افزایش کارآیی پرس‌وجوهای پایگاه داده تحلیلی با نگاشت مکعب مفهومی به فضای دوبعدی

محمدکریم سهرابی و احمد عبدالله‌زاده بارفروش

چکیده: پایگاه داده تحلیلی و پردازش تحلیلی برخط، از اجزا اصلی سیستم‌های تصمیم‌یار به‌شمار می‌روند که به‌طور روزافزون در مباحث مرتبط با پایگاه‌های داده مورد توجه قرار گرفته‌اند. سیستم‌های تصمیم‌یار نسبت به سیستم‌های پردازش تراکنش برخط، نیازمندی‌های متفاوتی دارند. در این سیستم‌ها بهینه‌سازی پرس‌وجوها و پردازش کارای مکعب‌های داده‌ای، در ساختار پایگاه داده تحلیلی نقش اساسی در عملکرد سیستم ایفا می‌کند.

در این مقاله با به‌کارگیری تکنیک‌های محاسبه از پایین به بالای عناصر شبکه جستجو، روش کارایی برای پردازش پرس‌وجو در پایگاه داده تحلیلی و انجام محاسبات مکعب داده ارائه شده است. بررسی نتایج به‌دست آمده بر مبنای پارامترهای ارزیابی حکایت از آن دارد که الگوریتم ارائه‌شده در این مقاله نسبت به بهترین الگوریتم‌هایی که پیش از آن ارائه شده‌اند، عملکرد بهتری (بر اساس معیار زمان اجرا) از خود نشان می‌دهد و سرعت آن در اجرای پرس‌وجوهای یکنوا و با حجم داده‌های بسیار زیاد، به‌مراتب بهتر از الگوریتم‌های پیش از آن است. ضمن اینکه با توجه به نگاه دوبعدی ایجادشده توسط این الگوریتم به مسأله مکعب و تبدیل مکعب به ساختار ابرگراف، میزان حافظه مورد نیاز این الگوریتم در مواردی که مجموع‌سازی بر روی زیرمجموعه‌ای از ابعاد مکعب صورت پذیرد، کمتر از حافظه مصرف‌شده توسط الگوریتم‌های پیش از آن است.

کلید واژه: پایگاه داده تحلیلی، پردازش تحلیلی برخط، مدل داده چندبعدی، مکعب داده.

1- مقدمه

پایگاه داده تحلیلی، ابزاری برای پشتیبانی تصمیم است که از منابع داده سازمان‌ها و ارگان‌های متفاوت تهیه می‌شود. این پایگاه داده بستر مناسبی فراهم می‌آورد که داده‌های بایگانی‌شده در پایگاه‌های داده عملیاتی، به‌صورت مجتمع و سازمان‌یافته درآیند و برای استخراج دانش مناسب باشند. تعریف ارائه‌شده توسط اینمون[1] برای پایگاه داده تحلیلی به‌صورت زیر است: "پایگاه داده تحلیلی، یک مجموعه موضوع - گرا، یک‌پارچه، متکی بر بازه‌های زمانی متفاوت (متغیر با زمان) و تغییرناپذیر از داده‌ها است که برای پشتیبانی مدیریت پردازش تصمیم‌گیری (تصمیم‌یاری) به‌کار می‌رود [1]."

پایگاه‌های داده تحلیلی با توجه به اینکه اطلاعات مجتمع‌شده از چندین پایگاه داده عملیاتی را در خود دارند شامل حجم عظیمی از داده هستند. این مسأله سبب بروز مشکلاتی در نگهداری و به‌کارگیری پایگاه‌های مزبور می‌شود. برای اینکه بتوان به‌صورت کارا به پرس‌وجوهایی که از پایگاه داده تحلیلی می‌شود پاسخ داد، باید از روش‌هایی کارا برای دستیابی به داده و پردازش پرس‌وجو استفاده کرد. به این منظور الگوریتم‌های فراوانی ارائه شده‌اند که هدف آنها انجام پرس‌وجوها به نحوی است که زمان لازم برای پاسخگویی کمینه باشد و در عین حال فضای حافظه مورد نیاز برای انجام پردازش‌ها بهینه و یا دست کم قابل قبول باشد.

پایگاه‌های داده تحلیلی را می‌توان از دیدگاه مفهومی به‌صورت مجموعه‌ای چندبعدی از داده‌ها دید. این دیدگاه چندبعدی را مکعب داده می‌نامند. مکعب داده در [2] ارائه شده و برخی از ویژگی‌های مفید محاسبه مکعب بیان شده است. منظور از محاسبه مکعب داده، پاسخ‌گویی به پرس‌وجوهای تحلیلی است که از پایگاه داده تحلیلی متناظر با آن می‌شود. چون این پرس‌وجوها تحلیلی هستند و به بررسی داده‌ها در بعد کلان می‌پردازند، نیاز به مجموع‌سازی داده‌ها در ابعاد مختلف مکعب دارند.

سه نوع تابع برای انجام عملیات مجموع‌سازی تعریف می‌شوند [3]. مجموعه‌ای مانند T از چندگانه‌ها (رکوردها) را در نظر بگیرید. فرض کنید که مجموعه $\{S_i \mid i=1,2,...,n\}$ یک مجموعه کامل از زیرمجموعه‌های جدا از هم T باشد یعنی $\bigcup_i S_i = T$ و $\bigcap_i S_i = \{\}$.

- تابع مجموع‌ساز F تابع توزیعی خوانده می‌شود اگر تابعی مانند G موجود باشد طوری که $F(T) = G(\{F(S_i) \mid i=1,2,...,n\})$. توابع SUM، MIN، و MAX همگی توابعی توزیعی هستند که در آنها $G = F$ است. تابع $COUNT$ نیز یک تابع توزیعی است که در آن $G = SUM$ است.
- تابع مجموع‌ساز F جبری است اگر تابع M مقداری G و تابع H موجود باشند طوری که $F(T) = H(\{G(S_i) \mid i=1,2,...,n\})$ که در آن مقدار M برخلاف مقادیر $|T|$ و n ثابت است. همه تابع‌های توزیعی توابعی جبری نیز هستند. علاوه بر آنها توابعی نظیر میانگین، انحراف معیار استاندارد، $MaxN$ و $MinN$ نیز توابعی جبری هستند. به‌عنوان مثال، در تابع میانگین ($Average$)، تابع G مقادیر SUM و $COUNT$ را به‌دست می‌آورد و تابع H نتایج به‌دست آمده از تابع G را بر هم تقسیم می‌نماید.
- یک تابع مجموع‌ساز نظیر F را هولیستیک می‌نامند اگر جبری نباشد. برای مثال توابع میانه ($Median$) و رتبه ($Rank$) نمونه‌ای از توابع هولیستیک هستند.

