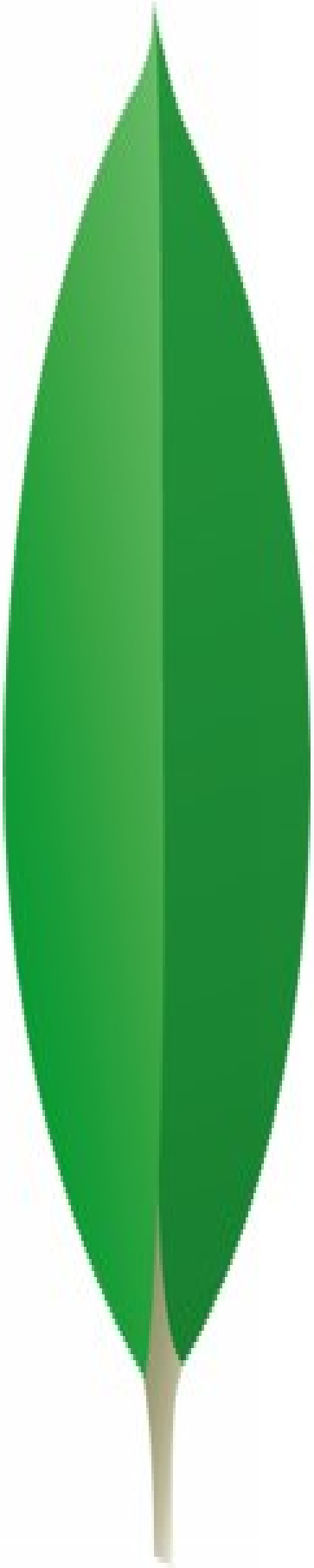

The Little MongoDB Book

by Karl Seguin

now updated for 2.6

The Little MongoDB Book

Karl Seguin

About This Book

License

စာရေးသူ အကြောင်း

Karl Seguin သည် နည်းပညာနှင့်ပတ်သတ်သော ဘာသာရပ်များတွင် အတွေ့အကြုံများစွာရှိသည့် developer တစ်ဦးဖြစ်သည်။ ၎င်းသည် .Net နှင့် Ruby ကျွမ်းကျင်သော developer တစ်ဦးဖြစ်သည့် အပြင် OSS Projects များကို တစ်စိတ်တစ်ပိုင်း contributor တစ်ဦးဖြစ်သည့် အပြင် နည်းပညာအကြောင်း ဟောပြောသူ နှင့် စာရေးသူ တစ်ဦးလည်း ဖြစ်သည်။ MongoDB နှင့်ပတ်သတ်၍လည်း C# ၏ MongoDB library ဖြစ်သော

NoRM ၏ အဓိက contributor တစ်ဦးဖြစ်ပြီး mongly နှင့် Mongo Web Admin တို့နှင့်ပတ်သတ်သော tutorial များကိုလည်းလည်း တွင်ရေးသားပါသေးသည်။ ၎င်း၏ game developer များအတွက် အခမဲ့ service ကို mogade.com တွင်တွေ့နိုင်ပြီး MongoDB ကိုအသုံးပြုထားသည်။

Karl သည် The Little Redis Book ကိုလည်းရေးသားခဲ့ပါသေးသည်။

သူ၏ blog ကို http://openmymind.net တွင်တွေ့နိုင်ပြီး ၎င်း၏ twitter handle မှာ [@karlseguin](http://twitter.com/karlseguin) ဖြစ်သည်။

ကျေးဇူးတင်လွှာ

သင်၏ မျက်လုံး စိတ်နှင့် ပြင်းထန်သော ဝါသနာများကို ငှားရမ်းမှုအတွက် Perry Neal ကို အထူးပဲကျေးဇူးတင်ရှိရပါတယ်။ သင့်ရဲ့ကူညီမှုက တန်ဖိုးဖြတ်၍မရပါဘူး။ ကျေးဇူးပါ။

http://github.com/karlseguin/the-little-mongodb-book.

နိဒါန်း

အခန်းတွေ တိုတိုလေးဖြစ်တာ ကျွန်တော် အပြစ်မဟုတ်ပါဘူး။ MongoDB ကလေ့လာရ လွယ်လို့ပါ။

နည်းပညာတွေက အရွှေ့တွေက မြန်သည်ဟု ပြောကြသည်။ တနေ့တနေ့ နည်းပညာ အသစ်တွေနှင့် technique အသစ်တွေပေါ်လာသည်က မှန်သည်။ သို့သော်လည်း programmer တွေအသုံးပြုသော အခြေခံနည်းပညာများမှာမူ ဖြေးဖြေးချင်းစီသာ ပြောင်းလဲပါသည်။ တစ်နှစ်တစ်နှစ်တွင် အနည်းငယ်မျှသာ လေ့လာသောလည်း အလုပ်ဖြစ်နိုင်ပါသည်။ မြန်သည်ကတော့ နည်းပညာအဟောင်းများ နေရာတွင် အစားထိုးမည့် အသစ်များပါ။ တနေ့ထဲတွင် အချိန်ကြာမြင့်စွာတည်ဆောက်ထားသော နည်းပညာများအသစ် deveoper များ၏ အပြောင်းအလဲကြောင့်ကို ကြောက်ရွံ့ရသည်။

နဂိုအသားကျနေသော relational database များမှ NoSQL နည်းပညာများသို့ ရုတ်တရက်အရွှေ့သည် ၎င်းကို ဖော်ပြနိုင်သော အကောင်းဆုံး ဥပမာဖြစ်သည်။ တနေ့တွင် web တစ်ခုလုံးသည် relational database များနှင့် NoSQL solution ငါခုလောက်ဖြင့်သာ အသုံးပြုကြတော့မည် ဟုထင်ခဲ့ကြသည်။

ၚင်း အပြောင်းအလဲများသည် နေ့ချင်းရက်ချင်းဖြစ်သော်လည်း တကယ်လက်တွေ့တွင် ထိုနည်းပညာများသည် တကယ့်လက်တွေ့တွင် သုံးနိုင်သည့် အခြေအနေရောက်အောင် နှစ်ပေါင်းများစွာကြာမြင့်ခဲ့သည်။ အစောပိုင်းတွင် developer အနည်းငယ်မျှသည် စိတ်အားထက်သန်မှုဖြင့် မောင်းနှင်ခဲ့ပြီး ပို၍ အဆင်ပြေလာသည်နှင့်အမျှ သင်ခန်းစာများရက နည်းပညာအသစ်၏ နေရာကိုရှာတွေ့ခဲ့ပြီး အခြားသူများမှာ တဖြည်းဖြည်း ၚင်းတို့အတွက် စမ်းကြည့်လာကြသည်။ ထပ်၍ ထုံးတမ်းစဉ်လာ storage solution များအနေဖြင့် အစားထိုး၍မရသော် နေရာများအတွက် NoSQL ၏ အနေအထားသည် မှန်သည်ဟုဆိုရမည်ဖြစ်ပြီး ထုံးတမ်းစဉ်လာ feature များထက် ဘယ်အချက်တွေ ပိုလာမလဲကိုသာ အဓိကပြောရမည်ဖြစ်သည်။

အပေါ်မှအတိုင်း ဆိုခဲ့ပြီးပါနောက် NoSQL ဆိုသည်မှာ ဘာလဲဆိုတာ ရှင်းပြရန်လိုမည်။ ၚင်းသည် ကွဲပြားသောသူများအတွက် ကွဲပြားသော အဓိပ္ပါယ်ဖွဲ့ဆိုချက်များ ဖြစ်သည်။ ကျွန်တော်အနေဖြင့်မူ data များ၏ သိမ်းဆည်းပုံနှင့် ပတ်သတ်၍ အဓိကကျသော နေရာတစ်ခုအဖြစ်ပါဝင်သော စနစ်တစ်ခုဟု အကြမ်းဖြင်းမှတ်ယူထားသည်။ တနည်းအားဖြင့် NoSQL သည် (ကျွန်တော်အတွက်) ခိုင်လုံသော စနစ်တစ်ခုသည် စက်တစ်လုံးထဲပေါ်တွင် မမူတည်နေသော ယုံကြည်ချက်ဖြစ်သည်။

ပုံမှန် relational database များ၏ ထုတ်လုပ်သူများသည် တောက်လျှောက် software တစ်ခုကို အားလုံးစုပြုံထားသော solution တစ်ခုအနေဖြင့် ပုံ

သွင်းရန်ကြိုးစားခဲ့သော်လည်း NoSQL အတွက်မူ သေးငယ်သော units များမှ တာဝန်အသီးသီးခွဲယူပြီး အကောင်းဆုံး tool များကို အသုံးပြု၍ တာဝန်တစ်ခုစီကို ထမ်းယူနိုင်ရန် ရည်ရွယ်သည်။ ထိုကြောင့် NoSQL stack တစ်ခုတွင် relational database နှင့်တွဲဖက်၍အသုံးပြုခြင်း ဆိုပါစို့ MySQL ကိုအသုံးပြုပေမယ့် Redis ကို စနစ်၏ အစိတ်အပိုင်းတချို့တွင် အသုံးပြုခြင်းနှင့် အကြီးအကျယ် data process ပြုလုပ်ပါက Haddop ကို အသုံးပြုခြင်းကဲ့သို့ပင် ရိုးရှင်းစွာပြောပါက NoSQL သည် အခြားအစားထိုး တစ်ခုအတွက်ကို ဖွင့်လှစ်ထားခြင်း ၊ ရှိပြီးသား နှင့် pattern များနှင့် tool များကိုအသုံးပြု၍ data များကို ကွပ်ကဲခြင်းဖြစ်သည်။

MongoDB သည် ၎င်းတို့အားလုံးကို ဖြေရှင်းနိုင်မလား ဟု တွေးကောင်းတွေးလိမ့်မည်။ document-oriented database တစ်ခု အနေဖြင့် MongoDB သည် အထွေထွေဆန်သည် NoSQL solution တစ်ခုဖြစ်သည်။ ၎င်းကို relational datbase များ၏ အစားထိုး ဟုမြင်နိုင်သည်။ relational database များကဲ့သို့ပင် တခြား NoSQL solution များနှင့် တွဲဖက်အသုံးပြုနိုင်သည်။ MongoDB တွင်အားသာချက်ကော အားနည်းချက်ပါရှိပြီး ၎င်းတို့ကို စာအုပ်၏ အခြားသောအပိုင်းများတွင် ဖော်ပြသွားမည်။

အစပျိုး

ဒီစာအုပ်၏ အများစုသည် MongoDB ၏ function များကို အာရုံစိုက်ထားမည်ဖြစ်၍ MongoDB Shell အပေါ်မှာ run မည်ဖြစ်သည်။ shell သည် administration ပြုလုပ်သည့် အပိုင်းများတွင် အသုံးဝင်သည်ဖြစ်သော်လည်း သင့် Code မှာမူ MongoDB driver ကိုအသုံးပြုမည်ဖြစ်သည်။

ထိုကြောင့် MongoDB နှင့်ပတ်သတ်၍ ပထမဆုံးသိရန်မှာ ၎င်း၏ driver များဖြစ်သည်။ MongoDB တွင် Language များအတွက် official drivers များစွာ ပါရှိသည်။ ၎င်း driver များကို အခြား သင်ရင်းနှီးပြီးသားဖြစ်သော် database driver များကဲ့သို့ မှတ်ယူနိုင်သည်။ ထိုအပြင် development community သည် language နှင့် framework အခြေပြု library များကို တည်ဆောက်ထားသည်။ ဥပမာ fluent-mongo ကဲ့သို့သော C# library သည် LINQ အတွက်ပါ support ပေးပြီး MongoMapper သည် Ruby အခြေပြု library ဖြစ်ပြီး ActiveRecord နှင့်နီးစပ်သည်။ သင့်အနေဖြင့် core MongoDB driver များကိုအခြေပြုပြီးရေးသားသည် ဖြစ်စေ အပေါ် တစ်ထပ်မှ library တစ်ခုခုအသုံးပြုသည်ဖြစ်စေ သင့်အပေါ်သာမူတည်သည်။ ထိုသို့ပြောရခြင်းမှာ MongoDB ကိုစတင်လေ့လာသူများသည် offical driver များနှင့် community library များကိုမြင်တွေ့ပြီး မျက်စိရှုပ်လေ့ရှိတက်သောကြောင့်ဖြစ်သည်။ ပထမတစ်ခုမှာ MongoDB ၏

communication နှင့် connectivity အပိုင်းကို အာရုံစိုက်ပြီး ဒုတိယတစ်ခုမှာ language နှင့် framework ဘက်မှ ချဉ်းကပ်ခြင်းဖြစ်သည်။

ဒီစာအုပ်ကိုဖတ်နေရင်း MongoDB နှင့် ကွျန်တော် ပြတဲ့ဟာတွေကို လိုက်စမ်းကြည့်ဖို့နှင့် မေးခွန်းတွေ ကိုယ်ဖာသာမေးကြည့်ဖို့ အားပေးပြါရစေ။

MongoDB တွင် Community version နှင့် Enterprise version ဟူ၍နှစ်မျိုးရှိသည်ဖြစ်ပြီး စတင်လေ့လာသူများအတွက် community version ကို အသုံးပြုရမည်ဖြစ်သည်။ ထိုအပြင် MongoDB ကို Install ပြုလုပ်ရာတွင် Install ပြုလုပ်နည်းသုံးနည်းရှိပါသည်။

1. သက်ဆိုင်ရာ Package Manager များမှ install ပြုလုပ်ခြင်း
2. Installable Package များမှ click နှိပ်ပြီး install ပြုလုပ်ခြင်း (ဥပမာ .msi,.deb)
3. Binary folder ကို download ပြုလုပ်၍ ညွှန်းဆိုအသုံးပြုခြင်း

