پردازش تراکنش ها در MongoDB

زمستان ۹۶

چکیده

در حال حاضر، برای مقابله با مقادیر زیادی از داده های متنوع در پایگاه داده، پایگاه داده های NoSQL پیشنهاد وجود دارد و پیشنهاد میشود. برای استفاده عملی از پایگاه داده های NoSQL ، از آنجا که اکثر آنها از پردازش تراکنش تنها در داده های تک رابطه ای پشتیبانی می کنند، متصویر می شود که داده های چندگانه را نمی توان در یک واحد با حفظ خواص ACID به روز رسانی کرد.

یک روش برای پردازش داده های چندگانه استفاده از MongoDB است ، که یک نوع پایگاه داده NoSQL سند گرا محسوب می شود. به طور خاص، هر داده قبل و بعد از روز رسانی، وضعیت هر تراکنش مدیریت می شود. سپس در مورد قبل از Commit، داده های قبلی مورد پرسش قرار می گیرد؛ پس از Commit ، اطلاعات دومی مورد پرسش قرار می گیرد. با این روش ما نشان می دهیم که جمع داده ها را می توان به عنوان یک transaction با سطح isolation خاص به روز کرد.

مقدمه

با توسعه داده های حجیم (big Data)، اطلاعات مختلف دنیای واقعی در پایگاه های داده منتشر می شود. سیستم های مدیریت پایگاه داده رابطه ای (RDBMS) هدف خاص شرکت، و عمدتا با ارزش مقابله character data که از پیش تعیین شده برای حفظ سازگاری بالا وجود دارد. با این حال، امروزه پایگاه های داده ای هستند دسترسی به کاربران جهانی، و داده های آن نیز به محدوده های مختلف مانند اسناد و فیلم ها گسترش می یابد. بنابراین، لازم است که با ویژگی 3V سازگار شود، یعنی حجم (مقدار بسیار بالا)، سرعت و تنوع (تنوع گسترده) بنابراین، سیستم های مختلف مدیریت پایگاه داده به نام پایگاه داده NoSQL، که متفاوت از RDBMS معمولی است، پیشنهاد می شود و مورد استفاده قرار می گیرد.

در اینجا، همانطور که برای RDBMS، کنترل همزمانی اجرا می شود. حتی در مواردی که تعدادی از transaction در یک زمان به پایگاه داده دسترسی پیدا می کنند، آن ها به صورت یکسان اجرا می شوند. در نتیجه، خواص transaction ACID ، یعنی Atomicity، Consistency، Isolation و Durability، حفظ می شوند. از سوی دیگر، برای مقابله با ویژگی ۷۳، بسیاری از پایگاه داده های NoSQL تنها ویژگی BASE را حفظ می کنند، یعنی، اساسا در دسترسی راحت و در نهایت سازگار است. همانطور که برای این ویژگی، خواص ACID ذکر شده در بالا فقط در مورد به روز رسانی یک داده یکتا نگهداری می شود. بنابراین، در مورد به روز رسانی داده های چندگانه (رابطه ای)، داده ها به روز شده یکی پس از دیگری، و در نهایت تبدیل شدن و بروزرسانی نهایی انجام می شود. به این ترتیب، این ناهنجاری در میان این به روز رسانی وجود دارد، از جمله یک داده به روز شده است و یکی دیگر به روز نشده است. ویژگی BASE برای تراکنش که نیازی به کنترل همزمان سخت افزار ندارند موثر است. به طور مثال فروشگاه ای را در نظر بگیرید که از کالاها پر شده است. در اینجا، حتی در موردی که در آن تراکنش چندگانه خرید همان کالاها در همان زمان، متصویر شماره ی نیست که هر یک از این تراکنش ابتدا اجرا شود. از سوی دیگر، در مواردی که سهام(و یا انتخاب ها) محدود است، مانند رزرو هتل، این ایجاد متصویر شماره می کند. بنابراین، سطح کنترل همزمانی برای پایگاه داده بستگی به فرآیند کسب و کار دارد. برای این متصویر شماره ، همانند RDBMS، سطوح جداسازی چندگانه در استاندارد SQL تعریف شده است و سطح برای هر تراکنش به صورت جداگانه انتخاب شده است. به طور مشابه، همانطور که برای پایگاه داده های NoSQL نیز شناخته شده است، چنین پردازش تراکنش ای برای برخی از فرآیندهای کسب و کار مورد نیاز است. بنابراین، دو مرحله انجام متد برای MongoDB استفاده می شود ، که یک نوع پایگاه داده NoSQL است. این روش به روزرسانی داده های چندگانه را مدیریت می کند. و در صورت قطع، تراکنش جبران خسارت را برای بازیابی داده ها قبل از به روز رسانی انجام می دهد. با این حال، از آنجایی که بر اساس ویژگی BASE است، آنومالی فوق در میان به روز رسانی و بازیابی داده ها باقی می ماند. هدف ما از این مقاله ارائه یک روش پردازش تراکنش با داشتن سطح isolation مشخص شده برای MongoDB است.

به طور خاص، هر داده دارای قبل و بعد از بروز رسانی وضعیت تراکنش مدیریت می شود. سپس، در مورد قبل از رویداد داده های قبلی مورد پرسش قرار می گیرد؛ پس از رویداد، اطلاعات دومی مورد پرسش قرار می گیرد. علاوه بر این، ما این روش را به عنوان برنامه های تجربی با استفاده از جاوا اجرا کردیم و تایید کرد که دسترسی به پایگاه داده می تواند همزمان به عنوان یک تراکنش با سطح isolation مشخص شده انجام شود.

باقی مانده از این مقاله به شرح زیر است. بخش ۲ متصویر شماره پردازش تراکنش MongoDB را نشان می دهد و روش ما برای این متصویر شماره را در بخش ۳ ارائه می کنیم. بخش ٤ اجرای این روش و نتایج تجربی را نشان می دهد. بخش ٥ ملاحظات را نشان می دهد، و بخش ٦ نتیجه گیری نهایی را خواهیم داشت.

پردازش تراکنش در RDBMS

برای روشن شدن پردازش تراکنش در MongoDB، ما نمای کلی از پردازش تراکنش در RDBMS را نشان می دهیم. همانطور که برای پردازش تراکنش در RDBMS، خواص ACID به عنوان ٤ ویژگی زیر تعریف می شود: Atomicity به معنی به روز رسانی تراکنش به طور کامل یا نه در همه؛ سازگاری به این معنی است که انطباق پایگاه داده پس از به روز شدن حفظ می شود. جداسازی به این معنی است که هر تراکنش بدون تأثیر بر سایر تراکنش ها انجام شده همزمان انجام می شود؛ پایایی به این معنی است که نتایج به روز رسانی پایایی داده ها و برنامه را از بین نمی برد. علاوه بر این، برای انجام تراکنش ها چندگانه همزمان با حفظ خواص ACID، کنترل همزمان انجام می شود.

