

بهبود الگوریتم انتخاب دید در پایگاه داده تحلیلی با استفاده از یافتن پرس‌وجوهای پر تکرار

ریحانه صباح گل و نگین دانشپور*

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



چکیده

پایگاه داده تحلیلی منبعی برای ذخیره‌سازی داده‌های تاریخی جهت تحلیل است. به طور معمول 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 مناسب ترین راهکار برای ذخیره‌سازی دید، ذخیره‌سازی دیدهای پر استفاده و پرکاربرد است. پرس‌وجوهایی که در قبل مورد استفاده پایگاه داده تحلیلی بوده‌اند، حاوی اطلاعات مهمی هستند که به احتمال 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 دیدهای توأمی پاسخ‌گویی را به بسیاری از پرس‌وجوهایی که در آینده اتفاق خواهند افتاد، دارند. روش پیشنهادی این مقاله از الگوریتم Index-BittableFI برای یافتن دیدهای پر تکرار استفاده کرده است که باعث بهبود روش‌های قبلی و کاهش زمان پاسخ به پرس‌وجوها شده است. آزمایش‌های انجام شده نشان می‌دهند که الگوریتم پیشنهادی از لحظه زمانی نسبت به الگوریتم‌های قبلی ۲۳ درصد و از لحظه فضای ذخیره‌سازی پنجاه درصد بهبود داشته است.

واژگان کلیدی: پایگاه داده تحلیلی، پرس‌وجوهای پر تکرار، 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 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 مهم‌ترین عامل در الگوریتم‌های انتخاب دید، زمان است، بنابراین در این مقاله، الگوریتمی پیشنهاد شده است که در زمان کمتری، به پرس و جوهای تحلیلی، پاسخ می‌دهد.

الگوریتم انتخاب دید پیشنهادشده در این مقاله، با استفاده از الگوریتم Index-BittableFI [4] پرس و جوهای پر تکرار را در هر خوشه به دست می‌آورد. الگوریتم ارائه شده در [3] و [2] و الگوریتم پیشنهادی این مقاله، پیاده‌سازی شده‌اند و مشاهده شده است که الگوریتم پیشنهادی نسبت به الگوریتم‌های بیان شده 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 جمله نخستین الگوریتم‌های مطرح شده 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 ازتابع 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 بعد کم مناسب هستند؛ ولی برای مسائل با بعد 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^۶. 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

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 تبلیغ^۱ و روش مشتق^۲. در روش نخست داده‌ها به وسیله پرس‌وجوهی کاربر یک‌پارچه^۳ می‌شوند. این روش، به علت این که داده‌ها پس از اجرای پرس‌وجوهی کاربر یک‌پارچه می‌شوند، دارای تأخیر زیادی است و درنتیجه، روش مناسبی نیست. روش دوم از پایگاه داده تحلیلی استفاده می‌کند. در این روش اطلاعات مورد نیاز از پایگاه‌های داده مختلف استخراج^۴ شده و پس از پاک‌سازی داده‌ها، در منبع بزرگی به نام پایگاه داده تحلیلی بارگذاری^۵ می‌شوند [1]. در این روش برای تصمیم‌گیری^۶ از پرس‌وجوهای تحلیلی استفاده می‌شود. امروزه بیشتر سازمان‌ها و ادارات از این روش برای پاسخ به پرس‌وجوهای تحلیلی تحلیل‌گران استفاده می‌کنند [2].

