بسمه تعالی

پروژه‌ی مطالعاتی

جرم شناسی پایگاه‌داده

رویدادنگاری، ممیزی و جرم‌شناسی در

پایگاه داده‌ی غیررابطه‌ای مانگو

فهرست مطالب

۱ مقدمه

پایگاه داده‌های رابطه‌ای اغلب برای ذخیره داده‌های مهم سازمان‌ها یا خدمات تحت وب استفاده می‌شوند. بااین‌حال، استفاده از آن‌ها برای تحلیل داده‌های حجیم، تحلیل رویدادها، شبکه‌های اجتماعی و برنامه‌های موبایل بسیار هزینه‌بر و پیچیده خواهد بود و کارایی لازم را نیز نخواهد داشت. برای مقابله با این محدودیت، انواع مختلفی از پایگاه داده‌های غیررابطه‌ای ارائه و جایگزین پایگاه داده‌های رابطه‌ای شده‌اند.

پایگاه داده‌های غیررابطه‌ای ابزاری کارآمد برای ذخیره‌سازی و دسترسی به داده‌های حجیم هستند چراکه کارگزارهای آن دارای قابلیت مقیاس‌پذیری افقی هستند. مدل داده‌ای این پایگاه داده‌ها شمای ثابتی ندارند و کاربران می‌توانند به‌سادگی، مدل داده‌ای برنامه خود را به صورت پویا تغییر دهند. این ویژگی پایگاه داده‌های غیررابطه‌ای منجر به استفاده روزافزون از آن‌ها در تحلیل‌های بلادرنگ، خدمات تحت وب مانند خدمات شبکه‌های اجتماعی، برنامه‌های کاربردی موبایل و ذخیره‌سازی داده‌های ماشین‌های تولیدکننده رویداد و اینترنت اشیا شده است.

از طرفی دیگر، استفاده روزافزون از پایگاه داده‌های غیررابطه‌ای احتمال موردحمله قرار گرفتن این پایگاه داده‌ها و دسترسی غیرمجاز به اطلاعات شخصی افراد و ذخیره‌ی آن‌ها در پایگاه داده غیررابطه‌ای، به‌منظور سرقت هویت را افزایش می‌دهد. به‌عنوان‌مثال در سال ۲۰۱۵، اطلاعاتی از یک شرکت نرم‌افزاری ایتالیایی فاش شد که بر اساس آن، این شرکت برنامه‌ی سیستم کنترل از راه دور RCS که به‌منظور جاسوسی تولید شده بود را به دولت‌های مختلفی فروخته است. این برنامه دستگاه‌های شخصی مانند کامپیوترها و تلفن‌های هوشمند را هک و اطلاعاتی از قبیل عکس‌ها، مستندات، ورودی‌های صفحه‌کلید و صوت مکالمات انجام‌شده را در پایگاه داده‌ی مانگو ذخیره می‌کرده است.

برخلاف مطالعات زیادی که در حوزه جرم‌شناسی پایگاه داده‌های رابطه‌ای (مانند اوراکل، MySQL و MSSQL) انجام‌شده، در حوزه پایگاه داده‌های غیررابطه‌ای تلاش کمی صورت گرفته است. بیشتر تحقیقات انجام‌شده در مورد پایگاه داده‌های غیررابطه‌ای به مقایسه کارایی، تفاوت آن با پایگاه داده رابطه‌ای و خصوصیات ویژه این پایگاه داده معطوف شده است.

در این گزارش راه‌کارهای ارائه شده در مقالات مرتبط با این موضوع بیان شده است. در ادامه چارچوب ارائه شده برای جرم‌شناسی سامانه‌های مدیریت پایگاه داده غیررابطه‌ای سند محور به‌ویژه پایگاه داده‌ی مانگو

به عنوان نمونه‌ای از این پایگاه داده‌ها پرداخته خواهد شد. منظور از پایگاه داده سند محور، پایگاه داده‌ای است که داده‌های خود را در غالب مستنداتی نظیر XML و JSON ذخیره می‌کند.

۲ آشنایی با پایگاه داده‌ی مانگو

پایگاه داده‌ی مانگو یک سیستم مدیریت پایگاه داده‌ی متن‌باز، از گونه پایگاه داده‌های غیررابطه‌ای سند محور است. کارایی بالای این پایگاه داده و امکان توسعه آن برای کار در حالت توزیع‌شده باعث گردیده تا این پایگاه داده غیررابطه‌ای توسط طیف وسیعی از کاربردها مورداستفاده قرار گیرد؛ به‌نحوی‌که بر اساس گزارش DB-ENGINES، اکنون به عنوان چهارمین پایگاه داده محبوب (پس از پایگاه داده‌های MySQL،Oracle و SQL server) شناخته می‌شود. در حال حاضر از پایگاه داده‌ی مانگو در برنامه‌های مختلف و شناخته‌شده‌ای مانند eBay[۱]، Expedia[۲]، MTV[۳]، McAfee[٤] و کمیسیون ارتباطات فدرال آمریکا استفاده می‌شود.

در پایگاه داده‌ی مانگو شمایی وجود ندارد و هر مستند در قالب JSON ذخیره می‌گردد. JSON یک قالب استاندارد باز برای تبادل داده‌های کم است که بر اساس زبان جاوا اسکریپت بنا نهاده شده است. در این قالب، یک شیء از مجموعه‌ای بدون ترتیب از زوج‌های نام/مقدار تشکیل شده است. در شکل ٤ سطح جزئیات پیغام‌های رویدادنگاری نمونه‌ای از این فایل که حاوی اطلاعات شخصی است نمایش داده شده است.

[۱] فروشگاه آنلاین
[۲] سرویس خدمات مسافرتی آنلاین
[۳] کانال موسیقی
[٤] شرکت امنیتی جهانی

```
{
   "name" : "Smith",
   "address":{
      "streetaddress":"3400 N Charles St, Mason Hall",
      "city":"Baltimore",
      "country":"USA",
      "postalCode":"MD 21218-2683"
   },
   "phoneNumbers": [
      "410-516-8000",
      "410-516-8171",
   ]
}
```

شکل ۱ نمونه‌ای از فایل JSON

ازآنجاکه پایگاه داده‌ی مانگو شمای از پیش تعریف‌شده‌ای ندارد، می‌توان مستندات JSON را به هر قالبی که هست در آن وارد کرد. بنابر همین ویژگی است که مدل داده‌ای پایگاه داده‌های مانگو انعطاف‌پذیر است و می‌توانند تغییر کنند.

این سیستم از مجموعه‌ای از برنامه‌ها و سرویس‌ها تشکیل شده است که برخی از مهم‌ترین آن‌ها که در این سند به آن‌ها خواهیم پرداخت عبارت‌اند از:

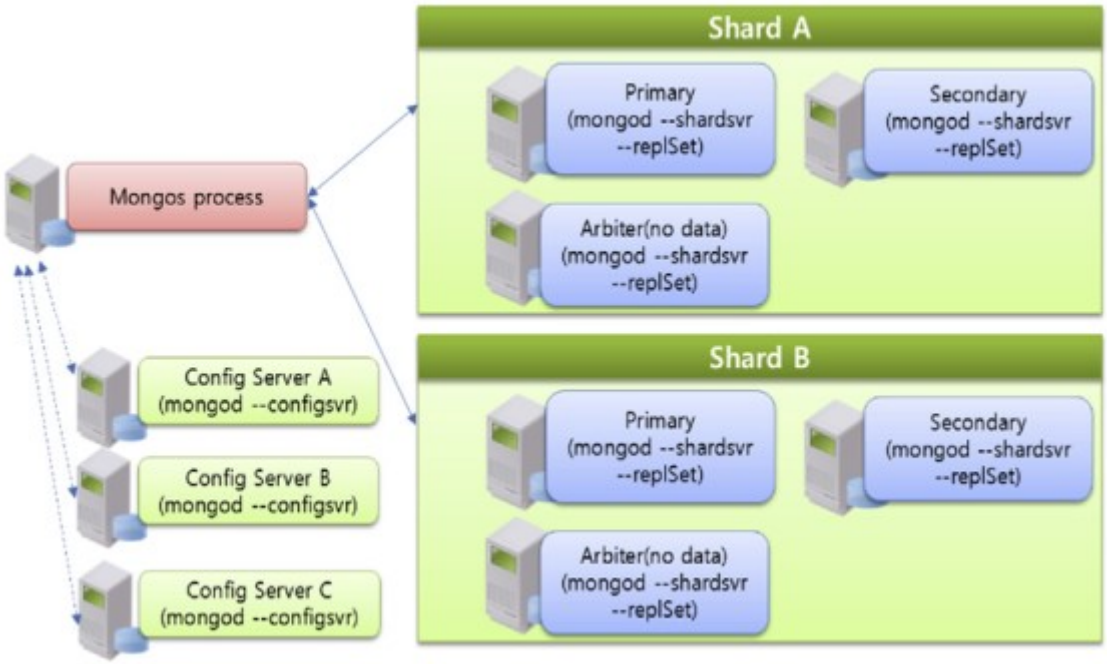
- mongod: سرویس اصلی پایگاه داده‌ی مانگو است که وظیفه‌ی مدیریت داده‌ها، درخواست‌های کاربران و نیز سایر عملیات مدیریتی را بر عهده دارد.
- mongo: واسط مبتنی بر خط فرمان است که امکان ارتباط کاربر با mongod را فراهم می‌نماید.
- mongos: درصورتی‌که پایگاه داده‌ی مانگو در حالت توزیع‌شده کار کند (Sharding)، داده‌ها در چند سایت (میزبان) مختلف قرار می‌گیرند. سرویس mongos با دریافت یک درخواست، تعیین می‌کند که این درخواست باید به کدام سایت ارجاع داده شود.

به دلیل اینکه پایگاه داده‌ی مانگو از محیط توزیع‌شده پشتیبانی می‌کند، می‌توان آن را به گونه‌های مختلفی مستقر کرد. سه نوع مختلف برای استقرار وجود دارد: مستقل، مجموعه‌ی تکثیر یا replica و خوشه‌ی توزیع‌شده یا sharded.

