فصلنامه علمی-ترویجی پدافند غیرعامل
سال نهم، شماره ۳، پاییز ۱۳۹۷، (پیاپی ۳۵): صص ۱۳۰-۱۱۹

انتخاب بانک اطلاعاتی توزیع‌شده مناسب جهت رسیدن به بهترین عملکرد در مدیریت داده‌های حجیم

سامان کشوری[۱]، مهدی نقوی[۲*]، ساناز کشوری[۳]

تاریخ دریافت: ۱۳۹۶/۰۸/۱۹
تاریخ پذیرش: ۱۳۹۶/۱۱/۰۸

چکیده

انتخاب بانک اطلاعاتی مناسب با توجه به نیازمندی‌های هر برنامه همواره یکی از دغدغه‌های توسعه‌دهندگان است. بانک‌های اطلاعاتی انواع مختلفی دارند که مشهورترین آن‌ها بانک اطلاعاتی رابطه‌ای است. با افزایش استفاده از اینترنت که منجر به رشد سریع ایجاد داده‌ها و نیازمندی‌های جدید در حوزه داده‌های حجیم شد، این نوع بانک‌های اطلاعاتی در مدیریت داده‌های غیر ساخت‌یافته و حجیم عملکرد خوبی نداشتند. جهت مدیریت داده‌هایی با چنین ویژگی‌ها، بانک‌های اطلاعاتی کلید- مقدار، سندگرا، ستون گسترده و مبتنی بر گراف ارائه شده است. این بانک‌ها هر یک با رویکرد خاصی راه‌حل‌هایی را ارائه می‌کنند. بانک‌های اطلاعاتی چند مدلی با رویکرد چندگانه با این ویژگی‌ها ارائه شده‌اند که قادر هستند راه‌حل‌های هم‌زمان برای چندین نیاز را پشتیبانی نمایند. در این مقاله ابتدا با مبنا قرار دادن مدل رابطه‌ای و با ذکر یک مثال، ضمن تشریح ساختار و امکانات انواع بانک‌های غیر رابطه‌ای مذکور، مثال مربوطه با توجه به هر یک از این مدل‌ها ارائه شده است. سپس از هر نوع، چندین بانک معرفی کرده و یک بانک پرکاربرد را جهت مقایسه ساختار و ویژگی‌ها با یکدیگر انتخاب و عملکرد آن‌ها مورد بررسی قرار گرفته است. معیارهای اصلی مقایسه سرعت ذخیره‌سازی و بازیابی اطلاعات بوده و در ارائه نتایج سعی شده معیارهای ارزیابی با توجه به رویکردها و نیازهای مختلف مورد بررسی و ارزیابی قرار گیرند. هدف این مقاله ارائه نمونه برتر از بانک‌های اطلاعاتی غیر رابطه‌ای نیست، بلکه هدف پیشنهاد بانک غیر رابطه‌ای مناسب برای کسب کارایی در کاربردهای مختلف بر اساس نیازهای تعریف شده است.

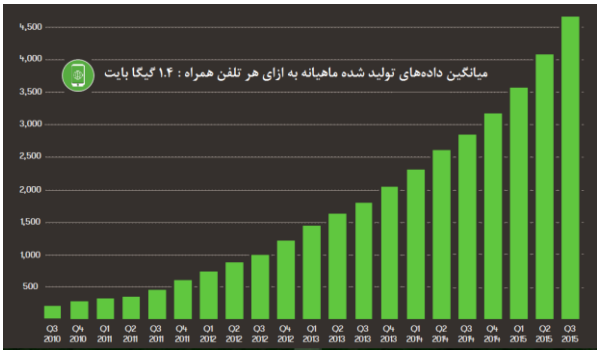
کلیدواژه‌ها: داده‌های حجیم، بانک‌های اطلاعاتی غیر ساخت‌یافته، بانک اطلاعاتی توزیع‌شده، NoSQL.

۱- دانشجوی کارشناسی ارشد نرم‌افزار، دانشگاه جامع امام حسین(ع)
۲- استادیارگروه کامپیوتر دانشگاه جامع امام حسین(ع)، mnaghavi@ihu.ac.ir- نویسنده مسئول
۳- دانشجوی دکتری نرم‌افزار، دانشگاه فردوسی مشهد

۱- مقدمه

در طول تاریخ استفاده از کامپیوتر انواع مدل مختلفی ازجمله مدل فایلی، مدل سلسله مراتبی و شبکه‌ای جهت ذخیره‌سازی اطلاعات مرسوم بوده است. در سال ۱۹۷۰ مدل رابطه‌ای که یک مدل منطقی بر مبنای ریاضیات است از منطق گزاره‌ها و نظریه مجموعه‌ها به‌عنوان زیربنا استفاده‌شده، توسط ادگارد کاد[1] ارائه شد. پایگاه داده رابطه‌ای با مدل رابطه‌ای مطابقت دارد و مجموعه‌ای از جدول‌ها است که از دید کاربر قابل‌درک هستند [۱]. در دهه ۱۹۸۰ زمانی قابلیت‌های پایگاه داده با قابلیت‌های زبان برنامه‌نویسی شیءگرا ترکیب شدند، پایگاه داده شیءگرا[2] به وجود آمد. این پایگاه داده به برنامه‌های شیءگرا اجازه می‌دهند که اطلاعات را به‌صورت اشیاء در بانک اطلاعاتی ذخیره، بازیابی و بروز رسانی کنند [۲]. علاوه بر مدل‌های داده‌ای و نحوه ذخیره‌سازی داده، نسل‌های مختلفی از سیستم‌های پایگاه داده ازجمله سیستم‌های مدیریت پایگاه داده بی‌درنگ[3] [۳]، سیستم مدیریت پایگاه داده تحمل‌پذیر در برابر خطا[4] [۴]، سیستم مدیریت پایگاه داده مطمئن[5] [۵]، سیستم مدیریت پایگاه داده ناهمگون[6] [۶]، سیستم مدیریت پایگاه داده چندرسانه‌ای[7] [۷]، سیستم مدیریت پایگاه داده متحد[8] و سیستم مدیریت پایگاه داده توزیع‌شده وجود دارد که بر توزیع داده و همچنین همبستگی فعالیت‌ها و کنترل روی اجزای توزیع‌شده سیستم دلالت دارند [۸]. سرعت رشد داده‌ها در جهان معاصر با افزایش نفوذ اینترنت در زندگی، خصوصاً استفاده از تلفن‌های همراه هوشمند و به وجود آمدن شبکه‌های اجتماعی، شتاب بسیار زیادی به خود گرفته است. به‌طور مثال در شکل ۱ میزان رشد داده‌هایی که از طریق موبایل در جهان از سال ۲۰۰۹ تا ۲۰۱۵ تولیدشده‌اند را نشان می‌دهد، طبق این آمار به ازای هر موبایل به‌طور میانگین حدود ۱,۴ گیگابایت داده در ماه تولید می‌شود [۹].

تعداد کاربرانی که در جهان از اینترنت استفاده می‌کنند بیش از ۳،۳۳۴،۵۵۰،۱۲۰ و تعداد وب‌سایت‌ها ۹۴۹،۸۸۷،۸۸۲ است. جدول (۱) آمار بیشتری را در این رابطه نشان می‌دهد[9]. در شبکه اجتماعی فیس‌بوک حدود ۱،۴۱۶،۴۲۴،۵۰۰ تعداد کاربر فعال حضور دارد این آمار برای شبکه‌های اجتماعی گوگل پلاس و توییتر به ترتیب ۱،۱۳۰،۳۶۴،۰۰۰ و ۳۰۹،۱۸۱،۲۰۰ عدد است [۹]. حجم پایگاه داده EBay به حدود ۹۰ پتابایت رسیده است [۱۰]. این آمارها به‌صورت لحظه‌ای رو به رشد هستند. با توجه به این حجم بالا، پردازش سنگین و همچنین تنوع زیادی که در داده‌ها مشاهده می‌شود، نیاز به روش‌هایی جهت ذخیره‌سازی این حجم از داده‌ها وجود دارد که بتواند اطلاعات موردنیاز را با سرعت و دقت بالا واکشی کند.



