بهبود به‌روزرسانی پایگاه داده تحلیلی نیمه‌آنی

عیسی حضرتی[1] و نگین دانشپور[2*]

[1] گروه مهندسی کامپیوتر، واحد میاندوآب، دانشگاه آزاد اسلامی، میاندوآب، ایران

[2] دانشکده مهندسی کامپیوتر، دانشگاه تربیت دبیر شهید رجایی، تهران، ایران

چکیده

امروزه تصمیم‌گیری سریع، اهمیت زیادی در محیط کسب و کار دارد. بنابراین مدیران سعی دارند تا از داده‌های موجود در پایگاه دادهٔ تحلیلی برای پیش‌بینی و تصمیم‌گیری درست استفاده کنند. برای داشتن داده‌های مناسب، باید تغییرات ایجادشده در منابع، با کم‌ترین تأخیر در پایگاه داده تحلیلی اعمال شوند. برای رسیدن به این هدف، الگوریتم‌های متعددی ارایه شده است که از آن جمله به الگوریتم X-HYBRIDJOIN می‌توان اشاره کرد. در این الگوریتم برای انتخاب پارتیشنی از لوح سخت که در حافظه اصلی بارگزاری می‌شود از روش مناسبی استفاده نشده است. در این مقاله الگوریتم جدیدی ارائه می‌شود که در آن تغییراتی در نحوه انتخاب پارتیشن یادشده، ایجاد شده است. بدین صورت که برای هر پارتیشنی از R که بر روی لوح سخت قرار دارد، تعداد رکوردهای موجود از آن پارتیشن در حافظه اصلی، شمارش شده و در آرایه‌ای ثبت می‌شود. با استفاده از آرایه به‌دست آمده، هر بار پارتیشنی را می‌توان انتخاب کرد که شامل بیشترین رکورد برای پیوست است. برای شمارش تعداد رکوردهای هر پارتیشن، در هنگام ورود جریان داده، بررسی می‌شود که جریان داده ورودی مربوط به کدام پارتیشن است. نتایج حاصل از اجرای الگوریتم جدید نشان می‌دهد که زمان پیوست و فضای مصرفی کاهش یافته است.

واژگان کلیدی: پایگاه داده تحلیلی نیمه‌آنی، پیوست، جریان داده، تصمیم‌گیری

Improving Near Real Time Data Warehouse Refreshment

Isa Hazrati[1] & Negin Daneshpour[2*]

[1]Department of Computer Engineering, Miandoab Branch, Islamic Azad University, Miandoab, Iran

[2]Faculty of Computer Engineering, Shahid Rajaee Teacher Training University, Tehran, Iran

Abstract

Near-real time data warehouse gives the end users the essential information to achieve appropriate decisions. Whatever the data are fresher in it, the decision would have a better result either. To achieve a fresh and up-to-date data, the changes happened in the side of source must be added to the data warehouse with little delay. For this reason, they should be transformed in to the data warehouse format. One of the famous algorithms in this area is called X-HYBRIDJOIN. In this algorithm the data characteristics of real word have been used to speed up the join operation. This algorithm keeps some partitions, which have more uses, in the main memory. In the proposed algorithm in this paper, disk-based relation is joined with input data stream. The aim of such join is to enrich stream. The proposed algorithm uses clustered index for disk-based relation and

* نویسندهٔ عهده‌دار مکاتبات

* Corresponding author

join attribute. Moreover, it is assumed that the join attribute is exclusive throughout the relation. This algorithm has improved the mentioned algorithm in two stages. At the first stage, some records of source table which are frequently accessible are detected. Detection of such records is carried out during the algorithm implementation. The mechanism is in the way that each record access is counted by a counter and if it becomes more than the determined threshold, then it is considered as the frequently used record and placed in the hash table. The hash table is used to keep the frequently used records in the main memory. When the stream is going to enter in to join area, it is searched in this table. At the second stage, the choice method of the partition which is going to load in the main memory has been changed. One dimensional array is used to choose the mentioned partition. This array helps to select a partition of source table with highest number of records for the join among all partitions of source table. Using this array in each iteration, always leads to choose the best partition loading in memory. To compare the usefulness of the suggested algorithm some experiments have been done. Experimental results show that the service rate acquired in suggested algorithm is more than the existing algorithms. Service rate is the number of joined records in a time unit. Increasing service rate causes the effectiveness of the algorithm.

Keywords: Near Real Time Data Warehouse, Join, Data Stream, Decision Making

۱- مقدمه

در اوایل ایجاد پایگاه‌های داده تحلیلی، بیش‌تر آن‌ها داده‌های خود را به‌طور دوره‌ای و در زمان‌های خاصی به‌روز می‌کردند. به‌عنوان مثال، در اواخر شب یا روزهای تعطیل که حجم تغییرات کاهش می‌یافت؛ ولی با گذشت زمان و با توجه به افزایش درخواست کاربران برای داشتن داده‌های به‌روز و جدید، این روش پاسخ‌گوی نیاز کاربران و مدیران نبود؛ بنابراین پژوهش‌گران، روش جدید افزایشی را ارائه دادند. در این روش دیگر لازم نبود در هر بار به‌روزرسانی، کل داده‌ها در پایگاه داده تحلیلی بارگزاری مجدد شوند، بلکه تنها داده‌های تغییریافته بارگزاری می‌شدند و این عمل تا حدودی زمان بارگزاری را کاهش می‌داد [26,27]. گرچه استفاده از این روش بسیار کارآمدتر و بهتر از روش اولیه بود؛ ولی باز هم پاسخ‌گوی نیازهای کاربران برای داشتن داده‌های به‌روز نبود؛ زیرا این روش نیز به‌طور دوره‌ای و در زمان‌های خاصی تغییرات را اعمال می‌کرد؛ بنابراین پژوهش‌گران تصمیم گرفتند روشی ارائه کنند که به‌جای به‌روزرسانی دوره‌ای، از به‌روزرسانی پیوسته استفاده نمایند تا تغییرات با کمترین تأخیر در پایگاه داده تحلیلی اعمال شوند [3,13]. این نگرش سبب به‌وجودآمدن فرضیه پایگاه داده تحلیلی نیمه‌آنی[1] شد و ابزارها و روش‌ها جهت رسیدن به این موضوع به‌سرعت گسترش یافتند [5,19]. درواقع در پایگاه داده تحلیلی نیمه‌آنی تغییرات رخ داده در سمت پایگاه داده‌های منابع با تأخیر کم‌تری به داده‌های پایگاه داده تحلیلی اضافه می‌شوند تا همواره بتوان داده‌های به‌روز و جدید را در اختیار کاربران نهایی قرار داد. برای داشتن چنین پایگاه داده تحلیلی، ابتدا باید تغییرات رخ‌داده در پایگاه داده‌های منابع، که از این به بعد جریان داده[2] (S) نامیده می‌شود، شناسایی شده و سپس به سمت پایگاه داده تحلیلی ارسال شوند. ولی قبل از ارسال لازم است، تغییراتی بر روی جریان داده‌های ارسالی انجام شود تا بتوان آن‌ها را به شکلی درآورد که در پایگاه داده تحلیلی نیاز است. این تغییرات ممکن است، شامل اضافه‌کردن یک فیلد به جریان داده یا تغییر فیلدهای موجود در آن باشد. برای این کار لازم است تا جریان داده (S)، با داده‌های حجیم موجود بر روی لوح سخت[3] (R) پیوست شود. برای درک بهتر موضوع، شکل (۱) را در نظر بگیرید. در این شکل منبع داده[4] همان تغییرات رخ‌داده در سمت پایگاه داده‌های منابع یا همان جریان داده‌ها است. دادهٔ اصلی[5] همان جدول دادهٔ موجود بر روی لوح سخت می‌باشد و پایگاه داده تحلیلی محلی است که نتیجه پیوست منبع داده و داده اصلی به سمت آن ارسال می‌شود. برای انجام پیوست باید برای هر جریان داده، رکورد متناظر با آن در لوح سخت جستجو شود. به‌عنوان یک راه حل کل رابطه R را وارد حافظه می‌توان کرد و پیوست را انجام داد؛ ولی با توجه به محدودیت حافظه اصلی، این عمل امکان‌پذیر نیست و تنها بخشی از رابطه R را در حافظه اصلی می‌توان بارگزاری کرد. این چالشی است که باید راه‌حل مناسبی برای آن پیدا کرد.