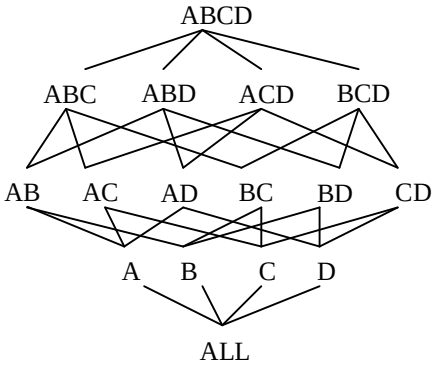
توابع جبری دارای خاصیتی کلیدی هستند که بر اساس آن مجموعه‌های جزئی‌تر (یعنی با تعداد ابعاد بیشتر) می‌توانند برای محاسبه مجموع‌های کلی‌تر مورد استفاده قرار گیرند. این خاصیت منجر به وجودآمدن یک ترتیب جزئی (یعنی یک شبکه) بر روی همه گروه‌بندی‌های یک مکعب می‌گردد. یک گروه‌بندی به‌عنوان فرزند یک

این مقاله در تاریخ 10 اردیبهشت ماه 1386 دریافت و در تاریخ 10 شهریور ماه 1386 بازنگری شد.

محمدکریم سهرابی، آزمایشگاه سیستم‌های هوشمند، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر، تهران (email: Amir_sohraby@yahoo.com).

احمد عبدالله‌زاده بارفروش، آزمایشگاه سیستم‌های هوشمند، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر، تهران (email: Ahmad@aut.ac.ir).

1. Inmon



شکل 1: شبکه چهاربعدی.

گروه‌بندی‌های یک مکعب می‌گردد. یک گروه‌بندی به‌عنوان فرزند یک گروه‌بندی پدر شناخته می‌شود اگر گروه‌بندی پدر بتواند در محاسبه فرزند به کار رود (و هیچ گروه‌بندی دیگری بین پدر و فرزند نباشد). شکل 1 یک شبکه ساده را نشان می‌دهد که در آن A، B، C و D ابعاد مکعب هستند. گروه‌بندی‌ها، گره‌های آن هستند و یال‌ها نشان‌دهنده رابطه پدر و فرزندی می‌باشند. هدف نهایی همه الگوریتم‌ها و تکنیک‌های محاسبه مکعب این است که بتوانند نودهای شبکه فوق را طوری محاسبه نمایند که میزان هم‌پوشانی محاسبات انجام‌شده کمینه باشد. بنابراین همه این الگوریتم‌ها شبکه مزبور را تشکیل داده و مورد بررسی قرار می‌دهند. تفاوت روش‌ها و الگوریتم‌های مختلف در نحوه ایجاد و بررسی درخت است. مقاله حاضر نیز برای پرهیز هرچه بیشتر از هم‌پوشانی محاسباتی، از رویکرد پایین به بالا برای این محاسبات استفاده می‌کند.

سازمان‌دهی مطالب در این مقاله بر اساس ساختار زیر است. بخش اول این مقاله به توضیح مقدمات لازم در زمینه مسأله محاسبات مکعب داده پرداخته است. در بخش دوم ضمن تعریف مسأله پرس‌وجوهای آستانه‌ای برای مکعب داده، به ارائه الگوریتم‌های پیشین می‌پردازیم و نحوه مواجهه این الگوریتم‌ها را با مسأله، مورد بررسی قرار خواهیم داد. بخش سوم به ارائه الگوریتم جدید Ex - Cube اختصاص دارد. در این بخش ساختار ابرگراف معرفی می‌شود. روش حل مسأله بر اساس آن مورد بررسی قرار می‌گیرد و برای آن الگوریتم ارائه می‌شود. ضمن آنکه کارآیی الگوریتم ارائه‌شده نیز از دیدگاه زمان اجرا و فضای حافظه مورد تحلیل نظری قرار خواهد گرفت. در بخش چهارم نتایج حاصل از پیاده‌سازی الگوریتم و مقایسه آن با الگوریتم‌های پیشین بر اساس پارامترهای مطرح مورد ارزیابی قرار می‌گیرد. نتیجه‌گیری در بخش آخر صورت می‌پذیرد و در پایان، مراجع مورد استفاده معرفی می‌گردند.

2- پرس‌وجوهای آستانه‌ای

یک پرس‌وجوی آستانه‌ای، یک تابع مجموع‌ساز را بر روی یک ویژگی (یا مجموعه‌ای از ویژگی‌ها) اعمال می‌کند و سپس مجموع‌هایی را که مقدار آنها از حد معینی کمتر باشند، حذف می‌کند. چنین پرس‌وجوهایی به این دلیل آستانه‌ای خوانده می‌شوند که تعداد مقادیر جداگانه *target*ها (که همانند یک کوه یخ هستند) بسیار زیاد است، در حالی که پاسخ پرس‌وجو یعنی *target*هایی که معمولاً اتفاق می‌افتند (قسمت آستانه‌ای از کوه یخ که روی آب قرار دارد)، بسیار کم است.

پرس‌وجوی آستانه‌ای نمونه‌ای که به‌عنوان مدل در این بخش ملاحظه می‌شود مبتنی بر یک رابطه با نام R و با ساختار $R(target1, target2, \ldots, targetk, rest)$ و یک حد آستانه‌ای به نام T است. این پرس‌وجو به‌صورت زیر است

جدول 1: جدول فروش (SALES).

Cust-gr (مشتری)	Mon (ماه)	City (شهر)	Product (محصول)	Price (قیمت)
Busi	Jan	Vanc	Printer	800
Edu	Jan	Toro	TV	500
Busi	Feb	Mon	Camera	1200
Edu	Feb	Vanc	Laptop	100
Edu	Jan	Vanc	HD	200

```
SELECT   target1, target2, …, targetk, count(rest)
FROM     R
GROUPBY  target1, target2, …, targetk
HAVING   count(rest)>=T
```

به‌عنوان مثال جدول 1 را در نظر بگیرید.

جدول دارای چهار بعد (محصول، شهر، ماه و نوع مشتری) است و یک ویژگی قابل اندازه‌گیری (قیمت) دارد. بر روی این جدول می‌توان پرس‌وجوهای مختلفی را در نظر گرفت که بر اساس برخی از ابعاد، مقدار قیمت را مجموع‌سازی می‌نمایند. اما یک در پرس‌وجوی آستانه‌ای حد پایینی برای مقدار این مجموع‌سازی ذکر می‌شود. مثلاً این پرس‌وجو که در کدام شهر، در عرض یک ماه بیش از 1000 دلار فروش داشته‌ایم با پرس‌وجوی آستانه‌ای زیر قابل پاسخ‌گویی است

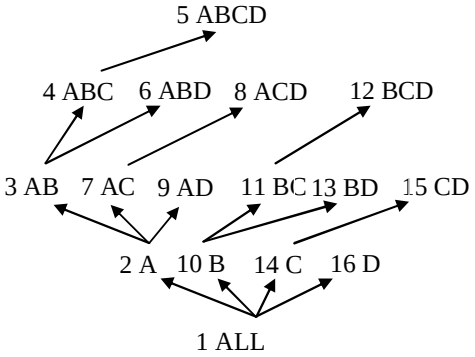
```
SELECT   Mon, City, Sum(Price)
FROM     Sales
GROUPBY  Mon, City
HAVING   Sum(Price)>=1000
```

به‌دلیل کم‌اهمیت بودن داده‌هایی که حاصل مجموع‌سازی بر روی آنها از حد آستانه‌ای کمتر است، از یک سو، و تأثیر حذف این داده‌ها از محاسبات در افزایش کارآیی مجموع‌سازی از سوی دیگر، در بسیاری از کاربردها استفاده از مکعب‌های آستانه‌ای جایگزین مکعب‌های کامل می‌گردد. بسیاری از الگوریتم‌های ارائه‌شده برای محاسبه مکعب نیز دارای قابلیت محاسبه هر دو نوع آستانه‌ای و کامل مکعب می‌باشند.

