# BAB II
# LANDASAN TEORI

## 2.1 Tinjauan Pustaka

### 2.1.1 Penelitian Sebelumnya

Adapun perbandingan antara penelitian analisis data yang dibuat dengan penelitian analisis data sebelumnya terkait Skripsi ini dapat dilihat seperti pada tabel 2.1.

Tabel 2.1 Perbandingan dengan Penelitian Sebelumnya

| Perbandingan | Penelitian 1 | Penelitian 2 | Penelitian Sekarang |
|---|---|---|---|
| Objek Yang Diteliti | Keranjang Pasar | Data Transaksi Peminjaman Buku | Data Peminjaman Buku Perpustakaan |
| Tujuan | Untuk Mendadapatkan Strategi Yang Digunakan Oleh Perusahaan Pemasaran (Swalayan) | Untuk Mengetahui Seberapa Kuat Rekomendasi Sistem Yang Dibuat | Untuk Mengetahui Pola/Rule Yang Terbentuk Dari Peminjaman Buku |
| Studi Kasus | Salah Satu Supermarket | Perpustakaan UKDW | Perpustakaan USAHID Surakarta |
| Algoritma | *FP-Growth* | *FP-Growth* | *FP-Growth* |
| *Software* Pengembang | - | *Web application* | *Dekstop* |

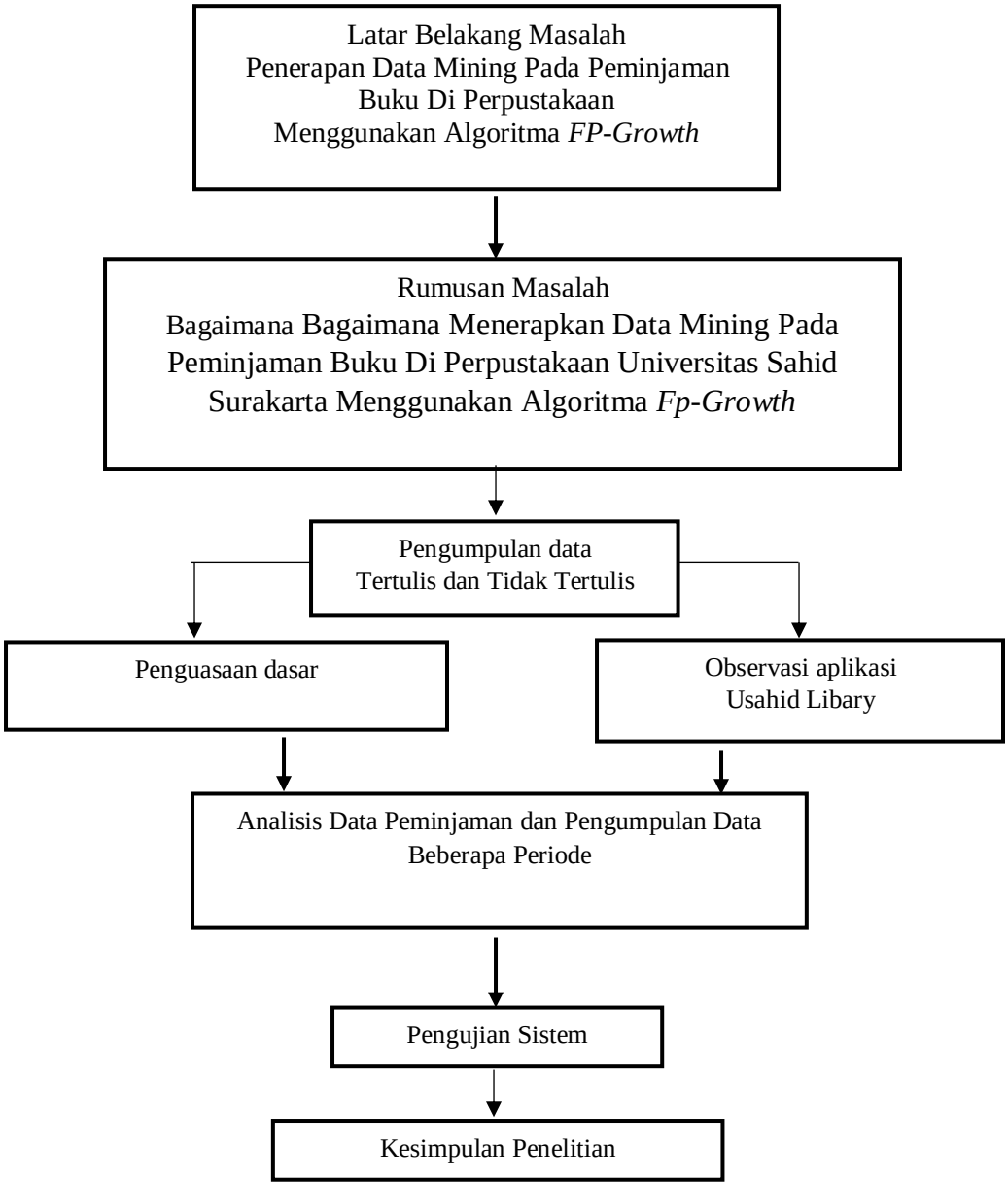
Asosiasi Data Mining Menggunakan Algoritma *FP-Growth* Untuk *Market Basket Analysis* (Fatihatul, 2014). Pada penelitian tersebut dilakukan analisa data untuk mengetahui perilaku pembelian dari konsumen yang dapat dimanfaatkan untuk strategi pemasaran. Penelitian tersebut tidak menerapkannya dalam bentuk sistem namun penelitian tersebut menggunakan *software* WEKA 3.6.4 hanya untuk mencari aturan yang dibentuk dari beberapa transaksi pembelian yang terjadi di *supermarket* dalam suatu periode tertentu. Berdasarkan aturan yang dihasilkan tersebut, maka strategi pemasaran yang dapat digunakan oleh *supermarket* tersebut adalah menempatkan *item-item* yang saling *frequent* secara berdekatan. Pada

penelitian tersebut menggunakan salah satu algoritma dari metode asosiasi yaitu algoritma *FP-Growth*. Pengolahan data pada penelitian sebelumnya menggunakan WEKA sehingga format *file* di *Excel* harus .csv terlebih dahulu. Meskipun melalui WEKA dapat diketahui *rule association* dengan cepat dan mudah, namun jika dilakukan secara manual seperti pada Analisis data Pola Peminjaman Buku Menggunakan Algoritma *FP-Growth* akan lebih menyenangkan dan mudah dimengerti perbedaan antara algoritma *FP-Growth* dengan algoritma lainnya.

