بهبود الگوریتم انتخاب دید در پایگاه دادهٔ تحلیلی با استفاده از یافتن پرس‌وجوهای پرتکرار

ریحانه صباغ گل و نگین دانشپور*

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

چکیده

پایگاه دادهٔ تحلیلی منبعی برای ذخیره‌سازی داده‌های تاریخی جهت تحلیل است. به‌طورمعمول زمان پاسخ به پرس‌وجوهای تحلیلی، زمانی طولانی است. استفاده از دید به جای دسترسی مستقیم به پایگاه داده، سرعت پاسخ‌گویی را بهبود می‌دهد. راه‌کارهای مختلفی برای ذخیره‌سازی دید وجود دارد؛ که مناسب‌ترین راهکار برای ذخیره‌سازی دید، ذخیره‌سازی دیدهای پراستفاده و پرکاربرد است. پرس‌وجوهایی که درقبل مورد استفادهٔ پایگاه دادهٔ تحلیلی بوده‌اند، حاوی اطلاعات مهمی هستند که به‌احتمال زیاد در آینده نیز مورد استفاده خواهند بود. این مقاله، الگوریتمی برای ذخیره‌سازی دیدهای پرکاربرد ارائه می‌دهد. این الگوریتم با استفاده از پرس‌وجوهای قبلی، دیدهای پرکاربرد را یافته و آن‌ها را ذخیره می‌کند. این دیدها توانایی پاسخ‌گویی را به بسیاری از پرس‌وجوهایی 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 روش‌های قبلی و کاهش زمان پاسخ به پرس‌وجوها شده است. آزمایش‌های انجام‌شده نشان می‌دهند که الگوریتم پیشنهادی از لحاظ زمانی نسبت به الگوریتم‌های قبلی ۲۳ درصد و از لحاظ فضای ذخیره‌سازی پنجاه درصد بهبود داشته است.

واژگان کلیدی: پایگاه دادهٔ تحلیلی، پرس‌وجوهای پرتکرار، خوشه‌بندی، ذخیره‌سازی 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 queriesbe ing posed in the future. This paper uses

* نویسندهٔ عهده‌دار مکاتبات

[DOI: 10.18869/acadpub.jsdp.14.1.29]

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

۱-مقدمه

امروزه سازمان‌ها و ادارات بزرگ دارای حجم بسیار بالایی از داده‌ها هستند. این داده‌ها در پایگاه داده‌های مختلفی ذخیره شده‌اند. برای دسترسی به این داده‌ها دو روش وجود دارد: روش تنبل[1] و روش مشتاق[2]. در روش نخست داده‌ها به‌وسیلهٔ پرس‌وجوی کاربر یک‌پارچه[3]می‌شوند. این روش، به‌علت این‌که داده‌ها پس از اجرای پرس‌وجوی کاربر یکپارچه می‌شوند، دارای تأخیر زیادی است و درنتیجه، روش مناسبی نیست. روش دوم از پایگاه دادهٔ تحلیلی استفاده می‌کند. در این روش اطلاعات مورد نیاز از پایگاه‌های داده مختلف استخراج[4] شده و پس از پاک‌سازی داده‌ها، در منبع بزرگی به نام پایگاه دادهٔ تحلیلی بارگذاری[5] می‌شوند[1] . در این روش برای تصمیم‌گیری[6] از پرس‌وجوهای تحلیلی استفاده می‌شود. امروزه بیش‌تر سازمان‌ها و ادارات از این روش برای پاسخ به پرس‌وجوهای تحلیلی تحلیل‌گران استفاده می‌کنند [2].

پایگاه دادهٔ تحلیلی به‌منظور کاهش زمان پاسخ به پرس‌وجوهای تحلیلی، از دید ذخیره‌شده[7] استفاده می‌کند. با توجه به محدودیت فضای ذخیره‌سازی، و نیز زمان زیاد به‌روزنگه‌داشتن دیدها، نمی‌توان تمام دیدهای ممکن را ذخیره‌کرد؛ بنابراین باید از بین تمامی دیدهای ممکن، تنها مجموعه‌ای از دیدها را ذخیره کرد. دیدها می‌توانند بر اساس پرس‌وجوهایی که درقبل مورد استفادهٔ پایگاه دادهٔ تحلیلی بوده‌اند، انتخاب و ذخیره شوند[3]،[2]. علت استفاده از پرس‌وجوهای قبلی، این است که به‌احتمال زیاد در آینده دوباره اتفاق خواهند افتاد. روند کار الگوریتم‌های بیان‌شده در [3]،[2] به این صورت است که ابتدا پرس‌وجوهای قبلی خوشه‌بندی می‌شوند؛ سپس در هر خوشه پرس‌وجوهای پرتکرار به‌دست می‌آیند. در مرحلهٔ بعد پرس‌وجوهای بهینه انتخاب می‌شوند و درنهایت پرس‌وجوهای بهینه در هر خوشه پیوند[8] می‌شوند. این الگوریتم به‌علت این‌که برای یافتن پرس‌وجوهای پرتکرار در هر خوشه از الگوریتم‌های مقدماتی همچون Apriori و DIC استفاده کرده است، دارای زمان اجرای بالایی است و سرعت پاسخ به پرس‌و جوهای تحلیلی بسیار کم است. از طرفی دیگر با توجه به این‌که مهم‌ترین عامل در الگوریتم‌های انتخاب دید، زمان است، بنابراین در این مقاله، الگوریتمی پیشنهاد شده است که در زمان کم‌تری، به پرس‌وجوهای تحلیلی، پاسخ می‌دهد.

