تکه تکه شدن و تخصیص سلسله مراتبی همزمان عمودی با استفاده از الگوریتم اصلاح شده انرژی پیوندی در پایگاه داده های توزیع شده

چکیده

طراحی یک سیستم پایگاه داده توزیع شده موثر (DDBS) به عنوان یکی از چالش برانگیز ترین مشکلات در نظر گرفته می شود به دلیل عوامل متعدد وابسته که بر روی عملکرد آن تاثیر گذار هستند . تخصیص و تکه تکه شدن دو فرآیند هستند که کارآیی و صحت آنها می تواند عملکرد DDBS را تحت تاثیر قرار دهد . بنابراین ، تکه تکه شدن کارآمد داده ها و تخصیص قطعات در سراسر قسمت های شبکه به عنوان یک حوزه مهم پژوهشی در طراحی پایگاه داده های توزیع شده به شمار می آید . در این مقاله ما یک روشی را ارائه می دهیم که در آن به طور همزمان به تکه تکه شدن داده ها و تخصیص قطعات مناسب در سراسر شبکه خواهیم پرداخت . الگوریتم انرژی پیوندی (BEA) با اندازه وابستگی بهتری برای بهبود خوشه های تولید شده از این ویژگی ها مورد استفاده قرار می گیرد . این الگوریتم به طور همزمان خوشه هایی از این ویژگی ها را تولید می کند، هزینه تخصیص هر خوشه به هر کدام از این محل ها را مورد محاسبه قرار می دهد و هر کدام از این خوشه ها را به مناسب ترین محل تخصیص می دهد.

1. مقدمه

پایگاه داده های توزیع شده سبب کاهش هزینه و افزایش کارایی و در دسترس بودن می شوند ، اما طراحی سیستم های مدیریتی پایگاه داده های توزیع شده (DDBMS) پیچیده است . برای امکان پذیر شدن این فرآیند ، آن را به دو مرحله تقسیم می کنیم : تکه تکه شدن و تخصیص . در مرحله تکه تکه شدن سعی می کنیم که داده ها را به تکه

هایی تقسیم کنیم ، که باید به محل هایی در طول شبکه در مرحله تخصیص اختصاص داده شود. فرآیند تکه تکه شدن به دو دسته تقسیم می شود : تکه تکه شدن عمودی و تکه تکه شدن افقی . تکه تکه شدن عمودی (VF) از لحاظ پارتیشن بندی رابطه R در مجموعه های متلاشی شده(disjoint) از روابط کوچکتر است در حالیکه تکه تکه شدن افقی (HF) پارتیشن بندی مجموعه R به چندتایی های متلاشی شده است . مشکل تخصیص شامل پیدا کردن توزیع بهینه از تکه تکه شدن مجموعه F در محل مجموعه S است . چهار نوع استراتژی قابل اجرا برای تخصیص داده در رابطه با یک پایگاه داده توزیع شده وجود دارد : متمرکز شده ، تکه تکه شده (پارتیشن)، تکرار کامل و تکرار جزئی (انتخابی) [10]. زمانی که داده ها اختصاص داده شده است ، ممکن است از آنها کپی گرفته شود یا اینکه کپی گرفته نشود . برای همین ، تخصیص قطعه می تواند به صورت فاقد افزونه و با افزونه باشد . در طی طرح تخصیص فاقد افزونه ، دقیقا یک کپی از هر قطعه در سراسر تمامی سایت ها وجود دارد ، در حالیکه در طرح تخصیص افزونه ، بیشتر از یک کپی از هر قطعه در سراسر تمامی سایت ها وجود خواهد داشت [12] . در این کار ، ما تکه تکه شدن را با تکرار جزئی برخی از خوشه های ویژگی ها را انجام می دهیم.

تخصیص و تکه تکه شدن وابسته هستند و تخصیص موثر قطعات داده نیاز به در نظرگیری محدودیت های تخصیص در فرآیند تکه تکه شدن دارد ، اما در اکثر کارهای گذشته این دو مرحله جدا شده است .

دو روش کلی در حل مسائل پارتشین بندی وجود دارد . یکی پیدا کردن راه حلی کارآمد با در نظر گیری برخی از محدودیت ها است . هوفر [13] ظرفیت ذخیره سازی و محدودیت های هزینه های بازیابی را به عنوان عوامل اصلی در نظر گرفته است . هر کدام از این عوامل بر اساس میزان اثر آنها وزن شده است . هدف ما به حداقل رساندن هزینه های کلی بوده است . این وزن ها با استفاده از روش برنامه ریزی خطی مورد محاسبه قرار گرفته است تا مجموع وزن ها برابر با 1 شود:

$$min(c_1 * storage\ cost + c_2 * retrieval\ cost) \qquad (1)$$

یک مثال خوب دیگر از اولین مجموعه روش ها در شکولنیک [21] ارائه شده است . این روش سعی می کند که خوشه ای از سوابق را در داخل ساختار سلسله مراتبی سیستم مدیریتی اطلاعاتی (IMS) داشته باشد . درخت سلسله مراتبی