شکل ۱- نمودار داده‌های تولیدشده توسط موبایل [۹]

جدول ۱-آمار ترافیک چندین سایت معتبر ۱۰

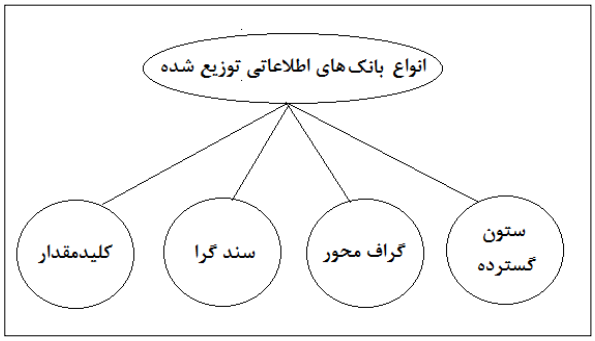
عنوان	تعداد در ثانیه	تعداد در روز
جستجو گوگل	۲۷،۷۱۹	۲،۳۹۴،۹۲۱،۶۰۰
ایمیل‌های ارسالی	۲،۳۹۸،۸۷۰	*۲۰۷،۲۶۲،۳۶۸،۰۰۰
مشاهده فیلم (یوتیوب)	۱۰۱،۶۷۶	۸،۷۸۴،۸۰۶،۴۰۰
آپلود اینستاگرام	۲،۱۵۶	۱۸۶،۲۷۸،۴۰۰
توییت در توییتر	۹،۲۹۸	۸۰۳،۳۴۷،۲۰۰

* طبق آمار ارائه‌شده حدود ۶۷٪ ایمیل‌ها هرزنامه هستند.

با توجه به ضعف‌هایی که سیستم‌های ذخیره‌سازی سنتی دارند، این نوع ساختارها جوابگو برخی از نیازهای امروزی نیستند [۱۱]. مدل‌های رابطه‌ای، بیش از بیست سال در سیستم‌های مدیریت داده‌ای در سازمان‌های بزرگ استفاده شدند، به‌طوری‌که برای نسل‌های مختلفی از توسعه‌دهندگان، تصور مدل داده‌ای بدون سطر و ستون، امری غیرممکن به نظر می‌رسد؛ بااین‌حال، کارایی پایین این مدل سنتی در داده‌های عظیم تجاری و برنامه‌های بزرگ تحت وب بسیار مورد هجمه واقع‌شده و بسیاری را به فکر ارائه راه‌حل‌های جایگزین انداخت. بااینکه در عصر اینترنت و داده‌های حجیم، یک پایگاه داده قوی و غنی سنتی کاربردی است، اما این مدل که برای برنامه‌های ساده در دهه ۱۹۵۰ طراحی‌شده، پاسخ‌گوی نیازهای امروزی نیست. جهت پاسخگویی به نیازهای امروزی، پایگاه‌های داده‌های جدیدی با عنوان NoSQL به وجود آمدند. کارلو استروزی

1 -Edgar.F. Codd
2 -Object Oriented Database
3 -Real-Time Database Management System
4 -Fault Tolerant Database Management System
5 -Secure Database Management System
6 -HTDBMS (Heterogeneous Database Management System)
7 -Multimedia Database Management System
8 -Federated Database Management System
9 -http://www.internetlivestats.com

نخستین بار در سال ۱۹۹۸ عبارت NoSQL را برای اشاره به پایگاه‌های داده‌ای سبک و متن‌باز به کار گرفت که از رابط SQL استفاده نمی‌کردند. بعدها وی به این نکته اشاره کرد که این عبارت و مفهوم پشت آن، از مدل رابطه‌ای جداشده و بهتر است NoREL نامیده شود. در سال ۲۰۰۰ میلادی، اریک بریور با ارائه نظریه CAP به کمبودها و محدودیت‌های مدل رابطه‌ای در سیستم‌های توزیع‌شده اشاره کرد. طبق نظریه CAP ثبات و دسترس‌پذیری بالا، هر دو در یک پایگاه داده در یک شبکه گسترده و وسیع قابل فراهم شدن نیستند [۱۲]. بر طبق این نظریه اهمیت داده‌های گسترده سطح شبکه و مدل‌هایی با تأکید بر جداسازی و دسترس‌پذیری بالا به‌عنوان نیازمندی اصلی و با در نظر گرفتن ثبات به‌عنوان اولویت بعدی که امکان به تأخیر انداختن آن در مقایسه با سایر اولویت‌ها نیز وجود دارد، از اهمیت بالایی برخوردارند. به همین دلیل، عبارت NoSQL مفهومی برای مشخص کردن پایگاه‌های داده‌هایی است که به‌شدت با پایگاه‌های داده رابطه‌ای سنتی متفاوت هستند. این پایگاه‌های داده‌ای اغلب با مفاهیم سنتی نظیر جدول‌ها، سطر و ستون‌های ثابت بیگانه است. در اغلب موارد، عملیات Join در آن‌ها بی‌معنی بوده و به‌صورت افقی مقیاس‌پذیر هستند. در سال ۲۰۱۲ میلادی، کار روی پروژه‌ای بانام UnQL آغاز شد که هدف آن، تدوین استاندارد زبان پرس‌وجو در پایگاه‌های داده‌ای NoSQL بود. این زبان به‌جای جدول‌های سنتی و سطر و ستون‌های داده‌ای، به ترتیب مجموعه‌ها، اسناد و فیلدهای مشخصی را مورد پرس‌وجو قرار می‌دهد. این زبان مجموعه‌ای ورای SQL به شمار آمده و NoSQL را می‌توان نمونه بسیار محدودشده آن به شمار آورد. بااین‌حال، زبان UnQL عبارت‌های DDL را پوشش نمی‌دهد [۱۳]. با توجه به رشد استفاده کاربران از اینترنت و تنوع در تولید داده‌ها همچنین افزایش حجم اطلاعات، نیاز به روش‌های نوینی جهت ذخیره‌سازی و پردازش این حجم از اطلاعات وجود دارد. جهت رفع مشکلات ناشی از افزایش حجم اطلاعات از بانک‌های اطلاعاتی توزیع‌شده استفاده می‌شود. لذا جهت ایجاد برنامه‌هایی که پاسخگوی این حجم از اطلاعات باشند باید بانک‌های اطلاعاتی توزیع‌شده به‌خوبی شناخته‌شده و ساختار و نمونه‌های آن نیز مورد ارزیابی قرار گیرند تا بتوان بانک اطلاعاتی را به‌گونه‌ای انتخاب و طراحی نمود که بتواند پاسخگوی نیازهای برنامه باشند. در این مقاله ابتدا در بخش ۲ انواع ساختار بانک اطلاعاتی NoSQL که به چهار دسته کلید-مقدار، سندگرا، ستون گسترده و مبتنی بر گراف تقسیم می‌شوند با مقایسه ساختار ذخیره‌سازی آن‌ها با بانک‌های رابطه‌ای همراه با نحوه گسترش و کاربرد هرکدام آمده است. در ادامه بانک‌های اطلاعاتی چند مدلی که از چندین نوع از این بانک‌ها پشتیبانی می‌کنند نیز معرفی شده‌اند. در این مقاله برای هرکدام از این انواع چندین نمونه بانک اطلاعاتی معرفی شده که در بخش ۳ جهت مقایسه کارایی آن‌ها در آزمایشی با داده‌های یکسان از هرکدام یک نمونه انتخاب شده است. درنهایت در قسمت ارزیابی و نتیجه‌گیری بخش ۵ با مقایسه کارایی هرکدام در امکانات و آزمایش به انتخاب بهترین بانک اطلاعاتی با توجه به کاربرد رسیده‌ایم.



شکل ۲- انواع بانک اطلاعاتی توزیع‌شده NoSQL

۲- بانک‌های اطلاعاتی NoSQL

بانک‌های اطلاعاتی توزیع‌شده که بر مبنای داده‌های حجیم کار می‌کنند به چهار دسته تقسیم می‌شوند که در شکل (۲) مشاهده می‌شوند. جهت معرفی ساختار هرکدام از بانک‌های اطلاعاتی و مقایسه آن‌ها با ساختار رابطه‌ای، در این مقاله پایگاه داده یک کتابخانه در شکل (۳) در نظر گرفته می‌شود. ساختارهای بانک‌های اطلاعاتی NoSQL، با این بانک اطلاعاتی مقایسه می‌شوند. در این بانک دو موجودیت دانشجو و کتاب در نظر گرفته‌شده و یک جدول که رابطه بین آن‌ها را نشان می‌دهد.