Implementasi Algoritma *FP-Growth* Untuk Sistem Rekomendasi Buku Di Perpustakaan (Miraldi, Rachmat, & Susanto, 2014). Pada penelitian tersebut dilakukan analisa data untuk mengetahui buku yang akan direkomendasikan untuk keperluan perpustakaan. Penelitian tersebut menerapkannya dalam bentuk sistem namun penelitian tersebut menggunakan pemrograman *web* hanya untuk mencari merekomendasi buku di perpustakaan UKDW dalam suatu periode tertentu. Untuk data yang di ambil yaitu secara acak dan penelitian ini sebagai alat bantu berupa sistem pencarian untuk mempermudah pengunjung perpustakaan mencari buku.

## 2.2 Kerangka Pemikiran

Kerangka pemikiran dari penelitian Skripsi ini telah tercantum pada Gambar 2.1.




**Gambar 2.1. Kerangka Pemikiran**

## 2.3 Dasar Teori

### 2.3.1 Pengertian Data *Mining*

Data *mining* adalah kegiatan menemukan pola yang menarik dari data dalam jumlah besar, data dapat disimpan dalam *database*, data *warehouse*, atau penyimpanan informasi lainnya (Han & Kamber, 2006). Data *mining* berkaitan dengan bidang ilmu-ilmu lain, seperti *Database system*, data *warehousing*, statistik, *machine learning*, *information retrieval*, dan komputasi tingkat tinggi. Selain itu, data *mining* didukung oleh ilmu lain seperti *neural network*, pengenalan pola, *spatial* data *analysis*, *image database*, *signal processing* (Han & Kamber, 2006). Data *mining*, sering juga disebut sebagai *Knowledge Discovery In Database* (KDD). KDD adalah kegiatan yang meliputi pengumpulan, pemakaian data historis untuk menemukan keteraturan, pola atau hubungan dalam set data berukuran besar (Han & Kamber, 2006). Data *mining* didefinisikan sebagai proses menemukan pola-pola dalam data. Proses ini otomatis atau seringnya semiotomatis. Pola yang ditemukan harus penuh arti dan pola tersebut memberikan keuntungan, biasanya keuntungan secara ekonomi. Data yang dibutuhkan dalam jumlah besar (Witten & Frank, 2005). Karakteristik data *mining* sebagai berikut (Davies, 2004) :

a Data *mining* berhubungan dengan penemuan sesuatu yang tersembunyi dan pola data tertentu yang tidak diketahui sebelumnya.

b Data *mining* biasa menggunakan data yang sangat besar. Biasanya data yang besar digunakan untuk membuat hasil lebih dipercaya.

c Data *mining* berguna untuk membuat keputusan yang kritis, terutama dalam strategi.

Berdasarkan beberapa pengertian tersebut dapat ditarik kesimpulan bahwa data *mining* adalah suatu teknik menggali informasi berharga yang terpendam atau tersembunyi pada suatu koleksi data (*Database*) yang sangat besar sehingga ditemukan suatu pola yang menarik yang sebelumnya tidak diketahui.

### 2.3.2 *Association Rules*

Analisis asosiasi atau *association rule mining* adalah teknik data mining untuk menemukan aturan asosiatif antara suatu kombinasi item (Han & Kamber,

2006). Contoh dari aturan asosiatif dari analisis pembelian di suatu pasar swalayan adalah dapat diketahuinya berapa besar kemungkinan seorang pelanggan membeli roti bersamaan dengan susu. Dengan pengetahuan tersebut pemilik pasar swalayan dapat mengatur penempatan barangnya atau merancang kampanye pemasaran dengan memakai kupon diskon untuk kombinasi barang tertentu. Karena analisis asosiasi menjadi terkenal karena aplikasinya untuk menganalisis isi keranjang belanja di pasar swalayan, analisis asosiasi juga sering disebut dengan istilah market basket analysis. Analisis asosiasi dikenal juga sebagai salah satu metode data mining yang menjadi dasar dari berbagai metode data mining lainnya. Khususnya salah satu tahap dari analisis asosiasi yang disebut analisis pola frekuensi tinggi (*frequent pattern mining*) menarik perhatian banyak peneliti untuk menghasilkan algoritma yang efisien. Pembacaan/pengartian makna hasil pola/rule yang diperoleh, berdasarkan pada interpretasi masing-masing individu sehingga kemungkinan akan diperoleh makna yang berbeda meskipun dari hasil pola/rule yang sama (Berry & Linoff, 2004).

**2.3.2.1.*Support***

Support diartikan sebagai peluang kejadian sebuah rule terhadap seluruh transaksi dalam dataset (Han & Kamber, 2006). Support dapat diartikan pula sebagai peluang kejadian dalam dataset yang mengandung A dan B , misal bentuk implikasi A → B, nilai support terhadap implikasi tersebut dihitung dengan persamaan 1 yaitu sebagai berikut.

$$\textbf{Support } A \rightarrow B = P(A \cup B) = \frac{\textbf{Support Count } A \cap B}{\textbf{Total Transaksi}} \qquad (1)$$

Dengan P(A∩B )merupakan peluang kejadian A dan B muncul bersamaan. Support count $A \cap B$ menunjukkan frekuensi kemunculan A dan B bersamaan total transaksi menunjukan total keseluruhan transaksi dataset.

**2.3.2.2.*Confidence***

Confidence diartikan sebagai peluang kejadian dalam transaksi yang mengandung kejadian A dan B (mengambil contoh kasus sebelumnya), sehingga dalam confidence dihitung seberapa sering kejadian B muncul bersamaan dengan

kejadian A juga muncul. Nilai confidence dapat dihitung dengan persamaan 2 yaitu sebagai berikut.

$$\textbf{Confidence } A \rightarrow B = P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{\textbf{Support Count } A \cap B}{\textbf{Support Count } (A)} \qquad (2)$$

Dengan P(B|A) merupakan peluang kejadian bersyarat B muncul jika A muncul. Support count A ∩ B menunjukkan frekuensi kemunculan A dan B bersamaan. Support count A menunjukkan frekuensi kemunculan A dalam dataset.