الگوریتم انتخاب دید پیشنهادشده در این مقاله، با استفاده از الگوریتم Index-BittableFI [4] پرس و جوهای پرتکرار را در هر خوشه به‌دست می‌آورد. الگوریتم ارائه‌شده در [3]،[2] و الگوریتم پیشنهادی این مقاله، پیاده‌سازی شده‌اند و مشاهده شده است که الگوریتم پیشنهادی نسبت به الگوریتم‌های بیان‌شده در منبع [3]،[2] بهبود داشته است.

ساختار این مقاله به شرح زیر است: در بخش ۲ مقالات و الگوریتم‌های مرتبط با این مقاله ارائه می‌شود. در بخش ۳ الگوریتم انتخاب دیدی که این مقاله ارائه می‌دهد بیان شده است. بخش ۴، به شرح پیاده‌سازی الگوریتم می‌پردازد و بخش ۵ به نتیجه‌گیری می‌پردازد.

۲-پیشینهٔ پژوهش

از جملهٔ نخستین الگوریتم‌های مطرح‌شده برای انتخاب دید می‌توان به الگوریتم حریصانه که در سال ۹۷ مطرح شده است، اشاره کرد [5]. الگوریتم حریصانه به این صورت عمل می‌کند که با استفاده از تابع سودی که تعریف شده است، دیدهای با سود بیش‌تر را انتخاب و با توجه به محدودیت‌های موجود از جمله فضای نگه‌داری آن دیدها را نگه‌داری می‌کند. البته این الگوریتم‌ها برای مسائل با ابعاد کم مناسب هستند؛ ولی برای مسائل با ابعاد بزرگ مناسب نیستند.

برخی دیگر از روش‌ها بر اساس گرافی که رسم شده است تصمیم‌گیری می‌کنند، برخی از این گراف‌ها عبارتند از گراف AND-OR یا گراف MVPP [9]. بعد از رسم این گراف‌ها بر اساس تابع هزینه‌ای که تعریف شده است، تصمیم‌گیری می‌شود که کدام دیدها انتخاب و ذخیره شوند[8]-[6]. این

[1] lazy
[2] eager
[3] integrate
[4] extraction
[5] load
[6] Decision making
[7] Materialized View
[8] Join
[9] Multi-View Processing Plan

[DOI: 10.18869/acadpub.jsdp.14.1.29]

الگوریتم‌ها با توجه به این‌که 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[1] 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[2] 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. پیش‌بینی‌کردن 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 جملهٔ این الگوریتم‌ها عبارتند از روش‌های انتخاب دید با استفاده از مسئلهٔ ارضای محدودیت‌ها[3][14]-[12].

برخی دیگر از این الگوریتم‌ها، با استفاده از روش برنامه‌نویسی صحیح[4] مسئله را مدل کرده و با استفاده از ابزارهای حل مسئلهٔ ریاضی، مسئلهٔ انتخاب دید را که به‌صورت فرمول درآمده است، حل‌کرده 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 این‌که محدودیت‌های مورد نظر را برای هر پرس‌وجو در نظر می‌گیرند، برای مسائل با ابعاد بزرگ پیچیده است و دارای زمان اجرای بالایی خواهند بود.

[1] Aggregate
[2] DAG
[3] Constraint Satisfaction Problem
[4] integer programming

برخی از الگوریتم‌ها، بهبودیافتهٔ الگوریتم حریصانه هستند. این الگوریتم‌ها علاوه‌بر معیارهای موجود در الگوریتم حریصانه، بر اساس اندازهٔ پرس‌وجو، فرکانس تکرار پرس‌وجوها و میزان توانایی پرس‌وجو در تصمیم‌گیری[5] دیدهای مناسب را انتخاب کرده و ذخیره‌سازی می‌کنند[17] . این الگوریتم‌ها با توجه به این که از شبکه‌ای از کعب[6] ها استفاده می‌کنند، برای مسائل با ابعاد کوچک مناسب هستند.

