آموزش کاربردی پایگاه داده‌ی نارابطه‌ای

MongoDB

کیل بانکر، پیتر باکوم، شاون ورچ

ترجمه و تألیف: مهدی مرسلی

انتشارات پندار پارس

شابک : 978-600-8201-17-5 : ۲۰۰۰۰۰ریال
شماره کتابشناسی ملی : ۴۳۶۳۴۹۸
عنوان و نام پدیدآور : آموزش کاربردی پایگاه داده‌ی نارابطه‌ای MongoDB/ کیل بانکر، پیتر باکوم، شاون ورچ [صحیح: کیل بانکر... [و دیگران]]؛ ترجمه مهدی مرسلی.
مشخصات نشر : تهران : پندار پارس ، ۱۳۹۵.
مشخصات ظاهری : ۲۶۰ ص.: مصور، جدول.
یادداشت : بخش اعظم کتاب حاضر ترجمه کتاب MongoDB in action , 2nd. ed, [2016] تالیف کایل بنکر... [و دیگران] است.
یادداشت : کتابنامه.
موضوع : مونگو دی بی، MongoDB
موضوع : پایگاه‌های اطلاعاتی شئ‌گرا، Object-oriented databases
موضوع : پایگاه‌های اطلاعاتی -- مدیریت -- نرم‌افزار، Database management -- Software
موضوع : نرم‌افزار متن‌باز، Open source software
موضوع : پایگاه‌های اطلاعاتی – مدیریت، Database management
رده بندی دیویی : ۷۴/۰۰۵
رده بندی کنگره : ۹/۷۶QA ۱۳۹۵ ۴۴م۲پ/
سرشناسه : مرسلی، مهدی، ۱۳۶۰ -، گردآورنده، مترجم
شناسه افزوده : بنکر، کایل، ۱۹۸۰ - م. Banker, Kyle
وضعیت فهرست نویسی : فیپا

انتشارات پندارپارس

دفتر فروش: انقلاب، ابتدای کارگرجنوبی، کوی رشتچی، شماره ۱۴، واحد ۱۶ www.pendarepars.com
تلفن: ۶۶۵۷۲۳۳۵ - تلفکس: ۶۶۹۲۶۵۷۸ همراه: ۰۹۲۱۴۳۷۱۹۶۴ info@pendarepars.com

نام کتاب : آموزش کاربردی پایگاه داده‌ی نارابطه‌ای MongoDB
ناشر : **انتشارات پندار پارس**
تالیف : **کیل بانکر، پیتر باکوم، شاون ورچ**
ترجمه و تألیف : **مهدی مرسلی**
چاپ نخست : **مهر ۹۵**
شمارگان : **۵۰۰ نسخه**
طرح جلد : **رامین شکرالهی**
چاپ، صحافی : **روز**
قیمت : **۲۰۰۰۰ تومان** **شابک : ۵-۱۷-۸۲۰۱-۶۰۰-۹۷۸**

گوهر معرفت اندوز که با خود ببری
که نصیب دگران است نصاب زر و سیم
"حافظ"

پیشگفتار

کتابی که در دست دارید، منبعی برای آموزش نرم‌افزار مدیریت پایگاه داده‌های MongoDB می‌باشد. با توجه به نبود منابع فارسی برای آموزش نرم‌افزارهای تخصصی کامپیوتر، بر آن شدیم تا گام کوچکی در این راه برداریم و راه را برای آموزش یکی این نرم‌افزارها کوتاه‌تر کنیم.

گرچه، شاید خواننده فرهیخته و علاقه‌مند، مشکلی در خواندن منابع به زبان انگلیسی نداشته باشد؛ با این وجود مطالعه به زبان دوم همیشه زمان‌بر بوده و ممکن است باعث کندی پیشرفت در کار باشد.

با توجه به علاقه شخصی حقیر به پایگاه داده‌های رابطه‌ای، در آغازِ راه، پذیرش لازم بودن آموزش پایگاه داده‌های نارابطه‌ای کمی دشوار بود ولی با مطالعاتی که در زمینه پایگاه داده‌های نارابطه‌ای انجام شد به این نتیجه رسیدیم که چه ما دوست داشته باشیم و چه دوست نداشته باشیم، دنیای وب، پایگاه داده‌ای مناسب نیاز دارد و MongoDB می‌تواند پاسخی مناسب برای این نیاز روز افزون باشد. از سوی دیگر، بیشتر منابع در این زمینه دارای حجم بسیار زیادی بودند و به مطالبی پرداخته بودند که بیشتر به کار متخصصان پایگاه داده و برنامه‌نویسان باتجربه می‌آمد و به جای اینکه راهگشای دانشجویان و جوانان علاقه‌مند باشد، باعث سردرگمی بیشتر آنان می‌شد. لذا تصمیم گرفتیم منبعی کاربردی و قابل فهم فراهم آوریم تا بتواند در سطح مبتدی و متوسط راه‌گشا بوده و گرهی از مشکلات علاقه‌مندان به زمینه وب و پایگاه داده‌های نارابطه‌ای بگشاید.

با قاطعیت می‌توان گفت که جز فصل نخست کتاب که معرفی MongoDB می‌باشد، در دیگر فصل‌ها از آوردن مطالب تئوری محض دوری کرده و با آوردن مثال‌های کاربردی و انجام یک پروژه وب سایت تجارت الکترونیک، کوشیده‌ایم به خواننده نگوییم چگونه می‌تواند کار کند، بلکه روش انجام کار را نشان دهیم. " صالح و طالح متاع خویش نمودند، تا که قبول افتد و که در نظر آید".

برای سادگی استفاده از کتاب، کدهای مثال‌های کتاب را در صفحه مربوط به کتاب در سایت انتشارات قرار داده‌ایم و می‌توانید این کدها را دانلود کرده و استفاده نمایید.

در پایان از دوستانی که این کتاب را مطالعه می‌کنند، نهایت سپاسگزاری را داریم که وقت گرانبهای خود را صرف مطالعه این اوراق کرده‌اند و برای تمام کاستی‌ها از یکایک آنان پوزش می‌طلبیم. تمام خطاهای برگردان و نگارش و . . . این کتاب بر عهده حقیر می‌باشد و از طریق آدرس پست الکترونیکی mehdi_morsali@yahoo.com جویای نظرات اصلاحی دوستان هستم.