جدول کتاب

ID	Name	Author	Year
4182	C#	Deitel	2005
4023	Java	Null	Null

جدول واسط دانشجو و کتاب

Student_id	Book_id
2536	4182
2537	4023

جدول دانشجو

ID	Name	Family	Age
2536	Saman	Keshvari	Null
2537	Ali	Null	24

شکل ۳- نحوه طراحی بانک اطلاعاتی کتابخانه در ساختار رابطه‌ای [۱۴]

۲-۱-پایگاه داده کلید- مقدار

در این مدل پایگاه‌های داده، جفت‌های داده‌ای کلید-مقدار به‌صورت جدول درهم‌ریزی[1] توزیع‌شده ذخیره می‌شوند به‌گونه‌ای که سرویس

1- Hash

بازیابی مشابه جدول درهم‌ریزی را فراهم می‌سازند. جهت بازیابی، کافی است کلید موردنظر درخواست شود تا با سرعتِ بالا داده‌های ذخیره‌شده برگشت داده شود. این سرعت بالا به دلیل ساده بودن عملکرد بازیابی اطلاعات در این نوع بانک‌های اطلاعاتی است، جهت بازیابی یک رکورد کافی است که کلید موردنظر به سیستم داده شود تا تنها در یک دسترسی به حافظه مقداری که برای آن کلید ذخیره شده است بازیابی شود [۱۵]. در این نوع پایگاه داده نوع داده‌ای که در قسمت مقدار واردشده است اهمیتی نداشته و سیستم مدیریت، آن مقدار را به برنامه تحویل داده و در برنامه عملیات مورد نیاز جهت نشان دادن آن داده صورت می‌پذیرد.

در پایگاه‌های داده کلید-مقدار، هر ماشین عضو خوشه پایگاه داده موجود در سیستم می‌تواند مقدار مرتبط با یک کلید را به‌صورت بهینه بازیابی کند. مسئولیتِ نگهداری نگاشت کلیدها و مقدارها در میان گره‌های موجود توزیع‌شده است به‌طوری‌که هرگونه تغییر در مجموعه گره‌ها، کم‌ترین میزان قطع شدن خدمات سیستم را به بار خواهد آورد [۱۶]. از کاربردهای معمول آن‌ها، کش کردن محتوا، ارجاعات سریع به داده‌ها، نمایه‌گذاری و ساختار کلید-مقدار است که در مقدار آن می‌تواند فرمت‌های مختلف ازجمله JSon, BSon, Xml را داشت که در ارتباطات برنامه و بانک اطلاعاتی مؤثر است. مزیت این پایگاه‌های داده در مقابل بانک‌های رابطه‌ای، تأخیر کم، توزیع‌پذیری و جایگزینی پذیری بالا است که با استفاده از یک API ساده قابل‌دسترسی هستند. از ضعف‌های این نوع می‌توان عدم داشتن شمای طراحی پایگاه داده را نام برد [۱۶][۱۷]. در جدول (۲) نام و میزان مقبولیت ده بانک اطلاعاتی که ساختار کلید-مقدار دارند مشاهده می‌شوند. در این جدول و جدول‌هایی که در ادامه آمده است، منظور از رتبه کل، رتبه در میان کل پایگاه‌های داده اهم از رابطه‌ای، کلید-مقدار، ستون گسترده و مبتنی بر گراف است.

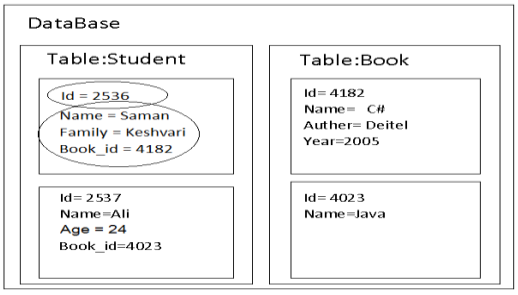
جدول ۲- بانک‌های اطلاعاتی کلید- مقدار [۱۸]

نام	رتبه	رتبه در کل
Redis	۱	۱۰
Memcached	۲	۲۲
Amazon DynamoDB	۳	۲۶
Riak	۴	۳۱
Ehcache	۵	۳۹
Hazelcast	۶	۴۰
OrientDB	۷	۴۱
Aerospike	۸	۵۵
Berkeley DB	۹	۵۹
Oracle Coherence	۱۰	۶۲

۲-۱-۱- نحوه ذخیره‌سازی، مقیاس‌پذیری[1] و سازگاری[2] در بانک‌های اطلاعاتی کلید-مقدار

بانک‌های اطلاعاتی کلید-مقدار به دلیل عملکرد ساده بسیار مقیاس‌پذیر هستند. برخلاف بانک‌های اطلاعاتی رابطه‌ای، بانک‌های کلید-مقدار به‌صورت عمودی رشد نمی‌کنند. این دسته از بانک‌های اطلاعاتی بدون نیاز به طراحی مجدد بین چندین ماشین یا دستگاه گسترش[3] پیدا می‌کنند. مقیاس‌پذیری در این بانک‌های اطلاعاتی دارای هزینه پایین‌تری نسبت به بانک‌های اطلاعاتی رابطه است [۱۹]. جهت گسترش بانک در بانک‌های رابطه‌ای گاهی نیاز به توقف سرور جهت تغییر الگو[4]ی پایگاه داده وجود دارد. یک پایگاه داده کلید-مقدار منحنی هزینه را خطی نگه‌داشته و از بروز منحنی نمایی جهت افزایش هزینه جلوگیری می‌کند، این نتیجه به دلیل طراحی این نوع بانک‌های اطلاعاتی است زیرا به‌گونه‌ای طراحی‌شده‌اند که بدون هیچ‌گونه الگوی از پیش تعیین‌شده‌ای داده‌ها را مدیریت نمایند.

اگر پایگاه داده کتابخانه شکل (۳) فرض شود، در ساختار کلید-مقدار کافی است برای دانشجویان و کتاب‌ها یک کلید در نظر گرفته شود. به‌عنوان‌مثال در شکل (۴) کلید برای دانشجویان و کتاب‌ها یک فیلد (id) نشان داده‌شده است ولی قسمت مقدار آن‌ها با یکدیگر متفاوت است. در این نمونه برای کتاب پارامترهایی متفاوت از دانشجو وجود دارد. حتی در موجودیت دانشجو نیز امکان وجود فیلدهای متفاوتی وجود دارد، به‌عنوان‌مثال برای دانشجوی اول: نام، نام خانوادگی، شناسه کتاب ولی برای دانشجوی دوم: نام، سن و شناسه کتاب در نظر گرفته‌شده درصورتی‌که در سیستم رابطه‌ای در خانه‌های خالی ردیف‌های جدول، مقادیر Null قرار می‌گیرد که این موجب نرمال نبودن بانک اطلاعاتی می‌شود. باید توجه داشت که مقدار کلید به ازای هر موجودیت در بانک اطلاعاتی کلید-مقدار به‌صورت یکتاست [۱۴].



شکل ۴- نحوه طراحی بانک اطلاعاتی کتابخانه در ساختار کلید-مقدار [۱۴]

1 -Scalability
2 -Consistency
3- Scale Over
4 -Schema

۲-۲- مدل داده‌ای سندگرا[1]

بانک‌های اطلاعاتی NoSQL سندگرا، جهت تهیه برنامه‌های مدیریت محتوا مناسب هستند. در این نوع برنامه‌ها، محتواهایی که دارای فراداده‌ای هستند، ذخیره می‌شوند. در بانک‌های اطلاعاتی سندگرا با استفاده از مفهوم برچسب‌ها، امکان اتصال فایل‌های متناظری به اسناد پیش‌بینی‌شده است. عملیات خواندن در این پایگاه‌های داده، با یک سیستم کنترل نسخه انجام می‌شود که به‌موجب آن هر مشتری می‌تواند یک نمونه یکتای کامل از یک سند را در کل عملیات خواندن در اختیار بگیرد. یکپارچگی داده‌ها به‌صورت اسناد در پایگاه داده در مقابل داده‌هایی که در میان تعداد زیادی از جدول‌ها گسترده می‌شوند، مزایای بسیاری دارد؛ در این حالت وقتی اسناد روی دیسک نوشته می‌شوند، فیلدها و فراداده‌های آن‌ها پی‌درپی در بافرها گنجانده می‌شوند [۱۶]، به‌این‌ترتیب امکان ایجاد دید فراهم می‌شود.

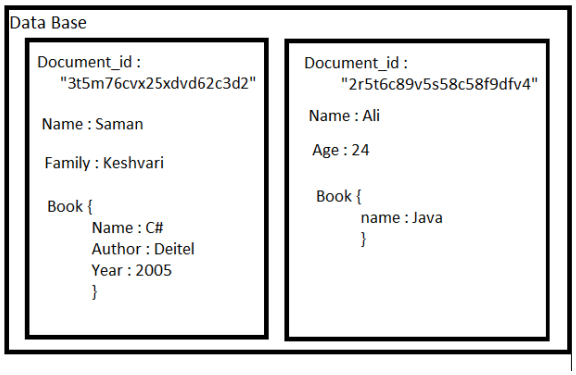
جدول۲- بانک‌های اطلاعاتی سندگرا [۲۰]

نام	رتبه	رتبه در کل
MongoDB	۱	۴
CouchBase	۲	۲۴
CouchDB	۳	۲۵
Amazon DynamoDB	۴	۲۶
MarkLogic	۵	۳۴
OrientDB	۶	۴۱
RavenDB	۷	۴۵
RethinkDB	۸	۴۶
Cloudant	۹	۵۰
GemFire	۱۰	۵۸

در بانک‌های اطلاعاتی سندگرا هر سند را می‌توان با یک URI آدرس‌دهی کرد و اغلب از جاوا اسکریپت[2] به‌عنوان زبان بومی[3] خود جهت سهولت کار با اشیاء JSON بهره می‌برند. در جدول (۲)، نام و میزان مقبولیت ده بانک اطلاعاتی که ساختار سندگرا دارند مشاهده می‌شوند.

