متد: فراخوانی تابع رشدFP درخت FP ، Null

رویه FP-growth(Tree,a):

اگر Tree شامل یک مسیر پسوندی باشد.

فرض می کنیم p تک مسیر پیشوندی Tree باشد

فرض می کنیم q قسمت چند مسیره با جایگزینی بالاترین راس شاخه ها با یک ریشه نال باشد

برای هر ترکیب b از راس ها در p انجام می دهیم

الگوی b اجتماع a را به وسیله ی support = minimum support of nudes in b ایجاد می کنیم

فرض کنید freq-pattern-set(P) مجموعه الگوهای ایجاد شده باشد.

پایان

پایان

در غیر اینصورت

فرض می کنیم Q همان Tree باشد

پایان

برای هر عضو ai که درون q باشد انجام بده

الگوی b= ai  را به وسیله ی support = ai .support می سازیم

اساس الگوی شرطی b را ایجاد می کنیم و سپس درخت FT شرطی b را می سازیم و آنرا Treeb نامگذاری می کنیم

اگر Treeb برابر تهی نباشد آنگاه FP-growth(Treeb,b) را فراخوانی می کنیم

فرض کنید freq-pattern-set(q) مجموعه الگوهای ایجاد شده باشد.

پایان

بازگردان

freq-pattern-set(p) freq-pattern-set(q) (freq-pattern-set(p) x freq-pattern-set(q))

3.1.3 استخراج الگو پی در پی

مسیله استخراج پی در پی در ابتدا در (۱۱) معرفی شد . دو مسیله الگوی پی در پی عبارتند از : ٬{۸۰ ٪ افرادی که یک تلویزیون می خرند در طول یک روز یک دوربین ویدیویی هم می خرند} و {هر زمان که سهام مایکروسافت ۵٪ سقوط می کند سهام آی بی ام در طول ۳ روز حداقل ۴٪ افت می کند} الگوهای بالا می توانند برای تعیین استفاده بهینه از فضای ویترین برای راحتی خریدار و یا تعیین درست قدم بعدی در یک بحران اقتصادی استفاده شود. استخراج الگوی پی در پی همچنین برای آنالیز داده های زیستی در حالتی که الفبای بسیار کوچک(برای نمونه ۴ برای دنباله دی ان ای و ۲۰ برای دنباله پروتین)

.و الگوهای طولانی با طول مخصوص چند صد یا حتی چند هزار مکررا ظاهر می شوند بسیار مهم است

یافتن دنباله می تواند به اساسا به عنوان یافتن ارتباط در یک دیتابیس موقت در نظر گرفته شود. در حالی که قواعد ارتباط فقط الگوهای داخل رویدادی را تشخیص می دهد . استخراج الگوی پی در پی الگوهای میان رویدادی را تشخیص می دهد . وظایف بسیار مهمی در ارتباط با استخراج قواعد ارتباطی مانند ارتباط .علیت . قسمت ها . الگوهای چند بعدی . الگوهای ماکس . تناوب بخشی و پدیدار شدن الگوها. کاوش عمیق مسایل استخراج الگوی پی در پی قطعا به یافتن راه حل بهینه برای مسایل تحقیقاتی ذکر شده در بالا کمک خواهد کرد.

متدولوژی های استخراج الگوی پی در پی به صورت گسترده در مسایل مرتبط بسیاری از جمله استخراج الگوی پی در پی جامع و استخراج الگوی پی در پی بر اساس مانع و استخراج الگوی پی در پی افزایشی و استخراج بخش مکرر و استخراج الگوی پی در پی تخمینی و استخراج الگوی متناوب بخشی و استخراج الگوی موقت در جریان های داده و استخراج الگوی دنباله بسته و ماکسیمال . در این بخش به دلیل محدودیت فضا ما بر روی معرفی الگوریتم استخراج الگوی پی در پی جامع که به دلیل فایده رسانی به استراتژی های آن(برای مثال اپریوری ابتکاری و رشد الگو بر اساس تجسم) به بقیه نوع ها پایه ای ترین نوع است. جزییات بیشتر و بررسی استخراج الگوی پی در پی را می توان در (۲۴۹و۱۷۲) یافت.