در نتیجه، اگر چه بسیاری از تراکنش ها در همان زمان انجام می شود، نتایج به روز رسانی آنها همانند یکسانی اجرا می شوند. این کنترل همگانی معمولا توسط روش قفل گذاری انجام می شود. یعنی، با انجام قفل قبل از دسترسی به داده ها، تداخل دسترسی به این داده ها از تراکنش ها دیگر محافظت می شود. همانطور که برای پردازش تراکنش، نشان داده شده است که پروتکل قفل دو مرحله ای (زیر، PL۲) مورد نیاز است، که در آن تمام قفل ها قبل از بازکردن اجرا می شوند. برای حفاظت از قطع آبشار، PL۲ دقیق مورد نیاز است، که تمام قفل تا زمانی که رویداد قطع شوند. برای باز کردن یک تراکنش دیگر صبر می کنند بنابراین تاخیر طولانی ممکن است در موارد زیر اتفاق بیفتد: درگیری با تراکنش طولانی مدت؛ تمرکز دسترسی به داده های خاص. سطوح isolation چندگانه تراکنش، همانطور که در جدول شماره ۱ نشان داده شده، تعریف شده است. در اینجا، تراکنش 2PL را برای به روز رسانی داده ها در هر سطح isolation احراز می کند. Read uncommitted اجازه می دهد تا تراکنش برای پرس و جو اطلاعات به روز رسانی قبل از آن commit شود، و قفل برای پرس و جو اجرا نمی شود. Read committed اجازه می دهد تا تراکنش برای پرس و جوی داده های به روز شده پس از آن commit شده باشد، و قفل مشترک تنها در طول پرس و جو اجرا می شود. Repeatable read اجازه می دهد تا تراکنش به طور مکرر پرس و جو یکسان داده ها را بدون تغییر توسط تراکنش ها دیگر، و تراکنش 2PL همچنین به درخواست داده ها انجام دهد. در حقیقت، همانطور که برای سطح isolation نیز وجود دارد serializable برای جلوگیری از خواندن داده ها، که توسط پرس و جو قبلی مورد پرسش نیست، وبرای پرس و جو بعدی مورد درخواست نیست. با این حال، از آنجا که داده های هدف برای قفل وجود ندارد، این سطح را نمی توان با پروتکل قفل ساده اجرا کرد. بنابراین، serializable در این مقاله حذف شده است.

```
{ "_id" : Id1, "name" : { "first" : "Tsukasa", "last" : "Kudo" }, "address" : "Hukuroi-shi"}
```

تصویر شماره ۱ – ساختمان داده در MongoDB

علاوه بر این، نشان داده شده است که در مواردی که سطح جداسازی تمام تراکنش ها حداقل خوانده نشده باشد، سطح جداسازی هر تراکنش از نوع خودش حفظ می شود. به این معنی که اگر هر تراکنش یکی از پروتکل های قفل شده در جدول شماره ۱ را نشان دهد، هر تراکنش می تواند با سطح انحلال مشخص شده انجام شود. بنابراین، حتی در MongoDB، می توان هر تراکنش را با سطح isolation مشخص شده انجام داد، در صورتی که تراکنش ها دیگر با سطح جداسازی Read uncommitted انجام شود.

No.	Isolation level	Exclusive lock	Shared lock
(1)	Read uncommitted	2PL	(none)
(2)	Read committed	2PL	During query
(3)	Repeatable read	2PL	2PL

جدول شماره ۱ – سطح Isolation و پروتکل قفل گذاری

داده MongoDB به عنوان سند JSON (نماد جاوا اسکریپت) که در تصویر شماره نشان داده شده است پیکربندی شده و سند شامل زمینه ها است. به عنوان مثال، در تصویر شماره ۱، {id: ld1} فیلدی است که شناسه «id» است؛ مقدار "ld1" (در زیر، شناسه شیء) است. در اینجا، فیلد ID Object مربوط به کلید اولیه پایگاه داده رابطه ای است. همچنین می تواند به عنوان نشان داده شده با نام "name"، که دارای فیلد "name" (نام) و "last" (نام خانوادگی) نشان داده شده است. از آنجایی که سند چنین ساختار دارد، ساختار پایگاه داده به عنوان RDBMS تعریف نشده است. بنابراین، زمینه هر سند می تواند در هر زمان اضافه یا حذف شود. به این معناست که ممکن است هر سند دارای زمینه های مختلفی باشد به غیر از id. در ضمن، مجموعه ای از اسناد مجموعه "مجموعه" را تشکیل می دهد، و هر کدام به رکوردها و جدول در یک پایگاه داده رابطه ای متصل می شود.

علاوه بر این، مانند SQL در RDBMS، روش های CRUD (ایجاد، خواندن، به روز رسانی، حذف): قرار دادن، پیدا کردن (مطابق با انتخاب)، به روز رسانی و حذف (مربوط به حذف). با این حال MongoDB قصد ندارند اسناد جمع را به عنوان یک تراکنش مشابه با RDBMS دستکاری کنند. ویژگی های تراکنش آنها فقط در سند تک ارائه می شود. به عبارت دیگر، در مورد روز رسانی اسناد جمع در یک مجموعه ، یک متصویر شماره وجود دارد که وضعیت میانی به روز رسانی می تواند توسط تراکنش ها دیگر مورد سوال قرار گیرد، یعنی خواص ACID نمی تواند حفظ شود. در ضمن، در مورد به روز شدن اسناد چندگانه با روش یکتا در MongoDB، گزینه "isolated $" را می توان تعیین کرد. با استفاده از این گزینه، می توان داده های هدف را از راه های دیگر به روز کرد. با این حال، متصویر شماره ی وجود دارد که این گزینه خاصیت Atomic را تضمین نمی کند. همچنین، آن را نمی توان به روز رسانی متتصویر شماره از انواع مختلفی از روش ها به تصویب رساند. علاوه بر این، پردازش تراکنش بر اساس همزمان سازی خوش بینانه با استفاده از نشانه زمانی برای H Base ، که نوعی از پایگاه داده NoSQL است و نسخه ای از هر داده را با Timestampمدیریت می کند، پیشنهاد می شود. برای اعمال این روش به MongoDB، لازم است که توابع مدیریت مشابه نسخه را به H Base اعمال کنیم.

برای این متصویر شماره ، دو مرحله انجام متد با حفظ Atomicity به روز می شود. در این روشبه عنوان مثال در صورت انتقال حساب بانکی از حساب A به حساب B شماره تراکنش به مجموعه مدیریت منتقل می شود. قبل از اینکه اسناد حساب A و B به روز شود، این شماره به دو اسناد ذخیره می شود تا اسناد به روز رسانی توسط این تراکنش مدیریت شود. در صورت اتمام موفقیت آمیز، شماره حذف می شود و پردازش پایان می یابد؛ در مورد شکست، تراکنش جبران شده برای بازیابی داده ها قبل از به روز رسانی انجام می شود.

