*Majalah Ilmiah Teknologi Elektro, Vol. 17, No. 2,Mei - Agustus 2018*
*DOI: https://doi.org/10.24843/MITE.2018.v17i02.P06*


# Implementasi Algoritma *FP-Growth* dengan *Closure Table* untuk Penemuan *Frequent Itemset* pada Keranjang Belanja

I Gusti Agung Indrawan[1], Made Sudarma[2], Lie Jasa[3]

***Abstract*— FP-growth algorithm is a data mining algorithm used to find frequent itemset in market basket data. Frequent itemset is a group of items that are often purchased together in one market basket. Analysis of frequent itemset will result in association rules. FP-growth algorithm finds frequent itemset by compressing market basket data into a tree structure called FP-tree. FP-tree is then analyzed to extract frequent itemset. Market basket data is always increasing for every transaction that occurs, so the process of mining frequent itemset will re-create FP-tree from scratch repeatedly every time FP-growth algorithm executed. In order for FP-tree not to be re-created from the beginning every time FP-growth algorithm executed, FP-tree needs to be stored on storage media using format suitable for tree structure. This research stores FP-tree to table in database using closure table structure. The structure of closure table has several advantages that are suitable for storing tree structure. The results obtained from storing FP-tree to table in database using closure table structure is FP-tree that has been stored in database can be analyzed repeatedly without needing to be re-created from scratch, and only updated when market basket data increases.**

***Keywords*— *closure table, data mining,* FP-*growth,* FP-*tree, frequent itemset, market basket*.**

***Intisari*— Algoritma *FP-growth* adalah algoritma *data mining* yang digunakan untuk menemukan *frequent itemset* pada data keranjang belanja. *Frequent itemset* adalah kelompok barang yang sering dibeli bersamaan dalam satu keranjang belanja. Analisa *frequent itemset* menghasilkan aturan asosiasi. Algoritma *FP-growth* menemukan *frequent itemset* dengan mengkompresi data keranjang belanja ke struktur data *tree* yang disebut *FP-tree. FP-tree* kemudian dianalisa untuk mengekstrak *frequent itemset*. Data keranjang belanja selalu bertambah jumlahnya untuk setiap transaksi yang terjadi, sehingga proses penemuan *frequent itemset* akan membuat *FP-tree* dari awal secara berulang-ulang setiap kali algoritma *FP-growth* dijalankan. Agar *FP-tree* tidak perlu dibuat ulang dari awal setiap kali algoritma *FP-growth* dijalankan, maka *FP-tree* perlu disimpan pada media penyimpanan menggunakan format yang sesuai dengan struktur data *tree*. Penelitian ini menyimpan *FP-tree* ke tabel pada *database* dengan struktur *closure table*. Struktur *closure table* memiliki beberapa keunggulan sehingga cocok digunakan untuk menyimpan struktur data *tree*. Hasil yang didapatkan dari penyimpanan *FP-tree* ke tabel pada *database* menggunakan struktur *closure table adalah FP-tree* yang telah tersimpan pada *database* dapat dianalisa berulang kali tanpa perlu dibuat dari awal, dan hanya di-*update* ketika data keranjang belanja bertambah.**

[1]*Mahasiswa Magister Teknik Elektro, Universitas Udayana, Kampus Universitas Udayana Jl. P.B. Sudirman Denpasar Bali 80232 (tlp: 0361-239599; e-mail: agung.indrawan@gmail.com)*
[2, 3] *Dosen Magister Teknik Elektro, Universitas Udayana, Kampus Universitas Udayana JL. P.B. Sudirman Denpasar Bali 80232 (tlp: 0361-239599; e-mail: [2]msudarma@unud.ac.id, [3]liejasa@unud.ac.id)*