الگوریتم [18] با لحاظ‌کردن مقادیر تکرار هر پرس‌وجو و استفاده از فرمول سود جدید توانسته است بهبودی نسبت به الگوریتم حریصانه حاصل کند. این الگوریتم در ابتدا شبکهٔ پایگاه دادهٔ تحلیلی 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 دادهٔ تحلیلی 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[7] 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 دادهٔ تحلیلی 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 مرحلهٔ بعد، پس از اجرای چندین زیر برنامهٔ 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

[DOI: 10.18869/acadpub.jsdp.14.1.29]

الگوریتم معرفی‌شده در [23] تعدادی دید نماینده[1] انتخاب می‌کند؛ سپس عملیات حذف و اضافه‌کردن دیدها انجام می‌شود؛ به این صورت که دیدهایی که هزینهٔ بالایی دارند، حذف شده و دیدهایی که هزینهٔ کمتری دارند اضافه می‌شوند. درصورتی‌که این الگوریتم دیدهای 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، دیدهای مناسب انتخاب می‌شوند [26]-[24]،[3]،[2]. در این الگوریتم‌ها از پرس‌وجوهای قبلی استفاده می‌شود؛ علت این امر، این است که به‌احتمال زیاد در آینده نیز، این پرس‌وجوها اتفاق خواهند افتاد. این الگوریتم‌ها از روش MVCF[2] 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[3] 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.

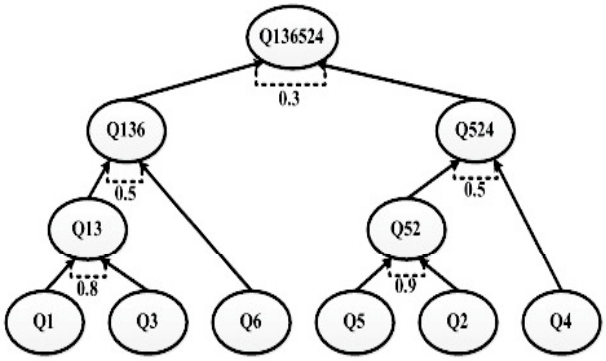
۳-الگوریتم انتخاب دید پیشنهادی

در این بخش الگوریتم ارائه‌شده 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 پرس‌وجوهایی که درقبل مورد استفادهٔ پایگاه دادهٔ تحلیلی بوده‌اند، استفاده شده است؛ زیرا این پرس‌وجوها حاوی اطلاعات مفیدی هستند که به‌احتمال زیاد در آینده نیز مورد استفاده خواهند بود. در الگوریتمی که در ادامه توضیح داده می‌شود، روش یافتن پرس‌وجوهای پرتکرار در روش MVCF بیان‌شده در منبع [3]،[2] بهبود داده شده که باعث بهبود نهایی الگوریتم انتخاب دید با استفاده از روش MVCF است. الگوریتم بیان‌شده در منابع [3]،[2] 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[4]، دسته‌بندی می‌شوند. به هریک از این دسته‌ها، خوشه[5] یا دامنه[6] گفته می‌شود. برای خوشه‌بندی‌کردن پرس‌وجوها از معیار شباهت جاکارد[7] [2] استفاده می‌شود.

برای یافتن خوشه‌هایی که دارای پرس‌وجوهای پرتکرار هستند، ابتدا باید الگوریتم ادغام پرس‌وجوها[8] [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 به‌صورت 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 مرحلهٔ همهٔ پرس‌وجوها در یک خوشه قرار می‌گیرند و درنهایت درخت ساخته می‌شود. نمونه‌ای از این درخت در شکل (۱) آمده است.



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

[1] Candidate
[2] Materialized Views Construction Framework
[3] Dynamic Itemset Counting
[4] Hierarchical clustering
[5] cluster
[6] domain, area, subject area
[7] Jaccard's coefficient
[8] Query Merger

پس از ساخت درخت، گره‌های درخت از بالا به پایین ملاقات[1] می‌شوند تا زمانی که خوشه‌ای پیدا شود که درجهٔ شباهتی بیش‌تر از حداقل شباهت حد آستانه[2]داشته باشد. این خوشه‌ها نشان‌دهندهٔ دامنه‌ها خواهند بود. به‌عنوان مثال در شکل (۱)، اگر حد آستانه ۰/۴ باشد خوشه‌ها Q136 و Q524 خواهند بود. توجه شود که Q13 به این معنی است که این خوشه شامل Q1 و Q3 است.

