

بهبود الگوریتم انتخاب دید در پایگاه داده تحلیلی با استفاده از یافتن پرس وجوهای پرتکرار

ریحانه صباغ گل و نگین دانشپور*

دانشکده مهندسی کامپیوتر، دانشگاه تربیت دبیر شهید رجایی، لویزان، تهران، ایران



چکیده

پایگاه داده تحلیلی منبعی برای ذخیره سازی داده های تاریخی جهت تحلیل است. به طور معمول زمان پاسخ به پرس وجوهای تحلیلی، زمانی طولانی است. استفاده از دید به جای دسترسی مستقیم به پایگاه داده، سرعت پاسخ گویی را بهبود می دهد. راه کارهای مختلفی برای ذخیره سازی دید وجود دارد؛ که مناسب ترین راهکار برای ذخیره سازی دید، ذخیره سازی دیدهای پر استفاده و پر کاربرد است. پرس وجوهای که در قبل مورد استفاده پایگاه داده تحلیلی بوده اند، حاوی اطلاعات مهمی هستند که به احتمال زیاد در آینده نیز مورد استفاده خواهند بود. این مقاله، الگوریتمی برای ذخیره سازی دیدهای پر کاربرد ارائه می دهد. این الگوریتم با استفاده از پرس وجوهای قبلی، دیدهای پر کاربرد را یافته و آنها را ذخیره می کند. این دیدها توانایی پاسخ گویی را به بسیاری از پرس وجوهای که در آینده اتفاق خواهند افتاد، دارند. روش پیشنهادی این مقاله از الگوریتم Index-BittableFI برای یافتن دیدهای پرتکرار استفاده کرده است که باعث بهبود روش های قبلی و کاهش زمان پاسخ به پرس وجوها شده است. آزمایش های انجام شده نشان می دهند که الگوریتم پیشنهادی از لحاظ زمانی نسبت به الگوریتم های قبلی ۲۳ درصد و از لحاظ فضای ذخیره سازی پنجاه درصد بهبود داشته است.

واژگان کلیدی: پایگاه داده تحلیلی، پرس وجوهای پرتکرار، خوشه بندی، ذخیره سازی دید

An Improved View Selection Algorithm in Data Warehouses by Finding Frequent Queries

Reyhaneh Sabbagh Gol and Negin Daneshpour*

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

Abstract

A data warehouse is a source for storing historical data to support decision making. Usually analytic queries take much time. To solve response time problem it should be materialized some views to answer all queries in minimum response time. There are many solutions for view selection problems. The most appropriate solution for view selection is materializing frequent queries. Previously posed queries on the data warehouse have profitable information. These queries probably will be used in the future. So, previous queries are clustered using clustering algorithms. Then frequent queries are found using data mining algorithms. Therefore optimal queries are found in each cluster. In the last stage optimal queries are merged to produce one (query) view for each cluster, and materializes this view. This paper proposes an algorithm for materializing frequent queries. The algorithm finds profitable views using previously posed queries on the data warehouse. These views can answer the most of the queries being posed in the future. This paper uses

Index-BittableFI algorithm for finding frequent views. Using this algorithm improves previous view selection algorithms and reduces the response time. The experiments show that the proposed algorithm has %23 improvement in response time and %50 improvement in storage space.

Keywords: Data warehouse, Frequent queries, View materialization, Clustering

اجرای بالایی است و سرعت پاسخ به پرس و جوهای تحلیلی بسیار کم است. از طرفی دیگر با توجه به این که مهم ترین عامل در الگوریتم های انتخاب دید، زمان است، بنابراین در این مقاله، الگوریتمی پیشنهاد شده است که در زمان کمتری، به پرس و جوهای تحلیلی، پاسخ می دهد.

الگوریتم انتخاب دید پیشنهاد شده در این مقاله، با استفاده از الگوریتم Index-BittableFI [4] پرس و جوهای پرتکرار را در هر خوشه به دست می آورد. الگوریتم ارائه شده در [2]، [3] و الگوریتم پیشنهادی این مقاله، پیاده سازی شده اند و مشاهده شده است که الگوریتم پیشنهادی نسبت به الگوریتم های بیان شده در منبع [2]، [3] بهبود داشته است.

ساختار این مقاله به شرح زیر است: در بخش ۲ مقالات و الگوریتم های مرتبط با این مقاله ارائه می شود. در بخش ۳ الگوریتم انتخاب دیدی که این مقاله ارائه می دهد بیان شده است. بخش ۴، به شرح پیاده سازی الگوریتم می پردازد و بخش ۵ به نتیجه گیری می پردازد.

۲-پیشینه پژوهش

از جمله نخستین الگوریتم های مطرح شده برای انتخاب دید می توان به الگوریتم حریصانه که در سال ۹۷ مطرح شده است، اشاره کرد [5]. الگوریتم حریصانه به این صورت عمل می کند که با استفاده از تابع سودی که تعریف شده است، دیدهای با سود بیش تر را انتخاب و با توجه به محدودیت های موجود از جمله فضای نگهداری آن دیدها را نگهداری می کند. البته این الگوریتم ها برای مسائل با ابعاد کم مناسب هستند؛ ولی برای مسائل با ابعاد بزرگ مناسب نیستند.

برخی دیگر از روش ها بر اساس گرافی که رسم شده است تصمیم گیری می کنند، برخی از این گراف ها عبارتند از گراف AND-OR یا گراف MVPP^۹. بعد از رسم این گراف ها بر اساس تابع هزینه ای که تعریف شده است، تصمیم گیری می شود که کدام دیدها انتخاب و ذخیره شوند [8]-[6]. این

۱-مقدمه

امروزه سازمان ها و ادارات بزرگ دارای حجم بسیار بالایی از داده ها هستند. این داده ها در پایگاه داده های مختلفی ذخیره شده اند. برای دسترسی به این داده ها دو روش وجود دارد: روش تنبل^۱ و روش مشتاق^۲. در روش نخست داده ها به وسیله پرس و جوی کاربر یک پارچه^۳ می شوند. این روش، به علت این که داده ها پس از اجرای پرس و جوی کاربر یک پارچه می شوند، دارای تأخیر زیادی است و در نتیجه، روش مناسبی نیست. روش دوم از پایگاه داده تحلیلی استفاده می کند. در این روش اطلاعات مورد نیاز از پایگاه های داده مختلف استخراج^۴ شده و پس از پاک سازی داده ها، در منبع بزرگی به نام پایگاه داده تحلیلی بارگذاری^۵ می شوند [1]. در این روش برای تصمیم گیری^۶ از پرس و جوهای تحلیلی استفاده می شود. امروزه بیش تر سازمان ها و ادارات از این روش برای پاسخ به پرس و جوهای تحلیلی تحلیل گران استفاده می کنند [2].

