BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisis sistem adalah penguraian dari suatu sistem 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. Tahap analisis sistem merupakan tahapan yang sangat kritis dan penting, karena kesalahan di dalam tahap ini akan menyebabkan juga keselahan di tahap selanjutnya. Tugas utama analisis sistem dalam tahap ini adalah menemukan kelemahan-kelemahan dari sistem yang berjalan sehingga dapat diusulkan perbaikannya. Dalam analisa sistem ini meluputi beberapa bagian, yaitu :

1. Analisis Masalah
2. Analisis Prosedur Penempatan Penyusunan Layut Makanan
3. Analisis Prosedur Penempatan Penentuan Paket Makanan Hemat
4. Analisis Penerapan Metode CRISP-DM
5. Analisis Spesifikasi Kebutuhan Perangkat Lunak

3.1.1 Analisis Masalah

Analisis masalah yang dilakukan adalah adalah melihat dan mengidentifikasikan permasalahan atau kendala di dalam penelitian yang dilakukan. Masalah dapat didefinisikan sebagai suatu pertanyaan yang diinginkan untuk dipecahkan. Perlu ada analisis masalah melalui rumusan masalah yang sudah ditentukan sebelumnya. Berdasarkan hasil pengamatan, diketahui bahwa analisis masalahnya adalah sebagai berikut :

1. Rumah makan Roso Echo masih memiliki kesulitan dalam penyusunan *layout* makanan prasmanan yang disajikannya, dikarenakan penyusunan *layout* makanan masih dilakukan secara acak dan belum adanya aturan dalam penyusunan *layou*t makanan prasmanan ini.

2. Belum adanya informasi penentuan paket makanan hemat yang dapat ditawarkan kepada konsumen, guna meminimalisir antrian panjang pada waktu-waktu tertentu.

3.1.2 Prosedur Penempatan Penyusunan Layout Makanan

3.1.2.1 Prosedur Penempatan Menu Pokok

Berdasarkan obervasi ke rumah makan Roso Echo bahwa prosedur atau alur penempatan menu pokok adalah sebagai berikut :

1. Menu pokok yang sudah siap disajikan atau dimasak akan ditempatkan berdasarkan tempat menu menu makan yang sudah tersedia secara acak
2. Proses dalam penempatan menu pokok tersebut mempunyai tahap yaitu pada saat menu pokok sudah siap disajikan atau dimasak, bagian pelayanan akan bertanggung jawab untuk mengantarkan makanan tersebut ke tempat menu menu makanan yang sudah tersedia.

3.1.2.2 Prosedur Penempatan Menu Sampingan

Menu sampingan terdiri dari satu kategori yaitu jenis lain-lain. Jenis lain-lain ini terdiri dari nasi, nasi bakar,dan makanan-makanan ringan lainnya. Adapun prosedur atau alur penempatan menu pokok adalah sebagai berikut :

1. Dalam proses penempatan atau alur dari menu sampingan berbeda-beda.
2. Pada proses penempatan kategori makanan seperti nasi sudah tersedia dan diletakkan di bagian menu-menu makanan yang sudah tersedia secara acak.
3. Proses penempatan kategori makanan ringan sudah tersedia di meja makan yang sudah disiapkan buat para konsumen.

3.1.3 Prosedur Penempatan Penentuan Paket Makanan Hemat

Berdasarkan wawancara ke pihak pengelola rumah makan Roso Echo bahwa proses yang sedang berjalan sekarang adalah hanya proses prosedur penempatan menu pokok dan menu sampingan. Untuk proses prosedur penempatan penentuan paket makanan hemat belum ada. Berdasarkan data yang

ada bahwa penentuan paket makanan hemat yang ditawarkan ke konsumen belum ada, dikarenakan masalah harga yang mungkin menjadi salah satu penyebabnya. Untuk mengatasi antrian panjang konsumen tersebut pihak pengelola rumah makan Roso Echo akan menawarkan penentuan paket makanan hemat kepada konsumen tetapi dengan harga yang tidak terlalu mahal. Adapun prosedur atau alur proses penempatan penentuan paket makanan hemat yang akan dilakukan oleh pihak pengelola rumah makan Roso Echo adalah sebagai berikut :

1. Menu makanan yang sudah siap disajikan atau yang dimasak, bagian catering akan bertanggung jawab untuk menyiapkan segala keperluan yang akan dibutuhkan.
2. Pada proses penempatan paket makanan hemat mempunyai tahap-tahap yaitu setelah bagian catering sudah menyiapkan segala makanan yang di paket, bagian catering akan menyusun paketan tersebut serapih mungkin dan diletakkan di bagian menu menu makanan lainnya dengan berbentuk paket.

3.1.4 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).

3.1.4.1 Pemahaman Bisnis

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

1. Tujuan Bisnis

   Dalam proses bisnisnya rumah makan Roso Echo mempunyai tujuan bisnis yaitu untuk memenuhi permintaan konsumen sehingga dapat meningkatkan kepuasan dan pendapatan.

2. Penentuan Sasaran *Data Mining*

   Tujuan dari penerapan data mining pada penjualan menu menu makanan ini adalah untuk mengetahui bagaimana pola pembelian dari kosumen dan makanan apa saja yang sering dibeli secara bersamaan.

3.1.4.2 Pemahaman Data

Berdasarkan tujuan bisnis rumah makan Roso Echo, data yang digunakan dalam penelitian ini adalah data transaksi yang terjadi di rumah makan Roso Echo selama tahun 2014 yang berasal dari *file microsoft* excel (.xlsx). Untuk lebih jelasnya dapat dilihat di Tabel A-1 pada lampiran A. Atribut yang terdapat pada laporan transaksi penjualan terdiri dari :

1. No
2. NoTable
3. NoBill
4. Date
5. Time
6. Server
7. Cetak
8. NamaMenu
9. Kategori
10. Qty
11. Harga
12. SubTotal
13. Pajak
14. GrandTotal
15. Cash
16. ChangeDue

Berikut merupakan penjelasan dari masing-masing atribut yang terdapat pada tabel data transaksi yang dapat dilihat pada tabel 3.1 dibawah.

Tabel 3. 1 Struktur Data Transaksi

Dokumen	Keterangan		
Detail data transaksi	Deskripsi	Data ini berisi transaksi yang terjadi di rumah makan Roso Echo	
	Fungsi	Untuk mengetahui kenis makanan yang sering diminati oleh konsumen berdasarkan kategorit yaitu makanan pokok dan makanan sampingan	
	Format	Microsoft Excel (.xlsx)	
	Atribut	No	Nomor urut
		NoTable	Nomor meja
		NoBill	Nomor struk
		Date	Tanggal pencetakan strukt
		Time	Waktu pencetakan struk
		Server	Nama kasir
		Cetak	Pencetak struk
		NamaMenu	Nama menu yang dibeli
		Kategori	Kategori menu yang dibeli
		Qty	Jumlah menu yang dibeli
		Harga	Harga dikalikan dengan qty
		SubTotal	Total harga menu yang dibeli
		Pajak	Pajak dari rumah makan
		GrandTotal	Jumlah yang harus dibayar
		Cash	Jumlah uang yang dibayarkan
		ChangeDue	Jumlah uang kembalian

3.1.4.3 Persiapan Data

Tahapan persiapan data harus dilakukan dalam proses *data mining* karena tidak semua *record* atau *atribute* digunakan dalam proses *data mining*. Proses ini

dilakukan agar data yang akan digunakan sesuai dengan kebutuhan. Adapun tahapan-tahapan persiapan data dalam penelitian ini adalah sebagai berikut :

1. Pemilihan Atribut
    a. Pemilihan atribut penyusunan layout makanan (menu pokok yaitu jenis makanan).

        Berdasarkan informasi yang ingin didapat oleh pengguna mengenai makanan apa saja yang dibeli secara bersamaan oleh konsumen, maka dalam tahap ini atribut yang akan digunakan dari tabel D-1 dalam lampiran D hanya NoBill, NamaMenu, Kategori yang bertujuan untuk menyeleksi data apa saja yang relevan dan diperlukan *database* untuk keperluan analisa. Hasil Pemilihan atribut dapat dilihat pada tabel D-4 dalam lampiran D.

    b. Pemilihan atribut penentuan paket makanan hemat (menu pokok dan menu sampingan yaitu jenis makanan, jenis lain-lain dan jenis minuman)

        Berdasarkan informasi yang ingin didapat oleh pengguna mengenai makanan apa saja yang dibeli secara bersamaan oleh konsumen, maka dalam tahap ini atribut yang akan digunakan dari tabel D-1 dalam lampiran D hanya NoBill, NamaMenu, Kategori yang bertujuan untuk menyeleksi data apa saja yang relevan dan diperlukan *database* untuk keperluan analisa. Hasil Pemilihan atribut dapat dilihat pada tabel D-32 dalam lampiran D

2. Pembersihan Data
    a. Pembersihan data penyusunan layout makanan

    Proses menghilangkan *noise* dan menghilangkan data tidak relevan atau inkonsisten disebut pembersihan data. Dalam data transaksi yang sudah ada bahwa data transaksi yang akan digunakan atau diolah sebagai perhitungan adalah menu pokok yang terdiri dari jenis makanan, sedangkan jenis lain-lain dan jenis minum, dihilangkan karena tidak sesuai

dengan tujuan *data* mining yaitu untuk penyusunan layout makanan. Hasil pembersihan data dapat dilihat pada tabel D-8 dalam lampiran D.

b. Pembersihan data penentuan paket makanan hemat

   Dalam penentuan paket makanan hemat atribut atau kategori tidak ada yang dihilangkan, karena semua menu pokok dan meu sampingan yang terdiri dari jenis makanan, jenis lain-lain dan jenis minuman digunakan untuk tahap proses selanjutnya.

Selanjutnya data yang sudah dibersihkan baik itu penyusunan layout makanan dan penentuan paket makanan hemat yang terdapat pada tabel D-8 dalam lampiran D akan diolah lagi sampai menjadi data akhir yang digunakan untuk proses *data mining*.

Adapun tahap-tahap untuk mendapatkan nilai akhir dari setiap pemodelan adalah sebagai berikut :

a. Tahap pemodelan penyusunan layout makanan

- Data yang sudah dibersihkan, akan di sederhanakan (simplikasi), agar data-data makanan yang sama dijadikan dalam satu nama makanan saja.dapat dilihat pada tabel D-11 dalam lampiran D
- Untuk proses berikutnya atribut NamaMenu akan diubah menjadi KodeMenu berbentuk integer yang akan mempermudah saat proses *mining.* Dapat dilihat pada tabel D-12 dalam lampiran D.
- Pengurutan data (*ascending*) untuk mempermudah pembacaan. Hasil Pengurutan dapat dilihat pada tabel D-13 dalam lampiran D.
- Setelah proses pengurutan dilakukan, proses terakhir adalah proses untuk persediaan data (data akhir), dapat dilihat pada tabel D-14 dalam lampiran D

b. Tahap pemodelan penentuan paket makanan hemat

- Data yang sudah dipilih atributnya akan di sederhanakan (simplikasi), agar data-data jenis makanan, jenis lain-lain, dan jenis minuman yang sama dijadikan dalam satu nama, dapat dilihat pada tabel D-36 dalam lampiran D.
- Untuk proses berikutnya atribut NamaMenu akan diubah menjadi KodeMenu berbentuk integer yang akan mempermudah saat proses *mining*. Dapat dilihat pada tabel D-37 dalam lampiran D.
- Pengurutan data (*ascending*) untuk mempermudah pembacaan. Hasil Pengurutan dapat dilihat pada tabel D-39 dalam lampiran D
- Setelah proses pengurutan dilakukan, proses terakhir adalah proses untuk persediaan data (data akhir), dapat dilihat pada tabel D-41 dalam lampiran D

3. Penyiapan Data Awal

   Setelah semua proses penyusunan layout makanan dan penentuan paket makanan hemat berhasil dilakukan dan data transaksi telah sesuai dengan kebutuhan yang diperlukan dalam proses *data mining*, maka data transaksi sudah dapat digunakan untuk proses selanjutnya dalam sistem *data mining*.

3.1.4.4 Pemodelan

3.1.4.4.1 Pemodelan Untuk Penyusunan Layout Makanan

Merupakan tahap untuk membuat model atau desain sistem yang akan dibangun. Data yang digunakan untuk pengolahan data adalah data menu pokok yaitu jenis makanan. Di dalam tahap ini akan dipilih metode yang cocok untuk menyelesaikan masalah di rumah makan Roso Echo. Metode yang akan digunakan adalah *association rule* dengan algoritma *fp-growth*.

Data yang dipakai untuk penyusunan layout makanan adalah pada tabel D-14 dalam lampiran D. Data tersebut akan diolah dengan seberapa banyak NamaMenu yang muncul. Hasil pengolahan dapat dilihat pada tabel 3.2 dibawah ini :

Tabel 3. 2 ***Frequent Item***

Item	***Frequent***
1	6
2	3
3	1
4	6
5	4
6	4
7	14
8	3
9	2
10	1
11	1
12	3
13	1
14	2
15	3
16	3
17	2
18	4
19	2
20	2
21	1
22	7
23	3
24	6
25	2
26	2
27	3
28	14

Dimisalkan nilai *minimum support* yang digunakan dalam contoh kasus ini adalah sebesar 4, karena dengan nilai *minimum support* tersebut tujuan *data mining* yaitu untuk menyusun pola *layou*t makanan dan pencarian *frequent itemset* dapat dimaksimalkan. Kenapa *minimum support* diambil 4 , supaya pada proses penyusunan layout makanan yaitu yang menu pokok yang terdiri jenis makanan, walaupun menu menu makanan tidak semua dimasukkan, tetapi dalam penyusnannya sudah tergambar bahwa makanan apa saja yang akan dimasukan dalam penyususnan layout makanan.

1. Langkah pertama adalah men-*scan* data transaksi yang ada pada tabel D-14 dalam lampiran D, untuk mengetahui *support count* dan nilai *support* dari setiap menu makanan yang ada. Hasil *support count* didapatkan dari hasil kemunculan *frequent item* yang ada pada tabel 3.2 diatas, sedangkan nilai *support* didapat menggunakan persamaan 2.1 yang terdapat pada bab 2. Berikut adalah hasil dari proses *scan* yang telah dilakukan :

Tabel 3. 3 Tabel Hasil *Scan*

KodeMenu	***Support Count***	**Nilai *Support***
1	6	(6/22)*100 = 27,27
2	3	(3/22)*100 = 13,63
3	1	(1/22)*100 = 4,54
4	6	(6/22)*100 = 27,27
5	4	(4/22)*100 = 18,18
6	4	(4/22)*100 = 18,18
7	14	(14/22)*100 = 63,63
8	3	(3/22)*100 = 13,63
9	2	(2/22)*100 = 9,09
10	1	(1/22)*100 = 4,54
11	1	(1/22)*100 = 4,54
12	3	(3/22)*100 = 13,63
13	1	(1/22)*100 = 4,54

KodeMenu	*Support Count*	Nilai *Support*
14	2	(2/22)*100 = 9,09
15	3	(3/22)*100 = 13,63
16	3	(3/22)*100 = 13,63
17	2	(2/22)*100 = 9,09
18	4	(4/22)*100 = 18,18
19	2	(2/22)*100 = 9,09
20	2	(2/22)*100 = 9,09
21	1	(1/22)*100 = 4,54
22	7	(7/22)*100 = 31,81
23	3	(3/22)*100 = 13,63
24	6	(6/22)*100 = 27,27
25	2	(2/22)*100 = 9,09
26	2	(2/22)*100 = 9,09
27	3	(3/22)*100 = 13,63
28	14	(14/22)*100 = 63,63

2. Langkah kedua yaitu menghilangkan data yang memiliki nilai *minimum support yang* kurang dari *minimum support* yang ditentukan sebelumnya. Karena semua data memenuhi nilai *minimum support*, maka tidak ada data yang dihilangkan.
3. Setelah didapatkan data yang memenuhi nilai *minimum support* seperti pada tabel 3.3 diatas, selanjutnya mencari *frequency of occurrence*. Urutkan tabel 3.3 berdasarkan *frequent* terbesar sampai dengan terkecil, kemudian memberikan *priority* dimulai dari data frekeunsi terbesar sampai dengan terkecil.

Tabel 3. 4 *Frequency Of Occurrence*

Item	*Frequent*	*Priority*
7	14	1
28	14	2

Item	*Frequent*	*Priority*
22	7	3
1	6	4
4	6	5
24	6	6
5	4	7
6	4	8
18	4	9
2	3	10
8	3	11
12	3	12
15	3	13
16	3	14
23	3	15
27	3	16
9	2	17
14	2	18
17	2	19
19	2	20
20	2	21
25	2	22
26	2	23
3	1	24
10	1	25
11	1	26
13	1	27
21	1	28

4. Setelah mendapatkan hasil dari *frequency of occurrence* proses selanjutnya adalah mengurutkan data yang ada pada tabel D-14 dalam lampiran D

sesuai dengan hasil penentuan *priority* pada tabel 3.4 diatas Berikut adalah hasil dari pengurutan data yang telah dilakukan :

Tabel 3. 5 Hasil Pengurutan Data Berdasarkan Priority

NoBill	*Ordered Items*
006477	13
006523	28, 15, 26, 10
006093	7, 24, 5
006097	7, 4, 18, 14
006111	7, 24, 6, 2
006752	7, 28, 22, 1, 4, 8, 27, 9, 17, 20, 26, 11, 21
006754	7, 28, 1
006553	7, 28, 22, 1, 24, 6, 18, 2, 8, 23, 27, 9, 14, 25
006554	28, 5, 15
006555	7, 28, 22, 16, 23, 19
006556	7, 28
006557	28, 22, 24, 5, 2, 12
006558	7, 28, 22, 4, 6, 19, 3
006559	7, 18, 17
007072	28, 22, 1, 4, 8, 12, 15, 16
007073	7, 28, 4
007074	7, 28, 1
007075	7, 28, 24, 5, 23, 25
007076	7, 28, 22, 1, 4, 24, 6, 18, 12, 16, 27, 20

5. Setelah mendapatkan data yang telah diurutkan, selanjutnya membuat *fp-tree*. Berikut adalah tahap-tahap dalam pembuatan *fp-tree* yaitu :
   a. Lakukan pembacaan pada data pertama yang akan diproses.
   b. Lakukan pengecekan pada *root*, jika root belum mempunyai anak (*child)*, maka data yang dibaca tadi menjadi anak dari *root*. Dan *support count* dari masing masing *node* bertambah 1, kecuali *root*.

c. 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.

d. Lakukan pengecekan pada *root*, jika *root* sudah mempunyai anak, maka 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 bertambah 1.

e. Lakukan langkah tersebut hingga semua data yang keluar.

f. Jika ada data yang sama setelah pembacaan, namun berbeda *lintasan* maka *nod*e pada *laintasan* yang lain tidak bertambah *support* count-nya, hanya data yang dibaca saja yang bertambah *support count*-nya.
Pembentukan *FP-Tree layout* makanan Dapat dilihat pada lampiran D-15 dalam lampiran D

6. Setelah tahap pembangunan *fp-tree* dari sekumpulan data transaksi dilakukan, maka akan diterapkan algoritma *fp-growth* untuk mencari *frequent itemset* yang signifikan. Berikut langkah-langkah utama algoritma *fp-growth* :

a. Tahap Pembangkitan *Conditional Pattern Base*

Pembangkitan *conditinal pattern base* didapatkan melalui hasil *FP-Tree* seluruhnya, dengan mencari *support count* terkecil sesuai dengan hasil pengurutan *priority* yang terdapat apda tabel 3.4 sebelumnya.

Dan telah didapatkan yaitu

:21,13,11,10,3,26,25,20,19,17,14,9,27,23,16,15,12,8,2,18,6,5,24,4,1,22,28,

7. Berikut adalah hasil tabel dari *conditional pattern base* :

Tabel 3. 6 *Conditional Pattern Base*

Item	***Conditonal Pattern Base***
21	• 7, 28, 22, 1, 4, 8, 27, 9, 17, 20, 26, 11, 21:1
13	• null, 13:1

Item	*Conditonal Pattern Base*
11	• 7, 28, 22, 1, 4, 8, 27, 9, 17, 20, 26, 11:1
10	• 28, 15, 26, 10:1
3	• 7, 28, 22, 4, 6, 19, 3:1
26	• 28, 15, 26:1 • 7, 28, 22, 1, 4, 8, 27, 9, 17, 20, 26:1
25	• 7, 28, 22, 1, 24, 6, 18, 2, 8, 23, 27, 9, 14, 25:1 • 7, 28, 24, 5, 23, 25:1}
20	• 7, 28, 22, 1, 4, 8, 27, 9, 17, 20:1 • 7, 28, 22, 1, 4, 24, 6, 18, 12, 16, 27, 20:1
19	• 7, 28, 22, 16, 23, 19:1 • 7, 28, 22, 4, 6, 19:1
17	• 7, 28, 22, 1, 4, 8, 27, 9, 17:1 • 7, 18, 17:1
14	• 7, 4, 18, 14:1 • 7, 28, 22, 1, 24, 6, 18, 2, 8, 23, 27, 9, 14:1
9	• 7, 28, 22, 1, 4, 8, 27, 9:1 • 7, 28, 22, 1, 24, 6, 18, 2, 8, 23, 27, 9:1
27	• 7, 28, 22, 1, 4, 8, 27:1 • 7, 28, 22, 1, 24, 6, 18, 2, 8, 23, 27:1 • 7, 28, 22, 1, 4, 24, 6, 18, 12, 16, 27:1
23	• 7, 28, 22, 1, 24, 6, 18, 2, 8, 23:1 • 7, 28, 22, 16, 23:1 • 7, 28, 24, 5, 23:1
16	• 7, 28, 22, 16:1 • 28, 22, 1, 4, 8, 12, 15, 16:1 • 7, 28, 22, 1, 4, 24, 6, 18, 12, 16:1
15	• 28, 15:1 • 28, 5, 15:1

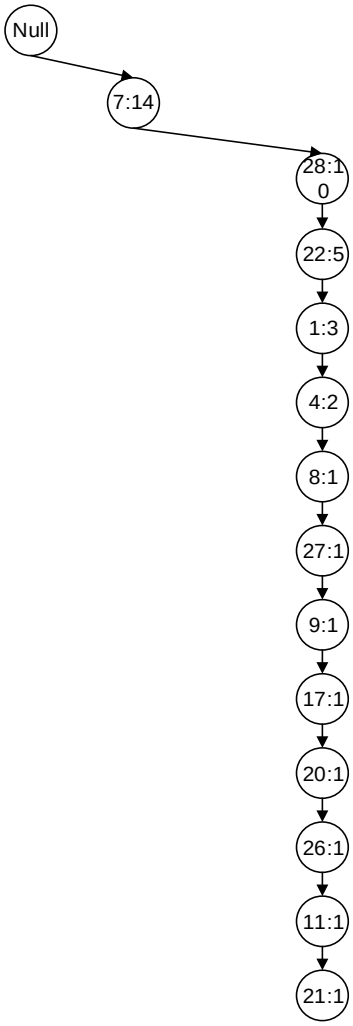
	• 28, 22, 1, 4, 8, 12, 15:1
12	• 28, 22, 24, 5, 2, 12:1 • 28, 22, 1, 4, 8, 12:1 • 7, 28, 22, 1, 4, 24, 6, 18, 12:1
8	• 7, 28, 22, 1, 4, 8:1 • 7, 28, 22, 1, 24, 6, 18, 2, 8:1 • 28, 22, 1, 4, 8:1
2	• 7, 24, 6, 2:1 • 7, 28, 22, 1, 24, 6, 18, 2:1 • 28, 22, 24, 5, 2:1
18	• 7, 4, 18:1 • 7, 28, 22, 1, 24, 6, 18:1 • 7, 18:1 • 7, 28, 22, 1, 4, 24, 6, 18:1
6	• 7, 24, 6:1 • 7, 28, 22, 1, 24, 6:1 • 7, 28, 22, 4, 6:1 • 7, 28, 22, 1, 4, 24, 6:1
5	• 7, 24, 5:1 • 28, 5:1 • 28, 22, 24, 5:1 • 7, 28, 24, 5:1
24	• 7, 24:2 • 7, 28, 22, 1, 24:1 • 28, 22, 24:1 • 7, 28, 24:1 • 7, 28, 22, 1, 4, 24:1
4	• 7, 4:1 • 7, 28, 22, 1, 4:2

	• 7, 28, 22, 4:1 • 28, 22, 1, 4:1 • 7, 28, 4:1
1	• 7, 28, 22, 1:3 • 7, 28, 1:2 • 28, 22, 1:1
22	• 7, 28, 22:5 • 28, 22:2
28	• null, 28:4 • 7, 28:10
7	• null, 7:14

b. Tahap Pembangkitan Conditional FP-Tree

Pada tahap ini, *support count* dari setiap item pada tabel *conditional pattern base* diatas dijumlahkan, lalu setiap item yang memiliki jumlah *support count* lebih besar atau sama dengan *minimum support* akan dibangkitkan dengan *conditional FP-Tree.*

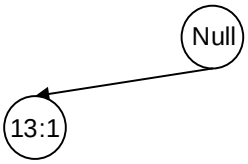
- *Condiitonal FP-Tree* untuk 21 : { 7, 28, 22, 1, 4, 8, 27, 9, 17, 20, 26, 11, 21:1}



Gambar 3. 1 *Conditional FP-Tree Untuk* 21

Karena niilai *support count* 21 adalah 1, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

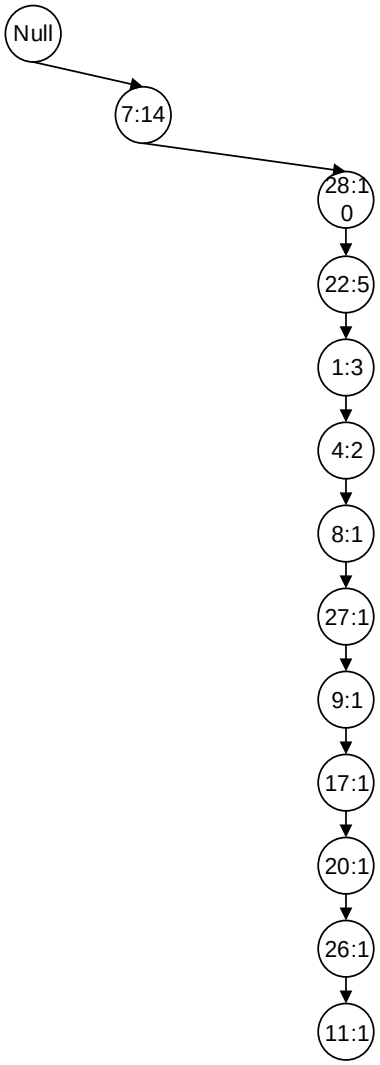
- *Conditional FP-Tree* untuk 13 : { null, 13:1}



Gambar 3. 2 *Conditional FP-Tree* 13

Karena niilai *support count* 13 adalah 1, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

- *Conditional FP-Tree* untuk 11 : {7, 28, 22, 1, 4, 8, 27, 9, 17, 20, 26, 11:1}



Gambar 3. 3 *Conditional FP-Tree* 11

Karena niilai *support count* 11 adalah 1, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

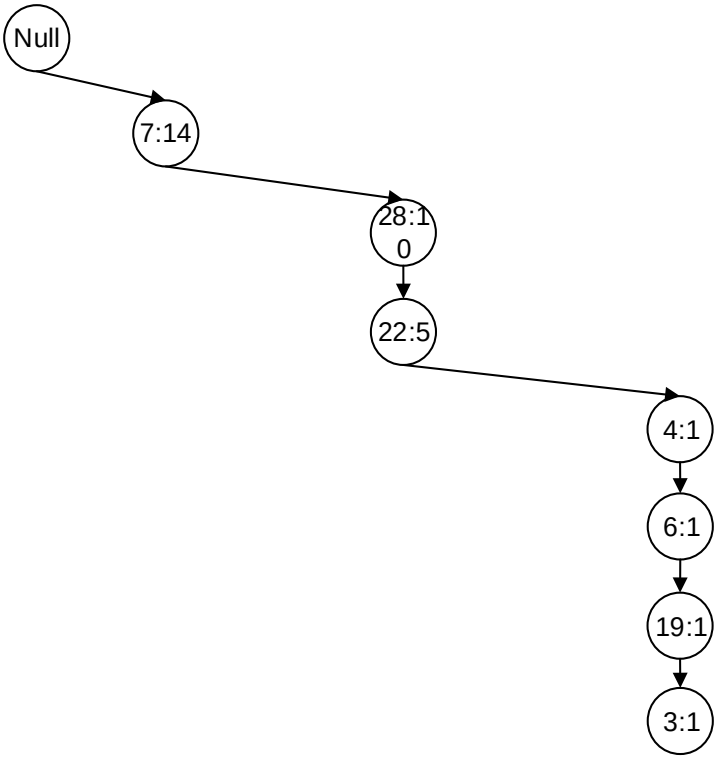
- *Conditional FP-Tree* untuk 10 : {28, 15, 26, 10:1}



Gambar 3. 4 *Conditional FP-Tree* 10

Karena niilai *support count* 10 adalah 1, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

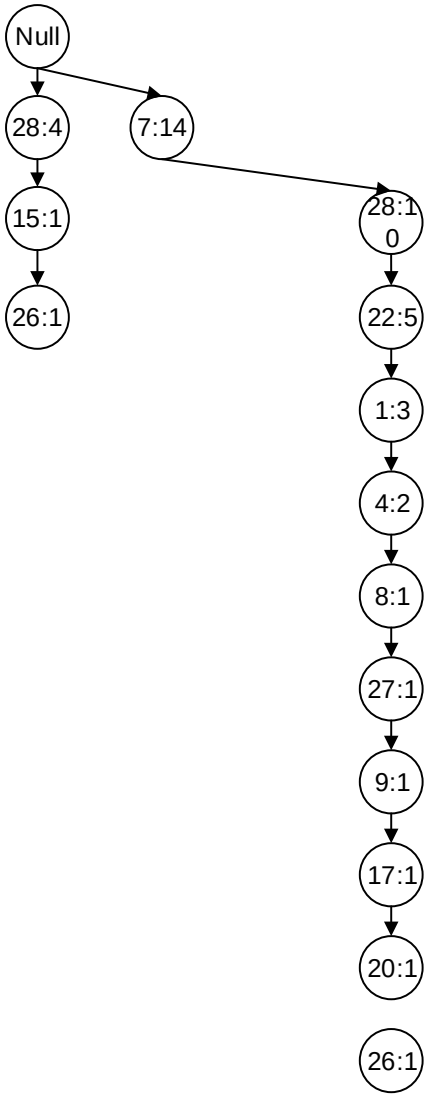
- *Conditional FP-Tree* untuk 3 : {7, 28, 22, 4, 6, 19, 3:1}



Gambar 3. 5 *Conditional FP-Tree* 3

Karena niilai *support count* 3 adalah 1, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

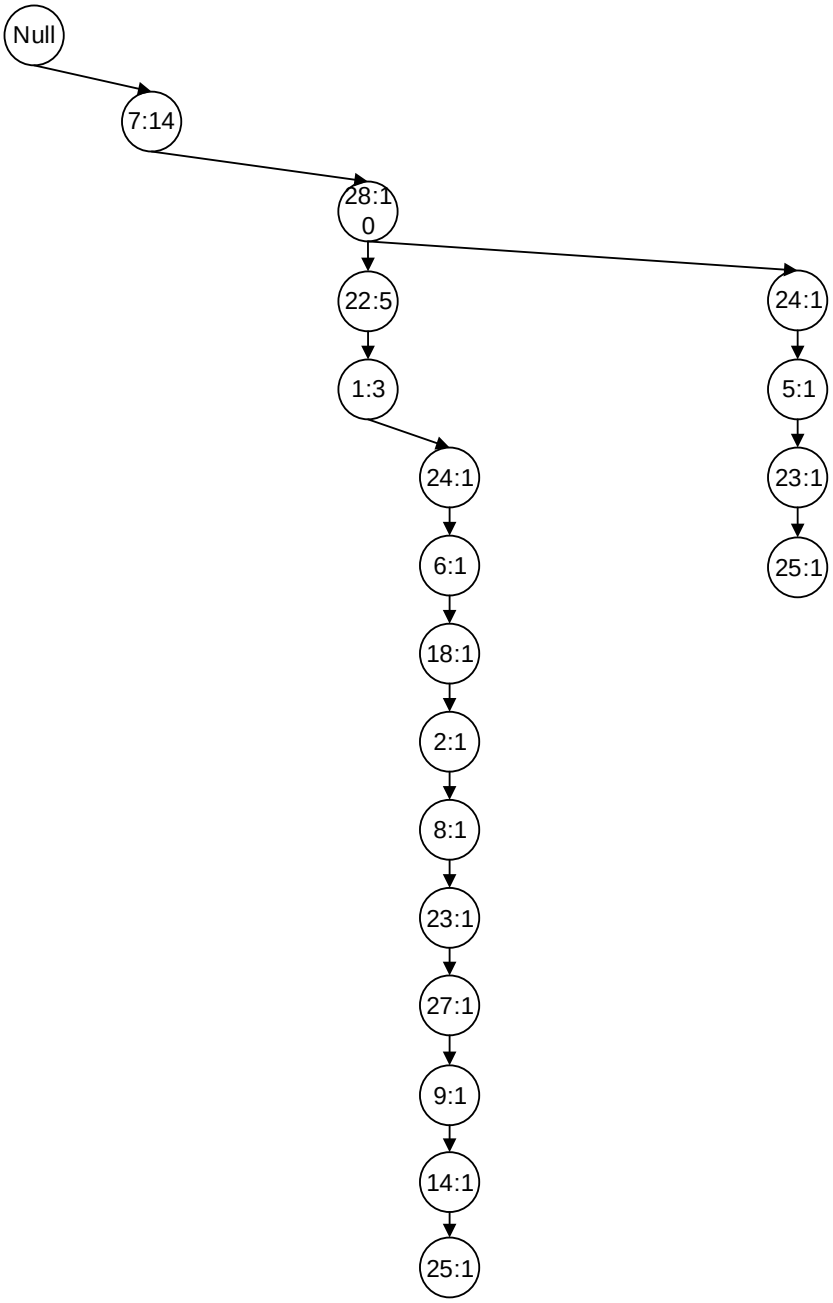
- *Conditional FP-Tree* untuk 26 : {28, 15, 26:1}, {7, 28, 22, 1, 4, 8, 27, 9, 17, 20, 26:1}



Gambar 3. 6 *Conditional FP-Tree 2*6

Karena niilai *support count* 26 adalah 2, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