موضوع دیگری که باید در در نظر داشت این است که دو منبع داده‌ای که برای پیوست لازم می‌باشند، نرخ دسترسی متفاوتی دارند. جریان داده (S)، یک جدول نیست و ماهیت انفجاری دارد؛ بدین مفهوم که ممکن است در هر

[1] Near-Real-Time Data Warehouse

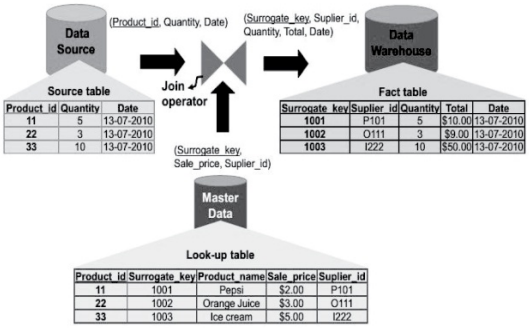
[2] Stream

[3] Disk-Base-Relation

[4] Source Data

[5] Master Data

لحظه تعداد متفاوتی از آن وارد محل پیوست شود؛ درصورتی‌که رابطه R، یک جدول بوده و بر روی لوح سخت قرار دارد و به‌دلیل مراجعه به لوح سخت (عمل I/O) نرخ دسترسی پایینی دارد. حال باید بتوان با روشی این تأخیر در دسترسی را به کمینه رساند تا بتوان تمام جریان داده‌های ورودی را حفظ و با رابطه R پیوست کرد.



Product_id	Quantity	Date
11	5	13-07-2010
22	3	13-07-2010
33	10	13-07-2010

Surrogate_key	Suplier_id	Quantity	Total	Date
1001	P101	5	$10.00	13-07-2010
1002	O111	3	$9.00	13-07-2010
1003	I222	10	$50.00	13-07-2010

Product_id	Surrogate_key	Product_name	Sale_price	Suplier_id
11	1001	Pepsi	$2.00	P101
22	1002	Orange Juice	$3.00	O111
33	1003	Ice cream	$5.00	I222

(شکل-۱): تغییر شکل جریان داده ورودی از طریق پیوست با داده‌های موجود بر روی لوح سخت [16]

(Figure- 1): Transformation of input stream by joining them with the data on the disk [16]

برای این منظور الگوریتم X-HYBRIDJOIN ارائه شده است که الگوریتم X-HYBRIDJOIN برای پیوست جریان داده ورودی (S) با داده‌های موجود بر روی لوح سخت (R) استفاده می‌شود [16]. الگوریتم مذکور از شاخص‌گذاری[1] بر روی صفت پیوست استفاده می‌کند که سبب می‌شود در انتخاب پارتیشن‌های مورد نظر عملکرد مطلوبی حاصل شود. این الگوریتم با درنظرگرفتن خصوصیات داده‌های دنیای واقعی، یک توزیع اریب[2] برای جریان داده‌های ورودی در نظر می‌گیرد. درواقع این نوع توزیع، بیان‌کنندهٔ همان قانون ۸۰/۲۰ است. برای مثال این قانون در مورد فروش یک فروشگاه بیان می‌کند که ۸۰ درصد از فروش در یک فروشگاه، مربوط به ۲۰ درصد از محصولات موجود در آن فروشگاه است. با توجه به این قانون، بخشی از رابطه R بیشتر مورد دسترسی قرار می‌گیرد؛ لذا الگوریتم یادشده سعی دارد با استفاده از خصوصیت داده‌های دنیای واقعی تعداد مراجعات به لوح سخت را، که بسیار پرهزینه است، کاهش دهد. در ادامه بخش‌های اصلی الگوریتم X-HYBRIDJOIN که برای این منظور به‌کار گرفته شده‌اند، بیان شده و مشکلات آن‌ها مطرح می‌شود. الگوریتم X-HYBRIDJOIN، از یک قسمت ثابت در حافظهٔ اصلی، برای نگهداری پارتیشنی که به‌طور دائمی در حافظه قرار می‌گیرد، استفاده می‌کند. درواقع با استفاده از این روش سعی می‌کند تعداد مراجعات به لوح سخت را کاهش دهد. با توجه به اینکه تنها در شروع کار، الگوریتم قادر به انتخاب بخش پراستفاده است، در ادامه بخش مورد نظر را نمی‌تواند تغییر دهد و این امر در بعضی موارد باعث کاهش کارایی می‌شود. این موضوع به‌عنوان مشکل نخست الگوریتم یادشده است. برای نمونه در همان مثال فروشگاه، با تغییر فصول، اولویت‌های خرید مشتریان تغییر می‌کند که منجر به تغییر داده‌های پراستفاده می‌شود؛ و این الگویتم قابلیت تطبیق با این تغییرات را ندارد. همچنین این الگوریتم از یک قسمت قابل تعویض برای بارگزاری سایر بخش‌های رابطه R استفاده می‌کند. دواقع در هر دور اجرای الگوریتم یک پارتیشن از رابطه R که بر روی لوح سخت قرار دارد، انتخاب و در این قسمت بارگزاری و پیوست انجام می‌شود. نحوه انتخاب این پارتیشن اهمیت زیادی دارد. در الگوریتم X-HYBRIDJOIN، هربار پارتیشنی انتخاب می‌شود که تنها وجود یک رکورد را برای پیوست تضمین می‌کند.

اگر انتخاب پارتیشن به‌گونه‌ای باشد که رکوردهای بیشتری برای پیوست در آن موجود باشد، به‌طور قابل ملاحظه‌ای کارایی الگوریتم افزایش می‌یابد؛ زیرا مراجعه به لوح سخت برای بارگزاری پارتیشن جدید بسیار پرهزینه بوده و باعث کاهش شدید کارایی الگوریتم می‌شود. الگوریتم X-HYBRIDJOIN قادر نیست پارتیشنی را که بیشترین رکورد برای پیوست دارد، انتخاب کند، و این مسأله نیز به‌عنوان مشکل دوم این الگوریتم، مطرح است.

در این مقاله الگوریتمی پیشنهاد می‌شود که مشکلات بیان‌شده برای الگوریتم X-HYBRIDJOIN را در دو مرحله برطرف می‌کند. الگوریتم پیشنهادی همانند الگوریتم X-HYBRIDJOIN بر پایه شاخص‌گذاری است که بر روی صفت پیوست اعمال می‌شود. برای برطرف‌کردن مشکل نخست که به‌عنوان بهبود مرحلهٔ نخست است، در الگوریتم پیشنهادی از جدول درهم‌سازی برای نگهداری رکوردهایی از رابطه R که به‌طور پیوسته مورد دسترسی قرار می‌گیرند، استفاده شده است. این جدول درهم‌سازی در حافظه اصلی قرار دارد. به‌کاربردن جدول درهم‌سازی، سبب کاهش عملیات حذف و اضافهٔ جریان داده‌های ورودی به محل پیوست می‌شود. بدین صورت که در ابتدای کار، برای هر جریان داده

[1] Indexing

[2] Skewed Distribution

بررسی می‌شود که رکورد متناظر آن، در جدول درهم‌سازی قرار دارد یا خیر. اگر رکورد متناظر جریان داده در جدول درهم‌سازی وجود داشته باشد، عمل پیوست انجام شده و نتیجهٔ حاصل، به سمت خروجی ارسال می‌شود. برای برطرف‌کردن مشکل دوم که به‌عنوان بهبود مرحله دوم است، از سازوکار جدیدی برای انتخاب پارتیشن بهینه، استفاده شده است. پارتیشن بهینه پارتیشنی است که دارای بیشترین رکود برای پیوست در میان تمام پارتیشن‌های موجود بر روی لوح سخت است.

در سازوکار ارائه‌شده، هر بار پارتیشن بهینه از رابطه R انتخاب می‌شود. برای انتخاب پارتیشن بهینه، تعداد رکوردهای متناظر از هر پارتیشن که به‌عنوان جریان داده در حافظه قرار دارند، شمارش شده و پارتیشنی که دارای بیشترین تعداد رکورد باشد، به‌عنوان پارتیشن بهینه انتخاب می‌شود. در بخش الگوریتم پیشنهادی، سازوکار مورد استفاده به‌طور کامل توضیح داده می‌شود.

درواقع می‌توان گفت برتری اصلی الگوریتم پیشنهادی نسبت به الگوریتم X-HYBRIDJOIN، در نحوهٔ انتخاب پارتیشنی است که قرار است از رابطه R انتخاب شده و در حافظه اصلی بارگزاری شود. همچنین مبنای کار الگوریتم X-HYBRIDJOIN بر اساس توزیع اریب داده‌ها است و بنابراین در مواردی که داده‌ها از این توزیع برخوردار نباشند عملکرد مناسبی ندارد. الگوریتم ارائه‌شده در این مقاله این محدودیت را ندارد و برای هر نوع توزیع داده‌یی عملکرد مناسبی دارد.

