BAB I

PENDAHULUAN

1. Latar Belakang

Saat ini semakin banyak *smartphone* dengan berbagai tipe, merk dan spesifikasi yang beredar di masyarakat. *Smartphone* yang saat ini banyak beredar dimasyarakat tersebut antara lain *smartphone* yang memiliki sistem operasi Android, iOS, Windows Phone, RIM, symbian dan lain sebagainya [ALA-13].

Semakin banyaknya *smartphone* yang beredar dimasyarakat tersebut membuat banyak *vendor* yang tertarik untuk membuat *smartphone* dengan spesifikasi yang sesuai dengan kebutuhan pengguna. Untuk mengembangkan sebuah *smartphone,* suatu *vendor* dapat melihat pola spesifikasi dari *smartphone* milik *vendor* lain yang sudah beredar lebih dulu di pasaran. Semakin banyak pengguna yang menggunakan *smartphone* dengan pola spesifikasi tertentu dapat menunjukkan bahwa pola spesifkasi tersebut merupakan pola spesifikasi yang memiliki performa yang sesuai dengan kebutuhan pengguna.

Pola spesifikasi *smartphone* dapat diperoleh dengan mengolah data spesifikasi-spesifikasi *smartphone* yang saat ini banyak dimiliki oleh pengguna. Data spesifikasi-spesifikasi *smartphone* tersebut menghasilkan banyak pola hubungan sehingga tiap pola hubungan memiliki frekuensi.

Untuk mengetahui pola kombinasi spesifikasi *smartphone* yang banyak dimiliki oleh pengguna, diperlukan suatu proses penggalian data atau *data mining*. Proses *data mining* ini dapat menghasilkan pola kombinasi sesuai dengan kesamaan ciri spesifikasi *smartphone* dari sebagian besar data yang ada. Dalam proses *data mining* ada beberapa teknik yang dapat digunakan sesuai dengan permasalahan yang sedang dihadapi. Pada permasalahan dalam menemukan pola spesifikasi *smartphone* yang banyak dimiliki pengguna saat ini digunakan teknik *association rule.*

*Association rule* merupakan suatu metode dalam *data mining* yang dapat digunakan untuk menemukan hubungan atau korelasi antara item data dalam jumlah yang banyak [TUD-08]. *Association rule* terdiri dari dua tahap proses, yaitu pencarian semua kombinasi *item* yang paling sering muncul (pencarian *frequent itemsets*) dan pembangkitan *rule* yang telah terbentuk dari *frequent itemsets* [HAN-01].

Terdapat beberapa algoritma *association rule* dan yang paling sering digunakan sampai saat ini adalah algoritma apriori [ERW-09]. Namun algoritma apriori tersebut masih mempunyai kekurangan yaitu dalam melakukan pencarian *frequent* *itemset*, algoritma apriori harus melakukan *scanning* *database* berulang kali untuk setiap kombinasi *item*. Hal tersebut menyebabkan banyaknya waktu yang dibutuhkan untuk melakukan *scanning* *database*. Algoritma apriori tersebut juga membutuhkan alokasi memori yang besar dalam pencarian *itemsets*. Selain itu, dibutuhkan *generate candidate* yang besar untuk mendapatkan kombinasi *item* dari *database* [ERW-09].

Untuk dapat menghemat pemakaian memori komputer dan menyingkat waktu yang dibutuhkan maka diperlukan suatu algoritma yang tidak memerlukan *scanning* *database* yang berulang kali untuk setiap kombinasi *item*. Algoritma yang cocok untuk tujuan tersebut adalah algoritma *Frequent Pattern Growth (FP-Growth)*, karena pada algoritma *FP-Growth* *scanning* *database* untuk pencarian *frequent* *itemset* hanya dilakukan dua kali, *scanning* pertama yaitu pada saat mengumpulkan *frequent* *items* dan *scanning* yang kedua sebelum pembentukan *FP-Tree* [HAN-00]. Dalam algoritma *FP-Growth generate candidate* tidak dilakukan karena *FP-Growth* menggunakan konsep pembangunan *tree* dalam pencarian *frequent itemsets* sehingga membuat algoritma *FP-Growth* lebih cepat dari algoritma *apriori* [ERW-09].