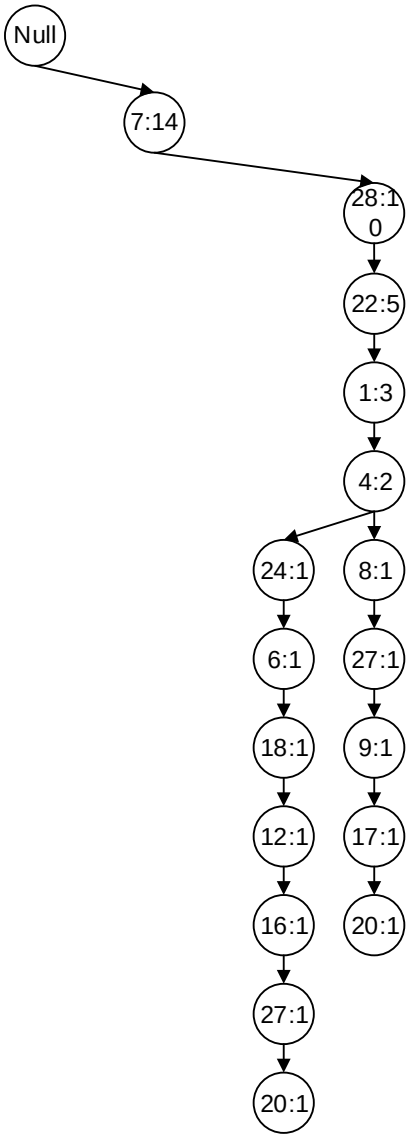
- *Conditional FP-Tree* untuk 25 : {7, 28, 22, 1, 24, 6, 18, 2, 8, 23, 27, 9,14, 25:1} {7, 28, 24, 5, 23, 25:1}



Gambar 3. 7 *Conditional FP-Tree* 25

Karena niilai *support count* 25 adalah 2, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

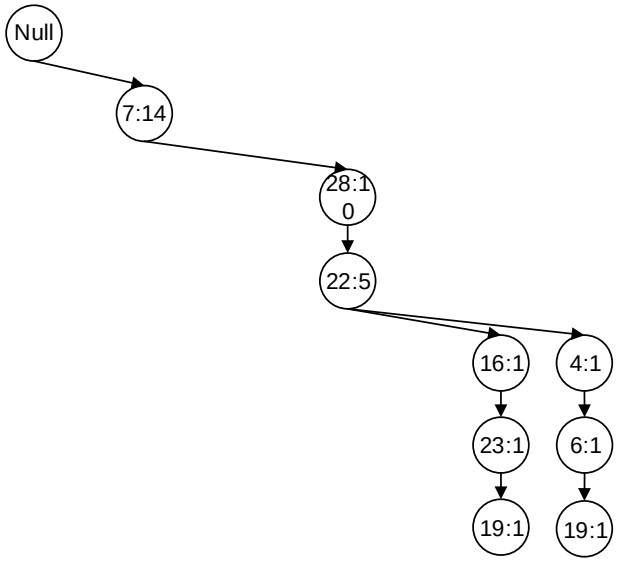
- *Conditional FP-Tree* untuk 20 : {7, 28, 22, 1, 4, 8, 27, 9, 17, 20:1 {7, 28, 22, 1, 4, 24, 6, 18, 12, 16, 27, 20:1}



Gambar 3. 8 *Conditional FP-Tree* 20

Karena niilai *support count* 20 adalah 2, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

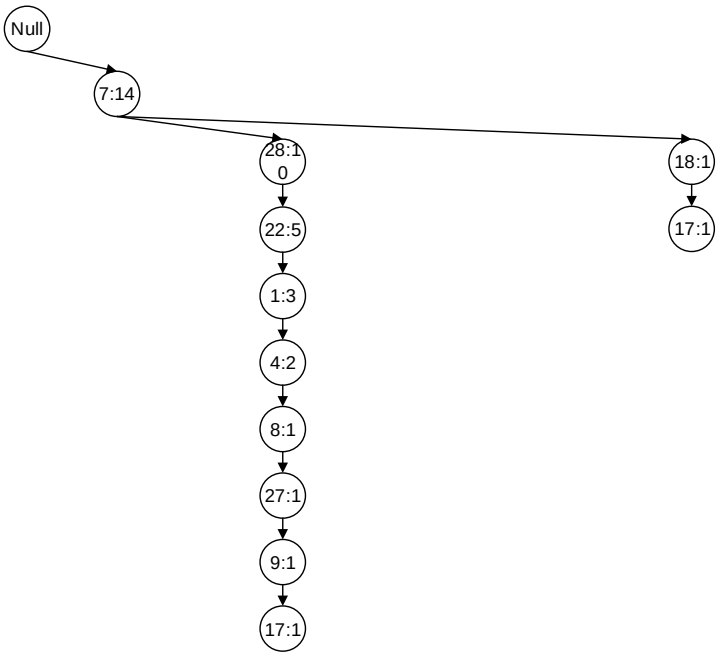
- *Conditional FP-Tree* untuk 19 : {7, 28, 22, 16, 23, 19:1} {7, 28, 22, 4, 6, 19:1}



Gambar 3. 9 *Conditional FP-Tree* 19

Karena niilai *support count* 19 adalah 2, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

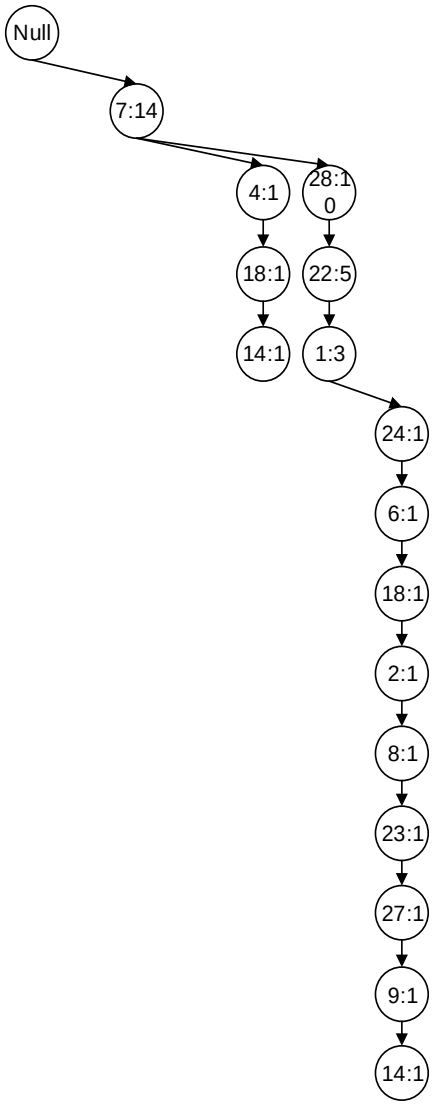
- *Conditional FP-Tree* untuk 17 : {7, 28, 22, 1, 4, 8, 27, 9, 17:1} {7, 18, 17:1}



Gambar 3. 10 *Conditional FP-Tree* 17

Karena niilai *support count* 17 adalah 2, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan..

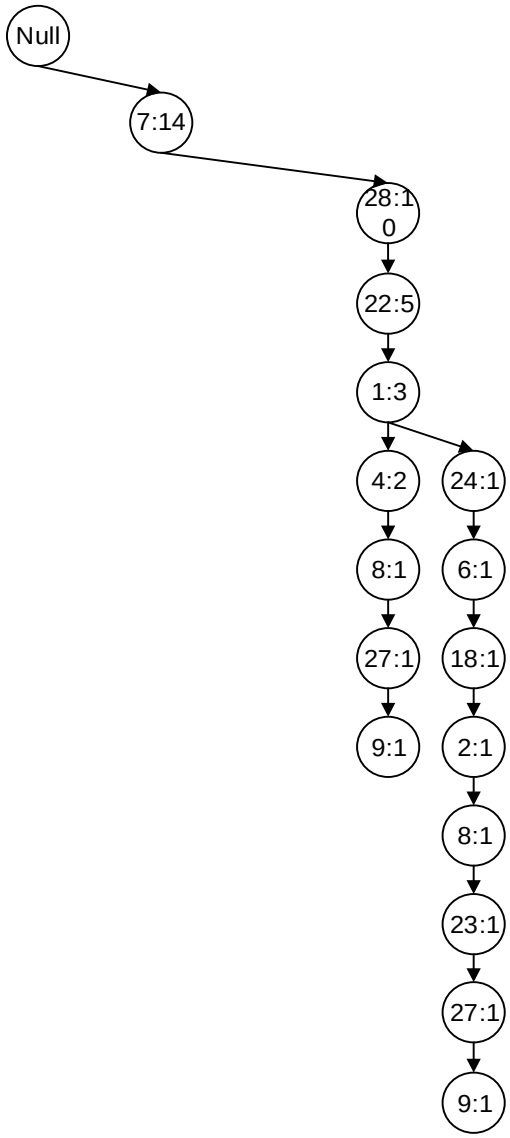
- *Conditional FP-Tree* untuk 14 : {7, 4, 18, 14:1} {7, 28, 22, 1, 24, 6, 18, 2, 8, 23, 27, 9, 14:1}



Gambar 3. 11 *Conditional FP-Tree* 14

Karena niilai *support count* 14 adalah 2, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan..

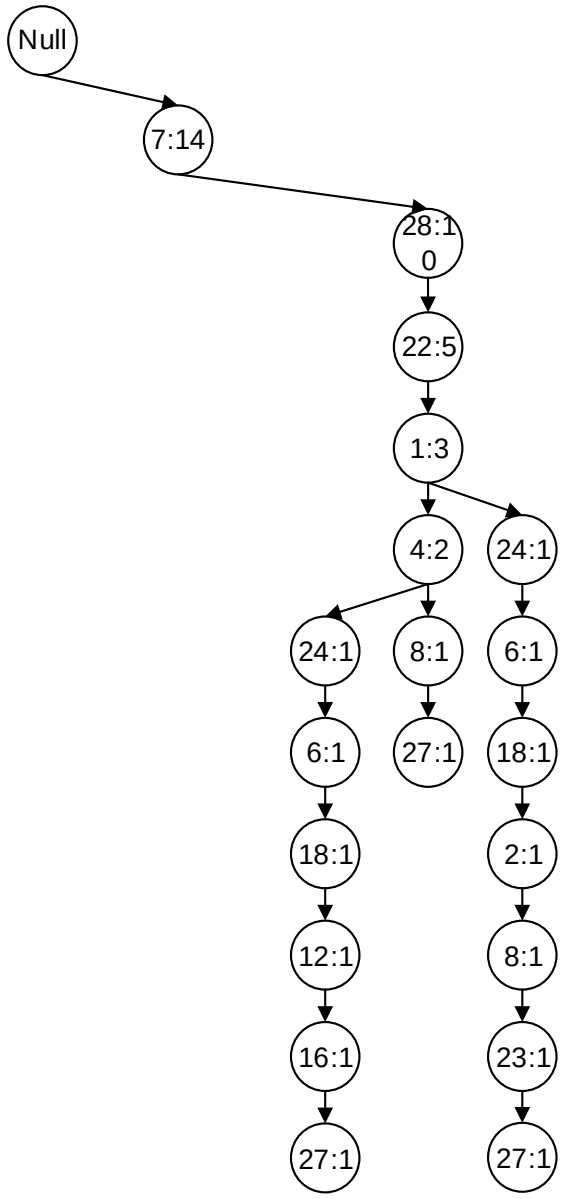
- *Conditional FP-Tree* untuk 9 : {7, 28, 22, 1, 4, 8, 27, 9:1} {7, 28, 22, 1, 24, 6, 18, 2, 8, 23, 27, 9:1}



Gambar 3. 12 *Conditional FP-Tree* 9

Karena niilai *support count* 9 adalah 2, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

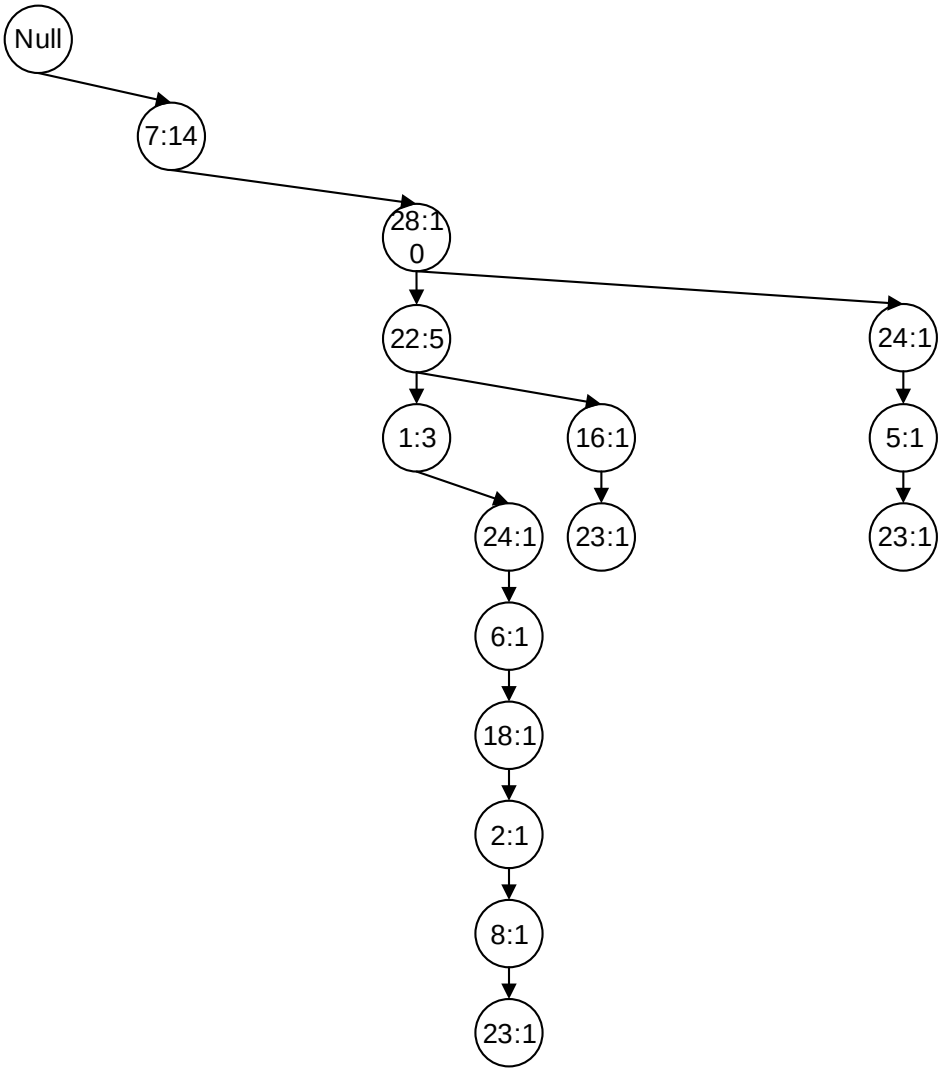
- *Conditional FP-Tree* untuk 27 : {7, 28, 22, 1, 4, 8, 27:1} {7, 28, 22, 1, 24, 6, 18, 2, 8, 23, 27:1} {7, 28, 22, 1, 4, 24, 6, 18, 12, 16, 27:1}



Gambar 3. 13 *Conditional FP-Tree* 27

Karena niilai *support count* 27 adalah 3, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

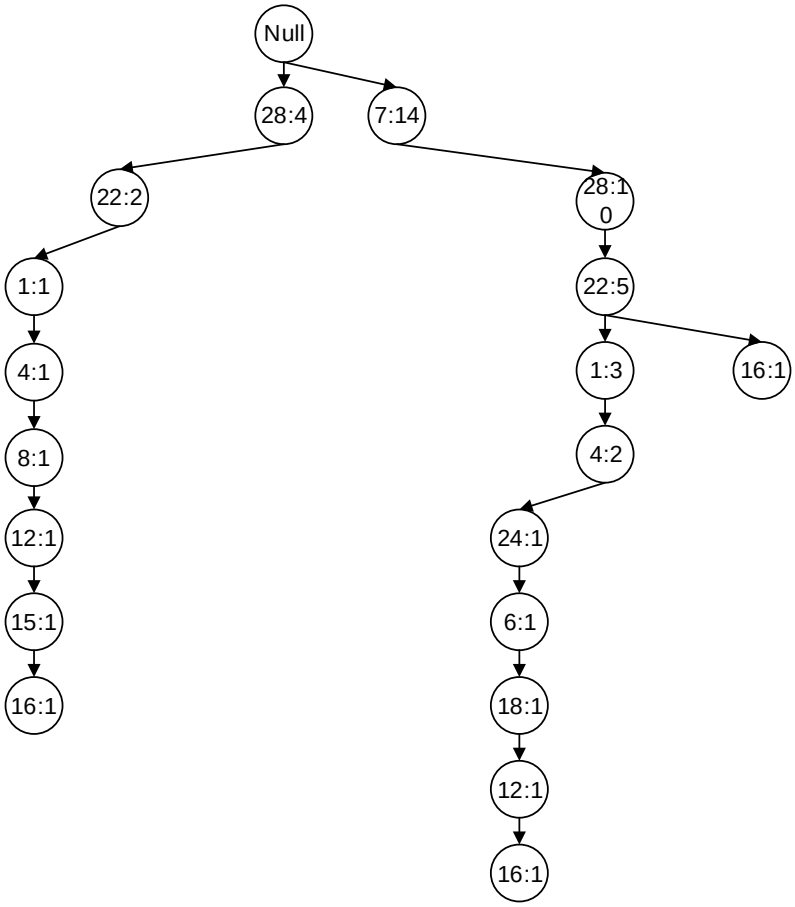
- *Conditional FP-Tree* untuk 23 : {7, 28, 22, 1, 24, 6, 18, 2, 8, 23:1} {7, 28, 22, 16, 23:1} {7, 28, 24, 5, 23:1}



Gambar 3. 14 *Conditional FP-Tree* 23

Karena niilai *support count* 23 adalah 3, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

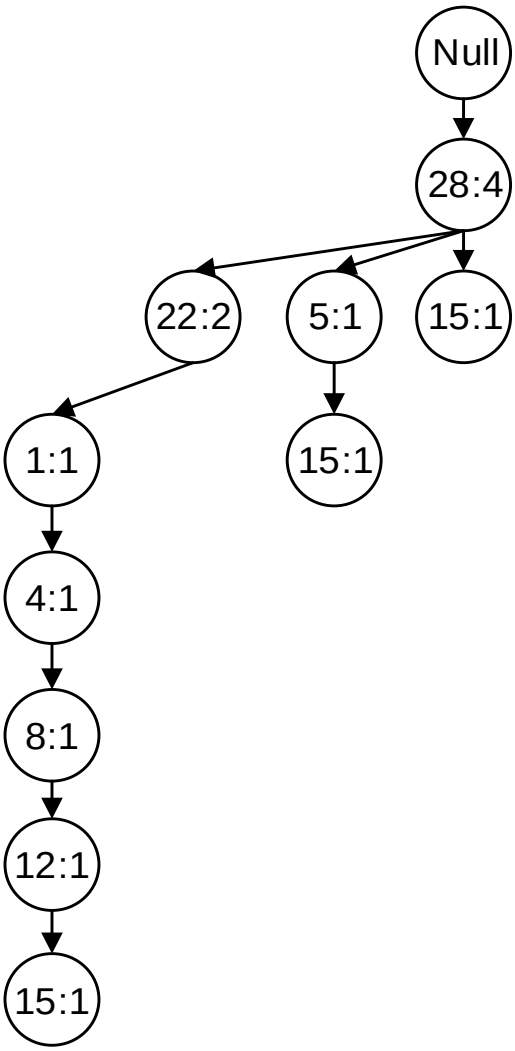
- *Conditional FP-Tree* untuk 16 : {7, 28, 22, 16:1}{28, 22, 1, 4, 8, 12, 15, 16:1} {7, 28, 22, 1, 4, 24, 6, 18, 12, 16:1}



Gambar 3. 15 *Conditional FP-Tree* 16

Karena niilai *support count* 16 adalah 3, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan..

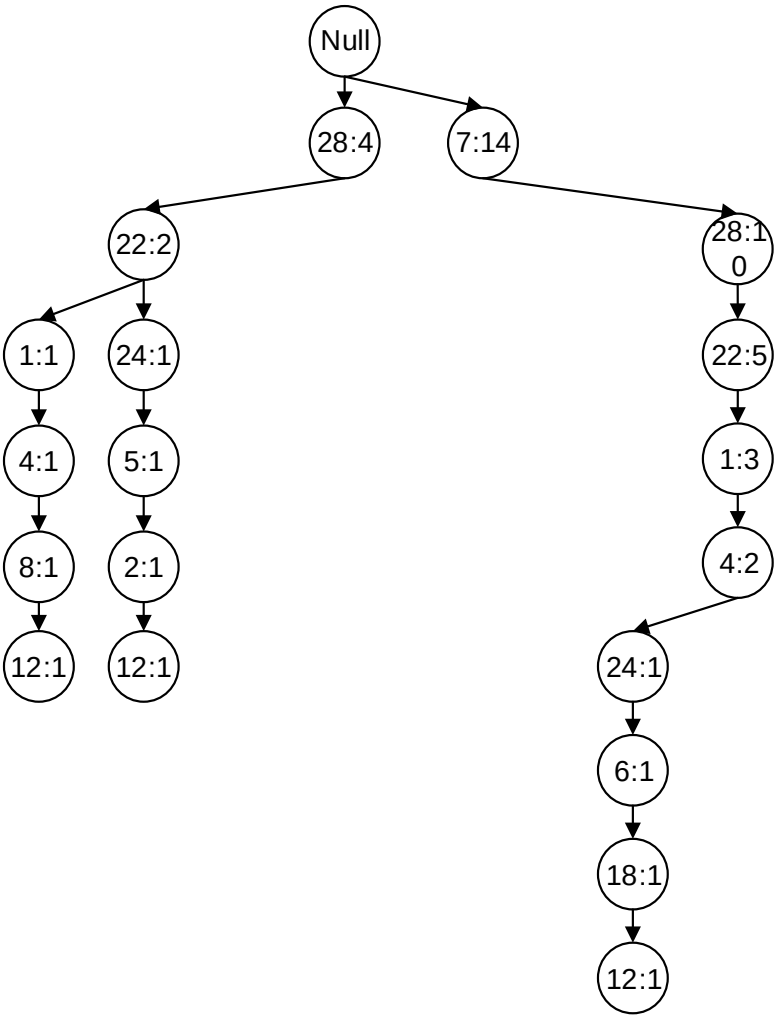
- *Conditional FP-Tree* untuk 15 : {28, 15:1} {28, 5, 15:1} {28, 22, 1, 4, 8, 12, 15:1}



Gambar 3. 16 *Conditional FP-Tree* 15

Karena niilai *support count* 15 adalah 3, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

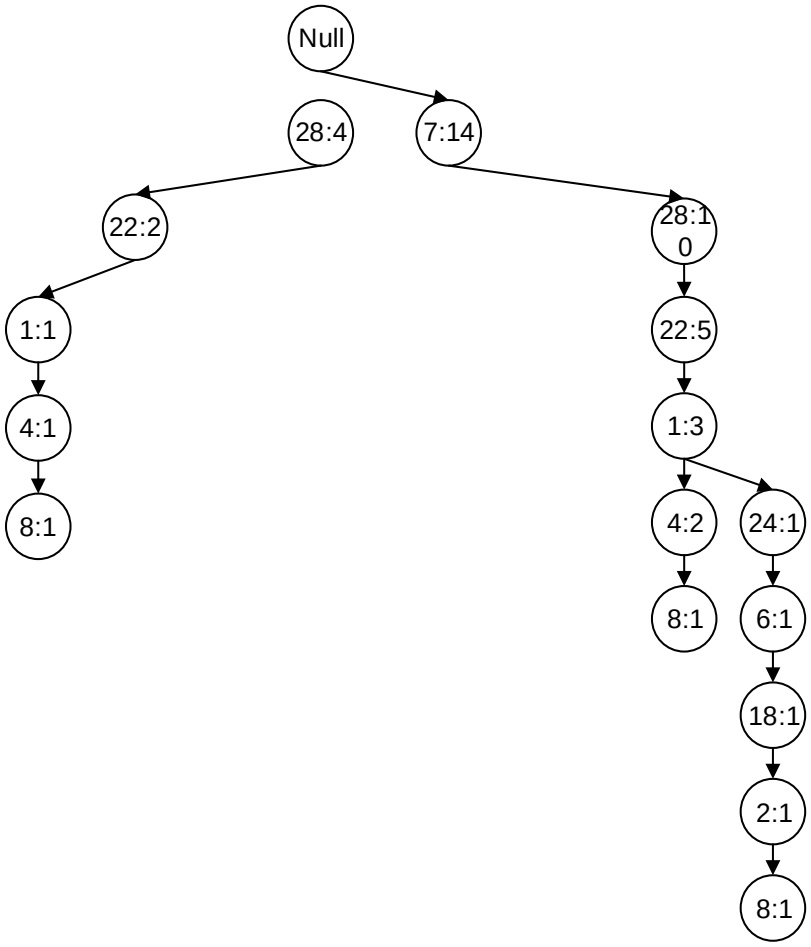
- *Conditional FP-Tree* untuk 16 : {28, 22, 24, 5, 2, 12:1}{28, 22, 1, 4, 8, 12:1} {7, 28, 22, 1, 4, 24, 6, 18, 12:1}



Gambar 3. 17 *Conditional FP-Tree* 12

Karena niilai *support count* 12 adalah 3, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

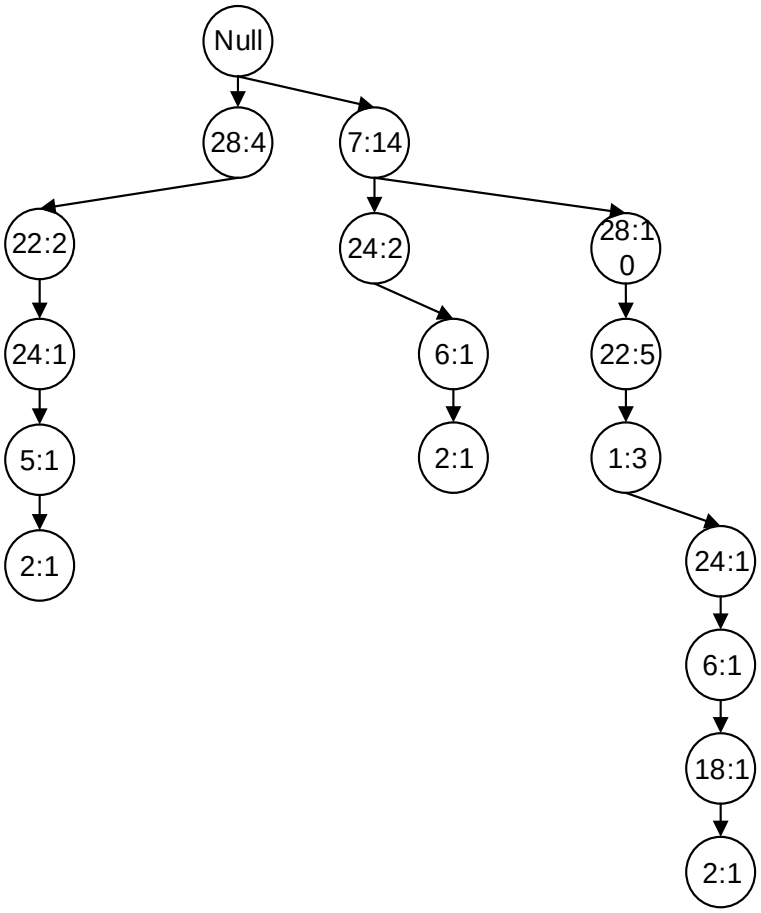
- *Conditional FP-Tree* untuk 8 **:** {7, 28, 22, 1, 4, 8:1} {7, 28, 22, 1, 24, 6, 18, 2, 8:1} {28, 22, 1, 4, 8:1}



Gambar 3. 18 *Conditional FP-Tree* 8

Karena niilai *support count* 8 adalah 3, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

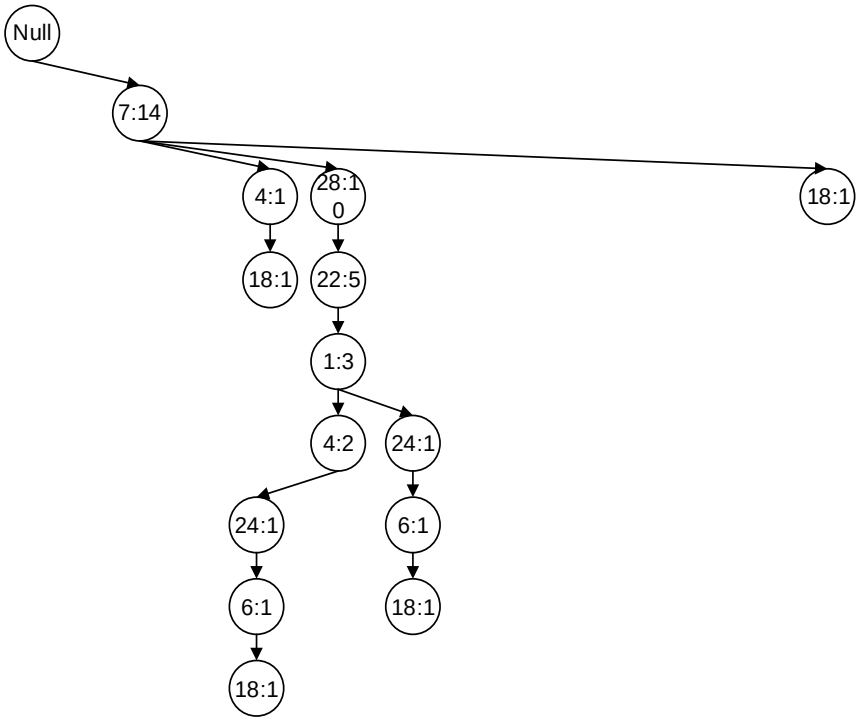
- *Conditional FP-Tree* untuk 2 : {7, 24, 6, 2:1} {7, 28, 22, 1, 24, 6, 18, 2:1} {28, 22, 24, 5, 2:1}



Gambar 3. 19 *Conditional FP-Tree* 2

Karena niilai *support count* 2 adalah 3, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

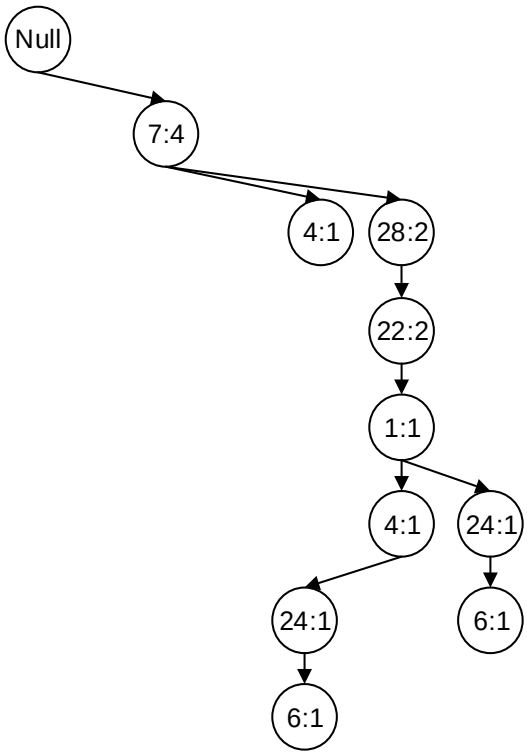
- *Conditional FP-Tree* untuk 18 **:** {7, 4, 18:1} {7, 28, 22, 1, 24, 6, 18:1} {7, 18:1} {7, 28, 22, 1, 4, 24, 6, 18:1}



Gambar 3. 20 *Conditional FP-Tree* 18

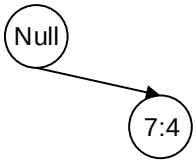
Karena niilai *support count* 18 adalah 4, maka memenuhi *minimum support* yaitu 4. Berikut adalah tahap-tahapannya :

- Jumlah *support count* yang memenuhi *minimum support* akan dibangkitkan dengan *conditional fp-tree*.
- Dalam conditional fp-tree untuk 18, ada 4 lintasan yaitu :
  - Lintasan pertama : {7,4,18:1}
  - Lintasan kedua : {7,28,22,1,4,24,6,18:1}
  - Lintasan ketiga : {7,28,22,1,24,6,18}
  - Lintasan keempat : {7, 18:1}
- Pada lintasan yang berakhiran {18:1} dengan *support count* adalah 1 atau lebih dari 1 akan dihilangkan dan *support count* yang melewati lintasan yang berakhiran {18}, akan sama *support count*-nya dengan *support count* 18 sesuai dengan *support* yang dimilikinya.
- Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka *support count*-nya akan selalu bertambah 1.



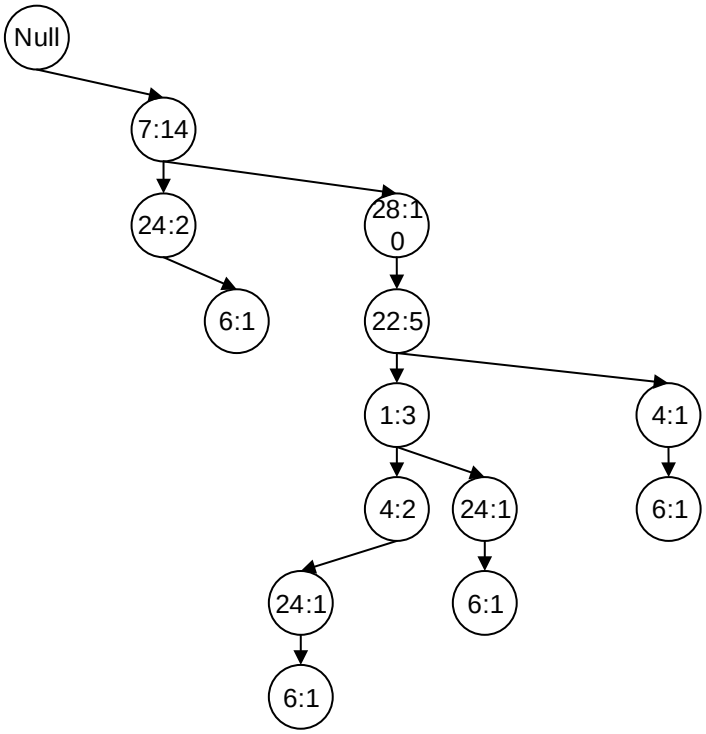
Gambar 3. 21 *Conditional FP-Tree* 18 Setelah Dibangkitkan

- Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequent itemset* yang keluar yang memenuhi *minimum support*. 7:4 adalah satu-satunya yang keluar yang dapat memenuhi *minimum support*. Dan *frequent itemset* yang didapatkan adalah {18, 7:4}



Gambar 3. 22 *Conditional FP-Tree* 18,7:4 Hasil *Frequent Itemset*

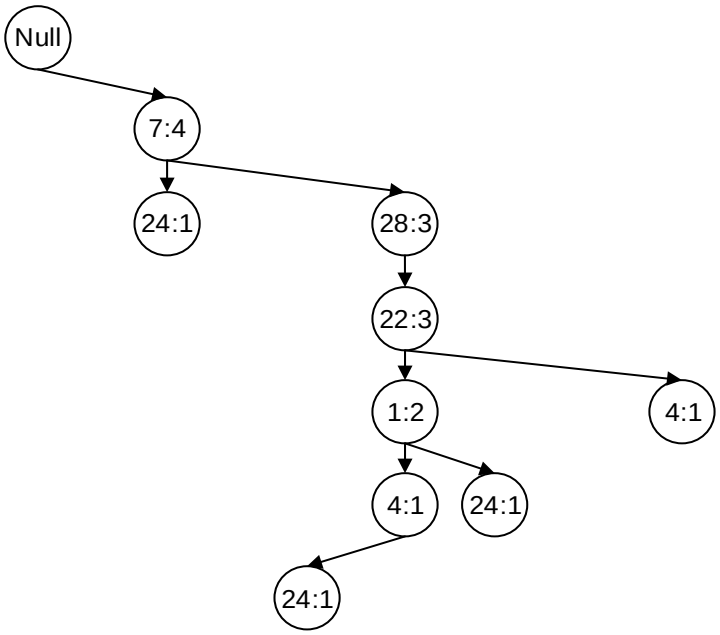
- *Conditional FP-Tree* untuk 6**:** {7, 24, 6:1} {7, 28, 22, 1, 24, 6:1} {7, 28, 22, 4, 6:1} {7, 28, 22, 1, 4, 24, 6:1}



Gambar 3. 23 *Conditional FP-Tree* 6

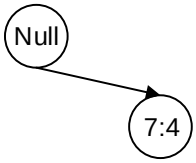
Karena niilai *support count* 6 adalah 4, maka memenuhi *minimum support* yaitu 4. Berikut adalah tahap-tahapannya :

- Jumlah *support count* yang memenuhi *minimum support* akan dibangkitkan dengan *conditional fp-tree*.
- Dalam conditional fp-tree untuk 6, ada 4 lintasan yaitu :
    - Lintasan pertama : {7,24,6:1}
    - Lintasan kedua : {7,28,22,1,4,24,6:1 }
    - Lintasan ketiga : {7,28,22,1,24,6:1}
    - Lintasan keempat : {7, 28,22,4,6:1}
- Pada lintasan yang berakhiran {6:1} dengan *support count* adalah 1 atau lebih dari 1 akan dihilangkan dan *support count* yang melewati lintasan yang berakhiran {6}, akan sama *support count*-nya dengan *support count* 6 sesuai dengan *support* yang dimilikinya.
- Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka *support count*-nya akan selalu bertambah 1.



