عنوان: **مروری بر مفاهیم مقدماتی NoSQL**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۶/۰۴ ۲۳:۵
آدرس: www.dotnettips.info
گروه‌ها: NoSQL

هدف از این مبحث، آشنایی با مفاهیم پایه‌ای اغلب بانک‌های اطلاعاتی NoSQL است که به صورت مشترکی در تمام آن‌ها بکار رفته است. برای مثال بانک‌های اطلاعاتی NoSQL چگونه مباحث یکپارچگی اطلاعات را مدیریت می‌کنند؟ نحوه ایندکس نمودن اطلاعات در آن‌ها چگونه است؟ چگونه از اطلاعات کوئری می‌گیرند؟ الگوریتم‌های محاسباتی مانند MapReduce چیستند و چگونه در اینگونه بانک‌های اطلاعاتی بکار رفته‌اند؟ همچنین الگوهای Sharding و Partitioning که در اغلب بانک‌های اطلاعاتی NoSQL مشترکند، به چه نحوی پیاده سازی شده‌اند.

لیست مشترکات بانک‌های اطلاعاتی NoSQL

قبل از اینکه بخواهیم وارد ریز جزئیات بانک‌های اطلاعاتی NoSQL شویم، نیاز است لیست و سرفصلی از مفاهیم اصلی و مشترک بین اینگونه بانک‌های اطلاعاتی را تدارک ببینیم که شامل موارد ذیل می‌شود:

الف) Non-Relational یا غیر رابطه‌ای

از کلمه NoSQL عموما اینطور برداشت می‌شود که در اینجا دیگر خبری از SQL نویسی نیست که در عمل برداشت نادرستی است. شاید جالب باشد که بدانید، تعدادی از بانک‌های اطلاعاتی NoSQL از زبان SQL نیز به عنوان اینترفیسی برای نوشتن کوئری‌های مرتبط، پشتیبانی می‌کنند.
کلمه NoSQL بیشتر به Non-Relational یا غیر رابطه‌ای بودن اینگونه بانک‌های اطلاعاتی بر می‌گردد. مباحثی مانند مدل‌های داده‌ای نرمال شده، اتصالات و Join جداول، در دنیای NoSQL وجود خارجی ندارند.

ب) Non-schematized/schema free یا بدون اسکیما

مفهوم مهم و مشترک دیگری که در بین بانک‌های اطلاعاتی NoSQL وجود دارد، بدون اسکیما بودن اطلاعات آن‌ها است. به این معنا که با حرکت از رکورد یک به رکورد دو، ممکن است با دو ساختار داده‌ای متفاوت مواجه شوید.

ج) Eventual consistency یا عاقبت یک دست شدن

عاقبت یک دست شدن، به معنای دریافت دستوری از شما و نحوه پاسخ دادن به آن (یا حتی پاسخ ندادن به آن) از طرف بانک اطلاعاتی NoSQL است. برای مثال، زمانیکه یک رکورد جدید را اضافه می‌کنید، یا اطلاعات موجودی را به روز رسانی خواهید کرد، اغلب بانک‌های اطلاعاتی NoSQL این دستور را بسیار سریع دریافت و پردازش خواهند کرد. اما تفاوت است بین دریافت پیام و پردازش واقعی آن در اینجا.
اکثر بانک‌های اطلاعاتی NoSQL، پردازش و اعمال واقعی دستورات دریافتی را با یک تاخیر انجام می‌دهند. به این ترتیب می‌توان خیلی سریع به بانک اطلاعاتی اعلام کرد که چه می‌خواهیم و بانک اطلاعاتی بلافاصله مجددا کنترل را به شما بازخواهد گرداند. اما اعمال و انتشار واقعی این دستور، مدتی زمان خواهد برد.

د) Open source یا منبع باز بودن

اغلب بانک‌های اطلاعاتی NoSQL موجود، منبع باز هستند که علاوه بر بهره بردن از مزایای اینگونه پروژه‌ها، استفاده کنندگان سورس باز دیگری را نیز ترغیب به استفاده از آن‌ها کرده‌اند.

