PENERAPAN METODE ASSOCIATION RULE DENGAN ALGORITMA FOLD-GROWTH PADA DATA TRANSAKSI TOKO BUKU

SKRIPSI

Oleh :

YUDI ARESTA SANJAYA
NIM. 0710963051

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2014

PENERAPAN METODE ASSOCIATION RULE DENGAN ALGORITMA FOLD-GROWTH PADA DATA TRANSAKSI TOKO BUKU

SKRIPSI

Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana
Dalam bidang Ilmu Komputer

Oleh :

YUDI ARESTA SANJAYA
NIM. 0710963051

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2014

LEMBAR PERSETUJUAN

PENERAPAN METODE ASSOCIATION RULE DENGAN ALGORITMA FOLD-GROWTH PADA DATA TRANSAKSI TOKO BUKU

SKRIPSI

Oleh :

YUDI ARESTA SANJAYA
NIM. 0710963051 – 96

Telah diperiksa dan disetujui oleh :

Pembimbing I,	Pembimbing II,
Edy Santoso, S.Si., M.Kom	**Dian Eka R, S.Si., M.Kom**
NIP. 19740414 2001312 1 004	**NIP. 19730619 200212 2 001**

LEMBAR PENGESAHAN SKRIPSI

PENERAPAN METODE ASSOCIATION RULE DENGAN ALGORITMA FOLD-GROWTH PADA DATA TRANSAKSI TOKO BUKU

SKRIPSI

Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana
Dalam Bidang Ilmu Komputer

Disusun Oleh :
YUDI ARESTA SANJAYA
NIM. 0710963051 – 96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 11 Agustus 2014
dan dinyatakan memenuhi syarat untuk memperoleh
gelar Sarjana dalam bidang Ilmu Komputer

Penguji I,	Penguji II,
Budi Darma Setiawan, S.Kom., M.Cs	**Rekyan Regasari MP., ST., MT.**
NIK. 841015 06 1 1 0090	**NIK. 770414 06 1 2 1253**

Penguji III,

Wijaya Kurniawan, ST.,MT
NIK. 820125 16 1 1 0418

Mengetahui,
Ketua Program Studi Informatika / Ilmu Komputer

Drs. Marji, M.T.
NIP. 19670801 199203 1 001

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, Agustus 2014
Penulis

Yudi Aresta Sanjaya
NIM. 0710963051

KATA PENGANTAR

Bismillahirrohmanirrohim

Alhamdulillah, puji syukur penulis panjatkan kehadirat Allah SWT, karena hanya dengan rahmat, hidayah, anugerah, bimbingan serta ridho-Nya penulis dapat menyesaikan skripsi yang berjudul **" Penerapan Metode Association Rule Dengan Algoritma Fold-Growth Pada Data Transaksi Toko Buku "**. Sholawat serta salam tak lupa penulis haturkan kepada Nabi Muhammad SAW, keluarga serta sahabatnya yang telah mengiringi penulis dalam menyelesaikan skripsi.

Selama penyelesaian skripsi, tak sedikit penulis mendapatkan bantuan petunjuk, ilmu, bimbingan, do'a, motivasi serta saran dari berbagai pihak sehingga penulis dapat menyelesaikan skripsi ini. Oleh karena itu segala rasa terima kasih penulis sampaikan kepada yang terhormat :

1. Edy Santoso, S.Si., M.Kom selaku dosen pembimbing utama, atas kesediaan menjadi pembimbing, ilmu yang berharga, segala masukan, kesabaran dalam membimbing, nasehat dan motivasi yang telah diberikan.
2. Dian Eka R., S.Si., M.Kom selaku dosen pembimbing pendamping, atas segala kesabaran dalam membimbing, ilmu yang berharga, nasehat dan motivasi yang telah diberikan.
3. Kedua orang tua penulis. Ibu tercinta atas segala kasih sayang, do'a yang tak henti, nasehat, pelajaran hidup yang berharga dan segala keikhlasan yang diberikan. Almarhum Ayah, atas segala pelajaran hidup yang cukup singkat dan berharga.
4. Istri tercinta dan anak-anakku tersayang, yang telah mengiringi baik suka dan duka penulis dalam menyelesaikan studi.
5. Drs. Marji, M.T, selaku Ketua Program Studi Ilmu Komputer. Terima kasih atas segala bantuan dan kemudahan yang telah diberikan.
6. Segenap bapak dan ibu dosen dengan ikhlas dan sabar telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di

Program Studi Teknik Informatika Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.

7. Adik dan semua keluarga besar. Terima kasih atas do'a yang tak henti-henti, motivasi dan kasih sayang yang selalu mengiringi penulis.
8. Marissa H., Supardi, Khoirun Nissa dan Nanang Akhmad C. Anam terima kasih atas segala diskusi, motivasi, semangat, segala bantuan dan do'a yang selalu mengiringi penulis.
9. Teman – teman seperjuangan, ilkom 2007 dan *all* ilkomers. Atas do'a, bantuan, motivasi dan pengalaman selama perkuliahan.
10. Serta semua pihak yang tidak dapat penulis sebutkan satu per satu, terima kasih atas segala bantuan dalam penyelesaian skripsi.

Penulis berharap agar apa yang telah penulis buat dapat bermanfaat nantinya. Dan besar harapan penulis semoga skripsi ini dapat bermanfaat bagi penulis khususnya dan semua pihak yang berkepentingan.

Penulis menyadari bahwa masih ada ketidaksempurnaan selama penyelesaian skripsi ini. Pada kesempatan ini pula penulis memohon maaf atas segala ketidaksempurnaan yang ada. Dengan tangan terbuka penulis menerima kritik dan saran dari berbagai pihak bersifat membangun untuk bahan perbaikan dimasa yang akan datang,yudie.ariesta@gmail.com.

Malang, Agustus 2014

Penulis

ABSTRAK

Yudi Aresta Sanjaya. 2014 : Penerapan Metode Association Rule Dengan Algoritma Fold-Growth Pada Data Transaksi Toko Buku

Dosen Pembimbing : Edy Santoso, S.Si., M.Kom danDian Eka R., S.Si., M.Kom

Strategi bisnis dapat dilakukan dengan menganalisa data transaksi penjualan sehingga diperoleh informasi tersembunyi atau tidak diketahui sebelumnya. Data transaksi penjualan merupakan data yang berskala besar sehingga dibutuhkan teknik khusus yang disebut penggalian data (data mining).

Dalam penelitian ini, penggalian dilakukan terhadap data transaksi penjualan alat tulis dan buku toko buku Togamas, untuk mendapatkan pola asosiasi buku dan alat tulis berdasarkan golongan item dengan rentang bulan data transaksi yang bervariasi, yaitu per bulan, per dua bulan, dan triwulan. Sistem akan melakukan pencarian aturan asosiasi dengan tahap awal adalah mencari frequent itemset dengan menerapkan algoritma fold-growth. Kemudian dari frequent itemset tersebut akan dilakukan pencarian pola asosiasi dengan menggunakan teknik association rule. Pada tahap akhir, sistem akan menghitung kekuatan pola asosiasi yang terbentuk dengan menggunakan lift ratio.

Pada penelitian ini didapatkan nilai rata – rata lift ratio dari rule yang dihasilkan dengan minimum confidence 50% sebesar 1.14. Nilai lift ratio tertinggi sebesar 1.26 pada periode bulan Januari dan nilai lift ratio terendah sebesar 1.02 pada periode bulan Februari rentang bulan per bulan.

Kata Kunci : data mining, association rule, fold-growth, lift ratio

ABSTRACT

Yudi Aresta Sanjaya. 2014 : The Application of Rule Association using Fold-Growth Algorithm on Book Store Transactional Data

Dosen Pembimbing : Edy Santoso, S.Si., M.Kom danDian Eka R., S.Si., M.Kom

Businees strategic decision can be done by analyzing the sales data so that the hidden or unknown information can be revealed. The sales data is kept in large scale database format so special technique called data mining will be needed.

In this research, mining is done on sales (book and stationary) transaction data from Togamas book store to get the book and stationary association pattern based on item classification and several transaction data range i.e within a month, two months or three months. System will search the association rule with its first step would be searching for frequent itemset using fold-growth algorithm. Based on the frequent itemset found, system will search the association pattern using association rule technique. In its final steps, the system will calculate the formed association pattern strength using lift ratio algorithm.

After conducting the research using the minimum confidence value of 50% it is found that the resulting average lift ratio value is 1.14. The highest lift ratio value is 1.26 which is occurred at January and the lowest lift ratio value is 1.02 which is occurred on February. This results achieved by using one month transactional data.

Keyword : data mining, association rule, fold-growth, lift ratio

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SOURCECODE

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Salah satu tantangan yang dihadapi perusahaan adalah untuk mengetahui pola konsumsi pembelinya. Pelanggan merupakan aset berharga bagi suatu perusahaan dan strategi pemasaran yang terbaik dalam industri akan lebih menguntungkan bila perusahaan tetap menjaga dan memuaskan pelanggan (Popovic, 2009). Salah satu cara untuk dapat mengetahui pola konsumsi pembeli adalah dengan melakukan analisa terhadap data transaksi penjualan karena data transaksi tersebut merupakan data riil yang terlibat langsung terhadap pembeli. Dengan mengetahui pola konsumsi pembelinya, perusahaan akan mendapatkan informasi yang dapat mendukung keputusan (*decision support*) dalam penyusunan strategi yang berdasarkan data faktual.

Pada penelitian ini, akan dilakukan analisa terhadap data transaksi penjualan alat tulis dan buku. Data transaksi penjualan alat tulis dan buku itu sendiri merupakan data yang berskala besar. *Data mining* merupakan salah satu cara efektif yang dapat digunakan untuk mengetahui adanya serangkaian pola informasi dari sejumlah besar data yang ada. *Data mining* sendiri, memiliki berbagai macam metode dimana penggunaannya bergantung terhadap permasalahan yang dihadapi. Dalam permasalahan pencarian pola konsumsi pembeli alat tulis dan buku ini, dibutuhkan metode *data mining* yang disebut metode *association rule*. Metode *association rule* merupakan metode *data mining* yang dapat mencari korelasi antara *item-item* yang berbeda dengan mengetahui pola asosiasinya (Rochmah, 2010).

Dalam mendapatkan pola asosiasi, metode *association rule* terdiri dari dua tahap, yaitu tahap mencari kombinasi *item* yang paling sering terjadi (mencari *frequent itemset*) dan membangkitkan *rule* yang telah terbentuk dari *frequent itemset* (Han dan Kamber, 2001). Salah satu alternatif algoritma yang dapat digunakan untuk menentukan himpunan data yang paling sering muncul (*frequent itemset*) dalam sebuah kumpulan data adalah *Fold Growth*. Algoritma *Fold Growth* merupakan hasil gabungan dari algoritma *FOLDRAM* dan *FP-Growth*

(Rully dan Ni Made, 2006). Algortima *Fold Growth* menggabungkan keuntungan dari algoritma *FOLDRAM* yang memiliki kinerja cepat pada saat ukuran itemset frequent maksimum (k_{max}) adalah kurang dari sama dengan 10 ($k_{max} \leq 10$), dan dengan keuntungan dari algoritma *FP-Growth* yang memiliki kinerja yang cepat pada saat ukuran itemset frequent maksimum (k_{max}) adalah lebih dari 10 ($k_{max} > 10$) (Rully dan Ni Made, 2006).

Dari penelitian sebelumnya, yang dilakukan oleh Soelaiman Rully dan Ni Made Arini (2006) tentang Analisis Kinerja *Fold Growth* dan *FP-Growth* Pada Penggalian Pola Asosiasi, didapatkan hasil bahwa algoritma *fold growth* lebih efisien karena menghasilkan *node* lebih sedikit dalam pembuatan *tree* dan durasi eksekusi *Fold Growth* selalu lebih cepat dari algoritma *FP-Growth*.

Kelebihan dari algoritma *Fold Growth* juga dijelaskan oleh Soelaiman Rully dan Ni Made Arini (2006) antara lain durasi eksekusi, skalabilitas, reliabilitas, dan utilisasi memori menggunakan algoritma *Fold Growth* lebih baik daripada algoritma *FP-Growth*

Berdasarkan latar belakang yang telah dijelaskan tersebut, maka teknik *data mining* yang dipilih pada penelitian ini yaitu *association rule* menggunakan *algoritma Fold Growth* dalam pencarian pola penjualan alat tulis dan buku, dengan judul penelitian **"Penerapan metode *association rule* dengan algoritma *fold growth* pada data transaksi penjualan alat tulis dan buku"**.

1.2 Rumusan Masalah

Permasalahan yang dijadikan subyek penelitian dari skripsi ini adalah sebagai berikut:

1. Bagaimana menerapkan metode *association rule* dengan algoritma *fold growth* dalam mengetahui pola penjualan alat tulis dan buku.
2. Bagaimana tingkat kekuatan (*lift ratio*) *association rule* hasil penerapan algoritma *fold growth*.

1.3 Batasan Masalah

Adapun batasan masalah pada penelitian ini adalah sebagai berikut :

1. Penelitian dilakukan dengan menggunakan data transaksi penjualan alat tulis dan buku toko buku Togamas Malang pada tahun 2010.
2. Data keseluruhan dibedakan menjadi dua data yaitu data latih untuk mendapatkan *rule* atau pola penjualan alat tulis dan buku
3. Input yang dibutuhkan untuk sistem pelatihan antara lain data transaksi penjualan buku dan alat tulis toko buku Togamas Malang sebagai data latih dengan jumlah data transaksi tertentu, besar minimum *support*, dan besar minimum *confidence*.
4. Input yang dibutuhkan untuk sistem pengujian adalah data transaksi penjualan buku dan alat tulis toko buku Togamas Malang dengan jumlah data transaksi tertentu.
5. Output yang dihasilkan dari sistem pelatihan adalah aturan asosiasi (*association rule*) beserta nilai *lift ratio*.

1.4 Tujuan

Tujuan dari skripsi ini adalah :

1. Mengetahui pola penjualan alat tulis dan buku dengan menggunakan metode *association rule* menggunakan algoritma *fold growth.*
2. Mengetahui tingkat kekuatan *rule* hasil dari penerapan algoritma *fold growth*.

1.5 Manfaat

Manfaat yang diperoleh dari penelitian ini adalah untuk mengetahui pola penjualan alat tulis dan buku dan diharapkan untuk pemanfaatan lebih lanjut, informasi pola penjualan alat tulis dan buku tersebut dapat digunakan untuk mendukung pengambilan keputusan dalam penyusunan strategi.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Yang Relevan

Pada penelitian sebelumnya Penerapan Metode Association Rule Dengan Algoritma *FP-Growth* Pada Data Transaksi Penjualan Buku, dapat diketahui bahwasannya algoritma *FP-Growth* menggunakan 1 struktur data namanya *FP-tree*, sehingga data dibaca dari scanning database ketika pembentukan *FP-tree*. Sedangkan algoritma *Fold-Grwoth* menggunakan 2 struktur data *tree* yaitu *SOTrieIT* dan *FP-tree*, dalam prosesnya *SOTrieIT* dibuat untuk menemukan hingga *2-itemset*. Scanning database dilakukan cukup 1 kali karena data direkap langsung ke *SOTrieIT* (Marissa, 2012).

2.2 Data

Data adalah nilai yang merepresentasikan sebuah fakta atau gambaran dari suatu objek atau kejadian (Kusrini, 2007). Atribut pada data menunjukkan karakteristik objek dan kualitas yang menentukan proses *mining* bergantung pada datanya (Prasetyo, 2003). Salah satunya fakta dari transaksi perusahaan di bidang perdagangan dapat berupa transaksi penjualan yang meliputi waktu transaksi, pelaku transaksi, barang yang ditransaksikan beserta jumlah dan harganya. Dalam menyatakan data dapat berupa nilai yang berbentuk angka, deretan karakter, atau simbol (Kusrini, 2007).

2.3 Informasi

Informasi berguna dalam pengambilan keputusan karena memberikan tambahan pengetahuan sehingga meminimalkan ketidak pastian dari suatu masalah. Informasi merupakan representasi data yang telah diproses menjadi sesuatu yang berguna. Data yang diproses menjadi informasi mengalami proses manipulasi atau pengubahan data yang bertujuan untuk meningkatkan fungsi dari data itu sendiri (Kusrini, 2007).

2.4 Data Transaksi

Data transaksi memiliki fungsi sebagai bahan dasar obyektif (relatif) di dalam proses penyusunan kebijaksanaan dan keputusan oleh pimpinan perusahaan (Kotler dan Kevin, 2007). Pada suatu perusahaan, transaksi yang terjadi terdiri dari dua kelompok transaksi, yaitu (Kotler dan Kevin, 2007):

1. Transaksi ekstern.

   Transaksi ekstern merupakan transaksi yang terjadi dengan pihak luar perusahaan, seperti penjualan, pembelian, pengeluaran, dan penerimaan uang.

2. Transaksi intern.

   Transaksi intern merupakan transaksi yang terjadi untuk keperluan perusahaan atau pembagian kembali biaya–biaya dalam perusahaan, seperti depresiasiaktiva tetap, pemakaian bahan baku untuk produksi, transfer dari barang dalam proses ke barang jadi.

Dalam menganalisis data transaksi perusahaan, lebih mudah dilakukan terhadap data transaksi ekstern karena data transaksi ekstern sebagian besar memberikan dampak penambahan dan pengurangan data perusahaan (Kotler dan Kevin, 2007).

Pada penelitian ini data transaksi yang digunakan merupakan data transaksi ekstern yaitu data transaksi penjualan.

2.5 *Data Mining*

2.5.1 Pengertian *Data Mining*

Arti dari kata *mining* adalah usaha untuk memperoleh sesuatu yang berharga dari material dasar dalam jumlah besar, maka *data mining* dapat diartikan usaha untuk memperoleh sesuatu yang berharga (informasi) yang tersembunyi dari data dalam jumlah besar (Pramudiono, 2003). Menurut Han dan Kamber (2001), kandungan informasi yang tersembunyi yang didapatkan dalam proses *data mining* ini sangat menguntungkan dari sudut pandang penelitian, bisnis dan lainnya.

”Proses *data mining* bertujuan untuk menemukan hubungan, pola, dan tren baru yang bermakna dengan menyaring data yang sangat besar, yang tersimpan

dalam penyimpanan, menggunakan teknik pengenalan pola seperti teknik statistik dan matematika" (Kusrini, 2007).

Bidang ilmu lain yang berkaitan dengan *data mining* antara lain *database system*, *data warehousing*, statistik, *machine learning*, *information retrieval*, dan komputasi tingkat tinggi. Selain itu, *data mining* juga didukung oleh ilmu lain seperti *neural networ*k, pengenalan pola, *spatial data analysis*, *image database*, *signal processing* (Han, 2006).

2.5.2 Fungsi *Data Mining*

Tugas *data mining* diklasifikasikan dalam dua kategori yaitu deskriptif dan prediktif. *Data mining* secara deskriptif adalah untuk mengklasifikasikan sifat umum suatu data di dalam *database*, sedangkan *data mining* secara prediktif adalah untuk mengambil kesimpulan terhadap data dalam membuat prediksi (Han dan Kamber, 2001).

Pola pencarian yang dapat ditemukan dalam *data mining* memiliki fungsi antara lain deskripsi kelas, analisis *association*, klasifikasi dan prediksi, dan analisis *cluster*. Deskripsi kelas berguna untuk mengasosiasikan data dalam bentuk kelas sehingga menggambarkan kelas secara individual dan terkonsep secara tepat. Analisis *association* adalah penemuan aturan asosiasi yang menunjukkan nilai kondisi dari atribut yang terjadi secara bersama–sama dan terus–menerus dalam membentuk sekumpulan data. Klasifikasi bertujuan agar sekumpulan model yang menggambarkan dan membedakan kelas data, dapat digunakan untuk memprediksi kelas suatu obyek yang belum diketahui kelasnya dimana kelas yang diprediksi dapat berupa data nominal maupun data diskrit. Berbeda dengan klasifikasi yang digunakan untuk menganalisis obyek data dari kelas yang telah diketahui, *clustering* digunakan untuk menganalisis obyek data dengan kelas yang belum diketahui. *Outlier* merupakan obyek yang jaraknya jauh dari *cluster* yang lain dan dilakukan dengan cara pengambilan sebuah distribusi dan probabilitas model untuk data menggunakan tes yang bersifat statistik atau menggunakan ukuran jarak. Analisis *evolution* digunakan untuk mencari model atau obyek data yang memiliki kebiasaan berubah setiap waktu atau berkaitan dengan *time-series*. Analisis *evolution* dapat berupa *characterization*,

discrimination, association, classification, dan *clustering* (Han dan Kamber, 2001).

2.5.3 Tahap – Tahap *Data Mining*

Data mining merupakan bagian dalam proses *knowledge discovery in database* (KDD) (Kusrini, 2007). Proses *knowledge discovery in database* (KDD) secara umum adalah sebagai berikut (Kusrini, 2007) :

1. ***Data Selection***

   Tahap penyeleksian data dari sejumlah besar data operasional. Hasil dari seleksi data disimpan dalam suatu berkas terpisah dari *database* operasional.

2. ***Preprocessing* atau *Cleaning***

   Pada tahap ini, dilakukan pembuangan data yang duplikat, pemeriksaan data yang tidak konsisten, dan perbaikan kesalahan pada data, serta memperkaya data yang sudah ada dengan data atau informasi eksternal.

3. ***Transformation***

   Tahap ini bergantung pada jenis atau pola informasi yang akan dicari dalam basis data karena transformasi merupakan proses pengubahan sesuai format yang dibutuhkan untuk digunakan dalam proses *data mining* (Huda, 2010).

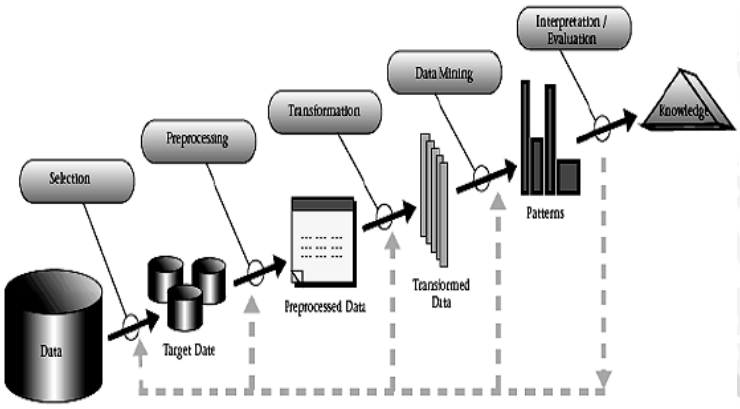
4. ***Data Mining***

   Pada tahap ini dilakukan pencarian pola atau informasi dari data yang terpilih dengan menggunakan metode atau teknik tertentu. Ketepatan metode atau teknik yang dipilih sangat bergantung pada tujuan dari proses ini.

5. ***Interpretation* atau *Evaluation***

   Hasil pola informasi dari proses *data mining* perlu ditampilkan dalam bentuk yang mudah dimengerti oleh pihak yang berkepentingan. Selain itu dilakukan pemeriksaan kesesuaian pola atau informasi yang ditemukan dengan fakta atau hipotesa yang ada sebelumnya.

Keseluruhan dari proses *knowledge discovery in database* (KDD) diilustrasikan pada Gambar 2.1.



Gambar 2.1 Tahap–tahap Proses *Knowledge Discovery in Database* (Han dan Kamber, 2001)

2.5.4 Teknik *Data Mining*

Teknik – teknik dalam *data mining* antara lain (Kantardzic, 2003) :

1. **Klasifikasi**

   Penemuan suatu fungsi yang dapat mengklasifikasikan data ke dalam salah satu kelas dari beberapa kelas yang telah ditentukan sebelumnya.

2. **Regresi**

   Penentuan suatu fungsi yang dapat memetakan data ke suatu nilai real dan variabel perkiraan

3. ***Clustering***

   Tugas deskriptif yang umumnya digunakan untuk menentukan himpunan berhingga dari kategori atau *cluster* untuk mendeskripsikan data.

4. ***Summarization* (Ringkasan)**

   Tugas deskripsi tambahan yang meliputi metode untuk menemukan deskripsi dari kumpulan data atau bagian dari kumpulan data.

5. ***Dependency Modeling* (Pemodelan Keterkaitan Antar Data)**

   Menemukan suatu model yang mendeskripsikan keterkaitan atau hubungan yang signifikan antara variabel atau nilai dari fitur – fitur dalam kumpulan data atau bagian dari kumpulan data.

6. ***Change and Deviation Detection* (Penemuan Perubahan dan Deviasi atau Simpangan)**

   Menemukan perubahan yang paling signifikan dalam kumpulan data.

2.6 *Association Rule*

Dalam pencarian hubungan antar item dalam suatu kumpulan data yang ditentukan, dibutuhkan suatu prosedur yang disebut *association rule*. Informasi yang diberikan dalam *association rule* berbentuk ”*if – then*” atau ”jika–maka” atau biasanya dikenal dengan istilah *antecedent* untuk mewakili bagian ”jika” dan *consequent* untuk mewakili bagian ”maka” (Santosa, 2007). *Rule* yang dihasilkan dari *association rule* menggambarkan isi dari basis data yang belum diketahui dan tidak terungkap secara eksplisit (Kantardzic, 2003).

Tugas dari *association rule* adalah mencari aturan yang tersembunyi untuk mengukur hubungan antara dua atau lebih atribut (Sholichah, 2009). *Association rule* membantu mengenali pola–pola tertentu di dalam kumpulan data yang besar (Ruldeviyani dan Muhammad, 2008).

Association rule meliputi dua tahap, yaitu mencari kombinasi yang paling sering terjadi dari suatu *itemset* dan membangkitkan *association rule* dari *frequent itemset* yang telah dibuat sebelumnya (Han dan Kamber, 2001).

Dalam *association rule* terdapat dua ukuran, yaitu *support* dan *confidence* (Han dan Kamber, 2001). *Rule* dikatakan *valid association rule* adalah yang memiliki *confidence* sama atau lebih dari *minimum confidence* (Ruldeviyani dan Muhammad, 2008). *Rule* dikatakan *strong rule* adalah memenuhi baik minimum *support* maupun minimum *confidence* (Handojo dkk, 2004).

2.6.1 *Support*

Nilai *support* digunakan untuk menentukan *frequent itemset*, sedangkan minimum *support* merupakan parameter yang digunakan sebagai batasan frekuensi kejadian atau jumlah *support* yang harus dipenuhi suatu kelompok data yang dijadikan aturan. *Itemset* yang nilai *support*-nya memenuhi parameter minimum *support* (min_sup) masuk dalam *frequent itemset* (Yulita, 2002).

Support dari aturan asosiasi adalah proporsi transaksi yang mengandung *antecedent* dan *consequent* (Rochmah, 2010). *Support* untuk aturan “X → Y” adalah probabilitas atribut atau kumpulan atribut X dan Y yang terjadi bersamaan dalam suatu transaksi (Yulita, 2002). Untuk menghitung nilai *support* suatu item dapat diperoleh dengan rumus pada Persamaan 2.1, sedangkan untuk menghitung

nilai *support* kombinasi item *antecedent* dan *consequent* digunakan rumus pada Persamaan 2.2 (Rochmah, 2010).

$$Support\ (A) = \frac{\text{Jumlah Transaksi yang Mengandung A}}{\text{Total Transaksi}} \quad (2.1)$$

$$Support\ (A, C) = P(A \cap C)$$
$$= \frac{\text{Jumlah Transaksi yang Mengandung A dan C}}{\text{Total Transaksi}} \quad (2.2)$$

2.6.2 *Confidence*

Ratio antara jumlah transaksi yang meliputi semua item dalam *antecedent* dan *consequent* dengan jumlah transaksi meliputi semua item dalam *antecedent* (Rochmah, 2010).Menurut Yulita (2002), *confidence* adalah probabilitas kejadian produk yang dibeli secara bersamaan dimana salah satu produk sudah pasti dibeli, misalnya jika terdapat n transaksi dimana X yang dibeli dan m transaksi dimana X dan Y yang dibeli bersamaan, maka *confidence* dari aturan X → Y adalah m dibagi n. *Association rule* yang nilai *confidence*-nya memenuhi parameter *threshold* minimum *confidence* (min_conf) termasuk dalam *strong association rule*. Minimum *confidence* adalah parameter yang mendefinisikan minimum *level* dari *confidence* yang dipenuhi oleh aturan yang berkualitas (Yulita, 2002). Untuk menghitung nilai *confidence* dapat diperoleh dengan rumus pada Persamaan 2.3 dan dijabarkan lebih khusus pada Persamaan 2.4 (Rochmah, 2010).

$$Confidence\ (A \rightarrow B) = \frac{\text{Support A} \cap \text{B}}{\text{Support A}} \quad (2.3)$$

$$Confidence\ (A \rightarrow B) = P(B|A)$$
$$= \frac{\text{Jumlah Transaksi yang Mengandung A dan B}}{\text{Jumlah Transaksi yang Mengandung A}} \quad (2.4)$$

2.6.3 Algoritma FOLDRAM

FOLDARM (*Fast Online Dynamic Association Rule Mining*) merupakan algoritma *data mining* yang menggunakan struktur data *SOTrieIT* yang memiliki kinerja cepat pada saat ukuran *itemset frequent* maksimumnya atau $k_{max} \leq 10$ (Soelaiman, 2006).

2.6.4 *FP-tree*

FP-tree adalah struktur data yang dibangun dengan pemetaan setiap data transaksi pada setiap lintasan tertentu, sehingga setiap transaksi yang memiliki item yang sama akan dapat dimampatkan dengan saling menimpa. Selain itu, kelebihan dari *FP-tree* adalah hanya membutuhkan pemindaian data transaksi sebanyak dua kali yang terbukti sangat efisien (Samuel, 2008). Menurut Han dan Kamber (2001), keuntungan struktur data *FP-tree*, selain lebih padat, juga tetap lengkap, artinya meskipun dilakukan penghapusan informasi yang tidak relevan dan penghilangan *item–item* yang tidak sering muncul akan tetapi kelengkapan informasi yang dibutuhkan untuk *frequent pattern mining* tetap terjaga.

FP-tree adalah sebuah *tree* yang terdiri dari satu *header* tabel, satu *root* yang diberi label "null", dan satu himpunan *item prefix subtree* sebagai node dari anak *root*. Masing–masing *entry* dalam *header table* adalah *frequent item*, dan setiap *record* terdiri dari dua atribut yaitu nama *item* dan *head of node-link*. Setiap node anak terdiri dari atribut : nama *item*, *count*, dan *node-link*. *Node-link* menghubungkan node ke node berikutnya pada *tree* tersebut yang mempunyai nama *item* yang sama atau *null* jika tidak ada (Febriyana, 2009).