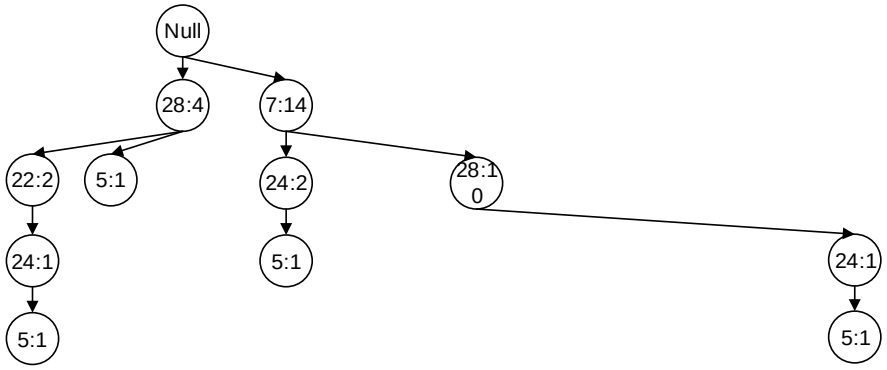
Gambar 3. 24 *Conditional FP-Tree* 6 Setelah Dibangkitkan

- Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequent itemset* yang keluar yang memenuhi *minimum support*. 7:4 adalah satu-satunya yang keluar yang dapat memenuhi *minimum support*. Dan frequent itemset yang didapatkan adalah {6, 7:4}



Gambar 3. 25 *Conditional FP-Tree* 6,7:4 Hasil *Frequent Itemset*

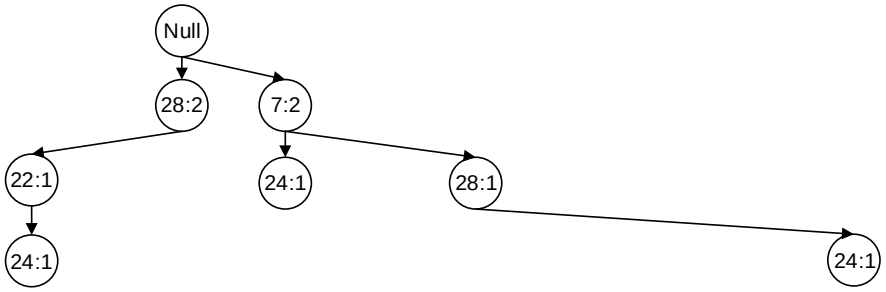
- *Conditional FP-Tree* untuk 5**:** 7, 24, 5:1} {28, 5:1} {28, 22, 24, 5:1} {7, 28, 24, 5:1}



Gambar 3. 26 *Conditional FP-Tree* 5

Karena niilai *support count* 5 adalah 4, maka memenuhi *minimum support* yaitu 4. Berikut adalah tahap-tahapannya :

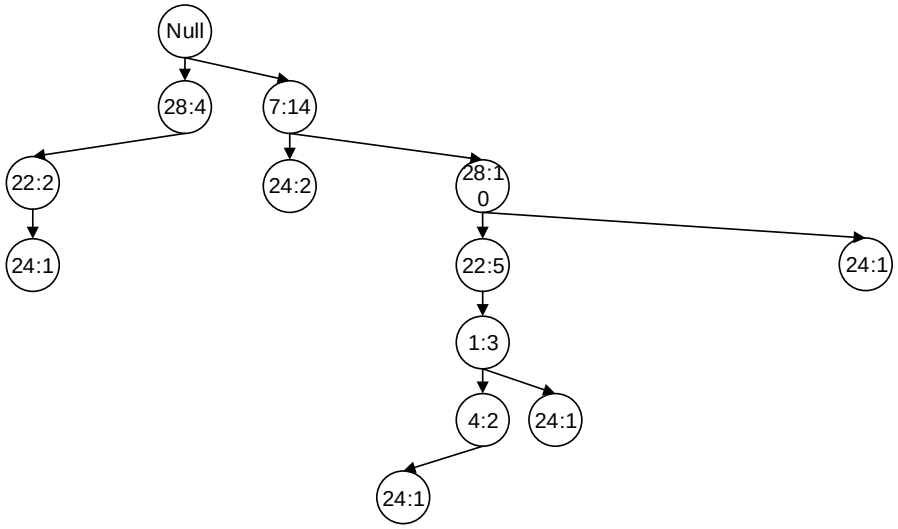
- Jumlah *support count* yang memenuhi *minimum support* akan dibangkitkan dengan *conditional fp-tree*.
- Dalam conditional fp-tree untuk 5, ada 4 lintasan yaitu :
    - Lintasan pertama : {28,22,24, 5:1}
    - Lintasan kedua : {28, 5:1 }
    - Lintasan ketiga : {7,24,5:1}
    - Lintasan keempat : {7, 28,24, 5:1}
- Pada lintasan yang berakhiran {5:1} dengan *support count* adalah 1 atau lebih dari 1 akan dihilangkan dan *support count* yang melewati lintasan yang berakhiran {5}, akan sama *support count*-nya dengan *support count* 5sesuai dengan *support* yang dimilikinya.
- Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka *support count*-nya akan selalu bertambah 1.



Gambar 3. 27 *Conditional FP-Tree* 5 Setelah Dibangkitkan

- Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequent itemset* yang keluar yang memenuhi *minimum support*. Dari *fp-ree* diatas tidaka da yang memenuhi *minimum support*, maka tidak ada yang keluar *frequent itemset*-nya

- *Conditional FP-Tree* untuk 24**:** {7, 24:2} {7, 28, 22, 1, 24:1} {28, 22, 24:1} {7, 28, 24:1} {7, 28, 22, 1, 4, 24:1}

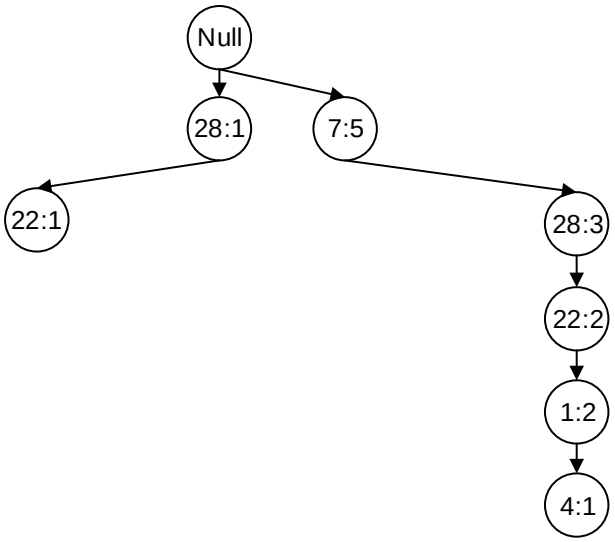


Gambar 3. 28 *Conditional FP-Tree* 24

Karena niilai *support count* 24 adalah 6, maka memenuhi *minimum support* yaitu 4. Berikut adalah tahap-tahapannya :

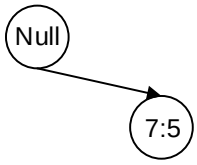
- Jumlah *support count* yang memenuhi *minimum support* akan dibangkitkan dengan *conditional fp-tree*.

- Dalam *conditional fp-tree* untuk 24, ada 5 lintasan yaitu :
    - Lintasan pertama : {28,22, 24:1}
    - Lintasan kedua : {7,24:2 }
    - Lintasan ketiga : {7,28,22,1,4,24:1}
    - Lintasan keempat : {7,28,22,1,24:1}
    - Lintasan Kelima : {7,28, 24:1}
- Pada lintasan yang berakhiran {24:1} dengan *support count* adalah 1 atau lebih dari 1 akan dihilangkan dan *support count* yang melewati lintasan yang berakhiran {24}, akan sama *support count*-nya dengan *support count* 24 sesuai dengan *support* yang dimilikinya.
- Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka *support count*-nya akan selalu bertambah 1.

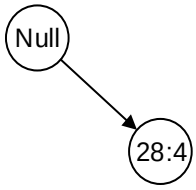


Gambar 3. 29 *Conditional FP-Tree* 24 Setelah Dibangkitkan

- Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequent itemset* yang keluar yang memenuhi *minimum support*. 7:5 dan 28:4 adalah yang keluar yang dapat memenuhi *minimum support*. Dan frequent itemset yang didapatkan adalah {24, 7:5},{24,28:4}

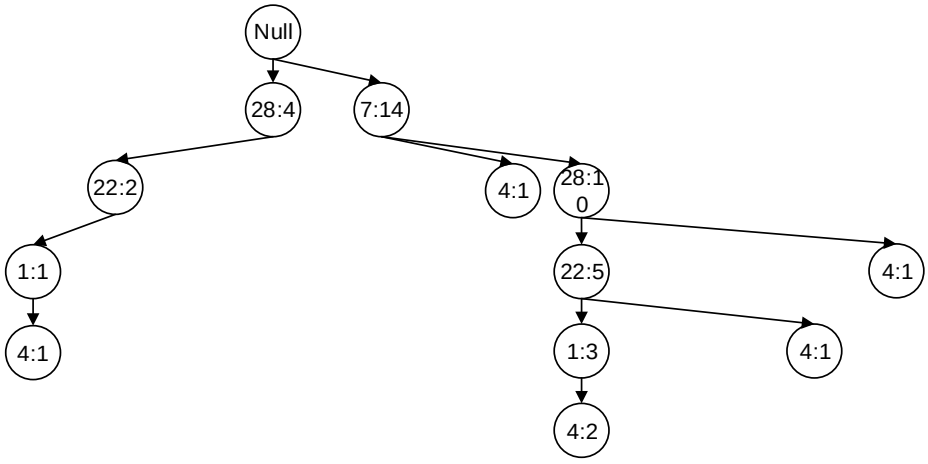


Gambar 3. 30 *Conditional FP-Tree* 24,7:5 Hasil *Frequent Itemset*



Gambar 3. 31 *Conditional FP-Tree* 24,28:4 Hasil *Frequent Itemset*

- *Conditional FP-Tree* untuk 4**:** {7, 4:1} {7, 28, 22, 1, 4:2} {7,28,22,4:1 } {28,22,1,4:1} {7,28,4:1}



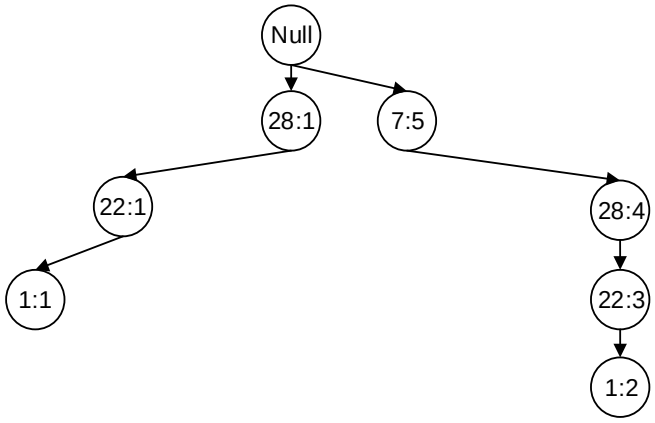
Gambar 3. 32 *Conditional FP-Tree* 4

Karena niilai *support count* 4 adalah 6, maka memenuhi *minimum support* yaitu 4. Berikut adalah tahap-tahapannya :

- Jumlah *support count* yang memenuhi *minimum support* akan dibangkitkan dengan *conditional fp-tree*.
- Dalam *conditional fp-tree* untuk 4, ada 5 lintasan yaitu :
  - Lintasan pertama : {7,4:1}

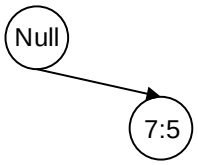
- Lintasan kedua : {7,28,22,1,4:2}
- Lintasan ketiga : {7,28,22,4:1}
- Lintasan keempat : {28,22,1,4:1}
- Lintasan Kelima : {7,28,4:1 }

- Pada lintasan yang berakhiran {4:1} dengan *support count* adalah 1 atau lebih dari 1 akan dihilangkan dan *support count* yang melewati lintasan yang berakhiran {4}, akan sama *support count*-nya dengan *support count* 4sesuai dengan *support* yang dimilikinya.
- Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka *support count*-nya akan selalu bertambah 1.

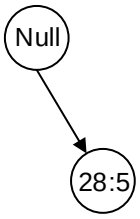


Gambar 3. 33 *Conditional FP-Tree* 4 Setelah Dibangkitkan

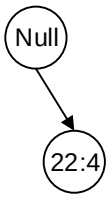
- Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequent itemset* yang keluar yang memenuhi *minimum support*. 7:5, 28:5 dan 22:4 adalah yang keluar yang dapat memenuhi *minimum support*. Dan *frequent itemset* yang didapatkan adalah {4, 7:5},{4,28:5}, {4,22:4}



Gambar 3. 34 *Conditional FP-Tree* 4,7:5 Hasil *Frequent Itemset*

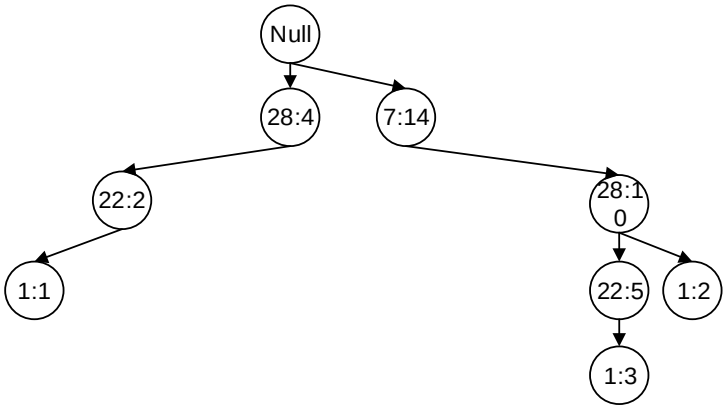


Gambar 3. 35 *Conditional FP-Tree* 4,28:5 Hasil *Frequent Itemset*



Gambar 3. 36 *Conditional FP-Tree* 4,22:4 Hasil *Frequent Itemset*

- *Conditional FP-Tree* untuk 1**:** {7,28,22,1:3},{7,28,1:2},{28,22,1:1}



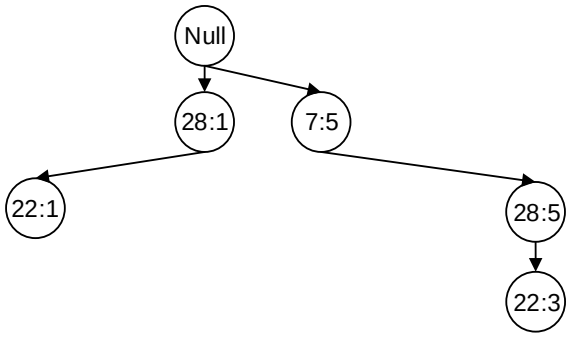
Gambar 3. 37 *Conditional FP-Tree* 1

Karena niilai *support count* 1 adalah 6, maka memenuhi *minimum support* yaitu 4. Berikut adalah tahap-tahapannya :

- Jumlah *support count* yang memenuhi *minimum support* akan dibangkitkan dengan *conditional fp-tree*.
    - Dalam *conditional fp-tree* untuk 1, ada 3 lintasan yaitu :

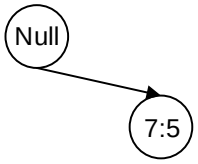
- Lintasan pertama : {7,28,22,1:3}
- Lintasan kedua : {7,28,1:2}
- Lintasan ketiga : {28,22,1:1}

- Pada lintasan yang berakhiran {1} dengan *support count* adalah 1 atau lebih yang dimilkinya akan dihilangkan dan *support count* yang melewati lintasan pertama yang berakhiran {1}, akan disamain dengan sesuai dengan *support count* yang dimilikinya..
- Proses selanjutnya adalah apabila dalam suatu lintasan, melintasi dua atau lebih, maka *support count*-nya akan selalu bertambah 1.

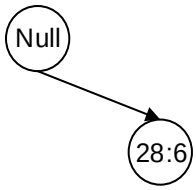


Gambar 3. 38 *Conditional FP-Tree* 1 Setelah Dibangkitkan

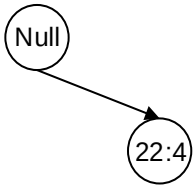
1. Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequent itemset* yang keluar yang memenuhi *minimum support*. 7:5 dan 28:6 adalah yang keluar yang dapat memenuhi *minimum support*. Dan *frequent itemset* yang didapatkan adalah {1, 7:5},{1,28:6}, {1, 22:4}



Gambar 3. 39 *Conditional FP-Tree* 1,7:5 Hasil *Frequent Itemset*

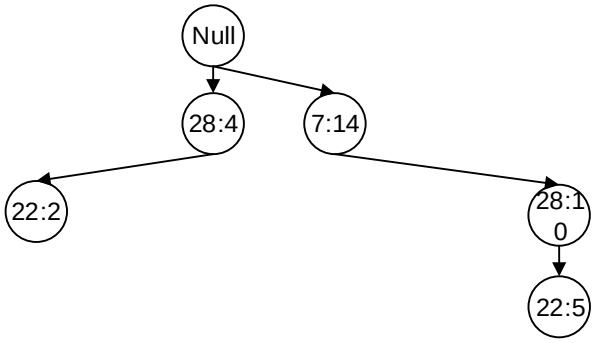


Gambar 3. 40 *Conditional FP-Tree* 1,28:6 Hasil *Frequent Itemset*



Gambar 3. 41 *Conditional FP-Tree* 1,22:4 Hasil *Frequent Itemset*

- *Conditional FP-Tree* untuk 22**:** {7,28,22:5},{28,22:2}

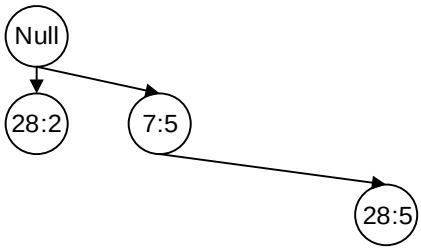


Gambar 3. 42 *Conditional FP-Tree* 22

Karena niilai *support count 2*2 adalah 7, maka memenuhi *minimum support* yaitu 4. Berikut adalah tahap-tahapannya :

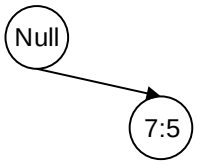
- Jumlah *support count* yang memenuhi *minimum support* akan dibangkitkan dengan *conditional fp-tree*.
- Dalam *conditional fp-tree* untuk 22, ada 2 lintasan yaitu :
    - Lintasan pertama : {28,22:2}
    - Lintasan kedua : {7,28,22:5}

- Pada lintasan yang berakhiran {22:1} dengan *support count* adalah 1 atau lebih dari 1 akan dihilangkan dan *support count* yang melewati lintasan yang berakhiran {22}, akan sama *support count*-nya dengan *support count* 22 sesuai dengan *support* yang dimilikinya.
- Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka *support count*-nya akan selalu bertambah 1

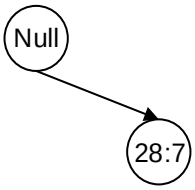


Gambar 3. 43 *Conditional FP-Tree* 22 setelah dibangkitkan

- Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequent itemset* yang keluar yang memenuhi *minimum support*. 7:5 dan 28:7 adalah yang keluar yang dapat memenuhi *minimum support*. Dan *frequent itemset* yang didapatkan adalah {22, 7:5},{22,28:7}

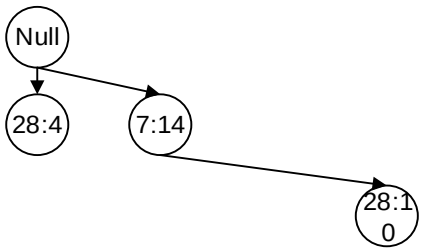


Gambar 3. 44 *Conditional FP-Tree* 22,7:5 Hasil *Frequent Itemset*



Gambar 3. 45 *Conditional FP-Tree* 22,28:7 Hasil *Frequent Itemset*

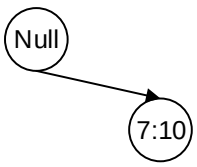
- *Conditional FP-Tree* untuk 28**:** {null, 28:4}, {7,28:10}



Gambar 3. 46 *Conditional FP-Tree* 28

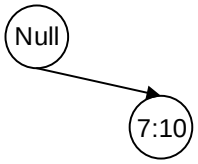
Karena niilai *support count* 28 adalah 14, maka memenuhi *minimum support* yaitu 4. Berikut adalah tahap-tahapannya :

- Jumlah *support count* yang memenuhi *minimum support* akan dibangkitkan dengan *conditional fp-tree*.
- Dalam *conditional fp-tree* untuk 28, ada 2 lintasan yaitu :
    - Lintasan pertama : {null,28:4}
    - Lintasan Kedua :{7,28:10}
- Pada lintasan yang berakhiran {28:1} dengan *support count* adalah 1 atau lebih dari 1 akan dihilangkan dan *support count* yang melewati lintasan yang berakhiran {28}, akan sama *support count*-nya dengan *support count* 28 sesuai dengan *support* yang dimilikinya.
- Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka *support count*-nya akan selalu bertambah 1.



Gambar 3. 47 *Conditional FP-Tree* 28 Setelah Dibangkitkan

- Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequent itemset* yang keluar yang memenuhi *minimum support*. 7:10 adalah yang keluar yang dapat memenuhi *minimum support*. Dan *frequent itemset* yang didapatkan adalah {28, 7:10}



Gambar 3. 48 *Conditional FP-Tree* 28,7:10 hasil *Frequent Itemset*

- *Conditional FP-Tree* untuk 7 : {null, 7:14}



Gambar 3. 49 *Conditional FP-Tree* 7

c. Tahap Pencarian *frequent itemset*

Setelah mendapatkan hasil dari *conditional fp tree* dari tabel *conditional pattern base*, ditemukan beberapa *frequent itemset* yang keluar. Hasilnya dapat dilihat apda tabel 3.7 dibawah ini :

Tabel 3. 7 Hasil *Conditional Fp-Tree*

Item	***Conditional FP-Tree***
21	Empty
13	Empty
11	Empty
10	Empty

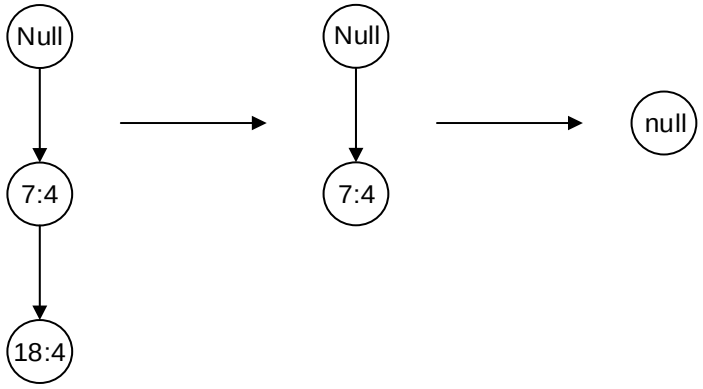
Item	*Conditional FP-Tree*
3	Empty
26	Empty
25	Empty
20	Empty
19	Empty
17	Empty
14	Empty
9	Empty
27	Empty
23	Empty
16	Empty
15	Empty
12	Empty
8	Empty
2	Empty
18	{18, 7:4}
6	{6, 7:4}
5	Empty
24	{24, 7:5} {24, 28:4}
4	{4, 7:5} {4, 28:5}, {4,22:4}
1	{1, 7:5} {1,28:6}, {1,22:4}
22	{22, 7:5} {22, 28:7}
28	{null} {28, 7:10}
7	{null}

Setelah membuat *conditional fp-tree* dari hasil pencarian *frequent itemset* maka dilakukan tahap berikutnya yaitu :

a. Akumulasikan *count* untuk setiap *item frequent itemset* yang ada pada tabel 3.7 diatas.

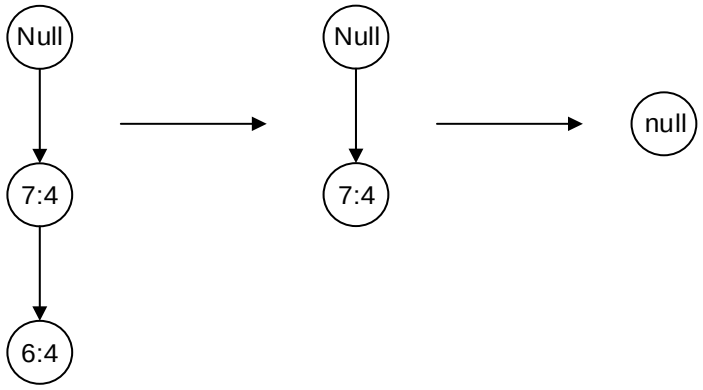
b. Buat *conditional fp-tree* untuk *frequent itemset* pada tabel 3.7 diatas.

- Untuk *Item* 18, *item frequent itemset* yang keluar adalah {18, 7:4}



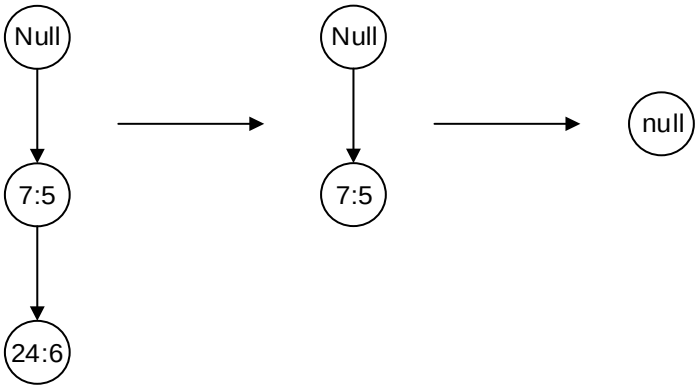
Gambar 3. 50 *Conditional FP-Tree* Untuk *Item Frequenset* 18, 7:4

- Untuk *item* 6, *item frequent itemset* yang keluar adalah {6, 7:4}

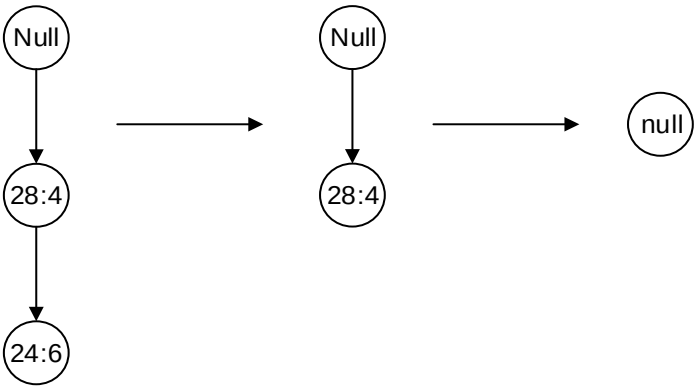


Gambar 3. 51 *Conditional FP-Tree* Untuk *Item Frequenset* 6, 7:4

- Untuk *item* 24, item *frequent itemset* yang keluar adalah {24, 7:5}, {24, 28:4}

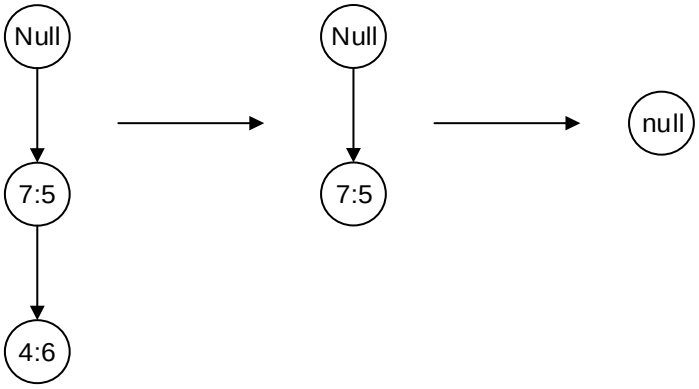


Gambar 3. 52 *Conditional FP-Tree* Untuk *Item Frequenset* 24, 7:5

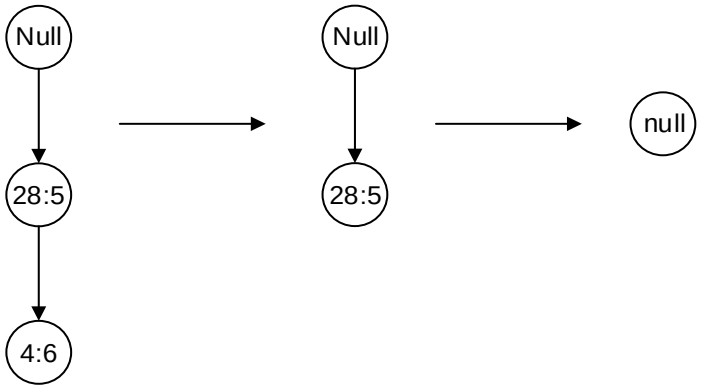


Gambar 3. 53 *Conditional FP-Tree* Untuk *Item Frequenset* 24, 28:4

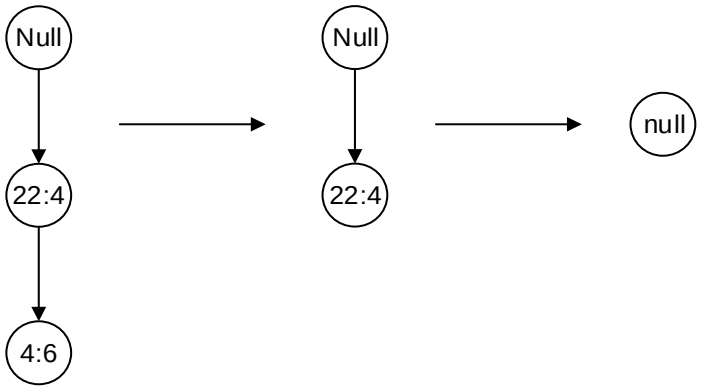
- Untuk *item* 4, *item frequent itemset* yang keluar adalah {4, 7:5}, {4, 28:5}



Gambar 3. 54 *Conditional FP-Tree* Untuk *Item Frequenset* 4, 7:5

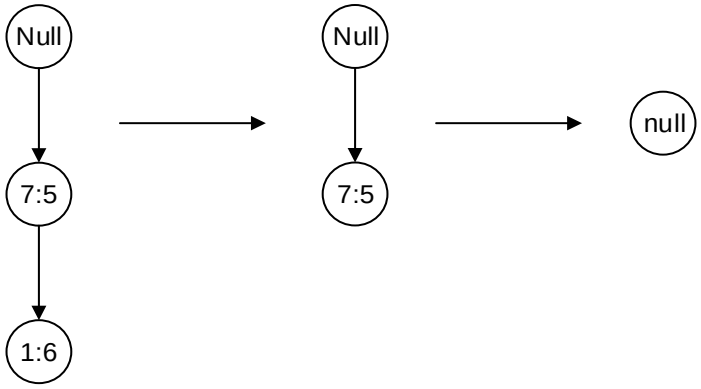


Gambar 3. 55 *Conditional FP-Tree* Untuk *Item Frequenset* 4, 28:5

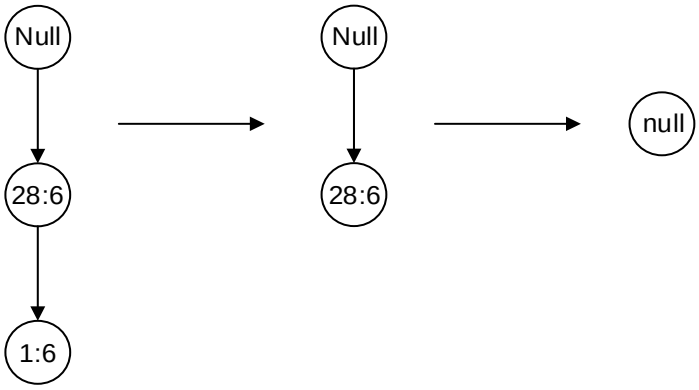


Gambar 3. 56 *Conditional FP-Tree* Untuk *Item Frequenset* 4, 22:4

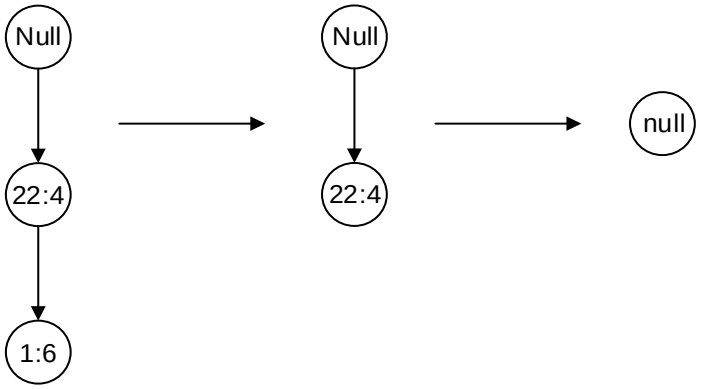
- Untuk *item* 1, *item frequent itemset* yang keluar adalah {1, 7:5} {1,28:6}, {1, 22:4}