در ادامه، در بخش دوم به بررسی کارهای پیشین پرداخته شده است و دلایل ارائه الگوریتم جدید بیان شده است. در بخش سوم الگوریتم ارائه‌شده در این مقاله به‌طور کامل شرح داده شده است. در بخش چهارم نتایج آزمایش‌های انجام‌شده بیان شده و در بخش پنجم نتیجه‌گیری و کارهای آینده گفته شده است.

۲- پیشینهٔ پژوهش

مقالات مختلفی در مورد پایگاه داده تحلیلی وجود دارند که هر کدام از جنبه خاصی به موضوع پرداخته‌اند. از میان آن‌ها [9-11, 15, 21] به بررسی معماری پایگاه داده تحلیلی آنی می‌پردازند. [7, 8] با استفاده از فناوری محاسبات ابری، پایگاه داده تحلیلی آنی را پیاده‌سازی کرده‌اند. همچنین [12] با استفاده از مفهوم چندهسته‌ای اقدام به پیاده‌سازی پایگاه داده تحلیلی آنی کرده است. مقالات بسیاری نیز به پیاده‌سازی پایگاه داده تحلیلی نیمه‌آنی پرداخته‌اند که از آن جمله به الگوریتم‌های [14-23, 25] می‌توان اشاره کرد. در ادامه به بررسی کامل‌تر الگوریتم‌هایی پرداخته شده است که سعی کرده‌اند مشکلات مطرح‌شده در این مقاله را برطرف کنند.

الگوریتم MESHJOIN، برای پیوست جریان داده ورودی با رابطه موجود بر روی لوح سخت (رابطهٔ R) استفاده می‌شود [22]. ابتدا با یک مثال ساده ایده اصلی الگوریتم بیان می‌شود. فرض می‌شود که رابطهٔ R شامل دو صفحه (p1 وp2) است و آن‌که الگوریتم فضای کافی برای ذخیرهٔ پنجره‌ای شامل دو رکورد ورودی اخیر (s1 و s2) را دارد.

در لحظه $t = 0$، الگوریتم رکورد ورودی نخست و صفحه نخست را می‌خواند و در حافظه پیوست می‌کند. در لحظه $t = 1$، الگوریتم رکورد ورودی دوم و صفحه دوم را به حافظه می‌آورد. در این لحظه، صفحه دوم با دو رکورد ورودی پیوست شده است، علاوه‌براین، رکورد ورودی s1 با کل رابطهٔ R پیوست شده است و از حافظه می‌تواند حذف شود. در لحظه $t = 2$، الگوریتم دوباره به هر دو ورودی، به‌صورت متوالی دسترسی پیدا می‌کند و رکوردهایی را که در حافظه قرار دارند، به‌روزرسانی می‌کند. به‌طور دقیق‌تر، پیمایش رابطهٔ R و بارگزاری صفحه نخست را از سر می‌گیرد و به‌طور هم‌زمان رکورد s1 را با رکورد ورودی بعدی یعنی s3، جایگزین می‌کند. صفحهٔ نخست با رکوردهای s2 و s3 پیوست شده و رکورد s2 از حافظه می‌تواند خارج شود.

با توجه به ایده بیان‌شده الگوریتم MESHJOIN، ابتدا رابطهٔ R را به پارتیشن‌هایی که بتواند آن‌ها را در حافظه بارگزاری کند، تقسیم‌بندی می‌کند؛ سپس، به نوبت پارتیشن‌ها را وارد حافظه کرده و با جریان داده‌های ورودی پیوست می‌کند. این عمل به دفعات تکرار می‌شود. این الگوریتم هیچ فرضیه‌ای در مورد توزیع داده‌ها ندارد و از استراتژی مناسبی برای انتخاب پارتیشن‌های رابطهٔ R استفاده نمی‌کند و همه پارتیشن‌ها را به نوبت وارد حافظه می‌کند؛ حتی اگر هیچ عامل پیوستی در آن پارتیشن وجود نداشته باشد. این‌کار باعث افزایش زمان پیوست می‌شود که در نتیجهٔ آن کارایی الگوریتم کاهش می‌یابد.

الگوریتم HYBRIDJOIN، همانند MESHJOIN، برای پیوست جریان داده‌ها با رابطهٔ R طراحی شده است [18]. این الگوریتم نیز از همان ایده مطرح‌شده در الگوریتم MESHJOIN استفاده و ابتدا رابطهٔ R را پارتیشن‌بندی

می‌کند. الگوریتم یادشده در هر دور پارتیشنی از رابطهٔ R را انتخاب می‌کند که دست‌کم یک عامل پیوست در آن موجود باشد. در این الگوریتم، بر خلاف MESHJOIN، پارتیشنی از رابطهٔ R که هیچ رکورد متناظری در میان جریان داده‌ها ندارد در حافظه بارگزاری نمی‌شود. این عمل سبب کاهش مراجعات به لوح سخت می‌شود. انتخاب پارتیشن یادشده با استفاده از یک شاخص‌گذاری بر روی صفت پیوست صورت می‌پذیرد. این الگوریتم نیز همانند MESHJOIN هیچ فرضیه‌ای در مورد توزیع داده‌ها ندارد. به همین خاطر هیچ پیش‌بینی در مورد پارتیشن پراستفادهٔ رابطه R نمی‌تواند داشته باشد.

الگوریتم M-HYBRIDJOIN، میزان بافر تخصیص یافته به قسمت‌های قابل تعویض و غیرقابل تعویض بافر لوح سخت در الگوریتم X-HYBRIDJOIN را تنظیم می‌کند. با این عمل تعداد پارتیشن‌های مقیم در بخش غیرقابل تعویض بافر لوح سخت را کنترل می‌کند و تعداد مراجعات به لوح سخت و عملیات I/O را کاهش می‌دهد؛ اما همچنان مشکلات موجود در الگوریتم X-HYBRIDJOIN برطرف نشده است [24].

الگوریتم SSJ، بر خلاف الگوریتم X-HYBRIDJOIN، به‌جای این‌که بخش خاصی از رابطهٔ R به‌طور دائمی در قسمت ثابت بافر لوح سخت قرار گیرد، رکوردهایی در آن قرار داده می‌شوند که بیشترین تعداد استفاده در پیوست را داشته باشند. درواقع با این عمل، رکوردهای پراستفاده در کل رابطهٔ R شناسایی شده و در بافر قرار می‌گیرند. این روش سبب افزایش نرخ سرویس می‌شود. این الگوریتم نیز همانند سایر الگوریتم‌ها روش مناسبی برای انتخاب پارتیشن‌ها از رابطهٔ R ارائه نکرده است [20].

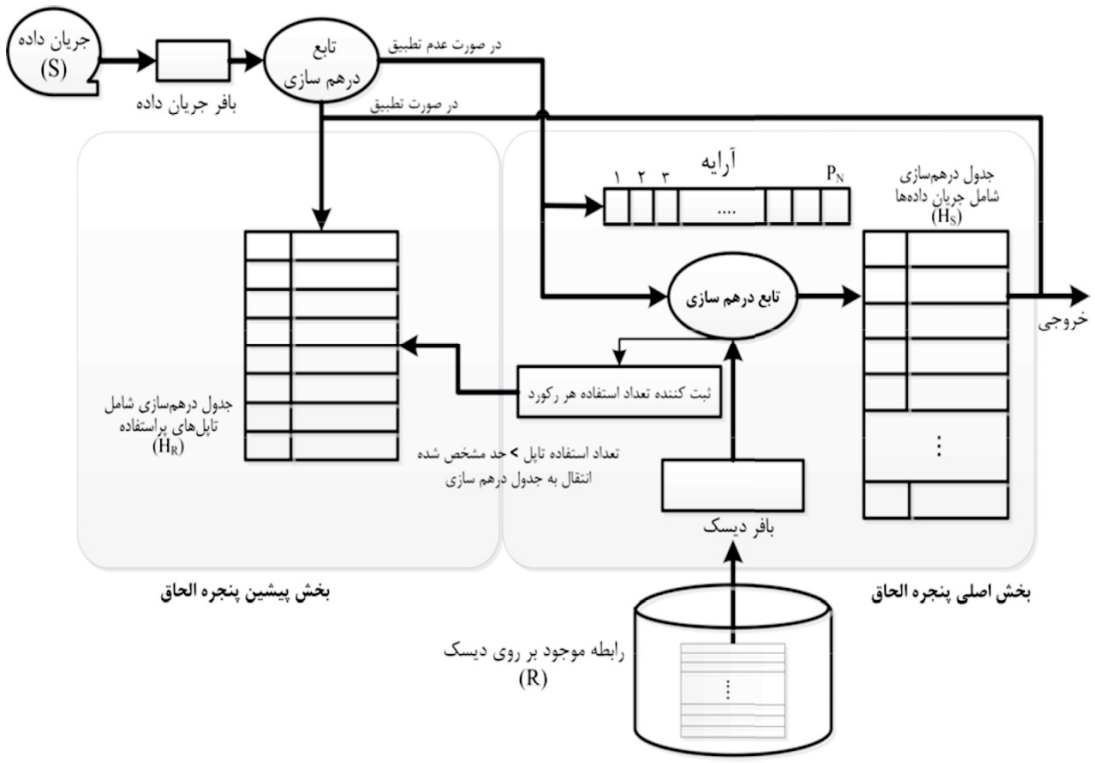
مقاله [4] یک پلتفرم برای پایگاه داده تحلیلی نیمه‌آنی ارائه نکرده و آن پلتفرم را برای محل خاص و مشخص‌شده‌ای مورد استفاده قرارا داده است. در این مقاله از نحوه پیوست جریان داده‌ها مطلبی بیان نشده و برای حل مشکلات موجود روشی ارائه نشده است.