مسایل استخراج الگو پی در پی

فرض کنید I={i1 , i2 ,……, ik}مجموعه ای از اشیا باشد. یک زیرمجموعه از I یک عنصر یا مجموعه شی خوانده می شود. دنباله s عبارتست از (t1 , t2 ,……, tl) به گونه ای که tj یک عنصر باشد برا مثال t1 برای . عنصر tj عبارتست از

(x1 x2 …… xm) در صورتی که xk  یک عضو باشد مثلا xk برای برای اختصار براکت ها در عناصری که فقط یک عضو دارند حذف شده است. به این معنی که عنصر (x) به صورت x نوشته می شود.تعداد اعضا درون یک دنباله طول آن دنباله خوانده می شود. یک دنباله با طول به صورت خوانده می شود. یک دنباله در دنباله دیگری مانند

وجود دارد اگر اعداد صحیح وجود داشته باشد به گونه ای که , ، ... ، باشد. ما را یک زیر دنباله از می نامیم و یک ابر دنباله از است. با داشتن دنباله

و یک عضو ، به معنی الحاق s به a است که دو فرم ممکن دارد مانند توسعه مجموعه عضوی(IE) ، و توسعه دنباله ای (SE) ، . اگر

در آنصورت p یک پیشوند است و s یک پسوند است.

یک دیتابیس دنباله S یک مجموعه از تاپلهای (sid,s) است به گونه ای که sid یک sequence\_id است و s یک sequence است. گفته می شود یک تاپل (sid,s) شامل یک دنباله b باشد ، اگر b یک زیردنباله از s باشد . support یک دنباله،b، در یک دیتابیس دنباله ، تعداد تاپل های درون دیتابیس است که شاملb می شوند و با support(b) نامیده می شود. با دادن یک عدد صحیح مثبت مشخص شده توسط کاربر یک دنباله b یک الگوی دنباله ای مکرر خوانده می شود اگر

الگوریتم های موجود برای استخراج الگو پی در پی

الگوریتم های استخراج الگو پی در پی می توانند در دو گروه دسته بندی شوند یک نوع Apriori مانند هستند مثل Apriori-all ، GSP ،Spade و Spam. نوع دیگر رشد الگو بر اساس نمایش است مانند PrefixSpan

AprioriAll

استخراج الگوی دنباله ای در ابتدا به وسیله ی آگراوال معرفی شد ، و سه الگوریتم اپریوری پایه پیشنهاد شد. دیتابیس تراکنشی با 3 مقدار customer-id ، trasnsaction-time ، purchased-time داده شده است پروسه استخراج به 5 فاز تقسیم می شود:

* **فاز مرتب سازی:** دیتابیس تراکنش اولیه براساس customer و transaction time مرتب می شود. شکل 3.3 اطلاعات تراکنشی مرتب شده را نمایش می دهد.
* **فاز مجموعه عنصرهای L:** دیتابیس مرتب شده در ابتدا بررسی می شود تا مجموعه عنصر های l مکرر بر اساس استانه support مشخص شده توسط کاربر به دست آید . فرض کنید support حداقلی 70% است. در این حالت تعداد ساپورت حداقلی 2 است و نتیجه مجموعه عنصرهایl در شکل 3.4 لیست شده است.

شکل 3و3

شکل 4و3

شکل 5و3

* **فاز تراکنش:** تمام مجموعه عنصر های بزرگ به یک سری از اعداد صحیح نگاشت می شوند و دیتابیس اولیه با جایگذاری مجموعه عنصرها تبدیل می شود. برای نمونه با کمک جدول نگاشت در شکل 3.4 دیتابیس تبدیل شده به دست می آید همان گونه که در شکل 3.5 نشان داده شده است.
* **فاز دنباله:** دیتابیس تبدیل شده را کاوش می کند و تمام الکوهای دنباله ای مکرر را می یابد.
* **فاز حداکثری:** الگوهایی را که در سایر الکوهای دنباله ای یافت می شوند حذف می کند . به عبارت دیگر فقط الگوهای دنباله ای حداکثری باقی می مانند.