- در مدل استقرار مستقل، تنها یک پردازه mongod در یک کارگزار اجرا می‌شود. از این مدل معمولاً برای توسعه یا تحقیق استفاده می‌کنند و به‌ندرت برای خدمات عمومی مورداستفاده قرار می‌گیرد.
- در مدل مجموعه تکثیر، یک کارگزار اصلی و فرعی وجود دارد که چندین پردازه mongod روی آن‌ها اجرا می‌شود. در این روش داده‌ها به صورت خودکار بین کارگزار اصلی و فرعی همگام‌سازی

شده و این اطمینان را می‌دهد که در صورت شکست هر یک از کارگزارها، سرویس به صورت خودکار به کار خود ادامه خواهد داد.

- در مدل خوشه‌ی توزیع‌شده، داده به تکه‌هایی تقسیم‌شده و روی چندین کارگزار ذخیره می‌شود. علاوه بر این، داده‌ها به صورت خودکار بر اساس سایزی که باعث بهبود کارایی گردد، میان کارگزارها تقسیم می‌شود. در روش خوشه توزیع‌شده که در آن ممکن است چندین کارگزار وجود داشته باشد، داده در واحدهایی به نام shard تقسیم می‌شود. برای برقراری یکپارچگی لازم است داده‌های توزیع‌شده از طریق پردازه mongos باهم در ارتباط باشند. یک کارگزار پیکربندی نیز وجود دارد که در آن وضعیت کنونی هر shard و فراداده‌ها قرار می‌گیرد. نمونه‌ای استقرار خوشه‌ی توزیع‌شده در شکل ۵ رویدادهای ثبت‌شده در فایل mongod.log قابل‌مشاهده است.

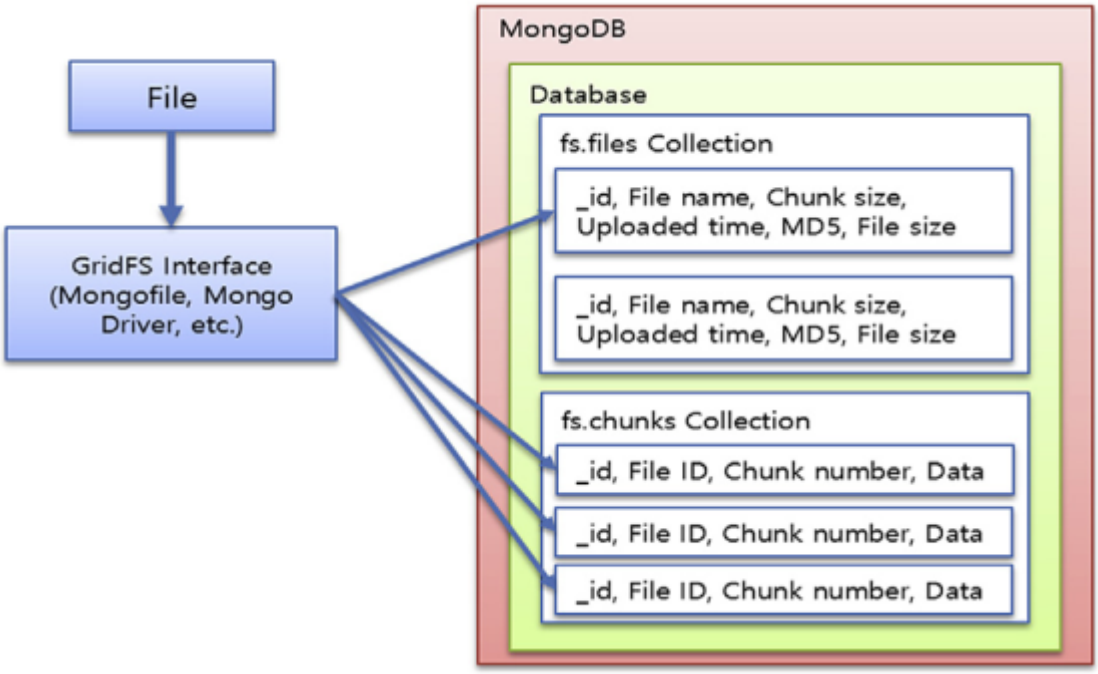


شکل ۲ نمونه‌ای از استقرار خوشه‌ی توزیع‌شده

GridFS یکی از بخش‌های ویژه‌ی پایگاه داده‌ی مانگو است که از آن برای تسهیل در ذخیره و بازیابی فایل‌ها استفاده می‌شود. GridFS یک فرمت داده‌ای نیست، اما مشخصاتی را برای ذخیره فایل‌های بزرگ در پایگاه داده‌ی مانگو تعیین می‌کند. برای بررسی لیست فایل‌ها و به دست آوردن فایل‌های ذخیره‌شده در پایگاه داده‌ی مانگو، محققین باید دید مناسبی از معماری GridFS داشته باشند.

درصورتی‌که فایلی توسط GridFS ذخیره شود، مجموعه‌ای از fs.files و fs.chunks در پایگاه داده ایجاد می‌شود. فراداده‌ها نیز، مانند نام فایل، سایز هر تکه، زمان بارگذاری، مقدار درهم فایل و سایز فایل در

fs.files به صورت یک سند ذخیره می‌شود. خود داده نیز به تعدادی تکه تقسیم می‌شود و در مجموعه‌ای از fs.chunks به عنوان مستندی، ذخیره می‌گردد. معماری GridFS در شکل ٦ خروجی دستور show log global برای مشاهده‌ی رویدادهای ثبت‌شده نشان داده شده است.



شکل ۳ معماری GridFS

۳ نحوه‌ی ثبت وقایع[۵] در پایگاه‌داده MongoDB

در این بخش انواع فایل‌های رویدادنگاری موردبررسی قرار گرفته است. همچنین گزینه‌های قابل تنظیم برای نحوه‌ی نظارت و ثبت وقایع مشخص شده‌اند. این بخش از گزارش بر روی MongoDB نسخه‌ی ۳،٤،۱۰ بر روی سیستم عامل ویندوز ۷ تهیه شده‌است.

۳-۱ انواع فایل‌های رویدادنگاری

در جدول ۱ انواع فایل‌های رویدادنگاری در انواع فایل‌های رویدادنگاری در MongoDB به‌اختصار توضیح داده‌شده‌اند.

---

۵ Logging

جدول ۱ انواع فایل‌های رویدادنگاری در MongoDB

نوع فایل رویدادنگاری	اطلاعات ثبت‌شده در فایل
فایل رویدادنگاری mongod.log	ثبت تمامی پرسمان‌ها و عملیات‌ها
فایل Oplog	ثبت عملیات‌های نوشتاری برای استفاده در تکرار[۶]

در ادامه هر یک از این فایل‌ها موردبررسی قرار گرفته‌است.

۳-۱-۱ فایل رویدادنگاری mongod.log

پس از نصب MongoDB بر روی ویندوز ۷، فایل پیکربندی برای راه‌اندازی سرویس Mongod در مسیر C:\Program Files\MongoDB\Server\۳٫٤ با نام mongod.cfg قرار داده می‌شود. محتوای این فایل به صورت زیر تنظیم می‌شود:

```
systemLog:
    destination: file
    path: c:\data\log\mongod.log
storage:
    dbPath: c:\data\db
```

همان‌گونه که از تنظیمات فوق مشخص است، پایگاه‌های داده و محتوای آن‌ها در مسیر dbPath قرار می‌گیرند و ثبت وقایع در فایل mongod.log موجود در مسیر c:\data\log انجام می‌شود. حال پس از ایجاد فولدرهای c:\data\db و c:\data\log، سرویس mongod در خط فرمان با دستور زیر پیکربندی می‌شود:

mongod.exe --config "C:\Program Files\MongoDB\Server\۳٫٤\mongod.cfg" --install

با دستور net start mongodb سرویس راه‌اندازی خواهد شد. حال می‌توان mongo.exe را اجرا و دستورات و پرسمان‌های مختلف را در آن محیط اجرا کرد.

---

[۶] Replication

۱-۱-۱-۳ سطح جزئیات پیغام‌های رویدادنگاری

پیغام‌های رویدادنگاری از اجزای زیر ساخته شده‌اند [۱]:

```
<timestamp> <severity> <component> [<context>] <message>
```

سطح جزئیات پیغام‌های خطا با اعداد صفر تا پنج تنظیم می‌شود. صفر سطح پیش‌فرض است و دارای جزئیات کمتری است، سطح پنج با جزئیات بیشتر است. برای مشاهده‌ی سطح جزئیات پیغام‌ها برای هر یک از مولفه‌ها از دستور زیر استفاده می‌شود:

```
db.getLogComponents()
```

همان طور که در شکل ٤ سطح جزئیات پیغام‌های رویدادنگاری دیده می‌شود سطح پیش‌فرض برای تمامی‌مولفه‌ها صفر است (عدد ۱- به معنی ارث‌بری سطح جزئیات از مولفه‌ی والد است)

```
> db.getLogComponents()
{
        "verbosity" : 0,
        "accessControl" : {
                "verbosity" : -1
        },
        "command" : {
                "verbosity" : -1
        },
        "control" : {
                "verbosity" : -1
        },
        "executor" : {
                "verbosity" : -1
        },
        "geo" : {
                "verbosity" : -1
        },
        "index" : {
                "verbosity" : -1
```

شکل ٤ سطح جزئیات پیغام‌های رویدادنگاری

برای تنظیم سطح جزئیات از دستور زیر استفاده می‌شود:

```
db.setLogLevel(<level>, <component>)
```

به عنوان مثال دستور زیر سطح جزئیات پیغام‌های تولید شده توسط تمامی‌مولفه‌ها را به پنج تغییر می‌دهد:

```
db.setLogLevel(5)
```

برای آنکه تمامی پرسمان‌های اجرایی ثبت شوند، کافی است سطح رویدادنگاری به یک تغییر داده شود.

۲-۱-۱-۳ پروفایلر در پایگاه‌داده

پروفایلر[۷] داده‌های ریزدانه در مورد عملیات نوشتن در MongoDB و دستورات پایگاه‌داده در حین اجرای mongod را جمع‌آوری می‌کند. پروفایلر تمامی داده‌های جمع‌آوری شده را در مجموعه‌ی system.profile می‌نویسد. لازم به ذکر است که پروفایلر به صورت پیش‌فرض غیرفعال است [۱].