پایگاه داده تحلیلی به منظور کاهش زمان پاسخ به پرس و جوهای تحلیلی، از دید ذخیره شده^۷ استفاده می کند. با توجه به محدودیت فضای ذخیره سازی، و نیز زمان زیاد به روزنگه داشتن دیدها، نمی توان تمام دیدهای ممکن را ذخیره کرد؛ بنابراین باید از بین تمامی دیدهای ممکن، تنها مجموعه ای از دیدها را ذخیره کرد. دیدها می توانند بر اساس پرس و جوهایی که در قبل مورد استفاده پایگاه داده تحلیلی بوده اند، انتخاب و ذخیره شوند [3]، [2]. علت استفاده از پرس و جوهای قبلی، این است که به احتمال زیاد در آینده دوباره اتفاق خواهند افتاد. روند کار الگوریتم های بیان شده در [3]، [2] به این صورت است که ابتدا پرس و جوهای قبلی خوشه بندی می شوند؛ سپس در هر خوشه پرس و جوهای پرتکرار به دست می آیند. در مرحله بعد پرس و جوهای بهینه انتخاب می شوند و در نهایت پرس و جوهای بهینه در هر خوشه پیوند^۸ می شوند. این الگوریتم به علت این که برای یافتن پرس و جوهای پرتکرار در هر خوشه از الگوریتم های مقدماتی همچون Apriori و DIC استفاده کرده است، دارای زمان

¹ lazy
² eager
³ integrate
⁴ extraction
⁵ load

⁶ Decision making
⁷ Materialized View
⁸ Join
⁹ Multi-View Processing Plan

الگوریتم‌ها با توجه به این که باید گراف‌های مذکور رسم شوند، کمی پیچیده هستند و برای مسائل با ابعاد بزرگ مناسب نیستند.

برخی دیگر از الگوریتم‌ها تأکید زیادی روی تابع هزینه دارند [9]. این روش‌ها تابع سود یا هزینه‌ای تعریف کرده و دیدهایی با بیش‌ترین سود و کم‌ترین هزینه را انتخاب و ذخیره‌سازی می‌کنند. از جمله عواملی که در [9] مطرح شده است، عبارتند از: فرکانس تکرار هر پرس‌وجو، زمان اجرای هر پرس‌وجو، تعداد پیوندها و مجتمع‌های^۱ موجود در هر پرس‌وجو، هزینه‌ی نگه‌داری دید، فرکانس به‌روزرسانی جداول پایه، تعداد درج‌ها، تعداد تغییرات و تعداد حذف‌ها.

با استفاده از روش بیان‌شده در [10]، گراف بدون دور مستقیم^۲ مربوط به هر دید رسم شده و پس از یافتن کوتاه‌ترین مسیر، دیدهای با بیش‌ترین سود انتخاب و ذخیره می‌شوند. در این گراف یال‌ها، دیدها می‌باشند. البته این الگوریتم برای مسائل با ابعاد کم مناسب و برای مسائل با ابعاد بزرگ، پیچیده است.

الگوریتم ارائه‌شده در [11] پرس‌وجوی بعدی را پیش‌بینی کرده و بر اساس این پیش‌بینی دیدهای مورد نیاز را ذخیره‌سازی می‌کند. پیش‌بینی کردن پرس‌وجوی بعدی باعث می‌شود، دید مورد نیاز برای آن پرس‌وجو، به‌درستی ذخیره شود؛ اما اگر تابع پیش‌بینی، پرس‌وجوی بعدی را اشتباه پیش‌بینی کند، زمان زیادی برای پاسخ‌گویی به آن پرس‌وجو صرف خواهد شد.

برخی دیگر از الگوریتم‌ها از مدل‌های ریاضی استفاده می‌کنند. از جمله این الگوریتم‌ها عبارتند از روش‌های انتخاب دید با استفاده از مسئله ارضای محدودیت‌ها^۳ [14]-[12].

برخی دیگر از این الگوریتم‌ها، با استفاده از روش برنامه‌نویسی صحیح^۴ مسئله را مدل کرده و با استفاده از ابزارهای حل مسئله ریاضی، مسئله انتخاب دید را که به‌صورت فرمول درآمده است، حل کرده و دیدهای بهینه را انتخاب و ذخیره‌سازی می‌کنند [16]، [15]. این الگوریتم‌ها با توجه به این که مسئله را با استفاده از معادلات ریاضی حل می‌کنند، جزء قدرتمندترین روش‌های حل مسائل هستند. اما با توجه به این که محدودیت‌های مورد نظر را برای هر پرس‌وجو در نظر می‌گیرند، برای مسائل با ابعاد بزرگ پیچیده است و دارای زمان اجرای بالایی خواهند بود.

برخی از الگوریتم‌ها، بهبودیافته الگوریتم حریصانه هستند. این الگوریتم‌ها علاوه بر معیارهای موجود در الگوریتم حریصانه، بر اساس اندازه پرس‌وجو، فرکانس تکرار پرس‌وجوها و میزان توانایی پرس‌وجو در تصمیم‌گیری^۵ دیدهای مناسب را انتخاب کرده و ذخیره‌سازی می‌کنند [17]. این الگوریتم‌ها با توجه به این که از شبکه‌ای از کعب^۶‌ها استفاده می‌کنند، برای مسائل با ابعاد کوچک مناسب هستند.

الگوریتم [18] با لحاظ کردن مقادیر تکرار هر پرس‌وجو و استفاده از فرمول سود جدید توانسته است بهبودی نسبت به الگوریتم حریصانه حاصل کند. این الگوریتم در ابتدا شبکه پایگاه داده تحلیل را همراه با اندازه هر کعب و تکرار هر پرس‌وجو می‌گیرد؛ سپس با استفاده از فرمول سود هر دید، دیدهای با سود بیشتر را به‌صورت حریصانه انتخاب می‌کند. این الگوریتم هزینه نگهداری دیدهای ذخیره‌شده را در نظر نمی‌گیرد. همچنین به‌علت پیچیدگی برای مسائل بزرگ، این الگوریتم برای پایگاه داده تحلیل با تعداد بعد زیاد مناسب نیست.

الگوریتم [19] همانند الگوریتم حریصانه دیدهایی با بهترین سود را برای ذخیره‌سازی انتخاب می‌کند؛ با این تفاوت که مقادیر تکرار هر پرس‌وجو^۷ را نیز در فرمول سود خود لحاظ کرده است. این الگوریتم هزینه نگهداری دیدهای ذخیره‌شده را در نظر نمی‌گیرد. همچنین به‌علت پیچیدگی برای مسائل بزرگ، این الگوریتم برای پایگاه داده تحلیل با تعداد بعد زیاد مناسب نیست.

الگوریتم [20] دارای چندین بخش است. در این الگوریتم ابتدا پرس و جوها بر اساس معیار شباهت و الگوریتم خوشه‌بندی سلسله‌مراتبی، دسته‌بندی می‌شوند. در مرحله بعد، پس از اجرای چندین زیر برنامه دیدهای منتخب برای ذخیره‌سازی معین می‌شوند.

الگوریتم بیان‌شده در [21] با استفاده از جفت‌گیری زنبور عسل دیدهای مناسب را انتخاب و ذخیره می‌کند. این الگوریتم در مقایسه با الگوریتم حریصانه دارای کارایی بالاتری است.

الگوریتم بیان‌شده در [22] با استفاده از روش‌های داده‌کاوی پرس و جوهای پرتکرار را می‌یابد. در این الگوریتم با تغییر داده‌ها، تمامی جدول‌ها نیز به‌روزرسانی می‌شوند. این الگوریتم نسبت به الگوریتم حریصانه بهتر عمل می‌کند.