۲-۲-۱- نحوه ذخیره‌سازی بانک‌های سندگرا

بانک‌های اطلاعاتی سندگرا در مقایسه با بانک‌های رابطه‌ای به‌جای ردیف‌ها سند دارند. ساختار سندها نیز عموماً بر مبنای اشیاء JSON تعریف می‌گردد، بنابراین هر سند دارای تعدادی خاصیت است چون اشیاء JSON به این نحو تعریف می‌شوند. این نوع بانک‌های اطلاعاتی شبیه به بانک‌های کلید- مقدار به نظر می‌رسند اما در حین تعریف اشیاء JSON، ممکن است یک مقدار صرفاً یک مقدار ساده نبوده و گاهی شامل یک شیء کامل دیگر است؛ به همین جهت عده‌ای به این نوع بانک‌های اطلاعاتی، بانک‌های اطلاعاتی کلید- مقدار سفارشی و خاص می‌گویند [۲۱]. این نوع ساختار منعطف، برای ذخیره‌سازی اطلاعات اشیاء تودرتو و درختی مناسب است. در بانک‌های اطلاعاتی سندگرا، اسناد می‌توانند حاوی پیوست‌هایی مانند پیوست یک فایل به یک سند باشند [۲۲]. بانک اطلاعاتی سندگرا شکل (۳) را می‌توان مانند شکل (۵) پیاده‌سازی کرد. همان‌طور که در این شکل مشاهده می‌شود برای کتابی که شخص از کتابخانه گرفته است می‌توان به‌صورت تودرتو ویژگی‌های آن را ذخیره نمود. در ساختار سندگرا برای هر موجودیت -در اینجا فرد- یک شناسه سند حتماً باید در نظر گرفته شود که با توجه به آن می‌توان به آن سند دسترسی داشت. در اینجا نیز هیچ‌گونه مقدار Null وجود ندارد یعنی برخلاف سیستم رابطه‌ای الزامی نیست که به دلیل مدنظر قرار داده نام خانوادگی برای دانشجوی اول، برای دانشجوی دوم چنین فیلدی در نظر گرفته شود. نکته‌ای که باید به آن توجه کرد این است که مقدار شناسه سند به‌صورت یکتاست.



شکل ۵- نحوه ذخیره‌سازی بانک‌های سندگرا

۲-۳- مدل داده‌ای ستون‌گسترده۴ (شبه رابطه‌ای، ستون- گرا)

این نوع بانک‌های اطلاعاتی زمانی که داده‌هایی با توالی خواندن بالا و نوشتن کم مدنظر است بهترین کارایی را از خود نشان می‌دهند یعنی سرعت بازیابی در آن‌ها بالاست. این نوع از بانک‌های اطلاعاتی توزیع‌شدگی را در ذخیره‌سازی به‌خوبی در سیستم فایل‌های رایج انجام می‌دهند که این ویژگی‌ها باعث شده در سایت‌های بزرگ اینترنتی که با حجم عظیمی از داده‌ها سروکار دارند استفاده شوند و به‌خوبی جوابگوی کار آن‌ها باشند.

1 -Document-Oriented
2 -JavaScript
3 -native language
4- Wide-Column

جدول ۳- بانک‌های اطلاعاتی ستون‌گسترده [۲۴]

نام	رتبه	رتبه در کل
Cassandra	۱	۸
HBase	۲	۱۶
Accumulo	۳	۶۱
Hypertable	۴	۱۳۴
Sqrrl	۵	۱۶۹
Google Cloud BigTable	۶	۲۱۲
ScyllaDB	۷	۱۶۹

این بانک‌های اطلاعاتی با سیستم فایل‌های معمولی به‌خوبی ارتباط برقرار کرده و زمانی که نیاز به نمایه‌گذاری دستی است، به‌خوبی می‌توانند جوابگو باشند [۲۳]. نقطه‌ضعف این نوع بانک‌های اطلاعاتی استفاده از Apiهای سطح پایین است که توسعه‌دهندگان نیازمند ابزارهای کمکی برای ارتباط با این نوع بانک‌ها هستند. در جدول۳ نام و میزان مقبولیت هفت بانک اطلاعاتی که از ساختار ستون‌گسترده پیروی می‌کنند مشاهده می‌شوند.

۲-۳-۱- مقایسه ذخیره‌سازی سطری با ستون گرا

در پایگاه داده ستون‌گسترده بحث از ذخیره‌سازی ستون گرا مطرح است، در جدول (۴) ساختار جدولی یک بانک اطلاعاتی نمونه و در شکل (۶) ساختار ستون‌گرای آن بانک مشاهده می‌شود.

جدول ۴- ساختار جدولی بانک فرض شده

id	Name	Family	Salary
8941	Saman	Alimi	5600
8942	Ali	Razavi	NULL
8943	payam	NULL	5500
8944	Ahmad	Amini	NULL

id	Family
8941	Alimi
8942	Razavi
8944	Amini

id	Name
8941	Saman
8942	Ali
8943	payam
8944	Ahmad

id	Salary
8941	5600
8943	5500

شکل ۶- ساختار ستون‌گرای بانک فرض شده

جدول (۵) مقایسه بین کارایی بانک‌های اطلاعاتی با ذخیره‌سازی ستونی را با ذخیره‌سازی سطری نشان می‌دهد.

جدول ۵- مقایسه بانک‌های اطلاعاتی سطری و ستونی

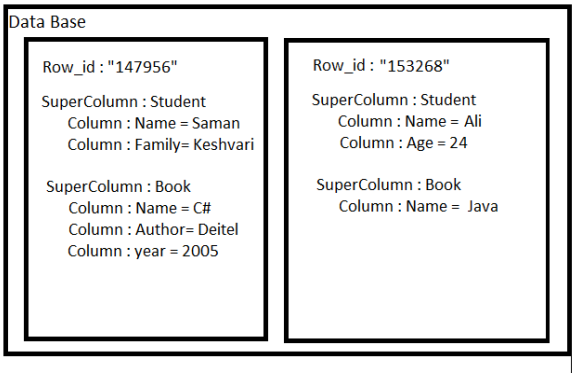
عملیات	سطری	ستونی
محاسبه مجموع مقادیر یک ستون	کند	سریع
فشرده‌سازی	-	بالا [۱]
بازیابی چند ستون در میان ستون‌های زیاد	کند [۲]	سریع
ورود/به‌روزرسانی یک رکورد جدید	سریع	کند
بازیابی یک رکورد	سریع	کند

۱- زمانی که داده‌هایی از یک نوع ذخیره شوند.

۲- به دلیل جستجو در میان داده‌های غیرضروری

۲-۳-۲- نحوه ذخیره‌سازی بانک‌های ستون‌گسترده

بانک‌های اطلاعاتی ستون‌گسترده دارای جدول‌هایی هستند که درون آن‌ها ستون‌هایی قابل‌تعریف است، درون این ستون‌ها که مانند بانک‌های اطلاعاتی رابطه‌ای هستند، اطلاعات به شکل کلید- مقدار با ساختاری متفاوت، قابل ذخیره‌سازی هستند. هر ستون شامل گروهی از ستون‌ها (ستون‌های چندوجهی) که بر اساس مفاهیم جفت‌های کلید- مقدار کار می‌کنند، است. در این نوع از بانک‌های اطلاعاتی نیز جداول و ردیف‌ها وجود دارند و هر ستون عضوی از خانواده یک ابر ستون است [۱۶]. در شکل (۷) ساختار ستون گسترده بانک اطلاعاتی رابطه‌ای شکل (۳) آمده است.



شکل ۷- شمای ذخیره‌سازی ستون‌گسترده

۲-۴- مدل داده‌ای مبتنی بر گراف

این نوع پایگاه داده جهت ردیابی ارتباطات بین اطلاعات (همچون شبکه‌های اجتماعی) طراحی‌شده‌اند.