Dalam *FP-tree* terdapat proses *conditional FP-tree* yaitu sama seperti membangun *FP-tree* namun dibangun dari pola dasar *conditional* (*conditional pattern base*). *Conditional pattern base* adalah himpunan *prefix path* dari node tiap item yang diperoleh dari penelusuran *node-link* (Febriyana, 2009).

Sebelum memulai membangun *FP*-tree, langkah awal yang perlu dilakukan adalah pembacaan basis data dan mendapatkan kumpulan *frequent 1-itemset* beserta nilai *support* kemudian dilakukan pengurutan berdasarkan nilai *support* secara *descending*. Membangun *FP-tree* dimulai dengan pembuatan *root* yang diberi nama *null*. Kemudian dilakukan pembuatan cabang pada *FP-tree*

untuk tiap transaksi yang telah diurutkan item transaksinya sesuai dengan urutan *frequent 1-itemset*. Setiap akan dilakukan pembuatan cabang baru dalam *FP-tree*, akan dilakukan penelusuran terlebih dahulu keberadaan node dengan item yang sama. Jika node sudah ada maka tidak perlu dilakukan pembuatan cabang baru melainkan dilakukan penambahan nilai *support* sebanyak 1 pada node yang sudah ada tersebut (Yiapanis, 2006).

Dalam proses pembuatan *FP-tree* juga dilakukan pembuatan tabel untuk melacak keberadaan setiap item ditempatkan dalam *FP-tree*. Setiap item yang sama pada cabang yang berbeda dihubungkan dengan *link* yang dinamakan *nodelink* (Yiapanis, 2006). *Pseudocode* dari algoritma pembentukan *FP-tree* dapat dilihat pada *Pseudocode* 2.1.

Masukan : Basis data Transaksi(D), dan nilai

minimum *support*(minsup)

Keluaran: FP-tree *Tree*;

Fungsi : FP-tree dibentuk dengan cara berikut

1. Transaksi di database (D) ditelusuri sekali agar didapat himpunan *frequent item*(F) dan nilai *support*nya.
2. Lalu F diurutkan mulai dari yang mempunyai nilai *support* yang terbesar sampai terkecil.
3. Buat *root* dari FP-tree(T) dan diberi nama null.
4. Untuk tiap transaksi Trans di D lakukan{
6. Panggil insert_tree([p|P],T);
7. Fungsi **insert_tree([p|P],T)** terdiri dari langkah – langkah berikut :
8. inisialisasi N = anak dari T
9. inisialisasi nama_item = nama dari item yang terdapat pada T
10. Jika (N.nama_item == p.nama_item){N.count ++;**}**
11. Jika tidak {buat node baru N;N.count = 1;N.*parent_link* =T; *N_link* = node yang sama nama *item*nya dihubungkan oleh struktur *node_link*}

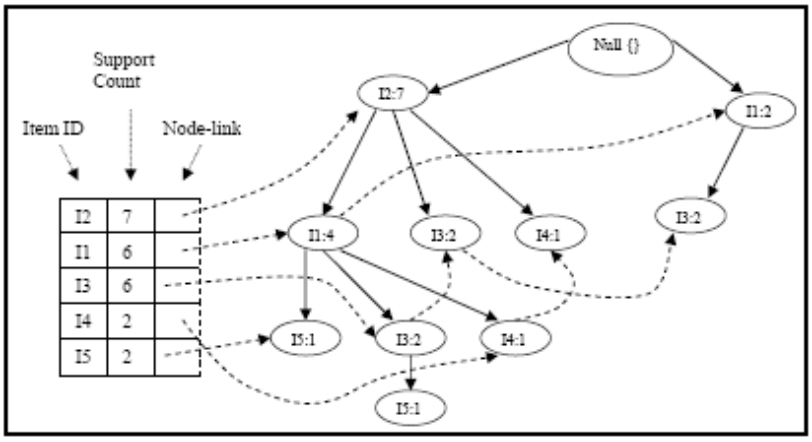
5. Pilih dan dilakukan pengurutan *frequent item* dalam Trans(disebut [p|P]) mengacu pada F dimana p adalah elemen pertama dan P adalah *item* berikutnya.	12. Jika P tidak kosong {*insert_tree*(P,N)}

***Pseudocode* 2.1** Pembentukan *FP-tree* (Yiapanis, 2006)

Penerapan algoritma pembuatan *FP-tree* adalah dimisalkan terdapat data transaksi pada Tabel 2.1 dengan *minimum support* adalah 2 maka hasil *frequent 1-itemset* adalah F = {(I2 :7), (I1 :6), (I3 :6), (I4 :2), (I5 :2)} karena item–item tersebut memiliki *support* lebih besar atau sama dengan *minimum support*. Hasil *FP-tree* setelah pembacaan semua transaksi dari Tabel 2.1 dapat dilihat pada Gambar 2.2 (Yiapanis, 2006).

Tabel 2.1 Contoh Data Transaksi (Yiapanis, 2006)

TID	Items
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3



Gambar 2.2 *FP-tree* (Yiapanis, 2006)

2.6.5 Algoritma *FP-Growth*

Algoritma *Frequent Patten Growth (FP–Growth)* adalah salah satu alternatif algoritma yang dapat digunakan untuk menentukan himpunan data yang paling sering muncul (*frequent itemset*) yang hanya memerlukan dua kali pemindaian basis data (Samuel, 2008). Dalam pencarian *frequent* itemset, algoritma *FP-Growth* menggunakan konsep pembangunan *tree* bukan membangkitkan semua kemungkinan kombinasi item sehingga memiliki waktu kompilasi yang lebih cepat. Struktur data *tree* yang digunakan algoritma *FP-Growth* disebut dengan *FP–tree*. Penggunaan *FP–tree* membuat algoritma *FP–Growth* dapat langsung mengekstrak *frequent itemset* (Erwin, 2009). Algoritma *FP-Growth* memiliki kinerja yang cepat pada saat $k_{max} \geq 10$ (Soelaiman, 2006).

Algoritma *FP– Growth* menggunakan *devide* dan *conquer*. *Devide* diartikan membagi suatu permasalahan besar menjadi permasalahan–permasalahan yang lebih kecil, sedangkan *conquer* diartikan pembagian dilakukan terus menerus sampai ditemukan bagian masalah kecil yang mudah untuk dipecahkan (Febriyana, 2009).

Dua fase yang dimiliki oleh algoritma *FP-growth* yaitu fase pembentukan *FP-tree* dan fase proses *mining FP-growth*. Langkah dari algoritma *FP-growth* dimulai dengan operasi pola *suffix* pada pola *frequent 1-itemset* dimana dibuat *conditional pattern base* untuk tiap *suffix*. *Conditional pattern base* adalah himpunan yang terdiri dari node pada lintasan awal yang disebut *prefix path* untuk

suffix tertentu. Setelah dilakukan proses *conditional pattern base,* selanjutnya dilakukan pembangunan *conditional FP-tree* yaitu proses *mining* secara rekursif. Kombinasi dari pola akhiran (*suffix*) dengan pola *frequent* yang dibangkitkan dari setiap *conditional FP-tree* dengan lintasan tunggal akan menjadi pola hasil dari algoritma *FP-growth* (Yiapanis, 2006).

Fase pembentukan *FP-tree* telah dijelaskan pada subbab *FP-tree*. Sedangkan fase dari proses *mining FP-growth* dijelaskan pada *Pseudocode* 2.2.

Masukan : *FP-tree* yang telah dibentuk sebelumnya pada langkah awal.
Keluaran : Sekumpulan dari pola *frequent*
Fungsi : panggil fungsi *fpGrowth(FP-Tree,null)*
Prosedur : *FP-Growth(Tree,α)*
{
1. jika Tree mengandung lintasan(P) tunggal;
2. maka untuk setiap kombinasi(ß) dari node-node dalam lintasan P lakukan
3. pembangkitan pola ß yang mengandung item α dengan support = minsup dari node – node dalam kombinasi(ß);
4. jika tidak untuk setiap a_i pada header Tree lakukan{
5. pembangkitan pola kombinasi (ß) = a_i yang mengandung item α dengan support =a_1.support;
6. bentuk conditional pattern base dari ß;
7. kemudian bentuk conditional FP-tree dari ß ($Tree_ß$);
8. jika $Tree_ß \neq \varnothing$ maka panggil FP-Growth(Tree,ß)}
}

***Pseudocode* 2.2** Algoritma *FP – Growth* (Yiapanis, 2006)

Dengan menggunakan permisalan pada data transaksi Tabel 2.1, operasi pola *suffix* dilakukan terhadap item yang memiliki nilai *support* terkecil dari pola *frequent 1-itemset*, yaitu I5. Pola dengan *suffix* I5 pada *FP-tree* yang telah terbangun pada Gambar 2.4 terdiri dari dua cabang yaitu lintasan <I2,I1,I5 :1> dan lintasan <I2,I1,I3,I5 :1> sehingga lintasan awal <I2,I1 :1> dan <I2,I1,I3 :1>

merupakan *conditional pattern base* dari I5. Namun, *conditional FP-tree* yang terbentuk adalah lintasan <I2, I1 : 2> karena nilai *support* I3 tidak memenuhi *minimum support*. Hasil penambangan yang dilakukan pada *conditional FP-tree* tersebut menghasilkan pola *frequent* {(I2,I5 :2),(I1,I5 :2),(I2,I1,I5 :2)}. Hasil dari fase proses penggalian *FP-growth* keseluruhan dari permasalahan ini dapat dilihat pada Tabel 2.2.

Tabel 2.2 Hasil Fase Proses Penggalian *FP-growth*

Item	Conditional Pattern Base	Conditional FP-tree	Frequent Pattern yang Dihasilkan
I5	{(I2,I1 :1), (I2,I1,I3 :1)}	<I2:2, I1:2>	{I2,I5 :2}, {I1,I5 :2}, {I2,I1,I5 ;2}
I4	{(I2,I1 :1), (I2:1)}	<I2:2>	{I2,I4 :2}
I3	{(I2,I1 :2), (I2:2), (I1:2)}	<I2:4, I1:2>, <I1:2>	{I2,I3 :4}, {I1,I3 :4}, {I2,I1,I3 :2}
I1	{(I2 :4)}	<I2 :4>	{I2,I1 :4}

2.7 Struktur Data *SOTrieIT*

Sub Ordered Trie Itemset (*SOTrieIT*) merupakan struktur data *tree* yang digunakan pada algoritma *Fold Growth* untuk *mining* kandidat *1-itemset* dan *2-itemset*. *SOTrieIT* memiliki dua tingkatan dari *node-node Tree* dengan tiap *node w* memiliki sebuah label *l* yang menyatakan *item* dan sebuah notasi *j* yang menyatakan nilai *support*. Setiap *node tree* terhubung pada beberapa *item* yang terdapat dalam *itemset I* (dinotasikan $a_i \in I$) maka untuk w_i mengacu pada *node* yang memiliki hubungan dengan $a_i \in I$. *SOTrieIT* dimungkinkan memiliki *parent node* yang berbeda-beda seperti w_1 , w_2 ,..., w_N, yang dibangun dari sebuah kumpulan data untuk menyimpan *support count* dari semua *1-itemset* dan *2-itemset*. Oleh karena itu dibutuhkan *node* khusus yang dinamakan *ROOT* untuk menghubungkan semua *SOTrie* bersama-sama (Soelaiman, 2006).

Pseudocode pembentukan *SOTrieIT* dapat dilihat pada *Pseudocode* 2.3.

Masukan:Basis data Transaksi(D),dan nilai minimum *support*(minsup)
Keluaran: Sekumpulan dari pola *frequent*
Fungsi : SOTrie IT dibentuk dengan cara berikut
1. Membuat himpunan SOTrieIT (Y)
2. untuk (tidak iterasi (k) = 1; Himpunan kandidat k-itemset(L) < 2; k++) maka dimulai dengan
3. Mendapatkan semua k-itemset dari transaksi dan
4. Menyimpannya di Himpunan kandidat k-itemset (Ck)
5. untuk setiap itemset (X) anggota dari(E) Himpunan kandidat k-itemset (Ck) lakukan dimulai dengan
6. Lintasi himpunan dari SOTrieIT (Y) untuk menempati node sepanjang jalur yang menunjukkan itemset (X)
7. jika ada himpunan node yang sudah ada di himpunan dari SOTrieIT (Y) maka
8. Tambahkan support count di daun node
9. Sortir node yang baru berdasarkan Anaknya untuk support count baru di urutan menurun kalau

   tidak maka
10. Membuat himpunan node baru dengan *support count* dari 1 yang menunjukkan jalur ke itemset (X)
11. Memasukkannya ke dalam himpunan dari SOTrieIT (Y) berdasatkan *support countnya* ke urutan menurun dari kiri
12. akhir dari jika
13. akhir dari looping
14. akhir dari loopig

***Pseudocode* 2.3** : *Pseudocode* Pembentukan *SOTrieIT* (Das, Ng, dan Woon, 2001)

Dalam pembentukan *SOTrieIT*, 1-*Itemset* dan 2-*Itemset* dari setiap transaksi yang dibaca diekstraksi. Tahap-tahap dari algoritma pembentukan *SOTrieIT* dijelaskan sebagai berikut (Das, Ng, dan Woon, 2001) :

1. Penentuan 1-*itemset* dan 2-*itemset* dari setiap transaksi.
2. Penambahan *node* baru pada *tree* dengan nilai *support* adalah 1(satu) jika *node* tersebut tidak terdapat pada level 1 dan level 2, sedangkan jika *node* dengan

label item yang sama sudah ada dalam *tree* maka dilakukan penambahan nilai *support* di *node* tersebut dan dilakukan *rebalancing* antar *node* dalam *tree* tersebut.

Ilustrasi untuk memperjelas pembangunan *SOTrieIT* tersebut dijelaskan dengan menggunakan data transaksi pada Tabel 2.3 (Das, Keong, dan Woon, 2001)..

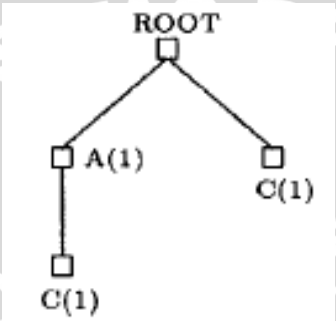
Tabel 2.3 Contoh data transaksi

TID	Items
100	A C
200	B C
300	A B C
400	A B C D

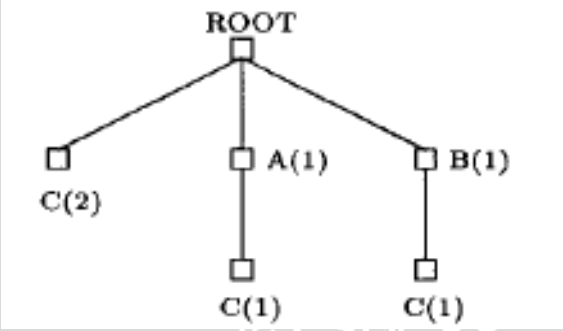
Tiap transaksi akan dibaca dan dilakukan ekstraksi 1-*itemset* dan 2-*itemset*, maka ketika transaksi dengan TID 100 dibaca, ekstraksi 1– *itemset* yang didapatkan : {A, C} dan ekstraksi 2–*itemset* yang didapatkan : {AC} sehingga pembentukan *SOTrieIT* untuk data transaksi dengan TID 100 dilihat pada Gambar 2.3. Selanjutnya pembacaan transaksi dengan TID 200, ekstraksi 1-*itemset* yang didapatkan : {B, C} dan 2-*itemset* yang didapatkan : {BC} sehingga pada *SOTrieIT* terbentuk *node* baru yaitu *node* B pada level 1 dengan nilai *support* 1 dan *node* C pada level 2 atau *node* anak dari *node* B dengan nilai *support* 1. Selain itu *node* C pada level 1 nilai *support* bertambah 1. *SOTreeIT* yang terbentuk dari pembacaan transaksi dengan TID 200 dapat dilihat pada Gambar 2.4.

Pada pembacaan transaksi dengan TID 300, ekstraksi 1-*itemset* yang didapatkan adalah {A, B, C} dan ekstraksi 2-*itemset* yang didapatkan adalah {AB, BC, AC}. Untuk hasil ektraksi 1-*itemset* A dan 2-*itemset* AB dan AC, pada *SOTrieIT node* dengan label A ditambahkan nilai *support* sebanyak 1 sehingga menjadi 2. *Node* A pada *SOTrieIT* memiliki *node* anak C maka node anak C nilai *support* ditambahkan 1 karena terdapat hasil ekstraksi dengan lintasan sama yaitu AC. Namun *node* A tidak memiliki *node* anak B sedangkan hasil ektraksi ada AB

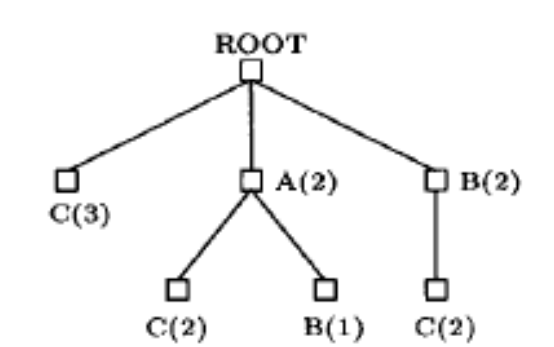
sehingga dibentuk *node* baru yang merupakan *node* anak dari *node* A dengan nilai *support* 1. Kemudian untuk hasil ektraksi BC, karena pada *SOTrieIT* terdapat cabang dengan lintasan yang sama maka *node* pada cabang tersebut ditambahkan nilai *support* sebanyak 1. *SOTrieIT* yang terbentuk dari pembacaan transaksi dengan TID 300 dapat dilihat pada Gambar 2.5. Langkah yang sama dilakukan sampai pembacaan transaksi terakhir yaitu TID 400 sehingga *SOTrieIT* yang terbentuk setelah pembacaan data transaksi keseluruhan dapat dilihat pada Gambar 2.6.



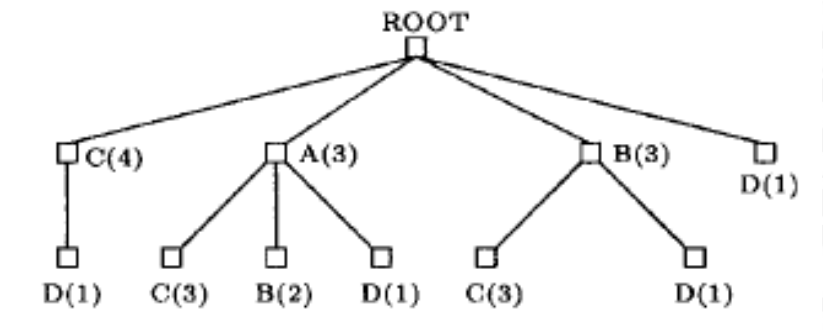
Gambar 2.3 *SOTrieIT* Pembacaan Transaksi TID 100



Gambar 2.4 *SOTrieIT* Pembacaan Transaksi TID 200



Gambar 2.5 *SOTrieIT* Pembacaan Transaksi TID 300



Gambar 2.6 *SOTrieIT* Pembacaan Transaksi TID 400

2.8 Algoritma *FOLD-Growth*

Algoritma *FOLD-Growth* merupakan hasil gabungan dari algoritma *FOLDARM* dan *FP-Growth*. Pada algoritma *FOLD-growth* ini, diharapkan dapat menggabungkan keuntungan dari algoritma *FOLDARM* yang memiliki kinerja cepat pada saat ukuran *itemset frequent* maksimum (k_{max}) adalah kecil atau $k_{max} \leq 10$ dengan keuntungan dari algoritma *FP-Growth* yang memiliki kinerja yang cepat pada saat $k_{max} > 10$. *Pseudocode* dari algoritma *FOLD-growth* ditunjukkan pada *Pseudocode* 2.4 (Soelaiman, 2006).

1: Menggunakan SOTrie untuk menemukan $L1$ dan dengan cepat $L2$
2: **jika** $L1=0 \vee L2=0$
3: algoritma dihentikan
4: **berakhir jika**
5: **untuk** transaksi T ∈ D **maka lakukan**
6: Hilangkan item yang tidak memberikan kontribusi pada *Lk* dimana k > 2, menggunakan $L1$ dan $L2$
7: Urutkan item berdasarkan support terbesar
8: Bangun/Ubah FP-tree dengan T yang telah dipangkas dan diurutkan
9: **berakhir untuk**
10: Jalankan algoritma FP-growth pada FP-tree yang dibangun

***Pseudocode* 2.4** : *Pseudocode* Algoritma *FOLD-Growth*

Tahap - tahap algoritma *FOLD-Growth* terdiri dari (Prastowo, 2008) :

a. Pencarian *1-itemset* dan *2-itemset* dengan menggunakan *SOTrieIT*

Pencarian *1-itemset* dan 2-*itemset* dilakukan dengan melakukan pembacaan basis data transaksi sebanyak satu kali. Kemudian akan dibangkitkan semua kemungkinan *1-itemset* dan *2-itemset* untuk setiap transaksi yang selanjutnya disimpan dalam *SOTrieIT*.

b. Pemangkasan item-item yang tidak *frequent*

Pemangkasan item – item yang tidak *frequent* pada setiap transaksi yang ada dalam basis data dengan menggunakan *1-itemset* dan *2-itemset*. Untuk setiap transaksi *T*, pada itemset L_k pada transaksi tersebut dimana panjang *k* lebih dari 2, dilakukan pengecekan dengan menggunakan *1-itemset* dan *2-itemset* sehingga akan dipangkas item-item yang dianggap tidak *frequent*. Item yang dikatakan tidak *frequent* adalah item yang memiliki nilai *support* kurang dari nilai *minimum support* yang telah ditentukan. Setelah item tidak *frequent* dipangkas terhadap transaksi *T*, maka item – item yang terdapat dalam transaksi tersebut akan diurutkan berdasarkan nilai *support* secara *descending* (terurut dari nilai yang besar ke nilai yang kecil). Dari tahap ini akan dihasilkan *Ordered Frequent Items*.

c. Pembangunan *FP-tree*

Pada pembangunan *FP-Tree* data transaksi T yang digunakan adalah data transaksi T yang telah dipangkas dan diurutkan berdasarkan nilai *support* secara *descending*.

d. *Mining itemset frequent* dengan algoritma *FP-growth*.

Tahap *mining itemset frequent* dilakukan dengan menggunakan algoritma *FP-growth* pada *FP-tree* yang telah terbangun pada tahap sebelumnya

.

Aturan asosiasi yang didapatkan berdasarkan basis data transaksi pada Tabel 2.3 dan dengan *minimum support* dan *minimum confidence* adalah 50% dalam proses penggalian rekursif di atas, dapat dilihat pada Tabel 2.4 (Soelaiman, 2006).

Tabel 2.4 *Rule* yang Dihasilkan

Pola Asosiasi	*Support*	*Confidence*
{A} → {C}	0.75	1
{C} → {A}	0.75	0.75
{B} → {C}	0.75	1
{C} → {B}	0.75	0.75

2.9 *Lift Ratio*

Lift ratio merupakan suatu ukuran untuk mengevaluasi kekuatan sebuah aturan asosiasi yang didapatkan melalui perbandingan antara *confidence* sebuah aturan dengan nilai *benchmark confidence*. *Benchmark confidence* itu sendiri merupakan perbandingan antara jumlah semua *item* yang menjadi *consequent* (bagian maka) terhadap total jumlah transaksi (Santosa, 2007). Untuk menghitung nilai *benchmark confidence* digunakan rumus yang dapat dilihat pada Persamaan 2.5, sedangkan untuk menghitung nilai *lift ratio* digunakan rumus pada Persamaan 2.6.

$$Benchmark\ Confidence = \quad Nc/N \qquad (2.5)$$

Keterangan rumus 2.5 :

Nc = jumlah transaksi dengan *item* yang menjadi *consequent*

N = jumlah transaksi basis data

$$Lift\ Ratio = \frac{Confidence\ (A,C)}{Benchmark\ Confidence} \qquad (2.6)$$

Suatu aturan asosiasi dikatakan menunjukkan adanya manfaat jika nilai *lift ratio* aturan tersebut lebih besar dari 1 dan semakin tinggi nilai *lift ratio* maka semakin besar kekuatan asosiasinya (Santosa, 2007).

BAB III
METODOLOGI DAN PERANCANGAN

Pada bab ini, pembahasan meliputi metode dan perancangan langkah–langkah yang dilakukan dalam penelitian untuk penerapan metode *association rule* dengan menggunakan algoritma *Fold-Growth* pada data transaksi penjualan buku.

Langkah–langkah penelitian ini dapat digambarkan seperti pada Gambar 3.1.

Gambar 3.1 Langkah – Langkah Penelitian

Keterangan :

1. Melakukan studi literatur mengenai *association rule*, algoritma *Fold-Growth.*
2. Mengumpulkan data–data yang diperlukan dalam penelitian ini yang berupa data transaksi penjualan buku dan alat tulis.
3. Menganalisa dan melakukan perancangan sistem untuk proses *mining association rule* dari data transaksi penjualan buku.

4. Mengimplementasikan rancangan yang dilakukan pada tahap sebelumnya menjadi sebuah perangkat lunak untuk membantu proses analisa pola *association rule* dari data transaksi penjualan buku.
5. Melakukan uji coba terhadap perangkat lunak menggunakan data transaksi penjualan buku yang telah tersimpan dalam *database*.
6. Mengevaluasi hasil analisa yang dilakukan oleh sistem.

3.1. Pengumpulan Data

Data yang akan digunakan dalam penelitian ini dikumpulkan dengan menggunakan metode pengumpulan data lapangan. Data yang dikumpulkan adalah data transaksi penjualan buku dan alat tulis. Sumber data yang digunakan adalah laporan transaksi penjualan buku dari toko buku Togamas Malang. Data yang diperoleh adalah data transaksi penjualan buku dan alat tulis dari bulan Januari sampai bulan Desember tahun 2010 dengan rincian sebagai berikut :

1. Transaksi bulan Januari : 65530 *record* data.
2. Transaksi bulan Februari : 65530 *record* data.
3. Transaksi bulan Maret : 65530 *record* data.
4. Transaksi bulan April : 64090 *record* data.
5. Transaksi bulan Mei : 61897 *record* data.
6. Transaksi bulan Juni : 65530 *record* data.

Atribut yang terdapat dalam *dataset* terdiri dari atribut kode transaksi, tanggal transaksi, nama buku, penerbit buku dan golongan buku. Kode transaksi terdiri dari informasi kode kasir, tahun transaksi(*yy*), bulan transaksi(*mm*), tanggal transaksi(*dd*), dan nomor urut transaksi pada kasir dan tanggal bersangkutan . Atribut tanggal transaksi memiliki format tanggal, bulan, dan tahun transaksi (*dd*/*mm*/*yyyy*). Berdasarkan informasi dan hasil pengamatan data transaksi penjualan buku yang terkumpul, jenis buku yang ada pada toko buku Togamas Malang adalah sebanyak 84 jenis buku dan terbagi menjadi 13 golongan buku. Sedangkan data alat tulis yang terdapat pada toko buku Togamas Malang dikelompokkan menjadi 4 kelompok.

3.2. Analisis dan Perancangan Sistem

3.2.1. Deskripsi Sistem

Pada penelitian ini, akan dibuat sistem untuk melakukan analisis terhadap data transaksi penjualan buku dan alat tulis sehingga didapatkan pola penjualan buku dan alat tulis. Dalam analisis, parameter yang akan digunakan adalah nilai *minimum support*, nilai *minimum confidence*, dan periode data transaksi yaitu per bulan, per dua bulan, dan per tiga bulan. Sistem akan menghasilkan keluaran berupa pola *association rule*. Pola *association rule* yang dihasilkan oleh sistem akan menunjukkan keterkaitan antara buku yang satu dengan buku yang lain, alat tulis yang satu dengan alat tulis yang lain, dan buku dengan alat tulis.

Sistem akan dibuat secara keseluruhan menggunakan metode *association rule* dimana untuk melakukan *frequent itemset mining* dilakukan dengan menerapkan algoritma *Fold-Growth*. Itemset dalam sistem berupa himpunan golongan buku dan kelompok alat tulis. *Frequent itemset* dalam sistem berupa himpunan golongan buku dan kelompok alat tulis yang frekuensi kemunculan pada data transaksi penjualan memenuhi nilai minimum *support* yang menjadi masukan dalam sistem. *Frequent itemset* tersebut selanjutnya akan dihitung nilai *confidence*-nya. Melalui nilai *confidence* tersebut akan dilakukan pembangkitan dari *frequent itemset* menjadi *rule*, yaitu *frequent itemset* yang memiliki nilai *confidence* yang memenuhi (lebih besar atau sama dengan) nilai *minimum confidence* yang menjadi masukan juga pada sistem. Proses pembangkitan dari *frequent itemset* menjadi *rule* inilah akan diterapkan metode *association rule* itu sendiri.

Keluaran dari sistem adalah *rule* baik dalam bentuk kode maupun keterangan beserta nilai *support*, nilai *confidence* dan nilai *lift ratio*. Sistem akan menyimpan hasil keluaran ini dalam bentuk file *excel* (berformat xls).

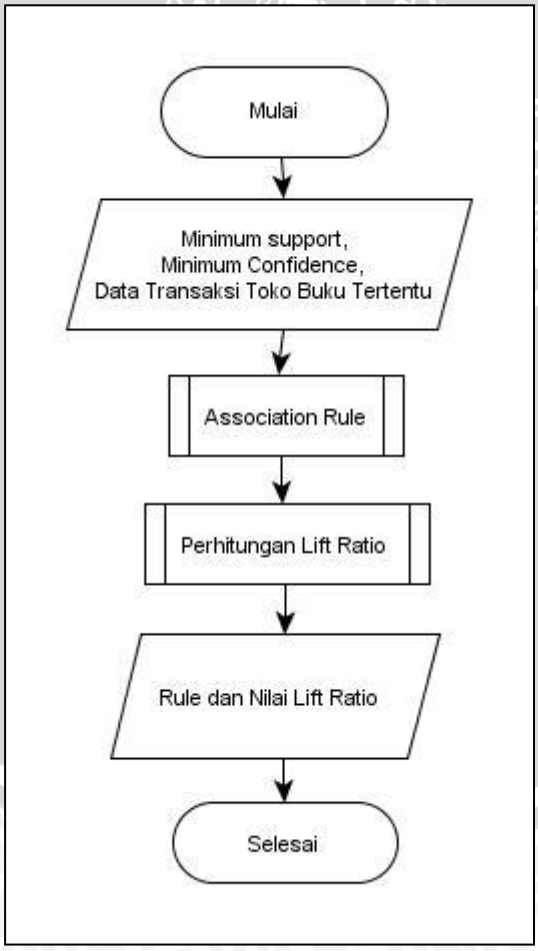
3.2.2. Rancangan Pembuatan Sistem

Untuk pembuatan sistem yang menerapkan metode *association rule* dengan algoritma *Fold-Growth* ini dibutuhkan proses perancangan sistem yang terdiri dari rancangan proses, rancangan basis data, dan rancangan antar muka sistem.