پایگاه داده تحلیلی به منظور کاهش زمان پاسخ به پرس‌وجوهای تحلیلی، از دید ذخیره‌شده^۷ استفاده می‌کند. با توجه به محدودیت فضای ذخیره‌سازی، و نیز زمان زیاد به روزنگه‌داشتن دیدها، نمی‌توان تمام دیدهای ممکن را ذخیره کرد؛ بنابراین باید از بین تمامی دیدهای ممکن، تنها مجموعه‌ای از دیدهای را ذخیره کرد. دیدهای می‌توانند بر اساس پرس‌وجوهایی که در قبل مورد استفاده پایگاه داده تحلیلی بوده‌اند، انتخاب و ذخیره شوند [3],[2]. علت استفاده از پرس‌وجوهای قبلی، این است که به احتمال زیاد در آینده دوباره اتفاق خواهد افتاد. روند کار الگوریتم‌های بیان شده در [3],[2] به این صورت است که ابتدا پرس‌وجوهای قبلی خوشه‌بندی می‌شوند؛ سپس در هر خوشه پرس‌وجوهای پر تکرار به دست می‌آیند. در مرحله بعد پرس‌وجوهای بهینه انتخاب می‌شوند و درنهایت پرس‌وجوهای بهینه در هر خوشه پیوند^۸ می‌شوند. این الگوریتم به علت این که برای یافتن پرس‌وجوهای پر تکرار در هر خوشه از الگوریتم‌های مقدماتی همچون Apriori و DIC استفاده کرده است، دارای زمان

⁶ Decision making

⁷ Materialized View

⁸ Join

⁹ Multi-View Processing Plan

¹ lazy

² eager

³ integrate

⁴ extraction

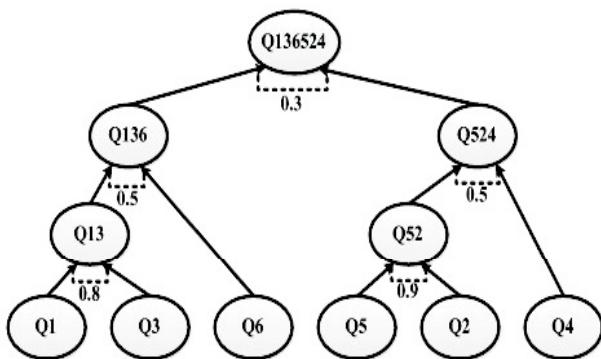
⁵ load

حاوی اطلاعات مفیدی هستند که به احتمال زیاد در آینده نیز مورد استفاده خواهد بود. در الگوریتمی که در ادامه توضیح داده می‌شود، روش یافتن پرس‌وجوهای پرتکرار در روش MVCF بیان شده در منبع [3].² بهبود داده شده که باعث MVCF نهایی الگوریتم انتخاب دید با استفاده از روش است. الگوریتم بیان شده در منبع [3].² با توجه به نام نویسنده‌گان، 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 سلسله‌مراتبی³، دسته‌بندی می‌شوند. به هر یک از این دسته‌ها، خوش⁴ یا دامنه⁵ گفته می‌شود. برای خوش‌بندی کردن پرس‌وجوهای از معیار شباهت جاکارد⁶ [2] استفاده می‌شود.

برای یافتن خوش‌هایی که دارای پرس‌وجوهای پرتکرار هستند، ابتدا باید الگوریتم ادغام پرس‌وجوها⁸ [2] اجرا شود تا براساس معیار شباهت جاکارد، پرس‌وجوها با هم ادغام شوند؛ سپس براساس ترتیب ادغام پرس‌وجوها درخت آن را تشکیل داده و خوش‌های از روی درخت یافته شوند.

برای ساخت درخت از روی الگوریتم ادغام پرس‌وجوها، روند ساخت خوش‌ها به صورت درخت رسم می‌شود. با توجه به الگوریتم ادغام پرس‌وجوها، خوش‌ها براساس بیشترین میزان شباهت به هم از پایین به بالا ادغام می‌شوند تا این که در آخرین مرحله همه پرس‌وجوها در یک خوش قرار می‌گیرند و درنهایت درخت ساخته می‌شود. نمونه‌ای از این درخت در شکا، (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 دیدهای انجام می‌شود؛ به این صورت که دیدهایی که هزینه بالایی دارند، حذف شده و دیدهایی که هزینه کمتری دارند اضافه می‌شوند. در صورتی که 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 خوش‌بندی و داده‌کاوی، دیدهای مناسب انتخاب می‌شوند [26]-[24],[3]. 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 خوش‌بندی، پرس‌وجوهای قبلی خوش‌بندی می‌شوند.