برای بررسی این الگوریتم‌ها تمرکز خود را معطوف به رابطه R، به‌عنوان جدول حاوی عناصر مکعب می‌نماییم. در ابتدا فرض می‌کنیم که رابطه R تنها دارای زوج $<target, rest>$ است و از قبل محاسبه و ذخیره‌سازی گشته است. این فرض برای سادگی صورت گرفته و در ادامه نشان داده شده است که الگوریتم‌های ارائه‌شده به‌سادگی قابل تعمیم به حالت‌هایی هستند که در آنها R دارای مجموعه‌ای از *target*های چندگانه باشد و پیش‌محاسبه نیز نشده باشد.

یکی از الگوریتم‌های بررسی‌شده در این بخش، BUC (محاسبات از پایین به بالای مکعب - BottomUp Cubing) است و برای انجام محاسبات مکعب‌های کم‌تراکم و آستانه‌ای کاربرد دارد.

الگوریتم BUC برای نخستین بار در [4] ارائه شده است. ایده اصلی BUC این است که کارایی ورودی/خروجی الگوریتم‌های PartitionedCube/MemoryCube با هم ترکیب شود، ولی برای به دست‌آوردن پشتیبانی کمینه بهتر، مانند الگوریتم Apriori [5] از هرس‌کردن استفاده می‌شود. BUC از الگوریتم‌های ارائه‌شده در [6] به‌ویژه PartitionedCube الهام گرفته است. در حقیقت BUC شبیه نسخه‌ای از الگوریتم PartitionedCube است که هرگز MemoryCube را فراخوانی نمی‌کند.



شکل 2: درخت پردازش الگوریتم BUC.

برای دست‌یابی به مزایای هرس‌نمودن، BUC عملیات خود را از پایین شبکه آغاز می‌کند (یعنی از کوچک‌ترین گروه‌بندی که دارای بیشترین مجموع‌سازی نیز هست) و کار خود را به سمت بالای شبکه، که دارای گروه‌بندی‌های بزرگ‌تر و با مجموع‌های کمتر است، ادامه می‌دهد. همه الگوریتم‌های ارائه‌شده پیش از BUC، محاسبات را در مسیر عکس انجام می‌دادند. در این نوع محاسبه از آنجا که گروه‌بندی‌های پدر برای محاسبه گروه‌بندی‌های فرزندان مورد استفاده قرار می‌گرفتند امکان اجتناب از محاسبه گروه‌بندی‌های پدر وجود نداشت. درخت پردازش BUC در شکل 2 نشان داده شده است.

در [7] الگوریتمی به نام CCUBE ارائه شده که در آن از تکنیک پارتیشن‌کردن (قسمت‌بندی) برای کم‌کردن میزان حافظه انبارشی مورد نیاز استفاده شده است.

فرض می‌کنیم که r رابطه‌ای باشد که بر روی شمای R تعریف شده است. دو نوع ویژگی داریم: 1) مجموعه Dim از ابعاد که تحلیل بر روی آنها انجام می‌شود، و 2) یک ویژگی قابل اندازه‌گیری M که برای تحلیل اندازه‌ها به‌کار می‌رود. X نیز مجموعه‌ای از ابعاد به‌صورت $\{A_1, A_2, ...\}$ است و $X \subseteq Dim$. فرض می‌کنیم که شرط محدودکننده $Cond$ و تابع مجموع‌کننده f در قالب پرس‌وجو توسط کاربر داده می‌شوند. با استفاده از تعاریف، مثال‌ها و قضایای زیر به بیان و توضیح روش پرداخته شده است. با الهام‌گرفتن از مفهوم افراز که در مورد پایگاه‌های داده استنتاجی به‌کار گرفته شده است، مفهوم افراز قابل اندازه‌گیری ابعادی متناظر با مجموعه‌ای از ابعاد ارائه شده است.

تعریف 1: تعریف کلاس‌های قابل اندازه‌گیری ابعادی: فرض کنید که r یک رابطه بر روی شمای R با تعریف زیر باشد $R=(RowId, Dim, M)$ و $X \subseteq Dim$ باشد. یک کلاس متعادل قابل اندازه‌گیری ابعادی برای یک چندگانه مانند $t \in r$ متناظر با X به‌صورت $[t]_X$ نمایش داده می‌شود و به شکل زیر تعریف می‌شود

$$[t]_X = \{(u[RowId], u[M]) \,/\, u \in r, \; u[X] = t[X]\}. \quad (1)$$

تعریف 2: قطعه قابل اندازه‌گیری ابعادی: فرض کنیم که r رابطه‌ای بر روی شمای R، X مجموعه‌ای از ابعاد، $X \subseteq Dim$ و $Cond$ یک شرط باشد. یک قطعه قابل اندازه‌گیری ابعادی متناظر با X رابطه r به‌صورت $\prod_X(r)$ نمایش داده می‌شود و به شکل زیر تعریف می‌شود

$$\prod_X(r) = \{[t]_X \mid= Cond \,/\, t \in r\}. \quad (2)$$

قضیه 1: ضرب قطعه‌های قابل اندازه‌گیری ابعادی تحت محدودیت شرط: فرض کنیم r یک رابطه و $\prod_X(r)$ و $\prod_Y(r)$ دو قطعه قابل اندازه‌گیری ابعادی به‌ترتیب بر اساس X و Y باشند. حاصل‌ضرب دو قطعه که به‌صورت $\prod_X(r) \bullet_c \prod_Y(r)$ نشان داده می‌شود به‌صورت زیر به‌دست می‌آید و برابر است با

$$\prod_{X \cup Y} : \prod_X(r) \bullet_c \prod_Y(r) = \{[t]_X \cap [t]_Y \mid= Cond \,/\, [t]_X \in \prod_X(r), [t]_Y \in \prod_Y(r)\} = \prod_{X \cup Y}(r) \quad (3)$$

الگوریتم CCUBE با به‌کارگیری قضایای بالا روشی برای انجام محاسبات مکعب تدارک می‌بیند که بر مبنای قطعه‌بندی است و از نظر زمانی بهتر از BUC عمل می‌کند. در حقیقت قضایای فوق این امکان را می‌دهند که مکعب قطعه‌بندی‌شده، قطعات مزبور مورد محاسبه قرار گرفته و نتایج برای به‌دست آمدن نتیجه نهایی برای مکعب کامل، ترکیب گردند. در [7]، زمان به‌عنوان معیار مقایسه دو الگوریتم به کار رفته و نتایج به‌دست آمده بر اساس این معیار مورد مقایسه و سنجش قرار گرفته‌اند. اگرچه با تغییر ساختار عملکرد BUC تا حدودی بهتر شده است اما ساختار ارائه‌شده در BUC روش مناسبی برای انجام محاسبات است که با اعمال تغییراتی می‌تواند به مراتب بهتر عمل کند. در این مقاله، الگوریتم Ex - Cube می‌کوشد با بهره‌گیری از مزایای CCUBE و اعمال ساختار پایین به بالای BUC، روش سریع‌تری برای محاسبات مکعب ارائه کند.