الگوریتم X-HYBRIDJOIN، از مشاهدات و خصوصیات داده‌های دنیای واقعی برای کاهش زمان پیوست استفاده می‌نماید [16]. در دنیای واقعی بیش‌تر داده‌ها توزیع اریب دارند [6]. به‌طور مثال، در یک فروشگاه تعداد مشخصی از محصولات فروش بیشتری نسبت به سایر محصولات دارند. درنتیجه جریان داده تولیدشدهٔ این تعداد از محصولات بیشتر خواهد بود؛ لذا الگوریتم X-HYBRIDJOIN پارتیشنی از رابطه R را که مراجعه بیشتری به آن وجود دارد، به‌طور ثابت در حافظه قرار می‌دهد. استفاده از این قسمت ثابت سبب می‌شود تا تعداد مراجعات به لوح سخت کاهش یابد. پنجره پیوست در این الگوریتم شامل موارد زیر است:

بافر جریان داده[1] که برای نگهداری موقت جریان داده‌های ورودی استفاده می‌شود. بافر لوح سخت[2] که خود شامل دو قسمت ثابت[3] و قابل تعویض[4] است. قسمت ثابت شامل همان پارتیشنی از رابطه R است که مراجعات بیشتری به آن می‌شود و قسمت قابل تعویض در هر دور اجرا، شامل یک پارتیشن از رابطه R است. جدول درهم‌سازی که دربرگیرندهٔ جریان داده‌ها است و یک صف[5] که فیلد کلیدی جریان داده‌های موجود در جدول درهم‌سازی را در خود نگه می‌دارد.

روش کار در الگوریتم X-HYBRIDJOIN بدین صورت است که ابتدا تمام جریان داده‌های ورودی درون جدول درهم‌سازی قرار می‌گیرند تا ظرفیت آن تکمیل شود. سپس در هر دور اجرا، ابتدا برای تمام رکوردهای موجود در قسمت ثابت بافر لوح سخت بررسی می‌شود که آیا آن رکورد در جدول درهم‌سازی وجود دارد یا خیر. اگر وجود داشته، باشد دو رکورد با هم پیوست شده و نتیجه به خروجی ارسال می‌شود. برای هر رکوردی که پیوست می‌شود متناظر آن از جدول درهم‌سازی و صفت کلیدی آن از صف حذف می‌شود. بعد از اتمام بررسی رکوردهای قسمت ثابت، نوبت به قسمت قابل تعویض می‌رسد و همین عمل تکرار می‌شود. در پایان هر دور، به تعداد رکوردهای حذف‌شده از جدول درهم‌سازی، رکورد جدید از بافر جریان داده خوانده شده و وارد جدول درهم‌سازی می‌شود. به‌ازای هر رکوردی که وارد جدول درهم‌سازی می‌شود، صفت کلیدی آن نیز در صف قرار می‌گیرد. از صف مذکور برای انتخاب پارتیشنی از رابطه R که قرار است در قسمت تعویض‌پذیر بافر لوح سخت قرار گیرد، استفاده می‌شود. بدین‌صورت که ابتدا صفت کلیدی از ابتدای صف خوانده می‌شود و سپس از رابطه R پارتیشنی انتخاب می‌شود که شامل رکوردی با این صفت کلیدی است. درواقع با این کار، الگوریتم تضمین می‌کند که دست‌کم یک رکورد در پارتیشن انتخاب‌شده برای پیوست وجود دارد. مشکلی که در این الگوریتم وجود دارد این است که: ۱) جریان داده‌هایی که ممکن است متناظر آن‌ها در قسمت

[1] Stream Buffer
[2] Disk Buffer
[3] Non-swappable
[4] Swappable
[5] Queue



(شکل-۲): معماری حافظه استفاده‌شده در الگوریتم پیشنهادی
(Figure-2): Memory Architecture used in the Proposed Algorithm

ثابت بافر لوح سخت وجود داشته باشند، ابتدا به جدول درهم‌سازی و صف اضافه می‌شود و سپس بررسی می‌شود که آیا آن جریان داده در آن قسمت وجود دارد یا خیر. اگر وجود داشته باشد دوباره همان رکورد از جدول درهم‌سازی و صف حذف می‌شود که این زمان پیوست را به‌شدت افزایش می‌دهد. ۲) در این الگوریتم، در انتخاب پارتیشنی که قرار است در قسمت تعویض‌پذیر بافر لوح سخت قرار گیرد، به‌صورت بهینه عمل نمی‌شود. هر بار پارتیشنی انتخاب می‌شود که تنها وجود یک رکورد را برای پیوست تضمین می‌کند. در ادامه الگوریتمی ارائه می‌شود که مشکلات مطرح‌شده در مورد این الگوریتم را برطرف خواهد کرد.

با توجه به اینکه میزان فضای اختصاص‌یافته به بخش‌های مختلف موجود در الگوریتم X-HYBRIDJOIN اهمیت زیادی دارد، در [28] به بررسی نحوه اختصاص میزان حافظه به قسمت‌های مختلف پرداخته شده است. در این مقاله، میزان فضای اختصاص‌یافته به هر بخش بررسی شده و مقدار بهینه برای هر بخش به‌دست آمده است. این عمل سبب بهبود عملکرد الگوریتم می‌شود؛ ولی همچنان مشکلات مطرح‌شده برای الگوریتم وجود دارند.

در [29] برای بهبود عملکرد الگوریتم X-HYBRIDJOIN از یک جدول درهم‌سازی برای نگهداری پارتیشن ثابت استفاده شده است. گرچه استفاده از جدول درهم‌سازی سبب کاهش زمان جستجو در آن جدول می‌شود، ولی همچنان مشکلات مطرح‌شده برای الگوریتم X-HYBRIDJOIN برطرف نشده‌اند.

۳- الگوریتم پیشنهادی

در این قسمت الگوریتم پیشنهادی این مقاله شرح داده می‌شود. در این الگوریتم رابطهٔ موجود بر روی لوح سخت با جریان داده‌های ورودی پیوست می‌شود. هدف از این پیوست تغییر شکل‌دادن جریان داده‌ها است. الگوریتم پیشنهادی از شاخص‌گذاری برای رابطه موجود بر روی لوح سخت و بر روی صفت پیوست استفاده می‌کند. همچنین فرض می‌شود که صفت پیوست در کل رابطه، منحصربه‌فرد است. معماری حافظهٔ استفاده‌شده در این الگوریتم در شکل (۲) نشان داده شده است.

معماری الگوریتم پیشنهادی همانند الگوریتم X-HYBRIDJOIN است که دو تغییر در آن انجام شده است. تغییر نخست در مورد قسمت ثابت بافر لوح سخت است که به‌جای آن از جدول درهم‌سازی H_R استفاده شده است [1].

همان‌طورکه بیان شد رکوردهایی از رابطه R در قسمت ثابت بافر لوح سخت قرار می‌گیرند که به دفعات در پیوست استفاده می‌شوند؛ لذا شناسایی رکوردهایی که به‌طور مکرر در پیوست شرکت می‌کنند اهمیت زیادی دارد. روش کار بدین صورت است که توسط یک شمارنده تعداد دفعاتی که هر رکورد مورد دسترسی قرار می‌گیرد شمارش شده و اگر از حد آستانهٔ مشخص‌شده بیشتر باشد، به‌عنوان رکورد پراستفاده در جدول H_R قرار می‌گیرد. در ادامهٔ الگوریتم، هرگاه ظرفیت جدول H_R تکمیل شد رکورد تازه وارد با قدیمی‌ترین رکورد موجود در H_R جایگزین می‌شود و این عمل به کرّار در طول اجرای الگوریتم تکرار می‌شود. استفاده از جدول H_R به‌جای بخش ثابت باعث می‌شود تا تمام رکوردهایی که در کل رابطه R بیشتر مورد دسترسی قرار می‌گیرند، شناسایی شوند. این در حالی است که در الگوریتم X-HYBRIDJOIN تنها رکوردهای یک پارتیشن قابل انتخاب بودند که منجر به انعطاف پایین الگوریتم می‌شد. بهبود بیان‌شده، در شکل (۲) در بخش پیشین قابل مشاهده است. در ادامه، کاربرد بخش پیشین در روند اجرای الگوریتم بیان می‌شود.