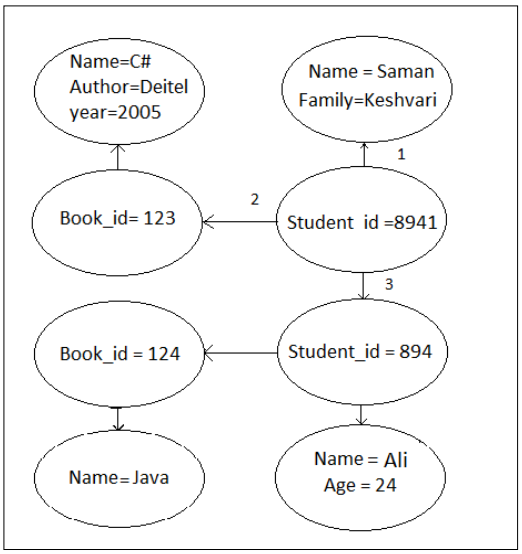
۲-۴-۱- نحوه ذخیره‌سازی بانک‌های مبتنی بر گراف

همان‌طور که در بخش قبل توضیح داده شد این نوع از بانک‌های اطلاعاتی از گره به‌عنوان موجودیت و یال برای ارتباط استفاده می‌کنند. مثال شکل (۳) برای بانک‌های اطلاعاتی مبتنی بر گراف همانند شکل (۸) طراحی می‌شود. در این شکل گره‌ها معرف موجودیت دانشجو و کتاب هستند و یال‌ها برای ارتباط بین آن‌ها استفاده‌شده است. به‌عنوان نمونه یال ۱ برای معرفی کردن

ویژگی‌های دانشجوی با شناسه ۸۹۴۱ است و ارتباط (یال) ۲ معرف قرض گرفتن کتاب با شناسه ۱۲۳ توسط این دانشجو است. به‌عنوان‌مثال در ارتباط ۳ در بانک اطلاعاتی همکلاس بودن دانشجویان نیز مدنظر بوده، می‌توان آن را نیز به‌صورت یال در نظر گرفت. در اینجا یال سه به‌عنوان ارتباط همکلاس بودن در نظر گرفته‌شده است. سایر ارتباطات در پایگاه‌های داده مبتنی بر گراف با یال نشان داده می‌شوند.

جدول ۶- بانک‌های اطلاعاتی گراف محور [۲۵]

نام	رتبه	رتبه در کل
Neo4j	۱	۲۱
OrientDB	۲	۴۱
Titan	۳	۴۴
Viruoso	۴	۶۴
ArangoDB	۵	۹۵
Giragh	۶	۱۱۶
AllegroGraph	۷	۱۳۳
Stardog	۸	۱۴۹
Sqrrl	۹	۱۶۹
InfiniteGraph	۱۰	۱۷۲



شکل ۸- نحوه ذخیره‌سازی بانک‌های گراف محور

۲-۵- بانک‌های اطلاعاتی چند مدلی

همزیستی مسالمت‌آمیز[1] این بحث را مطرح می‌کند که لازم نیست در یک برنامه یا وب‌سایت تنها یک پایگاه داده برای ذخیره‌سازی و مدیریت اطلاعات استفاده شود. با توجه به کارایی هر بخش برای رسیدن به بهترین عملکرد برای آن بخش، می‌توان از بهترین گزینه برای پایگاه داده به‌صورت هم‌زمان در کنار بانک‌های اطلاعاتی سایر بخش‌ها استفاده نمود. به‌عنوان مثال جهت ذخیره‌سازی داده‌های بار ارزش و کم‌حجم مانند پروفایل کاربران یا صورت‌حساب آن‌ها، بهتر است از پایگاه داده رابطه‌ای مانند Mysql استفاده شود. جهت ذخیره‌سازی داده‌های حجیم مانند ثبت صفحات از بانک‌های اطلاعاتی سندگرا مانند MongoDB استفاده شود. در صورت فراهم بودن سخت‌افزار کافی بهتر است از رویکرد همزیستی مسالمت‌آمیز استفاده نمود [۳۴]. در صورت عدم توانایی در استفاده از چندین بانک در کنار یکدیگر، می‌توان از بانک‌های اطلاعاتی چند مدلی استفاده کرد [۲۶]. بانک‌های اطلاعات چند مدلی از چند نوع بانک اطلاعاتی که در قسمت‌های قبل بررسی شدند -با توجه به ساختاری که دارند- پشتیبانی می‌کنند. مثلاً با توجه به جدول‌های (۱-۳ و ۶) نام چند بانک اطلاعاتی در آن‌ها تکرار شده است که این نشان از پشتیبانی این نوع بانک‌ها از چند نوع از بانک‌های اطلاعاتی NoSQL است. انواع این بانک‌های اطلاعاتی و ساختارهایی که پشتیبانی می‌کنند در جدول (۷) آمده است.

جدول ۷- بانک‌های اطلاعاتی چند مدلی

نام	رابطه‌ای	کلید-مقدار	سندگرا	ستون‌گسترده	مبتنی بر گراف
Amazon DynamoDB	×	✓	✓	×	×
OrientDB	×	✓	✓	×	✓
Virtuoso	✓	×	✓	×	✓
ArangoDB	×	✓	✓	×	✓
Sqrrl	×	✓	✓	✓	✓
GlobalsDB	×	✓	✓	×	✓
AlaSQL	✓	×	✓	×	×
Amisa Server	✓	×	✓	×	✓
CortexDB	×	✓	✓	×	×

۳- مقایسه و ارزیابی

تا اینجا بانک‌های اطلاعاتی توزیع‌شده‌ای که در ذخیره‌سازی داده‌های حجیم کاربرد دارند بررسی شدند. جهت مقایسه این نوع بانک‌های اطلاعاتی از هرکدام یک نوع انتخاب شده و آن‌ها با معیارهای مختلف مقایسه شده‌اند. جهت انتخاب نمونه از هر نوع معیارهای مختلفی مد نظر قرار داده شده که در توضیح هرکدام آمده است. بانک اطلاعاتی membase که یک نمونه موفق در پیاده‌سازی بانک اطلاعاتی کلید-مقدار است، به‌عنوان نمونه بانک اطلاعاتی کلید-مقدار در نظر گرفته می‌شود. بانک اطلاعاتی MongoDB که با زبان C++ در سال ۲۰۰۹ ایجادشده است در دسته بانک‌های اطلاعاتی سندگرا قرار دارد به‌عنوان نمونه بانک اطلاعاتی سندگرا در نظر گرفته می‌شود [۲۷]. این بانک اطلاعاتی محبوب‌ترین، پرکاربردترین و معروف‌ترین بانک اطلاعاتی NoSQL است. بانک اطلاعاتی Cassandra که با زبان جاوا پیاده‌سازی شده و یکی از بانک‌های معروف مورداستفاده شرکت‌هایی

1- Polyglot persistence

نظیر ناسا، فیس‌بوک، IBM و سیسکو است، به‌عنوان بانک اطلاعاتی ستون‌گسترده انتخاب می‌شود [۲۸]. پایگاه داده Hbase مورد دیگر برای انتخاب بانک‌های اطلاعاتی ستون‌گسترده بود، ولی به دلیل نیازمند بودن این بانک اطلاعاتی به امکانات سخت‌افزاری بالا و عدم کارایی آن در داده‌های غیر دسته‌ای، این بانک اطلاعاتی انتخاب نشد [۲۹].

برای بانک‌های اطلاعاتی مبتنی بر گراف Neo4j که در سال ۲۰۰۷ با زبان جاوا به وجود آمده است، به دلیل محبوبیت بالای این بانک اطلاعاتی در نزد توسعه‌دهندگان، انتخاب می‌شود [۳۰].

بانک اطلاعاتی OrientDB که در سال ۲۰۱۰ با زبان جاوا نوشته‌شده و از ساختارهای کلید-مقدار، سندگرا و مبتنی بر گراف پشتیبانی می‌کند به‌عنوان نمونه بانک اطلاعاتی چند مدلی در نظر گرفته می‌شود [۳۱]. OrientDB در مقایسه‌ای که انجام شده [۲۶]، بهترین عملکرد را در بین بانک‌های اطلاعاتی چند مدلی را از خود نشان داده است.

جهت کار با بانک‌های اطلاعاتی، استفاده از APIهای مختلف می‌تواند به توسعه‌دهندگان کمک شایانی بکند. REST[1] متکی بر یک پروتکل ارتباطی بدون حالت، مشتری-خدمتگذار[2] و باقابلیت نهان‌سازی است که در اکثر موارد در پروتکل HTTP مورداستفاده قرار می‌گیرد.