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

ج) سپس در هر خوش، پرس‌وجوهای بهینه انتخاب می‌شوند.

د) در این مرحله، در هر خوش پرس‌وجوهای بهینه ادغام می‌شوند تا بهزاری هر خوش، یک دید به دست آید و آن دید ذخیره می‌شود.

در [3],[2] از الگوریتم‌های پایه مانند الگوریتم Apriori و DIC^۳ برای یافتن آیتم‌های پرتکرار در هر خوش 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 پرس‌وجوهایی 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 I -itemset‌های پر تکرار است. هر عضو این آرایه به صورت زوج دوتایی ($item, subsume$) است، که $item$ I -itemset ($item$, $subsume$) است، که $subsume(item) = \{j \in I \mid item \prec j \wedge g(item) \subseteq g(j)\}$ (۲) $subsume(item)$ به این معنی است که اگر $j \in subsume(item)$ بعد از $item$ قرار دارد. برای پیدا کردن آیتم‌های پر تکرار، ابتدا باید در هر خوشة، آرایه اندیسی برای هر آیتم به دست آید؛ سپس با در دست داشتن آرایه اندیسی تمام آیتم‌ها، با استفاده از الگوریتم Index-BittableFI آیتم‌های پر تکرار در هر خوشه به دست آورده شوند. در قسمت (الف) الگوریتم ساختن آرایه اندیسی توضیح داده شده است و در قسمت (ب) الگوریتم Index-BittableFI توضیح داده شده است. اساس الگوریتم‌های بیان شده در قسمت‌های الف و ب بر پایه‌ی منبع [۴] است و در این مقاله برای این مسئله بازنویسی شده است.

الف) الگوریتم ساختن آرایه اندیسی:

ابتدا الگوریتم ساختن آرایه اندیسی مورد بررسی قرار می‌گیرد. 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
           $index[f].subsume \leftarrow index[f].item$ ;
        end if
      end for
    end for
    Write Out  $index\ array[i]$ ;
  end for
End
```

(شکل-۲): الگوریتم ایجاد آرایه اندیسی برای الگوریتم

SRTTU-2015
(Figure-2): Producing Index Array for SRTTU-2015 algorithm

³ itemset

⁴ Index array

پس از ساخت درخت، گره‌های درخت از بالا به پایین ملاقات^۱ می‌شوند تا زمانی که خوشه‌ای پیدا شود که در جهش باهشتی بیشتر از حداقل شباهت حد آستانه^۲ داشته باشد. این خوشه‌ها نشان‌دهنده دامنه‌ها خواهند بود. به عنوان مثال در شکل (۱)، اگر حد آستانه $4/0$ 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 پرس و جوهای پر تکرار در هر خوشه یافته شوند. روش یافتن پرس و جوهای پر تکرار برای هر خوشه، در الگوریتم‌های KD2011 و KD2013، به ترتیب روش Apriori و DIC است؛ الگوریتم‌های Apriori و DIC الگوریتم‌های قدیمی هستند و سبب می‌شوند که در زمان اجرای بالایی به پرس و جوهای 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 [۴] برای بهبود زمان اجرا استفاده شده است. این الگوریتم برای پرس و جوهای هر خوشه بررسی می‌شود تا پرس و جوهای پر تکرار در هر خوشه به دست آیند. در این مرحله، هر کدام از پرس و جوهای را یک مجموعه آیتم^۳ می‌نامند. یک مجموعه آیتم که دارای k تا آیتم است، k -itemset نامیده می‌شود. 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 می‌شود. X نشان‌دهنده تعداد پرس و جوهایی است که شامل آیتم X هستند. اگر $sup(X) > min_sup$ یک آیتم پر تکرار نامیده می‌شود.

در الگوریتم Index-BittableFI، برای کاهش فضای جستجو، آرایه اندیسی^۴ تعریف می‌شود. 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 می‌شود و D ، پایگاه داده‌ای است که شامل تمامی پرس و جوهای هر خوشه است و t ، مجموعه‌ای از آیتم‌هایی است که هر پرس و جو شامل می‌شود. این آیتم‌ها فیلد‌های مورد استفاده برای هر پرس و جو هستند.

¹ visit

² Minimum threshold

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

