BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisis sistem adalah penguraian dari suatu informasi yang utuh kedalam bagian-bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan, kesempatan, hambatan yang terjadi dan kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan. Analisis Sistem ini bertujuan untuk menjelaskan permasalahan yang muncul pada saat pembangunan sistem, hal ini bertujuan untuk membantu ketika proses perancangan sistem berlangsung. Tugas utama analisis sistem dalam tahap ini adalah menemukan kelemahan-kelemahan dari sistem yang berjalan sehingga dapat diusulkan perbaikannya. Dalam analisis sistem ini meliputi beberapa bagian, yaitu :

1. Analisis Masalah
2. Analisis Penerepan Metode *CRISP-DM*
3. Analisis Spesifikasi Kebutuhan Perangkat Lunak

3.1.1 Analisis Masalah

Analisis masalah yang dilakukan adalah melihat dan mengidentifikasi permasalahan atau kendala di dalam penelitian yang dilakukan. Masalah dapat diartikan sebagai suatu pernyataan yang menyatakan tentang situasi yang memerlukan pemecahan. Perlu ada analisis masalah melalui rumusan masalah yang sudah ditentukan sebelumnya. Berdasarkan hasil pengamatan, diketahui bahwa analisis masalahnya adalah sebagai berikut :

1. Kartin Collection masih mengalami masalah dalam penentuan model tas yang akan diproduksi
2. Belum adanya perekomendasian untuk menentukan model tas yang harus diproduksi.

3.1.2 Analisis Penerapan Metode *CRISP-DM*

Metode pembangunan perangkat *data mining* yang digunakan dalam penelitian ini adalah *Cross-Industry Standard Process for Data Mining (CRISP-DM).*

3.1.2.1 Pemahaman Bisnis

Tahapan pemahaman bisnis merupakan tahapan pertama dilakukan dalam kerangka kerja *CRISP-DM.* Dalam tahapan bisnis ini terdapat beberapa tahapan lainnya, yaitu :

1. Tujuan Bisnis

   Dalam proses bisnisnya Kartin Collection mempunyai tujuan bisnis yaitu untuk meningkatkan mutu kualitas tas sesuai dengan permintaan dari pelanggan tetap.

2. Penentuan Sasaran *Data Mining*

   Tujuan dari penerapan *data mining* nya adalah untuk mementukan model tas apa saya yang akan diproduksi, sesuai dengan permintaan pelangan tetap atau pelanggan lainnya.

3.1.2.2 Pemahaman *Data*

Dalam penelitian ini adalah *data* transaksi yang terjadi di Kartin Collection yang berasal dari *file microsoft excel* (.xlsx). Berikut merupakan penjelasan dari masing-masing atribut yang terdapat pada tabel *Data* transaksi yang dapat dilihat pada tabel 3.1 dibawah ini :

Tabel 3. 1 Struktur *Data* Transaksi

Keterangan		
Fungsi	Untuk mengetahui jenis produk –produk yang paling sering terjual oleh customer	
Format	*Microsoft Excel (.xlsx)*	
Atribut	No	No urut
	Tgl	Tanggal proses pembelian
	Lokasi	Tempat pembelian
	Tuan/Toko	Nama pemilik yang melayani
	No Transaksi	Nomor pembelian
	Banyaknya	Banyaknya barang yang dibeli
	Kode Barang	Kode dari setiap barang
	Nama Barang	Nama barang yang dibeli
	Hitam	Warna hitam dari setiap nama barang
	Abu	Warna abu dari setiap nama barang
	Cream	Warna cream dari setiap nama barang
	Merah	Warna merah dari setiap nama barang
	Coklat	Warna coklat dari setiap nama barang
	Denim Hitam	Warna denim hitam dari setiap nama barang
	Denim Biru	Warna biru dari setiap nama barang
	Harga	Harga barang yang dibeli
	Jumlah	Jumlah barang yang dibeli
	Return	Barang yang dikembalikan
	Ket	Alasan dari pengembalian barang

3.1.2.3 Persiapan *Data*

Sebelum dilakukan proses *data mining* harus dipersiapkan karena tidak semua *data* atau atribut dalam *data* digunakan pada proses *mining*. Proses ini dilakukan agar *data* dapat digunakan sesuai kebutuhan. Adapun tahapan – tahapan persiapan *data* adalah sebagai berikut :

1. *Ekstraksi Data*

   Ekstraksi Data adalah proses pengambilan *data* transaksi penjualan dikartin collection dari sumber *data* yang berasal dari *file* berformat *microsoft excel*. *Data* yang telah diekstrak akan disimpan kedalam *database,* untuk melanjutkan proses pengolahan *data* ke tahap selanjutnya. Berikut adalah *data* transaksi penjualan yang terjadi di kartin collection dapat dilihat pada lampiran E.

2. Pemilihan atribut

   Berdasarkan imformasi yang ingin didapat oleh pengguna mengenai model produk yang dibeli secara bersamaan, maka atribut yang digunakan dalam *data* transaksi diatas adalah No Transaksi dan Kode Barang. No transaksi adalah ID dari transaksi dan atribuit Kode Barang adalah jenis barang yang dibeli. Ini bertujuan untuk proses *mining*, karena tidak semua *data* yang digunakan dalam *data* transaksi digunakan untuk proses *mining.* Berikut dibawah ini table proses pemilihan atribut dapat dilihat pada tabel 3.2 :

Tabel 3. 2Pemilihan atribut

No Transaksi	KodeBarang
1	14
1	9
2	9
2	2
2	14
3	9
3	14
3	2
3	1
4	12
4	7

No Transaksi	KodeBarang
4	5
4	15
4	10
4	2
4	1
5	14
5	5
5	9
5	1
5	7
5	12
5	5
5	15
6	14
6	9
6	5
6	2
6	10
6	3
6	7
6	15
7	9
7	14
7	2
8	9
8	14
9	2
9	1
10	12
10	7
11	2
11	1
11	14
12	5
12	9
13	1
14	7
14	12
14	5
15	14

No Transaksi	KodeBarang
15	13
15	7
15	15
16	14
16	2
16	1
17	12
17	7
18	15
18	10
18	8
19	11
19	9
19	14
19	2
20	1
20	12
20	7
20	5
20	15
20	14
20	6
20	9
21	1
21	2
21	14
21	15
22	14
22	15
22	4
22	2
23	1
23	14
23	12
23	15
23	7
24	14
24	9
24	16
24	6

No Transaksi	KodeBarang
25	9
25	1
25	2
26	14
26	15
26	12
26	7

3. Pembersihan *Data*

Pembersihan *Data* merupakan proses menghilangkan *data* dari hasil pemilihan atribut pada tabel 3.2. *Data* yang dihilangkan adalah *ata* pada satu transaksi yang hanya terdapat satu pembelian model tas, karena *data* tersebut tidak memiliki hubungan *assosiasi* dengan *Item* lain. *data* yang telah dibersihkan dapat dilihat pada tabel 3.3 :

Tabel 3. 3 Hasil Pembersihan *Data*

No Transaksi	KodeBarang
1	14
1	9
2	9
2	2
2	14
3	9
3	14
3	2
3	1
4	12
4	7
4	5
4	15
4	10
4	2
4	1
5	14
5	5
5	9
5	1

No Transaksi	KodeBarang
5	7
5	12
5	5
5	15
6	14
6	9
6	5
6	2
6	10
6	3
6	7
6	15
7	9
7	14
7	2
8	9
8	14
9	2
9	1
10	12
10	7
11	2
11	1
11	14
12	5
12	9
14	7
14	12
14	5
15	14
15	13
15	7
15	15
16	14
16	2
16	1
17	12
17	7
18	15
18	10

No Transaksi	KodeBarang
18	8
19	11
19	9
19	14
19	2
20	1
20	12
20	7
20	5
20	15
20	14
20	6
20	9
21	1
21	2
21	14
21	15
22	14
22	15
22	4
22	2
23	1
23	14
23	12
23	15
23	7
24	14
24	9
24	16
24	6
25	9
25	1
25	2
26	14
26	15
26	12
26	7

4. Penyiapan *Data* awal

   Setelah proses semua dilakukan dan *data* telah siap di *mining*, maka *data* transaksi sudah dapat digunakan proses selanjutnya yaitu pemodelan.

3.1.2.4 Pemodelan

Pemodelan merupakan tahap untuk membuat model atau desain sistem yang akan dibangun. *Data* yang digunakan untuk pengolahan *data* ini adalah *data* yang terdapat dalam table 3.3. Didalam tahap ini metode yang dipakai adalah metode *association rule* dengan algoritma *fp-growth*.

Metode *assosiation rules* terbagi menjadi dua tahap yaitu mencari pola frekuensi dan pembentukan atau perhitungan aturan *asosiasi*. Pencarian frekuensi adalah tahap mencari nilai frekuensi dari setiap *Item*, sedangkan pembentukan *asosiasi* adalah proses perhitungan untuk mencari pola *Item* yang memenuhi nilai *minimum support*.

Nilai *minimum support* yang digunakan adalah 10% dari 26 sampel *data* transaksi yaitu 2.6≈ 3. yang bertujuan untuk melihat banyaknya variasi *data* yang terjadi dalam proses pencarian hasil akhir yaitu *rule.*

Berikut adalah tahap pencarian frekuensi dari setiap *Item* yaitu menuliskan setiap kode barang dan memberikan nilai frekuensi dengan cara menghitung kemunculan setiap barang. Hasil pencarian frekuensi dapat dilihat pada tabel 3.4 :

Tabel 3. 4Perhitungan Frekuensi

Kode Barang	Frekuensi
1	10
2	11
3	1
4	1
5	6
6	2

Kode Barang	Frekuensi
7	9
8	2
9	12
10	3
11	1
12	8
14	18
15	10
16	1

Adapun tahapan dalam proses pencarian selanjutnya adalah sebagai berikut :

1. Langkah pertama menghilangkan d*ata* yang tidak memenuhi *minimum support* yang sudah ditentukan sebelumnya yaitu 3, berikut *data* yang memenuhi nilai *minimum support* dapat dilihat pada tabel 3.5 :

Tabel 3. 5 *Data* Yang Memenuhi *Minimum Support*

Kode Barang	*Support count*
1	10
2	11
5	6
7	9
9	12
10	3
12	8
14	18
15	10

2. Langkah kedua adalah memberikan nilai *priority* untuk setiap *Item*. Setelah mendapatkan *data* yang memenuhi nilai *minimum support* seperti pada tabel 3.5 diatas, selanjutnya melakukan proses pengurutan secara as*cending* dari kemunculan setiap *support count* yang muncul dan memberikan *priority* sebagai tanda dari setiap *Item* yang memiliki *support count* paling besar sampai dengan terkecil. Jika terdapat kode barang dengan *support count* yang sama maka yang akan didahulukan adalah kode barang yang kemunculannya lebih awal pada *data* transaksi dibandingkan yang lainnya. Berikut adalah hasil pengurutan berdasarkan *priority*dapat dilihat pada tabel 3.6 :

Tabel 3. 6 Pengurutan Secara *Ascending*

Kode Barang	*Support Count*	*Priority*
14	18	1
9	12	2
2	11	3
1	10	4
15	10	5
7	9	6
12	8	7
5	6	8
10	3	9

3. Langkah ketiga setelah semua *Item* mendapatkan nilai *priority* maka *data* transaksi yang memenihi *minimum support* akan diurutkan berdasarkan *priority*. Pada tahap ini *Item* yang memiliki nilai *priority* tertinggi pada satu transaksi akan didahulakan penulisannya. Hasil pengurutan berdasarkan *priority* dapat dilihat pada tabel 3.7:

Tabel 3. 7 Pengurutan *Data* Berdasarkan *Priority*

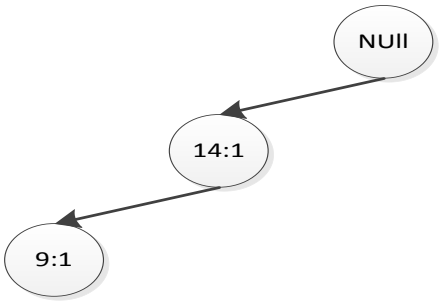
No Transaksi	Kode Barang
1	14
1	9
2	14
2	9
2	2
3	14
3	9
3	2
3	1
4	2
4	1
4	15
4	7
4	12
4	5
4	10
5	14
5	9
5	1
5	15
5	7
5	12
5	5
6	14
6	9
6	2
6	15
6	7

No Transaksi	Kode Barang
6	5
6	10
7	14
7	9
7	2
8	14
8	9
9	2
9	1
10	7
10	12
11	14
11	2
11	1
12	9
12	5
14	7
14	12
14	5
15	14
15	15
16	14
16	2
16	1
17	7
17	12
18	15
18	10
19	14

No Transaksi	Kode Barang
19	9
20	14
20	9
20	1
20	15
20	7
20	12
20	5
21	14
21	2
21	1
21	15
22	14
22	2
22	15
23	14
23	1
23	15
23	7
23	12
24	14
24	9
25	14
25	9
25	2
25	1
26	14
26	15
26	7

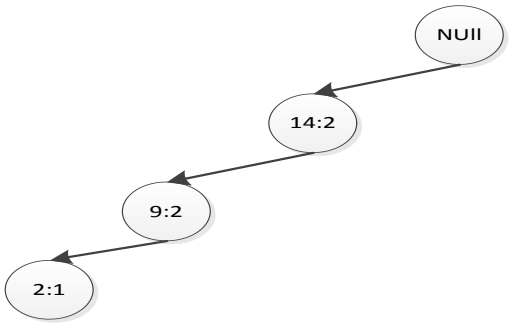
No Transaksi	Kode Barang
26	12

4. Langkah keempat adalah membuat *fp-tree*. Berikut proses pembentukan *FP-Tree* dibawah ini :
    1. Transaksi 1 : 14,9
        1) Lakukan pembacaan pada *data* transaksi pertama yang akan diproses.
        2) Lakukan pengecekan pada *root* yang diinisialisasikan dengan *Null*, jika *root* belum mempunyai anak (*child)*, maka d*ata* yang dibaca tadi menjadi anak pertama dari *root*. Dan *support count* dari masing masing *node* bertambah 1, kecuali *root*.



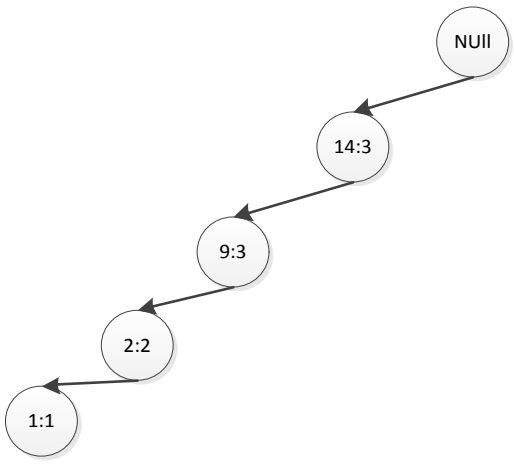
Gambar 3. 1 *Tree* No transaksi

    2. Transaksi 2 : 14,9,2
        1) Lakukan pembacaan pada *data* transaksi kedua yang akan diproses.
        2) Jika *data* yang keluar berikutnya sama dengan *data* yang pertama, maka itu bukan anak kedua dari *root*, tetapi tetap anak dari *root* yang sebelumnya keluar, yang berubah adalah *support count*-nya menjadi bertambah 1.



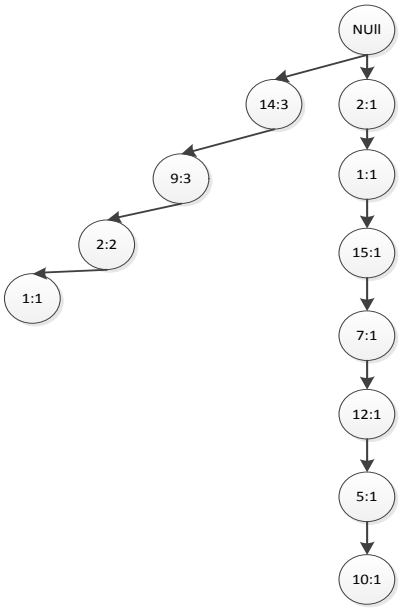
Gambar 3. 2 *Tree* No transaksi 2

3. Transaksi 3 : 14,9,2,1
   1) Lakukan pembacaan pada *data* transaksi ketiga yang akan diproses.
   2) Jika *data* yang keluar berikutnya sama dengan *data* yang pertama, maka itu bukan anak kedua dari *root*, tetapi tetap anak dari *root* yang sebelumnya keluar, yang berubah adalah *support count*-nya menjadi bertambah 1.



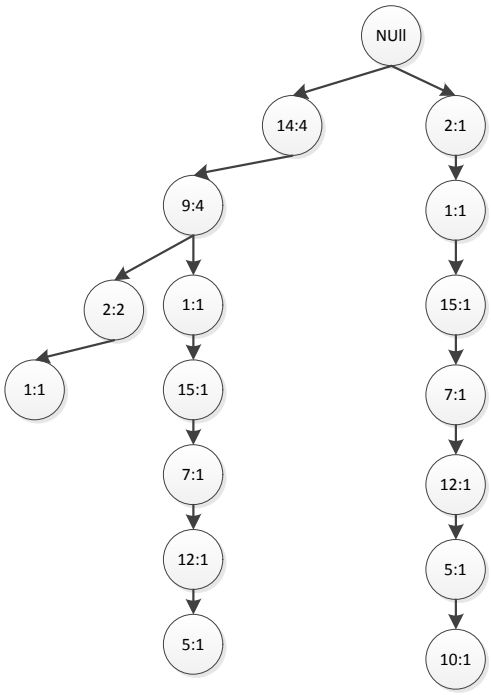
Gambar 3. 3 *Tree* No transaksi 3

4. Transaksi 4 : 2,1,15,7,12,5,10
   1) Lakukan pembacaan pada *data* transaksi keempat yang akan diproses.
   2) Jika *data* yang dibaca atau *data* yang keluar berbeda dengan *data* yang kelaur sebelumnya maka itu menjadi anak terakhir dari *root* dan menjadi sibling dari *node data* pertama. Dan *support count* dari *node* tersebut bertambah1.



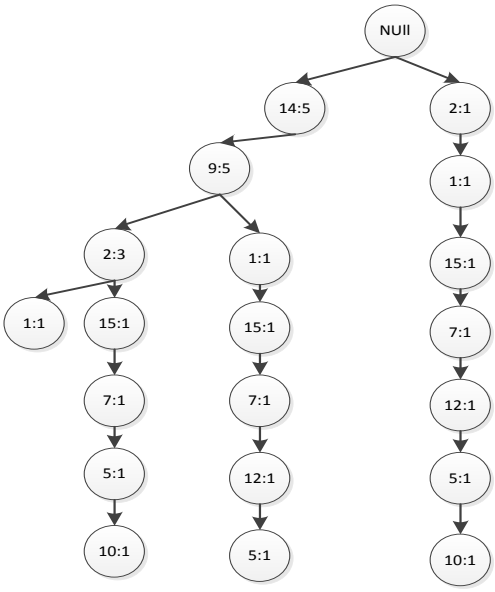
Gambar 3. 4 *Tree* No transaksi 4

5. Transaksi 5 : 14,9,1,15,7,12,5
   1) Lakukan pembacaan pada *data* transaksi kelima yang akan diproses.
   2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support* count-nya, hanya *data* yang dibaca sama saja yang bertambah *support count*-nya.



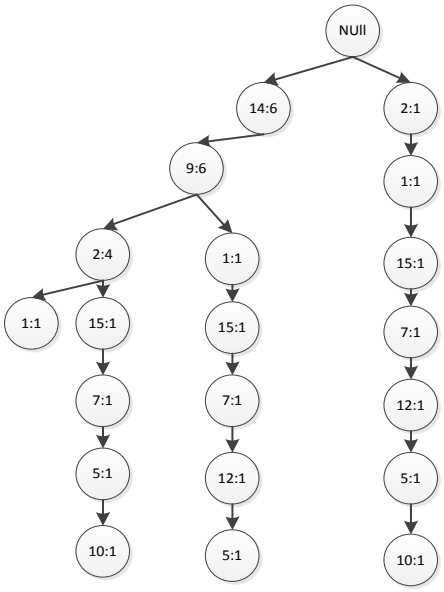
Gambar 3. 5 *Tree* No transaksi 5

6. Transaksi 6: 14,9,2,15,7,5,10
   1) Lakukan pembacaan pada *data* transaksi keenam yang akan diproses.
   2) Jika *Data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *data* yang dibaca sama saja yang bertambah *support count*-nya.



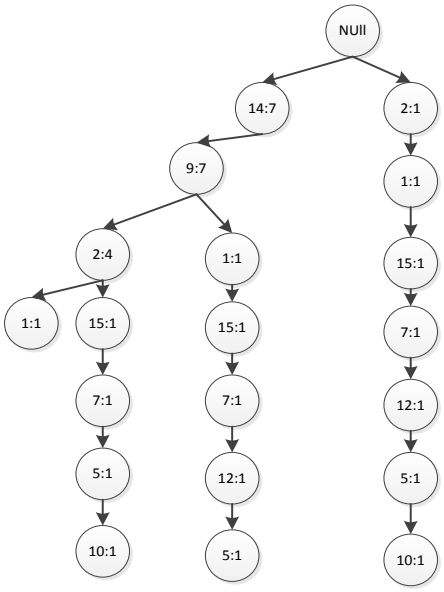
Gambar 3. 6 *Tree* No transaksi 6

7. Transaksi 7 : 14,9,2
    1) Lakukan pembacaan pada *data* transaksi ketujuh yang akan diproses.
    2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *data* yang dibaca sama saja yang bertambah *support count*-nya.



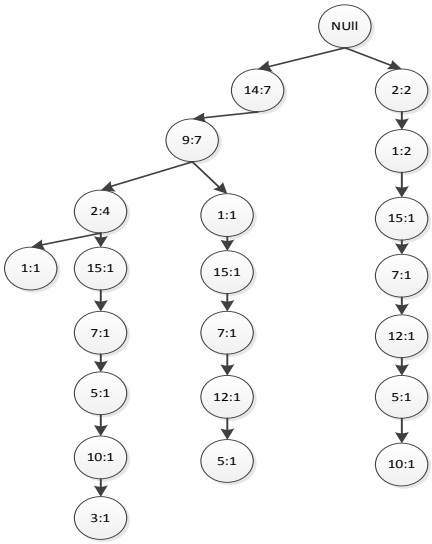
Gambar 3. 7 *Tree N*o transaksi 7

8. Transaksi 8 : 14,9
    1) Lakukan pembacaan pada *data* transaksi kedelapan yang akan diproses.
    2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *data* yang dibaca sama saja yang bertambah *support count*-nya.



Gambar 3. 8 *Tree* No transaksi 8

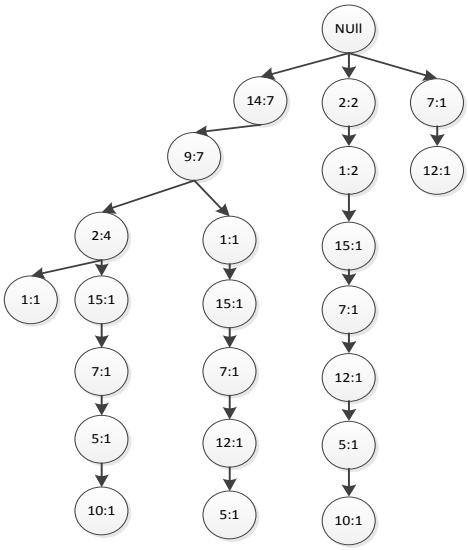
9. Transaksi 9 : 2,1
    1) Lakukan pembacaan pada *data* transaksi kesembilan yang akan diproses.
    2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *data* yang dibaca sama saja yang bertambah *support count*-nya.