زمانی که جریان داده وارد چرخه الگوریتم می‌شود در بدو ورود، جدول H_R جستجو می‌شود و اگر متناظر آن یافت شود، عمل پیوست در همان ابتدای کار انجام و حاصل به‌سمت خروجی هدایت می‌شود. با توجه به اینکه بیش‌تر رکوردهای پر استفاده، شناسایی شده و در جدول H_R قرار دارند و با درنظرگرفتن قانون ۸۰/۲۰، احتمال این‌که رکورد مورد نظر در H_R یافت شود، بسیار بالا است. بنابراین استفاده از جدول درهم‌سازی به‌عنوان بافر، باعث می‌شود جریان داده‌هایی که متناظر آن‌ها در این پارتیشن وجود دارند، وارد جدول درهم‌سازی H_S نشوند و در همان ابتدای الگوریتم پیوست شده و به خروجی ارسال شوند. استفاده از این روش سبب بهبود کارایی و کاهش زمان پیوست می‌شود و همچنین سبب برطرف‌شدن نخستین مشکل مطرح‌شده در مورد الگوریتم X-HYBRIDJOIN می‌شود.

تغییر دوم در مورد نحوهٔ انتخاب پارتیشنی از رابطه R که قرار است در قسمت قابل تعویض بافر لوح سخت قرار گیرد، رخ داده است [2]. به‌موجب این تغییر، صف موجود در الگوریتم X-HYBRIDJOIN حذف شده است و به‌جای آن از آرایه‌ای یک‌بعدی استفاده می‌شود. با استفاده از این آرایه همواره پارتیشنی از رابطه R را می‌توان انتخاب کرد که بیشترین رکورد را برای پیوست، در میان تمام پارتیشن‌های رابطه R داشته باشد. روشی ارایه می‌شود که با استفاده از آن تعداد رکوردهای موجود در هر پارتیشن از رابطه R را که در جدول H_S وجود دارند می‌توان شمارش کرد و هنگام انتخاب، پارتیشنی را انتخاب کرد که بیشترین تعداد رکورد را برای پیوست دارد. استفاده از این روش دو مزیت اساسی دارد. ۱) باعث می‌شود در هر دور پارتیشنی انتخاب شود که بیشترین رکورد را برای پیوست دارد. ۲) باعث می‌شود الگوریتم برای هر نوع توزیع داده‌ای عملکرد مناسب و یکسانی نشان دهد؛ زیرا با شمارش رکوردهای موجود در حافظه اقدام به انتخاب پارتیشن می‌کند. استفاده از این روش نیز سبب برطرف‌شدن دومین مشکل مطرح‌شده در مورد الگوریتم X-HYBRIDJOIN می‌شود.

برای اینکه در هر بار بتوان بهترین پارتیشن از رابطه R را انتخاب کرد، باید به شکل زیر عمل کرد:

با توجه به اینکه از شاخص‌گذاری بر روی صفت پیوست استفاده شده است و این باعث می‌شود که رکوردهای رابطه R بر اساس صفت پیوست، به‌صورت مرتب باشند؛ بنابراین با استفاده از این خصوصیت ابتدا اقدام به پارتیشن‌بندی رابطه R می‌شود. اگر تعداد کل رکوردهای موجود در رابطه R برابر $M = 2^k$ در نظر گرفته شود و تعداد رکوردهای موجود در هر پارتیشن برابر $n = 2^{k'}$ باشد، آن‌گاه تعداد کل پارتیشن‌های رابطه R، با توجه به رابطه (۱)، برابر $P_N = 2^{k''}$ خواهد بود ($k, k', k'' \in N$).

$$P_N = \frac{M}{n} \qquad (۱)$$

بعد از آن‌که تعداد کل پارتیشن‌های رابطه R به‌دست آمد، حال آرایه‌ای یک‌بعدی به‌طول P_N تعریف می‌شود که اندیس خانه‌های آن متناظر با شماره پارتیشن‌های رابطه R است و محتوای هر خانه از آرایه، نشان‌دهنده تعداد رکوردهای موجود از پارتیشن متناظر با اندیس آن خانه، در جدول درهم‌سازی H_S است. در ادامه برای هر جریان داده‌ای که قرار است به جدول H_S افزوده شود باید پارتیشن متناظر با آن، در رابطه R شناسایی شده و در آرایهٔ یادشده ثبت شود.

با توجه به این‌که مقدار صفت پیوست رابطه R و جریان داده‌های ورودی، دامنهٔ یکسانی دارند، پارتیشن

متناظر هر جریان داده با استفاده از رابطه (۲) به‌دست می‌آید (صفت پیوست جریان ورودی a_i در نظر گرفته می‌شود).

$$\text{پارتیشن متناظر با ورودی } a_i \text{ در } R = \frac{a_i}{n} \qquad (۲)$$

زمانی که جریان داده‌ای وارد جدول درهم‌سازی H_S می‌شود، ابتدا با استفاده از رابطه (۲) شماره پارتیشن آن به‌دست می‌آید؛ سپس با استفاده از شمارهٔ به‌دست‌آمده که همان اندیس آرایه N_P است، به آرایه N_P مراجعه نموده و محتوای خانه با اندیس به‌دست آمده، یک واحد افزایش داده می‌شود. با این عمل تعداد رکوردهای هر پارتیشن شمارش شده و در آرایه ثبت می‌شود. با استفاده از این آرایه همواره پارتیشنی از رابطه R که بیشترین رکورد در جدول درهم‌سازی H_S را دارد می‌توان، انتخاب کرد.

شبه‌کد روش پیشنهادی در الگوریتم (۱) بیان شده است:

(الگوریتم-۱): شبه‌کد الگوریتم پیشنهادی
(Algorithm-1): pseudo-code for proposed algorithm

```
Input: A disk-based relation R with an index on join
       attribute and a stream of updates S.
Output: S∞R.
Parameters: w tuples of S and a partition p_i of R.
Method:
1:begin
2: DO INITIALIZE OPERATION
3: w ← h_s
4: LOAD first partition p_1 of R into the non-
   swappable part of the disk buffer(H_R).
5: while (True) do
6:       READ the index of maximum value of
         array
7:       LOAD a disk partition p_i of R into the
         swappable part of disk buffer using the index
         of p_i
8:       array[index] ← 0
9:       for each tuple r into p_i do
10:        if r ∈ H_S then
11:          OUTPUT r ∞ H_S
12:          w ← w + number of matching tuples
             found in H_S
13:          DELETE all matched tuples from H_S
14:        end if
15:      end for
16:      while (w    0) do
17:          READ one tuple r from stream buffer
18:          if r ∈ H_R then
19:            OUTPUT r ∞ H_R
20:          else
21:            ADD r into H_S and increase
               corresponding index of array by one
22:          end if
23:      end while
24: end while
25:end
```

الگوریتم یادشده، یک الگوریتم تکرارشونده است و به‌صورت زیر عمل می‌کند:

ابتدا تا زمانی که ظرفیت جدول H_S تکمیل نشده است، جریان داده‌های ورودی به آن اضافه و همچنین تغییرات مربوط به آرایه N_P، همانگونه که شرح داده شد، انجام می‌شود و سپس شمارندهٔ w، برابر صفر قرار داده می‌شود (خط ۲ و ۳). بعد از تکمیل ظرفیت جدول H_S پارتیشنی که مراجعه بیشتری به آن خواهد شد، تعیین شده و رکوردهای آن درون جدول H_R قرار داده می‌شوند (خط ۴). حال باید پارتیشنی که قرار است، درون بخش قابل تعویض بافر لوح سخت قرار گیرد، انتخاب شود. برای انتخاب این پارتیشن با تابع بیشینه، خانه‌ای از آرایه N_P که بیشترین مقدار را دارد، پیدا شده و از اندیس آن خانه(Index) مشخص می‌شود؛ سپس با استفاده از اندیس به‌دست آمده و رابطهٔ (۳)، ابتدای پارتیشن یادشده به‌دست می‌آید. در ادامه با استفاده از رابطهٔ (۴) انتهای پارتیشن مورد نظر مشخص می‌شود:

$$begin = Index * n \qquad (۳)$$

$$end = begin + n \qquad (۴)$$

begin: رکورد شروع پارتیشن ، *end*: رکورد پایان پارتیشن

بعد از مشخص‌شدن پارتیشن یادشده، آن پارتیشن در قسمت قابل تعویض بافر لوح سخت بارگزاری و محتوای خانه متناظر با آن در آرایه، برابر صفر می‌شود (خطوط ۶ تا ۸). برای همهٔ رکوردهای موجود در بخش قابل تعویض بافر لوح سخت، به‌صورت جداگانه برای هر رکورد، بررسی می‌شود که آیا آن رکورد در جدول H_S وجود دارند یا خیر. اگر وجود داشته باشند، پیوست صورت گرفته و رکورد یادشده از جدول H_S حذف می‌شود و شمارندهٔ w به تعداد رکوردهای حذف‌شده، افزایش می‌یابد (خطوط ۹ تا ۱۵). شمارنده w برای نگهداری تعداد رکوردهای حذف‌شده از جدول H_S استفاده می‌شود. بعد از اتمام بررسی تمامی رکوردها، باید به تعداد رکوردهای حذف‌شده، رکورد جدید از بافر جریان داده وارد جدول H_S شود؛ اما هر رکوردی که قرار است وارد جدول H_S شود، ابتدا در جدول H_R (همان پارتیشنی که به‌طور دائمی در حافظه قرار دارد) جستجو می‌شود؛ اگر رکورد متناظرش در آن جدول موجود باشد، پیوست شده و به

خروجی می‌رود. درصورتی‌که در آن جدول یافت نشود به جدول H_S اضافه می‌شود و خانه متناظر آن در آرایه، یک واحد افزایش می‌یابد، خطوط (۱۶ تا ۲۳). بعد از پرشدن جدول H_S الگوریتم دوباره تکرار می‌شود. تکرار الگوریتم تا زمانی صورت می‌پذیرد که هیچ جریان داده‌ای در بافر جریان داده موجود نباشد.

در ادامه مدل هزینه‌ای برای الگوریتم پیشنهادی ارائه می‌شود. در این مدل، فرمول‌هایی برای محاسبه فضای مورد نیاز الگوریتم و زمان اجرای آن وجود دارد. علائم به‌کاررفته در جدول (۱) نشان داده شده است.

(جدول-۱): علائم به‌کار رفته در مدل هزینه‌ای
(Table-1): the Symbols used in Cost Model

نام پارامتر	نشانه
کل حافظه اختصاصی(بایت)	M
نرخ سرویس(رکورد بر ثانیه)	μ
اندازه ورودی(= تعداد رکوردهای پیوست شده در دور قبلی)	W
اندازه رکورد جریان داده(بایت)	v_s
اندازه هر یک از قسمت‌های قابل تعویض و ثابت بافر(بایت)	v_p
اندازه یک رکورد از رابطه R(بایت)	v_R
اندازه هر یک از قسمت‌های قابل تعویض و ثابت بافر(رکورد)	$d_T = v_P / v_R$
ضریب حافظه برای جدول درهم‌سازی	α
ضریب حافظه برای آرایه	$1-\alpha$
هزینه خواندن یک پارتیشن از لوح سخت به بافر لوح سخت(نانوثانیه)	$c_{I/O}(v_P)$
هزینه جستجوی یک رکورد در جدول درهم‌سازی(نانوثانیه)	c_H
هزینه تولید خروجی برای یک رکورد(نانوثانیه)	c_O
هزینه حذف یک رکورد از جدول درهم‌سازی(نانوثانیه)	c_E
هزینه خواندن یک رکورد جریان داده از بافر(نانوثانیه)	c_S
هزینه افزودن یک رکورد به جدول درهم‌سازی و تغییر آرایه(نانوثانیه)	c_A
کل هزینه برای یک دور اجرای الگوریتم پیشنهادی	c_{loop}

میزان فضای مصرفی توسط بخش‌های مختلف الگوریتم قابل محاسبه است. میزان فضای مصرفی توسط قسمت‌های ثابت و قابل تعویض لوح سخت برابر با $2v_p$ است. حافظه مورد استفاده برای جدول درهم‌سازی از رابطه ۵ و آرایه از رابطه (۶) به‌دست می‌آید. کل فضای مصرفی توسط الگوریتم نیز توسط رابطه (۷) قابل محاسبه است.

$$\text{حافظه مصرفی جدول درهم‌سازی} = \alpha(M-2v_p) \quad (۵)$$

$$\text{حافظه مصرفی آرایه} = (1-\alpha)(M-2v_p) \quad (۶)$$

$$\text{کل فضای مصرفی} = 2v_p + \alpha(M-2v_p) + (1-\alpha)(M-2v_p) \quad (۷)$$

با توجه به اینکه الگوریتم پیشنهادی، یک الگوریتم تکرارشونده است، زمان اجرای یک تکرار از آن محاسبه می‌شود که در رابطه (۸) بیان شده است.

$$c_{loop}(\sec s) = 10^{-9}[c_{I/O}(v_p) + d_T.c_{II} + w(c_O + c_E + c_S + c_A + c_H)] \quad (۸)$$

نرخ سرویس نهایی الگوریتم پیشنهادی نیز در رابطه (۹) بیان شده است.

$$\mu = \frac{w}{c_{loop}} \quad (۹)$$

۴- نتایج آزمایش‌ها

برای اینکه کارایی الگوریتم پیشنهادی را بتوان بررسی کرد، آزمایش‌هایی انجام شده است که در این بخش ارائه می‌شوند. با توجّه به این‌که سخت‌افزار و داده‌هایی که در انجام آزمایش مورد استفاده قرار می‌گیرند، اهمیّت ویژه‌ای در این الگوریتم دارند، در ابتدا به معرفی آن‌ها می‌پردازیم.

داده‌های استفاده‌شده در این بخش، دارای توزیع اریب می‌باشند و به‌گونه‌ای طراحی شده‌اند که از قانون ۸۰/۲۰ پیروی می‌کنند. برای اینکه بتوان نتایج الگوریتم به‌دست‌آمده را با الگوریتم X-HYBRIDJOIN مقایسه کرد از داده‌هایی با توزیع اریب استفاده شده است. روش تولید و دلیل استفاده از این نوع داده در [16] توضیح داده شده است. این در حالی است که الگوریتم پیشنهادی برای هر نوع توزیع داده‌ای عملکرد مناسبی دارد.

خصوصیّات کامل این داده‌ها در جدول (۲) نمایش داده شده است.

(جدول-۲): مشخصات کامل داده‌های استفاده شده در آزمایش‌ها

(Table-2): Complete Specifications of the Data used in the Experiments

پارامتر	مقدار
داده‌های موجود بر روی لوح سخت	
اندازه رابطه موجود بر روی لوح سخت	0.5 میلیون تا 8 میلیون رکورد
اندازه هر رکورد	120 بایت
جریان داده	
اندازه هر رکورد	20 بایت
ویژگی	انفجاری و دارای توزیع اریب

برای نگهداری و مدیریت داده‌ها از SQL Server 2008 استفاده شده است. پیاده‌سازی الگوریتم نیز در نرم‌افزار C# نسخه ۲۰۱۲ انجام شده است.

سخت‌افزاری که الگوریتم‌ها بر روی آن اجر شده‌اند، دارای پردازندهٔ Core i3 M330 2.13GHz با حافظه اصلی ۳ گیگا بایت و حافظه جانبی ۵۰۰ گیگا بایت است. تمامی الگوریتم‌ها تحت سیستم عامل ویندوز ۷ نسخه Ultimate اجرا شده‌اند.

همان‌طورکه در بخش (۳) بیان شد، الگوریتم پیشنهادی دارای دو بهبود اساسی نسبت به الگوریتم موجود است که نتایج آزمایش‌ها برای هر مرحله از بهبود، ثبت شده است. نتایج به‌دست آمده از انجام آزمایش‌ها، پس از اعمال بهبود مرحله نخست در شکل (۳) نشان داده شده است. با توجه به نمودار مشخص است که الگوریتم پیشنهادی پس از بهبود مرحلهٔ نخست نیز، کارایی بهتری نسبت به الگوریتم‌های موجود، دارد.