```

Procedure IndexBitTableFI
Input: index array for each cluster  $index[i]$ , min_sup
Output: frequent itemsets
Begin
for each cluster  $i$  do
    for each element  $index[j]$  of  $index[i]$  do
        Write Out  $index[j].item$  and its support;
        if ( $index[j].subsum$  ==  $\emptyset$ ) 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  $s\text{-item} \subseteq index[j].subsum$  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  $index[j].subsum$ ;
                // delete items included by  $index[j].subsum$  from
                //  $t(index[j].item)$ 
                Depth_First ( $index[j].item$ , tail);
                for each element  $s\text{-item} \subseteq index[j].subsum$  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

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

$$\text{Maximise } 13Q_1 + 15Q_7 + 28Q_6 + 35Q_{17} + 32Q_{20} \quad (3)$$

subject to

$$120Q_1 + 250Q_7 + 430Q_6 + 310Q_{17} + 360Q_{20} \leq 1000$$

and

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

³ Optimal Queries

در ابتدا پایگاه داده مربوط به هر خوشه یعنی D_i پویش می‌شود تا آیتم‌های پر تکرار به دست آیند؛ سپس آیتم‌های پر تکرار به صورت صعودی مرتب می‌شوند و آیتم‌ها به ترتیب به هر عضو از آرایه اندیسی خوشة i نسبت داده می‌شوند. در مرحله بعد جدول بیتی¹ از پایگاه داده D_i ساخته می‌شود. نحوه ساخت این جدول به این صورت است که اگر پرس‌وجوی T شامل آیتم t باشد، آن‌گاه بیت t 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 پرس‌وجوهایی که شامل $index[j].item$ می‌شوند، اشتراک گرفته می‌شوند؛ سپس تمام آیتم‌هایی که دارای اندیس بالاتری نسبت به ز هستند، در $index[j].subsum$ قرار می‌گیرد؛ درنهایت به عنوان خروجی برای هر خوشه، یک آرایه اندیسی برگردانده می‌شود.

ب) الگوریتم :Index-BitTableFI

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

در ابتدا آیتم‌ها و $support$ آن‌ها نوشته می‌شود. در مرحله بعد $subsum$ مربوط به هر آیتم بررسی می‌شود. در صورت تهی بودن $subsum$ آن، اگر $support$ آن آیتم از min_sup بیشتر باشد، آنگاه تابع نخست‌عمق² اجرا می‌شود. حال اگر $subsum$ آن غیر تهی باشد، تمام زیرمجموعه‌های غیر تهی $subsum$ را به دست آورده، سپس آیتم مذکور را با هر کدام از زیرمجموعه‌ها ادغام می‌کنیم. در این حالت هر کدام از مجموعه‌های به دست آمده یک مجموعه آیتم پر تکرار هستند که 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 به دست آوردن پرس‌وجوهای پر تکرار در هر خوشه، باید پرس‌وجوهای بهینه³ با توجه به فضای موجود برای هر خوشه در هر خوشه به دست 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 دیدهای ذخیره شده 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، مدت زمان پاسخ به پرس‌وجوهای آزمایش است.

در این آزمایش‌ها برای به دست آوردن مقدار معیار مربوطه، پنج بار 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 به دست آوردن 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، $0.4/0$ است. بنابراین همان‌طور که در بخش ۳ توضیح داده شد، با استفاده از این پارامتر، خوش‌ها 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 دیدهای ذخیره شده نهایی است.