Berdasarkan latar belakang yang telah dijelaskan tersebut, maka dalam penelitian ini penulis akan membuat suatu sistem dengan metode *association rule* menggunakan algoritma *Frequent Pattern Growth (FP-Growth)* untuk mengetahui pola spesifikasi *smartphone* yang banyak dimiliki oleh pengguna saat ini. Diharapkan hasil dari sistem yang dibangun ini dapat digunakan oleh *vendor* *smartphone* untuk referensi dalam produksi *smartphone* yang sesuai dengan kebutuhan banyak pengguna.

1. Rumusan Masalah

Permasalahan yang dijadikan subyek dalam skripsi ini adalah sebagai berikut:

1. Bagaimana menerapkan metode *association rule* menggunakan algoritma *Frequent Pattern Growth (FP-Growth)* untuk mengetahui pola spesifikasi *smartphone*.
2. Bagaimana kekuatan *rule* *(lift ratio)* dari *rule* hasil penerapan algoritma *FP-Growth*.
3. Bagaimana pengaruh nilai *minimum support* dan *minimum confidence* terhadap rata-rata *lift ratio*.
4. Batasan Masalah

Adapun batasan permasalahan dari skripsi ini adalah sebagai berikut:

1. Penelitian dilakukan dengan menggunakan data hasil *survey* mengenai spesifikasi-spesifikasi *smartphone* yang dimiliki oleh pengguna saat ini.
2. Penelitian dilakukan hanya terhadap *smartphone* yang memiliki sistem operasi Android, dan Windows Phone.
3. Penelitian dilakukan dengan menemukan pola spesifikasi *smartphone* banyak dimiliki pengguna saat ini.
4. Penelitian dilakukan dengan menerapkan algoritma *FP-Growth*.
5. Tujuan

Tujuan dari skripsi ini adalah:

1. Mengetahui pola kombinasi spesifikasi *smartphone* dengan menerapkan metode association rulemenggunakan algoritma *FP-Growth*.
2. Mengetahui tingkat kekuatan rule (lift ratio) hasil dari penerapan algoritma *FP-Growth*.
3. Mengetahui pengaruh nilai *minimum support* dan *minimum confidence* terhadap rata-rata *lift ratio*.
4. Manfaat

Manfaat yang diperoleh dari penelitian ini adalah sebagai berikut:

1. Memberikan informasi mengenai pola kombinasi spesifikasi *smartphone* yang memiliki sistem operasi Android, dan Windows Phone.
2. Untuk pemanfaatan lebih lanjut, informasi mengenai pola kombinasi spesifikasi *smartphone* dapat digunakan oleh *vendor* *smartphone* dalam pengembangan *smartphone* dengan spesifikasi yang sesuai dengan kebutuhan pengguna.
3. Sistematika Penulisan

Sistematika penulisan skripsi ini dibagi menjadi enam bab dengan masing-masing bab diuraikan sebagai berikut :

**BAB I PENDAHULUAN**

Memuat latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika penulisan.

**BAB II TINJAUAN PUSTAKA**

Memuat teori dasar dan teori penunjang penelitian yang berkaitan dengan *smartphone, data mining, association rule, FP-Tree, FP-Growth* dan *lift ratio.*

**BAB III METODOLOGI DAN PERANCANGAN**

Membahas tentang perancangan perangkat lunak yang dibangun, meliputi perancangan proses, perancangan tabel, dan perancangan antar muka.

**BAB IV IMPLEMENTASI DAN PEMBAHASAN**

Membahas tentang implementasi perangkat lunak yang digunakan untuk mengetahui pola spesifikasi *smartphone*.

**BAB V PENGUJIAN DAN ANALISIS**

Membahas mengenai pengujian serta analisis terhadap pola yang dihasilkan dari perangkat lunak yang telah dibuat.

**BAB V PENUTUP**

Memuat kesimpulan dari hasil penelitian serta saran sebagai pengembangan penelitian selanjutnya.

BAB II

TINJAUAN PUSTAKA

Pada bab ini akan dijelaskan mengenai dasar teori yang berkaitan dengan penelitian yang dilakukan. Dasar teori tersebut antara lain mengenai: *Smartphone,* konsep *Data mining,* *Association rule, FP-Tree, FP-Growth,* dan *Lift Ratio.*

**2.1 *Smartphone***

Arti istilah *Smartphone* yaitu sebuah ponsel yang mempunyai sistem operasi yang dipersembahkan untuk masyarakat umum, dimana pengguna dapat secara bebas menambahkan aplikasi, menambahkan fungsi, atau menyesuikan fungsi sesuai dengan keinginan pengguna [SHI-2010].