پروفایلر سه سطح دارد: سطح صفر پروفایلر را غیرفعال می‌کند؛ سطح یک، عملیات‌هایی که کند هستند و اجرای آن‌ها از حد آستانه‌ای بیشتر زمان می‌برد را ثبت می‌کند و سطح دو تمامی عملیات‌ها را ثبت می‌کند. سطحی که انتخاب می‌شود سبب می‌شود تا کارگزار، محتوای پرسمان‌ها را در فایل رویدادنگاری نیز ثبت کند. به منظور ثبت تمامی عملیات‌ها در فایل رویدادنگاری، از دستور زیر استفاده می‌شود [۱]:

```
db.setProfilingLevel(2,0)
```

نمونه ای از رویداد ثبت‌شده در شکل ۵ نمایش داده‌شده‌است:

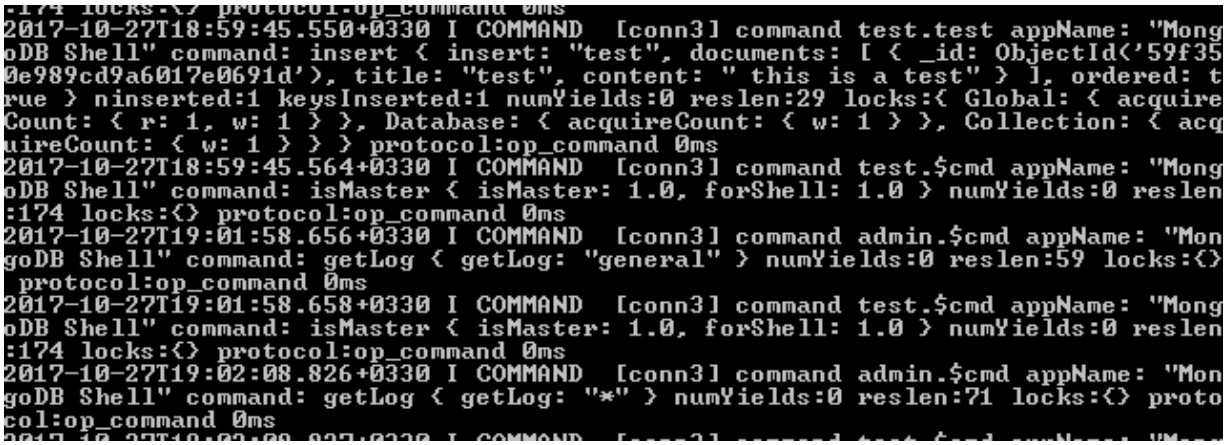
```
isMaster { isMaster: 1.0, forShell: 1.0 } numYields:0 reslen:174 locks:{} protocol:op_command 0ms
insert { insert: "test", documents: [ { _id: ObjectId('59f350e889cd9a6017e0691b'), title: "test", content:
isMaster { isMaster: 1.0, forShell: 1.0 } numYields:0 reslen:174 locks:{} protocol:op_command 0ms
insert { insert: "test", documents: [ { _id: ObjectId('59f350e889cd9a6017e0691c'), title: "test", content:
isMaster { isMaster: 1.0, forShell: 1.0 } numYields:0 reslen:174 locks:{} protocol:op_command 0ms
insert { insert: "test", documents: [ { _id: ObjectId('59f350e989cd9a6017e0691d'), title: "test", content:
isMaster { isMaster: 1.0, forShell: 1.0 } numYields:0 reslen:174 locks:{} protocol:op_command 0ms
 getLog { getLog: "general" } numYields:0 reslen:59 locks:{} protocol:op_command 0ms
isMaster { isMaster: 1.0, forShell: 1.0 } numYields:0 reslen:174 locks:{} protocol:op_command 0ms
 getLog { getLog: "*" } numYields:0 reslen:71 locks:{} protocol:op_command 0ms
isMaster { isMaster: 1.0, forShell: 1.0 } numYields:0 reslen:174 locks:{} protocol:op_command 0ms
 getLog { getLog: "global" } numYields:0 reslen:102380 locks:{} protocol:op_command 0ms
isMaster { isMaster: 1.0, forShell: 1.0 } numYields:0 reslen:174 locks:{} protocol:op_command 0ms
 getLog { getLog: "global" } numYields:0 reslen:102616 locks:{} protocol:op_command 0ms
isMaster { isMaster: 1.0, forShell: 1.0 } numYields:0 reslen:174 locks:{} protocol:op_command 0ms
```

شکل ۵ رویدادهای ثبت‌شده در فایل mongod.log

برای مشاهده‌ی رویدادهای ثبت‌شده علاوه‌بر فایل رویدادنگاری mongod.log می‌توان از دستور زیر نیز استفاده کرد (شکل ٦):

```
show log global
```

[۷] Profiler



شکل ٦ خروجی دستور show log global برای مشاهده‌ی رویدادهای ثبت‌شده

۲-۱-۳ Oplog

Oplog یک مجموعه[۸] است که تمامی عملیات‌هایی که به نحوی داده‌های ذخیره شده در پایگاه‌داده را تغییر می‌دهند را ثبت می‌کند. MongoDB عملیات‌های پایگاه‌داده را بر روی کارگزار اصلی اعمال و عملیات‌های اجرا شده را بر روی Oplog در کارگزار اصلی ذخیره می‌کند. اعضای ثانویه، Oplog را کپی و عملیات‌های درون آن را اعمال می‌کنند. تمامی اعضای مجموعه‌ی تکرار[۹] شامل یک کپی از Oplog هستند [۱].

٤ ممیزی[۱۰] در پایگاه‌داده‌ی MongoDB

MongoDB نسخه‌ی Enterprise، دارای قابلیت ممیزی از عملیات‌های مختلف است. ممیزی به مدیران و کاربران این امکان را می‌دهد که فعالیت‌های سیستمی را ردگیری کنند [۱].

هنگامی‌که ممیزی فعال شود، موارد زیر ثبت خواهند شد:

- شما (DDL)
- مجموعه‌ی تکرار[۱۱] و خوشه‌ی انحصاری[۱۲]

---

[۸] Collection

[۹] Replica set members

[۱۰] Auditing

[۱۱] Replica set

[۱۲] Sharded cluster

- تصدیق اصالت و مجوزها[۱۳]
- عملیات‌های CRUD (Create, Read, Update, Delete): به منظور تهیه‌ی ممیزی از این دسته از عملیات‌ها بایستی پارامتر auditAuthorizationSuccess با مقدار true تنظیم شود.

رویدادهای ممیزی می‌توانند در میز فرمان[۱۴]، syslog (این گزینه در ویندوز قابل تنظیم نیست)، فایل JSON یا فایل BSON نوشته شوند. به منظور فعال کردن ممیزی می‌توان از گزینه‌ی auditDestination در دستور mongod استفاده کرد. با استفاده از این گزینه، محل خروجی رویدادهای ممیزی تعیین می‌شود. به عنوان نمونه، برای تعیین فایل JSON برای نوشتن رویدادهای ممیزی از دستور زیر استفاده می‌شود:

```
C:\Program Files\MongoDB\Server\3.4\bin>mongod.exe --config "C:\Program Files\Mo
ngoDB\Server\3.4\mongod.cfg" --auditDestination file --auditFormat JSON --auditP
ath c:\data\log\auditlog.json –install
```

همچنین می‌توان این تنظیمات را در فایل پیکربندی mongod.cfg به صورت زیر وارد کرد:

```
auditLog:
   destination: file
   format: JSON
   path: c:\data\log\auditlog.json
setParameter: { auditAuthorizationSuccess: true }
```

حال می‌توان رویدادهای ممیزی موجود در فایل auditlog.json را مشاهده کرد (شکل ۷).

```
"isMaster" : 1, "client" : { "application" : { "name" : "MongoDB Shell" }, "driver" : { "name" : "MongoDB
{ "insert" : "test", "documents" : [ { "_id" : { "$oid" : "59f36dbd35e8e5abf2d3b987" }, "title" : "test",
{ "insert" : "test", "documents" : [ { "_id" : { "$oid" : "59f36dbe35e8e5abf2d3b988" }, "title" : "test",
{ "insert" : "test", "documents" : [ { "_id" : { "$oid" : "59f36dbe35e8e5abf2d3b989" }, "title" : "test",
```

شکل ۷ فایل ممیزی در پایگاه‌داده MongoDB

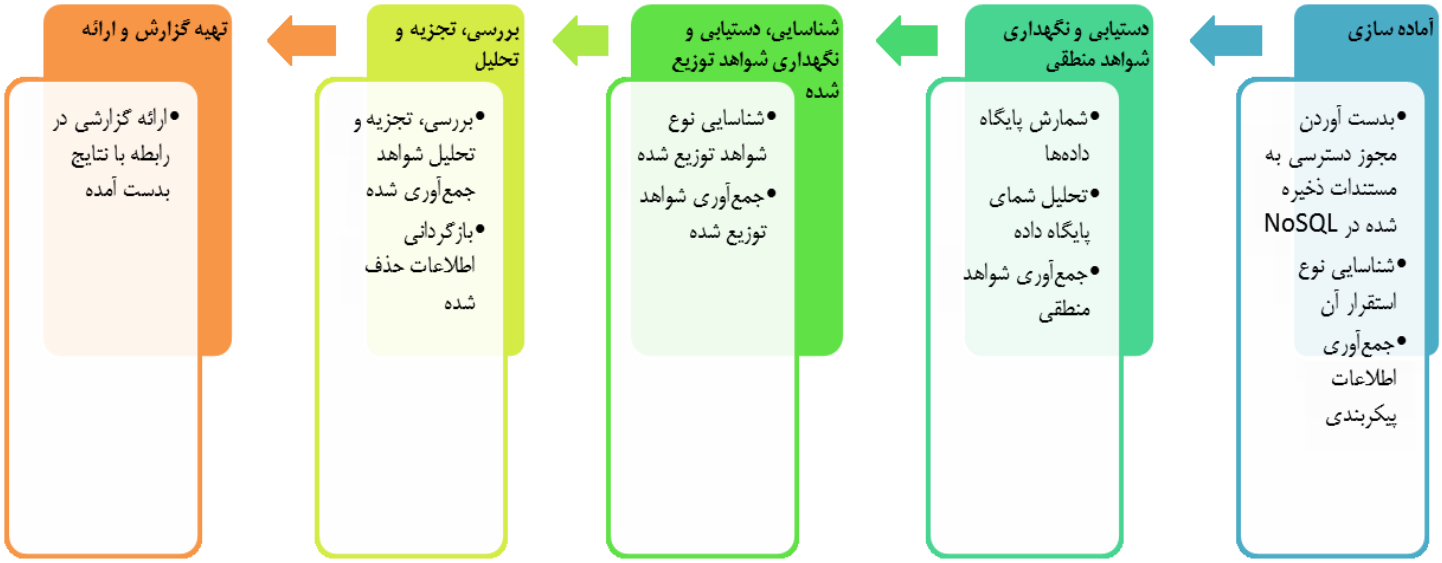
[۱۳] Authentication and authorization

[۱۴] Console

یک راه‌حل کامل برای ممیزی باید تمامی کارگزارهای mongod و پردازه‌های mongos را شامل شود. در خوشه‌های توزیع‌شده، در صورتی که ممیزی روی نمونه‌های mongos فعال باشد، باید ممیزی روی تمامی نمونه‌های mongod در خوشه یعنی shard–ها و کارگزارهای پیکربندی فعال باشد [۱].