ه) Distributed یا توزیع شده

هرچند امکان پیاده سازی توزیع شده بانک‌های اطلاعاتی رابطه‌ای نیز وجود دارد، اما نیاز به تنظیمات قابل توجهی برای حصول این امر می‌باشد. در دنیای NoSQL، توزیع شده بودن جزئی از استاندارد تهیه اینگونه بانک‌های اطلاعاتی است و بر اساس این مدل ذهنی شکل گرفته‌اند. به این معنا که اطلاعات را می‌توان بین چندین سیستم تقسیم کرد، که حتی این سیستم‌ها ممکن است فواصل جغرافیایی قابل توجهی نیز با یکدیگر داشته باشند.

و) Web scale یا مناسب برای برنامه‌های تحت وب پر کاربر

امروزه بسیاری از کمپانی‌های بزرگ اینترنتی، برای مدیریت تعداد بالایی از کاربران همزمان خود، مانند فیس‌بوک، یاهو، گوگل، Linkedin، مایکروسافت و غیره، نیاز به بانک‌های اطلاعاتی پیدا کرده‌اند که باید در مقابل این حجم عظیم درخواست‌ها و همچنین اطلاعاتی که دارند، بسیار بسیار سریع پاسخ دهند. به همین جهت بانک‌های اطلاعاتی NoSQL ابداع شده‌اند تا بتوان برای این نوع سناریوها پاسخی را ارائه داد.

و نکته مهم دیگر اینجا است که خود این کمپانی‌های بزرگ اینترنتی، بزرگترین توسعه دهنده‌های بانک‌های اطلاعاتی NoSQL نیز هستند.

نحوه مدیریت یکپارچگی اطلاعات در بانک‌های اطلاعاتی NoSQL

مدیریت یکپارچگی اطلاعات بانک‌های اطلاعاتی NoSQL به علت ذات و طراحی توزیع شده آن‌ها، با نحوه مدیریت یکپارچگی اطلاعات بانک‌های اطلاعاتی رابطه‌ای متفاوت است. اینجا است که تئوری خاصی به نام CAP مطرح می‌شود که شامل یکپارچگی یا Consistency به همراه Availability یا دسترسی پذیری (همیشه برقرار بودن) و partition tolerance یا توزیع پذیری است. در تئوری CAP مطرح می‌شود که هر بانک اطلاعاتی خاص، تنها دو مورد از سه مورد مطرح شده را می‌تواند با هم پوشش دهد.

به این ترتیب بانک‌های اطلاعاتی رابطه‌ای عموما دو مورد C و P یا یکپارچگی (Consistency) و partition tolerance یا میزان تحمل تقسیم شدن اطلاعات را ارائه می‌دهند. اما بانک‌های اطلاعاتی NoSQL از این تئوری، تنها دو مورد A و P را پوشش می‌دهند (دسترسی پذیری و توزیع پذیری مطلوب).

بنابراین مفهومی به نام ACID که در بانک‌های اطلاعاتی رابطه‌ای ضامن یکپارچگی اطلاعات آن‌ها است، در دنیای NoSQL وجود خارجی ندارد. کلمه ACID مخفف موارد ذیل است:

Durability و Atomicity، Consistency، Isolation

ACID در بانک‌های اطلاعاتی رابطه‌ای تضمین شده است. در این نوع سیستم‌ها، با ایجاد تراکنش‌ها، مباحث ایزوله سازی و یکپارچگی اطلاعات به نحو مطلوبی مدیریت می‌گردد؛ اما دنیای NoSQL، دسترسی پذیری را به یکپارچگی ترجیح داده است و به همین جهت پیشتر مطرح شد که مفهوم «Eventual consistency یا عاقبت یک دست شدن» در این نوع بانک‌های اطلاعاتی در پشت صحنه بکار گرفته می‌شود. یک مثال دنیای واقعی از عاقبت یک دست شدن اطلاعات را حتما در مباحث DNS مطالعه کرده‌اید. زمانیکه یک رکورد DNS اضافه می‌شود یا به روز خواهد شد، اعمال این دستورات در سراسر دنیا به یکباره و همزمان نیست. هرچند اعمال این اطلاعات جدید در یک نود شبکه ممکن است آنی باشد، اما پخش و توزیع آن در سراسر سرورهای DNS دنیا، مدتی زمان خواهد برد (گاهی تا یک روز یا بیشتر).