پس از یافتن خوشه‌ها، باید پرس‌وجوهای پرتکرار در هر خوشه یافته شوند. روش یافتن پرس‌وجوهای پرتکرار برای هر خوشه، در الگوریتم‌های KD2011 و KD2013، به‌ترتیب روش Apriori و DIC است؛ الگوریتم‌های Apriori و DIC الگوریتم‌هایی قدیمی هستند و سبب می‌شوند که در زمان اجرای بالایی به پرس‌وجوهای تحلیلی پاسخ داده شود. از طرفی دیگر مهم‌ترین عامل در الگوریتم‌های انتخاب دید، زمان اجرا است؛ بنابراین در الگوریتم پیشنهادی این مقاله، از الگوریتم Index-BittableFI [4] برای بهبود زمان اجرا استفاده شده است. این الگوریتم برای پرس‌وجوهای هر خوشه بررسی می‌شود تا پرس‌وجوهای پرتکرار در هر خوشه به‌دست آیند. در این مرحله، هر کدام از پرس‌وجوها را یک مجموعهٔ آیتم[3] می‌نامند. یک مجموعهٔ آیتم که دارای k تا آیتم است، k-*itemset* نامیده می‌شود. با توجه به این‌که هر پرس و جو در قسمت from خود دارای تعدادی جدول است، بنابراین هر کدام از این جداول، یک آیتم نامیده می‌شود. *sup(X)* نشان‌دهندهٔ تعداد پرس‌وجوهایی است که شامل آیتم X هستند. اگر $sup(X)>min\_sup$ باشد، آنگاه X یک آیتم پرتکرار نامیده می‌شود.

در الگوریتم Index-BittableFI، برای کاهش فضای جستجو، آرایهٔ اندیسی[4] تعریف می‌شود. فرمول (۱) تعریف ریاضی آرایهٔ اندیسی است.

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

که در آن X، آیتمی است که آرایهٔ اندیسی آن یافته می‌شود و D، پایگاه داده‌ای است که شامل تمامی پرس‌وجوهای هر خوشه است و t، مجموعه‌ای از آیتم‌هایی است که هر پرس‌وجو شامل می‌شود. این آیتم‌ها فیلدهای مورد استفاده برای هر پرس‌وجو هستند.

به عبارت دیگر آرایهٔ اندیسی، یک آرایه با اندازهٔ m_1 است که m_1، تعداد *1-itemset*های پرتکرار است. هر عضو این آرایه به‌صورت زوج دوتایی (*item* , *subsume*) است، که *item*، *1-itemset* است و *subsume* در فرمول (۲) توصیف شده است.

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

که *subsume(item)* به این معنی است که اگر $j\in subsume(item)$ باشد، آن‌گاه بر اساس ترتیب $\prec$، j بعد از *item* قرار دارد. برای پیداکردن آیتم‌های پرتکرار، ابتدا باید در هر خوشه، آرایهٔ اندیسی برای هر آیتم به‌دست آید؛ سپس با دردست‌داشتن آرایهٔ اندیسی تمام آیتم‌ها، با استفاده از الگوریتم Index-BittableFI آیتم‌های پرتکرار در هر خوشه به دست آورده شوند. در قسمت (الف) الگوریتم ساختن آرایهٔ اندیسی توضیح داده شده است و در قسمت (ب) الگوریتم Index-BittableFI توضیح داده شده است. اساس الگوریتم‌های بیان‌شده در قسمت‌های الف و ب بر پایه‌ی منبع [4] است و در این مقاله برای این مسأله بازنویسی شده است.

الف) الگوریتم ساختن آرایهٔ اندیسی:

ابتدا الگوریتم ساختن آرایهٔ اندیسی مورد بررسی قرارمی‌گیرد. این الگوریتم در شکل (۲) بیان شده است.

```
Procedure IndexArray
Input: The set of queries in each cluster Di, min_sup.
Output: index array for each cluster index array[i]
Begin
for each cluster i do
  Scan database Di once. Delete infrequent items;
  Sort frequent single items in supports ascending order as
  a1, a2, …, am;
  for each element index[j] of index array[i] of cluster i do
    index[j].item=aj;
  Represent the database Di with BitTable;
  for each element index[j] in index array[i] of cluster i do
    index[j].subsume=∅;
    candidate = ⋂ t∈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 ← index[f].item;
      end if
    end for
  end for
Write Out index array[i];
end for
End
```

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

[1] visit
[2] Minimum threshold
[3] itemset
[4] Index array

[DOI: 10.18869/acadpub.jsdp.14.1.29]

