عنوان: **چه زمانی بهتر است از بانک‌های اطلاعاتی NoSQL استفاده کرد و چه زمانی خیر؟**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۶/۰۷ ۲۱:۰
آدرس: www.dotnettips.info
برچسب‌ها: NoSQL

در سناریوهای خاصی، بانک‌های اطلاعاتی NoSQL خوش می‌درخشند و در بسیاری از موارد دیگر، بانک‌های اطلاعاتی رابطه‌ای بهترین گزینه انتخابی می‌باشند و نه بانک‌های اطلاعاتی NoSQL. در ادامه به بررسی این موارد خواهیم پرداخت.

در چه برنامه‌هایی استفاده از بانک‌های اطلاعاتی NoSQL مناسب‌تر است؟

1) برنامه‌های مدیریت محتوا
2) کاتالوگ‌های محصولات (هر برنامه‌ای با تعدادی شیء و خصوصا متادیتای متغیر)
3) شبکه‌های اجتماعی
4) Big Data
5) سایر

1) برنامه‌های مدیریت محتوا

بانک‌های اطلاعاتی NoSQL سندگرا، جهت تهیه برنامه‌های مدیریت محتوا، بسیار مناسب هستند. در این نوع برنامه‌ها، یک سری محتوا که دارای متادیتایی هستند، ذخیره خواهند شد. این متادیتاها مانند نوع، گروه و هر نوع خاصیت دیگری، می‌تواند باشند. برای ذخیره سازی این نوع اطلاعات، جفت‌های key-value بسیار خوب عمل می‌کنند. همچنین در بانک‌های اطلاعاتی سندگرای NoSQL، با استفاده از مفهوم برچسب‌ها، امکان الصاق فایل‌های متناظری به اسناد پیش بینی شده‌است. همانطور که در قسمت قبل نیز ذکر شد، در Document stores، نگارش‌های قدیمی اسناد نیز حفظ می‌شوند. به این ترتیب، این خاصیت و توانمندی توکار، امکان دسترسی به نگارش‌های مختلف یک محتوای خاص را به سادگی میسر می‌سازد. به علاوه اکثر Document stores امکان دسترسی به این مستندات را به کمک URLها و REST API، به صورت خودکار فراهم می‌سازند.

برای نمونه به CouchDB ، عنوان Web database نیز داده شده است؛ از این جهت که یک برنامه وب را می‌توان داخل بانک اطلاعاتی آن قرار داد. در اینجا منظور از برنامه وب، یک وب سایت قابل دسترسی از طریق URLها است و نه برنامه‌های سازمانی وب. برای نمونه ساختاری شبیه به برنامه معروف EverNote را می‌توان داخل این نوع بانک‌های اطلاعاتی به سادگی ایجاد کرد (خود بانک اطلاعاتی تشکیل شده است از یک وب سرور که REST API را پشتیبانی کرده و امکان دسترسی به اسناد را بدون نیاز به کدنویسی اضافه‌تری، از طریق URLها و HTTP Verbs استاندارد مهیا می‌کند).

2) کاتالوگ‌های محصولات

محصولات در یک کاتالوگ، ویژگی‌های مشابه یکسان فراوانی دارند؛ اما تعدادی از این محصولات، دارای ویژگی‌هایی خاص و منحصربفردی نیز می‌باشند.

مثلا یک شیء محصول را درنظر بگیرید که دارای خواص مشترک و یکسان شماره، نام، توضیحات و قیمت است. اما بعضی از محصولات، بسته به رده‌ی خاصی که دارند، دارای ویژگی‌های خاصی مانند قدرت تفکیک، رنگ، سرعت و غیره نیز هستند که از هر گروه، به گروه دیگری متغیر است.

برای مدیریت یک چنین نیازی، هر دو گروه key-value stores و wide column stores بانک‌های اطلاعاتی NoSQL مناسب هستند؛ از این جهت که در یک key-value store نیازی به تعریف هیچ نوع ساختار خاصی، در ابتدای کار نیست و این ساختار می‌تواند از هر رکورد، به رکورد دیگری متفاوت باشد.

یا برای نمونه، یک برنامه فرم ساز را درنظر بگیرید که هر فرم آن، هر چند دارای یک سری خواص ثابت مانند نام، گروه و امثال آن است، اما هر کدام دارای فیلدهای تشکیل دهنده متفاوتی نیز می‌باشد. به این ترتیب با استفاده از key-value stores، دیگری نیازی به نگران بودن در مورد نحوه مدیریت اسکیمای متغیر مورد نیاز، نخواهد بود.

3) شبکه‌های اجتماعی

همانطور که در قسمت قبل نیز بحث شد، نوع خاص Graph databases برای کاربردهای برنامه‌های شبکه‌های اجتماعی و ردیابی تغییرات آن‌ها بسیار مفید و کارآ هستند. برای مثال در یک شبکه افراد دارای تعدادی دنبال کننده هستند؛ عضو گروه‌های مختلف می‌باشند، در قسمت‌های مختلفی نظر و مطلب ارسال می‌کنند. در اینجا، اشیاء نسبت به یکدیگر روابط مختلفی دارند. با استفاده از

Graph databases، تشکیل روابط self-joins و تو در تو و بسیاری از روش‌های خاص، مانند روابط many-to-many که در بانک‌های اطلاعاتی رابطه‌ای با تمهیدات ویژه‌ای قابل تشکیل هستند، با سهولت بهتری مدیریت خواهند شد.