**2.1.1 *Smartphone* Android**

*Smartphone* Android adalah sebuah *smartphone* yang berjalan pada sistem operasi Android yang berbasis *open-source* milik perusahaan Google [CAS-14]. Sistem operasi Android dibangun dengan memungkinkan siapa saja *developer* yang berkeinginan untuk membuat aplikasi *mobile* dapat secara bebas membuat aplikasi *mobile* semenarik mungkin agar dapat mengambil keuntungan dari aplikasi yang dibuatnya. Karena sistem operasi Android berbasis *open-source* maka dapat secara bebas dikembangkan dan ditambahkan berbagai macam teknologi-teknologi baru sehingga dapat digunakan untuk membangun perangkat lunak yang inovatif [OPE-08].

Banyak produsen *smartphone* yang memproduksi *smartphone* yang memiliki sistem operasi Android, antara lain yaitu Samsung, LG, Lenovo, Sony, HTC, dan Motorola [FOG-14]. Perangkat yang menggunakan sistem operasi Android sampai saat ini adalah yang terbanyak dibandingkan perangkat yang menggunakan sistem operasi lainnya. Dimana pada tahun 2013 saja total pengiriman perangkat yang memiliki sistem operasi Android mencapai 878 juta dan diperkirakan tahun 2014 akan mecapai 1,1 miliar [GAR-14].

**2.1.2 *Smartphone* Windows Phone**

Windows Phone adalah sistem operasi *mobile* yang dikembangkan oleh Microsoft. Microsoft memberikan lisensi perangkat lunak untuk produsen *third-party hardware*, tetapi microsoft memberikan persyaratan minimum yang ketat untuk memastikan produk yang dihasilkan adalah yang terbaik. Nokia merupakan perangkat yang telah memilih Windows Phone sebagai OS untuk produk *smartphone*-nya [GSM-14].

**2.2 *Data mining***

**2.2.1 Pengertian *Data mining***

*Data mining* merupakan suatu teknik penemuan pola yang tersembunyi dalam suatu dataset yang besar yang berfokus pada hal yang berkaitan dengan kelayakan, kegunaan, efektifitas dan skalabilitas. *Data mining* terdiri dari algoritma yang memungkinkan seseorang untuk mendapatkan suatu pengetahuan dari data yang berjumlah besar [ZAK-13].

*Data mining* merupakan suatu proses yang digunakan untuk menemukan pola informasi yang tersembunyi pada suatu data dengan jumlah yang besar yang disimpan dalam *database*, data warehouse, atau repositori informasi lainnya [HAN-01].

*Data mining* merupakan teknik analisis yang dilakukan pada set data berjumlah besar yang ingin diteliti untuk menemukan hubungan tak terduga dan untuk meringkas data dalam cara baru agar lebih mudah dimengerti dan dapat berguna bagi pemilik data [HAD-01].

2.2.2 Fungsionalitas *Data Mining*

*Data mining* mempunyai fungsi yaitu digunakan untuk menentukan suatu pola yang dapat ditemukan. Tugas *data mining* secara umum dapat diklasifikasikan kedalam dua kategori, yaitu kategori deskriptif dan prediktif. Tugas *data mining* secara deskriptif yaitu mengkarakterisasi sifat umum dari suatu data di dalam *database*. Sedangkan tugas *data mining* secara prediktif yaitu melakukan inferensi pada data yang digunakan saat ini untuk membuat suatu prediksi [HAN-01].

Fungsionalitas *data mining* dan macam-macam pola pencarian yang dapat ditemukan dalam *data mining* adalah sebagai berikut [HAN-01]:

1. **Deskripsi Kelas/Konsep**

Data dalam kelas dikarakterisasi agar dapat membedakan dengan data dalam kelas lainnya, hal ini berguna untuk menggambarkan kelas secara individual dan konsep secara tepat.

1. **Analisis *Frequent* Patterns, Asosiasi, dan Korelasi**

Penemuan frekuensi dari pola yang sering terjadi pada data penemuan asosiasi dan korelasi dari data yang menunjukkan nilai kondisi dari atribut-atribut yang terjadi secara bersama-sama dan berulang kali.