ၚင်းတို့အနက် မိမိတို့ အသုံးပြုသည် OS သည် windows မဟုတ်ပါက Package Manager မှ install ပြုလုပ်ရန်အကြံပြုလိုသည်။ တကယ်လက်တွေ့တွင် deploy ပြုလုပ်သော server များတွင် ထိုနည်းများကို အသုံးပြုရမည် ဖြစ်သည်။ ထိုအပြင် upgrade ပြုလုပ်သည့်တာဝန်များကိုလည်း package manager များမှ တာဝန်ယူထားသဖြင့် ပိုမိုလွယ်ကူပါသည်။ ထိုနောက် မိမိတို့ အသုံးပြုမည့် Operation System ပေါ်မူတည်၍

အောက်ဖော်ပြပါလင့်များ အတွင်းသွား၍ install ပြုလုပ်ကြပါ။ (Linux) (Windows) (Mac)

`mongo` shell ကိုအသုံးပြုနိုင်ပါက အရင်ဆုံး `db.version()` ဟုရိုက်ထည့်ရင်း မိမိတို့အသုံးပြုသည့် database ၏ version ကိုကြည့်နိုင်ပါသည်။

အခန်း(၁) အခြေခံ

ပထမဆုံးခြေလှမ်းကိုတော့ MongoDB မည့်သို့အလုပ်လုပ်သနည်း ဆိုသည့် အခြေခံကိုပြောရင်းဖြင့် စပါမည်။ ၎င်းသည် MongoDB ကိုနားလည်ရာတွင် အဓိကအကျဆုံးအချက်ဖြစ်ပြီး MongoDB သည် မည်သို့သော အခြေအနေမျိုးနှင့် ကိုက်ညီသည် ဆိုသည့် မေးခွန်းများကို ဖြေရာတွင်လည်း အထောက်အကူပြုသည်။

ရှေးဦးစွာ အချက် ၆ ခုကို နားလည်ထားရန်လိုသည်။

1. MongoDB သည် အခြားသော database များကဲ့သို့ `database` ဟုသော concept အပေါ်တွင် အခြေခံသည်။

2. database တစ်ခုတွင် `collection` များပါရှိပြီး collection များသည် `table များနှင့်ဆင်၍ နှစ်ခုစလုံး အတူတူပဲဟု အစောပိုင်းမှတ်ယူနိုင်သည်။

3. Collection များသည် `document` များနှင့်ဖွဲ့စည်းထားပြီး ၎င်းတို့ကို `row` ကဲ့သို့ မှတ်ယူနိုင်သည်။

4. document တစ်ခုတွင် `field` များပါဝင်ပြီး ၎င်းတို့သည် `column` သဘောတရားနှင့် ဆင်တူသည်။

5. MongoDB တွင်ရှိသည့် `Index` များ၏ သဘောသဘာဝသည် RDBMS များနှင့်ဆင်သည်။

6. ၎င်းတွင်ပါရှိသော `Cursor` များသည် အခြား concept များနှင့်ကွဲပြားပြီး တခြားတရံချန်လှပ်ထားတက်သည်။ သို့သော် ၎င်းတို့ကို ဆွေးနွေးရန်ထိုက်တန်သည် ဟုမြင်ပါသည်။ အဓိကအချက်မှာ MongoDB မှ data ကိုလှမ်းယူလိုက်ပါက ပါသမျှ အကုန်ထုတ်ပေးသည်မဟုတ်ပဲ data များကို cursor ဟုခေါ်သည့် pointer တစ်ခုကို wrap လုပ်ပေးထားပြီ။ ၎င်းတို့ကို data တကယ်မဆွဲခင်ကတည်းက count ပြုလုပ်ရာတွင် ၎င်း ၊ ကျော်ရာတွင် ၎င်း အသုံးဝင်သည်။

အတိုချုပ်ပြောပါက MongoDB သည် `document` များပါဝင်သည့် `collection` များကိုစုစည်းထားသည့် `database` များနှင့်ဖွဲ့စည်းထားပြီး document တစ်ခုခြင်းစီတွင် `field` များပါဝင်သည်။ `collection` များကို ရှာဖွေရာတွင်နှင့် စီရာတွင် ပိုမိုမြန်ဆန်ကောင်းမွန်စေရန် `index` ပြုလုပ်နိုင်သည်။ နောက်ဆုံးတွင် mongodb မှ data များကို တိုက်ရိုက်ရသည်မဟုတ်ပဲ လိုအပ်မှ ဆွဲထုတ်ပေးသည့် `cursor` များမှ တဆင့်ရရှိသည်။

ဒါဆို ဘာလို့ စကားလုံးအသစ်တွေဖြစ်သည့် table အစား collection ၊ row အစား document ၊ column အစား field ဟု သုံးရသနည်း။ ပို၍ပ်အောင်လား ? အမှန်မှာ ၎င်း concept များသည် Relational Database များမှ

concept များနှင့် ဆင်သော်လည်း ထပ်တူမဟုတ်ပေ။ အဓိက ကွဲပြားချက်မှာ relational database များတွင် `column` များကို `table` အဆင့်တွင်တွင် သတ်မှတ်ပေးရပြီး document အထူးပြု database များတွင် `field` များကို `document` အဆင့်တွင် သတ်မှတ်ပေးရသည်။ ထို့ကြောင့် `collection` တစ်ခု အတွင်းရှိ `document` တစ်ခုချင်းစီသည် အမျိုးအစားမတူညီသော `field` များရှိနိုင်သည်။ ထိုကြောင့် `collection` သည် `table` နှင့်နှိုင်းစာလျင် ပေါ့ပေါ့လျော့လျော့ရှိပြီး `document` တွင် `row` ထက် အချက်အလက်ပိုများနိုင်သည်။

ငင်းကို နားလည်ရန် အရေးကြီးသော်လည် အခုလက်ရှိတွင် သိပ်မရှင်းသေးပါက ပြဿနာမရှိပါ။ ငင်းကို နားလည်ရန် data အထည့်အသွင်းအကြိမ် အနည်းငယ် ပြုလုပ်ပါက နားလည်သွားပါလိမ့်မည်။ အထူးသတိပြုရန်မှာ collection များသည် မည်သည့် အချက်အလက်အမျိုးအစား ထည့်သွင်းထားသည်ကို restrict ပြုလုပ်မည် မဟုတ်ပါ။ (schema-less ဖြစ်သော်ကြောင့်) ငင်း၏ အားသာချက်နှင့် အားနည်းချက်ကို နောက်အခန်းများတွင် ရှင်းပြသွားပါမည်။

စကြရအောင်။ သင့်အနေနဲ့ ဘာမှ မ run ရသေးပါက `mongod` ဆိုပြီး server ကိုစတင်လိုက်ပြီး mongo shell ထဲဝင်လိုက်ပါ။ ထို shell သည် Javascript ဖြင့် run ပြီး global command များဖြစ်သည့် `help` တို့ `exit` တို့ကိုလည်း အသုံးပြုနိုင်သည်။ လက်ရှိ database တစ်ခုလုံးစာ အရာများကို

manageလုပ်ရန် `db` object အတွင်းရှိ command များပါရှိသည်။ (ဥပမာ `db.help()`၊ `db.stats()`) သို့မဟုတ် လက်ရှိ collection တစ်ခုစာ အရာများကို execute လုပ်ရန်မူ `db.COLLECTION_NAME` ဟု object များပါရှိသည်။ (ဥပမာ `db.unicorns.help()`၊ `db.unicorns.count()`)

`db.help()` ဟု ရိုက်ပြီးရှာကြည့်ပါ။ သင့်အနေဖြင့် `db` object အတွင်းတွင် execute ပြုလုပ်နိုင်သော commands list ကျလာပါလိမ့်မည်။

မှတ်သားရန် တစ်ခုမှာ Javascript Shell ဖြစ်တာကြောင့် method တစ်ခုကို ရိုက်ထည့်ရာတွင် parenthese ဖြစ်သည့် `()` ကို ဖြုတ်ပြီး ရိုက်ထည့်ကြည့်ပါက method ကိုခေါ်မည့် အစား method body ကိုတွေ့ရမည်ဖြစ်သည်။ ပြောရသည်မှာ သင့်အနေဖြင့် ပထမဆုံး response တစ်ခုမှာ `function (...){` ဟုပုံစံဖြင့် လာပါက အရမ်း မအံ့သြရန်ဖြစ်သည်။ ဥပမာ `db.help` ဟုရိုက်ထည့်ပါက `help` method ၏ internal implementation ကိုတွေ့ရမည်ဖြစ်သည်။

ပထမဦးစွာ `use` ဆိုသည့် global helper ကိုသုံး၍ database များကို switch ပြုလုပ်နိုင်သည်။ ထိုကြောင့် `use learn` ဆိုပြီးရိုက်လိုက်ပါ။ database မရှိသေးလည်း ကိစ္စမရှိပါ။ ပထမဆုံး collection ကိုတည်ဆောက်လိုက်ပါက `learn` database ပါ တခါတည်း ဆောက်သွားမည်ဖြစ်သည်။ အခု database ၏ အတွင်းထဲကိုရောက်သွားပြီ ဖြစ်၍ database command

များဖြစ်သည့် `db.getCollectionNames()` ကိုသုံးနိုင်မည်ဖြစ်သည်။ အခု ရိုက်ကြည့်ပါက array အလွတ် (`[ ]`) ကိုသာတွေ့မည်ဖြစ်သည်။ collection များမှာ schema မရှိသဖြင့် တကူးတက ဆောက်ရန်မလိုပေ။ document ကို insert လုပ်လိုက်ပါက collection ဖြစ်ပေါ်လာမည်ဖြစ်သည်။ ထိုသို့ ပြုလုပ်ရန် `insert` command ကိုအသုံးပြုရမည်ဖြစ်ပြီး document ကို အောက်ပါအတိုင်း insert ပြုလုပ်လိုက်ပါ။

```
db.unicorns.insert({name: 'Aurora',
    gender: 'f', weight: 450})
```

parameter အနေဖြင့် pass ပြီး `unicorns` collection ကို `insert` ပြုလုပ်မည်ဖြစ်သည်။ MongoDB ၏ အတွင်းပိုင်းတွင် BSON ဟုခေါ်သည့် Binary serialized JSON format ကိုအသုံးပြုပြီး ထိုကြောင့် အပြင်ပိုင်းတွင် JSON ကို အဓိကအသုံးပြုသည်ကို ပြော၍ရပြီး ကျွန်တော်တို့ လက်ရှိဥပမာတွင် မြင်တွေ့နိုင်သည်။ အကယ်၍ ဒီတစ်ခါ `db.getCollectionNames()` ရိုက်ကြည့်ပါက `unicorns` ကိုမြင်တွေ့ရမည်ဖြစ်သည်။

ယခု `unicorns` အတွင်းရှိ document များကို `find` ကိုအသုံးပြု၍ ရှာဖွေနိုင်ပါပြီ။

```
db.unicorns.find()
```

သတိထားမိမည်မှာ မိမိတို့ထည့်ထားသည့် data များအပြင် `_id` ဟုသည့် field တစ်ခု အပိုပါနေသည်ကိုတွေ့ရမည်။ document တိုင်းတွင် သီးသန့်

`_id` field ပါရှိသည်။ သင့်အနေဖြင့် ကိုယ်ဖာသာကိုယ် generate ပြုလုပ်နိုင်သလို MongoDB မှ generate ပြုလုပ်ထားသည့် `ObjectId` ကိုလည်း အသုံးပြုနိုင်သည်။ အခြေအနေတော်တော်များများတွင် `ObjectId` ကို အသုံးပြုဖို့များပါသည်။ default အနေဖြင့် `_id` သည် index ပြုလုပ်ထားသည်။ ၎င်းကို `getIndexes` command ကိုအသုံးပြု၍ သိရှိနိုင်သည်။

```
db.unicorns.getIndexes()
```

index ၏ အမည်ကိုတွေ့ပါက database နှင့် collection နှင့် field တို့ပါ ထို index အတွင်းတွင် ပါရှိသည်ကို သတိထားမိမည်ဖြစ်သည်။

အခု ဆက်၍ schema-less collection အကြောင်း ဆွေးနွေးကြပါစို့။ `unicorns` အတွင်းသို့ လုံးဝမတူညီသည့် document တစ်ခုကို insert ပြုလုပ်ကြည့်ကြပါ။

```
db.unicorns.insert({name: 'Leto',
    gender: 'm',
    home: 'Arrakeen',
    worm: false})
```

ထိုနောက် `find` ကိုအသုံးပြု၍ document များကို list လုပ်ပါ။ ထပ်၍လေ့လာပြီးပါက MongoDB ၏ စိတ်ဝင်စားစရာ သဘောတရားကို ဆွေးနွေးပါမယ်။ သို့သော် ယခု မြင်သာလာမည်က သမာရိုးကျ အသုံးအနှုန်းများသည် ၎င်းနှင့် မကိုက်ဆိုသည့် အချက်ကို။