Gambar 3. 9 *Tree* No transaksi 9

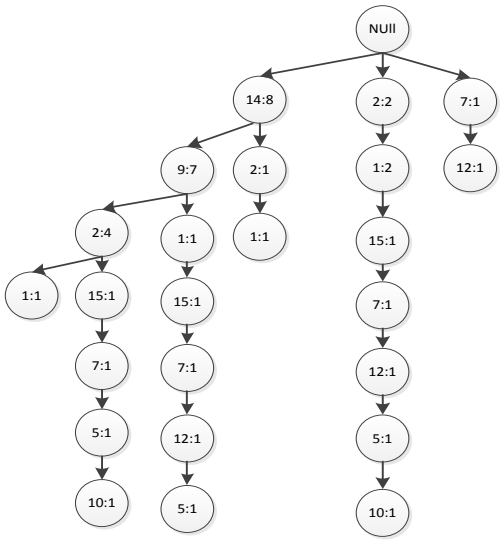
10. Transaksi 10 : 7,12

1) Lakukan pembacaan pada *data* transaksi kesepuluh yang akan diproses.
2) Jika *data* yang dibaca atau *data* yang keluar berbeda dengan *data* yang kelaur sebelumnya maka itu menjadi anak terakhir dari *root* dan menjadi *sibling* dari *node data* pertama. Dan *support count* dari *node* tersebut bertambah1.



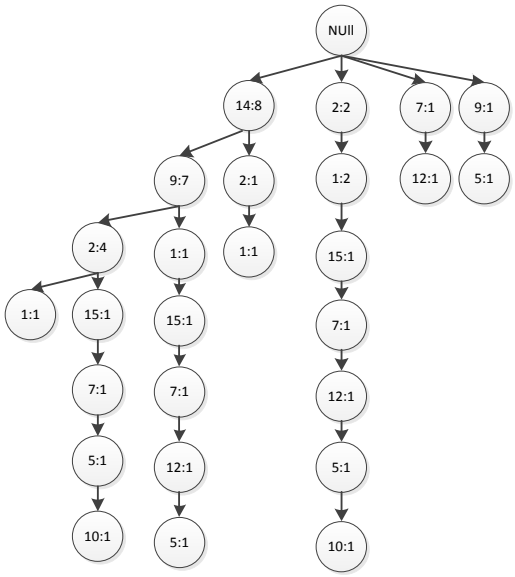
Gambar 3. 10 *Tree* No transaksi 10

11. Transaksi 11 : 14,2,1
    1) Lakukan pembacaan pada *data* transaksi kesebelas yang akan diproses.
    2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *data* yang dibaca sama saja yang bertambah *support count*-nya.



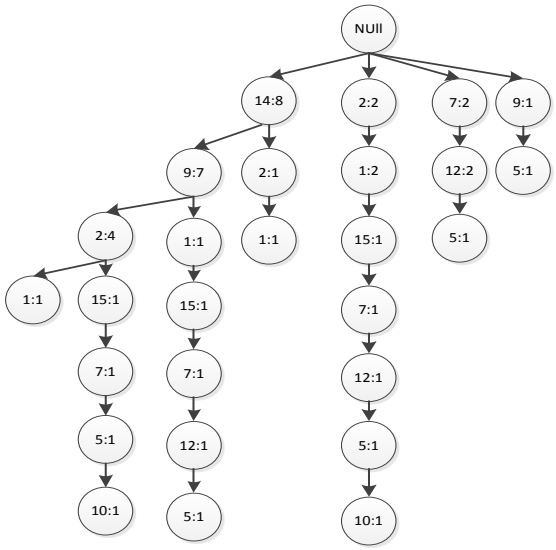
Gambar 3. 11 *Tree* No transaksi 11

12. Transaksi 12 : 9,5
    1) Lakukan pembacaan pada *data* transaksi kedua belas yang akan diproses.
    2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *data* yang dibaca sama saja yang bertambah *support count*-nya.



Gambar 3. 12 *Tree* No transaksi 12

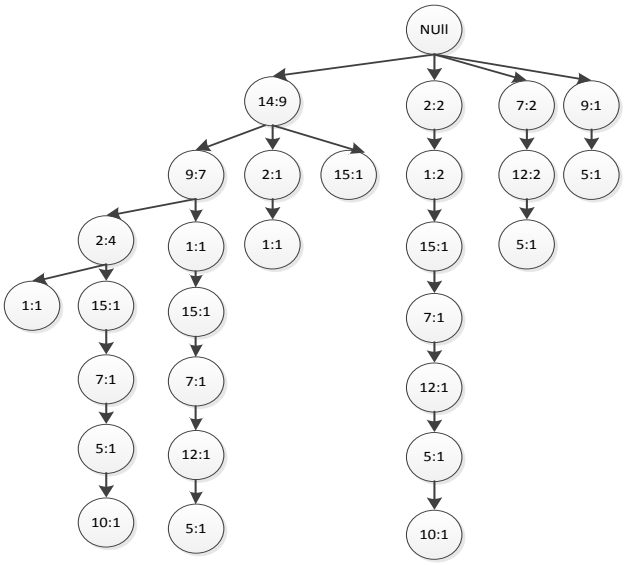
13. Transaksi 14 : 7,12,5
    1) Lakukan pembacaan pada *data* transaksi keempat belas yang akan diproses.
    2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *data* yang dibaca sama saja yang bertambah *support count*-nya.



Gambar 3. 13 *Tree* No transaksi 14

14. Transaksi 15 : 14,15

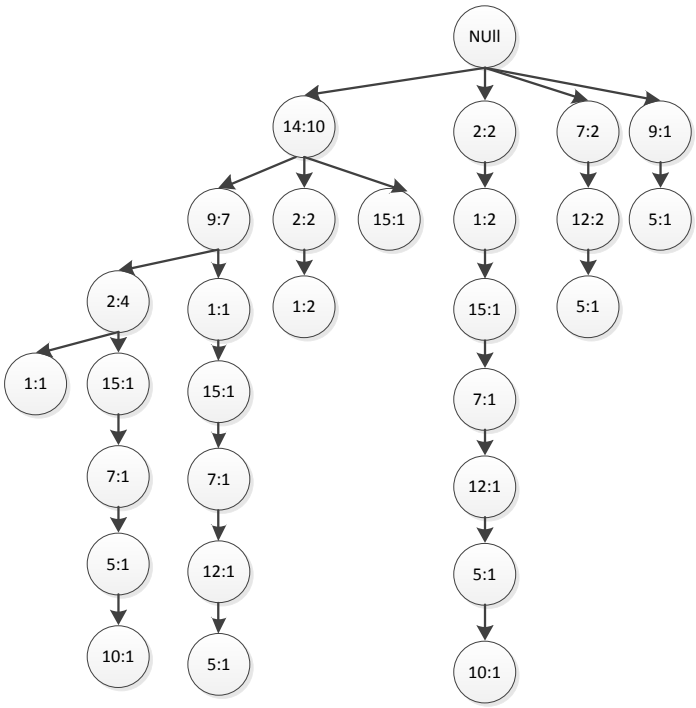
1) Lakukan pembacaan pada *data* transaksi kelima belas yang akan diproses.
2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *data* yang dibaca sama saja yang bertambah *support count*-nya.



Gambar 3. 14 *Tree* No transaksi 15

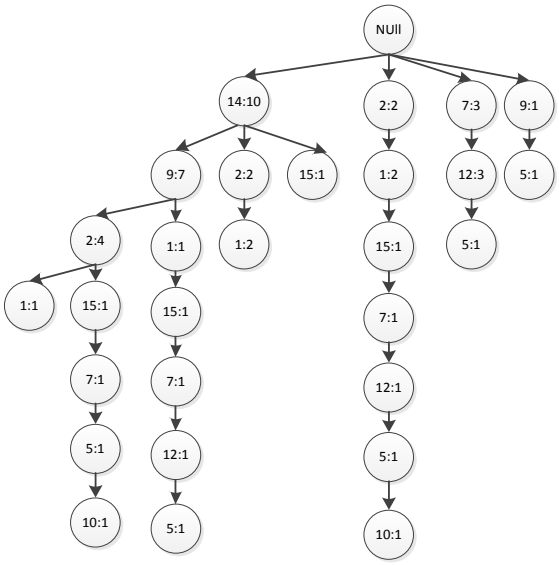
15. Transaksi 16 : 14,2,1

1) Lakukan pembacaan pada *data* transaksi keenam belas yang akan diproses.
2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *data* yang dibaca sama saja yang bertambah *support count*-nya.



Gambar 3. 15 *Tree* No transaksi 16

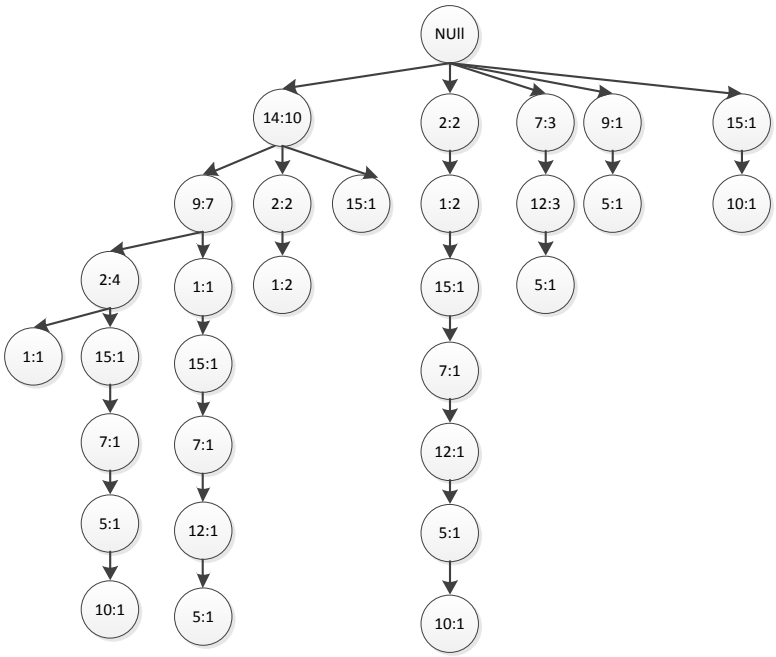
16. Transaksi 17 : 7,12
    1) Lakukan pembacaan pada *data* transaksi ketujuh belas yang akan diproses.
    2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *data* yang dibaca sama saja yang bertambah *support count*-nya.



Gambar 3. 16 *Tree* No transaksi 17

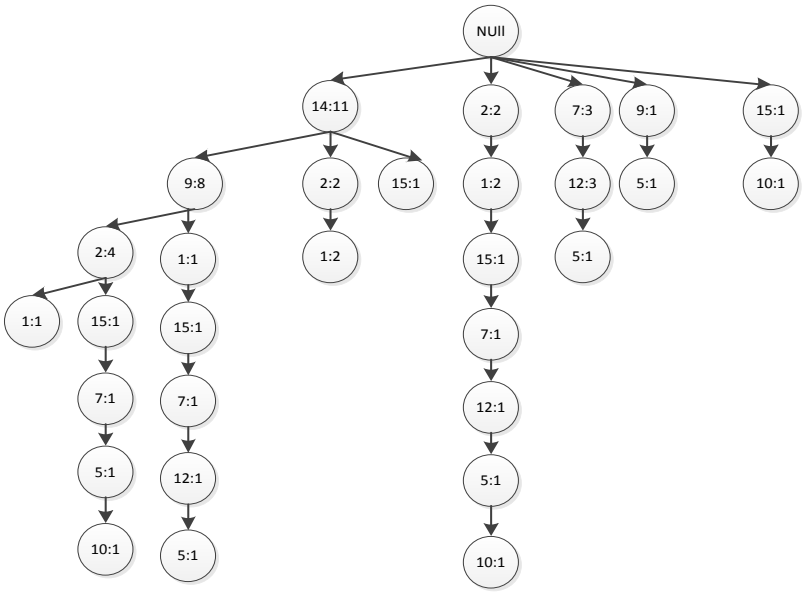
17. Transaksi 18 : 15,10

1) Lakukan pembacaan pada *data* transaksi kedelapan belas yang akan diproses.
2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *Data* yang dibaca sama saja yang bertambah *support count*-nya.



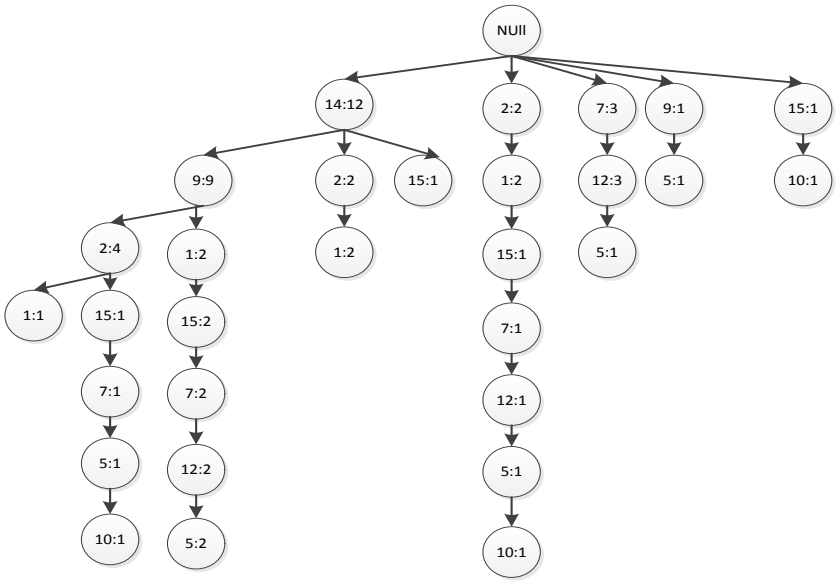
Gambar 3. 17 *Tree* No transaksi 18

18. Transaksi 19 : 14,9
    1) Lakukan pembacaan pada *data* transaksi kesembilan belas yang akan diproses.
    2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasanmaka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *data* yang dibaca sama saja yang bertambah *support count*-nya.



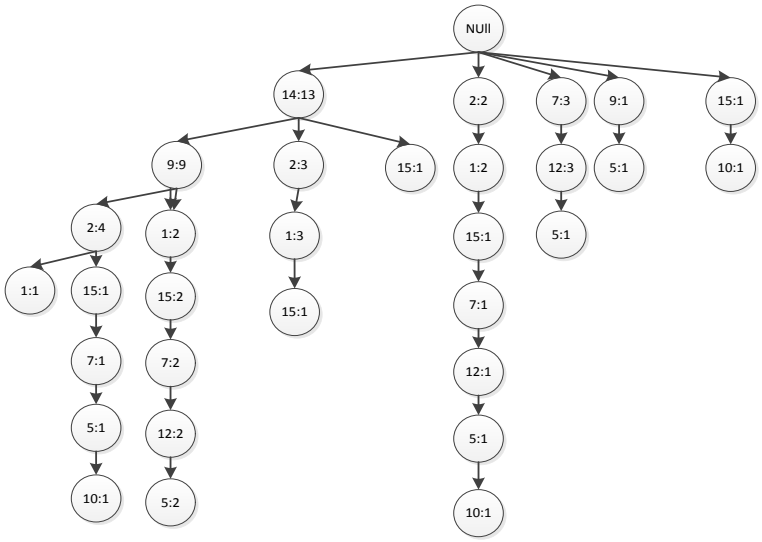
Gambar 3. 18 *Tree* No transaksi 19

19. Transaksi 20 : 14,9,1,15,7,12,5
    1) Lakukan pembacaan pada *data* transaksi kedua puluh yang akan diproses.
    2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *data* yang dibaca sama saja yang bertambah *support count*-nya.



Gambar 3. 19 *Tree* No transaksi 20

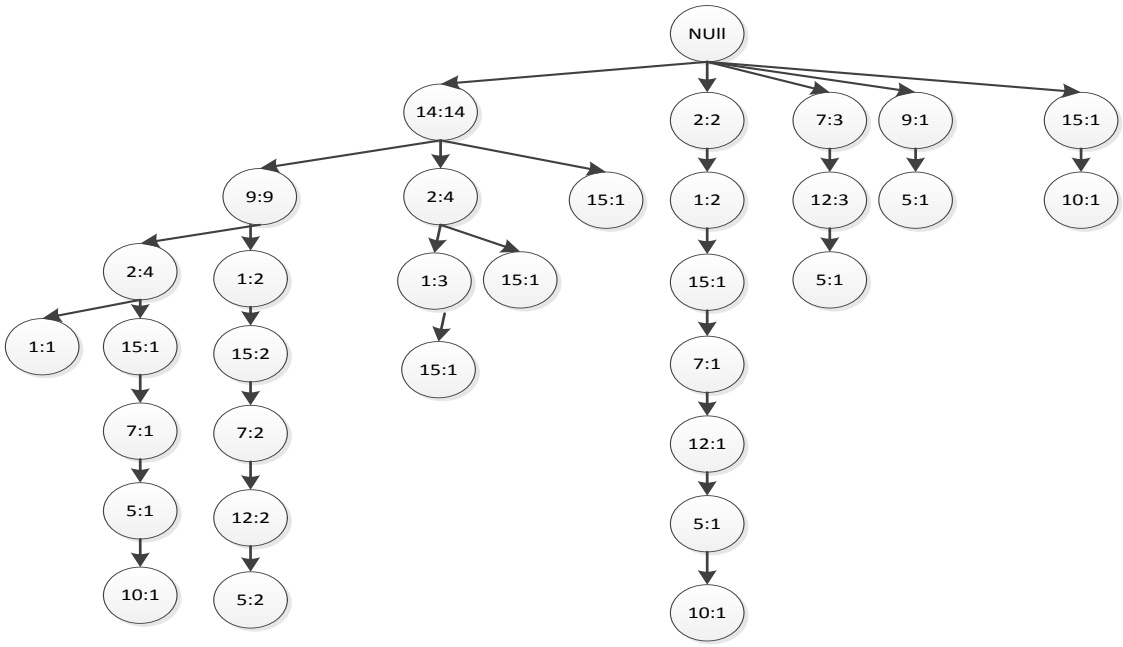
20. Transaksi 21 : 14,2,1,15
    1) Lakukan pembacaan pada *data* transaksi kedua puluh satu yang akan diproses.
    2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *data* yang dibaca sama saja yang bertambah *support count*-nya.



Gambar 3. 20 *Tree* No transaksi 21

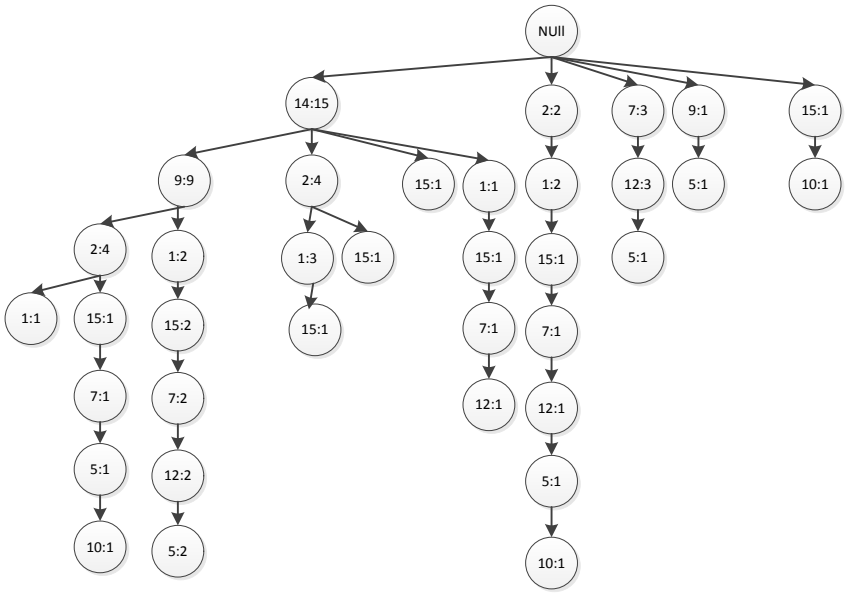
21. Transaksi 22 : 14,2,15

1) Lakukan pembacaan pada *data* transaksi keduapuluh dua yang akan diproses.
2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *data* yang dibaca sama saja yang bertambah *support count*-nya.



Gambar 3. 21 *Tree* No transaksi 22

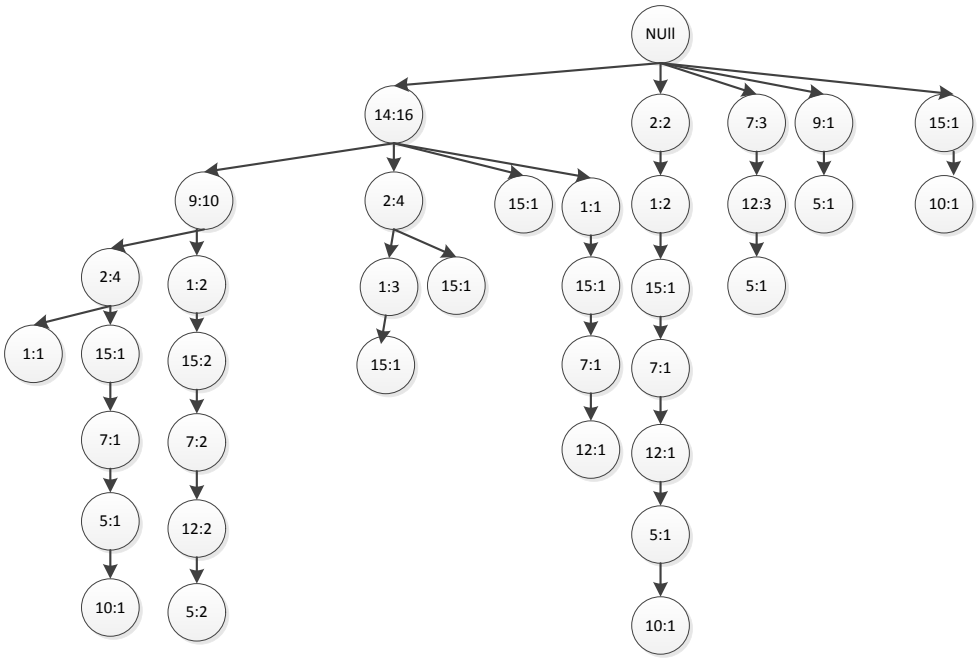
22. Transaksi 23 : 14,1,15,7,12
    1) Lakukan pembacaan pada *data* transaksi keduapuluh tiga yang akan diproses.
    2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *data* yang dibaca sama saja yang bertambah *support count*-nya.



Gambar 3. 22 *Tree* No transaksi 23

23. Transaksi 24 : 14,9

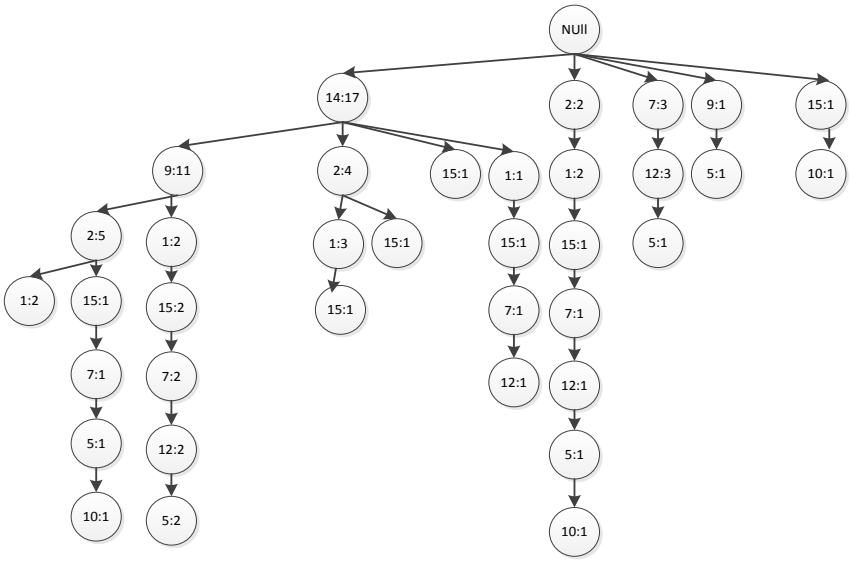
1) Lakukan pembacaan pada *data* transaksi kedua puluh empat yang akan diproses.
2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *data* yang dibaca sama saja yang bertambah *support count*-nya.