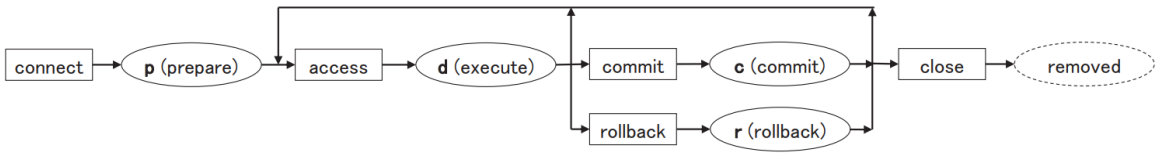
```
{ "_id" : Object ID of this document, "data0" : data before update,
  "ctl" : { "rn:" : number of query transactions, "r_id" : [ Object ID of query transaction 1, 2, ··· ],
  "w_id" : Object ID of update transaction }, "data1" : data after update }
```

Document and data collection

```
{ "_id" : Object ID of transaction, "tno" : transaction number, "st" : processing state,
  "level" : isolation level of this transaction }
```

Document of TP(Transaction Prossing Managment) collection

در واقع، مبلغ موقتا تغییر یافته توسط تراکنش ها دیگر مورد پرسش قرار می گیرد. علاوه بر این یک متصویر شماره دیگر نیز وجود دارد: در مورد داده های مرتبط با یکدیگر، هماهنگی داده ها ممکن است حفظ نشود حتی اگر به روز رسانی با موفقیت انجام شد. بدین ترتیب، از آنجا که پردازش تراکنش MongoDB نمیتواند اسناد چندگانه را به صورت یکپارچه با حفظ خواص ACID به روز رسانی کند، این متصویر شماره وجود دارد که نمیتوان آن را در سیستمهای کسب و کار مورد نیاز برای دستکاری چنین اطلاعاتی مورد استفاده قرار داد.



روش پردازش تراکنش در MongoDB

ساختار داده و بروز رسانی

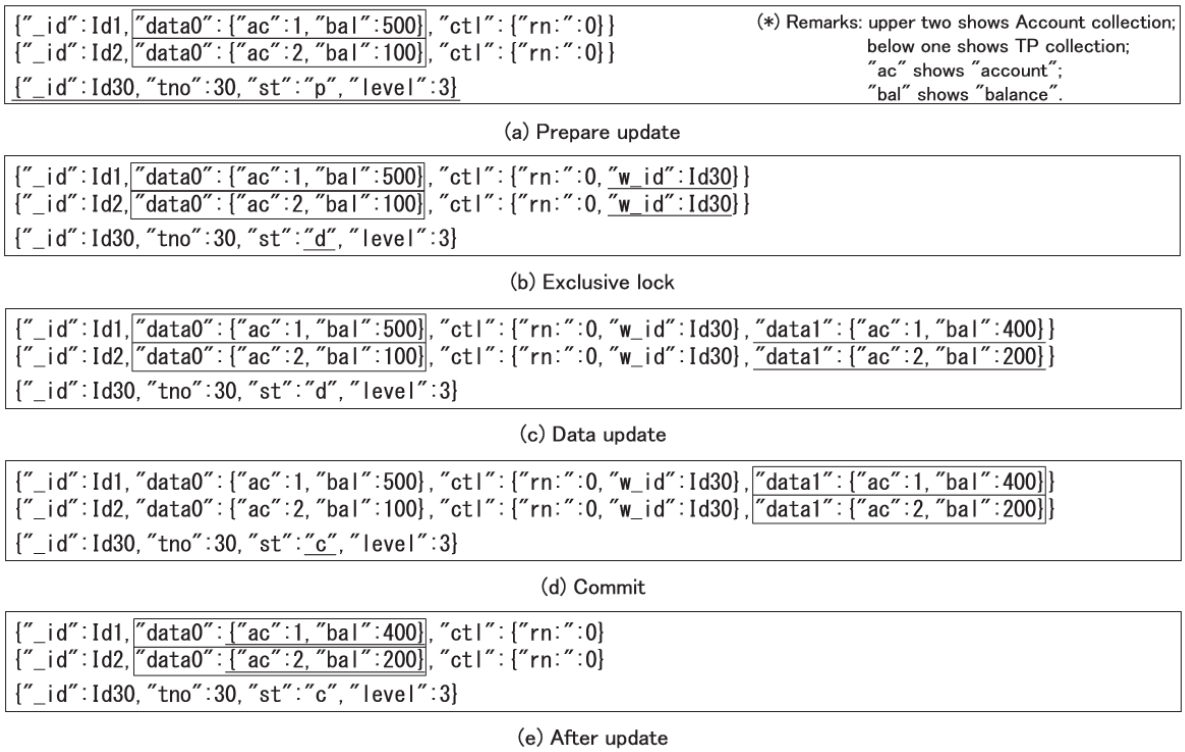
برای به روز شدن اسناد چندگانه یک تراکنش در MongoDB، تعهد یا عقبگرد باید در تمام مراحل مورد نظر همزمان انجام شود. بنابراین هر سند قبل و بعد از بروز رسانی برای پنهان کردن نتیجه به روز رسانی commit است. همچنین دارای فیلد اطلاعات قفل برای انجام کنترل منحصر به فرد است. علاوه بر این، TP (مدیریت پردازش مدیریت تراکنش ها) اضافه شده است برای مدیریت تراکنش ها در حال اجرا. تصویر شماره ۲ ساختارهای اسناد این دو مجموعه را نشان می دهد.

(a) یک مجموعه داده را نشان می دهد. انواع مختلفی از جمع آوری داده ها وجود دارد و آنها اطلاعات مربوط به کسب و کار را ذخیره می کنند. آنها شامل این موارد هستند: "data0" قبل از به روز رسانی داده ها را ذخیره می کند؛ "ctl" وضعیت قفل این سند را ذخیره می کند؛ "data1" پس از به روز رسانی داده ها را ذخیره می کند. همچنین فیلد "ctl" شامل موارد زیر است: "rn" تعداد تراکنش ها را قفل می کند که این داده ها را با قفل مشترک به اشتراک می گذارند، که تراکنش ها این سند را جستجو می کنند. "r id" آرایه شناسه شیء خود را در مجموعه TP ذخیره می کند؛ به طور مشابه، "w id" شناسه Object transaction را ذخیره می کند که این سند را منحصرا قفل می کند. در حقیقت، زمینه های زیر را در تصویر شماره ۲ اضافه می کند زمانی که داده ها به نظر می رسد و زمانی که داده ها از بین می روند حذف می شوند.

(b) مجموعه TP را نشان می دهد که اطلاعات مربوط به تراکنش را ذخیره می کند و سند آن وقتی که تراکنش مربوطه به MongoDB متصل می شود وارد می شود؛ هنگامی که تراکنش این اتصال را متوقف می کند حذف می شود. فیلد "tno" آن شماره شناسایی تراکنش را ذخیره می کند. "st" وضعیت تراکنش ذکر شده در

زیر را ذخیره می کند؛ "level" سطوح isolation تراکنش را که در جدول شماره ۱ نشان داده شده است را ذخیره می کند. تصویر شماره ۳ انتقال وضعیت تراکنش ها را نشان می دهد: p (آماده سازی) تراکنش متصل به پایگاه داده را نشان می دهد؛ d (اجرای) نشان می دهد که اولین دستکاری داده انجام شده است؛ c (مرتکب) نشان می دهد که شروع به انجام آن به دلیل اتمام موفقیت آمیز؛ r (رولپول) نشان می دهد که به دلیل شکست به عقب بازگشته است.