***Kata Kunci*— *closure table, data mining,* FP-*growth,* FP-*tree, frequent itemset,* keranjang belanja.**

## I. PENDAHULUAN

Usaha *retail* yang menggunakan sistem *point-of-sales* menyimpan data transaksi penjualan barang dalam jumlah besar. Data transaksi penjualan barang per pelanggan disebut data keranjang belanja. Data transaksi yang tersimpan dalam jumlah besar ini berpotensi memiliki *knowledge* yang belum diketahui. Sebagai contoh adalah *knowledge* tentang barang yang sering dibeli secara bersamaan dalam satu transaksi, yang dalam istilah *data mining* disebut *frequent itemset. Knowledge* tersebut dapat digunakan oleh *retailer* untuk meningkatkan penjualan, misal dengan membuat promosi bundel barang-barang tertentu dan meletakkan barang yang sering dibeli secara berdekatan pada rak penjualan. Penelitian untuk menemukan *frequent itemset* pada keranjang belanja sebelumnya dilakukan oleh [1].

Algoritma *FP-growth* menyimpan data keranjang belanja ke struktur data *tree* yang disebut FP-*tree*. Algorima FP-*growth* yang dijabarkan oleh [2] dan [3] membangun dan menyimpan *FP-tree* pada *random access memory* (RAM). Kelemahan penyimpanan *FP-tree* pada RAM adalah *FP-tree* akan terhapus ketika program dihentikan atau komputer dimatikan. Solusi dari masalah tersebut adalah menyimpan *FP-tree* yang telah di-*generate* ke media penyimpanan data pada komputer. Struktur *closure table* adalah solusi yang elegan untuk menyimpan hierarki *tree* pada *database*. *Closure table* menyimpan seluruh *path* pada *tree* untuk *node* tertentu, tidak hanya *node* dengan relasi *parent-child* [4]. Keunggulan struktur *closure table* adalah *query* untuk menemukan seluruh *child node* untuk *parent node* tertentu hanya memerlukan satu kali *query*.

Penggunaan *database* untuk menyimpan *FP-tree* menggunakan struktur *closure table* diharapkan menghilangkan keterbatasan algoritma *FP-growth* dimana ukuran *FP-tree* dapat melebihi kapasitas memori utama yang tersedia pada komputer, terutama pada *dataset* keranjang belanja berukuran besar.

## II. STUDI LITERATUR

### *A. Data Mining*

Istilah *data mining* dapat dijelaskan sebagai proses menemukan pengetahuan (*knowledge*) atau pola (*pattern*) dari data berukuran besar [5]. Seiring dengan meningkatnya komputerisasi kegiatan masyarakat, kapasitas penyimpanan data yang semakin besar dan kemampuan *processing* komputer yang semakin cepat, mengakibatkan data yang dihasilkan dan disimpan oleh perorangan maupun organisasi tiap harinya sangatlah besar. Kegiatan sehari-hari seperti transaksi pembelian yang diproses dengan terminal *point-of-*


p-ISSN:1693 – 2951; e-ISSN: 2503-2372

*sales*, penggunaan situs media sosial untuk melakukan *sharing* berita dan foto serta video, adalah contoh kegiatan yang menghasilkan data dalam jumlah besar.

Jumlah data yang sangat besar yang tidak dibarengi dengan wawasan (*insight*) yang cukup (*knowledge*) terhadap data tersebut disebut *data rich but information poor*. Hal ini diakibatkan tidak terpenuhinya kebutuhan terhadap alat analisis data yang handal. Situasi inilah yang kemudian memunculkan teknologi *data mining*.

### *B. Aturan Asosiasi*

*Frequent patterns* adalah pola yang sering muncul pada *dataset*. *Frequent patterns* dapat berupa *itemset, subsequences* atau *substructures.* Kumpulan barang yang sering muncul bersamaan pada satu transaksi data, sebagai contoh susu dan roti yang sering dibeli bersamaan pada satu transaksi penjualan barang, disebut *frequent itemset* [6]. Penemuan *frequent itemset* pada data transaksi keranjang belanja memungkinkan perusahaan *retail* mengoptimalkan penjualan barang, misal dengan meletakkan barang-barang yang sering dibeli bersamaan pada rak yang berdekatan atau mengadakan promosi barang bundel.

