آبگیر داده: رویکردی نوین جهت مدیریت و تحلیل بی‌درنگ داده‌های حجیم

سامان کشوری[۱،*]، حسن نادری[۲] و مجید غیوری ثالث[۳]

اطلاعات مقاله

دریافت مقاله: ۱۳۹۵/۰۳/۲۶

پذیرش مقاله: ۱۳۹۶/۱۱/۱۷

واژگان کلیدی:

داده‌های حجیم،

بانک‌های اطلاعاتی NoSQL،

آبگیر داده،

دریاچه داده،

انبار داده بی‌درنگ.

چکیده

با افزایش سرعت تولید داده‌ها، نیاز به پردازش، ذخیره‌سازی و تحلیل داده‌های حجیم، روزبه‌روز در حال افزایش است. کارهای مرتبطی برای ایجاد انبار داده بی‌درنگ انجام شده است، ولی با توجه به وجود داده‌های ناساخت‌یافته در داده‌های حجیم، انبار داده با ساختار قدیمی، پاسخ‌گوی نیازمندی‌های جدید مدیریت این نوع داده‌ها نیست. به‌تازگی دریاچه داده برای داده‌های ناساخت‌یافته (با خصوصیت BASE) مطرح شده است؛ اما وجود داده‌های حساس ساخت‌یافته (با خصوصیت ACID) و داده‌های با حساسیت کمتر غیرساخت‌یافته در داده‌های حجیم از طرفی باعث بروز مشکلاتی جدید در مدیریت داده‌های حجیم با استفاده از این روش‌ها شده است. در این مقاله راه‌حلی ارائه خواهد شد که قادر خواهد بود داده‌های ساخت‌یافته و ناساخت‌یافته با خصوصیات متفاوت را به‌صورت هم‌زمان ذخیره‌سازی و به پرس‌وجوهای کاربر به‌صورت بی‌درنگ پاسخ دهد. روش مذکور پس از بررسی انبار داده و دریاچه داده، مشخص کردن قوت‌ها و ضعف‌ها و در نهایت، با تلفیق این دو روش مطرح شده است. به‌عنوان یکی از نتایج مهم این تحقیق پس از مقایسه انبار داده و دریاچه داده خواهیم دید، دریاچه داده جایگزینی برای انبار داده نبوده، انبار داده کاربردهای خاص خود را مخصوصاً در داده‌های مالی دارد؛ زیرا از نظریه ACID پیروی کرده، دریاچه داده نیازمندی‌های نظریه BASE را رفع می‌کند. ایده مطرح‌شده در این مقاله با عنوان آبگیر داده، دارای سه مزیت اصلی است: ۱. استفاده هم‌زمان از انبار داده و دریاچه داده برای پاسخگویی بی‌درنگ به انواع نیازهای داده‌ای سازمان با بهره‌گیری از مزایای آن‌ها؛ ۲. تفکیک داده‌های جدید از قدیمی برای رسیدن به بی‌درنگی؛ ۳. ایجاد توازی و در نتیجه عدم هم‌زمانی بارگذاری داده و پردازش پرس‌وجو جهت کاهش هزینه زمانی.

۱- مقدّمه

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، شکل (۱) میزان رشد تولید داده‌هایی را که از طریق موبایل در جهان از سال ۲۰۰۹ تا ۲۰۱۵ تولیدشده‌اند، نشان می‌دهد. طبق این آمار به‌ازای هر موبایل به‌طور میانگین حدود ۱,۴ گیگابایت داده تولید

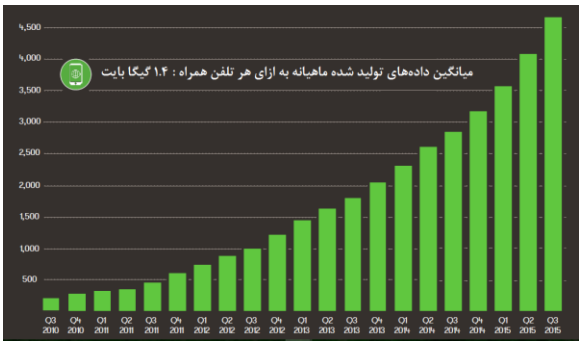
* پست الکترونیک نویسنده مسئول: SaKeshvari@ihu.ac.ir

۱. دانشجوی کارشناسی ارشد، مهندسی کامپیوتر، دانشگاه جامع امام حسین(ع)

۲. استادیار، مهندسی کامپیوتر، دانشگاه علم و صنعت ایران

۳. استادیار، مهندسی کامپیوتر، دانشگاه جامع امام حسین(ع)

می‌شود [۱]. 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.[۱]



شکل ۱- نمودارداده‌های تولیدشده توسط موبایل [۱]

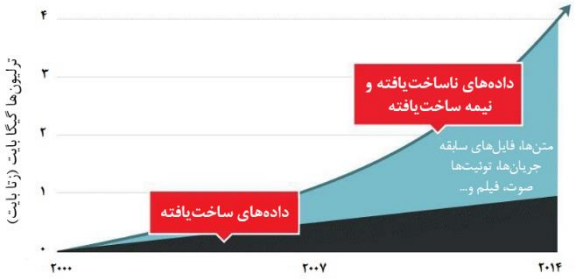
در شبکه اجتماعی فیس‌بوک حدود ۱،۴۱۶،۴۲۴،۵۰۰ تعداد کاربر فعال حضور دارد. این آمار برای شبکه‌های اجتماعی گوگل پلاس و توییتر به‌ترتیب ۱،۱۳۰،۳۶۴،۰۰۰ و ۳۰۹،۱۸۱،۲۰۰ است.[۱]

حجم انبار داده Ebay به حدود ۹۰ پتابایت رسیده است [۲]. این آمارها به‌صورت لحظه‌ای در حال رشد هستند.

جدول ۱- آمار ترافیک چند سایت معتبر[۱]

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

با توجه به این حجم بالا، پردازش سنگین و همچنین تنوع زیادی که در داده‌ها مشاهده می‌شود، نیاز به روش‌هایی برای ذخیره‌سازی، مدیریت و تحلیل این حجم از داده‌ها وجود دارد که بتواند عملیات موردنظر را با سرعت و دقت بالا انجام دهد. با توجه به ضعف‌هایی که سیستم‌های ذخیره‌سازی سنتی دارند، ساختارهای موجود جواب‌گوی برخی از نیازهای اساسی امروزی نیست [۳]. جهت ذخیره‌سازی این حجم از داده‌ها باید سیستم‌های ذخیره‌سازی توزیع‌شده استفاده شود. برای پاسخگویی به نیازهای جدیدی که در بانک اطلاعاتی به وجود آمد و عدم‌کارایی مدل رابطه‌ای، پایگاه‌های داده‌های جدیدی با عنوان NoSQL[۳] به وجود آمدند [۴]. کارلو استروزی نخستین بار در سال ۱۹۹۸ عبارت NoSQL را برای اشاره به پایگاه‌های داده‌ای سبک و متن‌باز رابطه‌ای به کار گرفت که از رابط SQL استفاده نمی‌کردند.[۴] بعدها وی به این نکته اشاره کرد که این عبارت و مفهوم پشت آن، از مدل رابطه‌ای جداشده و بهتر است NoREL[۵] نامیده شود. در سال ۲۰۰۰ میلادی، اریک بریور با ارائه نظریه CAP[۶] به کمبودها و محدودیت‌های مدل رابطه‌ای در سیستم‌های توزیع‌شده اشاره کرد.

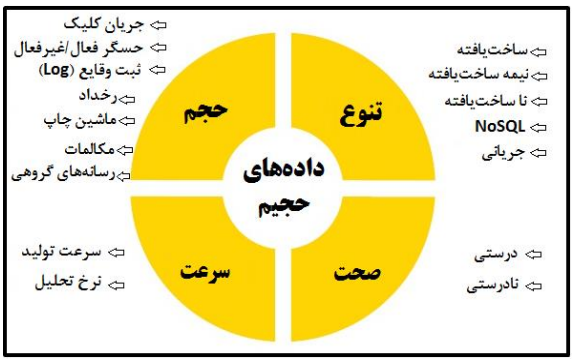


شکل ۲- حجم داده‌های ساخت‌یافته و ناساخت‌یافته در جهان [۶]

طبق نظریه CAP، در یک پایگاه داده در یک شبکه توزیع‌شده و وسیع، سازگاری و دسترس‌پذیری بالا باهم قابل فراهم شدن نیستند [۵]. درواقع، با استفاده از سیستم فایل‌های توزیع‌شده و بانک‌های اطلاعاتی NoSQL می‌توان مشکلات ذخیره‌سازی و بانک اطلاعاتی این حجم از داده‌ها را برطرف کرد.

مشکل دیگر، تغییر در نوع داده‌ها است. همان‌طور که در شکل (۲) مشاهده می‌شود، در سال‌های اخیر داده‌های ساخت‌یافته، مقدار کمی از داده‌ها را به خود اختصاص داده‌اند. درواقع، زمانی که بحث داده‌های حجیم مطرح می‌شود، مسائلی همچون تنوع، حجم، سرعت و صحت داده‌ها پیش می‌آید (شکل ۳). به این موارد می‌توان مصورسازی، تغییر و مقدارپذیری را نیز افزود [۷].

[1]. http://www.internetlivestats.com
[2] طبق آمار ارائه‌شده، حدود ۶۷٪ ایمیل‌ها هرزنامه هستند.
[3].Not Only SQL
[4].http://www.strozzi.it/cgibin/CSA/tw7/I/en_US/NoSQL/Home%20Page
[5].Not Only Relational
[6].Consistency- Availability- Partition Tolerance



شکل ۳- ویژگی داده‌های حجیم

به این دلایل، تحلیل بر روی داده‌ها به کمک انبار داده سنتی که برای داده‌های ساخت‌یافته طراحی شده‌اند، مناسب نیست. همچنین نمی‌توان عملیات‌هایی از جمله تحلیل کلیک‌های جریانی را که توسط کاربران انجام می‌شود، با سرعت‌بالا تحلیل کرد. از طرفی، وجود داده‌های ساخت‌یافته، نیمه‌ساخت‌یافته و ناساخت‌یافته در کنار یکدیگر منجربه پیچیدگی کار شده است. جهت رفع این اشکالات، دریاچه داده توسط مارتین فلاور معرفی شده است [۸]. برخلاف انبارهای داده که به‌صورت خاص برای مقاصد تحلیل و گزارش‌گیری‌های مدیریتی طراحی می‌شوند و شکل داده‌های ورودی را به قالب موردنیاز خود تبدیل و ذخیره می‌کنند، دریاچه داده، اطلاعات و داده‌ها را به‌صورت خام ذخیره می‌کند. این امکان باعث می‌شود در آینده، ابزارهای پردازشی و تحلیلگران، خود تصمیم بگیرند که چه تبدیل یا پردازشی روی آن‌ها انجام بدهند.

با توجه به اهمیت تصمیم‌گیری بر اساس تحلیل داده‌ها و تراکنش‌های موجود، مخصوصاً در تراکنش‌های مالی بانک‌ها و شرکت‌هایی که خدمات‌دهنده کارت‌های الکترونیک هستند، انجام تحلیل به‌صورت بی‌درنگ، نیاز به ایجاد بستر مناسب از جمله انبار داده‌های بی‌درنگ دارد که با توجه به ضعف انبار داده این ساختار به‌خوبی پاسخگوی نیازها در داده‌های حجیم نیست. در استفاده از دریاچه داده نیز به دلیل حجم بالای اطلاعات تحلیل به‌صورت بی‌درنگ انجام نمی‌شود. با توجه به ضعف‌ها و قوت‌های دریاچه و انبار داده، در این مقاله، ساختاری با عنوان آبگیر داده جهت انجام تحلیل به‌صورت بی‌درنگ ارائه شده است.

در بخش ۲ این مقاله، ابتدا کارهای مرتبط با موضوع تحقیق ارائه شده، در ادامه و در بخش ۲ ضمن تشریح ساختار و چالش‌های دریاچه داده و انبار داده، به مقایسه بین این دو پرداخته می‌شود. برای انجام تحلیل‌های موردنیاز بر روی داده‌ها، انبارهای داده و دریاچه داده دارای ضعف‌ها و قوت‌هایی هستند. با توجه به ضعف‌های این دو، راهکارها و ابزارهای ارائه‌شده با یکدیگر مقایسه شده‌اند. در ادامه، معماری آبگیر داده و نحوه پیاده‌سازی آن ارائه شده است. در پایان نیز نتیجه‌گیری آمده است.

۲-کارهای مرتبط

در این مقاله یک معماری نو مدیریت داده برای تحلیل بی‌درنگ معرفی می‌شود. کارهای مرتبط با این مقاله تحقیقاتی شامل دریاچه داده، انبار داده بی‌درنگ و انبار داده جریانی هستند. اخیراً شرکت‌ها و مؤسسات از انبار داده بی‌درنگ استفاده می‌کنند. در مقاله لنگست نگرانی برای تراکنش‌های مالی و اهداف کسب‌وکار مطرح شده است [۹]. رویکردی که در بیشتر مقالات مرتبط، جهت رسیدن به بی‌درنگ بودن به کار گرفته می‌شود، ایجاد جدول‌های موقتی برای تشکیل انبار داده است [۱۱ و ۱۰]. در تحقیق دیگری [۱۲] علاوه بر ایجاد جدول‌های موقت، قطعه‌بندی را به تشکیل جدول‌های موقت اضافه کرده‌اند که در آن به‌عنوان‌مثال، قطعه‌ای برای ساعت قبل و قطعه‌ای دیگر، داده‌های روز قبل را در خود خواهد داشت.