با توجه به اهمیت زبان جاوا، API های این زبان -که استفاده زیادی در ارتباط با بانک‌های اطلاعاتی NoSQL دارد- در اکثر بانک‌ها دیده‌شده است. برخی از بانک‌های اطلاعاتی زبان پرس‌وجوی اختصاصی جهت پاسخ به درخواست‌ها دارند. روش نگاشت-کاهش که توسط گوگل ابداع‌شده است برای تقسیم بار پردازشی در بین سرورهای مختلف به کار می‌رود در برخی بانک‌های اطلاعاتی به‌عنوان یک امکان مناسب وجود دارد. امکانات پرس‌وجوی موجود در بانک‌های اطلاعاتی در جدول (۸) آمده است.

با توجه به اینکه بانک‌های اطلاعاتی بررسی‌شده در این مقاله به‌صورت توزیع‌شده هستند هنگامی چندین درخواست به‌صورت موازی به این بانک‌های اطلاعاتی وارد شود روش کنترل هم‌روندی از مسائل مهم در این نوع بانک‌های اطلاعاتی است که باید بررسی شود. کنترل هم‌زمانی چند نسخه‌ای (MVCC[4]) اولویت را به کارایی می‌دهد؛ در این روش دسترسی هم‌زمانی با قفل‌گذاری مدیریت نشده، بلکه با چندین نسخه با برچسب زمانی غیرقابل تغییر سازمان‌دهی می‌شود [۲۲]. تا زمانی که مجموعه داده در اختیار یک دسترسی اختصاصی قرار گیرد، درخواست‌های خواندنی با ارائه آخرین نسخه پاسخ داده می‌شوند، این در حالی است که در زمان نوشتن هم‌زمان به‌منظور مقابله با نوشتن متضاد هر پردازه علاوه بر ذخیره‌سازی مقدار جدید، اتصالی به مقداری که قبلاً خوانده‌شده ذخیره می‌شود؛ بنابراین، الگوریتم اجراشده در پایگاه داده یا مشتری، امکان رفع مسئله تصادم را با روش‌های مختلف وجود دارد. در این روش به دلیل ذخیره‌سازی چندین نسخه از داده به فضای بیشتری نیاز دارد، علاوه بر این به دلیل روش‌های مدیریت و جمع‌آوری زباله‌ها[5] که راه‌حلی برای مسئله بی‌ثباتی است، منجر به پیچیدگی روش MVCC شده است [۲۲].

جدول ۸- مقایسه امکانات مختلف بانک‌های اطلاعاتی NoSQL نمونه

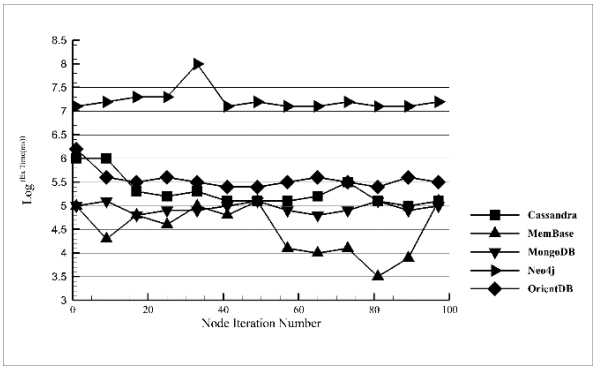
بانک اطلاعاتی	نوع	امکانات پرس و جوی				کنترل هم‌روندی			قطعه‌بندی		نسخه‌برداری		
		Rest API	Java API	زبان پرس و جوی	نگاشت-کاهش	قفل‌گذاری	قفل خوش‌بینانه	MVCC	بر اساس محدوده	درهم‌سازی سازگار	خواندن از کپی	نوشتن در کپی	ثبات
Membase	کلید-مقدار	✓	✓	×	×	×	✓	×	×	✓	×	×	✓
MongoDB	سندگرا	✓	✓	×	✓	×	×	×	✓	×	✓	×	×/✓
Cassandra	ستون‌گسترده	×	✓	✓	✓	×	×	×	×	✓	✓	×	×/✓
Neo4j	مبتنی بر گراف	✓	✓	✓	×	✓	×	×	×	×	✓	✓	×
OrientDB	چند مدلی	✓	✓	✓	×	✓	×	×	×	✓	✓	✓	×

پشتیبانی از تراکنش‌ها به‌منظور دسترسی انحصاری بدون رزرو پایگاه داده چندگانه، قفل خوش‌بینانه[3] با کمک ذخیره‌سازی چندگانه فراهم می‌شود. قبل از تثبیت داده‌های تغییریافته، هر تراکنش با تراکنش‌هایی که به‌صورت موازی با تراکنش موردنظر بر روی مجموعه داده مشخص تغییرات ایجاد می‌کنند، ممکن است این کار منجر به بروز تصادم شود که در صورت رخ دادن آن، تراکنش به حالت اولیه بازمی‌گردد. این روش برای مجموعه تراکنش‌هایی که در آن‌ها امکان بروز تصادم کم است، به‌خوبی عمل می‌کند. در این روش

1- Representational State Transfe
2- Client-Server
3- MapReduce
4- Multiversion concurrency control
5- Garbages

بررسی و بازگشت دارای هزینه کمتری نسبت به قفل‌گذاری مجموعه داده به‌منظور دسترسی اختصاصی است.

با افزایش حجم ذخیره و بازیابی اطلاعات یک سرور پاسخگوی این حجم از اطلاعات نیست، لذا باید مجموعه داده در بخش‌های مختلف در میان خوشه‌ها تقسیم شوند. بانک‌های اطلاعاتی NoSQL با توجه به مدل ذخیره‌سازی با دو روش قطعه‌بندی را انجام می‌دهند. با توجه به اینکه مدل‌های داده‌ای کلید-مقدار، ستون‌گسترده و سندگرا کلید محور هستند مکانیسم قطعه‌بندی آن‌ها بر اساس کلید است که داده‌های در محدوده مشخص در سرورهای مشخص قرار می‌گیرند و هر گره موجود در خوشه مسئول پاسخگویی به درخواست‌های در محدوده خود است که این محدوده‌ها توسط سرور مسیریابی سازمان‌دهی می‌شوند. حساسیت سرور مسیریابی موجب می‌شود که اطلاعات آن در چندین سرور تکرار شود که اگر یکی از آن‌ها در دسترس نبود، سرور جایگزین جهت پاسخگویی قرار گیرد. روش دیگری جهت قطعه‌بندی وجود دارد که کلیدها و اطلاعات با کمک تابع درهم‌سازی سازگار[1] توزیع می‌شوند [۲۲]. در این روش اشتراک‌گذاری معماری خاصی وجود ندارد و هر سرور به‌منظور پاسخگویی به محدوده درهم‌سازی خاصی در نظر گرفته می‌شود.



شکل ۹- مقایسه زمان اجرای عملیات درج گره در درخت [۳۳]

آدرس یک کلید مشخص بدون نیاز به خوشه‌بندی به تابع درهم‌سازی داده‌شده که منجر به سریع‌تر شدن محاسبه مکان داده می‌شود. درهم‌سازی سازگار منجر به توزیع بار به‌صورت تصادفی در بین سرورها می‌شود، تابع درهم‌سازی که کلیدها را به‌صورت متوازن در بین سرورهای مختلف توزیع کند، مناسب است.

علاوه بر توازن بار جهت افزایش کارایی، تکرار داده‌ها در بین سرورهای مختلف در سیستم‌های توزیع‌شده منجر به افزایش دسترسی‌پذیری و ماندگاری می‌شود. به گفته بریور نمی‌توان به‌صورت هم‌زمان ثبات و دسترسی‌پذیری را داشت. اگر تمام کپی‌ها در سرورهای مختلف به‌صورت هم‌زمان به‌روز شوند سیستم تا زمانی که کپی‌ها در تمام سرورها نوشته شود در دسترس نیست. اگر به دلیل خطای شبکه پیام از دست برود سیستم در مدت‌زمان بیشتری نمی‌تواند در دسترس قرار بگیرد. برای بسترهایی که در آن‌ها دسترس‌پذیری بالا مدنظر است، روش کپی هم‌زمان روش مناسبی نیست زیرا زمان‌های کوتاه در حد میلی‌ثانیه در آن‌ها از اهمیت بالایی برخوردار است. افزایش دسترسی‌پذیری کاهش ثبات داده را به دنبال دارد. اگر تکرار داده‌ها به‌صورت غیرهمزمان صورت پذیرد، خواندن اطلاعات برای مدت‌زمان محدودی بی‌ثبات است. این سیستم‌ها با عنوان سیستم‌هایی که از نظریه BASE[2] که توسط بریور مطرح‌شده است پیروی می‌کنند، مطرح هستند. پس اگر نیاز به ثبات داده، درواقع دقت بالابود باید از بانک‌های اطلاعاتی که از نظریه ACID پیروی می‌کنند استفاده شود و اگر نیاز به‌سرعت بالا در دسترسی به اطلاعات بود از بانک‌های اطلاعاتی که از نظریه BASE پیروی می‌کنند باید استفاده شود [۳۲].