Pada contoh sebelumnya, disebutkan susu dan roti sering dibeli bersamaan pada satu transaksi penjualan barang. Pola tersebut dapat dituliskan dalam bentuk aturan asosiasi seperti berikut:

$$\{Penghapus\} \rightarrow \{Pensil\} \quad (1)$$

Kekuatan aturan asosiasi dapat diukur menggunakan *support* dan *confidence*. *Support* menentukan seberapa sering aturan asosiasi $X \rightarrow Y$ berlaku pada suatu dataset [6]. *Support* dapat didefinisikan sebagai berikut:

$$Support, s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N} \quad (2)$$

*Confidence* menentukan seberapa sering item Y muncul pada transaksi yang mengandung item X [6]. *Confidence* dapat didefinisikan sebagai berikut:

$$Confidence, c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} \quad (3)$$

### *C. Algoritma FP-growth*

Algoritma *FP-growth* (FP: *frequent-pattern*) mengkodekan data menggunakan struktur data yang disebut *FP-tree* dan mengekstrak *frequent itemset* dari *FP-tree* tersebut. *FP-tree* adalah representasi terkompresi dari data keranjang belanja. *FP-tree* dibangun dengan membaca *dataset* transaksi per satu transaksi dan melakukan *mapping item* tiap transaksi tersebut ke sebuah *path* pada *FP-tree* [6]. Dikarenakan setiap transaksi berbeda dapat memiliki beberapa item transaksi yang sama, *path* item transaksi yang sama tersebut akan *overlap* pada *path* di *FP-tree*. Semakin banyak *path* yang *overlap* pada *FP-tree*, semakin besar kompresi data yang bisa dicapai.

Penelitian sebelumnya tentang implementasi algoritma *FP-growth* antara lain:

1. Referensi [7] membahas tentang optimasi algoritma *FP-growth* yang dinamakan APFTC (Apriori with *FP-tree and Correlation*), dimana *FP-tree mining* untuk menemukan *frequent itemset* menggunakan algoritma *apriori*. Algoritma APFTC mengenalkan konsep korelasi untuk meningkatkan kualitas *interestingness* dari aturan asosiasi yang ditemukan.
2. Referensi [8] membahas tentang algoritma *FP-growth* yang dieksekusi menggunakan Apache Hadoop. Apache Hadoop adalah *framework* yang memungkinkan pemrosesan *datasets* berukuran besar secara paralel pada komputer terkluster. Penelitian ini menggunakan implementasi *FP-growth* dari Apache Mahout. Apache Mahout adalah *library* untuk mengembangkan aplikasi *machine learning*.
3. Referensi [9] membahas tentang optimasi algoritma *FP-growth* yang dinamakan algoritma *painting-growth* dan *n-painting-growth*.

### *D. Closure Table*

Struktur *closure table* menyimpan seluruh *path* pada *tree* untuk *node* tertentu, tidak hanya *node* dengan relasi *parent-child* [4]. Keunggulan struktur *closure table* adalah kemudahan melakukan *query* seluruh *child node* untuk *parent node* tertentu, sehingga cocok digunakan sebagai struktur tabel untuk menyimpan struktur data *FP-tree* yang digunakan algoritma *FP-growth*. Struktur *closure table* diimplementasikan sebagai tabel tambahan. Tabel tersebut bernama *treepaths* dan didefinisikan dengan SQL *query* sebagai berikut:

```
CREATE TABLE Treepaths (
  ancestor BIGINT UNSIGNED NOT NULL,
  descendant BIGINT UNSIGNED NOT NULL,
  PRIMARY KEY(ancestor, descendant),
  FOREIGN KEY (ancestor) REFERENCES trx(id),
  FOREIGN KEY (descendant) REFERENCES trx(id)
);
```

Tabel *treepaths* mereferensi kolom id yang merupakan *primary key* dari tabel trx (transaksi/*market basket*). Salah satu ciri struktur *closure table* adalah setiap *node* mereferensi dirinya sendiri. Setiap *node* juga menyimpan referensi/*path* ke semua *child node* yang ada dibawahnya. Tabel 1 menunjukkan data tabel nama barang, sedangkan tabel 2 menunjukkan data tabel *treepaths*.