در مقاله نگویت و تجاو، نویسنده ابزاری پیشنهاد می‌دهد که مبتنی بر داده‌های جریانی است و به داده‌ها اجازه تازگی[1] و ادغام متداوم را می‌دهد. نتایج به‌دست‌آمده در این مقاله تقریبی هستند. علاوه بر سایر معیارهایی که توسط نویسنده ارزیابی شده، نتایج استفاده از داده‌های جریانی هزینه زیادی دارند [۱۳]. در همین زمینه، در تحقیق دیگری فواید، چالش‌ها و روش‌های پیاده‌سازی انبار داده جریانی مطرح شده است [۱۴]. در کار ژو و همکاران، راه‌حلی مبتنی بر معماری سرویس‌گرا[2] ارائه شده است. در این رویکرد، ابتدا استخراج داده‌ها از طریق وب سرویس انجام شده، نتایج در حافظه

[1].Freshness

[2].SOA

پنهان ذخیره می‌شوند. سطوح مختلفی از حافظه نهان وجود دارد؛ به‌عنوان‌مثال ۵، ۱۰، ۳۰ و ۶۰ دقیقه پس از گذشت زمان، داده‌ها از طریق حافظه‌های پنهان به انباره داده انتقال پیدا می‌کنند. این راه‌حل دارای قسمتی است که اطلاعات را از حافظه‌های پنهان پیوند می‌دهد [۱۵]. در تحقیق دیگری با استفاده از خدمات وب، نمونه‌ای اولیه برای اجرای انبار داده بی‌درنگ پیشنهاد شده است [۱۶].

در مقاله واسیلاید و اسمیت، نویسنده‌ها دو مسئله سنتی و بی‌درنگ را در انباره داده مطرح می‌کنند. معماری پیشنهادی آن‌ها که ثابت بودن نرخ داده‌ها را در عملیات بازسازی در انباره داده تضمین می‌کند، دارای پنج سطح است [۱۷]:

۱. استخراج اطلاعات و تحویل آن به دارندگان اطلاعات؛

۲. همگام‌سازی داده‌ها از دارندگان داده‌ها و داده‌های تبادل‌شده در فواصل معین یا مبتنی بر فشار[1] به سطح متوسط شناخته‌شده به‌عنوان منطقه پردازشی داده؛

۳. سطح PDA[2] یا همان منطقه پردازشی داده مسئول انجام عملیات تبدیل (دگرگونی)؛

۴. سطح همگام‌سازی PDA و انبار داده؛

۵. سطح انبار داده.

دو مشکل دربارهٔ کار آن‌ها وجود دارد. نویسندگان در حین بارگذاری و فاز همگام‌سازی وجود شاخص‌ها، مارت‌های داده[3] و خلاصه‌های محقق‌شده[4] را که در چه صورت در انبار داده بی‌درنگ نیاز هستند، در نظر نگرفته‌اند؛ زیرا این موارد باعث کاهش کارایی و سرعت ایجاد انبار داده می‌شوند. از آن مهم‌تر اینکه مطالعه آن‌ها فرضی است. آن‌ها روش پیشنهادی خود را ارزیابی نکرده‌اند. به‌طور خلاصه مشکلات عملکرد معرفی‌شده، عدم در نظر گرفتن و ارزیابی نکردن هم‌زمانی بین پرس‌وجوهای آنلاین و بارگذاری مداوم داده است. در تحقیقی دیگر، اشکالات این روش رفع شده، پرس‌وجوها و بارگذاری داده‌ها با حداقل تنزل کارایی رخ می‌دهد؛ زیرا آن‌ها می‌توانند در نمونه‌های مختلفی از پایگاه داده در ماشین‌های مختلف انجام پذیرند. در این تحقیق با جداسازی دو بخش داده‌های جدید و قدیم به‌ترتیب در انبار داده‌های پویا و ایستا، نویسندگان توانسته‌اند زمان پاسخگویی به پرس‌وجوها را به‌صورت چشمگیری کاهش دهند [۱۸]. معماری گزارش‌شده در مقاله مذکور، شامل سه بخش زیر است:

۱. انبار داده پویا (D-DW[5])؛

۲. انبار داده ایستا (S-DW[6])؛

۳. ادغام کننده[7].

داده‌های جدید در انبار داده پویا و داده‌های قدیمی در انبار داده ایستا قرار دارند. درصورتی‌که پرس‌وجو به صورتی بود که در آن نیاز به داده‌های قدیمی و جدید وجود داشت، پاسخ هر قسمت با کمک ادغام‌کننده با یکدیگر ادغام شده، پاسخ به کاربر نشان داده می‌شود. روش ارائه‌شده در جاهایی که نیاز به انباره داده بی‌درنگ مطرح است می‌تواند مورد استفاده قرار گیرد؛ اما با توجه به نیازهای امروزی و ضعف‌های انبار داده که در مقدمه به قسمتی از آن اشاره شده و در ادامه بیشتر به آن خواهیم پرداخت، شرکت‌ها ملزم به استفاده از دریاچه داده هستند. پس باید برای تحلیل بی‌درنگ در دریاچه داده نیز پیکربندی و معماری مناسبی ارائه کرد که یکی از اهداف این مقاله است.

هوانگ در مقاله خود [۱۹] ضمن معرفی، ویژگی‌ها و قابلیت‌های دریاچه داده، اهمیت این مفهوم را برای اکوسیستم مدیریت داده‌ها برشمرده است. نویسنده در این مقاله، افکار و اعمال خود را در رابطه با دریاچه داده تشریح کرده است. در این تحقیق، مقایسه‌ای بین انبار داده و دریاچه داده نیز ارائه شده که در مقاله ما این مقایسه بر اساس معیارهای بیشتر و دقیق‌تر انجام گرفته است.

ریحان و همکارانش[۲۰] یک سیستم دریاچه داده با مدیریت فراداده خبره به نام کانستنس[8] ارائه کردند که توانایی واکشی داده‌ها از منابع ناهمگن داده را دارد. کانستنس همچنین توانایی پوشش، واکشی و خلاصه کردن فراداده ساخت‌یافته از منابع داده را داشته، داده و فراداده را با اطلاعات معناداری برای رفع ابهامات، حاشیه‌نویسی می‌کند. با جایگذاری موتور بازنویسی پرس‌وجو، این سیستم داده‌های ساخت‌یافته و نیمه‌ساخت‌یافته نیز پشتیبانی می‌شود. کانستنس یک رابط

[1].Push-based
[2].Process Data Area
[3].Data Marts
[4].Materialized summaries
[5].Dynamic Data Warehouse
[6].Static Data Warehouse
[7].Merger
[8].Constance

یکپارچه برای پردازش پرس‌وجو و شناسایی داده برای کاربران فراهم می‌سازد.

مقاله آلرهامی و والکر [۲۱] دریاچه داده شخصی را ارائه کرده است. دریاچه داده شخصی، امکانات یکپارچه ذخیره‌سازی، تحلیل و پرس‌وجوی داده‌های شخصی را فراهم می‌سازد. در این مقاله برای داده‌های ناساخت‌یافته، کشش جاذبه[۱] فعال‌سازی شده است؛ به این معنا که به کمک افزونه سوم، داده‌های ناساخت‌یافته می‌توانند تحلیل و پرس‌وجو شوند.

۳- مفاهیم پایه

آشنایی با مفاهیمی از جمله نظریه‌های بانک اطلاعاتی، انباره داده و دریاچه داده از نیازمندی‌های این مقاله بوده کهدر این بخش به آن‌ها خواهیم پرداخت.

۳-۱- نظریه‌های مطرح شده در بانک‌های اطلاعاتی

نظریه‌های مختلفی در استفاده از داده‌های نو وجود دارد که متناسب با کاربرد و اهمیت آن‌ها در معماری باید موردتوجه قرار گیرند. در ادامه به‌صورت خلاصه به برخی از آن‌ها اشاره می‌شود.

۳-۱-۱- نظریه ACID

ACID[۲]، یک روش استاندارد برای قضاوت درباره جامعیت داده بین سیستم‌های مدیریت داده است. این مجموعه از خصوصیات، تضمین می‌دهند که تراکنش‌های پایگاه داده با اطمینان انجام می‌شوند [۲۲]. ACID به این صورت تعریف می‌شود:

- تجزیه‌ناپذیر[۳] به معنی این است که هر تراکنش یا باید کامل اجرا شود یا اصلاً اجرا نشود.
- سازگاری[۴] تضمین می‌دهد که هر تراکنش، پایگاه داده را از یک وضعیت سالم و معتبر به یک وضعیت سالم و معتبر دیگر می‌برد.
- جداسازی[۵]تضمین می‌دهد که اجرای هم‌زمان تراکنش‌ها مشکلی در سلامت پایگاه داده ایجاد نکند.
- ماندگاری[۶] به‌معنی ذخیره کردن تغییرات داده‌ای اعمال‌شده توسط هر تراکنش به‌منظور بازگرداندن پایگاه داده به وضعیت قبل از تراکنش در زمان‌های سقوط[۷] یا رخ دادن خطا است.

تمرکز اصلی بانک‌های اطاعاتی NoSQL بر در دسترس بودن و توزیع شدن داده‌ها در بین سرورهای مختلف است، بنابراین، در مواردی که پشتیبانی از ACID ملاک انتخاب پایگاه داده است، انتخاب NoSQL انتخاب مناسبی نبوده، بهتر است از سیستم‌های رابطه‌ای که در آن‌ها ACID تضمین‌شده، استفاده شود. هرچند در بعضی از بانک‌های اطلاعاتی NoSQL ادعا می‌کنند که به‌طور کامل از ACID پشتیبانی می‌کنند[۲۳].

۳-۱-۲- نظریه BASE[۸]

عبارت BASE در تقابل با نظریه ACID ارائه شده است. NoSQL بر روی در دسترس بودن[۹] و کارایی بالا تأکید دارد. طبق نظریه CAP، ساخت پایگاه داده توزیع‌شده‌ای که ویژگی‌های ACID را فراهم کند و همواره در دسترس باشد، امکان‌پذیر نیست؛ بنابراین، در برنامه‌هایی که از BASE پیروی می‌کنند، سازگاری[۱۰] و جداسازی[۱۱] اغلب نادیده گرفته می‌شوند. یکی از مفاهیم اصلی BASE، فراهم‌سازی سازگاری از طرف توسعه‌دهنده است و نباید آن را برعهده بانک اطلاعاتی گذاشت [۲۴] که خود، موجب به وجود آمدن سبک جدیدی در مدیریت داده و برنامه‌نویسی شده است.

این مفهوم شامل موارد زیر است:

دسترسی‌پذیری اساسی[۱۲]: این محدودیت بیان می‌کند سیستم، در دسترس بودن داده‌ها را تضمین می‌کند و یک پاسخ برای هر درخواستی وجود دارد؛ اما داده‌ای که به‌عنوان پاسخ ارائه می‌شود، ممکن است به سازگاری نرسیده یا در حال تغییر باشد. درواقع، پاسخ ممکن است با شکست مواجه شود. حالت نرم[۱۳]: حالات سیستم باید در هرزمانی تغییر یابد، پس ممکن است با عدم ورود داده‌ها به سیستم، رخدادهایی جهت رسیدن به سازگاری نهایی[۱۴] انجام شود، بنابراین سیستم همواره در حالت نرم قرار خواهد داشت.

---

1. gravity pull
2. Atomicity, Consistency, Isolation, Durability
3. Atomicity
4. Consistency
5. Isolation
6. Durability
7. Crash
8. Basically Available, Soft state, Eventual consistency
9. Availability
10. Consistency
11. Isolation
12. Basically Available
13. Soft state
14. Eventual consistency

سازگاری نهایی: هنگامی‌که دریافت داده از طرف آن متوقف می‌شود، در نهایت، سیستم باید به یک حالت پایدار برسد. داده‌ها به هرجایی زودتر یا دیرتر پخش می‌شوند، اما سیستم به دریافت داده‌ها همچنان ادامه داده، سازگاری هر تراکنش را قبل از رفتن به تراکنش بعد بررسی نمی‌کند.

در جدول ۲، دو نظریه ACID و BASE به‌صورت خلاصه آمده است.

جدول ۲- ویژگی‌های دو نظریه BASE و ACID [۲۴] (با تغییرات)

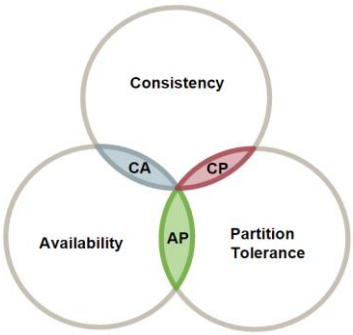
ACID	تجزیه‌ناپذیری	انجام تمام یا هیچ قسمتی از عملیات. تثبیت یا عقب‌گرد
	سازگاری	عدم مشاهده داده‌های ناسازگار توسط تراکنش
	جداسازی	عدم آگاهی و تأثیر تراکنش‌های هم‌روند از یکدیگر
	ماندگاری	تمامی رخدادهای تراکنش‌ها در سیستم باقی می‌ماند.
BASE	دسترسی‌پذیری اساسی	در دسترس بودن به کمک کپی‌برداری
	حالت نرم	محول کردن سازگاری به برنامه کاربر
	سازگاری نهایی	سازگاری ضعیف، پایگاه داده در زمان طولانی اجرا به ثبات رسیده و استفاده از داده‌های قدیمی مشکل ندارد.