3- الگوریتم EX-CUBE

در این بخش الگوریتم ارائه‌شده در این مقاله را تشریح خواهیم نمود و در ارتباط با نحوه طراحی آن توضیح می دهیم. در این راه، در ابتدا تعاریف و مقدمات را در قالب نمادها ارائه می‌کنیم و در ارتباط با ساختار ابرگراف توضیح خواهیم داد و پس از آن الگوریتم ارائه‌شده را تشریح می‌نماییم.

3-1 تعاریف و نمادها

سلول $a = (a_1, a_2, ..., a_n, measure_a)$ در یک مکعب n بعدی، یک سلول m بعدی (که سلولی در یک مکعب‌واره m بعدی است) نامیده می‌شود اگر و تنها اگر دقیقاً m مقدار $(m \le n)$ در مجموعه $\{a_1, a_2, ..., a_n\}$ وجود داشته باشد که در گروه‌بندی و تشکیل سلول شرکت کرده‌اند و تمامی مقادیر آنها مورد نظر قرار ندارد. به‌عبارت دیگر m مقدار وجود دارند که به جای آنها ALL یا * قرار نگرفته است. اگر $m = n$ باشد، سلول مزبور را سلول پایه می‌نامند. سلول‌های غیرپایه مقادیر مجموع را در خود ذخیره می‌کنند بنابراین در برخی موارد با عنوان سلول‌های مجموع شناخته می‌شوند.

در مکعب داده n بعدی، سلول i بعدی $a = (a_1, a_2, ..., a_n, measure_a)$ را نیای یک سلول j بعدی مانند $b = (b_1, b_2, ..., b_n, measure_b)$ و سلول b را نواده سلول a می‌نامند اگر و تنها اگر $i < j$ باشد و به‌ازای هر m که $(1 \le m \le n)$ داشته باشیم $a_m = b_m$ به شرط آنکه a_m دارای مقدار باشد (* نباشد). در حالت خاصی که $j = i + 1$ باشد، سلول a را پدر b، و b را فرزند سلول a می‌نامند.

برای هر مکعب آستانه‌ای ICube، مکعب کاملی را که توسط پرس‌وجوی مشابه این مکعب آستانه‌ای (بدون شرط HAVING) ساخته می‌شود، مکعب پس‌زمینه ICube می‌نامیم و آن را با B(ICube) نمایش می‌دهیم. به‌عنوان مثالی از تعاریف عنوان‌شده جدول 2 را در نظر بگیرید. بر اساس داده‌های موجود در این جدول $(Jan, *, *, 1200, 2800)$ و $(*, Toronto, *, 800, 1200)$ سلول‌های یک‌بعدی و $(Jan, *, Edu, 600, 250)$ یک سلول دوبعدی است. سلول $(Jan, Toronto, Busi, 1500, 45)$ سه‌بعدی است. سلول‌های یک‌بعدی

جدول 2: جدول اطلاعات فروش (SALES - INFO).

Mon (ماه)	Day (روز)	City (شهر)	Cust-gr (مشتری)	Product (محصول)	Cost (هزینه)	Price (قیمت)
Jan	10	Tor	Edu	Printer	500	485
Jan	15	Tor	Hoshld	TV	800	1200
Jan	20	Tor	Edu	Camera	1160	1280
Feb	20	Mon	Busi	Laptop	1500	2500
Mar	4	Van	Edu	HD	540	520

$a=(Jan,*,*,1200,2800)$ و دوبعدی $b=(Jan,*,Edu,600,250)$ نیای سلول سه‌بعدی $c=(Jan,Toronto,Busi,1500,45)$ هستند و c نواده آنها محسوب می‌شود. همچنین b پدر c و c فرزند b است. مکعب پس‌زمینه Sales-Iceberg، مکعبی است که با پرس‌وجوی زیر ساخته می‌شود

CREATE TABLE Sales_Cube AS
SELECT Month, City, Customer-group,
AVG(Price), COUNT(*)
FROM Sales_Info
CUBEBY Month, City, Customer-group

یک مکعب آستانه‌ای مانند *ICube* را یکنوا می‌نامند اگر و تنها اگر به‌ازای هر سلول مانند c در $B(Icube)$، اگر محدودیت مشخص‌شده توسط *ICube* را (که در *HAVING* آمده است) ارضا نمی‌کند، همه نواده‌های c نیز این محدودیت را ارضا نکنند.

به‌عنوان مثال مکعب آستانه‌ای *Count-Iceberg* یکنوا است

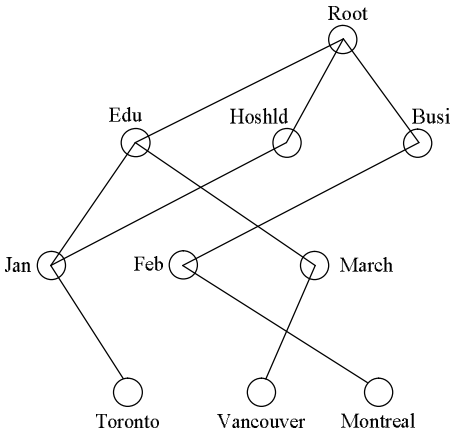
CREATE TABLE Count_Iceberg AS
SELECT Month, City, Customer-group,
COUNT(*)
FROM Sales_Info
CUBEBY Month, City, Customer-group
HAVING COUNT(*)>=50

در واقع اگر یک سلول مانند c در *Count-Iceberg* شرط موجود در *HAVING* را ارضا ننماید، یعنی تعداد رکوردهای آن کمتر از 50 باشد، آنگاه تعداد رکوردهایی که برای تشکیل هر یک از سلول‌های نواده c به‌کار می‌روند نیز کمتر از 50 خواهد بود و بنابراین هیچ یک از نواده‌های آن در قالب شرط پرس‌وجوی آستانه‌ای فوق نمی‌گنجند.

این در حالی است که اگر در این مکعب به جای شرط *HAVING*، عبارت *HAVING AVERAGE(Price)>=50* قرار گیرد، مکعب آستانه‌ای مزبور دیگر یکنوا نیست. زیرا حتی اگر میانگین قیمت همه کالاهای فروخته‌شده در ماه مارس کمتر از 800 دلار باشد یعنی $(March,*,*,600,1800)$، قیمت متوسط برای یک زیرمجموعه از آن که تنها شامل اجناس فروخته‌شده در ماه مارس و به مشتریان تاجر است یعنی $(March,*,*,1300,360)$ می‌تواند بیش از 800 دلار باشد و شرط *HAVING* را ارضا نماید.