در ابتدا پایگاه دادهٔ مربوط به هر خوشه یعنی D_i پویش می‌شود تا آیتم‌های پرتکرار به‌دست آیند؛ سپس آیتم‌های پر تکرار به‌صورت صعودی مرتب می‌شوند و آیتم‌ها به‌ترتیب به هر عضو از آرایه اندیسی خوشهٔ i نسبت داده می‌شوند. در مرحلهٔ بعد جدول بیتی[1] از پایگاه دادهٔ D_i ساخته می‌شود. نحوهٔ ساخت این جدول به این صورت است که اگر پرس‌وجوی T شامل آیتم f باشد، آن‌گاه بیت fام مرتبط با پرس‌وجوی T، یک خواهد شد؛ در غیر این صورت، صفر خواهد بود؛ سپس در آخرین حلقه، آرایهٔ اندیسی تولید می‌شود. ابتدا تمام پرس‌وجوهایی که شامل $index[j].item$ می‌شوند، اشتراک گرفته می‌شوند؛ سپس تمام آیتم‌هایی که دارای اندیس بالاتری نسبت به j هستند، در $index[j]$.subsume قرارمی‌گیرد؛ درنهایت به‌عنوان خروجی برای هر خوشه، یک آرایهٔ اندیسی برگردانده می‌شود.

ب) الگوریتم Index-BitTableFI:

این الگوریتم با استفاده از آرایهٔ 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 مرحلهٔ بعد *subsume* مربوط به هرآیتم بررسی می‌شود. در صورت تهی‌بودن *subsume* آن، اگر *support* آن آیتم از *min_sup* بیش‌تر باشد، آنگاه تابع نخست‌عمق[2] 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 هرکدام از مجموعه‌های به‌دست‌آمده 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 به‌دست‌آوردن پرس‌وجوهای پرتکرار در هر خوشه، باید پرس‌وجوهای بهینه[3] 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 ظرفیت کوله‌پشتی، پرس‌وجوها را در آن می‌توان قرارداد [26]. پس از حل معادلات مسألهٔ 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 ==∅) 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-item⊆ index[j].subsume do
        Write Out index[j].item∪s-item and its support;
        // The support of any index[j].item∪s-item equals to
        //support of index[j].item
      end for
      if (sup(index[j].item)>min_sup) then
        tail ← t(index[j].item)\ 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-item⊆ index[j].subsume do
          Depth_First (index[j].item∪s-item, tail);
        end for
      end if
    end else
    end if
  end for
end for
Procedure Depth_First (itemset, tail)
if (tail==∅) then return;
for each i∈ tail do
  f-itemset ← itemset∪i;
  if (sup(f-itemset) ≥ min_sup) then
    Write Out f-itemset and its support;
    tail ← tail\i;
    Depth_First (f-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 رابطهٔ (۳) بیان شده است.

$$\text{Maximise } 13Q_1+15Q_7+28Q_{16}+35Q_{17}+32Q_{20} \qquad (۳)$$

subject to

$$120Q_1+250Q_7+430Q_{16}+310Q_{17}+360Q_{20}\le 1000$$

and

$$Q_i s=0 \text{ or } 1 \text{ where } i=1,7,16,17,20$$

[1] BitTable
[2] depth-first
[3] Optimal Queries

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 دادهٔ مورداستفادهٔ این مقاله، Microsoft SQL Server است.

داده‌های این پایگاه داده، به‌صورت دادهٔ 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 دادهٔ خاص حساس نیست. سامانهٔ مورد استفاده برای پیاده‌سازی الگوریتم‌ها دارای 4GB RAM و CPU Corei3 2.2GHz است. پایگاه دادهٔ تحلیلی مورد استفادهٔ این آزمایش‌ها، دارای ده جدول بعد[1] و یک جدول حقیقت[2] می‌باشد.

برای مقایسهٔ 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 دیدهای ذخیره‌شده: پس از اجرای الگوریتم انتخاب دید، دیدهایی برای ذخیره‌سازی انتخاب می‌شوند، این معیار تعداد سطرهای دیدهای 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[3] 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 آستانهٔ پوشش را نشان می‌دهد. در این نمودار محور افقی، مقدار حد آستانه پوشش و محور عمودی مقدار زمان کل است.