۳-۱-۳- نظریه CAP

عبارت CAP از ترکیب سرنام مفاهیم زیر است: سازگاری[1]، تمامی گره‌ها اطلاعات یکسانی را در لحظه دارند؛ دسترسی[2]، هر درخواست حتماً یک پاسخ دریافت خواهد کرد؛ تحمل پارتیشن[3]، اگر بخشی از شبکه دچار مشکل شد، بقیهٔ سیستم به کار خود ادامه خواهد داد. اشکال در بخشی از شبکه نباید منجربه توقف کار کل سیستم شود.

نظریه CAP بیانگر این موضوع است که در سیستم‌های توزیع‌شده فقط امکان فراهم کردن دو گزینه از سه گزینه سازگاری، دسترسی و تحمل‌پذیری پارتیشن وجود دارد و گزینه باقی‌مانده، قربانی فراهم کردن سایر گزینه‌ها خواهد شد [۲۵]. با توجه به مشکلات شبکه‌ای که در هر سیستمی امکان بروز آن وجود دارد، توصیه متخصصان، انتخاب قطعی تحمل پارتیشن به‌عنوان یکی از معیارهای گزینش‌شده است. البته حجم بسیار زیاد داده‌های حجیم نیز این انتخاب را تقریباً به یک اجبار تبدیل کرده است. انتخاب گزینه بعدی از بین دسترسی بالا و سازگاری، به ماهیت نرم‌افزار و اولویت‌ها بستگی دارد که البته در بیشتر کاربردهای کلان داده این وزنه به‌سمت دسترسی متمایل است. سیستمی با حجم زیاد داده که به‌صورت توزیع‌شده پارتیشن شده باشد، اگر بخواهد شرط سازگاری را رعایت کند، باید به علت قفل‌های زیادی که بر روی فیلدهای داده‌ای قرار می‌دهد، به دفعات از دسترس کاربر خارج شده، وی را منتظر بگذارد. تصور چنین تصویری در سیستمی همچون تلگرام، در مقابل احتمال بسیار کم جابه‌جا شدن دو پیغام که گاه‌گاهی برای یک کاربر رخ می‌دهد، دلیل انتخاب دسترس‌پذیری بالا را بر سازگاری منسجم در بیشتر سیستم‌های کلان داده به‌خوبی روشن می‌کند.

انتخاب‌ها در نظریه CAP در شکل (۴) آمده است که در ادامه، توضیحات وضعیت‌های گزینشی بررسی خواهند شد[۲۴].



شکل ۴- نظریه CAP [۲۶]

با توجه به نیازمندی‌های پروژه، هرکدام از انتخاب‌های زیر را می‌توان داشت. در کارهای قبلی، ما به تأثیر این نظریه بر انتخاب مناسب بانک اطلاعاتی بر پایه این نظریه و نظریه هم‌زیستی مسالمت‌آمیز که در ادامه شرح داده می‌شود،

[1].Consistency
[2].Availability
[3].Partition Tolerance

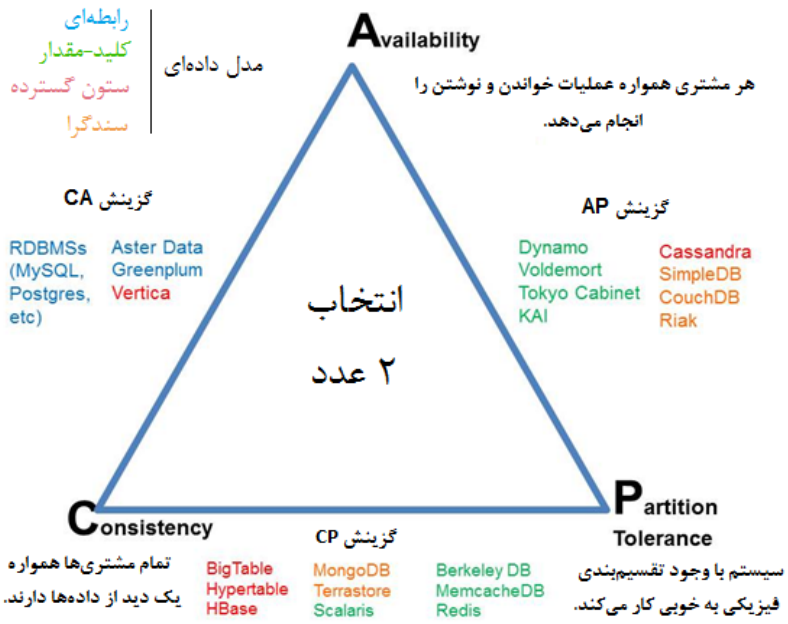
پرداخته‌ایم[۲۷].

جدول ۳- ACID در مقابل BASE [۲۴] (با تغییرات)

BASE [A+P | P+C]	ACID [C+A]
انجام دادن سازگاری، تجزیه‌ناپذیری و تحمل‌پذیری به‌صورت در نهایت	سازگاری قوی
اولویت در دسترس بودن	جداسازی
بهترین تلاش	تمرکز بر تثبیت
پاسخ تقریبی	تراکنش‌های تودرتو
خوش‌بین: پذیرش بانک‌های اطلاعاتی موقت ناسازگار یا در نهایت سازگار	بدبین: سازگاری اجباری در پایان تراکنش
ساده‌تر، سرعت و تکامل آسان	دشواری تکامل
مناسب برای برنامه‌های غیرمالی مبتنی بر وب	مناسب برای تراکنش‌های مالی
سریع	امن
عدم اشتراک‌گذاری	اشتراک برخی دستگاه‌ها نظیر حافظه
توسعه‌پذیری نامحدود	توسعه‌پذیری محدود
کدهای پیچیده و بانک اطلاعاتی ساده	کدهای ساده، بانکاطلاعاتی قوی
خوشه‌بندی	ماشین تک کاربرِ
مطابق با نظریه CAP	A + C
توسعه‌پذیری افقی	توسعه‌پذیری عمودی
API های اختصاصی	SQL
اغلب به‌وسیله کلید نمایه‌گذاری شده‌اند	نمایه‌گذاری کامل
در دسترس بودن	در دسترس بودن
تهاجمی (خوش‌بینانه)	محافظه‌کارانه (قفل بدبینانه)

- **گزینش AP:** انتخاب دسترسی و تحمل پارتیشن به این معنی است که هر درخواستی، پاسخی دریافت خواهد کرد. این پاسخ تا حد ممکن، شامل جدیدترین اطلاعات خواهد بود، ولی امکان قدیمی بودن آن نیز وجود دارد. از طرفی، نوشتن اطلاعات جدید بر روی گره‌ها ممکن است در لحظه، امکان‌پذیر نباشد. بااین‌حال، سیستم به کار خود ادامه خواهد داد و در نهایت، به ثبات اطلاعات خواهد رسید. در گزینش AP ، در دسترس بودن سیستم و سرعت بیشتر آن، اولویت بالاتری نسبت به سازگاری و دوام اطلاعات در لحظه دارد [۲۷]. این انتخاب مطابق با نیازهای مطرح‌شده در نظریه BASE است [۲۴].
- **گزینش**CP: انتخاب سازگاری و تحمل پارتیشن به این معنی است که هر درخواستی برای دریافت اطلاعات، دقیقاً باید آخرین نسخه موجود را به‌عنوان پاسخ ارسال کند. از طرفی، این آخرین نسخه موجود از هر اطلاعاتی بر روی سیستم کاملاً مشخص و واضح است؛ یعنی اگر دو درخواست یکسان از دو مکان مختلف به سیستم ارسال شود، سیستم جواب یکسان و مشخصی را به هر دو خواهد داد. هر میزان که سازگاری موردنیاز افزایش داده شود، سیستم دچار رکورد شده، سرعت پاسخ‌دهی آن کاهش می‌یابد [۲۷].
- **گزینش**CA: انتخاب دسترسی و ثبات موجب می‌شود اطلاعات در یک خوشه باشند؛ بنابراین، تمام گره (خوشه) در حال استفاده است. پس اگر در این قسمت خطایی رخ دهد، سیستم کارایی خود را از دست خواهد داد [۲۸]. این گزینش مطابق با نیازهای مطرح‌شده در نظریه ACID است [۲۴].

با توجه به نیازهای مطرح‌شده در نظریه CAP و دو نظریه ACID و BASE در جدول ۳ ویژگی دو حالت انتخاب CA و AP آمده است.



شکل ۵-گزینش بانک‌های اطلاعاتی با توجه به نظریه CAP [۲۹]

- **گزینش**CP: انتخاب سازگاری و تحمل پارتیشن به این معنی است که هر درخواستی برای دریافت اطلاعات، دقیقاً باید آخرین نسخه موجود را به‌عنوان پاسخ ارسال کند. از طرفی، این آخرین نسخه موجود از هر اطلاعاتی بر روی سیستم کاملاً مشخص و واضح است؛ یعنی اگر دو درخواست یکسان از دو مکان مختلف به سیستم ارسال شود، سیستم جواب یکسان و مشخصی را به هر دو خواهد داد. هر میزان که سازگاری موردنیاز افزایش داده شود، سیستم دچار رکورد شده، سرعت پاسخ‌دهی آن کاهش می‌یابد [۲۷].
- **گزینش**CA: انتخاب دسترسی و ثبات موجب می‌شود اطلاعات در یک خوشه باشند؛ بنابراین، تمام گره (خوشه) در حال استفاده است. پس اگر در این قسمت خطایی رخ دهد، سیستم کارایی خود را از دست خواهد داد [۲۸]. این گزینش مطابق با نیازهای مطرح‌شده در نظریه ACID است [۲۴].

با توجه به نیازهای مطرح‌شده در نظریه CAP و دو نظریه ACID و BASE در جدول ۳ ویژگی دو حالت انتخاب CA و AP آمده است.

معیارهای مختلفی در انتخاب بانک‌های اطلاعاتی با توجه به نظریه‌های بیان‌شده وجود دارد. در شکل (۵) نمونه‌ای از انتخاب بانک‌های اطلاعاتی انتخاب‌شده نشان داده می‌شود. همین امر موجب استفاده هم‌زمان از چندین بانک اطلاعاتی در یک برنامه شده است. در ادامه، با ارائه نظریه هم‌زیستی مسالمت‌آمیز[1] به این مفهوم بیشتر خواهیم پرداخت.

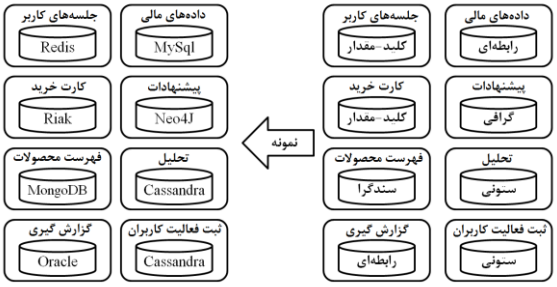
۳-۱-۴- نظریه هم‌زیستی مسالمت‌آمیز

طبق نظریه هم‌زیستی مسالمت‌آمیز، الزامی در استفاده تنها یک پایگاه داده برای ذخیره‌سازی اطلاعات برنامه نیست. با توجه به نیاز هر بخش برای رسیدن به بهترین کاربرد برای آن بخش، می‌توان از بهترین پایگاه داده استفاده کرد. به‌عنوان‌مثال، برای ذخیره‌سازی داده‌های باارزش و کم‌حجم، مانند پروفایل کاربران یا صورت‌حساب آن‌ها می‌توان از پایگاه داده رابطه‌ای مانند MySQL استفاده کرد. جهت ذخیره‌سازی داده‌های حجیم، مانند ثبت صفحات از بانک‌های اطلاعاتی سندگرا همچون مانگودی‌بی[2] استفاده شود، همچنین برای ذخیره‌سازی اطلاعات کم‌ارزش و حجیم، می‌توان از بانک اطلاعاتی کلید-مقدار، مانند ردیس[3] در برنامه استفاده کرد[۲۷].

[1].Polyglot persistence
[2].MongoDB
[3].Redis

هم‌زیستی مسالمت‌آمیز، علاوه بر فراهم کردن راهی مناسب برای توسعه برنامه‌های کاربردی جدید، این امکان را مهیا می‌سازد که توسعه‌دهندگان بتوانند برنامه‌های کاربردی کنونی خود را بر اساس آن تغییر داده، از مزایای هر دو نوع پایگاه داده استفاده کنند. بنابراین، همه توسعه‌دهندگان برای تولید برنامه‌های جدید می‌توانند در کنار استفاده از مدل سنتی رابطه‌ای، از امکانات پایگاه‌های داده‌هایNoSQL در برنامه خود بهره ببرند [۲۹].