3.2.2.1 Rancangan Proses

Dalam proses penelitian ini, metode *data mining* yang digunakan adalah *association rule* dengan menggunakan algoritma *Fold–Growth*. Metode ini diterapkan untuk menganalisa frekuensi kemunculan masing–masing golongan buku dengan menggunakan parameter waktu transaksi.

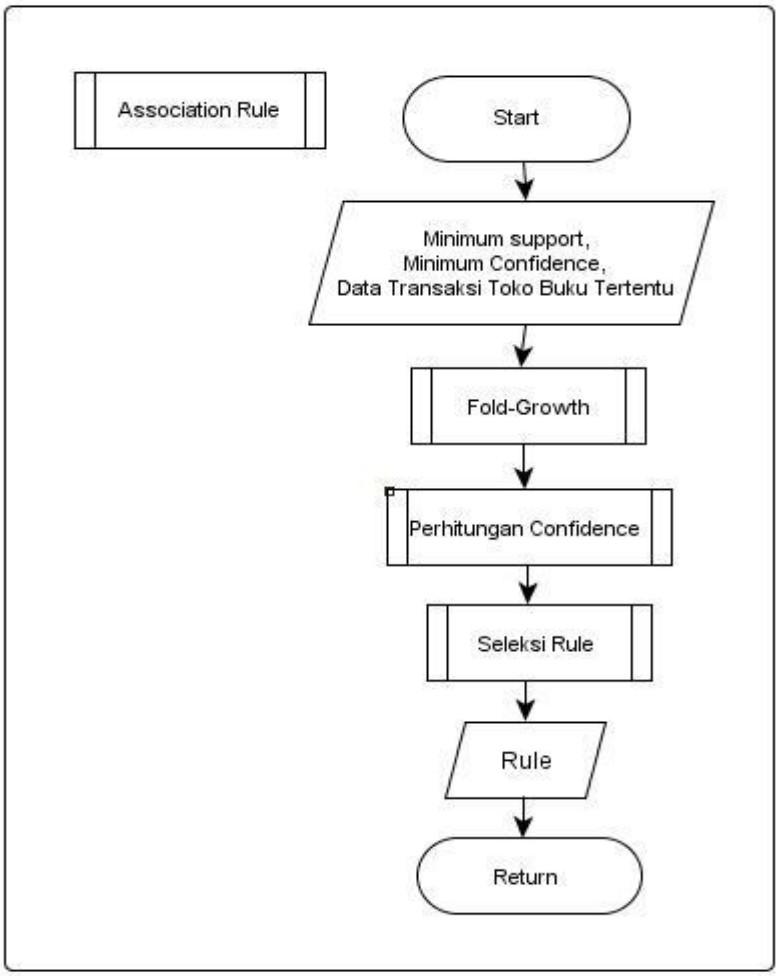
Sebelum melakukan pembuatan sistem untuk penerapan algoritma *Fold-Growth* ini, perlu dilakukan rancangan proses. Rancangan proses secara umum digambarkan pada Gambar 3.2. Secara umum, sistem membutuhkan data masukan berupa minimum *support*, minimum *confidence*, dan data transaksi periode tertentu. Data masukan tersebut akan di proses melalui beberapa proses antara lain, proses *association rule* ,dan perhitungan *lift ratio*. Hasil keluaran sistem berupa *rule*,dan nilai *lift ratio*.



Gambar 3.2 *Flowchart* Sistem Secara Umum

Proses *association rule* sendiri, dijelaskan lebih detail pada Gambar 3.3. Pada proses *association rule*, data masukan yang berupa minimum *support*,

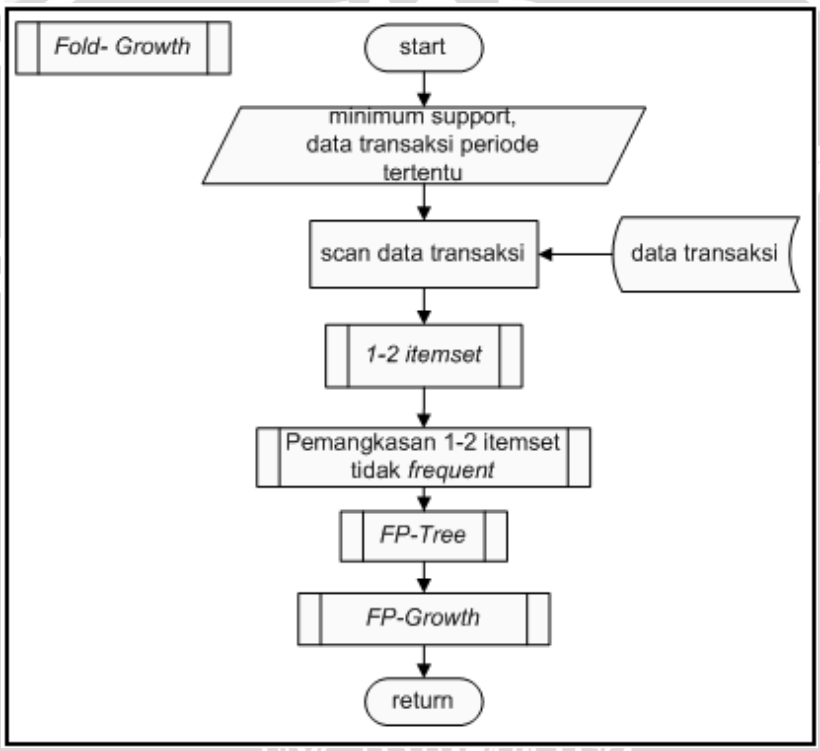
minimum *confidence*, dan data transaksi akan diproses melalui proses *Fold-Growth*. Proses *Fold-Growth* digunakan untuk mendapatkan *frequent itemset* dari keseluruhan data transaksi. Selanjutnya dilakukan proses perhitungan *confidence* dari setiap *frequent itemset* yang didapatkan melalui proses *Fold-Growth*. Dengan informasi nilai *confidence* tiap *frequent itemset* tersebut, selanjutnya dilakukan proses seleksi *rule*, yaitu proses pembangkitan *frequent itemset* menjadi *rule*. *Frequent itemset* yang terseleksi adalah *frequent itemset* yang memiliki nilai *confidence* yang memenuhi nilai *minimum confidence*



Gambar 3.3 *Flowchart Association Rule*

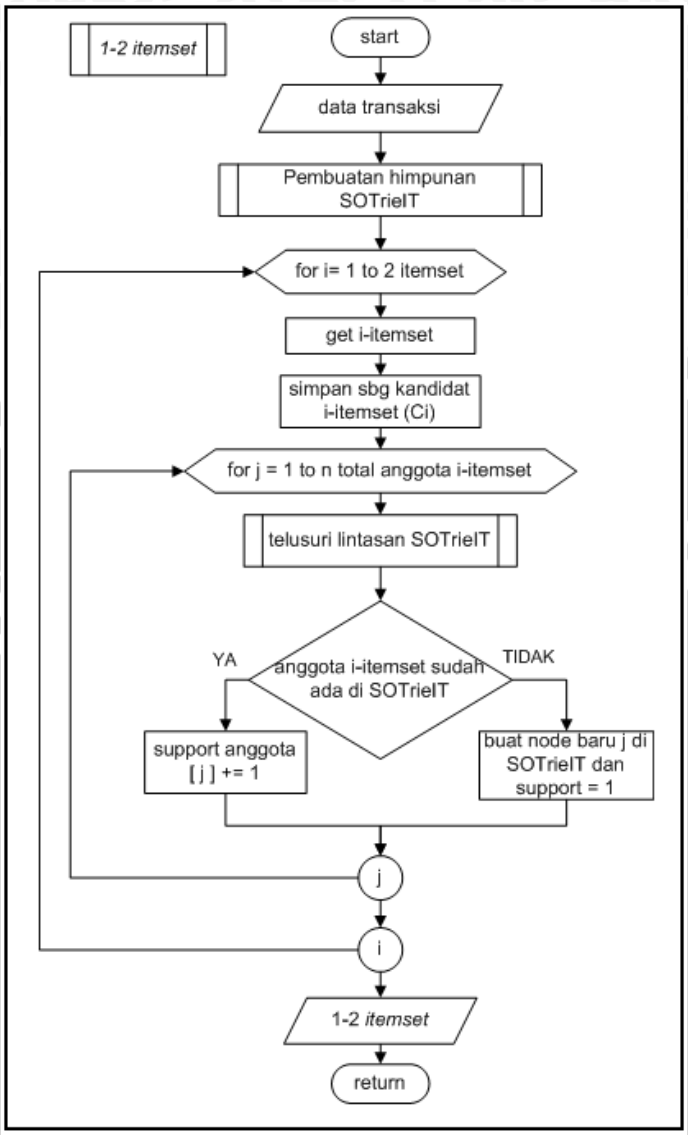
Rancangan proses *Fold-Growth* dijelaskan lebih detail pada Gambar 3.4. Proses *Fold-Growth* membutuhkan data input berupa minimum *support* dan data transaksi periode tertentu. Proses ini diawali dengan *scan* data transaksi dari

database. Setiap data transaksi dibaca dilakukan pencarian 1 *itemset* dan 2 *itemset*. Setelah semua data transaksi didapatkan 1-2 *itemset*, selanjutnya dilakukan proses pemangkasan itemset yang tidak *frequent*. Itemset yang tidak *frequent* adalah itemset yang tidak memenuhi nilai *minimum support*. Untuk data 1 *itemset* dan 2 *itemset* yang *frequent* selanjutnya dilakukan proses pembuatan *FP-Tree*. Setelah *FP-Tree* terbentuk, dilakukan proses *mining frequent itemset* dengan menggunakan *FP-Growth*.

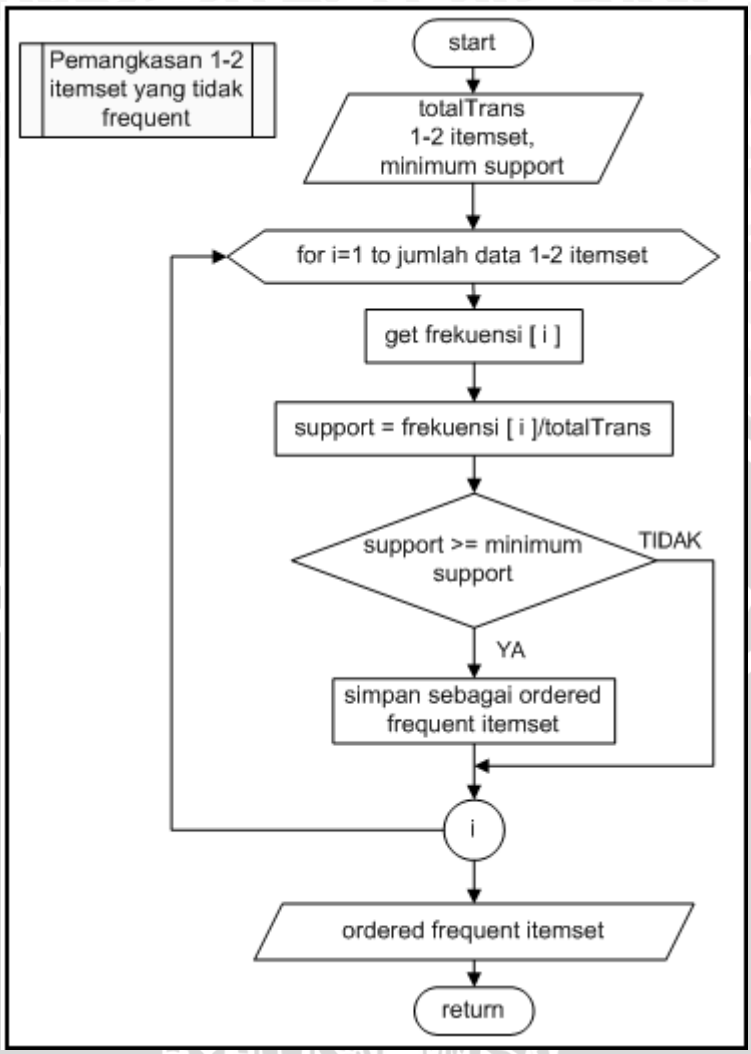


Gambar 3.4 *Flowchart Fold-Growth*

Gambar 3.5 menjelaskan proses 1-2 itemset lebih detail. Proses 1-2 itemset ini digunakan untuk mendapatkan 1-itemset dan 2-itemset dari setiap data transaksi sebagai kandidat 1-2 itemset dan selanjutnya dilakukan perhitungan frekuensi kemunculan tiap 1–2 itemset. Proses ini membutuhkan data input berupa data transaksi. Awal proses adalah dilakukan pembuatan himpunan *SOTrieIT*. Kemudian untuk setiap i itemset dilakukan penelusuran pada *SOTrieIT* sehingga didapatkan frekuensi kemunculannya pada data transaksi.



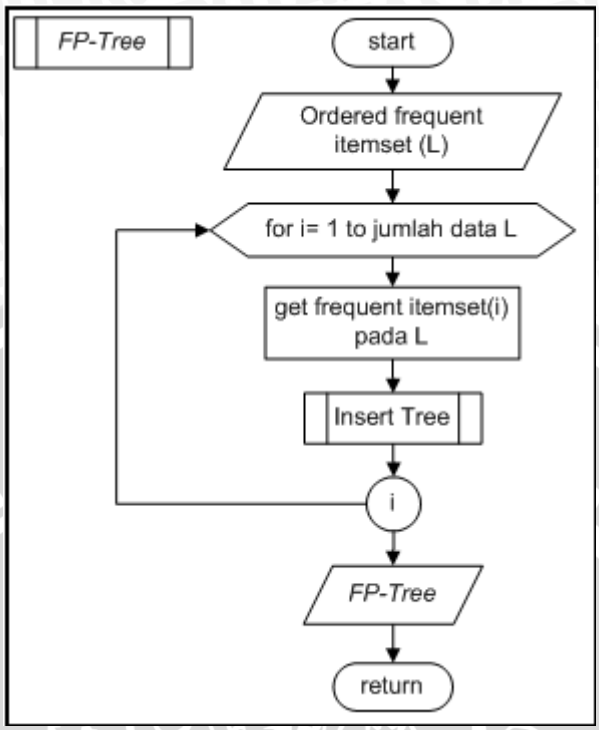
Gambar 3.5 *Flowchart 1-2 Itemset*



Gambar 3.6 *Flowchart* Pemangkasan 1-2 Itemset

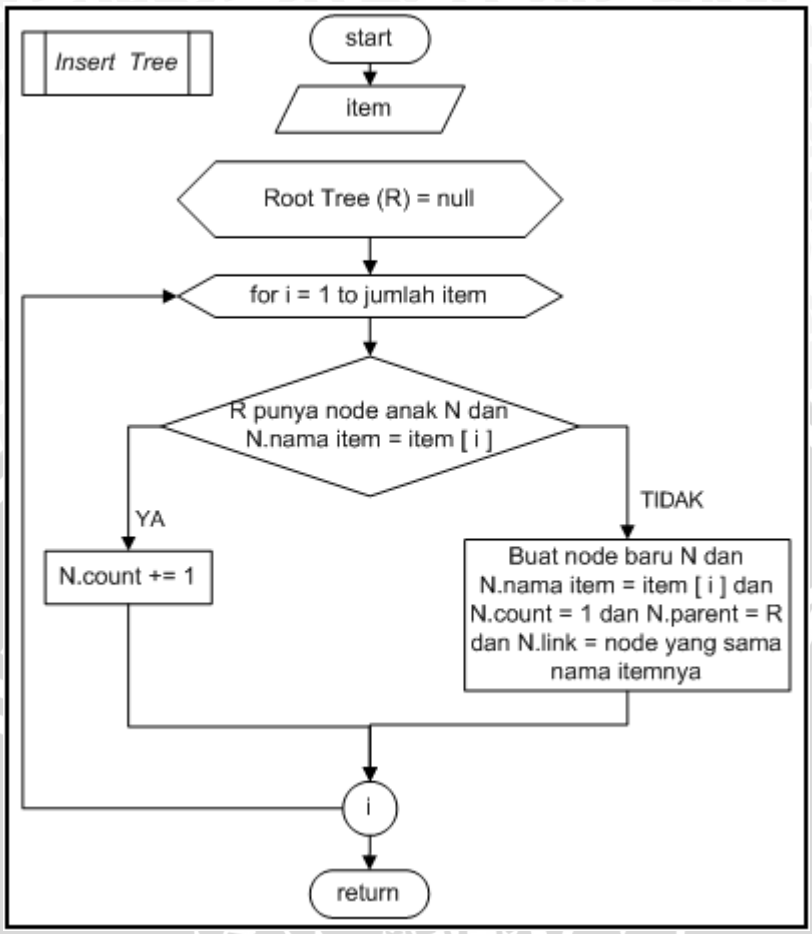
Pada Gambar 3.6 dijelasakan diagram alur dari proses pemangkasan 1-2 itemset. Proses ini digunakan untuk mendapatkan himpunan 1-*itemset* dan 2-*itemset* yang nilai *support*-nya memenuhi (lebih besar atau sama dengan) nilai *minimum support*. Selanjutnya himpunan 1-*itemset* dan 2-*itemset* yang terseleksi tersebut menjadi *ordered frequent itemset*.

Proses *FP-Tree* dijelaskan lebih detail pada Gambar 3.7. Proses ini merupakan proses untuk membangun *FP-Tree*. Setiap *itemset* pada *ordered frequent itemset* dimasukkan ke dalam *FP-Tree* sehingga hasil keluaran dari proses ini adalah *FP-Tree.*



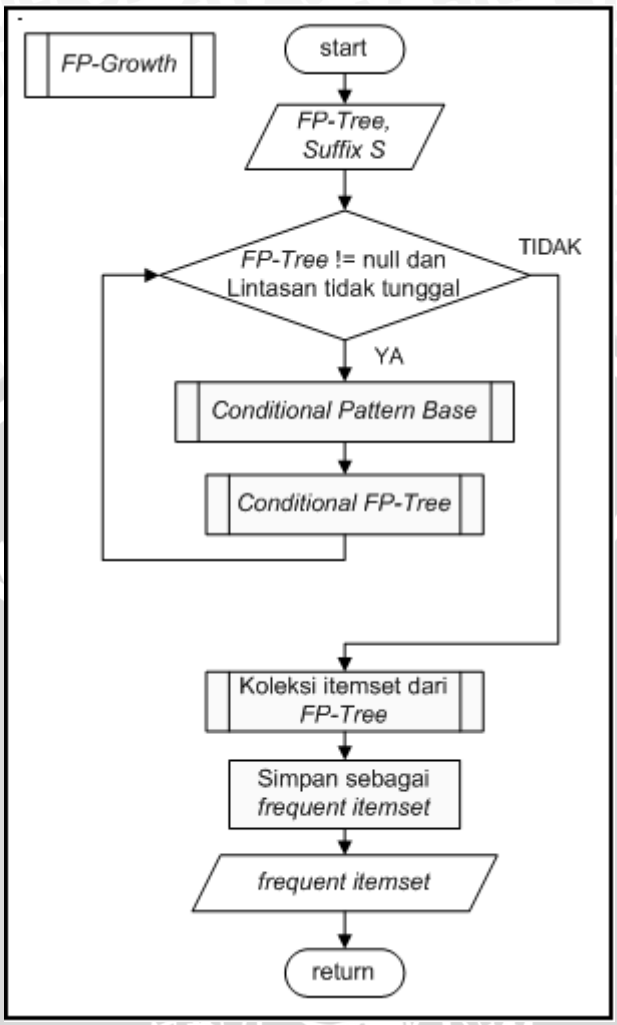
Gambar 3.7 *Flowchart FP-Tree*

Proses *insert tree* dijelaskan lebih detail pada Gambar 3.8. Proses ini merupakan proses untuk memasukkan *item* ke dalam *FP-Tree*. Proses *insert tree* diawali dengan inisialisasi *root* dari *FP-Tree* yang bernilai *null*. Kemudian untuk tiap *itemset* dilakukan pembacaan item pada itemset. Kemudian dilakukan pengecekan pada *root* apakah memiliki *node* anak yang nama itemnya sama dengan item yang sedang dibaca, jika ada maka *count node* tersebut ditambahkan nilainya sebanyak 1. Sedangkan jika belum ada maka dilakukan pembuatan *node* baru yang memiliki nama item item tersebut dan *parent node* adalah *root* dan *link node* adalah *node* yang sama nama itemnya.



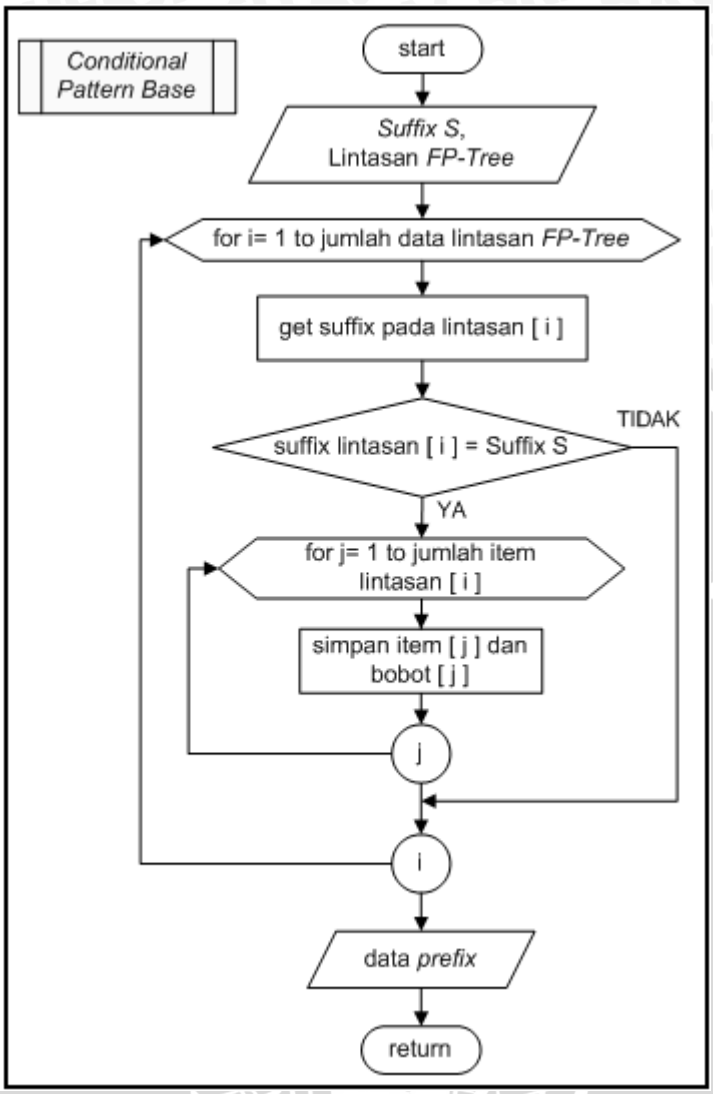
Gambar 3.8 *Flowchart Insert Tree*

Gambar 3.9 menjelaskan detail proses *FP-Growth*. Proses ini digunakan untuk mendapatkan *frequent itemset*. Data input yang dibutuhkan pada proses ini adalah *FP-Tree* dan item yang menjadi *suffix*. Pada awal proses dilakukan pengecekan terlebih dahulu apakah *FP-Tree* tidak *null* dan lintasan tidak tunggal jika ya, maka dilakukan proses *conditional pattern base* dan dilanjutkan dengan proses *conditional FP-Tree*. Sedangkan jika tidak maka dilakukan koleksi *itemset* pada *FP-Tree* dan setiap *itemset* disimpan sebagai *frequent itemset*.



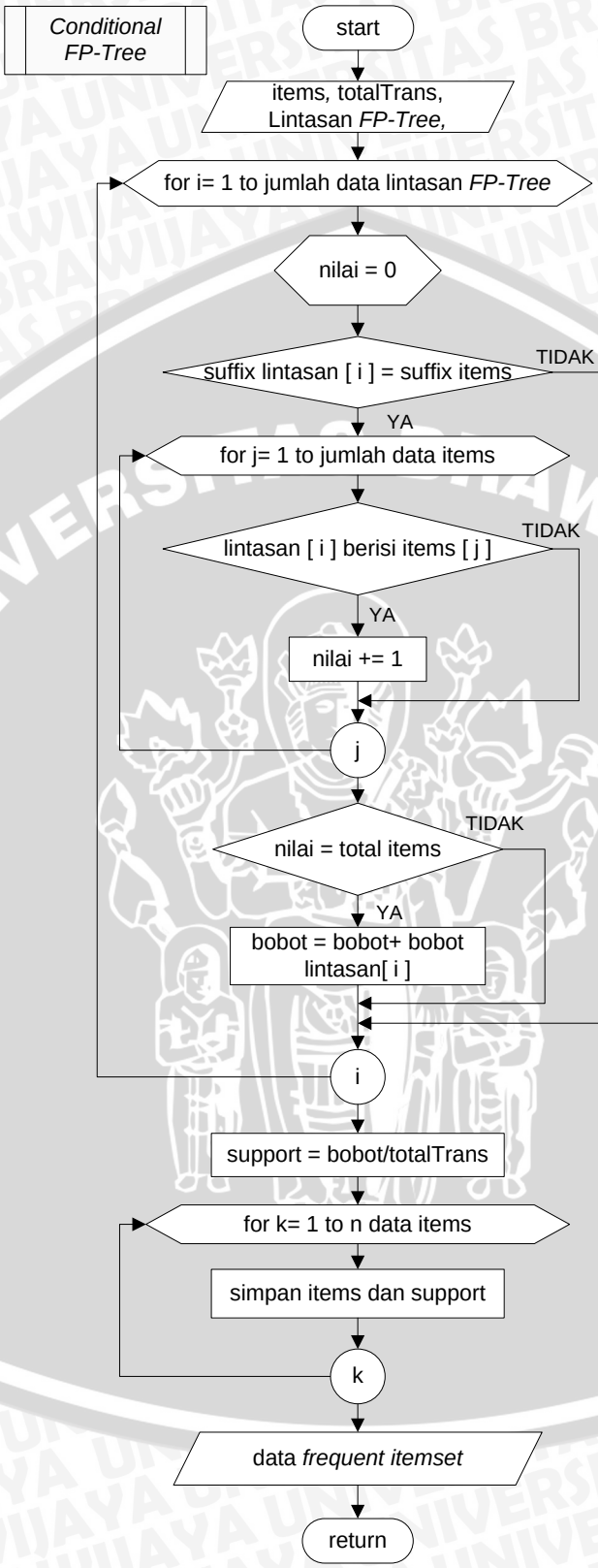
Gambar 3.9 *Flowchart FP-Growth*

Proses *conditional pattern base* merupakan proses untuk mendapatkan informasi lintasan tiap *suffix*. Proses ini digambarkan pada Gambar 3.10. Data input yang dibutuhkan pada proses ini adalah *item* yang menjadi *suffix*, dan data lintasan *FP-Tree*. Setiap lintasan akan dicek apakah memiliki *suffix* yang sama seperti *suffix* yang dicari, jika sama maka tiap *item* yang menyusun lintasan tersebut akan disimpan beserta informasi bobotnya.

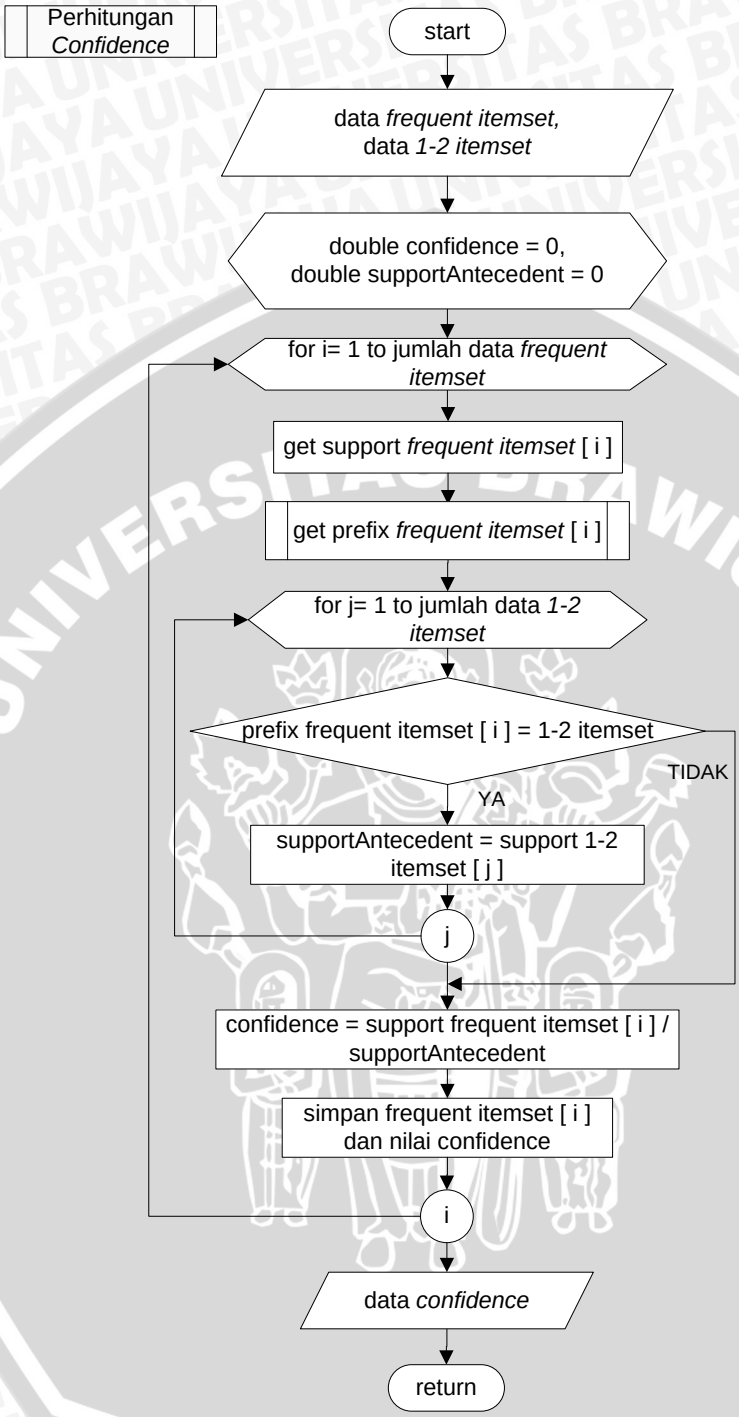


Gambar 3.10 *Flowchart Conditional Pattern Base*

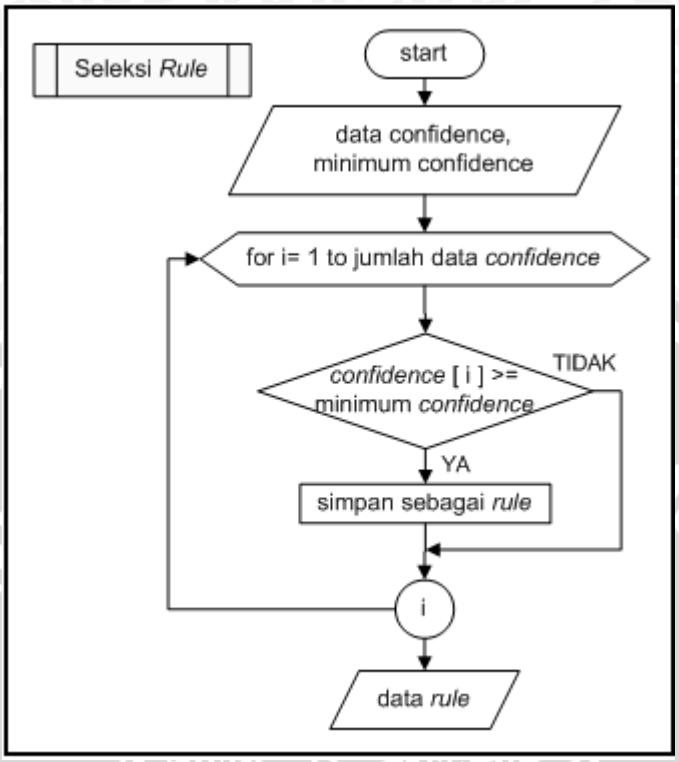
Proses *conditional FP-Tree* merupakan proses untuk melakukan perhitungan *support* tiap *itemset*. Proses ini digambarkan pada Gambar 3.11. Proses dilakukan dengan mencari lintasan yang memiliki *suffix* yang dicari dan memiliki item penyusun yang sama.