Mastering Selectors

အပေါ်မှ အချက် ခြောက်ချက်ကို စူးစမ်းနေရင်း အဆင့်မြင့်တန်း ကို ဆက်၍မလေ့လာမီ MongoDB ၏ လက်တွေ့ကျသည့် အချက် တနည်း အားဖြင့် query selectors များအကြောင်းကို နားလည်ထားရန်လိုသည်။ collection မှ ရှာသောခါ ၊ ရေတွက်သောအခါ ၊ ပြင်ဆင်သောအခါ နှင့် ဖျက်သောအခါများတွင် MongoDB ၏ query selector သည် SQL ၏ `where` statement များဖြင့် ခပ်ဆင်ဆင်ပင်ဖြစ်သည်။ JSON object သည် selector ဖြစ်ပြီး အရိုးရှင်းဆုံး ပုံစံသည် `{}` ဖြစ်ပြီး ၎င်းသည် document အကုန်လုံးဖြင့် match ဖြစ်မည်ဖြစ်သည်။ အကယ်၍ female unicorns များကိုသာရှာလိုပါက `{gender:'f'}` ဟု အသုံးပြုနိုင်သည်။

selector များအကြောင်း ထဲထဲဝင်ဝင် မဆင်းခင် data များထည့်ပြီး စမ်းသပ်ကြည့်ကြရအောင်။ ပထမဆုံး unicorns တွင်ရှိပြီးသား collection ကို `db.unicorns.remove({})` ရိုက်ပြီး ရှင်းလင်းလိုက်ပါ။ ထိုနောက် အောက်ကအတိုင်း (copy paste လုပ်ရန် အားပေးပါသည်) ရိုက်ထည်းလိုက်ပါ။

```
db.unicorns.insert({name: 'Horny',
    dob: new Date(1992,2,13,7,47),
    loves: ['carrot','papaya'],
    weight: 600,
    gender: 'm',
    vampires: 63});
db.unicorns.insert({name: 'Aurora',
    dob: new Date(1991, 0, 24, 13, 0),
    loves: ['carrot', 'grape'],
    weight: 450,
    gender: 'f',
    vampires: 43});
```

```
db.unicorns.insert({name: 'Unicrom',
    dob: new Date(1973, 1, 9, 22, 10),
    loves: ['energon', 'redbull'],
    weight: 984,
    gender: 'm',
    vampires: 182});
db.unicorns.insert({name: 'Roooooodles',
    dob: new Date(1979, 7, 18, 18, 44),
    loves: ['apple'],
    weight: 575,
    gender: 'm',
    vampires: 99});
db.unicorns.insert({name: 'Solnara',
    dob: new Date(1985, 6, 4, 2, 1),
    loves:['apple', 'carrot',
        'chocolate'],
    weight:550,
    gender:'f',
    vampires:80});
db.unicorns.insert({name:'Ayna',
    dob: new Date(1998, 2, 7, 8, 30),
    loves: ['strawberry', 'lemon'],
    weight: 733,
    gender: 'f',
    vampires: 40});
db.unicorns.insert({name:'Kenny',
    dob: new Date(1997, 6, 1, 10, 42),
    loves: ['grape', 'lemon'],
    weight: 690,
    gender: 'm',
    vampires: 39});
db.unicorns.insert({name: 'Raleigh',
    dob: new Date(2005, 4, 3, 0, 57),
    loves: ['apple', 'sugar'],
    weight: 421,
    gender: 'm',
    vampires: 2});
db.unicorns.insert({name: 'Leia',
    dob: new Date(2001, 9, 8, 14, 53),
    loves: ['apple', 'watermelon'],
    weight: 601,
    gender: 'f',
    vampires: 33});
db.unicorns.insert({name: 'Pilot',
    dob: new Date(1997, 2, 1, 5, 3),
    loves: ['apple', 'watermelon'],
    weight: 650,
    gender: 'm',
```

```
    vampires: 54});
db.unicorns.insert({name: 'Nimue',
    dob: new Date(1999, 11, 20, 16, 15),
    loves: ['grape', 'carrot'],
    weight: 540,
    gender: 'f'});
db.unicorns.insert({name: 'Dunx',
    dob: new Date(1976, 6, 18, 18, 18),
    loves: ['grape', 'watermelon'],
    weight: 704,
    gender: 'm',
    vampires: 165});
```

အခု data တွေရှိသွားပြီဆိုတော့ selector ကိုစမ်းလို့ရပါပြီ။ `{field: value}` ပုံစံဖြင့် ရှာရမည်ဖြစ်ပြီး `field` သည် `value` နှင့်ညီသည့်အရာများကိုရှာပေးမည်ဖြစ်သည်။ `{field1: value1, field2: value2}` ပုံစံသည် `and` statment နှင့်သွားတူမည်ဖြစ်သည်။ အထူး operator များဖြစ်သော `$lt`၊ `$lte`၊ `$gt`၊ `$gte` နှင့် `$ne` တို့သည် ငယ်သော ၊ ငယ်သည်နှင့်တူသည်နှစ်ခုထဲမှ တစ်ခုကိုက်ညီသော ၊ ကြီးသော ၊ ကြီးသည်နှင့် တူသည် နှစ်ခုထဲမှ ကိုက်ညီသော နှင့် မတူညီသော အရာများကိုရှာဖွေရန်အသုံးပြုနိုင်သည်။ ဥပမာ unicorn များအနက်မှာ male ဖြစ်ပြီး weight ပေါင် ၇၀၀ ထက်ကျော်သည်ကိုရှာလိုပါက အောက်ပါအတိုင်း

```
db.unicorns.find({gender: 'm',
    weight: {$gt: 700}})
//or (not quite the same thing, but for
//demonstration purposes)
db.unicorns.find({gender: {$ne: 'f'},
    weight: {$gte: 701}})
```

`$exists` operator သည် field တစ်ခုရှိမရှိကို ဆန်းစစ်ရာတွင် အသုံးပြုနိုင်သည်။ ဥပမာ

```
db.unicorns.find({
    vampires: {$exists: false}})
```

သည် document တစ်ခုသာပြန်လိမ့်မည်ဖြစ်သည်။ '$in' သည် value များစွာပါဝင်သော array တစ်ခုထဲတွင် မိမိတို့အလို့ရှိသည့် value များပါသည့် array များနှင့် ကိုက်ညီခြင်းရှိမရှိ တိုက်စစ်ရာတွင် အသုံးပြုနိုင်သည်။

```
db.unicorns.find({
    loves: {$in:['apple','orange']}})
```

၎င်းသည် apple နှင့် orange ဖြစ်ဖြစ် ကြိုက်နှစ်သက်သော unicorn များ return ပြန်လိမ့်မည်။

field များကိုစစ်ရာတွင် AND အစား OR operator ကိုအသုံးပြုလိုပါက `$or` ကိုအသုံးပြုနိုင်ပြီး selector array ကို pass ပြုလုပ်နိုင်သလို အောက်ပါအတိုင်းလည်း အသုံးပြုနိုင်သည်။

```
db.unicorns.find({gender: 'f',
    $or: [{loves: 'apple'},
          {weight: {$lt: 500}}]})
```

အပေါ်မှ အတိုင်းရေးပါက ပေါင် 500 အောက်ဖြစ်သည်ဖြစ်စေ ၊ apple ကိုနှစ်သက်သော female unicorn များကိုထုတ်ပေးမည်ဖြစ်သည်။

အပေါ်မှ ဥပမာနှစ်ခုတွင် သတိထားစရာအချက် ၂ ခုရှိသည်။ ပထမတစ်ခုမှာ `loves` ဆိုသော field သည် array ဖြစ်သည်။ MongoDB သည် array များကို first class object အနေဖြင့် အသိအမှတ်ပြုသည်။ ၎င်းသည်အလွန်

အသုံးဝင်သော feature ဖြစ်ပြီး ရင်းနှီးသွားပါက ၎င်းမပါပဲ မသုံးတက်တော့ချေ။ ပို၍ စိတ်ဝင်စားစရာကောင်းသည်က array ၏ value ကို ရွေးချယ်ရာတွင် လွယ်ကူခြင်းဖြစ်ပြီး `{loves: 'watermelon'}` ဟုအသုံးပြုပါက `loves` ၏ တန်ဖိုး `watermelon` ဖြစ်သောအရာများပါလာမည် ဖြစ်သည်။

ယခု မြင်နေကြများအပြင် အခြား သုံးနိုင်သည့် operator များလည်းရှိပါသေးသည်။ ၎င်းတို့ကို MongoDB Manual ၏ Query Selectors section တွင်ဖတ်နိုင်သည်။ ယခုဖော်ပြခဲ့သော အရာများအခြေခံဖြစ်ပြီး ၎င်းတို့ကို အများစုအနေဖြင့် အသုံးပြုရမည်ဖြစ်သည်။

`find` command ကိုအသုံးပြု၍ selector များကိုအသုံးပြုသည်ကိုတွေ့ပြီးပြီဖြစ်သည်။ ၎င်းတို့ကိုလည်း ကျွန်တော်တို့ တစ်ခေါက်အသုံးပြုဖူးသည့် `remove` command ဖြင့်လည်းတွဲဖက်အသုံးပြုနိုင်သလို `count` ဖြင့်လည်း သုံးနိုင်သည်။ ထိုနောက် `update` command ကို အချိန်တစ်ခုပေး၍ ရှင်းပြပါမည်။

MongoDB မှ `_id` field အတွက် generate ပြုလုပ်ပေးသော `ObjectId` ကို အောက်ပါအတိုင်းမြင်တွေ့နိုင်သည်။

```
db.unicorns.find(
    {_id: ObjectId("TheObjectId")})
```

နောက်တစ်ခန်းမဖတ်ခင်

`update` command ကိုမကြည့်ရသေးသလို `find` ဖြင့်ပြုလုပ်၍ရသော အချို့သော မိုက်သည့် အရာများ မစမ်းကြည့်ရသေးသော်လည်း MongoDB ကို install ပြုလုပ်ပြီး run ခြင်း၊ `insert` နှင့် `remove` ပြုလုပ်ခြင်း တို့ကို လုပ်ဆောင်ပြီးဖြစ်သည်။ ထိုနောက် `find` အကြောင်းနှင့် MongoDB မှ `selector` အကြောင်းကို မိတ်ဆက်ပေးခဲ့ပြီးဖြစ်သည်။ သင်ယုံချင်မှယုံမည်ဖြစ်သော်လည်း ယခုအခြေအနေလောက်ဖြင့် MongoDB နှင့်ပတ်သတ်သော အခြေခံတော်တော်များများကို သင်သိရှိပြီးဖြစ်သည်။ ထိုမျှပင် သင်ရလွယ်ကူသလို အသုံးပြုရ လွယ်ကူသည်။ သို့သော် နောက်အဆင့်များမတိုင်ခင် local မှာပင် အကြိမ်ကြိမ်စမ်းသပ်ရန် အကြံပြုလိုသည်။ မတူညီသော documents များ insert ပြုလုပ်ခြင်း၊ collection အသစ်များဖန်တီးခြင်း၊ နှင့် selector များကိုရင်းနှီးအောင် လေ့လာထားသင့်သည်။ ကိုယ့်ဖာသာကိုယ် `find` I `count` နှင့် `remove` များအသုံးပြုပြီး အကြိမ်ကြိမ်စမ်းသပ်ပါကတဖြည်းဖြည်းရင်းနှီးလာပြီး အသားကျလာပါလိမ့်မည်။

Chapter 2 - Updating

အခန်း (၁)တွင် CRUD (Create, Read, Update and Delete) ၏ သုံးခုကို မိတ်ဆက်ပေးခဲ့သည်။ ယခု အခန်းတွင် `update` တစ်ခုကို အာရုံစိုက်ရန် ရည်ရွယ်ထားသည်။ `Update` ပြုလုပ်ခြင်းသည် အံသြဖွယ်ရာ လုပ်ဆောင်ချက်များပါဝင်ပြီး ထိုကြောင့် ၎င်းအတွက် သီးသန့် အခန်းတစ်ခု ရေးရခြင်းဖြစ်သည်။

Update: Replace Versus $set

`update` ၏ အရိုးရှင်းဆုံးပုံစံတွင် parameter နှစ်ခုလက်ခံသည်။ selector (ဘယ်အချိန်မှာ) နှင့် field များကို ဘာ update ပြုလုပ်မည်တို့ဖြစ်သည်။ Roooooodles ပို၍ဝလာပါက အောက်ပါအတိုင်း execute ပြုလုပ်ရမည် ဖြစ်သည်။

```
db.unicorns.update({name: 'Roooooodles'},
    {weight: 590})
```

(`unicorns` collection ကိုစမ်းရင်းနှင့် နဂို data အတိုင်းမဟုတ်ပဲပြောင်းလဲသွားပါက document များအားလုံးကို `remove` ပြုလုပ်ပြီး အခန်း (၁) က အတိုင်း ပြန်၍ insert ပြုလုပ်ရန်လိုပါမည်)

update ပြုလုပ်ထားသည့် record ကိုကြည့်ကြည့်ပါ

```
db.unicorns.find({name: 'Roooooodles'})
```

`update` ၏ ပထမဆုံး အံ့ဖွယ်ကို မြင်တွေ့ရမည်ဖြစ်သည်။ မိမိတို့ apply ပြုလုပ်သော parameter မှ အခြား parameter များကိုမြင်တွေ့ရမည် မဟုတ်ပေ။ update ပြုလုပ်ရာတွင်လိုအပ်သော operator များအသုံးမပြု သဖြင့် မူလ original document ကို အသစ်သွင်းသည့် parameter ဖြင့် လုံးလုံး အစားထိုးလိုက်ခြင်းဖြစ်သည်။ ၎င်းကဲ့သို အရာများသည် SQL update များတွင်ဖြစ်မည်မဟုတ်ပေ။ ထိုကြောင့် MongoDB တွင် value တစ်ခုနှင့် တစ်ခုထက်ပိုသော field များကိုပြင်လိုပါက `$set` operator ကို အသုံးပြုရသည်။ အောက်ပါအတိုင်း ပြန်ရိုက်ထည့်၍ ပျက်သွားသော field များကိုပါ ပြင်ကြည့်လိုက်ပါ။

```
db.unicorns.update({weight: 590}, {$set: {
    name: 'Roooooodles',
    dob: new Date(1979, 7, 18, 18, 44),
    loves: ['apple'],
    gender: 'm',
    vampires: 99}})
```

၎င်းတွင် `weight` ကို မသတ်မှတ်ထား၍ overwrite ဖြစ်မည်မဟုတ်ပေ။ ၎င်းကို execute လုပ်ပါက

```
db.unicorns.find({name: 'Roooooodles'})
```

လိုချင်သည့် result ကိုရမည်ဖြစ်သည်။ ထိုကြောင့် weight ကိုချည်းသက်သက် update ပြုလုပ်လိုပါက ပြင်ရမည့်ပုံစံသည် အောက်ပါအတိုင်းဖြစ်သည်။

```
db.unicorns.update({name: 'Roooooodles'},
    {$set: {weight: 590}})
```