Gambar 3. 23 *Tree* No transaksi 24

24. Transaksi 25 : 14,9,2,1

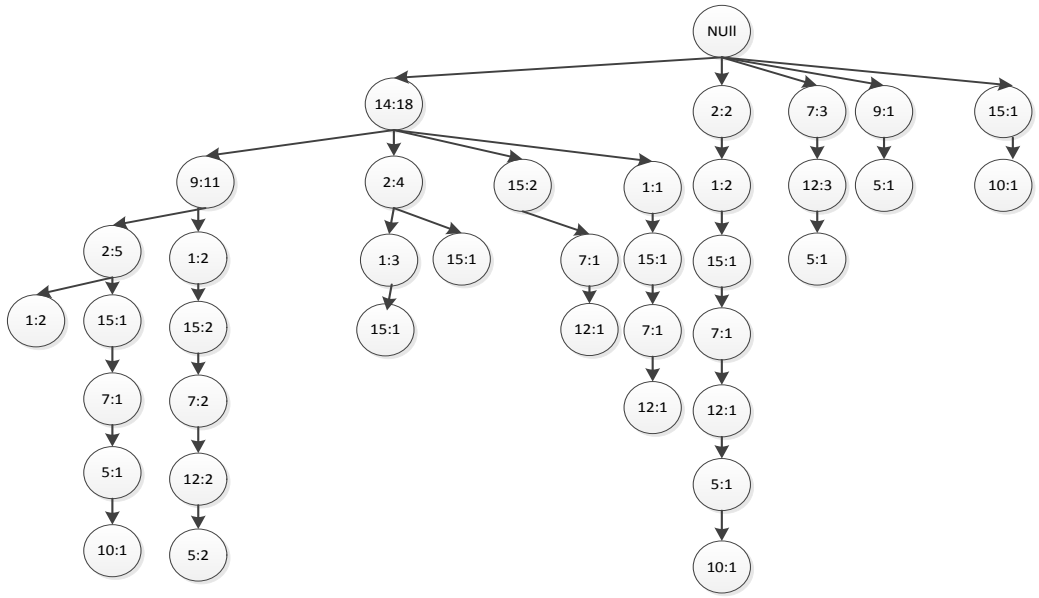
1) Lakukan pembacaan pada *data* transaksi keduapuluh lima yang akan diproses.
2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *data* yang dibaca sama saja yang bertambah *support count*-nya.



Gambar 3. 24 *Tree* No transaksi 25

25. Transaksi 26 : 14,15,7,12

1) Lakukan pembacaan pada *data* transaksi kedua puluh enam yang akan diproses.
2) Jika *data* yang keluar sama setelah pembacaan, namun berbeda lintasan maka *nod*e pada lintasan yang lain tidak bertambah *support count*-nya, hanya *cata* yang dibaca sama saja yang bertambah *support count*-nya.



Gambar 3. 25 *Tree* No transaksi 26

Setelah pembuatan *fp-tree* dilakukan langkah selanjutnya adalah tahap Pembangkitan *Conditional Pattern Base*, Tahap Pembangkitan *Conditional FP-Tree* dan pencarian *Frequent Pattern*. Adapun langkahnya yaitu :

a. Tahap Pembangkitan *Conditional Pattern Base*

   Mencari *support count* terkecil dari setiap *Item* sampai dengan terbesar berdasarkan hasil pembentukan *fp-tree* sebelumnya . Berikut ini adalah *support count* terkecil sampai dengan terbesar dapat dilihat pada tabel 3.8 :

Tabel 3. 8 *Condiitonal Pattern Base*

Item	***Conditonal Pattern Base***
10	• 14,9,2,15,7,5:1 • 2,1,15,7,12,5:1 • 15:1
5	• 14,9,2,15,7:1 • 14,9,1,15,7,12:2

	• 2,1,15,7,12:1 • 7,12:1 • 9:1
12	• 14,9,1,15,7:2 • 14,15,7:1 • 14,1,15,7:1 • 2,1,15,7:1 • 7:3
7	• 14,9,2,15:1 • 14,9,1,15:2 • 14,15:1 • 14,1,15:1 • 2,1,15:1
15	• 14,9,2:1 • 14,9,1:2 • 14,2,1:1 • 14,2:1 • 14:2 • 14,1:1 • 2,1:1
1	• 14,9,2:2 • 14,9:2 • 14,2:3 • 14:1 • 2:2
2	• 14,9:5 • 14:4
9	• 14:11

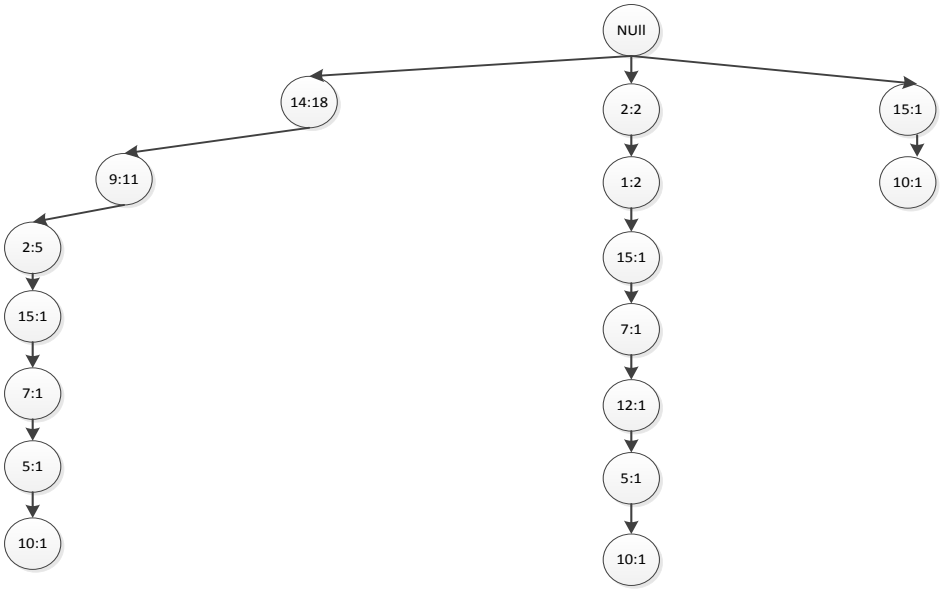
b. Tahap Pembangkitan *Conditional FP-Tree*

Pada tahap ini setiap *Item-Item*akan dibuat *conditional fp-tree*, dan apabila ada *Item* yang memenuhi *minimum support* atau sama dengan 3, maka *Item* tersebut akan dibangkitkan lagi untuk mencari *frequent pattern*. Berikut adalah *conditional fp-tree* dari tiap *Item.*

1. *Conditional FP-Tree Item* 10:

   Tahap pembentukan *Conditional FP-Tree* untuk *Item*10 dengan *Conditional pattern base*

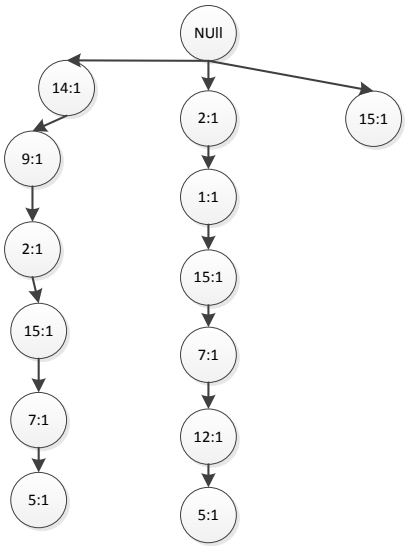
   {14,9,2,15,7,5:1},{2,1,15,7,12,5:1},{15:1}



Gambar 3. 26 *Conditional FP-Tree Item*

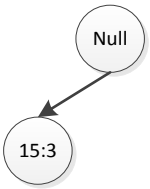
Karena *support count Item* 10 memenuhi *minimum support* maka *Item*10 akan dibangkitkan dengan *conditional fp-tree*, berikut langkah langkahnya :

1. *Item* {10} akan dihilangkan dan *support count* yang satu lintasan dengan {10} akan sama dengan *support count* {10} sebelumnya.
2. Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka *support count*-nya akan selalu bertambah 1.



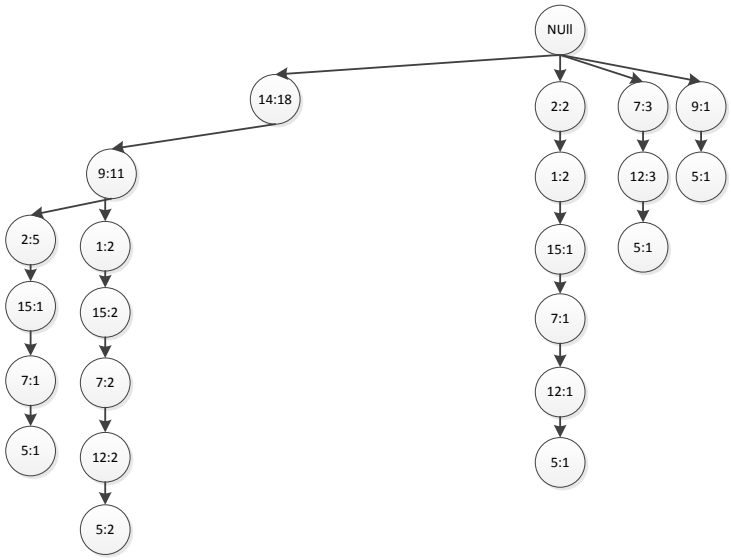
Gambar 3. 27 *Conditional FP-TreeItem* 10 setelah dibangkitkan

3. Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequent pattern* yang memenuhi *minimum support*.{15:3} adalah *Item* yang keluar yang dapat memenuhi *minimum support*.



Gambar 3. 28 *Conditional FP-Tree* 10,15:3 Hasil *Frequent Pattern*

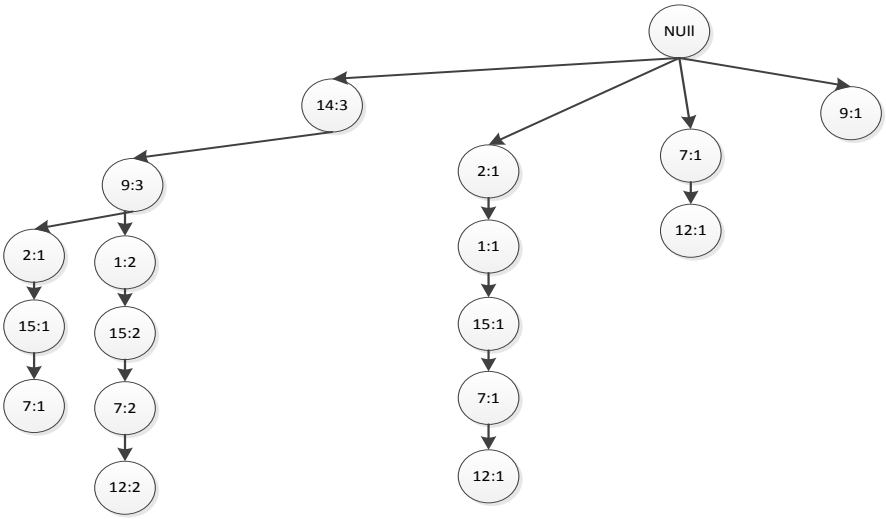
2. *Conditional FP-Tree* untuk *Item* 5 dengan *conditional pattern base* {14,9,2,15,7:1} {14,9,1,15,7,12:2} {2,1,15,7,12:1} {7,12:1} {9:1}



Gambar 3. 29 *Conditional FP-Tree Item* 5

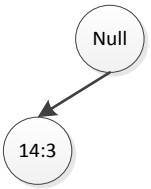
Karena *support count Item* 5 memenuhi *minimum support* maka *Item* 5 akan dibangkitkan dengan *conditional fp-tree*, berikut langkah langkahnya :

1. *Item* {5} akan dihilangkan dan *support count* yang satu lintasan dengan {5} akan sama dengan *support count* {5} sebelumnya.
2. Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka *support count*-nya akan selalu bertambah 1.

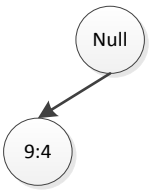


Gambar 3. 30 *Conditional FP-TreeItem*5 setelah dibangkitkan

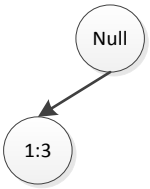
3. Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequent pattern* yang memenuhi *minimum support*. {14:3},{9:4},{1:3},{15:4},{7:5},{12:4} adalah *Item* yang keluar yang dapat memenuhi *minimum support*.



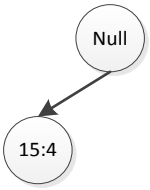
Gambar 3. 31 *Conditional FP-Tree* 5,14:3 Hasil *Frequent pattern*



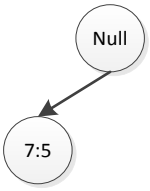
Gambar 3. 32 *Conditional FP-Tree* 5,9:4 Hasil *FrequentPattertn*



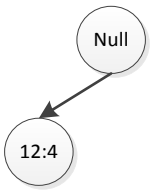
Gambar 3. 33 *Conditional FP-Tree* 5,1:3 Hasil *Frequent Pattern*



Gambar 3. 34 *Conditional FP-Tree* 5,15:4 Hasil *FrequentPattertn*

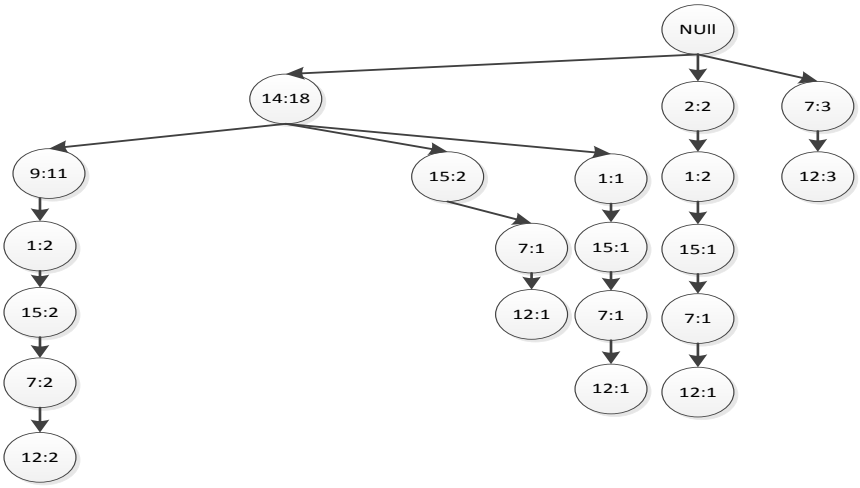


Gambar 3. 35 *Conditional FP-Tree* 5,7:5 Hasil *FrequentPattertn*



Gambar 3. 36 *Conditional FP-Tree* 5,12:4 Hasil *FrequentPattertn*

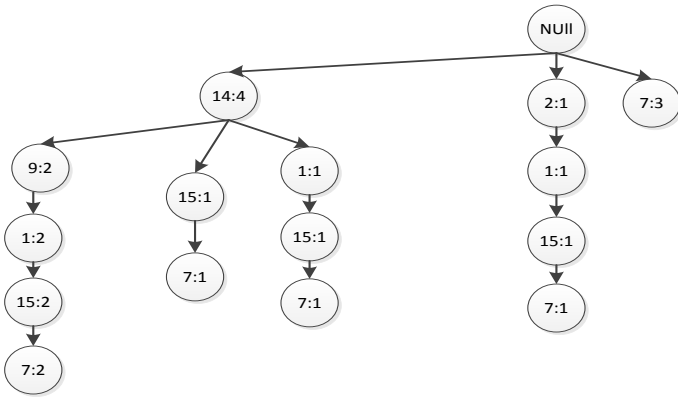
3. *Conditional FP-Tree* untuk *Item* 12 dengan *conditional pattern base* {14,9,1,15,7:2} {14,15,7:1} {14,1,15,7:1} {2,1,15,7:1} {7,:3}



Gambar 3. 37 *Conditional FP-Tree Item* 12

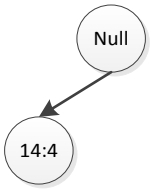
Karena *support count Item* 12 memenuhi *minimum support* maka *Item* 12 akan dibangkitkan dengan *conditional fp-tree*, berikut langkah langkahnya :

1. *Item* {12} akan dihilangkan dan *support count* yang satu lintasan dengan {12} akan sama dengan *support count* {12} sebelumnya.
2. Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka *support count*-nya akan selalu bertambah 1.

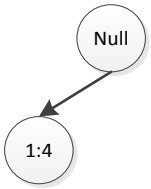


Gambar 3. 38 *Conditional FP-Tree Item* 12 setelah dibangkitkan

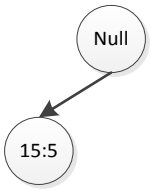
3. Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequen patterns* harus memenuhi *minimum support*. {14:4},{1:4},{15:5},{7:8} adalah *Item* yang memenuhi *minimum support*.



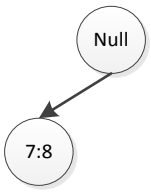
Gambar 3. 39 *Conditional FP-Tree* 12,14:4 Hasil *Frequent Pattertn*



Gambar 3. 40 *Conditional FP-Tree* 12,1:4 Hasil *Frequent Pattertn*

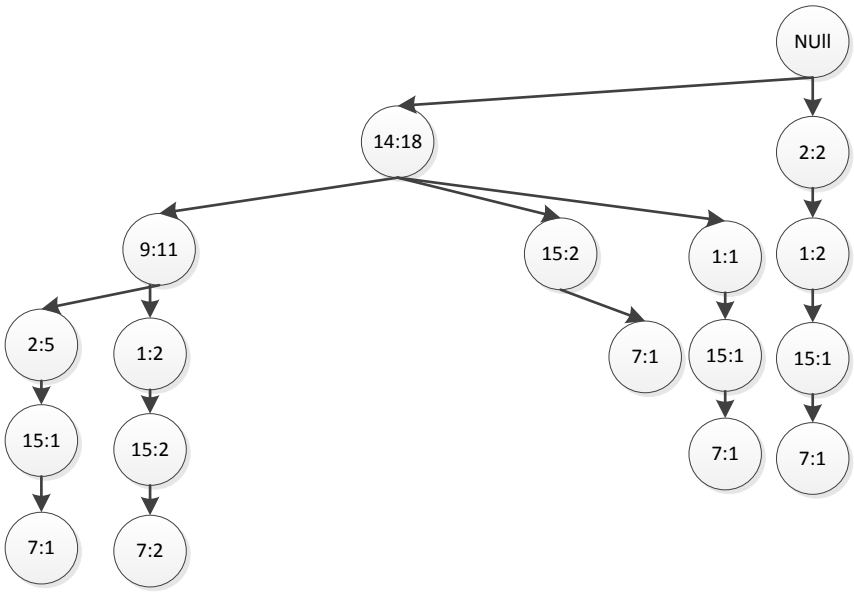


Gambar 3. 41 *Conditional FP-Tree* 12,15:5 Hasil *Frequent Pattertn*



Gambar 3. 42 *Conditional FP-Tree* 12,7:8 Hasil *Frequent Patterntn*

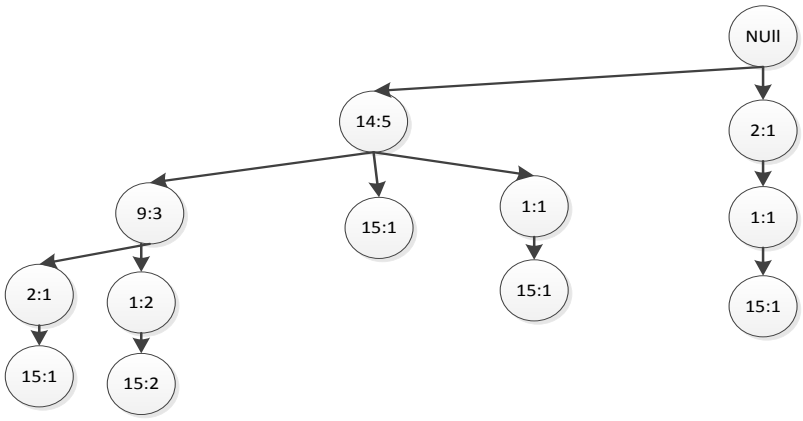
4. *Conditional FP-Tree* untuk *tem* 7 dengan *conditional pattern base* {14,9,2,15:1 } {14,9,1,15:2} {14,15:2} {14,1,15:1} {2,1,15:1}



Gambar 3. 43 *Conditional FP-Tree Item* 7

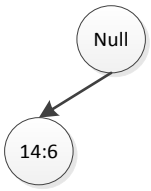
Karena *support count Item* 7 memenuhi *minimum support* maka *Item* 7 akan dibangkitkan dengan *conditional fp-tree*, berikut langkah langkahnya :

1. *Item* {7} akan dihilangkan dan *support count* yang satu lintasan dengan {7} akan sama dengan *support count* {7} sebelumnya.
2. Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka *support count*-nya akan selalu bertambah 1.

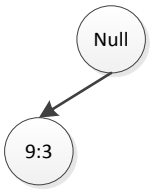


Gambar 3. 44 *Conditional FP-Tree Item* 7 setelah dibangkitkan

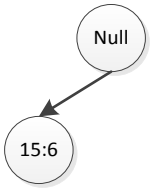
3. Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequen patterns* harus memenuhi *minimum support*.{14:5},{9:3}, {15:6},{1:4} adalah *Item* yang memenuhi *minimum support*.



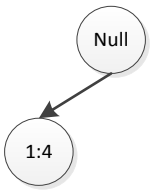
Gambar 3. 45 *Conditional FP-Tree* 7,14:5 Hasil *Frequent Patterntn*



Gambar 3. 46 *Conditional FP-Tree* 7,9:3 Hasil *Frequent Patterntn*

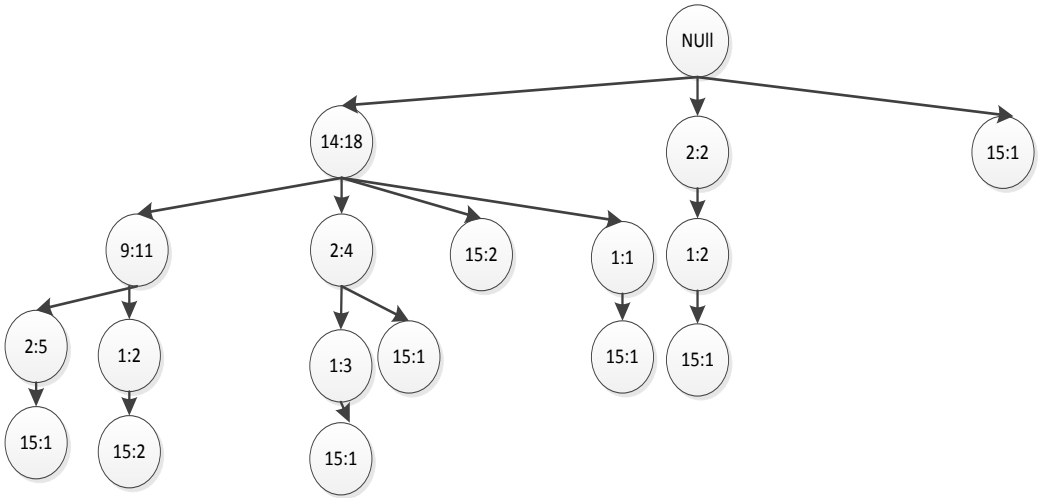


Gambar 3. 47 *Conditional FP-Tree* 7,15:6 Hasil *Frequent Patterntn*



Gambar 3. 48 *Conditional FP-Tree* 7,1:4 Hasil *Frequent Pattertn*

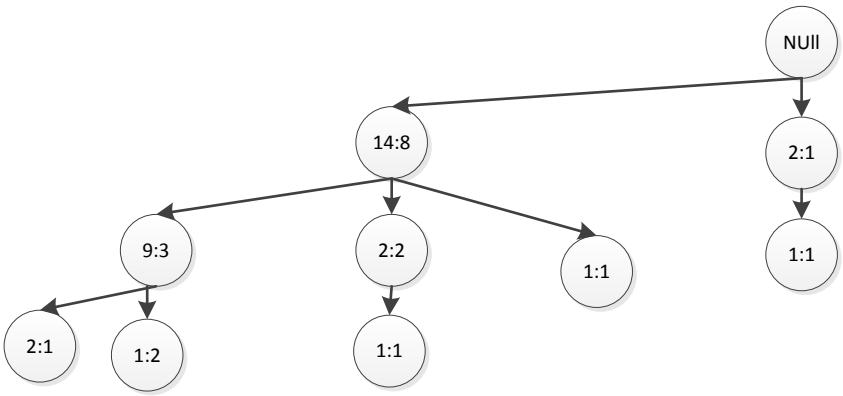
5. *Conditional FP-Tree* untuk *Item*15 dengan *conditional pattern base* {14,9,2:1} {14,9,1:2} {14,2:2} {14:2} {14,1:1} {2,1:1}



Gambar 3. 49 *Conditional FP-Tree Item* 15

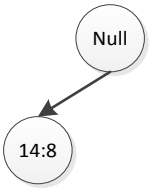
Karena *support count Item* 15 memenuhi *minimum support* maka *Item* 15 akan dibangkitkan dengan *conditional fp-tree*, berikut langkah langkahnya :

1. *Item* {15} akan dihilangkan dan *support count* yang satu lintasan dengan {15} akan sama dengan *support count* {15} sebelumnya.
2. Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka support count-nya akan selalu bertambah 1.