Gambar 3.11 *Flowchart Conditional FP-Tree*



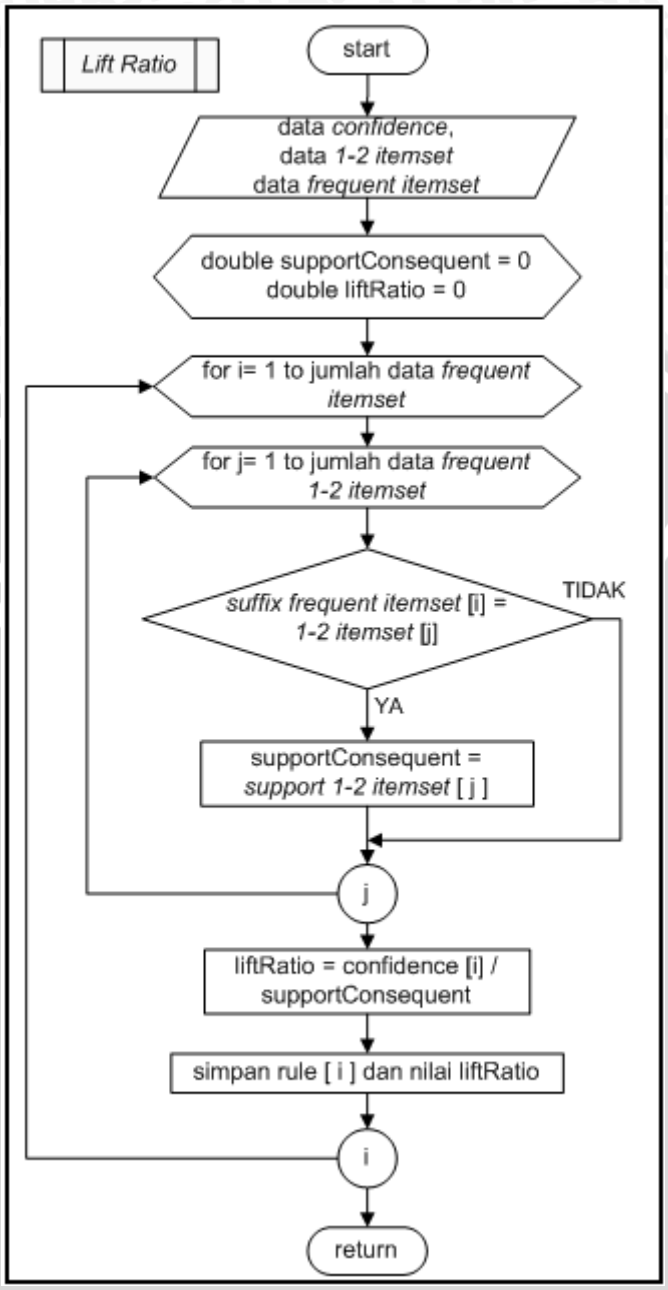
Gambar 3.12 *Flowchart* Perhitungan *Confidence*



Gambar 3.13 *Flowchart* Seleksi *Rule*

Proses perhitungan *confidence* merupakan proses untuk menghitung nilai *confidence* tiap *frequent itemset*. Proses ini digambarkan pada Gambar 3.12. Proses dilakukan dengan mencari *prefix* dari *frequent itemset*, kemudian dilakukan pengecekan pada data *1-2 itemset* yang *item*-nya sama dengan *prefix frequent itemset* tersebut. Untuk *1-2 itemset* dengan *item* yang sama dengan *prefix frequent itemset*, diambil nilai *support*-nya sebagai *support antecedent*. Kemudian dilakukan perhitungan *confidence frequent itemset* tersebut dengan cara *support frequent itemset* dibagi dengan *support antecedent*.

Proses seleksi *rule* pada Gambar 3.13, digunakan untuk menyeleksi *frequent itemset* yang dapat dijadikan *rule*, yaitu *frequent itemset* yang memiliki nilai *confidence* lebih besar atau sama dengan nilai *minimum confidence*. *Frequent itemset* hasil seleksi disimpan sebagai *rule*.



Gambar 3.14 *Flowchart* Perhitungan *Lift Ratio*

Proses perhitungan *lift ratio* merupakan proses untuk menghitung nilai *lift ratio* tiap *rule*. Proses ini digambarkan pada Gambar 3.14. Proses dilakukan dengan mencari *suffix* dari *frequent itemset*, kemudian dilakukan pengecekan pada data *1-2 itemset* yang *item*-nya sama dengan *suffix frequent itemset* tersebut. Untuk *1-2 itemset* dengan *item* yang sama dengan *suffix frequent itemset*, diambil nilai *support*-nya sebagai *support consequent*. Kemudian dilakukan perhitungan

lift ratio frequent itemset tersebut dengan cara *confidence frequent itemset* dibagi dengan *support consequent*.

3.2.2.2. Rancangan Basis Data

Pada penelitian ini, dalam pembuatan sistem dibutuhkan suatu basis data untuk penyimpanan data transaksi. Namun, sebelum dilakukan penyimpanan data transaksi ke dalam basis data, dilakukan terlebih dahulu proses *preprocessing* dan transformasi data. Proses *preprocessing* adalah dilakukan pembersihan data dari atribut data yang tidak dibutuhkan seperti atribut nama penerbit dan atribut *quantity*. Selain itu, pembersihan data dari data dengan nilai yang hilang (*missing value*) dan memperbaiki kesalahan pada data seperti pengisian nilai pada atribut yang tidak tepat. Hasil dari proses *preprocessing* data selanjutnya ditransformasi ke dalam format data yang dibutuhkan sesuai dengan pola informasi yang akan dicari. Karena pola informasi yang akan dicari adalah asosiasi antar golongan item baik buku maupun alat tulis yang dibeli maka basis data yang dibutuhkan terdiri dari tabel transaksi, tabel jenis item, dan tabel golongan item.

Tabel transaksi digunakan untuk menyimpan data transaksi penjualan buku. Tabel jenis item digunakan untuk menyimpan data jenis – jenis item, sedangkan tabel golongan item digunakan untuk menyimpan golongan – golongan item. Tabel transaksi terdiri dari atribut id transaksi sebagai *primary key*, no faktur, tanggal dan kode jenis item yang dibeli sebagai *foreign key* yang merupakan referensi kode jenis item dari tabel jenis item seperti pada Tabel 3.1. Tabel jenis item terdiri dari atribut kode jenis item sebagai *primary key*, nama jenis item, dan kode golongan item sebagai *foreign key* yang merupakan referensi kode golongan item dari tabel golongan item seperti pada Tabel 3.2. Tabel golongan item terdiri dari atribut kode golongan item dan nama golongan seperti pada Tabel 3.3. Sedangkan Gambar rancangan basis data dan relasi basis data dapat dilihat pada Gambar 3.15.

Tabel 3.1 Tabel Transaksi

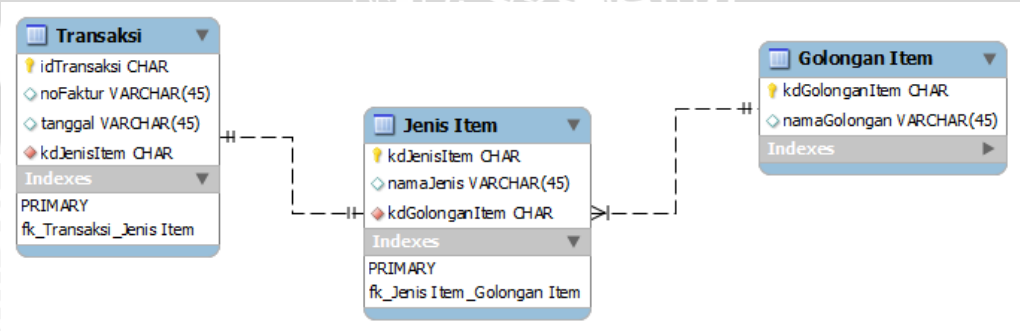
Atribut	Tipe Data
idTransaksi	Char
noFaktur	varchar
tanggal	varchar
kdJenisItem	Char

Tabel 3.2 Tabel Jenis Item

Atribut	Tipe Data
kdJenisItem	char
namaJenis	varchar
kdGolonganItem	char

Tabel 3.3 Tabel Golongan Item

Atribut	Tipe Data
kdGolonganItem	Char
namaGolongan	Varchar



Gambar 3.15 Diagram Relasi Basis Data

Berdasarkan diagram relasi basis data pada Gambar 3.15, tabel yang berelasi antara lain tabel transaksi dengan tabel jenis item, dan tabel jenis item dengan tabel golongan item. Meskipun pada sebenarnya, tiap transaksi dapat melakukan banyak pembelian jenis item, namun pada relasi tabel transaksi terhadap tabel jenis item memiliki kardinalitas relasi satu ke satu. Hal tersebut

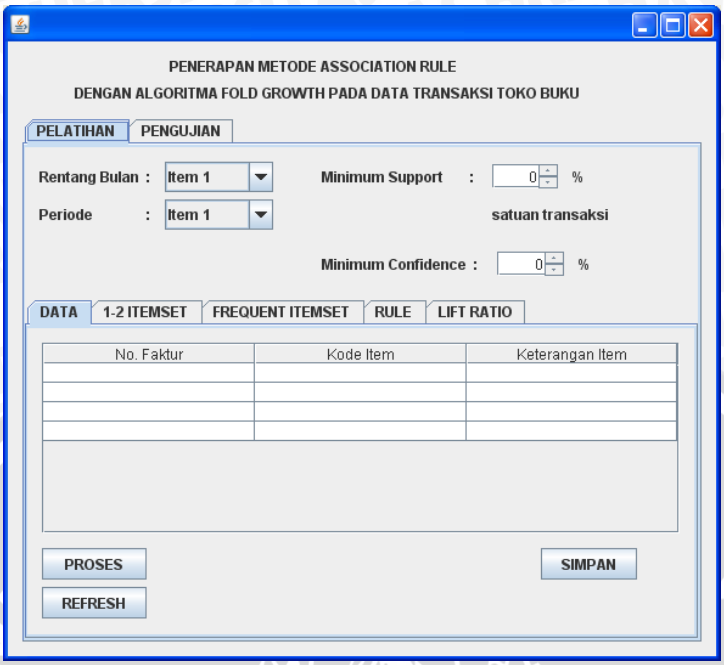
dikarenakan pembuatan tabel transaksi dan tabel jenis buku pada basis data bertujuan untuk pemetaan data. Sedangkan tabel golongan item terhadap tabel jenis item memiliki kardinalitas relasi satu ke banyak.

3.2.2.3. Rancangan Antar Muka

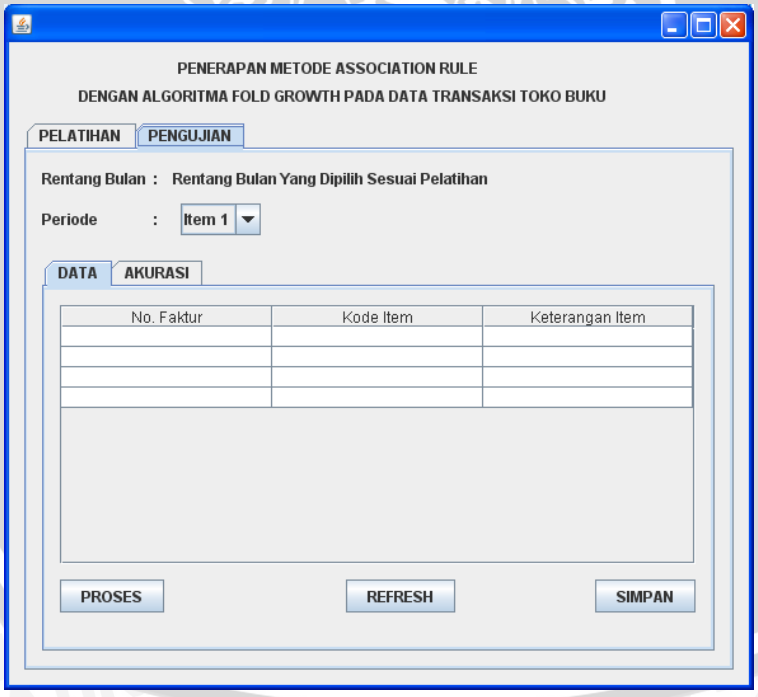
Rancangan antar muka secara umum terdiri dari bagian menu utama, tombol proses, tombol *refresh*, dan tombol simpan. Tombol proses digunakan untuk menjalankan perintah pemrosesan pada sistem. Tombol *refresh* digunakan untuk mengembalikan sistem pada posisi baik data *input* maupun data *output* kosong. Sedangkan tombol simpan berguna sebagai perintah untuk menyimpan hasil *output* sistem yang di-*export* ke bentuk file berformat xls.

Pada bagian menu utama terdiri dari menu pelatihan dan menu pengujian. Menu utama pelatihan memiliki bagian *input* yang terdiri dari pilihan rentang bulan, pilihan periode, data *input* minimum *support*, dan data *input* minimum *confidence*. Sedangkan bagian *output* menu utama pelatihan terdiri dari 5 submenu, yaitu submenu data, submenu 1-2 *itemset*, submenu *frequent itemset*, submenu *rule*, submenu *lift ratio*. Submenu data menampilkan tabel yang berisi data transaksi sesuai data *input* rentang bulan dan periode yang dipilih. Submenu 1-2 *itemset* menampilkan 1-*itemset* dan 2-*itemset* yang dihasilkan dalam bentuk tabel. Submenu *frequent itemset* menampilkan *frequent itemset* yang dihasilkan dalam bentuk tabel. Submenu *rule* menampilkan *rule* yang dihasilkan dalam bentuk tabel dan submenu *lift ratio* menampilkan hasil perhitungan *lift ratio* dalam bentuk tabel. Gambar rancangan antar muka menu utama pelatihan dapat dilihat pada Gambar 3.16.

Bagian *input* menu utama pengujian terdiri dari rentang bulan dimana secara otomatis sistem akan memberi nilai sesuai dengan rentang bulan yang dipilih pada data *input* pelatihan, dan data *input* pilihan periode.Sedangkan bagian *output* terdiri dari submenu data yang menampilkan tabel data uji. Gambar rancangan antar muka menu utama pengujian dapat dilihat pada Gambar 3.17.



Gambar 3.16 Rancangan Antar Muka Menu Pelatihan



Gambar 3.17 Rancangan Antar Muka Menu Pengujian

3.3. Perhitungan Manual

Perhitungan manual dilakukan pada sampel data yang diambil sebanyak 10 data transaksi dengan minimum *support* 20% dan minimum *confidence* 50%.

Sampel data yang digunakan telah melalui tahap *preprocessing* terhadap kode jenis item, yaitu pada kode faktur yang sama kode jenis item yang sama hanya dianggap satu kemunculan, dapat dilihat pada Tabel 3.4. Untuk jenis item yang ada pada data sampel dijelaskan pada Tabel 3.5 dan untuk golongan item yang ada pada data sampel dijelaskan pada Tabel 3.6. Sedangkan untuk data lengkap Golongan Item ada pada lampiran A.2 halaman 104.

Tabel 3.4 Sampel Data Transaksi *Preprocessing* Jenis Item

Kode Faktur	Tanggal	Kode Jenis Item	Kode Gol Item
650DEV-10090200103	02/09/2010	1601	16
650DEV-10090200103	02/09/2010	2231	22
650DEV-10090200103	02/09/2010	1404	14
650DEV-10090200068	02/09/2010	1601	16
650DEV-10090200068	02/09/2010	2232	22
650DEV-10090200068	02/09/2010	1403	14
650DEV-10090300068	03/09/2010	2232	22
650DEV-10090300068	03/09/2010	1405	14
650DEV-10090300068	03/09/2010	1902	19
650DEV-10090300023	03/09/2010	2232	22

650DEV-10090300023	03/09/2010	2213	22
650DEV-10090300023	03/09/2010	1405	14
650DEV-10090300023	03/09/2010	1113	11
650DEV-10090300074	03/09/2010	1405	14
650DEV-10090300074	03/09/2010	2112	21
650DEV-10090300074	03/09/2010	2131	21
650DEV-10090400056	04/09/2010	2232	22
650DEV-10090400056	04/09/2010	2111	21
650DEV-10090400056	04/09/2010	1402	14
650DEV-10090400071	04/09/2010	2111	21
650DEV-10090400071	04/09/2010	1404	14
650DEV-10090400080	04/09/2010	2111	21
650DEV-10090400080	04/09/2010	1401	14
650DEV-10090400080	04/09/2010	2112	21
650DEV-10090400080	04/09/2010	2131	21

650DEV-10090500089	05/09/2010	2111	21
650DEV-10090500089	05/09/2010	2213	22
650DEV-10090500089	05/09/2010	1404	14
650DEV-10090500089	05/09/2010	2112	21
650DEV-10090500089	05/09/2010	2131	21
Kode Faktur	**Tanggal**	**Kode Jenis Item**	**Kode Gol Item**
650DEV-10090500094	05/09/2010	1702	17
650DEV-10090500094	05/09/2010	1404	14
650DEV-10090500094	05/09/2010	1705	17

Tabel 3.5 Tabel Kode Jenis Item

Kode Jenis Item	Nama Jenis Item
1113	MATEMATIKA
1401	LANGUAGE TEACHING METHODOLOGY
1402	KOMUNIKASI
1403	SASTRA
1404	NOVEL
1405	KOMIK
1601	AGAMA ISLAM

1702	CERITA/DONGENG
1705	TK/PRA TK
1902	MAJALAH
2111	BALLPEN
2112	PENSIL
2131	PERANGKAT ATK
2213	BUKU AGENDA
2221	KERTAS
2223	KERTAS FANCY
2231	ARSIP BUKU
2232	ARSIP KERTAS

Tabel 3.6 Tabel Kode Golongan Item

Kode Golongan Item	Nama Golongan Item
11	TEKNIK
14	BAHASA DAN SASTRA
16	AGAMA
17	BUKU ANAK
19	MEDIA
21	ALAT TULIS KANTOR
22	BUKU DAN KERTAS

Kemudian dilakukan tahap *preprocessing* terhadap golongan item, yaitu setiap kode faktur yang sama kode golongan buku yang sama dianggap satu kemunculan, dapat dilihat Tabel 3.7.

Tabel 3.7 Sampel Data Transaksi *Preprocessing* Golongan Item

Kode Faktur	Tanggal	Kode Gol Item
650DEV-10090200103	02/09/2010	16
650DEV-10090200103	02/09/2010	22
650DEV-10090200103	02/09/2010	14
650DEV-10090200068	02/09/2010	16
650DEV-10090200068	02/09/2010	22
650DEV-10090200068	02/09/2010	14
650DEV-10090300068	03/09/2010	22
650DEV-10090300068	03/09/2010	14
650DEV-10090300068	03/09/2010	19
650DEV-10090300023	03/09/2010	22
650DEV-10090300023	03/09/2010	14
650DEV-10090300023	03/09/2010	11
650DEV-10090300074	03/09/2010	14

650DEV-10090300074	03/09/2010	21
650DEV-10090400056	04/09/2010	22
650DEV-10090400056	04/09/2010	21
650DEV-10090400056	04/09/2010	14
650DEV-10090400071	04/09/2010	21
650DEV-10090400071	04/09/2010	14
650DEV-10090400080	04/09/2010	21
650DEV-10090400080	04/09/2010	14
650DEV-10090500089	05/09/2010	21
650DEV-10090500089	05/09/2010	22
650DEV-10090500089	05/09/2010	14
650DEV-10090500094	05/09/2010	17
650DEV-10090500094	05/09/2010	14

Tahap selanjutnya adalah transformasi dimana data hasil *preprocessing* data diubah bentuk sesuai dengan kebutuhan, hasil transformasi sampel data transaksi dapat dilihat pada Tabel 3.8.

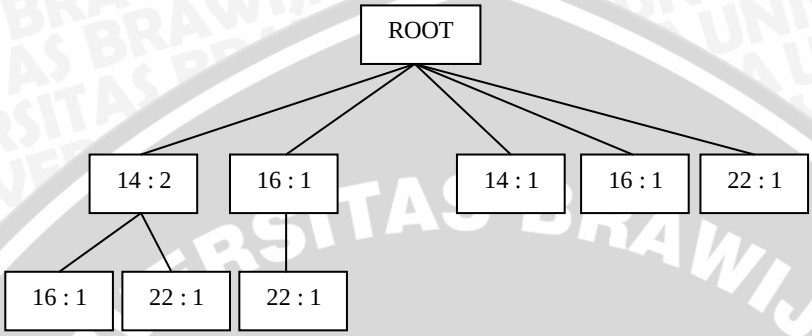
Tabel 3.8 Hasil Transformasi Sampel Data Transaksi

Kode Faktur	Tanggal	Kode Gol Item
650DEV-10090200103	02/09/2010	14, 16, 22
650DEV-10090200068	02/09/2010	14, 16, 22
650DEV-10090300068	03/09/2010	14, 19, 22
650DEV-10090300023	03/09/2010	11, 14, 22
650DEV-10090300074	03/09/2010	14, 21
650DEV-10090400056	04/09/2010	14, 21, 22
650DEV-10090400071	04/09/2010	14, 21
Kode Faktur	**Tanggal**	**Kode Gol Item**
650DEV-10090400080	04/09/2010	14, 21
650DEV-10090500089	05/09/2010	14, 21, 22
650DEV-10090500094	05/09/2010	14, 17

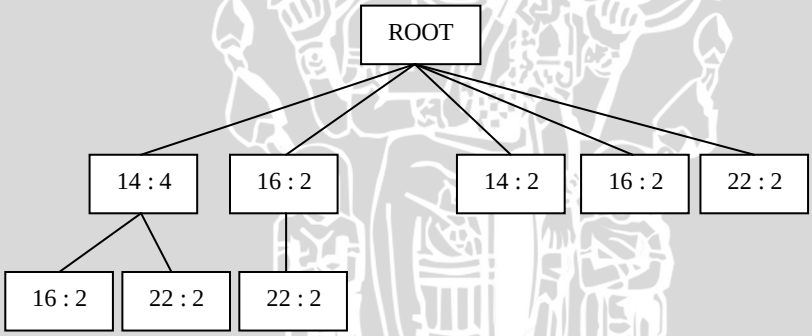
Langkah ketiga adalah tahap *mining* 1 *itemset* dan 2 *itemset.* Proses ini dilakukan dengan membangun struktur data *SOTrieIT* dengan melakukan pembacaan data transaksi yang telah di transformasi, dan tiap data transaksi yang dibaca maka kode golongan item dimasukkan ke dalam *SOTrieIT*. Pembuatan *SOTrieIT* ini dapat dilihat pada Gambar 3.18 sampai 3.27.

Node pada *SOTrieIT* selalu dimulai dengan root yang bernilai null. Kemudian setiap *node* merepresentasikan nama item dan jumlah frekuensi

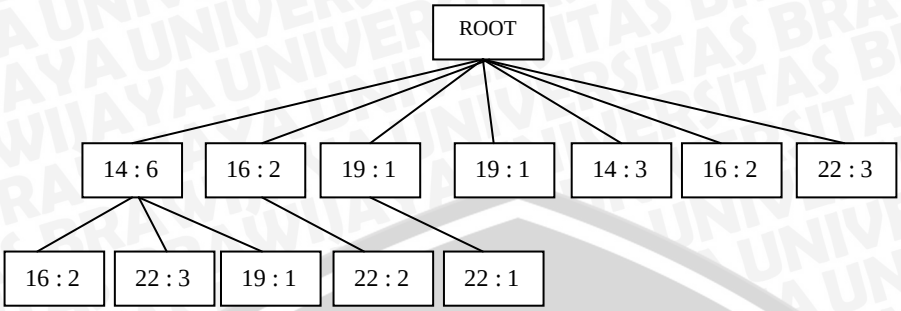
kemunculan. Pembentukan *node* hanya sampai dua tingkat kedalaman, karena hanya sampai 2 *itemset* (maksimal banyaknya item yang berpasangan dalam setiap data transaksi adalah 2 item).



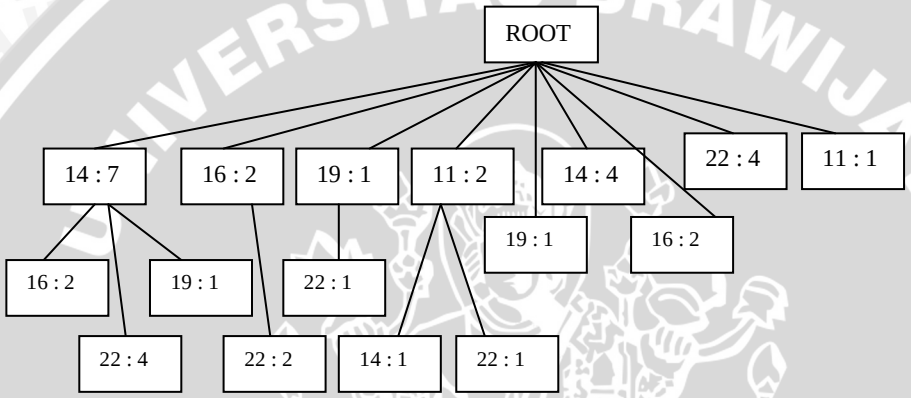
Gambar 3.18 *SOTrieIT* Pembacaan Data Transaksi 650DEV-10090200103



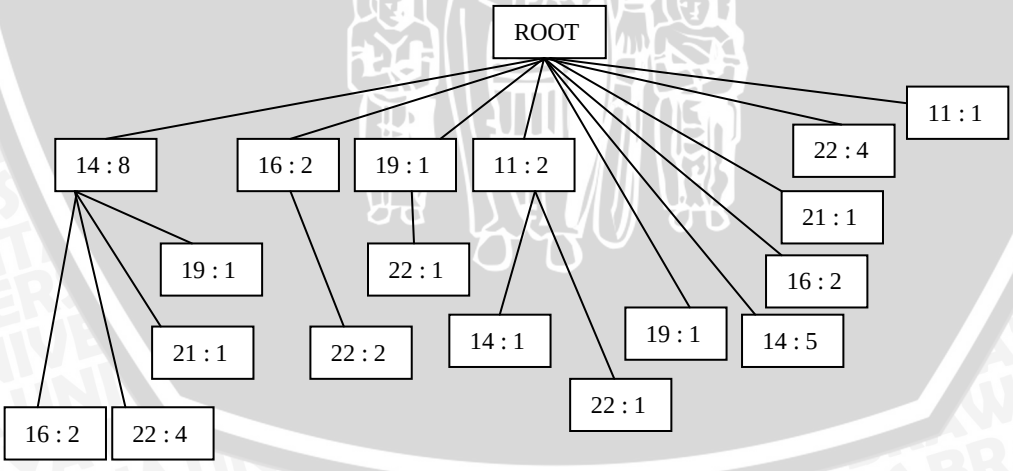
Gambar 3.19 *SOTrieIT* Pembacaan Data Transaksi 650DEV-10090200068



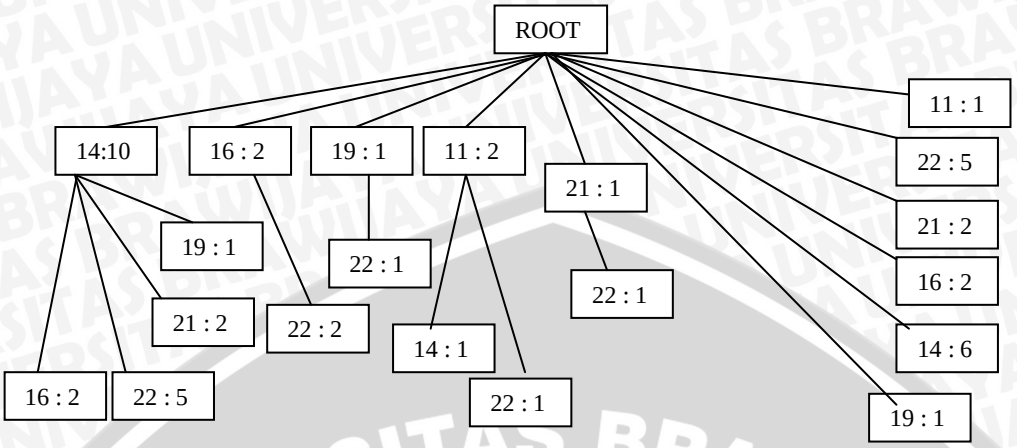
Gambar 3.20 *SOTrieIT* Pembacaan Data Transaksi 650DEV-10090300068