3-2 ساختار ابرگراف Ex-Graph

در این بخش یک ساختار ابرگراف به‌نام Ex-Graph(EG) معرفی می‌کنیم و بر اساس ساختار آن الگوریتم کارآیی برای محاسبه مکعب‌ها ارائه خواهیم کرد که تکنیک‌های به‌کار رفته در BUC و CCUBE را برای بهینه‌سازی پردازش پرس‌وجوهای تحلیلی یکنوا به‌کار می‌گیرد. برای



شکل 3: تشکیل Ex-Graph برای Sales-Info.

آسان‌شدن انتقال مفاهیم مربوط به EG، طریقه ساخت آن را برای جدول 2 مرحله به مرحله نشان می‌دهیم. یک گراف EG برای چنین پایگاه داده‌ای طی مراحل زیر ساخته می‌شود:

1. در ابتدا گراف تنها دارای یک ریشه است که مقداری در آن قرار نمی‌گیرد. ابعاد را به‌ترتیب کاردینالیتی و به‌صورت صعودی مرتب می‌کنیم. در مورد این مثال ابتدا بعد مشتریان (که دارای کمترین کاردینالیتی است)، پس از آن بعد زمان (ماه‌های سال) و در نهایت بعد مکان (یعنی شهرها) قرار دارد.
2. یک جدول تیتر ساخته می‌شود که هر رکورد موجود در آن شامل اطلاعات مربوط به یک مقدار خاص از مجموعه مقادیر ویژگی‌های موجود است.
3. نخستین رکورد یعنی $t_1=(Edu,Jan,Toronto,485)$ با افزودن سه نود به گراف اضافه می‌شود. نودهای *Edu*، *Jan* و *Toronto* به‌ترتیب به گراف افزوده می‌شوند و نخستین شاخه را شکل می‌دهند و قیمت 485 برای به‌هنگام‌سازی مقادیر مربوط به *Edu*، *Jan* و *Toronto* در جدول تیتر مورد استفاده قرار می‌گیرند.
4. به‌صورت مشابه $t_2=(Hoshld,Jan,Toronto,1200)$ نیز به گراف اضافه می‌شود. چون نودهای *Jan* و *Toronto* موجود هستند تنها نود *Hoshld* را تشکیل می‌دهیم و ارتباط‌ها را برقرار می‌نماییم.
5. چون $t_3=(Edu,Jan,Toronto,1200)$ همان مقادیر ویژگی‌های t_1 را دارد مسیر خود را با t_3 به اشتراک می‌گذارد و تنها مقادیر متناظر در جدول تیتر به‌هنگام می‌شوند و نودی افزوده نمی‌گردد.
6. سایر نودها نیز به‌صورت مشابه به گراف افزوده می‌شوند و گراف مانند شکل 3 تشکیل می‌شود.

3-3 ذخیره‌سازی دوبعدی Ex-Graph

در گراف EG به‌عنوان وزن هر یال از گراف، زوج مرتب (x, y) نگهداری می‌شود. با افزودن یک یال به این گراف، مقدار اولیه‌ی x در زوج مرتب آن یال برابر یک می‌گردد و y نیز برابر price متناظر با رکوردی می‌شود که سبب افزوده‌شدن این یال گشته است. اگر رکوردی را مشاهده کنیم که مسیر متناظر با آن (یا بخشی از مسیر آن) پیش از این ایجاد شده است، وزن یال‌هایی را که در مسیر مزبور قرار دارند به این صورت تغییر می‌دهیم که به x آنها یک واحد می‌افزاییم و مقدار y را با مقدار price رکورد مزبور جمع می‌کنیم.

در این حالت به‌سادگی می‌توان پاسخ پرس‌وجوهایی را که به دنبال *SUM* یا *COUNT* برای معیارهای قابل اندازه‌گیری رکوردها هستند، یافت. برای مثال برای پاسخ‌گویی به پرس‌وجوی

```
1. EG=ConstructExGraph (input, numDim);
2. Procedure ExCube (input, EG, dim)
3.   for d=dim; d<numDims; d++ do
4.     let C=cardinality[d];
5.     Partition (input, d, C, EG)
6.     let k=0;
7.     for i=0; i<C; i++ do // for each partition
8.       let s=0;
9.       for (j=1; j<d; j++) s=s+cardinality[j]
10.        let c=EG[Root][s+i+1].count;
11.          // EG[Root][s+i+1].sum
12.        if c>=minsup then
13.          outputRec.dim[d]=EG[Root][d];
14.          BottomUpCube (input [k…k+c], d+1);
15.        end if
16.        k+=c
17.      end for
18.      let m=0;
19.      for (j=1; j<=d; j++) m=m+EG[Root][j];
20.      outputRec.dim[d]=m;
21.    end for
```

شکل 4: شبه‌کد الگوریتم Ex-Cube.

```
CREATE TABLE    T1 AS
SELECT    Customer-group, COUNT(*)
FROM    Sales_Info
GROUPBY    Customer-group
HAVING    COUNT(*)>=50
```

کافی است مقادیر موجود در سطر R از ماتریس همسایگی (همجواری) گراف را به تفکیک چاپ نماییم. علاوه بر آن برای پاسخ‌گویی به پرس‌وجوهایی که مقادیر سلول‌های مکعب کامل را محاسبه می‌کنند نیز می‌توان از همین روش استفاده کرد

```
CREATE TABLE    T1 AS
SELECT    Customer-group, COUNT(*)
FROM    Sales_Info
GROUPBY    Customer-group
```

برای محاسبه پرس‌وجوهایی که عمل گروه‌بندی را بر روی دو ویژگی انجام می‌دهند، کافی است خانه مشترک میان دو ویژگی مزبور را نمایش دهیم. مثلاً برای پاسخ‌گویی به پرس‌وجوی

```
CREATE TABLE    T2 AS
SELECT    Customer-group, month, COUNT(*)
FROM    Sales_Info
GROUPBY    Customer-group, month
```

کافی است داشته باشیم

$$\forall x \in Cost-Group, \forall y \in Month \rightarrow \text{print}\,(EG[x][y])$$

برای آنکه قادر باشیم با استفاده از ماتریس همسایگی گراف فوق عمل گروه‌بندی را بر روی هر دو بعد مورد نظر از میان ابعاد موجود انجام دهیم، لازم است گراف را کامل فرض نماییم. این موضوع به این صورت محقق می‌شود که وقتی به‌واسطه ارتباط Busi و Jan در یک رکورد، وزن یال متصل‌کننده آنها را به‌هنگام‌سازی می‌کنیم و به خاطر ارتباط Jan و Vanc در همان رکورد، با همان اطلاعات یال متصل‌کننده آن دو را نیز به‌هنگام می‌کنیم، وزن یال متصل‌کننده Busi و Vanc را نیز تغییر دهیم. یعنی وقتی اطلاعات یک رکورد را به گراف خود می‌افزاییم، تمامی زوج ویژگی‌های متناظر با آن رکورد را به‌هنگام کنیم.