Update Operator များ

`$set` operator အပြင် တခြားသော operator များလည်းရှိပါသေးသည်။ update operator များသည် field အတွက်သာ အလုပ်လုပ်မည်ဖြစ်ပြီး document တစ်ခုလုံးကို အပြတ်ရှင်းမည်မဟုတ်ပါ။ ဥပမာ `$inc` operator သည် အပေါင်းနှင့် အနှုတ်ကိန်းများကို တိုးရန် အသုံးပြုမည်ဖြစ်သည်။ ဥပမာ Pilot ၏ vampire ကိုနှိမ်နှင်းခဲ့သည့် အရေအတွက်သည် မှားယွင်းနေပါက အောက်ပါအတိုင်း execute လုပ်၍ ပြင်နိုင်သည်။

```
db.unicorns.update({name: 'Pilot'},
    {$inc: {vampires: -2}})
```

အကယ်၍ Aurora တစ်ယောက် အချို့ကြိုက်လာပါကလား ၎င်း `loves` field ထဲသို့ `$push` operator အသုံးပြု၍ ထပ်ဖြည့်နိုင်သည်။

```
db.unicorns.update({name: 'Aurora'},
    {$push: {loves: 'sugar'}})
```

MongoDB Manual တွင်ပါရှိသည့် Update Operators section ကိုဖတ်၍ အခြား အသုံးပြုနိုင်သော update operator များအကြောင်းကိုလေ့လာနိုင်သည်။

Upserts

`update` ကိုအသုံးပြုသောအခါ စိတ်ချမ်းသာစရာတစ်ခုကတော့ `upsert` ကို အသုံးပြုနိုင်ခြင်းဖြစ်သည်။ `upsert` သည် data ရှိပါက update ပြုလုပ်ပြီး

မရှိပါက insert ပြုလုပ်ပေးခြင်းဖြစ်သည်။ upsert များသည် အချို့သော အခြေအနေများအတွက် အလွန်အသုံးဝင်ပြီး သုံးနေရင်း သိလာပါလိမ့်မည်။ upsert ကိုအသုံးပြုရန် update ၏ တတိယ parameter အနေဖြင့် `{upsert:true}` ဟုထည့်ပေးရသည်။

ခပ်ရိုးရိုး ဥပမာအနေဖြင့် ပြောရလျှင် website အတွက် hit counter တစ်ခု ဆိုပါစို့။ ဥပမာ real time အနေဖြင့် count များကို စုစည်းဖော်ပြလိုသည် ဆိုပါစို့။ ထို page အတွက် record မှာရှိပြီးသားမရှိသေးသည်လားကို ဆန်းစစ်ပြီး update သို့မဟုတ် insert ကို run ရမည်။ upsert option ကိုမသုံးပဲ သို့မဟုတ် false လုပ်ထားပါက အောက်ပါ option မှာ မည်သို့မှ အလုပ်လုပ်မည်မဟုတ်ပေ။

```
db.hits.update({page: 'unicorns'},
    {$inc: {hits: 1}});
db.hits.find();
```

သို့သော် upsert option ကိုထည့်လိုက်ပါက ရလဒ်မှာပြောင်းလဲသွားသည်။

```
db.hits.update({page: 'unicorns'},
    {$inc: {hits: 1}}, {upsert:true});
db.hits.find();
```

`page` သည် `unicorns` နှင့်တူသော document ကိုရှာမတွေ့ပါက အသစ်အနေဖြင့် insert လုပ်သွားမည်ဖြစ်သည်။ အကယ်၍ ဒုတိယ အကြိမ်ထပ် run ပါက ရှိပြီးသား document သည် update ဖြစ်သွားပြီး `hits` သည် ၂ သို့ပြောင်းသွားမည်ဖြစ်သည်။

```
db.hits.update({page: 'unicorns'},
    {$inc: {hits: 1}}, {upsert:true});
db.hits.find();
```

Multiple Updates

`update` ၏အံဖွယ်နောက်တစ်ခုမှာ default အနေဖြင့် document တစ်ခုကိုသာ update လုပ်ခြင်းဖြစ်သည်။ အထက်က ဥပမာတွေဆိုလျင် အလုပ်ဖြစ်နေမည်ဖြစ်သော်လည်း သင့်အနေဖြင့် အောက်ကအတိုင်းဆောင်ရွက်လိုပါက

```
db.unicorns.update({},
    {$set: {vaccinated: true }});
db.unicorns.find({vaccinated: true});
```

သင့်၏ unicorn တွေအားလုံး vaccinated ပြုလုပ်မည်ထင်ထားမည်ဖြစ်သော်လည်း ထိုသို့ပြုလုပ်ပါက `multi` ဟုသော option ကို true အနေဖြင့် ပေးမှရမည်။

```
db.unicorns.update({},
    {$set: {vaccinated: true }},
    {multi:true});
db.unicorns.find({vaccinated: true});
```

ယခုအခန်းတွင်

ယခုအခန်းတွင် CRUD operation များ၏ အခြေခံများကိုလေ့လာခဲ့ပြီးဖြစ်သည်။ `update` ၏ စိတ်ဝင်စားစရာကောင်းသည့် အခြေအနေ ၃ခုကို သိရှိရမည်ဖြစ်သည်။ ပထမအချက်မှာ MongoDB တွင် operator များ

အသုံးမပြုပဲ update ပြုလုပ်ပါက replace ဖြစ်သွားမည်ဖြစ်သည်။ ထို ကြောင့် `$set` အပါအဝင် operator များကိုအသုံးပြုရမည်ဖြစ်သည်။ ဒုတိယအနေဖြင့် `update` လုပ်ရာတွင် data ရှိမရှိ စစ်စရာမလိုတော့သည့် `upsert` option ကိုအသုံးပြုနိုင်သည်။ နောက်ဆုံးအနေဖြင့် `update` သည် မူလသဘောအရ match ဖြစ်သည့် ပထမဆုံးသော document ကိုသာ update ပြုလုပ်ပြီး အကုန်လုံးကိုပြောင်းလဲစေချင်ပါက `multi` option ကို အသုံးပြုရသည်။

အခန်း (၃) ကို find ကို အသေးစိတ် လေ့လာခြင်း

အခန် (၁) တွင် `find` command ကိုအကြမ်းဖြင်းတွေ့ရမည်ဖြစ်သည်။ `selector` များတွင်သာမက `find` တွင် တခြားလေ့လာစရာများရှိပါသေးတယ်။ `find` မှရွာတွေ့သည်များကို `cursor` အနေဖြင့်ရသည်ကို ပြောပြခဲ့ပြီးဖြစ်ပါသည်။ ယခုအခါတွင် အသေးစိတ် ဘာကိုဆိုလိုသည်ကို ဆက်သွားပါမည်။

Field များရွေးချယ်ခြင်း

`cursor` များအကြောင်း မပြောမီ `find` သည် projection ဟုခေါ်သည့် ဒုတိယ optional parameter အကြောင်းကိုလေ့လာကြပါစို့။ ၎င်း parameter သည် မိမိတို့ အလိုရှိသည့် field များနှင့်ချန်ထားလိုသည့် field များကို သတ်မှတ်ရာတွင် အသုံးဝင်သည်။ ဥပမာ unicorn များ၏ အမည်ကိုသာ အလိုရှိပြီး ကျန်သည့် field များကိုချန်ထားလိုပါက

```
db.unicorns.find({}, {name: 1});
```

ပုံမှန်အားဖြင့် `_id` field သည်အမြဲပါရှိမည်ဖြစ်ပြီး ၎င်းကို ချန်လှပ်လိုပါက `{name:1, _id: 0}` ဟု အသုံးပြုနိုင်သည်။ `_id` field မှအပ အခြားသော field များကိုရွေးချယ်ရာတွင် ပါချင်သည်များကို ရွေးခြင်း သို့မဟုတ် မပါချင်သည်များကို ရွေးခြင်း တစ်ခုခုကိုသာ ရွေးချယ်အသုံးပြုသင့်ပါသည်။

သင့်အနေဖြင့် fields များကို ရွေးချယ်ခြင်း သို့မဟုတ် ပယ်ထုတ်ခြင်းသည် ပို၍ ရှင်းလင်းပြီး အဓိပ္ပါယ်ရှိသည်။

Ordering

`find` သည် cursor ကို return ပြန်သည်ကိုသိထားပြီးနောက် ၎င်းကို shell တွင်အသုံးပြုပါက ချက်ချင်း execute ပြုလုပ်သည်ကို တွေ့ရမည်ဖြစ်သော်လည်း ၎င်း shell ၏ behaviour တစ်ခုသာဖြစ်ပြီး `cursor` များအလုပ်တကယ်လုပ်ပုံကိုမူ ordering ပြုလုပ်မှသာ သိနိုင်သည်။ ပထမဆုံးကြည့်ရမည်မှာ `sort` ဖြစ်သည်။ မိမိတို့၏ JSON document အလိုက် sort ပြုလုပ်ပါက အစဉ်အတိုင်းဖြစ်လျင် 1 ပြောင်းပြန်ဆိုပါက -1 ကိုအသုံးပြုနိုင်သည်။ ဥပမာ

```
//heaviest unicorns first
db.unicorns.find().sort({weight: -1})

//by unicorn name then vampire kills:
db.unicorns.find().sort({name: 1,
    vampires: -1})
```

Relational Database ကဲ့သို့ပင် MongoDB တွင် sorting အတွက် index ကိုအသုံးပြုနိုင်သည်။ ၎င်းကို index များအကြောင်းရှင်းပြမှသာ အသေးစိတ်ပြောပါတော့မည်။ သို့သော်သိထားရမည်မှာ MongoDB သည် index မပါပါက sorting ပြုလုပ်နိုင်သော ပမာဏကို limit ပြုလုပ်ထားပါသည်။ အကယ်၍ သင့်အနေဖြင့် index မပြုလုပ်ထားသော အလွန်

ကြီးမားသော collection တစ်ခုကို sort ပြုလုပ်ပါက error တက်မည် ဖြစ်သည်။ အမှန်အတိုင်းဆိုရလျှင် database များသည် ၎င်းကဲ့သို့ optimize မပြုလုပ်ထားသော လျော့တိလျော့ရဲရေးထားတဲ့ query များကို တားဆီးထားသည်ကို မြင်ချင်မိပါသည်။ (MongoDB ဘက်ကို ရှေ့နေလိုက်ပေးခြင်းတော့ မဟုတ်သော်လည်း ဆိုးဆိုးရွားရွား optimize ပြုလုပ်ထားသော database များမြင်ရတိုင်း strict-mode ရှိပါစေဟု ဆုတောင်းမိသည်)

Paging

`limit` နှင့် `skip` cursor method များကို အသုံးပြု၍ paging result များကို ရယူနိုင်သည်။ ဒုတိယနှင့် တတိယ အလေးဆုံး unicorn ကိုသိလိုပါက အောက်ပါအတိုင်း ရှာယူနိုင်သည်။

```
db.unicorns.find()
    .sort({weight: -1})
    .limit(2)
    .skip(1)
```

Using `limit` in conjunction with `sort`, can be a way to avoid running into problems when sorting on non-indexed fields.

Count

Shell တွင် တိုက်ရိုက် `count` လုပ်နိုင်ပြီး ဥပမာ အနေဖြင့် အောက်ပါအတိုင်း ဖြစ်သည်။

```
db.unicorns.count({vampires: {$gt: 50}})
```

လက်တွေ့တွင်မူ `count` သည် `cursor` method တစ်ခုဖြစ်ပြီး shell မှာ ၎င်း၏ shortcut အနေဖြင့် support ပြုလုပ်ထားသည်။ တချို့ support မ ပြုလုပ်သော driver များတွင်မူ အောက်ပါအတိုင်း ပြုလုပ်ရပါသည်။ (၎င်း နည်းလမ်းမှာ နှစ်ခုလုံးတွင် အသုံးပြု၍ရသည်။)

```
db.unicorns.find({vampires: {$gt: 50}})
    .count()
```

ယခုအခန်းတွင်

`find` နှင့် `cursor` များသည်ရိုးရိုးရှင်းရှင်းဖွဲ့စည်းထားသည်။ အခြား အပို ဆောင်း command များကို နောက်ပိုင်းအခန်းများတွင် ဖော်ပြသွားမည် ဖြစ်သည်။ ယခု အခြေအနေတွင် mongo shell တွင် အဆင်ပြေစွာသုံးနိုင် ရန် လေ့ကျင်ထားရန်လိုပြီး MongoDB ၏ အခြေခံကိုလည်း သိထားရန်လို သည်။

အခန်း (၄) - Data Modeling

နောက်ထပ် ဂီယာ ပြောင်းပြီး MongoDB အကြောင်း အကြမ်းဖြင်းဆွေးနွေးကြတာပေါ့။ အချက်အလက် အသစ်များနှင့် syntax အသစ်များကို ရှင်းပြရသည်က မခက်လှပါ။ သို့သော် နည်းပညာအသစ်ဖြင့် modeling ပြုလုပ်ရမည့် အကြောင်း ဆွေးနွေးရသည်မှာမူ မလွယ်လှပေ။ NoSQL database များအားလုံးတွင် document အခြေပြု database များမှာ relational datbase များနှင့် အနီးစပ်ဆုံးဖြစ်သော်လည်း ကွဲပြားခြားနားချက်များရှိပြီး ၎င်းတို့သည် အရေးကြီးသော အချက်များဖြစ်သည်။

Join မရှိ

MongoDB နှင့်အသားကျရန် လေ့လာရာတွင် အဓိကအကျဆုံး ကွာခြားချက်မှာ ၎င်းတွင် join မရှိပါ။ join ကို အဘယ်ကြောင့် support မလုပ်ခဲ့သည်ကို မသိသော်လည်း ကျွန်တော် မြင်သလောက် join သည် scale ပြုလုပ်ရန်ခက်ခဲ့သည်။ ထိုကြောင့် data များကို ဖြန့်ခွဲသိမ်းဆည်းပါက application layer တွင် ပြန်၍ join ပြုလုပ်ရသည် ထိုကြောင့် data များသည် relational *ဖြစ်မြဲ*ဖြစ်ပြီး MongoDB သည် join ကို support မပြုလုပ်သည်က အမှန်။