Gambar 3. 57 *Conditional FP-Tree* Untuk *Item Frequenset* 1, 7:5

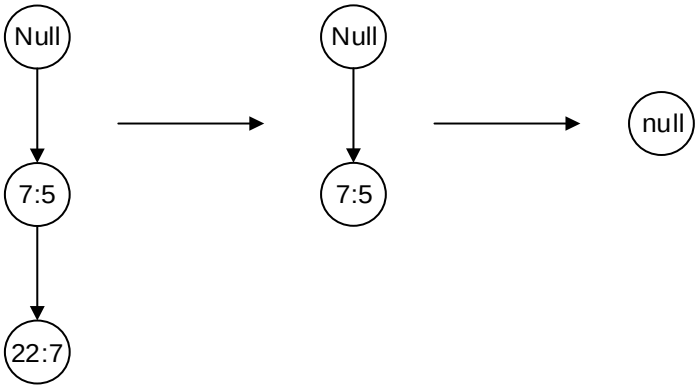


Gambar 3. 58 *Conditional FP-Tree* Untuk *Item Frequenset* 1, 28:6

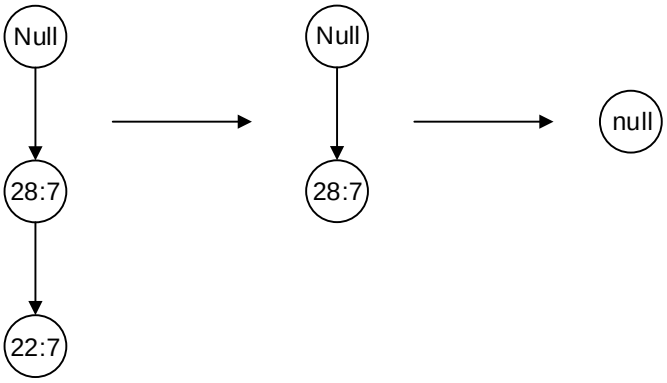


Gambar 3. 59 *Conditional FP-Tree* Untuk *Item Frequenset* 1, 22:4

- Untuk *item* 22, item *frequent itemset* yang keluar adalah {22, 7:5} {22, 28:7}

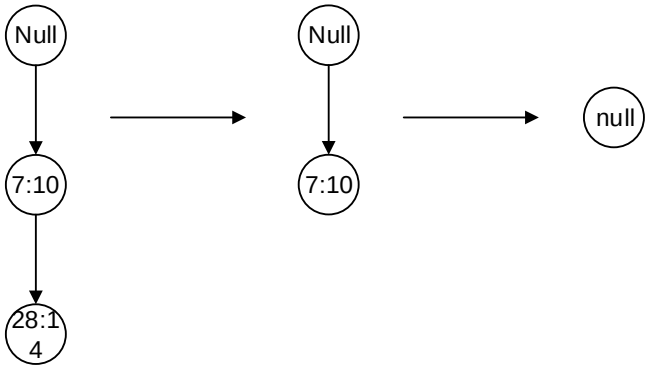


Gambar 3. 60 *Conditional FP-Tree* Untuk *Item Frequenset* 22, 7:5



Gambar 3. 61 *Conditional FP-Tree* Untuk *Item Frequenset* 22, 28:7

- Untuk *item* 28, *item frequent itemset* yang keluar adalah {28, 7:10}



Gambar 3. 62 *Conditional FP-Tree* Untuk *Item Frequenset* 28, 7:10

7. Setelah mendapatkan hasil *frequent itemset* dari tiap *condiitonal fp-tree,* selanjutnya adalah menentukan *frequent itemset* setiap *suffix*

Tabel 3. 8 Hasil *Frequent Itemset*

Suffix	***Frequent Itemset***
18	18 - 7
6	6 - 7
24	24 – 7 24 - 28

Suffix	*Frequent Itemset*
4	4 - 7 4 – 28 4 - 22
1	1 - 7 1 – 28 1 - 22
22	22 - 7 22 - 28
28	28 - 7

Dari hasil *frequent Itemset* diatas akan di-*generate* untuk mendapatkan *rule* dengan *minimum support* 4 dan *minimum confidence* 60% dengan masing-masing *itemset* dikombinasikan dengan *itemset* lain. Kenapa nilai *confidence* 60% karena hasil *frequent itemset* yang keluar adalah hasil yang sering paling muncul dalam data transaksi, jadi penyusunan layout makanannya bisa diatur sesuai dengan *frequent* yang muncul sesuai dengan nilai *confidence*. Dengan menggunakan persamaan (2.2) yang ada di bab 2 untuk memperoleh nilai *support* dan persamaan(2.4) untuk memperoleh nilai *confidence.* Maka *rule* yang sudah didapat akan diberikan nilai *support* dan *confidence* menggunakan persamaan yang sudah disebutkan di atas. Berikut ini adalah hasil dari pemberian nilai untuk *support* dan *confidence* dari tiap *rule.*

Tabel 3. 9 Hasil Perhitungan *Support dan Confidence*

Rule	*Minimun Support*	*Minimum Confidence*
18→7	4/19*100=21,05	4/4*100=100%
7→18	4/19*100 = 21,05	4/14*100=28,57%
6→7	4/19*100=21,05	4/6*100=66,66%
7→6	4/19*100=21,05	4/14*100=28,57%
24→7	5/19*100=26,31	5/6*100=83,33%
7→24	5/19*100=26,31	5/14*100=35,71%

Rule	***Minimun Support***	***Minimum Confidence***
24→28	4/19*100=21,05	4/6*100=66,66%
28→24	4/19*100=21,05	4/14*100=28,57%
4→7	5/19*100=26,31	5/6*100=83,33%
7→4	5/19*100=26,31	5/14*100=35,71%
4→28	5/19*100=26,31	5/6*100=83,33%
28→4	5/19*100=26,31	5/14*100=35,71%
4→22	4/19*100=21,05	4/6*100=66,66%
22→4	4/19*100=21,05	4/7*100=57,14%
1→7	5/19*100=26,31	5/6*100=83,33%
7→1	5/19*100=26,31	5/14*100=35,71%
1→28	6/19*100=31,57	6/6*100=100%
28→1	6/19*100=31,57	6/14*100=42,85%
1→22	4/19*100=21,05	4/6*100=66,66%
22→1	4/19*100=21,05	4/7*100=57,14%
22→7	5/19*100=26,31	5/7*100=71,42%
7→22	5/19*100=26,31	5/14*100=35,71%
22→28	7/19*100=36,84	7/7*100=100%
28→22	7/19*100=36,84	7/14*100=50%
28→7	10/19*100=52,63	10/14*100=71,42%
7→28	10/19*100=52,63	10/14*100=71,42%

Setelah melakukan perhitungan *minimum support* dan *minimum confidence* pada tabel diatas, proses selanjutnya yang akan dilakukan adalah mencari *generate rule* yang memenuhi nilai *minimum support* dan *confidence* yang telah ditentukan sebelumnya. Jika terdapat *itemset* yang tidak memenuhi *suppor*t dan *confidence* akan dihapus.

Tabel 3. 10 Hasil *Generate Rule*

Rule	***Minimum Support***	***Minimum Confidence***
18→7	4/19*100=21,05	4/4*100=100%
6→7	4/19*100=21,05	4/6*100=66,66%
24→7	5/19*100=26,31	5/6*100=83,33%
24→28	4/19*100=21,05	4/6*100=66,66%
4→7	5/19*100=26,31	5/6*100=83,33%
4→28	5/19*100=26,31	5/6*100=83,33%
4→22	4/19*100=21,05	4/6*100=66,66%
1→7	5/19*100=26,31	5/6*100=83,33%
1→28	6/19*100=31,57	6/6*100=100%
1→22	4/19*100=21,05	4/6*100=66,66%
22→7	5/19*100=26,31	5/7*100=71,42%
22→28	7/19*100=36,84	7/7*100=100%
28→7	10/19*100=52,63	10/14*100=71,42%
7→28	10/19*100=52,63	10/14*100=71,42%

Berdasarkan hasil *rule* yang didapatkan, maka pihak pengelola rumah makan Roso Echo dapat menyusun *layout* makanan sesuai dengan *rule* yang sudah didapatkan. Dalam penysusunan tabel dibawah ini hasil penysunan *layout makanan* berdasarkan *confidence*. Berikut hasil penyusunan *layout* makanan dalam bentuk awal.

Tabel 3. 11 Hasil Penyusunan *Layout* Makanan Dalam Bentuk Awal

Rule	**Susunan Layout Makanan**	***Confidence***
18→7	Oseng Jamur → Gorengan / Bacem	100%
6→7	Garang Asam→ Gorengan / Bacem	66,66%
24→7	Sate Jamur→ Gorengan / Bacem	83,33%
24→28	Sate Jamur→ Tumisan / Sayur /Urap Mie	66,66%
4→7	Cumi→ Gorengan / Bacem	83,33%

Rule	Susunan Layout Makanan	*Confidence*
4→28	Cumi→ Tumisan / Sayur /Urap Mie	83,33%
4→22	Cumi→ Sambal Goreng Krecek	66,66%
1→7	Ayam Bakar / Cabe Ijo→ Gorengan / Bacem	83,33%
1→28	Ayam Bakar / Cabe Ijo→ Tumisan / Sayur /Urap Mie	100%
1→22	Ayam Bakar / Cabe Ijo→ Sambal Goreng Krecek	66,66%
22→7	Sambal Goreng Krecek→ Gorengan / Bacem	71,42%
22→28	Sambal Goreng Krecek→ Tumisan / Sayur /Urap Mie	100%
28→7	Tumisan / Sayur /Urap Mie → Gorengan / Bacem	71,42%
7→28	Gorengan / Bacem→ Tumisan / Sayur /Urap Mie	71,42%

3.1.4.4.2 Pemodelan Untuk Penentuan Paket Makanan Hemat

Merupakan tahap untuk membuat model atau desain sistem yang akan dibangun. Data yang digunakan untuk pengolahan data adalah data menu pokok dan menu sampingan yang terdiri dari yaitu jenis makanan, jenis lain-lain dan jenis minuman. Di dalam tahap ini akan dipilih metode yang cocok untuk menyelesaikan masalah di rumah makan Roso Echo. Metode yang akan digunakan adalah *association rule* dengan algoritma *fp-growth*.

Data yang dipakai untuk penentuan paket makanan hemat adalah pada tabel D-41 dalam lampiran D Data tersebut akan diolah dengan seberapa banyak NamaMenu yang muncul. Hasil pengolahan dapat dilihat pada tabel 3.12 dibawah ini :

Tabel 3. 12 *Frequent Item*

Item	***Frequent***
1	6
2	3
3	1
4	6
5	4
6	2
7	4
8	14
9	3
10	2
11	1
12	2
13	1
14	1
15	1
16	1
17	1

Item	*Frequent*
18	2
19	1
20	1
21	3
22	1
23	2
24	17
25	4
26	1
27	2
28	3
29	3
30	2
31	4
32	2
33	2
34	3
35	1
36	7
37	1
38	3
39	6
40	2
41	3
42	3
43	1
44	2
45	3
46	14

Item	*Frequent*
47	1
48	1

Dimisalkan nilai *minimum support* yang digunakan dalam contoh kasus ini adalah sebesar 5, karena dengan nilai *minimum support* tersebut tujuan *data mining* yaitu untuk penentuan paket makanan hemat dan pencarian *frequent itemset* dapat dimaksimalkan. Kenapa *minimum support* diambil 5, supaya pada proses penentuan paket makanan hemat yaitu yang menu pokok dan menu sampingan yang terdiri jenis makanan, jenis lain-lain dan jenis minuman terlihat jelas dipaketin sesuai item yang keluar pada proses berikutnya. Dan bisa dilihat dari tabel 3.12 diatas bahwa *frequent- frequent* yang besar itu sebagai acuan dalam penentuan paket makanan hemat, karena *frequent* yang jarang muncul jarang dibeli oleh kosumen.

1. Langkah pertama adalah men-*scan* data transaksi yang ada pada tabel D-41 dalam lampiran D, untuk mengetahui *support count* dan nilai *support* dari setiap menu makanan yang ada. Hasil *support count* didapatkan dari hasil kemunculan *frequent* yang ada pada tabel 3.12 diatas, sedangkan nilai *support* didapat menggunakan persamaan 2.1 yang terdapat pada bab 2. Berikut adalah hasil dari proses *scan* yang telah dilakukan :

Tabel 3. 13 Tabel Hasil *Scan*

KodeMenu	*Support Count*	Nilai *Support*
1	6	(6/22)*100 = 27,27
2	3	(3/22)*100 = 13,63
3	1	(1/22)*100 = 4,54
4	6	(6/22)*100 = 27,27
5	4	(4/22)*100 = 18,18
6	2	(2/22)*100 = 9,09
7	4	(4/22)*100 = 18,18

KodeMenu	*Support Count*	Nilai *Support*
8	14	(14/22)*100 = 63.63
9	3	(3/22)*100 = 13,63
10	2	(2/22)*100 = 9.09
11	1	(1/22)*100 = 4,54
12	2	(2/22)*100 = 9.09
13	1	(1/22)*100 = 4,54
14	1	(1/22)*100 = 4,54
15	1	(1/22)*100 = 4,54
16	1	(1/22)*100 = 4,54
17	1	(1/22)*100 = 4,54
18	2	(2/22)*100 = 9.09
19	1	1/22)*100 = 4,54
20	1	(1/22)*100 = 4,54
21	3	(3/22)*100 = 13,63
22	1	(1/22)*100 = 4,54
23	2	(2/22)*100 = 9.09
24	17	(17/22)*100 = 77,27
25	4	(4/22)*100 = 18,18
26	1	(1/22)*100 = 4,54
27	2	(2/22)*100 = 9.09
28	3	(3/22)*100 = 13,63
29	3	(3/22)*100 = 13,63
30	2	(2/22)*100 = 9.09
31	4	(4/22)*100 = 18,18
32	2	(2/22)*100 = 9.09
33	2	(2/22)*100 = 9.09
34	3	(3/22)*100 = 13,63
35	1	(1/22)*100 = 4,54
36	7	(7/22)*100 = 31,81

KodeMenu	*Support Count*	Nilai *Support*
37	1	(1/22)*100 = 4,54
38	3	(3/22)*100 = 13,63
39	6	(6/22)*100 = 27,27
40	2	(2/22)*100 = 9,09
41	3	(3/22)*100 = 13,63
42	3	(3/22)*100 = 13,63
43	1	(1/22)*100 = 4,54
44	2	(2/22)*100 = 9,09
45	3	(3/22)*100 = 13,63
46	14	(14/22)*100 = 63,63
47	1	(1/22)*100 = 4,54
48	1	(1/22)*100 = 4,54

2. Langkah kedua yaitu menghilangkan data yang memiliki nilai *support yang* kurang dari *minimum support* yang ditentukan sebelumnya yaitu 5.

Tabel 3. 14 *Support* Yang Memenuhi

KodeMenu	*Support Count*	Nilai *Support*
1	6	(6/22)*100 = 27,27
2	3	(3/22)*100 = 13,63
4	6	(6/22)*100 = 27,27
5	4	(4/22)*100 = 18,18
6	2	(2/22)*100 = 9,09
7	4	(4/22)*100 = 18,18
8	14	(14/22)*100 = 63.63
9	3	(3/22)*100 = 13,63
10	2	(2/22)*100 = 9.09
12	2	(2/22)*100 = 9.09
18	2	(2/22)*100 = 9.09

KodeMenu	*Support Count*	Nilai *Support*
21	3	(3/22)*100 = 13,63
23	2	(2/22)*100 = 9.09
24	17	(17/22)*100 = 77,27
25	4	(4/22)*100 = 18,18
27	2	(2/22)*100 = 9.09
28	3	(3/22)*100 = 13,63
29	3	(3/22)*100 = 13,63
30	2	(2/22)*100 = 9.09
31	4	(4/22)*100 = 18,18
32	2	(2/22)*100 = 9.09
33	2	(2/22)*100 = 9.09
34	3	(3/22)*100 = 13,63
36	7	(7/22)*100 = 31,81
38	3	(3/22)*100 = 13,63
39	6	(6/22)*100 = 27,27
40	2	(2/22)*100 = 9,09
41	3	(3/22)*100 = 13,63
42	3	(3/22)*100 = 13,63
44	2	(2/22)*100 = 9,09
45	3	(3/22)*100 = 13,63
46	14	(14/22)*100 = 63,63

3. Setelah didapatkan data yang memenuhi nilai *support* seperti pada tabel 3.14 diatas, selanjutnya mencari *frequency of occurrence*. Urutkan tabel 3.14 berdasarkan *frequent* terbesar sampai dengan terkecil, kemudian memberikan *priority* dimulai dari data frekeunsi terbesar sampai dengan terkecil.

Tabel 3. 15 ***Frequency Of Occurrence***

Item	***Frequent***	***Priority***
24	17	1
8	14	2
46	14	3
36	7	4
1	6	5
4	6	6
39	6	7
5	4	8
7	4	9
25	4	10
31	4	11
2	3	12
9	3	13
21	3	14
28	3	15
29	3	16
34	3	17
38	3	18
41	3	19
42	3	20
45	3	21
6	2	22
10	2	23
12	2	24
18	2	25
23	2	26
27	2	27

Item	*Frequent*	*Priority*
30	2	28
32	2	29
33	2	30
40	2	31
44	2	32

4. Setelah mendapatkan hasil dari *frequency of occurrence* proses selanjutnya adalah mengurutkan data yang ada pada tabel D-41 dalam lampiran D sesuai dengan hasil penentuan *priority* pada tabel 3.15 diatas. Tetapi sebelum pengurutan data yang ada pada tabel D-41, dihilangkan data yang tidak memenuhi *minimum support* yang tidak memenuhi. Berikut adalah hasil dari pengurutan data yang telah dilakukan :

Tabel 3. 16 Hasil Pengurutan data Berdaarkan Priority

NoBill	NamaMenu
006346	6, 18, 23
006467	6, 18
006477	42
006523	24, 46, 28, 41, 44
006093	24, 8, 39, 5
006097	24, 8, 4, 31, 27
006111	24, 8, 39, 7, 2
006715	23
006752	24, 8, 46, 36, 1, 4, 9, 34, 45, 10, 30, 33, 44
006754	24, 8, 46, 1, 41
006553	24, 8, 46, 36, 1, 39, 7, 31, 2, 9, 38, 45, 10, 27, 40
006554	24, 46, 5, 28
006555	24, 8, 46, 36, 29, 38, 32
006556	24, 8, 46
006557	24, 46, 36, 39, 5, 25, 2, 21
006558	8, 46, 36, 4, 7,25, 32
006559	24, 8, 31, 30
007072	24, 46, 36, 1, 4, 9, 21, 28, 29, 34
007073	24, 8, 46, 4, 42
007074	24, 8, 46, 1, 42

NoBill	NamaMenu
007075	24, 8, 46, 39, 5, 25, 34, 38, 12, 40
007076	24, 8, 46, 36, 1, 4, 39, 7, 25, 31, 21, 29, 41, 45, 12, 33

5. Setelah mendapatkan data yang telah diurutkan, selanjutnya membuat *fp-tree*. Berikut adalah tahap-tahap dalam pembuatan *fp-tree* yaitu :
    a. Lakukan pembacaan pada data pertama yang akan diproses.
    b. Lakukan pengecekan pada *root*, jika root belum mempunyai anak (*child)*, maka data yang dibaca tadi menjadi anak dari *root*. Dan *support count* dari masing masing *node* bertambah 1, kecuali *root*.
    c. 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.
    d. Lakukan pengecekan pada *root*, jika *root* sudah mempunyai anak, maka 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 bertambah 1.
    e. Lakukan langkah tersebut hingga semua data yang keluar.
    f. Jika ada data yang sama setelah pembacaan, namun berbeda *lintasan* maka *nod*e pada *laintasan* yang lain tidak bertambah *support* count-nya, hanya data yang dibaca saja yang bertambah *support count*-nya.

    Pembentukan *fp-tree* paket makanan hemat dapat dilihat pada tabel D-42 dalam lampiran D.

6. Setelah tahap pembangunan *fp-tree* dari sekumpulan data transaksi dilakukan, maka akan diterapkan algoritma *fp-growth* untuk mencari *frequent itemset* yang signifikan. Berikut langkah-langkah utama algoritma *fp-growth* :

a. Tahap Pembangkitan *Conditional Pattern Base*

Pembangkitan *conditional pattern base* didapatkan melalui hasil *fp-tree* seluruhnya, dengan mencari *support count* terkecil sampe terbesar sesuai dengan hasil pencarian *priority* pada tabel 3.15.Dan telah didapatkan yaitu : 44,40,33,32,30,27,23,18,12,10,6,45,42,41,38,34,29,28,21,9,2,31,25,7,5 ,39,4,1,36,46,8,24. Berikut adalah hasil tabel dari *conditional pattern base* :

Tabel 3. 17 *Conditional Pattern Base*

Item	***Conditional Pattern Base***
44	• 24,46,28,41,44:1 • 24,8,46,36,1,4,9,34,45,10,30,33,44:1
40	• 24,8,46,36,1,39,7,31,2,9,38,45,10,27,40:1 • 24,8,46,39,5,25,34,38,12,40:1
33	• 24,8,46,36,1,4,9,34,45,10,30,33:1 • 24,8,46,36,1,4,39,7,25,31,21,41,29,45,12,33:1
32	• 24,8,46,36,29,38,32:1 • 8,46,36,4,7,25,32:1
30	• 24,8,46,36,1,4,9,34,45,10,30:1 • 24,8,31,30:1
27	• 24,8,4,31,27:1 • 24,8,46,36,1,39,7,31,2,9,38,45,10,27:1
23	• 6,18,23:1 • Null, 23:1
18	• 6,18:1 • 6,18:1
12	• 24,8,46,36,1,4,39,7,25,31,21,41,29,45,12:1 • 24,8,46,39,5,25,34,38,12:1
10	• 24,8,46,36,1,4,9,34,45,10:1

Item	*Conditional Pattern Base*
	• 24,8,46,36,1,39,7,31,2,9,38,45,10:1
6	• Null, 6:2
45	• 24,8,46,36,1,4,9,34,45:1 • 24,8,46,36,1,4,39,7,25,31,21,41,29,45 • 24,8,46,36,1,39,7,31,2,9,38,45:1
42	• Null, 42:1 • 24,8,46,1,42:1 • 24,8,46,4,42:1
41	• 24,46,28,41:1 • 24,8,46,36,1,4,39,7,25,31,21,41:1 • 4,8,46,1,41:1
38	• 24,8,46,36,1,39,7,31,2,9,38:1 • 24,8,46,29,38:1 • 24,8,46,39,5,25,34,38:1
34	• 24,46,36,1,4,9,21,28,29,34:1 • 24,8,46,36,1,4,9,34:1 • 24,8,46,39,5,25,34:1
29	• 24,46,36,1,4,9,21,28,29:1 • 24,8,46,36,1,4,39,7,25,31,21,41,29:1 • 24,8,46,36,29:1
28	• 24,46,28:1 • 24,46,5,28:1 • 24,46,36,1,4,9,21,28:1
21	• 24,46,36,39,5,25,2,21:1 • 24,46,36,1,4,9,21:1 • 8,46,36,1,4,39,7,25,31,21:1
9	• 24,46,36,1,4,9:1 • 24,8,46,36,1,4,9:1

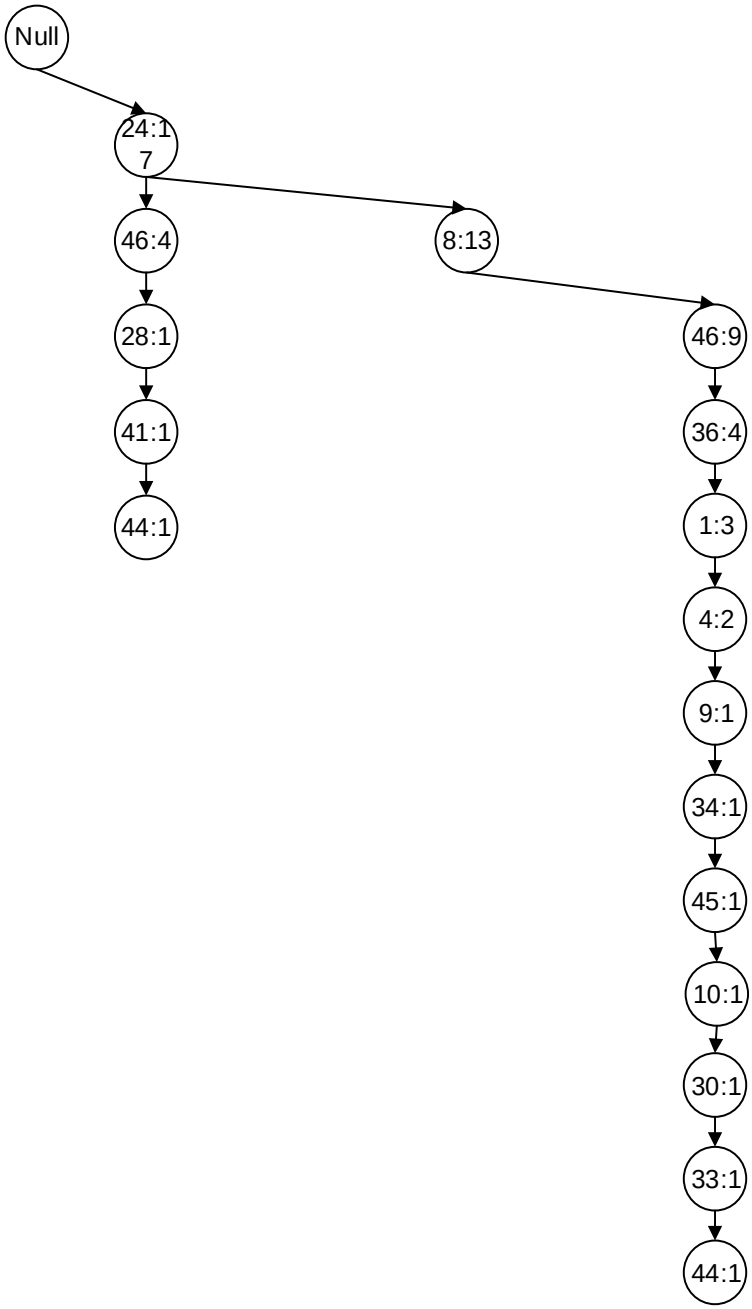
Item	*Conditional Pattern Base*
	• 24,8,46,36,1,39,7,31,2,9:1
2	• 24,46,36,39,5,25,2:1 • 24,8,39,7,2:1 • 24,8,46,36,1,39,7,31,2:1
31	• 24,8,4,31:1 • 24,8,46,36,1,4,39,7,25,31:1 • 24,8,46,36,1,39,7,31:1 • 24,8,31:1
25	• 24,46,36,39,5,25:1 • 24,8,46,36,1,4,39,7,25:1 • 24,8,46,39,5,25:1 • 8,46,36,4,7,25:1
7	• 24,8,39,7:1 • 24,8,46,36,1,4,39,7:1 • 24,8,46,36,1,39,7:1 • 8,46,36,4,7:1
5	• 24,46,5:1 • 24,46,36,39,5:1 • 24,8,39,5:1 • 24,8,46,39,5:1
39	• 24,46,36,39:1 • 24,8,39:2 • 24,8,46,36,1,4,39:1 • 24,8,46,36,1,39:1 • 24,8,46,39:1
4	• 24,46,36,1,4:1 • 24,8,4:1 • 24,8,46,36,1,4:2

Item	***Conditional Pattern Base***
	• 24,8,46,4:1 • 8,46,36,4:1
1	• 24,46,36,1:1 • 24,8,46,36,1:3 • 24,8,46,1:2
36	• 24,46,36:2 • 24,8,46,36:4 • 8,46,36:1
46	• 24,46:4 • 24,8,46:9 • 8,46:1
8	• 24,8:13 • Null, 8:1
24	• Null, 24:17

b. Tahap Pembangkitan *Conditional FP-Tree*

Pada tahap ini, *support count* dari setiap *item* pada tabel *conditional pattern base* diatas dijumlahkan, lalu setiap *item* yang memiliki jumlah *support count* lebih besar atau sama dengan *minimum support* akan dibangkitkan dengan *conditional fp-tree.*

- *Conditional FP-Tree* untuk 44 : {24,46,28,41,44:1} {24,8,46,36,1,4,9,34,45,10,30,33,44:1}



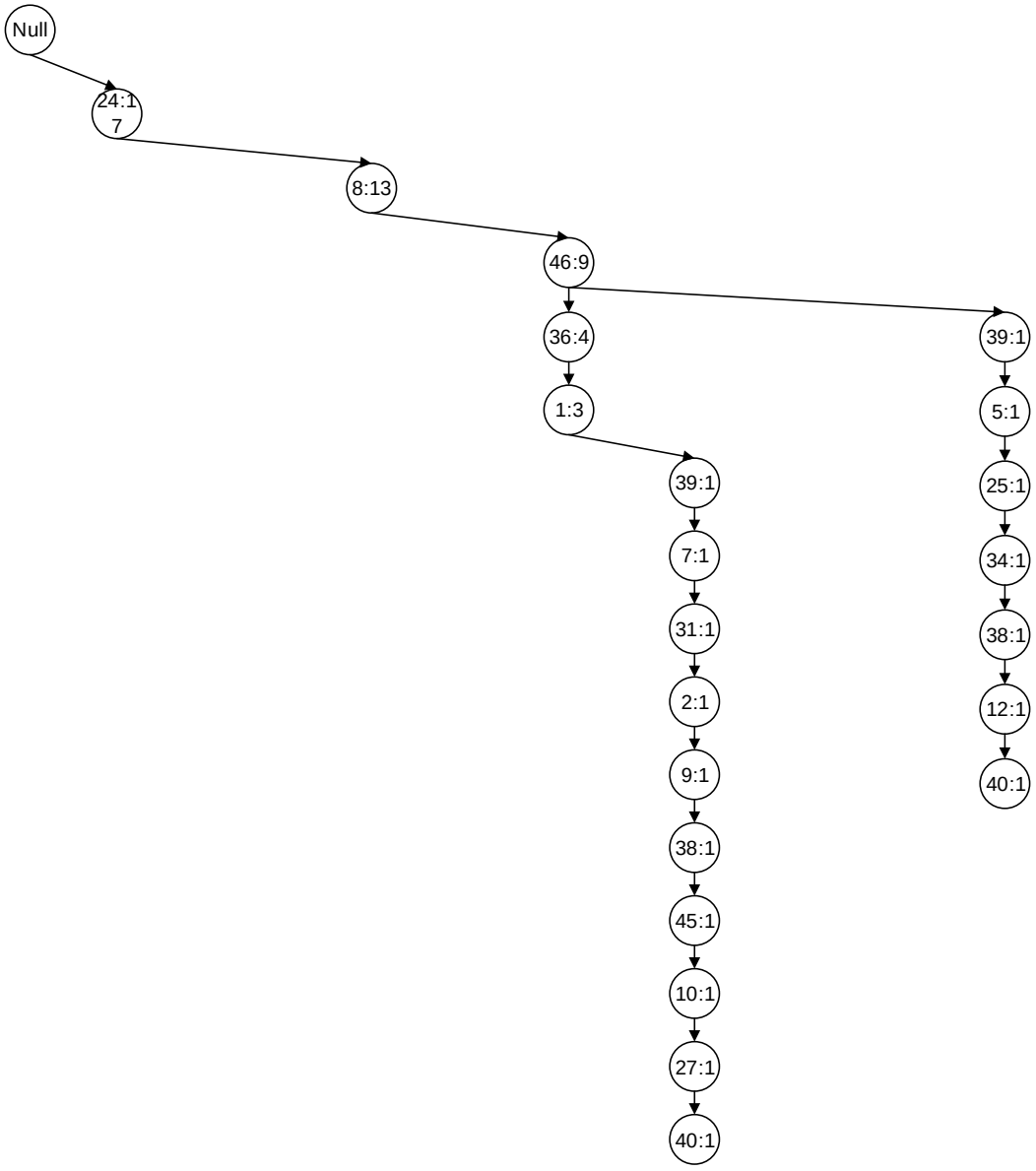
Gambar 3. 63 *Conditional FP-Tree Untuk* 44

Karena niilai *support count* 44 adalah 2, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

- *Conditional FP-Tree* untuk 40

  {24,8,46,36,1,39,7,31,2,9,38,45,10,27,40:1}

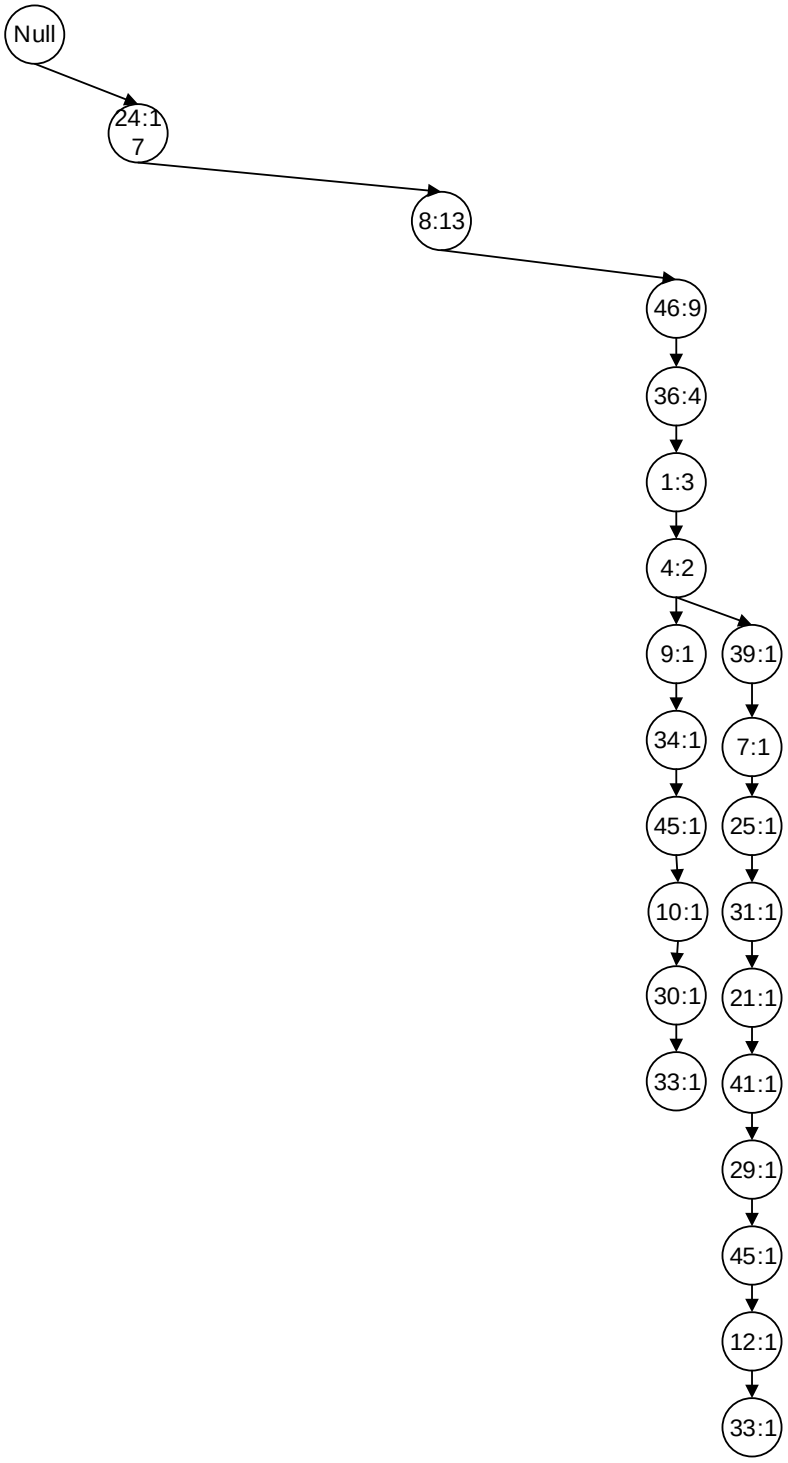
  {24,8,46,39,5,25,34,38,12,40:1}



Gambar 3. 64 *Conditional FP-Tree Untuk* 40

Karena niilai *support count* 40 adalah 2, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