تولید شده از لحاظ تعداد گره ها خطی است . گروه بندی اکتشافی با استفاده از روش ارائه شده به وسیله همر و نیامیر [3] صورت گرفته است . این قضیه با تخصیص ویژگی ها به موقعیت های مختلفی آغاز می شود . تمامی نوع های احتمالی از گروه بندی در نظر گرفته می شود و نوعی که نشان دهنده بهترین بهبود در طی نامزد های گروه بندی فعلی است به عنوان نامزد جدید مطرح می شود. گروه بندی و تجدید گروه بندی تا زمانی تکرار می شود که دیگر هیچ پیشرفت بیشتری را انتظار نداشته باشیم . اصلی ترین مشکل در این بین ، جهت حرکت است ، که دارای اثری غالب در کارآیی الگوریتم دارد . یکی دیگر از روش های اکتشافی توسط ما و همکارانش[5] ارائه شده است ، که از یک مدل هزینه و هدف به صورت جهانی و برای به حداقل رساندن هزینه ها استفاده می شود . هدف اصلی ایت است که قطعه بندی بر اساس بهره وری از رایج ترین کوئری ها است . در رفرنس [14] هافر و سورانس خوشه هایی از ویژگی های مشابه را با استفاده از اندازه گیری موثر جفت ویژگی هایی که در رابطه با الگوریتم انرژی پیوندی (BEA) وجود دارد انجام داده اند. یکی از اصلی ترین نقاط ضعف تعداد ویژگی ها در خوشه هایی است که قابل تصمیم گیری نیستند ، و از آنجایی که تنها شباهت های ویژگی هایی دو به دو در نظر گرفته می شود ، برای تعداد بزرگتری از ویژگی ها لازم است . تکه تکه شدن عمودی نیز می تواند در بیشتر از یک فاز انجام شود . این روش توسط نواسه و همکارانش در [23] ارائه شده است . در روش جداسازی دو فازی قطعات به تکه هایی که با یکدیگر تداخل دارند و به قطعاتی که با یکدیگر تداخل ندارند تقسیم می شود . فاز اول بر اساس تابع هدف تجربی است و سپس اجرای آن سبب بهینه سازی هزینه ها با ترکیب دانش از یک محیط خاص نرم افزاری در فاز دوم می شود . این روش توسط لاتیفول و شهیدلو [6] ارائه شده است و یک روشی برای طراحی پایگاه داده های توزیع داده شی است که شامل یک فاز تجزیه و تحلیل برای نشان دادن مناسب ترین روش تکه تکه شدن است ، یک الگوریتم تکه تکه شدن افقی کلاس ، و یک الگوریتم تکه تکه شدن کلاس عمودی است . فاز تجزیه و تحلیل مسئول انتخاب بین تکنیک های پارتیشن بندی افقی و عمودی است ، یا حتی ترکیب هر دوی این تکنیک ها ، به منظور کمک به طراحان توزیع در مرحله تکه تکه شدن پایگاه داده های شی است . بایائو و همکارانش [8] یک روش سه مرحله ای برای طراحی پایگاه داده های توزیع شده شامل فاز تجزیه و تحلیل ، فاز الگوریتم تکه تکه شدن افقی و فاز تکه تکه شدن عمودی کلاس است . این روش توسط ابولیامان در [7]

انجام گرفته است که به طور تجربی نشان می دهد که انتقال یک ویژگی که به طور آزادانه در یک پارتیشن قرار دارد سبب بهبود آمار موفقیت ویژگی در پارتیشن می شود.

یک روشی برای تکه تکه شدن افقی هماهنگ و تخصیص توسط عبدالله [4] ارائه شده است . این روش یک مدل هزینه سلسله مراتبی را برای یافتن قطعه و تخصیص بهینه را ارائه می دهند . تکه تکه شدن بر اساس یک مجموعه ای از حالت های ساده است ، و تخصیص بهینه سبب به حداقل رسیدن تابع هزینه می شود . یک روش پارتیشن بندی عمودی سازگار توسط جین و می یانگ [15] ارائه شده است . این مقاله به بررسی پارتیشن بندی عمودی باینری (BVP) [18] می پردازد و نتایج آن را با روش پارتیشن بندی عمودی سازگار (AVP) مقایسه می کند که در آن از یک روش سلسله مراتبی برای روش قطعه قطعه شدن استفاده می شود ، یک درختی از پارتیشن را ایجاد می کند و در نهایت بهترین نتایج آن را انتخاب می کند . یک روش سلسله مراتبی در کار آدرین رونسنو [1] پیاده سازی شده است . در آن روش تدوین تابع هدف اعمال می شود ، با نام ارزیاب پارتیشن [2] ، پیش از اینکه الگورتم های (اکتشافی) را برای مشکلات پارتیشن بندی توسعه دهیم . این روش سبب می شود که ما قادر به مطالعه خواص الگورتیم با توجه به تابع هدف توافق شده باشیم ، و همچنین در مقایسه با الگوریتم های مختلف برای خوبی استفاده از معیار یکسان برای تکه تکه شدن عمودی پایگاه داده های توزیع شده است . یک الگوریتم ابتکاری جدید بر اساس یک تکنیک تجزیه به وسیله محمود و رویردن [16] توسعه داده شده است که تا حد زیادی سبب کاهش مشکل پیچیدگی محاسباتی تخصیص فایل و تخصیص ظرفیت در یک توپولوژی ثابت از شبکه توزیع شده می شود .

جدول 1 – نمادهای مدل

ماتریس ویژگی های وابستگی	AFF
ماتریس دسترسی به کوئری	QA
ماتریس میل خوشه	CA
ماتریس فاصله	DM
ماتریس استفاده از ویژگی ها	AU
هزینه کلی ذخیره سازی	TSC

حجم داده های تخصیص داد شده که مشخصات اندازه های آنها اندازه گیری شده است	V
هزینه ذخیره سازی قطعه i در سایت j	SC_{ij}
وابستگی ویژگی های A_i , A_j	$aff(A_i,A_j)$
تناوب دسترسی به کوئری k بر روی محل i	$Freq_1(q_k)$
دسترسی به ازای هر اجرای کوئری k بر روی محلi	$Acc_1(q_k)$
اندازه گیری تشابه بین A_i و A_j	S_{ij}
حداقل دسترسی به کوئری	MQA
هزینه ذخیره سازی	SC
تکرار خوشه ورودی (که تکرار بعدی را تغذیه می کند)	IIC
خوشه برگ	LC

اگرچه استفاده از یک طرح تکرار جزئی سبب افزایش بهره وری پایگاه داده می شود ، اما این مزیت همراه با هزینه است. این هزینه ، که به طور بالقوه بالا می باشد ، شامل کل هزینه ذخیره سازی ، هزینه پردازش محلی ، و هزینه های ارتباطاتی [19] است . برخی از روش های تکه تکه شدن همراه با بهینه سازی کوئری ، بهینه سازی توزیع ، و بهینه سازی پیوستن است که توسط حکیم زاده و هارون ربابا در [9] پوشش داده شده است . در اینجا ما ارتباطات و هزینه های پردازش های محلی را در ترکیب با دسترسی به کوئری را به حساب می آوریم و سپس هزینه ذخیره سازی کل را به طور جداگانه محاسبه می کنیم .