به‌منظور بررسی کارایی بانک‌های اطلاعاتی که در جدول (۸) بررسی‌شده‌اند در محیط واقعی یک آزمایش طراحی‌شده است که در آن جهت ارزیابی کارایی چندین عملیات کلیدی که شامل درج گره، بازیابی ویژگی، بازیابی ویژگی-مقدار در یک درخت انجام شده که شامل چهار آزمایش است [۳۳]:

۱) درج گره در یک درخت که درمجموع دارای ۱۰۰،۰۰۰ گره است به این صورت که هر گره دارای ۱۰ گره فرزند است که تا عمق ۵ پیش رفته است. مقادیر ویژگی ۱۰ گره فرزند شامل ترکیبی از اعداد شناور، صحیح، رشته و بولی است که از یک مخزن داده انتخاب می‌شوند. شکل (۹) مقایسه زمان اجرای آن در بانک‌های اطلاعاتی را نشان می‌دهد.

۲) زمان بازیابی زیر درخت که در سطح ۲ درخت قرار دارند به‌گونه‌ای که درمجموع شامل ۱۰۰۰ گره است. شکل (۱۰) مقایسه زمان اجرای این عملیات را در بانک‌های اطلاعاتی مختلف نشان می‌دهد.

۳) میانگین زمان اجرای ۱۰۰ پرس‌وجو که در آن ویژگی‌های مشخص از قبل تعیین‌شده برای بانک‌های اطلاعاتی محاسبه‌شده که در شکل (۱۱) مشخص است.

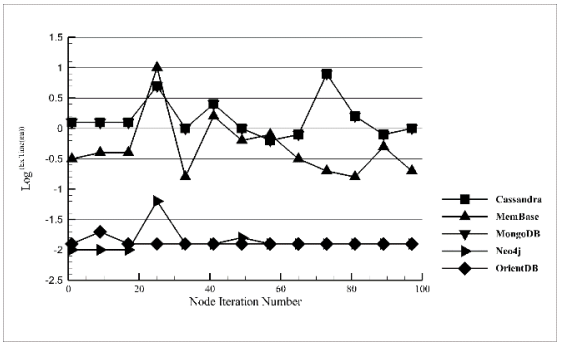
۴) میانگین زمان اجرای ۱۰۰ پرس‌وجو که شامل "ویژگی : مقدار" است که در شکل (۱۲) مشخص است.

در عملیات ساخت گره Membase کمترین تأخیر را داراست که بهترین عملکرد را در بین بانک‌های اطلاعاتی دارد. در عوض برای

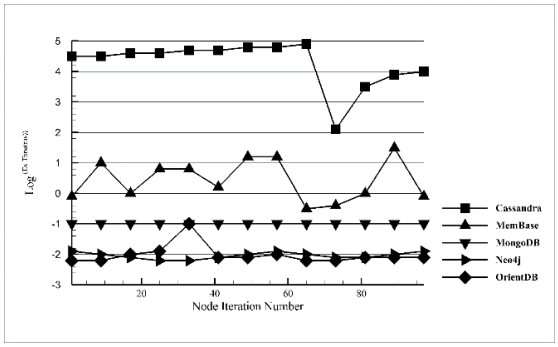
1 -Consistent Hashing

2 -Basically Available, Soft-State, Eventually Consistent

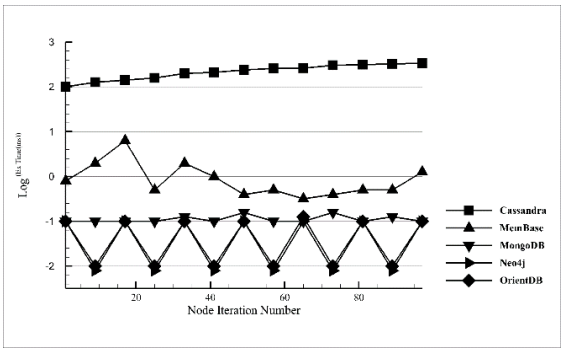
عملیات بازیابی بانک اطلاعاتی Neo4j که نوعی بانک مبتنی بر گراف است بهترین عملکرد را داراست. نکته قابل‌توجه اینکه بانک اطلاعاتی چند مدلی OrientDB که با مراجعه به جدول ۷ از بانک اطلاعاتی مبتنی بر گراف و کلید-مقدار پشتیبانی می‌کند، نیز دارای عملکرد خوبی در درج و بازیابی است.



شکل۱۰- مقایسه زمان اجرای عملیات بازیابی زیر درخت [۳۳]



شکل ۱۱- مقایسه زمان اجرای پرس‌وجوی نام ویژگی [۳۳]



شکل ۱۲- مقایسه زمان اجرای پرس‌وجوی ویژگی : مقدار [۳۳]

۴- نتیجه‌گیری

در این مقاله انواع بانک‌های اطلاعاتی NoSQL شامل انواع کلید-مقدار، سندگرا، ستون گسترده و مبتنی بر گراف و بانک‌های اطلاعاتی چند مدلی بررسی و هریک از انواع بانک‌های اطلاعاتی یک نمونه جهت مقایسه ویژگی و کارایی، با توجه به کاربرد انتخاب شد. این بانک‌های انتخاب‌شده ازنظر مشخصاتی همچون قطعه‌بندی، نسخه‌برداری داده‌ها و کنترل هم‌روندی مقایسه شدند. بانک‌های اطلاعاتی انتخاب‌شده بر اساس سرعت ذخیره و بازیابی اطلاعات مورد ارزیابی قرار گرفتند. نتایج این ارزیابی نشان می‌دهد که در استفاده از بانک‌های اطلاعاتی هرگاه نیاز به نهان‌سازی محتوا، ارجاعات سریع به حافظه و نمایه‌گذاری بود، انتخاب بانک اطلاعاتی کلید-مقدار با توجه به تأخیر اندک در عملیات ساخت، مناسب خواهد بود. در آزمایش انجام‌شده، در بانک اطلاعاتی Membase به‌عنوان بانک اطلاعاتی کلید-مقدار، عملیات ساخت با تأخیر کمی همراه بوده که نشان‌دهنده درستی این ادعا است. در کاربردهای مربوط به مدیریت محتوا که مبتنی بر سند و فراداده هستند، استفاده از بانک اطلاعاتی سندگرا مانند MongoDB که در آن امکان آدرس‌دهی سندها با URI وجود دارد، بسیار مناسب است. طبق نتایج ارائه‌شده، با توجه به امکان خواندن از نسخه‌های پشتیبان که جهت پشتیبانی از داده‌های اصلی در نظر گرفته شده است، سرعت پاسخگویی به درخواست‌های خواندن بسیار مناسب‌تر است. از دیگر نتایجی که در این بررسی حاصل‌شده این است که علی‌رغم مناسب بودن بانک‌های اطلاعاتی ستون‌گسترده در برنامه‌های مبتنی بر داده‌های حجیم، در کارهای غیر دسته‌ای، استفاده از این نوع بانک‌ها مناسب نخواهد بود. در نتایج نیز نشان داده شد که Cassandra به‌عنوان یک بانک اطلاعاتی ستون‌گسترده، سرعت پاسخگویی پایینی به درخواست‌های غیر دسته‌ای دارد. در برنامه‌هایی مبتنی بر شبکه‌های اجتماعی که نیازمند کاوش در روابط بین موجودیت‌ها است، استفاده از بانک‌های اطلاعاتی مبتنی بر گراف کارایی را بهبود می‌بخشد. طبق نتایج آزمایش Neo4j به‌عنوان بانک اطلاعاتی مبتنی بر گراف، دارای سرعت خوبی در پاسخگویی به درخواست‌هایی است که بین موجودیت‌های آن رابطه وجود دارد. در صورت نیاز به چندین رویکرد هم‌زمان بانک‌های اطلاعاتی NoSQL استفاده از بانک‌های اطلاعاتی چند مدلی کارگشا است. البته باید در نظر گرفت که هرچند این بانک‌ها چندین نوع بانک را پشتیبانی می‌کنند ولی انتظار کامل در همه ابعاد موردنظر را نمی‌توان از آن‌ها داشت؛ بنابراین در صورت فراهم بودن سخت‌افزار کافی بهتر است از رویکرد همزیستی مسالمت‌آمیز استفاده نمود. در این رویکرد با توجه به خصایص داده‌ها مانند حجم، دسته‌ای یا غیر دسته‌ای بودن، شبکه محور بودن اطلاعات از بانک‌های مذکور که توانایی لازم را بنا به مورد استفاده دارند، در کنار یکدیگر به کار برد.