1. **Klasifikasi dan Prediksi**

Klasifikasi merupakan suatu proses untuk menemukan sekumpulan model atau fungsi yang menjelaskan dan membedakan kelas/konsep data, yang bertujuan agar model atau fungsi tersebut dapat digunakan untuk memprediksi kelas dari suatu obyek yang belum diketahui kelasnya.

1. **Analisis *Cluster***

Analisis *cluster* digunakan untuk menganalisa obyek data tanpa perlu memperhatikan label kelas yang telah tersedia. *Clustering* dapat digunakan untuk menghasilkan label tersebut. Dimana data-data yang memiliki kemiripan tinggi dikelompokkan dalam *cluster* dan setiap *cluster* yang terbentuk dapat dijadikan sebagai kelas-kelas.

1. **Analisis *Outlier***

Analisis *outlier* dapat digunakan untuk menemukan objek data yang tidak sesuai dengan model data. *Outlier* bisa dideteksi menggunakan tes uji yang bersifat statistik yang mengasumsikan sebuah distribusi dan probabilitas model untuk data atau menggunakan ukuran jarak dimana suatu obyek yang jaraknya jauh dari *cluster* yang lain maka dianggap sebagai outlier.

1. **Analisis *Evolution***

Analisis *evolution* digunakan mencari model atau obyek yang memiliki kebiasaan yang berubah-ubah setiap waktu.

2.2.3 Proses *Knowledge Discovery from Database (KDD)*

*Data mining* dianggap memiliki persamaan kata dengan istilah *Knowledge Discovery from database (KDD)*. *Data mining* adalah suatu langkah penting yang terdapat dalam proses *knowledge discovery* [HAN-01].

Proses dalam *knowledge discovery* diuraikan dalam langkah-langkah berikut [HAN-01]:

1. ***Data Cleaning***

Merupakan suatu tahap yang dilakukan untuk menghilangkan *noise* dan data-data yang tidak konsisten.

1. ***Data Integration***

Merupakan tahap untuk mengkombinasikan beberapa sumber data.

1. ***Data Selection***

Merupakan tahap untuk memilih data-data yang relevan dengan tugas analisis dari *database*.

1. ***Data Transformation***

Merupakan suatu tahap dimana data-data diubah atau dikonsolidasikan ke dalam bentuk yang dibutuhkan untuk proses *mining*, misalnya dengan melakukan ringkasan atau operasi agregasi terhadap data.

1. ***Data mining***

Merupakan suatu proses dimana suatu metode atau teknik diaplikasikan untuk mendapatkan pola atau informasi tersembunyi dari data yang terpilih.

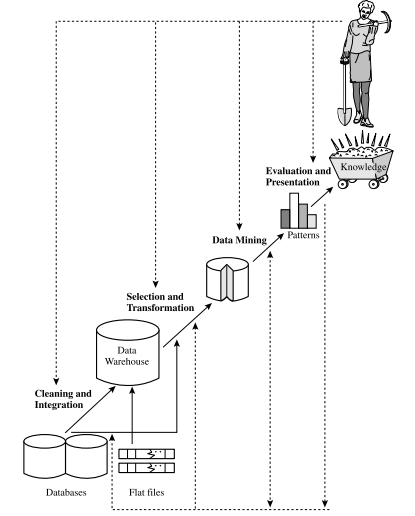
1. ***Pattern Evaluation***

Merupakan tahap untuk mengidentifikasi pola atau informasi yang benar-benar mewakili dari pengetahuan yang dimaksud.

1. ***Knowledge Presentation***

Merupakan tahap dimana visualisasi dan teknik representasi pengetahuan digunakan untuk menyajikan pengetahuan baru yang sudah didapat kepada pengguna.

Proses dari *Knowledge Discovery in Database* diilustrasikan pada Gambar 2.1 berikut:



Gambar 2.1 Proses *Knowledge Discovery in Database* [HAN-01].

### 2.2.4 Teknik *Data Mining*

Teknik-teknik yang digunakan dalam data *mining* antara lain [HAN-01]:

1. *Association rule Mining*

*Association rule Mining* merupakan suatu teknik dalam data *mining* yang digunakan untuk menemukan aturan asosiasi (*association rule*) antara suatu kombinasi *item*.