۵ چارچوب جرم‌شناسی در پایگاه داده غیررابطه‌ای سند محور

در جرم‌شناسی پایگاه داده رابطه‌ای ساختار و ویژگی‌های خود پایگاه داده بسیار مهم است و بر مبنای آن جرم‌شناسی انجام می‌گیرد؛ بنابراین نمی‌توان آن‌ها را روی پایگاه داده غیررابطه‌ای که ساختار متفاوتی دارد اعمال کرد.



شکل ۸ چارچوب ارائه شده برای جرم‌شناسی پایگاه داده‌ی غیررابطه‌ای

در جرم‌شناسی پایگاه داده غیررابطه‌ای لازم است شمای غیرثابت تحلیل و مکان فیزیکی شواهد توزیع‌شده شناسایی شود. این شواهد از اهمیت بسیار زیادی برخوردار هستند زیرا اطلاعاتی در رابطه با گزارش تراکنش‌ها و فایل‌های داده‌ای در بین آن‌ها وجود دارد. بر این اساس، یک چارچوب پنج مرحله‌ای طراحی شده است که می‌توان در شکل ۸ چارچوب ارائه شده برای جرم‌شناسی پایگاه داده‌ی غیررابطه‌ای آن را مشاهده کرد.

- **مرحله اول: آماده‌سازی**

در این مرحله نوع پایگاه داده‌ی غیررابطه‌ای مشخص و مجوزها و دسترسی‌های متناسب برای عملیات جرم‌شناسی کسب می‌شود. علاوه بر این لازم است اطلاعاتی راجع به محیط توزیع و تکثیر پایگاه داده به دست آید. سپس باید اطلاعاتی در مورد محیط اجرایی سیستم هدف، ازجمله نوع استقرار پایگاه داده (مستقل[۱۵]، تکثیر[۱۶] و توزیع[۱۷])، درگاه و آی‌پی[۱۸] سرویس، نقش کارگزار و مکان فیزیکی جمع‌آوری شود.

- **مرحله دوم: دستیابی و نگهداری شواهد منطقی**

شواهد منطقی درواقع اطلاعاتی هستند که می‌توان بدون در نظر گرفتن نحوه‌ی استقرار پایگاه داده غیررابطه‌ای جمع‌آوری کرد. این قسمت مشابه فرآیند جمع‌آوری اطلاعات از پایگاه داده‌ی رابطه‌ای است. داده‌های ذخیره‌شده در پایگاه داده محاسبه‌شده و سپس شمای آن تحلیل می‌شود. ازآنجاکه پایگاه داده غیررابطه‌ای سند محور، برخلاف پایگاه داده رابطه‌ای شمای ثابتی ندارد، انجام این قسمت به توجه و دقت زیادی نیاز دارد. به دلیل اینکه جمع‌آوری کل داده‌ها مشکل است، باید تنها داده‌های مرتبط با موضوع انتخاب شوند. علاوه بر این داده‌های جمع‌آوری‌شده باید یکپارچگی خود را در فرآیند جرم‌شناسی نیز حفظ کنند.

- **مرحله سوم: شناسایی، دستیابی و نگهداری شواهد توزیع‌شده**

در این مرحله بر روی یکی از خصوصیات کلیدی پایگاه داده‌های غیررابطه‌ای، یعنی توزیع و تکثیر تمرکز می‌شود. در این قسمت باید شواهدی که لازم است برای مرحله تحلیل و بررسی در یک کارگزار فیزیکی ذخیره شوند، شناسایی و آن‌ها را از کارگزارهای توزیع‌شده، جمع‌آوری کرد. برخی شواهد توزیع‌شده شامل گزارش رویدادهای عمومی و تراکنش‌ها می‌شوند. علاوه بر این، برای دسترسی به داده‌های حذف‌شده، ابتدا باید کارگزار فیزیکی‌ای که داده در آن ذخیره‌شده است، شناسایی و سپس فایل‌های داده‌ای آن بازگردانی شوند.

---

[۱۵] Standalone

[۱۶] Replication

[۱۷] Distribution

[۱۸] IP

- **مرحله چهارم: بررسی، تجزیه‌وتحلیل**

در این مرحله تحلیلگر، شواهد جمع‌آوری‌شده را بر اساس روشی فنی، طی فرآیندی بررسی، آزمایش و تحلیل می‌کند. ازآنجاکه تاکنون ابزارهای بسیار کمی برای جرم‌شناسی و تحلیل داده‌های پایگاه داده غیررابطه‌ای سند محور ارائه شده است، بهترین و کارآمدترین راه این است که ابتدا در آزمایشگاه تحلیل جرم‌شناسی، یک پایگاه داده غیررابطه‌ای سند محور ایجاد و سپس داده‌ها در آن وارد گردد. علاوه بر این، تحلیل گزارش رویدادها و بازگردانی داده‌های حذف‌شده نیز باید در این مرحله صورت بگیرد.

- **مرحله پنجم: تهیه گزارش و ارائه**

در مرحله آخر باید تحلیلگر گزارشی برای ارائه‌ی نتایج تحلیل و آزمایش‌های خود، در قالبی مناسب ارائه دهد.

٦ اعمال چارچوب در پایگاه داده‌ی مانگو

در این قسمت، بر اساس چارچوب ارائه شده در بخش قبل، جرم‌شناسی بر روی پایگاه داده‌ی مانگو شرح داده خواهد شد.

٧ مرحله اول: آماده‌سازی

در این مرحله محقق اطلاعاتی در مورد محیط عملیاتی پایگاه داده‌ی مانگو جمع‌آوری می‌کند. اطلاعات محیطی پایگاه داده‌ی مانگو را می‌توان با اتصال از طریق ابزار mongo به پایگاه داده به دست آورد. نوع استقرار پایگاه داده‌ی مانگو را می‌توان با استفاده از ظاهر Shell Prompt شناسایی کرد:

- درصورتی‌که مستقل باشد، به شکل `>` خواهد بود.
- اگر مجموعه‌ی تکثیر باشد، به صورت `<Replica set name: member type>` خواهد بود.
- در خوشه‌ی توزیع‌شده نیز از `mongos>` استفاده می‌شود.

برای جمع‌آوری اطلاعات اولیه از پایگاه داده‌ی مانگو از دستور `db.serverStatus()` استفاده می‌شود. خروجی این دستور شامل اطلاعات پایه‌ی کارگزار، اطلاعات قفل، تعداد ارتباطات، نرخ استفاده از شبکه و حافظه است. از منظر جرم‌شناسی، اطلاعات پایه‌ی کارگزار، از اطلاعات مربوط به کارایی ارزشمندتر است.

ازجمله اطلاعاتی که می‌توان از این دستور به دست آورد، نام و IP میزبان کارگزار، نسخه پایگاه داده‌ی مانگو، نوع پردازه (mongos یا mongod)، شناسه‌ی پردازه، زمان بالا بودن و زمان محلی کارگزار است.

برای مشاهده جزئیات بیشتر تنظیمات پایگاه داده‌ی مانگو از دستور `db.serverCmdLineOpts()` استفاده می‌شود. نمونه‌ای از اجرای این دستور در شکل ۹ خروجی دستور db.serverCmdLineOpts قابل‌مشاهده است.

```
mongos> db.serverCmdLineOpts()
{
        "argv" : [
                "mongos",
                "--config",
                "/data/conf/mongos.conf"
        ],
        "parsed" : {
                "config" : "/data/conf/mongos.conf",
                "net" : {
                        "bindIp" : "192.168.229.141",
                        "port" : 10000
                },
                "processManagement" : {
                        "fork" : true
                },
                "sharding" : {
                        "chunkSize" : 1,
                        "configDB" : "192.168.229.141:20000,192.168.229.142:20000,192.168.229.143:20000"
                },
                "systemLog" : {
                        "destination" : "file",
                        "logAppend" : true,
                        "path" : "/data/log/mongos.log"
                }
        },
        "ok" : 1
}
```

شکل ۹ خروجی دستور db.serverCmdLineOpts

قسمت `argv`، آرگومان‌های استفاده‌شده در خط فرمان سیستم‌عامل را، از زمان آغاز به کار پردازه mongos یا mongod، نشان می‌دهد. در قسمت `parsed`، اطلاعات پیکربندی که از فایل پیکربندی یا گزینه‌های شروع که از خط فرمان دریافت می‌شود، قرار می‌گیرد. ازآنجایی‌که گزینه `logAppend` دارای مقدار `true` است، می‌توان فایل‌های رویداد را در `/data/log/mongos.log` مشاهده کرد.