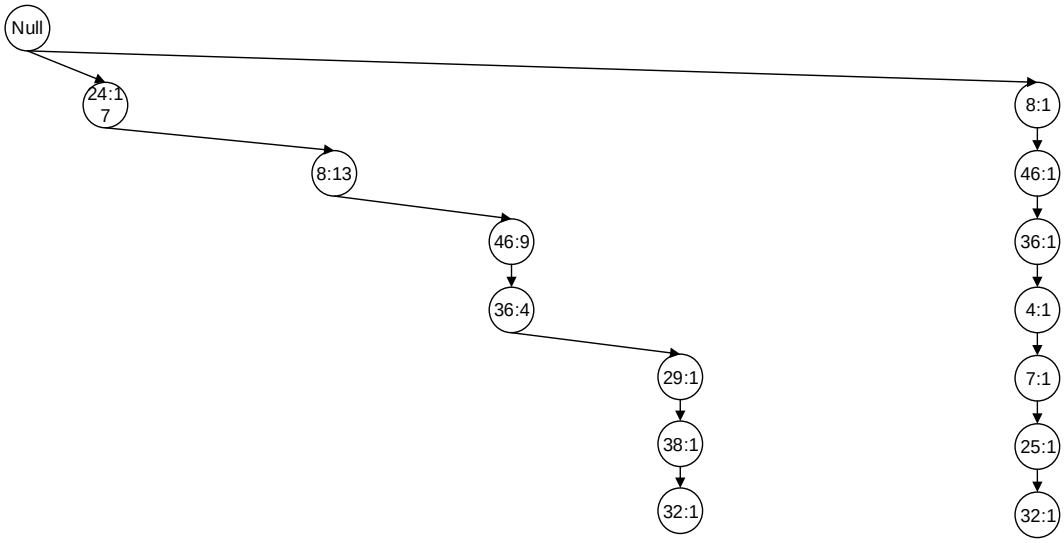
- *Conditional FP-Tree* untuk 33:{24,8,46,36,1,4,9,34,45,10,30,33:1}
  {24,8,46,36,1,4,39,7,25,31,21,41,29,45,12,33:1}



Gambar 3. 65 *Conditional FP-Tree Untuk* 33

Karena niilai *support count* 33 adalah 2, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

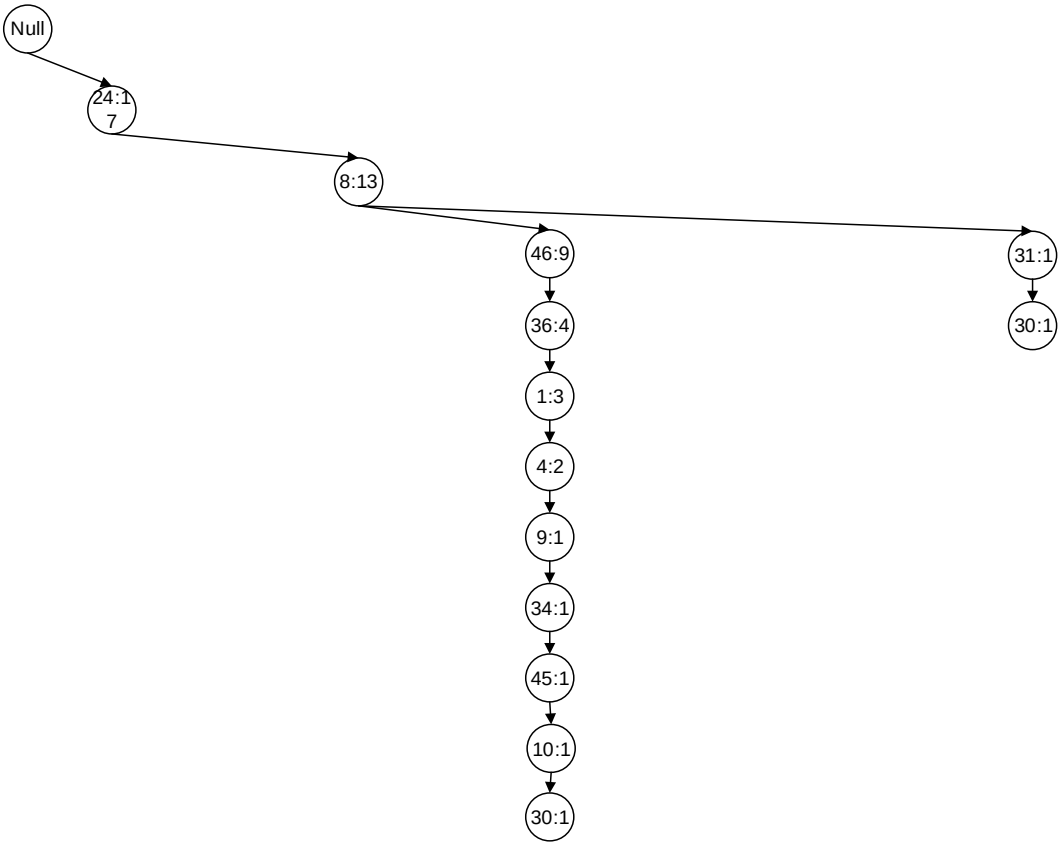
- *Conditional FP-Tree* untuk 32 : {24,8,46,36,29,38,32:1}
  {8,46,36,4,7,25,32:1}



Gambar 3. 66 *Conditional FP-Tree Untuk* 32

Karena niilai *support count* 32 adalah 2, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

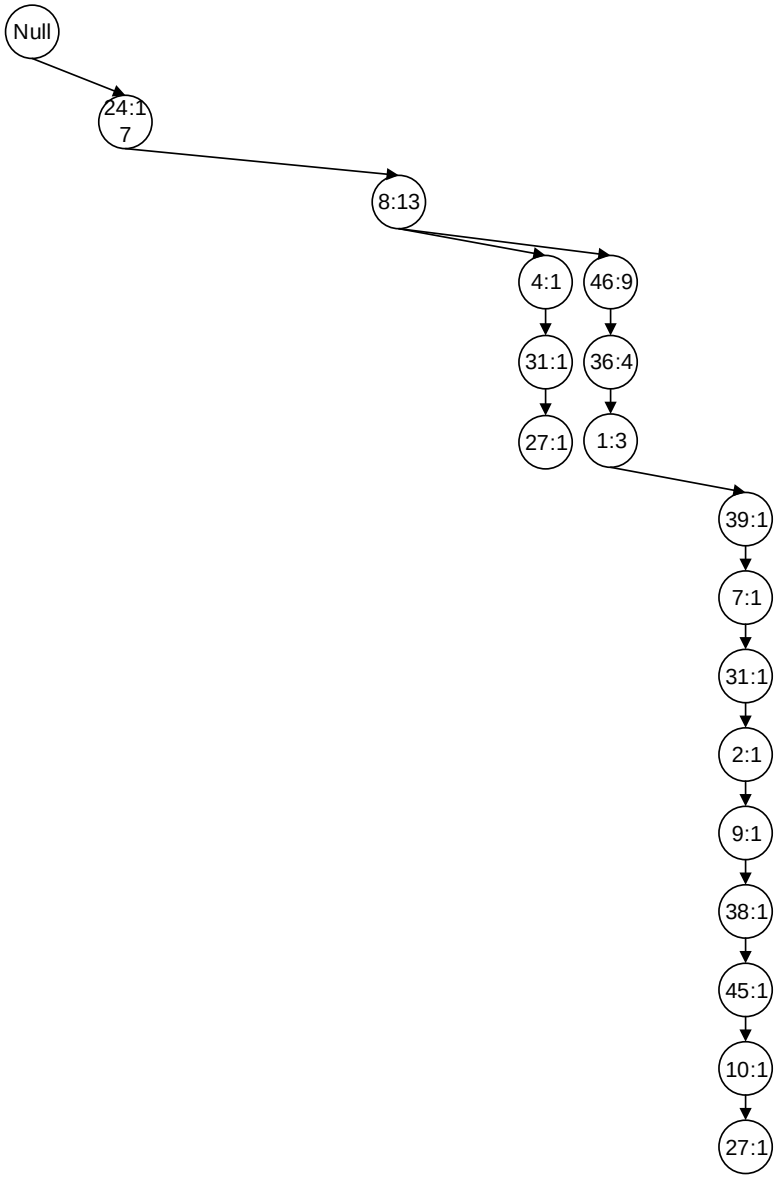
- *Conditional FP-Tree* untuk 30 : {24,8,46,36,1,4,9,34,45,10,30:1}
  {24,8,31,30:1}



Gambar 3. 67 *Conditional FP-Tree Untuk* 30

Karena niilai *support count* 30 adalah 2, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

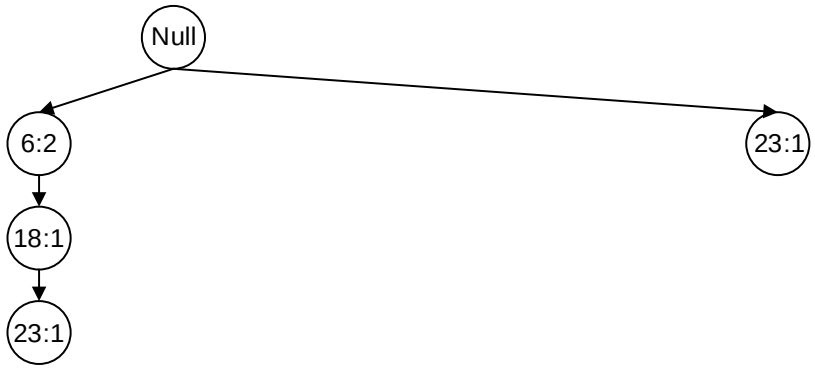
- *Conditional FP-Tree* untuk 27 : {24,8,4,31,27:1}
  {24,8,46,36,1,39,7,31,2,9,38,45,10,27:1}



Gambar 3. 68 *Conditional FP-Tree Untuk* 27

Karena niilai *support count* 27 adalah 2, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

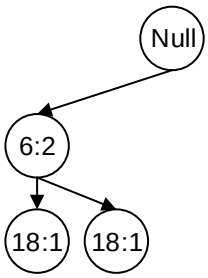
- *Conditional FP-Tree* untuk 23 : {6,18,23:1} {Null, 23:1}



Gambar 3. 69 *Conditional FP-Tree Untuk* 23

Karena niilai *support count* 23 adalah 2, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

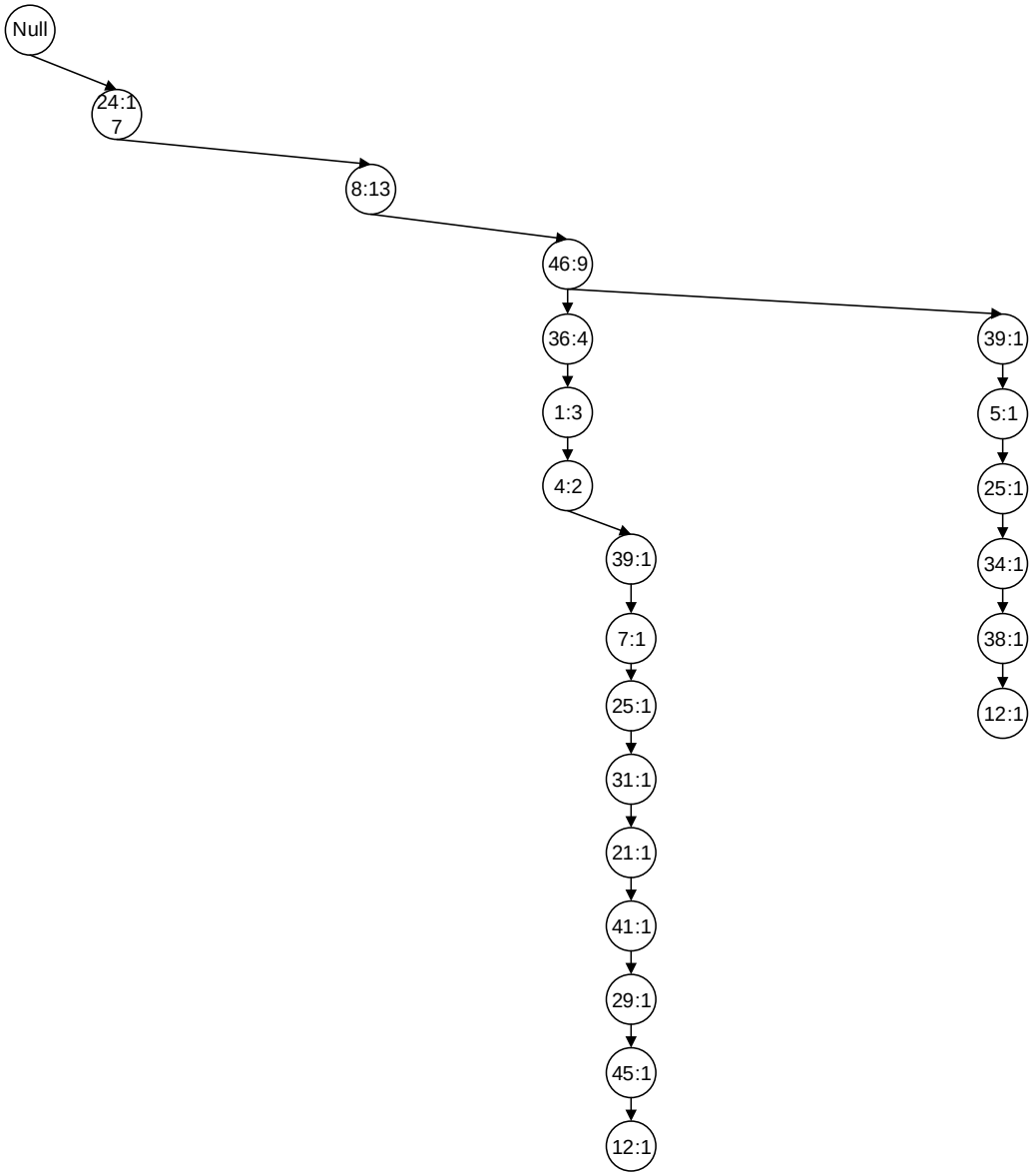
- *Conditional FP-Tree* untuk 18 : {6,18:1} {6,18:1}



Gambar 3. 70 *Conditional FP-Tree Untuk* 18

Karena niilai *support count* 18 adalah 2, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

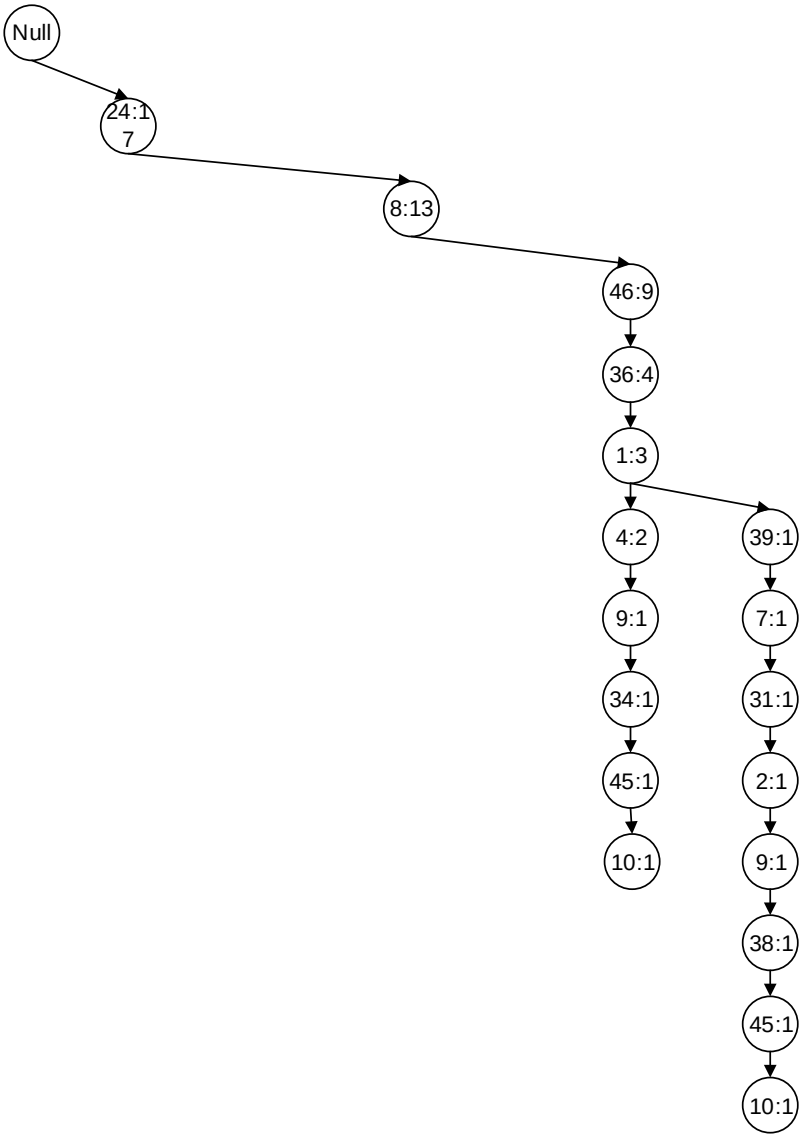
- *Conditional FP-Tree* untuk 12 : {24,8,46,36,1,4,39,7,25,31,21,41,29,45,12:1}{24,8,46,39,5,25,34,38,12:1}



Gambar 3. 71 *Conditional FP-Tree Untuk* 12

Karena niilai *support count* 12 adalah 2, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

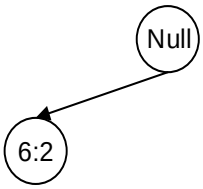
- *Conditional FP-Tree* untuk 10 : {24,8,46,36,1,4,9,34,45,10:1}
  {24,8,46,36,1,39,7,31,2,9,38,45,10:1}



Gambar 3. 72 *Conditional FP-Tree Untuk* 10

Karena niilai *support count* 10 adalah 2, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

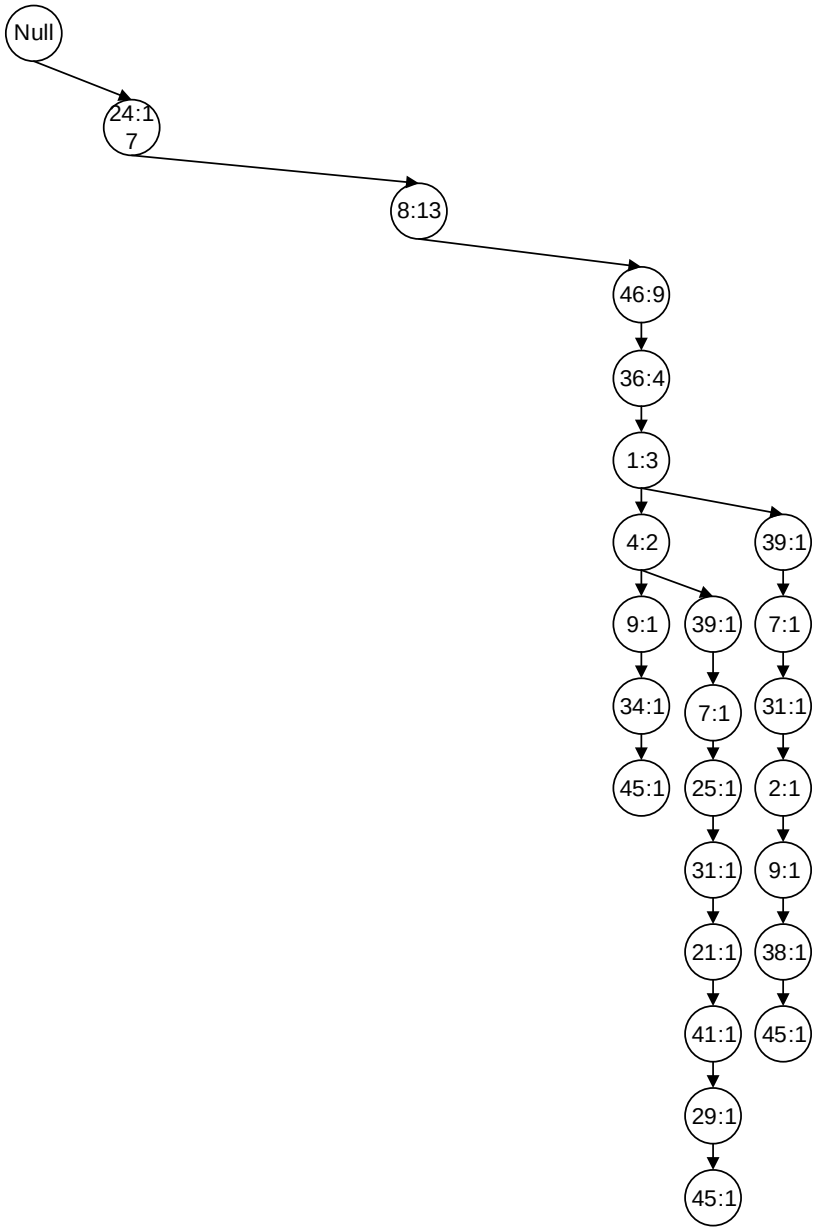
- *Conditional FP-Tree* untuk 6 : {Null, 6:2}



Gambar 3. 73 *Conditional FP-Tree Untuk* 6

Karena niilai *support count* 6 adalah 2, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

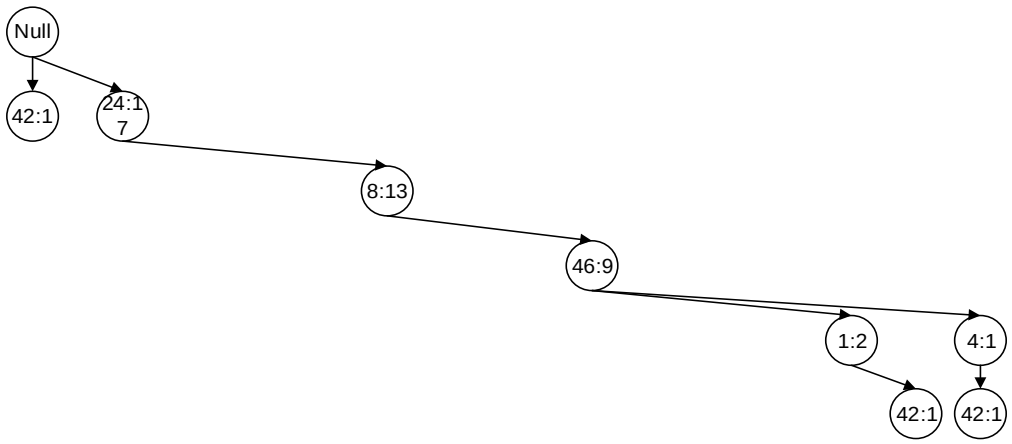
- *Conditional FP-Tree* untuk 45 : {24,8,46,36,1,4,9,34,45:1} {24,8,46,36,1,4,39,7,25,31,21,41,29,45}{24,8,46,36,1,39,7,31,2,9, 38,45:1}



Gambar 3. 74 *Conditional FP-Tree Untuk* 45

Karena niilai *support count* 45 adalah 3, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

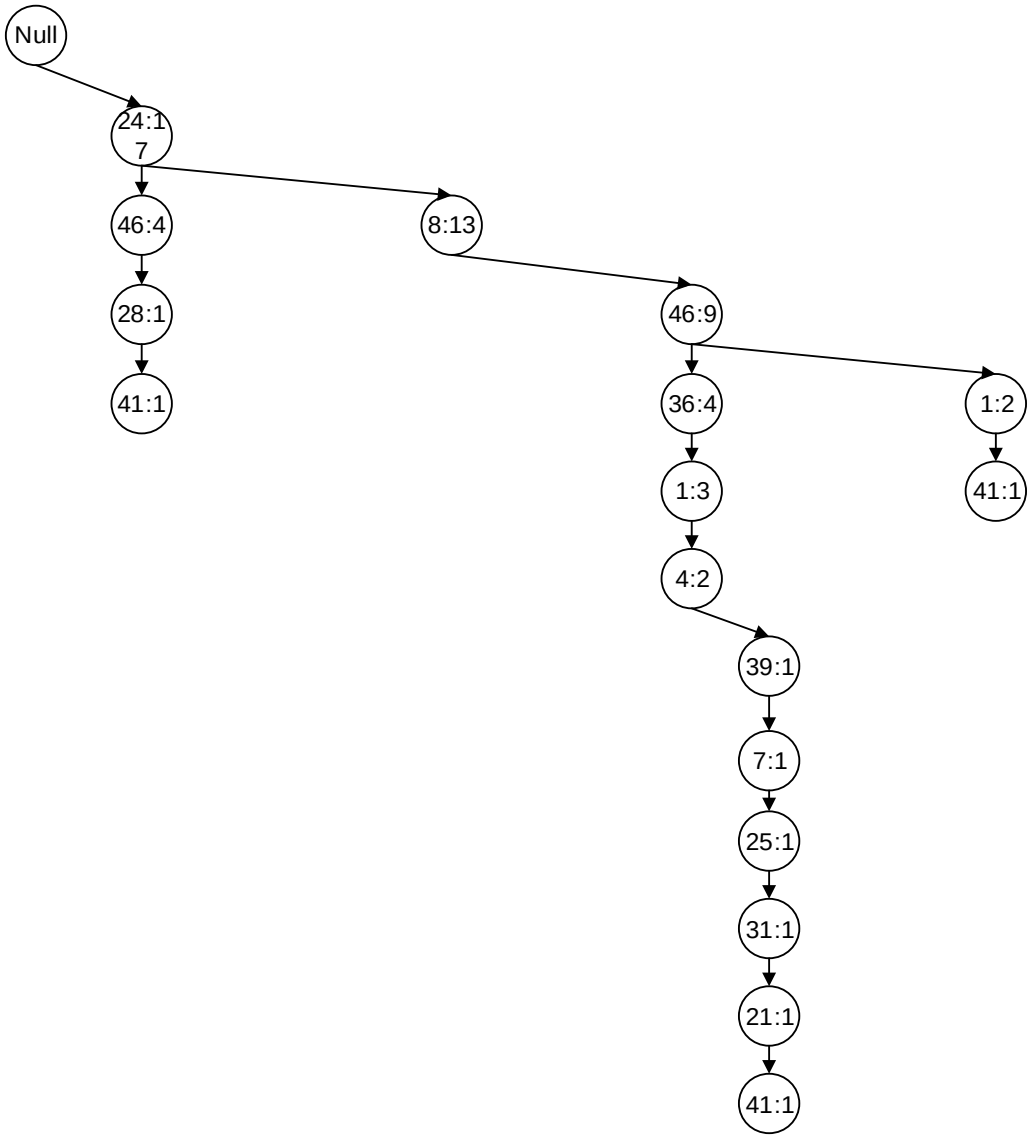
- *Conditional FP-Tree* untuk 42 : {Null, 42:1}{24,8,46,1,42:1} {24,8,46,4,42:1}



Gambar 3. 75 *Conditional FP-Tree Untuk* 42

Karena niilai *support count* 42 adalah 3, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

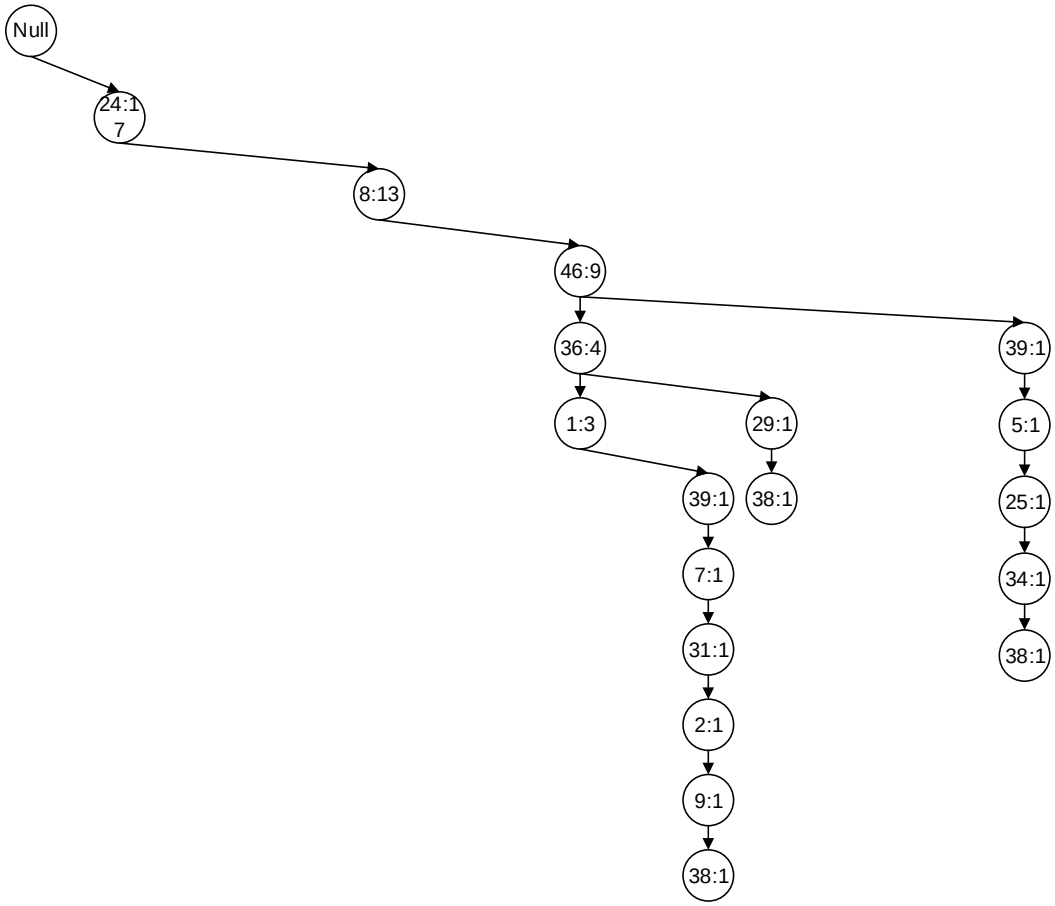
- *Conditional FP-Tree* untuk 41: {24,46,28,41:1} {24,8,46,36,1,4,39,7,25,31,21,41:1} {4,8,46,1,41:1



Gambar 3. 76 *Conditional FP-Tree Untuk* 41

Karena niilai *support count* 41 adalah 3, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

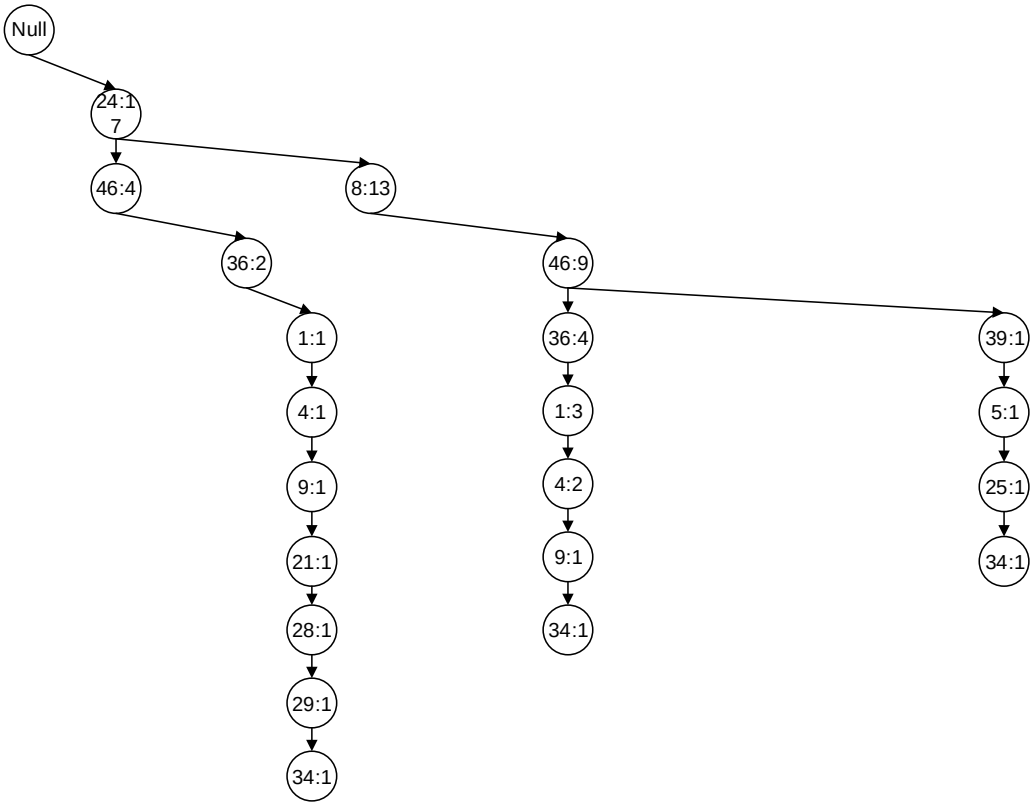
- *Conditional FP-Tree* untuk 38 : {24,8,46,36,1,39,7,31,2,9,38:1} {24,8,46,29,38:1} {24,8,46,39,5,25,34,38:1}



Gambar 3. 77 *Conditional FP-Tree Untuk* 38

Karena niilai *support count* 38 adalah 3, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