با توجه به نمودار رسم شده در شکل (۷) مشخص می‌شود که تعداد سطرهای دیدهای ذخیره شده توسط الگوریتم پیشنهادی SRTTU-2015 همیشه کمتر از تعداد سطرهای به دست 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.

³ Support threshold

پس از به دست آوردن پرس‌وجوهای بهینه در هر خوش، باید این پرس‌وجوهای ادغام شوند تا بازای هر خوش، یک دید ذخیره شود. هدف از این مرحله این است که برای هر دامنه، یک پرس‌وجوی بهینه، ذخیره شود. در این مرحله جداول Natural Outer Join مربوط به هر پرس‌وجو در هر خوش، آن دید ذخیره می‌شود [۳]. شکل (۴) الگوریتم پیشنهادی را به صورت روند نما نشان می‌دهد.

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

۴-نتایج شبیه‌سازی

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

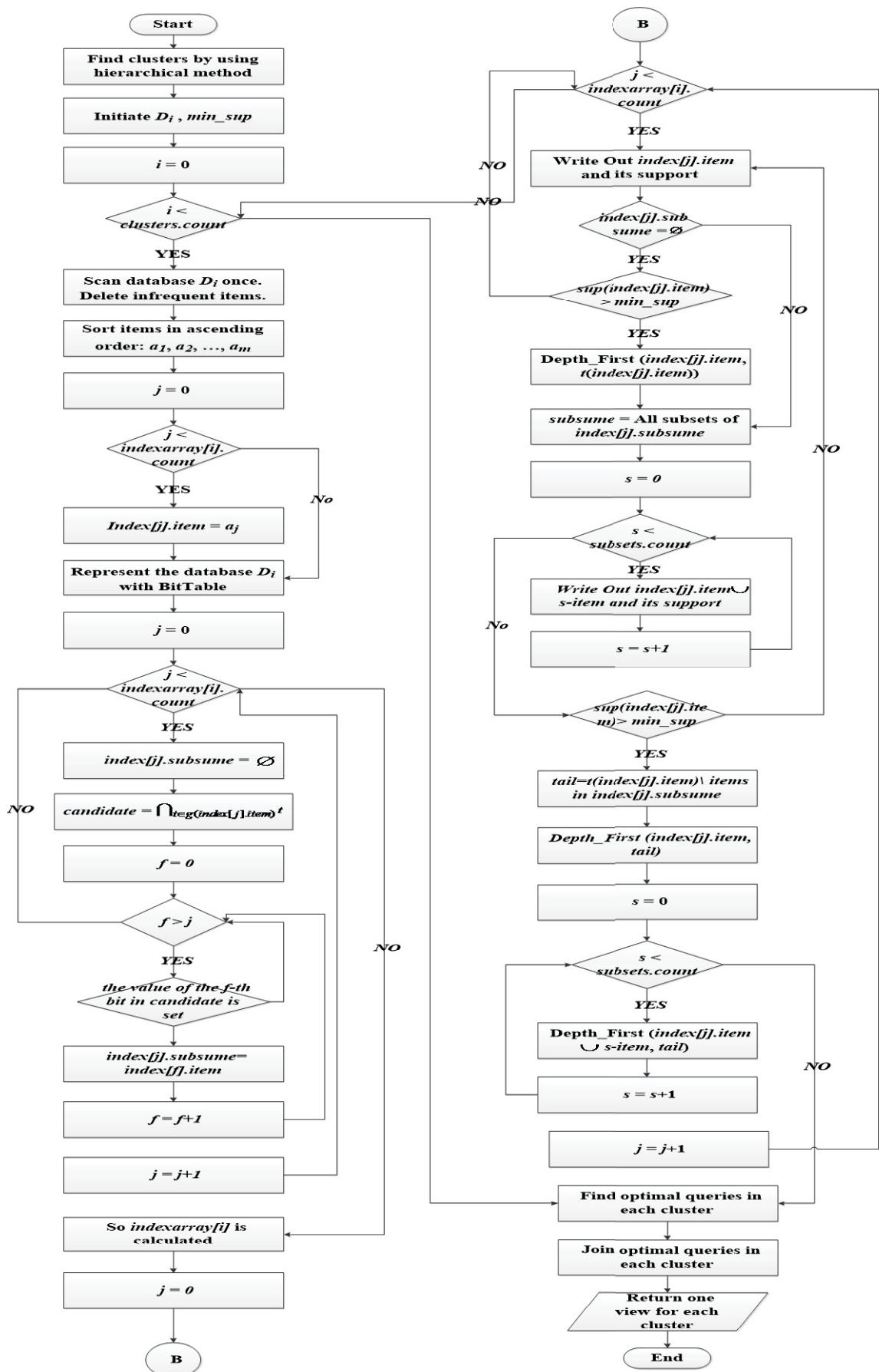
داده‌های این پایگاه داده، به صورت داده تصادفی توسط نرم‌افزار Microsoft Visual Studio تولید شده است. با توجه به این که داده‌ها به صورت تصادفی تولید شده‌اند، پس در آزمایش‌های صورت گرفته نتایج حاصل به یک نوع پایگاه داده خاص حساس نیست. سامانه مورد استفاده برای پیاده‌سازی الگوریتم‌ها دارای CPU Corei3 2.2GHz و 4GB RAM است. پایگاه داده تحلیلی مورد استفاده این آزمایش‌ها، دارای ۵ جدول بعد^۱ و یک جدول حقیقت^۲ می‌باشد.