တခြား ဘာမှမသိထားပဲ join မရှိသော ကမ္ဘာတွင်နေနိုင်ရန် application code တွင် ကိုယ့်ဖာသာကိုယ် join ရပါသည်။ ထိုကြောင့် ဒုတိယ query တွင် သက်ဆိုင်သော အချက်အလက်ကို collection မှ `find` နိုင်ရန် အရေးကြီးသည်။ data ကို setting ပြုလုပ်ချင်းသည် relational database များတွင် foriegn key အသုံးပြုခြင်းနှင့် သိပ်၍ မကွာခြားလှပါ။ အရင်ဆုံး `unicorns` collection မှ ခဏခွာ၍ `employees` collection ဘက်ကိုလှည့်ပါစို့။ ပထမဦးစွာ employee ကို create ပြုလုပ်ရန်လိုပါသည်။ (ယခု ဥပမာတွင် `_id` ကိုတမင်တကာထည့်ပေးထားသည်)

```
db.employees.insert({_id: ObjectId(
    "4d85c7039ab0fd70a117d730"),
    name: 'Leto'})
```

ထိုနောက် တခြား employee များကိုလည်း ထပ်ဖြည့်ပြီး ၎င်း၏ manager ကို `Leto` ဟုထားလိုက်ပါ။

```
db.employees.insert({_id: ObjectId(
    "4d85c7039ab0fd70a117d731"),
    name: 'Duncan',
    manager: ObjectId(
    "4d85c7039ab0fd70a117d730")});
db.employees.insert({_id: ObjectId(
    "4d85c7039ab0fd70a117d732"),
    name: 'Moneo',
    manager: ObjectId(
    "4d85c7039ab0fd70a117d730")});
```

(မှတ်ထားရန်တစ်ခုမှာ `_id` သည် unique ဖြစ်သည့် value တစ်ခုခုဖြစ်လျင် ရသည်။ သင့်အနေဖြင့် လက်တွေ့တွင် `ObjectId` ဟု အသုံးပြုရသည် အခါ

မျိုးလည်းရှိမည်ဖြစ်သဖြင့် တခါတည်း စမ်းကြည့်ပါ။)

Leto ၏ အလုပ်သမားများ အားလုံးကို လိုအပ်ပါက အောက်ပါအတိုင်း execute ပြုလုပ်နိုင်သည်။

```
db.employees.find({manager: ObjectId(
    "4d85c7039ab0fd70a117d730")})
```

ဘာမှထူးခြားသည်တော့ မဟုတ် ၊ အဆိုးဆုံးသော အနေဖြင့် အချိန်တော်တော်များများတွင် join မရှိသဖြင့် query တကြောင်းပိုရေးရမည်ဖြစ်သည်။

Arrays and Embedded Documents

MongoDB တွင် join မရှိသဖြင့် ၎င်းတွင် trick များမရှိသည် မဟုတ်။ MongoDB တွင် array များကို first class အနေဖြင့် support ပြုလုပ်သဖြင့် ၎င်းသည် many-to-one နှင့် many-to-many relationship များကို ကိုင်တွယ်ရာတွင် အသုံးဝင်သည်။ ဥပမာ အနေဖြင့် manager နှစ်ယောက်ရှိသော employee အတွက် အောက်ပါအတိုင်း array အတွင်းထည့်သွင်းနိုင်သည်။

```
db.employees.insert({_id: ObjectId(
    "4d85c7039ab0fd70a117d733"),
    name: 'Siona',
    manager: [ObjectId(
    "4d85c7039ab0fd70a117d730"),
    ObjectId(
    "4d85c7039ab0fd70a117d732")] })
```

စိတ်ဝင်စားစရာကောင်းသည်မှာ အချို့သော document များတွင် `manager` သည် scaler ဖြစ်လျင်ဖြစ်ပြီးတချို့အတွက် array ဖြစ်ပါလိမ့်မည်။ မူလ find query သည်နှစ်ခုစလုံးအတွက် အလုပ်လုပ်မည်ဖြစ်သည်။

```
db.employees.find({manager: ObjectId(
    "4d85c7039ab0fd70a117d730")})
```

value array မှာ many-to-many join table များထက်ပို၍ အသုံးပြုရ အဆင်ပြေသည်ကို သတိထားမိမည်ဖြစ်သည်။

Array များအပြင် MongoDB သည် document အတွင်း document များ အဖြစ်အသုံးပြုနိုင်သည်။ အောက်ပါ nested document ကို insert ပြုလုပ် ကြည့်ပါ။

```
db.employees.insert({_id: ObjectId(
    "4d85c7039ab0fd70a117d734"),
    name: 'Ghanima',
    family: {mother: 'Chani',
        father: 'Paul',
        brother: ObjectId(
    "4d85c7039ab0fd70a117d730")}})
```

ဘယ်လိုပြန်ရှာရမည် စဉ်းစားနေပါက embedded document များသည် dot-notation ဖြင့်ရှာနိုင်သည်။

```
db.employees.find({
    'family.mother': 'Chani'})
```

embedded doument များကို မည်ကဲ့သို့ အသုံးဝင်သည်နှင့် မည်သို့ အသုံးပြုရမည်ကို အကြမ်းဖြင်းရှင်းပြပါမည်။

ထို concept နှစ်ခုကို ပေါင်းစပ်၍ embedded ပြုလုပ်ထားသော documents array ကိုအသုံးပြုနိုင်သည်။

```
db.employees.insert({_id: ObjectId(
    "4d85c7039ab0fd70a117d735"),
    name: 'Chani',
    family: [ {relation:'mother',name: 'Chani'},
        {relation:'father',name: 'Paul'},
        {relation:'brother', name: 'Duncan'}]})
```

Denormalization

Join အစားပြုလုပ်နိုင်သည့် နည်းလမ်းတစ်ခုမှာ data များကို denormalize ပြုလုပ်ခြင်းဖြစ်သည်။ တောက်လျှောက် denormalize ပြုလုပ်ရသည့် ရည်ရွယ်ချက်မှာ performance ပိုမိုကောင်းမွန်ရန်နှင့် snapshot (audit log ကဲ့သို့) ပြုလုပ်နိုင်ရန်ဖြစ်သည်။ သို့သော် NoSQL ကျော်ကြားလာသည်အမျှ join မရှိသည်က များသဖြင့် Modeling ပြုလုပ်ရာတွင် denormalize ပြုလုပ်လာသည်က ပုံမှန်လိုဖြစ်လာသည်။ သို့သော် ရှိသမျှ အချက်အလက်တိုင်းကို duplicate ပြုလုပ်ရမည်ဟု မဆိုလိုပါ။ သို့သော် duplicate ဖြစ်မှုကို ကြောက်နေမည့်အစား document ကိုကြည့်၍ လိုအပ်သလို model ပြုလုပ်ရမည်ဖြစ်သည်။

အကယ်၍ forum application တစ်ခုရေးသည် ဆိုပါစို့။ ထုံတမ်းစဉ်လာအရ ဆိုလျှင် `user` တစ်ယောက်၏ `post` သည် `posts` table အတွင်းရှိ `userid` အနေဖြင့် တည်ရှိနေမည်ဖြစ်သည်။ ထို Model အရဆိုပါက `user` ဖြင့် မ

join ပဲ `post` ကိုထုတ်ပြနိုင်မည် မဟုတ်ပေ။ တခြား နည်းလမ်းတစ်ခုမှာ `name` အပြင် `userid` တို့ကို `post` တိုင်းတွင် store ပြုလုပ်ရမည်ဖြစ်သည်။ Embedded document တွင်မူ `user: {id: ObjectId('Something'), name: 'Leto'}` ဟူ၍ ထပ်ဆင့် သိမ်းထားနိုင်သည်။ အကယ်၍ user များက ၎င်း၏ အမည်ကိုပြောင်းလဲပါက document တိုင်းကို update ပြုလုပ်ရမည်ဖြစ်မည်။(multi-update ကိုအသုံးပြုရမည်။)

ထိုကဲ့သို့ approach ကိုပြောင်းလဲရသည်မှာ လွယ်ကူလှသည်တော့မဟုတ်။ တခြားသို့သော အခြေအနေများမှာ မှားနေသည်ဟု ခံစားရမည်ဖြစ်သည်။ သို့သော် ထို approach ကိုစမ်းသပ်ကြည့်ရန် မကြောက်ရွံ့ပါနှင့်။ အချို့သော အခြေအနေများအတွက် အဆင်ပြေရုံသာမက အကောင်းဆုံးဖြေရှင်းနည်းပါ ဖြစ်ချင်ဖြစ်တက်ပါသည်။

Which Should You Choose?

id array များသည် one-to-many နှင့် many-to-many အခြေအနေများတွင် အသုံးဝင်သော ဖြေရှင်းနည်းဖြစ်သည်။ များသောအားဖြင့် developer များသည် ထိုကဲ့သို့ manual reference ပြုလုပ်ခြင်းထက် embedded documents များကို ပို၍ အသုံးပြုကြသည်။

ပထမဦးစွာ သိထားရမည်မှာ document တစ်ခုသည် 16megabyes သာခွင့်ပြုမည်ဖြစ်သည်။ document များမှာ size limit ရှိသဖြင့် ၎င်းတို့ကို မည်သို့

အသုံးပြုရမည်ကို အကြမ်းဖြင့်ခန့်မှန်း၍ရသည်။ ထိုအခြေအနေတွင် developer အတော်များများသည် relationship တော်တော်များများကို manual reference ပြုလုပ်ကြသည်။ အချို့သော အခြေအနေများတွင်မူ embedded documents များကို အသုံးပြုကြသော်လည်း data size သေးငယ်သည်များကို အများအားဖြင့် parent document မှ ထည့်ထားချင်သည့် အခါတွင်သုံးသည်။ လက်တွေ့ ဥပမာအနေဖြင့် user တစ်ဦးချင်း၏ `addresses` document များကိုအောက်ပါအတိုင်း ထည့်သွင်းသည်ကို တွေ့ရမည်။

```
db.users.insert({name: 'leto',
    email: 'leto@dune.gov',
    addresses: [{street: "229 W. 43rd St",
                 city: "New York", state:"NY",zip:"10036"},
                {street: "555 University",
                 city: "Palo Alto", state:"CA",zip:"94107"}]})
```

သို့သော်လည်း embedded document များကို minor data များကို ထည့်သွင်းရန် သို့မဟုတ် သိပ်အသုံးမဝင်ဟု ဆိုလိုခြင်းမဟုတ်ပါ။ data model မှ object သို့ တိုက်ရိုက်ဆက်စပ်ခြင်းသည် တချို့အရာများအတွက် အတော်ရိုးရှင်းသွားကာ join ပြုလုပ်ရန်လိုအပ်ခြင်းမှ အတော်နည်းပါးသွားပြီး အထူးသဖြင့် MongoDB တွင် embedded documents နှင့် array များကို index ပြုလုပ်နိုင်ခြင်းကြောင့်လည်း ပါဝင်သည်။

Collection အနည်းအများ

Collection များမှာ schema များကို enforce မပြုလုပ်သော်လည်း collection တစ်ခုထဲမှ အမျိုးစုံ document များ ဗျောက်သောက် ပစ်ထည့်သည် စနစ်ကို တည်ဆောက်သည်က ရှိနိုင်သေးပြီး ၎င်းမှာ အတော် အခြေအနေဆိုးသည့် လုပ်ရပ်ဖြစ်သည်။ MongoDB systems များသည် relational system များတွင်တွေ့ရှိရမည်နှင့် ဆင်တင်တင်ဖြစ်ပြီး collections များမှာ ပို၍နည်းနိုင်သည်။ တနည်းအားဖြင့် relational database မှ table တစ်ခုသည် MongoDB တွင် collection တစ်ခုဖြစ်နိုင်သည်။ (Many to many join table နှင့် One to Many join table များမှာ မူ ထိုဥပမာထဲမပါဝင်ပါ။)

embedded documents များနှင့် ဆက်စပ်စဉ်းစားပါက ပို၍စိတ်ဝင်စားစရာကောင်းလာပါသည်။ အများစုရင်းနှီးကြမည့် ဥပမာမှာ blog ဖြစ်သည်။ `posts` နှင့် `comments` များသည် သီးသန့် collection အဖြင့်ရှိသင့်ပါသလား `post` တစ်ခုတိုင်းတွင် `comment` များသည် array အနေဖြင့်ရှိသင့်ပါသလား။ 16MB document size limit ကိုခဏမေ့ထား၍ (Sheakspear ၏ ဝတ္ထုတစ်အုပ်လုံး၏ text size သည်ပင် 200kb သာရှိသည်) developers အများစုသည် ခွဲရေးကို ပိုသဘောကျကြသည်။ ၎င်းသည် ပို၍ရှင်းလင်းပြီး performance အနေဖြင့် ပိုကောင်းသည်။ MongoDB ၏ flexible ဖြစ်သော approach ကြောင့် blog post တစ်ခုမှ comment များကို တချို့တဝက် embedded ပြုလုပ်ပြီး (ပထမဆုံး အနည်းငယ်) ကျန်သည်များကို သီးသန်း

သိမ်းဆည်းသည့် ပုံစံမျိုးကို အသုံးပြုနိုင်သည်။ ထိုသို့ဖြင့် data များကို အလိုရှိပါက query တစ်ခုတည်းဖြင့် ယူနိုင်သည့် စည်းကမ်းကို လိုက်နာရာကြသည်။

16MB limit မှလွဲ၍ တင်းကျပ်ထားသော စည်းကမ်းမရှိပါ။ မျိုးစုံ စမ်းသပ်ကြည့်ပါ မည်သည်က ပို၍ အဆင်ပြေသည် မပြေသည်ကို တွေ့လာရပါမည်။