مهدی مرسلی، تابستان ۹۵

فهرست

فصل نخست

پایگاه داده‌ای برای وب مدرن

این فصل شامل مطالب زیر است:

- تاریخچه MongoDB، اهداف طراحی و خصوصیات کلیدی
- معرفی کوتاه پوسته و راه‌اندازها
- کاربردها و محدودیت‌ها
- تغییرات جدید در MongoDB

اگر در چند سال گذشته برنامه تحت وب ساخته باشید، شاید از یک پایگاه داده رابطه‌ای به‌عنوان ابزار ذخیره‌سازی اصلی استفاده کرده باشید. اگر با SQL آشنا باشید، حتما از مفید بودن یک مدل نرمال‌سازی شده خوب آگاهید و لزوم استفاده از تراکنش‌ها را می‌دانید و از اطمینان فراهم شده توسط این موتورهای ذخیره‌سازی باخبر هستید. زبان‌های ساده پایگاه داده‌های رابطه‌ای، بسیار شناخته شده و بالغ هستند. وقتی برنامه‌نویسان به دفاع از ابزارهای ذخیره‌سازی جدید پرداختند پرسش‌های زیادی درباره حیات و کارایی این فناوری جدید مطرح شد. آیا این ابزارهای ذخیره‌سازی داده جایگزینی برای سیستم‌های پایگاه داده رابطه‌ای هستند؟ چه کسی از آنها در تولید نرم‌افزار استفاده می‌کند و چرا؟ بایدها و نبایدهای رفتن به یک پایگاه داده نارابطه‌ای چیستند؟ پاسخ به پرسش‌های بالا منجر به پاسخ به این پرسش می‌شود که: چرا برنامه‌سازان به MongoDB علاقه دارند؟

MongoDB یک سیستم مدیریت پایگاه داده است که برای ساخت سریع برنامه‌های تحت وب و زیرساخت اینترنت طراحی شده است. مدل داده و استراژی‌های ماندگاری این سیستم برای خواندن و نوشتن با توان عملیاتی بالا و قابلیت گسترش سریع و کنترل خطای خودکار ساخته شده‌اند. نرم‌افزار چه به یک گره پایگاه داده نیاز داشته باشد و چه به ده‌ها گره، MongoDB کارایی شگفت

آوری دارد. اگر متخصص گسترش پایگاه داده‌های رابطه‌ای باشید، این برای شما خبر خوبی است. البته ممکن است همه نخواهند در مقیاس بزرگ کار کنند. شاید بخواهید تنها یک سرور پایگاه داده داشته باشید؛ در این صورت چرا باید از MongoDB استفاده کنید؟

بزرگ‌ترین دلیل برای استفاده برنامه‌نویسان از MongoDB قابلیت گسترش آن نیست بلکه بیشتر مدل داده‌ای قابل درک آن است. MongoDB اطلاعات خود را به صورت اسناد ذخیره می‌کند که با سطرهای پایگاه داده رابطه‌ای متفاوت است. سند چیست؟ به مثال زیر توجه کنید:

```
{
  _id: 10,
  username: 'peter',
  email: 'pbbakkum@gmail. com'
}
```

این یک سند ساده و زیباست که چند فیلد اطلاعاتی درباره یک کاربر را ذخیره می‌کند. اینک، امتیاز این مدل چیست؟ حالتی را در نظر بگیرید که می‌خواهید چند ایمیل برای هر کاربر ذخیره کنید. در پایگاه داده رابطه‌ای برای انجام این کار باید جدول‌های جداگانه‌ای برای ایمیل و کاربر ساخت که با هم در ارتباط باشند. اما MongoDB برای شما راه‌حل تازه‌ای دارد:

```
{
  _id: 10,
  username: 'peter',
  email: [
    'pbbakkum@gmail. com',
    'pbb7c@virginia. edu'
  ]
}
```

و به همین‌ترتیب می‌توانید به سادگی، آرایه‌ای از ایمیل‌ها داشته باشید و مشکل خود را حل کنید. به‌عنوان یک برنامه‌ساز، برای شما بسیار مفید خواهد بود که بتوانید سند ساخت یافته‌ای مانند این را بدون نگرانی درباره شمای مناسب یا افزودن جدول‌های جدید هنگام تغییر داده‌ها ذخیره کنید.

قالب سند MongoDB بر پایه JSON بنا شده است که یک شِمای[1] محبوب برای ذخیره ساختارهای داده دلخواه می‌باشد. JSON کوته نوشت Java Script Object Notation است. احتمالا بدانید که ساختار JSON از کلیدها و مقادیر تشکیل شده است و می‌تواند عمق دلخواهی داشته باشد. این ساختارها با واژه نامه‌ها و نقشه‌های درهم‌سازی در زبان‌های برنامه‌سازی دیگر قابل مقایسه‌اند.

مدل داده مبتنی بر سند می‌تواند ساختارهای داده‌ای سلسله مراتبی قدرتمند را بازنمایی کند و این کار اغلب بدون نیاز به ارتباط‌های میان جدولی است که در پایگاه داده‌های رابطه‌ای معمول است.

---

[1] Schema

برای نمونه، فرض کنید محصولات یک سایت تجارت الکترونیک را مدل‌سازی می‌کنید. با یک مدل داده‌ای کاملا نرمال‌سازی شده، اطلاعات یک محصول ممکن است به ده‌ها جدول تقسیم شود. اینک اگر بخواهید بازنمایی محصول را از پوسته پایگاه داده به دست آورید باید یک کوئری بنویسید که شامل چندین پیوند میان جدول‌ها باشد.

در عوض با استفاده از مدل سند، بیشتر اطلاعات محصول می‌تواند در یک سند بازنمایی شود. وقتی پوسته جاوا اسکریپت MongoDB را باز می‌کنید، می‌توانید یک بازنمایی قابل درک سلسله مراتبی سازماندهی شده مانند ساختار JSON را داشته باشید که می‌توانید به آسانی آن را ویرایش کرده و کوئری‌های خود را روی آن اجرا کنید. قابلیت‌های کوئری‌سازی MongoDB به‌ویژه برای ویرایش اسناد ساخت‌یافته طراحی شده است، بنابراین کاربرانی که از پایگاه داده‌های رابطه‌ای آمده‌اند، قدرت کوئری‌سازی مشابهی را تجربه می‌کنند. افزون بر این، اکنون بیشتر برنامه‌سازان با زبان‌های شیءگرا کار می‌کنند و به ابزار ذخیره‌سازی داده‌ای نیاز دارند که با اشیاء، بهتر نگاشت شود. با MongoDB، یک شیء که در زبان برنامه‌نویسی تعریف شده است، اغلب می‌تواند با حذف برخی از پیچیدگی‌های نگاشت اشیاء، به همان صورت باقی بماند. اگر تجربه کار در پایگاه داده رابطه‌ای را دارید، مهارت شما می‌تواند در کوچ به MongoDB مفید باشد.

اگر تمایز میان بازنمایی جدولی و بازنمایی شیءگرای داده‌ها برای شما موضوع جدیدی است، شاید پرسش‌های زیادی در ذهن شما ایجاد شده باشد. مطمئن باشید در پایان این بخش، یک خلاصه کامل از خصوصیات و اهداف طراحی MongoDB به دست خواهید آورد. تاریخچه MongoDB را مرور کنید تا با خصوصیات اصلی پایگاه داده‌ها آشنا شوید و پس از آن با راه حل‌های جایگزین خانواده NoSQL آشنا شوید و ببینید که MongoDB در کجای این طیف قرار دارد. در پایان خواهید آموخت که MongoDB بهترین عملکرد را در کجا دارد و دیگر پایگاه داده‌ها، در کجا به MongoDB ترجیح داده می‌شوند.