**2.3.2.3.*Lift***

Penentuan rule asosiasi dengan menggunakan support dan confidence saja tidak cukup untuk menunjukkan hubungan yang terdapat antar item dalam sebuah rule. Oleh karena itu, dibutuhkan perhitungan korelasi untuk mengetahui korelasi yang dimiliki antar item dalam sebuah rule. Ada banyak cara dalam menghitung korelasi, salah satu diantaranya adalah lift. Lift merupakan cara perhitungan korelasi yang paling sederhana. Perhitungan dengan lift dapat menunjukkan apakah hubungan antar item dalam rule saling berkolerasi atau tidak. Dicontohkan dalam sebuah implikasi, misal A→B, dari bentuk implikasi tersebut tidak dapat diketahui apakah terdapat korelasi antara A dan B. Lift dapat menunjukkan korelasi diantara keduanya dengan persamaan 3 yaitu sebagai berikut.

$$\textbf{Lift } (A \rightarrow B) \frac{P(A \cap B)}{P(A) \,.\, P(B)} = \frac{\textbf{Confidence } A \cap B}{\textbf{Support } (B)} \qquad (3)$$

Dimana P(A ∩ B) menunjukan peluang kejadian A dan B muncul bersamaan, P(A) menunjukkan peluang kejadian A, P(B) menunjukkan peluang kejadian B. Persamaan tersebut sama dengan perhitungan confidence(A→B)/ support(B) atau P(B|A)/P(B). Jika nilai lift yang diperoleh lebih besar dari 1 maka kejadian A berkorelasi positif terhadap kejadian B sebaliknya jika nilai lift kurang dari 1 maka kejadian A berkolerasi negatif. Jika nilai lift sama dengan 1 maka tidak terdapat korelasi antara kejadian A dan kejadian B (independent correlation) (Han & Kamber, 2006).

### 2.3.3 Algoritma *FP-Growth*

#### 2.2.3.1 *FP-Tree* Dalam Algoritma *FP-Growth*

Algoritma *FP-Growth* merupakan pengembangan dari algoritma *Apriori*, sehingga kekurangan dari algoritma *Apriori* diperbaiki oleh algoritma *FP-Growth*. *FP-Growth* merupakan salah satu alternatif algoritma yang dapat digunakan untuk menentukan himpunan data yang paling sering muncul (frequent itemset) dalam sebuah kumpulan data. *FP-Growth* menggunakan pendekatan yang berbeda dari paradigma yang selama ini sering digunakan, yaitu paradigma Apriori.

Paradigma Apriori yang dikembangkan oleh Agrawal dan Srikan (1994), yaitu anti-monotone Apriori Heuristic : Setiap pola dengan panjang pola k yang tidak sering muncul (tidak frequent) dalam sebuah kumpulan data, maka pola dengan panjang (k+1) yang mengandung sub pola k tersebut tidak akan sering muncul (tidak frequent) (Han & Kamber, 2006). Ide dasar paradigma Apriori ini adalah dengan mencari himpunan kandidat dengan panjang (k+1) dari sekumpulan pola frequent dengan panjang k, lalu mencocokkan jumlah kemunculan pola tersebut dengan informasi yang terdapat dalam *database*. Adapun hal ini akan mengakibatkan algoritma Apriori akan melakukan scanning *database* yang berulang-ulang, apalagi jika jumlah Data cukup besar. Berbeda dengan algoritma *FP-Growth* yang hanya memerlukan dua kali scanning *database* untuk menentukan frequent itemset.

Struktur Data yang digunakan untuk mencari frequent itemset dengan algoritma *FP-Growth* adalah perluasan dari penggunaan sebuah pohon prefix, yang biasa disebut dengan *FP-Tree* (Han & Kamber, 2006). Dengan menggunakan *FP-Tree*, algoritma *FP-Growth* dapat langsung mengekstrak frequent itemset dari *FP-Tree* yang telah terbentuk dengan menggunakan prinsip divide and conquer.

#### 2.2.3.2 Kelebihan Algoritma *FP-Growth* dengan Algoritma *Apriori*

Pada algoritma Apriori diperlukan generate candidate untuk mendapatkan frequent itemsets. Akan tetapi, di algoritma *FP-Growth* generate candidate tidak dilakukan karena *FP-Growth* menggunakan konsep pembangunan tree dalam pencarian frequent itemsets. Hal tersebutlah yang menyebabkan algoritma FP-

Growth lebih cepat dari algoritma Apriori (Surojo, 2012).

#### 2.2.3.3 Tahapan Algoritma *FP-Growth*

**1. *FP-Tree***

*FP-Tree* adalah struktur penyimpanan data yang dipadatkan. *FP-Tree* dibangun dengan memetakan setiap data transaksi ke dalam setiap lintasan tertentu dalam *FP-Tree*. Karena dalam setiap transaksi yang dipetakan mungkin ada transaksi yang memiliki item yang sama, maka lintasannya memungkinkan untuk saling menimpa. Semakin banyak data transaksi yang memiliki item yang sama, maka proses pemadatan dengan struktur data *FP-Tree* semakin efektif. Kelebihan dari *FP-Tree* adalah hanya memerlukan dua kali scanning data transaksi yang terbukti sangat efisien.

Definisi *FP-Tree* adalah sebuah pohon dengan (Samuel, 2008):

a *FP-Tree* dibentuk oleh sebuah akar yang bernama null, sekumpulan cabang yang terdiri dari item-item tertentu, dan sebuah tabel frequent header.

b Setiap simpul dalam *FP-Tree* mengandung 3 informasi penting, yaitu label item (menginformasikan jenis item yang direpresentasikan simpul tersebut), *support count* (merepresentasikan jumlah lintasan transaksi yang melalui simpul tersebut) dan pointer penghubung (yang menghubungka simpul-simpul dengan label item sama antar lintasan, ditandai dengan garis panah putus-putus).

Contoh kasus yang akan digunakan adalah 10 transaksi dengan minimum *support count* = 20% dan minimum *confidence* = 75%, seperti dalam tabel 2.2.

Tabel 2.2 Contoh Data Transaksi (Surojo, 2012).

| No. | Transaksi |
|---|---|
| 1 | 4, 5, 3 |
| 2 | 6, 9, 10, 4, 7 |
| 3 | 6, 1, 2, 3, 7 |
| 4 | 6, 1, 3 |
| 5 | 4, 8, 3, 7 |
| 6 | 4, 6, 3, 7 |
| 7 | 3, 5 |
| 8 | 4, 7, 3 |
| 9 | 4, 6, 3, 1 |
| 10 | 7, 1, 4 |

Dalam membangun *FP-Tree* diperlukan dua kali penelusuran *Database*. Penelusuran *database* yang pertama digunakan untuk menghitung nilai *support* masing-masing item dan memilih item yang memenuhi nilai minimum *support*. Hasil dari proses penelusuran *database* yang pertama adalah menghitung jumlah frekuensi kemunculan tiap item yang ada didalam *database* dan mengurutkannya berdasarkan jumlah frekuensi kemunculan terbesar, seperti yang terdapat dalam tabel berikut:

Tabel 2.3 Jumlah Frekuensi Tiap Item (Surojo, 2012).

| Item | Frekuensi |
|---|---|
| 3 | 8 |
| 4 | 7 |
| 7 | 6 |
| 6 | 5 |
| 1 | 4 |
| 5 | 2 |
| 2 | 1 |
| 8 | 1 |
| 9 | 1 |
| 10 | 1 |

Dari tabel di atas, diperoleh *itemset* yang memiliki frekuensi di atas minimum *support count* <= 2, yaitu 3, 4, 7, 6, 1 dan 5 yang selanjutnya diberi nama F-*list*. Keenam item ini akan berpengaruh saat pembuatan *FP-Tree*. Sedangkan 2, 8, 9 dan 10 dibuang karena tidak memenuhi minimum *support count* = 2. Tabel dibawah adalah tabel *header* atau sering disebut F-*list*:

Tabel 2.4 Tabel *Header* atau F-*list* (Surojo, 2012).