به این ترتیب در صورت n بعد در مکعب، در هر به‌هنگام‌سازی به انتخاب 2 از n یعنی $n(n-1)/2$ تغییر یا ایجاد وزن بر روی یال‌های گراف نیازمندیم. باید توجه شود که در این حالت، دیگر مجموع عناصر یک سطر، بیانگر بخشی از پاسخ گروه‌بندی (آنچنان که پیش از این اشاره شد) نیست ولی این موضوع خللی در روند عملیاتی Ex-Cube ایجاد نمی‌کند و همچنان مسیر مستقیم میان *Root* (سطر R) و هر یک از ویژگی‌ها در گراف کامل تشکیل‌شده، نشان‌دهنده حاصل مجموع‌سازی بر روی بعد آن ویژگی و به‌ازای آن ویژگی خواهد بود.

بر اساس توضیحات داده‌شده، روش Ex-Cube با به‌وجود آوردن گرافی که شامل همه اطلاعات موجود در مکعب ما باشد، علاوه بر حفظ اطلاعات مکعب، با توصیف گراف در قالب ماتریس همجواری سبب نگاشت اطلاعات از فضای چندبعدی مکعب به یک فضای دوبعدی در ماتریس همجواری شده است. به‌علاوه با فراهم‌شدن امکان محاسبات از پایین به بالا، الگوریتم باعث جلوگیری از هم‌پوشانی محاسباتی می‌شود و در محاسبات خود نودهایی را که قبلاً محاسبه شده‌اند، هرس می‌نماید.

الگوریتم کار در شکل 6 توضیح داده شده است. در خط اول با فراخوانی روال ConstructExGraph، به‌صورت توضیح داده شده، ابرگراف را می‌سازیم و آن را به‌عنوان پارامتر، به الگوریتم اصلی ExCube می‌دهیم. الگوریتم برای هر یک از ابعاد اعمال زیر را تکرار می‌کند: کاردینالیتی بعد را حساب می‌کند و در خط 5 با استفاده از روال Partition EG را بر اساس کاردینالیتی خرد می‌نماید. حلقه خطوط 7 تا 17 هر یک از قطعات را (که بر اساس روش CCUBE قطعه‌بندی شده‌اند) به روال BottomUpCube می‌سپارد تا آن را از پایین به بالا جستجو نماید. در پایان نتایج به‌دست آمده برای ترکیب نهایی ذخیره می‌شوند.

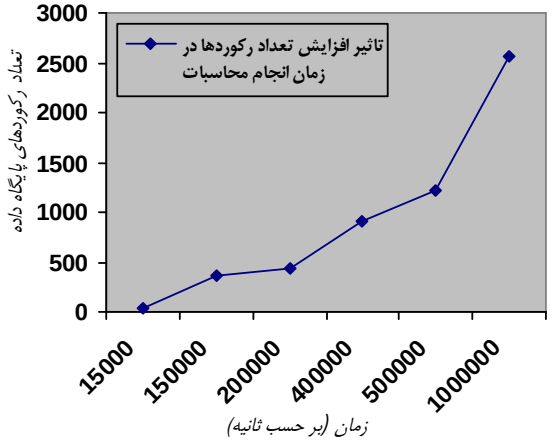
4-3 فضای لازم برای ذخیره‌سازی Ex-Graph

فرض کنید مکعبی دارای 10 بعد باشد و کاردینالیتی هر بعد را نیز 10 در نظر بگیرید. در این صورت فضای مورد نیاز برای آنکه مکعب مزبور را نگهداری کنیم برابر است با 10^{10}. اگر همین مکعب را به فرم ماتریس همسایگی Ex-Graph نگهداری کنیم در آن صورت فضای مورد نیاز گراف برابر است با $(10\times10)^2 = 100^2 = 10^4$.

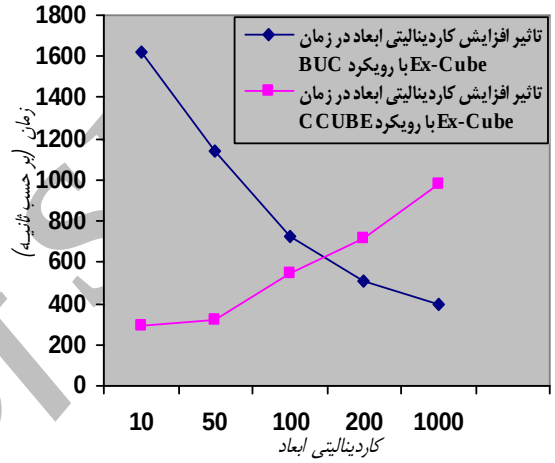
در حقیقت اگر n بعد داشته باشیم که به‌ترتیب با نام‌های A_1 تا A_n مشخص شوند و کاردینالیتی بعد A_i برابر C_i باشد، فضای لازم برای نگهداری یک مکعب کامل برابر با $\prod_{i=1}^{n} C_i$ است، در حالی که فضای لازم برای نگهداری و ذخیره‌سازی Ex-Graph برابر $(\sum_{i=1}^{n} C_i)^2$ است.

به‌سادگی می‌توان مشاهده کرد که Ex-Graph تأثیر شگفت‌آوری بر روی حافظه مورد نیاز مکعب می‌گذارد و آن را به میزان قابل ملاحظه‌ای کاهش می‌دهد. علاوه بر آن مجموع‌سازی در زمانی که گروه‌بندی بر روی یک یا دو بعد صورت می‌گیرد، کارآیی زمانی یکسانی در هر دو حالت دارد.

اگرچه پیش‌محاسبه و ذخیره‌سازی مکعب در قالب گراف، سبب سرعت‌بخشیدن به روند محاسبات می‌گردد و بخشی از عملیات محاسبه، که شامل حداقل یک بار پویش پایگاه داده و تشکیل گراف است، از پیش



شکل 5: افزایش زمان محاسبات بر اثر زیادشدن تعداد رکوردها.

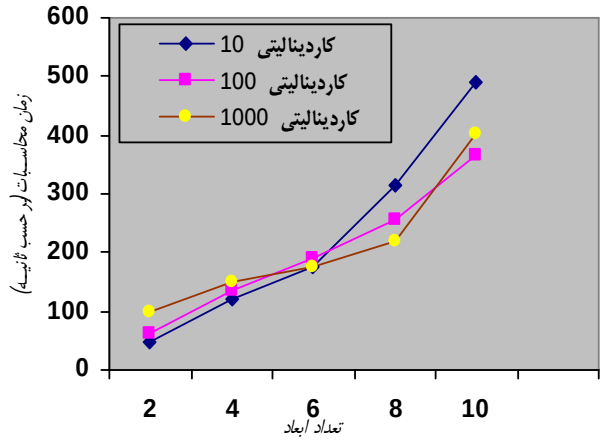


شکل 6: تغییر زمان محاسبات بر اثر زیادشدن کاردینالیتی ابعاد.

انجام و حاصل آن ذخیره می‌شود، ولی یکی از برتری‌های روش Ex-Cube در این است که اگر به‌طور کلی هیچ بخشی از مکعب پیش‌محاسبه نشده باشد نیز اشکالی در روند الگوریتم مزبور ایجاد نمی‌شود. Ex-Cube می‌تواند در ابتدای کار خود یک مرحله برای ساخت Ex - Graph در نظر بگیرد و پس از آن بر روی گراف تشکیل‌شده عملیات خود را انجام دهد.