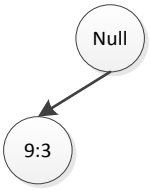


Gambar 3. 50 *Conditional FP-Tree Item* 15 setelah dibangkitkan

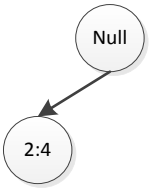
3. Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequen patterns* harus memenuhi *minimum support*. {14:8},{9:3},{2:4},{1:5} adalah *Item* yang memenuhi *minimum support*.



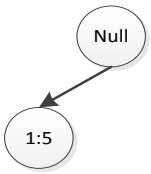
Gambar 3. 51 *Conditional FP-Tree* 15,14:8 Hasil *Frequent Pattertn*



Gambar 3. 52 *Conditional FP-Tree* 15,9:3 Hasil *Frequent Pattertn*

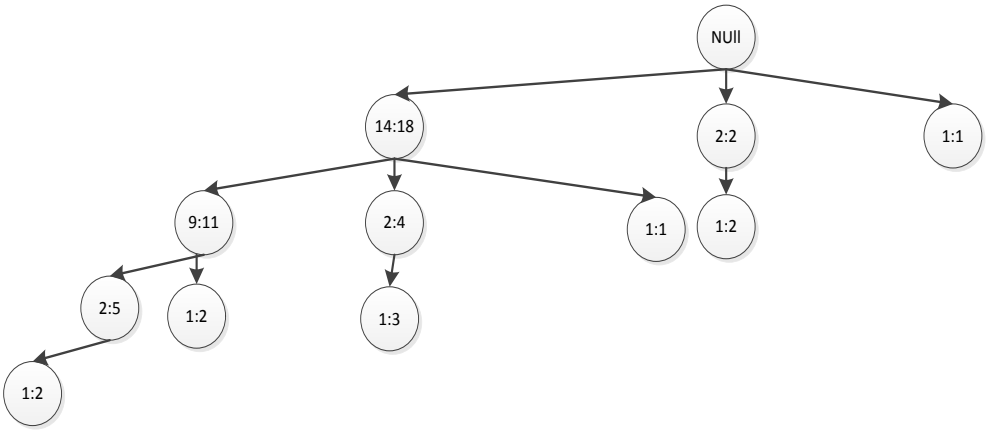


Gambar 3. 53 *Conditional FP-Tree* 15,2:4 Hasil *Frequent Pattertn*



Gambar 3. 54 *Conditional FP-Tree* 15,1:5 Hasil *Frequent Patterth*

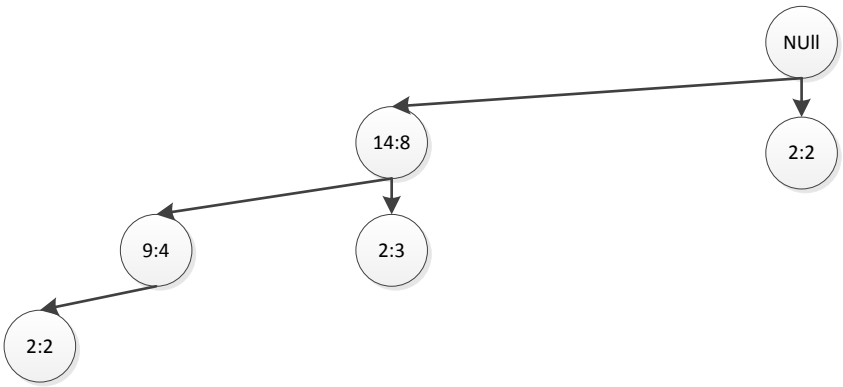
6. *Conditional FP-Tree* untuk *Item*1 dengan *conditional pattern base* {14,9,2:2} {14,9:2} {14,2:2} {14:1}{2:2}



Gambar 3. 55Conditional FP-Tree *Item* 1

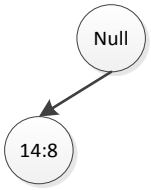
Karena *support count Item* 1 memenuhi *minimum support* maka *Item* 1 akan dibangkitkan dengan *conditional fp-tree*, berikut langkah langkahnya :

1. *Item* {1} akan dihilangkan dan *support count* yang satu lintasan dengan {1} akan sama dengan *support count* {1} sebelumnya.
2. Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka support count-nya akan selalu bertambah 1.

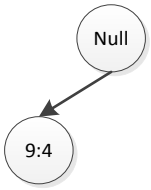


Gambar 3. 56 *Conditional FP-Tree Item* 1 setelah dibangkitkan

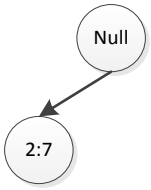
3. Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequen patterns* harus memenuhi *minimum support*. {14:8},{9:4},{2:7} adalah *Item* yang memenuhi *minimum support*.



Gambar 3. 57 *Conditional FP-Tree* 1,14:8 Hasil *Frequent Pattertn*

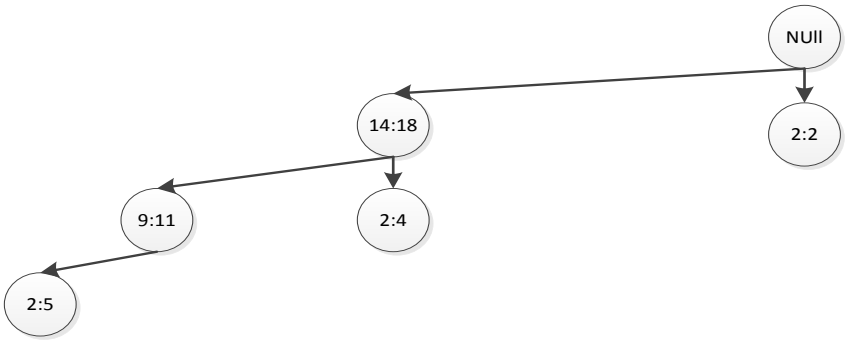


Gambar 3. 58 *Conditional FP-Tree* 1,9:4 Hasil *Frequent Pattertn*



Gambar 3. 59 *Conditional FP-Tree* 1,2:6 Hasil *Frequent Pattertn*

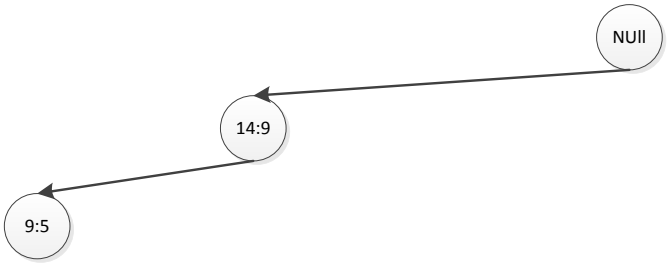
7. *Conditional FP-Tree*untuk *Item* 2 dengan *conditional pattern base* {14,9:5 } {14:2 }



Gambar 3. 60 *Conditional FP-Tree Item* 2

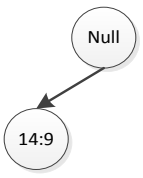
Karena *support count Item* 2 memenuhi *minimum support* maka *Item* 2 akan dibangkitkan dengan *conditional fp-tree*, berikut langkah langkahnya :

1. *Item* {2} akan dihilangkan dan *support count* yang satu lintasan dengan {2} akan sama dengan *support count* {2} sebelumnya.
2. Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka support count-nya akan selalu bertambah 1.

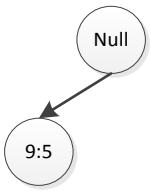


Gambar 3. 61 *Conditional FP-Tree Item* 2 setelah dibangkitkan

3. Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequen patterns* harus memenuhi *minimum support*. {14:9},{9:5} adalah *Item* yang memenuhi *minimum support*.

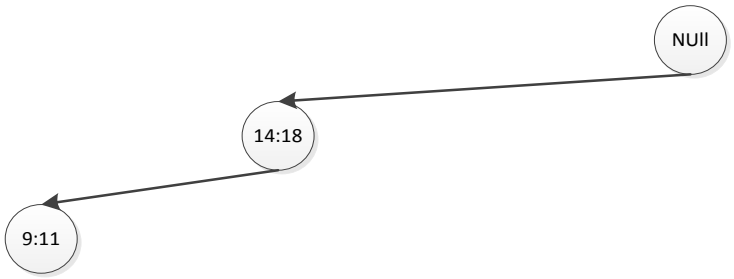


Gambar 3. 62 *Conditional FP-Tree* 2,14:9 Hasil *Frequent Patterntn*



Gambar 3. 63 *Conditional FP-Tree* 2,9:5 Hasil *Frequent Pattertn*

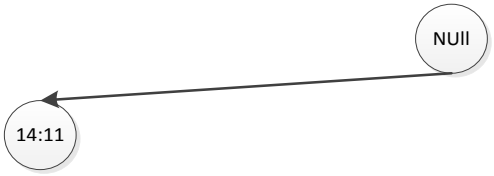
8. *Conditional FP-Tree* untuk *Item* 9 dengan *conditional pattern base* {14:11}



Gambar 3. 64 *Conditional FP-Tree Item* 9

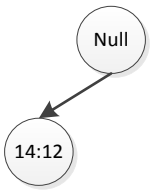
Karena *support count Item* 9 memenuhi *minimum support* maka *Item* 9 akan dibangkitkan dengan *conditional fp-tree*, berikut langkah langkahnya :

1. *Item* {9} akan dihilangkan dan *support count* yang satu lintasan dengan {9} akan sama dengan *support count* {9} sebelumnya.
2. Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka *support count*-nya akan selalu bertambah 1.



Gambar 3. 65*Conditional FP-TreeItem*9 setelah dibangkitkan

3. Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequen patterns* harus memenuhi *minimum support*. {14:11} adalah *Item* yang memenuhi *minimum support*.



Gambar 3. 66 *Conditional FP-Tree* 9,14:12 Hasil *Frequent Patterntn*

c. Tahap Pencarian *Frequnt pattern*

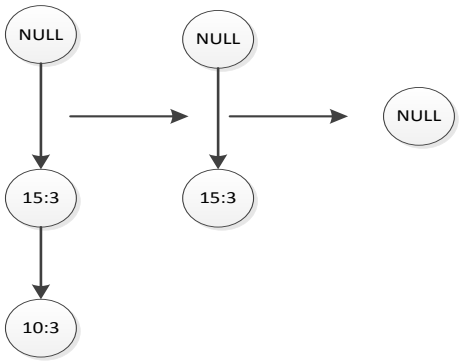
Setelah mendapatkan hasil dari *conditional fp-tree*, *frequent pattern* yang keluar, dapat dilihat pada tabel 3.9:

Tabel 3. 9*Frequent pattern*

Item	***Frequent Pattern***
10	{15,10:3}
5	{14,5:3},{9,5:4},{1,5:3},{15,5:4},{7,5:5},{12,5:4}
12	{14,12:4},{1,12:4},{15,12:5},{7,12:8}
7	{14,7:5},{9,7:3},{15,7:6},{1,7:4}
15	{14,15:8},{9,15:3},{2,15:4},{1,15:5}
1	{14,1:8},{9,1:4},{2,1:7}
2	{14,2:9},{9,2:5}
9	{14,9:11}

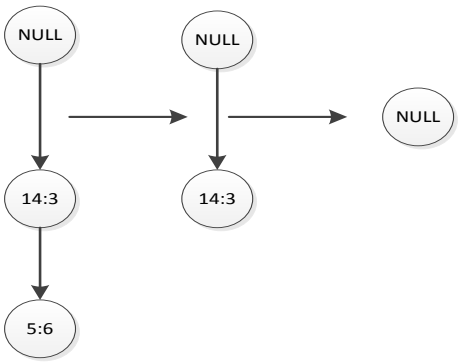
Setelah mendapatkan tabel dari *conditional fp-tre*, tahap selanjutnya adalah :

a. Akumulasikan setiap *Item* untuk setiap *frequent pattern* yang ada pada tabel 3.9 diatas.
b. Buat *conditional fp-tree* dari setiap *frequent pattern* yang keluar.
    1. Untuk *Item*10, *Itemfrequent pattern* yang keluar adalah {15,10:3}

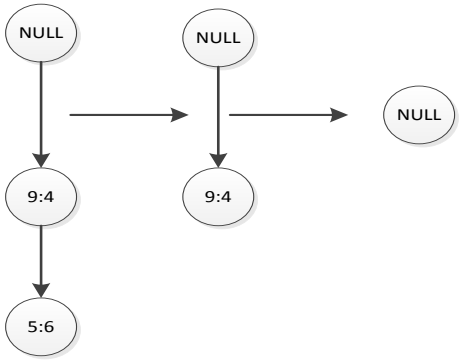


Gambar 3. 67 *Conditional FP-Tree Untuk Item Frequent* 10, 15:3

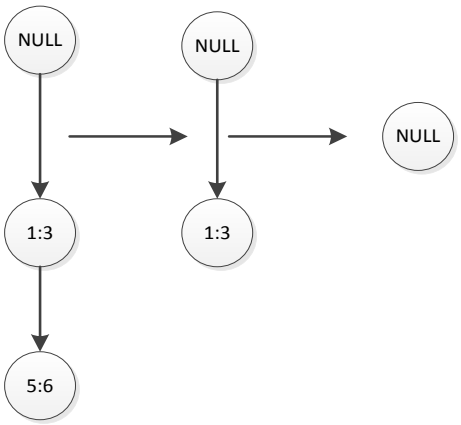
2. Untuk *Item* 5, *Item frequent pattern* yang keluar adalah {14,5:3},{9,5:4},{1,5:3},{15,5:4},{7,5:5},{12,5:4}



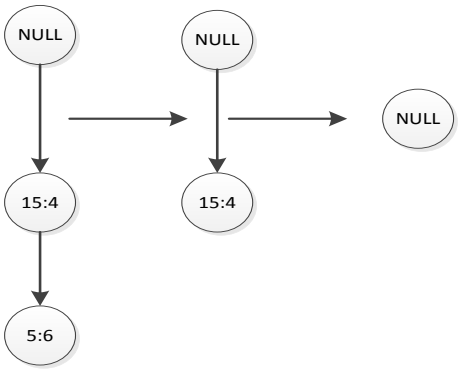
Gambar 3. 68 *Conditional FP-Tree Untuk Item Frequent* 5, 14:3



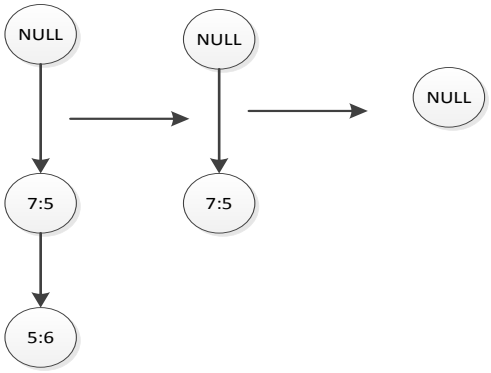
Gambar 3. 69 *Conditional FP-Tree Untuk Item Frequent* 5, 9:4



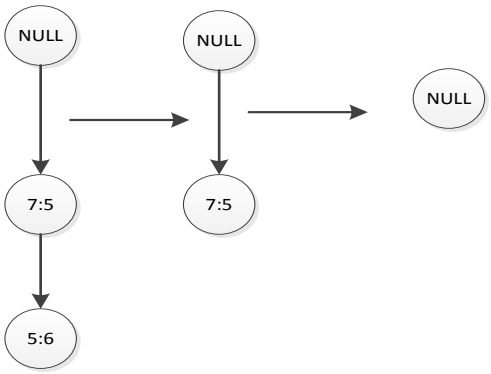
Gambar 3. 70 *Conditional FP-Tree Untuk Item Frequent* 5, 1:3



Gambar 3. 71 *Conditional FP-Tree Untuk Item Frequent* 5, 15:4

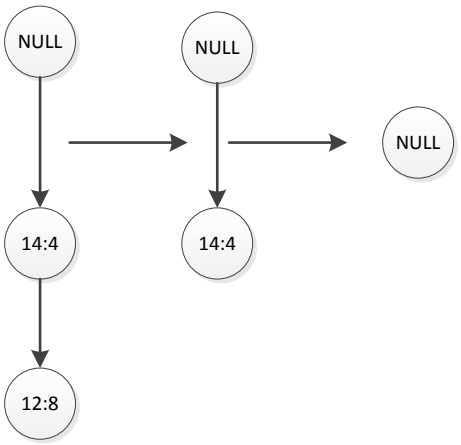


Gambar 3. 72 *Conditional FP-Tree Untuk Item Frequent* 5,7:5

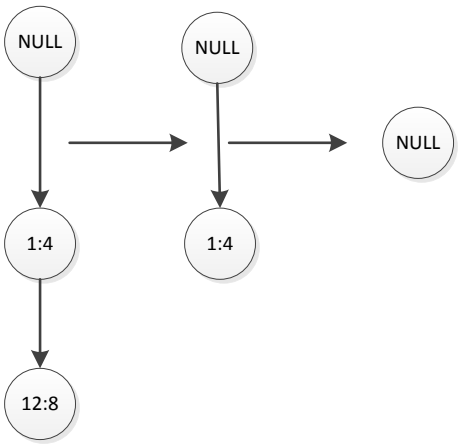


Gambar 3. 73 *Conditional FP-Tree Untuk Item Frequent* 5,12:5

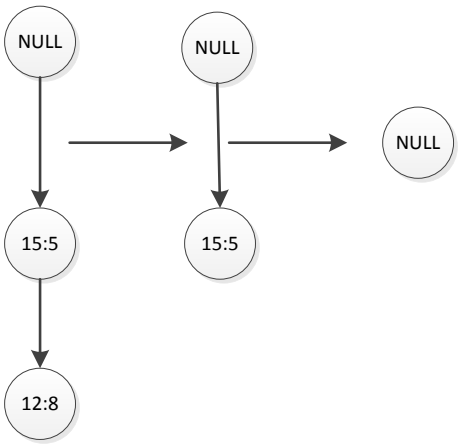
3. Untuk *Item* 12, *Item frequent Itemset* yang keluar adalah {14,12:4},{1,12:4},{15,12:5},{7,12:8}



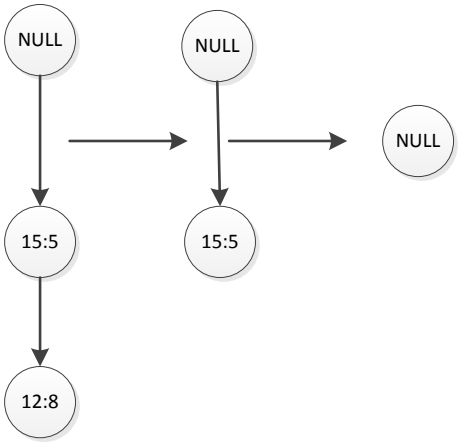
Gambar 3. 74 *Conditional FP-Tree Untuk Item Frequent* 12, 14:4



Gambar 3. 75 *Conditional FP-Tree Untuk Item Frequent* 12, 1:4

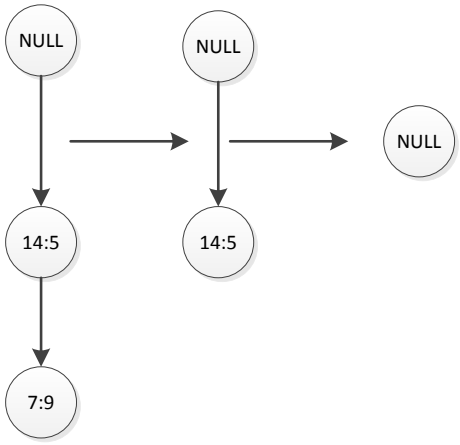


Gambar 3. 76 *Conditional FP-Tree Untuk Item Frequent* 12, 15:5

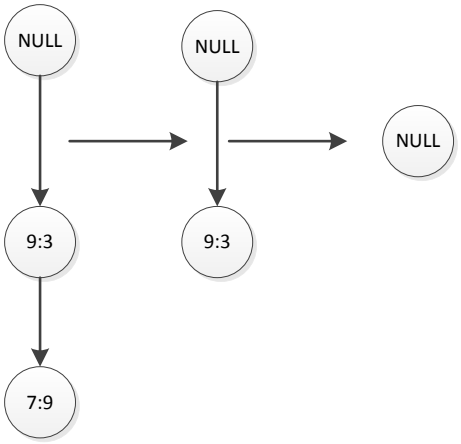


Gambar 3. 77 *Conditional FP-Tree Untuk Item Frequent* 12, 7:8

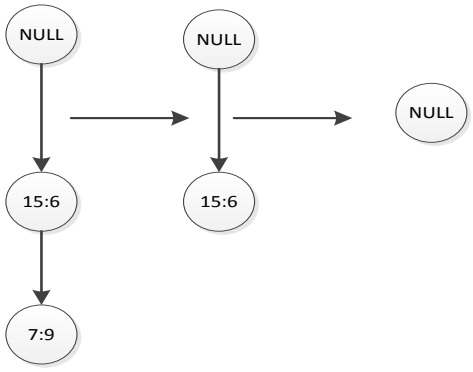
4. Untuk *Item* 7, *Itemfrequent Itemset* yang keluar adalah {14,7:5},{9,7:3},{15,7:6},{1,7:4}



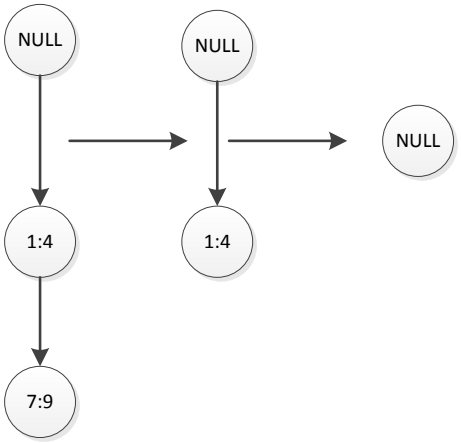
Gambar 3. 78 *Conditional FP-Tree Untuk Item Frequent* 7, 14:5



Gambar 3. 79 *Conditional FP-Tree Untuk Item Frequent* 7, 9:3



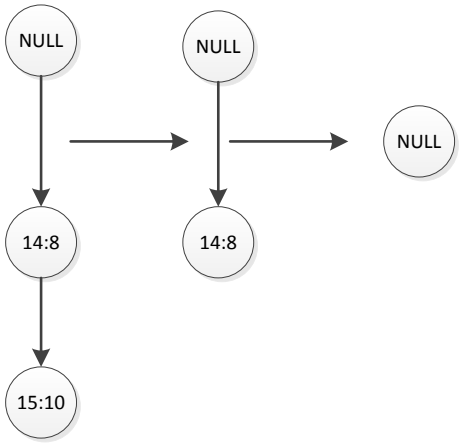
Gambar 3. 80 *Conditional FP-Tree Untuk Item Frequent* 7,15:6



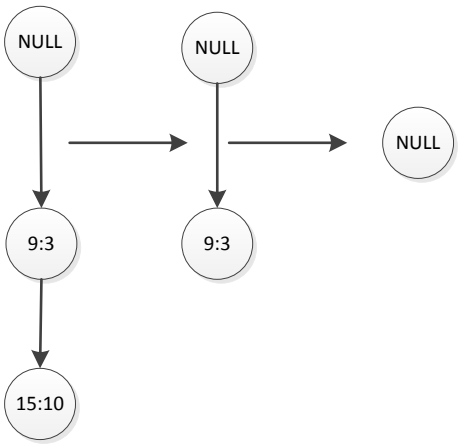
Gambar 3. 81 *Conditional FP-Tree Untuk Item Frequent* 7, 1:4