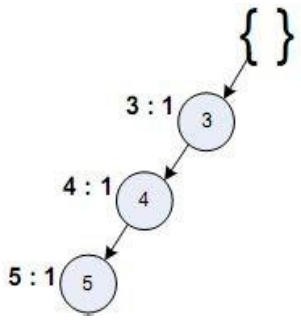
| Item | Frekuensi |
|---|---|
| 3 | 8 |
| 4 | 7 |
| 7 | 6 |
| 6 | 5 |
| 1 | 4 |
| 5 | 2 |

Setelah dibuat F-*list*, urutkan *itemset* pada tiap transaksi berdasarkan frekuensi paling tinggi atau menurut F-*list*. Setelah itu buatlah *Tree* berurut berdasarkan transaksi ID-nya, seperti yang terdapat dalam tabel 2.5.

Tabel 2.5 Data Transaksi dengan Minimum *Support* (Surojo, 2012).

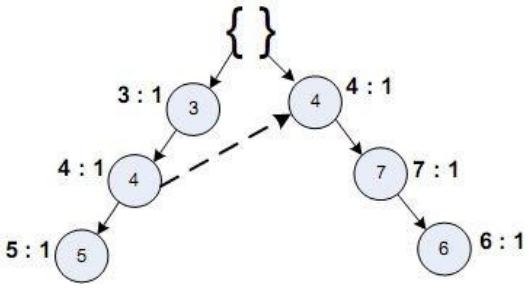
| TID | Item |
|---|---|
| 1 | {3,4,5} |
| 2 | {4,7,6} |
| 3 | {3,7,6,1} |
| 4 | {3,6,1} |
| 5 | {3,4,7} |
| 6 | {3,4,7,6} |
| 7 | {3,5} |
| 8 | {3,4,7} |
| 9 | {3,4,6,1} |
| 10 | {4,7,1} |

Setelah *itemset* disusun ulang berdasarkan F-*list*, dilakukan penelusuran *database* yang kedua yaitu membaca tiap transaksi diawali dengan membaca TID 1 untuk membuat *FP-Tree*. TID 1 {3,4,5} akan membuat simpul 3, 4 dan 5,sehinga terbentuk lintasan {} →3→4→5 dengan *support count* awal bernilai satu.



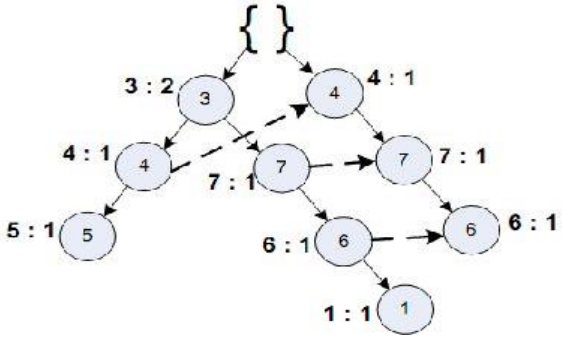

**Gambar 2.2 *FP-Tree* Pada Transaksi TID 1 *(*Surojo, 2012)**

Setelah pembacaan TID 1, maka selanjutnya membaca TID 2 yaitu {4,7,6} yang membentuk lintasan kedua yaitu *{} →4→7→6 dengan support count* awal bernilai satu juga. Walaupun 4 ada pada TID 1, tetapi karena *prefix* transaksinya tidak sama, maka TID 2 ini tidak dipadatkan ke lintasan TID 1.



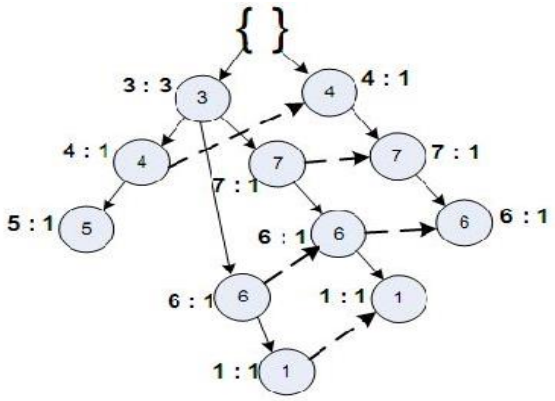

**Gambar 2.3 *FP-Tree* Pada Transaksi TID 2 *(*Surojo, 2012)**

Setelah pembacaan TID 2, maka selanjutnya membaca TID 3 yaitu {3,7,6,1}. Karena memiliki salah satu *prefix* yang sama dengan lintasan pertama, yaitu 3, maka lintasan TID 3 bisa dipadatkan pada lintasan TID 1. Selain itu tambahkan *support count* 3 menjadi dua karena telah dilewati atau dipadatkan sebanyak dua kali, sedangkan 7, 6 dan 1 masing-masing bernilai *support count* satu.



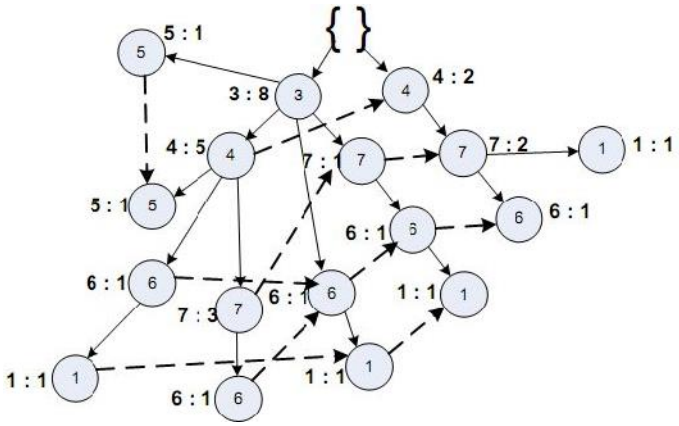

**Gambar 2.4 *FP-Tree* Pada Transaksi TID 3 *(*Surojo, 2012)**

Setelah pembacaan TID 3, maka selanjutnya membaca TID 4 yaitu {3,6,1}. Karena memiliki *prefix* yang sama dengan lintasan pertama, yaitu 3, maka lintasan TID 4 dapat ditimpakan di 3. Selain itu tambahkan *support count* 3 menjadi tiga karena telah dilewati atau dipadatkan sebanyak tiga kali, sedangkan 6 dan 1 masing-masing bernilai *support count* satu.