MongoDB گاهی به حق و گاه به ناحق مورد انتقاد واقع شده است و شما باید توانایی‌ها و محدودیت‌های آن را بیاموزید. MongoDB برای برخی از انواع داده‌ها بسیار مناسب است و نداشتن شِما به این معنی است که می‌تواند یکی از مهم‌ترین ابزارها برای ساخت سریع یک نرم‌افزار باشد. هدف ما این است که به شما اطلاعاتی بدهیم تا بتوانید به درستی تصمیم بگیرید که آیا MongoDB برای شما مناسب است یا خیر و توضیح دهیم که چگونه به طور کارآمدی از آن استفاده کنید.

۱-۱- ساخته شدن برای اینترنت

تاریخچه MongoDB کوتاه ولی با ارزش است، زیرا MongoDB از دل یک پروژه جاه‌طلبانه برآمده است. در میانه سال ۲۰۰۷ یک پروژه تازه کار در شهر نیویورک که 10gen نامیده می‌شد شروع به کار روی یک PaaS[1] کرد که از یک اپلیکیشن سرور و یک پایگاه داده تشکیل شده بود و باید از برنامه‌های تحت وب پشتیبانی می‌کرد و در صورت نیاز آنها را گسترش می‌داد.

سکوی 10gen مانند موتور googles APP طراحی شده بود تا مشکل گسترش‌پذیری و مدیریت زیرساخت سخت‌افزاری و نرم‌افزاری را به صورت خودکار انجام داده و کاربر را از تمرکز انحصاری بر روی کد برنامه خلاص کند. 10gen در نهایت کشف کرد که برنامه‌سازان با دادن کنترل به دست فناوری چندان راحت نیستند ولی فناوری پایگاده داده جدید 10gen را می‌خواستند. این موضوع باعث شد 10gen تنها بر روی پایگاه داده تمرکز کند که MongoDB نام گرفت.

10gen نام خود را به MongoDB تغییر داده و به پشتیانی خود از پایگاه داده‌ها به صورت یک پروژه متن باز ادامه داد. کدهای پروژه در دسترس عمومی هستند و تغییر و استفاده از آنها بر اساس موافقت‌نامه، آزاد است و جامعه استفاده کننده، به گزارش دادن خطاها و ثبت patchها تشویق می‌شوند. اکنون بیشتر برنامه‌سازان هسته‌ی MongoDB، مؤسسان و کارمندان شرکت MongoDB هستند و پروژه را برپایه‌ی نقشه راه خود ادامه می‌دهند تا به نیازهای جامعه کاربران پاسخ دهند و هدف خود را به ساخت یک پایگاه داده که بهترین خصوصیات پایگاه داده رابطه‌ای و ذخیره‌سازهای کلید- مقدارِ توزیع شده را داشته باشد، تغییر داده‌اند. بنابراین ساختار مدل تجاری شرکت MongoDB بی‌شباهت به دیگر شرکت‌های متن باز خوشنام نیست: پشتیبانی از ساخت یک محصول متن باز و ارائه تأییدیه به کاربران نهایی. مهم‌ترین چیزی که باید از تاریخچه MongoDB به یاد داشته باشیم این است که MongoDB تلاش می‌کند به‌عنوان بخشی از پشته نرم‌افزارهای تحت وب، تا حد بسیار زیادی ساده و انعطاف‌پذیر باشد.

---

[1] Platform as a Service

۱-۲- خصوصیات کلیدی MongoDB

یک پایگاه داده عمدتا با مدل داده‌ای خود شناخته می‌شود. در این بخش نگاهی به مدل داده مبتنی بر سند خواهیم داشت و خصوصیاتی از MongoDB را خواهیم دید که به ما اجازه می‌دهد تعامل کارآمدی با این مدل داشته باشیم.

۱-۲-۱- مدل داده‌ای مبتنی بر سند

مدل داده‌ای MongoDB سندگراست. اگر با اسناد در مبحث پایگاه داده‌ها آشنا نیستید، این مفهوم به سادگی با یک مثال روشن می‌شود. یک سند JSON در همه جا به جز مقادیر عددی نیاز به دابل کوتیشن دارد. لیست۱-۱ نسخه جاوا اسکریپتی از JSON را نمایش می‌دهد که در آن دابل کوتیشن لازم نیست.

این لیست یک سند JSON را نمایش می‌دهد که مطلبی را روی یک خبر اجتماعی (مانند Reddit یا twitter) بازنمایی می‌کند. همان‌گونه که می‌بینید یک سند عبارت است از مجموعه‌ای از نام مشخصه‌ها و مقادیر آنها. مقادیر می‌توانند انواع داده‌ای ساده مانند رشته‌ها، اعداد و تاریخ‌ها باشند. اما این مقادیر همچنین می‌توانند آرایه‌ها و یا سندهای JSON دیگر نیز باشند.❷ این ساختار به سند اجازه بازنمایی ساختارهای داده‌ای قدرتمندی را می‌دهد. همچنین می‌بینید که سند، مشخصه tags نیز دارد که برچسب‌های مطلب را در آرایه‌ای ذخیره می‌کند❶ و البته جالب‌تر از همه، مشخصه‌ی comments می‌باشد❸ که آرایه‌ای از اسناد comment می‌باشد.

MongoDB در درون خود اسناد را با فرمت JSON دودویی یا BSON ذخیره می‌کند. BSON ساختاری مانند JSON دارد ولی می‌تواند اسناد زیادی را ذخیره کند. وقتی که در MongoDB یک کوئری می‌سازید و پاسخ می‌گیرید، کوئری شما به یک ساختار داده قابل خواندن آسان[1] برگردان می‌شود. پوسته MongoDB از جاوا اسکریپت استفاده می‌کند و اسناد را به زبان JSON می‌گیرد که ما نیز در بیشتر مثال‌های خود از آن استفاده خواهیم کرد. در فصل‌های آینده درباره توانایی‌های BSON بیشتر صحبت خواهیم کرد.