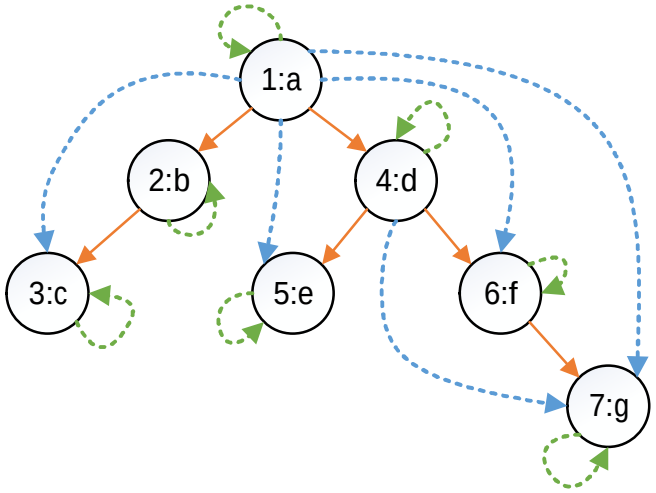
Visualisasi data yang ada pada tabel 2 ditunjukkan di gambar 1. Garis biru menandakan referensi *foreign keys* untuk kolom *ancestor* dan *descendant* pada tabel *treepaths*. Garis hijau menandakan referensi *foreign keys* ke *node* itu sendiri.

Tabel 1: NAMA BARANG

| id | nama |
|---|---|
| 1 | a |
| 2 | b |
| 3 | c |
| 4 | d |
| 5 | e |
| 6 | f |
| 7 | g |

Tabel 2: DATA PADA TABEL *TREEPATHS*

| ancestor | descendant | ancestor | descendant |
|---|---|---|---|
| 1 | 1 | 3 | 3 |
| 1 | 2 | 4 | 4 |
| 1 | 3 | 4 | 5 |
| 1 | 4 | 4 | 6 |
| 1 | 5 | 4 | 7 |
| 1 | 6 | 5 | 5 |
| 1 | 7 | 6 | 6 |
| 2 | 2 | 6 | 7 |
| 2 | 3 | 7 | 7 |




Gambar 1. Visualisasi *Tree*

## III. METODE

### A. *Alur Implementasi Algoritma FP-growth Menggunakan Closure Table*

Alur implementasi algoritma *FP-growth* dengan *closure table* dijelaskan sebagai berikut:

1. Data keranjang belanja yang berbentuk file teks di-transformasi agar dapat disimpan dalam bentuk tabel pada *database*.
2. Tentukan *minimum support count* untuk penemuan *frequent itemset*. Item yang tidak memenuhi *minimum support count*, dianggap *infrequent* dan tidak disertakan dalam proses *mining*. *Update* kolom is_min_sc pada tabel items dan tabel trx_details dengan nilai *true/false* sesuai dengan kondisi apakah frekuensi kemunculan barang pada *dataset* keranjang belanja memenuhi *minimum support count* atau tidak.
3. *Scan* tabel trx_detail untuk menentukan semua ID transaksi detail yang valid. ID transaksi detail dianggap valid ketika ID transaksi detail tersebut memiliki ID transaksi master dan frekuensi kemunculan barang pada transaksi detail tersebut pada *dataset* keranjang belanja memenuhi syarat *minimum support count*.
4. *Scan* tabel trx_detail untuk menemukan frekuensi kemunculan item pada keranjang belanja. *Update* jumlah frekuensi kemunculan item ke kolom freq pada tabel item.
5. Tahap berikutnya adalah pembuatan *FP-tree*. Item transaksi yang telah berurut secara *descending*, di-*mapping* ke tabel *treepaths* yang disimpan pada DBMS MySQL menggunakan struktur *closure table*.
6. Setelah *FP-tree* terbentuk, tahap berikutnya adalah ekstraksi *frequent itemset*.
7. *Frequent itemset* yang telah ditemukan, digunakan untuk membuat aturan asosiasi. Aturan asosiasi kemudian ditampilkan dalam bentuk laporan, beserta perhitungan *support* dan *confidence* aturan asosiasi tersebut.