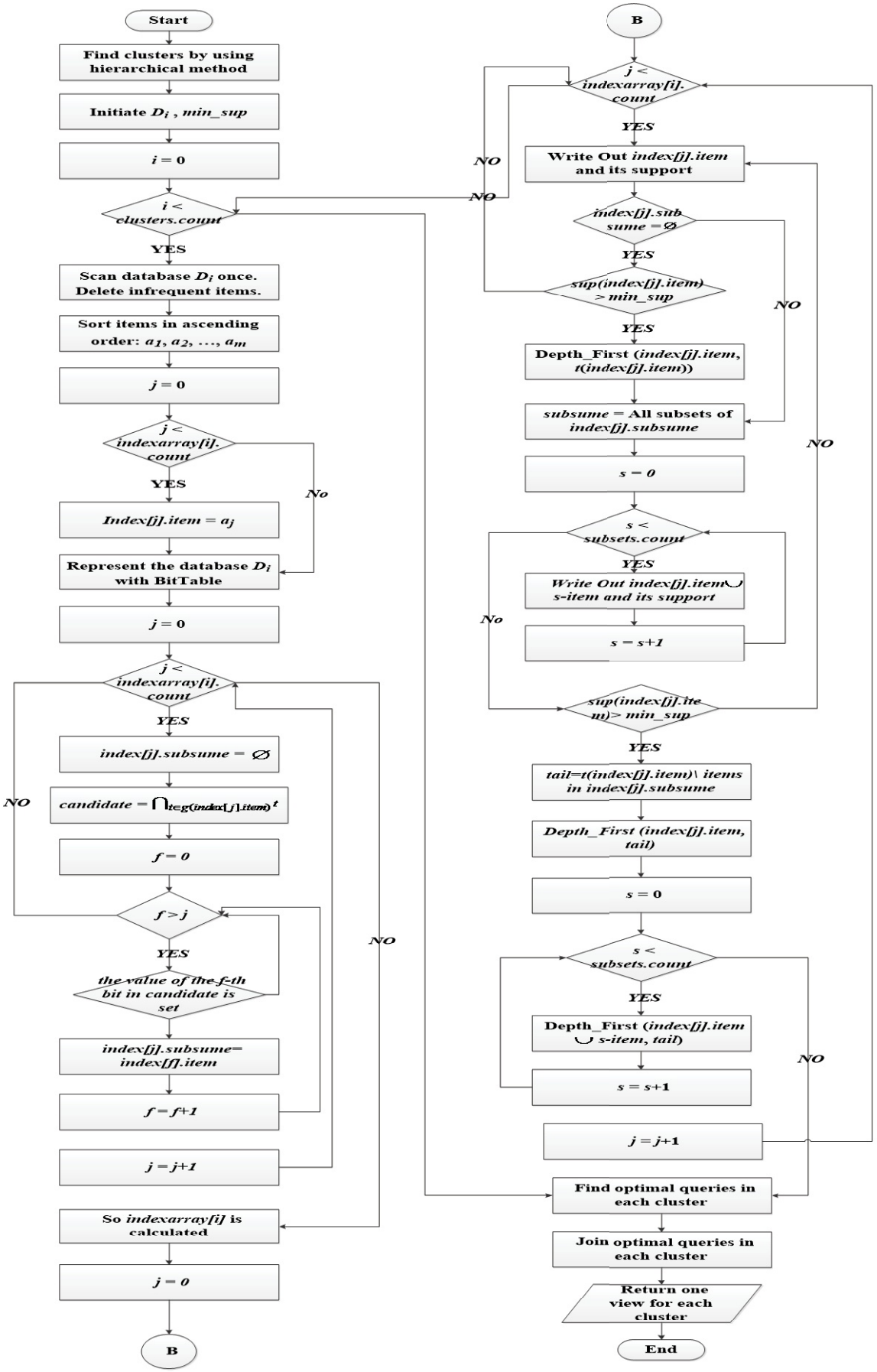
همان‌طورکه در شکل (۶) مشاهده می‌شود بهترین مقدار برای حد آستانه پوشش، ۰/۴ است. بنابراین همان‌طورکه در بخش۳ توضیح داده شد، با استفاده از این پارامتر، خوشه‌ها مشخص می‌شوند.

شکل (۷) نشان‌دهندهٔ مقایسهٔ الگوریتم‌های KD2011 و KD2013 و SRTTU-2015 بر اساس معیار شمارهٔ (۱) است. در این نمودار محور افقی نشان‌دهندهٔ تعداد پرس‌وجوها و محور عمودی نشان‌دهندهٔ تعداد رکوردهای دیدهای ذخیره‌شدهٔ نهایی است.

با توجه به نمودار رسم‌شده در شکل (۷) مشخص می‌شود که تعداد سطرهای دیدهای ذخیره‌شده توسط الگوریتم پیشنهادیSRTTU-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.