همان‌گونه که پایگاه داده رابطه‌ای، دارای شماری جدول است، MongoDB نیز داری مجموعه (collection) است. به دیگر سخن، MYSQL (یک پایگاه داده رابطه‌ای محبوب) داده‌های خود را در

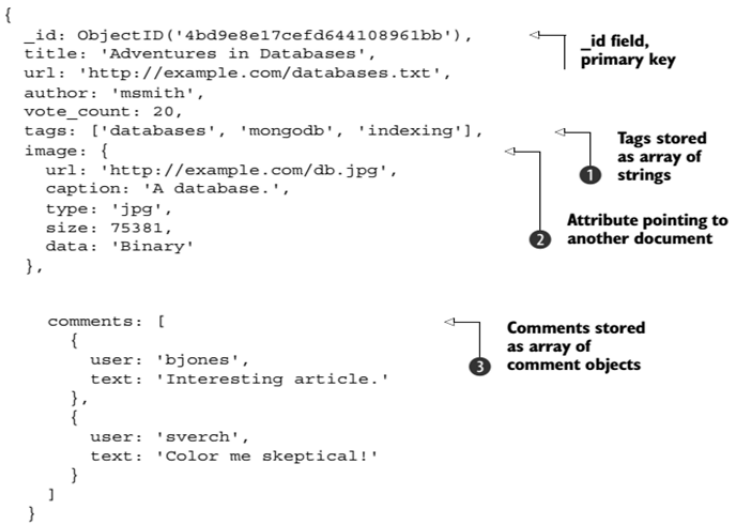
---

[1] Easy-to-read Data Structure

جدول‌هایی از سطرها ذخیره می‌کند؛ در حالی که MongoDB داده‌های خود را در مجموعه‌هایی از سندها نگهداری می‌کند که می‌توانید آن را گروهی از اسناد در نظر بگیرید. مجموعه‌ها مفاهیم مهمی در MongoDB هستند. داده‌ها در مجموعه‌ها بر روی دیسک ذخیره می‌شوند و بیشتر کوئری‌ها از شما می‌خواهند که مشخص کنید، مجموعه هدف شما کدام است.

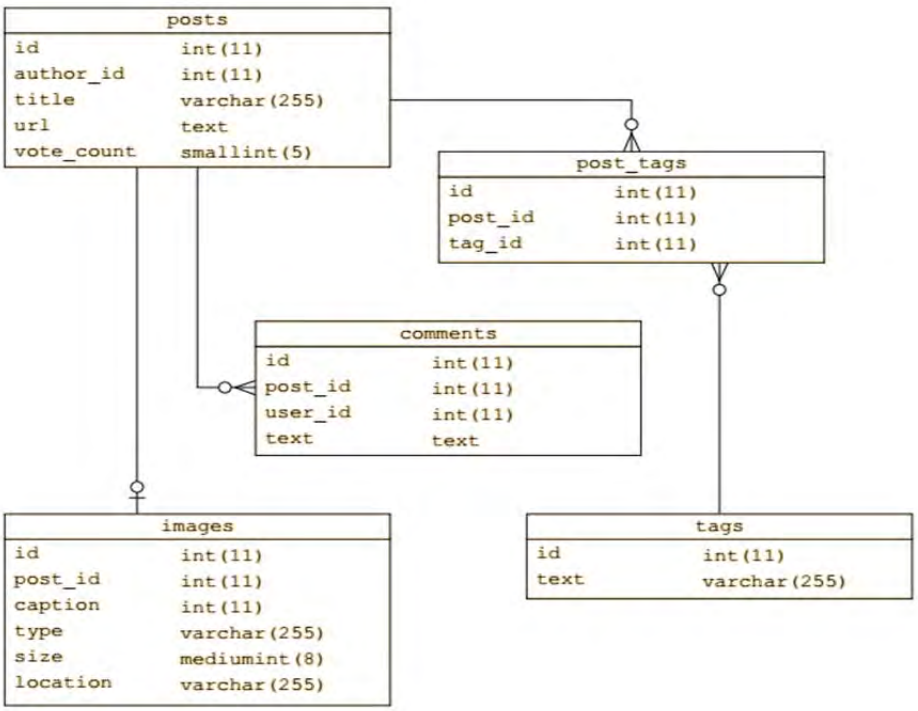
بیایید برای لحظه‌ای مجموعه‌های MongoDB را با بازنمـایی اسـتاندارد پایگاه داده رابطه‌ای مقایسه کنیم. شکل۱-۱چنین پایگاه داده‌ای را نمایش می‌دهد. چون جدول‌ها اصولا تخت هستند، بازنمایی ارتباط‌های یک به چند در سند post به چندین جدول نیاز دارد. پس با جدول posts آغاز می‌کنید که شامل اطلاعات اصلی برای هر پست است، سپس سه جدول دیگر می‌سازید که هر کدام از آنها شامل فیلد post_id برای ارجاع به post اصلی می‌باشند. تکنیک پخش داده‌های اشیاء به چند جدول مانند آنچه می‌بینید نرمال‌سازی نامیده می‌شود. یک مجموعه داده نرمال‌سازی شده، همراه با چیزهای دیگر، این اطمینان را می‌دهد که هر واحد داده، تنها در یک مکان بازنمایی شده است.

اما نرمال‌سازی سخت‌گیرانه، بدون هزینه نیست و نیاز به چند اسمبلی داریم. برای نمایش پستی که دیدیم نیاز به ایجاد یک ارتباط، میان جدول‌های post و comments داریم. پاسخ به اینکه نرمال‌سازی تا چه‌اندازه لازم است، بستگی به داده‌ای دارد که مدل‌سازی می‌کنید و در فصل چهارم بیشتر در این باره خواهیم گفت. آنچه که در اینجا باید گفته شود، این است که مدل داده‌ای سندگرا، داده‌ها را در قالب تجمیع شده بازنمایی می‌کند، از comment تا tag می‌توانند در یک شئ پایگاه داده قرار گیرند.



لیست ۱-۱- یک سند که یک ورودی روی یک سایت اخبار اجتماعی را بازنمایی می کند

شاید متوجه شده‌اید که اسناد افزون بر اینکه ساختار قدرتمند دارند، نیاز به شمای دقیق نیز ندارند. با یک پایگاه داده رابطه‌ای، داده‌ها را در سطرهای یک جدول ذخیره می‌کنید. هر جدول یک شِمای تعریف شده‌ی دقیق دارد که چه ستون‌ها و انواع داده‌هایی در آن مجاز هستند. اگر یکی از سطرهای یک جدول نیاز به یک فیلد اضافه داشته باشد، باید کل جدول را تغییر دهید. MongoDB اسناد را در مجموعه‌هایی گروه‌بندی می‌کند که هیچ شِمایی ندارند. به طور نظری هر سند در یک مجموعه می‌تواند یک ساختار کاملا متفاوت داشته باشد. درعمل، اسناد یک مجموعه، قالب نسبتا یکسانی دارند. برای نمونه، هر سند در مجموعه post فیلدهایی برای title، tags، comments و . . . دارد.



شکل ۱-۱- یک مدل داده‌ای رابطه‌ای برای ورودی‌های یک سایت اخبار اجتماعی. خط‌های با پایان صلیب مانند، یک ارتباط یک به یک را بازنمایی می کنند، بنابراین تنها یک سطر از جدول images به یک سطر از جدول پست وابسته است. خطوطی که پایان آنها چند شاخه شده است، یک ارتباط یک به چند را بازنمایی می‌کنند، بنابراین چند سطر در جدول comments می‌توانند با یک سطر از جدول posts در ارتباط باشند

مزایای مدل بدون شِما

نداشتن شِما مزایایی دارد. نخست اینکه ساختار داده را کد برنامه، و نه پایگاه داده مشخص می‌کند که می‌تواند در زمان‌هایی که شِما به تناوب عوض می‌شود، ساخت برنامه‌ی ابتدایی را سریع‌تر کند.