ယခုအခန်းတွင်

ဒီအခန်း၏ အဓိကရည်ရွယ်ချက်မှာ MongoDB တွင် data modeling ပြုလုပ်ရာတွင် အသုံးဝင်သည့် guideline များကို ညွှန်ပြခြင်းဖြစ်သည်။ Document အခြေပြုစနစ်တွင် Modeling ပြုလုပ်ရာတွင် Relational World နှင့် ကွဲပြားသောလည်း အရမ်း မခြားနားလှပါ။ ပို၍ flexible ဖြစ်ပြီး constraint အနည်းငယ်မျှသာရှိပြီး စနစ်အသစ်များအတွက်မူ ပို၍ အဆင်ပြေလေ့ရှိသည်။

အခန်း (၅) - MongoDB ကိုဘယ်အချိန်မှာအသုံးပြုမလဲ

အခုအနေအထားအရဆိုလျင် သင့်အနေဖြင့် MongoDB သည် သင့်ရှိပြီးသား စနစ်၏ မည့်သည်နေရာတွင် ကိုက်ညီမည်နည်းဆိုသည်ကို သိနိုင်မည် ဖြစ်သည်။ ရှိပြီးသားနှင့် အသစ်ပေါ်လာသော နည်းပညာများစွာရှိပြီး ၎င်းတို့သည် ယခု လိုအပ်ချက်များကို ဖြည့်ဆည်းပေးနိုင်စွမ်းရှိသည်။

ကျွန်တော်အတွက် အရေးအကြီးဆုံး သင်ခန်းစာမှာ MongoDB နှင့်မဆိုင်သော်လည်း သင့်၏ data များနှင့်ပတ်သတ်၍ solution တစ်ခုတည်းကို မမှီခိုရန်ဖြစ်သည်။ solution တစ်ခုတည်းကို အသုံးပြုခြင်းသည် သိသာထင်ရှားသော အားသာချက်များလည်းရှိပြီး Project အတော်များများအတွက် တစ်ခုတည်း အသုံးပြုခြင်းသည် ဖြစ်နိုင်သော လမ်းကြောင်းတစ်ခုဖြစ်သည်။ သင့်အနေဖြင့် မတူညီသော နည်းပညာများကို *မဖြစ်မနေ* အသုံးပြုရမည်ဟု ဆိုလိုခြင်း မဟုတ်ပဲ လိုအပ်ပါက *အသုံးပြုနိုင်အောင်* လေ့လာထားရမည်ဖြစ်ပြီး ၎င်း၏ အားသာချက်နဲ့ ကုန်ကျစရိတ်ကို စဉ်းစားဆုံးဖြတ်နိုင်ရမည်။

ထိုသို့ ပြောခဲ့သော်လည်း ယခုအထိ အသုံးပြုပြီးနောက်ပိုင်းတွင် သင့်အနေဖြင့် MongoDB ကို ယေဘူယျ solution တစ်ခုဟုမြင်စေရန် မျှော်လင့်ပါသည်။ အစောပိုင်းမှ ပြောခဲ့သလို Document Database များသည်

Relational Database များနှင့် အတော်ဆင်တူလေ့ရှိသည်။ ထို့ကြောင့် ဝေဝိုက်ပြောမနေပဲ MongoDB သည် relational database များအတွက် အခြားတစ်ဖက် အနေဖြင့် မြင်နိုင်သည်။ Lucense ကို Relational database များမှ full text indexing အတွက်သော်လည်းကောင်း၊ Redis ကို persistant key-value store တစ်ခုအနေဖြင့် လည်းကောင်း၊ MongoDB ကို ၎င်း data များအားလုံးအတွက် ဗဟိုချက် repository အဖြစ်လည်းကောင်း စဉ်းစားနိုင်သည်။

ကျွန်တော်အနေဖြင့် MongoDB ကို relational database များ၏ အစားထိုး ဟု မပြောခဲ့ပဲ အခြားတဖက် ဟု ပြောခဲ့သည်ကို သတိထားမိလိမ့်မည်။ ၎င်း သည် အခြားကရိယာအတော်များများ ပြုလုပ်နိုင်သည့် အရာများကို စွမ်းဆောင်နိုင်သည့် ကရိယာတစ်ခုပင်ဖြစ်သည်။ တချို့အရာများတွင် MongoDB တွင်ပိုအဆင်ပြေပြီး တချို့အရာများသည် ပိုချာလိမ့်မည်။ ၎င်း အရာများကို ဆက်လက်စုံစမ်းကြည့်ကြပါစို့။

Flexible Schema

document-oriented database ၏ အများဆုံးပြောကြလေ့ရှိသော အားသာ ချက်မှာ schema အသေမဟုတ်ခြင်းဖြစ်သည်။ ၎င်းအတွက် traditional database မှ table များထက် ပို၍ flexible ဖြစ်သည်။ flexible schame

ဖြစ်တိုင်းကောင်းသည် ဟု ဆိုလိုခြင်းမဟုတ်ပဲ လူအများပြောကြသည်ကို ပြောပြခြင်းဖြစ်သည်။

Schema-less ဖြစ်သည် လူအများက ချီးမွမ်းပြီးနောက်ပိုင်း တကယ့် တကယ် အလုပ်လုပ်ရာတွင် အမျိုးအစားစုံလင်ပြီး မကိုက်ညီသော data တစ်ပုံတခေါင်းများ ရောက်လာသည်ကို တွေ့ရမည်။ MongoDB မသုံးပဲ တချို့သော domain နှင့် dataset များသည် relational database များ အသုံးပြုပါက အလွန်တိုင်ပတ်မည်ဖြစ်သော်လည်း ထိုကိစ္စများကို ဖြစ်တောင့်ဖြစ်ခဲ ဟုယူဆကြပါစို့။ schema-less ဖြစ်ခြင်းသည် မိုက်သော်လည်း သင်အသုံးပြုသော data အများစုသည် structure အထပ်ထပ် ပြုလုပ်ထားသည်များဖြစ်တက်သည်။ Feature အသစ်ထပ်ထည့်သောအခါ ရံဖန်ရံခါ မကိုက်ညီခြင်းသည် တိုင်ပတ်သလို တကယ့်လက်တွေ့တွင် nullable column ကိုအသုံးပြုခြင်းသည်လည်း ပိုကောင်းသည့် ဖြေရှင်းမှုဟု ဆိုမရပါ။

ကျွန်တော် အမြင်အရ Dynamic Schema ၏အဓိက အားသာချက်မှာ setup ပြုလုပ်စရာမလိုခြင်းနှင့် OOP နှင့် ပွတ်တိုက်မှုလျော့နည်းခြင်း ဖြစ်သည်။ ၎င်းသည် သင့်အနေဖြင့် static language များနှင့် အလုပ်လုပ်ပါက ပို၍ သိသာပါလိမ့်မည်။ ကျွန်တော် C# နှင့် Ruby ကို အသုံးပြု၍ MongoDB နှင့် အလုပ်လုပ်ဖူးပြီး ခြားနားချက်မှာ အတော်ပင်ဖြစ်သည်။ Ruby ၏ dyanmism နှင့် ActiveRecord implementation သည် object

များနှင့် relational database များအကြားတွန်းအားကို အတော်ပင် လျော့နည်းပြီးသားဖြစ်သည်။ ထိုသို့ပြောခြင်းဖြင့် MongoDB နှင့် Ruby သည်မသင့်တော်ဟု ဆိုလိုခြင်း မဟုတ်၊ ကျွန်တော့်အမြင်အရ Ruby developer များသည် MongoDB ကို ၎င်းတို့၏ ခြေလှမ်းတစ်ခုအဖြင့်သာ မြင်ပြီး C# နှင့် Java developer များအတွက်မူ data များနှင့် interact ပြုလုပ်ရာတွင် အခြေခံပိုင်းဆိုင်ရာ လုံးဝပြောင်းလဲမှုတစ်ခုဖြစ်သည်။

driver ကို develop ပြုလုပ်သူတစ်ယောက်အနေဖြင့် တွေးကြည့်ပါ။ object ကို save ခြင်လား JSON အဖြစ် ပြောင်းလိုက်ပါ (BSON ဟုပြောလျှင်ပို မှန်မည်ဖြစ်သော်လည်း ထားပါတော့) ပြီးနောက် MongoDB သို့ပို့လိုက်ပါ။ property ကော type mapping များလည်းမရှိပါ။ ထိုသို့ရိုးရှင်းမှုသည် ၎င်းမှ တဆင့် သင့်အထိ စီးဆင်းသွားသည်။

Writes

MongoDB ၏ အထူးပြုသုံးစွဲသည့် နေရာတစ်ခုမှာ logging ဖြစ်သည်။ MongoDB ၏ အချက်နှစ်ချက်သည် မြန်မြန်ဆန်ဆန် write နိုင်ရန် အထောက်အကူပေးသည်။ ပထမအချက်မှာ write command ကိုအသုံးပြု ပြီးနောက် အသိအမှတ်ပြုသည်ကို မစောင့်ပဲ return ပြန်လာသည် option တစ်ခုရှိသည်။ နောက်တစ်ခုမှာ data ခိုင်မာမှု၏ write behavior ကို ကိုယ်တိုင် ထိန်းချုပ်နိုင်သည်။ ၎င်း setting များအပြင် server ဘယ်နှစ်ခု

ကို အောင်မြင်စွာ write ပြုလုပ်ပြီးမှ အသိအမှတ်ပြုသည်ကိုက အစ configure ပြုလုပ်နိုင်ပြီး write performance နှင့် data ခိုင်မာမှု အကြား control အများကြီး ပေးစွမ်းနိုင်ပါသည်။

Performance အပိုင်းများအပြင် Log data များသည် schema မရှိခြင်း အားသာချက်ကို ရယူနိုင်သော data set အမျိုးအစားတစ်ခုဖြစ်သည်။ နောက်ဆုံးတွင် MongoDB တွင် [capped collection] ဟုပါရှိလာသည်။ အပေါ်မှာ တောက်လျှောက် create ပြုလုပ်ခဲ့သော collection များမှာ ပုံမှန် ဖြစ်ပြီး capped collection များကို `db.createCollection` ဟူသော command ကိုအသုံးပြုပြီး capped flag ကိုထည့်၍ အသုံးပြုနိုင်သည်။

```
//limit our capped collection to 1 megabyte
db.createCollection('logs', {capped: true,
    size: 1048576})
```

ကျွန်တော်တို့၏ capped collection သည် document အဟောင်းများကို အလိုအလျောက်ဖျက်ပစ်မည်ဖြစ်သည်။ `max` ကိုအသုံးပြု၍ document ၏ size အစား အရေအတွက် ဖြင့်လည်း ထိန်းချုပ်နိုင်သည်။ capped collection များတွင် စိတ်ဝင်စားစရာ ဂုဏ်သတ္တိများလည်းရှိသေးသည်။ ဥပမာ document ကို update ပြုလုပ်နိုင်သော်လည်း size ကိုမပြောင်းလဲနိုင်ပေ။ insert ပြုလုပ်သည့် order သည် အစဉ်လိုက်ထားသဖြင့် time based sorting ကိုရရှိရန် index ပြုလုပ်စရာမလိုပေ။ unix တွင် `tail -f`

`<filename>` ဟု tail ပြုလုပ်သလို capped collection များကို tail ပြုလုပ်နိုင်ပြီး data အသစ်ရောက်လာသည်နှင့် query အသစ်ပြန်ရေးစရာမလိုပေ။

collection ၏ size အစား အချိန်ကာလ အလျောက် expire ပြုလုပ်လိုပါက TTL Indexesများကိုအသုံးပြုနိုင်သည်။ TTL ၏ အရှည်ကောက်မှာ “time-to-live” ဖြစ်သည်။

Durability

Version 1.8 မတိုင်မီအထိ MongoDB တွင် single-server durability မပါရှိပေ။ ထိုကြောင့် server crash ဖြစ်ပါက data lost ဖြစ်လျှင်ဖြစ် မဖြစ်ပါက corrupt ဖြစ်လေ့ရှိသည်။ ထိုကြောင့် MongoDB ကို အမြဲတမ်း Multi-server အနေဖြင့်အသုံးပြုရန် (MongoDB တွင် replication ကို support လုပ်သည်) အားပေးလေ့ရှိသည်။ 1.8 တွင် အဓိကပါဝင်သော feature မှာ Journaling ဖြစ်ပြီး 2.0 မှစ၍ MongoDB တွင် default အနေဖြင့် Journaling ကို enable ပြုလုပ်လေ့ရှိပြီး ထိုကြောင့် crash သို့မဟုတ် power loss ဖြစ်ပါက လျှင်မြန်စွာ recover ပြုလုပ်နိုင်သည်။

Full Text Search

MongoDB တွင် full text search ကိုအသုံးပြုနိုင်ပြီး ဘာသာစကား ၁၅ မျိုး၏ stop words နှင့် steeming များကို support ပြုလုပ်သည်။

MongoDB ၏ array နှင့် full text search များကြောင့် သင့်အနေဖြင့် ပို၍ powerful ဖြစ်သော full text search engine များမှအပ ပုံမှန်အားဖြင့် အသုံးပြုနိုင်သည်။

Transactions

MongoDB 4.2 မှစ၍ transaction ကိုအသုံးပြုနိုင်ပြီး အစောပိုင်း version များတွင်မူတွင် transaction မရှိသော်လည်း တစ်ဖက်လှည့်အနေဖြင့် နှစ်မျိုးအသုံးပြုနိုင်သည်။ ပထမတစ်မျိုးမှာ ကောင်းမွန်သော်လည်း အကန့်အသတ်ရှိပြီး ၊ ဒုတိယ တစ်မျိုးမှာ အလုပ်ရှုပ်သော်လည်း flexible ဖြစ်သည်။ ပထမတစ်မျိုးကို အစောပိုင်းကတည်းက တွေးပြီးပြီဖြစ်သည် ဥပမာ `$inc` နှင့် `$set` တို့ဖြစ်သည်။ တချို့သော command များဖြစ်သော `findAndModify` သည် update သို့မဟုတ် delete ပြုလုပ်နိုင်ပြီး အလိုအလျောက် return ပြုလုပ်နိုင်သည်။ Atomic operation များမှာ မလုံလောက်ပါက two-phase-commit) များကိုအသုံးပြုနိုင်သည်။ ၎င်းသည် relational ကမ္ဘာတွင် ကျော်ကြားပြီး database များစွာတွင် transaction များ implement ပြုလုပ်ရာတွင်အသုံးပြုသည်။ MongoDB တွင် nested document များနှင့် schema မှာ အရှင်ဖြစ်သဖြင့် အနည်းငယ်သက်သာသော်လည်း အခုမှ စလေ့လာပါက လွယ်ကူသည်တော့မဟုတ်ပေ။