اطلاعات مربوط به محیط تکثیر و توزیع با استفاده از دستورات `rs.status()` و `sh.status()` قابل کسب است. با استفاده از `rs.status()` نام میزبان‌ها و درگاه‌های اعضایی که در مجموعه تکثیر قرار دارند، در کنار وضعیت و نقششان نشان داده می‌شود. نام shard ها، آدرس IP ها و درگاه‌های هریک از اعضای استقرار توزیع‌شده نیز با استفاده از دستور `sh.status()` به دست می‌آید. شکل ۱۰ نمایی از اجرای دستور sh.status خروجی این دستور را نشان می‌دهد.

```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "version" : 4,
        "minCompatibleVersion" : 4,
        "currentVersion" : 5,
        "clusterId" : ObjectId("54be721519c327853cf576fd")
}
  shards:
        {  "_id" : "rs-a",  "host" : "rs-a/192.168.229.141:30000,192.168.229.142:30001" }
        {  "_id" : "rs-b",  "host" : "rs-b/192.168.229.142:40000,192.168.229.143:40001" }
        {  "_id" : "rs-c",  "host" : "rs-c/192.168.229.141:50001,192.168.229.143:50000" }
```

شکل ۱۰ نمایی از اجرای دستور sh.status

۱-۷ مرحله دوم: دستیابی و نگهداری شواهد منطقی

در این مرحله ابتدا باید به شمارش پایگاه داده‌ها و مجموعه‌ها پرداخت. برای این کار لازم است از دستور `show dbs` استفاده شود. نتیجه اجرای این دستور در شکل ۱۱ نمایی از اجرای دستور show dbs نشان داده شده است.

```
mongos> show dbs
admin          (empty)
config         0.047GB
drupal         0.359GB
ebookfiles2    55.835GB
test           (empty)
testDB         0.156GB
```

شکل ۱۱ نمایی از اجرای دستور show dbs

اطلاعات جزئی دیگر مانند سایز داده و تعداد مجموعه‌ها را نیز می‌توان با سوئیچ کردن در پایگاه داده با دستور `use<database name>` و اجرای دستور `db.stats()` به دست آورد. برای شمارش مجموعه‌ها باید پس از وارد شدن در پایگاه داده، دستور `show collections` اجرا شود. شکل ۱۲ نمایی از اجرای دستور show collections اجرای این دستور را نشان می‌دهد.

```
mongos> use drupal
switched to db drupal
mongos> show collections
fields_current.node
fields_revision.node
system.indexes
watchdog
```

شکل ۱۲ نمایی از اجرای دستور show collections

برای به دست آوردن تعداد کل مستندات در هر مجموعه می‌توان از دستور `db.<collection name>.count()` استفاده کرد. اطلاعات بیشتر در مورد مجموعه‌ها نیز، مانند تعداد قطعات، وضعیت shard مجموعه‌ها و سایز داده‌ها با استفاده از دستور `db.<collection name>.stats()` به دست می‌آید.

در قدم بعدی، لازم است شمای مجموعه‌ها تحلیل گردد. به دلیل وجود نداشتن شما در پایگاه داده‌ی مانگو، یافتن همه‌ی فیلدهای مجموعه، کار بسیار مشکلی است؛ اما بااین‌وجود می‌توان با ارسال درخواست روی مستندات موجود در یک مجموعه مدلی برای آن‌ها ایجاد کرد، چراکه بیشتر برنامه‌ها مدل داده‌ای سازگاری دارند. با استفاده از دستور `db.<collection name>.find().limit(<document counts to print>)` می‌توان مستندات داخل مجموعه را با فرمت JSON مشاهده کرد. اگر به انتهای درخواست عبارت `.pretty()` نیز اضافه شود، نتیجه به صورت مستند خوانای JSON خواهد بود. شکل ۱۳ نمایی از اجرای دستور بالا نتیجه اجرای این دستور را نشان می‌دهد.

```
mongos> db.fields_current.node.find().limit(10).pretty()
{
        "_id" : NumberLong(4706),
        "_type" : "node",
        "_bundle" : "article",
        "_revision_id" : NumberLong(4706),
        "nid" : NumberLong(4706),
        "vid" : NumberLong(4706),
        "type" : "article",
        "language" : "und",
        "title" : "Beginning Lua Programming",
        "uid" : NumberLong(1),
        "status" : NumberLong(1),
        "created" : NumberLong(1421740829),
        "changed" : NumberLong(1421740829),
        "comment" : NumberLong(2),
        "promote" : NumberLong(1),
        "sticky" : NumberLong(0),
        "tnid" : NumberLong(0),
        "translate" : NumberLong(0)
}
```

شکل ۱۳ نمایی از اجرای دستور بالا

علاوه بر این می‌توان از variety.js، تحلیل‌کننده متن‌باز شمای پایگاه داده‌ی مانگو استفاده کرد. درصورتی‌که تعداد مستندات یا فیلدهای تودرتو زیاد باشد، این ابزار مفید خواهد بود. نتیجه‌ی استفاده از این ابزار در شکل ١٤ نتیجه‌ی استفاده از variety.js نشان داده شده است.

```
H:\mongodb>mongo 192.168.229.141:10000/drupal --eval "var collection='fields_current.node'" variety.js
MongoDB shell version: 2.6.5
connecting to: 192.168.229.141:10000/drupal
Variety: A MongoDB Schema Analyzer
Version 1.2.4, released 05 December 2013
Using query of { }
Using limit of 3597
Using maxDepth of 99
creating results collection: fields_current.nodeKeys
removing leaf arrays in results collection, and getting percentages
{ "_id" : { "key" : "_id" }, "value" : { "type" : "Object" }, "totalOccurrences" : 3597, "percentConta
{ "_id" : { "key" : "_type" }, "value" : { "type" : "String" }, "totalOccurrences" : 3597, "percentCon
{ "_id" : { "key" : "nid" }, "value" : { "type" : "Object" }, "totalOccurrences" : 3597, "percentConta
{ "_id" : { "key" : "vid" }, "value" : { "type" : "Object" }, "totalOccurrences" : 3597, "percentConta
{ "_id" : { "key" : "type" }, "value" : { "type" : "String" }, "totalOccurrences" : 3597, "percentCont
{ "_id" : { "key" : "title" }, "value" : { "type" : "String" }, "totalOccurrences" : 3597, "percentCon
{ "_id" : { "key" : "uid" }, "value" : { "type" : "Object" }, "totalOccurrences" : 3597, "percentConta
{ "_id" : { "key" : "tnid" }, "value" : { "type" : "Object" }, "totalOccurrences" : 3597, "percentCont
{ "_id" : { "key" : "_id.floatApprox" }, "value" : { "type" : "Number" }, "totalOccurrences" : 3597, "
{ "_id" : { "key" : "_bundle" }, "value" : { "type" : "String" }, "totalOccurrences" : 3597, "percentC
{ "_id" : { "key" : "_revision_id" }, "value" : { "type" : "Object" }, "totalOccurrences" : 3597, "per
{ "_id" : { "key" : "_revision_id.floatApprox" }, "value" : { "type" : "Number" }, "totalOccurrences"
{ "_id" : { "key" : "nid.floatApprox" }, "value" : { "type" : "Number" }, "totalOccurrences" : 3597, "
{ "_id" : { "key" : "vid.floatApprox" }, "value" : { "type" : "Number" }, "totalOccurrences" : 3597, "
{ "_id" : { "key" : "language" }, "value" : { "type" : "String" }, "totalOccurrences" : 3597, "percent
{ "_id" : { "key" : "uid.floatApprox" }, "value" : { "type" : "Number" }, "totalOccurrences" : 3597, "
{ "_id" : { "key" : "status" }, "value" : { "type" : "Object" }, "totalOccurrences" : 3597, "percentCo
{ "_id" : { "key" : "status.floatApprox" }, "value" : { "type" : "Number" }, "totalOccurrences" : 3597
{ "_id" : { "key" : "created" }, "value" : { "type" : "Object" }, "totalOccurrences" : 3597, "percentC
{ "_id" : { "key" : "created.floatApprox" }, "value" : { "type" : "Number" }, "totalOccurrences" : 359
{ "_id" : { "key" : "changed" }, "value" : { "type" : "Object" }, "totalOccurrences" : 3597, "percentC
{ "_id" : { "key" : "changed.floatApprox" }, "value" : { "type" : "Number" }, "totalOccurrences" : 359
{ "_id" : { "key" : "comment" }, "value" : { "type" : "Object" }, "totalOccurrences" : 3597, "percentC
{ "_id" : { "key" : "comment.floatApprox" }, "value" : { "type" : "Number" }, "totalOccurrences" : 359
{ "_id" : { "key" : "promote" }, "value" : { "type" : "Object" }, "totalOccurrences" : 3597, "percentC
{ "_id" : { "key" : "promote.floatApprox" }, "value" : { "type" : "Number" }, "totalOccurrences" : 359
{ "_id" : { "key" : "sticky" }, "value" : { "type" : "Object" }, "totalOccurrences" : 3597, "percentCo
{ "_id" : { "key" : "sticky.floatApprox" }, "value" : { "type" : "Number" }, "totalOccurrences" : 3597
{ "_id" : { "key" : "tnid.floatApprox" }, "value" : { "type" : "Number" }, "totalOccurrences" : 3597,
{ "_id" : { "key" : "translate" }, "value" : { "type" : "Object" }, "totalOccurrences" : 3597, "percen
{ "_id" : { "key" : "translate.floatApprox" }, "value" : { "type" : "Number" }, "totalOccurrences" : 3
{ "_id" : { "key" : "field_files" }, "value" : { "type" : "null" }, "totalOccurrences" : 1, "percentCo
{ "_id" : { "key" : "field_file" }, "value" : { "type" : "null" }, "totalOccurrences" : 1, "percentCon
{ "_id" : { "key" : "field_nf" }, "value" : { "type" : "null" }, "totalOccurrences" : 1, "percentConta
```

شکل ١٤ نتیجه‌ی استفاده از variety.js

در قدم بعدی باید از پایگاه داده یا مجموعه، رونوشت[١٩] تهیه شود. برای این کار می‌توان از ابزارهای mongodump استفاده کرد. با استفاده از این ابزار می‌توان از تمامی داده‌ها، یک پایگاه داده‌ی مشخص و یا از یک مجموعه رونوشت تهیه کرد.

[١٩] Dump

در مرحله بعدی لازم است داده‌ها انتخاب گردند. به این منظور می‌توان از دستور `find()` برای جستجوی داده استفاده کرد. برای استفاده از این دستور باید از عبارت `db.<collection` `<name>.find({'<field>':'<value>'})` استفاده کرد. عبارت موردنظر باید در قسمت value وارد گردد. با استفاده از / می‌توان یک عبارت را در قسمت جستجو وارد کرد، مانند `db.fields_current.node.find({'title':/MongoDB/})`. از ابزار mongoexport نیز می‌توان برای دست‌یابی به اطلاعات خاصی در پایگاه داده، مجموعه یا نتیجه یک درخواست به صورت JSON یا CSV استفاده کرد.