⁵ Decision making

⁶ Lattice of cuboids

⁷ Query Frequency

¹ Aggregate

² DAG

³ Constraint Satisfaction Problem

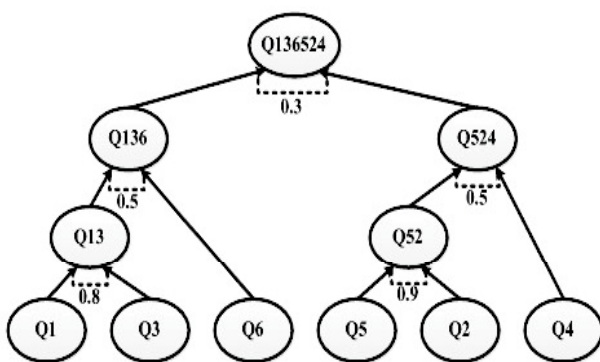
⁴ integer programming

حاوی اطلاعات مفیدی هستند که به احتمال زیاد در آینده نیز مورد استفاده خواهند بود. در الگوریتمی که در ادامه توضیح داده می‌شود، روش یافتن پرس‌وجوهای پرتکرار در روش MVCF بیان شده در منبع [2],[3] بهبود داده شده که باعث بهبود نهایی الگوریتم انتخاب دید با استفاده از روش MVCF است. الگوریتم بیان شده در منابع [2],[3] با توجه به نام نویسندگان، الگوریتم KD2011 و KD2013 نامیده می‌شود و الگوریتم پیشنهادی در این مقاله، الگوریتم SRTTU-2015 نامیده شده است. الگوریتم پیشنهادی در ادامه توضیح داده می‌شود.

ابتدا پرس‌وجوهای قبلی با استفاده از روش سلسله‌مراتبی^۴، دسته‌بندی می‌شوند. به هریک از این دسته‌ها، خوشه^۵ یا دامنه^۶ گفته می‌شود. برای خوشه‌بندی کردن پرس‌وجوها از معیار شباهت جاکارد^۷ [2] استفاده می‌شود.

برای یافتن خوشه‌هایی که دارای پرس‌وجوهای پرتکرار هستند، ابتدا باید الگوریتم ادغام پرس‌وجوها^۸ [2] اجرا شود تا براساس معیار شباهت جاکارد، پرس‌وجوها با هم ادغام شوند؛ سپس براساس ترتیب ادغام پرس‌وجوها درخت آن را تشکیل داده و خوشه‌ها از روی درخت یافته شوند.

برای ساخت درخت از روی الگوریتم ادغام پرس‌وجوها، روند ساخت خوشه‌ها به صورت درخت رسم می‌شود. با توجه به الگوریتم ادغام پرس‌وجوها، خوشه‌ها براساس بیش‌ترین میزان شباهت به هم از پایین به بالا ادغام می‌شوند تا این‌که در آخرین مرحله همه پرس‌وجوها در یک خوشه قرار می‌گیرند و در نهایت درخت ساخته می‌شود. نمونه‌ای از این درخت در شکل (۱) آمده است.



شکل-۱: درخت ساخته شده از روی الگوریتم ادغام پرس‌وجوها
(Figure-1): The tree made from merging queries' algorithm

⁵ cluster

⁶ domain, area, subject area

⁷ Jaccard's coefficient

⁸ Query Merger

الگوریتم معرفی شده در [23] تعدادی دید نماینده^۱ انتخاب می‌کند؛ سپس عملیات حذف و اضافه کردن دیده‌ها انجام می‌شود؛ به این صورت که دیده‌هایی که هزینه بالایی دارند، حذف شده و دیده‌هایی که هزینه کمتری دارند اضافه می‌شوند. در صورتی که این الگوریتم دیده‌های نماینده را به درستی انتخاب نکند، زمان پاسخ الگوریتم بسیار بالا خواهد بود.

در برخی دیگر از روش‌ها با استفاده از الگوریتم‌های خوشه‌بندی و داده‌کاوی، دیده‌های مناسب انتخاب می‌شوند [26]-[24],[3],[2]. در این الگوریتم‌ها از پرس‌وجوهای قبلی استفاده می‌شود؛ علت این امر، این است که به احتمال زیاد در آینده نیز، این پرس‌وجوها اتفاق خواهند افتاد. این الگوریتم‌ها از روش MVCF^۲ استفاده می‌کنند. این معماری از چهار مرحله تشکیل شده است:

(الف) ابتدا با استفاده از روش‌های خوشه‌بندی،

پرس‌وجوهای قبلی خوشه‌بندی می‌شوند.

(ب) سپس با استفاده از روش‌های داده‌کاوی برای یافتن آیتم‌های پرتکرار، در هر خوشه پرس‌وجوهای پرتکرار یافته می‌شوند.

(ج) سپس در هر خوشه، پرس‌وجوهای بهینه انتخاب می‌شوند.

(د) در این مرحله، در هر خوشه پرس‌وجوهای بهینه ادغام می‌شوند تا به‌ازای هر خوشه، یک دید به‌دست آید و آن دید ذخیره می‌شود.

در [3],[2] از الگوریتم‌های پایه مانند الگوریتم Apriori و DIC^۳ برای یافتن آیتم‌های پرتکرار در هر خوشه استفاده شده است و این امر باعث شده تا الگوریتم انتخاب دید، کارایی مناسبی نداشته باشد. بنابراین در این مقاله از الگوریتم جدیدتری برای پیاده‌سازی مراحل روش MVCF استفاده شده است تا الگوریتم انتخاب دید کارایی بالاتری داشته باشد.

۳- الگوریتم انتخاب دید پیشنهادی

در این بخش الگوریتم ارائه شده در این مقاله برای انتخاب دید توصیف می‌شود. این الگوریتم از روش MVCF که در بخش کارهای مرتبط توضیح داده شده است، استفاده می‌کند. در این الگوریتم از پرس‌وجوهای که در قبل مورد استفاده پایگاه داده تحلیلی بوده‌اند، استفاده شده است؛ زیرا این پرس‌وجوها

¹ Candidate

² Materialized Views Construction Framework

³ Dynamic Itemset Counting

⁴ Hierarchical clustering

به عبارت دیگر آرایه اندیسی، یک آرایه با اندازه m_1 است که m_1 تعداد l -itemset های پرتکرار است. هر عضو این آرایه به صورت زوج دوتایی ($item, subsume$) است، که $item$ l -itemset است و $subsume$ در فرمول (۲) توصیف شده است.

$$subsume(item) = \{j \in I \mid item \prec j \wedge g(item) \subseteq g(j)\} \quad (2)$$

که $subsume(item)$ به این معنی است که اگر $j \in subsume(item)$ باشد، آن گاه بر اساس ترتیب $\prec$ ، j بعد از $item$ قرار دارد. برای پیدا کردن آیتم های پرتکرار، ابتدا باید در هر خوشه، آرایه اندیسی برای هر آیتم به دست آید؛ سپس با در دست داشتن آرایه اندیسی تمام آیتم ها، با استفاده از الگوریتم Index-BitableFI آیتم های پرتکرار در هر خوشه به دست آورده شوند. در قسمت (الف) الگوریتم ساختن آرایه اندیسی توضیح داده شده است و در قسمت (ب) الگوریتم Index-BitableFI توضیح داده شده است. اساس الگوریتم های بیان شده در قسمت های الف و ب بر پایه ی منبع [4] است و در این مقاله برای این مسأله بازنویسی شده است.