1. *Characterization*

*Characterization* merupakan teknik *data mining* yang dilakukan dengan meringkas karakteristik umum dari kelas data untukmembedakan suatu kelas data dengan ciri objek data secara umum, hal ini berguna untuk menggambarkan ciri kelas secara individual dan konsep secara tepat.

1. *Classification*

*Classification* merupakan suatu teknik dalam *data mining* yang digunakan untuk menemukan model/fungsi yang menjelaskan atau membedakan konsep/kelas data yang bertujuan untuk dapat memperkirakan kelas dari suatu objek yang labelnya tidak diketahui sebelumnya.

1. *Clustering*

*Clustering* merupakan suatu teknik dalam *data mining* dengan cara pengelompokan data tanpa berdasarkan kelas data/label tertentu. *Clustering* dapat digunakan untuk memberikan label pada kelas data yang belum diketahui.

**2.3 *Association Rule***

*Association rule mining* merupakan suatu prosedur untuk mencari hubungan antar-*item* dalam suatu *dataset*. *Association rule* biasanya digunakan pada analisis keranjang pasar untuk menemukan hubungan antar item yang dibeli pada *database* transaksi [HAN-01].

Secara umum, aturan *association rule* mining terdiri dari dua proses [HAN-01]:

1. Mencari semua *frequent* *itemsets*

Dimana setiap *itemsets* akan mempunyai frekuensi paling tidak seperti *minimum support* yang telah ditentukan sebelumnya.

1. Menghasilkan *association rule* yang kuat dari *frequent* *itemsets*

Dimana yang akan dijadikan hasil *rule* adalah *rule* yang memenuhi *minimum support* dan *minimum confidence*.

*Minimum support* adalah suatu parameter yang digunakan sebagai batasan frekuensi kejadian (*support count*) yang harus dipenuhi suatu kelompok data untuk dapat dijadikan *rule*. Sedangkan *Minimum confidence* adalah parameter yang mendefinisikan minimum level dari *confidence* yang harus dipenuhi oleh *rule* yang berkualitas [YUL-02].

*Association rule* mempunyai suatu bentuk yaitu "*If Antecedent Then Consequent*" dan dilengkapi dengan suatu nilai ukuran yaitu nilai *support* dan *confidence* dari masing-masing *rule* [LAR-08].

**2.3.1 *Support***

Nilai *Support* merupakan suatu nilai ukuran yang menunjukkan seberapa besar tingkat dominasi suatu *itemset* dari keseluruhan transaksi dalam *database* [HAN-01].

*Support* merupakan probabilitas pelanggan membeli beberapa produk secara bersamaan dari seluruh transaksi. *Support* untuk aturan “X=>Y” adalah probabilitas atribut atau kumpulan atribut X dan Y yang terjadi bersamaan [YUL-02].

Untuk mencari nilai *support* sebuah *item* diperoleh dengan rumus berikut:

(2.1)

**2.3.2 Confidence**

Nilai *Confidence* merupakan suatu nilai ukuran yang menunjukkan hubungan antara dua atau lebih *item* secara *conditional* [HAN-01].

*Confidence* merupakan probabilitas kejadian beberapa produk dibeli secara bersamaan dimana salah satu produk sudah pasti dibeli. Contoh: jika ada n transaksi dimana X dibeli, dan ada m transaksi dimana X dan Y dibeli bersamaan, maka *confidence* dari aturan if X then Y adalah m/n [YUL-02].

Untuk mencari nilai *confidence* sebuah item diperoleh dengan rumus berikut:

(2.2)

**2.4 *Frequent Pattern Tree* (*FP-Tree*)**

**2.4.1 Definisi *FP-Tree***

*FP-Tree* merupakan suatu struktur penyimpanan data yang dimampatkan, dimana *FP-Tree* bekerja dengan memetakan setiap data transaksi ke dalam setiap lintasan tertentu. Lintasan dalam *FP-Tree* tersebut dapat saling menimpa apabila terdapat transaksi yang mempunyai item yang sama. Struktur data pada *FP-Tree* akan semakin efektif bila memiliki semakin banyak item yang sama dalam suatu transaksi. Proses dalam *FP-Tree* sangat efisien karena hanya perlu dua kali pemindaian data [SAM-08].

*FP-Tree* merupakan variasi dari pohon struktur data yang merupakan struktur pohon awalan (*prefix*) untuk menyimpan informasi penting dan dikompresi tentang *frequent* *pattern* [Zen-02].