**Gambar 2.5 Hasil Pembetukan *FP-Tree* Setelah Pembacaan TID 4 *(*Surojo, 2012)**

Proses selanjutnya sama seperti pada gambar 2.2-2.5 untuk setiap transaksi TID 5-10. Pada TID 10 yaitu {4,7,1}, karena memiliki *prefix* yang sama dengan lintasan kedua, yaitu 4 dan 7, maka lintasan TID 10 dapat ditimpakan di 4 dan 7. Selain itu tambahkan *support count* 4 menjadi dua dan *support count* 7 menjadi dua, karena 4 telah dilewati atau dipadatkan sebanyak dua kali dan 7 telah dilewati atau dipadatkan sebanyak dua kali, sedangkan 1 bernilai *support count* satu.




**Gambar 2.6 Hasil Pembetukan FP-Tree Setelah Pembacaan TID 10 (Surojo, 2012)**

Setelah penelusuran *database* selesai, maka proses pembuatan *FP-Tree* telah selesai dan hasil *Tree* yang didapat dari contoh kasus ini dapat dilihat pada gambar 2.6. Gambar 2.2 - 2.6 menunjukkan proses terbentuknya *FP-Tree* setiap TID dibaca. Setiap simpul pada *FP-Tree* mengandung nama sebuah item dan *count*er *support* yang berfungsi untuk menghitung frekuensi kemunculan item tersebut dalam tiap lintasan transaksi.

**2. Penerapan Algoritma *FP-Growth***

Algoritma *FP-Growth* merupakam algoritma pencarian frequent pattern yang efisien karena menggunakan struktur data tree (*FP-Tree*). Algoritma FP-Growth menemukan frequent itemset yang berakhiran suffix tertentu dengan menggunakan metode divide and conquer untuk memecah problem menjadi subproblem yang lebih kecil (Samuel, 2008).

Setelah tahap pembangunan *FP-Tree* dari sekumpulan data transaksi, akan diterapkan algoritma *FP-Growth* untuk mencari frequent itemset yang signifikan. Algoritma *FP-Growth* dibagi menjadi tiga langkah utama, yaitu (Han & Kamber, 2006):

a Tahap Pembangkitan Conditional pattern base

Conditional pattern base merupakan sub *database* yang berisi *prefix path* (lintasan *prefix*) dan *suffix pattern* (pola akhiran). Pembangkitan *conditional pattern base* didapatkan melalui *FP-Tree* yang telah dibangun sebelumnya.

b Tahap Pembangkitan Conditional *FP-Tree*

Pada tahap ini, support count dari setiap item pada setiap *conditional pattern base* dijumlahkan, lalu setiap item yang memiliki jumlah *support count* lebih besar sama dengan minimum support count akan 1

c Tahap Pencarian *frequent itemset*

Apabila Conditional *FP-Tree* merupakan lintasan tunggal (single path), maka didapatkan frequent itemset dengan melakukan kombinasi item untuk setiap conditional *FP-Tree*. Jika bukan lintasan tunggal, maka dilakukan pembangkitan *FP-Growth* secara rekursif.

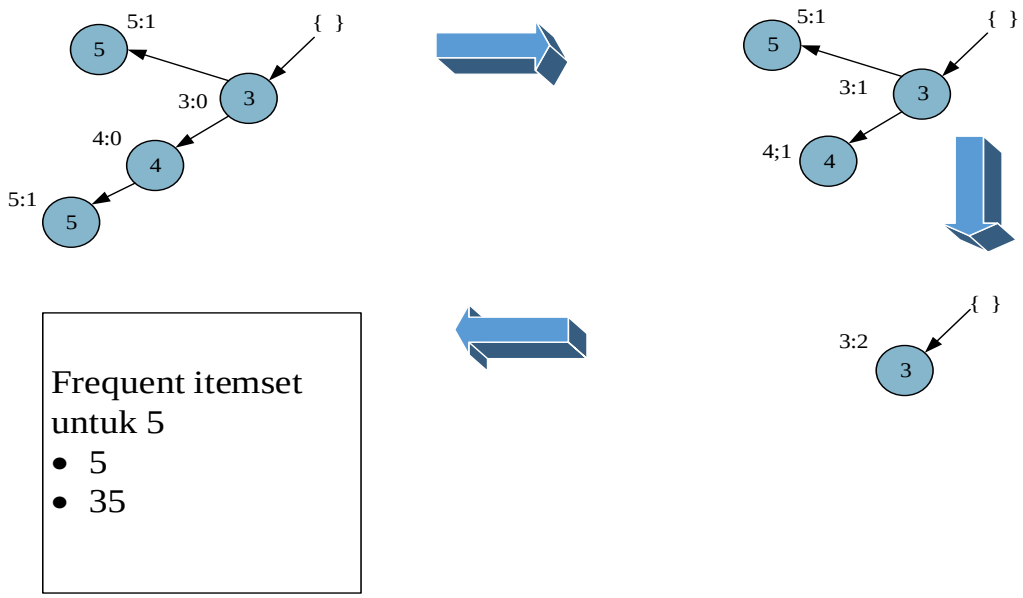
Ketiga tahap tersebut merupakan langkah yang akan dilakukan untuk mendapat *frequent itemset*, yang dapat dilihat pada algoritma berikut (Erwin, 2009):

```
01: if Tree mengandung single path P;
02: then untuk tiap kombinasi (dinotasikan
β)
dari node-node dalam path do
03: bangkitkan pola β α dengan supportt
dari
node-node dalam β;
04: else untuk tiap a1 dalam header dari
Tree
do
{
05: bangkitkan pola
06: bangun β = a1 α dengan supportt = a1.
supportt
07: if Tree β = Θ
08: then panggil FP-Growth (Tree, β)
}
}
```

Dari contoh kasus pada tabel 2.2, akan dicari semua *subsets* yang memungkinkan dengan cara membangkitkan *conditional FP-Tree* dan mencari *frequent itemset*. Membangkitkan *conditional FP-Tree* dilakukan berurut sesuai F-*list*, hanya saja untuk membangkitkan *conditional FP-Tree* dilakukan dari bawah ke atas atau dari item yang jumlah frekuensi kemunculannya terkecil.

1. Misalkan kondisi *FP-Tree* untuk 5

   Pertama-tama, ekstrak semua lintasaan yang berakhiran 5. Selain *path* 5, nol kan semua nilai *path*. Untuk lebih memperjelas, dapat dilihat dibawah ini contoh menemukan *frequent itemset* yang berakhiran dengan item 5.