برای مقایسه الگوریتم‌های KD2011 و 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، به صورت داده‌های تصادفی تولید می‌شوند؛ پس از اجرای الگوریتم انتخاب دید، این الگوریتم توسط پرس‌وجوهای آزمایش مورد آزمایش قرار می‌گیرد.

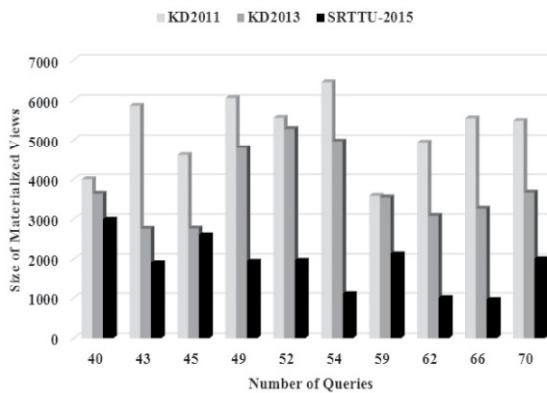
معیارهای مورد بررسی در این آزمایش‌ها به شرح ذیل هستند:

۱. تعداد سطرهای دیدهای ذخیره شده: پس از اجرای الگوریتم انتخاب دید، دیدهایی برای ذخیره‌سازی انتخاب