تکه تکه شدن و تخصیص معمولا به طور جداگانه ای صورت می پذیرد در حالیکه این دو در مرحله طراحی DBMS توزیع شده به طور نزدیکی به یکدیگر مرتبط هستند . دلیل جداسازی این طراحی توزیع شده به دو قسمت به این علت است که بهتر بتوان با پیچیدگی مشکلات مقابله کرد [17].

در اینجا ما یم روش برای VF را ارائه می کنیم ، که بر روی BEA سلسله مراتبی با اندازه گیری شباهت اصلاح شده و به طور همزمان با تخصیص قطعات به مناسب ترین سایت رخ می دهد . علائم این مدل در جدول 1 لیست شده است.

بقیه این مقاله به شرح زیر است. روش ها و عوامل مختلف تاثیر گذار در قسمت 2 مورد بحث قرار گرفته اند . الگوریتم با ذکر جزئیات در بخش 3 توضیح داده شده است . بخش 4 نتایج مقایسه ای از اعمال BEA کلاسیک و روش ارائه شده را بر روی پایگاه داده ها ارائه داده است . در نهایت ، نتیجه گیری و کارهای آینده در بخش 5 ارائه شده است .

2. روش ها

تخصیص و تکه تکه شدن مشکلاتی مرتبط به یکدیگر هستند که حل آنها به طور همزمان دشوار می باشد ولی منجر به عملکرد بهتر در نرم افزار ها خواهد شد . برای بهترین دانش ما ،BEA به طور همزمان بر روی تخصیص و تکه تکه شدن اعمال نمی شود . زیرا در پارتیشن بندی عمودی ویژگی هایی که معمولا با هم دیده می شوند معمولا در یک قطعه قرار می گیرند ، و اندازه گیری دقیق آنها با یکدیگر ضروری می باشد .BEA از وابستگی ویژگی ها برای ایجاد خوشه هایی از ویژگی ها استفاده می کند که بیشتر شبیه هم هستند . آن را با استفاده از ویژگی (AU) و دسترسی به کوئری (QA) ماتریس ویژگی های وابستگی (AFF) و در نهایت خوشه های ماتریس وابستگی (CA) به وسیله در موقعیت قرار دادن و مجددا در موقعیت قرار دادن ستون ها و سطر هایی از ویژگی ها صورت می پذیرد . اندازه گیری وابستگی ها بسیار ساده است . اندازه گیری این وابستگی ارائه شده در BEA اصولا بر اساس دسترسی همزمان به ویژگی مبتنی برA_i و ویژگی A_j از رابطه $R(A_1,A_2,\ldots,A_n)$ با کوئری q_k برای هر کوئری در $Q=(q_1,q_2,\ldots,q_n)$ است . به عبارت دیگر ، دو ویژگی زمانی شبیه به هم در نظر گرفته می شوند که اگر با یک کوئری بتوان هر دوی آنها را دید . این موضوع را با AU به وسیله $A_{ij}=1$ و $A_{ik}=1$ به طور همزمان برای ویژگی j و k که به وسیله کوئری i دیده شده است می باشد و با در نظر گیری تمایلات ویژگی های A_i و A_j به عنوان $aff(A_i,A_j)$ صورت می پذیرد ، تعداد تکرار دسترسی به کوئری K بر سایت I را به شکل $Freq_1(q_k)$ نشان می دهیم ، و دسترسی به اجرای کوئری K بر سایت I را به شکل $acc_1(q_k)$ نشان می دهیم ، معادله برای این وابستگی به شکل زیر ارائه شده است :

$$aff(A_i, A_j) = \sum_{k|use(q_k,A_i)=1 \wedge use(q_k,A_j)=1} \sum_{\forall S_l} freq_l(q_k) * acc_l(q_k) \qquad (2)$$

پس از تولید AFF با استفاده از اندازه گیری وابستگی توصیف شده ، خوشه هایی از ویژگی ها با استفاده از تابع تقسیم ایجاد می شود. Split(AFF) ماتریس AFF را به عنوان ورودی می گیرد ، و ستون ها و سطر های آن را تغییر می دهد و سپس ماتریس CA را تولید می کند. جایگشت سپس در راهی صورت می گیرد که سبب به حداکثر رسیدن اندازه گیری های جهانی زیر شود:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} aff_{ij}[aff_{i,j-1} + aff_{i,j+1} + aff_{i-1,j} + aff_{i+1,j}] \quad (3)$$

در جایی که

$$aff_{i,0} = aff_{0,j} = aff_{i,n+1} = aff_{n+1,j} = 0 \quad (4)$$

آخرین مجموعه از شرایط به مراقبت از مواردی می پردازد که یک ویژگی در CA به سمت چپ از چپ ترین ویژگی یا به سمت راست از راست ترین ویژگی در طول جایگشت ستون قرار داده شده است ، و قبل از بالاترین ردیف و زیر آخرین ردیف در طول جایگشت ردیف قرار داده شده است . در فرآیند تقسیم باند بین دو ویژگی i و j و سهم خالص به اندازه گیری وابستگی جهانی به اندازه گیری قرار دادن ویژگی k بین i و j می پردازد که نقش کلیدی را ایفا می کند . پیوند بین ویژگی های i و j به شکل زیر تعریف می شود:

$$bond(A_i, A_j) = \sum_{k=1}^{n} aff(A_k, A_i)aff(A_k A_j) \quad (5)$$

سهم خالص قرار دادن ویژگی k بین i و j به شکل زیر تعریف شده است:

$$\begin{aligned} cont(A_i, A_k, A_j) \quad &= 2bond(A_i, A_k) + 2bond(A_k, A_j) \\ &\quad -2bond(A_i, A_j) \end{aligned} \quad (6)$$

تابع تقسیم بندی ماتریس وابستگی خوشه شده را در دو مرحله تولید می کند :

الگوریتم 1 . VF و تخصیص همزمان

Require:
Communication Cost Matrix
Attribute Usage Matrix (AU)
Query Access Matrix (QA)
Number of attributes
Output:
Clustered Attribute Matrices as a Tree
Allocated clusters to sites
1: Optimizing Communication Cost Matrix and generating DM
2: Generating Minimized Query Access matrix (MQA)
$MQA = \sum_i \sum_k DM * QA$
3: $IIC \leftarrow AU$
4: **while** *Number of attributes in IIC* > 3 **do**
Run Modified BEA Algorithm (IIC,MQA)
Add LC and IIC to Tree
5: **end while**
6: Calculate Storage Cost
$\sum_{i=1}^{m} X_{ij} * SC_{ij} * V$
7: Allocate each cluster to site with minimum cost

مقدار دهی اولیه: یکی از ستون های ماتریس وابستگی AFF را در داخل ماتریس CA تنظیم می کند و قرار می دهد.