[1] Dimension
[2] Fact
[3] Support threshold

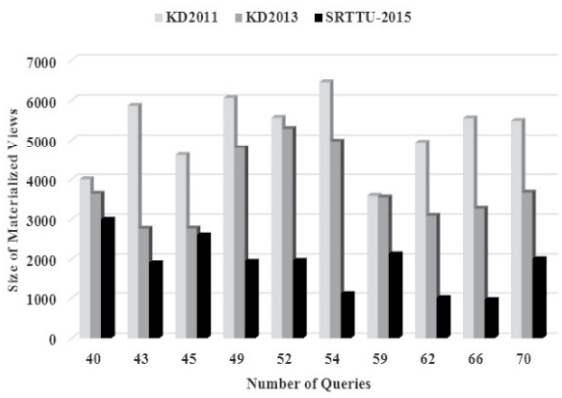
[DOI: 10.18869/acadpub.jsdp.14.1.29]



(شکل-۴): نمودار جعبه‌ای الگوریتم SRTTU-2015

(Figure-4): Flowchart for SRTTU-2015 algorithm

[DOI: 10.18869/acadpub.jsdp.14.1.29]

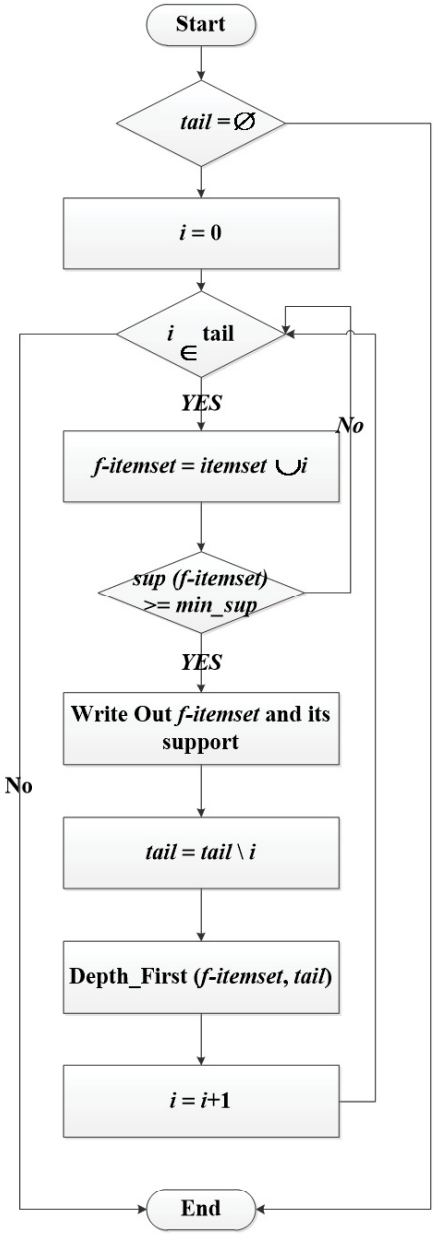


(شکل-۷): مقایسهٔ تعداد سطرهای دیدهای ذخیره‌شده با افزایش تعداد پرس‌وجوها

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

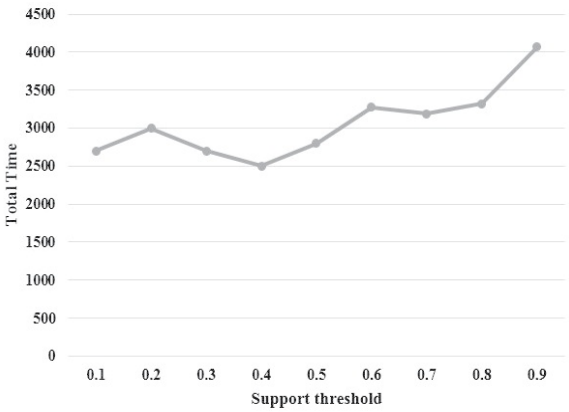
شکل (۸) مقایسهٔ الگوریتم‌های KD2011 و KD2013 و SRTTU-2015 بر اساس معیار شمارهٔ (۲) است. محور افقی نشان‌دهندهٔ تعداد پرس‌وجوها است و محور عمودی نشان‌دهندهٔ زمان کل است. این زمان شامل زمان اجرای الگوریتم و زمان صرف‌شده برای پاسخ به پرس‌وجوهای آزمایش است.

نمودار رسم‌شده در شکل (۸) نشان می‌دهد که زمان کل الگوریتم پیشنهادیSRTTU-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 پرس‌وجوهای آزمایش پاسخ دهد، الگوریتم انتخاب دید مناسب‌تری خواهد بود. با توجه به نمودار رسم‌شده در شکل (۸)، مشاهده می‌شود که الگوریتم پیشنهادی SRTTU-2015 دارای زمان پاسخ کم‌تری بوده است.