### B. *Struktur Tabel di Database*

Struktur data *FP-tree* yang dihasilkan oleh algoritma *FP-growth* disimpan sebagai *closure table* pada *database* MySQL. Tabel yang menyimpan *FP-tree* pada *database* MySQL dinamakan tabel *treepaths* dan didefinsikan dengan SQL *query* sebagai berikut:

```
CREATE TABLE treepaths(
   id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
   ancestor BIGINT UNSIGNED NOT NULL,
   descendant BIGINT UNSIGNED NOT NULL,
   path_length INTEGER UNSIGNED NOT NULL,
   PRIMARY KEY (id));

ALTER TABLE treepaths ADD CONSTRAINT
fk_treepaths_anc_ref_items_id FOREIGN KEY (ancestor)
REFERENCES items (id_item);

ALTER TABLE treepaths ADD CONSTRAINT
fk_treepaths_des_ref_items_id FOREIGN KEY (descendant)
REFERENCES items (id_item);
```

Struktur tabel *treepaths* dapat dilihat pada tabel 3. Kolom id menyimpan *identity* dari tiap *node*, yaitu angka bulat (*integer*) yang membedakan tiap *node*. Kolom ancestor menyimpan *id node* yang merupakan *ancestor* dari *node* saat ini, dan kolom descendant menyimpan *id node* yang merupakan *descendant* dari *node* saat ini. Kolom path_length menyimpan jarak *node* antara *node ancestor* dan *descendant*.

Tabel 3: STRUKTUR TABEL *TREEPATHS*

| No | Nama Kolom | Tipe Data | Key |
|---|---|---|---|
| 1 | id | Big Integer Unsigned | PK, NN, AI |
| 2 | ancestor | Big Integer Unsigned | NN, FK |
| 3 | descendant | Big Integer Unsigned | NN, FK |
| 4 | path_length | Integer Unsigned | NN |

Selain tabel *treepaths*, terdapat tabel lain yang digunakan untuk menyimpan data transaksi penjualan barang. Nama dan struktur data tabel tersebut dapat dilihat pada tabel 4, 5, 6 dan 7.

Tabel 4: STRUKTUR TABEL TRX_MASTER

| No | Nama Kolom | Tipe Data | Key |
|---|---|---|---|
| 1 | tid | Big Integer Unsigned | PK |

Tabel 5: STRUKTUR TABEL TRX_DETAILS

| No | Nama Kolom | Tipe Data | Key |
|---|---|---|---|
| 1 | tid_detail | Big Integer Unsigned | PK |
| 2 | tid | Big Integer Unsigned | FK |
| 3 | id_item | Big Integer Unsigned | FK |
| 4 | is_min_sc | Boolean | - |


p-ISSN:1693 – 2951; e-ISSN: 2503-2372

Tabel 6: STRUKTUR TABEL ITEMS

| No | Nama Kolom | Tipe Data | Key |
|---|---|---|---|
| 1 | id_item | Big Integer Unsigned | PK |
| 2 | freq | Integer Unsigned | NN |
| 3 | is_min_sc | Boolean | - |

Tabel 7: Struktur Tabel items_name

| No | Nama Kolom | Tipe Data | Key |
|---|---|---|---|
| 1 | id_item | Integer | PK |
| 2 | Name | Varchar(255) | NN |

*C. Penyimpanan FP-tree ke Closure Table*

Algoritma penyimpanan *FP-tree* ke *closure table* dijelaskan sebagai berikut:

1. Langkah pertama adalah meng-*insert root node FP-tree* ke tabel *treepaths*. Pada algoritma *FP-growth*, *root node* dari *FP-tree* adalah *node* dengan data *null*, dan pada algoritma ini diimplementasikan dengan angka nol sebagai nilai kolom *ancestor* dan *descendant*. Kolom *path_length* berisi nilai nol karena baris data ini adalah *node* pertama.
2. *Item* pada keranjang belanja yang memenuhi syarat *minimum support count*, diurutkan secara *descending* berdasarkan jumlah kemunculan item tersebut pada *dataset* keranjang belanja.
3. *Item* keranjang belanja yang sudah terurut, di-*insert* ke tabel *treepaths* berdasarkan *pseudo code* sebagai berikut:

```
For j = 0 To jml_item_krnjng_blnj
    If j = 0 Then
        For k = 0 To jml_item_krnjng_blnj – 1
            idItem = TrxDetails.Rows(k).idItem
            ExecQuery(INSERT INTO treepaths(ancestor,
            descendant, path_length) VALUES(0, idItem,
            k+1))
        Next
    Else
        For k = j To jml_item_krnjng_blnj
            idItemAncstr = TrxDetails.Rows(j-1).idItem
            idItemDescndnt = TrxDetails.Rows(k-1).idItem
            ExecQuery(INSERT INTO treepaths(ancestor,
            descendant, path_length) VALUES(idItemAncstr,
            idItemDescndnt, k-j))
        Next
    End If
Next
```

*D. Ekstraksi Frequent Itemset Dari Closure Table*

Algoritma ekstraksi *frequent itemset* dari tabel *treepaths* dijelaskan sebagai berikut:

1. Scan tabel items untuk menemukan *item* yang memenuhi kriteria *minimum support count*. Untuk semua item yang memenuhi kriteria tersebut, dapatkan *ancestor*-nya menggunakan SQL *query* sebagai berikut:
   ```
   Select ancestor FROM treepaths
   WHERE descendant = idItem
   ```
2. Hasil dari *query* tersebut adalah *multiple paths* dimana tiap *path* berisi *item* yang dibeli bersamaan dengan *item* yang sedang diproses. Tiap *path* pada *multiple paths* dipisahkan dengan data *ancestor* = 0 (nol).
3. Langkah ketiga adalah memecah *multiple paths* menjadi *path* individual. Pemecahan *multiple paths* kemungkinan menghasilkan *path* duplikat yang memiliki urutan/*sequence* id item yang sama. Jumlah kemunculan *path* duplikat yang memiliki urutan/*sequence* id item yang sama tersebut disimpan sebagai *support count path* tersebut. *Path* yang duplikat kemudian dihapus sehingga tersisa satu *path* individual yang memiliki *support count* yang bersesuaian.
4. Langkah keempat adalah menemukan semua id item unik yang disimpan sebagai *end node* di *path* individual. Semua id item unik yang ditemukan disimpan dalam *list*.
5. Langkah kelima adalah melakukan iterasi untuk semua id item unik yang ada pada *list*, temukan *path* individual yang memiliki *end node* sama dengan id item tersebut. Tambahkan *support count* semua *path* individual yang memiliki *end node* sama dengan id item, sebagai *support count* id item tersebut. Tentukan juga *support count* untuk *node* lain pada *path* individual dengan *end node* sama dengan id item yang sedang diproses.
6. Bila *support count* id item ≥ *minimum support count*, maka id item tersebut *frequent*. Pada titik ini penemuan *frequent* 1-*itemset* sudah selesai.
7. Langkah ketujuh adalah penemuan *frequent* 2-*itemset*. Pada tahap 5, *support count* untuk *node* lain pada *path* individual dengan *end node* sama dengan id item yang sedang diproses sudah dihitung. Bila *support count node* lain tersebut ≥ *minimum support count*, maka node lain tersebut adalah *frequent* dan anggota *frequent* 2-*itemset* dari id item yang sedang diproses.

## IV. HASIL DAN PEMBAHASAN

Pembahasan implementasi algoritma *FP-growth* menggunakan *closure table* untuk penemuan *frequent itemset* menggunakan contoh data keranjang belanja alat tulis yang dapat dilihat pada tabel 8 dan tabel 9.

