GPG MongoDB در ترمینال اجرا کنید:

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10
```

فایل `etc/apt/sources.list.d/mongodb.list/` را با دستور زیر بسازید:

```
echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen' | sudo tee /etc/apt/sources.list.d/mongodb.list
```

اکنون دستور زیر را برای بروز رسانی مخازن پکیج‌ها اجرا کنید:

```
sudo apt-get update
```

حالا MongoDB را با استفاده از دستور زیر نصب کنید:

```
apt-get install mongodb-10gen=2.2.3
```

در دستور نصب فوق، به نسخه‌ی 2.2.3 از MongoDB انتشار شده است. همیشه مطمئن شوید که آخرین نسخه را نصب کرده اید. اکنون MongoDB با موفقیت نصب شده است.

راه اندازی MongoDB

```
sudo service mongodb start
```

متوقف کردن MongoDB

```
sudo service mongodb stop
```

راه اندازی مجدد MongoDB

```
sudo service mongodb restart
```

برای استفاده از MongoDB از دستور زیر استفاده کنید:

Mongo

این دستور شما را به نمونه‌ی در حال اجرای Mongod متصل خواهد کرد.

راهنمای MongoDB

برای دریافت لیست دستورات، `db.help()` را در نسخه کلاینت MongoDB تایپ کنید. این دستور، لیست دستورات را مانند تصویر زیر به شما می‌دهد:

```

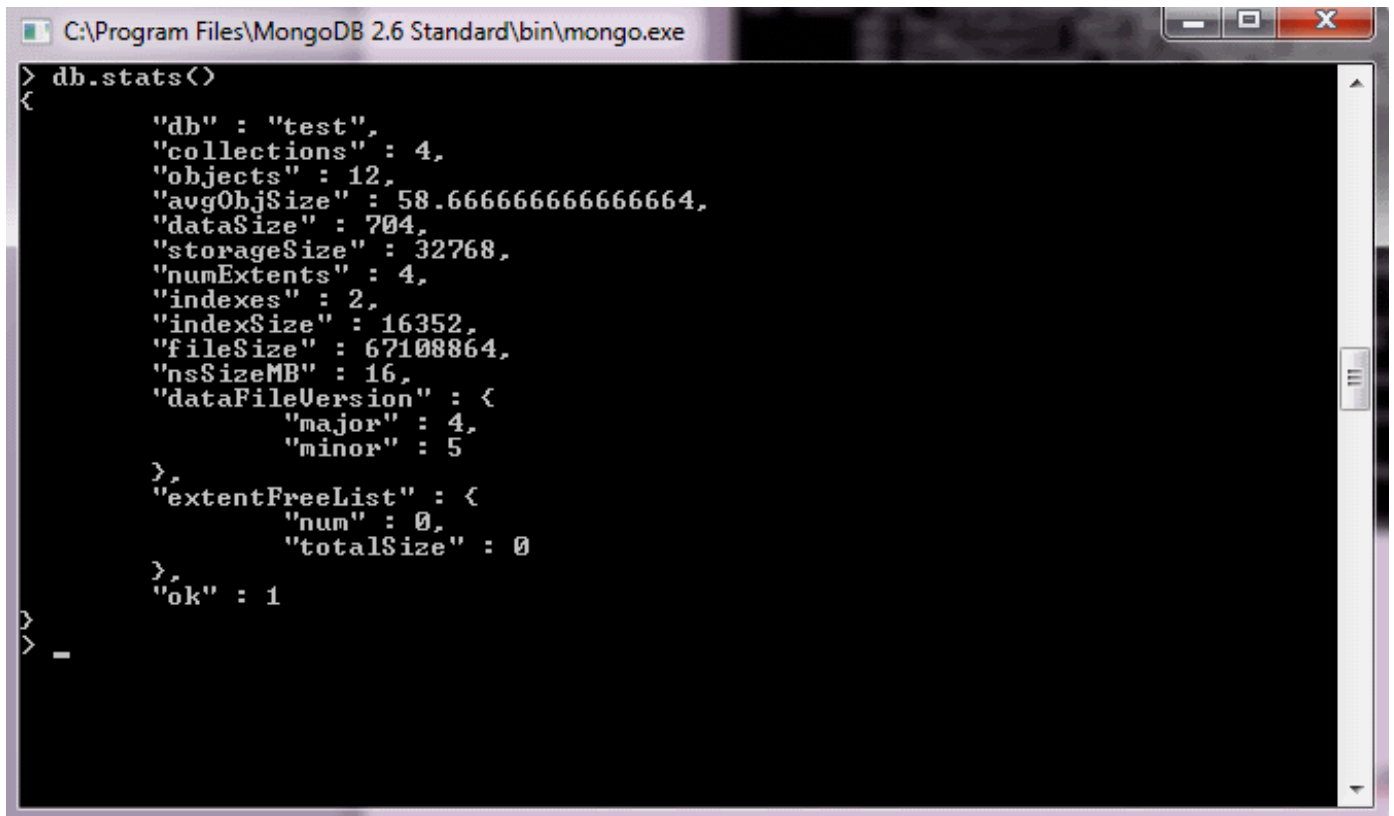
C:\Program Files\MongoDB 2.6 Standard\bin\mongo.exe

db.auth(username, password)
db.cloneDatabase(fromhost)
db.commandHelp(name) returns the help for the command
db.copyDatabase(fromdb, todb, fromhost)
db.createCollection(name, { size : ..., capped : ..., max : ... } )
db.createUser(userDocument)
db.currentOp() displays currently executing operations in the db
db.dropDatabase()
db.eval(func, args) run code server-side
db.fsyncLock() flush data to disk and lock server for backups
db.fsyncUnlock() unlocks server following a db.fsyncLock()
db.getCollection(cname) same as db['cname'] or db.cname
db.getCollectionNames()
db.getLastErrorMessage() - just returns the err msg string
db.getStatusObj() - return full status object
db.getMongo() get the server connection object
db.getMongo().setSlaveOk() allow queries on a replication slave server
db.getName()
db.getPrevError()
db.getProfilingLevel() - deprecated
db.getProfilingStatus() - returns if profiling is on and slow threshold
db.getReplicationInfo()
db.getSiblingDB(name) get the db at the same server as this one
db.getWriteConcern() - returns the write concern used for any operations
on this db, inherited from server object if set
db.hostInfo() get details about the server's host
db.isMaster() check replica primary status
db.killOp(opid) kills the current operation in the db
db.listCommands() lists all the db commands
db.loadServerScripts() loads all the scripts in db.system.js
db.logout()
db.printCollectionStats()
db.printReplicationInfo()
db.printShardingStatus()
db.printSlaveReplicationInfo()
db.dropUser(username)
db.repairDatabase()
db.resetError()
db.runCommand(cmdObj) run a database command. if cmdObj is a string, tu
rns it into { cmdObj : 1 }
db.serverStatus()
db.setProfilingLevel(level,<slowms>) 0=off 1=slow 2=all
db.setWriteConcern( <write concern doc> ) - sets the write concern for w
rites to the db
db.unsetWriteConcern( <write concern doc> ) - unsets the write concern f
or writes to the db
db.setVerboseShell(flag) display extra information in shell output
db.shutdownServer()
db.stats()
db.version() current version of the server
>

```

## آمار و ارقام در MongoDB

برای گرفتن آمار و ارقام از MongoDB سرور، دستور `db.stats()` را در نسخه کلاینت MongoDB تایپ کنید. این دستور نام پایگاه داده، تعداد مجموعه‌ها و سندهای موجود در پایگاه داده را نمایش می‌دهد:



The screenshot shows a Windows command prompt window titled "C:\Program Files\MongoDB 2.6 Standard\bin\mongo.exe". The prompt is at the MongoDB shell, where the command `db.stats()` has been entered. The output is a JSON document providing statistics for the 'test' database. The statistics include the number of collections (4), objects (12), average object size (58.666666666666664), data size (704), storage size (32768), number of extents (4), number of indexes (2), index size (16352), file size (67108864), namespace size in MB (16), data file version (major: 4, minor: 5), extent free list (num: 0, total size: 0), and a status 'ok' of 1.