پارامترهایی که در بررسی عملکرد الگوریتم‌ها مد نظر قرار گرفته شده است، به شرح زیر هستند:

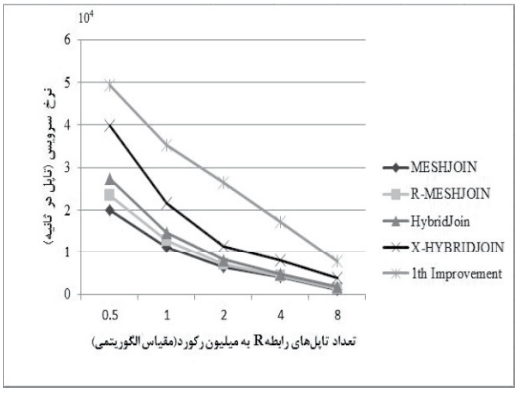
- پارامتر نخست تعداد رکوردهای موجود در رابطه R که بر روی لوح سخت قرار دارد، است. افزایش رکوردهای رابطه R تأثیر مستقیمی بر روی نرخ سرویس الگوریتم‌ها دارد؛ لذا در تمامی آزمایش‌ها این پارمتر از ۵/ میلیون رکورد تا هشت میلیون رکورد به‌صورت لگاریتمی افزایش داده شده و نتایج آزمایش‌ها ثبت شده است.
- پارامتر دوم تعداد مراجعات به لوح سخت است. همراه با افزایش رکوردهای رابطه R، تعداد مراجعات به لوح سخت در الگوریتم‌ها افزایش می‌یابد. نتایج به‌دست‌آمده در این مورد نیز ثبت شده است.

نتایج به‌دست‌آمده از بهبود مرحله دوم که درواقع همان الگوریتم نهایی است، در شکل (۴) نشان داده شده است. با توجه به نتایج به‌دست آمده، مشاهده می‌شود که نرخ سرویس به‌دست‌آمده برای الگوریتم پیشنهادی، اختلاف قابل توجهی با سایر الگوریتم‌های موجود دارد. دلیل این اختلاف در دو مورد زیر بیان می‌شود:

دلیل نخست، حذف صف موجود در بخش اصلی الگوریتم: برای هر جریان داده‌ای که وارد جدول H_S می‌شود،

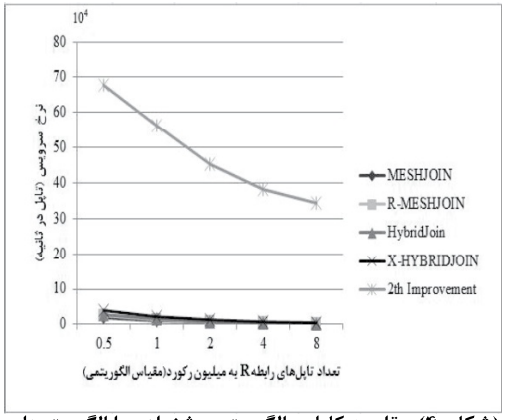
شناسه آن نیز در صف قرار داده می‌شود. همچنین برای هر جریان داده‌ای که از جدول Hs حذف می‌شود، باید شناسه آن نیز از صف یادشده حذف شود؛ چون ممکن است حذف شناسه از هر مکانی از صف صورت گیرد، صف مذکور با استفاده از فهرست پیوندی دوطرفه پیاده‌سازی شده است. در فهرست پیوندی دوطرفه برای عنصری که حذف می‌شود، چهار پیوند و برای عنصری که اضافه می‌شود، سه پیوند به‌روز می‌شود که عملی پرهزینه می‌باشد. در الگوریتم پیشنهادی صف مذکور حذف شده است و به‌جای آن آرایه‌ای یک‌بعدی استفاده شده است. در آرایه استفاده‌شده، در هنگام افزودن یک جریان داده به جدول Hs در زمان O(k) به خانهٔ مربوط به آن جریان داده در آرایه می‌توان دسترسی داشت. همچنین در هنگام حذف از جدول Hs هیچ مراجعه‌ای به آرایه وجود ندارد.

دلیل دوم، کاهش تعداد مراجعات به لوح سخت: در شکل (۵) تعدا مراجعات به لوح سخت در الگوریتم‌ها، نشان داده شده است که در الگوریتم پیشنهادی تعداد مراجعات بسیار کمتر از الگوریتم موجود است. همان‌گونه که در بخش قبلی بیان شد، در هر دور اجرای الگوریتم، پارتیشنی انتخاب می‌شود که دارای بیشترین تعداد رکورد در جدول Hs است. درواقع استفاده از این پیوند سبب می‌شود که با کمترین تعداد مراجعه به لوح سخت، بیشترین رکورد ممکن پیوست شود. با توجه به اینکه مراجعه به لوح سخت بخاطر عملیات I/O، عملی پرهزینه است، کاهش تعداد مراجعات به آن سبب کاهش زمان پیوست می‌شود. در نتیجهٔ این کاهش، نرخ سرویس الگوریتم به میزان قابل توجهی افزایش می‌یابد.



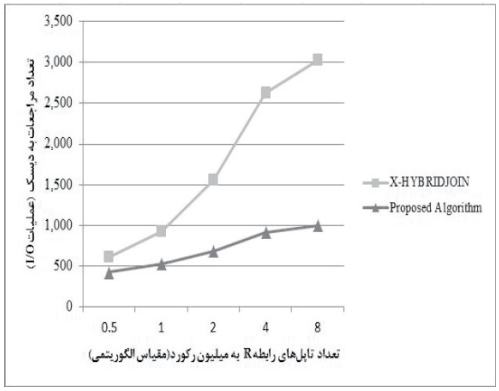
(شکل-۳): مقایسه کارایی الگوریتم پیشنهادی با الگوریتم‌های موجود پس از بهبود مرحله اول

(Figure-3): Compare the performance of the proposed algorithm after the first improvement



(شکل-۴): مقایسه کارایی الگوریتم پیشنهادی با الگوریتم‌های موجود پس از بهبود مرحله دوم

(Figure-4): Compare the performance of the proposed algorithm after the second improvement



(شکل-۵): مقایسه تعداد مراجعات به لوح سخت در الگوریتم‌ها

(Figure-5): Compare the number of disk access

۵- نتیجه‌گیری و کارهای آینده

در این مقاله الگوریتم جدیدی برای عملیات پیوست در پایگاه داده تحلیلی نیمه‌آنی، ارائه شد. الگوریتم پیشنهادی، عملیات پیوست موجود را، در دو مرحله بهبود داده است. در مرحله نخست از بهبود، از یک جدول درهم‌سازی برای نگهداری پارتیشن مقیم در حافظه استفاده شده است. زمانی که جریان داده‌ای می‌خواهد وارد محدوده پیوست شود، ابتدا در این جدول جستجو می‌شود. اگر متناظر آن یافت شود، عمل پیوست همان لحظه انجام می‌شود و خروجی تولید می‌شود. استفاده از این روش سبب می‌شود تا جریان داده‌هایی که در پارتیشن مقیم وجود دارند، به محدوده پیوست وارد نشوند که این باعث کاهش زمان پیوست می‌شود. در مرحله دوم از بهبود، نحوه انتخاب پارتیشنی که قرار است در بخش قابل تعویض لوح سخت قرار گیرد، تغییر داده شد. برای انتخاب

پارتیشن یادشده از یک آرایه یک‌بعدی استفاد شده که در آن اندیس هرخانه نشان‌دهندهٔ شماره پارتیشن متناظر در رابطه R و محتوای آن نشـــان‌دهندهٔ تعداد رکوردهای موجود از هر پارتیشن در حافظه است. با استفاده از این آرایه در هر مرحله از انتخـاب، همواره بهترین پـارتیشـــن را برای بـارگزاری در حافظه می‌توان انتخاب کرد. بهترین پارتیشـــن، پارتیشـــنی است که بیشترین تعداد رکورد را در حافظه دارد.

مشکلی که با اجرای الگوریتم پیشنهادی مشاهده شد، انتظـار نـامحدود برخی از جریان داده‌های موجود در حافظه اسـت. با توجه به اینکه در هر بار انتخاب، بهترین پارتیشن از لوح سـخت انتخاب می‌شـود، این مشـکل به‌وجود می‌آید. در کار آینده مشـکل مطرح‌شـده با اسـتفاده از روش مناسـبی برطرف خواهد شد.

6- References ۶- مراجع

[۱] حضـــرتی آغبلاغ، عیســـی و دانشـــپور، نگین، "RX-HYBRIDJOIN: الگوریتمی بهبود یافته برای پایگاه داده تحلیلی نیمه‌آنی،" دهمین سـمپوزیوم پیشـرفت علوم و تکنولوژی، مشهد، موسسه آموزش عالی خاوران. ۱۳۹۴.