4- شبیه‌سازی و نتایج به‌دست آمده

در این بخش به ارائه نتایج حاصل از پیاده‌سازی الگوریتم Ex-Cube می‌پردازیم. این نتایج بر اساس پیاده‌سازی الگوریتم مزبور بر روی سیستم پنتیوم با کلاک پردازنده 2000 مگاهرتز و با ظرفیت حافظه 512 مگابایت به‌دست آمده است. اجرای الگوریتم‌های پیاده‌سازی‌شده روی سیستم‌های ضعیف‌تر نیز از برتری‌های Ex-Cube به‌همین صورت ارائه‌شده حکایت دارد با این تفاوت که در این سیستم‌ها، کاهش قدرت و امکانات کامپیوتر به کاهش کارآیی در عملکرد الگوریتم‌ها منجر شده است. توجه به این نکته حائز اهمیت است که پارامترهایی که در اینجا به‌عنوان مشخصات بستر پیاده‌سازی ذکر شده‌اند، مواردی هستند که تغییر آنها به عوض‌شدن نتایج می‌انجامد و بر روی معیارهای ارزیابی تأثیر مستقیم دارند. علاوه بر موارد فوق، زبان برنامه‌نویسی نیز یکی از اجزای تشکیل‌دهنده بستر پیاده‌سازی است. اگرچه شبیه‌سازی الگوریتم‌های ارائه‌شده در این مقاله در نگارش 8 زبان برنامه‌نویسی Delphi انجام شده است ولی تغییر زبان برنامه‌نویسی، تغییر قابل ملاحظه‌ای در مقادیر اندازه‌گیری‌شده برای پارامترهای ارزیابی ایجاد نخواهد کرد.



شکل 7: تغییر زمان محاسبات بر اثر افزایش تعداد ابعاد.

4-1 تأثیر افزایش حجم پایگاه داده بر روی شمای ثابت بر سرعت محاسبات

در مورد پایگاه داده مشاهده می‌شود که اگر تعداد رکورد موجود در یک پایگاه داده افزایش یابد (بدون آنکه تغییری در شمای ویژگی‌های پایگاه داده روی دهد)، زمان انجام محاسبات افزایش می‌یابد که این موضوع روندی طبیعی و کاملاً قابل انتظار است. روند افزایش زمان محاسبات در شکل 5 ترسیم شده است.

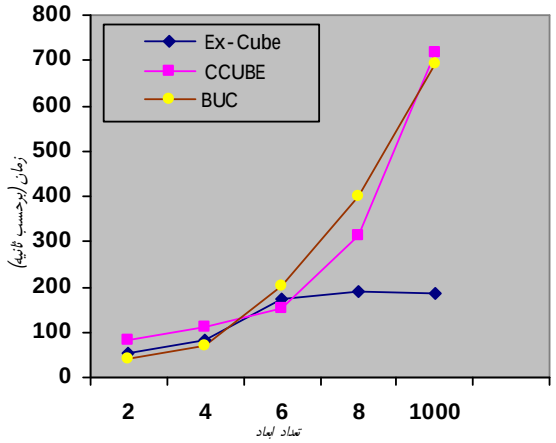
4-2 تغییرات زمان با افزایش کاردینالیتی ابعاد

در شکل 6 نحوه اثر افزایش کاردینالیتی ابعاد، بر روی عملکرد دو رویکرد BUC و CCUBE از الگوریتم Ex-Cube نشان داده شده است. زمانی که کاردینالیتی ابعاد افزایش می‌یابد مقادیر مختلفی که ویژگی‌های یک بعد می‌توانند اختیار کنند زیاد می‌شود. این موضوع سبب می‌گردد که در حجم ثابت تعداد رکوردها، پراکندگی مقادیر مختلف کاهش یابد و در نتیجه مکعب کم‌تراکم‌تر گردد. نسخه‌ای از Ex-Cube که با الهام از رویکرد BUC پیاده‌سازی شده است می‌تواند عملکرد بسیار مناسبی در ارتباط با چنین مکعب‌هایی از خود نشان دهد چرا که همان‌طور که در توصیف BUC مشاهده کردیم، ساختار این الگوریتم چنان است که برای مکعب‌های کم‌تراکم عملکرد بسیار مناسبی از خود نشان می‌دهد. با کاهش کاردینالیتی ابعاد مکعب به‌سوی چگال‌شدن پیش می‌رود که نتیجه آن کاهش کارآیی نسخه BUC از Ex-Cube است. با چگال‌شدن مکعب می‌توان از نسخه دوم Ex-Cube که با الهام از CCUBE طراحی شده استفاده کرد زیرا چنانکه مشاهده کردیم با افزایش چگالی مکعب، CCUBE عملکرد بسیار موفقی دارد. در پیاده‌سازی Ex-Cube می‌توانیم حدی از چگالی را برای مکعب مورد محاسبه تعیین کنیم که در آن حد، رویکرد محاسبه در Ex-Cube تغییر نماید.

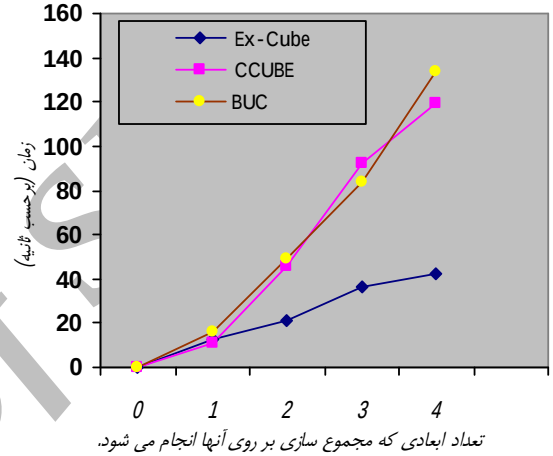
4-3 افزایش تعداد ابعاد

آنچه تاکنون ارائه‌شده حاصل انجام آزمایشات مزبور بر روی مکعبی 3بعدی است. با افزایش تعداد ابعاد، به‌صورت طبیعی زمان پردازش افزایش خواهد یافت. تغییر عملکرد Ex-Cube بر اثر افزایش تعداد ابعاد در شکل 7 نشان داده شده است.

در این شکل کاردینالیتی ابعاد متفاوت فرض شده است. بر اساس هر یک از کاردینالیتی‌های مزبور نتایجی به‌دست آمده که در شکل دیده می‌شود.



شکل 8: مقایسه عملکرد BUC، CCUBE و Ex-Cube.



شکل 9: مقایسه عملکرد سه الگوریتم بر اساس داده واقعی.

4-4 مقایسه الگوریتم‌ها

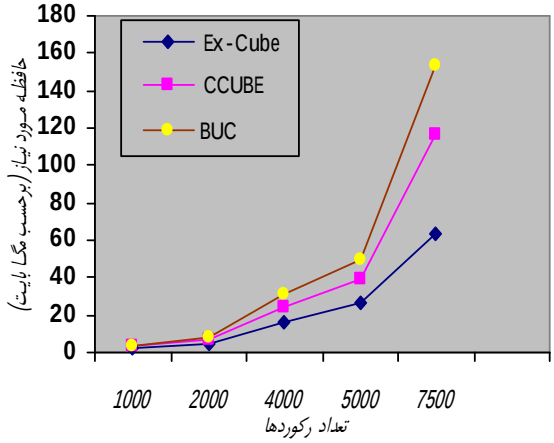
در نمودار شکل 8 مقایسه عملکرد الگوریتم‌های BUC، CCUBE و Ex-Cube نشان داده شده است. کاردینالیتی همه ابعاد در این مقایسه 50 قرار داده شده است.