دوم اینکه مدل بدون شِما، اجازه می‌دهد که داده‌های با مشخصات متغیر را بازنمایی کنید. برای نمونه، تصور کنید در حال ساخت یک کاتالوگ تجارت الکترونیک هستید. راهی وجود ندارد که بدانید در آینده محصول شما چه صفاتی ممکن است داشته باشد، بنابراین برنامه باید برای تغییرات آماده باشد.

روش سنتی مدیریت این نیاز با پایگاه داده با شِمای ثابت که از الگوی موجودیت–صفت–مقدار استفاده می‌کند، در شکل ۱–۲ نمایش داده شده است.

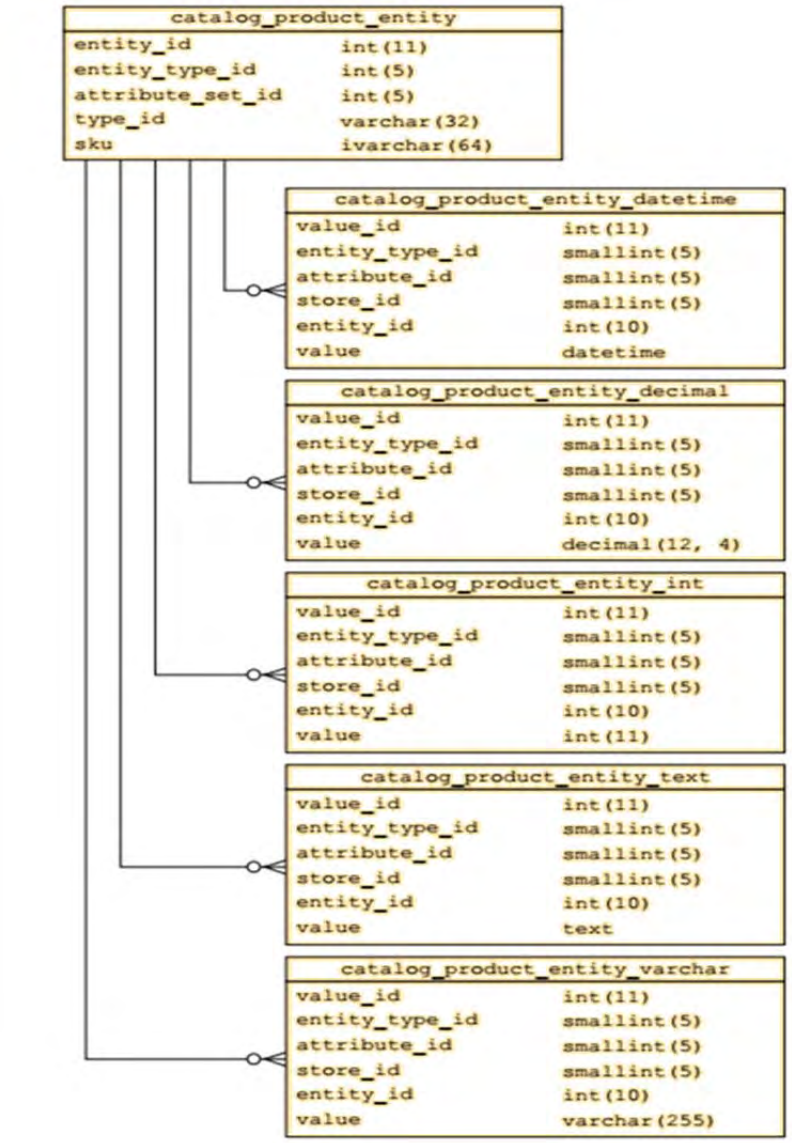
آنچه که می‌بینید یک بخش از مدل داده‌ای یک چارچوب تجارت الکترونیک است. به مجموعه جدول‌هایی که در اساس همه آنها یکسان هستند و تنها نوع داده آنها متفاوت است، توجه کنید. این ساختار به یک مدیر اجازه می‌دهد تا انواع محصولات و صفات مختلفی برای آنها تعریف کند اما نتیجه کار، به طور چشم‌گیری پیچیده است. تصور کنید پوسته MYSQL را برای آزمایش یا به‌روز رسانی یک محصول مدل شده به این روش، فعال کنیم. پیوندهای SQL مورد نیاز برای بازسازی محصول بسیار پیچیده خواهند بود، اما در مدل‌سازی به صورت سند هیچ پیوندی لازم نیست.

هرچند، تمام مدل‌های رابطه‌ای، این پیچیدگی را ندارند ولی نکته اینجاست که وقتی با استفاده از MongoDB برنامه‌ای را تولید می‌کنید، لازم نیست نگران فیلدهای داده‌ای باشید که در آینده به آنها نیاز خواهید داشت.

۱–۲–۲ کوئرهای تخصصی

برای اینکه بگوییم سیستمی از کوئری‌های تخصصی پشتیبانی می‌کند باید نشان دهیم که لازم نیست از ابتدا مشخص کنیم چه نوع کوئری‌هایی در آینده قابل پذیرش خواهند بود. پایگاه داده‌ای رابطه‌ای این خصوصیات را دارند. آنها هرگونه کوئری SQL خوش ساختی را با هر تعداد شرط، به خوبی اجرا می‌کنند. کوئری‌های تخصصی می‌توانند در هر پایگاه داده رابطه‌ای اجرا شوند اما تمام پایگاه داده‌ها از کوئری‌های پویا پشتیبانی نمی‌کنند. برای نمونه ذخیره‌سازهای کلید–مقدار توان کوئری نویسی بالا را قربانی قابلیت گسترش آسان مدل می‌کنند. یکی از اهداف طراحی MongoDB این است که توان کوئری‌نویسی را در حد دنیای پایگاه داده رابطه‌ای نگه دارد.

برای مشاهده اینکه زبان کوئری‌نویسی MongoDB چگونه کار می‌کند، اجازه دهید مثال ساده‌ای را ارائه دهیم که شامل posts و comments می‌شود. فرض کنید می‌خواهید تمام پست‌هایی را بیابید که با بر چسب politics بیش از ۱۰ رأی آورده‌اند.



شکل ۱-۲- بخشی از شمای یک برنامه تجارت الکترونیک. این جدول‌ها تعریف صفات پویا برای محصولات را آسان‌تر می‌کنند.