طبق این نظریه در برنامه می‌توان به‌صورت جداگانه و طبق نیاز، دو نظریه BASE و ACID را پشتیبانی کرد. شکل(۶)نمونه‌ای از پیاده‌سازی این نظریه را نشان می‌دهد. با توجه به نیاز هر بخش می‌توان از بانک اطلاعاتی NoSQL یا رابطه‌ای و حتی نیمه‌ساخت‌یافته بهره برد. استفاده از چندین بانک اطلاعاتی منجربه بروز مشکل در آرشیو اطلاعات و تحلیل بر روی آن‌ها می‌شود؛ در نتیجه، انبار داده که برای داده‌های ساخت‌یافته به وجود آمده است، کارایی چندانی برای حل این مسئله ندارد. این مسئله منجربه استفاده از دریاچه داده شده است. در ادامه، ضمن بیان ضعف‌های انبار داده، این روند تشریح و تفاوت، مشکلات و چالش‌های این دو سیستم مدیریت داده بررسی می‌شود.



شکل ۶- نمونه‌ای از به‌کارگیری هم‌زیستی مسالمت‌آمیز [۲۷]

۳-۲- انبارداده و چالش‌های آن

یک انبار داده مکانی به‌عنوان مخزن اصلی داده‌های قدیمی یک سازمان است که برای گزارش‌گیری، تجزیه‌وتحلیل بهینه می‌شود [۳۰]. در انبار داده، داده‌ها بسیار ساختمند بوده، متخصصان داده ملزم به بازتغییر شکل داده‌ها به حالتی هستند که ساختار انبار داده مشخص می‌کند. برای ساخت انبار داده با توجه به موضوع ساماندهی‌شده برای کسب‌وکار، عموماً از یکی از دو روش ساخت انبار داده، یعنی روش رالف کیمبال[1] [۳۱] یا بیل آینمون[2] [۳۲] پیروی می‌شود. انبار داده دارای ویژگی‌های مفید زیر است:

- قابلیت اعتماد در یکپارچه‌سازی داده‌ها بالا است.
- تولید گزارش‌های تحلیلی بر اساس داده‌های سازمان انجام می‌شود.
- حاکمیت[3] و امنیت مؤثر
- کیفیت بالای داده
- اصالت داده
- کارایی تکرارپذیری[4]

اما این سیستم مدیریت داده، ضعف‌هایی نیز دارد. توسعه انبار داده، نیازمند زیرساخت و منابع وسیع است که منجربه بالا رفتن هزینهٔ آن می‌شود.

در انبار داده، داده‌ها به‌صورت تجمعی ذخیره می‌شوند. به‌عنوان‌مثال، آمار فروش سالیانه یا ماهیانه ذخیره‌شده و ذخیره‌سازی به‌صورت ساعتی و لحظه‌ای صورت نمی‌پذیرد. در انبار داده، مراحلی همچون آماده‌سازی و نرمال‌سازی، زمان زیادی را به خود اختصاص داده، منجربه جلوگیری از چابکی در تحلیل می‌شود. در انبار داده، برخی از قواعد حاکمیت داده رعایت نمی‌شود، از جمله:

- حفظ بافت اصلی داده
- اصالت داده
- یکپارچگی داده‌های سازمانی

با پدیدآمدن بسترهای NoSQL برای ذخیره‌سازی و واکشی اطلاعات، به علت نبود ساختار منسجم داده، در تحلیل داده‌های حجیم خلأ ایجاد شده است. با توجه به این نیاز، دریاچه داده پیشنهاد شده است که در ادامه، ساختار و چالش‌های آن را بیان خواهیم کرد. در بخش بعدی نیز به تفاوت این دو خواهیم پرداخت.

۳-۳- دریاچه داده و چالش‌های آن

دریاچه داده، اصطلاحی است که یکی از اجزا و قسمت‌های

[1].Ralph Kimball
[2].Bill Inmon
[3].Governance
[4].Repeatable performance

مهم در تحلیل‌های خط لوله در داده‌های حجیم به شمار می‌رود [۸]. دریاچه داده، روش ذخیره‌سازی حجم انبوهی از داده‌ها بر مبنای هزینه پایین ذخیره‌سازی، پالایش، آرشیو و شناسایی داده‌های جدید است. دریاچه داده، شامل داده‌های ناساخت‌یافته یا چند ساختاری است که اغلب، میزان ارزش آن‌ها برای سازمان مشخص نیست [۱۹].

در دریاچه داده، داده‌ها به‌صورت خام ذخیره می‌شوند؛ به همین دلیل، نیاز به تغییر شکل و قالب داده‌ها و همچنین پالایش داده در دریاچه داده وجود ندارد. در دریاچه داده، داده‌های اصلی برای رفع مشکل ذخیره‌سازی تجمعی نگهداری می‌شوند؛ مثلاً اگر نیاز به گزارش هفتگی، ماهیانه، سالیانه و از طرفی روزانه و لحظه‌ای باشد، دریاچه داده به‌خوبی پاسخگوی آن خواهد بود.

در این سیستمِ مدیریت داده، هزینه راه‌اندازی، پایین است و توانایی ترکیب پرس‌وجو داده‌های بدون ساختار، نیمه‌ساخت‌یافته و NoSQL وجود دارد. دریاچه داده، تمام داده‌ها را ذخیره می‌کند، اعمال تغییر در آن ساده بوده، نگرش درباره داده‌ها را با سرعت فراهم می‌سازد. ذخیره‌سازی در دریاچه داده برخلاف انبار داده بدون جابه‌جایی داده‌ها انجام شده، تغییرات در آن ساده است. دریاچه داده از تمام انواع کاربرانی که در [۳۳] ذکر شده‌اند، پشتیبانی می‌کند.

در دریاچه داده، برای تعریف ساختار داده‌ای که زمان استفاده از آن بهترین کارایی را در تجزیه تحلیل داشته باشد، به‌صورت «طراحی برای خواندن»[1] ایجاد می‌شود.

در رابطه میان دریاچه داده و سایر داده‌ها و نرم‌افزارهای یک سازمان، دریاچه داده به‌عنوان یک محیط جنبی و کاملاً مجزا از داده‌های عملیاتی، به ذخیره داده‌ها و اطلاعات در قالب‌های باز (مانند جیسان[2]، اکس‌ام‌ال[3] و متن) یا بانک‌های اطلاعاتیNoSQL بدون ساختار می‌پردازد و تحلیلگران از روی این داده‌ها، ساختارهای تحلیلی خود را طراحی و اجرا می‌کنند. نمایی پیشنهادی از این ساختار در شکل (۷) قابل‌مشاهده است.

اهداف متخصصان از به‌کارگیری دریاچه داده، انجام عملیات بی‌درنگ و رفع نیازهای تجزیه تحلیل عصر کنونی است که شامل تجزیه تحلیل داده‌های جریانی و حجیم است. دریاچه داده، داده‌ها را به همان صورتی که هستند، از منابع داده ناهمگن، اکتساب و در مخازن داده ذخیره می‌کند و به کاربران اجازهٔ اجرای پرس‌وجو بر روی آن‌ها را می‌دهد.

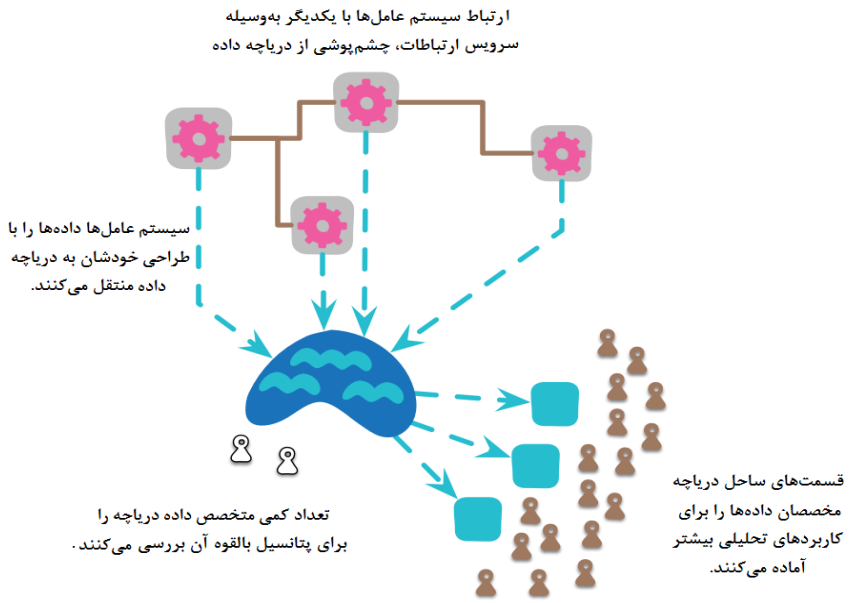
برای استفاده از دریاچه داده به‌عنوان سیستم مدیریت داده، باید به چالش‌های این سیستم توجه کرد. تعیین اصالت داده، به این معنا که داده از کجا و در چه زمانی ایجاد شده و در زمان موردنیاز توسط چه کاربرانی قابل‌دسترس است و از دسترسی چه کاربرانی به آن باید جلوگیری کرد. تعیین کیفیت داده و امنیت داده‌های موجود در دریاچه نیز از اهمیت زیادی برخوردار است. ارائه اصالت داده[4] و عملکرد بالا در پاسخگویی به نیازهای کاربران، از موارد مهم در دریاچه داده است.

در ابتدای تعریف دریاچه داده، طراحی اطلاعات، نگاشت‌ها[5] و محدودیت‌های دیگر به‌صراحت تعریف نشده است. در مرحله اکتساب[6] داده، استخراج ابر داده از منابع داده از اهمیت زیادی برخوردار است. مدیریت فراداده برای استدلال داده[7]، پردازش پرس‌وجو و مدیریت کیفیت داده اهمیت دارد. بدون فراداده، دریاچه داده به‌سختی به‌عنوان ساختاری ساخت‌یافته و قابل‌استفاده است و معانی داده‌ها قابل‌فهم نخواهد بود؛ به این صورت، دریاچه داده به‌سرعت به یک باتلاق داده[8] تبدیل خواهد شد [۲۰].

معمولاً ۸۰ درصد از زمان متخصصان داده، صرف آماده‌سازی داده‌ها می‌شود [۳۴]؛ زیرا داده‌ها برای تحلیل آماده نیستند.

در دریاچه داده، توسعه‌پذیری بسیار بالا است و شتاب بیشتری برای دسترسی کاربران وجود دارد، به همین دلیل روش‌های سنتی ذخیره و بازیابی اطلاعات، پاسخگوی نیازهای دریاچه داده نیست. با توجه به تنوع در داده‌های دریاچه داده، باید این مقدار حجم و تنوع، به‌خوبی مدیریت شود. استفاده کسب‌وکارهای کوچک و متوسط از دریاچه داده نباید برای آن‌ها هزینهٔ زیادی به همراه داشته باشد تا تمام متخصصان داده بتوانند از این سیستم مدیریت داده استفاده کنند.

[1].schema on read
[2].JSON
[3].XML
[4].data proven
[5].mappings
[6].ingestion
[7].data reasoning
[8].data swamp



شکل ۷- ارتباط دریاچه داده با سایر نرم‌افزارها [۸]

با توجه به حرکت از ذخیره‌سازی داده‌ها و مدیریت فایل‌ها به‌سمت حاکمیت داده، این مبحث برای دریاچه داده از اهمیت زیادی برخوردار است و یکی از چالش‌های مهم دریاچه داده به شمار می‌رود. طیفی از کاربران، از داده‌های خام استفاده می‌کنند و تعیین اصالت داده، اهمیت زیادی دارد. در انبار داده، سازوکار این معیار، معین است، اما در دریاچه داده، برای رسیدن به تکرار[1]، این معیار کاهش می‌یابد، پس باید میان حاکمیت و تکرار، با توجه به حجم درخواست‌های سیستم، تعادل ایجاد کرد. به عقیده برین نمی‌توان هم‌زمان، سرعت در پاسخگویی به پرس‌وجوها و پشتیبانی کامل از حاکمیت داشت [۳۵]. شرکت آپاچی، «اطلس»[2] را برای حاکمیت و چارچوب فراداده برای هدوپ[3] معرفی کرده است. اطلس مجموعه توسعه‌پذیر و مقیاس‌پذیر برای ارائه سرویس‌های اصلی حاکمیت داده، است که امکان ادغام داده‌های سازمانی را به‌صورت مؤثر و کارآمد مطابق با نیازها بر بستر هدوپ ارائه می‌دهد [۳۶]. اطلس به‌صورت مجزا از هدوپ، به‌صورت مستقل از بستر[4] اجرا می‌شود. شکل (۸) معماری اطلس را نشان می‌دهد.

با توسعه چابک نرم‌افزار باید به ارزش داده‌ها توجه بیشتری کرد؛ زیرا ممکن است داده‌ای در حال حاضر دارای ارزش چندانی نبوده، در آینده از ارزش زیادی برخوردار باشد.