```
> db.stats()
{
  "db" : "test",
  "collections" : 4,
  "objects" : 12,
  "avgObjSize" : 58.666666666666664,
  "dataSize" : 704,
  "storageSize" : 32768,
  "numExtents" : 4,
  "indexes" : 2,
  "indexSize" : 16352,
  "fileSize" : 67108864,
  "nsSizeMB" : 16,
  "dataFileVersion" : {
    "major" : 4,
    "minor" : 5
  },
  "extentFreeList" : {
    "num" : 0,
    "totalSize" : 0
  },
  "ok" : 1
}
```

### مدل کردن داده در MongoDB

داده در MongoDB شمای منطقی دارد. سندها در یک مجموعه به تعدادی از فیلدها با ساختاری شبیه به هم نیازی ندارند و فیلدهای مشترک در یک سند مجموعه ممکن است نوع‌های داده‌ی متفاوتی را نگهداری کنند.

#### برخی ملاحظات هنگام طراحی شمای در MongoDB

شمای خود را بر اساس نیازمندی‌های کاربر طراحی کنید.

آبجکت‌هایی را که از آنها باهم استفاده می‌کنید، داخل یک سند ترکیب کنید؛ در غیر اینصورت آنها را جدا کنید (اما مطمئن شوید که نباید نیازی به استفاده از Join باشد).

داده را بصورت محدود تکثیر کنید؛ چون فضای دیسک ارزاتر است از محاسبه زمان.

عمل Join را هنگام ذخیره کردن انجام دهید، نه موقع خواندن.

شمای خود را برای بیشترین موارد استفاده بهینه کنید.

تجمع‌های پیچیده (Complex Aggregation) را در شمای انجام دهید.

### مثال

فرض کنید یک کاربر نیاز به طراحی یک پایگاه داده برای وب سایت خود دارد. تفاوت‌های طراحی شمای بین MongoDB و RDBMS را در ادامه ملاحظه خواهید کرد. وب سایت نیازمندی‌های زیر را دارد:

هر پست یک عنوان یکتا، توضیحات و آدرس اینترنتی دارد.

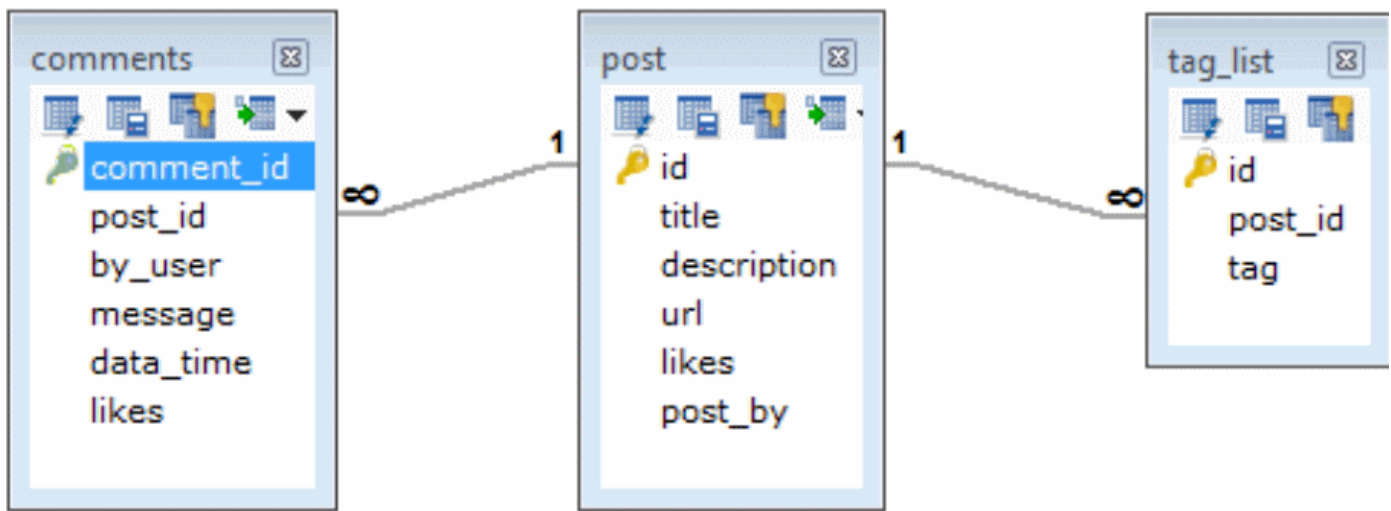
هر پست می‌تواند یک یا چند برچسب داشته باشد.

هر پست نام نویسنده و تعداد Likeها را دارد.

هر پست تعدادی نظر معین را توسط کاربران همراه با نامشان، پیام، تاریخ ثبت و تعداد Likeها، دارد.

در هر پست صفر یا چند نظر وجود دارد.

در طراحی شمای توسط RDBMS برای نیازمندی‌های فوق، حداقل سه جدول نیاز است.



درحالیکه در طراحی شمای توسط MongoDB یک مجموعه از پست را با ساختار زیر خواهیم داشت:

```

{
  _id: POST_ID,
  title: TITLE_OF_POST,
  description: POST_DESCRIPTION,
  by: POST_BY,
  url: URL_OF_POST,
  tags: [TAG1, TAG2, TAG3],
  likes: TOTAL_LIKES,
  comments: [
    {
      user: 'COMMENT_BY',
      message: TEXT,
      dateCreated: DATE_TIME,
      like: LIKES
    },
    {
      user: 'COMMENT_BY',
      message: TEXT,
      dateCreated: DATE_TIME,
      like: LIKES
    }
  ]
}

```

بنابراین هنگام نمایش داده در RDBMS نیاز به تهیه Join بین سه جدول دارید؛ اما در MongoDB داده‌ها از یک مجموعه نمایش داده خواهند شد.

عنوان:	MongoDB #5
نویسنده:	جوادی
تاریخ:	۰۵:۱۳۹۳/۱۱/۰۲
آدرس:	<a href="http://www.dotnettips.info">www.dotnettips.info</a>
گروه‌ها:	NoSQL, Database, mongodb

## ساخت و حذف پایگاه داده در MongoDB

### دستور Use

در MongoDB از دستور `use DATABASE_NAME` برای ساخت پایگاه داده استفاده می‌شود. این دستور یک پایگاه داده جدید را ایجاد می‌کند و اگر از قبل موجود باشد، یک پایگاه داده موجود را برمی‌گرداند.

### گرامر (Syntax):

گرامر پایه عبارت `use DATABASE` به شکل زیر است:

```
use DATABASE_NAME
```

### مثال:

اگر می‌خواهید یک پایگاه داده را با نام `<mydb>` بسازید، عبارت `use DATABASE` به شکل زیر در می‌آید:

```
>use mydb
switched to db mydb
```

از دستور `db`، برای بررسی انتخاب صحیح نام پایگاه داده استفاده کنید:

```
>db
mydb
```

اگر می‌خواهید لیست پایگاه‌های داده‌ی خود را چک کنید، از دستور `show dbs` استفاده کنید:

```
>show dbs
local 0.78125GB
test 0.23012GB
```

پایگاه داده ساخته شده توسط شما (`mydb`) ممکن است در لیست نمایش داده نشود. برای نمایش پایگاه داده، نیاز به درج حداقل یک سند داخل آن، دارید.

```
>db.movie.insert({"name":"tutorials point"})
>show dbs
local 0.78125GB
mydb 0.23012GB
test 0.23012GB
```

در MongoDB، پایگاه داده پیش فرض تست است. اگر هیچ پایگاه داده‌ای نساخته‌اید، مجموعه در پایگاه داده `test` ذخیره می‌شود.

## حذف یک پایگاه داده در MongoDB

### متد `dropDatabase()`

از دستور `db.dropDatabase()` برای حذف یک پایگاه داده موجود استفاده می‌شود.

### گرامر:

گرامر پایه دستور `dropDatabase()` به شکل زیر است:

```
db.dropDatabase()
```



این دستور پایگاه داده‌ی انتخاب شده را حذف می‌کند. اگر هیچ پایگاه داده‌ای را انتخاب نکرده باشید، بصورت پیش فرض پایگاه داده test حذف خواهد شد.

#### مثال:

اول، لیست پایگاه‌های داده‌ی موجود را با استفاده از دستور show dbs مشاهده کنید:

```
>show dbs
local 0.78125GB
mydb 0.23012GB
test 0.23012GB
>
```

اگر می‌خواهید پایگاه داده <mydb> را حذف کنید، از دستور dropDatabase به شکل زیر استفاده کنید:

```
>use mydb
switched to db mydb
>db.dropDatabase()
>{ "dropped" : "mydb", "ok" : 1 }
>
```

اکنون لیست پایگاه‌های داده را بررسی کنید:

```
>show dbs
local 0.78125GB
test 0.23012GB
>
```

## ساخت مجموعه در MongoDB

### متد CreateCollection ()

دستور db.createCollection(name, options) در MongoDB برای ساخت مجموعه بکار برده می‌شود.

### گرامر:

گرامر پایه دستور createCollection() به شکل زیر است:

db.createCollection(name, options)

در این دستور، پارامتر name نام مجموعه‌ای است که باید ساخته شود. پارامتر Option یک سند است و برای تعیین پیکربندی مجموعه استفاده می‌شود.

پارامتر	نوع	توضیحات
name	رشته	نام مجموعه‌ای است که باید ساخته شود
Option	سند	(اختیاری) تعیین اختیارات برای اندازه حافظه و ایندکس گذاری

پارامتر Option اختیاری است. در جدول زیر لیست اختیاراتی را که می‌توانید استفاده کنید آمده است:

فیلد	نوع	توضیحات
capped	Boolean	(اختیاری) اگر مقدار آن true باشد یک مجموعه‌ی پوشیده (capped) در اختیار می‌گذارد. مجموعه‌ی پوشیده یک مجموعه با اندازه ثابت است که وقتی به حداکثر اندازه خود برسد، داده‌های جدید را بصورت اتوماتیک جایگزین قدیمی‌ترین داده‌ها می‌کند. اگر این پارامتر را true تنظیم کرده باشید، باید پارامتر size را هم مقداردهی کنید.
AutoIndexID	Boolean	(اختیاری) اگر true باشد، بصورت اتوماتیک روی فیلد id_ ایندکس می‌سازد. مقدار پیش فرض این پارامتر false است.
size	number	(اختیاری) تعیین کننده‌ی حداکثر اندازه به بایت برای مجموعه پوشیده. اگر پارامتر capped برابر true باشد آنگاه نیاز دارید این پارامتر را نیز مقداردهی کنید.
max	number	(اختیاری) تعیین کننده حداکثر تعداد سندهای مجاز در یک مجموعه پوشیده

هنگام درج یک سند، MongoDB ابتدا فیلد capped و سپس فیلد max را بررسی می‌کند.

### مثال:

گرامر پایه متد `createCollection()` بدون اختیارات به شکل زیر است:

```
>use test
switched to db test
>db.createCollection("mycollection")
{ "ok" : 1 }
>
```

با استفاده از دستور `show collection` می‌توانید مجموعه ساخته شده را بررسی کنید:

```
>show collections
mycollection
system.indexes
```

مثال زیر گرامر متد `createCollection()`، با اختیارات مهم‌تر را نمایش می‌دهد:

```
>db.createCollection("mycol", { capped : true, autoIndexID : true, size : 6142800, max : 10000 } )
{ "ok" : 1 }
>
```

در MongoDB، نیازی به ساخت مجموعه ندارید. وقتی یک سند را درج کنید، MongoDB بصورت اتوماتیک مجموعه را می‌سازد.

```
>db.tutorialspoint.insert({"name" : "tutorialspoint"})
>show collections
mycol
mycollection
system.indexes
tutorialspoint
>
```

## حذف مجموعه‌ها در MongoDB

### متد `drop()`

دستور `db.collection.drop()` برای حذف یک مجموعه از پایگاه داده استفاده می‌شود.

### گرامر:

گرامر پایه دستور `drop()` به شکل زیر است:

```
db.COLLECTION_NAME.drop()
```

### مثال:

ابتدا همه مجموعه‌های موجود در پایگاه داده `mydb` را بررسی کنید:

```
>use mydb
switched to db mydb
```

```
>show collections
mycol
mycollection
system.indexes
tutorialspoint
>
```

اکنون مجموعه با نام mycollection را حذف کنید:

```
>db.mycollection.drop()
true
>
```

دوباره لیست مجموعه‌های داخل پایگاه داده را بررسی کنید:

```
>show collections
mycol
system.indexes
tutorialspoint
>
```

اگر مجموعه انتخاب شده به موفقیت حذف شود، متد drop() مقدار true در غیر این صورت مقدار false را برمی‌گرداند.

عنوان:	MongoDB #7
نویسنده:	جوادی
تاریخ:	۱۳:۵۰ ۱۳۹۳/۱۱/۰۵
آدرس:	<a href="http://www.dotnettips.info">www.dotnettips.info</a>
گروه‌ها:	NoSQL, Database, mongodb

## انواع داده‌ها در MongoDB

MongoDB انواع داده‌هایی را که در زیر لیست شده اند، پشتیبانی می‌کند:

**String** : این نوع پرکاربردترین نوع داده برای ذخیره اطلاعات است. رشته در MongoDB باید بصورت یونیکد (utf-8) معتبر باشد.

**Integer** : این نوع برای ذخیره کردن یک مقدار عددی استفاده می‌شود. Integer بسته به نوع سرور می‌تواند 32 یا 64 بیت باشد.

**Boolean** : این نوع برای ذخیره کردن یک مقدار بولی (true/false) استفاده می‌شود.

**Double** : این نوع برای مقادیر با ممیز شناور استفاده می‌شود.

**Min/Max** : این نوع برای مقایسه یک مقدار با کمترین یا بیشترین عناصر BSON استفاده می‌شود.

**Array** : این نوع برای ذخیره آرایه‌ها یا لیست یا چندین مقدار در یک کلید استفاده می‌شود.

**Timestamp** : این نوع می‌تواند برای ضبط زمان تغییرات (مثلا وقتی یک سند درج می‌شود یا تغییر می‌کند) مفید باشد.

**Object** : این نوع برای سندهای توکار استفاده می‌شود.

**Null** : این نوع برای ذخیره مقدار تهی (Null) استفاده می‌شود.

**Symbol** : این نوع بطور یکسان برای ذخیره رشته استفاده می‌شود، اما عموماً برای زبان‌هایی که از یک نوع نماد (Symbol) مشخص استفاده می‌کنند تعبیه شده است.

**Date** : این نوع برای ذخیره تاریخ یا زمان جاری به فرمت زمان در یونیکس (UNIX) استفاده می‌شود. با ساخت یک شی از نوع Date و ارسال روز، ماه و سال به آن می‌توانید تاریخ مشخص خود را داشته باشید.

**Object ID** : ای نوع برای ذخیره سازی شناسه سند استفاده می‌شود.

**Binary Data** : این نوع برای ذخیره سازی داده باینری استفاده می‌شود.

**Code** : این نوع برای ذخیره سازی کد جاوا اسکریپت داخل سند استفاده می‌شود.

**Regular Expression** : این نوع برای ذخیره سازی عبارت باقاعده استفاده می‌شود.

## درج سند در MongoDB متد Insert ()

برای درج داده در یک مجموعه نیاز است تا از متد insert() یا save() در MongoDB استفاده کنید.

### گرامر

گرامر پایه دستور insert() به شکل زیر است:

```
>db.COLLECTION_NAME.insert(document)
```

```
>db.mycol.insert({
  _id: ObjectId(7df78ad8902c),
  title: 'MongoDB Overview',
  description: 'MongoDB is no sql database',
  by: 'tutorials point',
  url: 'http://www.tutorialspoint.com',
  tags: ['mongodb', 'database', 'NoSQL'],
  likes: 100
})
```

در اینجا mycol نام مجموعه‌ی ما است. اگر مجموعه از قبل در پایگاه داده موجود نباشد، MongoDB این مجموعه را خواهد ساخت؛ سپس سند را داخل آن درج خواهد کرد.

در درج این سند، اگر پارامتر id\_ را مشخص نکنید، MongoDB یک ObjectId یکتا را به این سند اختصاص می‌دهد. برای درج چند سند در یک کوئری می‌توانید آرایه‌ای از سندها را به دستور insert() پاس دهید.

#### مثال