الف) الگوریتم ساختن آرایه اندیسی:

ابتدا الگوریتم ساختن آرایه اندیسی مورد بررسی قرار می گیرد. این الگوریتم در شکل (۲) بیان شده است.

```

Procedure IndexArray
Input: The set of queries in each cluster  $D_i, min\_sup$ .
Output: index array for each cluster  $index\_array[i]$ 
Begin
for each cluster  $i$  do
    Scan database  $D_i$  once. Delete infrequent items;
    Sort frequent single items in supports ascending order as
     $a_1, a_2, \dots, a_m$ ;
    for each element  $index[j]$  of  $index\_array[i]$  of cluster  $i$  do
         $index[j].item = a_j$ ;
    Represent the database  $D_i$  with BitTable;
    for each element  $index[j]$  in  $index\_array[i]$  of cluster  $i$  do
         $index[j].subsume = \emptyset$ ;
         $candidate = \bigcap_{t \in g(index[j].item)} t$ ;
        for each  $f > j$  do
            if ( the value of the  $f$ -th bit in  $candidate$  is set) then
                 $index[j].subsume \leftarrow index[f].item$ ;
            end if
        end for
    end for
Write Out  $index\_array[i]$ ;
end for
End
    
```

(شکل-۲): الگوریتم ایجاد آرایه ی اندیسی برای الگوریتم SRTTU-2015
(Figure-2): Producing Index Array for SRTTU-2015 algorithm

پس از ساخت درخت، گره های درخت از بالا به پایین ملاقات^۱ می شوند تا زمانی که خوشه ای پیدا شود که درجه شباهتی بیش تر از حداقل شباهت حد آستانه آ داشته باشد. این خوشه ها نشان دهنده دامنه ها خواهند بود. به عنوان مثال در شکل (۱)، اگر حد آستانه 0.4 باشد خوشه ها Q136 و Q524 خواهند بود. توجه شود که Q13 به این معنی است که این خوشه شامل Q1 و Q3 است.

پس از یافتن خوشه ها، باید پرس وجوهای پرتکرار در هر خوشه یافته شوند. روش یافتن پرس وجوهای پرتکرار برای هر خوشه، در الگوریتم های KD2011 و KD2013، به ترتیب روش Apriori و DIC است؛ الگوریتم های Apriori و DIC الگوریتم هایی قدیمی هستند و سبب می شوند که در زمان اجرای بالایی به پرس وجوهای تحلیلی پاسخ داده شود. از طرفی دیگر مهم ترین عامل در الگوریتم های انتخاب دید، زمان اجرا است؛ بنابراین در الگوریتم پیشنهادی این مقاله، از الگوریتم Index-BitableFI [4] برای بهبود زمان اجرا استفاده شده است. این الگوریتم برای پرس وجوهای هر خوشه بررسی می شود تا پرس وجوهای پرتکرار در هر خوشه به دست آیند. در این مرحله، هر کدام از پرس وجو ها را یک مجموعه آیتم^۳ می نامند. یک مجموعه آیتم که دارای k تا آیتم است، k -itemset نامیده می شود. با توجه به این که هر پرس و جو در قسمت from خود دارای تعدادی جدول است، بنابراین هر کدام از این جداول، یک آیتم نامیده می شود. $sup(X)$ نشان دهنده تعداد پرس وجو هایی است که شامل آیتم X هستند. اگر $sup(X) > min_sup$ باشد، آنگاه X یک آیتم پرتکرار نامیده می شود.

در الگوریتم Index-BitableFI، برای کاهش فضای جستجو، آرایه اندیسی^۴ تعریف می شود. فرمول (۱) تعریف ریاضی آرایه اندیسی است.

$$g(X) = \{t \in D \mid \forall i \in X, i \in t\} \quad (1)$$

که در آن X ، آیتمی است که آرایه اندیسی آن یافته می شود و D ، پایگاه داده ای است که شامل تمامی پرس وجو های هر خوشه است و t ، مجموعه ای از آیتم هایی است که هر پرس وجو شامل می شود. این آیتم ها فیلدهای مورد استفاده برای هر پرس وجو هستند.

¹ visit

² Minimum threshold

³ itemset

⁴ Index array

ظرفیت کوله‌پشتی، پرس‌وجوها را در آن می‌توان قرارداد [26]. پس از حل معادلات مسئله کوله‌پشتی، پرس‌وجوهای بهینه در هر خوشه به‌دست می‌آید.

```

Procedure IndexBitTableFI
Input: index array for each cluster  $index\ array[i]$ ,  $min\_sup$ 
Output: frequent itemsets
Begin
for each cluster  $i$  do
  for each element  $index[j]$  of  $index\ array[i]$  do
    Write Out  $index[j].item$  and its support;
    if ( $index[j].subsume == \emptyset$ ) then
      if ( $sup(index[j].item) > min\_sup$ ) then
        Depth_First ( $index[j].item$ ,  $t(index[j].item)$ );
        //  $t(index[j].item)$  is the set of frequent single items
        // that after  $index[j].item$ , according to support ascending
        // order
      end if
    else
      for each element  $s\text{-}item \in index[j].subsume$  do
        Write Out  $index[j].item \cup s\text{-}item$  and its support;
        // The support of any  $index[j].item \cup s\text{-}item$  equals to
        // support of  $index[j].item$ 
      end for
      if ( $sup(index[j].item) > min\_sup$ ) then
         $tail \leftarrow t(index[j].item) \setminus items\ in\ index[j].subsume$ ;
        // delete items included by  $index[j].subsume$  from
        //  $t(index[j].item)$ 
        Depth_First ( $index[j].item$ ,  $tail$ );
        for each element  $s\text{-}item \in index[j].subsume$  do
          Depth_First ( $index[j].item \cup s\text{-}item$ ,  $tail$ );
        end for
      end if
    end else
  end if
end for
end for
Procedure Depth_First ( $itemset$ ,  $tail$ )
if ( $tail == \emptyset$ ) then return;
for each  $i \in tail$  do
   $f\text{-}itemset \leftarrow itemset \cup i$ ;
  if ( $sup(f\text{-}itemset) \geq min\_sup$ ) then
    Write Out  $f\text{-}itemset$  and its support;
     $tail \leftarrow tail \setminus i$ ;
    Depth_First ( $f\text{-}itemset$ ,  $tail$ );
  end if
end for
End

```

(شکل-۳): الگوریتم Index-BitTableFI برای الگوریتم

پیشنهادی SRTTU-2015

(Figure-3): Index-BitTableFI algorithm that is needed for SRTTU-2015 proposed algorithm

به‌عنوان مثال نمونه‌ای از معادلات در رابطه (۳) بیان شده است.

$$\begin{aligned}
 & \text{Maximise } 13Q_1 + 15Q_7 + 28Q_{16} + 35Q_{17} + 32Q_{20} \\
 & \text{subject to} \\
 & 120Q_1 + 250Q_7 + 430Q_{16} + 310Q_{17} + 360Q_{20} \leq 1000 \\
 & \text{and} \\
 & Q_i = 0 \text{ or } 1 \text{ where } i = 1, 7, 16, 17, 20
 \end{aligned} \quad (3)$$

³ Optimal Queries

در ابتدا پایگاه داده مربوط به هر خوشه یعنی D_i پوش می‌شود تا آیتم‌های پرتکرار به‌دست آیند؛ سپس آیتم‌های پرتکرار به‌صورت صعودی مرتب می‌شوند و آیتم‌ها به‌ترتیب به هر عضو از آرایه اندیسی خوشه i نسبت داده می‌شوند. در مرحله بعد جدول بیتی^۱ از پایگاه داده D_i ساخته می‌شود. نحوه ساخت این جدول به این صورت است که اگر پرس‌وجوی T شامل آیتم f باشد، آن‌گاه بیت f ام مرتبط با پرس‌وجوی T ، یک خواهد شد؛ در غیر این صورت، صفر خواهد بود؛ سپس در آخرین حلقه، آرایه اندیسی تولید می‌شود. ابتدا تمام پرس‌وجوهای که شامل $index[j].item$ می‌شوند، اشتراک گرفته می‌شوند؛ سپس تمام آیتم‌هایی که دارای اندیس بالاتری نسبت به j هستند، در $index[j].subsume$ قرار می‌گیرد؛ در نهایت به‌عنوان خروجی برای هر خوشه، یک آرایه اندیسی برگردانده می‌شود.