**2.4.2 Struktur *FP-Tree***

Struktur *Tree* dalam *Frequent* *Pattern* *Tree* (*FP-Tree*) yaitu [HAN-00] :

1. Terdiri dari satu *root* dengan label "null", satu himpunan *item-prefix* *substrees* sebagai anak dari *root*, dan sebuah tabel *header* *frequent* *item*.
2. Setiap node dalam *item-prefix substrees* terdiri dari tiga informasi yaitu:

* *Label name* yang memiliki fungsi untuk menginformasikan nama item yang direpresentasikan oleh simpul tersebut.
* *Support* *count* yang memiliki fungsi untuk merepresentasikan jumlah transaksi yang melalui simpul tersebut.
* *Pointer* penghubung yang ditandai dengan garis panah putus-putus dimana *pointer* penghubung tersebut mempunyai fungsi untuk menghubungkan simpul-simpul dengan nama *item* yang sama antar lintasan.

1. Setiap *entry* dalam tabel *header* *frequent item* terdiri dari dua informasi, yaitu nama *item* dan kepala dari node penghubung (sebuah pointer menunjuk ke node pertama pada *FP-Tree* dengan membawa nama *item*.

**2.4.3 Algoritma *FP-Tree***

Algoritma dari pembentukan *FP-Tree* dapat dilihat pada Gambar 2.2 [HAN-00]:

**Input :** Sebuah *database* transaksi (DB) dan nilai *minimum support*.

**Output :** *FP-Tree*, *Frequent* *pattern tree* dari DB.

**Method :** *FP-Tree* dibentuk dengan cara berikut ini.

1. Pemindaian *database* transaksi (DB) sekali. Mengumpulkan *Frequent* *item* (F), dan nilai *support* dari masing-masing *frequent* *item*. Kemudian urutkan *Frequent* *item* (F) secara *support*-*descending* mulai dari yang meiliki nilai *support* terbesar sampai terkecil dan hasil pengurutannya dilambangkan dengan *List of frequent* *item* (FItem-name).
2. Buat *root* dari *FP-Tree*, T, dan diberi label “null”. Untuk masing-masing transaksi (Trans) di DB lakukan langkah berikut ini:

2a. Pilih *frequent* *items* pada Trans dan urutkan berdasarkan urutan FItem-name. *Frequent* *item list* diurutkan di Trans be[p | P] dimana p adalah elemen pertama dan P adalah *list* berikutnya. Panggil insert\_*tree*([p|P], T).

2b. Fungsi insert\_*tree*([p|P], T) adalah seperti langkah berikut. **If** T memiliki anak N dimana N.*item-name* = p.item-name, kemudian *increment* nilai N dengan 1; **else** buat node baru N dan beri nilai 1, *parent link* nya dihubungkan dengan T, dan node-link dihubungkan dengan node yang memiliki *item-name* sama melalui struktur node-link. **If** P tidak kosong, **then** panggil insert\_*tree*(P, N) secara rekursif.

Gambar 2.2 Algoritma *FP-Tree*

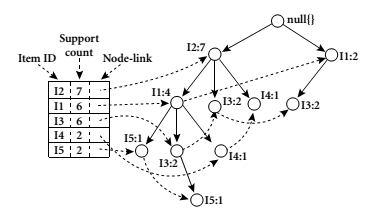
Contoh penerapan pembuatan *FP-Tree* yaitu dimisalkan terdapat data transaksi seperti pada Tabel 2.1.

Tabel 2.1 Contoh data transaksi [HAN-01]

|  |  |
| --- | --- |
| **TID** | **List Items** |
| T100 | 11, 12, 15 |
| T200 | 12, 14 |
| T300 | 12, 13 |
| T400 | 11, 12, 14 |
| T500 | 11, 13 |
| T600 | 12, 13 |
| T700 | 11, 13 |
| T800 | 11, 12, 13, 15 |
| T900 | 11, 12, 13 |

Yang pertama kali dilakukan untuk pembangunan *FP-Tree* adalah *scan database* untuk menemukan frekuensi items dan *frequent 1-itemset* (item yang memenuhi *minimum support*) berdasarkan *minimum support* yang telah ditentukan yaitu 2. *Frequent 1-itemset* diurutkan berdasarkan nilai *support* yang terbesar. Hasil *frequent 1-itemset* yang didapat setelah perhitungan *support* adalah L ={{I2: 7}, {I1: 6}, {I3: 6}, {I4: 2}, {I5: 2}}.