به همین جهت است که بانک‌های اطلاعاتی رابطه‌ای در حجم‌های عظیم اطلاعات و تعداد کاربران همزمان بالا، کند عمل می‌کنند. حجم اطلاعات بالا است، مدتی زمان خواهد برد تا تغییرات اعمال شوند، و چون مفهوم ACID در این نوع بانک‌های اطلاعاتی تضمین شده است، کاربران باید مدتی منتظر بمانند و نمونه‌ای از آن‌ها را با lock وdead های شایع، احتمالا پیشتر بررسی یا تجربه کرده‌اید. در مقابل، بانک‌های اطلاعاتی NoSQL بجای یکپارچگی، دسترسی پذیری را اولویت اول خود می‌دانند و نه یکپارچگی اطلاعات را. در یک بانک اطلاعاتی NoSQL، دستور ثبت اطلاعات دریافت می‌شود (این مرحله آنی است)، اما اعمال نهایی آن آنی نیست و مدتی زمان خواهد برد تا تمام اطلاعات در کلیه سرورها یک دست شوند.

نحوه مدیریت Indexing اطلاعات در بانک‌های اطلاعاتی NoSQL

اغلب بانک‌های اطلاعاتی NoSQL تنها بر اساس اطلاعات کلیدهای اصلی جداول آن‌ها index می‌شوند (البته نام خاصی به نام «جدول»، بسته به نوع بانک اطلاعاتی NoSQL ممکن است متفاوت باشد، اما منظور ظرف دربرگیرنده تعدادی رکورد است در اینجا). این ایندکس نیز از نوع clustered است. به این معنا که اطلاعات به صورت فیزیکی، بر همین مبنا ذخیره و مرتب خواهند شد.

یک مثال: بانک اطلاعاتی NoSQL خاصی به نام Hbase که بر فراز distributed file system Hadoop طراحی شده است، دقیقا به همین روش عمل می‌کند. این فایل سیستم، تنها از روش Append only برای ذخیره سازی اطلاعات استفاده می‌کند و در آن مفهوم دسترسی اتفاقی یا random access پیاده سازی نشده است. در این حالت، تمام نوشتن‌ها در بافر، لاگ می‌شوند و در بازه‌های زمانی متناوب و مشخصی سبب باز تولید فایل‌های موجود و مرتب سازی مجدد آن‌ها از ابتدا خواهند شد. دسترسی به این اطلاعات پس از تکمیل نوشتن، به علت مرتب سازی فیزیکی که صورت گرفته، بسیار سریع است. همچنین مصرف کننده سیستم نیز چون بلافاصله پس از ثبت اطلاعات در بافر سیستم، کنترل را به دست می‌گیرد، احساس کار با سیستمی را خواهد داشت که بسیار سریع است.

به علاوه Indexهای دیگری نیز وجود دارند که بر اساس کلیدهای اصلی جداول تولید نمی‌شوند و به آن‌ها ایندکس‌های ثانویه یا secondary indexes نیز گفته می‌شود و تنها تعداد محدودی از بانک‌های اطلاعاتی NoSQL از آن‌ها پشتیبانی می‌کنند. این مساله هم از اینجا ناشی می‌شود که با توجه به بدون اسکیما بودن جداول بانک‌های اطلاعاتی NoSQL، چگونه می‌توان اطلاعاتی را ایندکس کرد که ممکن است در رکورد دیگری، ساختار متناظر با آن اصلا وجود خارجی نداشته باشد.