ب) الگوریتم Index-BitTableFI

این الگوریتم با استفاده از آرایه اندیسی تولیدشده در هر خوشه، آیتم‌های پرتکرار را پیدا می‌کند. این الگوریتم در شکل (۳) ارائه شده است.

در ابتدا آیتم‌ها و $support$ آن‌ها نوشته می‌شود. در مرحله بعد $subsume$ مربوط به هر آیتم بررسی می‌شود. در صورت تهی بودن $subsume$ آن، اگر $support$ آن آیتم از min_sup بیش‌تر باشد، آنگاه تابع نخست‌عمق^۲ اجرا می‌شود. حال اگر $subsume$ آن غیرتهی باشد، تمام زیرمجموعه‌های غیر تهی $subsume$ را به‌دست آورده، سپس آیتم مذکور را با هر کدام از زیرمجموعه‌ها ادغام می‌کنیم. در این حالت هر کدام از مجموعه‌های به‌دست آمده یک مجموعه آیتم پرتکرار هستند که $support$ آن‌ها همان $support$ آیتم مذکور است. در اینجا نیز، اگر $support$ آیتم مذکور بیش‌تر از min_sup باشد، تابع نخست‌عمق صدازده می‌شود؛ سپس آیتم مورد نظر با تمام زیرمجموعه‌های غیر تهی ادغام شده و برای هر کدام تابع نخست‌عمق صدازده می‌شود.

پس از به‌دست آوردن پرس‌وجوهای پرتکرار در هر خوشه، باید پرس‌وجوهای بهینه^۳ با توجه به فضای موجود برای هر خوشه در هر خوشه به‌دست آیند. فرض می‌شود فضایی که می‌تواند برای ذخیره‌سازی دیدهای هر خوشه استفاده شود، فضای کوله‌پشتی باشد؛ در این صورت نمی‌توان تمام پرس‌وجوها را در کوله‌پشتی قرارداد و با توجه به محدودیت

¹ BitTable

² depth-first

می‌شوند، این معیار تعداد سطرهای دیده‌های ذخیره‌شده را نشان می‌دهد.

۲. زمان کل: این زمان مجموع دو مدت زمان است: اولی مدت زمانی است که الگوریتم انتخاب دید اجرا می‌شود؛ و دومی مدت زمانی است که پرس‌وجوهای آزمایش پاسخ داده می‌شوند.

۳. زمان آزمایش: این زمان، مدت‌زمان پاسخ به پرس‌وجوهای آزمایش است.

در این آزمایش‌ها برای به‌دست‌آوردن مقدار معیار مربوطه، پنج‌بار آزمایش‌ها تکرار شده و میانگین عددی پنج عدد حاصله، در نمودارهای این بخش ترسیم شده است.

تنها پارامتر موجود در این مسأله، پارامتر حد آستانه پوشش^۳ است. برای به‌دست‌آوردن بهترین مقدار پارامتر حد آستانه پوشش آزمایش‌های مختلفی انجام شده است. در این آزمایش‌ها تعداد پرس‌وجوهای ورودی و آزمایش، برابر پنجاه است. شکل (۶) نمودار تغییرات زمان کل در برابر تغییرات حد آستانه پوشش را نشان می‌دهد. در این نمودار محور افقی، مقدار حد آستانه پوشش و محور عمودی مقدار زمان کل است. همان‌طور که در شکل (۶) مشاهده می‌شود بهترین مقدار برای حد آستانه پوشش، $0/4$ است. بنابراین همان‌طور که در بخش ۳ توضیح داده شد، با استفاده از این پارامتر، خوشه‌ها مشخص می‌شوند.

شکل (۷) نشان‌دهنده مقایسه الگوریتم‌های KD2011 و KD2013 و SRTTU-2015 بر اساس معیار شماره (۱) است. در این نمودار محور افقی نشان‌دهنده تعداد پرس‌وجوها و محور عمودی نشان‌دهنده تعداد رکوردهای دیده‌های ذخیره‌شده نهایی است.

با توجه به نمودار رسم‌شده در شکل (۷) مشخص می‌شود که تعداد سطرهای دیده‌های ذخیره‌شده توسط الگوریتم پیشنهادی SRTTU-2015 همیشه کمتر از تعداد سطرهای به‌دست آمده توسط الگوریتم‌های KD2011 و KD2013 است. این بدین معناست که الگوریتم پیشنهادی نیاز به فضای ذخیره‌سازی کم‌تری دارد؛ ولی الگوریتم‌های KD2011 و KD2013 به فضای ذخیره‌سازی بیش‌تری احتیاج دارند. این مقایسه نشان می‌دهد که الگوریتم پیشنهادی SRTTU-2015 از لحاظ فضای ذخیره‌سازی بهتر است.

پس از به‌دست‌آوردن پرس‌وجوهای بهینه در هر خوشه، باید این پرس‌وجوها ادغام شوند تا به‌ازای هر خوشه، یک دید ذخیره شود. هدف از این مرحله این است که برای هر دامنه، یک پرس‌وجوی بهینه، ذخیره شود. در این مرحله جداول مربوط به هر پرس‌وجو در هر خوشه، Natural Outer Join می‌شوند. درنهایت برای هر دامنه یک دید به‌دست می‌آید و آن دید ذخیره می‌شود [3]. شکل (۴) الگوریتم پیشنهادی را به‌صورت روندنما نشان می‌دهد.

روندنمای شکل (۴) مراحل کار الگوریتم پیشنهادی SRTTU-2015 را نشان می‌دهد. در این الگوریتم از تابع نخست‌عمق استفاده شده است که این تابع در روندنمای شکل (۵) نشان داده شده است.

۴- نتایج شبیه‌سازی

برای پیاده‌سازی الگوریتم‌های KD2011، KD2013 و SRTTU-2015 از نرم‌افزار Microsoft Visual Studio استفاده شده است. پایگاه داده مورد استفاده این مقاله، Microsoft SQL Server است.

داده‌های این پایگاه داده، به‌صورت داده تصادفی توسط نرم‌افزار Microsoft Visual Studio تولید شده است. با توجه به این‌که داده‌ها به‌صورت تصادفی تولید شده‌اند، پس در آزمایش‌های صورت گرفته نتایج حاصل به یک نوع پایگاه داده خاص حساس نیست. سامانه مورد استفاده برای پیاده‌سازی الگوریتم‌ها دارای CPU Core i3 2.2GHz و 4GB RAM است. پایگاه داده تحلیلی مورد استفاده این آزمایش‌ها، دارای ده جدول بعد^۱ و یک جدول حقیقت^۲ می‌باشد.