تصویر شماره ٤ نمونه ای از دستکاری داده ها به دلیل این روش را نشان می دهد که در آن انتقال حساب بانکی از یک حساب به حساب دیگر به عنوان یک تراکنش انجام می شود. از (a) تا تصویر شماره ٤، دو داده فوق دو اسناد جمع آوری حساب را نشان می دهند که نوعی جمع آوری داده ها از (a) در تصویر شماره ۲. در هر دو "data0" و "data1" فیلدها، "ac" فیلد شماره حساب است؛ "باله" تعادل آن است. سند بقیه مجموعه TP را نشان می دهد. در اینجا، خط زیر نشان می دهد که زمینه های تغییر یافته، فیلدهای محصور نشان می دهد که داده های مورد پرس و جو هستند. علاوه بر این، اگرچه تصویر شماره ٤ نشان داده شده است



تصویر شماره ٤ – بروزرسانی داده ها

مورد ساده برای به روز رسانی تنها دو اسناد به منظور وضوح، ممکن است هر شماری از اسناد مجموعه حساب به عنوان یک تراکنش به همان شیوه به روز شود.

قبل از به روز رسانی وضعیت را نشان می دهد و اطلاعات تراکنش به روز شده در مجموعه TP به عنوان سند دارای شناسه اشیاء "Id30" و حالت "p" ذخیره شده است. (b) دولت را پس از تکمیل قفل منحصر به فرد نشان می دهد. در اینجا، قفل منحصر به فرد به هر سند به صورت پیوسته اجرا می شود، بنابراین فیلد حالت از سند جمع آوری TP زمانی که فیش قفل اجرا می شود، انتقال "d" می شود. (d) نشان داده شده است که پس از ارسال اطلاعات، که به صورت پیوسته انجام می شود. (a) بعد از به روز رسانی نشان می دهد که در فرآیند های زیر در هر سند به طور همزمان انجام می شود: داده های "data1" به "data0" تغییر می کند و "data1"

حذف می شود؛ قفل انحصاری تحت مدیریت "w id" قفل شده است. این فرایند نیز بر روی هر سند پیوسته انجام می شود. در ضمن، در صورت بازگشت، فرآیند زیر انجام می شود: در (d)، حالت انتقال حالت به "r"؛ در (e)، فیلد data1 حذف می شود و فیلد data0 باقی می ماند. قفل انحصاری قفل شده توسط این اسمارت فون هاست. علاوه بر این، درون بسته بندی، این جزء بدون data0 در صورت حذف، "data1" در (c) و (d) وجود ندارد، سند مربوطه در (e) حذف می شود.

در این مورد، این داده ها از طریق تطبیق مقادیر در جدول شماره 1 انجام می شود. ابتدا، Read committed نتیجه پرس و جو از دو اسناد در مجموعه حساب باید به یکی از موارد زیر باشد: هر دو بروز نمی شوند؛ هر دو به روز شده اند. بنابراین، پرس و جو در روش زیر انجام می شود. از آنجا که تراکنش commit نیست، فیلد "data0" بین (a) و (c) مورد سوال قرار می گیرد. اگر چه اسناد به طور پیوسته در جریان این فرآیند به روز می شوند، "data0" به روز نشده است. بنابراین، آنها نتایج به دست می آورند. سپس در (d)، از زمان انجام تعهدات، "data1" کنار رفته است. همانطور که قبلا ذکر شد، "data1" در هر کدام به صورت "data1" به "data0" داده شده است، نتیجه به روز می شود به شرح زیر است: هر دو "data0" و "data1" به طور همزمان مورد پرسش قرار می گیرند؛ و در مواردی که "data1" وجود دارد، سوال شده است؛ در غیر این صورت، data0 مورد سوال است.

در این مورد، این داده ها از طریق تطبیق مقادیر در جدول شماره ۱ انجام می شود. ابتدا، Read committed نتیجه پرس و جو از دو اسناد در مجموعه حساب باید به یکی از موارد زیر باشد: هر دو به روز نمی شوند؛ هر دو به روز شده اند. بنابراین، پرس و جو در روش زیر انجام می شود. از آنجا که تراکنش commit نیست، فیلد "data0" بین (a) و (c) مورد سوال قرار می گیرد. اگر چه اسناد به طور پیوسته در جریان این فرآیند به روز می شوند، "data0" به روز نشده است. بنابراین، آنها نتایج به دست می آورند. بعد، در (d)، از زمان انجام تعهدات، "data1" فریب خورده است. همانطور که قبلا ذکر شد، "data1" در هر کدام به صورت "data1" به "data0" داده شده است، نتیجه به روز می شود به شرح زیر است: هر دو "data0" و "data1" به طور همزمان مورد پرسش قرار می گیرند؛ و در مواردی که "data1" وجود دارد، سوال شده است؛ در غیر این صورت، data0 مورد سوال است.

دوم، در صورت عدم تعهد انجام تعهدات به موقع انجام نشده، commit شده است. بنابراین به طور مرتب به وسیله دستورالعمل ها inRead (d) غیرقابل قبول است. به طور تصادفی مجموعه ای از تصمیم ها به طور یکنواخت به پایان می رسد، که می تواند پس از آن به نتیجه برسد: تعدادی داده قبل از به روز رسانی؛ اطلاعات دیگر پس از به روز رسانی است. در نهایت، در مورد Repeatable read فرآیند پرس و جو همان چیزی است که خوانده شده و انجام شده است، اما در پروتکل قفل کردن متفاوت است، همانطور که در بخش بعدی نشان داده شده است.

Lock to request	Lock allowed before		
	Shared lock	Exclusive lock	
	–	c (*1)	d (*2)
Shared lock	○	○	×
Exclusive lock	×	×	×

(*1) Transaction state is "commit" or "rollback"
(*2) Transaction state is "execute"

تصویر شماره ۵ – ماتریس سازگاری خواندن و کامیت کردن

۳,۲. اجرای کنتر بوسیله متد قفل گذاری

همانطور که در ستون "Exclusive lock" در جدول شماره ۱ نشان داده شده است، قفل بر اساس PL۲ به عنوان دستکاری به روز رسانی در تمام سطوح isolation اجرا می شود. در ضمن، در این روش، برای جلوگیری از رکود آبشاری، PL۲ دقیق به عنوان پروتکل جایگزین PL۲ انجام می شود. بدین معنی که بازکردن انجام نمی شود تا زمانی که تراکنش انجام نشده باشد. بنابراین، در صورت عدم وجود قرارداد انتظار می رود تا انجام تراکنش دیگری متوقف شود.