تکرار: هر کدام از n-i ستون های باقی مانده را که در آنها i شماره ستون هایی است که در حال حاضر در CA قرار گرفته اند را بر می دارد و سپس سعی می کند آن را در باقی مانده i+1 موقعیت ها در CA قرار دهد . انتخاب محل قرار گیری سبب می شود که بیشترین سهم اندازه گیری وابستگی جهانی که در بالا توضیح داده شد به وقوع بپیوندد . این قضیه را انقدر ادامه می دهیم که دیگر هیچ ستونی برای قرار دادن باقی نمانده باشد .

از آنجایی که نتیجه خوشه بندی BEA مرز تقسیم شده بین دو مجموعه از ویژگی ها است ، BEA برای پایگاه داده های بزرگ به خوبی عمل نمی کند . بنابراین ، ما نیاز به یک روش بهتر برای شناسایی نامزدهای پارتیشن بندی داریم . همانطور که ما متوجه شدیم شباهت های دو ویژگی زمانی است که آنها به طور همزمان در یک کوئری رخ می دهند ، عدم وجود همزمانی آنها در یک کوئری سبب می شود که ما اینگونه برداشت کنیم که وزن اندازه گیری شده برای شباهت های آنها هم یکسان باشد . علاوه بر این ، وقوع جداگانه هر یک از این ویژگی ها را می توان به عنوان یک اقدام وزنی از عدم تشابه در نظر گرفت . n_{00}, n_{11}, n_{01} را به عنوان تعداد غیبت های همزمان ویژگی ها در نظر بگیرد ، که نشان دهنده ویژگی ها هستند ، و وقوع هر کدام از این ویژگی ها برای هر کوئری در ماتریس استفاده از وابستگی نیز

به همین ترتیب خواهد بود . به طور مشابهی S_{ij} نیز اندازه گیری می شود که توسط ژو و وونشگ در [20] صورت گرفته است که در آن n_{11} و n_{00} نشان دهنده ذره هایی برای تشابه و n_{10} و n_{01} نشان دهنده عدم تشابه است.

$$S_{ij} = \frac{n_{11} + n_{00}}{n_{11} + n_{00} + w_1(n_{01} + n_{10})} \quad (7)$$

این اندازه گیری به محاسبه تطبیق بین دو جسم به طور مستقیم می پردازد . جفت های غیر مشابه بر اساس سهم خودشان به شباهت مورد وزن کشی قرار می گیرند . اگر یکی از این تطابق را مشابه با w_1 در نظر بگیریم برابر با یک می شود . در خوشه به معنی محدود [24] ضریب را برابر با 2 در نظر می گیریم . گاور [11] w_1 را برابر با ½ پیشنهاد می دهند . این گونه می توان جمع بندی کرد که ، انتخاب یک مقدار مناسب برای وزن w_1 بستگی به روش و همچنین ساختار و تعریف خود پایگاه داده دارد .

به هر یک از این کوئری ها می توان در زمان های مختلف و از نقاط مختلفی دسترسی داشت . تکرار دسترسی به کوئری در هر سایت در ماتریس دسترسی به کوئری (QA) تعریف شده است . ورودی QA_{ij} نشان دهنده تعداد دفعاتی است که در آن کوئری i در سایت j قابل دسترس است . از سوی دیگر هزینه های ارتباطی بین سایت هایی از یک پایگاه داده توزیع شده نقشی کلیدی در عملکرد یک پایگاه داده توزیع شده را ایفا می کند . ماتریس فاصله (DM) یک ماتریس مربعی نا متقارن است که نشان دهنده هزینه های است که باید با استفاده از روش تعریف شده توسط بنتلی و دیتمن [25] به حداقل برسد . ضرب DM در QA یک ماتریس جدید را تولید می کند که در آن هزینه های ارتباطی بین سایت ها و دسترسی به کوئری در هر ککانی به طور همزمان در نظر گرفته می شود و هم آن را تحت تاثیر قرار می دهد و از آنجایی که DM یک ماتریس فاصله به حداقل رسیده است پس ماتریس نتیجه شده ماتریس حداقل دسترسی به کوئری (MQA) است .

$$MQA = \sum_i \sum_k DM * QA \quad (8)$$

هزینه کلی ذخیره سازی (TSC) برای هر ویژگی در هر سایت بستگی به هزینه ذخیره سازی برای هر آیتم و نهایت تعداد سایت دارد.

$$TSC = \sum_{i=1}^{m} X_{ij} * SC_{ij} * V \quad (9)$$

در جایی که

$$X_{ij} = 1 \text{ if fragment } i \text{ is allocated to site } j \quad (10)$$

ویژگی هایی که دارای حداقل هزینه ذخیره سازی برای هر سایت هستند به همان سایت تخصیص داده می شوند . معادله 9 نیز بر روی ویژگی های باقی مانده و سایت هایی که دارای حداقل هزینه برای تخصیص به مقدار این ویژگی ها است اعمال می شود.