برای مقایسه الگوریتم‌های KD2011، KD2013 و SRTTU-2015، آزمایش‌های متعددی صورت گرفته است. در این آزمایش‌ها دو نوع پرس‌وجو وجود دارد: پرس‌وجوهای ورودی الگوریتم و پرس‌وجوهای آزمایش. پرس‌وجوهای ورودی الگوریتم که همان پرس‌وجوهای قبلی هستند، به‌صورت داده‌های تصادفی تولید می‌شوند؛ پس از اجرای الگوریتم انتخاب دید، این الگوریتم توسط پرس‌وجوهای آزمایش مورد آزمایش قرار می‌گیرد.

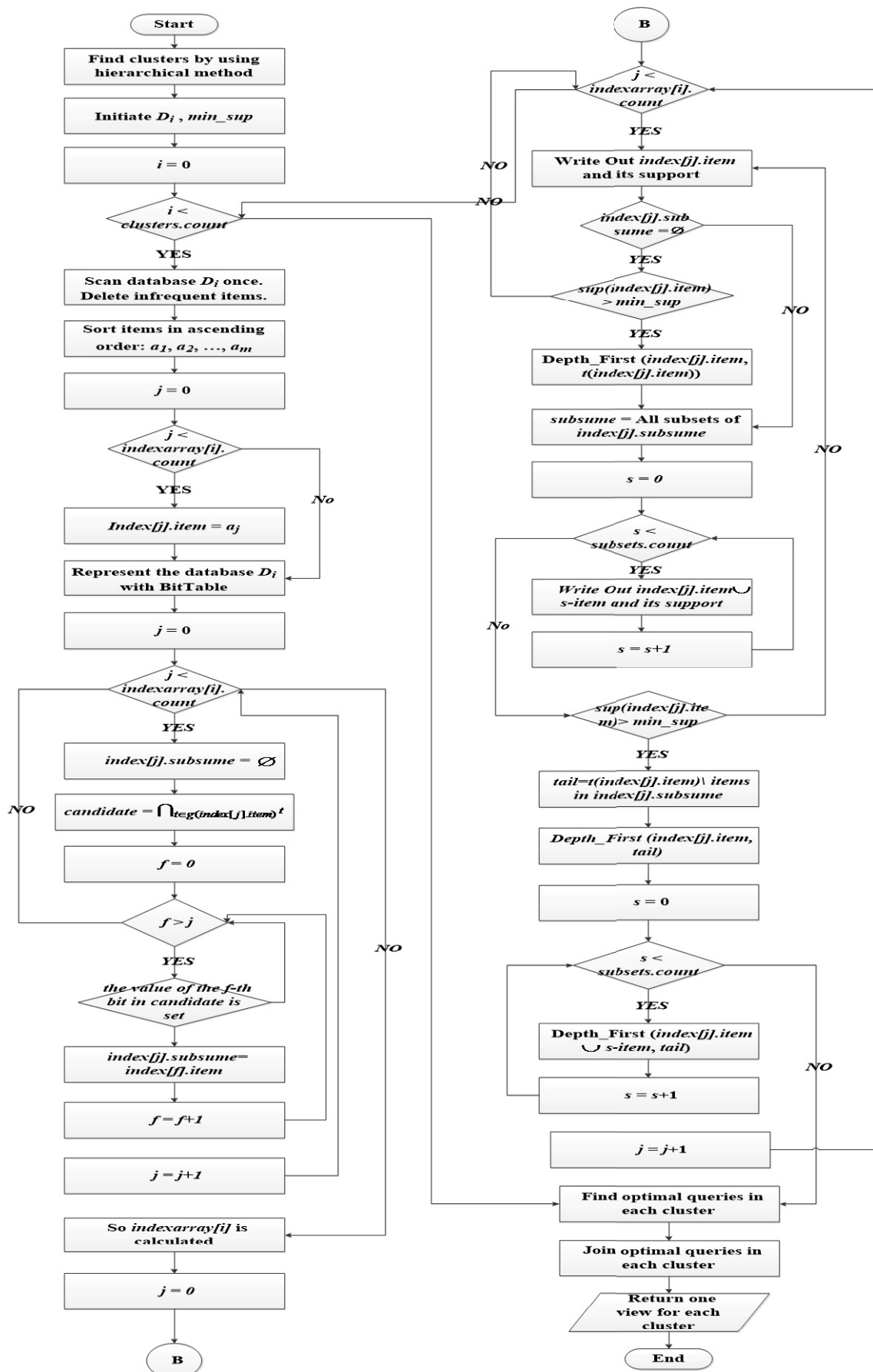
معیارهای مورد بررسی در این آزمایش‌ها به‌شرح ذیل هستند:

۱. تعداد سطرهای دیده‌های ذخیره‌شده: پس از اجرای الگوریتم انتخاب دید، دیده‌هایی برای ذخیره‌سازی انتخاب

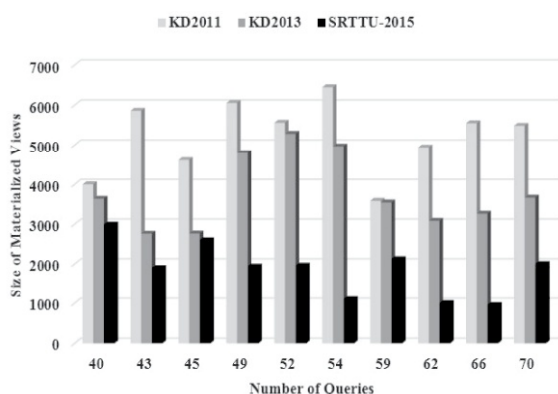
¹ Dimension

² Fact

³ Support threshold



(شکل-۴): نمودار جعبه‌ای الگوریتم SRTTU-2015
(Figure-4): Flowchart for SRTTU-2015 algorithm



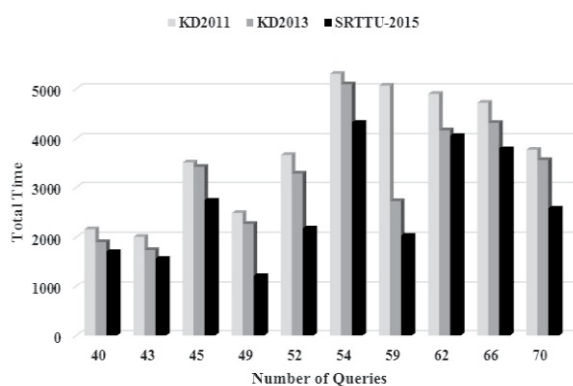
(شکل-۷): مقایسه تعداد سطرهای دیدهای ذخیره شده با افزایش