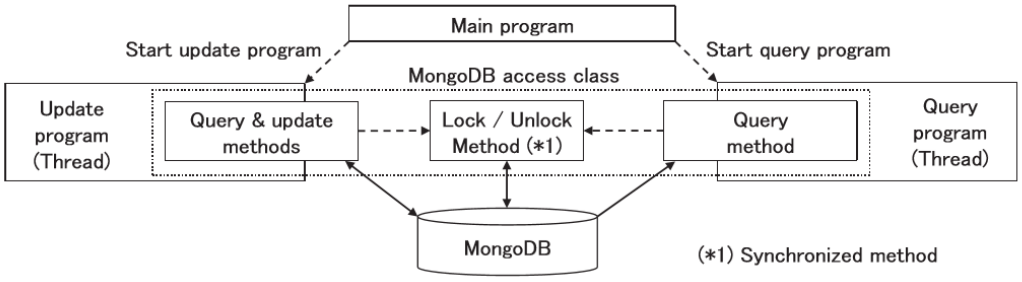
به طور مشابه، همانطور که در ستون "Shared Shuttle" نشان داده شده است، control Exclusive به سطح isolation مربوط به دستکاری پرس و جو بستگی دارد. اول، در مورد Read uncommitted، داده های هدف برای دستکاری پرس و جو قفل نیست. بنابراین، در صورتی که تراکنش تنها انجام دستکاری پرس و جو انجام می شود، می توان بدون وقفه به دلیل Conflict انجام شود. در ضمن، در مواردی که تراکنشهای Read uncommitted به روزرسانی شده و هر یک از سند را به یکی تحمیل می کند، مشابه با پردازش تراکنش MongoDB است،

انجام Transaction خواندن commit است که فقط در زمان درخواست انجام می شود. فیلد "rn" به عنوان فیلد "rn" تعریف شده در تصویر شماره ۲ نشان داده شده است. همانطور که در تصویر شماره ٤ نشان داده شده است، در این روش، تکمیل بروز رسانی بر اساس هر دو بستگی دارد: "w id" فیلد و state ("st" field) در مجموعه TP. بنابراین، در صورتی که تراکنش فوق (متضرر شده توسط "st") انجام شده باشد، قفل اشتراک مجاز است حتی اگر داده ها با قفل منحصر به فرد (نمایش داده شده توسط "w id") توسط این تراکنش قفل شده باشند. بنابراین، ماتریس سازگاری قفل مانند تصویر شماره ٥ نشان داده شده است. در اینجا، قفل منحصر به فرد "مجاز قبل" به دو مرحله تقسیم می شود: در صورتی که وضعیت تراکنش آن تعهد یا عقب نشینی باشد، قفل مشترک درخواست شده مجاز است؛ در غیر این صورت، رد می شود به طور مشابه، تراکنش تکرارپذیر نیز خواندن قفل مشترک را برای پرس و جو انجام می دهد، هرچند که تا زمانی که مرتکب یا رد شدن برگزار شود، برگزار می شود.

به طور مشابه، همانطور که برای قفل منحصر به فرد، همچنین ممکن است قفل انحصاری درخواست شده را در صورتی که تراکنش commit است، مجاز بدانیم. با این حال، در این مورد، از زمانی که بروزرسانی بعدی ممکن است قبل از اتمام بروزرسانی فعلی که در تصویر شماره ٤ (d) نشان داده شده، آغاز شود، لازم است که "data1" به عنوان آرایه باشد. به همین دلیل، در این مقاله، قفل انحصاری نباید سازگار باشد. ٥. در اینجا، همه ی آن ها برای قفل کردن قفل ها از طریق تراکنش های همزمان و همچنین RDBMS، قفل شده اند. به عبارت دیگر، دستکاری در "ctl" فیلد هر سند باید به ترتیب انجام شود شامل پرس و جو به مجموعه TP.

٤. ارزیابی اجرا

تصویر شماره ٦ ساختار برنامه تجربی را نشان می دهد. ما برای اجرای برنامه از جاوا (نسخه ١٧ ١,٦,٠) استفاده کردیم؛ MongoDB جاوا (نسخه ٢,١٣,٠) برای دسترسی به MongoDB (نسخه ٢,٦,٧). علاوه بر این، ما کلاس Thread از جاوا را برای اجرای همزمان اجرای تراکنش ها استفاده کردیم؛ و ما روش همگام سازی را برای پیاده سازی ترتیب انجام تراکنش ها ، که توسط دستکاری ها برای قفل و باز کردن استفاده می شود، استفاده کردیم. در ابتدای آزمایش، برنامه اصلی داده های تجربی را به مجموعه های MongoDB اختصاص داد. بعد، برنامه به روز رسانی و برنامه پرس و جو را با استفاده از موضوع شروع می کند. تا به طور همزمان به MongoDB دسترسی پیدا کند. ما کلاس دسترسی MongoDB را اجرا کردیم و دسترسی به MongoDB توسط روش های به روز رسانی و پرس و جو از این کلاس انجام می شود که در صورت لزوم با عملکرد قفل همراه است. برنامه بروزرسانی توازن سند هدف از مجموعه حساب را نمایش می دهد، سپس توازن سند را به روز می کند. بنابراین، ما برنامه Quantum را اجرا کردیم که با استفاده از آن، "program" را با عبارتی که با آن "select" می شود، اجرا می کنیم. برای پردازش در تصویر شماره ٦ ما روشهای زیر را در کلاس MongoDB Access آماده کردیم. در اینجا، روش های اساسی زیر حذف می شوند: تنظیم اتصال، بستن و سطح isolation.



تصویر شماره ٦ – اجرای برنامه ترکیبی

int read(int account)

این روش پرس و جو برای برنامه query string است، و توازن حساب مشخص را باز می گرداند. اول، اگر دولتی که از پروتکل جمع آوری اطلاعات (d) استفاده می کند، آن را انتقال می دهد "d". دوم، آن را به عنوان یک متخصص در این زمینه متمرکز کرده و آن را طبق جدول شماره ١ مشخص کرده است. موفقیت یا شکست قفل به تصویر شماره ٥ بستگی دارد. در صورت شکست، دوباره تلاش می کند؛ در مورد موفقیت، تعادل را نمایش می دهد. در اینجا، اگر سطح جداسازی Read uncommitted باشد، سند هدف را قفل نمی کند.

int readUp(int account)

این روش پرس و جو برای برنامه به روز رسانی است و تعادل را به همان شیوه ای که خوانده اید را باز می گرداند: ابتدا آن را انتقال می دهد؛ دوم، آن را به صورت جداگانه و با استفاده از آن به صورت جداگانه انجام می دهد. سپس، تعادل را نمایش می دهد. قفل تا زمانی که روش commit سازی یا بازپس گیری انجام نشود قفل نشده است.

WriteResult update(int account, int balance)

این تنظیم تعادل مشخصی را به حساب خاص اختصاص داده است. این سند هدف را همانند روش readUp قفل می کند، سپس سند را به روز می کند. در ضمن، WriteResult از مقدار بازگشتی یک کلاس از MongoDB راننده جاوا برای بیان نتایج به روز شده است.

void commit()

این تراکنش را تحمیل می کند. این حالت وضعیت مجموعه TP را به commit (c) تبدیل می کند و سپس اسناد هدف را با حذف اطلاعات قفل در فیلد ctl باز می کند. اسناد پس از دیگری قفل شده اند. در این زمان، اطلاعات ایمیلی با قفل قفل شده است، داده های "data1" به "data0" و "data1" حذف شده است، همانطور که در تصویر شماره ٤ نشان داده شده است.

void rollback()

این بازپرداخت تراکنش را انجام می دهد. روش آن برای جمع آوری TP و باز کردن همان روش commit است. اگر سند هدف با قفل منحصر به فرد قفل شود، فیلد "data1" حذف می شود.