*Scan database* yang kedua, item dari masing-masing transaksi diurutkan berdasarkan nilai *support* terbesar (berdasarkan L). Selanjutnya, untuk pembentukan *FP-Tree* yang pertama adalah membuat *root* dan diberi label null. Setiap cabang dari *FP-Tree* nantinya dibuat untuk masing-masing transaksi. Untuk transaksi pertama T100 = (11, 12, 15) yaitu terdiri dari tiga item (12, 11, 15) berdasarkan pengurutan *support*. Cabang pertama dalam *FP-Tree* dibuat untuk 3 node {12 : 1}, {11 : 1}, {15 : 1}, dimana node pertama yaitu node 12 dihubungkan ke *root*, node 11 menjadi anak dari node 12, dan node 15 menjadi anak dari node 11. Transaksi selanjutnya untuk pembentukan cabang kedua dari *FP-Tree* yang terdiri dari dua node yaitu 12 dan 14. Karena node 12 sudah terbentuk pada *FP-Tree* maka tidak perlu dibentuk node yang sama melainkan hanya menambah 1 nilai *count* dari node 12 yang telah ada sebelumnya menjadi {12 : 2} dan membuat node baru {14 : 1} menjadi anak dari node {12 : 2}. Hasil *FP-Tree* setelah pembacaan semua transaksi dapat dilihat pada Gambar 2.3 [HAN-01].



Gambar 2.3 Pembentukan *FP-Tree* lengkap [HAN-01]

**2.5 *Frequent Pattern Growth (FP-Growth)***

**2.5.1 Definisi *FP-Growth***

*Frequent Pattern Growth (FP-Growth)* adalah salah algoritma tercepat dan paling populer untuk menentukan himpunan data yang paling sering muncul (*frequent* *itemset*) dalam suatu *dataset*. Dalam penemuan *frequent itemset*, algoritma *FP-Growth* menggunakan *FP-Tree (Frequent Pattern Tree).* *FP-Growth* merupakan metode yang efisien untuk *data mining* dan mempunyai kinerja lebih cepat daripada apriori dan beberapa metode baru dalam *frequent* *pattern mining* [HAN-00].

*FP-Growth* merupakan representasi sebuah *prefix* *tree* dari *database* transaksi (yang disebut *FP-Tree*) yang dapat menghemat sejumlah besar memori untuk penyimpanan transaksi [BOR-05].

**2.5.2 Langkah Utama Algoritma FP-Growth**

Tiga langkah utama Algoritma *FP-Growth* [SAM-08]:

1. Tahap Pembangkitan *Conditional Pattern Base*

Arti istilah *Conditional Pattern Base* tersendiri adalah sub*database* yang berisi *prefix* *path* (lintasan awalan) dan *suffix* pattern (pola akhiran). Pembangkitan *conditional pattern base* ini didapatkan melalui *FP-Tree* yang sudah dibangun sebelumnya.

1. Tahap Pembangkitan *Conditional FP-tree*

Pada tahap pembangkitan Conditional *FP-Tree* ini, *support* *count* dari setiap item dalam setiap *conditional pattern* *base* dijumlahkan, lalu setiap item yang memiliki jumlah *support* *count* lebih besar atau sama dengan m*inimum support* *count* akan dibangkitkan dengan *conditional FP-Tree*.

1. Tahap Pencarian *Frequent* *Itemset*

Pada tahap Pencarian *frequent* *itemset* ini, apabila *Conditional FP-Tree* yang dibangkitkan sebelumnya merupakan lintasan tunggal, maka akan didapatkan *frequent* *itemset* dengan melakukan kombinasi item untuk setiap *conditional FP-Tree*. Sedangkan apabila bukan lintasan tunggal, maka dilakukan pembangkitan *FP-Growth* secara rekursif.

Ketiga tahap dalam algoritma *FP-Growth* tersebut adalah langkah yang akan dilakukan untuk mendapat *frequent* *itemset*.