به دلیل اینکه اکثر این فازها سر راست است تخقیقات بر فاز دنباله ای متمرکز شده است AprioriAll در ابتدا بر اساس الگوریتم اپریوری در استخراج فواعد ارتباطی معرفی شد. دو مرحبه برای استخراج الگوهای دنباله ای وجود دارد که عبارتند از ایجاد کاندید و تست کردن.

پروسه ایجاد کاندید شبیه AprioriGen است. مشخصه اپریوری برای هرس کردن دنباله های کاندیدی که زیر دنباله هایشان مکرر نیست به کار برده می شود. تفاوت این است که زمانی که مولف کاندید را با متصل کردن الگوهای مکرر درعبور های قبلی ایجاد می کند، ردیف های مختلف ترکیبات کاندید های مختلفی را به وجو می آورند. برای نمونه از عضو a و b سه کاندید (ab) ، (ba) و ((ba)) می تواند ایجاد شود . اما در استخراج قواعد ارتباطی فقط ((ba)) ایجاد می شود . علت این است که در استخراج قواعد ارتباطی ترتیب زمانی در نظر گرفته نمی شود. مشخصا تعداد دنباله های کاندید در استخراج الگوی دنباله بسیار بزرگتر از اندازه مجموعه عنصرهای کاندید در استخراج قواعد ارتباطی است در هنگام ایجاد دنباله های کاندید است. جدول 3.5 چگونگی ایجاد کاندید های دنباله 5 را با الحاق دنباله های بزرگ 4 نمایش می دهد . با بررسی مجموعه عناصر بزرگ 4 یافت می شود که در ابتدا مجموعه عنصرهای ((bc)ad) و سپس مجموعه عنصرهای ((bc)(ac)) دارای 3 عضو اول مشترک هستند بر اساس شرط اتصال اپریوری آنها برای ایجاد دنباله کاندید ((bc)(ac)d) اتصال داده می شوند. به صورت مشترک بقیه دنباله های 5 ایجاد می شوند.

پروسه آزمایش ساده و سرراست است . دیتابیس را برای شمارش ساپورت های آن دنباله های کاندید بررسی کنید سپس الگوهای دنباله ای مکرر یافت می شوند.

AprioriAll اولین الگوریتم استخراج الگوهای دنباله ای است و ایده اصلی به وسیله ی الگوریتم های بسیار دیگری به کار گرفته شده است. عیب این الگوریتم ایجاد تعداد زیاد کاندید و عبور های چند باره بر روی دیتابیس است که باعث بالا بودن هزینه محاسبه می شود.

GSP

سریکانت و آگراول تعریف مسیله استخراج الگوهای دنباله ای را به وسیله ی درگیر کردن مشخصه های جدید (محدیت های زمانی ، وقفه تراکنش و طبقه بندی) عمومی سازی نمودند. برای محدودیت های زمانی شکاف حداکثر و شکاف حداقل برای مشخص کردن فاصله بین هر دو تراکنش مجاور در دنباله تعریف می شوند. هنگام امتحان یک کاندید اگر هرکدام است شکاف های کاندید از دامنه بین شکاف حدکثر و شکاف حداقل بیرون بیافتد در آن صورت کاندید یک الگو نیست. علاوه بر آن مولفان مفهوم تراکنش را به وسیله ی یک پنجره لغزنده شبیه سازی کردند به این صورت که هنگامی که بازه زمانی بین دو عنصر کوچکتر از پنجره لغزنده است این دو عنصر در یک تراکنش یکسان فرض می شوند. طبقه بندی برای ساخت الگوهای دنباله ای چند سطحی استفاده می شود.

شکل 6و3

مولفان یک الگوریتم جدید را پیشنهاد دادند که GSP نام دارد و هدف آن یافتن الگوهای دنباله ای عام سازی شده است. مانند الگوریتم AprioriAll دو مرحله در GSP وجود دارد ، ایجاد کاندید و تست.