Gambar 3.21 *SOTrieIT* Pembacaan Data Transaksi 650DEV-10090300023

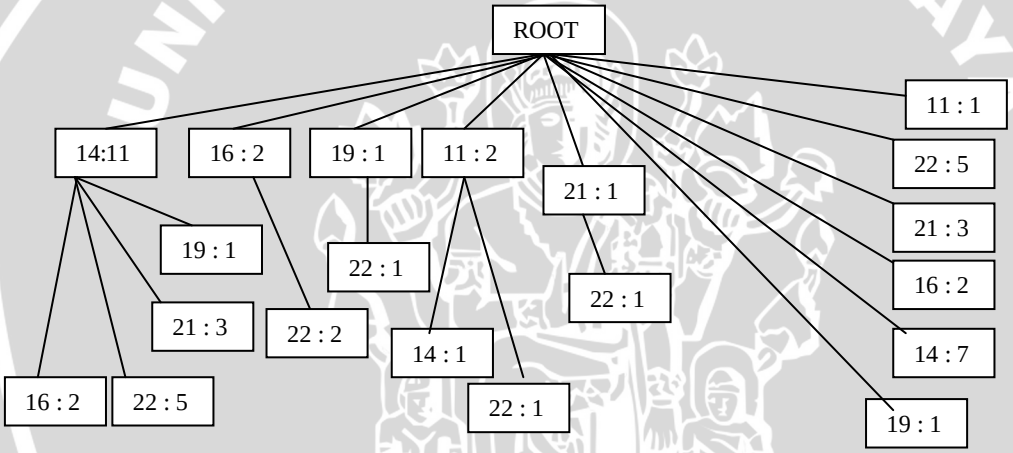


Gambar 3.22 *SOTrieIT* Pembacaan Data Transaksi 650DEV-10090300074



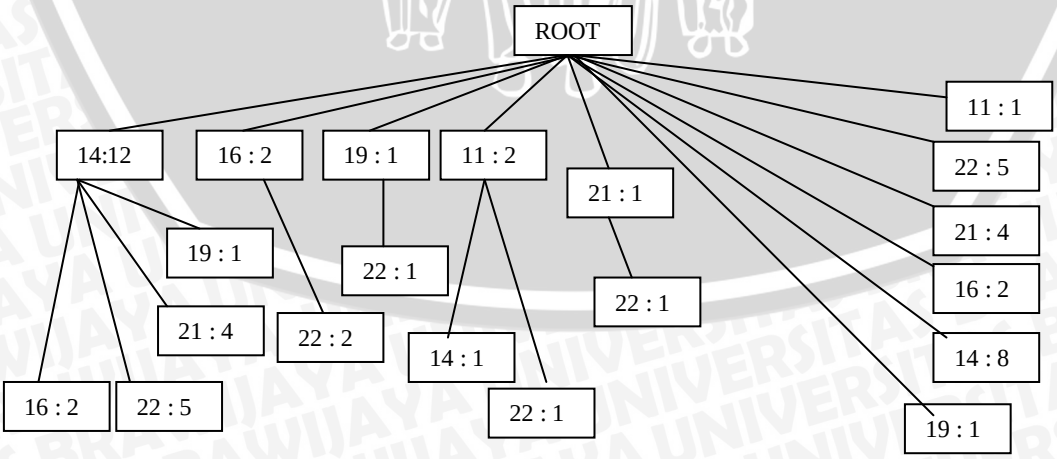
Gambar 3.23 *SOTrieIT* Pembacaan Data Transaksi

650DEV-10090400056



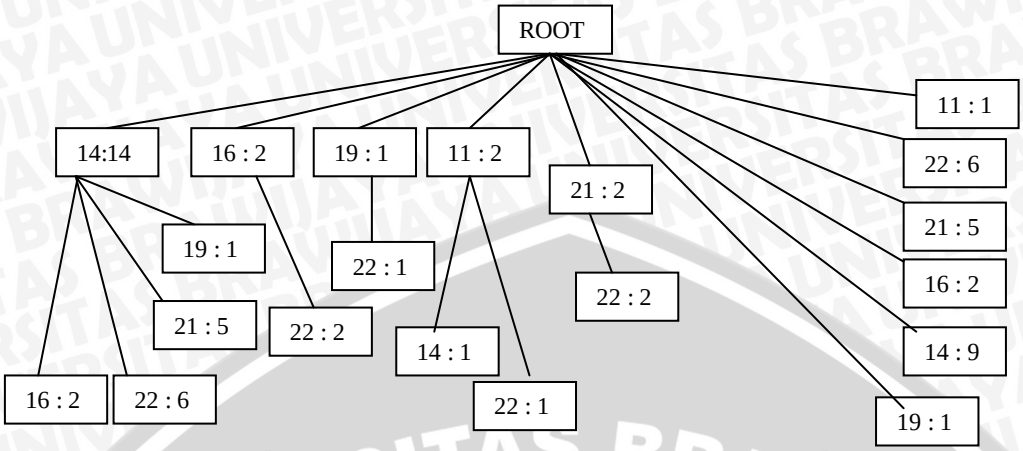
Gambar 3.24 *SOTrieIT* Pembacaan Data Transaksi

650DEV-10090400071

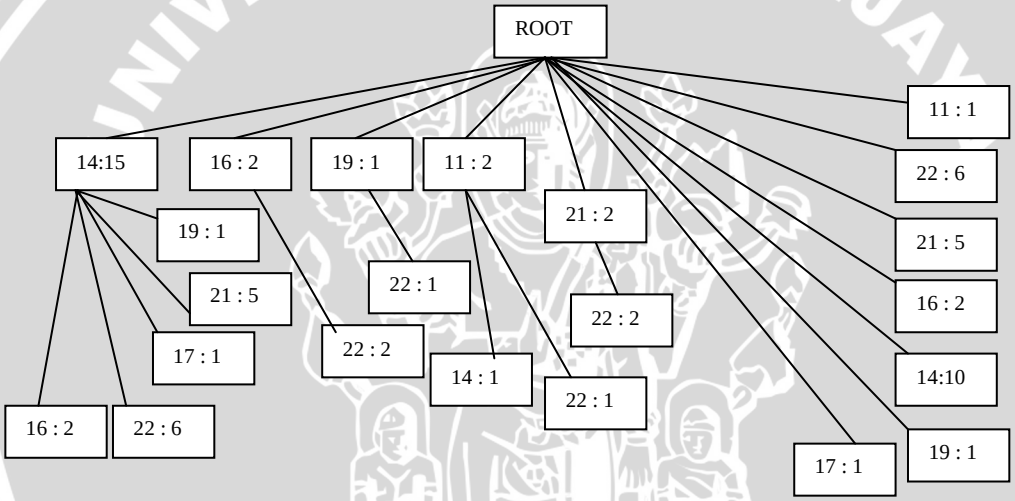


Gambar 3.25 *SOTrieIT* Pembacaan Data Transaksi

650DEV-10090400080



Gambar 3.26 *SOTrieIT* Pembacaan Data Transaksi 650DEV-10090500089



Gambar 3.27 *SOTrieIT* Pembacaan Data Transaksi 650DEV-10090500094

Hasil *SOTrieIT* yang terbangun dari seluruh pembacaan sampel data transaksi ditunjukkan pada Gambar 3.27 dimana berdasarkan *SOTrieIT* ini dapat diketahui 1-*itemset* dan 2-*itemset* yang terbentuk beserta frekuensi kemunculan dari keseluruhan sampel data transaksi. Dengan mengetahui frekuensi, maka dapat dihitung pula nilai *support* untuk masing – masing 1-*itemset* dan 2-*itemset*. Hasil perhitungan *support* 1-*itemset* dapat dilihat pada Tabel 3.9, sedangkan untuk *support* 2-*itemset* dapat dilihat pada Tabel 3.10.

Tabel 3.9 *Support* 1-*itemset*

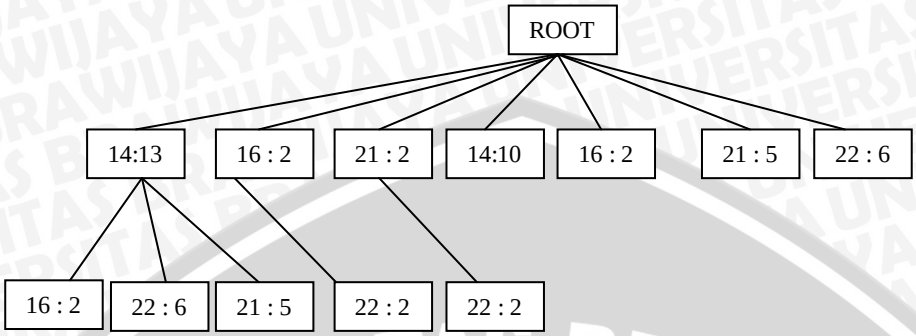
1-*itemset*	Frekuensi	*Support*
11	1	0.1
14	10	1
16	2	0.2
17	1	0.1
19	1	0.1
21	5	0.5
22	6	0.6

Tabel 3.10 *Support* 2-*itemset*

2-*itemset*	Frekuensi	*Support*
11,14	1	0.1
11,22	1	0.1
14,16	2	0.2
14,17	1	0.1
2-*itemset*	**Frekuensi**	***Support***
14,19	1	0.1
14,21	5	0.5
14,22	6	0.6
16,22	2	0.2
19,22	1	0.1
21,22	2	0.2

Setelah dilakukan perhitungan nilai *support* 1-*itemset* dan 2-*itemset* maka dilakukan proses pemangkasan item – item yang tidak *frequent* pada *SOTrieIT*, artinya 1-*itemset* dan 2-*itemset* yang memiliki nilai *support* kurang dari nilai minimum *support* dihapus dari *SOTrieIT* sehingga didapatkan kandidat 1-*itemset* seperti yang dapat dilihat pada Tabel 3.11 dan kandidat 2-*itemset* seperti yang dapat dilihat pada Tabel 3.12. Kandidat – kandidat tersebut diurutkan berdasarkan

nilai *support* dari yang tertinggi ke terendah (*descending*). Hasil pemangkasan ini dapat dilihat pada Gambar 3.28.



Gambar 3.28 *SOTrieIT* Setelah Pemangkasan

Tabel 3.11 Kandidat 1-*itemset*

1-itemset	Frekuensi	*Support*
14	10	1
22	6	0.6
21	5	0.5
16	2	0.2

Tabel 3.12 Kandidat 2-*itemset*

2-itemset	Frekuensi	*Support*
{14,22}	6	0.6
{14,21}	5	0.5
{14,16}	2	0.2
{16,22}	2	0.2
{21,22}	2	0.2

Tahap selanjutnya adalah pembuatan *FP-Tree* untuk kandidat 1-*itemset* dan kandidat 2-*itemset*. Pembuatan awal *FP-Tree* adalah pembuatan *node root* yang bernilai *null*. Selain itu *header table* untuk semua *head of link* bernilai *null* seperti pada Gambar 3.29.

Header Tabel

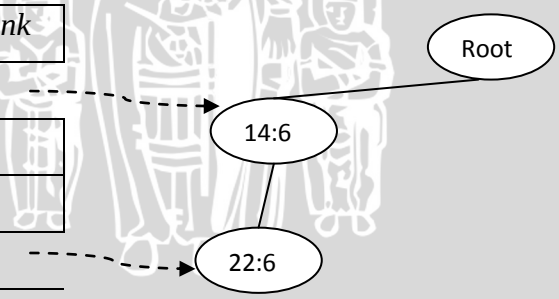
Itemset	*Head of link*
14	*Null*
16	*Null*
21	*Null*
22	*Null*

Gambar 3.29 Pembentukan Awal *FP-Tree*

Data yang dimasukkan pertama adalah data kandidat 2-*itemset* yang pertama yaitu {14,22}. Pada kandidat 1-*itemset* item 14 dengan item 22, *support* yang lebih tinggi nilainya dimiliki oleh item 14 oleh karena itu item 14 masuk terlebih dahulu pada *FP-Tree* dengan melakukan penelusuran pada *node* anak *root*, namun karena *root* belum memiliki anak sama sekali maka dilakukan pembuatan *node* baru dengan nama item 14 dan *support* sebanyak 6. Kemudian *parent* menjadi *node* dengan nama item 14 sehingga item 22 menjadi *node* anak dari *node* dengan nama item 14 dan diberi nilai *support* 6 seperti pada Gambar 3.30.

Header Tabel

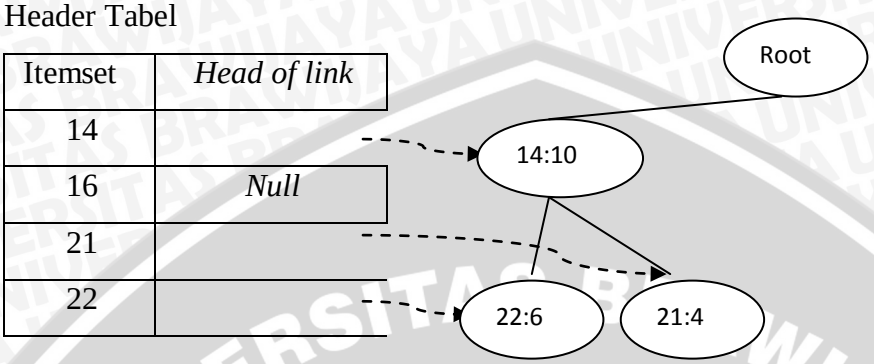
Itemset	*Head of link*
14	
16	*Null*
21	*Null*
22	



Gambar 3.30 *FP-Tree* Setelah Pembacaan Kandidat Pertama

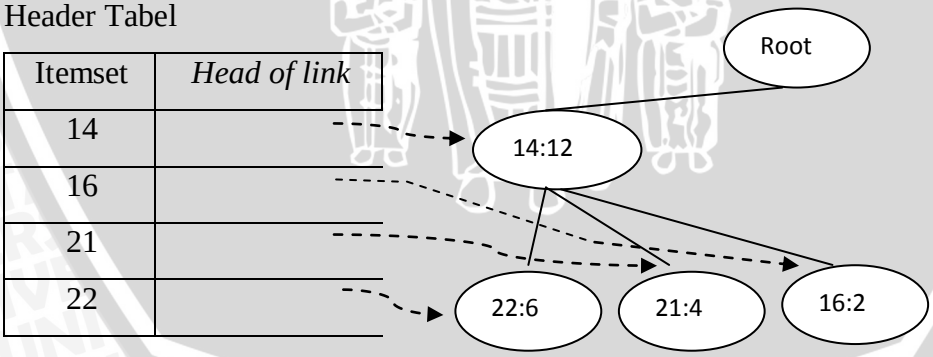
Data kedua adalah data kandidat 2-*itemset* {14,21}. Karena item 14 memiliki nilai *support* lebih tinggi daripada item 21 maka *FP-Tree* melakukan penelusuran pada *node* anak *root* yang nama itemnya 14. Kemudian *node* 14 ditambahkan nilai *support*-nya sebanyak 5, dan *parent* menjadi *node* dengan nama item 14. Karena *node* 14 tidak memiliki *node* anak dengan nama item 21 maka

dibuat *node* dengan nama item 21 yang menjadi *node* anak dari *node* dengan nama item 14 dan diberi nilai *support* 5 seperti pada Gambar 3.31.



Gambar 3.31 *FP-Tree* Setelah Pembacaan Kandidat Kedua

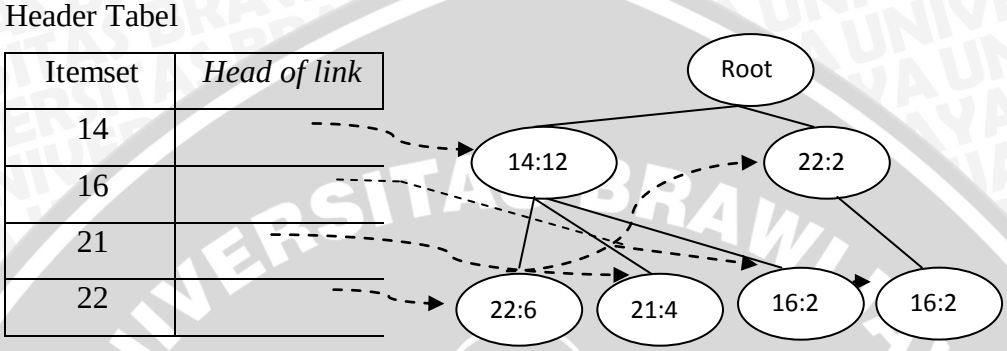
Data ketiga adalah data kandidat 2-*itemset* {14,16}. Karena item 14 memiliki nilai *support* lebih tinggi daripada item 16 maka *FP-Tree* melakukan penelusuran pada *node* anak *root* yang nama itemnya 14. Kemudian *node* 14 ditambahkan nilai *support*-nya sebanyak 2, dan *parent* menjadi *node* dengan nama item 14. Karena *node* 14 tidak memiliki *node* anak dengan nama item 16 maka dibuat *node* dengan nama item 16 yang menjadi *node* anak dari *node* dengan nama item 14 dan diberi nilai *support* 2 seperti pada Gambar 3.32.



Gambar 3.32 *FP-Tree* Setelah Pembacaan Kandidat Ketiga

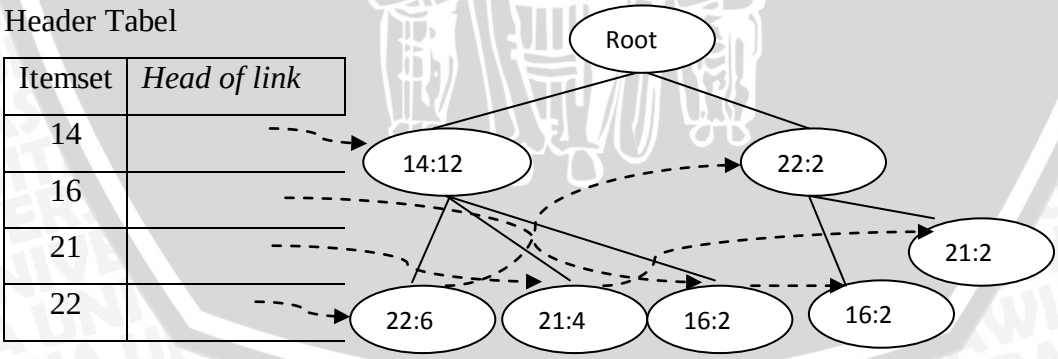
Data keempat adalah data kandidat 2-*itemset* {16,22}. Karena item 16 memiliki nilai *support* lebih rendah daripada item 22 maka *FP-Tree* melakukan penelusuran pada *node* anak *root* yang nama itemnya 22. *Root* tidak memiliki

node anak dengan nama item 22, sehingga dibuat *node* anak *root* baru dengan nama item 22 dan *support* sebanyak 2. *Parent node* sekarang menjadi *node* 22 sehingga dibuat *node* anak dengan nama item 16 dan *support* sebanyak 2 seperti yang dapat dilihat pada Gambar 3.33.



Gambar 3.33 *FP-Tree* Setelah Pembacaan Kandidat Keempat

Data terakhir adalah data kandidat 2-*itemset* {21, 22}. Karena item 21 memiliki nilai *support* lebih rendah daripada item 22 maka *FP-Tree* melakukan penelusuran pada *node* anak *root* yang nama itemnya 22 dan menambahkan nilai *support*-nya sebanyak 2. *Parent node* sekarang menjadi *node* 22 sehingga dibuat *node* anak dengan nama item 21 dan *support* sebanyak 2 seperti yang dapat dilihat pada Gambar 3.34.



Gambar 3.34 *FP-Tree* Setelah Pembacaan Kandidat Kelima

Gambar 3.31 merupakan gambar *FP-Tree* setelah keseluruhan data telah tersimpan pada *tree*. Langkah selanjutnya setelah *FP-Tree* terbentuk, proses *mining frequent itemset* dengan menggunakan algoritma *FP-Growth*. Tahap

pertama yang dilakukan tahap *conditional pattern base*, yaitu mencari lintasan tiap *suffix* item pada *FP-Tree* yang telah terbentuk. Item yang menjadi *suffix* pertama adalah kandidat 1-*itemset* dengan *support* paling kecil. Hasil *conditional pattern base* dapat dilihat pada Tabel 3.13.

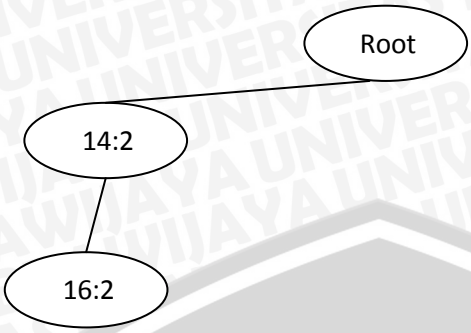
Tabel 3.13 Hasil *Conditional Pattern Base*

Suffix	**Lintasan Awal (*Prefix Path*)**
16	{14 : 2}
21	{22 : 2}, {14 :4}
22	{14 : 6}
14	*Null*

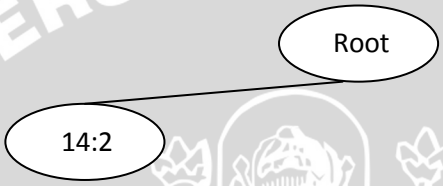
Tahap kedua dari *FP-Growth* adalah *conditional FP-Tree*. Pada tahap ini, *support count* setiap kode item yang sama dalam setiap *suffix* dijumlahkan. Kode item yang memiliki nilai *support* lebih besar atau sama dengan minimum *support* akan dibangkitkan dengan *conditional FP-Tree*.

Suffix {16}	:	*Support* {14}	$= {}^{2}/_{10}$	= 0.2
Suffix {21}	:	*Support* {22}	$= {}^{2}/_{10}$	= 0.2
		Support {14}	$= {}^{4}/_{10}$	= 0.2
Suffix {22}	:	*Support* {14}	$= {}^{6}/_{10}$	= 0.6

Pada Gambar 3.35 menggambarkan lintasan yang memiliki *suffix* item 16, sedangkan *conditional FP-Tree* untuk *suffix* item 16 digambarkan pada Gambar 3.35



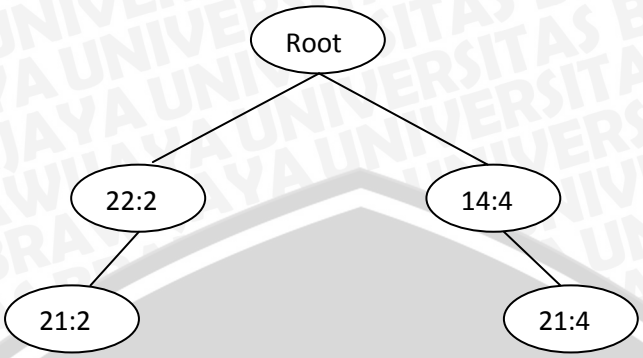
Gambar 3.35 Lintasan yang Memiliki *Suffix* 16



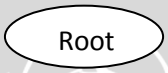
Gambar 3.36 *Conditional FP-Tree Suffix* 16

Berdasarkan Gambar 3.36 karena *prefix path* yang dihasilkan untuk *suffix* item 16 adalah tunggal dan nilai *support* memenuhi nilai minimum *support* maka langsung dibangkitkan menjadi *frequent itemset* {14, 16} dengan *support* sebesar 0.2.

Pada Gambar 3.37 menggambarkan lintasan yang memiliki *suffix* item 21. Berdasarkan Gambar 3.37 karena *prefix path* yang terbentuk tidak tunggal maka dilakukan proses *generate* kembali *conditional FP-Tree* untuk *suffix* {22, 21} yang digambarkan pada Gambar 3.38 dan *suffix* {14, 21} yang digambarakan pada Gambar 3.39.

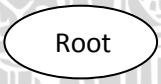


Gambar 3.37 Lintasan yang Memiliki *Suffix* 21



Gambar 3.38 *Conditional FP-Tree Suffix* {22, 21}

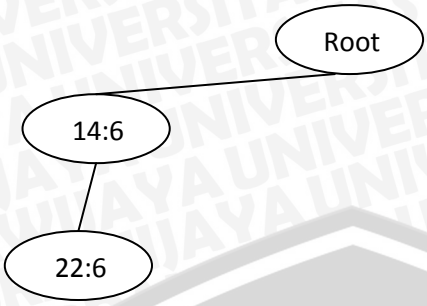
Berdasarkan Gambar 3.34 karena *prefix path* yang dihasilkan untuk *suffix* item {22, 21} adalah *null* sehingga yang menjadi *frequent itemset* adalah dirinya sendiri {22, 21} dengan *support* sebesar 0.2.



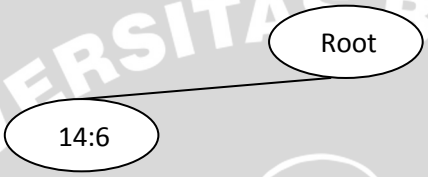
Gambar 3.39 *Conditional FP-Tree Suffix* {14, 21}

Berdasarkan Gambar 3.39 karena *prefix path* yang dihasilkan untuk *suffix* item {14, 21} adalah *null* sehingga yang menjadi *frequent itemset* adalah dirinya sendiri {14, 21} dengan *support* sebesar 0.4.

Pada Gambar 3.40 menggambarkan lintasan yang memiliki *suffix* item 22, sedangkan *conditional FP-Tree* untuk *suffix* item 22 digambarkan pada Gambar 3.41



Gambar 3.40 Lintasan yang Memiliki *Suffix* 22



Gambar 3.41 *Conditional FP-Tree Suffix* 22

Berdasarkan Gambar 3.41 karena *prefix path* yang dihasilkan untuk *suffix* item 22 adalah tunggal dan nilai *support* memenuhi nilai minimum *support* maka langsung dibangkitkan menjadi *frequent itemset* {14, 22} dengan *support* sebesar 0.6.

Hasil *frequent itemset* dari setiap proses conditional *FP-Tree* digabungkan menjadi satu dalam suatu tabel yaitu tabel *frequent itemset* seperti pada Tabel 3.14.

Langkah selanjutnya adalah melakukan perhitungan nilai *confidence* untuk setiap *frequent itemset* yang didapatkan. Hasil perhitungan nilai *confidence* dapat dilihat pada Tabel 3.15. Nilai *confidence* inilah yang akan menentukan kelayakan suatu *frequent itemset* untuk dibangkitkan menjadi *rule*. Nilai *confidence* yang lebih besar atau sama dengan nilai minimum *confidence* dianggap memenuhi nilai minimum *confidence* untuk suatu *frequent itemset* menjadi *rule*.

Tabel 3.14 Hasil *Frequent Itemset*

Frequent Itemset	**Frekuensi**	***Support***
{14,16}	2	0.2
{22,21}	2	0.2
{14, 21}	4	0.4
{14, 22}	6	0.6

Confidence {14} → {16} = $0.2/1$ = 0.2

Confidence {22} → {21} = $0.2/0.$ = 0.3

Confidence {14} → {21} = $0.4/1$ = 0.4

Confidence {14} → {22} = $0.6/1$ = 0.6

Tabel 3.15 Hasil Perhitungan *Confidence*

Frequent Itemset	***Support Frequent Itemset***	***Support Antecedent***	***Confidence***
{14,16}	0.2	1	0.2
{22,21}	0.2	0.6	0.3
{14, 21}	0.4	1	0.4
{14, 22}	0.6	1	0.6

Berdasarkan tabel 3.15 dan dengan nilai minimum *confidence* yang telah ditentukan maka yang memenuhi nilai minimum *confidecence* dan akan dibangkitkan menjadi *rule* adalah *frequent itemset* {14,22} dengan *confidence* sebesar 0.6 atau 60%. *Rule* yang terbentuk dapat dilihat pada Tabel 3.16.

Tabel 3.16 *Rule* yang Terbentuk

Rule	**Keterangan**
14 → 22	Jika Membeli BAHASA DAN SASTRA maka Membeli BUKU DAN KERTAS

Selanjutnya *rule* yang terdapat pada Tabel 3.16 dilakukan uji kekuatan *rule* dengan menggunakan *lift ratio*. Hasil perhitungan nilai *lift ratio* dapat dilihat pada Tabel 3.17.

Lift Ratio {14} → {22} = $0.6/0.6$ = 1

Tabel 3.17 *Lift Ratio Rules*

Rules	***Confidence***	**Frekuensi *Item Consequent***	***Benchmark Confidence***	***Lift Ratio***
14 →22	0.6	6	0.6	1.00

Berdasarkan hasil perhitungan *lift ratio* pada Tabel 3.17, dapat dilihat bahwa *rule* yang terbentuk tidak memiliki kekuatan yang bagus karena nilai *lift ratio rule* tersebut tidak lebih besar dari 1. Sehingga dapat dikatakan bahwa *rule* 14→ 22 bukan merupakan *rule* yang kuat dan mempunyai manfaat.

3.4. Rancangan Uji Coba

Uji coba sistem untuk penerapan algoritma *Fold-Growth* untuk data transaksi penjualan alat tulis dan buku ini akan melakukan evaluasi terhadap hasil analisa yang dihasilkan sistem. Tujuan dari uji coba ini adalah untuk mengetahui pengaruh nilai *minimum support* dan nilai *minimum confidence* yang digunakan terhadap jumlah *rule* yang dihasilkan. Tabel uji untuk pengaruh nilai *minimum support* dan nilai *minimum confidence* pada jumlah *association rule* yang terbentuk dapat dilihat pada Tabel 3.18.

Selanjutnya pada *association rule* yang terbentuk dilakukan uji kekuatan *association rule* dengan menggunakan *lift ratio*. Tujuan dari uji coba ini adalah untuk mengetahui kuat tidaknya *association rule* yang terbentuk. Nilai lift ratio menunjukkan ada atau tidaknya manfaat dari *association rule* yang terbentuk. Tabel uji kekuatan *association rule* dengan menggunakan *lift ratio* yang dihasilkan dapat dilihat pada Tabel 3.19.

Parameter yang digunakan untuk uji pengaruh nilai *minimum support* dan nilai *minimum confidence* terhadap jumlah *rule* yang dihasilkan adalah data periode waktu. Hal tersebut juga digunakan dalam uji kekuatan *rule* dengan menggunakan *lift ratio*. Periode waktu yang digunakan adalah 1 bulan, 2 bulan, dan 3 bulan.

Nilai *minimum support* yang digunakan pada uji pengaruh nilai *minimum support* dan nilai *minimum confidence* terhadap jumlah *rule* yang diterapkan adalah 10% sampai 40% tiap parameter dengan nilai *minimum confidence* 50% sampai 80% tiap nilai *minimum support*.