5. Untuk *Item* 15, *Itemfrequent Itemset* yang keluar adalah

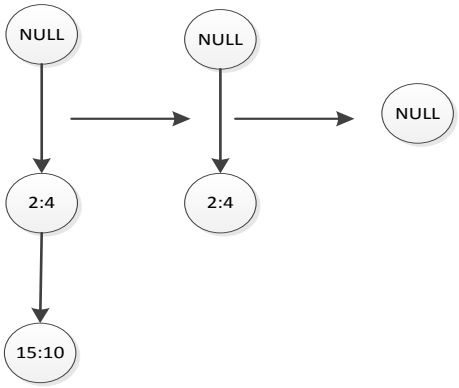
{14,15:8},{9,15:3},{2,15:4},{1,15:5}



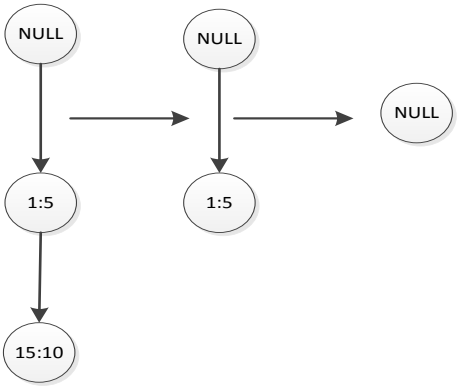
Gambar 3. 82 *Conditional FP-Tree Untuk Item Frequent* 15, 14:8



Gambar 3. 83 *Conditional FP-Tree Untuk Item Frequent* 15, 9:3



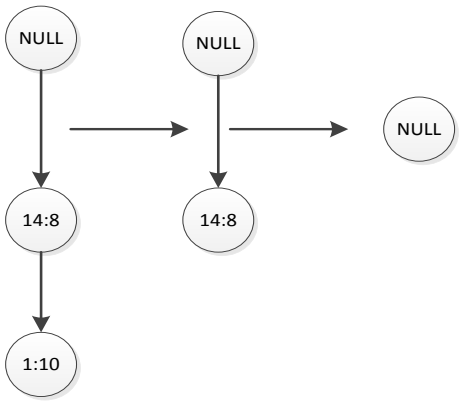
Gambar 3. 84 *Conditional FP-Tree Untuk Item Frequent* 15,2:4



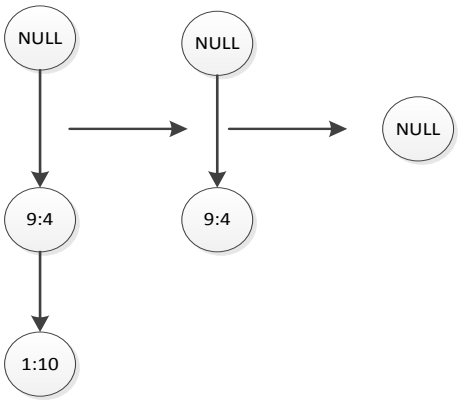
Gambar 3. 85 *Conditional FP-Tree Untuk Item Frequent* 15,1:5

6. Untuk *Item* 1, *Item frequent Itemset* yang keluar adalah

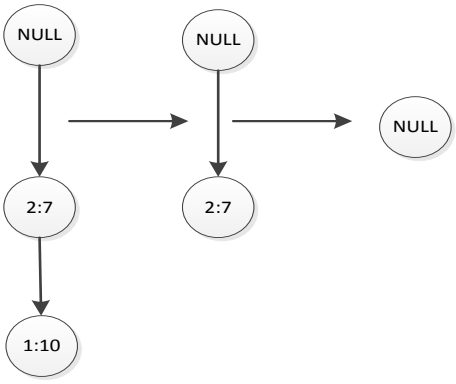
{14,1:8},{9,1:4},{2,1:7}



Gambar 3. 86 *Conditional FP-Tree Untuk Item Frequent* 1, 14:8



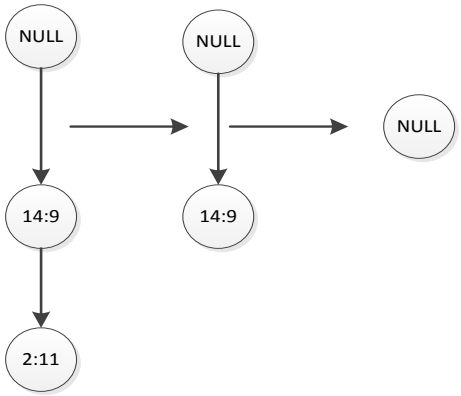
Gambar 3. 87 *Conditional FP-Tree Untuk Item Frequent* 1, 9:4



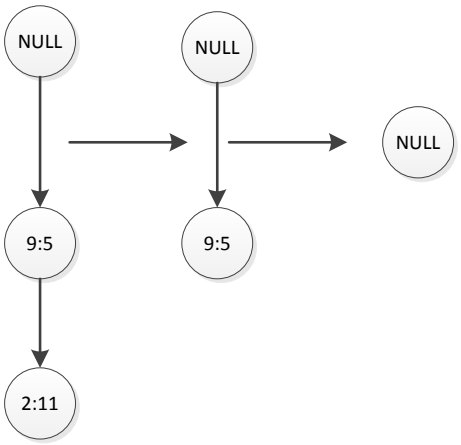
Gambar 3. 88 *Conditional FP-Tree Untuk Item Frequent* 1,2:7

7. Untuk *Item*2, *Item frequent Itemset* yang keluar adalah

   {14,1:9},{9,1:5}

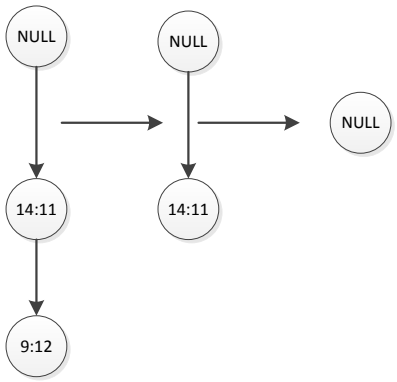


Gambar 3. 89 *Conditional FP-Tree Untuk Item Frequent* 2, 14:7



Gambar 3. 90 *Conditional FP-Tree Untuk Item Frequent* 2, 9:5

8. Untuk *Item*9, *Item frequent Itemset* yang keluar adalah {9, 14:11}



Gambar 3. 91 *Conditional FP-Tree Untuk Item Frequent* 9, 14:11

Setelah mendapatkan hasil *conditional fp tree* dari tiap *frequent pattern* selanjutnya adalah menentukan *suffix* dari tiap *frequent pattern* seperti pada table 3.10 :

Tabel 3. 10 Hasil *Frequent Patterns*

Suffix	***Frequent Patterns***
10	10,15
5	5,14 5,9 5,1 5,15 5,7 5,12
12	12,14 12,1 12,15 12,7
7	7,14 7,9 7,15

	7,1
Suffix	***Frequent Patterns***
15	15,14 15,9 15,2 15,1
1	1,14 1,9 1,2
2	2,14 2,9
9	9,14

Dari hasil *frequent patterns* yang didapatkan pada table 3.10 di atas akan di *generate* untuk mendapatkan *rule* dengan *minimum support* 3 dan *minimum confidence* 70 % , nilai minimum confidence ini untuk meyakinkan kuatnya hubungan antara *Item* yang satu dengan yang lain. Dengan menggunakan rumus yang ada pada bagian bab 2 untuk memperoleh nilai *support* dan nilai *confidence*.

$Confidence = \frac{\text{Jumlah transaksi yang mengandung A dan B}}{\text{Jumlah transaksi yang mengandung A}}$x100%..

Berikut hasil perhitungan *minimum confidence* dapat dilihat pada tabel 3.11 :

Tabel 3. 11Hasil Perhitungan *Minimum Confidence*

Rule	***Minimum Confidence***%
10→15	(3/3)*100=100%
15→10	(3/10)*100=30%
5→14	(3/6)*100=50%
14→5	(3/18)*100=17%
5 → 9	(4/6)*100 =66%
9→ 5	(4/12)*100 =33%
5→1	(3/6)*100=50%

1→5	(3/10)*100=33%
Rule	***Minimum Confidence*%**
5→15	(4/6)*100=66%
15→5	(4/10)*100=40%
5→ 7	(5/6)*100 =83%
7→ 5	(5/9)*100 =55%
5→12	(4/6)*100=66%
12→5	(4/8)*100=50%
12→14	(4/8)*100 =50%
14→12	(4/18)*100=22%
12→1	(4/8)*100=37%
1→12	(4/10)*100=44%
12→15	(5/8)*100=63%
15→12	(5/10)*100=55%
12→7	(8/8)*100=100%
7→12	(8/9)*100=88%
7 → 14	(5/9)*100 =55%
14 → 7	(5/18)*100 =27%
7→9	(3/9)*100=33%
9→7	(3/12)*100=25%
15→7	(6/10)*100=66%
7→15	(6/9)*100 =67%
7→1	(4/9)*100 =44%
1→7	(4/10)*100 =44%
15 → 14	(8/10)*100 =80%
14 → 15	(8/18)*100 =44%
15→9	(3/10)*100=30%
9→15	(3/12)*100=25%
15→2	(4/10)*100=40%
2→15	(4/11)*100=36%

15→1	(5/10)*100 =50%
Rule	***Minimum Confidence*%**
1→15	(5/10)*100 =50%
1 → 14	(8/10)*100 =80%
14 → 1	(8/18)*100 =44%
1→9	(4/10)*100=40%
9→1	(4/12)*100=33%
1→2	(7/10)*100=70%
2→1	(7/11)*100=63%
2→14	(9/11)*100=81%
14→2	(9/18)*100=50%
2→9	(5/11)*100=45%
9→2	(5/12)*100=42%
9→14	(11/12)*100=91%
14→9	(11/18)*100=61%

Setelah melakukan perhitungan *minimum confidence* pada table 3.11, proses selanjutnya adalah mencari *generate rule* yang memenuhi nilai *minimum confidence* yang telah ditentukan sebelumnya. Jika terdapat *Itemset* yang tidak memenuhi *confidence* akan dihilangkan atau dihapus. Hasil *generate* dapat dilihat pada tabel 3.12 :

Tabel 3. 12hasil *Generate Rule*

Rule	*Minimum Confidence*
10→15	(3/3)*100=100%
5→ 7	(5/6)*100 =83%
12→7	(8/8)*100=100%
7→12	(8/9)*100=88%
15 → 14	(8/10)*100 =80%
1 → 14	(8/10)*100 =80%
1→2	(7/10)*100=70%
2→14	(9/11)*100=81%
9→14	(11/12)*100=91%

Rule yang telah memenuhi nilai *minimum confidance* kemudian disesuaikan dengan nama barang. Sehingga kode barang yang terdapat pada *rule* sesuai dengan nama barang dan dapat dijadikan informasi. Tabelnama barang dapat dilihat pada tabel 3.13:

Tabel 3. 13 Nama Barang

Kode Barang	Nama Barang
1	Dompet
2	Gelang
3	Lidah Levis
4	S
5	Saku Dua
6	Sekolah Saku Dua
7	Sleting depan
8	Sleting Dua
9	Tali Panjang
10	Tali Sumbu
11	Tali Tutup
12	Tas Garis
13	Tas kecil
14	Tas sekolah
15	Tas Pasar
16	Tas Saku

Informasi yang dihasilkan berupa kombinasi *Item* dan nama barang berdasarkan kode barangnya sesuai dengan tabel 3.13. dapat dilihat pada tabel 3.14 :

Tabel 3. 14informasi yang dihasilkan dari *rule*

Rule	**Informasi Rekomendasi Produk**
10→15	tali sumbu → Tas pasar
5→ 7	saku dua → Sleting depan
12→7	Tas garis →sleting depan
7→12	Sleting depan → tas garis
15 → 14	tas pasar → Tas sekolah
1 → 14	dompet → Tas sekolah
1→2	dompet → Gelang
2→14	gelang→ Tas sekolah
9→14	tali panjang → Tas sekolah

Dari tabel informasi yang dihasilkan dari *rule* maka dijadikan informasi dalam perekomendasian produk tas apa saja yang akan diproduksi dibulan selanjutnya. Berikut adalah informasi model tas yang akan diproduksi dibulan selanjutnya :

Tali Sumbu
Tas Pasar
Saku Dua
Sleting Depan
Tas Garis
Tas Sekolah
Dompet
Gelang

Tali Panjang

3.1.3 Analisis Spesifikasi Kebutuhan Perangkat Lunak

Spesifikasi kebutuhan perangkat lunak adalah kebutuhan kebutuhan apa saja yang diperlukan untuk membangun sebuah perangkat lunak sesuai dengan kebutuhan penggunanya. Spesifikasi kebutuhan perangkat lunak dibagi menjadi dua yaitu kebuthan *non-fungsional* dan kebutuhan *fungsional*.

3.1.3.1 Analisis kebutuhan *Non-Fungsional*

Analisis kebutuhan *non fungsional* merupakan bagian penting lainnya agar simulasi perangkat lunak dapat berjalan dengan baik serta untuk mengetahui elemen-elemen yang berhubungan dengan sistem yang sedang berjalan. Tahapan analisis kebutuhan *non fungsional* dibagi menjadi beberapa tahap yaitu analisis kebutuhan perangkat keras, analisis kebutuhan perangkat lunak dan analisis kebutuhan pengguna (*user*).

1. Analisis Kebutuhan Perangkat Keras

   Perangkat keras pendukung dalam pembangunan perangkat lunak ini adalah sebagai berikut :

   a. Analisis kebutuhan perangkat keras pada sistem yang sedang berjalan di kartin Collection terdapat dua buah komputer dengan spesifikasi sebagai berikut, dapat dilihat pada tabel 3.15 :

Tabel 3. 15 Spesifikasi Perangkat Keras yang Sedang Berjalan

No	Perangkat Keras	Spesifikasi
1	RAM	1 GB
2	Harddisk	128 GB
3	Monitor	LCD 16 inchi
4	Processor	Intel Pentium Dual 3,0 Ghz

b. Analisis kebutuhan perangkat keras pada sistem yang akan dibangun membutuhkan spesifikasi *minimum* perangkat keras sebagai berikut, dapat dilihat pada tabel 3.16:

Tabel 3. 16 Spesifikasi Perangkat Keras yang Akan Dibangun

No	Perangkat Keras	Spesifikasi
1	RAM	512 MB
2	Harddisk	1 GB
3	Monitor	LCD 16 inchi
4	Processor	Berkecepatan minimal 1,9 Ghz

c. Evaluasi kebutuhan perangkat keras

Dengan membandingkan spesifikasi perangkat keras untuk menjalankan perangkat keras yang akan dibangun dengan spesifikasi perangkat keras yang ada pada kartin Collection, maka dapat disimpulkan bahwa perangkat keras yang tersedia sesuai dengan kebutuhan dan telah memenuhi spesifikasi seperti pada tabel 3.16 diatas.

2. Analisis Kebutuhan Perangkat Lunak

Perangkat lunak pendukung dalam pembangunan perangkat lunak iniadalah :

a. Analisis kebutuhan perangkat lunak pada sistem yang sedang berjalan di Kartin Colection adalah sebagai berikut, dapat dilihat pada tabel 3.17 :

Tabel 3. 17 Spesifikasi Perangkat Lunak yang Sedang Berjalan

No	Perangkat Keras	Spesifikasi
1	Sistem Operasi	Microsoft Windows XP

b. Analisis kebutuhan perangkat lunak pada sistem yang dibangun adalah sebagai berikut, pada dilihat pada tebel 3.18 :

Tabel 3. 18 Spesifikasi Perangkat Lunak yang Akan Dibangun

No	Perangkat Keras	Spesifikasi
1	Sistem Operasi	Microsoft Windows 7
2	*Web Server* Pembangun	Xampp Server
3	DBMS	MySQL
4	*Compiler*	Microsoft Visual Studio

c. Evaluasi kebutuhan perangkat lunak

Dengan membandingkan spesifikasi perangakat lunak yang dibutuhkan dengan perangkat lunak yang ada pada kartin Collection, dapat disimpulkan bahwa spesifikasi perangkat lunak yang ada di Kartin Collection masih belum memenuhi kebetuhan yang diperlukan karena pada perangkat lunak tidak terdapat *Database server*, sehingga perlu ditambahkan *xampp server*.

3. Analisis Pengguna (*User*)

Analisis kebutuhan perangkat pikir digunakan untuk menganalisa karakteristik pengguna perangkat lunak yang akan dibangun. Adapun beberapa karakteristiknya adalah sebagai berikut :

Tabel 3. 19 Analisa Kebutuhan di Kartin Collection

Pengguna	Tanggung Jawab	Tingkat Pendidikan	Tingkat Keterampilan

Pengguna	Merencanakan, melaksanakan, dan mengevaluasi terhadap strategi bisnis yang digunakan	Minimal Sekolah Menengah Atas (SMA)	Mampu mengoperasikan komputer

Adapun daftar pengguna yang dibutuhkan dalam menjalankan perangkat lunak yang akan dibangun adalah sebagai berikut :

Tabel 3. 20 Analisis Kebutuhan Pada Sistem

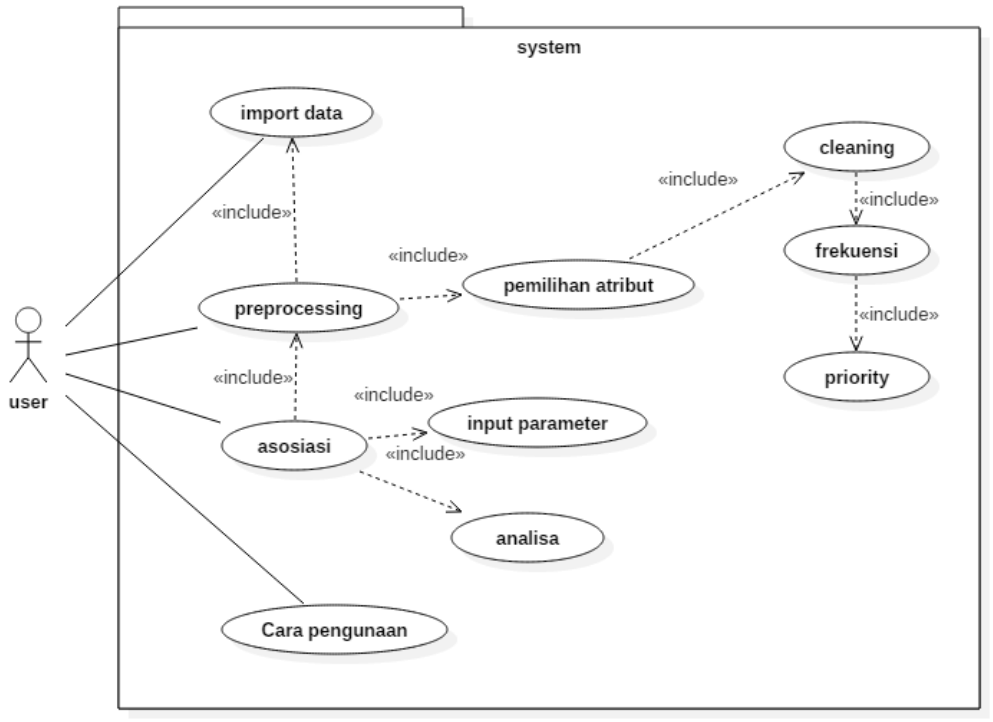
Pengguna	Hak Akses	Tingkat Pendidikan	Jenis Pelatihan
Pengguna	Memproses *Data* transaksi menjadi informasi yang berguna untuk menentukan strategi bisnis yang harus diterapkan, agar dapat meningkatkan kualitas mutu produk model tas nya	Minimal Sekolah Menengah Atas (SMA)	Pengoperasian komputer dan pemahaman konten yang tersedia pada program

3.1.3.2 Analisis Kebutuhan *Fungsional*

Analisis kebutuhan fungsional yang akan dibuat meliputi Diagram *use case, use case* skenario, *diagram aktivity*, *diagram class* dan *diagram sequence.*

1. *Diagram Use Case*

   Adapun diagram *use case* dari sistem yang akan dibuat dapat dilihat di gambar 3.92 :



Gambar 3. 92 *Diagram Use Case*

2. *Use Case Scenario*

1) Use Case Scenario Import Data

Skenario *use case import Data* menggambarkan langkah – langkah aksi *user* terhadap sistem untuk melakukan *import data* transaksi penjualan yang akan disimpan ke dalam *Database.* Skenario *use case import data* dapat dilihat pada tabel 3.21 :

Tabel 3. 21 *Use Case Scenario* Import *Data*

Use case Name	***Import Data***
Goal In Context	Import *Data* transaksi dan menyimpan ke dalam *Data*base
Precondition	Aktor berada pada *form import Data*
Successful End Condition	*Data* berhasil disimpan ke dalam *Data*base
Failed End Condition	*Data* nilai tidak berhasil disimpan ke dalam *Data*base
Actors	*User*

Triger	Aktor memilih menu *import Data*	
Main Flow	*Step*	*Action*
	1	Aktor mencari *file excel*
	2	Aktor menyimpan *file*
	3	Sistem menyimpan *file* ke *Data*base
	4	Sistem menampilkan *data file* yang dimasukan
Extension	*Step*	*Branching Action*
	1.1	Aktor memilih membatalkan proses *import*
	1.2	Sistem menampilkan pesan *data* tidak berhasil di*import*

2) *Use Case ScenarioPreprocesing*

Skenario *use case preprocessing* menjelaskan langkah-langkah untuk melakukan proses *cleaning* dan pemilihan atribut. Skenario *use case preprocesing* dapat dilihat pada Tabel 3.22 :

Tabel 3. 22 *Use Case Scenario Prepocessing*

Use case Name	***Preprocesing***	
Goal In Context	Sistem melakukan proses pemilihan atribut *Data* dan*Diagram Activity*	
Precondition	*Data* transaksi telah disimpan didalam *Database*	
Successful End Condition	Sistem berhasil melakukan proses *Data* pemilihan dan*Diagram Activity*	
Failed End Condition	Sistem tidak berhasil melakukan proses pemilihan atribut *Data* dan *Diagram Activity*	
Actors	*User*	
Triger	*User* melakukan *request* proses pemilihan atribut dan celaning *Data* kepada sistem	
Included Cases	Pemilihan atribut,*Diagram Activity* , frekuensi, *priority*	
Main Flow	*Step*	*Action*
	1	Aktor memilih menu *preprocessing*
	2	Aktor melakukan *request* pemilihan atribut
	3	Sistem melakukan sleksi atribut
	4	Sistem menampilkan *data* pemilihan atribut
	5	Aktor melakakukan *request Diagram Activity*
	6	Sistem melakukan proses *Diagram Activity*
	7	Sistem menampilkan hasil celaning

	8	Aktor melakukan *request* frekeunsi
	9	Sistem melakukan perhitungan pada *data* yang telah di*celaning*
	10	Sistem menampilkan *data* hasil perhitungan frekuensi
	11	Aktor melakukan *request priority*
	12	Sistem melakukan pengurtutan berdasarkan *priority*
	13	Sistem menampilkan hasil pengurutan berdasarkan *priority*
Extension	*Step*	*Branching Action*
	-	-

3) *Use Case Scenario* Pemilihan Atribut

Skenario *use case seleksi atribut* menggambarkan langkah-langkah melakukan pemilihan atribut terhadap d*ata* transaksi yang telah melewati proses *import data*. Skenario *use case* pemilihan atribut dapat dilihat pada table 3.23 :

Tabel 3. 23 *Use Case Scenario* Pemilihan Atribut

Use case Name	**Pemilihan Atribut**	
Goal In Context	Memilih atribut dalam *data* transaksi yang akan digunakan dalam proses asosiasi	
Precondition	*Data* masukan valid	
Successful End Condition	Sistem berhasil memilih atribut yang akan digunakan dalam proses asosiasi	
Failed End Condition	Sistem tidak berhasil memilih atribut yang akan digunakan dalam proses asosiasi	
Actors	Manager	
Triger	Aktor mengklik tombol *generate atribut*	
Main Flow	*Step*	*Action*
	1	Aktor melakukan *request* pemilihan atribut
	2	Sistem melakukan proses pemilihan terhadap kolom *data* transaksi
	3	Sistem menampilkan pemilihan atribut
Extension	*Step*	*Branching Action*
	-	-