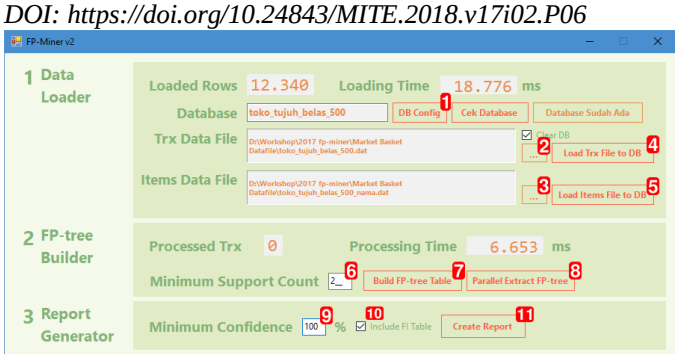
Tabel 8: Tabel item_name Alat Tulis

| id_item | nama |
|---|---|
| 1 | Pensil |
| 2 | Bolpoin |
| 3 | Buku Tulis |
| 4 | Penghapus |
| 5 | Rautan |
| 6 | Penggaris |
| 7 | Kertas |

Tabel 9: Data Keranjang Belanja Alat Tulis

| tid | id_item |
|---|---|
| 1 | 2 3 1 |
| 2 | 2 3 |
| 3 | 3 1 4 5 |
| 4 | 7 1 6 |
| 5 | 2 1 4 6 |
| 6 | 1 5 4 |

Tampilan program implementasi algoritma *FP-growth* dengan *closure table* yang diberi nama *FP-miner* dapat dilihat pada gambar 2.




Gambar 2. Tampilan Program *FP-miner*

Tahapan penggunaan program *FP-miner* dijelaskan sebagai berikut:

1. Langkah pertama adalah melakukan konfigurasi koneksi *database* MySQL.
2. Langkah kedua adalah melakukan transformasi data dari data keranjang belanja yang berupa file teks menjadi *record* pada *database*. Langkah transformasi data ini dilakukan di bagian (1 Data Loader) pada program *FP-miner*.
3. Langkah ketiga adalah membuat *FP-tree* yang akan disimpan pada tabel *treepaths* pada *database* MySQL. Langkah pembuatan *FP-tree* ini dilakukan di bagian (2 FP-tree Builder) pada program *FP-miner*. Penentuan *minimum support count* dilakukan pada langkah ini. *FP-tree* yang dihasilkan berdasarkan data keranjang belanja alat tulis pada tabel 9 disimpan pada tabel *treepaths* dan ditunjukkan pada tabel 10.
4. Langkah keempat adalah melakukan ekstraksi *frequent itemset* dari tabel *treepaths*. Langkah ini dilakukan dengan mengklik tombol (Extract FP-tree).
5. Langkah kelima adalah pembuatan laporan aturan asosiasi. Pembuatan laporan ini dilakukan dibagian (3 Report Generator) pada program *FP-miner*. Penentuan *minimum confidence* aturan asosiasi yang akan dilaporkan ditentukan pada tahap ini. Bila *minimum confidence* dikosongkan, maka program akan melaporkan semua aturan asosiasi yang ditemukan. Opsi (Include FI Table) bila dicentang maka pada laporan aturan asosiasi akan ditampilkan data *frequent itemset* yang ditemukan saat proses ekstraksi *frequent itemset* dari *FP-tree*. Tabel 11 menunjukkan *frequent itemset* untuk data keranjang belanja alat tulis.

Tabel 10: Tabel *Treepaths* Alat Tulis

| id | ancestor | descendant | path_length |
|---|---|---|---|
| 1 | 0 | 1 | 1 |
| 2 | 0 | 2 | 2 |
| 3 | 0 | 3 | 3 |
| 4 | 1 | 1 | 0 |
| 5 | 1 | 2 | 1 |
| 6 | 1 | 3 | 2 |
| 7 | 2 | 2 | 0 |
| 8 | 2 | 3 | 1 |
| 9 | 3 | 3 | 0 |
| ... | ... | ... | ... |
| 52 | 1 | 4 | 1 |
| 53 | 1 | 5 | 2 |
| 54 | 4 | 4 | 0 |
| 55 | 4 | 5 | 1 |
| 56 | 5 | 5 | 0 |