در پروسه ایجاد کاندید دنباله K کاندید بر اساس دنباله هایk-1 مکرر ایجاد می شود. با دادن یک دنباله و زیردنباله ی c،c یک زیردنباله پیوسته از s است اگر یکی از شروط زیر درست باشد: 1) c از s مشتق شده است با انداختن یک عنصر از یا 2) c از s مشتق شده است با انداختن یک عنصر از که حداقل 2 عضو دارد 3) c یک زیر دنباله پیوسته از c’ است و c’ یک زیر دنباله پیوسته از s است . به طور خاص کاندید ها در دو مرحله ساخته می شوند:

* **فاز اتصال:** دنباله k کاندید با اتصال دو دنباله k-1 که زیردنباله های پیوسته یکسانی داردند ایجاد می شود. هنگامی که ما دو دنباله را متصل می کنیم عضو می تواند به عنوان بخشی از عنصر یا به عنوان یک عنصر جداگانه وارد شود. برای نمونه به دلیل اینکه (d(bc)a) و (d(bc)d) هردو زیر دنباله پیوسته (d(bc)) را به صورت مشترک دارند در این صورت ما می دانیم که دنباله 5 (d(bc)(ad)) ، (d(bc)da)و (d(bc)(ad)) می توانند ایجاد شوند.
* **فاز هرس کردن:** الگوریتم دنباله های کاندیدی را که یک زیر دنباله پیوسته ای با تعداد ساپورت کمتر از ساپورت حداقل هستند حذف می کند. حلاوه بر آن از یک ساختار درخت درهم سازی برای کاهش تعداد کاندید ها استفاده می کند.

پروسه ایجاد کاندید برای دیتابیس مثال زده شده در شکل 3.6 نشان داده شده است. نکته چالش برانگیز برای GSP این است که ساپورت دنباله کاندید به دلیل قوانین عام سازی معرفی شده بسیار مشکل محاسبه می شود در حالی که برای AprioriAll این مشکل وجود ندارد. GSP یک استراتژی بهینه را در نظر می گیرد که شامل دو فاز (فاز پیشرفتی و فاز برگشتی)است. پروسه بررسی اینکه آیا یک دنباله d شامل یک دنباله کاندیدs است در ادامه نشان داده می شود(که تا زمانی که تمام عناصر یافت شوند تکرار می شود):

* **فاز پیشرونده :** منتظر عناصر بعدی s در d می ماند ، تا زمانی که تفاوت میان زمان انتهای عنصر و زمان شروع عنصر قبلی کمتر از شکاف حداکثر باشد. اگر اختلاف بیشتر از شکاف حداکثر باشد به فاز پسرونده سویچ می کند. اگر یک عنصر یافت نشد آنگاه s مشمول d نیست.

شکل 6و3

شکل 7و3

* **فاز پسرونده :** سعی می کند تا عنصر قبلی را به دست بیاورد. فرض کنید عنصر حال حاضر است و زمان پایان برابر t است. الگوریتم بررسی می کند آیا تراکنش های وجود دارد که شامل باشد و زمان های تراکنش منطبق بزرگتر از زمان حداکثر است. تا بعد از انتخاب اختلاف بین و به حداکثر شکاف برسد یا اولین عنصر اتنتخاب می شود. سپس الگوریتم به فاز پیشرونده سویچ می کند. اگر تمام عناصر را نتوان بالا آورد در آن صورت d شامل s نمی شود.

طبقه بندی ها درون دیتابیس اعمال می شوند با توسعه دنباله ها با طبقه بندی های منطبق با آنها. دنباله های اولیه با نسخه های توسعه یافته آنها جایگذاری می شوند. تعداد از قواعد بزرگتر می شود زیرا دنباله چگالتر می شود و قواعد زاید ایجاد می شوند. برای جلوگیری از ایجاد قواعد بی فایده در ابتدا اجداد(ریشه ها) به ازای هر عنصر پیش محاسبه می شوند و آنهایی که در کاندید ها نیستند حذف می شوند. علاوه بر این الگوریتم تعداد الگوهای دنباله ای را که شامل خود عنصر و اجدادش هستند را نمی شمارد.