¹ Dimension² Fact



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

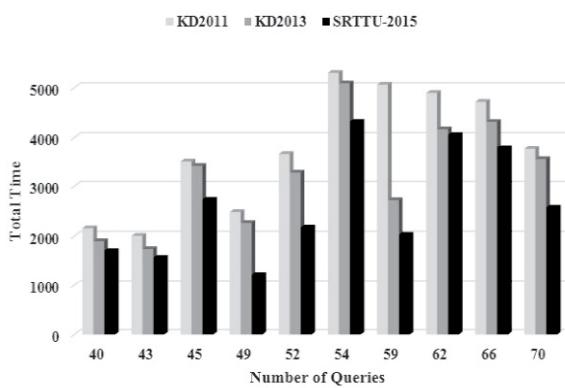


(شکل-۷): مقایسه تعداد سطرهای دیدهای ذخیره شده با افزایش تعداد پرس و جوهای

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

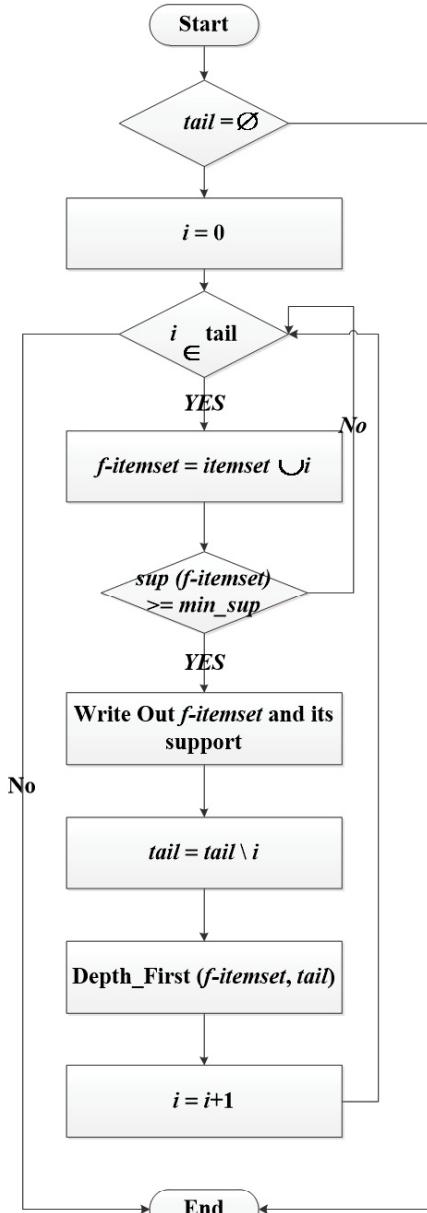
شکل (۸) مقایسه الگوریتم های KD2011 و KD2013 و SRTTU-2015 بر اساس معیار شماره (۲) است. محور افقی نشان دهنده تعداد پرس و جوهای این زمان شامل زمان اجرای الگوریتم و زمان صرف شده برای پاسخ به پرس و جوهای آزمایش است.

نمودار رسم شده در شکل (۸) نشان می دهد که زمان کل الگوریتم پیشنهادی SRTTU-2015 همیشه کمتر از زمان کل الگوریتم های KD2011 و KD2013 است. با توجه به اهمیت زمان، هر الگوریتم انتخاب دیدی که در زمان کمتری به پرس و جوهای آزمایش پاسخ دهد، الگوریتم انتخاب دید مناسب تری 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.

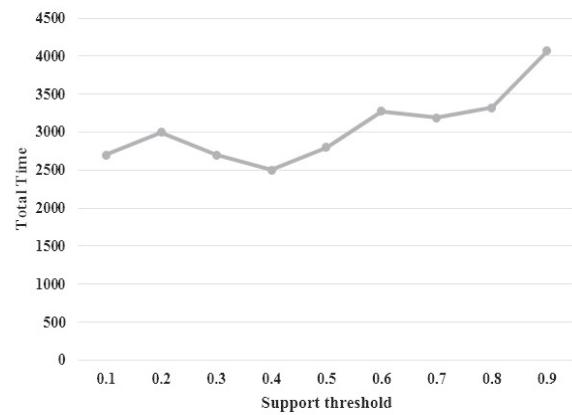


(شکل-۸): مقایسه زمان کل با افزایش تعداد پرس و جوهای

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



(شکل-۵): نمودار جعبه ایتابع 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 رسم شده در بخش قبل مشاهده می‌شود که الگوریتم SRTTU-2015 نسبت به الگوریتم KD2011 از نظر معیارهای زمانی (معیارهای شماره (۲) و (۳))، به طور متوسط ۲۵ درصد بهبود و از نظر معیار شماره (۱)، به طور 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)، به طور متوسط ۴۸ درصد بهبود داشته است. علت این بهبود، استفاده از الگوریتم DIC و Apriori در بدست آوردن آیتم‌های IndexBittableFI الگوریتم Index-BittableFI را به صورت مناسب‌تری 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 آیتم‌های پرتکرار را می‌یابد. با توجه به اینکه ر $O(n\log n)$ الگوریتم پیشنهادی از Index-BittableFI استفاده شده است، بنابراین الگوریتم پیشنهادی از SRTTU-2015 با پیچیدگی کمتری، دیدهای مناسب را انتخاب می‌کند.