نحوه پردازش Queries در بانک‌های اطلاعاتی NoSQL

بانک‌های اطلاعاتی NoSQL عموما از زبان کوئری خاصی پشتیبانی نمی‌کنند. در اینجا باید به اطلاعات به شکل فایل‌هایی که حاوی رکوردها هستند نگاه کرد. به این ترتیب برای پردازش و یافتن اطلاعات درون این فایل‌ها، نیاز به ایجاد برنامه‌هایی است که این فایل‌ها را گشوده و بر اساس منطق خاصی، اطلاعات مورد نظر را استخراج کنند. گاهی از اوقات زبان SQL نیز پشتیبانی می‌شود ولی آنچنان عمومیت ندارد. الگوریتمی که در این برنامه‌ها بکار گرفته می‌شود، Map Reduce نام دارد.
Map Reduce به معنای نوشتن کدی است، با دو تابع. اولین تابع اصطلاحا Map step یا مرحله نگاشت نام دارد. در این مرحله کوئری به قسمت‌های کوچکتری خرد شده و بر روی سیستم‌های توزیع شده به صورت موازی اجرا می‌شود. مرحله بعد Reduce step نام دارد که در آن، نتیجه دریافتی حاصل از کوئری‌های اجرا شده بر روی سیستم‌های مختلف، با هم یکی خواهند شد.
این روش برای نمونه در سیستم Hadoop بسیار مرسوم است. Hadoop دارای یک فایل سیستم توزیع شده است (که پیشتر در مورد آن بحث شد) به همراه یک موتور Map Reduce توکار. همچنین رده دیگری از بانک‌های اطلاعاتی NoSQL، اصطلاحا Wide column store نام دارند (مانند Hbase) که عموما به همراه Hadoop بکارگرفته می‌شوند. موتور Map Reduce متعلق به Hadoop بر روی جداول Hbase اجرا می‌شوند.
به علاوه Amazon web services دارای سرویسی است به نام Elastic map reduce یا EMR که در حقیقت مجموعه‌ی پردازش ابری است که بر مبنای Hadoop کار می‌کند. این سرویس قادر است با بانک‌های اطلاعاتی NoSQL دیگر و یا حتی بانک‌های اطلاعاتی رابطه‌ای نیز کار کند.
بنابراین MapReduce، یک بانک اطلاعاتی نیست؛ بلکه یک روش پردازش اطلاعات است که فایل‌ها را به عنوان ورودی دریافت کرده و یک فایل را به عنوان خروجی تولید می‌کند. از آنجائیکه بسیاری از بانک‌های اطلاعاتی NoSQL کار عمده‌اشان، ایجاد و تغییر فایل‌ها است، اغلب جداول اطلاعات آن‌ها ورودی و خروجی‌های معتبری برای یک موتور Map reduce به حساب می‌آیند.
در این بین، افزونه‌ای برای Hadoop به نام Hive طراحی شده است که با ارائه HiveSQL، امکان نوشتن کوئری‌هایی SQL مانند را بر فراز موتورهای Map reduce ممکن می‌سازد. این افزونه با Hive tables خاص خودش و یا با Hbase سازگار است.

آشنایی مقدماتی با مفاهیمی مانند الگوهای Sharding و Partitioning در بانک‌های اطلاعاتی NoSQL

Sharding (شاردینگ تلفظ می‌شود) یک الگوی تقسیم اطلاعات بر روی چندین سرور است که اساس توزیع شده بودن بانک‌های اطلاعاتی NoSQL را تشکیل می‌دهد. این نوع تقسیم اطلاعات، از کوئری‌هایی به نام Fan-out پشتیبانی می‌کند. به این معنا که شما کوئری خود را به نود اصلی ارسال می‌کنید و سپس به کمک موتورهای Map reduce، این کوئری بر روی سرورهای مختلف اجرا شده و نتیجه نهایی جمع آوری خواهد شد. به این ترتیب تقسیم اطلاعات، صرفا به معنای قرار دادن یک سری فایل بر روی سرورهای مختلف نیست، بلکه هر کدام از این سرورها به صورت مستقل نیز قابلیت پردازش اطلاعات را دارند.
امکان تکثیر و همچنین replication هر کدام از سرورها نیز وجود دارد که قابلیت بازیابی سریع و مقاومت در برابر خرابی‌ها و مشکلات را افزایش می‌دهند.
از آنجائیکه Shardها را می‌توان در سرورهای بسیار متفاوت و گسترده‌ای از لحاظ جغرافیایی قرار داد، هر Shard می‌تواند همانند مفاهیم CDN نیز عمل کند؛ به این معنا که می‌توان Shard مورد نیاز سروری خاص را در محلی نزدیک‌تر به او قرار داد. به این ترتیب سرعت عملیات افزایش یافته و همچنین بار شبکه نیز کاهش می‌یابد.