تعداد پرس وجوها

(Figure-7): Size of materialized views vs. number of queries

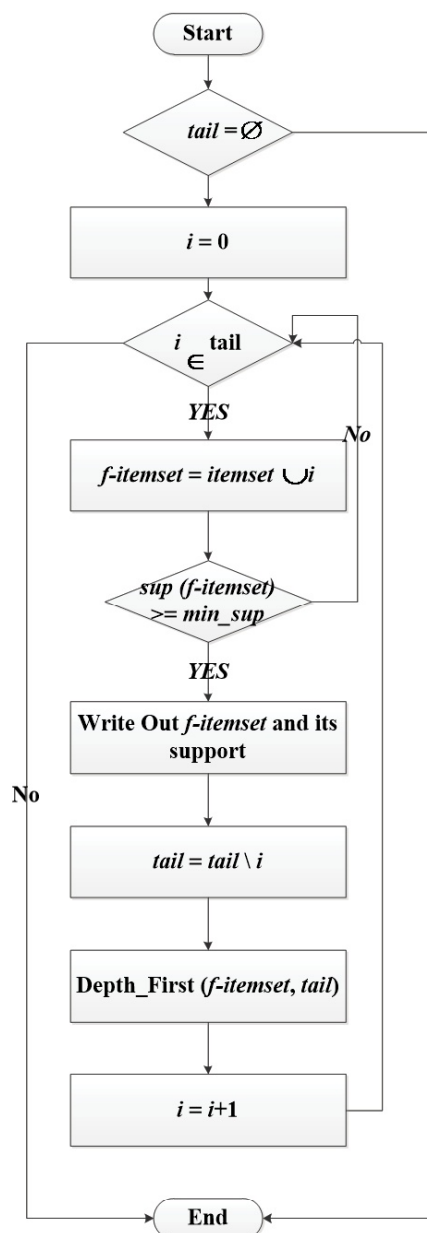
شکل (۸) مقایسه الگوریتمهای KD2011 و KD2013 و SRTTU-2015 بر اساس معیار شماره (۲) است. محور افقی نشان دهنده تعداد پرس وجوها است و محور عمودی نشان دهنده زمان کل است. این زمان شامل زمان اجرای الگوریتم و زمان صرف شده برای پاسخ به پرس وجوهای آزمایش است.

نمودار رسم شده در شکل (۸) نشان می دهد که زمان کل الگوریتم پیشنهادی SRTTU-2015 همیشه کمتر از زمان کل الگوریتمهای KD2011 و KD2013 است. با توجه به اهمیت زمان، هر الگوریتم انتخاب دیدی که در زمان کمتری به پرس وجوهای آزمایش پاسخ دهد، الگوریتم انتخاب دید مناسبتری خواهد بود. با توجه به نمودار رسم شده در شکل (۸)، مشاهده می شود که الگوریتم پیشنهادی SRTTU-2015 دارای زمان پاسخ کمتری بوده است.



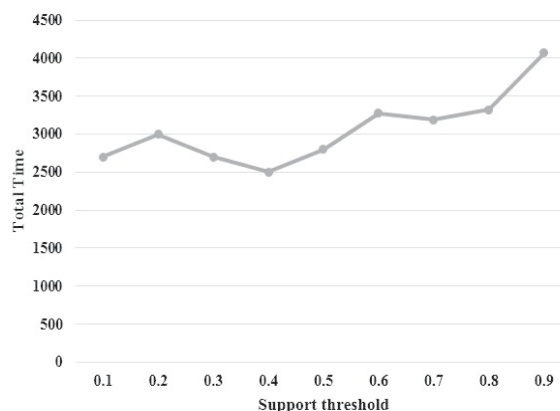
(شکل-۸): مقایسه زمان کل با افزایش تعداد پرس وجوها

(Figure-8): Total time vs. number of queries



(شکل-۵): نمودار جعبه ای تابع اول عمق

(Figure-5): Flowchart for first depth function



(شکل-۶): زمان کل در برابر تغییرات حد آستانه پوشش

(Figure-6): Total time vs. changes of support threshold

الگوریتم پیشنهادی SRTTU-2015 بهبود یافته الگوریتم‌های KD2011 و KD2013 است. در الگوریتم SRTTU-2015، روش یافتن پرس‌وجوهای پرتکرار بهبود یافته است. با توجه به نمودارهای رسم‌شده در بخش قبل مشاهده می‌شود که الگوریتم SRTTU-2015 نسبت به الگوریتم KD2011 از نظر معیارهای زمانی (معیارهای شماره (۲) و (۳))، به‌طور متوسط ۲۵ درصد بهبود و از نظر معیار شماره (۱)، به‌طور متوسط ۵۰ درصد بهبود داشته است. همچنین الگوریتم پیشنهادی نسبت به الگوریتم KD2013، از نظر معیارهای زمانی (معیارهای شماره (۲) و (۳))، ۲۲ درصد بهبود و از نظر معیار شماره (۱)، به‌طور متوسط ۴۸ درصد بهبود داشته است. علت این بهبود، استفاده از الگوریتم Index-BitableFI به جای الگوریتم‌های Apriori و DIC است. الگوریتم Index-BitableFI آیت‌های پرتکرار را به‌صورت مناسب‌تری نسبت به الگوریتم‌های Apriori و DIC پیدا می‌کند. الگوریتم Apriori با پیچیدگی زمانی $O(n^2)$ آیت‌های پرتکرار را و الگوریتم Index-BitableFI با پیچیدگی $O(n \log n)$ آیت‌های پرتکرار را می‌یابد. با توجه به اینکه الگوریتم پیشنهادی از Index-BitableFI استفاده شده است، بنابراین الگوریتم پیشنهادی SRTTU-2015 با پیچیدگی کم‌تری، دیدهای مناسب را انتخاب می‌کند.

سپاس‌گزاری

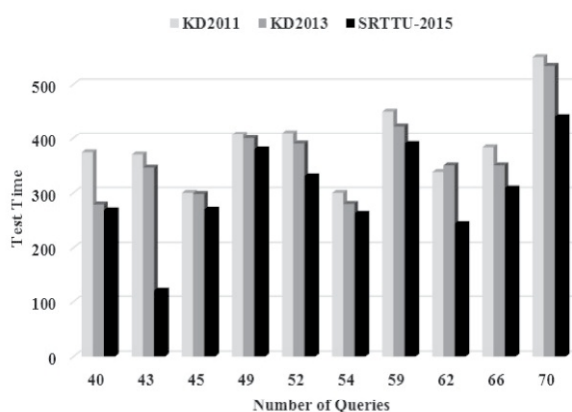
این پژوهش با حمایت مالی دانشگاه تربیت دبیر شهید رجایی طبق قرارداد شماره ۲۶۲۴۱ مورخ ۱۳۹۵/۱۰/۷ انجام شده است.

6-References

۶-مراجع

- [1] J. Widom, "Research Problems in Data Warehousing," in *International Conference on Information and Knowledge*, Baltimore, Maryland, 1995.
- [2] T. V. Kumar and K. Devi, "Frequent Queries Identification for Constructing Materialized Views," in *Electronics Computer Technology (ICECT)*, Kanyakumari, 2011.
- [3] T. V. V. Kumar, G. Dubey and A. Singh, "Frequent Queries Selection for View Materialization," *Advances in Computing and Information Technology*, vol. 177, pp. 521-530, 2013.
- [4] W. Song, B. Yang and Z. Xu, "Index-BitTableFI: An improved algorithm for mining

شکل (۹) نشان‌دهنده زمان لازم برای پاسخ به پرس‌وجوهای آزمایش در برابر تعداد پرس‌وجوها است. در این نمودار محور افقی نشان‌دهنده تعداد پرس‌وجوها است و محور عمودی نشان‌دهنده زمان صرف‌شده برای پاسخ به پرس‌وجوهای آزمایش است.