4) *Use Case Scenario Clenaing*

Scenario use case cleaning menggambarkan langkah-langkah melakukan pembersihan terhadap *data* transaksi penjualan yang hanya memiliki satu pembelian dalam satu transaksi. Scenario use case *cleaning* dapat dilihat pada table 3.24 :

Tabel 3. 24 *Use Case Scenario Clenaing*

Use case Name	***Clenaing***	
Goal In Context	Membersihkan *data* transaksi yang hanya memiliki satu pembelian dalam satu transaksi	
Precondition	*Data* sudah dilakukan pemilihan atribut	
Successful End Condition	Sistem berhasil membersihkan d*ata* pada transaksi yang hanya terdapat satu pembelian	
Failed End Condition	Sistem tidak berhasil membersihkan *data* pada transaksi yang hanya terdapat satu pembelian	
Actors	*User*	
Triger	Sudah melewati proses *import Data*	
Main Flow	*Step*	*Action*
	1	Aktor melakukan *request Diagram Activity*
	2	Sistem melakukan proses *Diagram Activity*
	3	Sistem menampilkan hasil proses *Diagram Activity*
Extension	*Step*	*Branching Action*
	-	-

5) *Use Case Scenario* Frekuensi

Scenario use case frekuensi menggambarkan langkah-langkah melakukan perhitungan frekuensi terhadap *Item* yang terdapat pada *data* transaksi. *Scenario* use case *data* frekuensi dapat dilihat pada tabel 3.25 :

Tabel 3. 25 *Use Case Scenario* Frekuensi

Use case Name	**Frekuensi**	
Goal In Context	Menghitung jumlah kemunculan tiap *Item* dalam *Data* transaksi	
Precondition	Posisi *Data* sudah dilakukan *Cleaning*	
Successful End Condition	Sistem berhasil menampilkan frekuensi tiap *Item*	
Failed End Condition	Sistem tidak berhasil melakukan perhitungan frekuensi*Item*	
Actors	Manager	
Triger	Aktor menekan tombol Frekuensi	
Main Flow	Step	Action
	1	Aktor melakukan *request* frekeunsi
	2	Sistem mengambil *data* hasil *cleaning*
	3	Sistem melakukan proses penghitungan frekuensi tiap *Item*
	4	Sistem menampilkan *Item* beserta frekuensi kemunculannya
Extension	*Step*	*Branching Action*
	-	-

6) Use Case Scenario *priority*

Scenario use case *priority* menggambarkan langkah-langkah melakukan pengurutan *Data* berdasarkan *priority*. *Scenario use case priority* dapat dilihat pada tabel 3.26 :

Tabel 3. 26 *Use Case Scenario priority*

Use case Name	***Prioritry***	
Goal In Context	*Data* terurut berdasarkan *priority*	
Precondition	*Data* sudah dihitung frekuensinya	
Successful End Condition	Sistem berhasil melakukan pengurutan berdasarkan *priority*	
Failed End Condition	Sistem tidak berhasil melakukan pengurutan *data* berdasarkan *priority*	
Actors	Manager	
Triger	Aktor menekan tombol *priority*	
Main Flow	*Step*	*Action*
	1	Aktor melakukan *request priority*
	2	Sistem melakukan proses pengurutan pada setiap transaksi berdasarkan prioritasnya
	3	Sistem menampilkan *data* yang sudah terurut
Extension	*Step*	*Branching Action*
	-	-

7) *Use Case Scenario Asosiasi*

Skenario *use case* proses *assosiasi* menggambarkan langkah-langkah aksi *user* untuk melakukan proses *frequent Itemset*, *rule*, dan melihat hasil *asosiasi*. Skenario *use case* proses asosiasi dapat dilihat pada table 3.27 :

Tabel 3. 27 *Use Case Scenario Assosiasi*

Use case Name	***Assosiasi***	
Goal In Context	Sistem mendapatkan *rule* lalu menampilkan *rule* tersebut menjadi sebuah informasi	
Precondition	telah melakukan proses *preprocessing*	
Successful End Condition	Sistem berhasil mendapatkan *rule* lalu menampilkan *rule* tersebut menjadi sebuah informasi	
Failed End Condition	Sistem tidak berhasil mendapatkan *rule*	
Actors	*User*	
Triger	*User* memasukan nilai *minimum support* dan *minimum confidence* yang valid	
Main Flow	*Step*	*Action*
	1	*User* memasukan nilai minimum *support*
	2	*User* memasukan nilai minimum *confidence*
	3	Sistem melakukan *generate rule*
	4	Sistem melakukan seleksi *rule* sesuai nilai minimum *confidence*
	5	Sistem mendapatkan *rule* yang sesuai dengan minimum *confidence*
	6	Sistem menampilkan hasil seleksi *rule*
Extension	*Step*	*Branching Action*
	1.1	Sistem menampilkan notifikasi nilai *minimum support* yang diinputkan tidak sesuai
	1.2	Sistem menampilkan notifikasi nilai minimum *confidence* yang diinputkan tidak sesuai
	1.3	Sistem tidak mendapatkan *rule* karena nilai minimum *confidence* yang diinputkan oleh user terlalu tinggi

8) Use Case Scenario Import Parameter

Scenario use case import parameter menggambarkan langkah-langkah melakukan *input* nilai *minimum support* dan *minimum confidence* yang dilakukan oleh *user*. Skenario *use case input parameter* dapat dilihat pada tabel3.28 :

Tabel 3. 28 *Use Case Skenario Input Parameter*

Use case Name	***Input Parameter***	
Goal In Context	Sistem mendapatkan nilai *minimum support* dan confidence untuk proses asosiasi	
Precondition	*User* menginputkan nilai *minimum support dan confidence*	
Successful End Condition	Sistem berhasil mendapatkan nilai *support* dan confidence sesuai dari *input*an *user*	
Failed End Condition	Sistem tidak berhasil mendapatkan nilai *support* dan confidence	
Actors	*User*	
Triger	*User* menginputkan nilai *minimum support dan confidence* yang valid	
Main Flow	Step	Action
	1	*User* menginputkan nilai *minimum support dan confidence*
	2	Sistem mendapatkan nilai *minimum support dan confidence*
Extension	*Step*	*Branching Action*
	2.1	Sistem menampilkan notifikasi nilai *minimum support* dan confidenceyang diinputkan tidak sesuai
	2.2	Sistem tidak mendapatkan nilai *minimum support dan confidence* karena nilai yang inputkan oleh *user* tidak sesuai

9) *Use Case Scenario* Analisa

Skenario *use case* analisa menggambarkan langkah-langkah aksi *user* untuk melakukan proses pembuatan *rule*. Skenario *use case analisa* dapat dilihat pada table 3.29 :

Tabel 3. 29 *Use Case Skenario* Analisa

Use case Name	***Analisa***	
Goal In Context	Sistem melakukan pembentukan *rule*	
Precondition	Sudah melewati proses *input nilai minimum support dan confidence*	
Successful End Condition	Sistem berhasil melakukan pembentukan *rule*	
Failed End Condition	Sistem tidak berhasil melakukan pembentukan *rule*	
Actors	*User*	
Triger	*User* menginputkan nilai minimum *support* dan *confidence*	
Included Cases	*minimum confidence*	
Main Flow	Step	Action
	1	*User* menginputkan nilai minimum *support* dan *confidence*
	2	Sistem mendapatkan nilai minimum *support* dan *confidence*
	3	Sistem melakukan *generate rule*
	4	Sistem memilah *rule* sesuai dengan minimum *confidence*
	5	Sistem mendapatkan *rule* yang sesuai dengan minimum *confidence*
Extension	Step	Branching Action
	1.1	Sistem menampilkan pesan nilai minimum *confidence* yang diinputkan tidak sesuai
	1.2	Sistem gagal mendapatkan *rule* karena nilai *confidence* yang diinputkan oleh *user* terlalu tinggi

10) *Use case Scenario* Cara Penggunaan

Skenario *use case* cara penggunaan menggambarkan langkah-langkah aksi *user* untuk menampilkan cara menggunakan aplikasi. Skenario *use case* cara penggunaan dapat dilihat pada table 3.30 :

Tabel 3. 30 Skenario *Use Case* Cara Penggunaan

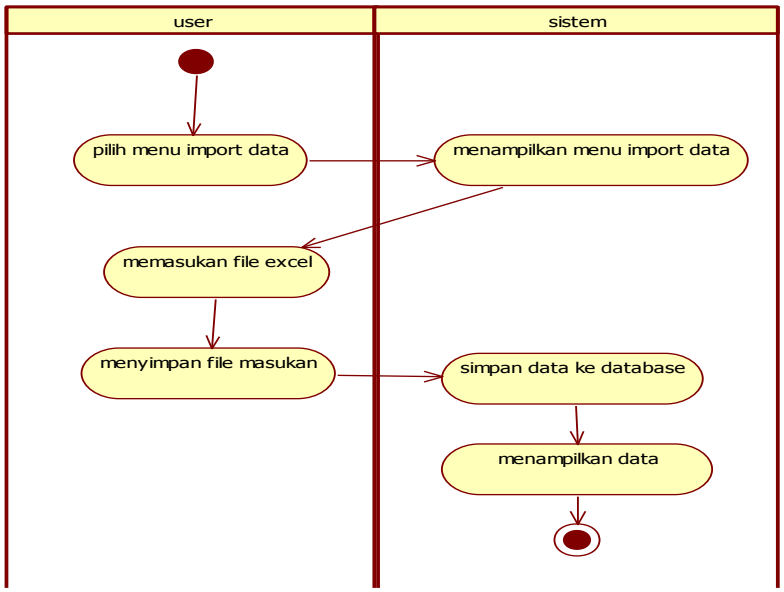
Use case Name	**Cara Penggunaan**	
Goal In Context	Menampilkan acara penggunaan aplikasi	
Precondition	Aktor berada pada menu home	
Successful End Condition	Berhasil menampilkan cara penggunaan	
Failed End Condition	Tidak berhasil menampilkan cara penggunaan	
Actors	User	
Triger	Aktor memilih menu cara penggunaan	
Main Flow	*Step*	*Action*
	1	Aktor memilih menu cara penggunaan
	2	Sistem menampilkan cara penggunaan
Extension	*Step*	*Branching Action*
	-	-

3. *Diagram Activity*

Diagram activity menggambarkan rangkaian aliran dari aktivitas dari setiap proses. Aktivitas diagram dari setiap proses dapat dilihat pada gambar-gambar berikut :

1) *Diagram Activity Import Data*

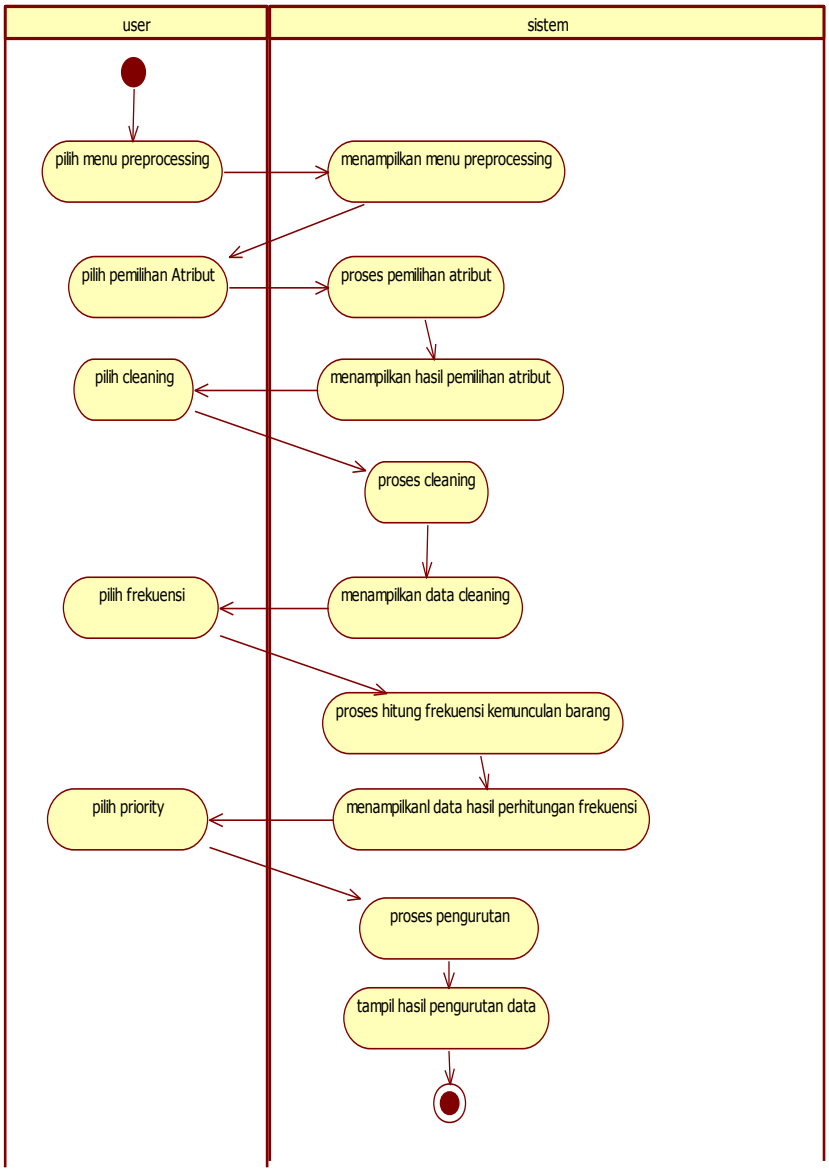
Diagram activity import data menjelaskan aktivitas aktor dalam melakukan proses *import data*. *Diagram activity import data* dapat dilihat pada gambar 3.93 :



Gambar 3. 93 *Diagram Activity Import Data*

2) *Diagram Activity Prepocessing*

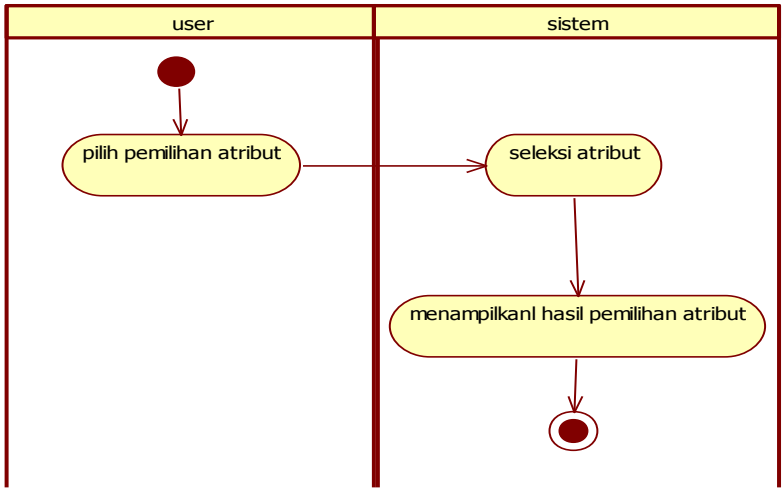
Diagram Activity preprocessing menjelaskan aktivitas sistem dalam melakukan proses pengolahan *data*. *Diagram activity preprocessing* dapat dilihat pada gambar 3.94 :



Gambar 3. 94 *Diagram Activity Prepocessing*

3) *Diagram Activity* Pemilihan Atribut

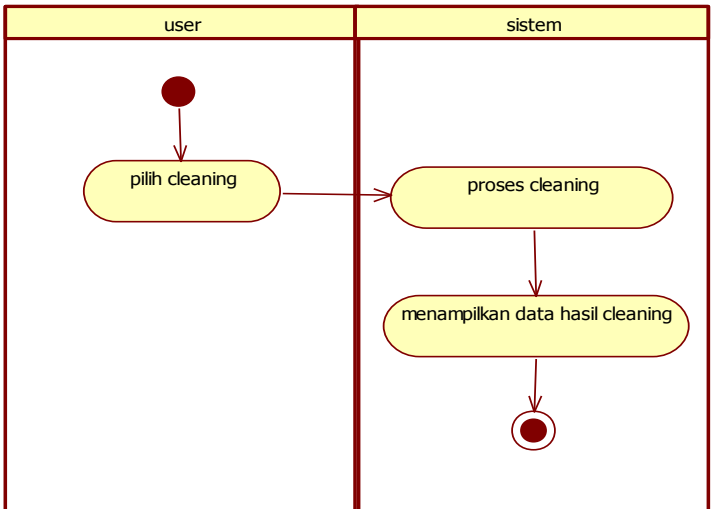
Diagram activity pemilihan atribut menjelaskan aktivitas sistem dalam melakukan pengaturan proses pemilihan atribut. *Diagram activity* pemilihan atributdapat dilihat pada gambar 3.95 :



Gambar 3. 95 *Diagram Activity* pemilihan atribut

4) *Diagram Activity Cleaning*

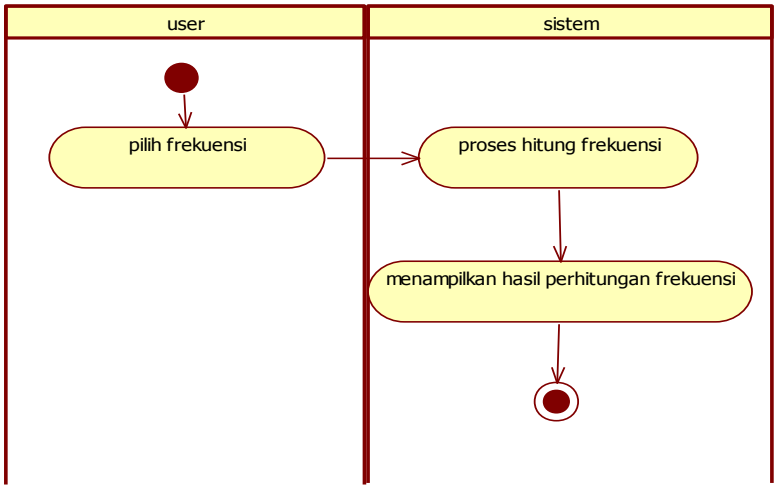
Diagram activity cleaning menjelaskan aktivitas sistem dalam proses pembersihan *data*. *Diagram activity cleaning* dapat dilihat pada gambar 3.96 :



Gambar 3. 96 *Diagram Activity Cleaning*

5) *Diagram Activity* Frekuensi

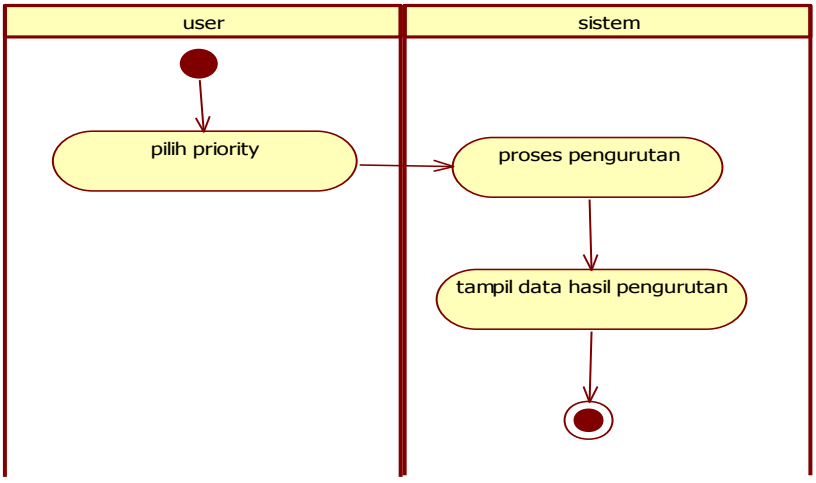
Diagram activity Frekuensi menjelaskan aktivitas sistem dalam melakukan proses Penghitungan jumlah kemunculan tiap *Item*. *Diagram activity* Frekuensi dapat dilihat pada gambar 3.97 :



Gambar 3. 97 *Diagram Activity* Frekuensi

6) *Diagram Activity Priority*

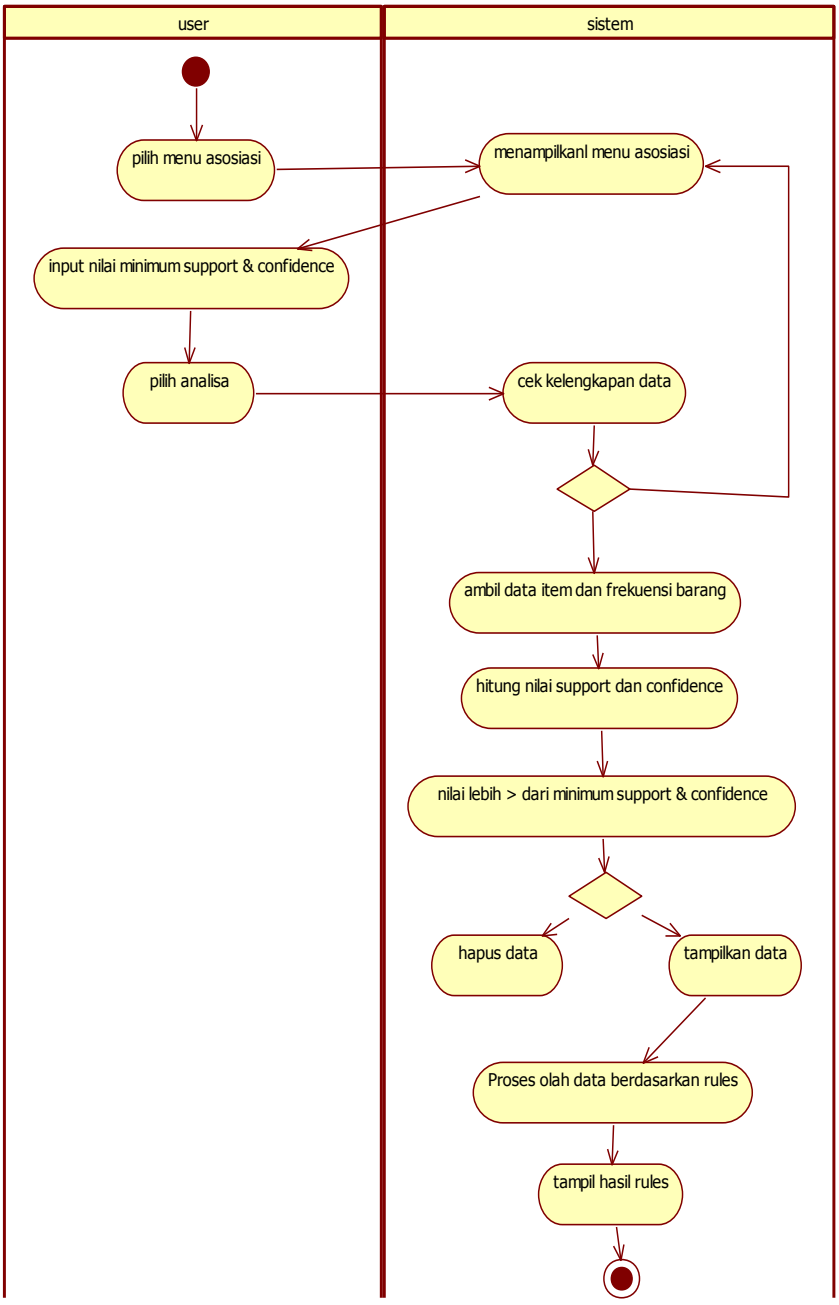
Diagram activity Priority menjelaskan aktivitas sistem dalam melakukan proses Pengurutan *Data* berdasarkan *priority*. *Diagram activity* Pengurutan *data* dapat dilihat pada gambar 3.98 :