SPADE

Spade یک الگوریتم پیشنهاد داده شده برای یافتن دنباله های مکرر با استفاده از تکنیک های شبکه ای بهینه و اتصالات ساده است. تمام الگوها می توانند با بررسی سه باره دیتابیس کشف شوند. این روش مسیله استخراج را برای حل کردن به مسایل کوچکتری تقسیم می کند و در همان حال این را ممکن می کند که تمام داده های مورد نیاز در حافظه یافت شود. الگوریتم spade که بر اساس ایده ی اکلات توسعه یافته است به حد زیادی بهره وری استخراج الگوی دنباله ای را افزایش داده است.

شکل 8و3

ایده کلیدی spade به این گونه می تواند توضیح داده شود. دیتابیس دنباله ای در ابتدا تبدیل به یک دیتابیس عمودی با فورمت لیست شناسه ای تبدیل می شود که در آن هر شناسه با دنباله و تراکنش های مشتری منطبق با خودش مرتبط می شود. نسخه عمودی دیتابیس اولیه (همان گونه که در شکل 3.3 نمایش داده شده است) در شکل 3.6 نمایش داده شده است برای نمونه ما می دانیم که لیست شناسه عنصرa، (100,1) ، (100,5) ،(100,6) ،(200,3) و (300,3) است به گونه ای که هر جفت (SID:TID) نشان دهنده ی دنباله و تراکنش خاصی است که a نشان می دهد. با بررسی دیتابیس عمودی دنباله های 1 مکرر به راحتی به دست می آید. برای یافتن دنباله های 2 مکرر ، دیتابیس اولیه دوباره بررسی می شود و دیتابیس افقی به عمودی جدید با دسته بندی کردن آن عنصرها با SID و به ترتیب افزایش TID ایجاد می شود که در شکل 3.7 نمایش داده شده است. با مرور دیتابیس الگوهای طول 2 می تواند کشف شود. یک شبکه بر اساس آن الگوهای طول 2 می تواند ایجاد شود و شبکه می تواند به کلاس های مختلفی تجزیه شود به گونه ای که الگوهایی که دارای پیشوند یکسانی هستند به کلاس یکسانی تعلق پیدا می کنند.این نوع از تجزیه این را ممکن می کند که بخش ها به اندازه کافی کوچک باشند تا درون حافظه بارگزاری شوند.SPADE سپس اتصالهای موقتی را برای یافتن تمام الگوهای بلندتر با شمردن شبکه به کار می برد.

دو استراتژی طی مسیر برای یافتن دنباله های مکرر در شبکه وجود دارد : breadth first search(BFS) ، depth first search(DFS) . برای DFS کلاس ها به صورت بازگشتی پایین به بالا ایجاد می شوند. برای نمونه برای ایجاد الگوهای طول 3 ، تمام الگوهای طول 2 باید به دست بیایند. بر خلاف آن DFS فقط نیاز به یک الگوی طول 2 و یک الگوی طولk برای ایجاد یک دنباله طول k+1 دارد. بنابراین همواره یک داد و ستد میان bfs و dfs وجود دارد : به صورتی که bfs نیاز به حافظه بیشتری برای ذخیره تمام الگوهای با طول 2 متوالی دارد ، این مزیت را دارد که اطلاعات بیشتری برای هرس کردن دنباله های طول k به دست می آید. تمام الگوهای طول k به صورت موقت یا اتصال تساوی الگوهای طول k-1 مکرر که پیشوند یکسان طولk-2 دارند کشف می شوند. تکنیک مشخصه هرس کردن apriori در spade به کار برده می شود.

شکل 3.8 یک نمونه از عملیات اتصال موقت را در spade نمایش می دهد. فرض کنید ما الگوهای طول 1 ،a و b را به دست آورده ایم. با اتصال این دو الگو ما می توانیم سه دنباله کاندید ab ، ba و (ab) را آزمایش کنیم.عملیات اتصال برای مقایسه زوج های sid , tid دو الگوی طول k-1 مورد نیاز است. برای نمونه الگوی b دو جفت {100,3} و {100,5} دارد که بزرگتر از یک جفت الگوی a است({100,1}) ، در دنباله همان مشتری.