3 . الگوریتم

الگوریتم 1 با هزینه ارتباطی بین شبکه سایت ها ، ماتریس QA ، ماتریس AU ، و ویژگی هایی که به عنوان ورودی به حساب می آیند کار می کند و درختی از ویژگی های خوشه شده به همراه تخصیص آنها به سایت ها را تولید می کند. این الگوریتم به شکل سلسله مراتبی کار می کند و به تدریج یک درخت خوشه ای ایجاد می کند . در هر بار تکرار آن دو مجموعه از ویژگی ها تولید می شود . مجموعه ای بزرگتر از ویژگی های مشابه که ما آن را به عنوان تکرار خوشه ورودی (IIC) می نامیم به عنوان ورودی برای تکرار بعدی مورد استفاده قرار می گیرد . سایر مجموعه های کوچک را با نام خوشه های برگ می شناسیم (LC) زیرا جداسازی شده است و به عنوان گره های برگ در درخت قرار می گیرند . در مرحله اول DM به وسیله ماتریس بهینه سازی هزینه های ارتباطاتی با استفاده از کار وایتن و همکارانش [25] صورت می پذیرد . سپس MQA به وسیله ضربت QA در ماتریس DM تولید می شود . قدم بعدی مقدار دهی اولیه IIC به وسیله ماتریس AU است . این الگوریتم با تکرار الگوریتم BEA اصلاح شده ادامه پیدا می کند ، که بعدا توضیح داده خواهد شد ، تا زمانی که ویژگی هایی که در IIC شمارش می شود برابر با 3 باشد . از آنجایی که هر تکرار دارای بیشترین شباهت باشد در یک گروه IIC قرار می گیرد ، ما فرض می کنیم پس از این تعداد تکرار ، IIC نتیجه شده شامل بیشترین ویژگی های مشابه از تمامی موارد است به همین دلیل نیازی نیست که از این بیشتر برویم . سپس ، هزینه ذخیره سازی برای هر ویژگی در هر سایت مورد محاسبه قرار می گیرد و در نهایت بر

اساس این هزینه ها ، هر خوشه از ویژگی ها به مناسب ترین سایت تخصیص پیدا می کند. آخرین IIC به تمامی سایت ها تخصیص پیدا کرده است.

الگوریتم BEA تغییر داده شده در حقیقت وابستگی اندازه گیری شده را در BEA اصلی تغییر می دهد . همانطور که پیش از این اشاره شد BEA به سادگی با استفاده از وقوع همزمان ویژگی ها ماتریس AFF را ایجاد می کند . در BEA اصلاح شده که در اینجا ارائه شده است ، سایر احتمالات نیز در نظر گرفته می شود . S_{ij} را از کار ژو و ووشج [20] می گیریم تا بتوانیم غیبت های نصفه را نیز در نظر بگیریم ، و n_{00} ، n_{01} و n_{10} را با وابستگی جدیدی که به وسیله S_{ij} اندازه گیری شده است را با استفاده از رابطه زیر مورد محاسبه قرار می دهیم:

$$S_{ij} = \frac{n_{11} + w_1 n_{00}}{n_{11} + w_1 n_{00} + coef} \qquad (11)$$

وزن w_1 وw_2 بین 0 و 1 است زیرا n_{00} ، n_{01} و n_{10} دارای اثر مثبت کمتری بر روی شباهت در مقایسه با n_{11} هستند. علاوه بر این ، اینگونه می توان استنباط کرد که w_1 باید بزرگتر از w_2 باشد . روش محاسبه مقدار هر کدام از این وزن ها بستگی به ساختار و تعریف جدول و روابط آنها در پایگاه داده ها دارد . اندازه گیری گوور و لجندر [11] و راجرز و تانیموتو [22] دارای برخی از روش ها برای محاسبه مقدار وزن هستند . هر کدام از این وزن ها با در نظر گیری ساختار ها و تعاریف پایگاه داده و کوئری های آن مورد محاسبه قرار می گیرد . ساختار پایگاه داده ها به ما برخی از اطلاعات در مورد روابط بین ویژگی های مختلف را می دهد . بنابراین ، با در نظر گیری کوئری ها ، مقادیر اولیه وزن ها استنباط می شود و پس از تولید نتایج اولیه ، مقادیر وزن ها به مقدار کمی تغییر پیدا می کند به طوری که نتایج نشان دهنده روابط واقعی مورد انتظار بین ویژگی ها با توجه به ساختار پایگاه داده ها است .

الگوریتم 2 . الگوریتم BEA اصلاح شده

Require:
Attribute Query Matrix
Query Access Matrix
Result:
AFF Matrix
1: $S \leftarrow MQA$
2: **for** *each attribute number i* **do**
3: $QS_i \leftarrow sum(S_{ij})$
4: **end for**
5: **for** *each attribute number i* **do**
6: **for** *each attribute number j* **do**
7: *initialize* $n_{00}, n_{11}, n_{01}, n_{10}$ *by* 0
8: **if** $(i == j)$ **then**
9: $AFF_{ij} \leftarrow sum(A_j) * QS$
10: **else**
11: **for** *each query number k* **do**
12: *calculating* $n_{00}, n_{11}, n_{01}, n_{10}$
13: **if** $(n_{01} == 0$ *and* $n_{10} > 0)$ *or* $(n_{10} == 0$ *and* $n_{01} > 0)$ **then**
14: $coef \leftarrow (-1)(n_{01} + n_{10}) * w_1$
15: **else**
16: $coef \leftarrow (|n_{01} - n_{10}|) * w_1$
17: **end if**
18: $S_{ij} \leftarrow (n_{11} + w_2 * n_{00})/(n_{11} + w_2 * n_{00}) + coef$
19: **end for**
20: **end if**
21: $AFF_{ij} \leftarrow S_i * QS_i$
22: **end for**
23: **end for**
24: *call Function Split(AFF)*

همانطور که پیش از این نیز ذکر شد در غیاب همزمانی ویژگی ها این روش می تواند به ما اطلاعاتی در مورد تشابه ویژگی ها می دهد . از سویی دیگر ، از آنجایی که این اثر در مقایسه با اثر حضور همزمان ناچیز است ، n_{00} دارای برخی اثر وزنی بر روی اندازه گیری وابستگی و بنابراین w_1 دارای مقداری بین 0 و 1 است .