**2.5.3 Algoritma FP-Growth**

Algoritma dari *FP-Growth* dapat dilihat pada Gambar 2.4 [HAN-00]:

**Input** : *Database* DB, *FP-tree* dan *minimum support*.

**Output** : Kumpulan lengkap dari *frequent* *patterns*.

**Method** : Memanggil FP-growth(FP-*tree*,null).

Procedure FP-growth(*Tree*,α) {

(1) **if** *Tree* mengandung *single* *prefix* *path*

(2) **then**{

(3) P menjadi *single* *prefix*-*path* bagian dari *Tree*;

(4) Q menjadi *multipath* *part* dengan node cabang atas diganti dengan “null root”;

(5) **for each** kombinasi (dinotasikan β) dari node-node di *path* P **do**

(6) Bangkitkan pola β ∪ α dengan *support*=*minimum* *support* dari node-node pada β;

(7) freq\_pattern\_set (P) menjadi kumpulan pola sehingga dibangkitkan;

}

(8) **else** Q menjadi *Tree*;

(9) **for each** item ai pada Q **do**{ // *Mining multipath FP-tree*

(10) Bangkitkan pola β=ai ∪ α dengan *support*=ai.*support*;

(11) Bangun β’s *conditional pattern-base* dan **then** β’s *conditional FP-tree* *Tree*β;

(12) **if** *Tree*β ≠ ∅

(13) **then** panggil FP-growth(*Tree*β,β);

(14) freq\_pattern\_set(Q) menjadi kumpulan pola sehingga dibangkitkan; }

(15) return(freq\_pattern\_set(P) ∪ freq\_pattern\_set(Q) ∪ (freq\_pattern\_set(P) × freq\_pattern\_set(Q)))

}

Gambar 2.4 Algoritma *FP-Growth*

Pada permasalahan data transaksi Tabel 2.1, pola *suffix* dimulai dengan item yang berada pada urutan terakhir dari pola *frequent 1-itemset*, yaitu item 15. Berdasarkan *FP-Tree* yang telah terbangun pada Gambar 2.3 maka untuk pola *suffix* 15 ditemukan pada dua cabang yaitu lintasan {12, 11, 15 : 1} dan lintasan {12, 11, 13, 15 : 1} sehingga lintasan {12, 11 : 1} dan lintasan {12, 11, 13 : 1} merupakan *conditional pattern base* dari 15. Selanjutnya, karena pada kedua lintasan tersebut node 12 dan 11 nilai *supportnya* memenuhi *minimum support* sedangkan node 13 tidak, maka *conditional FP-Tree* yang terbentuk yaitu {12 : 2, 11 : 2}. Hasil dari *conditional FP-Tree* tersebut menghasilkan *frequent itemset* yaitu {{12, 15 : 2}, {11, 15 : 2}, {12, 11, 15 : 2}}. Hasil dari keseluruhan proses *FP-Growth* pada contoh permasalahan ini dapat dilihat pada Tabel 2.2 [HAN-01].

Tabel 2.2 Hasil proses *FP-Growth* [HAN-01]

|  |  |  |  |
| --- | --- | --- | --- |
| ***Item*** | ***Conditional Pattern Base*** | ***Conditional FP-Tree*** | ***Frequent Pattern*** |
| 15 | {{12,11 :1},  {12,11,13 :1}} | {12:2, 11:2} | {12, 15 :2},  {11, 15 :2},  {12, 11, 15 :2} |
| 14 | {{12,11 :1},  {12 :1}} | {12:2} | {12, 14 :2} |
| 13 | {{12,11 :2},  {12 :2},  {11 :2}} | {12:4, 11:2},{11:2} | {12, 13 :4},  {11, 13 :4},  {12, 11, 13 :2} |
| 11 | {{12 :4}} | {12 :4} | {12, 11 :4} |

**2.6 *Lift Ratio***

*Lift ratio* adalah suatu cara untuk menilai kekuatan dari *association rule*. *Lift ratio* dilakukan dengan membandingkan nilai *confidence* dari *rule* dengan nilai *benchmark*, dimana diasumsikan bahwa terjadinya *consequent* kumpulan item pada sebuah transaksi adalah tidak tergantung dari terjadinya *antecedent* dari masing-masing *rule* [SHM-05].

Untuk mencari nilai *Lift ratio* diperoleh dengan persamaan berikut:

(2.3)

Dimana,

(2.4)

Apabila nilai *lift* *ratio* lebih besar dari 1, maka menunjukkan adanya kegunaan dari aturan tersebut. Lebih tinggi nilai *lift* *ratio* maka lebih besar kekuatan asosiasinya [SHM-05].