```
>db.post.insert([
{
  title: 'MongoDB Overview',
  description: 'MongoDB is no sql database',
  by: 'tutorials point',
  url: 'http://www.tutorialspoint.com',
  tags: ['mongodb', 'database', 'NoSQL'],
  likes: 100
},
{
  title: 'NoSQL Database',
  description: 'NoSQL database doesn't have tables',
  by: 'tutorials point',
  url: 'http://www.tutorialspoint.com',
  tags: ['mongodb', 'database', 'NoSQL'],
  likes: 20,
  comments: [
    {
      user: 'user1',
      message: 'My first comment',
      dateCreated: new Date(2013,11,10,2,35),
      like: 0
    }
  ]
}
])
```

برای درج سند می‌توانید از db.post.save(document) نیز استفاده کنید. اگر پارامتر id\_ را در سند مشخص نکنید، متد save() مانند متد insert() عمل خواهد کرد. اگر پارامتر id\_ را مشخص کنید، مقدار آن در سند، جایگزین مقدار پیش فرض id\_ می‌شود.

## اجرای کوئری در سند MongoDB متد find ()

برای اجرای یک کوئری نیاز دارید تا از متد find() در MongoDB استفاده کنید.

### گرامر:

گرامر پایه برای این متد به شکل زیر است:

```
>db.COLLECTION_NAME.find()
```

متد find() تمام سندها را در یک حالت بدون ساختار نمایش می‌دهد.

### متد Pretty ()

برای نمایش نتیجه، بصورت فرمت دهی شده و ساخت یافته می‌توانید از متد pretty() استفاده کنید.

### گرامر:

```
>db.mycol.find().pretty()
```

### مثال:

```
>db.mycol.find().pretty()
{
  "_id": ObjectId("7df78ad8902c"),
  "title": "MongoDB Overview",
  "description": "MongoDB is no sql database",
  "by": "tutorials point",
  "url": "http://www.tutorialspoint.com",
  "tags": ["mongodb", "database", "NoSQL"],
  "likes": "100"
}
```

جدای از متد find()، متد findOne() نیز وجود دارد که فقط یک سند را برمی گرداند.

## معادل‌های عبارت Where در MongoDB

برای اجرای کوئری‌های بر اساس چندین شرط بر روی سندها می‌توانید از عملگرهای زیر استفاده کنید:

عملگر	گرامر	مثال	مشابه در پایگاه داده رابطه ای
Equality	{<key>:<value>}	()db.mycol.find({"by":"tutorials point").pretty	' where by = 'tutorials point
Less Than	{<key>:{\$lt:<value>}}	()db.mycol.find({"likes":{\$lt:50}}).pretty	where likes < 50
Less Than Equals	{<key>:{\$lte:<value>}}	()db.mycol.find({"likes":{\$lte:50}}).pretty	where likes <= 50
Greater Than	{<key>:{\$gt:<value>}}	()db.mycol.find({"likes":{\$gt:50}}).pretty	where likes > 50
Greater Than Equals	{<key>:{\$gte:<value>}}	()db.mycol.find({"likes":{\$gte:50}}).pretty	where likes >= 50

عملگر	گرامر	مثال	مشابه در پایگاه داده رابطه ای
		<code>gte:50}}).pretty</code>	
Not Equals	<code>{&lt;key&gt;:{\$ne:&lt;value&gt;}}</code>	<code>()db.mycol.find({"likes":{\$ne:50}}).pretty</code>	<code>where likes != 50</code>

### عبارت And در MongoDB گرامر:

اگر چندین کلید را به متد `find()` پاس دهید و آنها را با کاما (,) از هم جدا کنید، MongoDB با آنها مانند عبارت And رفتار می‌کند. گرامر پایه عبارت AND در جدول زیر نشان داده شده است:

```
>db.mycol.find({key1:value1, key2:value2}).pretty()
```

### مثال:

در ادامه یک مثال آمده است که همه‌ی دوره‌های آموزشی که توسط 'tutorials point' ارائه شده‌اند و عنوان آنها 'MongoDB Overview' است را نشان می‌دهد:

```
>db.mycol.find({"by":"tutorials point","title": "MongoDB Overview"}).pretty()
{
  "_id": ObjectId(7df78ad8902c),
  "title": "MongoDB Overview",
  "description": "MongoDB is no sql database",
  "by": "tutorials point",
  "url": "http://www.tutorialspoint.com",
  "tags": ["mongodb", "database", "NoSQL"],
  "likes": "100"
}
```

برای مثال بالا، معادل عبارت `Where` آن `'where by='tutorials point' AND title='MongoDB Overview'` خواهد بود. شما می‌توانید چندین جفت کلید-مقدار به عبارت `find` پاس دهید.

### عبارت OR در MongoDB گرامر:

برای اجرای کوئری‌های مبتنی بر عبارت OR روی سند نیاز دارید تا از کلمه‌ی کلیدی `$or` استفاده کنید. گرامر پایه عبارت OR در زیر نشان داده شده است:

```
>db.mycol.find(
{
  $or: [
    {key1: value1}, {key2:value2}
  ]
}).pretty()
```

### مثال

در ادامه یک مثال آمده است که همه‌ی دوره‌های آموزشی را که توسط 'tutorials point' ارائه شده‌اند یا عنوان آنها 'MongoDB Overview' است، نشان می‌دهد:

```
>db.mycol.find({$or:[{"by":"tutorials point"},"title": "MongoDB Overview"}]}).pretty()
{
  "_id": ObjectId(7df78ad8902c),
  "title": "MongoDB Overview",
  "description": "MongoDB is no sql database",
  "by": "tutorials point",
  "url": "http://www.tutorialspoint.com",
  "tags": ["mongodb", "database", "NoSQL"],
  "likes": "100"
}
```



در ادامه یک مثال آمده است که سندهایی را که مقدار فیلد likes آنها بیشتر از 100 و عنوان آنها برابر 'MongoDB Overview' یا توسط 'tutorials point' ارائه شده‌اند، نشان خواهد داد. معادل عبارت Where آن برابر 'where likes>100 AND (by = 'tutorials point' OR title= 'MongoDB Overview')' است.

```
>db.mycol.find("likes": {$gt:100}, $or: [{"by": "tutorials point"}, {"title": "MongoDB Overview"}]
}).pretty()
{
  "_id": ObjectId("7df78ad8902c"),
  "title": "MongoDB Overview",
  "description": "MongoDB is no sql database",
  "by": "tutorials point",
  "url": "http://www.tutorialspoint.com",
  "tags": ["mongodb", "database", "NoSQL"],
  "likes": "100"
}
```

عنوان:	MongoDB #9
نویسنده:	جوادی
تاریخ:	۱۶:۲۵ ۱۳۹۳/۱۱/۱۱
آدرس:	<a href="http://www.dotnettips.info">www.dotnettips.info</a>
گروه‌ها:	NoSQL, Database, mongodb

### به‌روز رسانی سند در MongoDB

متدهای update() و save() هر دو برای به‌روز رسانی یک سند داخل یک مجموعه، استفاده می‌شوند. متد update() مقادیر موجود در سند را به‌روز رسانی می‌کند؛ درحالی‌که متد save() سند ارسالی به این متد را جایگزین سندی موجود در مجموعه می‌کند.

#### متد Update()

update() مقادیر موجود در سند را به‌روز رسانی می‌کند. گرامر:

گرامر پایه متد update() به شکل زیر است:

```
>db.COLLECTION_NAME.update(SELECTIOIN_CRITERIA, UPDATED_DATA)
```

#### مثال:

مجموعه mycol را ملاحظه کنید که داده‌های زیر را دارد:

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

مثال زیر عنوان 'New MongoDB Tutorial' را برای سندهایی که عنوان آن‌ها 'MongoDB Overview' است تنظیم می‌کند:

```
>db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB Tutorial'}})
>db.mycol.find()
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"New MongoDB Tutorial"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
>
```

بصورت پیش فرض، MongoDB فقط یک سطر را به‌روز رسانی خواهد کرد. برای به‌روز رسانی چندین سطر باید مقدار پارامتر 'multi' را به true تنظیم کنید.