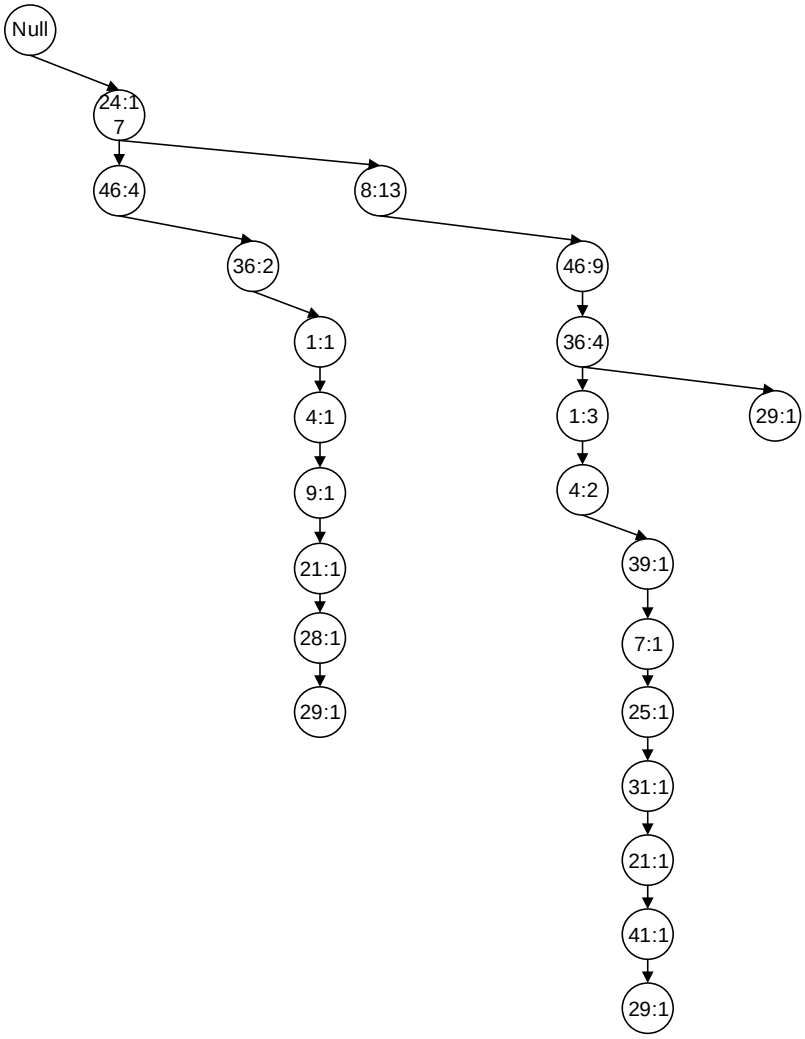
- *Conditional FP-Tree* untuk 34 : {24,46,36,1,4,9,21,28,29,34:1} {24,8,46,36,1,4,9,34:1} {24,8,46,39,5,25,34:1}



Gambar 3. 78 *Conditional FP-Tree Untuk* 34

Karena niilai *support count* 34 adalah 3, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

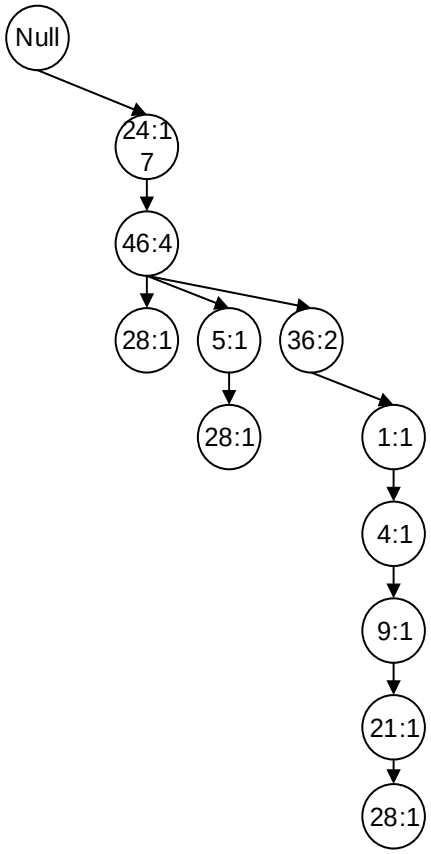
- *Conditional FP-Tree* untuk 29 : {24,46,36,1,4,9,21,28,29:1} {24,8,46,36,1,4,39,7,25,31,21,41,29:1} {24,8,46,36,29:1}



Gambar 3. 79 *Conditional FP-Tree Untuk* 29

Karena niilai *support count* 29 adalah 3, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

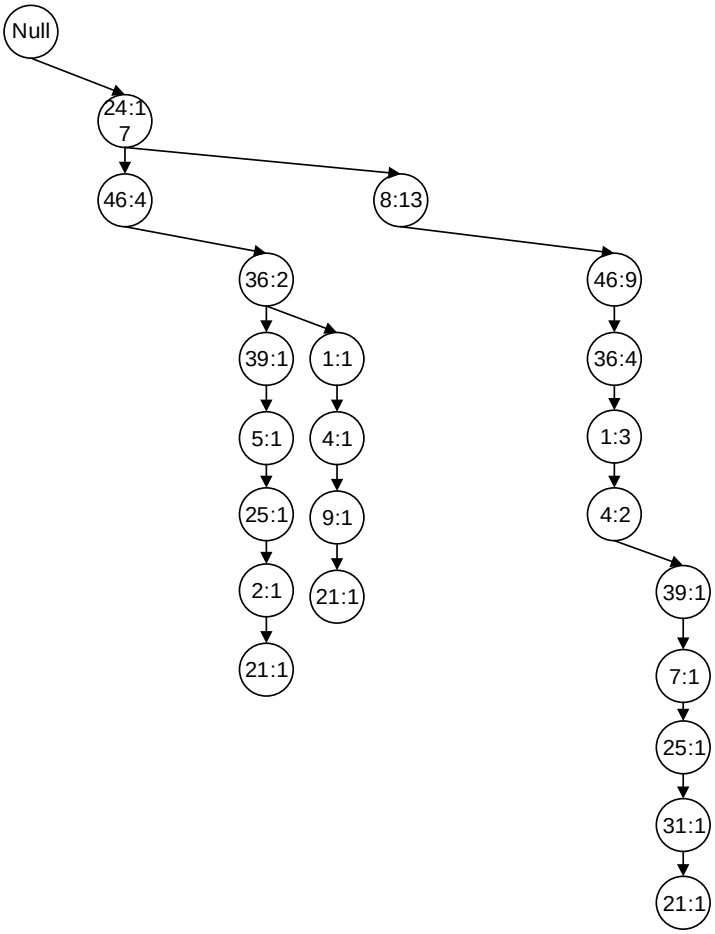
- *Conditional FP-Tree* untuk 28 : {24,46,28:1} {24,46,5,28:1} {24,46,36,1,4,9,21,28:1}



Gambar 3. 80 *Conditional FP-Tree Untuk* 28

Karena niilai *support count* 28 adalah 3, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

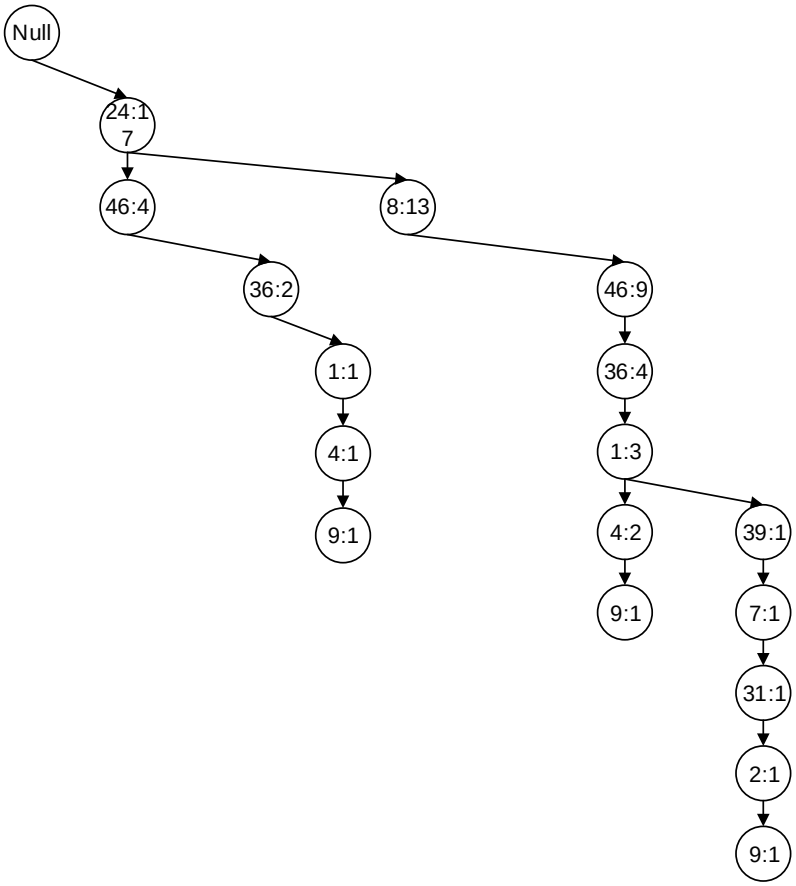
- *Conditional FP-Tree* untuk 21 : {24,46,36,39,5,25,2,21:1} {24,46,36,1,4,9,21:1} {8,46,36,1,4,39,7,25,31,21:1}



Gambar 3. 81 *Conditional FP-Tree Untuk* 21

Karena niilai *support count* 21 adalah 3, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

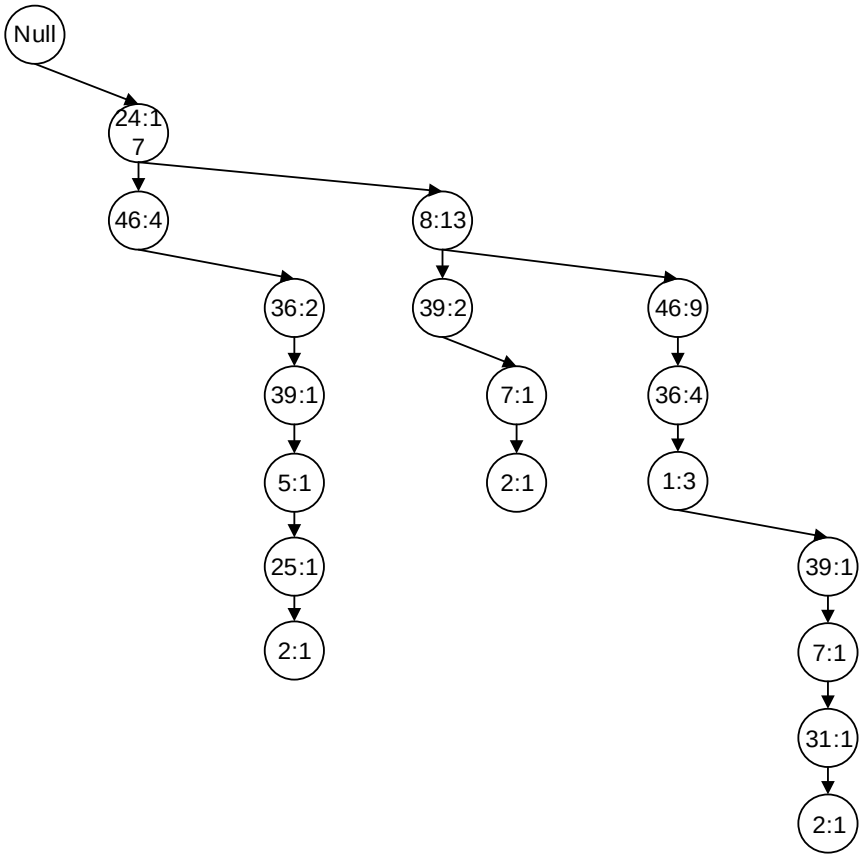
- *Conditional FP-Tree* untuk 9 : {24,46,36,1,4,9:1}
  {24,8,46,36,1,4,9:1} {24,8,46,36,1,39,7,31,2,9:1}



Gambar 3. 82 *Conditional FP-Tree Untuk* 9

Karena niilai *support count* 9 adalah 3, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

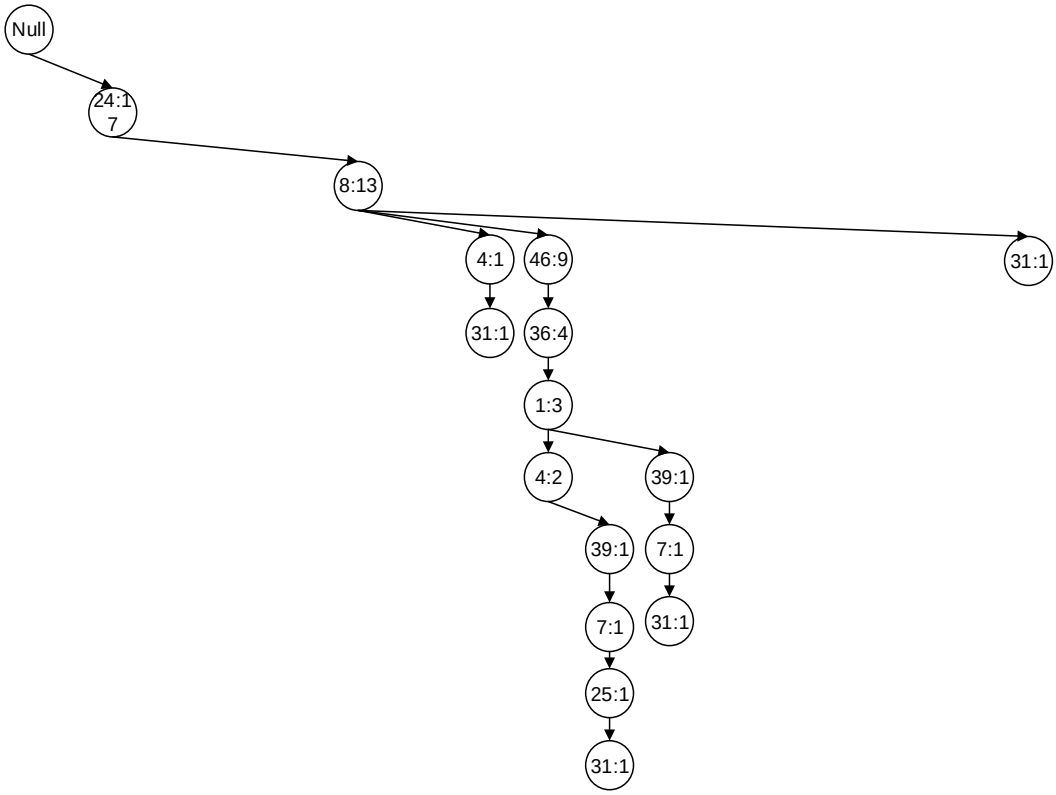
- *Conditional FP-Tree* untuk 2 : {24,46,36,39,5,25,2:1} {24,8,39,7,2:1} {24,8,46,36,1,39,7,31,2:1}



Gambar 3. 83 *Conditional FP-Tree Untuk* 2

Karena niilai *support count* 2 adalah 3, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan.

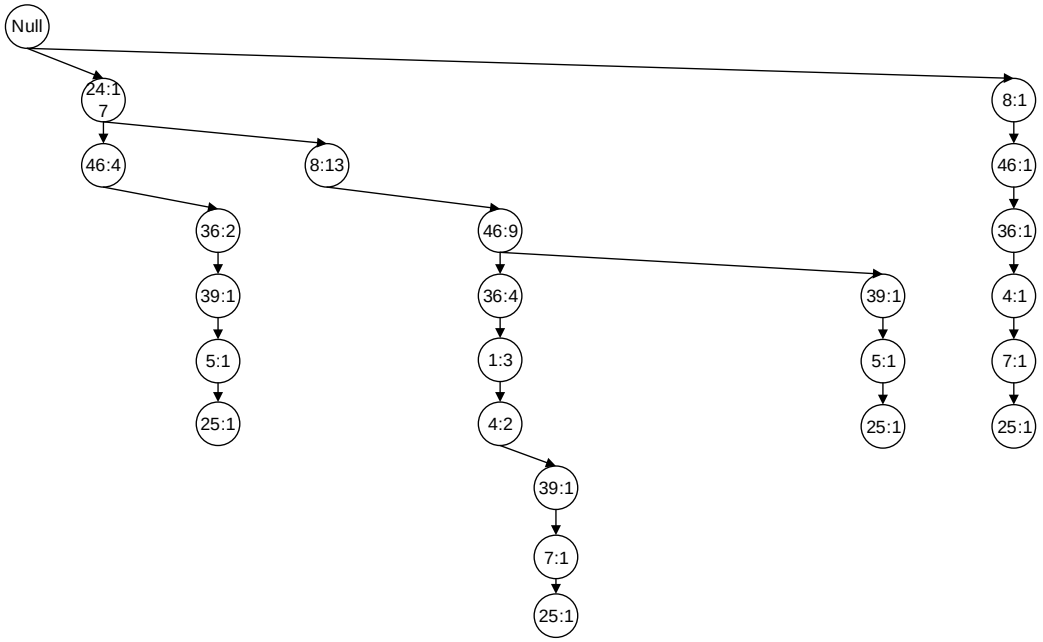
- *Conditional FP-Tree* untuk 31 : {24,8,4,31:1}
  {24,8,46,36,1,4,39,7,25,31:1} {24,8,46,36,1,39,7,31:1}
  {24,8,31:1}



Gambar 3. 84 *Conditional FP-Tree Untuk* 31

Karena niilai *support count* 31 adalah 4, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan

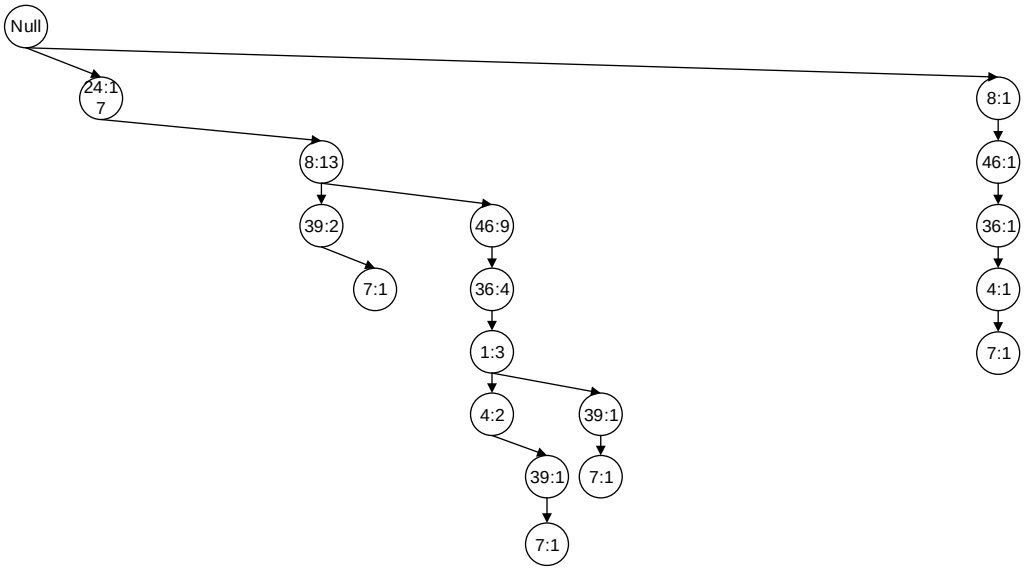
- *Conditional FP-Tree* untuk 25 : {24,46,36,39,5,25:1} {24,8,46,36,1,4,39,7,25:1} {24,8,46,39,5,25:1} {8,46,36,4,7,25:1}



Gambar 3. 85 *Conditional FP-Tree Untuk* 25

Karena niilai *support count* 25 adalah 4, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan

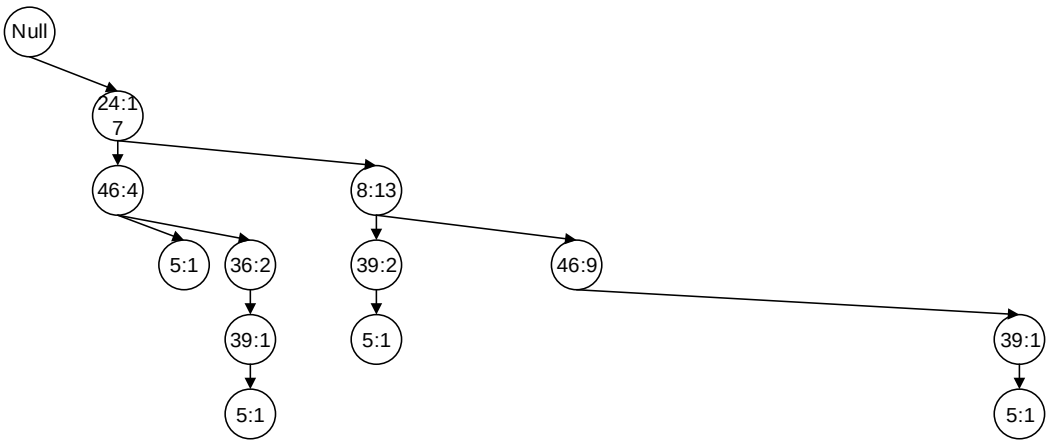
- *Conditional FP-Tree* untuk 7 : {24,8,39,7:1}
  {24,8,46,36,1,4,39,7:1} {24,8,46,36,1,39,7:1} {8,46,36,4,7:1}



Gambar 3. 86 *Conditional FP-Tree Untuk* 7

Karena niilai *support count* 7 adalah 4, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan

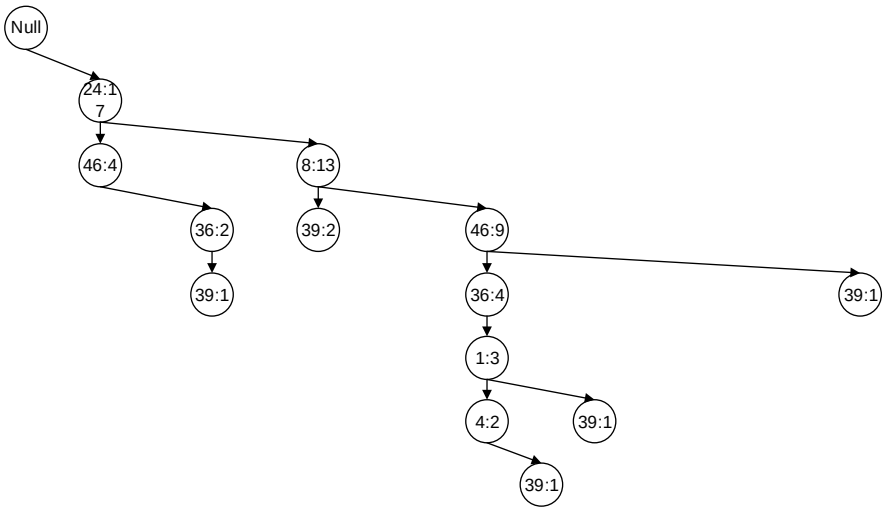
- *Conditional FP-Tree* untuk 5 : {24,46,5:1} {24,46,36,39,5:1} {24,8,39,5:1} {24,8,46,39,5:1}



Gambar 3. 87 *Conditional FP-Tree Untuk* 5

Karena niilai *support count* 5 adalah 4, maka tidak memenuhi nilai *minimum support*, maka tidak ada *frequent itemset* yang dibangkitkan

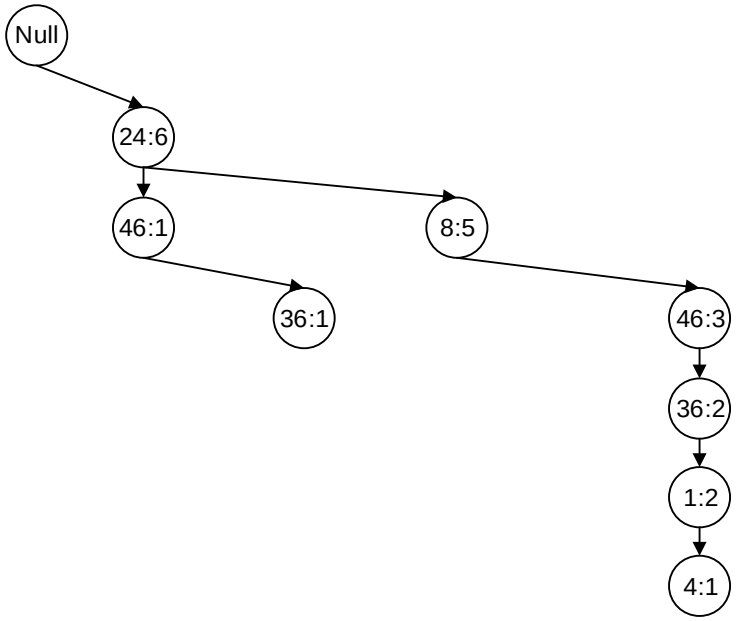
- *Conditional FP-Tree* untuk 39 : {24,46,36,39:1} {24,8,39:2} {24,8,46,36,1,4,39:1} {24,8,46,36,1,39:1} {24,8,46,39:1}



Gambar 3. 88 *Conditional FP-Tree Untuk* 39

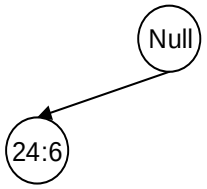
Karena niilai *support count* 39 adalah 6, maka memenuhi *minimum support* yaitu 5. Berikut adalah tahap-tahapannya :

- Jumlah *support count* yang memenuhi *minimum support* akan dibangkitkan dengan *conditional fp-tree*.
- Dalam conditional fp-tree untuk 39, ada 5 lintasan yaitu :
  - Lintasan Pertama : {24,46,36,39:1}
  - Lintasan Kedua : {24,8,39:2}
  - Lintasan Ketiga : {24,8,46,36,1,4,39:1}
  - Lintasan Keempat : {24,8,46,36,1,39:1}
  - Lintasan Kelima : {24,8,46,39:1}
- Pada lintasan yang berakhiran {39:1} dengan *support count* adalah 1 atau lebih dari 1 akan dihilangkan dan *support count* yang melewati lintasan yang berakhiran {39}, akan sama *support count*-nya dengan *support count* 39 sesuai dengan *support* yang dimilikinya.
- Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka *support count*-nya akan selalu bertambah 1.

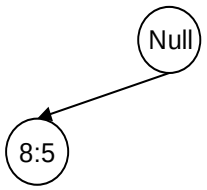


Gambar 3. 89 *Conditional FP-Tree* 39 Setelah Dibangkitkan

- Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequent itemset* yang keluar yang memenuhi *minimum support*. 24:6, dan 8:5 yang dapat meemnuhi *minimum support.* Dan *frequent itemset* yang didapatkan adalah {39,24:6}, {39,8:5}

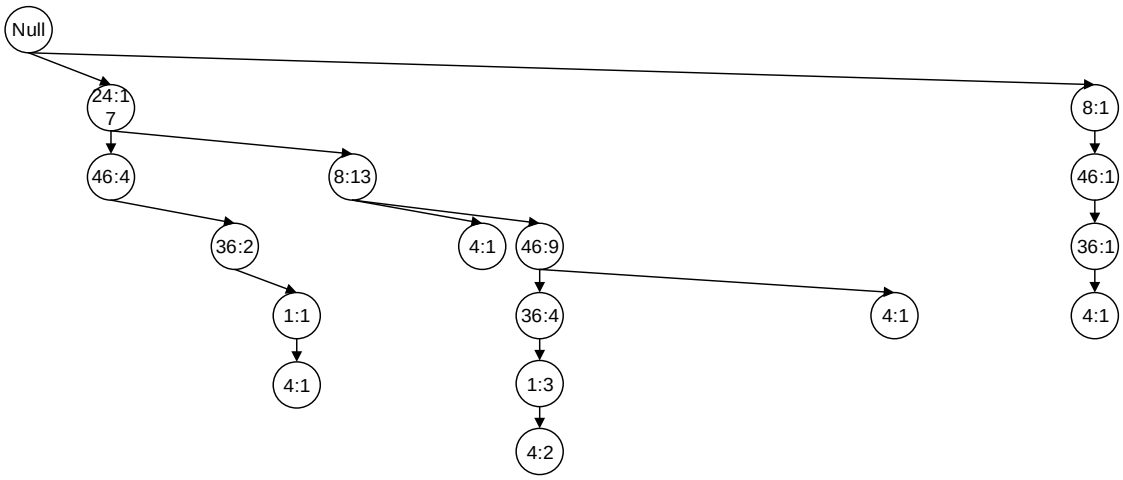


Gambar 3. 90 *Conditional FP-Tree* 39,24:6 Hasil *Frequent Itemset*



Gambar 3. 91 *Conditional FP-Tree* 39,8:5 Hasil *Frequent Itemset*

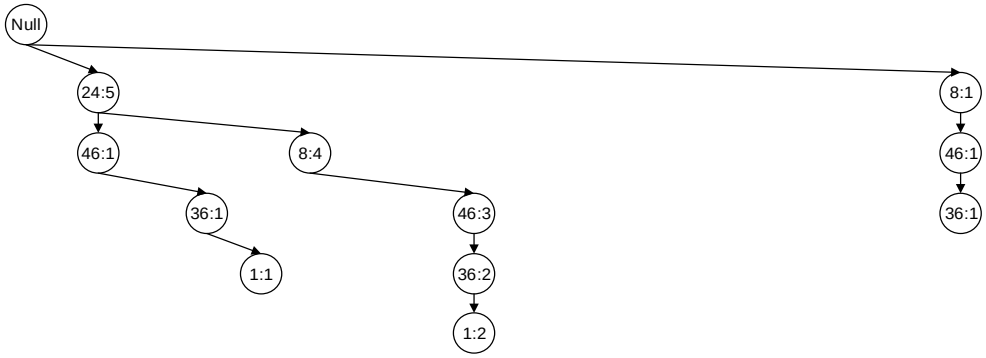
- *Conditional FP-Tree* untuk 4 : {24,46,36,1,4:1} {24,8,4:1} {24,8,46,36,1,4:2} {24,8,46,4:1} {8,46,36,4:1}



Gambar 3. 92 *Conditional FP-Tree Untuk* 4

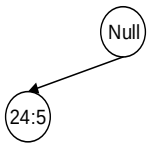
Karena niilai *support count* 4 adalah 6, maka memenuhi *minimum support* yaitu 5. Berikut adalah tahap-tahapannya :

- Jumlah *support count* yang memenuhi *minimum support* akan dibangkitkan dengan *conditional fp-tree*.
- Dalam conditional fp-tree untuk 4, ada 5 lintasan yaitu :
  - Lintasan Pertama : {24,46,36,1,4:1}
  - Lintasan Kedua : {24,8,4:1}
  - Lintasan Ketiga : {24,8,46,36,1,4:2}
  - Lintasan Keempat : {24,8,46,4:1}
  - Lintasan Kelima : {8,46,36,4:1}
- Pada lintasan yang berakhiran {4:1} dengan *support count* adalah 1 atau lebih dari 1 akan dihilangkan dan *support count* yang melewati lintasan yang berakhiran {4}, akan sama *support count*-nya dengan *support count* 4 sesuai dengan *support* yang dimilikinya.
- Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka *support count*-nya akan selalu bertambah 1.

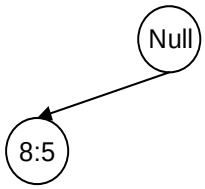


Gambar 3. 93 *Conditional FP-Tree* 4 Setelah Dibangkitkan

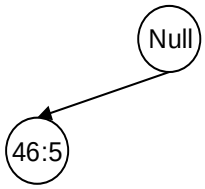
- Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequent itemset* yang keluar yang memenuhi *minimum support*. 24:5, 8:5, dan 46:5 yang dapat meemnuhi *minimum support.* Dan *frequent itemset* yang didapatkan adalah {4,24:5}, {4,8:5}, {4,46:5}



Gambar 3. 94 *Conditional FP-Tree* 4,24:5 Hasil *Frequent Itemset*

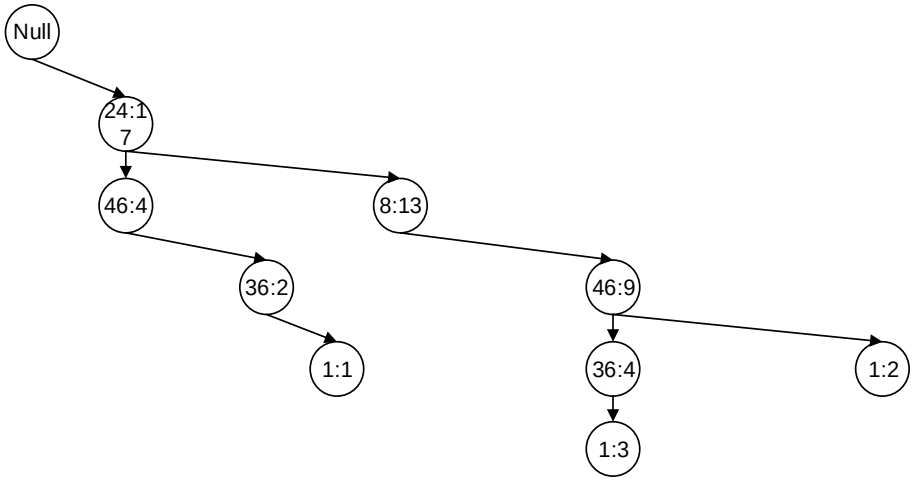


Gambar 3. 95 *Conditional FP-Tree* 4,8:5 Hasil *Frequent Itemset*



Gambar 3. 96 *Conditional FP-Tree* 4,46:5 Hasil *Frequent Itemset*

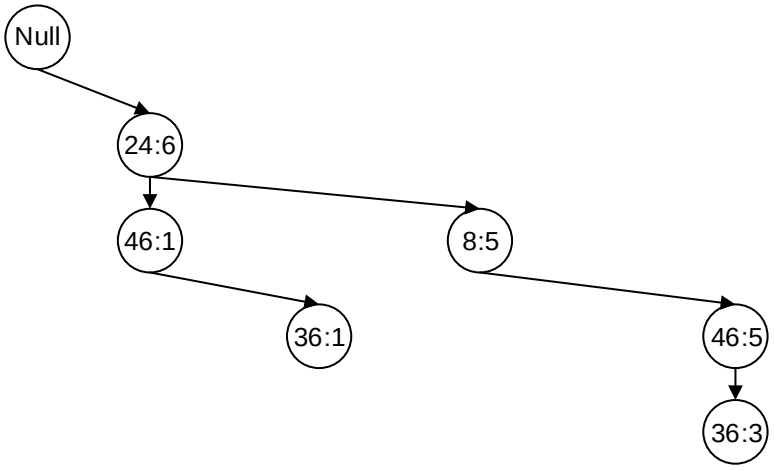
- *Conditional FP-Tree* untuk 1 : {24,46,36,1:1} {24,8,46,36,1:3} {24,8,46,1:2}



Gambar 3. 97 *Conditional FP-Tree Untuk* 1

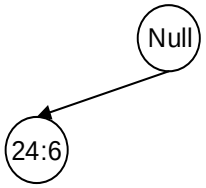
Karena niilai *support count* 1 adalah 6, maka memenuhi *minimum support* yaitu 5. Berikut adalah tahap-tahapannya :

- Jumlah *support count* yang memenuhi *minimum support* akan dibangkitkan dengan *conditional fp-tree*.
- Dalam conditional fp-tree untuk 1, ada 3 lintasan yaitu :
  - Lintasan Pertama : {24,46,36,1:1}
  - Lintasan Kedua : {24,8,46,36,1:3}
  - Lintasan Ketiga : {24,8,46,1:2}
- Pada lintasan yang berakhiran {1:1} dengan *support count* adalah 1 atau lebih dari 1 akan dihilangkan dan *support count* yang melewati lintasan yang berakhiran {1}, akan sama *support count*-nya dengan *support count* 1 sesuai dengan *support* yang dimilikinya.
- Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka *support count*-nya akan selalu bertambah 1.