متغیر coef در مخرج نشان دهنده اثر n_{01} و n_{10} است . چهار احتمال در این بین وجود دارد . زمانی که مقدار n_{01}> 0 باشد و n_{10} = 0 باشد ، نشان دهنده این است که برای دو ویژگی از A_i و A_j ، تمامی کوئری هایی که به A_i دسترسی دارند به A_j دسترسی ندارند . این بدان معنا است که این ویژگی ها دارای سطوحی از تشابه هستند . در نتیجه S_{ij} باید مقداری بیشتر از وزن اندازه گیزی شده در مخرج را داشته باشد ، W_2 ، باید مقدار منفی داشته باشد . این در خط 12 و 13 از الگوریتم 2 نشان داده شده است . همین قضیه برای موردی که در آن n_{10}>0 و n_{01}=0 باشد وجود دارد . احتمال دیگر این است که هم n_{01} و هم n_{10} بیشتر از 0 خواهند بود . این شرایط بدین معنی است که A_i و A_j دارای رفتاری مشابه با کوئری های مختلف که به آنها دسترسی دارند ، ندارند . این داری تاثیر منفی بر روی شباهت است ، برای همین وزن اندازه گیری شده در مخرج ، W_2 ، باید مثبت باشد . این قضیه در خط 15 نشان داده شده است .

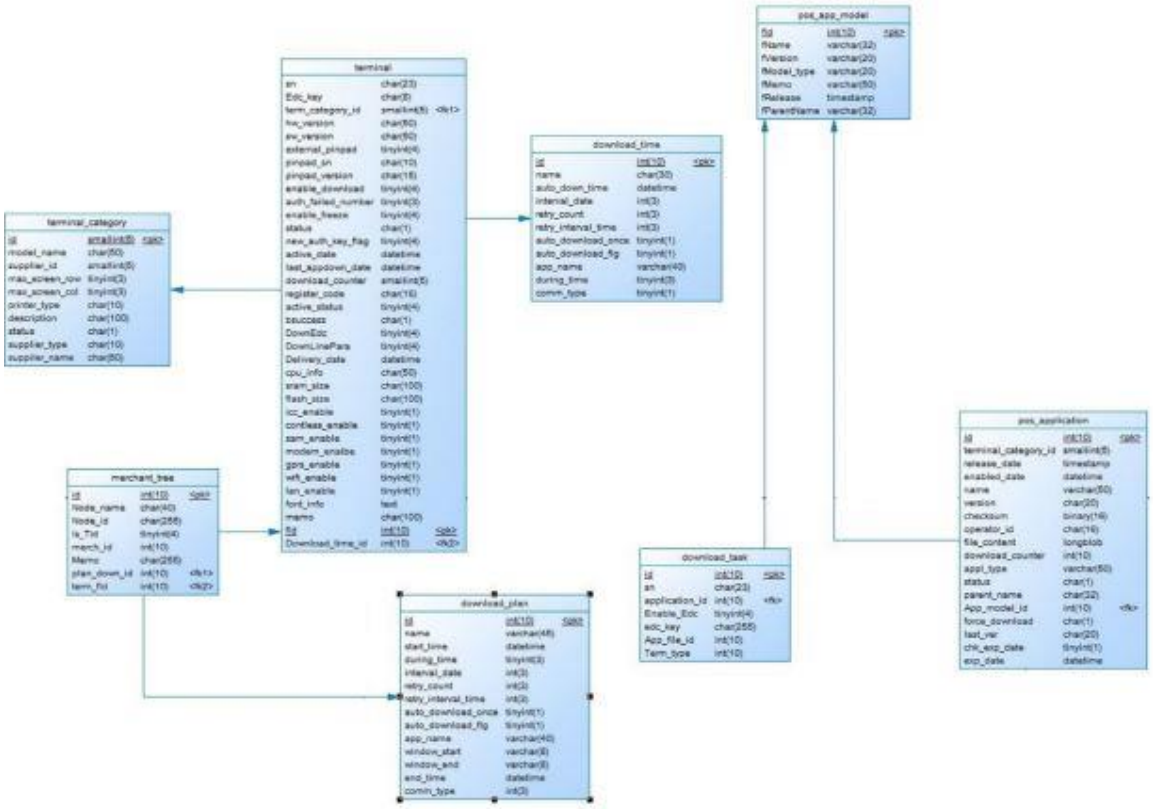
پس از اینکه ماتریس AFF مورد محاسبه قرار گرفت ، الگوریتم تابع تقسیم را فراخوانی می کند که ما در بخش 2 آن را شرح دادیم و سپس خوشه هایی از ویژگی ها را ایجاد می کند.

4. مورد مطالعاتی

برای تخمین میزان بهبودی و دقت الگوریتم ما ، ما هم BEA کلاسیک و هم الگوریتم خودمان را بر روی پایگاه داده ای از سیستم مدیریتی ترمینال (TMS) اعمال کردیم . TMS یک سروری است که به فروشگاه ها (یا سوپرمارکت ها) متصل شده است ، این ترمینال دارای شماره سریال منحصر به فردی است . بستگی به نوع ترمینال، اطلاعات ترمینالی یا اطلاعات سیستم عامل را می توان دانلود یا به روز رسانی کرد . از آنجایی که این فروشگاه ها در سایت های مختلفی واقع شده است ، TMS مسلما با پایگاه داده های توزیع شده بهتر می تواند عمل کند . هر ترمینال دارای یک شماره سریال منحصر به فرد است ، یکی از وظایف تعریف این شماره برای هر گروه ترمینال است . این وظایف شامل یک یا تعداد بیشتری از فایل ها است که می توانند همراه یک گروه پایانه باشند . هر ترمینال ، شامل گروهی از ترمینال و وظیفه است که دارای یک جدول است . یک نمای نمونه از جداول و روابط آنها در شکل 4 به تصویر کشیده شده است . پس از بررسی تراکنش ها ، هشت مورد از رایج ترین تراکنش ها و روابط آنها در TMS در نظر گرفته شده است ، 8 ویژگی (در جدول 2 به تصور کشیده شده است) برای توزیع در هفت سایت انتخاب شده است .