در قدم بعدی اطلاعات GridFS باید به دست بیاید. محققان در این مرحله لازم است فهرست فایل و فراداده را در مجموعه fs.files جستجو و داده‌های GridFS را به صورت بهینه جمع‌آوری کنند. نمونه‌ای از جستجو در لیست فایل و فراداده توسط دستور find() در شکل ۱۵ نمونه‌ای در دستور ()find نشان داده شده است.

```
· Searching all file list : db.fs.files.find()
· Searching file list contained specific keywords : db.fs.files.find({'filename':/<keyword>/})
· Searching file list uploaded after specific time : db.fs.files.find({'uploadDate': {$gte: new Date
                ("<Year>-<Month>-<Day>T<Hour>:<Minutes> : <Seconds><time offset(ex : +09:00)>")}})
```

شکل ۱۵ نمونه‌ای در دستور ()find

فایل‌های ذخیره‌شده در GridFS را می‌توان با استفاده از ابزار mongofiles نیز استخراج کرد. نمونه‌ای از استفاده از این ابزار در شکل ۱۶ نمونه‌ای از استفاده از ابزار monfogiles قابل‌مشاهده است.

```
H:\mongodb>mongofiles -h 192.168.229.141:10000 -d ebookfiles2 search "MongoDB"
connected to: 192.168.229.141:10000
50 Tips and Tricks for MongoDB Developers.PDF   1290218
Ruby and MongoDB Web Development.PDF    5376731
Building Node Applications with MongoDB and Backbone.PDF        5556406
Scaling MongoDB.PDF     1745053
MongoDB Applied Design Patterns.PDF     9813879
The Definitive Guide to MongoDB.PDF     5431598
MongoDB and PHP.PDF     7678309
MongoDB and Python.PDF  4398756
MongoDB in Action.PDF   8461360
PHP and MongoDB Web Development.PDF     6486087
Pro Hibernate and MongoDB.PDF   10159340

H:\mongodb>mongofiles -h 192.168.229.141:10000 -d ebookfiles2 get "MongoDB and Python.PDF"
connected to: 192.168.229.141:10000
done write to: MongoDB and Python.PDF
```

شکل ۱۶ نمونه‌ای از استفاده از ابزار monfogiles

۷-۲ مرحله سوم: شناسایی، دستیابی و نگهداری شواهد توزیع‌شده

در این مرحله ابتدا باید کارگزار فیزیکی که شواهد در آن ذخیره شده‌اند شناسایی شود. اطلاعات مربوط به وضعیت shard مجموعه، به‌منظور پیدا کردن کارگزار فیزیکی که در آن شواهد ذخیره شده است، باید چک شود. همانند شکل ۱۷ وضعیت shard، دستور `sh.status()` می‌تواند برای بررسی جزئیات وضعیت shard ها مورداستفاده قرار گیرد.

```
databases:
      {  "_id" : "admin",  "partitioned" : false,  "primary" : "config" }
      {  "_id" : "drupal",  "partitioned" : true,  "primary" : "rs-a" }
            drupal.fields_current.node
                  shard key: { "_id" : 1 }
                  chunks:
                        rs-b    1
                        rs-a    1
                        rs-c    1
                  { "_id" : { "$minKey" : 1 } } -->> { "_id" : NumberLong(3638) } on : rs-b Timestamp(2, 0)
                  { "_id" : NumberLong(3638) } -->> { "_id" : NumberLong(4706) } on : rs-a Timestamp(3, 1)
                  { "_id" : NumberLong(4706) } -->> { "_id" : { "$maxKey" : 1 } } on : rs-c Timestamp(3, 0)
```

شکل ۱۷ وضعیت shard

در قدم بعدی لازم است رویدادها جمع‌آوری شوند. در مرحله‌ی آماده‌سازی مکان فایل‌های رویداد مشخص شد. فایل مربوط به رویدادهای پردازه mongos شامل اطلاعاتی از قبیل زمان آغاز و پایان پردازه، ارتباطات کاربران، تغییرات وضعیت shard، مانند انتقال، بخش‌بندی و قفل‌کردن تکه‌ها است. در فایل رویداد پردازه mongod نیز اطلاعات مربوط به زمان آغاز و پایان پردازه، ارتباطات کاربران، ایجاد و دورانداختن پایگاه داده و ایجاد و دورانداختن مجموعه ذخیره می‌شود. در شکل ۱۸ کلیدواژه‌های مربوط به تراکنش‌های پایگاه داده‌ی مانگو در فایل رویداد تراکنش‌های پایگاه داده‌ی مانگو که می‌توان با استفاده از جستجو در فایل رویداد به دست آورد، نشان داده شده است.

MongoDB transaction	Keyword	
	mongos log	mongod log
Process start	/Mongos version.*starting:/	MongoDB starting
Process stop	terminated	shutdown
Client connection	connection accepted from	
Database create	put [	allocating new datafile
Database drop	DROP DATABASE	dropDatabase
Collection create	N/A	build index on:
Collection drop	DROP:	CMD: drop
Document insert	N/A	N/A
Document remove	N/A	N/A

شکل ۱۸ کلیدواژه‌های مربوط به تراکنش‌های پایگاه داده‌ی مانگو در فایل رویداد

قدم بعدی به دست آوردن Oplog است. فایل oplog، فایل رویدادی است که تغییرات رخ‌داده در داده‌ها و پایگاه داده‌ها را به‌منظور همگام‌سازی میان کارگزارها در مدل استقرار مجموعه تکثیر فراهم می‌آورد. این فایل در مجموعه oplog.rs، در پایگاه داده محلی ذخیره می‌شود. سایز این فایل بسته به محیط اجرایی، متفاوت است. فایل oplog برای جرم‌شناسی بسیار حائز اهمیت است، چراکه تحلیل گران با استفاده از آن می‌توانند آخرین تغییرات پایگاه داده را دریابند. نمونه‌ای از این فایل در شکل ۱۹ نمونه‌ای از فایل oplog می‌توان مشاهده کرد.

```
{
  "ts" : Timestamp(1422422806, 1),
  "h" : NumberLong("-4768623843016918112"),
  "v" : 2,
  "op" : "i",
  "ns" : "test3.coltest3",
  "o" : {
          "_id" : ObjectId("54c7ca1577bc813a4271ad30"),
          "test" : 1
        }
 }

· ts : the timestamp this operation occurred and its duration
· h : the unique ID of this operation
· v : a special version number for the oplog format
· op : type of this operation
    - i : document insertion        - u : document modification        - d : document deletion
    - c : database or collection drop    - n : indicates no-operation
· ns : the database and collection affected by this operation
· o : actual data (contents) of this operation
```

شکل ۱۹ نمونه‌ای از فایل oplog

همچنین محققین می‌توانند از طریق ابزار mongodump و mongoexport به فایل oplog دست پیدا کنند. همان‌طور که در شکل ۲۰ دستیابی به oplog از طریق mongoexport نشان داده شده است، لیست تراکنش حذف را می‌توان با استفاده از mongoexport به دست آورد.

```
mongoexport -h 192.168.229.143:50000 -d local -c oplog.rs -f ts,h,op,ns,o -q {'op':'d'}
-o deletiontransaction.csv --csv
```

شکل ۲۰ دستیابی به oplog از طریق mongoexport

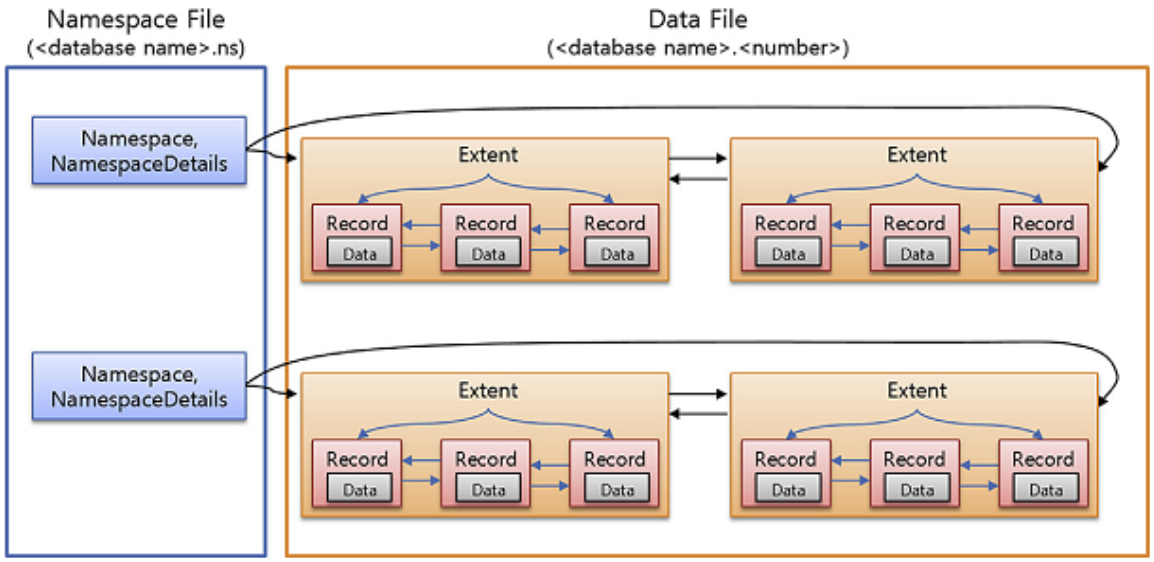
۳-۷ مرحله چهارم: بازگردانی اطلاعات حذف‌شده

به‌طورکلی دو نوع فایل داده‌ای در پایگاه داده‌ی مانگو وجود دارد. فایل‌هایی که فراداده در آن‌ها ذخیره می‌شود، Namespace files نام دارند. این فایل‌ها به شکل <database name>.ns نام‌گذاری می‌شوند و در دایرکتوری داده ذخیره می‌گردند؛ اما فایل‌هایی که داده‌های واقعی در آن‌ها ذخیره می‌شوند به صورت <DB name>.number نام‌گذاری می‌شوند. هرگاه سایز فایل داده از ۲ گیگابایت بیشتر شود، یک عدد به پسوند اضافه‌شده و فایل جدیدی ایجاد می‌شود. در شکل ۲۱ نحوه نام‌گذاری فایل‌های داده‌ای نمونه‌ای از نام‌گذاری این فایل‌ها وجود دارد.

```
4.0K 01-28 14:35 _tmp                2.0G 01-21 02:47 ebookfiles2.5
 64M 01-28 15:40 drupal.0            2.0G 01-21 03:10 ebookfiles2.6
 16M 01-28 15:40 drupal.ns           2.0G 01-27 15:44 ebookfiles2.7
 64M 01-27 15:44 ebookfiles2.0       2.0G 01-21 02:30 ebookfiles2.8
128M 01-21 01:37 ebookfiles2.1       2.0G 01-21 02:29 ebookfiles2.9
2.0G 01-21 03:10 ebookfiles2.10       16M 01-27 15:44 ebookfiles2.ns
2.0G 01-21 03:10 ebookfiles2.11      4.0K 01-28 14:36 journal
2.0G 01-21 03:10 ebookfiles2.12       64M 01-21 09:30 local.0
256M 01-21 02:51 ebookfiles2.2       2.0G 01-28 15:40 local.1
512M 01-21 01:51 ebookfiles2.3       2.0G 01-28 15:40 local.2
1.0G 01-21 03:06 ebookfiles2.4        16M 01-28 15:40 local.ns
```

شکل ۲۱ نحوه نام‌گذاری فایل‌های داده‌ای

به‌منظور بازگردانی مستندات حذف‌شده لازم است ساختار فایل داده‌ها مشخص شود. شکل ۲۲ ساختار داده‌ها و فایل داده‌ها ساختار کلی فایل داده‌های پایگاه داده‌ی مانگو را نشان می‌دهد.