شکل ۸- معماری آپاچیاطلس [۳۷]

یکی از چالش‌های دیگر دریاچه داده، ساخت دریاچه داده بدون شناخت از داده‌ها است، به این معنا که مشخص نیست داده‌ها دارای چه نوع و حجمی بوده یا با چه سرعتی به دریاچه وارد شوند. داده‌های فرار[5] مانند داده‌های یکتا و مهرهای زمانی[6] که نیاز به آن‌ها بدیهی است نیز از اهمیت زیادی برخوردارند. مدیریت این نوع داده‌ها در سیستم‌های

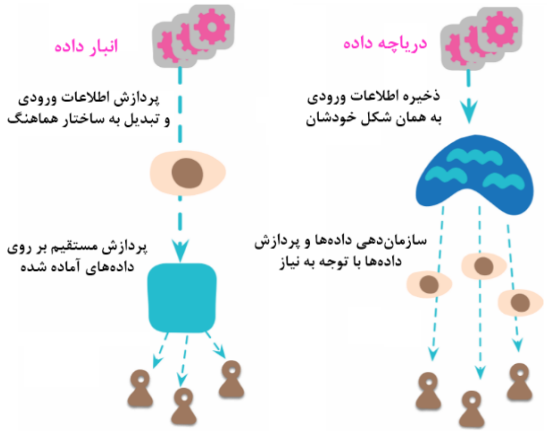
[1].iterative
[2].Atlas
[3].Hadoop
[4].Platform-agnostic
[5].from the get-go
[6].Time stamp

توزیع‌شده‌ای که دریاچه داده استفاده می‌کند، یکی دیگر از چالش‌ها است. در آغاز ایجاد دریاچه داده، توسعه‌دهنده ممکن است یک سیستم فایل معمولی یا بانک اطلاعاتی را برای سیستم خود در نظر بگیرد؛ ولی به دلیل اینکه ممکن است حجم داده‌ها به‌صورت نمایی رشد کند، باید با توجه به نیاز سیستم، چگونگی مدیریت حجم فزاینده داده‌ها را نیز در نظر داشته باشد. به همین دلایل، برای تولید خودکار کاتالوگ فایل‌ها، فیلدهای شامل فراداده و اصالت[1] داده، مدل‌های جدیدی ارائه شده که یکی از این مدل‌ها آبخور داده است. آبراه[2] داده یک کاتالوگ داده هوشمند برای یافتن، فهم و حاکمیت داده در دریاچه داده به‌منظور سرعت بخشیدن تحلیل‌های سلف‌سرویس، ارائه می‌دهد. کاتالوگ‌های داده با داده‌های واقعی تکمیل می‌شوند، پس کشف خودکار روابط داده‌ها، مجموعه داده‌هایی را که منجربه دانش موضوعی می‌شوند، تشکیل داده، چابکی در حاکمیت داده را در پی دارد [۳۸].

چالش مهم دیگر در دریاچه داده، فقدان تعاریف مناسب برای مفاهیم آن است. همچنین به‌دلیل استفادهٔ چندین‌سالهٔ متخصصان از انبار داده به‌عنوان سیستم مدیریت داده، پذیرش سیستم جدید برای این هدف، نیاز به زمان بیشتری دارد و در نتیجه، این حوزه با کمبود متخصص مواجه است.

۳-۴- مقایسه دریاچه داده و انبار داده

در انبار داده، ابتدا اطلاعات ورودی پردازش و به یک ساختار هماهنگ تبدیل و داده‌ها آماده می‌شوند. سپس پردازش‌ها به‌صورت مستقیم و با هزینه کمتر بر روی داده‌ها انجام می‌گیرد؛ اما در دریاچه داده، اطلاعات به‌دلیل حجم بسیار زیاد و سرعت بالای ورود اطلاعات، به همان شکلی که وارد شده‌اند، ذخیره می‌شوند. سپس کاربران (متخصصان داده، تحلیلگران و کاربران عادی) با توجه به نیاز، بخش‌هایی از داده‌ها را سازماندهی و پردازش می‌کنند. این روند در شکل (۹) نمایش داده شده است. انبار داده و دریاچه داده، هرکدام دارای قوت‌ها و ضعف‌هایی هستند. جدول ۴ قوت و ضعف هرکدام را نشان می‌دهد. انبار داده برای نوشتن، طراحی شده است؛ یعنی پرس‌وجوهایی که قرار است بر روی داده‌های انبار داده اعمال شود از قبل مشخص بوده، با توجه به آن‌ها داده‌ها نوشته می‌شوند. در دریاچه داده، طراحی برای خواندن است؛ بدین معنا که نیازی به پیش‌بینی اینکه چه پرس‌وجوهایی باید در آینده پاسخ داده شود، وجود ندارد. دریاچه داده مکانی مناسب برای ماسه‌بازی[3] داده است.



شکل ۹- تفاوت انبار داده و دریاچه داده [۸]

به‌دلیل انجام پردازش‌ها و اعمال تغییرات لازم در انبار داده قبل از استفاده از آن، حاکمیت و کیفیت داده به‌خوبی پشتیبانی شده، کارایی قابلیت تکرار، ثابت خواهد بود. با اعمال ملاحظات امنیتی در انبار داده قابلیت اطمینان به این سیستم مدیریت داده افزایش می‌یابد. با توجه به اینکه انبار داده به بلوغ رسیده، متخصصان زیادی در این حوزه وجود دارند، در حالی که برای دریاچه داده به این صورت نیست. از آنجا که در انبار داده، ساختار اطلاعات از قبل مشخص می‌شود، این سیستم دارای انعطاف کمی در برابر تغییرات است. انبار داده از داده‌های ناساخت‌یافته و نیمه‌ساخت‌یافته پشتیبانی نکرده، پیاده‌سازی و اعمال تغییرات در آن، هزینه زیادی در پی خواهد داشت. با توجه به اینکه داده‌ها به همان شکلی که هستند، در دریاچه داده ذخیره می‌شوند، این سیستم منعطف بوده، با هزینه کمتری نسبت به انبار داده راه‌اندازی می‌شود. همان‌طور که در بخش‌های قبل بیان شد، کنترل، امنیت و حاکمیت داده در آن، همچنان از چالش‌های این سیستم است. در جدول ۵ تفاوت دو سیستم مدیریت داده، انبار داده و دریاچه داده بر اساس معیارهای مختلف آمده است. دریاچه

[1].lineage
[2].Waterline
[3].Sandboxing

داده بیشتر بر روی انعطاف و انبار داده، بر روی کنترل داده‌ها تأکید دارند.

جدول ۴- ضعف و قوت انبارداده و دریاچه داده

سیستم	ضعف	قوّت
انبار داده	• غیرمنعطف بودن • مهارت‌های متنوع (گزارش‌گیری، مدیر پایگاه) • عدم طراحی برای استفاده مجدد • اعتماد تک‌مدلی • مرحله آماده‌سازی داده‌ها در ابتدا • گزارش دادن، عدم پاسخ به سؤالات • عدم پشتیبانی از داده‌های ناساخت‌یافته • هزینه بالای پیاده‌سازی و تغییرات	• طراحی برای نوشتن • کنترل/امنیت • حاکمیت داده • کیفیت داده • کارایی تکرار اصالت • کارایی سازگاری • قابلیت اطمینان • داشتن مهارت متخصصان • راحتی استفاده • ادغام داده‌ها • تجزیه تحلیل کاربردی • بارگذاری یک‌باره و استفاده چندین‌باره
دریاچه داده	• مهارت متخصصان و توسعه‌دهندگان • قابلیت استفاده مجدد • کنترل/امنیت • حاکمیت داده • یکپارچه‌سازی دیرهنگام داده‌ها • ادغام و استفاده توسط کاربر نهایی	• طراحی برای خواندن • انعطاف‌پذیری • هزینه • داده‌های ناساخت‌یافته • توسعه‌پذیری بسیار بالا • پشتیبانی از برنامه‌نویسی • مناسب برای سه خصوصیت کلان داده: حجم، تنوع و سرعت تغییر داده

بااینکه دریاچه داده پس از انبار داده، برای رفع مشکلات آن در استفاده از داده‌های حجیم به وجود آمده، با توجه به بحث‌های مطرح‌شده نباید این نتیجه‌گیری شود که دریاچه داده، جایگزینی برای انبار داده است. باید به این نکته توجه کرد که هنوز برای برخی از نیازها از جمله تراکنش‌های دقیق مالی، داده‌های کسب‌وکار، موافقت سطح خدمات[1]، تحمل‌پذیری بالا در برابر خطا، ETL[2] قوی، اصالت و حاکمیت داده، انبار داده بهتر از دریاچه داده پاسخگو است. با توجه به جدول ۵ انبار داده از نظریه ACID که در بخش ۳-۱-۱ به‌طور کامل شرح داده شد و دریاچه داده، از نظریه BASE که در بخش ۳-۱-۲ بیان شد، پیروی می‌کند. در استفاده از سیستم مدیریت داده در سازمان باید با توجه به نیازمندی و مسائل مطرح‌شده در این مقاله، سیستم مناسب را انتخاب کرد.

گاه در یک سازمان با توجه به نیاز، باید از دریاچه داده و انبار داده در کنار یکدیگر بهره برد تا بتوان پاسخگوی تمام نیازهای سازمان بود. در این حالت، برای تحلیل به‌صورت بی‌درنگ و تصمیم‌گیری سریع نیاز به در دسترس قرار دادن داده‌ها به‌صورت بی‌درنگ وجود دارد. در استفاده از انبار داده به‌عنوان تنها سیستم مدیریت داده، راهکارهای مختلفی برای ارائه آن به‌صورت بی‌درنگ وجود دارد که در بخش کارهای مرتبط به آن پرداخته شد. در ادامه، به معرفی یک معماری ترکیبی برای حل مشکلات هرکدام از دو راه‌حل فوق خواهیم پرداخت.

این معماری، سیستم مدیریت داده بی‌درنگی است که می‌تواند با توجه به نیاز، قابلیت‌های انبار داده و دریاچه داده را با هم ترکیب کرده، به‌صورت بی‌درنگ داده‌های موردنیاز را در اختیار تحلیلگران قرار دهد. با توجه به ساختار آن که مشابه یک آبگیر است، این نام برای آن انتخاب شده است.

۴- سیستم پیشنهادی

ایده سیستم پیشنهادی، جداسازی داده‌های دریاچه داده و انبار داده و درعین‌حال، جداکردن داده‌های جدید از داده‌های قدیمی است. در این ایده، سربار هم‌زمانی بارگذاری داده‌ها و پاسخگویی به درخواست از بین می‌رود که منجربه بی‌درنگی می‌شود. در ادامه، نحوه ذخیره‌سازی و پردازش پرس‌وجو در آبگیر داده، بیان است.

۴-۱- ذخیره‌سازی در آبگیر داده

همان‌طور که در شکل (۱۰) مشاهده می‌شود، ابتدا داده‌ها از منابع داده استخراج می‌شوند (فلش ۱). منابع داده، شامل اجزای شبکه، GPS، سیستم‌های موجود در سازمان یا سایر ابزارهای تولید داده هستند. این داده‌ها به‌صورت ساخت‌یافته،

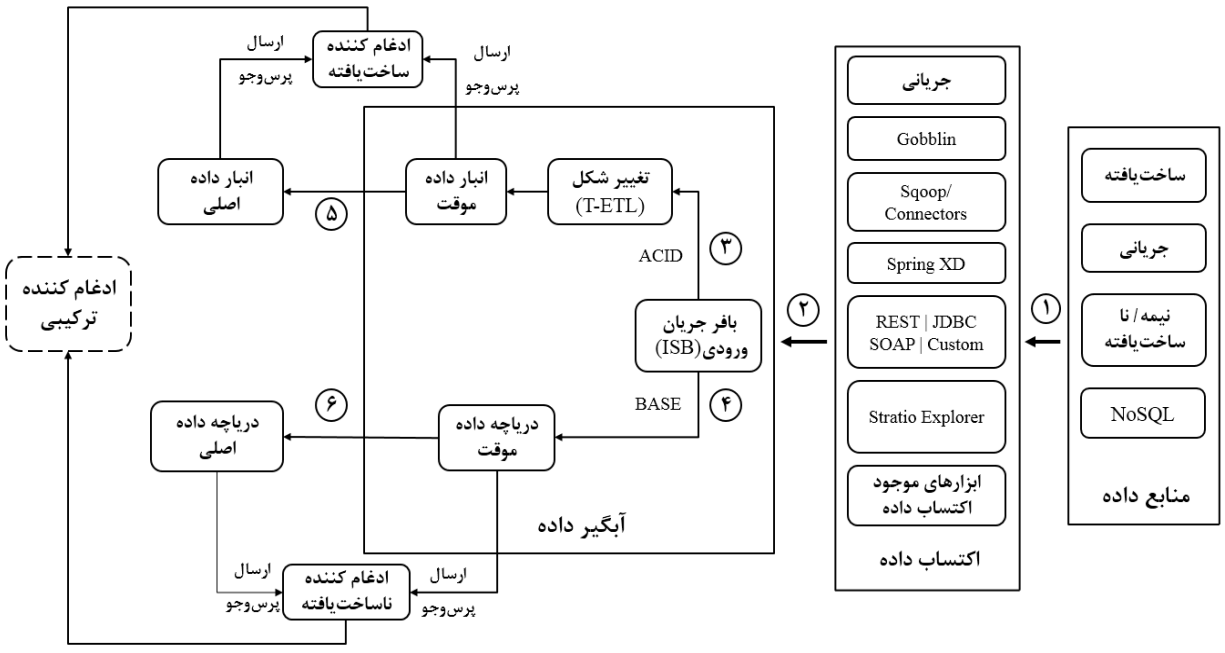
[1].Service-Level Agreement (SLA)

[2].Extraction, Transform, Load

نیمه‌ساخت‌یافته، ناساخت‌یافته، جریانی و NoSQL هستند. این تنوع در داده‌ها موجب استفاده از ابزارهای اکتساب داده متنوعی در برنامه می‌شود. در شکل (۱۰) چند نمونه از این ابزارها مشاهده می‌شود.