4) Big Data

الگوریتم MapReduce، برای کار با حجم داده‌های عظیم، طراحی شده است و در این بین، بانک‌های اطلاعاتی Wide column store (که در قسمت قبل بررسی شدند) و یا حتی Key-value store (مانند Amazon DynamoDB) بیشتر کاربرد دارند. در سناریوهای داده‌های عظیم، واژه‌های Hadoop و Hbase دنیای NoSQL را زیاد خواهید شنید. Hadoop نسخه سورس باز MapReduce گوگل است و Hbase نیز نسخه سورس باز BigTable گوگل می‌باشد. مفاهیم پایه‌ای Sharding و فایل سیستم‌های Append-only (با سرعت بالای نوشتن) نیز به مدیریت BigData کمک می‌کنند.

در اینجا بحث مهم، خواندن اطلاعات و آنالیز آن‌ها است و نه تهیه برنامه‌های معروف CRUD . بسیاری از اعمال آماری و ریاضی مورد نیاز بر روی داده‌های عظیم، نیازی به اسکیمای از پیش مشخص شده بانک‌های اطلاعاتی رابطه‌ای را ندارند و یا در اینجا قابلیت‌های نوشتن کوئری‌های پیچیده نیز آنچنان مهم نیستند.

5) سایر کاربردها

- هر سیستمی که اطلاعات Log مانند را تولید می‌کند. منظور از Log، اطلاعاتی است که در حین رخداد خاصی تولید می‌شوند. عموما مرسوم است که این نوع اطلاعات را در فایل‌ها، بجای بانک اطلاعاتی ذخیره کرد. بنابراین مدیریت این نوع فایل‌ها توسط بانک‌های اطلاعاتی NoSQL، قابلیت انجام امور آماری را بر روی آن‌ها ساده‌تر خواهد ساخت.
- مدیریت اطلاعات برنامه‌هایی مانند سیستم‌های EMail.

و در چه برنامه‌هایی استفاده از بانک‌های اطلاعاتی رابطه‌ای مناسب‌تر است؟

اگر تا اینجا به مزایای استفاده از بانک‌های اطلاعاتی NoSQL اشاره شد، بدین معنا نیست که بانک‌های اطلاعاتی رابطه‌ای، منسوخ شده‌اند یا دیگر قدر و قیمتی ندارند. واقعیت این است که هنوز بازه وسیعی از کاربردها را می‌توان به کمک بانک‌های اطلاعاتی رابطه‌ای بهتر از بانک‌های اطلاعاتی NoSQL مدیریت کرد. این کاربردها و مزیت‌ها در 5 گروه عمده خلاصه می‌شوند:

1) نیاز به تراکنش‌ها
2) اسکیمای پیش فرض
3) برنامه‌های LOB یا Line of business applications
4) زبان‌های کوئری نویسی پیشرفته
5) نیاز به امکانات گزارشگیری پیشرفته

1) نیاز به تراکنش‌ها

در سیستم‌های تجاری عمومی، نیاز به پیاده سازی مفهوم ACID که در قسمت‌های قبل به آن پرداخته شد، مانند Atomic transactions وجود دارد. Atomic transaction به زبان ساده به این معنا است که سیستم قادر است چندین دستور را در قالب یک گروه و در طی یک مرحله، به بانک اطلاعاتی اعمال کند و اگر یکی از این دستورات گروه در حال اعمال، با شکست مواجه شد، باید کل تراکنش برگشت خورده و امنیت کار تضمین گردد. در غیراینصورت با یک سیستم غیر هماهنگ مواجه خواهیم شد.

و همانطور که پیشتر نیز عنوان شد ، سیستم‌های NoSQL، مبنای کار را بر اساس «عاقبت یک دست شدن» اطلاعات قرار داده‌اند؛ تا دسترسی پذیری به آن‌ها افزایش یافته و سرعت عملیات به این نحو بهبود یابد. در این نوع سیستم‌ها تضمینی در مورد ACID وجود ندارد.

2) اسکیمای پیش فرض

پروسه‌های متداول، دارای ساختاری مشخص و معمولی هستند. زیرا طراحی اولیه یک پروسه، بر مبنای مجموعه‌ای از اطلاعات است که همیشه باید وجود داشته باشند و اگر همانند بحث کاتالوگ‌های محصولات، نیاز به متادیتای متغیر نباشد، ساختار و اسکیمای یک پروسه، از ابتدای طراحی آن مشخص می‌باشد.

و ... تمام این‌ها را به خوبی می‌توان توسط بانک‌های اطلاعاتی رابطه‌ای، با تعریف یک اسکیمای مشخص، مدیریت کرد.