Gambar 3. 98 *Diagram Activity Priority*

7) *Diagram Activity* assosiasi

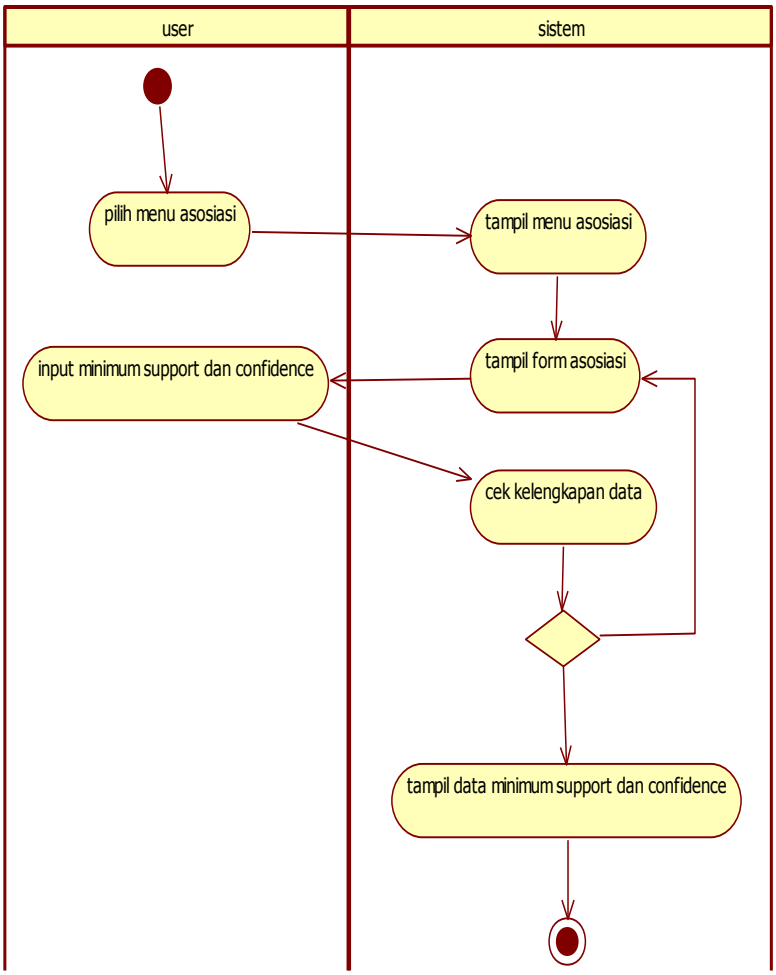
Diagram activity proses *asosiasi* menjelaskan aktivitas *user* dan sistem dalam melakukan proses pencarian *frequent Itemset* , *rule* dan melihat hasil *asosiasi. Diagram Aktivitas* proses *asosiasi* dapat dilihat pada gambar 3.99 :



Gambar 3. 99 *Diagram Activity Assosiasi*

8) *Diagram Avtivity Input Parameter*

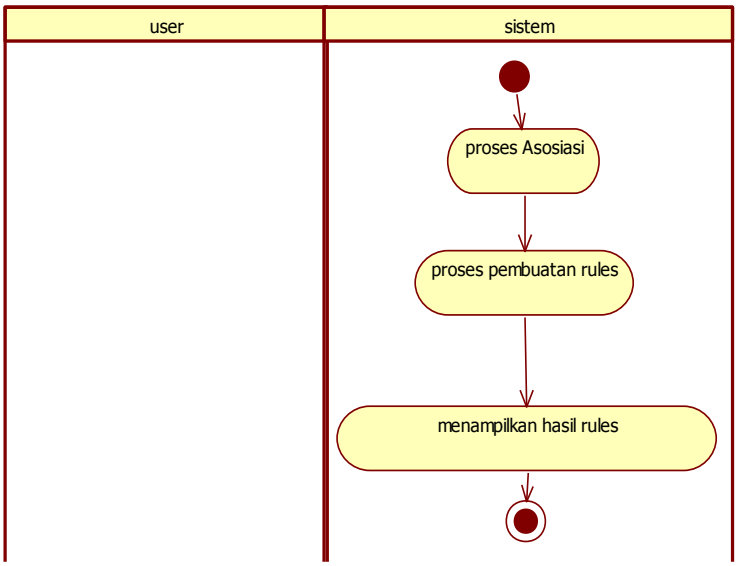
Diagram activity Input parameter menjelaskan aktivitas *user* dan sistem dalam melakukan proses input nilai *minimum support dan confidence*. *Diagram activity input parameter* dapat dilihat pada gambar 3.100 :



Gambar 3. 100 *Diagram Activity Input Parameter*

9) *Diagram Activity* Analisa

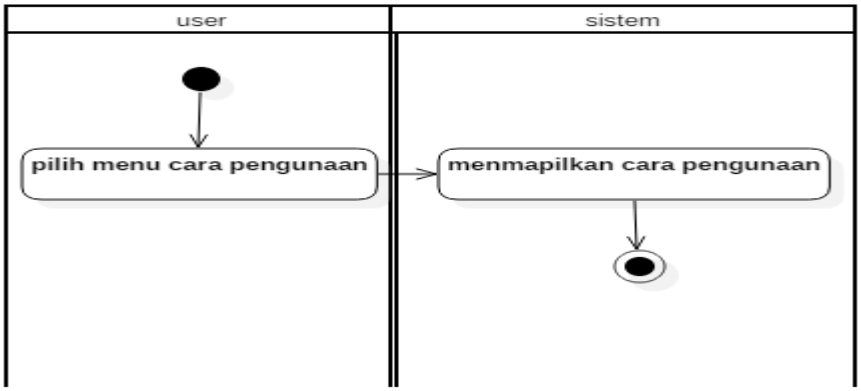
Diagram activity Analisa menjelaskan aktivitas sistem dalam melakukan proses menemukan *rule. Diagram activity* Analisa dapat dilihat pada gambar 3.101 :



Gambar 3. 101 *Diagram activity* Analisa

10) *Diagram activity* Cara Pengunaan

Diagram activity cara penggunaan menjelaskan aktivitas sistem dalam menampilkan cara penggunaan aplikasi. *Diagram activity* cara penggunaan dapat dilihat pada gambar 3.102:

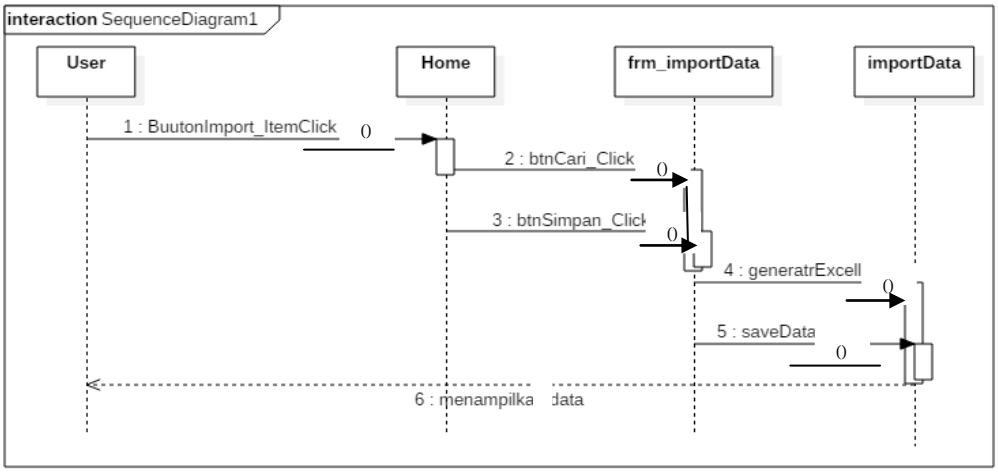


Gambar 3. 102 *Diagram Activity* Cara Penggunaan

4. *Diagram Sequence*

   1. *Diagram Sequence Import Data*

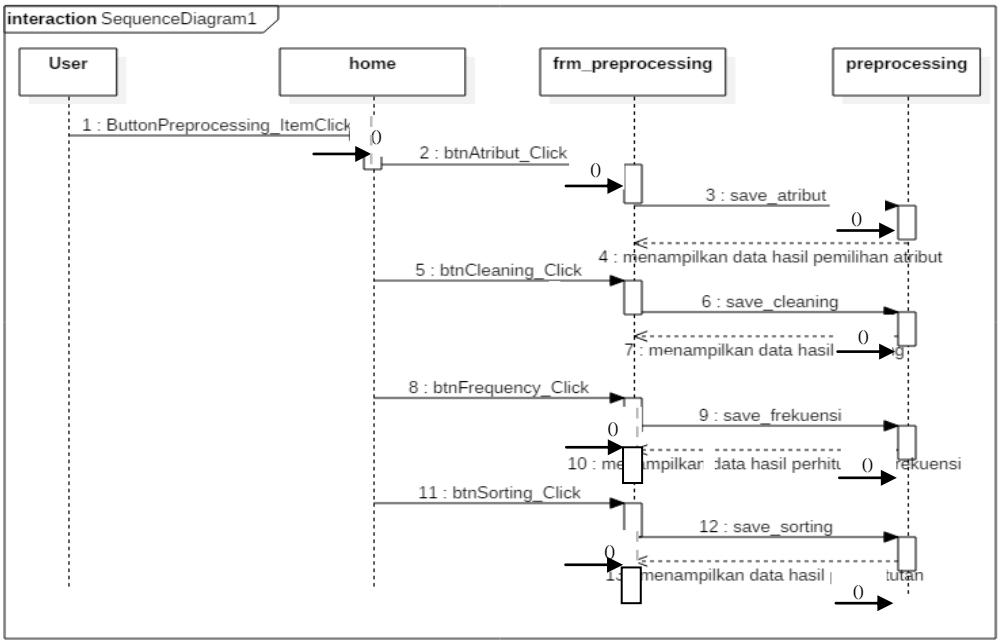
      Diagram Sequence Import Data dapat dilihat pada gambar 3.103 :



Gambar 3. 103 *Diagram Sequence Sequence Import Data*

   2. *Sequence Diagram Preprocessing*

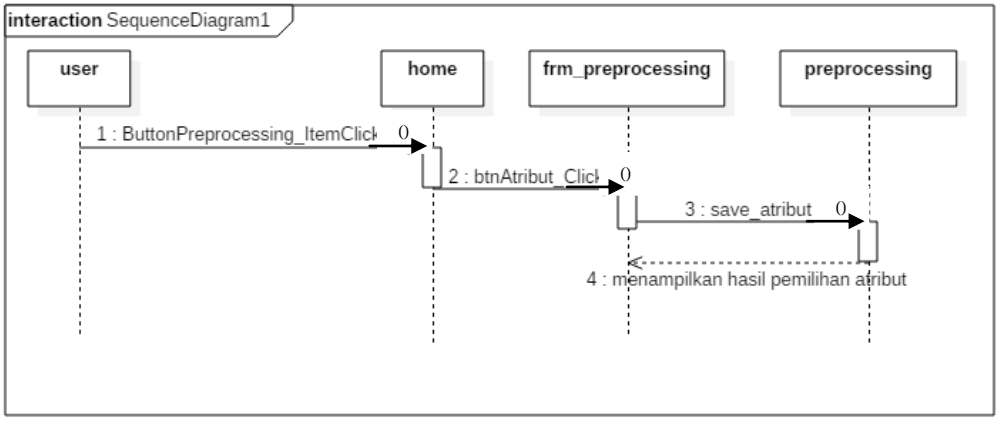
      Diagram Sequence Prepocessing dapat dilihat pada gambar 3.104 :



Gambar 3. 104 *Diagram Sequence Prepossesing*

a. *Diagram Sequance* Pemilihan Atribut

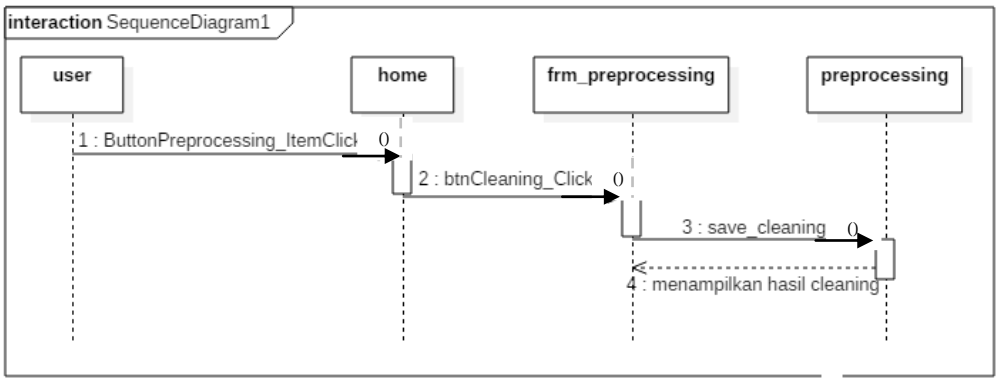
Diagram Sequence pemilihan atribut dapat dilihat pada gambar 3.105 :



Gambar 3. 105 ***Diagram Sequence Pemilihan Atribut***

b. *Diagram Sequance Cleaning*

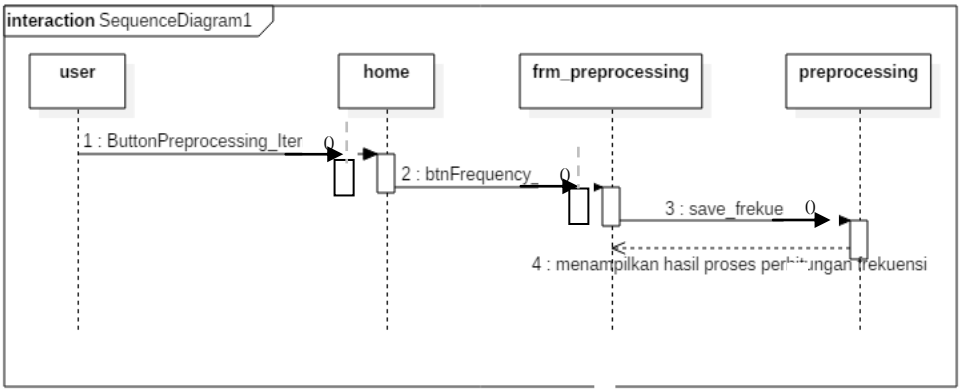
Diagram Sequence Cleaning dapat dilihat pasda gambar 3.106 :



Gambar 3. 106 ***Diagram Sequence Cleaning***

c. *Diagram Sequence* Frekuensi

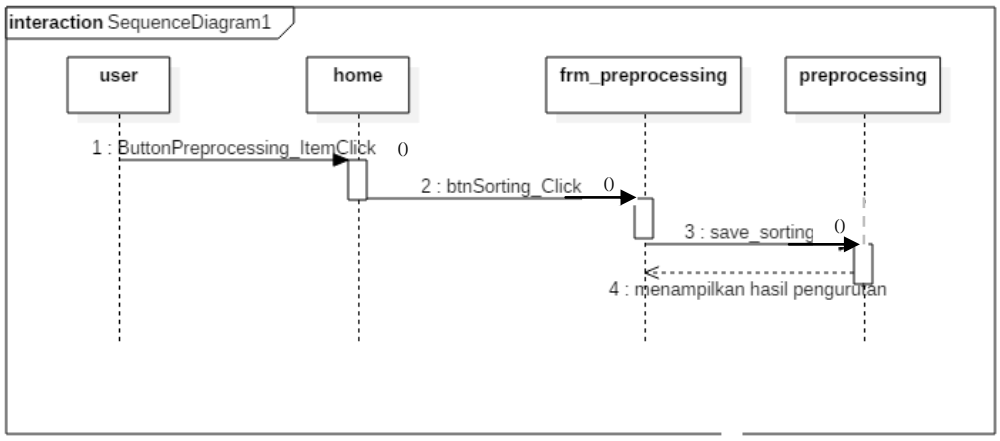
Diagram Sequence frekuensi dapat dilihat pada gambar 3.107 :



Gambar 3. 107 ***Diagram Sequence Hitung Frequent Item***

d. *Diagram Sequence Priority*

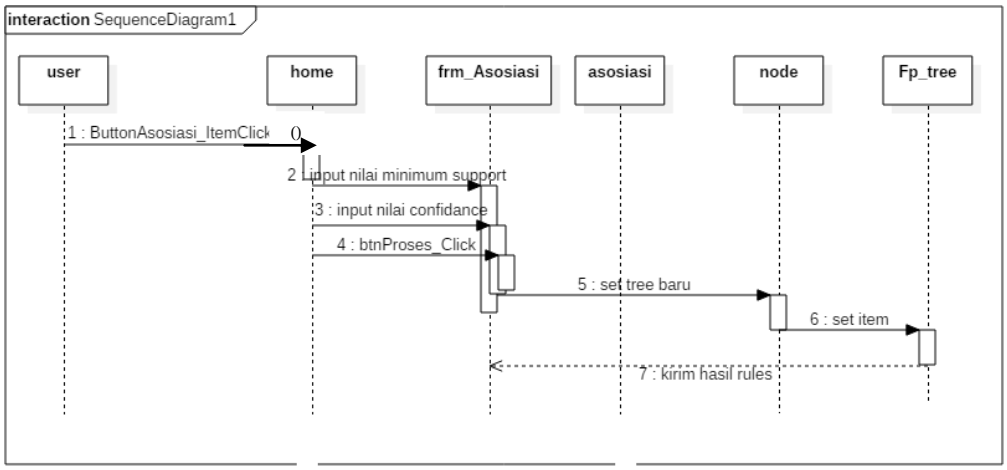
Diagram Sequence Priority dapat dilihat pada gambar 3.108 :



Gambar 3. 108 ***Diagram Sequence Proses Priority***

3. *Diagram Sequence Asosiasi* :

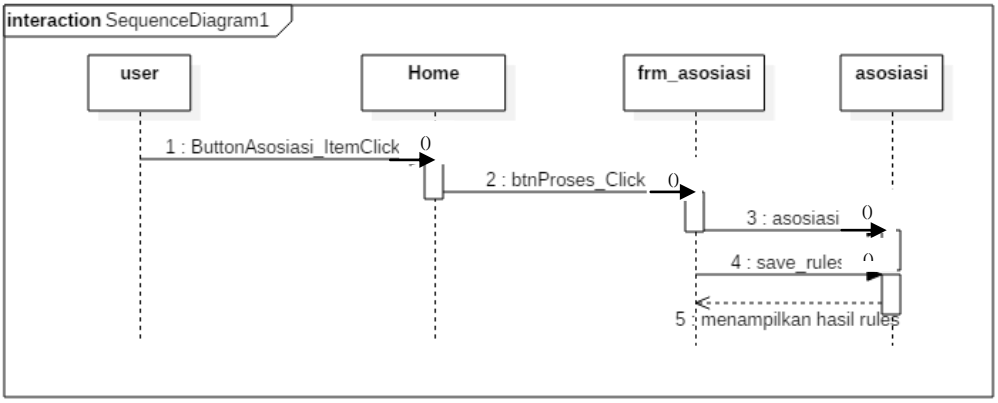
   Diagram Sequence assosiasi dapat dilihat pada gambar 3.109 :



Gambar 3. 109 *Diagram Sequence proses Asosiasi*

a. Diagram Sequence Analisa

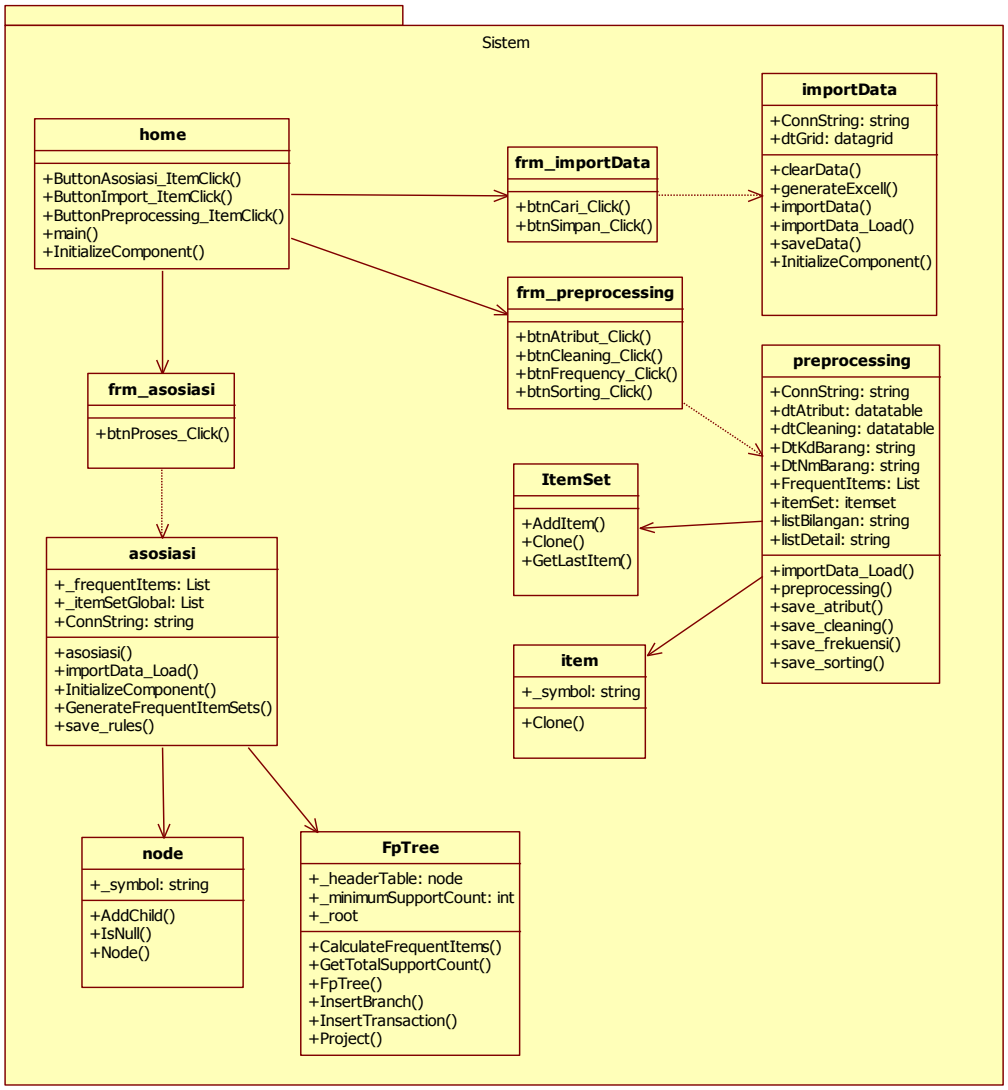
Diagram Sequence Analisa dapat dilihat pada gambar 3.110 :



Gambar 3. 110 *Diagram Sequence Analisa*

5. *Diagram Class*

Diagram Class dapat dilihat pada gambar 3.111 :



Gambar 3. 111 Diagram *Class*

3.2 Perancangan Sistem

Proses perancangan yang akan dilakukan pada tahapan ini adalah perancangan sistem basis *data*, perancangan stuktur menu, perancangan antar muka, perancangan pesan, jaringan semantik.