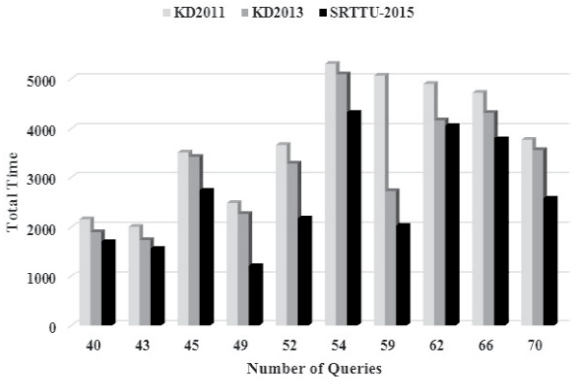
(شکل-۵): نمودار جعبه‌ای تابع اول عمق

(Figure-5): Flowchart for first depth function



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

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

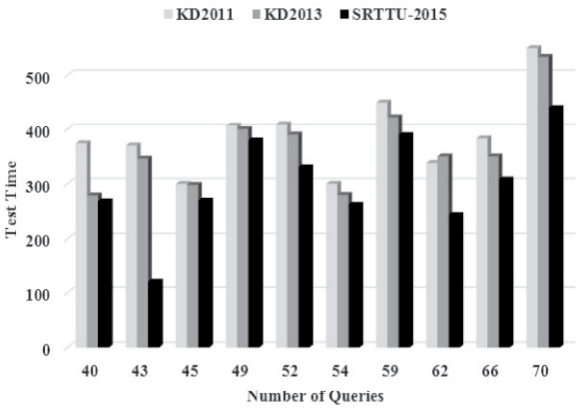


(شکل-۸): مقایسهٔ زمان کل با افزایش تعداد پرس‌وجوها

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

[DOI: 10.18869/acadpub.jsdp.14.1.29]

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



(شکل-۹): مقایسهٔ 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 پیشنهادیSRTTU-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 دادهٔ تحلیلی اجرا شده‌اند، کار می‌کند. در ابتدا بر اساس الگوریتم‌های خوشه‌بندی، پرس‌وجوهای قبلی خوشه‌بندی می‌شوند؛ سپس در هر خوشه، پرس‌وجوهای پرتکرار به‌دست می‌آیند. در مرحلهٔ بعد، پرس‌وجوهای بهینه به‌دست می‌آیند و درنهایت، پرس‌وجوهای بهینه در هر خوشه ادغام می‌شوند تا به‌ازای هر خوشه، یک دید به‌دست آید. الگوریتم پیشنهادی SRTTU-2015 بهبود یافتهٔ الگوریتم‌هایKD2011 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 شمارهٔ (۲) و (۳))، به‌طور متوسط ۲۵ درصد بهبود و از نظر معیار شمارهٔ (۱)، به‌طور متوسط ۵۰ درصد بهبود داشته است. همچنین الگوریتم پیشنهادی نسبت به الگوریتم KD2013، از نظر معیارهای زمانی (معیارهای شمارهٔ (۲) و (۳))، ۲۲ درصد بهبود و از نظر معیار شمارهٔ (۱)، به‌طور متوسط ۴۸ درصد بهبود داشته است. علت این بهبود، استفاده از الگوریتم IndexBittableFI به جای الگوریتم‌های Apriori و DIC است. الگویتم Index-BittableFI آیتم‌های پرتکرار را به‌صورت منساب‌تری نسبت به الگوریتم‌های Apriori و DIC پیدا می‌کند. الگوریتم Apriori با پیچیدگی زمانی $O(n^2)$ آیتم‌های پرتکرار را و الگوریتم Index-BittableFI با پیچیدگی O(nlogn) آیتم‌های پرتکرار را می‌یابد. با توجه به اینکه ر الگوریتم پیشنهادی از Index-BittableFI 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 ۲۶۲۴۱مورخ ۱۳۹۵/۱۰/۷ انجام شده است.

۶-مراجع 6-Refrences

[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

[DOI: 10.18869/acadpub.jsdp.14.1.29]

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 Barfourosh, "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 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.

ریحانه صباغ‌گل 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 پایان‌نامه ایشان، بهبود سرعت پاسخ‌گویی به پرس‌وجو در پایگاه دادهٔ تحلیلی با استفاده از گروه بندی پرس‌وجوها بوده است.

نشانی رایانامه ایشان عبارت است از:

Sabbgh.rsg@gmail.com

[DOI: 10.18869/acadpub.jsdp.14.1.29]

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

[DOI: 10.18869/acadpub.jsdp.14.1.29]