Data Processing

version 2.2 မတိုင်မီက MongoDB တွင် MapReduce ကိုအဓိကထား၍ data processing job များကို ဆောင်ရွက်ခဲ့သော်လည်း 2.2 မှစ၍ aggregation framework or pipeline] (http://docs.mongodb.org/manual/core/aggregation-pipeline/) ဟုသည် powerful ဖြစ်သည့် feature ပါဝင်လာပြီး ပုံမှန်ထက်ရှုပ်ထွေးသည့် ကိစ္စများတွင်မှသာ MapReduce ကိုအသုံးပြုရန်လိုသည်။ လက်ရှိတွင်မူ နှစ်မျိုးနှစ်စား ဟုယူဆ၍ `group by` ပြုလုပ်နိုင်သော feature များ (ထိုထက်ပို သော်လည်း) မှတ်ယူနိုင်ပါသည်။ အလွန်များပြားသော data များကို အပြိုင် process ပြုလုပ်ရာတွင်မူ Hadoop ကဲ့သို့သော distributed data processing များကို အားပိုကိုးရပြီး MongoDB နှင့်ချိတ်ဆက်အသုံးပြုရန် MongoDB connector for Hadoop လည်းရှိသည်။

Parallel data processing သည် relational database များတွင်လည်း ကောင်းလှသည်မဟုတ်ချေ။ MongoDB ၏နောက်ပိုင်း version များတွင် ပိုမိုကောင်းမွန်လာသည်ဟု မျှော်လင့်ရသည်။

Geospatial

MongoDB ၏ အခြားသော powerful ဖြစ်သော feature တစ်ခုမှာ geospatial indexes ဖြစ်ပြီး geoJSON အနေဖြင့်သော်လည်းကောင်း x နှင့် y coordinate များအဖြစ်လည်းကောင်း document များအတွင်း သိမ်းဆည်းနိုင်ပြီး `$near` သို့မဟုတ် `$within` ဖြင့် စတုရန်းဖြစ်စေ စက်ဝိုင်း

ပုံစံဖြစ်စေ ရှာဖွေနိုင်သည်။ ၎င်း feature သည် ပုံဖြင့်ရှင်းပြမှ ပို၍ နားလည်လွယ်မည်ဖြစ်ပြီး Find Restaurants with Geospatial Queries တွင် အသေးစိတ်လေ့လာနိုင်သည်။

Tools and Maturity

MongoDB တွင် language များစွာအတွက် driver များတည်ရှိရုံသာမက protocals များသည်လည်း ခေတ်မှီပြီး ရိုးရှင်းသည်။ MongoDB Tools တွင် အသုံးပြုနိုင်သည့် tools များကိုဖတ်ကြည့်နိုင်သည်။

In This Chapter

ယခု အခန်း၏ အဓိက သင်ခန်းစာမှာ MongoDB သည် အချို့သောနေရာများတွင် relational database များအစား အသုံးပြုနိုင်သည်။ ပို၍ရိုးရှင်းပြီး တည့်တိုးဖြစ်သည်အပြင် ပို၍ မြန်ဆန်ပြီး application developers များအတွက် ချည်နှောင်ခြင်းမှာ ပို၍နည်းပါသည်။ *လူတွေအနေဖြင့် Data Storge နယ်ပယ်တွင် MongoDB ၏နေရာမှာ ဘယ်နားမှာလဲ* ဟုမေးပါက အဖြေကရိုးရှင်းသည်။ *အလယ်တည့်တည့်မှာ* ဖြစ်သည်။

Chapter 6 - Aggregating Data

Aggregation Pipeline

Aggregation pipeline များသည် collection အထဲမှ document များကို ပြောင်းလဲ ပေါင်းစပ် အသုံးပြုနိုင်သည်။ documents များကို Unix မှ “pipe” နှင့် သဘောတရားချင်းဆင်တူသည့် pipeline တစ်ခုမှ တစ်ခုသို့ ပို့၍ အသုံးပြုကြသည်။

အလွယ်ကူဆုံးသော aggregation သောဥပမာမှာ သင့်နှင့်ရင်းနှီးနေပြီးသား ဖြစ်သော SQL မှ `group by` ဖြစ်သည်။ ကျွန်တော်တို့ `count()` ကိုတွေ့ပြီး ဖြစ်၍ unicorn များမှ မည်သည့်အရေအတွက်သည် အထီးဖြစ်မည် အမ ဖြစ်မည် ကို သိရှိနိုင်မည်နည်း။

```
db.unicorns.aggregate([{$group:{_id:'$gender',
    total: {$sum:1}}}])
```

Shell အတွင်းတွင် `aggregate` ဟုသော helper သည် pipeline operator များကို array အနေဖြင့် လက်ခံသည်ကိုတွေ့ရှိရမည်။ ရိုးရိုး group by တစ်ခု၏ အရေအတွက်ကိုသာ အလိုရှိပါက `$group` ဟုခေါ်သည့် operator တစ်ခုသာလိုသည်။ ၎င်းသည် SQL တွင်ရှိသည့် `GROUP BY` နှင့် အတူတူပင်ဖြစ်ပြီး `_id` field သည် မိမိတို့ group ပြုလုပ်လိုသော field (ယခုနေရာတွင် `gender`) နှင့် အခြား fields များသည် aggregation ၏ ရလဒ်ဖြင့် တွဲဖက်

ဖော်ပြလေ့ရှိပြီး ယခု ရလဒ်တွင်မူ `$sum` ဟု တိုက်ဆိုင်သည့် document တိုင်းကို ၁ ပေါင်းထည့်မည်ဖြစ်သည်။ သတိထားမိမည်က `_id` ကို assign လုပ်ထားသည်က `$gender` ဖြစ်ပြီး `gender` မဟုတ်ပါ။ `$` ထို field ၏ value ကိုအစားထိုးမည်ဟု သတ်မှတ်ထားခြင်းဖြစ်သည်။

တခြားဘယ် pipeline operator တွေရှိသေးလဲ? `$group` အပြင် အများဆုံး အသုံးပြုလေ့ရှိသည့် အရာမှာ `$match` ဖြစ်ပြီး ၎င်းသည် `find` နှင့်အတူတူ ပင်ဖြစ်ပြီး ၎င်း၏ရလဒ်မှ ကိုက်ညီသော document များကိုသာ သယ်ဆောင်သွားပြီး ကျန်သည်များကို ချန်လှစ်ထားခဲ့သည်။

```
db.unicorns.aggregate([{$match: {weight:{$lt:600}}},
    {$group: {_id:'$gender',  total:{$sum:1},
      avgVamp:{$avg:'$vampires'}}},
    {$sort:{avgVamp:-1}} ])
```

တခြား pipeline operator တစ်ခုမှာ `$sort` ဖြစ်ပြီး သင်ထင်ထားသည့် အတိုင်းပင် ဖြစ်သည်။ ထို့အပြင် `$skip` နှင့် `$limit` ။ ထိုအပြင် `$group` operator နှင့်တွဲဖက် အသုံးပြုလေ့ရှိသည့် `$avg` တို့လည်းရှိသည်။

MongoDB ၏ array များသည် powerful ဖြစ်ပြီး ၎င်းတို့၏ အတွင်းပိုင်းထဲထိ value များကို aggregate ပြုလုပ်နိုင်သည်။ ထိုသို့မပြုလုပ်မှီ “ဖြန့်ချ” ကာ ရေတွက်နိုင်သည်။

```
db.unicorns.aggregate([{$unwind:'$loves'},
    {$group: {_id:'$loves',  total:{$sum:1},
    unicorns:{$addToSet:'$name'}}},
    {$sort:{total:-1}},
    {$limit:1} ])
```

အခုဆိုရင်ဖြင့် unicorn အများစုမှာ မည်သည့် food item ကိုအကြိုက်ဆုံးဖြစ်သည်နှင့် ၎င်းတို့ကို ကြိုက်သည့် unicode အမည်များကိုပါ မြင်တွေ့နိုင်မည်ဖြစ်သည်။ `$sort` နှင့် `$limit` ကိုပေါင်းစပ်အသုံးပြုခြင်းဖြင့် ထိပ်ဆုံး ဘယ်နှစ်ခု ဆိုသော မေးခွန်းများ၏ အဖြေများကို ရှာနိုင်သည်။

နောက်ထပ် powerful ဖြစ်သည့် pipline operator တစ်ခုမှာ `$project` ဖြစ်ပြီး (`find` မှ ဒုတိယ parameter နှင့်ဆင်တူသည်) မည့်သည့် field များကို ရယူမည်နည်းနှင့် ရှိပြီးသား field မှ value များကို calculate ပြုလုပ်ပြီး field အသစ်များတည်ဆောက်နိုင်သည်။ ဥပမာ သင့်အနေဖြင့် ပျမ်းမျှမပြုလုပ်မီ fields များအချင်းချင်း ပေါင်းခြင်း ၊ သို့မဟုတ် fields များကို ဆက်စပ်ခြင်းများကို ပြုလုပ်နိုင်သည်။

၎င်းသည် aggregation နှင့်ပြုလုပ်နိုင်သည်၏ အပေါ်ယံပင်ရှိသေးသည်။ MongoDB ၏ version အသစ်ထွက်လာသည်နှင့်အမျှ aggregation တွင် operator အသစ်များ ပါပါလာပြီး ပို၍ powerful ဖြစ်လာသည်။ aggregate command သည် find ကဲ့သို့ပဲ cursor return ပြန်ပြီး collection အသစ်ဆီသို့ export ထုတ်ယူလိုပါက `$out` ဟုသော pipeline operator ကိုအသုံးပြုနိုင်သည်။ MongoDB တွင် support ပြုလုပ်သော pipeline operator အကြောင်းကို ဥပမာနှင့် တကွ သိရှိလိုပါက MongoDB manual တွင်ဖတ်နိုင်ပါသည်။

MapReduce

Mapreduce သည် data process ပြုလုပ်ရာတွင် နှစ်ဆင့်ခံ process တစ်ခု ဖြစ်သည်။ ပထမဦးစွာ map ပြီးနောက် reduce ပြုလုပ်သည်။ mapping အဆင့်တွင် input document များကို transform ပြုလုပ်ပြီး key => value အတွဲများအဖြစ် (key သက်သက် value သက်သက် သည်ပို၍ ရှုပ်ထွေးသည်) ပြန်ထုတ်ပေးပြီး ထို key value အတွဲများမှာ key အနေဖြင့် group by ပြုလုပ်ပြီး တူညီသော key များမှာ array တစ်ခုတည်းတွင် ကျရောက်လေ့ရှိသည်။ ထိုနောက် reduce မှာ key နှင့် array value ကိုရယူပြီး နောက်ဆုံး ရလဒ်ကို ထုတ်ပေးသည်။ map နှင့် reduce function များကို Javascript ဖြင့်ရေးသားထားသည်။

MongoDB တွင် `mapReduce` command ကို collection တစ်ခုတွင် အသုံးပြုနိုင်သည်။ `mapReduce` သည် map function တစ်ခု ၊ reduce function တစ်ခု နှင့် output directive တစ်ခု ပါရှိရန်လိုသည်။ Shell တွင် Javascript function တစ်ခု ပြုလုပ်ပြီး pass နိုင်သည်။ တချို့ library တွင်မူ function ကို string အနေဖြင့် (ရုပ်ဆိုးသော်လည်း) pass နိုင်သည်။ တတိယ parameter တစ်ခုပါရှိပြီး ၎င်းမှာ အပို options များဖြစ်သည့် မိမိတို့စိတ်ကြိုက် filter ၊ sort နှင့် limit ပြုလုပ်နိုင်သည်။ ထိုအပြင် `reduce` အဆင့်ပြီးသွားပါက နောက်ဆုံးတွင်ပြုလုပ်လိုသည်များကို `finalize` method အတွင်းတွင် ပြုလုပ်နိုင်သည်။

Aggregation အတော်များများအတွက် MapReduce အသုံးပြုရန် မလိုအပ်လှသော်လည်း လိုအပ်လာပါက ကျွန်တော်ဘလော့ နှင့် MongoDB manual များတွင်ဖတ်ရှုနိုင်သည်။

ယခု အခန်းတွင်

ယခု အခန်းတွင် MongoDB ၏ aggregation capabilities အကြောင်းပြောခဲ့ပြီးဖြစ်သည်။ Aggregation Pipeline များသည် သူ့၏တည်ဆောက်ကို နားလည်သည်နှင့် ရေးရသည်မှာ ရိုးရှင်းပြီး data များကို အုပ်စုဖွဲ့ရာတွင် အလွန် powerful ဖြစ်သည်။ MapReduce များမှာ ပို၍နားလည်ရန် ခက်ခဲသော်လည်း Javascript ဖြင့်ရေးသားနိုင်သောကြောင့် ၎င်း၏လုပ်ဆောင်နိုင်စွမ်းမှာ အကန့်အသတ်မရှိပေ။

အခန်း (၇) Performance နှင့် Tool များ

နောက်ဆုံးအခန်းတွင်မူ MongoDB ၏ performance နှင့် developer များအတွက် အသုံးပြုနိုင်သော tool များအကြောင်း ပြောသွားမည်။ ၎င်း topic များနှင့်ပတ်သတ်၍ အသေးစိတ် ဆင်းသွားမည်မဟုတ်သော်လည်း ၎င်းတို့၏ အဓိကကျသည့် အချက်များကို ကြည့်သွားမည်ဖြစ်သည်။

Index များ

အစောပိုင်းကတည်းက collection တစ်ခုမှ index များကိုထုတ်ပြသည့် `getIndexes` ဆိုသည့် command နှင့် ရင်းနှီးပြီးဖြစ်သည်။ MongoDB မှ index များသည် relational database များမှ index များနှင့် အတူတူပင်ဖြစ်ပြီး query နှင့် sorting ၏ performance ကိုပိုမိုကောင်းမွန်စေရန်ရည်ရွယ်သည်။ Index များကို `ensureIndex` ကို အသုံးပြုပြီး ဖန်တီးနိုင်ကာ

```
// where "name" is the field name
db.unicorns.ensureIndex({name: 1});
```

`dropIndex` ကိုအသုံးပြုပြီး drop ပြုလုပ်နိုင်သည်။

```
db.unicorns.dropIndex({name: 1});
```

ဒုတိယ parameter အနေဖြင့် `unique` ကို `true` ဟုပြောင်းလဲပြီး unique index တစ်ခုဖန်တီးနိုင်သည်။

```
db.unicorns.ensureIndex({name: 1},
    {unique: true});
```

Index များကို embedded field များနှင့် array များတွင်လည်း ဖန်တီးနိုင်သည်။ ထိုအပြင် compound index တစ်ခုကို အောက်ပါအတိုင်း ဖန်တီးနိုင်သည်။

```
db.unicorns.ensureIndex({name: 1,
    vampires: -1});
```

Index တစ်ခု၏ direction (1 မှာ အစဉ်အတိုင်း -1 မှာ ပြောင်းပြန်) သည် single key index အတွက် သိပ်အရေးမပါလှသော်လည်း တစ်ခုထက်ပိုသော အရာများဖြင့် sort ပြုလုပ်သောအခါ compound index များတွင်မူ အတော် ကွာခြားသည်။

The [indexes page] has additional information on indexes.