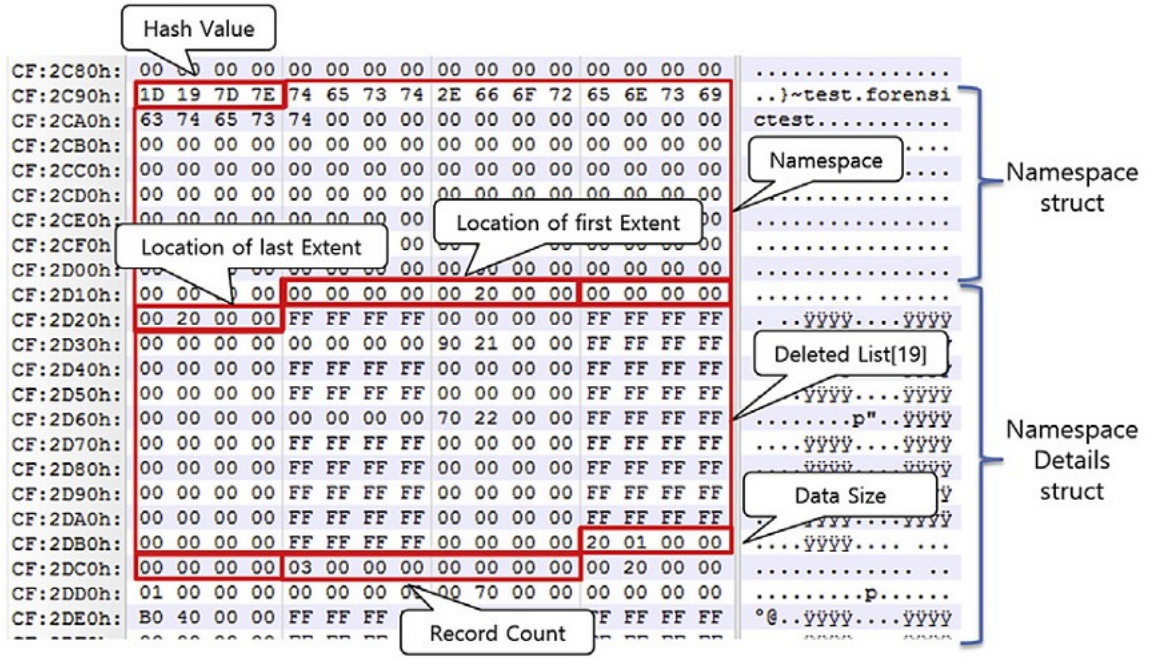
شکل ۲۲ ساختار داده‌ها و فایل داده‌ها

همان‌طور که مشاهده می‌شود، فایل داده‌ها شامل دو بخش Extent و record است. Extent مجموعه‌ای از بلاک‌های دنباله‌دار است که همه‌ی Extent های متعلق به یک Namespace، با استفاده از یک لینک لیست دوتایی به یکدیگر متصل هستند. هر Extent دارای تعدادی Record است که داده‌ها در آن به صورت مستندات BSON یا B-tree ذخیره شده‌اند. Record های مرتبط نیز با استفاده از لینک لیست دوتایی به هم متصل هستند.

جزئیات مربوط به Namespace و ساختار آن در فایل Namespace که برای هر مجموعه پایگاه داده ایجاد می‌شود، وجود دارد. همان‌طور که در شکل ۲۳ دیده می‌شود، ساختار Namespace شامل ٤ بایت مقدار درهم‌ریخته[۲۰] رشته‌ی Namespace و ۱۲۸ بایت بافر برای ذخیره رشته‌ی Namespace است. ١٦ بایت اول ساختار NamespaceDetails نیز شامل مکان اولین و آخرین Extent مربوط به یک Namespace می‌شود. مکان هر Extent با ۸ بایت نشان داده می‌شود که ٤ بایت اول شماره فایل داده و ٤ بایت بعدی آفست[۲۱] فایل داده است. ۱۵۲ بایت بعدی مکان رکوردهای حذف‌شده را به صورت آرایه نشان می‌دهد. ۸ بایت ادامه سایز داده و تعداد رکوردهای Namespace را مشخص می‌کند.

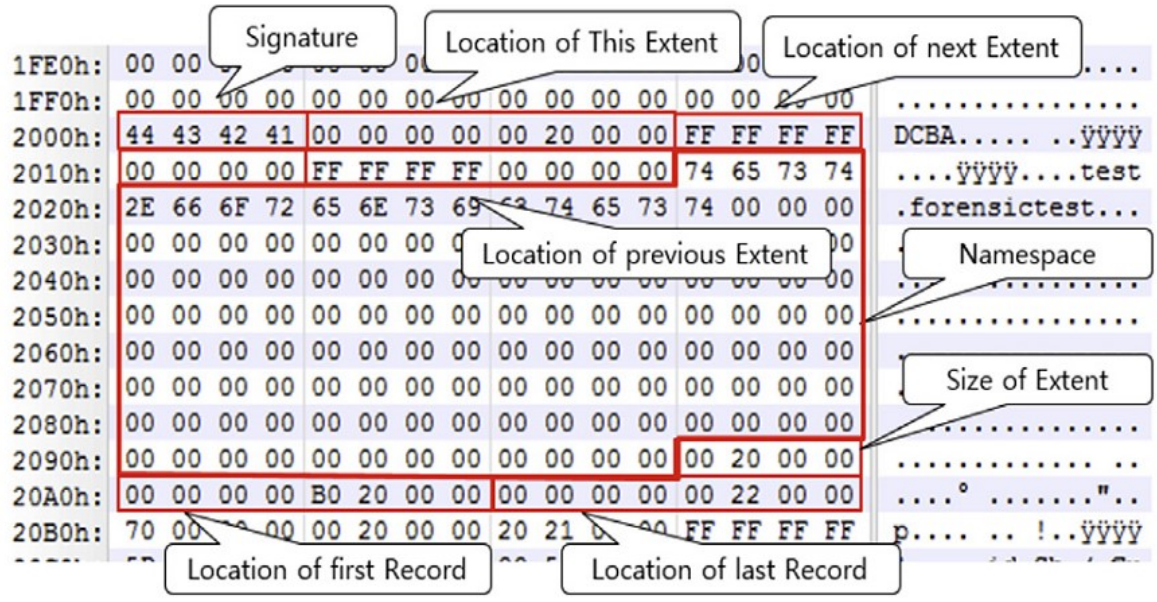
[۲۰] Hash

[۲۱] Offset



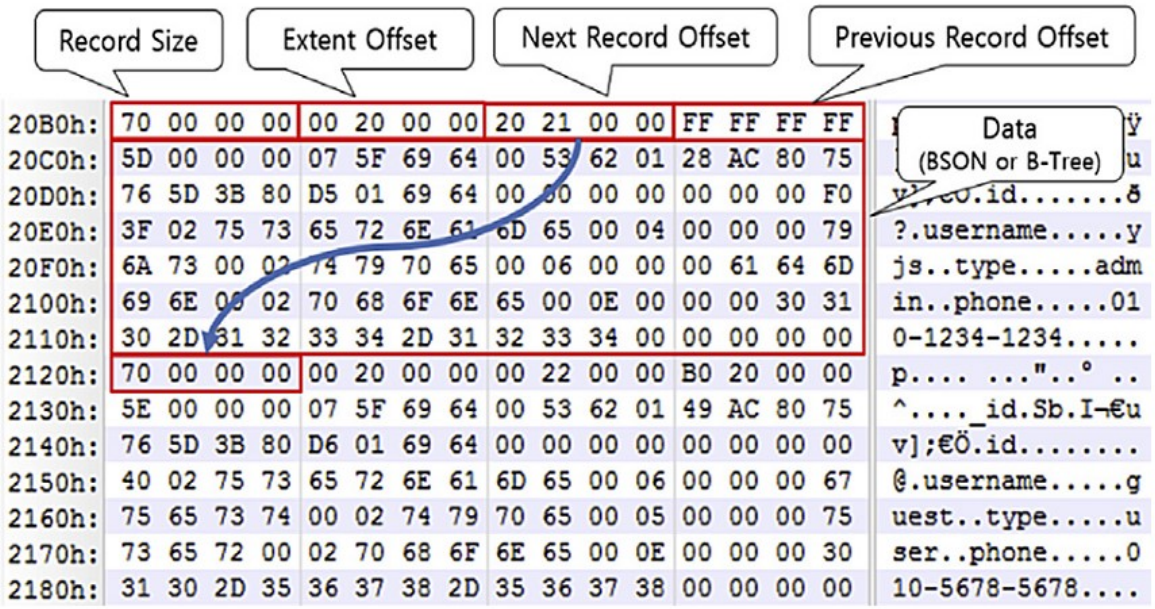
شکل ۲۳ قالب ساختار Namespace و NamespaceDetails

پس از یافتن مکان Extent اول از ساختار NamespaceDetails، همان‌طور که در شکل ۲٤ مشاهده می‌شود، ابتدا امضای ساختار Extent در آفست وجود دارد. ۸ بایت بعدی مکان خود Extent و ۱٦ بایت بعد، مکان Extent قبلی و بعدی را نشان می‌دهد. ۱۲۸ بایت بعدی بافر رشته Namespace و ٤ بایت بعد سایز Extent است. ۸ بایت موجود در ادامه، نشان‌دهنده مکان Record اول و آخر است.