Tabel 3.18 Tabel Uji Pengaruh Nilai *Minimum Support* dan *Nilai Confidence* terhadap Jumlah *Rule* yang Dihasilkan

Periode	*Minimum Support* (%)	*Minimum Confidence* (%)	Jumlah Rule yang dihasilkan

Tabel 3.19 Tabel Uji *Lift Ratio Association Rule*

Periode	*Rules*	*Confidence*	*Benchmark Confidence*	*Lift Ratio*

BAB IV
IMPLEMENTASI

4.1. Lingkungan Implementasi

Implementasi bertujuan untuk melakukan transformasi representasi rancangan ke dalam bahasa pemrograman sehingga dapat menjadi sistem yang dimengerti oleh komputer. Pada bab ini akan dijelaskan lingkungan implementasi yang meliputi lingkungan implementasi perangkat keras dan lingkungan implementasi perangkat lunak.

4.1.1. Lingkungan Perangkat Keras

Perangkat keras yang digunakan pada penelitian ini adalah sebuah PC(*Personal Computer*) dengan spesifikasi sebagai berikut :

1. *Processor* Intel® Core2 Duo 2.10GHz
2. Memori 2GB
3. *Harddisk* dengan kapasitas 320 GB
4. Monitor 14.0”
5. *Keyboard*
6. *Mouse*

4.1.2. Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan pada penelitian ini adalah sebagai berikut :

1. Sistem operasi *Microsoft Windows*8.
2. *Netbeans IDE* 6.8 dan JDK 1.6 dalam mengembangkan aplikasi untuk analisis pola penjualan buku.
3. MySQL dan JDBC

4.2. Implementasi Program

Berdasarkan rancangan pembuatan sistem pada bab 3, maka pada subbab ini akan dijelaskan implementasi proses – proses tersebut. Secara garis besar proses dikelompokkan menjadi 4 tahap, yaitu tahap *frequent 1-itemset* tahap

frequent itemset, tahap *generate rule*, dan tahap *lift ratio*. Implementasi program terdiri dari 18 *class* yang dijelaskan pada Tabel 4.1 beserta fungsinya.

Tabel 4.1 *Class* Pada Implementasi Program dan Fungsinya

Nama *Class*	Fungsi
Class HapusRedundansi	Menghapus kode golongan item(buku dan alat tulis) yang sama pada kode transaksi yang sama untuk proses *preprocessing*.Terdiri dari *method* test dan *method* main.
Class Data	Membaca data transaksi, membaca data golongan item (buku dan alat tulis), dan proses *frequent 1-itemset*. Terdiri dari *method* stat, *method* connect, *method* Transaksi, *method* getTrans, *method* getFrek, *method* urutFrek, *method* jumlahTrans, *method* oneItemset, *method* jumlahOneItemset, *method* transMinsup, *method* getJB, *method* maxSupport, *method* itungSupport.
Class Frek	Menyimpan data frekuensi semua kode golongan item (buku dan alat tulis). Terdiri dari *constructor* Frek.
Class JenisBuku	Menyimpan data golongan item (buku dan alat tulis).Terdiri dari *constructor* JenisBuku.
Class Support	Menyimpan data nilai *support* semua kode golongan item (buku dan alat tulis).Terdiri dari *constructor* Support.
Class Trans	Menyimpan data transaksi berupa kode transaksi dan kode golongan item (buku dan alat tulis) yang dimiliki tiap kode transaksi. Terdiri dari *constructor* Trans
Class Path	Menyimpan data lintasan. Terdiri dari *constructor* Path.
Class Prefix	Menyimpan data *prefix* dari semua *suffix*. Terdiri dari *constructor* Prefix.
Class FrequentItemset	Menyimpan data *frequent itemset*. Terdiri dari *constructor* FrequentItemset.
Class Confidence	Menyimpan data nilai *confidence*. Terdiri dari *constructor* Confidence.
Class Rule	Menyimpan data *rule*. Terdiri dari *constructor* Rule.
Class LiftRatio	Menyimpan data nilai *lift ratio*. Terdiri dari *constructor* LiftRatio.
Class TreeNode	Menyimpan data informasi *node* pada *Fold-Growth*. Terdiri dari *constructor*TreeNode, *method* displayNode, *method* NodeNext, *method* NodeSibling, *method* NodeParent.
Class TreeHeaderEntry	Sebagai *header table* yang menyimpan informasi total bobot tiap kode golongan item (buku dan alat tulis) yang menjadi *frequenst 1-2 itemset* dalam SOTreeIT dan yang menjadi ordered itemset dalam FP-Tree.Terdiri dari *constructor* TreeHeaderEntry, *method* DisplayHeaderEntry.
Class FPTree	Implementasi proses untuk tahap *frequent itemset*, proses untuk tahap koleksi *rule*, dan proses untuk tahap *lift ratio*.

	Terdiri dari *constructor* FPTree *method* insert, *method* cari, *method* simpanPath, *method* cariPrefix, *method* SupportPref, *method* ConditionalFPTree, *method* urutFI, *method* cekSupport, *method* FreqIt, *method* jumlahFrequentItemset, *method* ConfidenceFI, *method* JumlahRule, *method* RuleFI, *method* descRule, *method* hitungLift, *method* LiftRatio, *method* RuleConf.
Class SOTreeIT	Implementasi proses untuk tahap mendapatkan *frequent 1-2 itemset*. Terdiri dari *constructor* SOTreeIT, *method* insert, *method* cari, *method* simpanPath, *method* kombinasi, *method* pangkasItemset, *method* seleksi1Itemset, *method* urut1Itemset, *method* seleksi2Itemset, *method* jumlahTwoItemset.
Class Interface	*Main* program dan mengatur tampilan GUI. Terdiri dari *constructor* Interface, *method* tabelData, *method* oneItemset, *method* FrequentItemset, *method* Rule,*method* LiftRatio, *method* jComboBox1ActionPerformed, *method* refreshFreqOne, *method* refreshFreqItem, *method* refreshRule, *method* refreshLift, *method* jButton2ActionPerformed, *method* jButton1ActionPerformed, *method* jSpinner1StateChanged, *method* jSpinner2StateChanged, *method* jSpinner3StateChanged, *method* jRadioButton1MouseClicked, *method* jRadioButton2MouseClicked, *method* jRadioButton3MouseClicked, *method* jRadioButton4MouseClicked, *method* jRadioButton5MouseClicked, *method* jRadioButton6MouseClicked, *method* main.
Class ExportData	Mengekspor data dari *jTable* ke *excel*. Terdiri dari *method* exportTable.

4.2.1. Implementasi Tahap SOTreeIT

Pada tahap ini, dilakukan pembuatan SOTreeIT dari pembacaan data transaksi. Pembuatan SOTreeIT dilakukan dengan pengkoleksian tiap golongan item(buku dan alat tulis)yang disebut 1-*itemset*.Selanjutnya dilakukan pengkoleksian dua kombinasi dari setiap kode golongan item dalam setiap baris data transaksi yang disebut 2-*itemset*. Data 1-2 *itemset* selanjutnya dilakukan pemangkasan berdasarkan frekuensi kemunculannya pada data transaksi untuk menjadi *ordered frequent itemset. Ordered frequent itemset* merupakan *frequent*

1-*itemset* dan *frequent* 2-*itemset.* Tahap *frequent 1-itemset* dilakukan dengan mengoleksi setiap data golongan itemyang memiliki jumlah frekuensi terhadap jumlah data transaksi lebih besar atau sama dengan minimum *support*. Tahap *frequent 2-itemset* dilakukan dengan mengoleksi 2 kombinasi dari setiap kode golongan item yang memiliki jumlah frekuensi terhadap jumlah data transaksi lebih besar atau sama dengan minimum *support*. Tahap SOTreeIT ini terdiri dari tahap pembacaan data transaksi, tahap 1-*itemset*, tahap pembuatan *SOTreeIT*, tahap koleksi*ordered itemset*.

4.2.1.1. Implementasi Pembacaan Data Transaksi

Proses awal yang dilakukan adalah pembacaan data transaksi dalam periode waktu tertentu dan *load* data transaksi yang memiliki pembelian golongan item (buku dan alat tulis) lebih dari 3. Implementasi dari proses ini dapat ditunjukkan pada *source code* 4.1.

```
Temporary Transaksi(int periode, int bulan)
    {
        Temporary Save = new Temporary();
        String[] siBulan =
{"01","02","03","04","05","06","07",
"08","09","10","11","12"};
        int[] mulai = {0,1,2,3,4,5,6,7,8,9,10,11};
        bulan -= 1;
        int start = periode*bulan;
        int selesai = start + periode;
        try{
            String sql = "";
            for(int i=mulai[start];i<selesai;i++)
            {
                if(i==mulai[start])
                {
                    sql = "SELECT * FROM trans WHERE
(kd_trans LIKE '65%-
10"+siBulan[i]+"%'";
                }else{
                    sql = sql+" OR kd_trans LIKE '65%-
10"+siBulan[i]+"%'";
                }
            }
            sql = sql+")"+"GROUP BY kd_trans
HAVING count(kd_trans)>3";
            Save.kd_trans = stat(sql ,"kd_trans");
        } catch  (Exception e)
        {
            e.printStackTrace();
        }
        Save = new Temporary(Save.kd_trans);
        return Save;
    }

    Trans Transaksi(Temporary temp)
    {
        Trans T = new Trans();
        Object[] siKode = temp.kd_trans.toArray();
        String sql="";
        try{
            sql = "SELECT * FROM trans
WHERE kd_trans ='"+siKode[0]+"'";
            for (int i=1; i<siKode.length;i++){
                sql = sql+" OR kd_trans ='"+siKode[i]+"'";
            }
            T.kd_trans = stat(sql ,"kd_trans");
            T.kd_jb = stat(sql ,"kd_gb");
        }catch(Exception e){
            e.printStackTrace();
        }
        T = new Trans(T.kd_trans, T.kd_jb);
        return T;
    }
```

***Source Code* 4.1** Proses Pembacaan Data Transaksi

Fungsi Temporary Transaksi berfungsi untuk mengkoleksi kode transaksi dari tabel trans pada *database*, yang memiliki golongan item minimal 4 pada rentang bulan dan periode yang telah ditentukan. Kode transaksi hasil koleksi disimpan dalam ArrayList<String> kd_trans pada *class*Temporary yang dideklarasikan dengan nama *object*Save.

Fungsi dari Trans Transaksi berfungsi untuk mengkoleksi informasi kode golongan item dari kode transaksi hasil koleksi fungsi Temporary Transaksi yang menjadi parameter inputan dari fungsi ini yaitu Temporary temp. Hasil koleksi berupa kode transaksi dan kode golongan item. Kode transaksi disimpan dalam ArrayList<String> kd_trans sedangkan kode golongan item disimpan dalam ArrayList<Integer> kd_jb pada *class* Trans yang dideklarasikan dengan nama *object* T.

Pada kedua fungsi tersebut dilakukan pemanggilan fungsi stat yang berfungsi untuk mengeksekusi perintah SQL. Parameter dari fungsi stat adalah perintah SQL bertipe *String* dan nama kolom yang ingin diambil dari hasil eksekusi perintah SQL.

4.2.1.2. Implementasi 1-*itemset*

Proses selanjutnya adalah mengkoleksi kode golongan item yang ada pada data transaksi yang memiliki frekuensi kemunculan lebih dari sama dengan nilai *minimum support* (nilai batas minimal frekuensi kemunculan) dari masukan yang diberikan. Implementasi dari proses ini dapat ditunjukkan pada *source code* 4.2.

Pada fungsi getFrek, parameter inputan adalah data transaksi Trans T. Proses pada fungsi ini dilakukan dengan pembacaan setiap data transaksi dan diambil kode golongan itemnya pada index data transaksi yang sedang dibaca dengan perintah T.kd_jb.get(i). Setiap index data transaksi, dilakukan pengecekanpada *object*F, yang merupakan deklarasi dari *class* Frek. Apabila kode golongan buku belum ada di ArrayList<String> kodeItem pada *object* Fmaka kode golongan item tersebut disimpan dalam ArrayList<String> kodeItem dengan perintah F.kodeItem.add dan ArrayList<Integer> jumlahItem yang merupakan nilai frekuensi golongan item tersebut diberi nilai 1 dengan perintah F.jumlahItem.add(1)

Kode golongan item yang sudah ada pada F.kodeItem maka index dari F.kodeItem yang mengandung kode golongan buku tersebut disimpan dalam variabel pos. Kemudian frekuensi golongan buku tersebut diambil pada F.jumlahItem dengan index pos dan disimpan pada temp. Nilai frekuensi pada F.jumlahItem index pos dihapus kemudian pada index pos tersebut ditambahkan nilai frekuensi baru yaitu temp+1.

Pada fungsi itungSupport dilakukan perhitungan nilai *support* tiap kode golongan item dari nilai frekuensi kemunculannya dibagi dengan total transaksi. Kemudian nilai *support* tiap kode golongan item dibandingkan dengan nilai minimum *support* yang telah ditentukan. Golongan item yang memiliki nilai *support* lebih dari atau sama dengan minimum *support* yang akan menjadi *frequent 1-itemset*. Dalam menghitung nilai *support* dibutuhkan total transaksi yang disimpan dalam parameterjml. Pembacaan data Frek asli dilakukan setiap index data yang ada. Setiap index selanjutnya dilakukan perhitungan nilai frekuensinya asli.jumlahItem.get(i) dibagi dengan total transaksi jml. Hasil dari perhitungan nilai *support* tersebut disimpan dalam ArrayList<Double> bobot sedangkan golongan item dari asli.kodeItem disimpan ke dalam ArrayList<String> kodeItem pada *class* Support yang dideklarasikan dengan nama *object* S.

Fungsi oneItemset berfungsi untuk mengkoleksi kode golongan item yang memiliki nilai *support* lebih dari atau sama dengan nilai minimum *support*. Data nilai *support* dan data kode golongan item yang merupakan *arraylist* pada *object class* Support sdikonversi ke bentuk *array*. Nilai minimum *support* dihitung terlebih dahulu karena parameter inputan double persen merupakan minimum *support* yang diberikan penggunadalam bentuk prosentase. Kode golongan item yang memiliki nilai *support* lebih dari atau sama dengan minimum *support*minsup, disimpan ke dalam ArrayList<String> k untuk kode golongan item dan ke dalam ArrayList<Double> j untuk nilai *support* kode golongan tersebut. Selanjutnya ArrayList<String> k dan ArrayList<Double> j disimpan ke dalam *class* Supportdengan nama *object* hasil.

```
Frek getFrek(Trans T)
{
Frek F = new Frek();
for (int i =0; i<T.kd_trans.size();i++)
{
    if (F.kodeItem.contains(String.valueOf(T.kd_jb.get(i))))
    {
    int pos = F.kodeItem.indexOf(T.kd_jb.get(i));
    int temp = F.jumlahItem.get(pos);
    F.jumlahItem.remove(pos);
    F.jumlahItem.add(pos , temp+1);
    }else{
    F.kodeItem.add(String.valueOf(T.kd_jb.get(i)));
    F.jumlahItem.add(1);
    }
}
return F;
}

Support itungSupport(Frek asli, int jml){
Support S = new Support();
for(int i=0;i<asli.kodeItem.size();i++){
    S.bobot.add(i,     (double)     (asli.jumlahItem.get(i)     /
(double)jml));
}
S.kodeBukunya = asli.kodeItem;
return S;
}

Support oneItemset(Support s, double persen){

        Object[] sup = s.bobot.toArray();
        Object[] kd = s.kodeBukunya.toArray();
        double minsup = persen/100;
        ArrayList<String> k = new ArrayList<String>();
        ArrayList<Double> j = new ArrayList<Double>();
        for(int i=0;i<kd.length;i++)
        {
            if((Double)sup[i]>=minsup)
            {
                k.add((String)kd[i]);
                j.add((Double)sup[i]);
            }
        }
        Support hasil = new Support(k,j);
        return hasil;
    }
```

***Source Code* 4.2**Proses *1-itemset*

4.2.1.3. ImplementasiPembuatan *SOTreeIT*

Pada proses ini, dilakukan pembuatan *SOTreeIT* untuk mendapatkan *ordered itemset*. Proses ini dilakukan dengan mengkombinasi setiap golongan item pada tiap transaksi, menjadi kombinasi 2 item, dan kemudian dimasukkan ke

dalam struktur data *SOTreeIT* sehingga apabila terdapat hasil kombinasi 2 item yang sama pada transaksi lain maka data akan dimampatkan dalam node yang sama. Implementasi proses pembuatan *SOTreeIT* ditunjukkan pada *Source Code* 4.3.

```
ArrayList<ArrayList<Integer>> kombinasi
(ArrayList<Integer> item)
{
ArrayList<ArrayList<Integer>> itema = new
ArrayList<ArrayList<Integer>>();
ArrayList<Integer> titip;
ArrayList<Integer> titip1;
for(int i=0;i<item.size();i++){
for(int j=i+1;j<item.size();j++){
titip = new ArrayList<Integer>();
titip.add(item.get(i));
titip.add(item.get(j));
itema.add(titip);
}
}
for(int k=0;k<item.size();k++){
titip1 = new ArrayList<Integer>();
titip1.add(item.get(k));
itema.add(titip1);
}
return itema;
}

void insert(ArrayList<Integer> items, int count)
{
TreeNode current_node = root;
for (int index = 0; index < items.size(); index++)
{
     int entry_index =
     ((Integer)item2index.get(new
Integer(items.get(index)))).intValue();
header[entry_index].count += count;
TreeNode walker = current_node.child;
for ( ; walker != null; walker = walker.sibling)
if (walker.item == items.get(index))
break;
if (walker == null)
{
if (current_node.child != null)
hasMultiplePaths = true;
count_nodes++;
      TreeNode new_node= new TreeNode(items.get(index), count,
index + 1, entry_index, count_nodes,current_node,
       current_node.child,header[entry_index].head);
       header[entry_index].head = new_node;
       current_node.child = new_node;
       current_node = new_node;
}else{
walker.count += count;
current_node = walker;
```

```
}
}
}
```

***Source Code* 4.3** Proses *SOTreeIT*

Pada fungsi kombinasi, parameter yang dibutuhkan berupa golongan item untuk tiap transaksi yang diinisialisasi ArrayList<Integer> item. Pada implementasinya item merupakan data transaksi yang hanya mengandung golongan item yang menjadi *1-itemset*. Nilai balikan dari fungsi ini berupa ArrayList<ArrayList<Integer>> itema yang berisi dua kombinasi dari golongan item tiap transaksi dan golongan item itu sendiri. Dalam menggabungkan tiap golongan item menjadi kombinasi dua golongan item, setiap index golongan item dibaca dan disimpan pada ArrayList<Integer> titipyang secara bersamaan dilakukan pembacaan golongan item untuk index selanjutnya menggunakan *looping* bersarang dan disimpan pada ArrayList<Integer> titip sehingga pada titip terdapat kombinasi yang selanjutnya disimpan pada itema. Setelah setiap index golongan item di kombinasi, selanjutnya dilakukan penyimpanan tiap – tiap golongan item itu sendiri melalui ArrayList<Integer> titip1 yang selanjutnya disimpan dalam satu struktur data itema agar semua kandidat baik *1-2 itemset* terkoleksi di struktur data yang sama.

Pembentukan *SOTreeIT*diimplementasikan pada *class SOTreeIT*yaitu pada fungsi *insert*. Pada fungsi ini, parameter inputan berupa ArrayList<Integer> items yang merupakan semua kode golongan item yang dimiliki pada suatu kode transaksi dan int count yang merupakan jumlah bobot kode golongan item. Proses awal adalah inisialisasi *node root*. Kemudian dilakukan pembacaan tiap index data kode golongan item pada items, yang berarti kode golongan item pada index tersebut dimasukkan ke dalam struktur *SOTreeIT*.Setelah index dari kode golongan item didapatkan, maka pada *header table* data pada index tersebut ditambahkan 1 nilai *count*-nya. Proses selanjutnya adalah inisialisasi *node* pointer dengan nama walker dimana walker merupakan anak dari *node* sekarang. Selama *node* sekarang memiliki anak maka pointer akan menulusuri semua saudara dari anak *node* sekarang. Selama penelusuran, dilakukan pengecekan kode golongan item pada *node* tersebut sama atau tidak dengan kode golongan item yang dimasukkan ke dalam *SOTreeIT*. Jika ada yang sama maka posisi pointer berada

pada *node* tersebut, sedangkan bila tidak ada yang sama berarti belum pernah terbentuk *node* dengan kode golongan item yang dimasukkan ke dalam *FP-tree* yang menjadi anak dari *node* sekarang atau walker bernilai null.

Jika walker bernilai null maka diciptakan *node* baru yang dideklarasikan dengan nama new_node yang menjadi anak dari *node* sekarang dan *head node* pada *header table* untuk kode golongan buku tersebut. Kemudian *node* sekarang adalah *node* baru. Sedangkan, untuk walker yang tidak bernilai null maka bobot pada *node* yang dirujuk oleh pointer walker ditambahkan 1 dan *node* sekarang adalah *node* yang dirujuk oleh pointer walker.

4.2.1.4. Implementasi Koleksi*Ordered Itemset*

Implementasi dari proses *Ordered Itemset* dapat ditunjukkan pada S*ource Code* 4.4. Pada proses ini, dilakukan pengkoleksian *1-itemset* dan *2-itemset* yang memiliki nilai frekuensi kemunculan lebih dari atau sama dengan batas nilai minimal frekuensi kemunculan dari keseluruhan total transaksi (*minimum support*).

```
OrderedItemset PangkasItemset
(ArrayList<Path> pa,int jumlah,double persen)
{
OrderedItemset O= new OrderedItemset();
minsup = persen/100;
for(int i=0;i<pa.size();i++)
{
    double support =
    (double) pa.get(i).bobot.get(pa.get(i).kd_jb.size()
    -1)/(double) jumlah;
if(support>=minsup)
{
O.item.add(pa.get(i).kd_jb);
O.bobot.add
(pa.get(i).bobot.get(pa.get(i).kd_jb.size()-1));
O.support.add(support);
}
}
return O;
}
```

***Source Code* 4.4** Proses Koleksi *Ordered Itemset*

4.2.2. Implementasi Tahap *Frequent Itemset*

Pada tahap ini, dilakukan kombinasi kode golongan item yang menjadi *frequent 1-itemset* dan kemudian dilakukan perhitungan *support* untuk masing – masing kombinasi. Kombinasi yang memiliki nilai *support* lebih besar atau sama dengan nilai minimum *support* yang telah ditentukan, akan menjadi *frequent itemset*. Tahap *frequent itemset* terdiri dari 4 proses, yaitu proses pembentukan *FP-tree*, proses *conditional pattern base*, proses *conditional FP-tree*, dan proses koleksi *frequent itemset*.

4.2.2.1. Implementasi Pembentukan *FP-tree*

Proses awal yang dilakukan adalah pembentukan *FP-tree*. Proses ini bertujuan untuk menyimpan data transaksi sehingga tidak perlu dilakukan pembacaan data transaksi berulang kali. Pada proses ini kode golongan item setiap transaksi dimasukkan ke dalam *tree* sebagai satu lintasan. Node dari *FP-tree* menyimpan informasi kode golongan item, bobot, kedalaman, indeks pada *header table*, indeks node, node *parent*, node *sibling*, dan node *link*. Implementasi dari proses ini dapat ditunjukkan pada *source code* 4.6.

Pembentukan *FP-tree* diimplementasikan pada *class FPTree* yaitu pada fungsi *insert*. Pada fungsi ini, parameter inputan berupa ArrayList<Integer> items yang merupakan semua kode golongan item yang dimiliki pada suatu kode transaksi dan int count yang merupakan jumlah bobot kode golongan item. Proses awal adalah inisialisasi *node root*. Kemudian dilakukan pembacaan tiap index data kode golongan item pada items, yang berarti kode golongan item pada index tersebut dimasukkan ke dalam struktur *FP-tree*.

```
void insert(ArrayList<Integer> items,int count){
       FPTreeNode current_node = root;
       for (int index=0; index<items.size();
           index++){
..........................      int entry_index=
           ((Integer)item2index.get(new
           Integer(items.get(index)))).intValue();
           header[entry_index].count += count;
           FPTreeNode walker = current_node.child;
            for(;walker!=null;walker= walker.sibling)
```

```
.......        if (walker.item == items.get(index))
.................................          break;
          if(walker == null){
............        if (current_node.child != null)
                 hasMultiplePaths = true;
                 count_nodes++;
...................          FPTreeNode new_node=
                 new FPTreeNode(items.get(index),
                 count,index+1,entry_index,count_nodes,
.........           current_node,current_node.child,
                 header[entry_index].head);
....             header[entry_index].head = new_node;
...........           current_node.child = new_node;
.................          current_node = new_node;
.....................................     }else{
......................  walker.count += count;
             current_node = walker;}
       }
}
```

***Source Code* 4.6** Proses Pembentukan *FP-tree*

Kode golongan item yang dimasukkan pada struktur *FP-tree*, diambil indexnya pada item2index yang menyimpan index kode golongan item pada *header table*. *Header table* merupakan tabel informasi mengenai node kode golongan item yang sama yaitu yang menyimpan index node tiap kode golongan item, kode golongan itemnya, total frekuensi yang telah masuk ke dalam *FP-tree* tiap kode golongan item, dan *head node* dari tiap kode golongan item.

Setelah index dari kode golongan item didapatkan, maka pada *header table* data pada index tersebut ditambahkan 1 nilai *count*-nya. Proses selanjutnya adalah inisialisasi *node* pointer dengan nama walker dimana walker merupakan anak dari *node* sekarang. Selama *node* sekarang memiliki anak maka pointer akan menulusuri semua saudara dari anak *node* sekarang. Selama penelusuran, dilakukan pengecekan kode golongan item pada *node* tersebut sama atau tidak dengan kode golongan item yang dimasukkan ke dalam *FP-tree*. Jika ada yang sama maka posisi pointer berada pada *node* tersebut, sedangkan bila tidak ada yang sama berarti belum pernah terbentuk *node* dengan kode golongan item yang dimasukkan ke dalam *FP-tree* yang menjadi anak dari *node* sekarang atau walker bernilai null.

Jika walker bernilai null maka diciptakan *node* baru yang dideklarasikan dengan nama new_node yang menjadi anak dari *node* sekarang dan *head node*

pada *header table* untuk kode golongan item tersebut. Kemudian *node* sekarang adalah *node* baru. Sedangkan, untuk walker yang tidak bernilai null maka bobot pada *node* yang dirujuk oleh pointer walker ditambahkan 1 dan *node* sekarang adalah *node* yang dirujuk oleh pointer walker.

4.2.2.2. Implementasi *Conditional Pattern Base*

Pada proses ini, dilakukan pengkoleksian lintasan yang dimiliki tiap *suffix*, dimana *suffix* merupakan kode golongan item yang menjadi *frequent 1-itemset*. Selain itu, proses ini juga menyimpan bobot tiap node pada lintasan tersebut dengan tujuan untuk digunakan dalam menghitung nilai *support* tiap kode golongan item yang menjadi *prefix* pada tiap *suffix*. Implementasi dari proses ini dapat ditunjukkan pada *source code* 4.7.

Fungsi cariPrefix memiliki parameter inputan kode golongan item yang menjadi *suffix*int suffix dan data lintasan ArrayList<Path> p. Data lintasan yang mengandung suffix akan disimpan ke dalam ArrayList itemPref dan bobot *prefix* pada lintasan tersebut disimpan dalam ArrayList bobotPref. Kode golongan item yang menjadi *suffix* juga disimpan dalam ArrayList itemSuf.

```
Prefix cariPrefix(int suffix, ArrayList<Path> p){
   Prefix hasil = new Prefix();
   ArrayList itemPref = new ArrayList<Integer>();
   ArrayList itemSuf = new ArrayList<Integer>();
   ArrayList bobotPref = new ArrayList<Integer>();
   for(int i=0;i<(p.size());i++){
      itemSuf.add(suffix);
       if((p.get(i).kd_jb.get
         (p.get(i).kd_jb.size()-1))==suffix){
          for(int j=0;j<p.get(i).kd_jb.size()-1;j++){
            itemPref.add(p.get(i).kd_jb.get(j));
            bobotPref.add(p.get(i).bobot.get(j));
          }
         hasil=new Prefix(itemPref,itemSuf,bobotPref);
       }
    }return hasil;
}
```

***Source Code* 4.7** Proses *Conditional Pattern Base*

4.2.2.3. Implementasi *Conditional FP-tree*

Proses ini bertujuan untuk mengetahui nilai frekuensi tiap lintasan *prefix* tiap *suffix*. Implementasi dari proses ini dapat ditunjukkan pada *source code* 4.8.

```
FrequentItemset ConditionalFPTree
(ArrayList<Integer> items, ArrayList<Path> pa){
   int nilai;
   double bobot = 0;
ArrayList<Integer> fit = new ArrayList<Integer>();
ArrayList<Double> fitBo = new ArrayList<Double>();
   FrequentItemset hasil = new FrequentItemset();
   for(int i=0;i<pa.size();i++){
      nilai = 0;
        if(pa.get(i).kd_jb.get
          (pa.get(i).kd_jb.size()-1)==
          items.get(items.size()-1)){
          for(int x=0;x<items.size()-1;x++){
            if(pa.get(i).kd_jb.contains(items.get(x))){
               nilai+=1;}
          }
          if(nilai == items.size()-1){
             bobot=bobot+pa.get(i).bobot.get(0);}
        }
     }
     bobot = (bobot / total);
for(int j = 0; j<items.size();j++){
        fit.add(items.get(j));
        fitBo.add(bobot);}
     hasil = new FrequentItemset(fit, fitBo);
     return hasil;}
```

***Source Code* 4.8** Proses *Conditional FP-tree*

Fungsi ConditionalFPTree memiliki parameter inputan berupa kombinasi kode golongan item yang menjadi *frequent 1-itemset* yang dideklarasikan sebagai ArrayList<Integer> dengan nama variabel items dan data lintasanArrayList<Path> pa. Proses dilakukan dengan mengecek pada setiap data lintasansampai ditemukan data lintasan yang terdiri dari kode golongan item yang sama sejumlah kode golongan item pada data kombinasi pada items. Setiap data lintasan yang sesuai maka nilai bobotnya dijumlahkan. Setelah pengecekan dilakukan pada semua data lintasan, maka dihitung nilai *support* kombinasi kode golongan item tersebut dengan cara total bobot untuk kombinasi tersebut yang disimpan dalam variabel bobot dibagi dengan jumlah transaksi yang disimpan dalam variabel total. Kemudian kombinasi kode golongan item disimpan dalam ArrayList<Integer> fit

dan nilai *support* kombinasi kode golongan tersebut disimpan dalam ArrayList<Double> fitBo.

4.2.2.4. Implementasi Koleksi *Frequent Itemset*

Pada proses ini, dilakukan pengecekan nilai *support* setiap lintasan *prefix* tiap *suffix* terhadap nilai minimum *support* yang telah ditentukan. Lintasan *prefix* yang memiliki nilai *support* lebih besar atau sama dengan nilai minimum *support* akan menjadi *frequent itemset*. Implementasi dari proses ini dapat ditunjukkan pada *source code* 4.9.