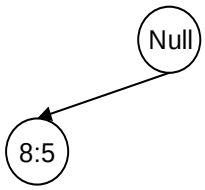


Gambar 3. 98 *Conditional FP-Tree* 1 Setelah Dibangkitkan

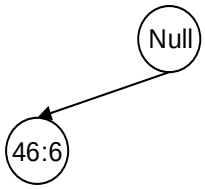
- Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequent itemset* yang keluar yang memenuhi *minimum support*. 24:6, 8:5, dan 46:6 yang dapat meemnuhi *minimum support.* Dan *frequent itemset* yang didapatkan adalah {1,24:6}, {1,8:5}, {1,46:6}



Gambar 3. 99 *Conditional FP-Tree* 1,24:6 Hasil *Frequent Itemset*

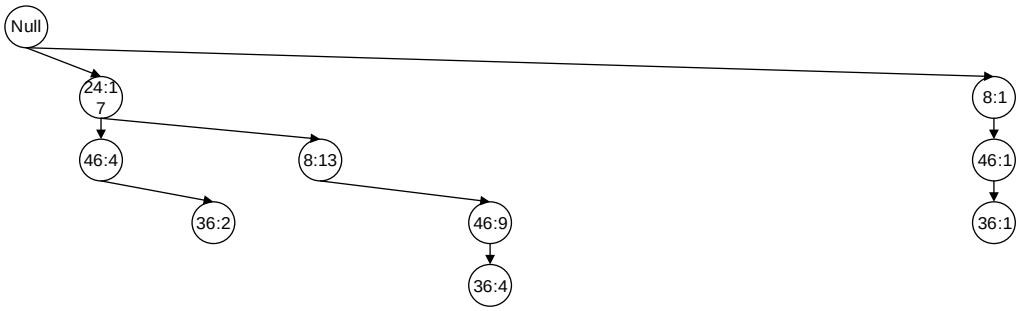


Gambar 3. 100 *Conditional FP-Tree* 1,8:5 Hasil *Frequent Itemset*



Gambar 3. 101 *Conditional FP-Tree* 1,46:6 Hasil *Frequent Itemset*

- *Conditional FP-Tree* untuk 36 : {24,46,36:2}{24,8,46,36:4} {8,46,36:1}

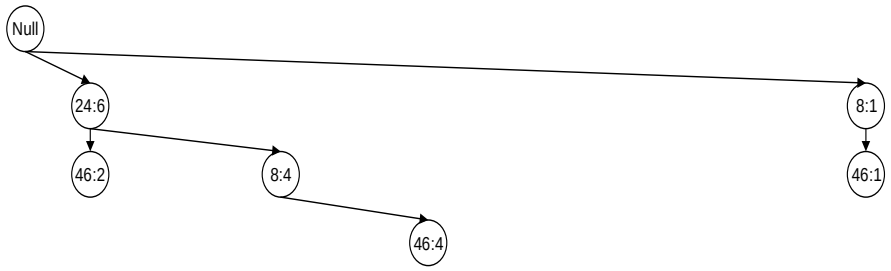


Gambar 3. 102 *Conditional FP-Tree Untuk* 36

Karena niilai *support count* 36 adalah 7, maka memenuhi *minimum support* yaitu 5. Berikut adalah tahap-tahapannya :

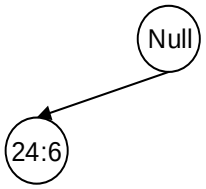
- Jumlah *support count* yang memenuhi *minimum support* akan dibangkitkan dengan *conditional fp-tree*.
- Dalam conditional fp-tree untuk 36, ada 3 lintasan yaitu :
    - Lintasan Pertama : {24,46,36:2}
    - Lintasan Kedua : {24,8,46,36:4}
    - Lintasan Ketiga : {8,46,36:1}

- Pada lintasan yang berakhiran {36:1} dengan *support count* adalah 1 atau lebih dari 1 akan dihilangkan dan *support count* yang melewati lintasan yang berakhiran {36}, akan sama *support count*-nya dengan *support count* 36 sesuai dengan *support* yang dimilikinya.
- Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka *support count*-nya akan selalu bertambah 1.

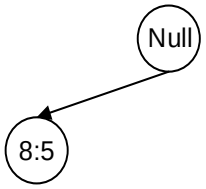


Gambar 3. 103 *Conditional FP-Tree* 36 Setelah Dibangkitkan

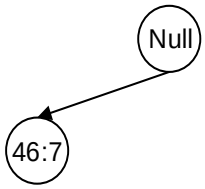
- Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequent itemset* yang keluar yang memenuhi *minimum support*. 24:6, 8:5, dan 46:7 yang dapat meemnuhi *minimum support.* Dan *frequent itemset* yang didapatkan adalah {36,24:6}, {36,8:5}, {36,46:7}



Gambar 3. 104 *Conditional FP-Tree* 36 ,24:6 Hasil *Frequent Itemset*

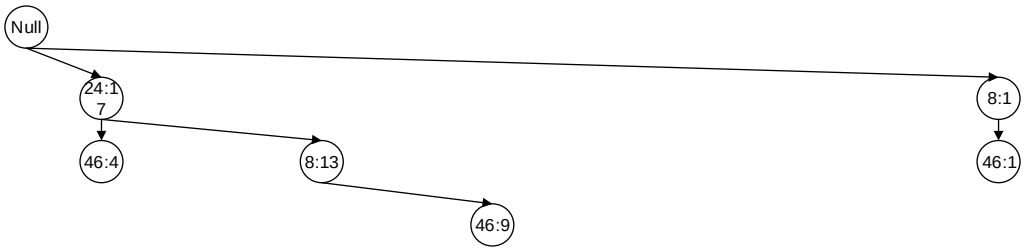


Gambar 3. 105 *Conditional FP-Tree* 36,8:5 Hasil *Frequent Itemset*



Gambar 3. 106 *Conditional FP-Tree* 36,46:7 Hasil *Frequent Itemset*

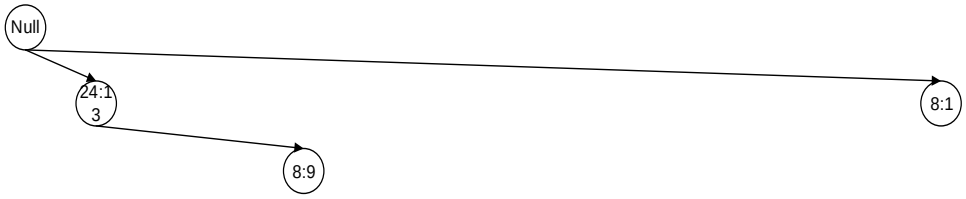
- *Conditional FP-Tree* untuk 46 : {24,46:4} {24,8,46:9} {8,46:1}



Gambar 3. 107 *Conditional FP-Tree Untuk* 46

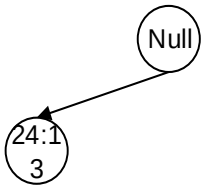
Karena niilai *support count* 46 adalah 14, maka memenuhi *minimum support* yaitu 5. Berikut adalah tahap-tahapannya :

- Jumlah *support count* yang memenuhi *minimum support* akan dibangkitkan dengan *conditional fp-tree*.
- Dalam conditional fp-tree untuk 46, ada 3 lintasan yaitu :
  - Lintasan Pertama : {24,46:4}
  - Lintasan Kedua : {24,8,46:9}
  - Lintasan Ketiga : {8,46:1}
- Pada lintasan yang berakhiran {46:1} dengan *support count* adalah 1 atau lebih dari 1 akan dihilangkan dan *support count* yang melewati lintasan yang berakhiran {46}, akan sama *support count*-nya dengan *support count* 46 sesuai dengan *support* yang dimilikinya.
- Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka *support count*-nya akan selalu bertambah 1.

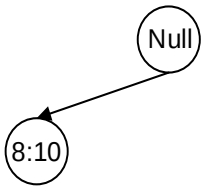


Gambar 3. 108 *Conditional FP-Tree* 46 Setelah Dibangkitkan

- Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequent itemset* yang keluar yang memenuhi *minimum support*. 24:13, 8:10 yang dapat meemnuhi *minimum support.* Dan *frequent itemset* yang didapatkan adalah {46,24:13}, {46,8:10}

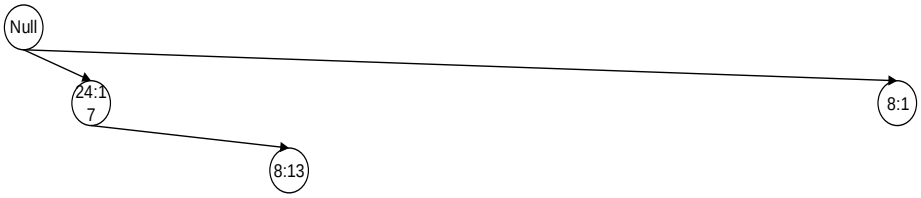


Gambar 3. 109 *Conditional FP-Tree* 46,24:13 Hasil *Frequent Itemset*



Gambar 3. 110 *Conditional FP-Tree* 46,8:10 Hasil *Frequent Itemset*

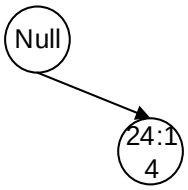
- *Conditional FP-Tree* untuk 8 : {24,8:13} {Null, 8:1}



Gambar 3. 111 *Conditional FP-Tree Untuk* 8

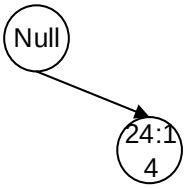
Karena niilai *support count* 8 adalah 14, maka memenuhi *minimum support* yaitu 5. Berikut adalah tahap-tahapannya :

- Jumlah *support count* yang memenuhi *minimum support* akan dibangkitkan dengan *conditional fp-tree*.
- Dalam conditional fp-tree untuk 8, ada 2 lintasan yaitu :
    - Lintasan Pertama : {24,8:13}
    - Lintasan Kedua : {Null, 8:1}
- Pada lintasan yang berakhiran {8:1} dengan *support count* adalah 1 atau lebih dari 1 akan dihilangkan dan *support count* yang melewati lintasan yang berakhiran {8}, akan sama *support count*-nya dengan *support count* 8 sesuai dengan *support* yang dimilikinya.
- Apabila dalam suatu lintasan melintasi satu atau dua lintasan, maka *support count*-nya akan selalu bertambah 1.



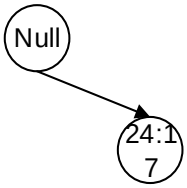
Gambar 3. 112 *Conditional FP-Tree* 8 Setelah Dibangkitkan

- Setelah proses pembangkitan, maka proses terakhir adalah mencari *frequent itemset* yang keluar yang memenuhi *minimum support*. 24:14 yang dapat meemnuhi *minimum support.* Dan *frequent itemset* yang didapatkan adalah {8,24:14}.



Gambar 3. 113 *Conditional FP-Tree* 8,24:14 Hasil *Frequent Itemset*

- *Conditional FP-Tree* untuk 24 : {Null, 24:17}



Gambar 3. 114 *Conditional FP-Tree Untuk* 24

c. Tahap Pencarian *Frequent Itemset*

Setelah mendapatkan hasil dari *conditional fp-tree* dari tabel *conditional pattern base* diatas, ditemukan beberapa *frequent itemset* yang keluar. Hasilnya dapat dilihat pada tabel 3.18 dibawah ini :

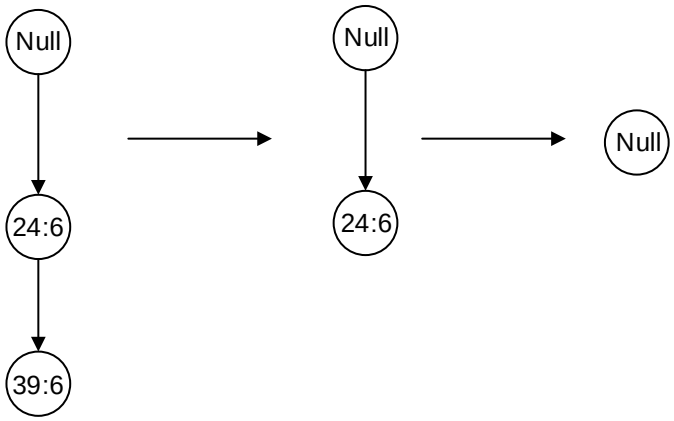
Tabel 3. 18 Hasil *Conditional Fp-Tree*

Item	***Conditional FP-Tree***
44	Empty
40	Empty
33	Empty
32	Empty
30	Empty
27	Empty
23	{Null}
18	Empty
12	Empty
10	Empty
6	{Null}
45	Empty
42	{Null}
41	Empty
38	Empty
34	Empty

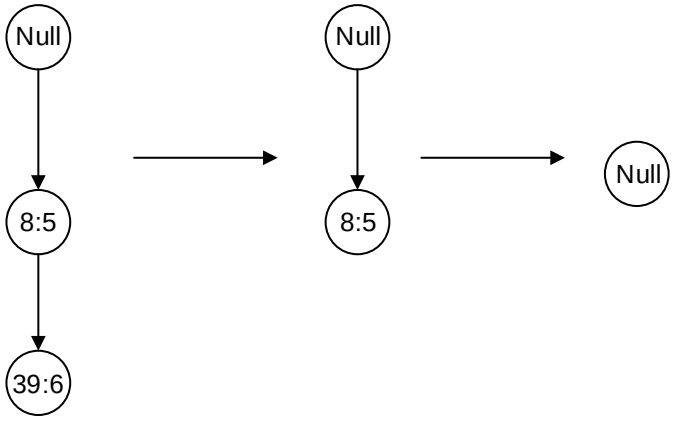
Item	*Conditional FP-Tree*
29	Empty
28	Empty
21	Empty
9	Empty
2	Empty
31	Empty
25	Empty
7	Empty
5	Empty
39	{39, 24:6}, {39,8:5}
4	{4,24:5}, {4,8:5}, {4,46:5}
1	{1,24:6}, {1,8:5}, {1,46:6}
36	{36,24:6},{36,8:5},{36,46:7}
46	{46,24:13}, {46,8:10}
8	{8,24:14}
24	{Null}

Setelah membuat *condiitonal fp-tree* dari hasil pencarian *frequent itemset*, maka dilakukan tahap berikutnya :

a. Akumulasikan *count* untuk setiap item frequent itemset yang ada pada tabel 3.18 diatas.
b. Buat *conditional fp-tree* untuk *frequent items* pada tabel 3.18 diatas.
    - Untuk *item* 39, *item frequent* yang keluar adalah {39, 24:6}, {39,8:5}

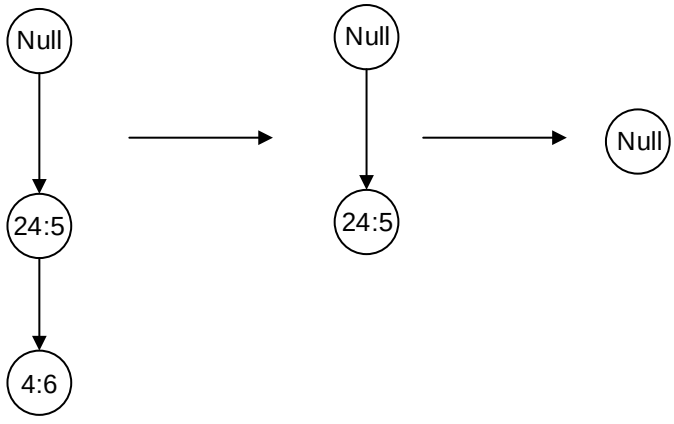


Gambar 3. 115 *Conditional FP-Tree* Untuk *Item Frequenset* 39, 24:6

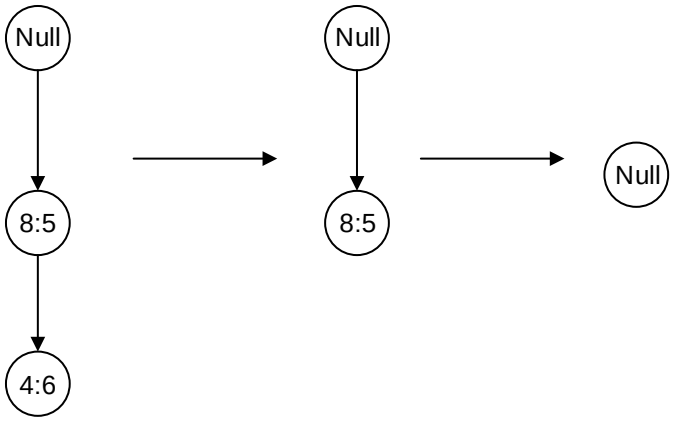


Gambar 3. 116 *Conditional FP-Tree* Untuk *Item Frequenset* 39, 8:5

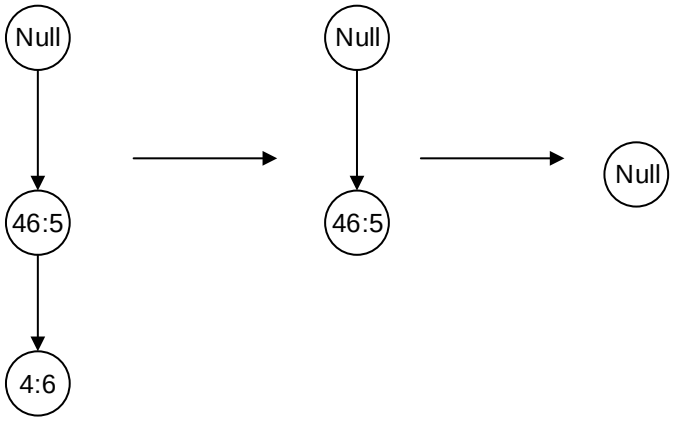
- Untuk *item* 4, *item frequent* yang keluar adalah {4,24:5}, {4,8:5}, {4,46:5}



Gambar 3. 117 *Conditional FP-Tree* Untuk *Item Frequenset* 4, 24:5

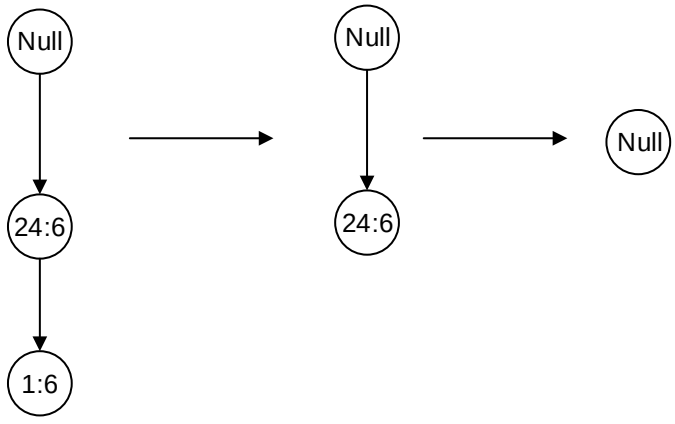


Gambar 3. 118 *Conditional FP-Tree* Untuk *Item Frequenset* 4, 8:5

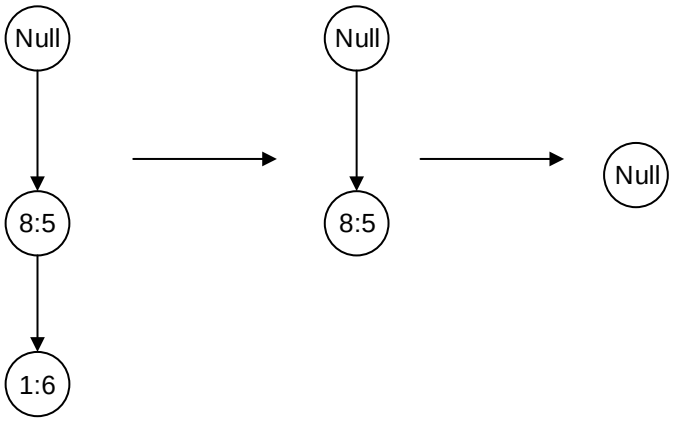


Gambar 3. 119 *Conditional FP-Tree* Untuk *Item Frequenset* 4, 46:5

- Untuk *item* 1, *item frequent* yang keluar adalah {1,24:6}, {1,8:5}, {1,46:6}



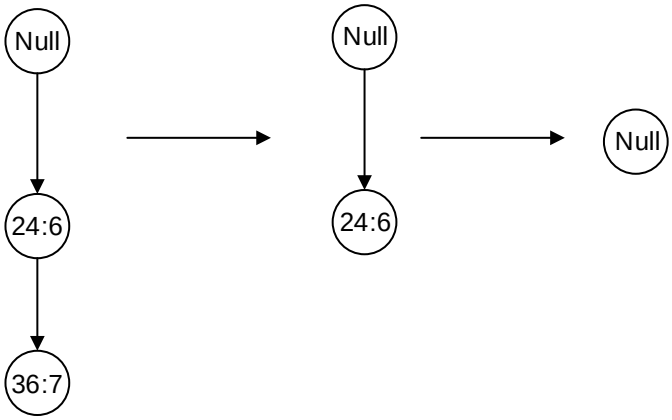
Gambar 3. 120 *Conditional FP-Tree* Untuk *Item Frequenset* 1, 24:6



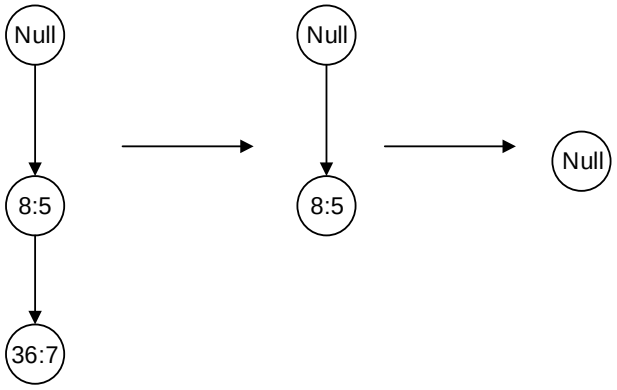
Gambar 3. 121 *Conditional FP-Tree* Untuk *Item Frequenset* 1, 8:5

Gambar 3. 122 Conditional FP-Tree Untuk Item Frequenset 1,46:6

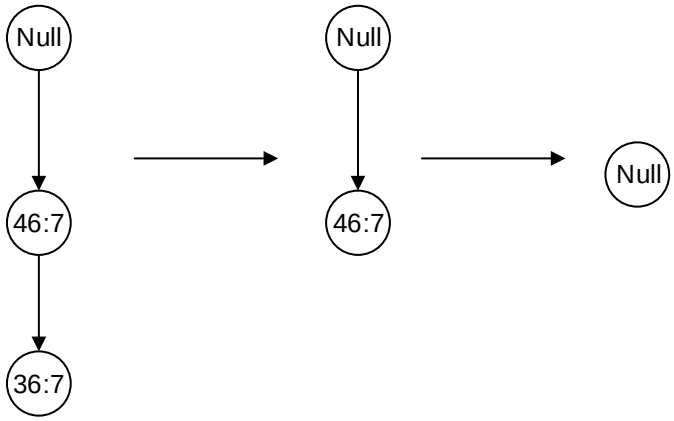
- Untuk *item* 36 *item frequent* yang keluar adalah {36,24:6},{36,8:5},{36,46:7}



Gambar 3. 123 *Conditional FP-Tree* Untuk *Item Frequenset* 36, 24:6

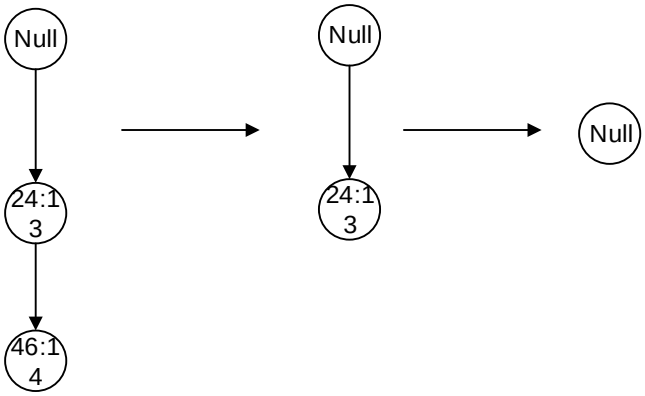


Gambar 3. 124 *Conditional FP-Tree* Untuk *Item Frequenset* 36, 8:5

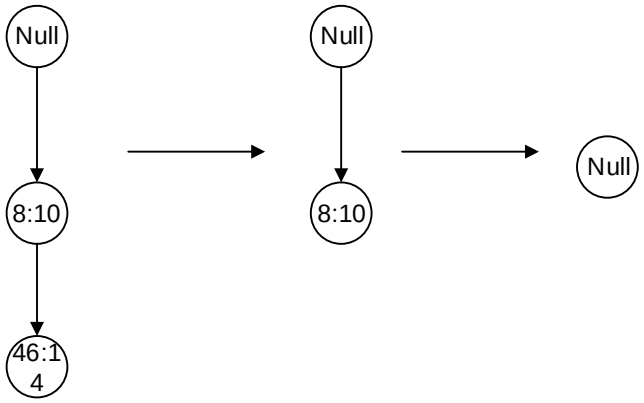


Gambar 3. 125 *Conditional FP-Tree* Untuk *Item Frequenset* 36, 46:7

- Untuk *item* 46, *item frequent* yang keluar adalah {46,24:13},{46,8:10}

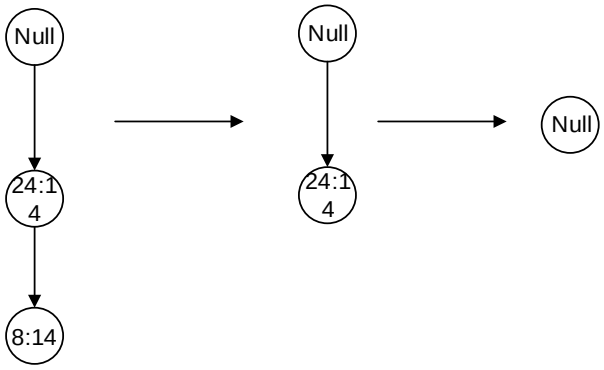


Gambar 3. 126 *Conditional FP-Tree* Untuk *Item Frequenset* 46, 24:13



Gambar 3. 127 *Conditional FP-Tree* Untuk *Item Frequenset* 46, 8:10

- Untuk *item* 8, *item frequent* yang keluar adalah {8,24:14}



Gambar 3. 128 *Conditional FP-Tree* Untuk *Item Frequenset* 8, 24:14

7. Setelah mendapatkan hasil *frequent itemset* dari setiap *conditional fp-tree*, selanjutnya adalah menentukan *frequent itemset* setiap *suffix*.

Tabel 3. 19 Hasil *Frequent Itemset*

Suffix	***Frequent Itemset***
39	39 - 24 39 - 8
4	4- 24

Suffix	*Frequent Itemset*
	4 - 8 4 - 46
1	1 - 24 1 – 8 1 - 46
36	36 - 24 36 - 8 36 - 46
46	46 - 24 46 - 8
8	8 - 24

Dari hasil *frequent Itemset* diatas akan di-*generate* untuk mendapatkan *rule* dengan *minimum support* 5 dan *minimum confidence* 60% dengan masing-masing *itemset* dikombinasikan dengan *itemset* lain. Kenapa nilai *confidence* 60% karena hasil *frequent itemset* yang keluar adalah hasil yang sering paling muncul dalam data transaksi, jadi penentuan paket makanan heamt bisa diatur sesuai dengan *frequent* yang muncul sesuai dengan nilai *confidence*. Dengan menggunakan persamaan (2.2) yang ada di bab 2 untuk memperoleh nilai *support* dan persamaan(2.4) untuk memperoleh nilai *confidence.* Maka *rule* yang sudah didapat akan diberikan nilai *support* dan *confidence* menggunakan persamaan yang sudah disebutkan di atas. Berikut ini adalah hasil dari pemberian nilai untuk *support* dan *confidence* dari tiap *rule.*

Tabel 3. 20 Hasil Perhitungan *Support* dan *Confidence*

Rule	*Minimum Support*	*Minimum Confidence*
39→24	6/22*100=27,27	6/6*100=100%
24→39	6/22*100=27,27	6/17*100=35,29%

Rule	*Minimum Support*	*Minimum Confidence*
39→8	5/22*100=22,72	5/6*100=83,33%
8→39	5/22*100=22,72	5/14*100=35,71%
4→24	5/22*100=22,72	5/6*100=83,33%
24→4	5/22*100=22,72	5/17*100=29,41%
4→8	5/22*100=22,72	5/6*100=83,33%
8→4	5/22*100=22,72	5/14*100=35,71%
4→46	5/22*100=22,72	5/6*100=83,33%
46→4	5/22*100=22,72	5/14*100=35,71%
1→24	6/22*100=27,27	6/6*100=100%
24→1	6/22*100=27,27	6/17*100=35,29
1→8	5/22*100=22,72	5/6*100=83,33%
8→1	5/22*100=22,72	5/14*100=35,71%
1→46	6/22*100=27,27	6/6*100=100%
46→1	6/22*100=27,27	6/14*100=42,85%
36→24	6/22*100=27,27	6/7*100=85,71%
24→36	6/22*100=27,27	6/17*100=35,29%
36→8	5/22*100=22,72	5/7*100=71,42
8→36	5/22*100=22,72	5/14*100=35,71%
36→46	7/22*100=31,81	7/7*100=100%
46→36	7/22*100=31,81	7/14*100=50%
46→24	13/22*100=59,09	13/14*100=92,85%
24→46	13/22*100=59,09	13/17*100=76,47%
46→8	10/22*100=45,45	10/14*100=71,42%
8→46	10/22*100=45,45	10/14*100=71,42%
8→24	13/22*100=59,09	13/14*100=92,85%
24→8	13/22*100=59,09	13/17*100=76,47%

Setelah melakukan perhitungan *minimum support* dan *minimum confidence* pada tabel 3.20 diatas, proses selanjutnya yang akan dilakukan adalah mencari

generate rule yang memenuhi nilai *minimum support* dan *confidence* yang telah ditentukan sebelumnya. Jika terdapat *itemset* yang tidak memenuhi *support* dan *confidence* akan dihapus. Berikut hasil *generate rule* pada tabel 3.21 dibawah ini :

Tabel 3. 21 Hasil *Generate Rule*

Rule	***Minimum Support***	***Minimum Confidence***
39→24	6/22*100=27,27	6/6*100=100%
39→8	5/22*100=22,72	5/6*100=83,33%
4→24	5/22*100=22,72	5/6*100=83,33%
4→8	5/22*100=22,72	5/6*100=83,33%
4→46	5/22*100=22,72	5/6*100=83,33%
1→24	6/22*100=27,27	6/6*100=100%
1→8	5/22*100=22,72	5/6*100=83,33%
1→46	6/22*100=27,27	6/6*100=100%
36→24	6/22*100=27,27	6/7*100=85,71%
36→8	5/22*100=22,72	5/7*100=71,42%
36→46	7/22*100=31,81	7/7*100=100%
46→24	13/22*100=59,09	13/14*100=92,85%
24→46	13/22*100=59,09	13/17*100=76,47%
46→8	10/22*100=45,45	10/14*100=71,42%
8→46	10/22*100=45,45	10/14*100=71,42%
8→24	13/22*100=59,09	13/14*100=92,85%
24→8	13/22*100=59,09	13/17*100=76,47%

Berdasarkan hasil *rule* yang didapatkan, maka pihak pengelola rumah makan Roso Echo dapat menentukan penentuan paket makanan hemat yang sesuai dengan *rule* yang didapatkan. Penentuan paket berdasarkan *confidence*. Berikut hasil penentuan paket makanan hemat dalam bentuk awal

Tabel 3. 22 Hasil Paket Makanan Hemat dalam Bentuk Awal

Rule	**Paket Makanan Hemat**	***Confidence***
39→24	Sate Jamur→Nasi	100%
39→8	Sate Jamur → Gorengan / Bacem	83,33%
4→24	Cumi→ Nasi	83,33%
4→8	Cumi→ Gorengan / Bacem	83,33%
4→46	Cumi→ Tumisan / Sayur / Urap / Mie	83,33%
1→24	Ayam Bakar / Cabe Ijo→ Nasi	100%
1→8	Ayam Bakar / Cabe Ijo→ Gorengan / Bacem	83,33%
1→46	Ayam Bakar / Cabe Ijo→ Tumisan / Sayur / Urap / Mie	100%
36→24	Sambal Goreng Krecek→Nasi	85,71%
36→8	Sambal Goreng Krecek→Gorengan / Bacem	71,42%
36→46	Sambal Goreng Krecek→ Tumisan / Sayur / Urap / Mie	100%
46→24	Tumisan / Sayur / Urap / Mie→Nasi	92,85%
24→46	Nasi→ Tumisan / Sayur / Urap / Mie	76,47%
46→8	Tumisan / Sayur / Urap / Mie→ Gorengan / Bacem	71,42%
8→46	Gorengan / Bacem→ Tumisan / Sayur / Urap / Mie	71,42%
8→24	Gorengan / Bacem→ Nasi	92,85%
24→8	Nasi→ Gorengan / Bacem	76,47%

3.1.5 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.*

3.1.5.1 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 ruma makan Roso Echo sendiri terdapat dua buah komputer dengan spesifikasi sebagai berikut :

Tabel 3. 23 Spesifikasi Perangkat Keras yang Sedang Berjalan

No	Perangkat Keras	Spesifikasi
1	RAM	1 GB
2	Harddisk	128 GB
3	Monitor	Touch-screen 15’’
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 :

Tabel 3. 24 Spesifikasi Perangkat Keras yang Akan Dibangun

No	Perangkat Keras	Spesifikasi
1	RAM	512 MB
2	Harddisk	1 GB
3	Monitor	Touch-screen
4	Processor	Berkecepatan minimal 1,9 Ghz

c. Evaluasi kebutuhan perangkat keras

Dengan memabndingkan spesifikasi perangkat keras untuk menjalankan perangkat lunak yang akan dibangun dengan spesifikasi perangkat keras yang ada pada rumah makan Roso Echo, maka dapat disimpulkan bahwa perangkat keras yang tersedia sesuai dengan kebutuhan dan telah memenuhi spesifikasi seperti pada tabel 3.24 diatas.