ماتریس های AU ، QA ، DM در شکل های 1 تا 3 به تصویر کشیده شده است . دسترسی به کوئری ورودی برای هر دو الگوریتم MQA بوده است . وزن های w_1 و w_2 در الگوریتم ما بر روی 7 . 0 و 3 . 0 تنظیم شده بود . درخت های خوشه ای نتیجه شده برای هر الگوریتم در شکل 5 نمایش داده شده است . همانطور که می توان مشاهده کرد ، هر دو الگوریتم دارای رفتاری یکسان تا تکرار 4 ام هستند . BEA کلاسیک ویژگی شماره 2 را جدا سازی می کند و ویژگی های شماره 3 و 4 و 5 و 7 را در داخل یک خوشه قرار می دهد.

شکل 4 – جدول رابط در TMS



جدول ۲ لیستی از ویژگی ها و جداول مربوطه

شماره	لیستی از ویژگی ها	جدول مربوطه
1	*model – name*	*terminal – category*
2	*term – fid*	*terminal*
3	*hw – version*	*terminal*
4	*pinpad – version*	*terminal*
5	*flash – size*	*terminal*
6	*app – name*	*download – time*
7	*interval – date*	*download – time*
8	*start – time*	*download – plan*

شکل 1 . ماتریس دسترسی کوئری (QA) برای هفت سایت

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
q_1	28	44	45	31	29	32	10
q_2	47	31	29	33	0	1	7
q_3	35	50	43	37	6	31	9
q_4	29	26	37	45	43	18	2
q_5	41	24	29	50	24	2	32
q_6	44	40	12	39	43	24	14
q_7	50	11	33	29	10	9	27
q_8	0	25	4	47	28	6	35

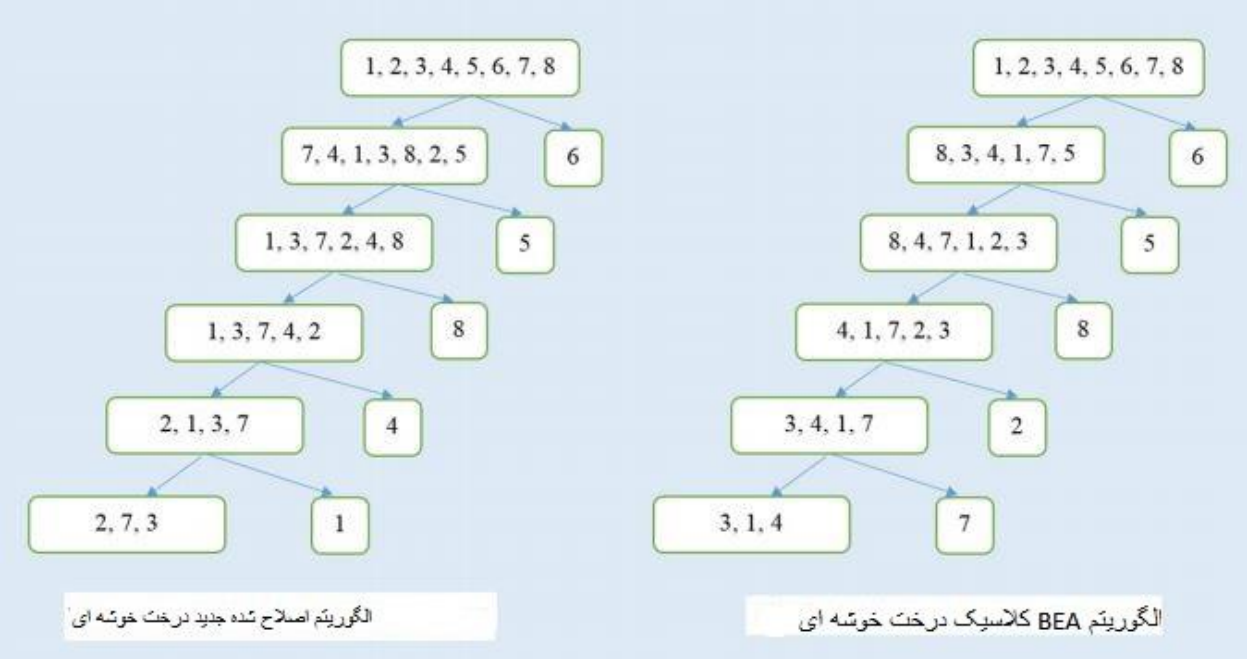
شکل 2 . ماتریس فاصله هزینه های ارتباطاتی برای هفت سایت

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
S_1	0	7	12	8	4	9	4
S_2	7	0	11	7	8	10	3
S_3	12	11	0	10	16	13	8
S_4	8	7	10	0	12	6	4
S_5	4	8	16	12	0	9	8
S_6	9	10	13	6	9	0	7
S_7	4	3	8	4	8	7	0

شکل 3 . ماتریس استفاده از ویژگی ها (AU) برای 8 کوئری

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
q_1	1	0	1	1	0	1	1	0
q_2	1	1	1	1	0	1	1	0
q_3	0	1	1	0	1	0	0	0
q_4	1	1	0	1	1	1	1	1
q_5	1	1	1	1	0	1	1	1
q_6	1	0	1	0	0	1	1	1
q_7	0	1	1	1	0	0	0	1
q_8	1	1	0	0	1	0	1	0

شکل 5 ویژگی های سلسله مراتبی درخت خوشه ای



جدول 3 شباهت ویژگی ها

ویژگی ها	n_{11}	n_{00}	n_{10}	n_{01}	*coef*	S_{ij}
A_2 *and* A_1	4	0	2	2	0	1
A_2 *and* A_3	4	0	2	2	0	1
A_2 *and* A_7	4	0	2	2	0	1
A_4 *and* A_1	4	1	2	1	0.35	0.93
A_4 *and* A_3	4	1	2	1	0.35	0.93
A_4 *and* A_7	4	1	2	1	0.35	0.93
A_1 *and* A_3	4	0	2	2	0	1
A_1 *and* A_7	6	2	0	0	0	1
A_3 *and* A_7	4	0	2	2	0	1

با توجه به شرایطی که در الگوریتم ما اعمال شده است ، Coef و S_{ij} مورد محاسبه قرار گرفته است و در جدول 3 آن را نشان دادیم . واضح است که A_4 کمتر به سایر ویژگی ها نسبت به A_2 شباهت ها دارد بنابراین به درستی جدا شده است. ما می توانیم نتیجه بگیریم که الگوریتم جدید اندازه گیری و خوشه بندی ویژگی ها را بسیار بهتر انجام می دهد.