٤,٢ آزمایشات پردازش تراکنش

ما تا اینجا ویژگی پردازش تراکنش با روش ارائه داده شده متوجه شدیم، آزمایشات زیر را انجام دادیم. این آزمایش ها بر روی ۱۰۰ حساب انجام شد. در مورد هر یک از سطوح برنامه پرس و جو، دو نمونه از آزمایشات انجام شد: تکمیل موفقیت آمیز و شکست. در ضمن، برنامه به روز رسانی با توجه به سخت افزار LP۲ ، سند هدف را با قفل منحصر به فرد قفل می کند.

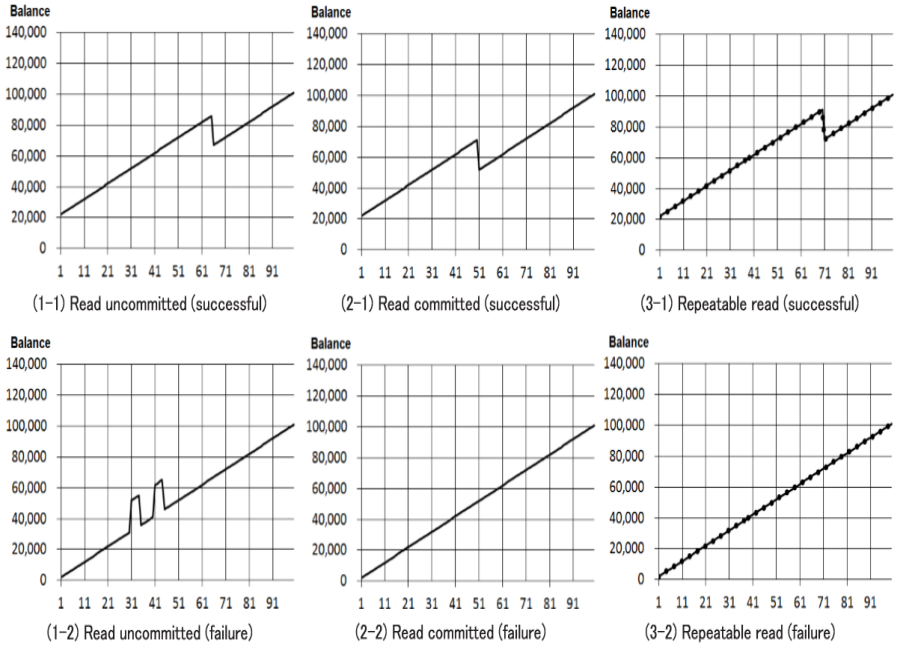
(۱) خواندن غیرقانونی: آن را نمایش داده ها یک به یک بدون قفل است.

(۲) خواندن commit: نمایش داده ها یک به یک با قفل مشترک؛ آن را در هر تکمیل پرس و جو باز کند.

(۳) Repeatable read: همچنین نمایش داده می شود یک به یک با قفل مشترک، و هر ۱۰ اسناد برای باز کردن انجام commit است.

Table 2. Delay time of executing program (millisecond).

Isolation level	Update transaction After query	After update	After commit	Query transaction Start delay	First read	After read	After commit
Read Uncommitted	20	20	0	1,500	–	20	–
Read Committed	10	0	150	500	–	5	–
Repeatable Read	10	0	100	500	25	–	50



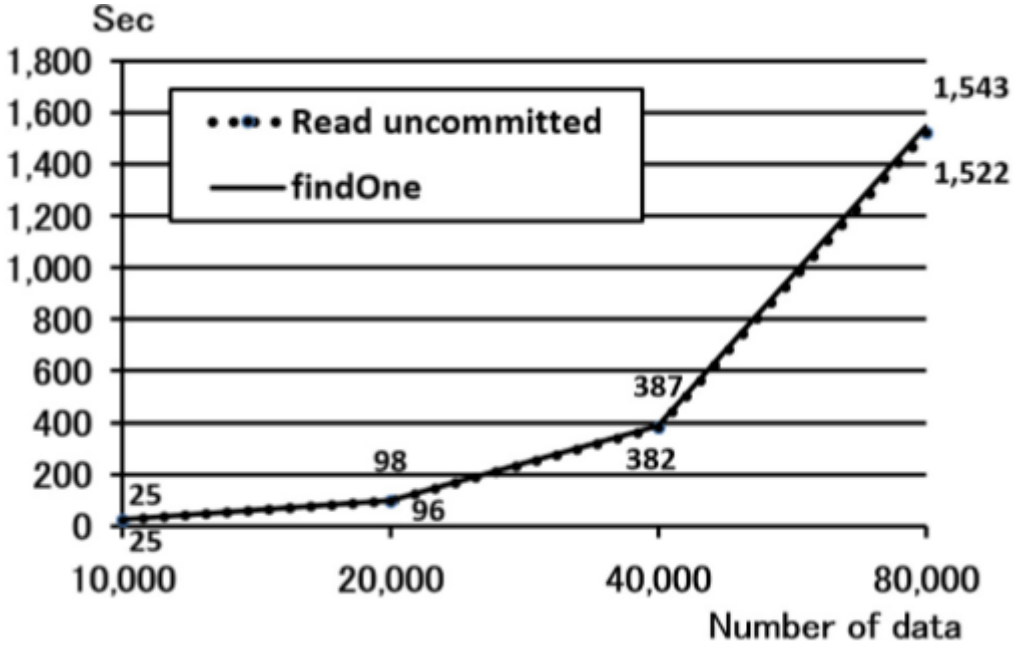
تصویر شماره ۷ – نتیجه آزمایش سطح isolation

برای جلوگیری از خرابی، هر دو برنامه به روز رسانی و پرس و جو با دسترسی به شماره حساب، به طور متوالی به مجموعه حساب دسترسی پیدا می کنند. برنامه به روز رسانی آغاز شده است و قفل های آن هر ۱۰ نسخه به روز رسانی می شوند. پس از آن، برنامه پرس و جو از پس از برنامه ریزی آغاز می