جدول ۵. مقایسه انبار داده و دریاچه داده

معیار	انبار داده	دریاچه داده
داده	ساخت‌یافته/ پردازش‌شده/ داده‌های تمیز و تجمیع‌شده	ساخت‌یافته/ نیمه‌ساخت‌یافته/ غیرساخت‌یافته، داده خام
پردازش	طراحی برای نوشتن	طراحی برای خواندن
ذخیره‌سازی	برای مقدار حجیم داده‌ها هزینه‌بر	طراحی‌شده برای هزینه پایین ذخیره‌سازی
بارکاری	پشتیبانی از کاربران تعاملی بیشتر، پردازش دسته‌ای داده‌ها در حجم بالا	قابلیت‌های مدیریت بارکاری پیشرفته، پردازش دسته‌ای
چابکی	کند، پیکربندی ثابت	چابک، پیکربندی و بازسازی مبتنی برنیاز
پیچیدگی	اتصالات پیچیده	پردازش پیچیده
امنیت	بلوغ	در حال بلوغ
کاربران	کسب‌وکار حرفه‌ای	متخصصان داده، تحلیلگران. کاربران عملیاتی
حجم داده	بزرگ (ترابایت)	بسیار حجیم (پتابایت)
روش دسترسی	SQL، روش جست‌وجو، ابزارهای هوش تجاری	NoSQL، روش پویش
توسعه‌پذیری	توسعه عمودی	توسعه افقی
سخت‌افزار	سخت‌افزار یا لوازم مخصوص	سخت‌افزار عادی
نظریه	ACID	BASE



شکل ۱۰- معماری آبگیر داده

داده‌های اکتساب‌شده، به بافر جریان ورودی[1] تحویل داده می‌شوند (فلش۲). پرداختن به چگونگی واکشی داده‌ها از منابع، خارج از محدوده این مقاله است. روش‌های مختلفی برای این کار در مقاله ژو و همکاران [۱۵]، شرکت اوراکل[2][۳۷]، آناندان و همکارانش [۴۰] و کیا و همکارانش[۴۱] ارائه شده است. فرض ما بر استفاده از راهکار مناسب با توجه به نیاز و نوع منبع داده است.

پس از استخراج، داده‌ها در بافر جریان ورودی ذخیره می‌شوند. نرخ موردنظر از یکپارچه‌سازی داده‌ها از منبع داده و بافر جریان ورودی، قابل پیکربندی است. بافر جریان ورودی داده‌ها را به‌صورت مینی‌دسته‌ای[3] تقسیم‌بندی کرده، حداکثر اندازه داده‌ها قبل از ادغام، برای جلوگیری از سربار در سیستم به‌صورت خودکار راه‌اندازی می‌شود.

بافر جریان ورودی می‌تواند به‌گونه‌ای پیکربندی شود که داده‌ها را در دیسک و حافظه ذخیره کند. این ذخیره‌سازی به حجم داده‌ها و معیارهای دیگر، همچون نیازمندی ماندگاری برای داده‌های ورودی بستگی دارد. در بافر جریان ورودی این امکان دیده شده که با توجه به نوع داده‌های ورودی و میزان انطباق نیاز به آن‌ها، به انبار داده یا دریاچه داده که در بخش‌های قبل مفصلاً بحث شد، انتقال دهد.

همان‌گونه که گفته شد، انبار داده پاسخگوی تمام نیازهای یک سیستم مدیریت داده نیست؛ به همین دلیل دریاچه داده به وجود آمد. درواقع، دریاچه داده جایگزینی برای انبار داده نیست و در برخی نیازها از جمله تراکنش‌های مالی که از نظریه ACID پیروی می‌کنند، استفاده از انبار داده همچنان پاسخگوی نیاز است.

برای ذخیره‌سازی داده‌ها در آبگیر داده یک انبار داده و یک دریاچه داده موقت وجود دارد که داده‌های جدید به‌صورت موقت در آن‌ها ذخیره می‌شوند. سپس در بازه زمانی که توسط مدیر داده مشخص می‌شود، به انبار داده یا دریاچه داده اصلی منتقل می‌شوند. اگر نیاز ذخیره‌سازی داده‌ها مطابق با نظریه ACID (گزینش C+A در نظریه CAP) بود، داده‌ها باید به انبار داده موقت و درغیراین‌صورت، یعنی اگر مطابق با نظریه BASE (گزینش C+P یا P+A در نظریه CAP) بود، به دریاچه داده انتقال می‌یابند. همان‌طور که در شکل (۱۰) مشاهده می‌شود، اگر نیاز به انتقال داده‌های جدید به انبار داده وجود داشت، باید ابتدا فاز تغییرشکل بر روی داده‌های ورودی انجام شود. به همین دلیل داده‌ها از بافر جریان ورودی به فاز تغییرشکل (فلش۳) و سپس به انبار داده موقت منتقل می‌شوند. در دریاچه داده به‌دلیل ذخیره‌سازی داده‌ها به همان شکل ورودی، نیازی به اعمال تغییرشکل در داده‌های ورودی به دریاچه داده موقت وجود ندارد (فلش۴)؛ پس داده‌ها به همان شکل به دریاچه داده موقت وارد می‌شوند.

ETL یک بخش مستقل بوده که مسئول تغییرشکل است. با توجه به مستقل بودن آن، فرض می‌کنیم که می‌توان آن را به روش‌های مختلف اجرا کرد. با هدف بهینه‌سازی این فرآیند، بحث‌هایی از جمله در پژوهش واس و همکارانش [۴۲] مطرح شده است.

داده‌های موجود در انبار داده و دریاچه داده موقت، داده‌هایی هستند که اخیراً به سیستم وارد شده‌اند و داده‌های قدیمی در انبار داده و دریاچه داده اصلی قرار دارند. دو واحد انبار داده قدیمی و جدید، مستقل از یکدیگر عمل می‌کنند. همچنین این شرایط برای دریاچه داده نیز وجود دارد. در مقاله فریرا و فورتادو [۱۸] جداسازی دو انبار داده موقت و اصلی مستقل که در آنجا با عنوان پویا و ایستا معرفی شده‌اند، برای رسیدن به بی‌درنگی، انجام شده است. همان‌گونه که در این مقاله بیان شده، شمای انبار داده موقت، مشابه انبار داده اصلی است و داده‌هایی که در انبار داده موقت ذخیره شده‌اند، ساده بوده، هیچ‌گونه نمایه یا کار اضافه‌ای برای رسیدن به بی‌درنگی بر روی آن‌ها انجام نمی‌شود. با توجه به اینکه در این مقاله، جداسازی دو واحد انبار داده موقت و انبار داده اصلی شرح داده شده، ما عملکرد این قسمت‌ها را مشابه آن مقاله در نظر گرفته و از بیان کردن عملکرد آن‌ها صرف‌نظر می‌کنیم. در ادامه، معماری و عملکرد سایر قسمت‌ها را شرح می‌دهیم.

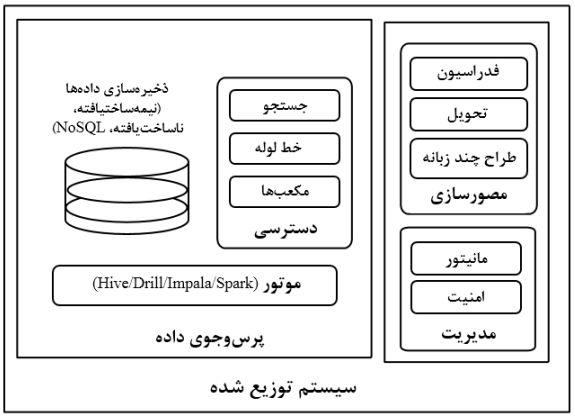
۴-۱-۱-دریاچه داده موقت

این بخش شامل داده‌هایی است که از نظریه BASE پیروی می‌کنند. این قسمت شامل داده‌های NoSQL، جریانی و گاهی ناساخت‌یافته یا نیمه‌ساخت‌یافته بوده که دارای حجم

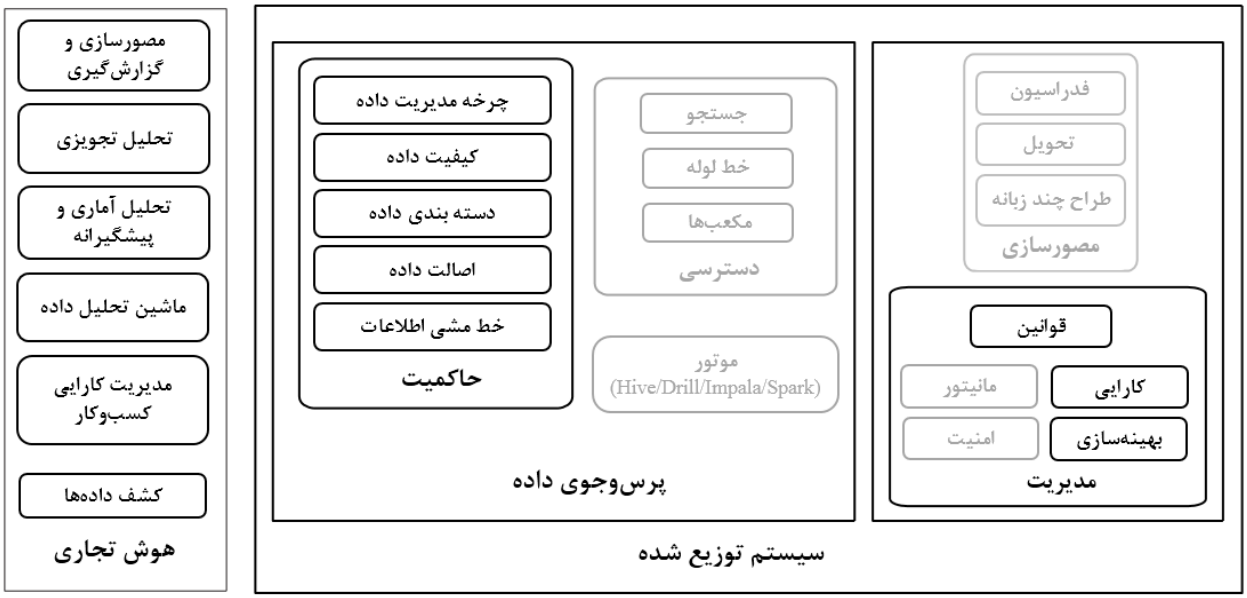
[1].Input Stream Buffer (ISB)
[2].Oracle
[3].Mini-batch

زیادی است و با نرخ بالایی به سیستم وارد می‌شوند. پس برای این قسمت باید حجم و قدرت پردازشی بالاتری نسبت به انبار داده موقت در نظر گرفته شود تا کارایی سیستم دچار افت نشود. بهتر است این قسمت بر روی یک سیستم واحد قرار گیرد، اما در صورت بالابودن حجم داده‌ها، می‌توان از سیستم توزیع شده استفاده کرد.

برای رسیدن به عملکرد بالا در پاسخگویی به پرس‌وجوها، در دریاچه داده موقت، از اعمال قوانینی که موجب از دست دادن زمان شود، پرهیز کرده، آن‌ها را به زمان انتقال به دریاچه اصلی موکول می‌کنیم (فلش۶).



شکل ۱۱- معماری دریاچه داده موقت



شکل ۱۲- معماری دریاچه داده اصلی

نکته‌ای که در این قسمت باید رعایت کرد، مشابه بودن طراحی دریاچه داده موقت با اصلی است تا هنگام انتقال، زمانی برای تغییر طراحی داده‌ها اختصاص نیابد؛ یعنی این قسمت نمونهٔ کوچک‌شدهٔ دریاچه داده اصلی است. با کاهش هزینه ذخیره‌سازی می‌توان داده‌های این قسمت را برای رسیدن به بی‌درنگی در ذخیره‌سازی، بازیابی و تحلیل با توجه به نیاز بر روی حافظه حالت‌جامد[1] ذخیره کرد. گاهی نیاز است تحلیل بر روی داده‌های جدید صورت پذیرد؛ به این دلیل باید امکان مصورسازی[2] در دریاچه داده موقت دیده شود. در شکل (۱۱) معماری دریاچه داده موقت مشاهده می‌شود. در صورت نیاز، این سیستم بر روی سیستم توزیع‌شده قرار می‌گیرد و برای راحتی و سرعت در ارتباط موتور پردازشی باید در آن در نظر گرفته شود. مکانیسم‌های دسترسی به داده‌های ذخیره‌شده نیز باید تعریف شوند.

برخی از کارهای امنیتی در این مرحله انجام می‌شود و دیگر عملیات‌ها به زمانی موکول می‌شود که داده‌ها به دریاچه داده اصلی منتقل می‌شوند.