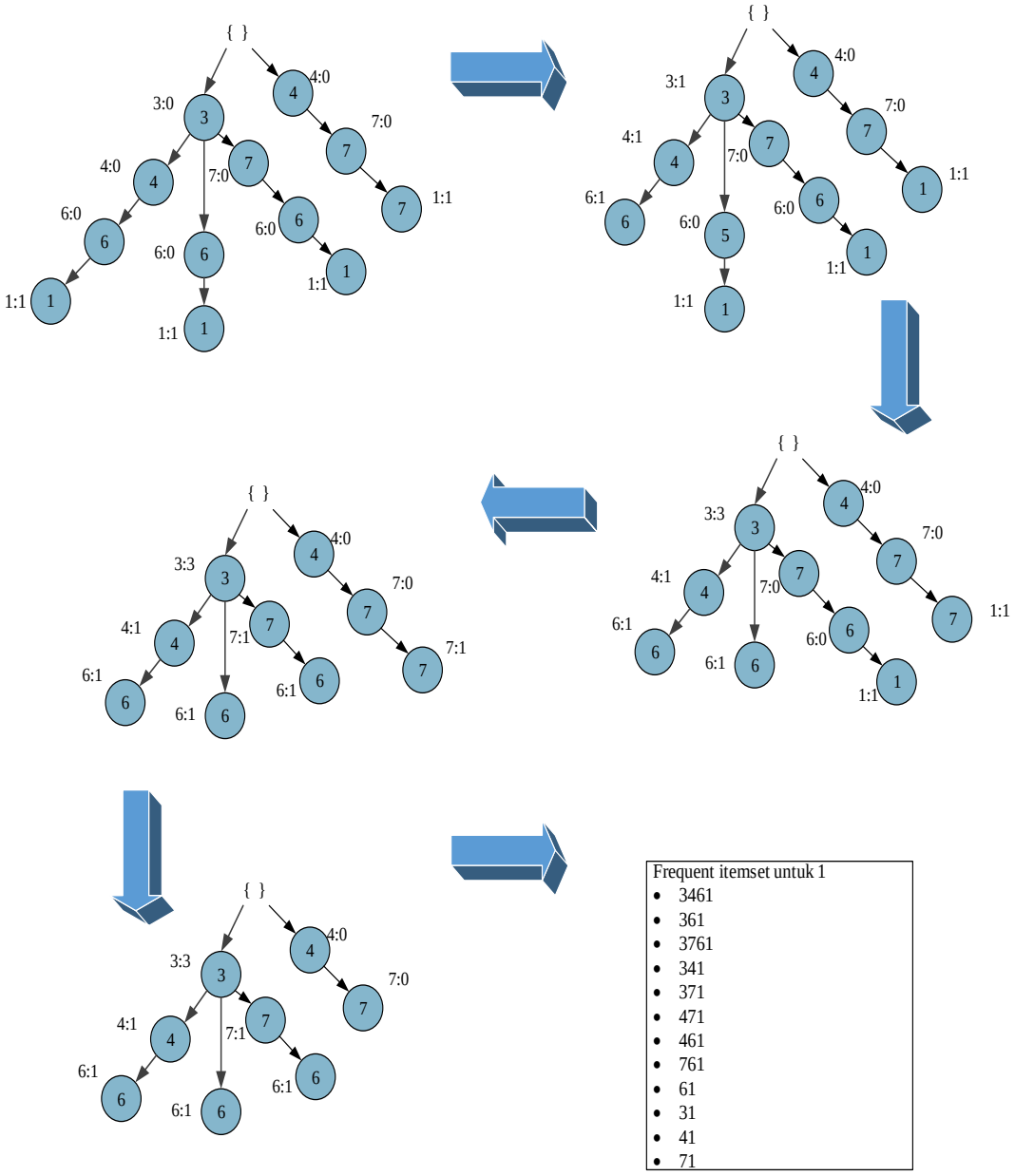



**Gambar 2.7 Kondisi *FP-Tree* Untuk 5 *(*Surojo, 2012)**

Setelah itu, buang satu persatu path 5 dan nilai path 5 dimasukan ke setiap path yang dilintasi dari path 5 sampai ke root. Pada kasus ini, 4 : 1 tidak frequent karena nilai support kurang dari 2 sehingga 4 dibuang. Setelah itu, bangun semua subsets dari {3}. Hasilnya, terdapat 2 subsets yang memungkinkan untuk 5 yang dapat dilihat pada gambar 2.7.

2. Misalkan kondisi *FP-Tree* untuk 1

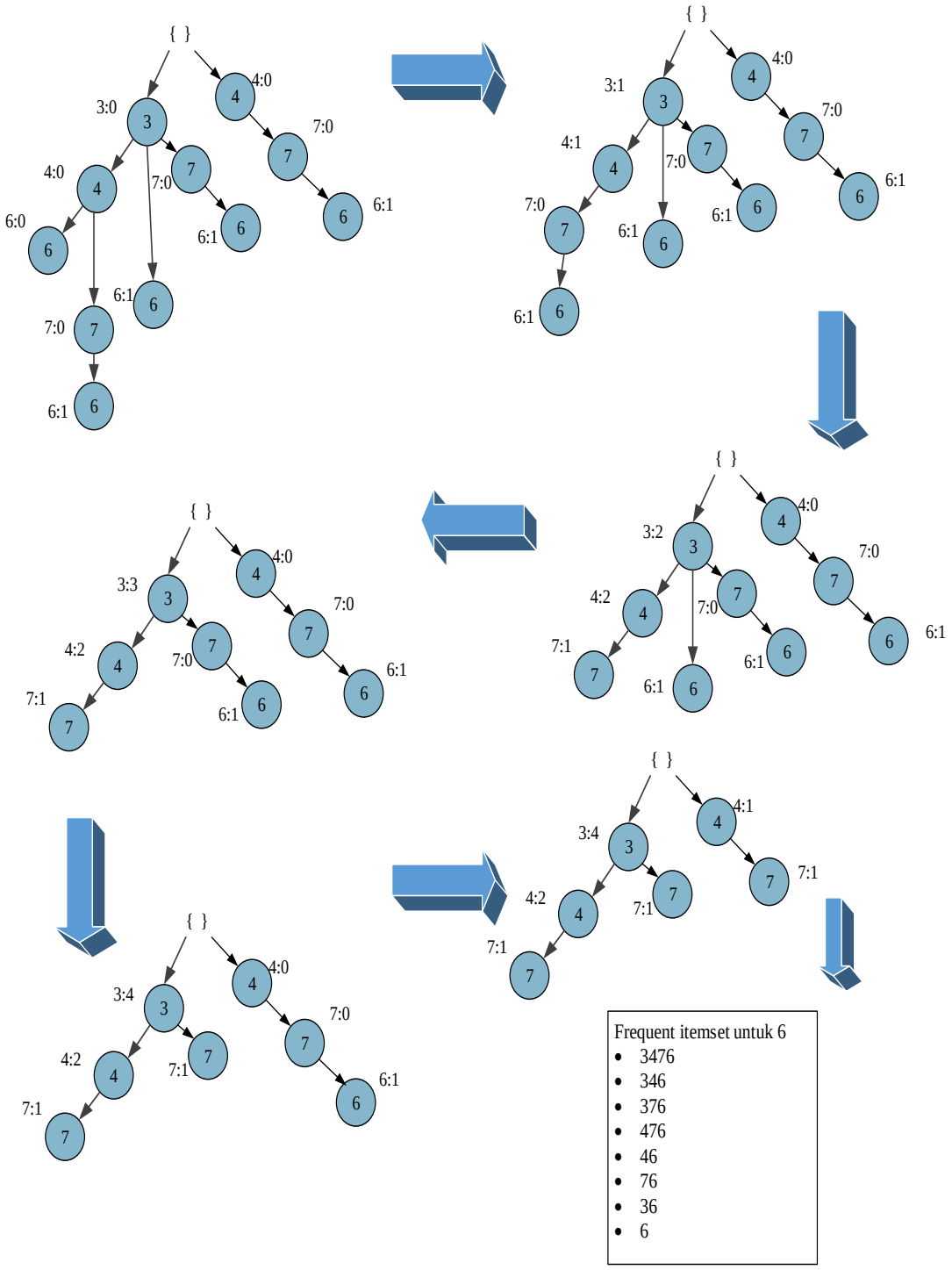
Ekstrak semua lintasaan yang berakhiran 1. Selain path 1, nol kan semua nilai path. Untuk lebih memperjelas, dapat dilihat dibawah ini contoh menemukan frequent itemset yang berakhiran dengan item 1