شکل ۲٤ قالب ساختار Extent

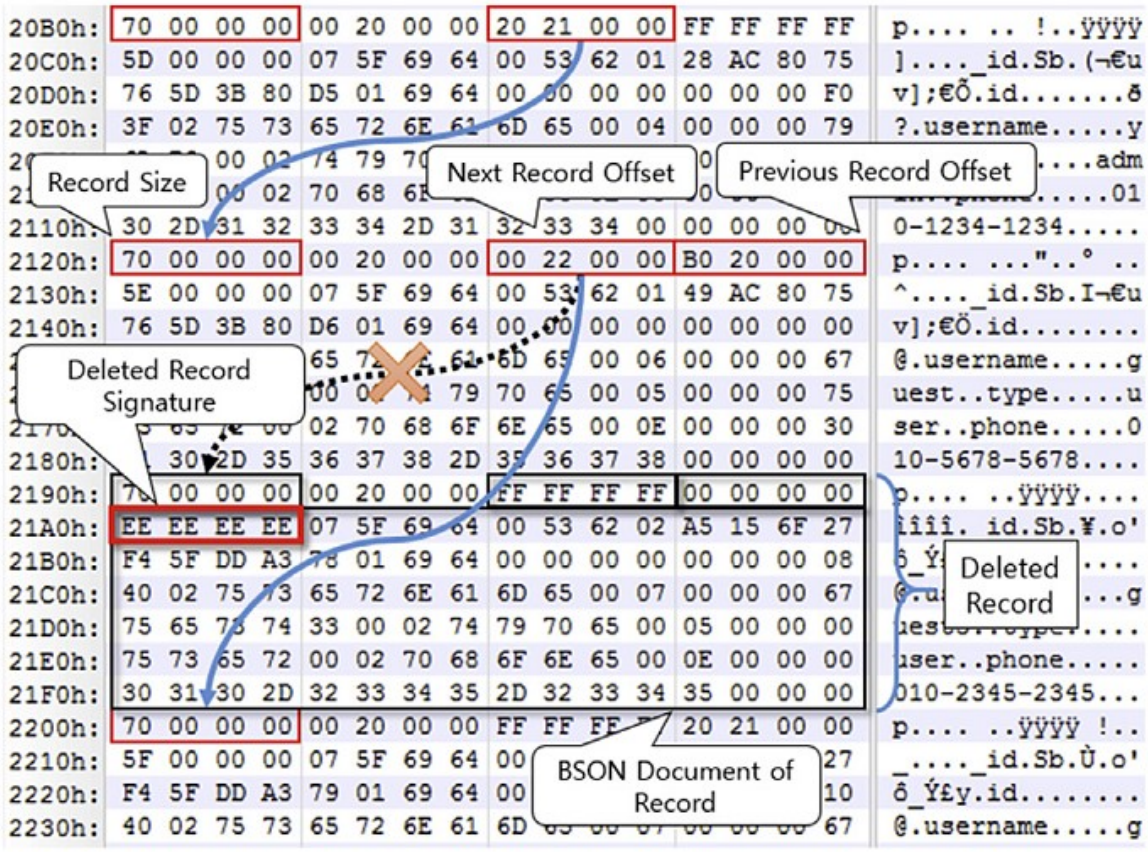
مکان اولین Record را می‌توان از ساختار Extent، به دست آورد. در شکل ۲۵ این ساختار نشان داده شده است. در ۱۶ بایت نخست به ترتیب سایز Record، آفست Extent که این Record متعلق به آن است، آفست Record بعدی و قبلی قرار گرفته است. پس‌ازآن نیز داده‌ها (BSON یا B-Tree) وجود دارد.



شکل ۲۵ قالب ساختار Record

درصورتی‌که مهاجم بخواهد عمداً داده‌ای را حذف یا پنهان کند، می‌توان آن را در فایل داده، با استفاده از ساختار Record بازیابی کرد. در پایگاه داده‌ی مانگو اطلاعات تا زمانی که مدیر دستور مدیریتی مانند repairDatabase یا collection compact را اجرا نکرده باشد، از پایگاه داده حذف نمی‌شود. هنگامی‌که Record توسط کاربری حذف شود، Record حذف‌شده علامت‌گذاری شده و آفست آن در لیست حذف‌شده‌ها، واقع در ساختار NamespaceDetails قرار می‌گیرد. همان‌طور که در شکل ۲۶ مشاهده می‌شود، زمانی که مستندی حذف می‌شود، ٤ بایت اول آن به مقدار ۰xEEEEEEEE و آفست Record قبلی و بعدی تغییر می‌یابد، درنتیجه Record حذف‌شده از ارتباطات بین Record ها حذف می‌شود.

برای بازگردانی مستند حذف‌شده، درصورتی‌که فایل Namespace موجود باشد، می‌توان با استفاده از مکان Record های حذف‌شده که در لیست قرار می‌گیرند، آن‌ها را بازیابی کرد. اگر فایل Namespace حذف‌شده باشد، می‌توان Record های حذف‌شده را از امضای ۰xEEEEEEEE که در ابتدای آن‌ها قرار می‌گیرد پیدا و بازیابی کرد.



شکل ۲۶ تغییرات انجام‌شده پس از حذف مستند

- **مرحله چهارم: بررسی، تجزیه‌وتحلیل**

در این قسمت تحلیلگر باید شواهد جمع‌آوری‌شده با استفاده از روش پیشنهادی در چهارچوب را بررسی و تحلیل کند. قبل از این کار، لازم است ابزارهای مناسب برای جرم‌شناسی پایگاه داده‌ی مانگو، ازجمله کارگزار آزمایشگاه جرم‌شناسی پایگاه داده‌ی مانگو نصب گردند.

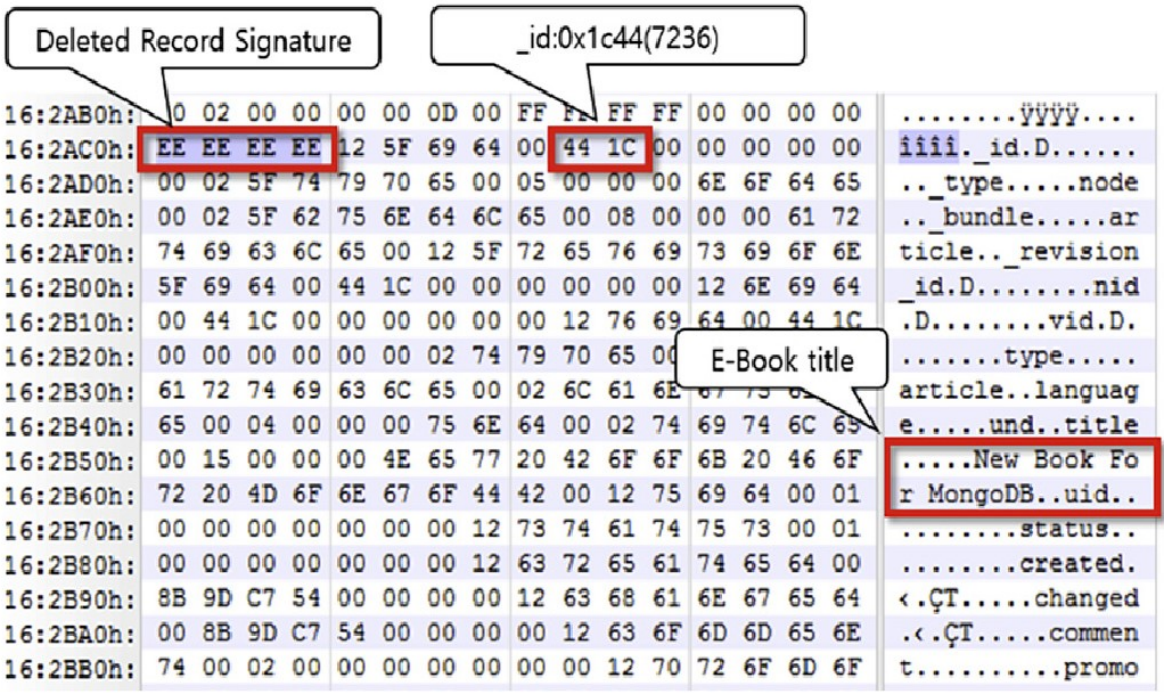
می‌توان فایل‌های رونوشت، فایل‌های CVS و JSON که در مرحله‌ی دوم جمع‌آوری شدند را در کارگزار پایگاه داده‌ی مانگو وارد کرد و با استفاده از ابزارهایی مانند mongorestore و mongoimport آن‌ها را بررسی و تحلیل کرد.

تراکنش‌های پایگاه داده‌ی مانگو با استفاده از oplog قابل‌پیگیری است و می‌توان از آن برای استخراج فهرست تراکنش‌های حذف بهره برد. در شکل ۲۷ نمونه‌ای تراکنش حذف در oplog نشان داده شده است.

```
{
        "ts" : Timestamp(1422434057, 1),
        "h" : NumberLong("-6257823605577556423"),
        "v" : 2,
        "op" : "d",
        "ns" : "drupal.fields_current.node",
        "b" : true,
        "o" : {
                "_id" : NumberLong(7236)
        }
}
```

شکل ۲۷ تراکنش حذف در فایل oplog

با استفاده از مقدار "_di" می‌توان فهمید که مستند در کدام shard قرار گرفته بوده است و چگونه می‌توان آن را بازگردانی کرد. به‌عنوان‌مثال، همان‌طور که در شکل ۲۸ نشان داده شده است، مستند حذف‌شده شامل اطلاعات مربوط به کتاب "New Book For MongoDB" است که مقدار "_id" آن ۷۲۳۶ بوده و مقدارش با مقدار استخراج‌شده از فایل oplog برابر است.



شکل ۲۸ مستند حذف‌شده در پایگاه داده‌ی مانگو

۸ جمع‌بندی

با استفاده روزافزون از پایگاه داده‌های غیر رابطه‌ای سند محور، نیاز به تجزیه‌وتحلیل و جمع‌آوری قانونی شواهد و جرم‌شناسی برای ارائه به دادگاه دیجیتال افزایش‌یافته است. ازآنجاکه این پایگاه داده‌ها شامل اطلاعات حجیم هستند و مدل داده‌ای آن‌ها انعطاف‌پذیر بوده و کارگزارهای آن‌ها توزیع‌شده است، لازم

است شرایط خاصی در جرم‌شناسی این پایگاه داده‌ها در نظر گرفته شود. در این گزارش یک چارچوب جرم‌شناسی پنج مرحله‌ای برای پایگاه داده‌های غیر رابطه‌ای سند محور معرفی شد. در این چارچوب سعی شد خصوصیات ویژه این پایگاه داده‌ها مدنظر قرار گرفته و بر اساس آن، چارچوبی مبتنی بر چارچوب‌های سنتی ارائه شود.

۹ منابع

[۱]. https://docs.mongodb.com/