سپاس گزاری

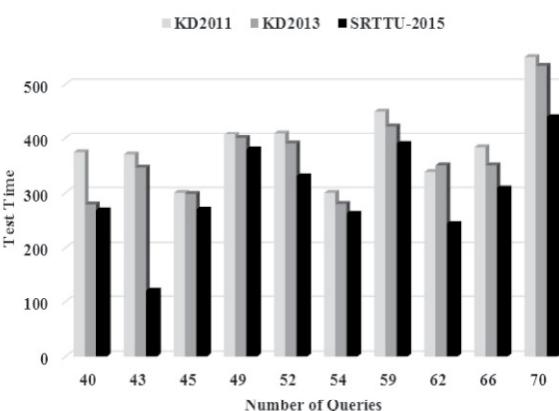
این پژوهش با حمایت مالی دانشگاه تربیت دبیر شهید رجایی طبق قرارداد شماره ۲۶۲۴۱ مورخ ۱۳۹۵/۱۰/۷ انجام شده است.

6-References

6-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 پرس‌وجوهای آزمایش است. در این نمودار محور افقی نشان‌دهنده تعداد پرس‌وجوهای آزمایش است و محور عمودی نشان‌دهنده زمان صرف شده برای پاسخ به پرس‌وجوهای آزمایش است.



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

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

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

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

5-نتیجه‌گیری

در این مقاله، الگوریتمی برای انتخاب دید ارائه شده است 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 به دست می‌آیند. در مرحله بعد، پرس‌وجوهای بهینه به دست می‌آیند و در نهایت، پرس‌وجوهای بهینه در هر خوشه ادغام می‌شوند تا به‌ازای هر خوشه، یک دید به دست 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.
- ريحانه صباح 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 پاسخ‌گویی به پرس‌وجوه در پایگاه داده تحلیلی با استفاده از گروه بندی پرس‌وجوها بوده است.**
- شناختی رایانمehr ایشان عبارت است از:
- Sabbagh.rsg@gmail.com



کرد. موضوع پایان نامه ایشان، بهبود سرعت پاسخ‌گویی به پرس‌وجوه در پایگاه داده تحلیلی با استفاده از گروه بندی پرس‌وجوها بوده است.

شناختی رایانمehr ایشان عبارت است از:

Sabbagh.rsg@gmail.com

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 Barfouros, "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*

نگین دانشپور استادیار دانشکده
مهندسی کامپیوتر دانشگاه تربیت دبیر
شهید رجایی است. نامبرده تحصیلات
خود را در مقطع کارشناسی مهندسی
کامپیوتر-سخت افزار در سال ۱۳۷۸ در



دانشگاه شهید بهشتی و کارشناسی ارشد مهندسی کامپیوتر-
نرم افزار در سال ۱۳۸۱ در دانشگاه صنعتی امیرکبیر به پایان
رسانده است، و در سال ۱۳۸۹ دکترای خود در رشته مهندسی
کامپیوتر-نرم افزار را از دانشگاه صنعتی امیرکبیر اخذ کرده
است. زمینه های پژوهشی مورد علاقه ایشان عبارتند از: پایگاه
داده، پایگاه داده تحلیلی، سامانه های تصمیم گیر، و داده کاوی.

نشانی ریاضی ایشان عبارت است از:

ndaneshpour@srttu.edu

فصلنامه

سال ۱۳۹۶ شماره ۱ پیاپی ۳۱