**Gambar 2.8 Kondisi *FP-Tree* Untuk 1 *(*Surojo, 2012)**

3. Kondisi *FP-Tree* untuk 6

Ekstrak semua lintasaan yang berakhiran 6. Selain *path* 6, nol kan semua nilai *path*. Untuk lebih memperjelas, dapat dilihat dibawah ini contoh menemukan *frequent itemset* yang berakhiran dengan item 6.




**Gambar 2.9 Kondisi *FP-Tree* Untuk 6 (Surojo, 2012)**

Setelah memeriksa semua *frequent itemset*, didapat 30 *possible subsets* yang hasilnya dirangkum dalam tabel berikut:

Tabel 2.6 Hasil *Frequent itemset* (Surojo, 2012).

| *Suffix* | *Frequent itemset* |
|---|---|
| 5 | {3,5}; {5} |
| 1 | {3,4,6,1};{3,6,1};{3,7,6,1};{3,4,1};{3,7,1};{4,7,1};{4,6,1};{7,6,1};{6,1};{3,1};{4,1};{7,1};{1} |
| 6 | {3,4,7,6},{3,4,6},{3,7,6},{4,7,6},{4,6},{7,6},{3,6},{6} |
| 7 | {3,4,7},{3,7},{4,7},{7} |
| 4 | {3,4},{4} |
| 3 | {3} |

**3. Pembuatan *Association Rule***

Dari 30 *possible subsets* yang hasilnya, tidak semua dihitung. Karena *rule* yang dihasilkan adalah jika kita membeli barang A, maka akan membeli barang B, maka *subsets* yang dihitung minimal berisi dua item. Maka yang akan dihitung *confidence*-nya adalah 24 *subsets*, yaitu : {3,4,6,1}, {3,6,1}, {3,7,6,1}, {3,4,1}, {3,7,1}, {4,7,1}, {4,6,1}, {7,6,1}, {3,5}, {6,1}, {3,1}, {4,1}, {7,1}, {3,4,7,6},{3,4,6}, {3,7,6}, {4,7,6}, {4,6}, {7,6}, {3,6}, {3,4,7}, {3,7}, {4,7}, {3,4}. Setelah didapatkan *frequent itemset*, selanjutnya adalah membuat *rule* dengan cara menghitung *confidence*-nya. Hanya kombinasi yang lebih besar sama dengan minimum *confidence* yang akan diambil atau *strong association rule*-nya saja. Adapun rumus menghitung *confidence* adalah:

$$\textbf{Confidence } (A \rightarrow B) = P(B|A) = \frac{\textbf{Support\_Count}(A \cup B)}{\textbf{Suppor\_Count}(A)} \qquad (4)$$

**Gambar 2.10 Rumus Menghitung *Confidence (*Surojo, 2012)**

Karena perhitungan yang sangat banyak, pada jurnal Erwin, 2009 mengambil contoh *itemset* {3,4,7} sebagai percontohan perhitungan *confidence*-nya adalah:

- 7→3∧4 =3/6 = 50 %
- 3→4∧7 =3/8 = 37 %
- 4→3∧7 =3/7 = 42,8 %
- 7→4= 5/6 =83,3%
- 4→7 =5/7 =71,4%
- 3→4 =5/8 =62,5 %
- 4→3 =5/7 =71,4 %
- 3→7 =4/8 = 50%
- 7→3 =4/6 =66,7 %

Karena minimum *confidence* adalah 75 %, maka yang termasuk *strong association rule* adalah 7→4 yang artinya jika konsumen membeli barang 7 dan 3, konsumen membeli barang 4 juga dan jika konsumen membeli barang 7, konsumen membeli barang 4 juga . Berikut hasil lengkap pola-pola atu *rules* yang dihasilkan. Tabel 2.7 Hasil *Rule* Yang Didapat Dari 24 *Subsets* Di Atas *(*Surojo, 2012).

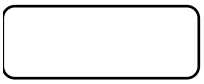
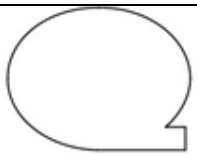
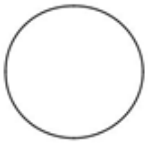
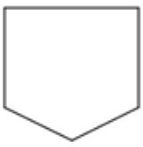
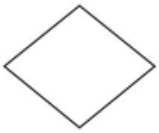
| *Rule* | *Confidence* |
|---|---|
| 1→3 | 75% |
| 3^4^1→6 | 100% |
| 4^6^1→3 | 100% |
| 5→3 | 100% |
| 1→3^6 | 75% |
| 3^1→6 | 100% |
| 6^1→3 | 100% |
| 3^6→1 | 75% |
| 7^6^1→3 | 100% |
| 1→6 | 75% |
| 3^7^1→6 | 100% |
| 6→3 | 80% |
| 7→4 | 83% |

### 2.3.4 Flow Chart (Diagram Alir)

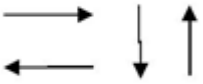
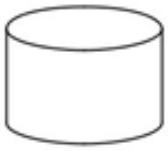
(Indrajani, 2015), "Flow chart adalah penggambaran secara grafik dari langkah-langkah dan urutan prosedur suatu program."

(Indrajani, 2015), menjelaskan simbol-simbol dalam Flow Chart dapat disajikan pada tabel 2.8.