Fungsi cekSupport berfungsi untuk mengkoleksi hasil dari proses *Conditional FP-tree* yang memiliki nilai *support* lebih besar atau sama dengan nilai *minimum support* yang telah ditentukan. Proses dilakukan dengan menghapus dari data *frequent itemset* yang dideklarasikan ArrayList<FrequentItemset> hasil untuk kandidat *frequent itemset* yang memiliki nilai *support* kurang dari *minimum support*.

```
ArrayList<FrequentItemset>
cekSupport(ArrayList<FrequentItemset> hasil){
  for(int i=0;i<hasil.size();i++){
     if((double)hasil.get(i).bobot.get(0)<minsup){
        hasil.remove(hasil.get(i));
        i=i-1;
     }
  }return hasil;
}
```

***Source Code* 4.9** Proses Koleksi *Frequent Itemset*

4.2.3. Implementasi Tahap *Generate Rule*

Pada tahap ini, dilakukan perhitungan nilai *confidence* pada setiap *frequent itemset*. Nilai *confidence* didapatkan dari nilai *support frequent itemset* dibagi dengan nilai *support* bagian *antecedent*. *Frequent Itemset* yang memiliki nilai *confidence* lebih besar atau sama dengan nilai minimum *confidence* yang telah ditentukan akan menjadi *rule*. Tahap *generate rule* terdiri dari 2 proses, yaitu hitung *confidence* dan proses koleksi *rule*.

4.2.3.1. Implementasi Hitung *Confidence*

Proses ini merupakan proses awal dari tahap *generate rule*. Pada proses ini, tiap *frequent itemset* dihitung nilai confidence-nya. Implementasi dari proses ini dapat ditunjukkan pada *source code* 4.10. Fungsi ConfidenceFI berfungsi untuk menghitung nilai *confidence* untuk setiap data *frequent itemset*. Parameter inputan yang dibutuhkan pada fungsi ini adalah data *frequent itemset* sebagai variabel hasil, data *support frequent 1-itemset* sebagai variabel S, dan nilai *minimum confidence* dalam bentuk prosentase sebagai variabel min.

Setiap data *frequent itemset* diambil *prefix*-nya yang disimpan sementara pada ArrayList<Integer> titip. Jika *prefix* dari index data *frequent itemset* tersebut terdiri lebih dari satu kode golongan item maka untuk mendapatkan nilai *support* kode golongan item yang menjadi *antecedent* (bagian jika) dilakukan pencarian terhadap data *frequent itemset* hasil yang terdiri dari kode golongan item dalam titip. Hasil pencarian tersebut berupa nilai *support* yang disimpan dalam supAntecedent dan untuk selanjutnya digunakan untuk perhitungan nilai *confidence* yaitu sebagai pembagi dari nilai bobot *frequent itemset*.

Sedangkan untuk *prefix* index data *frequent itemset* yang hanya terdiri dari satu kode golongan item, nilai *support* kode golongan item tersebut didapatkan dari pembacaan data *support frequent 1-itemset*. *Frequent itemset* yang memiliki nilai *confidence* lebih dari atau sama dengan minimum *confidence* akan disimpan data *frequent itemset*-nya dalam ArrayList<ArrayList<Integer>> f dan nilai *confidence frequent itemset* tersebut disimpan dalam ArrayList<Double> co. Data yang disimpan sementara pada titip selanjutnya dihapus semuanya karena akan digunakan untuk menyimpan data *prefix* index data *frequent itemset* selanjutnya.

```
Confidence ConfidenceFI(ArrayList<FrequentItemset>
hasil, Support S, double min){
    ArrayList<Integer> titip = new ArrayList<Integer>();
    ArrayList<ArrayList<Integer>> f=
    new ArrayList<ArrayList<Integer>>();
    ArrayList<Double> co = new ArrayList<Double>();
    Confidence hasilCo = new Confidence();
    double confidence= 0;
    double supAntecedent = 0;
    minconf = (min/100);
```

```
    for(int i=0;i<hasil.size();i++){
       for(int x=0;x<hasil.get(i).prefix.size()-1;x++)
{
          titip.add(hasil.get(i).prefix.get(x));
}
if(titip.size()>1){
         for(int a=0;a<titip.size();a++)
{
            for(int y=0;y<hasil.size();y++){
              if(hasil.get(y).prefix.equals(titip))
{
                supAntecedent=hasil.get(y).bobot.get(0);
}
            }
         }
         confidence=
(hasil.get(i).bobot.get(0)/supAntecedent);
       }else{
          for(int y=0;y<S.kodeBukunya.size();y++){
            if((Integer.parseInt
              (S.kodeBukunya.get(y)))==(titip.get(0))){
             confidence=(hasil.get(i).bobot.get(0)
                        /S.bobot.get(0));}
          }
       }
       if(confidence >= minconf){
          f.add(hasil.get(i).prefix);
          co.add(confidence);}
       titip.removeAll(titip);
       hasilCo = new Confidence(f,co);
   }return hasilCo;
}
```

***Source Code* 4.10** Proses Hitung *Confidence*

4.2.3.2. Implementasi Koleksi *Rule*

Pada proses ini, *frequent itemset* yang memiliki nilai *confidence* lebih besar atau sama dengan minimum *confidence* dibangkitkan sebagai *rule*. Implementasi dari proses ini dapat ditunjukkan pada *source code* 4.11.

Fungsi RuleFI terdiri dari parameter inputan data golongan item yaitu JenisBuku JB dan data *frequent itemset* yang lolos nilai *confidence*-nya yaitu ArrayList<Confidence> conf. Proses dalam fungsi ini adalah mengubah format penulisan *rule* dari kode golongan item menjadi nama golongan item. *Rule* dengan notasi kode golongan item disimpan dalam ArrayList<String> r, sedangkan *rule* dengan notasi nama golongan item disimpan dalam ArrayList<String> kt.

```
Rule RuleFI(ArrayList<Confidence> conf, JenisBuku JB){
   Rule R= new Rule();
   String ket,rule;
   String titip1 = "";
   ArrayList<String> r = new ArrayList<String>();
   ArrayList<String> kt = new ArrayList<String>();
   ArrayList<Double> con = new ArrayList<Double>();
   for(int i=0;i<conf.size();i++){
for(int j=0;j<conf.get(i).freqitem.size();j++){
rule = "";ket = "";
for(int l=0;l<conf.get(i).
freqitem.get(j).size();l++){
if(l!=conf.get(i).freqitem.get(j).size()-1){
for(int k=0;k<JB.kodeJenis.size();k++){
if(Integer.parseInt(JB.kodeJenis.get(k))==
conf.get(i).freqitem.
get(j).get(l)){
titip1=JB.namaJenis.get(k);}
}
if(l==0){
rule= conf.get(i).freqitem.
get(j).get(l).toString();
ket = "jika "+titip1;
}else{
rule=rule+","+conf.get(i).
freqitem.get(j).get(l).toString();
ket = ket+" dan "+titip1;}
}else{
for(int k=0;k<JB.kodeJenis.size();k++){
if(Integer.parseInt(JB.kodeJenis.get(k))==
conf.get(i).freqitem.get(j).get(l)){
titip1 = JB.namaJenis.get(k);}
}
rule=rule+"->"+conf.get(i).
freqitem.get(j).get(l).toString();
ket = ket+" maka "+titip1;}
        }
        con.add(conf.get(i).confidence.get(j));
        r.add(rule);
        kt.add(ket);
        R = new Rule(r,kt,con);
     }
  }return R;
}
```

***Source Code* 4.11** Proses Koleksi *Rule*

4.2.4. Implementasi Tahap *Lift Ratio*

Tahap *lift ratio* terdiri dari satu proses yaitu dilakukan perhitungan nilai *lift ratio* pada *frequent itemset* yang menjadi *rule*. Nilai *lift ratio* didapatkan dari hasil pembagian dari nilai *confidence* terhadap nilai *support* golongan item yang

menjadi *consequent* (bagian maka). Implementasi dari proses ini dapat ditunjukkan pada *source code* 4.12.

```
LiftRatio hitungLift(ArrayList<Confidence> conf, Support S,
ArrayList<Rule> rul)
{
   LiftRatio LR = new LiftRatio();
   double lift= 0;
   double supConsequent = 0;
   String fi = "";
   ArrayList<Double> L = new ArrayList<Double>();
   ArrayList<String> FIT = new ArrayList<String>();
   for(int i=0;i<conf.get(0).freqitem.size();i++)
   {
       for(int j=0;j<S.kodeBukunya.size();j++)
       {
           if(conf.get(0).freqitem.get(i).
           get(conf.get(0).freqitem.get(i).
           size()-1)==(Integer.parseInt
           (S.kodeBukunya.get(j)))){
           supConsequent = S.bobot.get(j);
           }
       }
       lift = conf.get(0).confidence.get(i)
       /supConsequent;
       fi = rul.get(0).freqitem.get(i);
       L.add(lift);
       FIT.add(fi);
       LR = new LiftRatio(FIT,L);
   }
   return LR;
}
```

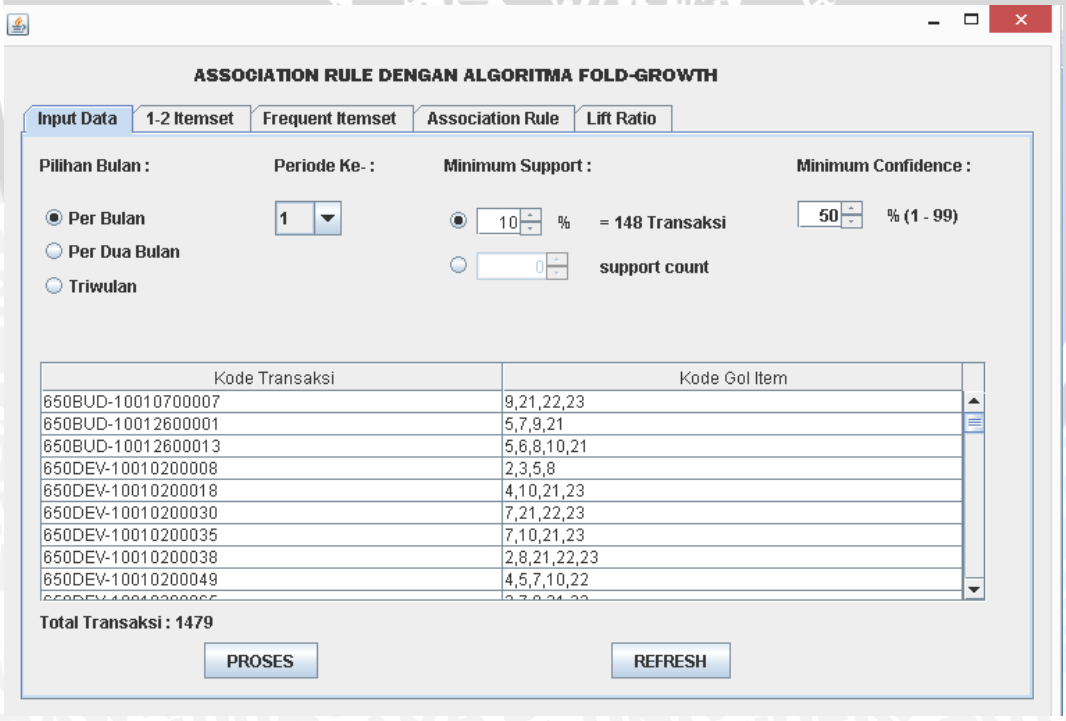
***Source Code* 4.12** Proses *Lift Ratio*

Fungsi hitungLift dimulai dengan mencari nilai *support* kode golongan item yang menjadi *consequent* (bagian maka) dari index data *confidence*. Nilai *support* didapatkan dari data *support frequent 1-itemset* yang dideklarasikan sebagai parameter inputan yaitu Support S. Nilai *lift ratio* didapatkan dari hasil pembagian dari nilai *confidence frequent itemset* tersebut dengan nilai *support consequent* dari *frequent itemset* tersebut. Nilai lift ratio disimpan dalam ArrayList<Double> L, sedangkan *frequent itemset* disimpan dalam ArrayList<String> FIT.

4.3. Implementasi Antarmuka

Implementasi antarmuka sistem seperti yang telah dijelaskan dalam rancangan antarmuka pada subbab 3.4, terdiri dari bagian *input*, bagian *output*, tombol proses dan tombol *refresh*. Bagian input terdiri dari tiga data *input*, yaitu data *input* pilihan bulan, data *input* periode bulan, data *input* minimum *support*, dan data *input* minimum *confidence*.Bagian output terdiri dari empat submenu yaitu submenu Data Transaksi, submenu *Frequent Itemset*, submenu *Association Rule*, dan submenu *Lift Ratio*.

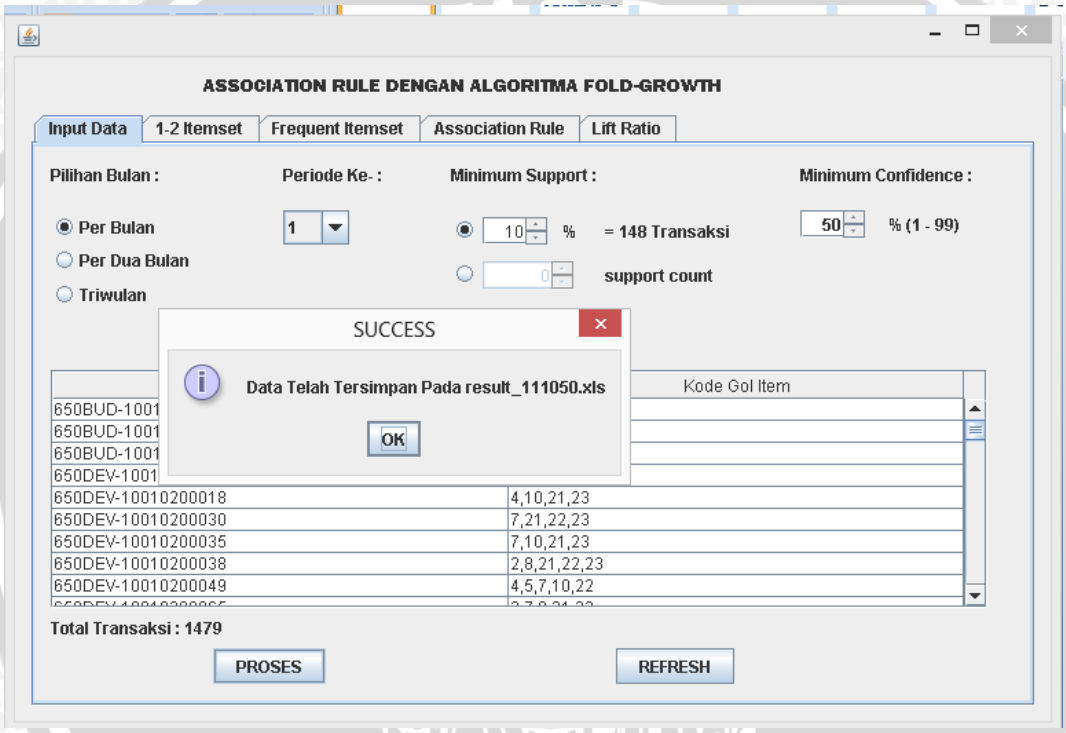
Bagian inputan yang aktif pertama adalah pilihan bulan yang berupa *RadioButton.* Pilihan bulan terdiri dari per bulan, per dua bulan, dan triwulan. Pilihan bulan yang dipilih mengubah isi pilihan pada *ComboBox* periode bulan. Setelah periode bulan dipilih, informasi mengenai total transaksi ditampilkan, *RadioButton* minimum *support* menjadi aktif dan bagian *ouput* submenu Data Transaksi muncul tabel transaksi yang berisi data transaksi untuk periode bulan yang telah dipilih seperti yang diilustrasikan pada Gambar 4.1.



Gambar 4.1 Tampilan Data Transaksi

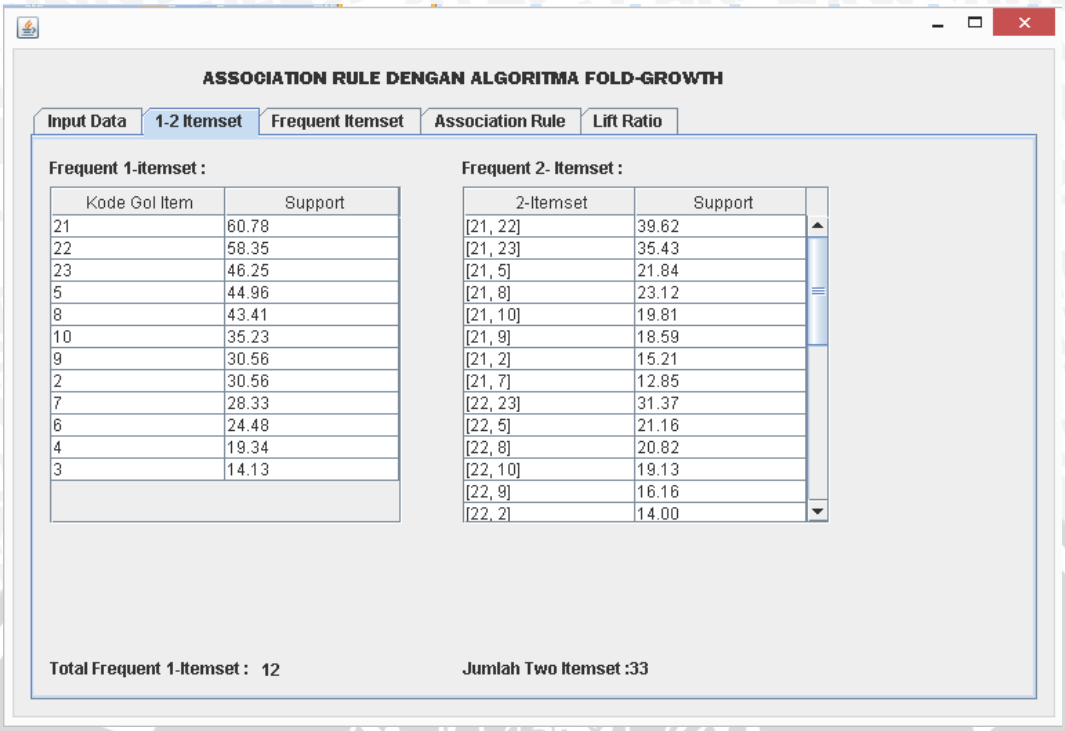
Data inputan untuk minimum *support* diberikan dua macam pilihan yaitu *RadioButton* pertama untuk inputan minimum *support* dalam bentuk prosentase dan *RadioButton* kedua untuk inputan minimum *support* dalam bentuk jumlah transaksi. Bagian inputan minimum *confidence* yang berupa *Spinner* menjadi aktif apabila salah satu *RadioButton* inputan minimum *support* telah dipilih.

Setelah data inputan terisi dengan benar, maka ketika tombol proses ditekan akan keluar *dialog box* bahwa data telah tersimpan seperti yang diilustrasikan pada Gambar 4.2.



Gambar 4.2 Tampilan *Dialog Box* Data Telah Tersimpan Pada File Format xls

Selain itu data *output* yaitu *frequent 1-itemset, frequent itemset, association rule,* dan hasil perhitungan *lift ratio* akan ditampilkan pada masing – masing tabel. Tabel *frequent 1-itemset* beserta jumlahnya dan tabel *frequent itemset* beserta jumlahnya ditampilkan pada submenu *Frequent Itemset* seperti yang diilustrasikan pada Gambar 4.3.

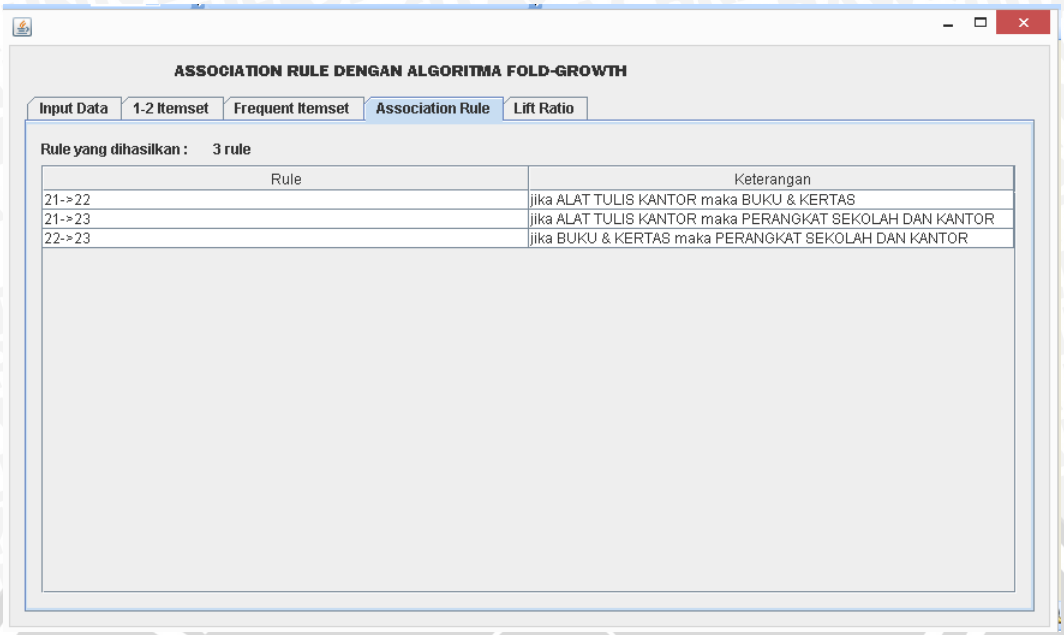


Gambar 4.3 Tampilan Submenu *1-2 Itemset*

Frequent 1-itemset yang memiliki nilai *minimum support* lebih dari atau sama dengan minimum *support* dibentuk kombinasinya untuk menjadi kandidat *frequent itemset*. Pembentukan kombinasi memiliki aturan yaitu yang menjadi bagian *suffix* adalah golongan item yang memiliki nilai *support* yang terkecil pada kombinasi tersebut.

Frequent Itemset yang memiliki nilai *support* lebih dari atau sama dengan minimum *support* akan menjadi kandidat *rule*. Kandidat *rule* yang menjadi *rule* adalah yang memiliki nilai *confidence* lebih dari atau sama dengan minimum *confidence*.

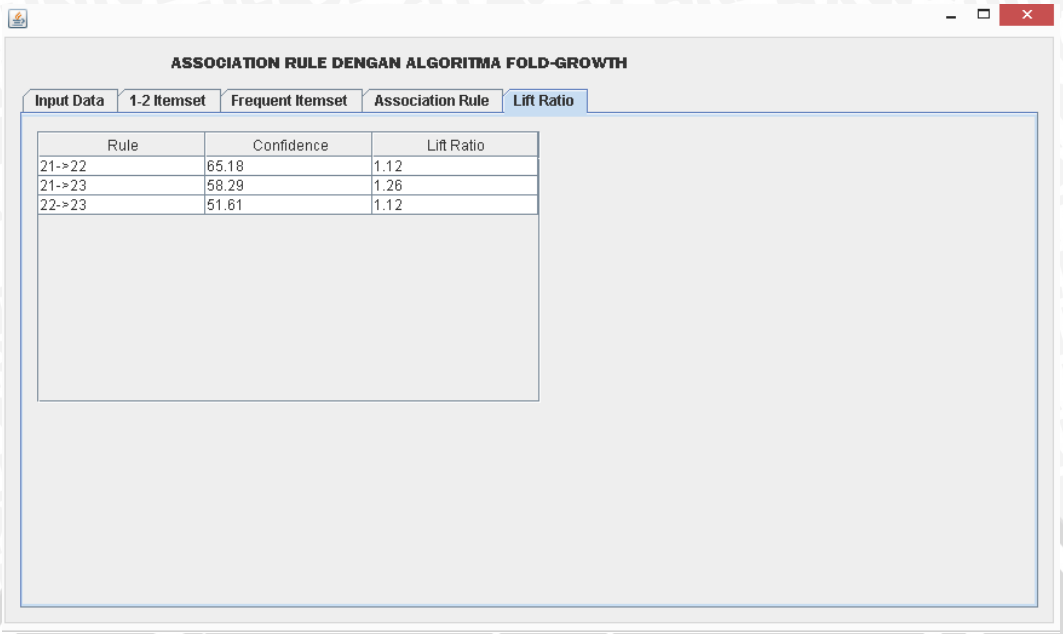
Rule yang dihasilkan ditampilkan pada tabel *Rule* yang terdapat pada submenu *Association Rule*. Tabel *Rule* terdiri dari dua kolom yaitu kolom *rule* yang berisi *rule* dalam bentuk kode dan kolom keterangan yang berisi keterangan dari kode *rule*. Pada submenu *Association Rule*, selain ditampilkan tabel *rule* juga ditampilkan informasi jumlah rule yang dihasilkan. Antarmuka submenu *Association Rule* ini diilustrasikan pada Gambar 4.4.



Gambar 4.4 Tampilan Submenu *Association Rule*

Rule yang terbentuk kemudian dihitung nilai *lift ratio*. Nilai *lift ratio* ini yang menentukan kekuatan *rule* berdasarkan manfaatnya karena nilai *lift ratio* ini merupakan nilai keseimbangan antara frekuensi kemunculan golongan item dalam *rule* tersebut muncul secara independen dan muncul secara bersama – sama.

Informasi nilai *confidence* dari *rule* yang dihasilkan dapat dilihat dalam tabel *lift ratio* pada submenu *Lift Ratio*. Hal ini dimaksudkan untuk mempermudah user melakukan analisis nilai *confidence* terhadap nilai *lift ratio* suatu rule. Tabel *lift ratio* terdiri dari kolom *rule* yang berisi kode *rule*, kolom *confidence* yang berisi nilai *confidence* untuk tiap *rule*, dan kolom *lift ratio* yang berisi nilai *lift ratio* untuk tiap *rule*. Antarmuka submenu *Lift Ratio* diilustrasikan pada Gambar 4.5.



4.5 Tampilan Submenu *Lift Ratio*

BAB V
PENGUJIAN DAN ANALISIS

5.1. Pengujian Sistem

Pengujian sistem dilakukan untuk masing – masing pilihan bulan, yaitu per bulan, per dua bulan, dan triwulan. Masing – masing pilihan bulan memiliki jumlah periode yang berbeda, yaitu data per bulan terdiri dari 6 periode, data per dua bulan terdiri dari 3 periode, dan data per triwulan terdiri dari 2 periode. Data transaksi yang digunakan dalam pengujian ini dibatasi hanya untuk transaksi yang memiliki pembelian golongan item minimal 4 buah golongan item.

Pengujian pertama dilakukan untuk mengetahui pengaruh nilai minimum *support* dan nilai minimum *confidence* yang diberikan terhadap jumlah *rule* yang dihasilkan. Pada pengujian pertama ini, minimum *support* yang digunakan adalah sejumlah 5% dan 10%. Minimum *confidence* yang digunakan adalah 50%. Hasil uji pengaruh nilai minimum *support* dan nilai minimum *confidence* terhadap jumlah *rule* yang dihasilkan dapat dilihat pada Tabel 5.1

Tabel 5.1 Hasil Uji Pengaruh Nilai Minimum *Support* dan Nilai Minimum *Confidence* terhadap Jumlah *Rule* yang Dihasilkan

	Periode	**Minimum *Support(%)***	**Minimum *Confidence(%)***	**Jumlah *Rule***
Per Bulan	**1**	**5**	**30**	
		10	**50**	**3**
	2	**5**	**30**	
		10	**50**	**3**
	3	**5**	**30**	
		10	**50**	**2**
	4	**5**	**30**	
		10	**50**	**2**
	5	**5**	**30**	

		10	50	3
	6	5	30	
		10	50	3
Per Dua Bulan	1	5	30	
		10	50	3
	2	5	30	
		10	50	2
	3	5	30	
		10	50	3
Triwulan	1	5	30	
		10	50	3
	2	5	30	
		10	50	2

Pengujian kedua dilakukan untuk mengetahui tingkat kekuatan (*lift ratio*) dari *rule* yang dihasilkan. Pada pengujian ini *rule* yang diuji adalah *rule* untuk minimum *confidence* 50%. Hasil pengujian kedua dapat dilihat pada Tabel 5.2 dan keterangan dari *rule* hasil pengujian dapat dilihat pada Tabel 5.3.

Tabel 5.2 Hasil Uji Kekuatan (*Lift Ratio*) *Rule* yang Dihasilkan

	Periode	Rule	Confidence (%)	Lift Ratio
Per Bulan	1	21 → 22	65.18	1.12
		21 → 23	58.29	1.26
		22 → 23	51.61	1.12
	2	21 → 22	68.99	1.07
		21 → 23	62.71	1.02
		22 → 23	55.64	1.07
	3	21 → 22	61.17	1.06

		21 → 23	54.98	1.16
	4	21 → 22	65.65	1.16
		21 → 23	54.58	1.21
	5	21 → 22	68.86	1.1
		21 → 23	57.88	1.26
		22 → 23	50.13	1.09
	6	21 → 22	70.23	1.12
		21 → 23	57.56	1.23
		22 → 23	50.8	1.08
Per Dua Bulan	1	21 → 22	67.06	1.1
		21 → 23	60.47	1.23
		22 → 23	53.6	1.09
	2	21 → 22	63.29	1.1
		21 → 23	54.79	1.19
	3	21 → 22	69.57	1.11
		21 → 23	57.72	1.24
		22 → 23	50.47	1.09
Triwulan	1	21 → 22	66.23	1.09
		21 → 23	59.7	1.22
		22 → 23	51.86	1.06
	2	21 → 22	69.01	1.12
		21 → 23	57.27	1.24

Tabel 5.3 Keterangan *Rule* Hasil Uji Kekuatan *Lift Ratio*

Rule	Keterangan
21 → 22	Jika alat tulis kantor maka buku dan kertas
21 → 23	Jika alat tulis kantor maka perangkat sekolah dan kantor
22 → 23	Jika buku dan kertasmaka perangkat

	sekolah dan kantor

Berdasarkan Tabel 5.2 didapatkan informasi bahwa rata – rata nilai *lift ratio rule* yang dihasilkan secara keseluruhan adalah 1.14. Rata – rata nilai *lift ratio rule* yang dihasilkan pada rentang per bulan adalah 1.13. *Rule* yang dihasilkan pada rentang rata – rata nilai *lift ratio* sebesar 1.15.

5.2. Analisis Hasil

Berdasarkan hasil pengujian pengaruh nilai minimum *support* dan nilai minimum *confidence* terhadap jumlah *rule* yang dihasilkan, didapatkan jumlah *rule* terbanyak yang dihasilkan adalah 3*rule* yaitu pada periode bulan Januari, periode bulan Februari, periode bulan Mei dan periode bulan Juni rentang bulan per bulan dengan minimum *support* 10% dan minimum *confidence* 50%.