Tabel 11: *Frequent Itemset* Keranjang Belanja Alat Tulis

| ***Suffix*** | ***Frequent Itemset*** |
|---|---|
| Penggaris | {Penggaris:2}<br>{Pensil,Penggaris:2} |
| Rautan | {Rautan:2}<br>{Penghapus,Rautan:2}<br>{Pensil,Rautan:2}<br>{Pensil,Penghapus,Rautan:2} |
| Penghapus | {Penghapus:3}<br>{Pensil,Penghapus:3} |
| Buku Tulis | {Buku Tulis:3}<br>{Bolpoin,Buku Tulis:2}<br>{Pensil,Buku Tulis:2} |
| Bolpoin | {Bolpoin:3}<br>{Pensil,Bolpoin:2} |
| Pensil | {Pensil:5} |

6. Laporan aturan asosiasi yang ditemukan disimpan dalam format file html pada *folder* yang sama dengan *folder* dimana program *FP-miner* dijalankan. Laporan aturan asosiasi yang ditemukan pada data keranjang belanja alat tulis adalah sebagai berikut:

1. {Penggaris → Pensil}
   a. *Support* = 2/6 = 33,33%
   b. *Confidence* = 2/2 = 100%
2. {Rautan → Penghapus}
   a. *Support* = 2/6 = 33,33%
   b. *Confidence* = 2/2 = 100%
3. {Rautan → Pensil}
   a. *Support* = 2/6 = 33,33%
   b. *Confidence* = 2/2 = 100%
4. {Rautan,Penghapus → Pensil}
   a. *Support* = 2/6 = 33,33%
   b. *Confidence* = 2/2 = 100%
5. {Penghapus → Pensil}
   a. *Support* = 3/6 = 50%
   b. *Confidence* = 3/3 = 100%

## V. KESIMPULAN

Kesimpulan yang bisa didapatkan dari implementasi algoritma *FP-growth* menggunakan *closure table* adalah struktur data *FP-tree* yang digunakan oleh algoritma *FP-growth* sebagai representasi terkompresi data keranjang belanja, dapat disimpan sebagai tabel di-*database* menggunakan struktur *closure table*. Penambahan data keranjang belanja baru tidak mengakibatkan pembuatan ulang *FP-tree* dari awal, cukup data keranjang belanja baru saja yang disimpan ke tabel *treepaths* sehingga mempercepat proses penemuan *frequent itemset* secara keseluruhan.

## REFERENSI

[1] A. W. O. Gama, I. K. G. D. Putra, I. P. A. Bayupati, "Implementasi Algoritma Apriori untuk Menemukan *Frequent Itemset* Dalam Keranjang Belanja", Majalah Ilmiah Teknologi Elektro, vol.15, Juli-Desember 2016.
[2] R. Agrawal, et al, *Mining Association Rules between Sets of Items in Large Databases*. Proceedings of the 1993 ACM SIGMOD Conference Washington DC, USA. 1993.
[3] R. Agrawal and R. Srikant, *Fast Algorithm for Mining Association Rules*. Proceeding of the 20th VLDB Conference Santiago, Chile. 1994.
[4] B. Karwin, *SQL Antipatterns: Avoiding the Pitfalls of Database Programming 1st Edition*, Texas: The Pragmatic Bookshelf, 2010.
[5] J. Han and M. Kamber, *Data Mining Concept and Techniques, 3rd edition*, USA: Elsevier, Inc, 2012.
[6] P. N. Tan, et al, *Introduction to Data Mining*, USA: Addison-Wesley, 2005.
[7] S. Dandu, "Improved Algorithm for Frequent Item sets Mining Based on Apriori and FP-Tree", *Global Journal of Computer Science and Technology*., vol. 13, no 2-C, 2013.
[8] R. An, J. Pan, "Association Analysis of Large Sample Data Based on Hadoop", in Proc. 2015 *International Industrial Informatics and Computer Engineering Conference*, 2015.
[9] Y. Zeng, S. Yin, J. Liu, and M. Zhang, "Research of Improved FP-Growth Algorithm in Association Rules Mining", *Scientific Programming*., 2015.