۴-۱-۲- دریاچه داده اصلی

معماری دریاچه داده در شکل (۱۲) مشاهده می‌شود. موارد کم‌رنگ قبلاً در قسمت دریاچه داده موقت اعمال شده‌اند، اما

[1].Solid-State Drive (SSD)

[2].Virtualization

باید توجه داشت که میزان اعمال آن‌ها به نیازهای سازمان بستگی دارد. به‌عنوان‌مثال، با انتقال داده‌ها از دریاچه موقت به اصلی، علاوه بر مانیتور و امنیت که در مراحل قبل بر روی داده‌ها اعمال شده بود، قوانین مدیریتی از جمله بهینه‌سازی، کارایی و قواعد[1] اعمال می‌گردد. البته میزان اعمال امنیت بر روی داده‌ها در دریاچه داده موقت و اصلی، به نیازهای سازمان بستگی دارد.

همان‌طور که در بخش‌های قبل اشاره شد، حاکمیت داده در دریاچه داده از اهمیت بسیار زیادی برخوردار است، پس در این معماری، جزء دیگر دریاچه داده اصلی، حاکمیت داده است. نگهداری کیفیت داده، یکی از مهم‌ترین عوامل این فاز است و تبعیت از مقررات کیفیت داده[2] برای جلوگیری از تبدیل دریاچه داده به مرداب، باید انجام شود.

دسته‌بندی داده‌های مرتبط منجربه راحتی دسترسی به داده‌های مرتبط و در نتیجه، افزایش سرعت در پاسخگویی به پرس‌وجوها خواهد بود. چرخه مدیریت داده و تعریف سیاست اطلاعات[3] با توجه به سطح نیاز، یکی دیگر از کارهای انجام‌شده در این بخش است.

نهایتاً داده‌ها برای هوش تجاری[4] آماده می‌شوند و می‌توان از آن‌ها بهره‌برداری کرد. پرس‌وجوها در آبگیر داده، انواع مختلفی دارند. در ادامه، انواع و بهترین راهکار برای پاسخ به آن‌ها شرح داده می‌شود.

۴-۲- پرس‌وجو در آبگیر داده

در آبگیر داده چند نوع مختلف درخواست داده به‌سمت سیستم وارد می‌شود. با توجه به جداسازی داده‌های قدیمی و جدید، برخی از تحلیل‌ها صرفاً نیاز به داده‌های قدیمی داشته، برخی دیگر به داده‌های جدید نیاز دارند. برای پاسخگویی به هرکدام، با توجه به اینکه از داده‌های انبار داده یا دریاچه داده استفاده می‌کنند باید پرس‌وجو بر روی هرکدام اعمال شود؛ یعنی پرس‌وجوهای مربوط به داده‌های جدید بر روی انبار داده و دریاچه داده موقت، پرس‌وجوی داده‌های قدیمی بر روی دریاچه داده و انبار داده قدیمی اعمال می‌شوند. اگر نیاز به اجماع پاسخ‌ها باشد، برای داده‌های قدیمی و جدید پاسخ پرس‌وجوها با یکدیگر ادغام می‌شوند. برای جلوگیری از پیچیدگی تصویر در شکل (۱۰) این موارد ترسیم نشده است. در آبگیر داده، داده‌های مالی و داده‌هایی که از نظریه ACID پیروی می‌کنند، به انبار داده موقت و بعد به انبار داده اصلی منتقل می‌شوند. سپس پرس‌وجوهایی که شامل داده‌های مالی هستند، به انبار داده اصلی یا موقت اعمال می‌شوند.

در پاسخ پرس‌وجوهایی که در آن‌ها به داده‌های قدیم و جدید نیاز است، ادغام پاسخ‌ها جای بحث دارد. در یک حالت پرس‌وجو:

- نیاز به داده‌های قدیم و جدید انبار داده موقت و اصلی دارد (ادغام‌کننده ساخت‌یافته در شکل ۱۰).
- نیاز به داده‌های قدیم و جدید دریاچه داده موقت و اصلی دارد (ادغام‌کننده ناساخت‌یافته در شکل ۱۰).
- نیاز به داده‌های قدیم و جدید کل سیستم‌های ذخیره‌سازی، انبارهای داده و دریاچه‌های داده، دارد (ادغام‌کننده ترکیبی در شکل ۱۰).

ادغام‌کننده ترکیبی، داده‌ها را با یکدیگر ادغام نمی‌کند. درواقع، این قسمت نشان دادن آمارهای کلی از سیستم است؛ زیرا در انبار داده، داده‌های ساخت‌یافته قرار داشته، ولی در دریاچه داده، داده‌ها ساخت‌یافته نیستند. دو حالت ادغام‌کننده ساخت‌یافته و ادغام‌کننده ناساخت‌یافته در ادامه شرح داده شده، یک نمونه از اجرای پرس‌وجوهایی که در آن ادغام‌کننده ترکیبی کاربرد دارد، بیان می‌شود.

۴-۲-۱- ادغام‌کننده ساخت‌یافته

داده‌های قدیمی که حجم بالایی دارند، در انبار داده اصلی ذخیره شده که این قسمت جدا از انبار داده موقت اجرا می‌شود. در انبار داده اصلی به‌دلیل وجود نمایه‌ها و دید اصلاح‌شده[5]، سربار چندانی در پردازش پرس‌وجوها وجود نخواهد داشت. ادغام انبار داده موقت به انبار داده اصلی از الگوی بارگذاری آفلاین انبار داده سنتی پیروی می‌کنند. این یعنی بازه زمانی آنلاین تعریف شده است. نمایه‌ها ممکن است تنها پس از کامل شدن بارگذاری رها یا بازسازی‌شده، بارگذاری کامل و بازسازی دید اصلاح‌شده انجام شود. پس از

[1].Provisions
[2].Data quality regulation adherence
[3].information policy
[4].Business Intelligence
[5].Materialized view

ادغام، انبار داده موقت خالی و آماده پذیرش مجدد داده‌ها می‌شود. با این رویکرد، انبار داده موقت شامل داده‌های به‌روز و پاسخگویی آن به پرس‌وجوها به‌صورت بی‌درنگ خواهد بود؛ اما در همان زمان در انبار داده اصلی که شامل حجم زیادی از داده‌های قدیمی است، به پرس‌وجوها پاسخ داده می‌شود. همان‌طور که قبلاً گفته شد، طراحی انبار داده موقت و انبار داده اصلی با هم مشابهت دارند؛ اگرچه، انبار داده موقت به بارگذاری سریع داده‌های جدید اختصاص دارد. این قابلیت با از بین بردن عواملی به دست می‌آید که در انبار داده اصلی موجب بارگذاری سنگین می‌شوند.

در انبار داده موقت هیچ‌گونه نمایه و محدودیتی و هیچ نوع خلاصه‌سازی یا دید اصلاح‌شده‌ای وجود ندارد. این اقدامات بدون کاهش چشمگیر کارایی میسر است؛ زیرا به‌طور منطقی، حجم داده‌های اخیر کوچک است. بنابراین زمان پردازش چشمگیری هنگام پردازش پرس‌وجوها زمانی که انبار داده موقت برای اجرای پرس‌وجوهای قسمت مربوط به خود به اشتراک می‌گذارند، رخ نمی‌دهد. بااین‌حال، روند پردازش پرس‌وجو در انبار داده موقت بدون بازنویسی درخواست‌ها است؛ زیرا شمای انبار داده موقت مشابه انبار داده اصلی است، به‌جز جمع‌آوری داده‌های محقق‌شده. در این حالت در انبار داده موقت، داده‌های اصلاح‌شده توسط دیدهای معمولی پایگاه داده جایگزین می‌شوند. در نتیجه، درخواست‌های مربوط به دیدهای اصلاح‌شده به‌سادگی توسط دیدهای متناظرشده (غیراصلاح‌شده) در انبار داده موقت پاسخ داده می‌شوند. با استفاده از این معماری، امکان تضمین نرخ ثابت ادغام داده‌های جدید و داده‌های قدیمی بدون کاهش چشمگیر هزینه کارایی فراهم می‌شود.

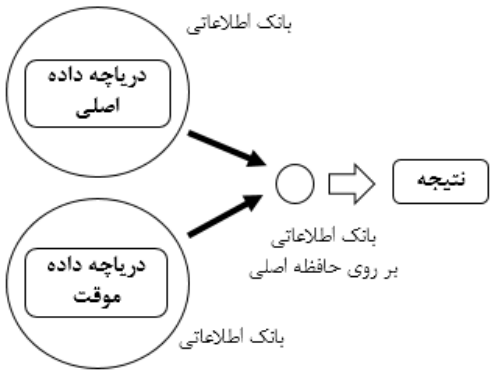
برای ادغام پاسخ‌های قدیمی و جدید در ادغام‌کننده ساخت‌یافته سه حالت وجود دارد:

- ادغام در مکان سوم: در این حالت پاسخ‌های پرس‌وجو از انبارهای داده قدیمی و جدید به مکانی مشخص انتقال‌یافته، با یکدیگر ادغام و پاسخ به کاربر ارائه می‌شود.
- ادغام همراه با انتقال: در این حالت، پاسخ‌های پرس‌وجو داده‌های جدید به مکان داده‌های قدیم، یا بالعکس انتقال می‌یابند. به‌دلیل کم بودن حجم داده‌های جدید نسبت به قدیمی، انتقال داده‌های جدید به مکان داده‌های قدیمی، دارای کارای بیشتری است.
- ادغام در مکان خاص بدون انتقال: در این حالت، یک پایگاه داده که دارای سرعت بالایی است، برای ادغام پاسخ‌ها در نظر گرفته می‌شود.

با توجه به نتایج ارزیابی در تحقیق انجام‌شده [۱۸]، ادغام در مکان خاص بدون انتقال، بهترین کارایی را داراست. فقط باید توجه کرد که در این حالت، بهره‌گیری از بانک اطلاعاتی که داده‌ها را بر روی حافظه اصلی ذخیره می‌کند، در افزایش سرعت پاسخگویی مؤثر است. در ادامه، نحوه کارکرد ادغام دریاچه داده اصلی و موقت آمده است.

۴-۲-۲- ادغام کننده ناساخت‌یافته

این بخش پاسخ پرس‌وجوهایی را که داده‌های آن در دریاچه داده اصلی و موقت هستند، با یکدیگر ادغام کرده، پاسخ را به کاربر نشان می‌دهد. برای این ادغام‌کننده نیز سه حالت ممکن وجود دارد. مشابه بخش قبلی، داده‌های جدید به دلیل حجم کمتر نسبت به داده‌های قدیمی به آنجا انتقال پیدا کنند. در حالت دیگر، داده‌های قدیمی و جدید در یک سیستم قرار بگیرند. این دو حالت به‌دلیل حجم بالای اطلاعات در دریاچه داده و سرعت انتقال آن‌ها از منابع داده، روش‌های مناسبی نیستند؛ پس بهتر است از حالت سوم که در آن یک بانک اطلاعاتی استفاده شود که بر روی حافظه اصلی، اطلاعات را ذخیره می‌کند. ساختار موردنظر در شکل (۱۳) مشاهده می‌شود.



شکل ۱۳- ادغام کننده ناساخت‌یافته

در دریاچه داده، چهار نوع بانک اطلاعاتی NoSQL می‌تواند

استفاده شود. این پایگاه‌های داده، شامل کلید-مقدار[1]، ستون گسترده[2]، گراف‌محور و سندگرا[3] هستند [۴۳]. یک نوع نیز حالت ترکیبی از این چهار بانک اطلاعاتی است. با توجه به نوع نیاز، می‌توان از هرکدام از این بانک‌های اطلاعاتی استفاده کرد که پرداختن به این موضوع در محدوده این مقاله نیست. در ادغام پاسخ‌ها باید نوع بانک‌های اطلاعاتی را در نظر گرفت. در انتخاب بانک‌های اطلاعاتی باید بهترین بانک اطلاعاتی را با توجه به نیازهای سازمان انتخاب کرد تا سیستم بی‌درنگ با مشکل مواجه نشود.

۴-۳- ادغام‌کننده ترکیبی

همان‌طور که گفته شد، این قسمت بیشتر به گزارش‌گیری پرداخته، پاسخ‌ها را با یکدیگر ادغام نمی‌کند. این قسمت زمانی که نیاز به برقراری ارتباط بین داده‌های مالی و داده‌های Nosql است، کاربرد دارد. به‌عنوان‌مثال، پرس‌وجویی بدین‌صورت نیاز است: «میانگین درآمد افرادی که در یک گروه در یک شبکه اجتماعی وجود دارند.»

همان‌طور که گفته شد، با توجه به تنوع، حجم و سرعت داده‌هایی که در شبکه‌های اجتماعی منتقل می‌شوند، الزام به استفاده از دریاچه داده برای مدیریت این نوع داده‌ها وجود دارد. از طرفی داده‌های مالی به‌دلیل اهمیت بالا در پشتیبانی از نظریه ACID در انبار داده ذخیره می‌شوند. این پرس‌وجو در سه مرحله پاسخ داده می‌شود:

۱. واکشی گروه‌های جدید و قدیمی تشکیل‌شده از دریاچه داده موقت و اصلی و ادغام آن‌ها توسط ادغام‌کننده ساخت‌یافته؛
۲. واکشی درآمد هریک از اعضای گروه موردنظر از انبار داده موقت و اصلی و ادغام پاسخ توسط ادغام‌کننده ناساخت‌یافته؛
۳. ارتباط دادن پاسخ‌های ادغام‌کننده‌های یک و دو و ارائه پاسخ موردنظر به کاربر.