[1] I. Hazrati, N. Daneshpour, "RX-HYBRIDJOIN: improved algorithm for near-real-time data warehouse," presented at the 10th Symposium on the Advancement of Science and Technology, Mashhad, Khavaran Higher Education Institution, 2015.

[۲] حضـــرتی آغبلاغ، عیســـی و دانشـــپور، نگین، "IX-HYBRIDJOIN: الگوریتمی بهبود یافته برای پایگاه داده تحلیلی نیمه‌آنی،" مقاله منتشـر شده در بیست و یکمین کـنـفرانس ملی کـامپیوتر ایران، تهران، پژوهشـــکـده دانش‌های بنیادین. ۱۳۹۴.

[2] I. Hazrati, N. Daneshpour, "IX-HYBRIDJOIN: improved algorithm for near-real-time data warehouse," presented at the 21th National Computer Conference of Iran, Tehran, Institute of Basic Sciences, 2015.

[3] A. Nguyen and A. Tjoa, "Zero-latency data warehousing for heterogeneous data sources and continuous data streams," Paper presented at the 5th International Conference on Information Integration and Web-based Applications Services, Austrian, 2003, pp. 55–64.

[4] A. Gupta, F. Yang, J. Govig, A. Kirsch, K. Chan, K. Lai, S. Wu, S. G. Dhoot, A. R. Kumar, A. Agiwal, S. Bhansali, M. Hong, J. Cameron, M. Siddiqi, D. Jones, J. Shute, A. Gubarev, S. Venka-taraman, and D. Agrawal, "Mesa: geo-replicated, near real-time, scalable data warehouse-ing," presented at the 40th International Conf-erence on Very Large Data Bases, China, 2014, pp. 1259-1270.

[5] A. Karakasidis, P. Vassiliadis, and E. Pitoura, " ETL queues for active data warehousing," presented at the 2th International Workshop on Information Quality in Information Systems, New York, 2005, pp. 28–39.

[6] C. Anderson, The Long Tail: Why the Future of Business is Selling Less of More, Hyperion, 2009.

[7] F. Dehne, Q. Kong, A. Rau-Chaplin, H. Zaboli, and R. Zhou, "Scalable real-time OLAP on cloud architectures," Journal of Parallel and Distributed Computing, vol. 79-80, pp. 31-41, 2015.

[8] F. Dehne, Q. Kong, A. Rau-Chaplin, H. Zaboli, and R. Zhou, "Distributed Tree Data Structure For Real-Time OLAP On Cloud Architectures," presented at the International Conference on Big Data, Silicon Valley, 2013, pp. 499-505.

[9] F. Majeed and S. Mahmood, "Efficient data streams processing in the real time data ware-house," presented at the 3rd IEEE Interna-tional Conference on Computer Science and Infor-mation Technology, Chengdu, 2010, pp. 57-61.

[10] F. Majeed, S. Mahmood, S. Ubaid, N. Khalil, S. Siddiqi, and F. Ashraf, "A burst resolution technique for data streams management in the real-time data warehouse," presented at the 7th Internat-ional Conference on Emerging Technologies, Islamabad, 2011, pp. 1-5.

[11] H. Zhou, D. Yang, and Y. Xu, "An ETL strategy for real-time data warehouse," presented at the International Conference on Intelligent Systems and Knowledge Engineering, Shanghai, 2011, pp. 329–336.

[12] H. Alzeini, SH. Hameed, and M. Habaebi, "A framework for developing real-time OLAP algorithm using multi-core processing and GPU: heterogeneous computing," presented at the 5th International Conference on Mechatronics, Kuala Lumpur. 2013.

[13] L. Golab, T. Johnson, J. S. Seidel, and V. Shkapenyuk, "Stream warehousing with data depot," presented at the 35th SIGMOD Interna-tional Conference on Management of Data, Rhode Island, 2009, pp. 847–854.

[14] L. Chen, W. Rahayu, and D. Taniar, "Towards near real-time data warehousing," presented at the 24th IEEE International Conference on Advanced Information Networking and Applications, Perth, 2011, pp. 1150-1157.

[15] M. Obal, B. Dursun, Z. Erdem, and A. Kadir, "A real-time data warehouse approach for data processing," presented at the Signal Processing

and Communications Applications Conference, Haspolat, 2013, pp. 1-4.

[16] M. A. Naeem, G. Dobbie, and G. Weber, "X-HYBRIDJOIN for near-real-time data warehousing," presented at the 28th British National Conference on Databases, Manchester, 2011, pp. 33–47.

[17] M. A. Naeem, G. Dobbie, and G. Weber, "A lightweight stream-based join with limited resource consumption" presented at the 14th International Conference DaWaK, Vienna, 2011, pp. 431-442.

[18] M. A. Naeem, G. Dobbie, and G. Weber, "Hybridjoin for near-real-time data warehousing," International Journal of Data Warehousing and Mining, vol. 7, no. 4, pp. 21-42, 2011.

[19] M. A. Naeem, G. Dobbie, and G. Weber, "An event-based near real-time data integration archite-cture," presented at the Enterprise Distributed Object Computing Conference Workshops, Munich, 2008, pp. 401–404.

[20] M. A. Naeem and N. Jamil, "An efficient stream-based join to procees end user transactions in real-time data warehousing," Journal of Digital Infor-mation Management, vol. 3, pp. 201-215, 2014.

[21] M. Thiele and W. Lehner, "Evaluation of load scheduling strategies for real-time data warehouse environments," presented at the 35th International Conference on Very Large Databases, Lyon, 2009, pp. 84-99.

[22] N. Polyzotis, S. Skiadopoulos, P. Vassiliadis, A. Simitsis, and N. Frantzell, "Meshing Streaming Updates with Persistent Data in an Active Data Warehouse," IEEE Transactions on Knowledge and Data Engineering, vol. 20, issue. 7, pp. 976-991, 2008.

[23] R. Abrahiem, "A new generation of middleware solutions for a near-real-time data warehousing architecture," presented at the 2007 IEEE International Conference on Electro/Information Technology, Chicago, 2007, pp. 192-197.

[24] S. Sudha and S. Manikandan, "M-hybridjoin- an adaptive approach for stream based near real-time data warehousing," International Journal of Ad-vanced Engineering Technology, vol. 7, issue 1, pp. 321-326, 2016.

[25] T. Jorg, and S. Dessloch, "Near real-time data warehousing using state-of-the-art ETL tools," presented at the 35th International Conference on Very Large Databases, Lyon. 2009.

[26] W. J. Labio, J. L. Wiener, H. Garcia, and V. Gorelik, "Efficient resumption of interrupted ware-house loads," SIGMOD Rec. vol. 29, no. 2, pp. 46–57, 2000.

[27] W. J. Labio, J. Yang, Y. Cui, H. Garcia, and J. Widom, "Performance issues in incremental warehouse maintenance," presented at the 26th International Conference on Very Large Data Bases, San Francisco, 2000, pp.461–472.

[28]] M. A. Naeem, G. Dobbie, and G. Weber, "Efficient usage of memory resources in near-real-time data warehousing," presented at the Emerging Trends and Applications in Information Communi-cation Technologies, Pakistan, 2012, pp. 326-337.

[29] M. A. Naeem, G. Dobbie, and G. Weber, "Optimised X-HYBRIDJOIN for near-real-time data warehousing" presented at the 23th Austra-lasian Database Conference, Melbourne, 2012, pp. 21-30.

عیسی حضرتی مدرک کارشناسی خود را در سال ۱۳۸۸ از دانشگاه ارومیه اخذ و همچنین مدرک کارشناسی ارشد خود را در سال ۹۴ از دانشگاه تربیت دبیر شهید رجایی تهران دریافت کرده است. زمینه‌های پژوهشی مورد علاقه ایشان عبارتند از: پایگاه داده تحلیلی، داده‌کاوی و پیش‌پردازش داده.

نشانی رایانامهٔ ایشان عبارت است از:

i.hazrati@sru.ac.ir

نگین دانشپور مدرک کارشناسی خود را در سال ۱۳۷۷ از دانشگاه شهید بهشتی تهران و درجهٔ کارشناسی ارشد و دکترای خود را از دانشگاه امیرکبیر در سال ۱۳۸۰ و ۱۳۸۸ اخذ کرده است. در حال حاضر، وی عضو هیئت علمی و استادیار دانشکده کامپیوتر دانشگاه شهید رجایی تهران است. زمینه‌های پژوهشی مورد علاقه ایشان عبارتند از: پایگاه دادهٔ تحلیلی، داده‌کاوی و پیش‌پردازش داده.

نشانی رایانامهٔ ایشان عبارت است از:

ndaneshpour@sru.ac.ir