یک کوئری SQL می‌تواند چیزی مانند این باشد:

```
SELECT * FROM posts
  INNER JOIN posts_tags ON posts. id = posts_tags. post_id
  INNER JOIN tags ON posts_tags. tag_id == tags. id
  WHERE tags. text = 'politics' AND posts. vote_count > 10;
```

کوئری معادل این کوئری در MongoDB با استفاده از یک سند به‌عنوان عامل مقایسه به صورت زیر خواهد بود. کلید $gt برای نمایش شرط "بزرگ‌تر از" استفاده می‌شود.

```
db. posts. find ({'tags': 'politics', 'vote_count': {'$gt': 10}}) ;
```

توجه کنید که دو کوئری براساس دو مدل داده‌ای متفاوت ساخته شده‌اند. کوئری SQL با تکیه بر مدل نرمال‌سازی بنا شده است که در آن posts و tags در جدول‌های مستقل ذخیره شده‌اند، در حالی که کوئری MongoDB بر این فرض بنا شده است که tags و posts در یک سند ذخیره شده‌اند. با این وجود هر دو کوئری توان کوئری‌سازی برای هر ترکیبی از صفات را نمایش می‌دهند.

۱-۲-۳- شاخص‌ها (Index)

عنصر بحرانی کوئری‌های تخصصی این است که دنبال مقادیری می‌گردند که هنگام ایجاد پایگاه داده از آنها بی‌اطلاع هستید. اگر شمار زیادی سند به پایگاه داده خود بیافزایید، هزینه جست‌وجوی یک مقدار افزایش می‌یابد. بنابراین نیاز به روشی برای جست‌وجوی با کارایی بالا در میان داده‌ها دارید. راه حل این مشکل، ایندکس یا همان شاخص است.

بهترین راه برای درک شاخص‌های پایگاه داده‌ها، مقایسه است: بیشتر کتاب‌ها شاخص‌هایی (فهرست‌هایی) دارند که کلیدواژه‌ها را به صفحات کتاب تطبیق می‌دهند. اینک فرض کنید یک کتاب آشپزی دارید و می‌خواهید تمام دستورهای غذایی را پیدا کنید که در آنها از گوشت استفاده شده است. راه زمان‌بر این است که صفحه به صفحه در دستورها دنبال گوشت بگردید، اما بیشتر افراد ترجیح می‌دهند که به ایندکس پایان کتاب مراجعه کرده و با رفتن به مدخل گوشت، لیستی از تمام دستورهایی را بیابند که در آنها از گوشت استفاده شده است. شاخص‌های پایگاه داده‌ها، ساختارهایی مانند جداول ایندکس یا همان واژه‌یاب موجود در پایان کتاب‌ها دارد.

شاخص‌ها در MongoDB به صورت ساختار داده B-tree پیاده‌سازی می‌شود. شاخص‌های B-tree که در بیشتر پایگاه داده‌های رابطه‌ای نیز استفاده می‌شود، برای انواعی از کوئری‌ها شامل کوئری‌های پویش کننده محدوده و کوئری‌های با کلاز sort بهینه‌سازی شده‌اند. اما wired Tiger از درخت‌های ادغام ثبت ساختار (LSM) پشتیبانی می‌کند.

بیشتر پایگاه داده‌ها به هر سند یا سطر، یک کلید اصلی اختصاص می‌دهند که یک شناسه برای آن داده است. معمولا کلید اصلی به صورت خودکار شاخص گذاری می‌شود؛ به گونه‌ای که هر داده می‌تواند با کارایی بالا به وسیله این کلید یکتا مورد دسترسی قرار گیرد که MongoDB نیز از این روش استفاده کرده است. البته تمام پایگاه داده‌ها به شما اجازه نمی‌دهند که داده‌های داخل یک سطر یا سند را شاخص گذاری کنید. این نوع شاخص‌ها، شاخص‌های ثانویه نامیده می‌شوند. شماری از پایگاه داده‌های NOSQL مانند HBASE به فکر ذخیره کلید-مقدار افتاده‌اند، زیرا آنها اجازه شاخص‌گذاری ثانویه را نمی‌دهند. شاخص‌گذاری ثانویه یکی از مهم‌ترین خصوصیات MongoDB است. با اجازه دادن به چندین شاخص ثانویه، MongoDB به کاربران اجازه می‌دهد تا انواع کوئری‌های خود را بهینه‌سازی کنند.

با MongoDB می‌توانید تا ٦٤ شاخص برای هر مجموعه داشته باشید. انواع شاخص‌هایی که پشتیبانی می‌شوند، همان‌هایی هستند که در RDBMSها پشتیبانی می‌شوند. همه‌ی انواع شاخص‌های صعودی، نزولی، یکتا، کلید مرکب، درهم سازی شده متنی و حتی فضایی، در MongoDB پشتیبانی می‌شوند.

چون MongoDB و RDBMSها از یک ساختار داده‌ای برای شاخص‌های خود استفاده می‌کنند، توصیه‌های مدیریتی شاخص‌ها در هر دو نوع سیستم یکسانند. در ادامه، نگاهی به شاخص‌ها خواهیم داشت و چون آموختن شاخص‌ها برای ساخت پایگاه داده‌ی کارا بسیار مهم است، فصل ۸ به این موضوع اختصاص دارد.

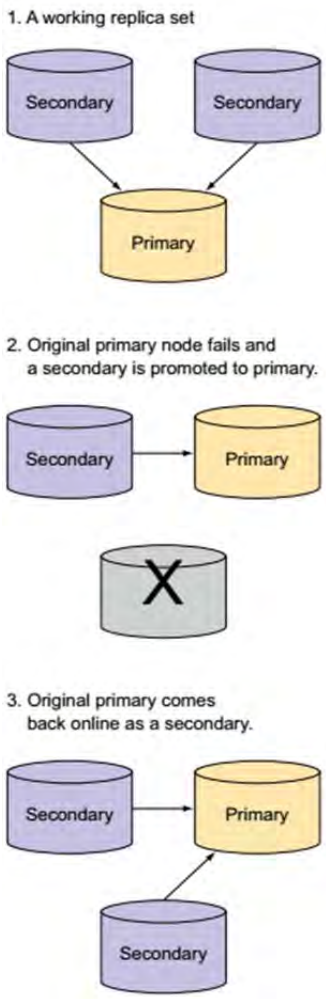
۱-۲-۴- تکرار (Replication)

MongoDB امکان تکرار از طریق یک توپولوژی به نام مجموعه تکرار را فراهم می‌آورد. مجموعه‌های تکرار، داده‌های توزیع شده‌ای از دو یا چند ماشین هستند که برای داشتن افزونگی و کنترل خطای خودکار، در مواقع در رویدادهای غیرمنتظره برای سرور و قطعی برق استفاده می‌شوند. افزون بر این، تکرار برای گسترش قابلیت خواندن از پایگاه داده نیز استفاده می‌شود.

اگر برنامه‌ای با شمار خواندن زیاد داشته باشید که معمولا در وب چنین است، این امکان وجود دارد که خواندن‌های پایگاه داده‌ها را روی ماشین‌های خوشه‌ی مجموعه‌ی تکرار، پخش کنید.