5-4 مقایسه زمان محاسبات الگوریتم‌ها بر اساس داده‌های واقعی

برای آنکه اثر انجام الگوریتم جدید مشخص‌تر باشد، علاوه بر داده‌های شبیه‌سازی شده، بر روی داده‌های بخش آموزش دانشگاه آزاد، نیز الگوریتم‌ها را اعمال و نتایج را بررسی و مقایسه نمودیم. مکعب ساخته‌شده بر روی این داده‌ها دارای پنج بعد دانشجو، سال و ترم ورود، شهر، مقطع، رشته، و ویژگی‌های قابل اندازه‌گیری معدل و تعداد واحدهای گذرانده است. کاردینالیتی شهر 87، کاردینالیتی رشته 23، کاردینالیتی مقطع 3 (کاردانی، کارشناسی و کارشناسی ارشد) و کاردینالیتی سال و ترم ورود 20 است (داده‌های 10 سال مورد بررسی قرار گرفته که هر سال دارای دو ترم برای پذیرش دانشجو است). بنابراین کاردینالیتی هر یک از ابعاد به‌صورت متوسط برابر 33 است. تعداد رکوردها برابر 7500 است. نمودار شکل 9، عملکرد الگوریتم‌ها از دیدگاه زمان اجرا را مقایسه می‌کند.

6-4 مقایسه حافظه مورد نیاز الگوریتم‌ها بر اساس داده‌های واقعی

علاوه بر زمان اجرا، روش ارائه‌شده سبب صرفه‌جویی در میزان حافظه



شکل 10: مقایسه فضای حافظه مورد نیاز هر یک از سه الگوریتم بر اساس داده واقعی.

مصرفی نیز می‌گردد. مقایسه میزان حافظه مورد نیاز هر یک از الگوریتم‌ها محاسبه مکعب داده‌های دانشجویی بخش 4-5، در شکل 10 نشان داده شده است. آنچه در این شکل مشاهده می‌شود، به‌آسانی و با استفاده از میزان فضای مصرفی هر یک از الگوریتم‌ها قابل محاسبه است. برای به‌دست آوردن فضای مورد نیاز هر یک از الگوریتم‌ها، میزان حافظه مورد نیاز الگوریتم را در اجرا در نظر گرفته‌ایم. تحلیلی مشابه آنچه در بخش 4-3 مشاهده کردیم نیز می‌تواند این مقایسه را از دیدگاه تحلیلی مورد بررسی قرار دهد.

5- نتیجه‌گیری

پاسخ‌گویی کارا به پرس‌وجوهای OLAP نیازمند پیش‌محاسبه نتایج پرس‌وجوها (در قالب مکعب داده) و ذخیره‌سازی آنها می‌باشد. با توجه به حجم عظیم داده‌ها، محاسبه مکعب داده از نظر زمان اجرا دارای هزینه بالایی است. از سوی دیگر نگهداری نتایج محاسبات در حافظه نیز به هدررفتن فضای زیادی از حافظه و دیسک می‌انجامد.

به‌دلیل ارتباط معکوس این دو موضوع با یکدیگر، هر یک از آنها به‌صورت جداگانه مورد بررسی قرار می‌گیرند. از یک سو الگوریتم‌های فراوانی برای محاسبه کارآمد مکعب‌های داده طراحی و پیاده‌سازی شده‌اند و از سوی دیگر روش‌هایی برای کمترکردن حافظه مورد استفاده ارائه‌شده و به مرحله اجرا رسیده‌اند. این دسته از الگوریتم‌ها یا بر پایه تکنیک‌های فیزیکی بنا شده‌اند یا از طریق انتخاب و پیش‌محاسبه بخشی از نتایج (و نه همه آنها) سبب صرفه‌جویی در حافظه مصرفی شده‌اند (پیش‌محاسبه جزئی) و یا اینکه متکی بر مدل تقریب هستند.

مراجع

[1] R. Kimball, *The Data Warehouse Toolkit*, John Wiley and Sons, Inc, 1996.

[2] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh, "Datacube: a relational aggregation operator generalizing group-by, cross-tab, and sub-totals," in *Proc. of the IEEE ICDE*, pp. 152-159, 1996.

[3] S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi, *On the Computation of Multidimensional Aggregates*, VLDB'1996.

[4] K. Beyer and R. Ramakrishnan, "Bottom-up computation of sparse and iceberg cubes", in *Proc. SIGMOD Conf.*, pp. 359-370, 1999.

[5] R. Agrawal and R. Srikant, Fast Algorithms for Mining Association Rules, VLDB'1994.

[6] K. A. Ross and D. Srivastava, "Fast computation of sparse datacubes," in *Proc. of the 23rd VLDB Conf.*, pp. 116-125, Athens, Greece, 1997.

[7] M. Laporte, N. Novelli, R. Cicchetti, and L. Lakhal, *Computation Full and Iceberg DataCubes Using Partitions*, 2002.

محمدکریم سهرابی مدرک کارشناسی مهندسی کامپیوتر- نرم‌افزار خود را از دانشگاه فردوسی مشهد در سال 1381 و مدرک کارشناسی ارشد مهندسی کامپیوتر- نرم‌افزار خود را از دانشگاه صنعتی امیرکبیر در سال 1383 دریافت نموده است و از سال 1384 تاکنون مشغول گذراندن دوره دکترای مهندسی کامپیوتر در دانشگاه صنعتی امیرکبیر است. نام‌برده از سال 1384 به‌عنوان عضو هیئت علمی دانشگاه آزاد سمنان استخدام و مشغول به کار است. ایشان در سال‌های 1383 و 1384 به‌ترتیب کتاب‌های "نظریه زبان‌ها و ماشین‌ها" و "پایگاه داده" را تألیف نمودند. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: پایگاه‌های داده تحلیلی، داده‌کاوی و الگوکاوی، سیستم‌های تصمیم‌یار، یادگیری تقویتی و نظریه محاسبات.

احمد عبدالله‌زاده بارفروش تحصیلات خود را در مقطع دکترای علوم کامپیوتر در دانشگاه بریستول انگلستان به پایان رسانده است و هم‌اکنون استاد دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر می‌باشد. نام‌برده از سال 1379 تا 1381 به‌عنوان استاد مدعو در دانشگاه‌های مریلند آمریکا و ارسی پاریس مشغول به کار بوده است. دکتر عبدالله‌زاده کتاب "مقدمه‌ای بر هوش مصنوعی توزیع‌شده" را در سال 1386 تألیف نمود. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از پایگاه‌های داده تحلیلی، داده‌کاوی، مهندسی نرم‌افزار، پردازش زبان‌های طبیعی، سیستم‌های تصمیم‌یار و واژگان‌شناختی.