```
>db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB Tutorial'}},{multi:true})
```

#### متد save()

متد save() سند ارسالی جدید به این متد را با سند موجود در مجموعه جایگزین می‌کند. گرامر در جدول زیر، گرامر پایه متد save() نشان داده شده است:

```
>db.COLLECTION_NAME.save({_id:ObjectId(),NEW_DATA})
```

#### مثال

مثال زیر سندی که مقدار id آن '5983548781331adf45ec7' است را جایگزین خواهد کرد:

```
>db.mycol.save(
{
  "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point New Topic", "by":"Tutorials
Point"
}
)
>db.mycol.find()
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"Tutorials Point New Topic", "by":"Tutorials Point"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
>
```

عنوان:	MongoDB #10
نویسنده:	جوادی
تاریخ:	۱۳۹۳/۱۱/۱۵
آدرس:	<a href="http://www.dotnettips.info">www.dotnettips.info</a>
گروه‌ها:	NoSQL, Database, mongodb

### حذف سند در MongoDB متد remove ()

متد remove() برای حذف یک سند از مجموعه، استفاده می‌شود. متد remove() دو پارامتر را می‌پذیرد:

1. deletion criteria (اختیاری): اسناد با توجه به شرط‌های تعیین شده در این پارامتر حذف خواهند شد.
2. justOne (اختیاری): اگر مقدار آن به true یا 1 تنظیم شود، فقط یک سند حذف می‌شود.

#### گرامر

گرامر پایه متد remove() به شکل زیر است:

```
>db.COLLECTION_NAME.remove(DELETION_CRITERIA)
```

#### مثال

مجموعه mycol را مشاهده کنید که داده‌های زیر را دارد:

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

مثال زیر همه سندهایی را که عنوان آنها 'MongoDB Overview' است، حذف می‌کند:

```
>db.mycol.remove({'title':'MongoDB Overview'})
>db.mycol.find()
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
>
```

### حذف فقط یک سند

اگر چندین سند وجود دارد و می‌خواهید فقط اولین رکورد را حذف کنید، مقدار پارامتر justOne را به true یا 1 تنظیم کنید:

```
>db.COLLECTION_NAME.remove(DELETION_CRITERIA,1)
```

### حذف همه‌ی اسناد

اگر نمی‌خواهید شرطی را برای حذف تعیین کنید، تمام اسناد یک مجموعه را حذف خواهد کرد. این معادل دستور truncate در SQL است:

```
>db.mycol.remove()
>db.mycol.find()
>
```

### پرتو در MongoDB

در MongoDB، پرتو (Projection) به معنی انتخاب داده‌های ضروری بجای انتخاب همه داده‌های یک سند است. اگر یک سند 5 فیلد دارد و شما نیاز به نمایش سه فیلد دارید؛ پس فقط باید 3 فیلد از آنها را انتخاب کنید.

#### متد find ()

متد find() که در قسمت اجرای کوئری در سند MongoDB توضیح داده شد، دو پارامتر اختیاری ورودی می‌گیرد که دومین پارامتر، لیست فیلدهایی است که می‌خواهید واکشی کنید. در MongoDB، وقتی متد find() را اجرا می‌کنید، همه فیلدهای یک سند به نمایش گذاشته می‌شوند. برای محدود کردن این متد، یک لیست از اسامی فیلدها با مقدار 0 یا 1 نیاز دارید. عدد 1 برای نمایش فیلد و عدد 0 برای عدم نمایش فیلد استفاده می‌شود. **گرامر**

گرامر پایه متد `find()` با پرتو بصورت زیر است:

```
>db.COLLECTION_NAME.find({}, {KEY:1})
```

### مثال

مجموعه `mycol` با داده زیر را ملاحظه کنید:

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

مثال زیر عناوین اسناد را هنگام اجرای کوئری نشان خواهد داد:

```
>db.mycol.find({}, {"title":1, _id:0})
{"title":"MongoDB Overview"}
{"title":"NoSQL Overview"}
{"title":"Tutorials Point Overview"}
>
```

توجه کنید که فیلد `id_` موقع اجرای متد `find()` همیشه نمایش داده خواهد شد. اگر نمی‌خواهید این فیلد را نمایش دهید، مقدار آنرا 0 تنظیم کنید.

<b>عنوان:</b>	<b>MongoDB #11</b>
<b>نویسنده:</b>	جوادی
<b>تاریخ:</b>	۱۳:۵۵ ۱۳۹۳/۱۱/۱۳
<b>آدرس:</b>	<a href="http://www.dotnettips.info">www.dotnettips.info</a>
<b>گروه‌ها:</b>	NoSQL, Database, mongodb

## بازگشت رکوردهایی محدود در MongoDB

**متد ( ) limit**

برای محدود کردن تعداد رکوردهای بازگشتی در MongoDB باید از متد limit() استفاده کنید. متد ( ) limit یک پارامتر عددی دارد که نشانگر تعداد سندهایی است که می‌خواهید نمایش دهید.

**گرامر**

گرامر پایه متد limit() به شکل زیر است:

```
>db.COLLECTION_NAME.find().limit(NUMBER)
```

**مثال**

مجموعه mycol را با داده‌های زیر، مشاهده کنید:

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

مثال زیر فقط 2 سند را هنگام اجرای کوئری نمایش می‌دهد:

```
>db.mycol.find({},{"title":1,_id:0}).limit(2)
{"title":"MongoDB Overview"}
{"title":"NoSQL Overview"}
>
```

اگر پارامتر عددی را وارد نکنید، متد limit() همه‌ی اسناد مجموعه را نمایش می‌دهد.

**متد ( ) skip**

بجز متد limit(), یک متد دیگر نیز به نام ( ) skip وجود دارد که آن نیز یک پارامتر عددی داشته و برای صرفنظر کردن از تعدادی سند استفاده می‌شود.

**گرامر**

گرامر پایه متد ( ) skip به شکل زیر است:

```
>db.COLLECTION_NAME.find().limit(NUMBER).skip(NUMBER)
```

**مثال**

مثال زیر فقط دومین سند را نمایش می‌دهد:

```
>db.mycol.find({},{"title":1,_id:0}).limit(1).skip(1)
{"title":"NoSQL Overview"}
>
```

توجه کنید که مقدار پیش فرض در متد skip() برابر صفر است.

## مرتب سازی اسناد در MongoDB

**متد ( ) sort**

برای مرتب سازی اسناد در MongoDB باید از متد sort() استفاده کنید. متد sort() یک سند را به همراه لیستی از اسامی فیلدها و با ترتیب مرتب سازی‌شان دریافت می‌کند. برای تعیین مرتب سازی از عدد 1 و -1 استفاده می‌شود. عدد 1 برای مرتب سازی صعودی و عدد -1 برای مرتب سازی نزولی استفاده می‌شود.

**گرامر**

گرامر پایه برای متد `sort()` به شکل زیر است:

```
>db.COLLECTION_NAME.find().sort({KEY:1})
```

**مثال**

مجموعه `mycol` را با داده‌های زیر، ملاحظه کنید:

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

مثال زیر داده‌های مرتب شده را بصورت نزولی بر اساس عنوان آن‌ها، نمایش می‌دهد:

```
>db.mycol.find({},{"title":1,_id:0}).sort({"title":-1})
{"title":"Tutorials Point Overview"}
{"title":"NoSQL Overview"}
{"title":"MongoDB Overview"}
>
```

اگر نمی‌خواهید اولویت مرتب سازی را مشخص کنید، متد `sort()` اسناد را بصورت صعودی مرتب می‌کند.

## ایندکس گذاری در MongoDB

ایندکس‌ها تاثیر بسیاری در اجرای کوئری‌ها دارند. بدون ایندکس‌ها، MongoDB باید تمام سندهای یک مجموعه را برای انتخاب سندهایی که با عبارت کوئری مطابقت دارند، اسکن کند. این اسکن بسیار ناکارآمد است و در این حالت MongoDB به پردازش حجم بزرگی از داده‌ها نیاز دارد.

ایندکس‌ها ساختارهای داده‌ی مخصوصی هستند که بخش کوچکی از مجموعه داده‌ها را به شکل ساده‌ای برای پیمایش، ذخیره می‌کنند. ایندکس، مقدار فیلد یا فیلدهای خاصی را که بعنوان ایندکس تعیین شده‌اند، ذخیره می‌کند.

### متد ensureIndex()

برای ساخت یک ایندکس باید از متد ensureIndex() در MongoDB استفاده کنید.

### گرامر

گرامر پایه متد ensureIndex() به شکل زیر است:

```
>db.COLLECTION_NAME.ensureIndex({KEY:1})
```

در اینجا، key نام فیلدی است که می‌خواهید بر روی آن ایندکس بسازید و مقدار 1 برای مرتب سازی بصورت صعودی است. برای ساخت ایندکس‌هایی با مرتب سازی نزولی باید از مقدار -1 استفاده کنید.

### مثال

```
>db.mycol.ensureIndex({"title":1})
```

در متد ensureIndex() برای ساخت ایندکس بر روی چندین فیلد، می‌توانید چندین فیلد را به آن پاس دهید:

```
>db.mycol.ensureIndex({"title":1,"description":-1})
```

متد ensureIndex() همچنین یک لیست از اختیارات را قبول می‌کند که در ادامه آمده‌اند:

پارامتر	نوع داده	توضیحات
background	Boolean	ایندکس را در پس زمینه می‌سازد؛ بنابراین عمل ساخت ایندکس، بقیه فعالیت‌های پایگاه داده را مسدود یا متوقف نمی‌کند. برای این کار مقدار را true تعیین کنید. مقدار پیش فرض false است.
unique	Boolean	یک ایندکس یکتا را می‌سازد. در این حالت مجموعه، اجازه درج سندهایی را که مقدار کلید یا کلیدهای آنها از قبل وجود دارند، نخواهد داد. برای ساخت یک ایندکس یکتا مقدار را true تعیین کنید. مقدار پیش فرض آن false است.
name	string	نام ایندکس. اگر تعیین نشود MongoDB نام خودکاری را توسط الحاق نام فیلدها و

پارامتر	نوع داده	توضیحات
		ترتیب مرتب سازی تولید می کند.
dropDups	Boolean	ایندکسی را بر روی فایل می سازد که ممکن است مقادیر تکراری داشته باشد. MongoDB فقط اولین پیشامد از یک کلید را ایندکس می کند و همه سندهایی را که پیشامد ثانویه کلید هستند، از مجموعه حذف می کند. برای ساخت ایندکس یکتا مقدار را true تعیین کنید. مقدار پیش فرض آن false است.
sparse	Boolean	اگر مقدار آن true تعیین شود، ایندکس فقط به سندهایی با فیلد تعیین شده رجوع می کند. این ایندکس ها از فضای کمی استفاده کرده و در برخی موقعیت ها متفاوت رفتار می کنند. مقدار پیش فرض آن false است.
expireAfterSeconds	integer	تعیین یک مقدار به ثانیه بعنوان TTL، برای کنترل کردن اینکه چه مدت MongoDB اسناد را در این مجموعه نگه دارد.
v	index version	شماره ی نسخه ایندکس. نسخه ی ایندکس پیش فرض بستگی به نسخه mongod در حال اجرای هنگام ساخت ایندکس دارد.
weights	document	وزن یک عدد بین 1 تا 99999 است و مشخص کننده ی اهمیت فیلد با دیگر فیلدهای ایندکس شده از نظر امتیاز است.
default_language	string	برای یک ایندکس متنی، زبانی برای تعیین کردن لیست کلمات متوقف کننده و نقش هایی برای ریشه یابی و نشانه گذاری کلمات.
language_override	string	برای یک ایندکس متنی، تعیین کننده نام فیلد در سند که شامل زبانی برای لغو کردن زبان پیش فرض است. مقدار پیش فرض آن language است.



## توابع جمعی در MongoDB

عملگرهای جمعی، رکوردهای اطلاعات را پردازش می‌کنند و نتیجه‌های محاسبه شده را برمی‌گردانند. عملیات جمعی مقادیر چندین سند را باهم گروه بندی می‌کند و می‌تواند یک نوع از عملگرها را روی اطلاعات دسته بندی شده انجام دهد تا یک نتیجه‌ی واحد را برگرداند. در sql، دستور count(\*) همراه Group by معادل یک تابع جمعی در MongoDB است.

## متد aggregate ()

برای توابع جمعی در MongoDB باید از متد aggregate() استفاده کنید.

## گرامر

گرامر پایه متد aggregate() به صورت زیر است:

```
>db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)
```

## مثال

در این مجموعه، داده‌های زیر را دارید:

```
{
  _id: ObjectId(7df78ad8902c)
  title: 'MongoDB Overview',
  description: 'MongoDB is no sql database',
  by_user: 'user1',
  url: 'http://www.site.com',
  tags: ['mongodb', 'database', 'NoSQL'],
  likes: 100
},
{
  _id: ObjectId(7df78ad8902d)
  title: 'NoSQL Overview',
  description: 'No sql database is very fast',
  by_user: 'user1',
  url: 'http://www.site.com',
  tags: ['mongodb', 'database', 'NoSQL'],
  likes: 10
},
{
  _id: ObjectId(7df78ad8902e)
  title: 'Neo4j Overview',
  description: 'Neo4j is no sql database',
  by_user: 'Neo4j',
  url: 'http://www.neo4j.com',
  tags: ['neo4j', 'database', 'NoSQL'],
  likes: 750
},
}
```

حالا اگر بخواهید از مجموعه‌ی بالا یک لیست را که تعداد دوره‌های نوشته شده توسط هر کاربر را نمایش می‌دهد، استخراج کنید، باید از متد aggregate () به صورت زیر استفاده نمائید:

```
> db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : 1}}}])
{
  "result" : [
    {
      "_id" : "user1",
      "num_tutorial" : 2
    },
    {
      "_id" : "Neo4j",
      "num_tutorial" : 1
    }
  ],
  "ok" : 1
}
```

معادل کوئری بالا در sql بصورت زیر خواهد بود:

```
select by_user, count(*) from mycol group by by_user
```

در مثال بالا، سندهای گروه بندی شده‌ی توسط فیلد by\_user را داریم و در هر اجرای by\_user مقدار قبلی جمع کلی افزایش می‌یابد. در اینجا لیست عبارتهای جمعی موجود، آمده است.

عبارت	توضیحات	مثال
\$sum	مقدار تعیین شده از همه سندهای مجموعه را جمع می‌کند.	db.mycol.aggregate([{\$group : {_id : "\$by_user", num_tutorial : {\$sum : "\$likes"}}}])
\$avg	میانگین همه مقادیر بدست آمده از سندهای مجموعه را محاسبه می‌کند.	db.mycol.aggregate([{\$group : {_id : "\$by_user", num_tutorial : {\$avg : "\$likes"}}}])
\$min	کمترین مقادیر مشابه را از همه سندهای مجموعه، بر می‌گرداند.	db.mycol.aggregate([{\$group : {_id : "\$by_user", num_tutorial : {\$min : "\$likes"}}}])
\$max	بیشترین مقادیر مشابه را از همه سندهای مجموعه، بر می‌گرداند.	db.mycol.aggregate([{\$group : {_id : "\$by_user", num_tutorial : {\$max : "\$likes"}}}])
\$push	یک مقدار را در سند نتیجه، در یک آرایه درج می‌کند.	db.mycol.aggregate([{\$group : {_id : "\$by_user", url : {\$push : "\$url"}}}])
\$addToSet	یک مقدار را در سند نتیجه در یک آرایه درج می‌کند، اما مقدار تکراری ایجاد نمی‌کند.	db.mycol.aggregate([{\$group : {_id : "\$by_user", url : {\$addToSet : "\$url"}}}])
\$first	اولین سند از اسناد را برطبق گروه بندی بر می‌گرداند. معمولا این عبارت بعد از عبارتهای مرتب سازی مرحله‌ای استفاده می‌شود.	db.mycol.aggregate([{\$group : {_id : "\$by_user", first_url : {\$first : "\$url"}}}])
\$last	آخرین سند از اسناد را برطبق گروه بندی بر می‌گرداند. معمولا این عبارت بعد از عبارتهای مرتب سازی مرحله‌ای استفاده می‌شود.	db.mycol.aggregate([{\$group : {_id : "\$by_user", last_url : {\$last : "\$url"}}}])

### مفهوم Pipeline

در Command shell یونیکس، خط لوله (Pipeline) به معنی امکان اجرای یک عملیات روی چندین ورودی و استفاده از خروجی بعنوان ورودی برای دستور بعدی و ادامه‌ی آن است. MongoDB نیز این مفهوم را در چارچوب توابع جمعی پشتیبانی می‌کند. یک مجموعه از مراحل وجود دارند که هرکدام از آنها یک مجموعه از اسناد را بعنوان ورودی می‌گیرند و یک مجموعه از سند را بعنوان نتیجه (یا نتیجه را بعنوان سند JSON در پایان خط لوله) ارائه می‌دهند. این عمل به نوبه خود می‌تواند برای مرحله بعد و یا مراحل بعدی، استفاده شود.

مراحل ممکن در چارچوب توابع جمعی در زیر آمده اند:

\$project: برای انتخاب چندین فیلد از یک مجموعه استفاده می‌شود.

\$match: این یک عملگر فیلترگذاری است که می‌تواند میزان اسنادی را که بعنوان ورودی در مرحله بعد گرفته می‌شوند، کاهش

دهد.

`group$` : این همان تابع جمعی است که در بالا توضیح داده شد.

`skip$` : توسط این عبارت، در یک لیست بدست آمده (نتیجه)، می‌توانید از لیست اسناد بصورت روبه جلو صرفنظر کنید.

`limit$` : این عبارت تعداد اسناد را توسط عدد گرفته شده، از موقعیت فعلی برای نمایش محدود می‌کند.

`unwind$` : این عبارت برای باز کردن (unwind) سندی که از آرایه‌ها بهره‌گیری می‌کند استفاده می‌شود. وقتی از آرایه استفاده می‌کنید، داده از نوع پیش پیوست (Pre-joined) است و با این نوع داده، این عمل برای داشتن سندهای اختصاصی نا تمام خواهد ماند. بنابراین با این مرحله می‌توانید میزان اسناد را برای مرحله بعد افزایش دهید.

## عمل تکثیر در MongoDB

عمل تکثیر (Replication) به فرآیند همزمان سازی داده در میان چند سرور گفته می‌شود. تکثیر، افزونگی را فراهم می‌آورد و دسترسی پذیری داده‌ها را توسط کپی داده در چندین سرور مختلف افزایش می‌دهد. این کار، یک پایگاه داده را در مقابل از دسترس خارج شدن یک سرور مفرد، محافظت می‌کند. همچنین امکان بازیابی از خرابی سخت افزار و وقفه‌های سرویس را به کاربر می‌دهد. توسط کپی برداری از اطلاعات، می‌توانید یکی از آنها را برای بازیابی، گزارشگیری و پشتیبان گیری اختصاص دهید.

### چرا تکثیر؟

برای ایمن نگه داری اطلاعات

دسترسی پذیری بالای اطلاعات (شبانه روزی)

بازیابی اطلاعات

نیازی به از کار افتادن هنگام انجام عملیات نگه‌داری ندارد

مقایسه پذیری خواندن داده‌ها (کپی برداری‌های اضافه برای عمل خواندن)

کپی اطلاعات برای نرم افزارها شفاف و قابل دستیابی است.

### تکثیر در MongoDB چگونه کار می‌کند

MongoDB عمل تکثیر را با استفاده از مجموعه کپی یا المثنی (Replica set) انجام می‌دهد. مجموعه کپی یک گروه از نمونه‌های mongod هستند که مجموعه داده یا دیتاست مشابهی را میزبانی (Host) می‌کنند. در یک کپی داده، یک گره، گره اصلی است که تمام عملیات نوشتن را دریافت می‌کند. بقیه‌ی نمونه‌های ثانویه، عملیات را از گره اصلی، دریافت و اعمال می‌کنند؛ بنابراین آنها هم دیتاست مشابهی دارند. مجموعه‌ی کپی تنها می‌تواند یک گره‌ی اصلی داشته باشد. یک مجموعه‌ی کپی، یک گروه از دو یا چند گره است. (عموما حداقل 3 گره نیاز است).

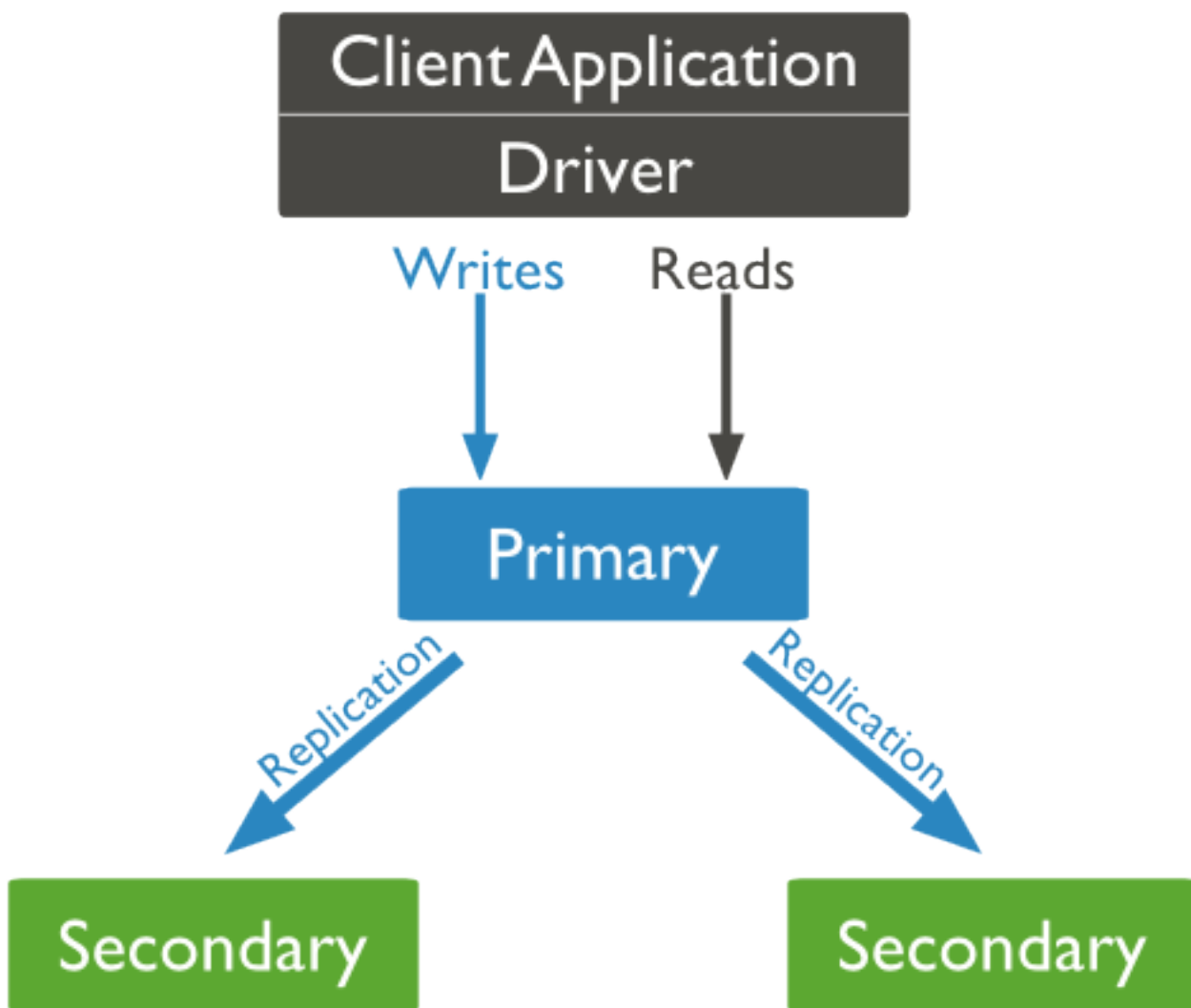
در یک مجموعه‌ی کپی، یک گره، گره اصلی است و بقیه گره‌ها گره‌های ثانویه هستند.

همه‌ی داده‌ها از گره‌ی اصلی به گره‌های ثانویه تکثیر می‌شوند.

هنگام انجام عملیات نگه داری یا ازدسترس خارج شدن سرور، گزینش برای گره اصلی و انتخاب گره اصلی جدید آغاز می‌شود.

گره از کار افتاده، بعد از بازیابی دوباره، به مجموعه کپی ملحق می‌شود و بعنوان یک گره ثانویه کار می‌کند.

در زیر یک نوع دیاگرام از تکثیر در MongoDB نشان داده شده است که در آن برنامه‌ی سمت کلاینت همیشه با گره‌ی اصلی در ارتباط است و گره‌ی اصلی، داده‌ها را گره‌های ثانویه تکثیر می‌کند.



ویژگی‌های مجموعه‌ی کپی

یک کلاستر از N عدد گره

هر گره‌ای می‌تواند گره اصلی باشد

همه‌ی عملیات نوشتن بر روی گره اصلی انجام می‌شود

عمل ازدسترس خارج شدن سرور و جایگزین شدن یک گره بصورت اتوماتیک

بازیابی بصورت اتوماتیک

همراهی و توافق در گزینش گره اصلی

ساختن یک مجموعه کپی

در اینجا می‌خواهیم یک نمونه از mongodb را به یک مجموعه‌ی کپی تبدیل کنیم. برای این کار مراحل زیر را انجام دهید:  
همه‌ی نمونه‌های در حال اجرای mongod را در سمت سرور، متوقف کنید.

اکنون mongod سمت سرور را با سوئیچ -replSet راه اندازی کنید.

گرامر پایه --replSet به شکل زیر است:

```
mongod --port "PORT" --dbpath "YOUR_DB_DATA_PATH" --replSet "REPLICA_SET_INSTANCE_NAME"
```

#### مثال

```
mongod --port 27017 --dbpath "D:\set up\mongodb\data" --replSet rs0
```

دستور فوق یک نمونه از mongod را با نام rs0، روی پورت 27017 راه اندازی می‌کند. اکنون command prompt را باز کنید و به این نمونه mongod متصل شوید. در سمت کلاینت، دستور rs.initiate() را برای شروع کردن یک مجموعه‌ی کپی جدید صادر کنید. برای چک کردن تنظیمات مجموعه‌ی کپی، دستور rs.conf() را صادر کنید. برای چک کردن وضعیت مجموعه کپی نیز دستور rs.status() را صادر کنید.

#### افزودن اعضا به مجموعه‌ی کپی

برای افزودن اعضا به مجموعه‌ی کپی، چند نمونه mongod را در چندین کامپیوتر راه اندازی کنید. اکنون برنامه‌ی سمت کلاینت را اجرا و دستور rs.add() را اجرا کنید.

#### گرامر

گرامر پایه دستور rs.add() به شکل زیر است:

```
>rs.add(HOST_NAME:PORT)
```

#### مثال

فرض کنید نام نمونه‌ی mongod شما mongod1.net و بر روی پورت 27017 در حال اجراست. برای افزودن این نمونه به مجموعه کپی، دستور rs.add() را در سمت کلاینت اجرا کنید.

```
>rs.add("mongod1.net:27017")
>
```

توجه کنید که فقط وقتی می‌توانید یک نمونه mongod را برای مجموعه کپی اضافه کنید که به گره اصلی متصل باشید. برای چک کردن اینکه به گره اصلی متصل هستید، دستور db.isMaster() را در سمت کلاینت صادر کنید.

## Sharding

Sharding فرآیند ذخیره سازی رکوردهای اطلاعاتی در چندین سرور است و این رویکرد MongoDB برای درخواست داده‌های در حال رشد است. همانطور که اندازه‌ی داده در افزایش است، شاید یک ماشین تنها برای ذخیره سازی داده‌ها کافی نباشد و یا نتواند کارآیی قابل قبولی را برای خواندن و نوشتن فراهم کند. Sharding این مشکل را با مقایس پذیری افقی حل نموده است. توسط Sharding، می‌توانید دستگاه‌های دیگری را برای پشتیبانی از داده‌های در حال رشد بیافزایید و عملیات خواندن و نوشتن را بیشتر پوشش دهید.

### چرا Sharding؟

در عمل تکثیر، همه‌ی نوشتن‌ها به سمت گره اصلی می‌روند.

پرس و جوهای حساس به تاخیر نیز به سمت گره اصلی می‌روند.

مجموعه کپی مفرد به 12 گره محدود است.

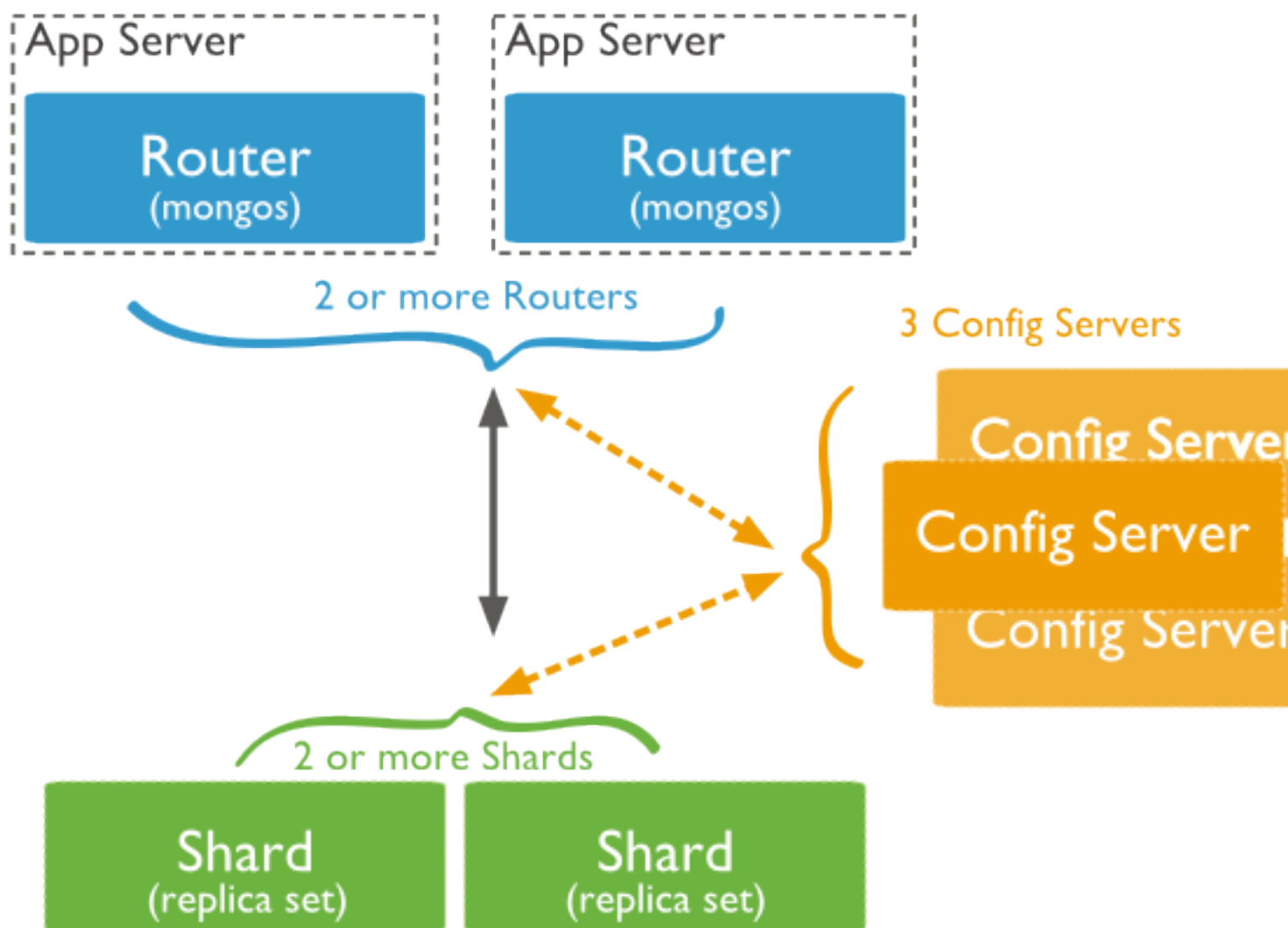
وقتی دیتاست خیلی بزرگ می‌شود، حافظه به اندازه کافی نمی‌تواند بزرگ شود.

دیسک محلی به اندازه‌ی کافی بزرگ نیست.

هزینه مقیاس پذیری عمودی بسیار بالاست.

### MongoDB در Sharding

دیاگرام زیر نحوه Sharding در MongoDB را نشان می‌دهد:



در دیاگرام بالا سه جز اصلی وجود دارند که در ادامه توضیح داده شده‌اند:

**Shard:** برای ذخیره داده استفاده می‌شود. آنها دسترس پذیری بالا و پایداری داده را فراهم می‌کنند. در محیط تولید هر Shard یک مجموعه کپی جداست.

**Config Server:** متا دیتای کلاستر را نگهداری می‌کند. این داده‌ها شامل اطلاعات نگاشت دیتاست کلاستر برای Shards است. مسیریاب کوئری (query router) از این متا دیتا برای نشان گذاری عملیات برای Shards تعیین شده استفاده می‌کند. در محیط تولید Shard شده، کلاسترها دقیقاً 3 سرور تنظیمات دارند.

**Query Routers:** مسیریاب کوئری‌ها بطور اساسی نمونه‌های mongos و واسط بین برنامه کلاینت هستند و عملیات را به Shard مناسب هدایت می‌کنند. مسیریاب کوئری عملیات را برای Shard، پردازش و نشان گذاری می‌کنند و نتیجه را برای کاربر برمی گردانند. یک Shard کلاستر شده می‌تواند شامل چندین مسیریاب کوئری (برای تقسیم بارگیری درخواست کلاینت) باشد. یک کلاینت می‌تواند درخواست هایش را به یک مسیریاب کوئری ارسال کند. عموماً یک Shard کلاستر شده چندین مسیریاب کوئری دارد.