مجموعه‌های تکرار، از تعدادی سرور MongoDB تشکیل شده‌اند که معمولا هر سرور روی یک ماشین فیزیکی جداگانه است که آنها را گره می‌نامیم. در هر زمان مشخصی، یک گره به گره اصلی

مجموعه تکرار سرویس می‌دهد. مانند تکرار master-slave که ممکن است در پایگاه داده‌های دیگر با آن آشنا شده باشید. گره اصلی مجموعه تکرار می‌تواند هم خواندن و هم نوشتن‌ها را بپذیرد ولی گره‌های ثانویه، تنها قابلیت خواندن دارند. آنچه مجموعه‌های تکرار را ویژه می‌کند، پشتیبانی آنها از کنترل خطای خودکار است: اگر گره اصلی با خطا مواجه شود، خوشه یکی از گره‌های ثانویه را گرفته و به صورت خودکار آن را به گره اصلی ارتقاء می‌دهد و وقتی گره اصلی به سیستم باز می‌گردد، مانند یک گره ثانویه رفتار می‌کند. شکل ۱-۳ این فرآیند را نمایش می‌دهد. تکرار یکی از مفیدترین خصوصیات MongoDB است و در ادامه کتاب با ژرفای بیشتر بررسی خواهد شد.



شکل۱-۳-کنترل خودکار خطا با replica

۱-۲-۵-سرعت و پایایی

برای درک درست رویکرد MongoDB در قبال پایایی، نخست باید به چند ایده اولیه توجه کرد. در قلمروی سیستم‌های پایگاه داده، یک نسبت عکس میان سرعت نوشتن و پایایی وجود دارد. سرعت نوشتن می‌تواند به‌عنوان تعداد درج‌ها، به‌هنگام‌سازی‌ها و حذف‌هایی درنظر گرفته شود که می‌توانند در یک بازه زمانی پردازش شوند. پایایی به سطحی از اطمینان گفته می‌شود که این عمل‌های نوشتن را ماندنی می‌کند.

برای نمونه فرض کنید ۱۰۰ رکورد ۵۰ کیلوبایتی را در پایگاه داده می‌نویسید و سپس بی‌درنگ برق سرور را قطع می‌کنید. آیا این رکوردها پس از بالا آمدن بعدی سرور قابل بازیابی هستند؟ پاسخ به سیستم پایگاه داده، پیکربندی آن و سخت‌افزار شما بستگی دارد. بیشتر پایگاه داده‌ها به صورت پیش فرض پایایی خوبی دارند و اگر چنین رویدادی رخ دهد خیال شما آسوده است. برای برخی برنامه‌ها، مانند ذخیره کننده‌های ثبت خطوط، داشتن سرعت نوشتن بیشتر، مهم‌تر از خطر از دست دادن داده‌هاست. مشکل اینجاست که نوشتن به یک هارد دیسک مغناطیسی بسیار کندتر از نوشتن در RAM است: برخی پایگاه داده‌ها مانند Memcached به طور انحصاری روی RAM می‌نویسند که آنها را بسیار سریع می‌کند ولی داده‌های آنها فرار است. از سوی دیگر شمار کمی از پایگاه داده‌ها به طور انحصاری روی دیسک می‌نویسند زیرا کارایی پایین آن قابل پذیرش نیست. بنابراین اغلب لازم است که طراحان پایگاه داده میان سرعت و پایایی، بهترین مصالحه را انجام دهند.

ثبت تراکنش‌ها

یک مصالحه میان سرعت و پایایی در موتور InnoDB از MYSQL قابل مشاهده است. InnoDB یک موتور ذخیره‌سازی تراکنشی است که بر حسب تعریف باید از پایایی برخوردار باشد. این موتور با استفاده از نوشتن به‌هنگام‌سازی‌ها در دو جا، از عهده این کار بر می‌آید: یکی در فایل ثبت‌تراکنش و دیگری در محل بافر درون حافظه‌ای. ثبت تراکنش‌ها بی‌درنگ با دیسک همگام می‌شود، در حالی که بافر تنها در مواقع لزوم با نخ‌های پس‌زمینه همگام می‌شود. دلیل این نوشتن دوگانه این است که خواندن و نوشتن تصادفی بسیار کندتر از خواندن و نوشتن سریال است. چون نوشتن در فایل داده اصلی، از خواندن و نوشتن تصادفی تشکیل شده است، با هر بار نوشتن، این تغییرات نخست در RAM نوشته می‌شود و اجازه می‌یابد که دیرتر با دیسک همگام شود. اما برخی از انواع نوشتن روی دیسک لازم است که پایایی را تأمین کنند و برای آنها مهم است که نوشتن آنها سریال و سریع باشد و این چیزی است که فایل ثبت تراکنش فراهم می‌کند. در هنگام خاموش شدن ناسالم، InnoDB

می‌تواند فایل ثبت تراکنش را تکرار کرده و داده‌های اصلی فایل‌های مربوطه را به‌هنگام‌سازی کند. این روش یک سطح قابل قبولی از کارایی را فراهم می‌کند و از سویی، پایایی سطح بالایی را تأمین می‌کند.

در MongoDB، کاربران مصالحه سرعت و پایایی را با انتخاب قواعد نوشتن و تصمیم گیری درباره فعال کردن journaling کنترل می‌کنند. Journaling تا MongoDBv2.0 به صورت پیش فرض فعال می‌شد. در راه‌اندازهای پس از نوامبر ۲۰۱۲، MongoDB به صورت پیش فرض تضمین می‌کند که نوشتن پیش از بازگشت برای کاربر در RAM نوشته می‌شود، هر چند این خصوصیت قابل پیکربندی است. می‌توانید MongoDB را به صورت fire-and-forget پیکربندی کنید که در این حالت نوشتن را بدون انتظار برای تأیید، به سرور ارسال می‌کند. همچنین می‌توان MongoDB را طوری پیکربندی کرد که تضمین کند نوشتن بدون نگرانی از کامل شدن، به replicaها منتقل شود. برای داده‌های با تعداد زیاد و مقادیر کم (مانند رشته کلیک و ثبت رویدادها) روش fire-and-forget می‌تواند ایده‌آل باشد. برای داده‌های با اهمیت بالا تنظیمات امن‌تری لازم است. مهم است بدانیم که در نسخه‌های پیش از MongoDB2.0 استراتژی ناامن fire-and-forget به صورت پیش فرض در نظر گرفته شده بود، زیرا وقتی 10gen شروع به ساخت MongoDB کرد تنها بر روی داده‌های وب کار می‌کرد، اما وقتی MongoDB برای موارد بیشتر و بیشتر مورد استفاده قرار گرفت، متوجه شد که این روش برای داده‌های با اهمیت بسیار ناامن است.

از MongoDB2.0 به بعد journaling به صورت پیش فرض فعال شد. با journaling نوشتن‌ها از هر ۱۰۰ میلی ثانیه به فایل journal نوشته می‌شوند.

