# PRÍLOHA C – Algoritmy dolovania uzavretých frekventovaných množín v prúde dát

V tejto prílohe uvádzame podrobnejší opis niektorých známych algoritmov dolovania uzavretých frekventovaných množín v prúde dát.

## Moment

Autori v práci . (Chi, Wang, Yu, & Muntz, 2004) navrhujú algoritmus hľadania uzavretých frekventovaných množín nad posúvajúcim sa oknom v prúde dát s názvom Moment. Je to prvá práca v ktorej navrhli inkrementálny prístup k dolovaniu uzavretých frekventovaných množín z prúdu dát. Okrem toho nájdené množiny nie sú len aproximáciou ale sú to exaktne nájdené uzavreté frekventované množiny aj napriek tomu, že algoritmus používa inkrementálne sa posúvajúce okno.

Autori navrhujú dátovú štruktúru CET (ang. closed enumeration tree), ktorá monitoruje uzavreté frekventované množiny a tiež tzv. okrajové množiny, ktoré tvoria hranicu medzi uzavretými frekventovanými množinami a ostatnými nefrekventovanými množinami. Každý uzol tejto štruktúry reprezentuje množinu *I*s atribútom aktuálnej podpory *sup(I).* Táto štruktúra je podobná prefixovému stromu *FP-tree* používaného v algoritme *FP-Growth* (pozri ...). Rozdiel je, že v strome neudržiava všetky množiny ale len frekventované a okrajové. Pokiaľ nedochádza k zmene stavu množiny, tak stačí inkrementovať hodnotu podpory pre zahrnuté podmnožiny. V CET sú 4 typy uzlov (teda stavov množín):

* je ***nefrekventovaný bránový uzol*** ak množina I, ktorú reprezentuje je nefrekventovaná. Rodičovský uzol je frekventovaný a množina I vznikne spojením rodičovskej množiny J s jedným zo súrodeneckých uzlov J’.
* je ***nenádejný bránový uzol*** ak I je frekventovaná množina a existuje uzavretá frekventovaná množina J taká, že
* je ***prechodný uzol*** ak *I* je frekventovaná množina a má ako potomka uzol taký, že nie je nenádejný bránový uzol.
* je ***uzavretý uzol*** reprezentujúci uzavretú frekventovanú množinu v aktuálnom okne. Môže to byť vnútorny uzol aj listový uzol stromu.

V práci ďalej dokazujú tieto tvrdenia o uzloch:

1. Ak je ***nefrekventovaný bránový uzol*** tak akýkoľvek uzol kde reprezentuje nefrekventovaný uzol (Vyplýva zo známej *apriori* vlastnosti). Tieto uzly tak môžu byť odstránené (Takto sa výrazne znižuje počet uzlov v štruktúre).

Uzol predstavuje okrajový uzol, ktorý si algoritmus „pamätá” a ak sa stane frekventovaným tak vie z neho nájsť ďalšie frekventované množiny.

1. Ak je ***nenádejný bránový uzol*** potom nie je uzavretý uzol a žiadny z jeho potomkov nie je uzavretý. Preto sú potomkovia uzla odrezaný, pretože uzol si “pamätá” hranicu a vieme z neho odvodiť ďalšie neuzavreté frekventované množiny.
2. Ak je ***prechodný uzol*** tak nie je uzavretý a má potomkov uzavreté uzly.

Uzol je teda štruktúrou :

* Informácia o type
* Množina *I*
* Podpora pre množinu *I – support(I)*
* Suma identifikátorov transakcií v ktorých sa *I*nachádza (*tid\_sum(I))*

Na urýchlenie častej operácie kontroly, či daný uzol je alebo nie je nenádejný bránový uzol, navrhli použiť hash tabuľku so všetkými uzavretými frekventovanými množinami. Kľúčom je dvojica (*support(I)*, *tid\_sum(I)*).

Štruktúra CET sa inkrementálne aktualizuje vychádzajúc z aktuálneho okna. Aktuálne okno je reprezentované štruktúrou FP-tree (ako sme ju opísali v časti ) s tým rozdielom, že obsahuje všetky množiny aj tie nefrekventované a navyše obsahuje tabuľku ukazovateľov TID na koncový uzol korešpondujúci s danou transakciou uloženou v strome.

Procedúra s názvom *Explore(, min\_support)* prehľadáva podstrom CET štruktúry do hĺbky, aktualizuje a označuje typy uzlov. Podprocedúra *leftcheck()* vie určiť či je ***nenádejný bránový uzol*** tak, že zisťuje v hash tabuľke, či predtým už existovala uzavretá množina s rovnakou podporou ako a obsahujúca *I.*

Pre každú prichádzajúcu transakciu T algoritmus aktualizuje hodnotu podpory a môže zmeniť typ korešpondujúceho uzla v CET štruktúre.

* Ak bol ***nefrekventovaný bránový uzol***. A teraz sa stáva frekventovaným tak pre každého ľavého súrodenca skúma či má byť vytvorený nový potomok s množinou obsahujúcou podmnožinu *I*. Keďže bol okrajový uzol tak musí byť ďalej skúmaný aj jeho celý odrezaný podstrom pomocou procedúry *Explore.*
* Ak bol ***nenádejný bránový uzol*** tak sa môže stať “nádejným”. Ako nádejný uzol označujú autori uzol, ktorý v CET nachvíľu nie je ani ***nefrekventovaný bránový uzol*** ani ***nenádejný bránový uzol***. V tom prípade treba preskúmať jeho odrezaný podstrom pomocou procedúry *Explore.*
* Ak bol ***prechodný uzol*** taksa môže stat ***uzavretým***,Ak T obsahuje *I*ale žiadny z potomkov , ktoré majú rovnakú podporu ako pred príchodom T.
* Ak bol ***uzavretý uzol*** tak zostane uzavretý.

Pre každú odchádzajúcu transakciu T algoritmus aktualizuje hodnotu podpory a môže zmeniť typ korešpondujúceho uzla v CET štruktúre.

* Ak bol ***nefrekventovaný bránový uzol*** tak sa nezmení.
* Ak bol ***nenádejný bránový uzol*** môže sa stať nefrekventovaným a teda bude odstránený.
* Ak bol ***nádejný uzol*** tak sa môže stať nenádejným (ak T obsahuje *I*). Použije sa procedúra *leftcheck().*
* Ak bol ***uzavretý uzol*** tak môže prestať byť uzavretým uzlom. Zistíme to kontrolou hodnôt podpory potomkov tohto uzla.

## CLAIM

## Citované práce

Chi, Y., Wang, H., Yu, P. S., & Muntz, R. R. (2004). Moment: Maintaining closed frequent itemsets over a stream sliding window. *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on* (s. 59-66). IEEE.