3.2.1 Perancangan *Class*

Berikut ini adalah perancangan *class* yang digunakan pada sistem yang akan dibangun :

Tabel 3. 31 Struktur *ClassHome*

Nama *Class* : home		
Nama	Jenis	Type
home()	Method	Public
ButtonAsosiasi_*Item*Click()	Method	Public
ButtonImport_*Item*Click()	Method	Public
ButtonPrepocessing_*Item*Click()	Method	Public
Main()	Method	Public
initializeComponent()	Method	Public

Tabel 3. 32 Struktur *Class frm_import Data*

Nama *Class* :frm_import*Data*		
Nama	Jenis	Type
btnCari_Click()	Method	Public
btnSimpan_Click()	Method	Public

Tabel 3. 33 Struktur *Class frm_preprocessing*

Nama *Class* : frm_preprocessing		
Nama	Jenis	Type
btnAtribut_Click()	Method	Public
btn*Diagram Activity* _Click()	Method	Public
btnFrequency_Click()	Methode	Public
btnSorting_Click()	Mrthode	Public

Tabel 3. 34 Struktur *Class frm_asosiasi*

Nama *Class* : frm_asosiasi		
Nama	Jenis	Type
btnProses_Click()	Method	Private

Tabel 3. 35 Struktur *Class import Data*

Nama *Class* : import*Data*		
Nama	Jenis	Type
clear*Data*()	Method	Public
generateExcell()	Method	Public
import*Data*()	Method	Public
import*Data*_Load()	Method	Public
save*Data*()	Method	Public
InitializeComponent()	Method	Public
ConnString: string	Atribut	Public
dtGrid: *Data*grid	Atribut	Public

Tabel 3. 36 Struktur *Class Preprocessing*

Nama *Class* : *preprocessing*		
Nama	Jenis	Type
importData_Load()	*Method*	*Public*
Prepocessing()	*Method*	*Public*
Save_atribut()	*Method*	*Public*
Save_Diagram Activity ()	*Method*	*Public*
Save_frekuensi()	*Method*	*Public*
Save_sorting()	*Method*	*Public*
ConnString: string	*Atribut*	*Public*
dtAtribut: Datatable	*Atribut*	*Public*
dtDiagram Activity : Datatable	*Atribut*	*Public*
dtKdBarang: string	*Atribut*	*Public*
DtNmBarang: string	*Atribut*	*Public*
FrequentItems: List	*Atribut*	*Public*
ItemSet: Itemset	*Atribut*	*Public*
listBilangan: string	*Atribut*	*Public*
listDetail: string	*Atribut*	*Public*

Tabel 3. 37 Struktur *Class ItemSet*

Nama *Class* : *Item*Set		
Nama	Jenis	Type
AddItemi()	*Method*	*Public*
Clone()	*Method*	*Public*
GetLastItem	*Method*	*Public*

Tabel 3. 38 Struktur *Class Item*

Nama *Class* : *Item*		
Nama	Jenis	Type
Clone()	*Method*	*Public*
_symbol:string	*Atribut*	*Public*

Tabel 3. 39 Struktur *Class Asosiasi*

Nama *Class* : *Asosiasi*		
Nama	Jenis	Type
Asosiasi()	*Method*	*Public*
importData_Load()	*Method*	*Public*
initializeComponent()	*Method*	*Public*
generateFrequentItemSet()	*Method*	*Public*
Save_rule()	*Method*	*Public*
_frequentItems:List	*Atribut*	*Public*
_ItemSetGlobal:List	*Atribut*	*Public*
ConnString: string	*Atribut*	*Public*

Tabel 3. 40 Struktur *Class Node*

Nama *Class : Node*		
Nama	Jenis	Type
AddChild()	*Method*	*Public*
IsNull()	*Method*	*Public*
Node()	*Method*	*Public*
symbol string	*Atribut*	*Public*

Tabel 3. 41 Struktur *Class Fp-Tree*

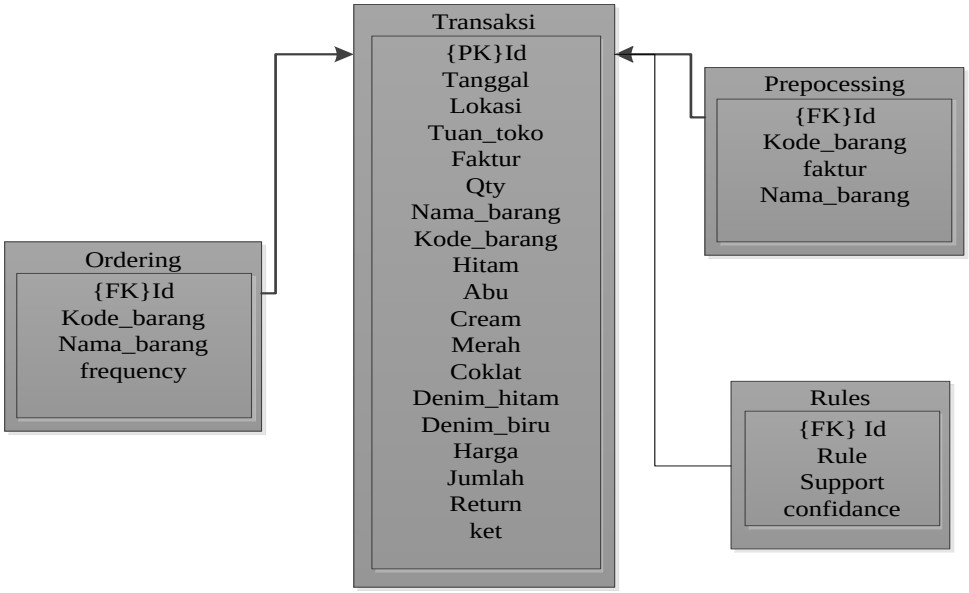
Nama *Class* : *Fp Tree*		
Nama	Jenis	Type
CalculateFrequentItems()	*Method*	*Public*
GetTotalSupportCount()	*Method*	*Public*
FpTree()	*Method*	*Public*
InsertBrach()	*Method*	*Public*
insertTransaction()	*Method*	*Public*
_Header Table: node	*Atribut*	*Public*
_minimumSupportCount:int	*Atribut*	*Public*
_root	*Atribut*	*Public*

3.2.2 Perancangan Basis *Data*

Perancangan basis *data* terhadap aplikasi yang akan dibangun meliputi struktur tabel.

3.2.1.1 Skema Relasi

Skema relasi dari aplikasi yang akan dibangun dapat dilihat pada gambar dibawah ini :



Gambar 3. 112 Skema Relasi

3.2.1.2 Struktur Tabel

Struktur table pada aplikasi yang akan dibangun terdiri dari 4 tabel, rinciannya sebagai berikut,dapat dilihat pada tabel 3.42 sampai tabel 3.35 :

Tabel 3. 42 Struktur Tabel *data* Transaksi

Nama Field	Tipe *Data*	Panjang	Keterangan
Id	Int	11	PK
Tanggal	Varchar	20	
Lokasi	Varchar	20	
Tuan/Toko	Varchar	50	
Faktur	Varchar	20	
Qty	Int	11	
Nama Barang	Varchar	50	
Kode Barang	Varchar	20	
Hitam	Int	20	
Abu	Int	20	
Cream	Int	20	
Merah	Int	11	
Coklat	Int	11	
Denim Hitam	Int	11	
Denim Biru	Int	11	
Harga	Int	11	
Jumlah	Int	11	
Return	Int	11	
Ket	Text		

Tabel 3. 43 Tabel *Ordering*

Nama Field	Tipe *Data*	Panjang	Keterangan
Id	Int	11	FK references Transaksi(Id)
Kode_barang	Varchar	10	
Nama_barang	Varchar	100	
frequency	Int	11	

Tabel 3. 44 Tabel *Preprocessing*

Nama Field	Tipe *Data*	Panjang	Keterangan
Id	Int	11	FK references Transaksi(Id)
Kode_barang	Varchar	10	
faktur	Varchar	50	
Nama_barang	Varchar	100	

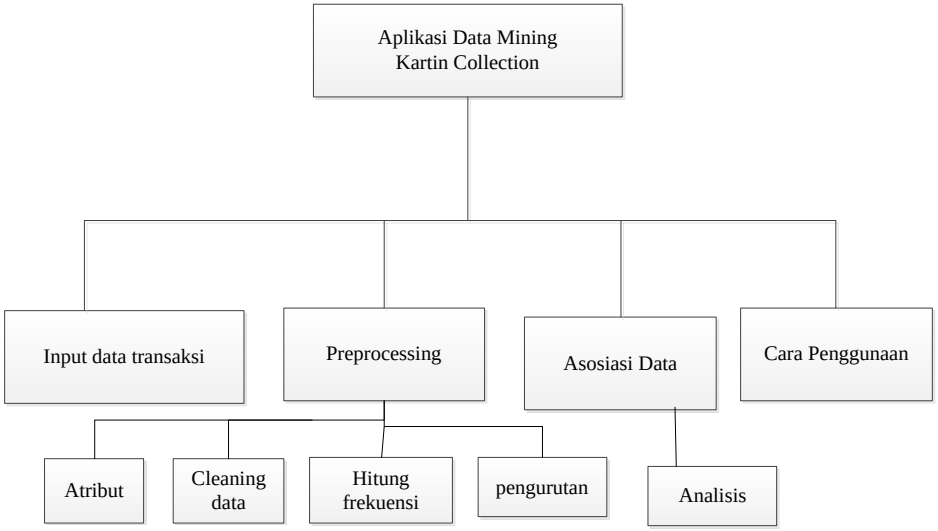
Karena dibutuhkannya sebuah tempat penyimpanan untuk hasil pencarian *frequent pattern*, maka dibuat sebuah tabel *rule*

Tabel 3. 45 Tabel *Rule*

Nama Field	Tipe *Data*	Panjang	Keterangan
Id	Int	11	FK references Transaksi(Id
Rule	Text	-	
Support	Int	11	
Confidence	Int	11	

3.2.2 Perancangan Struktur Menu

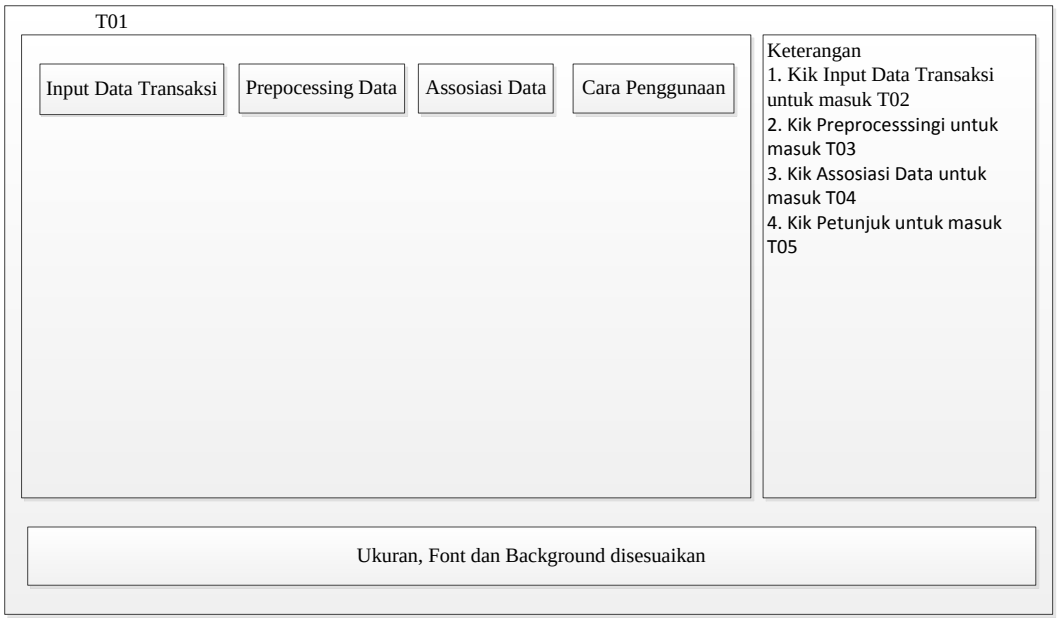
Dalam pembangunan perangkat lunak diperlukan struktur menu untuk memudahkan pengguna dalam menavigasi *fungsional* sistem. Adapun perancangan struktur menu untuk aplikasi yang akan dibangun dapat dilihat pada gambar berikut :



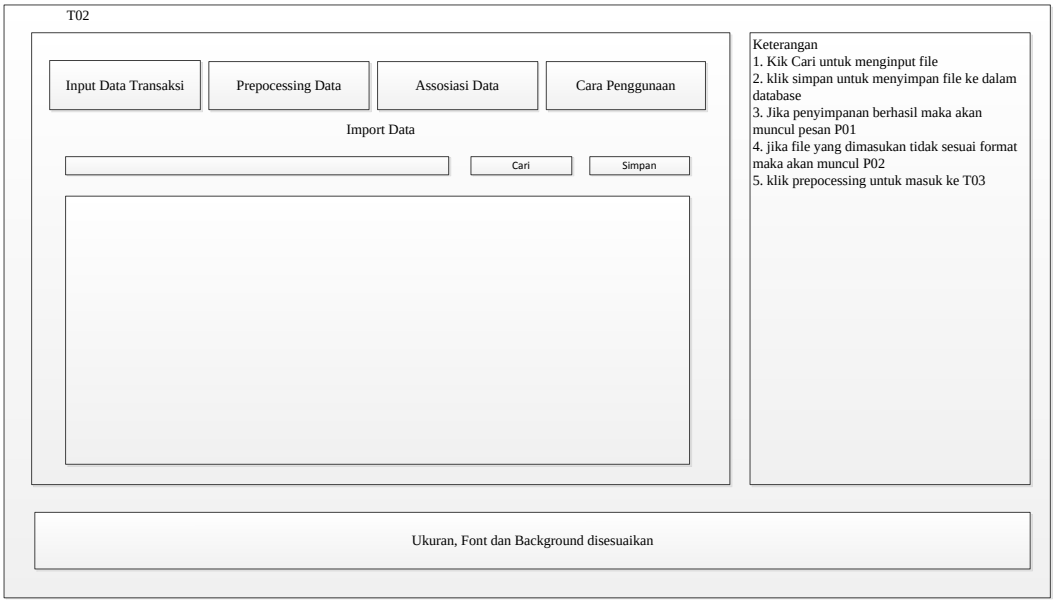
Gambar 3. 113 Struktur Menu

3.2.3 Perancangan Antar Muka

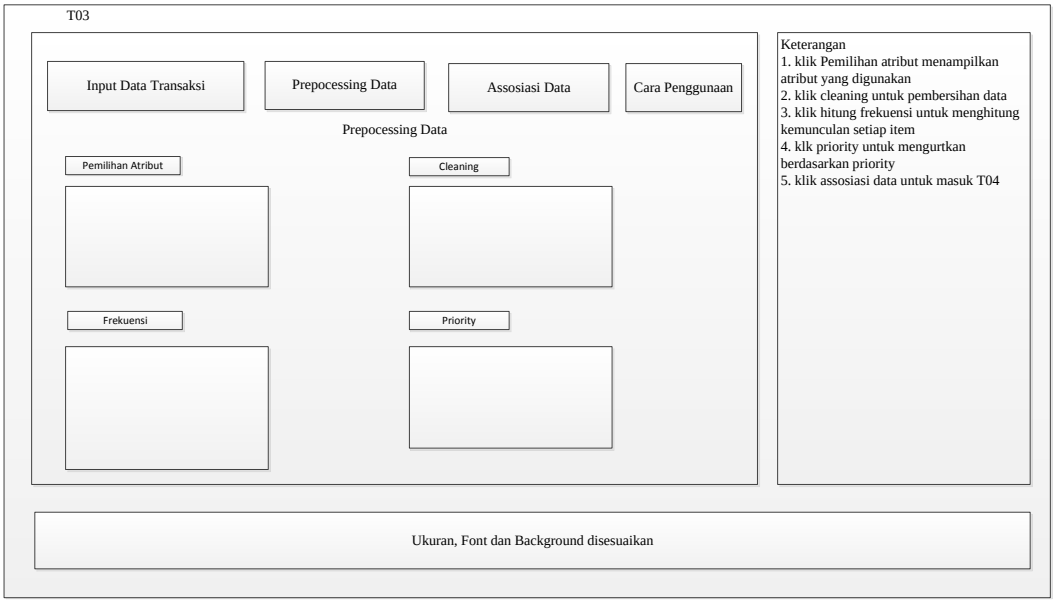
Perancangan antar muka berfungsi untuk memberi gambaran tentang pernagkat lunak yang akan dibangun, supaya lebih memudahkan dalam pembuatan perangkat lunak nantinya. Adapun perancangan antar muka untuk perangakt lunak yang akan dibangun akan dijelaskan berikut ini :



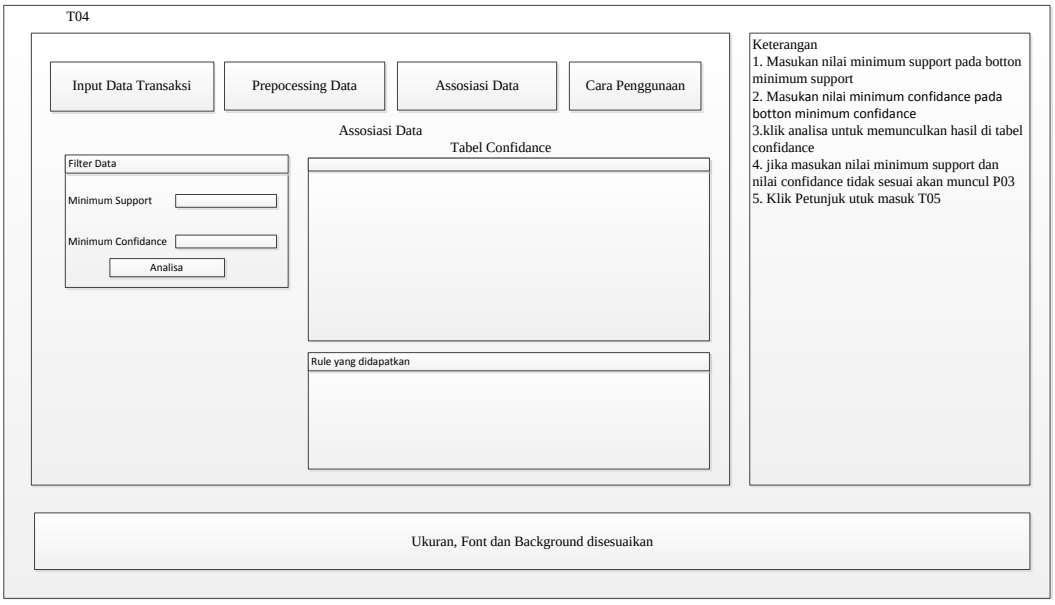
Gambar 3. 114 Perancangan Antarmuka Menu Utama



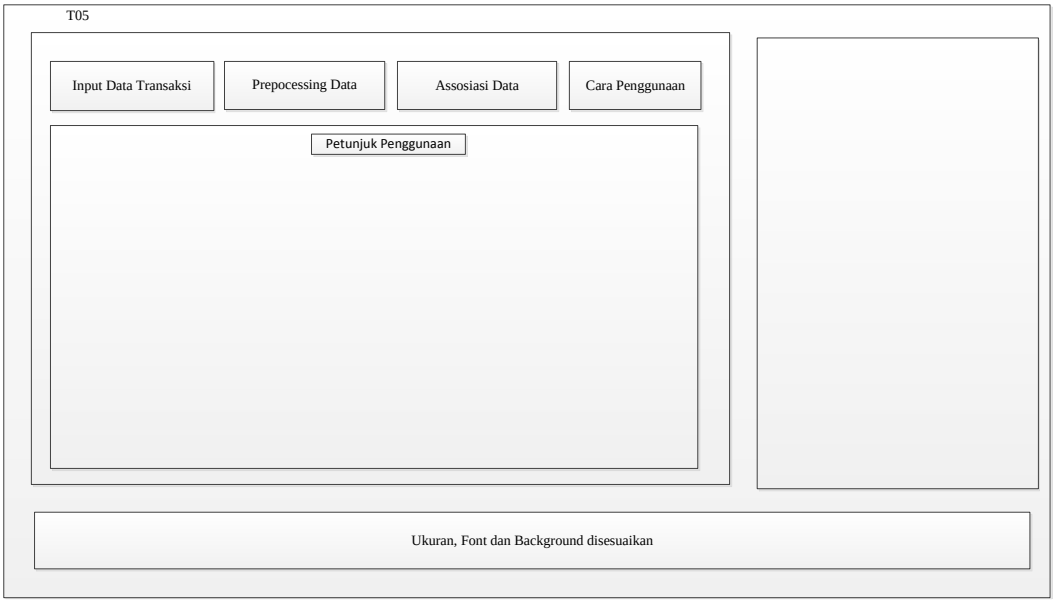
Gambar 3. 115 Perancangan Antarmuka *Import Data*



Gambar 3. 116 Perancangan Antarmuka *Preprocessing*



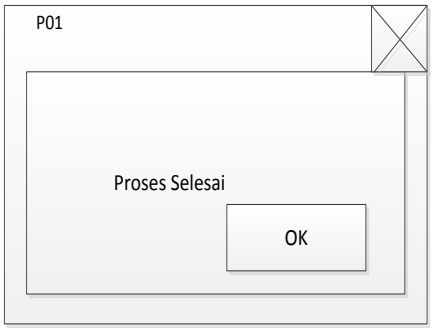
Gambar 3. 117 Perancangan Antar Muka Aturan*Asosiasi*

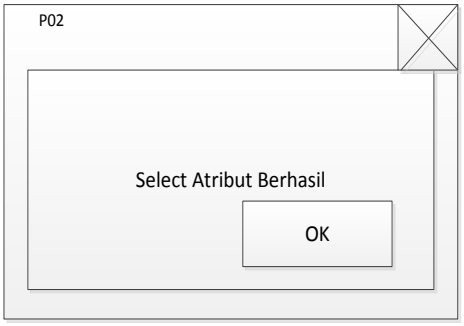


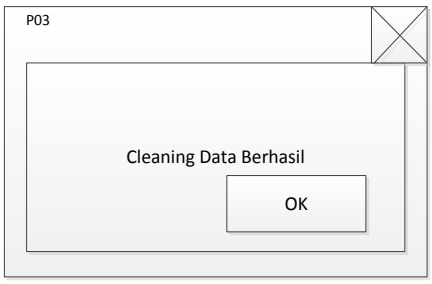
Gambar 3. 118 Perancangan Antar Muka Cara Penggunaan

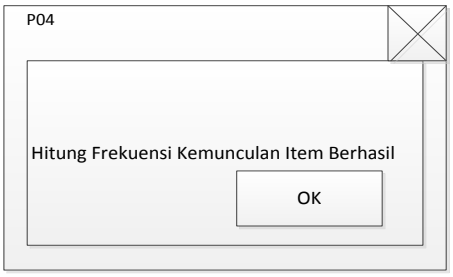
3.2.4 Perancangan Pesan

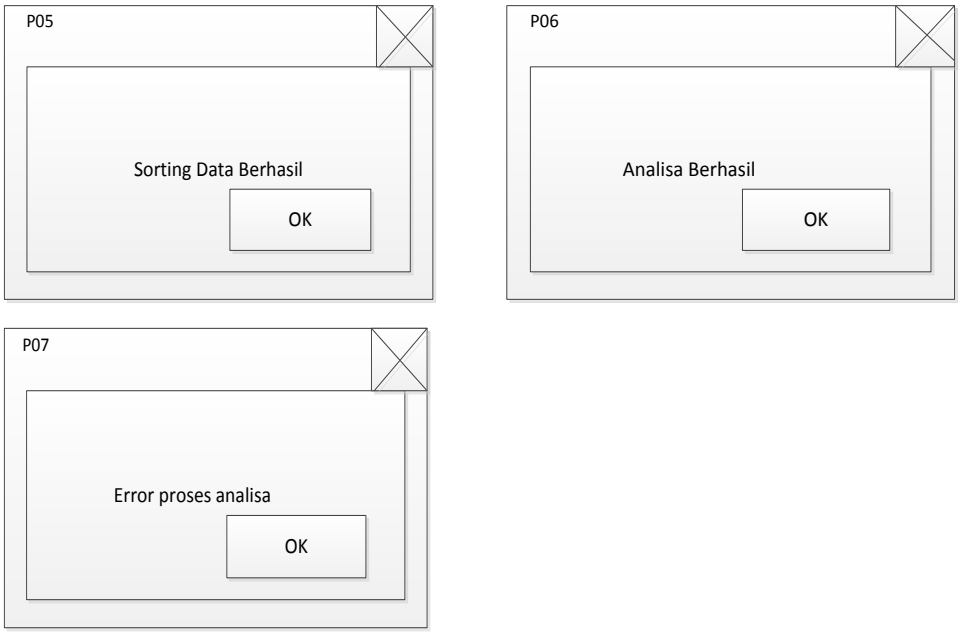
Adapun perancangan pesan dapat dilihat pada gambar dibawah ini :







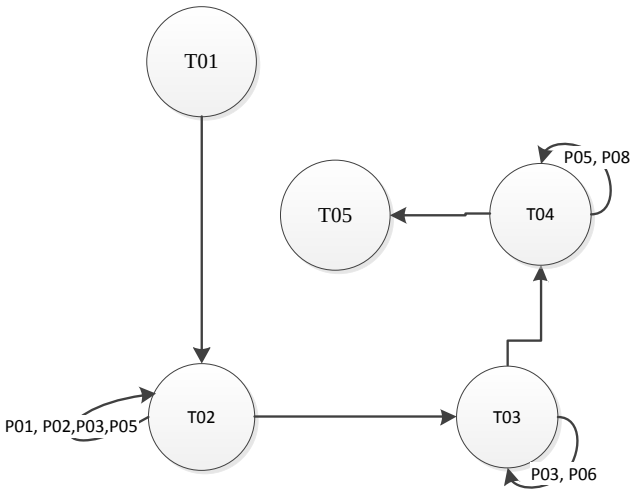




Gambar 3. 119 Perancangan Pesan

3.2.5 Jariangan Semantik

Jaringan semantik bertujuan untuk merepresentasikan pemetaan *Data* untuk mencegak duplikasi *data*. Adapun jaringan semantik yang menghubungkan antarmuka pada aplikasi yang akan dibangun dapat dilihat pada gambar berikut ini :



Gambar 3. 120 Jaringan Semantik