(شکل-۹): مقایسه زمان تست با افزایش تعداد

پرس‌وجوها

(Figure-9): Test time vs. number of queris

با توجه به نمودار رسم‌شده در شکل (۹) مشاهده می‌شود که زمان لازم برای پاسخ‌دادن به پرس‌وجوهای آزمایش، در الگوریتم پیشنهادی SRTTU-2015 نسبت به الگوریتم‌های KD2011 و KD2013، کم‌تر است. الگوریتمی که در زمان کم‌تری به پرس‌وجوهای تست پاسخ دهد الگوریتم مناسب‌تری است.

با توجه به نمودارهای رسم‌شده مشاهده می‌شود که الگوریتم پیشنهادی SRTTU-2015 نسبت به الگوریتم‌های KD2011 و KD2013 بهبود داشته است.

۵-نتیجه‌گیری

در این مقاله، الگوریتمی برای انتخاب دید ارائه شده است (SRTTU-2015). این الگوریتم بر اساس پرس‌وجوهای قبلی که بر روی پایگاه داده تحلیلی اجرا شده‌اند، کار می‌کند. در ابتدا بر اساس الگوریتم‌های خوشه‌بندی، پرس‌وجوهای قبلی خوشه‌بندی می‌شوند؛ سپس در هر خوشه، پرس‌وجوهای پرتکرار به‌دست می‌آیند. در مرحله بعد، پرس‌وجوهای بهینه به‌دست می‌آیند و درنهایت، پرس‌وجوهای بهینه در هر خوشه ادغام می‌شوند تا به‌ازای هر خوشه، یک دید به‌دست آید.

Extending database technology: Advances in database technology, France, 2008.

- [17] T. V. Kumar and M. Haider, "Query answering-based view selection," *International Journal of Business Information Systems*, vol. 18, no. 3, pp. 338-353, 2015.
- [18] V. T. Kumar and M. Haider, "Selection of views for materialization using size and query frequency," *Information Technology and Mobile Communication*, pp. 150-155, 2011.
- [19] V. T. Kumar and M. Haider, "Materialized views selection for answering queries," *Data Engineering and Management*, pp. 44-51, 2012.
- [20] M. S. Chaudhari and D. Chandrashekhar, "Dynamic materialized view selection algorithm: a clustering approach," *Data Engineering and Management*, pp. 57-66, 2012.
- [21] V. T. Kumar and B. Arun, "Materialized View Selection Using HBMO," *International Journal of System Assurance Engineering and Management*, pp. 1-14, 2015.
- [22] P. Vishwanath and R. Sridhar, "An Association Rule Mining for Materialized View Selection and View Maintenance," *International Journal of Computer Applications*, vol. 105, no. 5, 2015.
- [23] D. Yao, A. abulizi and R. Hou, "An improved algorithm of materialized view selection within the confinement of space," 2015.
- [24] T. V. Kumar and K. Devi, "Materialised view construction in data warehouse for decision making," *International Journal of Business Information Systems*, vol. 11, no. 4, pp. 379-396, 2012.
- [25] T. V. V. Kumar, A. Singh and G. Dubey, "Mining Queries for Constructing Materialized Views in a Data Warehouse," *Advances in Computer Science, Engineering & Applications*, pp. 149-159, 2012.
- [26] T. V. V. Kumar, A. Goel and N. Jain, "Mining information for constructing materialised views," *Int. J. Information and Communication Technology*, vol. 2, no. 4, pp. 386-405, 2010.
- frequent itemsets," *Knowledge-Based Systems*, vol. 21, pp. 507-513, 2008.
- [5] J. Yang, K. Karlapalem and Q. Li, "Algorithms for materialized view design in data warehousing environment," *VLDB*, vol. 97, 1997.
- [6] I. Mami and Z. Bellahsene, "A survey of view selection methods," *ACM SIGMOD*, vol. 41, no. 1, pp. 20-29, 2012.
- [7] C. A. Dhote and M. S. Ali, "Materialized view selection in data warehousing: a survey," *Journal of Applied sciences*, vol. 9, no. 3, pp. 401-414, 2009.
- [8] J.-S. Sohn, J.-H. Yang and I.-J. Chung, "Improved view selection algorithm in data warehouse," *IT Convergence and Security*, pp. 921-928, 2013.
- [9] A. B. Rashid, M. Islam and A. L. Hoque, "Dynamic Materialized View Selection Approach for Improving Query Performance," *Computer Networks and Information Technologies*, vol. 142, pp. 202-211, 2011.
- [10] W. Xu, D. Theodoratos, C. Zuzarte, X. Wu and V. Oria, "A dynamic view materialization scheme for sequences of query and update statements," *Data Warehousing and Knowledge Discovery*, pp. 55-56, 2007.
- [11] N. Daneshpour and A. Abdollahzadeh Barfouroush, "Dynamic view Management System for Query Prediction to view materialization," *International Journal of Data Warehousing and Mining*, vol. 7, no. 2, pp. 67-96, 2011.
- [12] I. Mami, R. Coletta and Z. Bellahsene, "Modeling view selection as a constraint satisfaction problem," in *International Conference on Database and Expert Systems Applications*, France, 2011.
- [13] I. Mami, Z. Bellahsene and R. Coletta, "View selection under multiple resource constraints in a distributed context," in *International Conference on Database and Expert Systems Applications*, Vienne, 2012.
- [14] I. Mami, Z. Bellahsene and R. Coletta, "A Declarative Approach to View Selection Modeling," *Transactions on Large-Scale Data- and Knowledge-Centered Systems*, pp. 115-145, 2013.
- [15] R. Huang, R. Chirkova and Y. Fathi, "Advances in Databases and Information Systems," in *Deterministic view selection for data analysis queries: Properties and algorithms*, Berlin, Springer Berlin Heidelberg, 2012, pp. 195-208.
- [16] Z. Asgharzadeh, R. Chirkova and Y. Fathi, "Exact and inexact methods for selecting views and indexes for olap performance improvement," in *international conference on*



ریحانه صباغ گل مدرک کارشناسی خود را در رشته مهندسی کامپیوتر- نرم افزار در سال ۱۳۹۲ اخذ کرد. ایشان در سال ۱۳۹۴ مدرک کارشناسی ارشد خود را در دانشگاه تربیت دبیر شهید رجایی اخذ کرده است. موضوع پایان نامه ایشان، بهبود سرعت پاسخ گویی به پرس وجود در پایگاه داده تجلی با استفاده از گروه بندی پرس وجوها بوده است. نشانی رایانامه ایشان عبارت است از:

Sabbgh.rsg@gmail.com



نگین دانشپور استادیار دانشکده

مهندسی کامپیوتر دانشگاه تربیت دبیر

شهید رجایی است. نامبرده تحصیلات

خود را در مقطع کارشناسی مهندسی

کامپیوتر-سخت‌افزار در سال ۱۳۷۸ در

دانشگاه شهید بهشتی و کارشناسی ارشد مهندسی کامپیوتر-

نرم‌افزار در سال ۱۳۸۱ در دانشگاه صنعتی امیرکبیر به پایان

رسانده است، و در سال ۱۳۸۹ دکترای خود در رشته مهندسی

کامپیوتر-نرم‌افزار را از دانشگاه صنعتی امیرکبیر اخذ کرده

است. زمینه‌های پژوهشی مورد علاقه ایشان عبارتند از: پایگاه

داده، پایگاه داده تحلیلی، سامانه‌های تصمیم‌یار، و داده‌کاوی.

نشانی رایانامه ایشان عبارت است از:

ndaneshpour@srttu.edu