بنابراین (ab) باید در همان دنباله وجود داشته باشد. به صورت مشابه ساپورت سایر دنباله های کاندید می تواند جمع آوری شود به همان صورت که در سمت راست تصویر 3.8 نمایش داده شده است.

شکل 9و3

SPAM

آیریس و همکاران الگوریتم spam را بر اساس ایده کلیدی spade پیشنهاد دادند. تفاوت در اینجاست که spam از یک نمایش بیت مپی از دیتابیس به جای جفت های {sid,tid} که در الگوریتم spade به کار برده می شود استفاده می کند. بنابراین spam با استفاده از عملگر های بیتی می تواند بسیار بهتر از spade و بقیه عمل کند.

هنگام بررسی دیتابیس برای بار اول ، یک بیت مپ عمودی برای هر عنصر در دیتابیس ایجاد می شود و هر بیت مپ یک معادل بیتی برای هر مجموعه عنصرهای دنباله ها در دیتابیس دارد. اگر یک عنصر در یک مجموعه عنصرها ظاهر شود ، معادل بیتی مجموعه عنصری بیت مپ برای عنصر برابر یک می شود ، در غیر این صورت بیت برابر صفر می گردد. اندازه یک دنباله تعداد مجموعه عنصرهای موجود در دنباله است. شکل 3.9 بیت مپ جدول عمودی موجود در شکل 3.5 را نمایش می دهد. یک دنباله در دیتابیس با سایز بین+1 تا به عنوان دنباله بیتی شناخته می شود. بیت مپ یک دنباله بر اساس بیت مپ آیتم های موجود در آن ایجاد می شود.

برای ایجاد و امتحان دنباله کاندید ، spamاز دو مرحله استفاده می کند ، مرحله S و مرحله I بر اساس مفهوم شبکه. به عنوان یک روش عمقی پروسه کلی از مرحله s شروع می شود و سپس مرحله I . برای توسعه یک دنباله ، مرحله s یک عنصر را به انتهای آن به عنوان آخرین عنصر جدید می افزاید ، و سپس مرحله I عنصر را در صورت امکان به آخرین عنصر آن می افزاید. هر بخش بیت مپ یک دنباله که می خواهیم توسعه بدهیم در ابتدا در مرحله s تبدیل می شود به گونه ای که تمام بیت های بعد از اولین بیت با مقدار یک ، یک مقدار دهی می شوند. سپس بیت مپ منتج شده از مرحله s با انجام عملیات AND برای بیت مپ تبدیل شده و بیت مپ عنصر افزوده شده به دست می آید.شکل 3.10 چگونگی اتصال دو الگو با طول 1،a و b را بر اساس دیتابیس مثال 3.5 نمایش می دهد. از طرف دیگر مرحله I فقط از بیت مپ های دنباله و عنصر های افزوده شده برای انجام عملیات And برای به دست آوردن بیت مپ نتیجه که الگوی (ab) را به کاندید (a(bc)) توسعه می دهد ، استفاده می کند. شمارش ساپورت تبدیل به یک بررسی ساده می شود که چه تعداد بخش های بیت مپ کلا صفر نیستند.

شکل 10و3

اشکال اصلی spam مصرف عظیم حافظه است. برای نمونه هر چند یک عنصر a در دنباله s وجود ندارد spam کماکان از یک بیت برای نمایندگی وجود آن استفاده می کند. این عیب مانع از تبدیل شدن SPAM به بهترین الگوریتم استخراج دیتابیس های بزرگ در شرایط منابع محدود می شود.