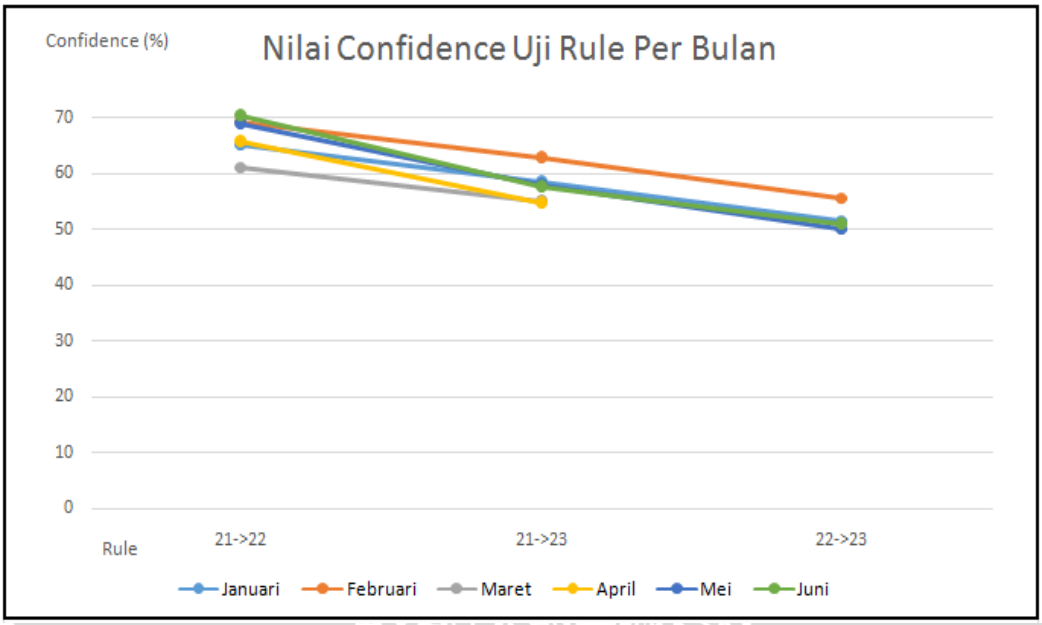
Pengaruh nilai minimum *support* dan nilai minimum *confidence* terhadap jumlah *rule* yang dihasilkan berdasarkan hasil pengujian adalah berbanding terbalik. Hal ini berarti semakin tinggi nilai minimum *support* dan nilai minimum *confidence* yang digunakan maka semakin sedikit jumlah *rule* yang dihasilkan.

Berbanding terbaliknya nilai minimum *support* dan nilai minimum *confidence* terhadap jumlah *rule* tersebut dikarenakan semakin tinggi nilai minimum *support* yang digunakan berarti semakin tinggi nilai batas *support* yang harus dicapai kandidat *frequent itemset* untuk menjadi *frequent itemset* sehingga jumlah *frequent itemset* semakin sedikit dan bila semakin tinggi nilai minimum *confidence* yang digunakan maka semakin tinggi nilai batas *confidence* yang harus dicapai *frequent itemset*, yang merupakan kandidat *rule*, untuk menjadi *rule* sehingga semakin sedikit jumlah *rule* yang dihasilkan.

Berdasarkan hasil pengujian kekuatan (*lift ratio*) *rule* yang dihasilkan dengan nilai minimum *confidence* 50% didapatkan rata – rata nilai *lift ratio* dari keseluruhan *rule* yang dihasilkan adalah 1.14. Nilai *lift ratio* tertinggi adalah *rule* jika membeli item**alat tulis kantor** maka membeli item**perangkat sekolah dan kantor** (21→ 23) sebesar 1.26 yang didapatkan pada rentang bulan per bulan periode bulan Januari dan Mei, sedangkan nilai *lift ratio* terendah adalah 1.02 yaitu pada *rule* jika membeli item**alat tulis kantor**maka membeli item**perangkat**

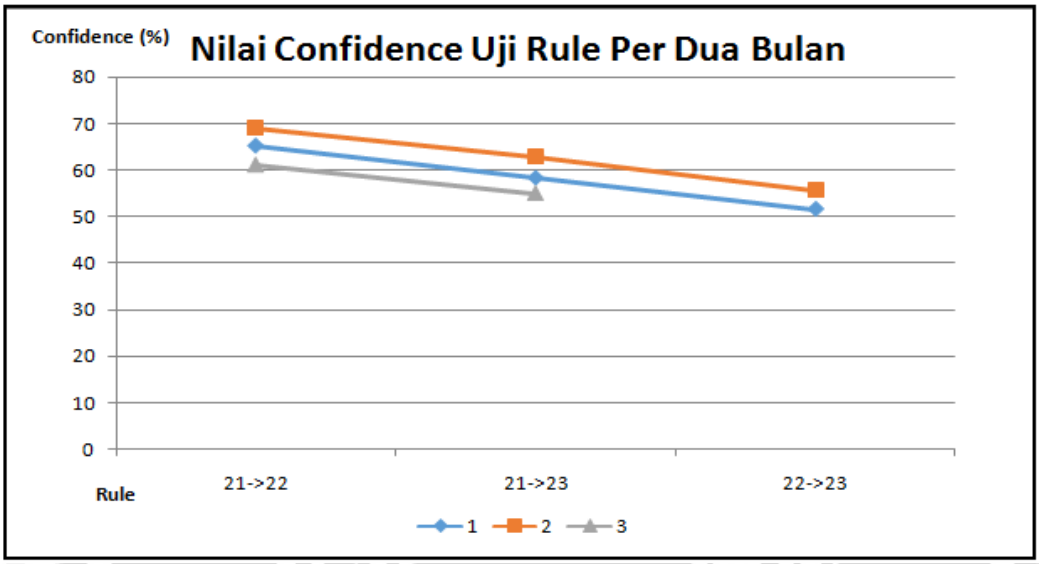
sekolah dan kantor(21→ 23) yang didapatkan pada rentang bulan per bulan periode bulan Februari.

Rentang bulan yang digunakan pada pengujian selain memberikan pengaruh terhadap variasi *rule* yang dihasilkan juga memberikan pengaruh terhadap nilai *confidence* dari *rule* yang dihasilkan. Hal ini digambarkan dengan grafik nilai *confidence rule* yang dihasilkan per rentang bulan yang digunakan yang dapat dilihat pada Gambar 5.1 sampai Gambar 5.3.



Gambar 5.1 Grafik Nilai *Confidence Rule* Hasil Uji Per Bulan

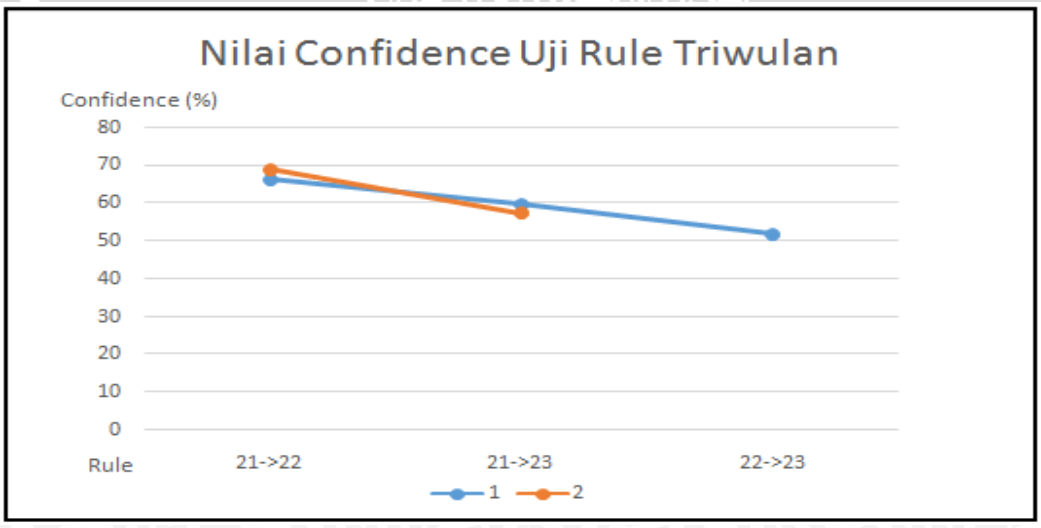
Pada grafik Gambar 5.1, *rule* 21→ 22 terlihat memiliki nilai *confidence* yang selalu lebih tinggi dari *rule* lain tiap periode. Hal ini menunjukkan bahwa konsistensi penjualan item **alat tulis kantor** dan **buku dan kertas** stabil dari periode ke periode. Selain itu, dapat disimpulkan bahwa hubungan item**alat tulis kantor**dan **buku dan kertas**memiliki nilai *confidence* yang konstan yaitu diatas 50% yang berarti dari jumlah frekuensi item**alat tulis kantor**yang terjual, 60% sampai 70% terjual bersama – sama dengan item**buku dan kertas.** R*ule* yang menggambarkan hubungan kedua golongan buku tersebut selalu dihasilkan tiap periode.



Gambar 5.2 Grafik Nilai *Confidence Rule* Hasil Uji Per Dua Bulan

Grafik Gambar 5.2 memperlihatkan bahwa *rule* 21→ 22 tetap selalu memiliki nilai *confidence* tertinggi dari *rule* lainnya tiap periode. Hal ini menjelaskan bahwa pada periode dua bulan, tingkat penjualan item **alat tulis kantor** dan **buku dan kertas** konsisten.

Rule yang lain seperti 21 → 23, dan 22→ 23masih dihasilkan bila dianalisis per dua bulan, hal ini menunjukkan bahwa hubungan antar item tersebut tetap memiliki nilai kepercayaan cukup tinggi yaitu minimal 50% ketika data transaksi dianalisis secara dua bulan.



Gambar 5.3 Grafik Nilai *Confidence Rule* Hasil Uji Per Triwulan

Gambar 5.3 memperlihatkan bahwa *rule* 21→ 22 terlihat memiliki nilai *confidence* yang menonjol dari *rule* lainnya meskipun data transaksi dianalisis per 3 bulan. *Rule* lain yang tetap dihasilkan adalah *rule* 21→ 23, dan 22→ 23. *Rule* 22→ 23 tidak dihasilkan pada periode bulan April sampai bulan Juni. Hal ini menunjukkan bahwa penjualan item **buku dan kertas**dan **perangkat sekolah dan kantor**mengalamin penurunan penjualan pada bulan April sampai bulan Juni bila dianalisis triwulan.

Berdasarkan nilai *confidence* dan nilai *lift ratiorule* yang dihasilkan dalam pengujian ini, *rule* yang memiliki nilai *confidence* tinggi belum tentu memiliki nilai *lift ratio* yang tinggi (lebih dari atau sama dengan 1). Hal ini menjelaskan bahwa meskipun nilai kepercayaan dari *rule* tersebut tinggi namun belum tentu *rule* tersebut bermanfaat karena nilai *lift ratio* menggambarkan keseimbangan antara frekuensi kemunculan tiap golongan buku dalam *rule* tersebut secara independen dengan frekuensi kemunculannya secara bersamaan, sedangkan nilai *confidence* menggambarkan tingkat kepercayaan golongan buku dalam *rule* tersebut muncul secara bersamaan. Akan tetapi nilai minimum *support* yang digunakan juga mempengaruhi untuk mendapatkan *rule* yang benar – benar bermanfaat. *Rule* yang dapat dikatakan bermanfaat dari hasil pengujian ini dapat dilihat pada Tabel 4.5 dan detail informasi *rule* dapat dilihat pada Tabel 4.6.

Tabel 5.4 *Rule* yang Bermanfaat

Rule	Keterangan
21 → 22	**Jika membeli item alat tulis kantor maka membeli item buku dan kertas**
21 → 23	**Jika membeli buku golongan alat tulis kantor maka membeli item perangkat sekolah dan kantor**
22 → 23	**Jika membeli item buku dan kertas maka membeli item perangkat sekolah dan kantor**

Tabel 5.5 Detail Informasi *Rule* yang Bermanfaat

Rule	*Support*	*Confidence*	*Lift Ratio*
21 → 22	**44.04**	**65.65**	**1.16**
21 → 23	**22,94**	**58.29**	**1.26**
22 → 23	**41.62**	**51.61**	**1.12**

BAB VI
KESIMPULAN DAN SARAN

6.1. Kesimpulan.

Berdasarkan hasil analisis uji coba dapat diambil kesimpulan:

Dalam menerapkan algoritma *Fold-Growth* untuk mengetahui pola penjualan buku dari data transaksi penjualan buku berdasarkan golongan buku dan alat tulis, parameter inputan yang berpengaruh untuk mendapatkan *rule* dengan nilai *confidence* dan nilai *lift ratio* tinggi adalah penggunaan batasan jumlah golongan buku dan alat tulis per transaksi, minimum *support*, dan penggunaan minimum *confidence*.

Rule yang dihasilkan dari penerapan algoritma *Fold-Growth* dalam permasalahan data transaksi penjualan buku ini memiliki rata – rata tingkat kekuatan (*lift ratio*) sebesar 1.14 dengan tingkat kekuatan (*lift ratio*) *association rule* terbesar yang dihasilkan adalah 1.26 dan terkecil adalah 1.02 untuk *rule* dengan tingkat kepercayaan minimal 50% item tersebut dibeli secara bersamaan. Berdasarkan tingkat kekuatan (*lift ratio*) yang dihasilkan, dapat dikatakan bahwa *rule* yang bermanfaat tidak hanya dilihat berdasarkan nilai *confidence*, akan tetapi berdasarkan tiga parameter penting yaitu nilai *confidence*, nilai *lift ratio*, dan nilai minimum *support* yang digunakan.

6.2 Saran

Saran yang dapat dijadikan bahan pertimbangan untuk penelitian selanjutnya adalah dalam melakukan analisis pola menggunakan metode *association rule* terhadap data dengan model data seperti data transaksi penjualan buku dan alat tulis, proses *preprocessing* data dan penggunaan nilai parameter seperti rentang bulan, nilai minimum *support*, dan nilai minimum *confidence* sangat berpengaruh penting terhadap jumlah *rule* dan kekuatan *rule* yang dihasilkan. Penggunaan algoritma *association rule* lain, juga dapat diterapkan untuk membandingkan pola dan kompleksitas waktu.

DAFTAR PUSTAKA

Das, Amitabha, Wee-Keong Ng, dan Yew-Kwong Woon. 2001. *Rapid Association Rule Mining*. Nanyang Technological University. Singapura.

Febriyana, Fatma Rika. 2009. *Perbandingan Kecepatan dalam Pencarian Frequent Itemset antara Algoritma FP – Growth dan Cut Both Ways*. Universitas Brawijaya. Malang

Han, Jiawei dan Micheline Kamber. 2001. *Data Mining : Concepts and Techniques*. Morgan Kaufmann, California

Han, Jiawei dan Micheline Kamber. 2006. *Data Mining : Concepts and Techniques 2nd Edition*. Morgan Kaufmann, California

Handojo, Andreas, Gregorius Satia Budhi, Hendra Rusly. 2004. *Aplikasi Data Mining untuk Meneliti Asosiasi Pembelian Item Barang di Supermarket dengan Metode Market Basket Analysis*. Seminar Nasional Teknologi Informasi Universitas Kristen Petra. Surabaya.

Huda, Nuqson Masykur. 2010. *Aplikasi Data Mining untuk Menampilkan Informasi Tingkat Kelulusan Mahasiswa*. Semarang : Universitas Diponegoro.

Kantardzic, Mehmed. 2003. *Data Mining : Concepts, Models, Methods, and Algorithms*. John Willey & Sons, Inc. New Jersey.

Kotler, Philip dan Kevin Lane Keller. 2007. *Manajemen Pemasaran*. PT. Indeks. Jakarta.

Kusrini, M.Kom. 2007. *Konsep dan Aplikasi Sistem Pendukung Keputusan*. CV Andi Offset. Yogyakarta.

Moertini, Veronika S. 2002. *Data Mining sebagai Solusi Bisnis*. Integral Majalah Ilmiah Matematika dan Ilmu Pengetahuan Alam, Volume 7 No.1. Jurusan Ilmu Komputer Universitas Katolik Parahyangan. Bandung.

Pramudiono, Iko. 2003. *Pengantar Data Mining : Menambang Permata Pengetahuan di Gudang Data*. http:// www.ilmukomputer.com, tanggal akses 15 Maret 2011.

Rochmah, Affriantari. 2010. *Perancangan Fitur Rekomendasi Film Website Solo Movie dengan Menggunakan Metode Algoritma Apriori*. Universitas Sebelas Maret. Surakarta.

Ruldeviyani, Yova dan Muhammad Fahrian. 2008. *Implementasi Algoritma – Algoritma Association Rules sebagai Bagian dari Pengembangan Data Mining Algorithms Collection*. Fakultas Ilmu Komputer Universitas Indonesia. Jakarta.

Samuel, David. 2008. *Penerapan Struktur FP – Tree dan Algoritma FP – Growth dalam Optimasi Penentuan Frequent Itemset*. Institut Teknologi Bandung

Soelaiman, Rully dan Ni Made Arini WP. 2006. *Analisis Kinerja Algoritma Fold – Growth dan FP – Growth pada Penggalian Pola Asosiasi*. Seminar Nasional Aplikasi Teknologi Informasi (SNATI) Institut Teknologi Sepuluh Nopember (ITS). Surabaya.

Santosa, Budi. 2007. *Data Mining Teknik Pemanfaatan Data untuk Keperluan Bisnis*. Graha Ilmu. Yogyakarta.

Sholichah, Alfiyatus. 2009. *Data Mining untuk Pembiayaan Murabahah Menggunakan Association Rule*. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri (UIN) Maulana Malik Ibrahim. Malang.

Verhein, Florian. 2008. *Frequent Pattern Growth (FP – Growth) Algorithm*. School of Information Technologies The University of Sydney. Australia.

Yiapanis, Paraskevas. 2006. *Parallel Mining of Minimal Sample Unique Itemsets*. School of Computer Science, University of Manchester.

Yulita, Marsela dan Veronica S. Moertini. 2002. *Analisis Keranjang Pasar dengan Algoritma Hash-Based pada Transaksi Penjualan di Apotek.* Integral Majalah Ilmiah Matematika dan Ilmu Pengetahuan Alam, Volume 9 No. 3. Jurusan Ilmu Komputer Universitas Katolik Parahyangan. Bandung.

LAMPIRAN A

A.1 Tabel Data Penjualan Toko Buku Togamas

FAKTUR	TANGGAL	JUDUL	GOLONGAN	NAMA GOLONGAN
650DEV-10010200001	1/2/2010	BALLPEN STANDARD DR GEL MIX/MILKY (BR,HT)	21.11	BALLPEN
650DEV-10010200001	1/2/2010	PERSIAPAN MENGHADAPI UNAS & UJIAN SEKOLAH SMA/MA IPA '09/'10 5TH KTSP	18.03	BUKU SMA
650DEV-10010200002	1/2/2010	PAKET DETIK-DETIK UNAS SMP/MTS 2009/2010 SMP + KUNCI,ALAT TULIS	18.02	BUKU SMP
650DEV-10010200002	1/2/2010	IPA TERPADU SMP IX JL3 KTSP 06	18.02	BUKU SMP
650DEV-10010200002	1/2/2010	ERASER FABER CASTELL DUST FREE BLACK 187530	21.31	PERANGKAT ATK
650DEV-10010200003	1/2/2010	BALLPEN BPT-P TUTUP PILOT	21.11	BALLPEN
650DEV-10010200004	1/2/2010	BALLPEN GEL INK PERAK/SILVER	21.11	BALLPEN
650DEV-10010200005	1/2/2010	MICROSOFT EXCEL	18.03	BUKU SMA
650DEV-10010200005	1/2/2010	SBS MICROSOFT OFFICE WORD 2007	11.04	TEKNIK INFORMATIKA
650DEV-10010200006	1/2/2010	SAINS SD IV JL4 KTSP 2006	18.01	BUKU SD
650DEV-10010200006	1/2/2010	SAINS SD III JL3 KTSP 06	18.01	BUKU SD
650DEV-10010200007	1/2/2010	PIN 6000 BAJUKU	23.32	FANCY
650DEV-10010200007	1/2/2010	PIN 4000 BAJUKU	23.32	FANCY
650DEV-10010200007	1/2/2010	PIN 4500 BAJUKU	23.32	FANCY
650DEV-10010200008	1/2/2010	KAMUS NOMENKLATUR ZOOLOGI & BOTANI	11.16	BIOLOGI
650DEV-10010200008	1/2/2010	101 EKSPERIMEN SAINS YANG MENGHIBUR	17.04	ENSIKLOPEDI
650DEV-10010200008	1/2/2010	WASPADA FLU BABI	12.05	KESEHATAN
650DEV-10010200008	1/2/2010	LIVING ENGLISH CONVERSATION [HVS]	14.01	LANGUAGE TEACHING METHODOLOGY
650DEV-10010200009	1/2/2010	STATISTIK NON-PARAMETRIK TEORI &..SPSS	11.13	MATEMATIKA
650DEV-10010200010	1/2/2010	AGAMA JALAN KEDAMAIAN/FARIDI	16.01	AGAMA ISLAM

A.2 Tabel Basis Data Golongan Item

kdGol	namaGol
1	BUKU IMPORT
2	TEKNIK
3	KEDOKTERAN
4	SOSIAL
5	BAHASA DAN SASTRA
6	BISNIS
7	AGAMA
8	BUKU ANAK
9	BUKU SEKOLAH
10	MEDIA
11	UMUM
12	KOMPUTER
13	STATISTIK
21	ALAT TULIS KANTOR
22	BUKU & KERTAS
23	PERANGKAT SEKOLAH DAN KANTOR
24	AKSESORIS KOMPUTER
25	FOOD & BEVERAGES

A.3 Tabel Hasil Pengujian Periode Bulan Januari

Kode Gol Item	Support
21	60,78
22	58,35
23	46,25
5	44,96
8	43,41
10	35,23
9	30,56
2	30,56
7	28,33
6	24,48
4	19,34
3	14,13

2-Itemset	Support	2-Itemset	Support
[21, 22]	39,62	[23, 5]	14,74
[21, 23]	35,43	[23, 8]	14,87
[21, 5]	21,84	[23, 10]	14,06

[21, 8]	23,12	[23, 9]	13,39
[21, 10]	19,81	[23, 2]	10,14
[21, 9]	18,59	[5, 8]	20,22
[21, 2]	15,21	[5, 10]	14,94
[21, 7]	12,85	[5, 9]	13,05
[22, 23]	31,37	[5, 2]	13,46
[22, 5]	21,16	[5, 7]	13,39
[22, 8]	20,82	[5, 6]	11,90
[22, 10]	19,13	[8, 10]	13,39
[22, 9]	16,16	[8, 9]	12,31
[22, 2]	14,00	[8, 2]	13,39
[22, 7]	11,97	[8, 7]	13,32
[22, 6]	10,28	[8, 6]	10,48
		[10, 2]	10,95

Rule	Keterangan
21->22	jika ALAT TULIS KANTOR maka BUKU & KERTAS
21->23	jika ALAT TULIS KANTOR maka PERANGKAT SEKOLAH DAN KANTOR
22->23	jika BUKU & KERTAS maka PERANGKAT SEKOLAH DAN KANTOR

Rule	Confidence	Lift Ratio
21->22	65,18	1,12
21->23	58,29	1,26
22->23	51,61	1,12

A.4 Tabel Hasil Pengujian Periode Bulan Februari

Kode Gol Item	Support
21	68,36
22	64,30
23	52,14
5	47,00
8	38,66
9	30,55
10	30,16
2	26,11
6	24,71
7	21,43
4	17,30
3	10,44

2-Itemset	Support	2-Itemset	Support
[21, 22]	47,16	[22, 10]	16,91
[21, 23]	42,87	[22, 2]	14,58
[21, 5]	27,90	[22, 6]	13,25
[21, 8]	22,76	[22, 7]	10,52
[21, 9]	21,20	[23, 5]	19,49
[21, 10]	18,00	[23, 8]	14,73
[21, 2]	13,25	[23, 9]	16,45
[21, 6]	11,30	[23, 10]	11,15
[21, 7]	12,00	[5, 8]	18,78
[22, 23]	38,04	[5, 9]	14,65
[22, 5]	24,71	[5, 10]	13,80
[22, 8]	20,19	[5, 2]	11,61
[22, 9]	17,23	[5, 6]	12,55
		[8, 10]	12,00

Rule	Keterangan
21->22	jika ALAT TULIS KANTOR maka BUKU & KERTAS
21->23	jika ALAT TULIS KANTOR maka PERANGKAT SEKOLAH DAN KANTOR
22->23	jika BUKU & KERTAS maka PERANGKAT SEKOLAH DAN KANTOR

Rule	Confidence	Lift Ratio
21->22	68,99	1,07
21->23	62,71	1,20
22->23	55,64	1,07

A.5 Tabel Hasil Pengujian Periode Bulan Maret

Kode Gol Item	Support
21	67,99
22	57,94
23	47,43
8	43,69
5	41,82
6	30,84
2	26,64
10	26,17
4	22,90
7	20,33
9	16,82
3	13,79

2-Itemset	Support	2-Itemset	Support
[21, 22]	41,59	[22, 6]	17,29
[21, 23]	37,38	[22, 2]	11,68
[21, 8]	25,93	[22, 10]	12,85
[21, 5]	25,23	[23, 8]	14,49
[21, 6]	15,65	[23, 5]	17,06
[21, 2]	14,95	[23, 2]	10,28
[21, 10]	15,89	[8, 5]	18,46
[21, 4]	12,15	[8, 6]	13,32
[21, 7]	13,08	[8, 2]	10,05
[21, 9]	10,05	[8, 10]	11,21
[22, 23]	28,04	[8, 4]	10,75
[22, 8]	19,39	[8, 7]	10,05
[22, 5]	19,86	[5, 6]	10,05
		[5, 10]	10,51

Rule	Keterangan
21->22	jika ALAT TULIS KANTOR maka BUKU & KERTAS
21->23	jika ALAT TULIS KANTOR maka PERANGKAT SEKOLAH DAN KANTOR

Rule	Confidence	Lift Ratio
21->22	61,17	1,06
21->23	54,98	1,16

A.6 Tabel Hasil Pengujian Periode Bulan April

Kode Gol Item	Support
21	60,37
22	56,68
8	51,15
23	44,93
5	43,32
10	31,11
6	29,49
2	27,88
7	23,96
4	18,20
9	17,97
3	12,21

2-Itemset	Support	2-Itemset	Support
[21, 22]	39,63	[22, 2]	11,98
[21, 8]	25,81	[22, 7]	10,14
[21, 23]	32,95	[8, 23]	17,74
[21, 5]	20,97	[8, 5]	22,58
[21, 10]	15,90	[8, 10]	14,52
[21, 6]	12,44	[8, 6]	13,36
[21, 2]	12,90	[8, 2]	13,13
[21, 7]	11,06	[8, 7]	12,90
[22, 8]	25,81	[23, 5]	16,36
[22, 23]	28,34	[23, 10]	12,67
[22, 5]	19,82	[5, 10]	11,98
[22, 10]	15,90	[5, 6]	13,13
[22, 6]	14,06	[5, 7]	11,06

Rule	Keterangan
21->22	jika ALAT TULIS KANTOR maka BUKU & KERTAS
21->23	jika ALAT TULIS KANTOR maka PERANGKAT SEKOLAH DAN KANTOR

Rule	Confidence	Lift Ratio
21->22	65,65	1,16
21->23	54,58	1,21

A.7 Tabel Hasil Pengujian Periode Bulan Mei

Kode Gol Item	Support
21	63,91
22	62,68
5	48,06
8	46,66
23	45,83
10	38,23
6	31,54
2	24,19
7	23,53
9	19,65
4	15,11
3	13,63

2-Itemset	Support	2-Itemset	Support
[21, 22]	44,01	[22, 2]	12,47
[21, 5]	25,76	[22, 7]	11,73
[21, 8]	26,67	[22, 9]	10,57
[21, 23]	36,99	[5, 8]	22,96
[21, 10]	22,79	[5, 23]	16,43
[21, 6]	14,20	[5, 10]	18,08
[21, 2]	11,89	[5, 6]	15,69
[21, 7]	11,23	[5, 7]	12,63
[21, 9]	10,65	[8, 23]	16,76
[22, 5]	25,52	[8, 10]	16,35
[22, 8]	23,70	[8, 6]	13,46
[22, 23]	32,04	[8, 7]	12,47
[22, 10]	22,38	[23, 10]	14,45
[22, 6]	15,11	[10, 6]	11,07

Rule	Keterangan
21->22	jika ALAT TULIS KANTOR maka BUKU & KERTAS
21->23	jika ALAT TULIS KANTOR maka PERANGKAT SEKOLAH DAN KANTOR
22->23	jika BUKU & KERTAS maka PERANGKAT SEKOLAH DAN KANTOR

Rule	Confidence	Lift Ratio
21->22	68,86	1,10
21->23	57,88	1,26
22->23	50,13	1,09

A.8 Tabel Hasil Pengujian Periode Bulan Juni

Kode Gol Item	Support
21	68,15
22	62,70
5	49,04
23	46,94
8	46,35
10	34,28
9	29,17
6	25,23
7	25,06
2	23,22
4	15,51
3	10,14

2-Itemset	Support	2-Itemset	Support
[21, 22]	47,86	[22, 7]	11,48
[21, 5]	30,09	[22, 2]	11,99
[21, 23]	39,23	[5, 23]	18,11
[21, 8]	29,84	[5, 8]	22,63
[21, 10]	21,37	[5, 10]	16,51
[21, 9]	19,28	[5, 9]	12,91
[21, 6]	11,99	[5, 6]	13,33
[21, 7]	11,57	[5, 7]	12,99
[21, 2]	10,90	[23, 8]	18,27
[22, 5]	26,32	[23, 10]	12,41
[22, 23]	34,62	[23, 9]	12,24
[22, 8]	24,14	[8, 10]	15,84
[22, 10]	18,11	[8, 9]	13,24
[22, 9]	17,60	[8, 6]	10,06
[22, 6]	11,15	[8, 7]	12,15

Rule	Keterangan
21->22	jika ALAT TULIS KANTOR maka BUKU & KERTAS
df21->23	jika ALAT TULIS KANTOR maka PERANGKAT SEKOLAH DAN KANTOR
22->23	jika BUKU & KERTAS maka PERANGKAT SEKOLAH DAN KANTOR

Rule	Confidence	Lift Ratio
21->22	70,23	1,12
21->23	57,56	1,23
22->23	50,80	1,08