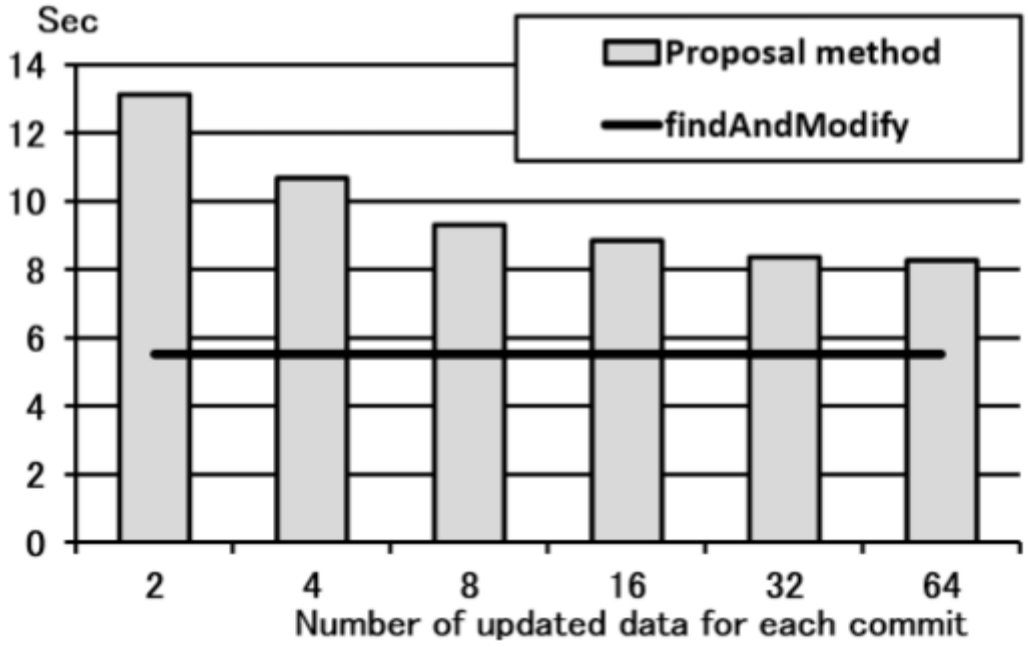
شود. بنابراین، ما هر دو برنامه را برای تأخیر در جدول ۲ نشان دادیم. در اینجا، برنامه پرس و جو از خواندن غیرقانونی، نیازی به انجام این کار نیست. بنابراین، برنامه زمانبندی روزانه "complete" را تضمین می کند. برای اندازه گیری سطح اطمینان، باید آنرا به روز رسانی کنید و به روز رسانی کنید. بنابراین، برنامه به روز رسانی مربوطه به تأخیر در commit بستگی دارد.

در اینجا، برنامه های در حال اجرا، برنامه های خود را به روز رسانی می کنند. و، به منظور تضمین سازگاری نتایج آزمایشی، دستورالعمل های زیر انجام شد: نتیجه هایی که از مجموعه فراخوانده شده به Resource انتخاب شد، که از طریق پردازش تراکنش ها انجام شد؛ برخی از انواع اطلاعات به کنسول منتقل شد. دومی اطلاعات قفل بود، اطلاعات بهروزرسانی، دوباره اطلاعات و session را دوباره بارگذاری کرد. مبلغ حساب هر حساب قبل از اعدام ۱،۰۰۰ + ۱،۰۰۰ × شماره حساب سپس، برنامه به روز رسانی تعادل هر حساب با اضافه کردن ۲۰،۰۰۰ در این آزمایشات به روز شد.

همانطور که برای برنامه پرس و جو از خواندن uncommitted (۱)، تصویر شماره ۷ (۱–۱) و (۲–۱) نتایج پرس و جو را در مورد اتمام موفقیت آمیز و شکست به ترتیب نشان می دهد. از آنجا که تعهد برنامه بروزرسانی هر ۱۰ به روز رسانی انجام می شود، محور عمودی نشان دهنده نقطه تسلیم است. در مورد اتمام موفقیت آمیز (۱–۱)، تا زمانی که برنامه پرس و جو برنامه را به اتمام برساند، اولین سؤال قبل از به روز رسانی داده ها را پرسید. در غیر این صورت، داده ها را بعد از به روز رسانی درخواست کرد. علاوه بر این، از آنجایی که داده های غیرقابل قبول را مورد پرسش قرار داده است، نقطه تغییر با نقطه تعهد سازگار نیست. در مورد شکست (۲–۱)، از سوی دیگر، در حساب های بین ۳۱ و ۳٤ و بین ٤۱ و ٤٤، برنامه پرس و جو داده های داده شده را پس از آن فراخوانی کرد. همانطور که در تصویر شماره زیر وجود دارد، باز گرداندن انجام شد



تصویر شماره ۷ – عملکرد READ UNCOMMITTED



تصویر شماره ۹- کارایی و بروز رسانی

زمانی که برنامه پرس و جو حساب ۳٤ را درخواست کرد. همانطور که برای دومین، از زمانی که برنامه پرس و جو برنامه به روز رسانی را از دست داده بود، داده ها قبل از رد درخواست مورد سوال قرار گرفتند.

به طور مشابه، تصویر شماره ۷ (۱-۲) و (۲-۲) نتایج تجربی خواندن commit (۲) را نشان می دهد. همانطور که برای اتمام موفقیت آمیز (۱-۲)، از آنجایی که خواندن کثیف رخ نمی داد، داده های درخواست شده قبل و بعد از داده های به روز شده توسط مرحله تساوی تقسیم شدند؛ همانطور که برای شکست (۲-۲)، تمام داده های پرس و جو قبل از به روز رسانی بود. علاوه بر این، از آنجا که قفل مشترک در برنامه پرس و جو اجرا می شود، تأخیر به علت اختلاف در ۳,٦ برابر در یک قفل مشترک، ۰,۳ بار در قفل منحصر به فرد به طور متوسط رخ داده است. در اینجا، سابق در برنامه پرس و جو رخ داد؛ دومی در برنامه به روز رسانی رخ داده است.

همانطور که برای Repeatable read(۳)، از زمانیکه تنظیم شده است، تکمیل شده است، برنامه پرس و جوی تجربی به صورت زیر تهیه شده است: ۱۰ داده را به صورت پیوسته نمایش می دهیم پس از تاخیر (۲۵ مگابایت)، این داده ها را یک بار دیگر به همان شیوه نمایش داده می شود؛ بعد، آن commitمی شود؛ پس از تاخیر (۵۰ مگابایت)، پردازش بعدی ۱۰ روزه شروع می شود. تصویر شماره ۷، این نتایج به صورت پیشفرض نتیجه می شود؛ نتایج پرس و جو نتایج نشان می دهد. در نتیجه، هر دو آنها یکسان بودند، یعنی نتایج پرس و جو تغییر نکرد حتی اگر دوباره آنها را مورد پرسش قرار داد. در اینجا، نتایج پرس و جو درست مثل (۲) در هر دو مورد از (۱-۳) و (۲-۳) است. علاوه بر این، تاخیر ناشی از تداخل ٤۸ برابر در قفل مشترک، ۱,٤ بار در قفل منحصر به فرد به طور متوسط رخ داده است.

دوم، ما مقایسه عملکرد بین سطح جداسازی "Read uncommitted" و "findOne" را مورد بررسی قرار دادیم. دومی، بیانیه استاندارد MongoDB برای پرسیدن اطلاعات تک مشخص شده است. در این آزمایش، تعداد مشخصی از داده ها قبل از آن در جدول حساب ذخیره شده بود، و همه داده ها یک به یک مورد پرس و جو قرار گرفتند. وجود داشت که در آن فایده نداشت. ۸ علاوه بر این، برای ارزیابی عملکرد همزمان بیش از دو اسناد همزمان، آزمایش

انجام شده برای به روز رسانی ۱۰۲٤ داده انجام دادیم. همانطور که در تصویر شماره ۹ نشان داده شده است، مرتکب برای هر بروز رسانی از شماره مشخص شده انجام شد. از طریق این آزمایش، ما نتیجه را دریافت کردیم تا بتوانیم بیش از ۳۰ اسناد را به طور همزمان در این مورد به روز کنیم.