Explain

မိမိတို့ အသုံးပြုသည့် query များသည် index ကို အသုံးပြုခြင်းရှိမရှိကို cursor အတွင်းရှိ `explain` method ကို အသုံးပြု၍ သိရှိနိုင်သည်။

```
db.unicorns.find().explain()
```

ရလဒ်အနေဖြင့် `BasicCursor` အနေဖြင့် မြင်တွေ့ရမည် ဖြစ်ပြီး (index မပြုလုပ်ရသေးဟု ဆိုလိုသည်) object 12 ခုကို scan ပြုလုပ်ကြောင်း၊

ဘယ်လောက်ကြာသည်၊ index ရှိခဲ့ပါက ဘယ် index ကိုအသုံးပြုသည် စသဖြင့် အခြားအသုံးဝင်သော အချက်အလက်များကို ဖော်ပြပေးသည်။

Index ကိုအသုံးပြုသော query ဖြင့်ရှာကြည့်ပါက index ဖြင့် ပတ်သတ်သော အချက်အလက်များ အပြင် `BtreeCursor` ဟု မြင်တွေ့ရမည် ဖြစ်သည်။

```
db.unicorns.find({name: 'Pilot'}).explain()
```

Replication

MongoDB ၏ replication သည် relational database များကဲ့သို့ပင် ခပ်ဆင်ဆင် အလုပ်လုပ်သည်။ production deployment အားလုံးသည် သုံးခုနှင့် ၎င်းထက်ပိုသော တူညီသော data များတည်ရှိသည် server များဖြင့် replica set များဖြစ်သင့်သည်။ Write များကို Primary ဖြစ်သည့် server တစ်ခုသို့ပို့ပြီး ထို့မှတဆင့် တခြား secondary server များသို့ asynchronous အနေဖြင့် replicate ပြုလုပ်သည်။ Read များကို secondary မှ ပြုလုပ်မည် မပြုလုပ်မည်ကို config ပြုလုပ်နိုင်ပြီး ၎င်းသည် primary သို့ အလှူပ်ရှုပ်မည့် query တချို့ကို လျော့ကျစေသည်။ Primary ကျသွားပါက Secondary များအနက် တစ်ခုသည် Primary အသစ်အဖြစ် ရွေးကောက်တင်မြောက်ခြင်းခံရမည် ဖြစ်သည်။ ထပ်၍ ပြောရမည်ဆိုလျင်

MongoDB ၏ replication ယခု စာအုပ် scope မှကျော်လွန်သွားပြီ ဖြစ်သည်။

Sharding

MongoDB သည် sharding ကိုအလိုအလျောက် support လုပ်သည်။ sharding သည် server သို့မဟုတ် cluster များမှ data များကို များပြားလာ ပါက scale လုပ်နိုင်ရန် data များကို ပိုင်းခြား၍ သိမ်းဆည်းပေးသော စနစ် ဖြစ်သည်။ အလွယ်ပြောရလျှင် user ၏ အမည်များကို သိမ်းရသည်ဆိုပါစို့ A မှ M အထိကို server တစ်လုံးတွင်သိမ်း၍ ကျန်သည်ကို နောက်တလုံး တွင် သိမ်းခြင်း ကဲ့သို့ပင်။ MongoDB ၏ sharding လုပ်နိုင်စွမ်းသည် ထို ထက်ပို၍ကျယ်ပြန့်သော်လည်း ယခုစာအုပ်တွင်တော့ အသေးစိတ် မဖော်ပြ တော့ပေ။ သို့သော် ထို feature များရှိသည်ဟု သိထားရန်လိုပြီး လိုအပ်ပါ က အသုံးပြုနိုင်ရန်ဖြစ်သည်။

replication သည် performance အတွက် တပိုင်းတစ ကောင်းမွန် သော်လည်း (အားစိုက်ရမည့် query များကို secondary များတွင် run ချင်း ဖြင့်သော်လည်းကောင်း တချို့သော query များအတွက် latency လျှော့ချ ခြင်းဖြင့်သော်လည်းကောင်း) ၎င်းအဓိက ရည်ရွယ်ချက်မှာ high availability အတွက်ဖြစ်သည်။ Sharding မှာမူ MongoDB cluster များကို

scale ပြုလုပ်ရန်ရွယ်ထားပြီး ၎င်းနှစ်ခုကို ပေါင်းစပ် အသုံးပြုခြင်းဖြင့် မိမိတို့လိုလားသော ရလဒ်ကို ရရှိအောင် လုပ်ယူရမည်။

Stats

database ၏ အချက်အလက်များကို `db.stats()` ဟုရိုက်ကြည့်နိုင်သည်။ အချက်အလက်အများစုမှာ database ၏ size နှင့်ပတ်သတ်သည်များဖြစ်သည်။ collection ၏ အချက်အလက်ကိုလည်း ဥပမာ `unicorns` ကိုသိချင်ပါက `db.unicorns.stats()` ဟုရိုက်ရှာနိုင်ပြီး collection နှင့် index များအကြောင်းကိုဖော်ပြပေးမည်ဖြစ်သည်။

Profiler

MongoDB ၏ profiler ကိုအောက်ပါအတိုင်း enable ပြုလုပ်နိုင်သည်။

```
db.setProfilingLevel(2);
```

enable ပြုလုပ်ပြီးနောက် အောက်ပါ command ကို run နိုင်သည်။

```
db.unicorns.find({weight: {$gt: 600}});
```

ထိုနောက် profiler ကို အောက်ပါအတိုင်းကြည့်နိုင်သည်။

```
db.system.profile.find()
```

မည်သည်ကနှင့် ဘယ်အချိန်က run သည်၊ document မည်မျှ scan ပြုလုပ်သည်၊ မည်မျှ data များ return သည်ကိုဖော်ပြပေးသည်။

`setProfilingLevel` ၏ parameter ကို `0` ထားပြီး profiler ကို disable ပြုလုပ်နိုင်သည်။ `1` ဟုထားပါက 100 milisecond ထက်များသော အချက်အလက်များကို profile ပြုလုပ်ထားမည်ဖြစ်သည်။ 100 milisecond သည် default ဖြစ်၍ မိမိတို့ဖာသာ တခြားအချိန်ကို ထားလိုပါက ဒုတိယ parameter အနေဖြင့် အောက်ပါအတိုင်း ပြင်နိုင်သည်။

```
//profile anything that takes
//more than 1 second
db.setProfilingLevel(1, 1000);
```

Backups and Restore

MongoDB ၏ `bin` folder အတွင်းတွင် `mongodump` ဟုသော executable ပါရှိသည်။ `mongodump` ကို execute ပြုလုပ်ခြင်းဖြင့် localhost ကို connect ပြုလုပ်ပြီး database များကို `dump` ဟုသော subfolder ထဲတွင် backup ပြုလုပ်သွားမည်။ `mongodump --help` ဟုရိုက်ပြီး အခြား parameter များကိုလေ့လာနိုင်သည်။ အသုံးများသော option များမှာ `--db DBNAME` ဟု၍ မိမိတို့ အလိုရှိသော database ကိုသာ backup ပြုလုပ်ခြင်း နှင့် `--collection COLLECTIONAME` ဟု၍ မိမိတို့ စိတ်ကြိုက် collection ကို backup ပြုလုပ်ခြင်းဖြစ်သည်။ ထိုနောက် bin folder အတွင်းရှိ `mongorestore` ဟုသော executable နောက်တစ်ခုကို အသုံးပြု၍ backup ပြုလုပ်ပြီးသားကို restore ပြုလုပ်နိုင်သည်။ ထိုအပြင် `--db` နှင့် `--collection` flag များကိုအသုံးပြု၍ ရွေးချယ် restore ပြုလုပ်နိုင်သည်။

`mongodump` နှင့် `mongorestore` တို့သည် MongoDB ၏ native format ဖြစ်သော BSON ဖြင့်အလုပ်လုပ်သည်။

ဥပမာ `learn` database ကို `backup` folder အတွင်းသို့ backup ပြုလုပ်နိုင်သည်။ (၎င်းသည် သီးသန့် executable ဖြစ်၍ mongo shell မှ မဟုတ်ပဲ အပြင်မှ command/terminal window မှ ခေါ်ရမည်ဖြစ်သည်)

```
mongodump --db learn --out backup
```

`unicorn` collection တစ်ခုတည်းသာ restore ပြုလုပ်လိုပါက အောက်ပါအတိုင်း ရိုက်နိုင်သည်။

```
mongorestore --db learn --collection unicorns \
    backup/learn/unicorns.bson
```

ထပ်၍ မှတ်သားရန်လိုအပ်သည်မှာ `mongoexport` နှင့် `mongoimport` ဟူသော executable နှစ်ခုလည်းရှိသေးပြီး ၎င်းတို့သည် JSON သို့မဟုတ် CSV သို့ export ၊ import ပြုလုပ်ရာတွင် အသုံးပြုသည်။ ဥပမာ JSON output ကိုလိုချင်ပါက အောက်ပါအတိုင်း ပြုလုပ်နိုင်သည်။

```
mongoexport --db learn --collection unicorns
```

CSV ကိုလိုချင်ပါက အောက်ပါအတိုင်းဖြစ်မည်။

```
mongoexport --db learn \
    --collection unicorns \
    --csv --fields name,weight,vampires
```

သတိပြုရန်မှာ `mongoexport` နှင့် `mongoimport` သည် သင့်၏ data ကို အမြဲတမ်း represent ပြုလုပ်နိုင်မည်မဟုတ်ပေ။ `mongodump` နှင့် `mongorestore` ကသာ တကယ့် backup အတွက် အသုံးပြုသင့်သည်။ အသေးစိတ်ကို backup ပြုလုပ်နိုင်သည့် option များတွင်ဖတ်ရှုနိုင်သည်။

ယခုအခန်းတွင်

ယခုအခန်းတွင် MongoDB အသုံးပြုပါက လိုအပ်သော tools များနှင့် performance အကြောင်းကို လေ့လာပြီးဖြစ်သည်။ အကုန်လုံး အသေးစိတ် မထိတွေ့ခဲ့ရသော်လည်း အသုံးများသည်များကို ဖော်ပြခဲ့သည်။ MongoDB မှ indexing သည် အခြား relational database များ၏ indexing နှင့် ဆင်တူသည်ဖြစ်ပြီး တခြား tools များမှာလည်း ထိုနည်းတူပင်။ သို့သော် MongoDB ကိုအသုံးပြုပါက ၎င်းတို့ကို လွယ်ကူစွာ အသုံးပြုနိုင်သည်ကို သတိထားမိမည်ဖြစ်သည်။

နိဂုံးချုပ်

ယခုအချိန်တွင် သင့်အနေဖြင့် လက်တွေ့တွင် MongoDB ကိုစတင်အသုံးပြုနိုင်ရန် အသင့်ဖြစ်နေပြီဟု ယူဆရသည်။ ပြောခဲ့သည်များထက်ပိုသော အချက်များသည် MongoDB တွင်ပါရှိသော်လည်း ယခုလေ့လာထားခဲ့သည်ကို အဓိကထား၍ ပေါင်းစပ်အသုံးပြုနိုင်ရန် ကြိုးစားသင့်သည်။ MongoDB website တွင် အသုံးဝင်သည့် အချက်အလက်များကို ဖတ်ရှုနိုင်ပြီး တရားဝင် MongoDB user group တွင်လည်း မေးမြန်းနိုင်ပါသည်။

NoSQL သည် လိုအပ်ချက်အရ ပေါ်ပေါက်လာရုံသာမက ဖြေရှင်းရန်နည်းလမ်းအသစ်များကို သယ်ဆောင်လာသည်။ ၎င်းသည် ကျွန်တော်တို့အသက်မွေးဝမ်းကြောင်းမှုဖြစ်သည့် နည်းပညာသည် အမြဲတမ်းပြောင်းလဲနေပြီး ကျွန်တော်တို့ တခါတရံ ကျရှုံးမှုများကို လက်မခံပဲ မစမ်းသပ်ကြည့်ပါက အောင်မြင်နိုင်မည် မဟုတ်ပါ။ ထိုအချက်သည် ကျွန်တော်တို့၏ professional ဘဝများကို ဦးဆောင်နိုင်မည် အချက်ပင်ဖြစ်သည်။