دلیل واکشی اطلاعات از انبار داده و دریاچه داده موقت، امکان ورود اطلاعات مالی کاربر جدید و تشکیل گروه جدید توسط کاربران است. درواقع، گاهی اطلاعات گروه تشکیل‌شده در دریاچه موقت بوده، هنوز به دریاچه داده اصلی انتقال پیدا نکرده است. این حالت برای داده‌های مالی نیز ممکن است.

۴-۴- افزایش بهره‌وری استفاده از آبگیر داده

در معماری پیشنهادی، با توجه به در نظر گرفتن دو سیستم مدیریت داده دریاچه و انباره داده و همچنین جداسازی داده‌های موقت از اصلی، طبیعتاً امکانات و دانش سخت‌افزاری و نرم‌افزاری بیشتری نسبت به قبل نیاز است. درصورتی‌که آبگیر داده و در راستای آن، انبار داده و دریاچه داده اصلی به‌خوبی طراحی شوند، نگرانی از بابت افزایش حجم داده‌ها (عمیق شدن دریاچه) و در دسترس بودن آن‌ها کاهش می‌یابد. درواقع، اگر طراحی به‌درستی انجام شود، هنگام افزایش حجم داده‌ها، سرعت و کارایی با افت چشمگیری همراه نخواهد بود. برای رسیدن به این هدف، طراح آبگیر داده باید برای قسمت‌های مختلف به‌طور مؤثر فراداده تعریف کند. این کار منجربه درک تدریجی کاربران از داده‌های موجود در آبگیر داده می‌شود. طراح آبگیر داده باید راه‌های مؤثری برای ورود اطلاعات به آبگیر داده طراحی کند، به‌طوری‌که داده‌ها همواره حالت تازگی[4] خود را حفظ کنند. مسئله دیگر، دسته‌بندی موضوعی داده‌ها است؛ این کار منجربه واکشی مؤثر داده‌ها توسط کاربران جهت تحلیل است. پرداختن به این موضوعات، به محققان به‌عنوان کارهای آینده پیشنهاد می‌شود.

۵- نتیجه‌گیری

انبار داده و دریاچه داده، دارای چالش‌ها و ضعف‌ها و قوت‌های مختلفی هستند که در این مقاله به آن‌ها پرداخته شد. با مقایسه انبار داده و دریاچه داده نتیجه‌گیری می‌شود که دریاچه داده، جایگزینی برای انبار داده نیست و انبار داده همچنان کاربردهای خاص خود را مخصوصاً در داده‌های حساس همچون داده‌های مالی دارد. آبگیر داده، داده‌های جدید را با توجه به نوع و نیاز به آن‌ها در انبار داده یا دریاچه داده موقت ذخیره می‌کند. داده‌های جدید در زمان تعیین‌شده توسط مدیر داده به دریاچه داده یا انبار داده اصلی انتقال می‌یابند. در مقاله، معماری ذخیره‌سازی انبار داده و دریاچه داده موقت و اصلی بیان شده است. در حالات ذخیره‌سازی موقت، داده‌ها به‌گونه‌ای ذخیره می‌شوند که پاسخ پرس‌وجوهای این قسمت سریع داده شود. درواقع، روی داده‌ها

[1] Key-Value
[2] Wide Column
[3] Document Oriented
[4] refreshed

کمتر عملیات‌هایی همچون نمایه‌گذاری و بهینه‌سازی انجام می‌شود. برای پاسخگویی به پرس‌وجوها در این معماری سه ادغام‌کننده در نظر گرفته شده است. در ادغام‌کننده اول و دوم به‌ترتیب، داده‌های انبار داده و دریاچه داده موقت و اصلی با یکدیگر ادغام می‌شوند. برای ادغام پاسخ‌ها در این حالت، بهترین راهکار استفاده از یک بانک اطلاعاتی مبتنی بر حافظه اصلی است. ادغام‌کننده سوم نیز که بیشتر جنبه گزارش‌گیری دارد، پاسخ دو ادغام‌کننده قبل را با یکدیگر ترکیب کرده، به کاربر ارائه می‌دهد. در پایان، راهکارها و مسائلی جهت بهبود کارایی استفاده از آبگیر داده بیان شده است.

منابع

[1] 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].

[2] L. Tay (2013), "Inside eBay’s 90PB data warehouse" [Online], Available: http://www.itnews.com.au/news/inside-ebay8217s-90pb-data-warehouse-342615, [Accessed: 27 February 2016]

[۳] س. کشوری، م. نقوی، س. کشوری، «معرفی، بررسی و مقایسه سیستم فایل‌های توزیع‌شده»، اولین همایش ملی فناوری‌های نوین رایانه و توسعه پایدار، تهران، ۱۳۹۴.

[4] A.B. Angadi, A.B. Angadi, K.C. Gull. (2013), "Growth of New Databases & Analysis of NOSQL Datastores", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, 2013, pp. 06-20.

[5] E.A. Brewer, “Towards Robust Distributed Systems”, Portland, Oregon, J. – Keynote at the ACM Symposium on Principles of Distributed Computing (PODC) on, 2000, PP. 07-19.

[6] M. Jacobsohn and M. Delurey, "HOW THE DATA LAKE WORKS", 2014, Available:https://www.boozallen.com/content/dam/boozallen/documents/Data_Lake.pdf, [Accessed: 07 August 2016].

[7] M. Ali-ud-din Khan, M. Fahim Uddin and N. Gupta, "Seven V's of Big Data understanding Big Data to extract value, American Society for Engineering Education (ASEE Zone 1) ,Bridgeport, CT 2014, PP. 1-5.

[8] M. Fowler, "DataLake"[Online], 2015, Available:http://martinfowler.com/bliki/DataLake.html, [Accessed: 07 August 2016].

[9] J. Langseth, "Real-Time Data Warehousing: Challenges and Solutions", DSSResources.COM, [Online], 2004, Available: http://dssresources.com/papers/features/langseth/langseth02082004.html, [Accessed: 07 August 2016].

[10] R. Santos and J.Bernardino, "Real-time data warehouse loading methodology", International Database Engineering and Applications Symposium (IDEAS), New York, NY, USA, 2008, PP. 49-58.

[11] T. Jain, S. Rajasree and S. Saluja, "Refreshing datawarehouse in near real-time", International Journal of Computer Applications, May, Vol. 46, No. 18, 2012, pp.0975–8887.

[12] J. Zuters, "Near real-time data warehousing with multi-stage trickle and flip", Perspectivesin Business Informatics Research, Vol. 90 of Lecture Notes in Business InformationProcessing, 2011, pp.73–82.

[13] M. Nguyen and A. Tjoav, “Zero-latency data warehousing for heterogeneous data sources and continuous data streams”, Fifth International Conference on Information and Web-based Applications and Services, Austrian Computer Society (OCG), 2003, PP 167 - 176.

[14] L. Golab and T. Johnson, “Data stream warehousing”, IEEE 30th International Conference on Data Engineering, Chicago, IL, 2014, PP. 1290-1293.

[15] Y. Zhu, L. an and S. Liu, “Data updating and query in real-time data warehouse system”, Computer Science and Software Engineering, International Conference on (Volume: 5), 2008, PP. 1295 - 1297.

[16] M. Obali, Z. Erdem and A.K. Görür, “A real time data warehouse approach for data processing”, Signal Processing and Communications Applications Conference (SIU),Haspolat, 2013, PP. 1 - 4.

[17] P. Vassiliadis and A. Simitsis, "Near real time ETL", in Springer journal Annals of Information Systems, Vol. 3, Special issue on New Trends in Data Warehousing and Data Analysis, Springer, 2008, pp. 1-31.

[18] N. Ferreira and P. Furtado, "Real-time data warehouse: a solution and evaluation", International Journal of Business Intelligence and Data Mining, Volume 8, Issue 3, 2013, PP. 244-263.

[19] H. Fang, “managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem”, Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2015 IEEE International Conference on, Shenyang, PP. 820 - 824.

[20] R. Hai, S. Geisler and C. Quix, “Constance: An Intelligent Data Lake System”, SIGMOD '16 Proceedings of the 2016 International Conference on Management of Data, New York, NY, USA, 2016, PP. 2097 - 2100.

[21] C. Walker and H. Alrehamy, “Personal Data Lake with Data Gravity Pull, Big Data and Cloud Computing (BDCloud)”, 2015 IEEE Fifth International Conference on, Dalian, PP. 160 - 167.

[22] Ch. Xie, Ch. Su, C. Littley and et. al., “High-performance ACID via modular concurrency control”, SOSP '15 Proceedings of the 25th Symposium on Operating Systems Principles, New York, USA, 2015, PP. 279-294.

[23] Ch. Kong, Sh. China, M. Gao and et. al., “ACID Encountering the CAP Theorem: Two Bank Case Studies”, 2015 12th Web Information System and Application Conference (WISA), Jinan, PP. 235 - 240.

[24] D.G. Chandra, "BASE analysis of NoSQL database", Future Generation Computer Systems, Volume 52, 2015, pp. 13–21.

[25] E. Brewer, "CAP twelve years later: How the "rules" have changed", Computer, vol. 45, 2012, pp. 23 - 29.

[26] R.M. Elsa Estrada-Guzman and L. Gómez, “NoSQL method for the metric analysis of Smart Cities”, presented at the IEEE Guadalajara Metrics for Smart Cities Working Group, 2015.

[۲۷] س. کشوری، ح. صابری و س. کشوری، «نقش نظریه CAP و هم‌زیستی مسالمت‌آمیز در انتخاب بانک‌های اطلاعاتی»، سومین کنفرانس بین‌المللی پژوهش‌های کاربردی در مهندسی کامپیوتر و فناوری اطلاعات، تهران، دانشگاه صنعتی مالک اشتر، ۱۳۹۴.

[28] S. Gilbert and N. Lynch, "Perspectives on the CAP Theorem", Computer, Volume: 45, Issue: 2, 2012, pp. 30-36.

[29] S. Prasad and M.S. Nunifar Sha, “NextGen data persistence pattern in healthcare: Polyglot persistence”, presented at the Computing, Communications and Networking Technologies (ICCCNT), Fourth International Conference on, Tiruchengode, 2013, PP. 1-8.

[30] M. Rifaie, K. Kianmehr, R. Alhajj and M.J. Ridley (2008), “Data warehouse architecture and design”, Information Reuse and Integration, 2008. IRI 2008. IEEE International Conference on, Las Vegas, NV, USA, 2008, PP. 58 - 63.

[31] R. Kimball and M. Ross, The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, Wiley Publishing, 2013.

[32] W.H. Inmon, K. Krishnan, Building the Unstructured Data Warehouse, Technics Publications, LLC , USA, 2011.

[33] Ch. CAMPBELL, "Top Five Differences between Data Lakes and Data Warehouses", 2015, Available: https://www.blue-granite.com/blog/bid/402596/Top-Five-Differences-between-Data-Lakes-and-Data-Warehouses, [Accessed: 14 August 2016].

[34] M. Refaat, Data Preparation for Data Mining Using SAS, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2007.

[35] G. Goth, "The Data Lake Concept Is Maturing", ACM NEWS, 2016, Available: http://cacm.acm.org/news/200095-the-data-lake-concept-is-maturing/fulltext, [Accessed: 16 August 2016].

[36] Apache Software Foundation, Apache Atlas, 2015Available: http://atlas.incubator.apache.org, [Accessed: 16 August 2016].

[37] A. Ahn, APACHE ATLAS PROJECT PROPOSED FOR HADOOP GOVERNANCE, Hortonworks, 2015, Available: http://hortonworks.com/blog/apache-atlas-project-proposed-for-hadoop-governance/, [Accessed: 16 August 2016].

[38] A. Gorelik, J. Chen, O. Claude and et. al., "waterline data", 2016, Available: http://www.waterlinedata.com, [Accessed: 16 August 2016].

[39] Oracle, "Best Practices for Real-time Data Warehousing", White Paper, and August 2012.

[40] S. Anandan, M. Bogoevici, G. Renfro and et. al, “Spring XD: a modular distributed stream and batch processing system”, DEBS '15 Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems, PP. 217-225.

[41] L. Qiao, Y. Li, S. Takiar and at. al, "Gobblin: unifying data ingestion for Hadoop", JournalProceedings of the VLDB Endowment - Proceedings of the 41st International Conference on Very Large Data Bases, Kohala Coast, Hawaii, Volume 8 Issue 12, 2015, PP. 1764-1769.

[42] F. Waas, R. Wrembel, T. Freudenreich and et. al, "On-Demand ELT Architecture for Right-Time BI: Extending the Vision", International Journal of Data Warehousing and Mining archiveVolume 9و Issue 2, 2013, PP. 21-38.

[۴۳] س. کشوری، م.ع. جوادزاده و م. نقوی، «ارزیابی و مقایسه کارایی پایگاه داده‌های کلید-مقدار با هدف انتخابِ مبتنی بر نیاز»، مجله علوم رایانشی، ۶، ۱۳۹۶، (در نوبت چاپ http://csj.isi.org.ir/page12.aspx).