ملاحظات

همانطور که در تصویر شماره ۷ نشان داده شده است. روش پیشنهادی به دست آوردن سطوح بازدهی مجدد، تصویر شماره ۷ می تواند حل شود. به این ترتیب، این برنامه ها می توانند از طریق روش های متفاوتی تحویل شوند. علاوه بر این، سطح ایزولاسیون را می توان برای هر تراکنش مشخص کرد. بنابراین، همانطور که برای حساب بانکی، که در آزمایش استفاده می شود، می توان هر تراکنش را با سطح isolation مورد نیاز کسب و کار به صورت زیر انجام داد. اول، در موردی که پرس و جو از تعادل به سفارش دقیق نیاز ندارد، می توان آن را با Read uncommitted انجام داد. به طور خاص، همانطور که برای تراکنش پرس و جو، از آنجایی که داده های هدف را قفل نمی کند، تاخیر ناشی از اختلال رخ نمی دهد. ثانیا، در صورت انتقال حساب بانکی بین دو حساب، تراکنش با Read انجام شده می تواند ثبات خواص ACID را حفظ کند. به عبارت دیگر، مقدار آنها به عنوان مقدار مشخصی مورد پرسش قرار می گیرد. با این حال، در این مورد، فرض می شود که هر یک از حساب ها با یک دستکاری به روز شده، مانند افزایش یا کاهش مقدار مشخص شده است. سوم، در صورتی که تراکنش اول تعادل را نمایش دهد و مقدار تخلیه را تعیین کند، ضروری است که پس از انجام این پرس و جو از بروزرسانی آن از تراکنش ها دیگر جلوگیری شود. بنابراین، در این مورد، تراکنش باید با خواندن تکرار پذیر انجام شود.

علاوه بر این، همانطور که برای پرس و جو با Read uncommitted از این روش، عملکرد آن تقریبا معادل پیدا کردن یک روش است که در تصویر شماره ۸ نشان داده شده است. در زیر، باید در نظر داشت که دسترسی معمولا بر روی یک سند در MongoDB انجام می شود؛ تراکنش ها با سطح جداسازی دیافراگم می تواند همزمان در این روش انجام شود. لذا با استفاده از این روش در نظر گرفته شده است که پردازش تراکنش مورد نیاز را می توان با حفظ قابلیت اطمینان به عنوان یک سیستم کل به شرح زیر استفاده کرد. تراکنش با دسترسی متداول با Read uncommitted انجام می شود؛ برای به روز رسانی چندین اسناد به طور همزمان، تراکنش با سطح isolation لازم انجام می شود، یعنی خواندن مرتکب یا Repeatable read. و، همانطور که در تصویر شماره ٦ نشان داده شده است، این روش می تواند به عنوان یک برنامه کاربردی در MongoDB اجرا شود. علاوه بر این، همانطور که در تصویر شماره ٤ نشان داده شده است، اطلاعات قفل در سند فردی جمع آوری داده ها ذخیره می شود؛ اطلاعات تراکنش به طور مشابه در مجموعه TP ذخیره می شود. بنابراین، این روش را می توان با روش های معمول MongoDB اجرا کرد. به عبارت دیگر، این روش نه تنها پردازش تراکنش را برای ارسال چندین اسناد از MongoDB استفاده کرد بلکه همچنین می تواند برای حفظ مقیاس پذیری با استفاده از توابع به عنوان یک پایگاه داده توزیع شده از MongoDB، مانند sharing باشد.

٦. نتیجه گیری

اگر چه MongoDB اطلاعات متنوعی را به دست میگیرد، که نمیتواند اسناد چندگانه را با حفظ خواص ACID تراکنش دستکاری کند. برای این منظور روش پردازش تراکنش را برای MongoDB ارئه می کنیم، که از مزایای ساختار داده انعطاف پذیر بدون طرح استفاده می کند: هر سند قبل و بعد از به روزرسانی است و داده های معتبر بر اساس وضعیت تراکنش سوالی می شود. علاوه بر این، از طریق آزمایش، ما اثبات کردیم که سطح جداسازی زیر می تواند توسط این روش تشکیل شود:

- Read uncommitted

- Read committed
- Readable repeatable

این سیستم را طراحی کرده است که بتواند اطلاعات زیادی را در اختیار شما قرار دهد. زمانی که بروز رسانی انجام می شود، داده های آن در MongoDB ذخیره می شود. با این حال، از آنجا که داده ها توسط دستگاه و پردازنده ها به طور همزمان افزوده شده و به روز می شوند، عملیات ترانزیت پردازش، نیاز به ترانزیت، مدیر ترانزیت خواهد داشت.

منابع

۱. Laney, D.. 3d data management: Controlling data volume, velocity and variety. META Group Research Note 2001;6:70. 2. Redmond, E.,Wilson, J.R.. Sevendatabasesinsevenweeks: aguidetomoderndatabasesandtheNoSQLmovement. PragmaticBookshelf; 2012. 3. Gray, J., Reuter, A.. Transaction processing: concepts and techniques. Elsevier; 1992. 4. Cattell, R.. Scalable SQL and NoSQL data stores. ACM SIGMOD Record 2011;39(4):12–27. 5. Tiwari, S.. Professional NoSQL. John Wiley & Sons; 2011. 6. Date, C.. An introduction to database systems. person. 2004. 7. MongoDB, I.. The MongoDB 3.2 Manual; 2015. URL: http://docs.mongodb.org/manual/. 8. Seguin, K.. The little mongodb book. 2011. URL: http://openmymind.net/mongodb.pdf. 9. Connolly, T.M.,Begg, C.E.. DatabaseSystems: A PracticalApproachto Design, Implementation and Management. AddisonWesleyPubl; 2009. 10. Gray, J.N., Lorie, R.A., Putzolu, G.R., Traiger, I.L.. Granularity of locks and degrees of consistency in a shared data base. In: IFIP Working Conference on Modelling in Data Base Management Systems. 1976, p. 365–394. 11. Sriparasa, S.S.. JavaScript and JSON Essentials. Packt Publishing Ltd; 2013. 12. Shastry, K., Madhyastha, S., Kumar, S., Bresniker, K.M., Battas, G.. Transaction support for HBase. In: Proceedings of the 20th International Conference on Management of Data; COMAD '14. 2014, p. 117–120. 13. Garcia-Molina, H., Salem, K.. Sagas. ACM 1987;16(3):249–259. 14. Kudo, T., Takeda, Y., Ishino, M., Saotome, K., Kataoka, N.. Evaluation of lump-sum update methods for nonstop service system. Int, Journal of Informatics Society 2013;5(1):21–28.