PrefixSpan

PrefixSpan در (201) پیشنهاد داده شد. ایده اصلی الگوریتم PrefixSpan به کار بردن نمایش دیتابیس برای کوچک کردن دیتابیس برای گذر بعدی و افزایش عملکرد است. مولفان ادعا کردند که در PrefixSpan هیچ نیازی برای ایجاد کاندید نیست . این روش به صورت بازگشتی دیتابیس را با الگوهای یافته شده کوتاه نمایش می دهد . متدهای نمایش مختلفی معرفی شدند مانند نمایش مرحله به مرحله ، نمایش دو مرحله ای و شبه نمایش.

مسیر کاری PrefixSpan به این صورت نمایش داده می شود . فرض کنید که عنصر ها در طول هر تراکنش به ترتیب الفبایی مرتب شده اند. مشابه با بقیه الکوریتم ها اولین مرحله PrefixSpan بررسی دیتابیس برای یافتن الگوهای با طول 1 است . سپس دیتابیس اولیه با توجه به الگوهای طول 1 مکرر و با فرض کردن الگوی مشابه به عنوان پیشوند به چندین بخش نمایش داده می شود. برای نمونه شکل 3.11 دیتابیس نمایش داده شده با لگوهای طول 1 به عنوان پیشوندشان را نشان می دهد. قدم بعدی بررسی دیتابیس نمایش داده شده y است به گونه ای که y می تواند هر کدام از الگوهای با طول 1 باشد. بعد از بررسی می توان الگوی با طول 1 مکرر را در دیتابیس نمایش داده شده به دست آورد. این الگوها در ترکیب با پیشوند مشترکشانy به عنوان الگوهای با طول 2 شناخته می شوند. پروسه به صورت بازگشتی اجرا می شود. به وسیله ی الگوهای با طول k جهت یافتن الگوهای طول k+1 تا زمانی که دیتابیس نمایش داده شده خالی باشد یا هیچ الگوی مکرر پیدا نشود دیتابیس نمایش داده شده بخش بندی می شوند.

استراتژی معرفی شده نمایش مرحله به مرحله نامیده می شود. هزینه اصلی محاسباتی زمان و مصرف فضا هنگام ایجاد و بررسی دیتابیس نمایش داده شده است همان گونه که در شکل 3.11 نشان داده شده است.

شکل 10و3

برای افزایش بهره وری استراتژی دیگری به نام نمایش دو مرحله ای جهت کاهش هزینه ساخت و بررسی دیتابیس نمایش داده شده پیشنهاد شد. تفاوت میان دو استراتژی نمایش در اینجاست که در دومین مرحله نمایش دو مرحله ای ، یک ماتریس مثلثی n\*n(s -matrix) با بررسی دوباره دیتابیس ایجاد می شود همان گونه که در شکل 3.11 نشان داده شده است. این ماتریس تمام ساپورتهای کاندیدهای طول 2 را نشان می دهد. برای نمونه

نشان می دهد که ساپورتهای ، و برابر 3 ،3 و 0 هستند. دیتابیس اولیه سپس با توجه به الگوهای طول 2 مکرر موجود در s -matrix نمایش داده می شود و دیتابیس نمایش داده شده بررسی می شود. این پروسه به صورت بازگشتی این عمل نمایش و بررسی را برای یافتن تمام الگوها دنبال می کند. البته این استراتژی همیشه به نظر بهینه نمی آید.

بهینه سازی پیشرفته تری به نام شبه نمایش برای به صرفه تر کردن نمایش هنگامی که دیتابیس نمایش داده شده می تواند در حافظه بارگزاری شود پیشنهاد شد. استراتژی با استفاده از یک جفت اشاره گر و آفست برای نشان دادن مکان هر دیتابیس نمایش به جای کپی کردن هر باره داده ها اجرا می شود. عیب این روش این است که اندازه دیتابیس نمی تواند خیلی بزرگ باشد.

به صورت خلاصه PrefixSan کارایی استخراج الگوهای دنباله ای را با استفاده از نمایش دیتابیس افزایش می دهد. این روش دیتابیس های کوچکتر نمایش داده شده را در هر گذر بررسی می کند. مشکل اصلی PrefixSan این است که در بررسی دیتابیس نمایش داده شده زمان گیر است که این زمان می تواند بسیار بزرگ باشد اگر دیتابیس اصلی عظیم باشد.