Tabel 2.8 Simbol-simbol dalam *Flow Chart* (Diagram Alir)

| NO. | NAMA SIMBOL | SIMBOL | KETERANGAN |
|---|---|---|---|
| 1. | *Start* atau *End* |  | Mendefinisikan awal atau akhir dari sebuah *flowchart.* |
| 2 | Proses |  | Simbol pemprosesan yang terjadi pada sebuah alur kerja. |
| 3. | Sub Program |  | Simbol yang menyatakan bagian dari program (sub program). |
| 4. | *Input/Output* |  | Simbol masukan atau keluaran dari atau ke sebuah pita *magnetic*. |
| 5. | *Input/Output* |  | Mendefinisikan masukan dan keluaran proses |
| 6. | Penghubung |  | Untuk menyambung proses pada lembar kerja yang sama |
| 7. | Penghubung |  | Untuk menyambung proses pada lembar kerja yang berbeda. |
| 8. | Dokumen |  | Simbol masukan atau keluaran dari atau ke sebuah dokumen. |
| 9. | Kondisi |  | Simbol untuk memutuskan proses lanjutan dari kondisi tertentu. |

Lanjutan Tabel 2.8 Simbol-simbol dalam *Flow Chart* (Diagram Alir)

| NO. | NAMA SIMBOL | SIMBOL | KETERANGAN |
|---|---|---|---|
| 10. | Output |  | Simbol yang menyatakan piranti keluaran, seperti layar monitor, *printer*, dll. |
| 11. | Proses Manual |  | Simbol yang mendefinisikan proses yang dilakukan secara manual. |
| 12 | Penghubung Antar Proses |  | Simbol untuk menghubungkan antar proses atau antar simbol |
| 13. | Database |  | Simbol database atau basis data. |

### 2.3.5 Analisa Konsistensi Pola

Pada tahap ini, dilakukan analisa terhadap pola yang dihasilkan. Pertama dicari pola yang terbentuk di tiap bulan, semester, dan tahun dengan cara menghitung nilai *support* , *confidence* , dan *lift* setiap pola. Kemudian dilakukan analisa konsistensi pola yang dihasilkan perbulan, dan tahun. Sebuah pola/*rule* dianggap menarik (*interesting*) jika memenuhi syarat : nilai *support* pola ≥ min_sup, nilai *confidence* pola ≥ min_conf, dan nilai *lift* pola > 1 (*possitive correlation*). Sebuah pola dikatakan konsisten jika pola tersebut sering muncul pada bulan–bulan dalam bulan ataupun tahun. Semakin sering sebuah pola muncul dalam rentang bulan atau tahun tertentu maka pola tersebut dapat dikatakan sebagai pola yang konsisten. Untuk mengetahui apakah sebuah pola memenuhi syarat sebagai pola yang konsisten, diberikan sebuah parameter konsistensi. Parameter konsistensi didefinisikan sebagai frekuensi kemunculan minimal yang harus dipenuhi oleh sebuah pola untuk dikatakan sebagai pola yang konsisten. Parameter

konsistensi berfungsi sebagai syarat nilai minimal sebuah pola dikatakan konsisten. Dengan mengetahui pola yang konsisten pada periode tertentu dapat diketahui karakteristik peminjam buku pada periode tersebut. Karakteristik peminjam buku pada periode tersebut dapat dilihat dari frekuensi kemunculan pola. Semakin besar frekuensi kemunculan pola menandakan bahwa pola tersebut semakin konsisten dan semakin menunjukkan karakteristik peminjam pada periode tersebut (Miranda N.Q.A, 2015).

## 2.4 Teori Umum

### 2.4.1 Aplikasi *Weka*

Weka adalah kumpulan algoritma pembelajaran mesin untuk tugas-tugas penambangan data. Algoritme dapat diterapkan langsung ke kumpulan data atau dipanggil dari kode Java Anda sendiri. Weka berisi alat untuk pra-pemrosesan data, klasifikasi, regresi, pengelompokan, aturan asosiasi, dan visualisasi. Ini juga sangat cocok untuk mengembangkan skema pembelajaran mesin baru (Waikato, 2018).

### 2.4.2 Perpustakaan

Kata perpustakaan berasal dari kata pustaka, yang berarti: (1) kitab, buku-buku, (2) kitab primbon. Kemudian kata pustaka mendapat awalan per dan akhiran an, menjadi perpustakaan. Perpustakaan mengandung arti: (1) kumpulan buku-buku bacaan, (2) bibliotek, dan (3) buku-buku kesusastraan (Kamus Besar Bahasa Indonesia-KBBI). Selanjutnya ada pula istilah pustakaloka yang berarti tempat atau ruangan perpustakaan. Pengertian perpustakaan yaitu mencakup suatu ruangan, bagian dari gedung/bangunan, atau gedung tersendiri, yang berisi bukubuku koleksi, yang disusun dan diatur sedemikian rupa, sehingga mudah untuk dicari dan dipergunakan. Perpustakaan dilengkapi dengan berbagai sarana prasarana, seperti ruangan baca, rak buku, rak majalah, meja kursi baca kartukartu katalog, system pengelolaan tertentu, dan ditempatkan karyawan atau pustakawan yang melaksanakan kegiatan perpustakaan.

Sedangkan menurut (Sutarno, 2003), ”Perpustakaan adalah suatu ruangan, bagian dari gedung/bangunan, atau gedung tersendiri, yang berisi buku-buku

koleksi, yang disusun dan diatur sedemikian rupa, sehingga mudah untuk dicari dan dipergunakan sewaktu-waktu diperlukan oleh pembaca".

Dari definisi di atas dapat disimpulkan bahwa Perpustakaan adalah suatu unit kerja dari suatu badan yang mengelola bahan pustaka, baik berupa buku maupun bukan buku yang disusun secara sistematis menurut aturan tertentu sehingga dapat digunakan sebagai sumber informasi oleh setiap pengguna perpustakaan

### 2.4.3 Pengertian Java

Java dikembangkan oleh perusahaan Sun Microsystem. Java menurut definisi dari Sun Microsystem adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer standalone ataupun pada lingkungan jaringan. Java 2 adalah generasi kedua dari java platform (Rosa & M, 2014).

### 2.4.4 NeatBeans

NetBeans merupakan salah satu proyek open source yang disponsori oleh Sun Microsystem. Proyek ini berdiri pada tahun 2000 dan telah menghasilkan 2 produk, yaitu NetBeans IDE dan NetBeans Platform. NetBeans IDE merupakan produk yang digunakan untuk melakukan pemrograman baik menulis kode, meng-compile, mencari kesalahan dan mendistribusikan program. Sedangkan NetBeans platform adalah sebuah modul yang merupakan kerangka awal/pondasi dalam membangun aplikasi desktop yang besar (Wahana Komputer, 2010).