5. جمع بندی

پایگاه داده های توزیع شده سبب کاهش هزینه به روز رسانی و بازیابی اطلاعات و افزایش عملکرد و در دسترس بودن می شوند ، اما طراحی DDBMS بسیار پیچیده تر از طراحی پایگاه داده های مرکزی است . یکی از اصلی ترین چالش هایی که بسیار بر روی عملکرد DDBS تاثیر گذار بوده است تکه تکه کردن و تخصیص این قطعات به سایت ها می باشد . تخصیص و تکه تکه شدن از لحاظ منطقی قابل ادغام است و می توان به طور همزمان آن را انجام داد . در این مقاله ما یک روشی که سبب ادغام تکه تکه شدن عمودی و تخصیص می شود را پیشنهاد دادیم . برای رسیدن به این هدف ما الگوریتم انرژی پیوندی را با اصلاح اندازه گیری وابستگی ها در یک فرآیند سلسله مراتبی اعمال کردیم و به طور همزمان هزینه تخصیص داده برای هر سایت و تخصیص این قطعات به سایت های مناسب را مورد محاسبه قرار دادیم . استفاده از فرآیند سلسله مراتبی در خوشه بندی مجموعه هایی که دارای ویژگی های مشابه هستند و برای تکه تکه شدن بهتر داده ها صورت می پذیرد . از سویی دیگر ، به وسیله اجرای همزمان هزینه محاسباتی که برای ما دارد وابسته به محاسبه تکه تکه شدن داده ها و تخصیص آنها دارد.

برای کارهای آینده می توان به بهینه سازی تابع هزینه برای تخصیص داده و با در نظر گیری بازیابی و بروزرسانی های مکرر برای هر ویژگی و اعمال روشی بهتر جهت محاسبه وزن هایی برای اندازه گیری شباهت ها است .

References

[1] Adrian Runceanu, Fragmentation in distributed databases, Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering, 2008, pp. 57–62.

[2] G.S. Chinchwadkar, A. Goh, An overview of vertical partitioning in object oriented databases, Comput. J. 42 (1) (1999).

[3] M. Hammer, B. Niamir, A heuristic approach to attribute partitioning, in: Proceedings ACM SZGMOD International Conference on Management of Data, Boston, Mass., ACM, New York, 1979.

[4] Hassan I. Abdalla, A synchronized design technique for efficient data distribution, Comput. Human Behav. 30 (2014) 427–435.

[5] H. Ma, K.D. Schewe, M. Kirchberg, A heuristic approach to vertical fragmentation incorporating query information, in: Proc. 7th International Baltic Conference on Databases and Information Systems (DB and IS), 2006, pp. 69–76.

[6] Latiful A.S.M. Hoque, Shahidul Islam Khan, A New Technique for Database Fragmentation in Distributed Systems, International Journal of Computer Applications 5(9):2024, August 2010. Published By Foundation of Computer Science.

[7] E.S. Abuelyaman, An optimized scheme for vertical partitioning of a distributed database, Int. J. Comput. Sci. Netw. Sec. 8 (1) (2008).

[8] F. Baiao, M. Mattoso, G. Zaverucha, A distribution design methodology for object DBMS, Distrib. Parallel Databases 16 (1) (2004) 4590.

[9] Haroun Rababaah, H. Hakimzadeh, Distributed Databases: Fundamentals and research, Advanced Database B561, Spring 2005.

[10] N. Iacob, Data replication in distributed environments, Annals of the Constantin Brncusi, University of Trgu Jiu, Economy Series, Issue 4, 2010.

[11] J. Gower, A general coefficient of similarity and some of its properties, Biometrics 27 (1971) 857872.

[12] J.O. Hauglid, N.H. Ryeng, DYFRAM: dynamic fragmentation and replica management in distributed database systems, Distrib. Parallel Databases 28 (2010) 157185.

[13] J.A. Hoffer, An integer programming formulation of computer database design problems, Znf. Sci. 11 (July 1976) 29–48.

[14] J.A. Hoffer, D.G. Severance, The use of cluster analysis in physical database design, in: Proceedings 1st International Conference on Very Large Databases, Framingham, Mass., 1975.

[15] Jin Hyun Son, Myoung Ho Kim, An adaptable vertical partitioning method in distributed systems, J. Syst. Softw. 73 (2004) 551–561.

[16] S. Mahmoud, J.S. Roirdon, Optimal allocation of resources in distributed information networks, ACM Trans. Database Syst. 1 (1976) 1.

[17] M.T. Ozsu, P. Valduriez, Principles of Distributed Database Systems, Alan Apt, New Jersey, 1999.

[18] S. Navathe, S. Ceri, G. Wiederhold, J. Dou, Vertical partitioning algorithms for database design, ACM Trans. Database Syst. 9 (4) (1984) 680710.

[19] S.K. Rahimi, F.S. Haug, Distributed Database Management Systems, A John Wiley and Sons Inc. Publication, IEEE Computer Society, 2010.

[20] Rui Xu, Donald Wunsch II, Survey of clustering algorithms, IEEE Trans. Neural Netw. 16 (3) (2005).

[21] M. Schkolnick, A clustering algorithm for hierarchical structure, ACM Trans. Database Syst. 2 (1977) 1.

[22] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Introduction to Data Mining, 2005, ISBN: 0-321-32136-7.

[23] Shamkant Navathe, Stefano Ceri, Gio Wiederhold, Jinglie Dou, Vertical partitioning algorithms for database design, ACM Trans. Database Syst. 9 (4) (December 1984).

[24] K. Wagstaff, S. Rogers, S. Schroedl, Constrained -means clustering with background knowledge, in: Proc. 8th Int. Conf. Machine Learning, 2001, pp. 577–584.

[25] J. Whitten, L. Bentley, K. Dittman, Systems Analysis and Design Methods, sixth ed., McGraw-Hill, 2004.