۵- منابع

1. A. Silberschatz, S. Sudarshan, and H. F. Korth, “Database System Concepts,” Sixth Edition, McGraw-Hill, 2010.

2. H. Ishikawa, “Object-Oriented Database System Design and Implementation for Advanced Applications,” Computer Science Workbench Series, Springer Japan, 2005.
3. V. Srinivasan, B. Bulkowski, W. Chu, S. Sayyaparaju, A. Gooding, R. Iyer, A. Shinde, and T. Lopatic, “Aerospike: architecture of a real-time operational DBMS. Proc. VLDB Endow.,” vol. 9, no. 13, pp. 1389-1400, 2016.
4. J. M. Crump and M. E. Tirado, “System and method for aggregating query results in a fault-tolerant database management system,” Google Patents, 2016. https://www.google.com/patents/US20160246857.
5. B. Thuraisingham, “Multilevel Secure Database Management System,” US, Springer, 2009.
6. M. Davidson, E. Hom, and R. Parks, “Heterogeneous database management system,” Google Patents, 2011. https://www.google.com/patents/US8060470.
7. Zh. Guo, Zh. Zhang, E. P. Xing, and C. Faloutsos, “Multimodal Data Mining in a Multimedia Database Based on Structured Max Margin Learning,” ACM Transactions on Knowledge Discovery from Data, vol. 10, no. 3, Article 23, 2016.
8. C. Coronel and S. Morris, “Database Systems Design, Implementation, and management,” Computers Ser., United States of America, 12 Edition. 2016.
9. We are Social, “Digital in 2016 report,” [Online], Available: http://wearesocial.com/uk/special-reports/digital-in-2016. Company number 06629464, London, [Accessed: 27 February 2016.[
10. L. Tay, “Inside eBay’s 90PB data warehouse,” [Online], Available: http://www.itnews.com.au/news/inside-ebay8217s-90pb-data-warehouse-342615, Accessed: 27 February 2016.
۱۱. کشوری، سامان، نقوی، مهدی. کشوری، ساناز، معرفی، بررسی ومقایسه سیستم فایلهای توزیع شده، اولین همایش ملی فناوری های نوین رایانه و توسعه پایدار، دانشگاه شهید بهشتی، تهران، ۱۳۹۴.
12. E. A. Brewer, “Towards Robust Distributed Systems,” Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing, Portland, Oregon, USA, P. 7, 2000.
13. S. Kempe, “UnQL: A Standardized Query Language for NoSQL Databases,” [Online], Available: http://www.dataversity.net/unql-a-standardized-query-language-for-nosql-databases/, (2012, March 29), [Accessed: 26 February 2016].
۱۴. کشوری، سامان، جوادزاده، محمدعلی، نقوی، مهدی، ارزیابی و مقایسه کارایی پایگاه داده‌های کلید- مقدار با هدف انتخابِ مبتنی بر نیاز، مجله علوم رایانشی، شماره ۶، ۱۳۹۶.
15. X. Yuan, X. Wang, C. Wang, C. Qian, and J. Lin, “Building an Encrypted, Distributed, and Searchable Key-value Store. Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (ASIA CCS '16),” ACM, New York, NY, USA, pp. 547-558, 2016.
16. J. Han, E. Haihong, et al., “Survey on NoSQL database,” Pervasive Computing and Applications (ICPCA), 2011 6th International Conference on Port Elizabeth, IEEE, pp. 363-366, 2011.
17. A. B. Angadi, A. B. Angadi, and K. C. Gull, “Growth of New Databases & Analysis of NOSQL Datastores,” International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, no. 6, 2013.
18. Solid IT gmbh (1), “DB-Engines Ranking of Key-value Stores,” [Online]. Available: http://db-engines.com/en/ranking/key-value+store, [Accessed: 27 February 2016].
19. M. Stonebraker, “SQL databases v. NoSQL databases,” Communications of the ACM,New York, NY, USA, vol. 53, no. 4, pp. 10-11, 2010.
20. solid IT gmbh (2), “DB-Engines Ranking of Document Stores,” [Online]. Available: http://db-engines.com/en/ranking/document+store, [Accessed: 27 February 2016].
21. J. Pokorný, “Database technologies in the world of big data,” Proceedings of the 16th International Conference on Computer Systems and Technologies (CompSysTech '15), Boris Rachev and Angel Smrikarov (Eds.). ACM, New York, NY, USA, pp. 1-12, 2015.
22. R. Hecht and S. Jablonski, “NoSQL evaluation: A use case oriented survey,” International Conference on Cloud and Service Computing (CSC) , Hong Kong, IEEE, pp. 336 – 341, 2011.
23. A. Nowosielski, P. A. Kowalski, and P. Kulczycki, “The column-oriented data store performance considerations,” Federated Conference on Computer Science and Information Systems (FedCSIS), Gdansk, pp. 877-881, 2016.
24. Solid IT gmbh (3), “DB-Engines Ranking of Wide Column Stores,” [Online]. Available: http://db-engines.com/en/ranking/wide+column+store, [Accessed: 27 February 2016].
25. Solid IT gmbh (4), “DB-Engines Ranking of Graph DBMS,” [Online]. Available: http://db-engines.com/en/ranking/graph+dbms, [Accessed: 27 February 2016].
26. F. R. Oliveira and L. d. V. Cura, “Performance Evaluation of NoSQL Multi-Model Data Stores in Polyglot Persistence Applications,” In Proceedings of the 20th International Database Engineering & Applications Symposium (IDEAS '16), Evan Desai (Ed.). ACM, New York, NY, USA, pp. 230-235, 2016.
27. Mongo DB, Inc, “the most popular document stores,” [Online]. Available: https://Mongodb.com/, [Accessed: 27 March 2017].
28. Apache Software Foundation, “Wide-column store based on ideas of BigTable and DynamoDB,” [Online]. Available: https://cassandra.apache.org, [Accessed: 27 March 2017].
29. S. N. Swaminathan and R. Elmasri, “Quantitative Analysis of Scalable NoSQL Databases,” IEEE International Congress on Big Data (BigData Congress), San Francisco, CA, pp. 323-326, 2016.

30. Neo Technology, “Open source graph database,” [Online]. Available: https://neo4j.com, [Accessed: 27 March 2017].
31. OrientDB LTD, “Multi-model DBMS,” [Online]. Available: https://orientdb.com, [Accessed: 27 March 2017].
32. D. G. Chandra, “BASE Analysis of NoSQL Database,” Future Generation Computer Systems, vol. 52, pp. 13–21, 2015.
33. D. Jayathilake, Ch. Sooriaarachchi and ET. Al., (2012). A study into the capabilities of NoSQL databases in handling a highly heterogeneous tree, Information and Automation for Sustainability (ICIAfS), IEEE 6th International Conference on Beijing, IEEE, 106 - 111.
۳۴. کشوری، سامان. صابری، حسن و کشوری، ساناز. (۱۳۹۴)، نقش نظریه CAP و همزیستی مسالمت‌آمیز در انتخاب بانک‌های اطلاعاتی، سومین کنفرانس بین المللی پژوهش‌های کاربردی در مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه تربیت مدرس، تهران.

Choosing the Right Distributed Database to Achieve the Best Performance in Management of Big Data

S. Keshvari, M. Naghavi*, S. Keshvari

Abstract

Selecting the appropriate database according to the requirements of each program is always a concern for developers. There are different types of databases among which the relational database is the most famous one. With the increasing use of the Internet, which led to the rapid growth of data and new requirements in the area of Big Data, this type of databases had a bad practice in the management of unstructured and large data. With such features to manage data, key-value, Document Oriented, Wide Column and graph-based databases are provided. Each of these Databases provides solutions with a specific approach. Multi-modal databases provided multiple approaches to these features that the solutions are able to support multiple requirements. In this paper, at first, emphasizing on the relational model and giving an example, along with describing the structure and features of a variety of NoSQL databases, we offer relevant examples with respect to each of these models. Then, we introduce a common database of any kinds of several banks to compare the structure and feature of each choice and evaluate their performance. The main criteria for comparing are the speed storage and retrieval of information, and in presenting the results of the evaluation criteria of the approach, it has been tried to test and evaluate different needs. The target of this article is not to provide a superior example of NoSQL databases, but to offer high suitable NoSQL database in various applications based on defined needs.

Key Words: *Big Data, NoSQL Databases, Distributed Database.*

* Imam Hossein Comprehensive University (mnaghavi@ihu.ac.ir) - Writer-in-Charge