3) برنامه‌های LOB یا Line of business applications

در برنامه‌های تجاری متداول، طراحی طرحبندی فرم‌های برنامه یا انقیاد داده‌ها، بر اساس یک اسکیما و ساختار مشخص صورت می‌گیرد. بدون داشتن یک اسکیمای مشخص، امکان تعاریف انقیاد داده‌ها به صورت strongly typed وجود نخواهد داشت. همچنین کل مفهوم Object relational mapping و ORMهای مختلف نیز بر اساس وجود یک اسکیمای مشخص و از پیش تعیین شده کار می‌کند. بنابراین بانک‌های اطلاعاتی رابطه‌ای، انتخاب بسیار مناسبی برای تهیه برنامه‌های تجاری روزمره هستند.

4) زبان‌های کوئری نویسی پیشرفته

همانطور که عنوان شد برای تهیه کوئری بر روی اغلب بانک‌های اطلاعاتی NoSQL، باید توسط یک برنامه ثانویه، کار پیاده سازی الگوریتم Map Reduce را انجام داد. هر چند تعدادی از این نوع بانک‌های اطلاعاتی به صورت توکار دارای موتور MapReduce هستند، اما بسیاری از آن‌ها خیر. به همین جهت برای تهیه کوئری‌های متداول، کار پیاده سازی این برنامه‌های ثانویه مشکل خواهد بود. به این ترتیب نوشتن Ad Hoc queries و گزارشگیری بسیار مشکل می‌شوند.

علاوه بر امکانات خوب کوئری گرفتن در بانک‌های اطلاعاتی رابطه‌ای، این کوئری‌ها در زمان اجرا نیز بر اساس اسکیمای موجود، بسیار بهینه و با سرعت بالا اجرا می‌شوند. قابلیتی که رسیدن به آن در بانک‌های اطلاعاتی با اسکیمای متغیر، کار ساده‌ای نیست و باید آن‌را با کدنویسی شخصی بهینه کرد. البته اگر تعداد این نوع برنامه‌های ثانویه که به آن‌ها imperative query در مقابل declarative query بانک‌های رابطه‌ای می‌گویند، کم باشد، شاید یکبار نوشتن و بارها استفاده کردن از آن‌ها اهمیتی نداشته باشد؛ در غیراینصورت تبدیل به یک عذاب خواهد شد.

5) نیاز به امکانات گزارشگیری پیشرفته

گزارشگیرهای برنامه‌های تجاری نیز بر اساس یک ساختار و اسکیمای مشخص به کمک قابلیت‌های پیشرفته کوئری نویسی بانک‌های اطلاعاتی رابطه‌ای به سادگی قابل تهیه هستند. برای تهیه گزارشاتی که قابلیت چاپ مناسبی را داشته باشند، محل قرارگیری فیلدهای مختلف در صفحه مهم هستند و با متغیر بودن آن‌ها، قابلیت طراحی از پیش آن‌ها را از دست خواهیم داد. در این حالت با اسکیمای متغیر، حداکثر بتوان یک dump از اطلاعات را به صورت ستونی نمایش داد.

بنابراین به صورت خلاصه، بانک‌های اطلاعاتی رابطه‌ای، جهت مدیریت کارهای روزمره تجاری اغلب شرکت‌ها، بسیار ضروری و جزو مسایل پایه‌ای به‌شمار می‌روند و به این زودی‌ها هم قرار نیست با نمونه‌ی دیگری جایگزین شوند.

نظرات خوانندگان

نویسنده: مرادی
تاریخ: ۱۳۹۲/۰۶/۱۷ ۸:۱۲

با سلام، با توجه به این که Raven Db یکی از دیتابیس‌های No SQL، در پشت صحنه خود از دیتابیس ویندوز با نام ESENT استفاده می‌کند، که مطابق با مستندات سایت‌های معتبر چون MSDN و ویکی پدیا هم Transactional است، و هم دارای امکانات دیگر، آیا می‌توان Transactional بودن را مزیتی صرف برای دیتابیس‌های رابطه ای به شمار آورد، و برای دیتابیس‌های No SQL این مزیت را قائل نشد ؟

http://blogs.msdn.com/b/windowssdk/archive/2008/10/23/esent-extensible-storage-engine-api-in-the-windows-sdk.aspx Windows comes with an embeddable, transactional database engine

http://en.wikipedia.org/wiki/Extensible_Storage_Engine

ESE provides transacted data update and retrieva

با سپاس

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۶/۱۷ ۹:۲۲

- بحث فوق، بحثی است عمومی و حاصل برآیند بررسی اکثر بانک‌های اطلاعاتی NoSQL.
- بدیهی است این بین ممکن است استثناءهایی هم وجود داشته باشند. برای مثال RavenDB داخل document store خود transactional عمل می‌کند؛ اما کوئری‌های آن (که بر روی ایندکس‌های لوسین اجرا می‌شوند) از اصل عاقبت یک‌دست شدن پیروی می‌کنند. همچنین این کوئری‌ها را هم می‌شود طوری تنظیم کرد که stale data باز نگردانند.