اگر سرور ناسالم خاموش شود (مثلا بر اثر قطعی برق) فایل journal جامعیت داده‌ای ما را پس از شروع به کار دوباره سیستم تضمین می‌کند. این روش، امن‌ترین راه استفاده از MongoDB است.

گاهی ممکن است ما سرور را به خاطر افزایش کارایی، بدون journaling راه‌اندازی کنیم. روی دیگر سکه این است که در چنین حالتی ممکن است فایل‌های داده، پس از خاموش شدن ناسالم سیستم، خراب شوند. نتیجه می‌گیریم که اگر کسی تصمیم بگیرد ژورنالینگ را غیر فعال کند، بهتر است که یک مرکز داده دیگر داشته باشد تا بتواند داده‌های اولیه دست نخورده داشته باشد.

MongoDB طوری طراحی شده که برای شما گزینه‌هایی برای مصالحه میان سرعت و پایایی فراهم کند، اما پیشنهاد ما این است که برای داده‌های مهم از تنظیمات امن استفاده کنید. موضوعات تکرار و

پایایی بسیار گسترده‌اند و برای اطلاعات بیشتر در این زمینه می‌توانید به کتاب‌های تخصصی‌تر و مسندات MongoDB مراجعه کنید.

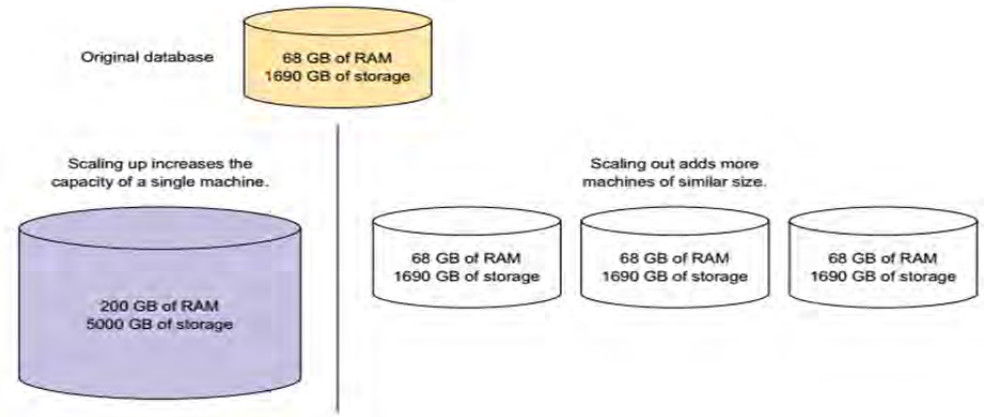
۱-۲-۶- قابلیت گسترش

یکی از ساده‌ترین راهها برای گسترش بیشتر پایگاه داده‌ها، ارتقای سخت‌افزار است. اگر برنامه شما روی یک گره اجرا می‌شود، می‌توانید ترکیبی از دیسک‌های سریع، حافظه بیشتر و پردازنده قوی برای برطرف کردن محدودیت‌های آن استفاده کنید. تکنیک افزودن سخت‌افزار به یک گره برای گسترش آن، گسترش عمومی یا گسترش به بالا نامیده می‌شود. گسترش عمومی امتیازهایی همچون سادگی، قابلیت اطمینان و کم هزینه بودن را تا یک نقطه معلوم دارد ولی زمانی به نقطه‌ای می‌رسید که امکان انتخاب سیستم بهتری را ندارید.

آنگاه باید به گسترش افقی یا گسترش به بیرون فکر کنید (شکل۱-٤ را ببینید). در این روش به جای افزایش قدرت یک گره، پایگاه داده‌ها روی چند ماشین توزیع می‌شود. یک معماری گسترش یافته افقی می‌تواند روی شمار زیادی ماشین کوچک و ارزان اجرا شود که معمولا باعث کاهش هزینه‌ها می‌شود. به جز این، توزیع داده‌ها بر روی ماشین‌های مختلف، هزینه خرابی سیستم‌ها را کمتر می‌کند. خرابی گاه به گاه ماشین‌ها گریزناپذیر است. در حالت گسترش عمومی، وقتی با خرابی سیستمی مواجه می‌شویم، حتی اگر نسخه دیگری از داده‌ها را نیز داشته باشیم، باز مشکل پیش آمده موجب تهدید کلیت سیستم‌ها می‌شود اما در حالت گسترش افقی، مصیبت از کار افتادن یک ماشین کمتر است، زیرا تنها درصدی از داده‌های شما در معرض تهدید هستند.

MongoDB طوری طراحی شده است تا بتواند گسترش افقی را مدیریت کند و این کار را از طریق یک مکانیزم پارتیشن‌بندی مبتنی بر محدوده انجام می‌دهد که sharding نامیده می‌شود. این مکانیزم به صورت خودکار، توزیع داده‌ها روی گره‌ها را مدیریت می‌کند.

سیستم sharding افزودن گره‌های shard شده را مدیریت می‌کند و امکاناتی برای کنترل خطای خودکار فراهم می‌آورد.



شکل ۱-۴- گسترش افقی در برابر گسترش عمودی

۱-۳- سرور اصلی و ابزارهای MongoDB

MongoDB به زبان C++ نوشته شده است و هنوز هم ساخت آن توسط شرکت MongoDB فعالانه ادامه دارد. پروژه روی بیشتر سیستم عامل‌های اصلی مانند Mac OS X، Windows، solaris و بیشتر نسخه‌های Linux کامپایل می‌شود و البته فایل‌های دودویی از پیش کامپایل شده قابل نصب برای بیشتر این سیستم عامل‌ها نیز موجود است. MongoDB متن باز است و تحت لیسانس AGPL می‌باشد. سورس برنامه به صورت رایگان در GitHub موجود است و تقاضای همکاری با جامعه MongoDB معمولا پذیرفته می‌شود ولی پروژه توسط شرکت MongoDB هدایت می‌شود.

MongoDB v1.0 در نوامبر ۲۰۰۹ منتشر شد. نسخه‌های پایدار برنامه، هر سه ماه یک بار منتشر می‌شود و اکنون آخرین نسخه برنامه، MongoDB 3.2.8 است.

۱-۳-۱- سرور مرکزی

سرور پایگاه داده مرکزی از یک فایل اجرایی به نام mongod (یا MongoDB.exe در ویندوز) اجرا می‌شود. فرآیند سرور mongod دستورها را از یک سوکت شبکه از یک پروتکل دودویی ویژه دریافت می‌کند. تمام فایل‌های فرآیند mongod به صورت پیش فرض در مسیر /data/db لینوکس و مسیر C: \data\db در ویندوز ذخیره می‌شوند. بیشتر مثال‌های این کتاب بیشتر مبتنی بر لینوکس می‌باشند. بیشتر سرورهای MongoDB به خاطر قابلیت اطمینان، تطبیق‌پذیری بالا و ابزارهای مناسب روی لینوکس اجرا می‌شوند.