2 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak pendukung dalam pembangunan perangkat lunak ini adalah :

a. Analisis kebutuhan perangkat lunak pada sistem yang sedang berjalan di rumah makan Roso Echo adalah sebagai berikut :

Tabel 3. 25 Spesifikasi Perangkat Lunak yang Sedang Berjalan

No	Perangkat Keras	Spesifikasi
1	Sistem Operasi	Microsoft Windows XP Profesional

b. Analisis kebutuhan perangkat lunak pada sistem yang dibangun adalah sebagai berikut :

Tabel 3. 26 Spesifikasi Perangkat Lunak yang Akan Dibangun

No	Perangkat Keras	Spesifikasi
1	Sistem Operasi	Microsoft Windows 7
2	Xampp server	Database Server

c. Evaluasi kebutuhan perangkat lunak

Dengan membandingkan spesifikasi perangakt lunak yang dibutuhkan dengan perangkat lunak yang ada pada rumah makan Roso Echo, dapat disimpulkan bahwa spesifikasi perangkat lunak yang ada di rumah makan Roso Echo masih belum memenuhi kebetuhan yang diperlukan karena pada perangkat lunak tidak terdapat *database server*, sehingga perlu ditambahkan *xampp server*.

1. 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. 27 Analisa Kebutuhan di Rumah Makan Roso Echo

Pengguna	Tanggung Jawab	Tingkat Pendidikan	Tingkat Keterampilan
Manager	Merencanakan, melaksanakan, mengawasi dan mengevaluasi terhadap strategi bisnis yang digunakan	Minimal S-1	Mampu mengoperasikan komputer dan menangani rumah makan

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

Tabel 3. 28 Analisis Kebutuhan Pada Sistem

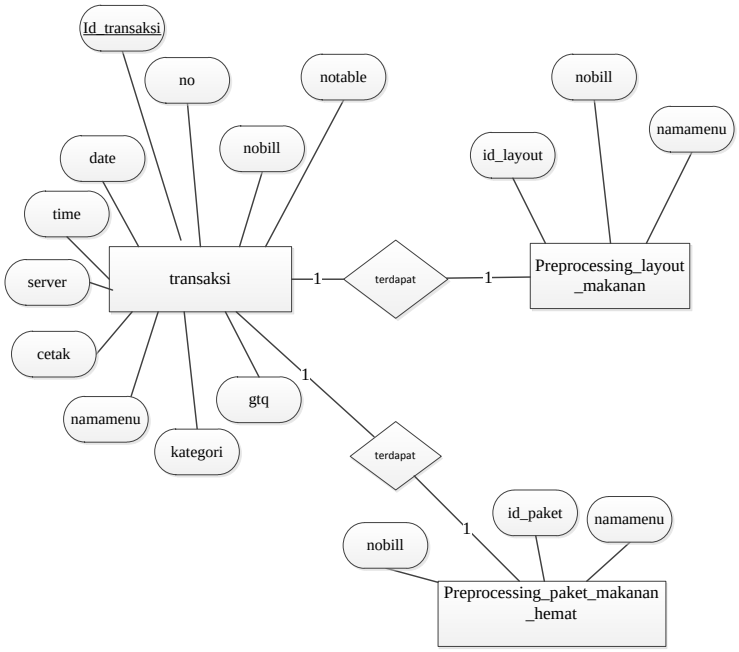
Pengguna	Hak Akses	Tingkat Pendidikan	Jenis Pelatihan
Manager	Memproses data transaksi menjadi informasi yang berguna untuk menentukan strategi bisnis yang harus diterapkan, agar dapat meningkatkan kualitas ruma makannya	S-1	Pengoperasian komputer dan penggunaan sistem

3.1.5.2 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional yang akan dibuat meliputi ERD (*Entity Relationship Diagram*) diagram konteks, Data Flow Diagram (DFD), spesifikasi proses dan kamus data.

1. ERD (*Entity Relationship Diagram*)

   Berikut ini adalah ERD dari perangkat lunak yang akan dibangun.

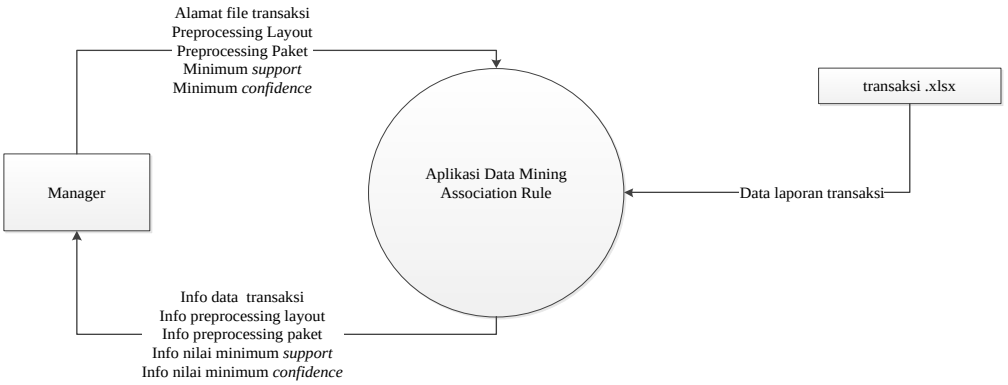


Gambar 3.128 ERD (*Entity Relationship Diagram*)

ERD diatas adalah merupakan gambaran dari database yabng digunakan yang saling berelasi, adapun data lain yang digunakan, akan tetapi tidak berelasi yaitu : data frekeunsi_makanan, data rule_layout_makanan, dan rule_paket_makanan_hemat

2. Diagram Konteks

Pembuatan diagram konteks dari sistem pendekatan struktur ini menggambarkan sistem secara garis besar yang kemudian akan dipecah menjadi bagian-bagian lebih terperinci. Adapun diagram konteks dari sistem yang akan dibangun dapat dilihat pada gambar berikut :

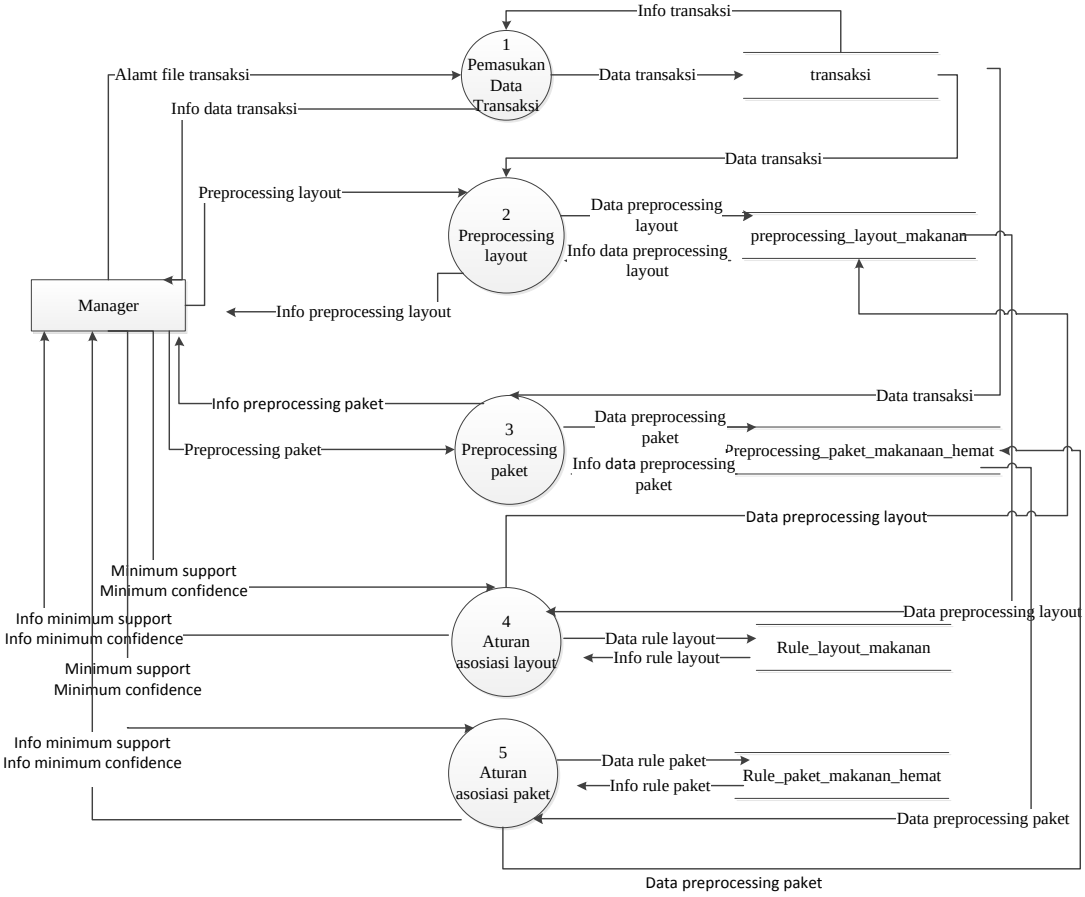


Gambar 3. 129 Diagram Konteks

3. Data Flow Diagram (DFD)

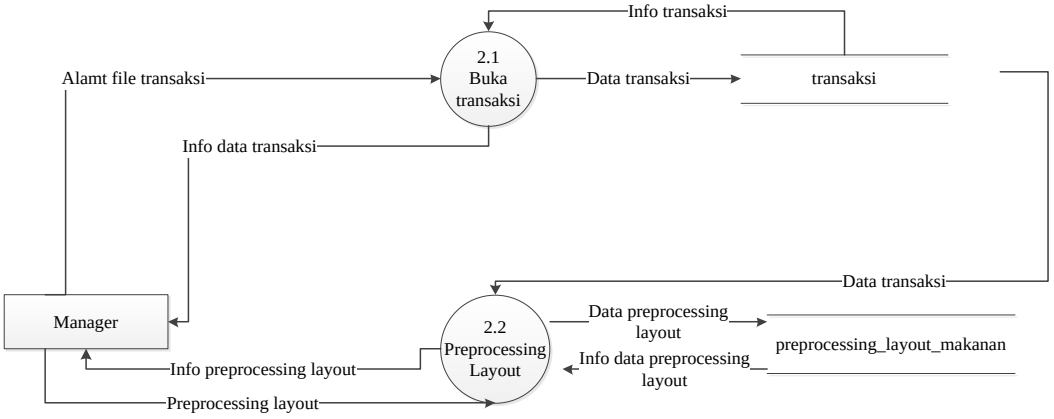
Data Flow Diagram (DFD) merupakan proses yang menggambarkan suatu alir informasi yang lebih detail dan terperinci yang merupakan pengembangan dari diagram konteks. Berikut ini merupakan *data flow diagram* dari sistem yang akan dibangun :

a. DFD Level 1 Pemasukan Data



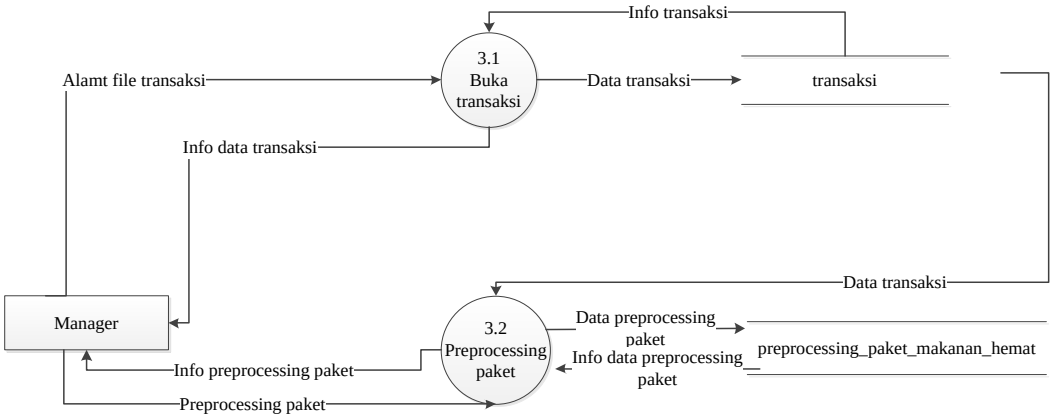
Gambar 3. 130 DFD Level 1 Pemasukan Data

b. DFD Level 2 Preprocessing Layout Makanan



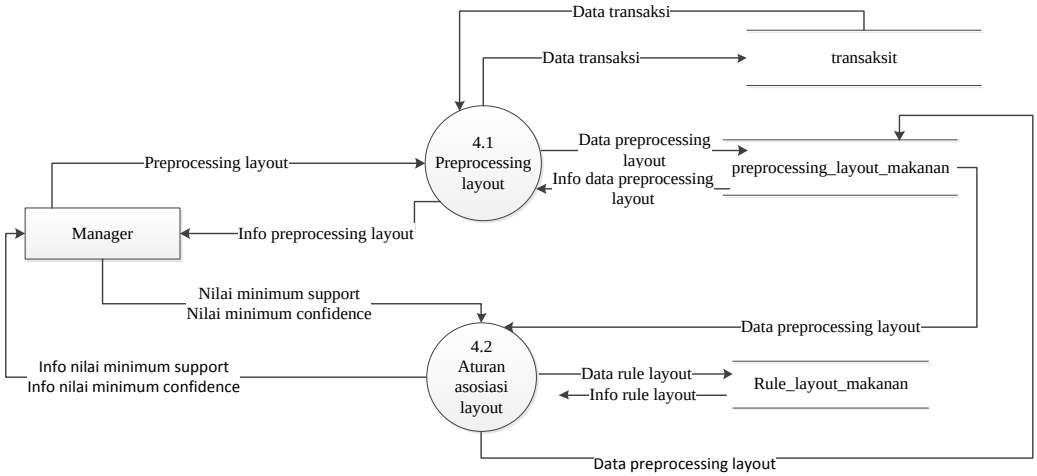
Gambar 3. 131 DFD Level 2 Preprocessing Layout Makanan

c. DFD Level 2 Preprocessing Paket Makanan Hemat



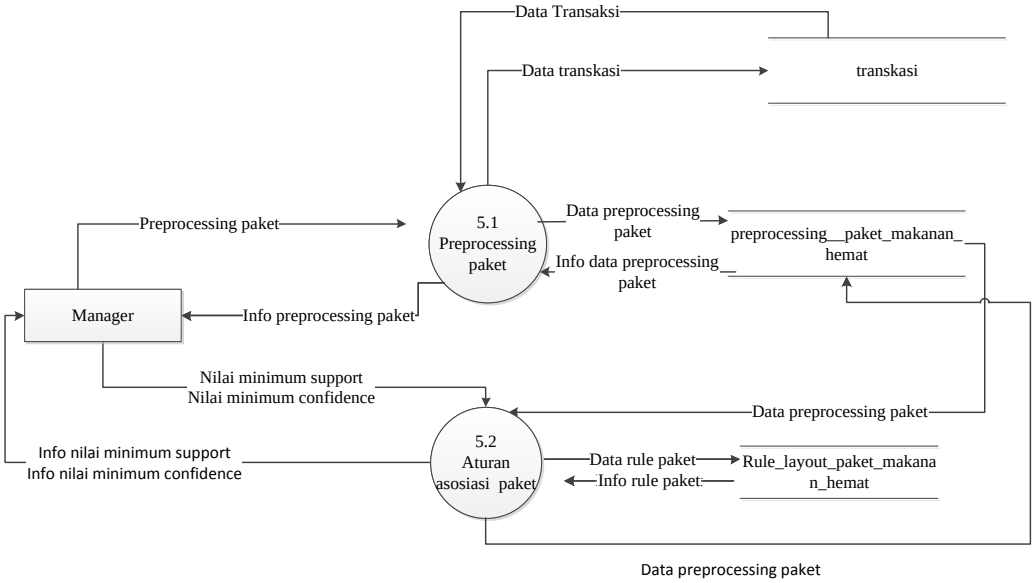
Gambar 3. 132 DFD Level 2 Preprocessing Paket Makanan Hemat

d. DFD Level 3 Aturan Asosiasi Layout Makanan



Gambar 3. 133 DFD Level 3 Atruan Asosiasi Layout Makanan

e. DFD Level 3 Aturan Asosiasi Paket Makanan Hemat



Gambar 3. 134 DFD Level 3 Aturan Asosiasi Paket Makanan Hemat

4.Spesifikasi Proses

Spesifikasi proses berfungsi untuk menjelaskan proses model aliran yang terhadap pada DFD. Adapun spesifikasi proses dari DFD yang telah dibuat akan dijelaskan pada tabel berikut :

Tabel 3. 29 Spesifikasi Proses

No	Proses	Keterangan
1	No.Proses	1.1
	Nama proses	Pemasukan data transaksi
	Source (sumber)	Manager
	Input	Alamat file transaksi
	Output	Info data transaksi
	Destination(tujuan)	Untuk menyimpan data transaksi ke tabel transaksi
	Logika proses	1. Manager memasukan alamat file excel 2. Sistem akan mengekstrak data excel ke dalam

		database
No	**Proses**	**Keterangan**
2	No.Proses	2.1
	Nama proses	Buka transaksi
	Source (sumber)	Data laporan transaksi
	Input	Alamat file transaksi
	Output	Info data transaksi
	Destination(tujuan)	Data transaksi
	Logika proses	1. Manager selesai menyimpan file ke database. 2. Sistem akan menampilkan hasil data yang telah disimpan
No	**Proses**	**Keterangan**
3	No.Proses	2.2
	Nama proses	Preprocessing Layout
	Source (sumber)	Data transaksi
	Input	Preprocessing layout
	Output	Info preprocessing layout
	Destination(tujuan)	Untuk menyimpan data ke dalam tabel prerocessing_layout_makanan
	Logika proses	1. Sistem menampilkan form preprocessing layout. 2. Sistem mengambil data transaksi yang sudah disimpan di database 3. Sistem memilih atribut nobill, namamenu, kategori 4. Sistem menghilangkan data transaksi yang memiliki item tunggal, null serta kategori yang bukan kedalam jenis makanan.
No	**Proses**	**Keterangan**
4	No.Proses	3.1
	Nama proses	Buka transaksi

	Source (sumber)	Data laporan transaksi
	Input	Alamat file transaksi
	Output	Info data transaksi
	Destination(tujuan)	Data transaksi
	Logika proses	1. Manager selesai menyimpan file ke database. 2. Sistem akan menampilkan hasil data yang telah disimpan
No	**Proses**	**Keterangan**
5	No.Proses	3.2
	Nama proses	Preprocessing Paket
	Source (sumber)	Data transaksi
	Input	Preprocessing paket
	Output	Info preprocessing paket
	Destination(tujuan)	Untuk menyimpan data ke dalam tabel prerocessing_paket_makanan_hemat
	Logika proses	1. Sistem menampilkan form preprocessing paket 2. Sistem mengambil data transaksi yang sudah disimpan di database 3. Sistem memilih atribut nobill, namamenu, kategori 4. Sistem menghilangkan data transaksi yang memiliki item tunggal, null.
No	**Proses**	**Keterangan**
6	No.Proses	4.1
	Nama proses	Preprocessing layout
	Source (sumber)	Data transaksi
	Input	Data transaksi
	Output	Info data preprocessing layout
	Destination(tujuan)	Data preprocessing_layout_makanan
	Logika proses	1. Sistem menampilkan data transaksi yang sudah

		disimpan 2. Sistem mengambil data transaksi yang sudah disimpan di database 3. Sistem memilih atribut nobill, namamenu, kategori 4. Sistem menghilangkan data transaksi yang memiliki item tunggal, null serta kategori yang bukan kedalam jenis makanan.
No	**Proses**	**Keterangan**
7	No.Proses	4.2
	Nama proses	Aturan asosiasi layout
	Source (sumber)	Manager, Data preprocessing layout
	Input	Data preprocessing layout
	Output	Info data preprocessing layout
	Destination(tujuan)	Untuk menyimpan data kedalam tabel rule_layout_makanan, dan menghasilkan aturan asosiasi dengan nilai minimum *support* dan *confidence*
	Logika proses	1. Manager memasukan nilai untuk minimum support dan *confidence*. 2. Sistem akan mengurutkan data secara *ascending* berdasarkan *confidence* 3. Kemudian sistem akan menghapus nillai minimum *support* dan *confidence* yang tidak memenuhi 4. Setelah data sudah diihilangkan sesuai masukan dari manager sebelumnya, sistem akan men-*scan* dari setiap transaksi secara berurutan lalu dibuat pohon *fp-tree*. 5. Kemudian sistem akan melakukan pembentukan *conditional pattern base*

No	**Proses**	**Keterangan**
8	No.Proses	5.1
	Nama proses	Preprocessing paket
	Source (sumber)	Data transaksi
	Input	Data transaksi
	Output	Info data preprocessing paket
	Destination(tujuan)	Data preprocessing_paket_makanan_hemat
	Logika proses	1. Sistem menampilkan data transaksi yang sudah disimpan 2. Sistem mengambil data transaksi yang sudah disimpan di database 3. Sistem memilih atribut nobill, namamenu, kategori 4. Sistem menghilangkan data transaksi yang memiliki item tunggal, null.
No	**Proses**	**Keterangan**
9	No.Proses	5.2
	Nama proses	Aturan asosiasi paket
	Source (sumber)	Manager, Data preprocessing paket
	Input	Data preprocessing paket
	Output	Info data preprocessing paket
	Destination(tujuan)	Untuk menyimpan data kedalam tabel rule_paket_makanan_hemat , dan menghasilkan aturan asosiasi dengan nilai minimum *support* dan *confidence*
	Logika proses	1. Manager memasukan nilai untuk minimum support dan *confidence*. 2. Sistem akan mengurutkan data secara *ascending* berdasarkan *confidence*

		3. Kemudian sistem akan menghapus nillai minimum *support* dan *confidence* yang tidak memenuhi 4. Setelah data sudah diihilangkan sesuai masukan dari manager sebelumnya, sistem akan men-*scan* dari setiap transaksi secara berurutan lalu dibuat pohon *fp-tree*. 5. Kemudian sistem akan melakukan pembentukan *conditional pattern base*

5. Kamus Data

Kamus data berfungsi untuk menjelaskan definisi semua data yang mengalir dan berisi dari DFD. Adapun kamus data dari *data flow diagram* perangkat lunak yang akan dibangun yang dijelaskan pada tabel berikut :

Tabel 3. 30 Kamus Data

Nama Data	Transaksi
Deskripsi	Memilih file data yang akan diproses
Bentuk Data	File excel .xlsx
Struktur Data	No, NoTable, NoBill, Date, Server, Time, Cetak, Qty, NamaMenu, Kategori, Harga, SubTotal, Pajak, GrandTotal, Cash, ChangeDue
No	{0..9}
NoTable	{0..9}
NoBill	{A..Z | a..z} {0..9}
Date	{A..Z | a..z}
Time	{00:00:00}
Server	{A..Z | a..z} {0..9}
Cetak	{A..Z | a..z} {0..9}
NamaMenu	{A..Z | a..z} {0..9}
Kategori	{A..Z | a..z}{0..9}
Qty	{0..9}

Harga	{A..Z | a..z} {0..9}
Subtotal	{A..Z | a..z} {0..9}
Pajak	{A..Z | a..z} {0..9}
Grandtotal	{A..Z | a..z} {0..9}
Cash	{A..Z | a..z}{0..9}
Changedue	{A..Z | a..z} {0..9}
Nama Data	**Preprocessing Layout Makanan**
Deskripsi	Hasil pemilihan atribut dan pembersihan data
Bentuk Data	File excel .xlsx
Struktur Daa	NoBill, NamaMenu
NoBill	{A..Z | a..z} {0..9}
NamaMenu	{A..Z | a..z} {0..9}
Nama Data	**Preprocessing Paket Makanan Hemat**
Deskripsi	Hasil pemilihan atribut dan pemebersihan data
Bentuk Data	File excel .xlsx
Struktur Data	NoBill, NamaMenu
NoBill	{A..Z | a..z} {0..9}
NamaMenu	{A..Z | a..z} {0..9}
Nama Data	**Frekeunsi Makanan Layout**
Deskripsi	Hasil penentuan kode menu, frekeunsi makanan, hasil scan, FOO
Bentuk Data	File excel .xlsx
Struktur Data	No, menumakanan, sc, ns
No	{0..9}
menumakanan	{A..Z | a..z}
Sc	{0..9}
ns	{0..9}
Nama Data	**Rule Layout Makanan**
Deskripsi	Hasil penentuan minimum support dan minimum confidence
Bentuk Data	File excel .xlsx
Struktur Data	Susunan_layout_makanan, confidence

Susunan_layout_makanan	{A..Z | a..z}
Confidence	{0..9}
Nama Data	**Rule Paket Makanan Hemat**
Deskripsi	Hasil penentuan minimum support dan minimum confidence
Bentuk Data	File excel .xlsx
Struktur Data	Paket_makanan_hemat, confidence
Paket_makanan_hemat	{A..Z | a..z}
Confidence	{0..9}

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 Basis Data

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

3.2.1.1 Struktur Tabel

Struktur tabel pada aplikasi yang akan dibangun terdiri dari 4 tabel. Rinciannya sebagai berikut :

Tabel 3. 31 Struktur Tabel Transaksi

Nama Field	Tipe Data	Panjang	Keterangan
Id_transaksi	Int	12	PK
No	Int	11	
NoTabel	Int	11	
NoBill	Varchar	7	
Date	Date	25	
Time	Varchar	25	
Server	Varchar	15	
Cetak	Varchar	15	
NamaMenu	Varchar	30	

Nama Field	Tipe Data	Panjang	Keterangan
Kategori	Varchar	20	
Qty	Int	11	
Harga	Varchar	15	
SubTotal	Varchar	15	
Pajak	Varchar	15	
GrandTotal	Varchar	15	
Cash	Varchar	15	
ChangeDue	Varchar	15	

Tabel 3. 32 Struktur Tabel Preprocessing Layout Makanan

Nama Field	Tipe Data	Panjang	Keterangan
Id_layout	Int	12	PK
Id_transaksi	Int	12	FK references transaksi.id_transaksi
NoBill	Varchar	7	
NamaMenu	Varchar	30	

Tabel 3. 33 Struktur Tabel Preprocessing Paket Makanan Hemat

Nama Field	Tipe Data	Panjang	Keterangan
Id_paket	Int	12	PK
Id_transaksi	Int	12	FK references transaksi.id_transaksi
NoBill	Varchar	7	
NamaMenu	Varchar	30	

Karena dibutuhkannya sebuah tempat penyimpanan untuk hasil pencarian *frequent itemset*, maka dibuatlah tabel frekeunsi_makanan, rule_layout_makanan, dan rule_paket_makanan_hemat dengan detail pada tiap tabel seperti dibawah ini :

Tabel 3. 34 Struktur Tabel Preprocessing Frekeunsi Makanan

Nama Field	Tipe Data	Panjang	Keterangan
No	int	11	
Menumakanan	text	30	
sc	Int	11	
ns	float	-	

Tabel 3. 35 Struktur Tabel Rule Layout Makanan

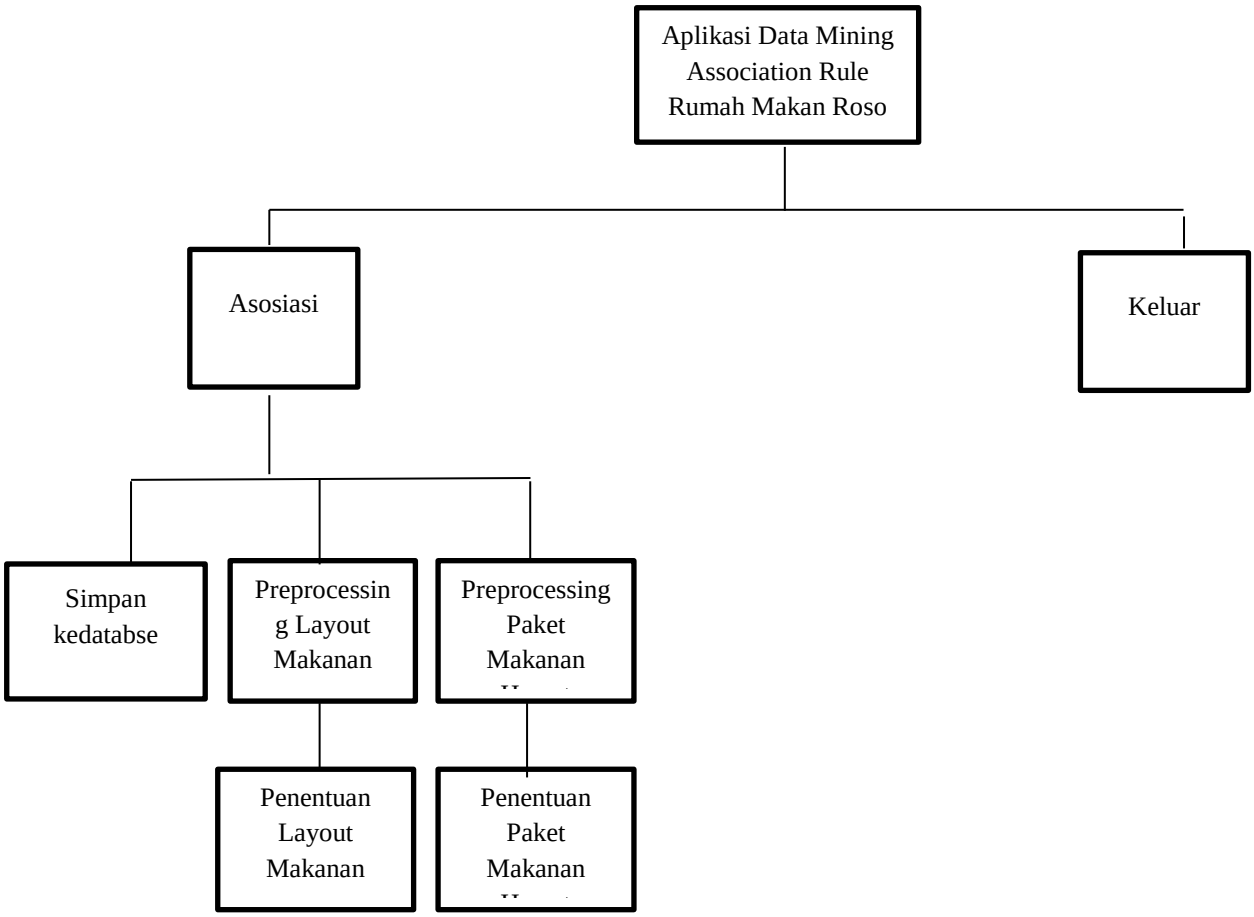
Nama Field	Tipe Data	Panjang	Keterangan
Susunan_layout_makanan	text	-	
confidence	float	-	

Tabel 3. 36 Struktur Tabel Rule Paket Makanan Hemat

Nama Field	Tipe Data	Panjang	Keterangan
Susunan_layout_makanan	text	-	
confidence	float	-	

3.2.2 Perancangan Struktur Menu

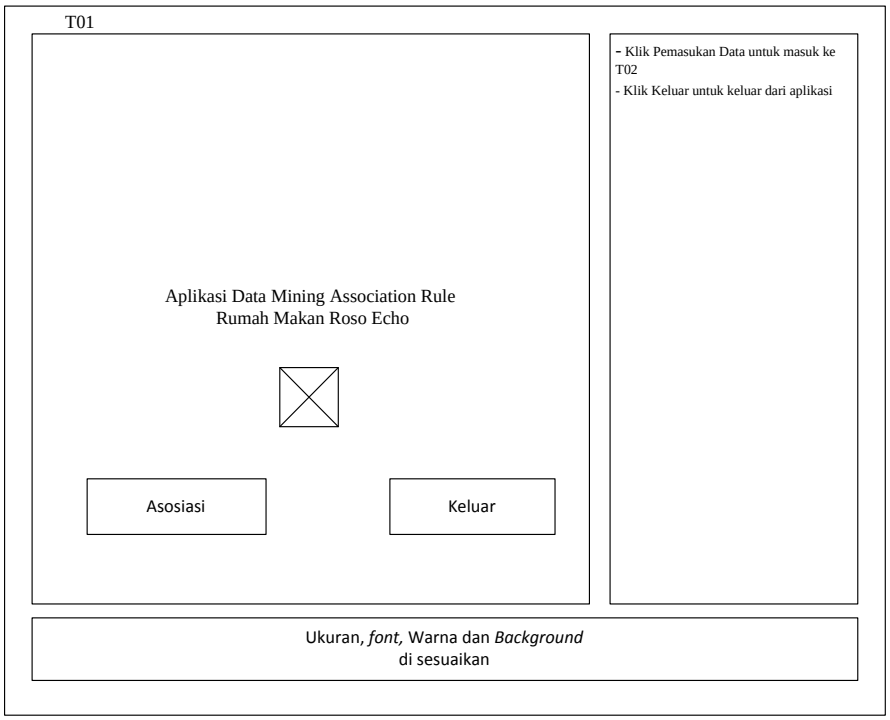
Dalam pembangunan perangkat lunak diperlukan struktur menu untuk memudahkan pengguna dalam menavigasi fungsional sistem. Adapun perancangan struktur menu untuk aplikasi yang akan dibangun dapat dilihat pada gambar berikut :



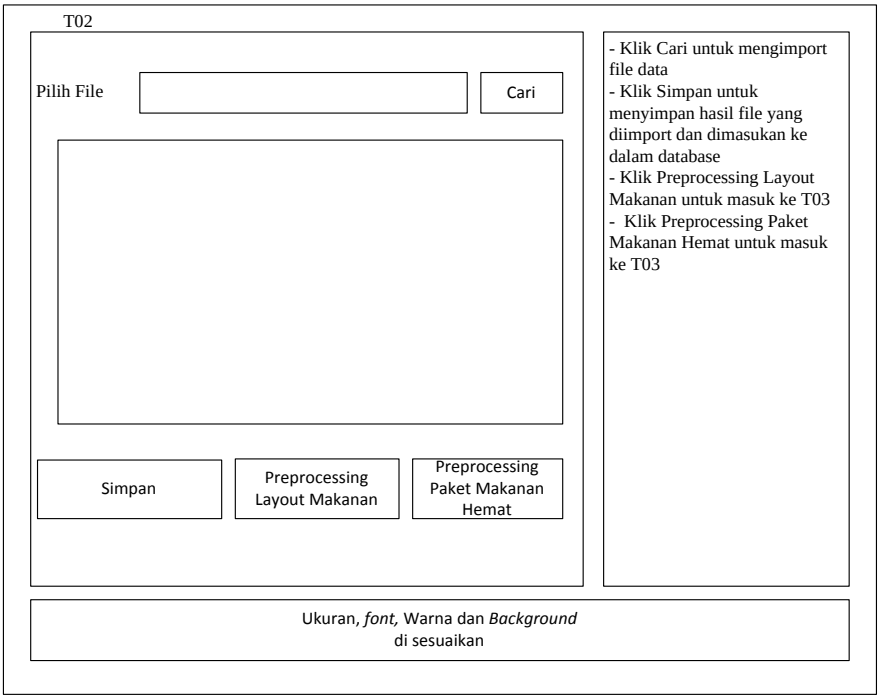
Gambar 3. 135 Struktur Menu

3.2.3 Perancangan Antar Muka

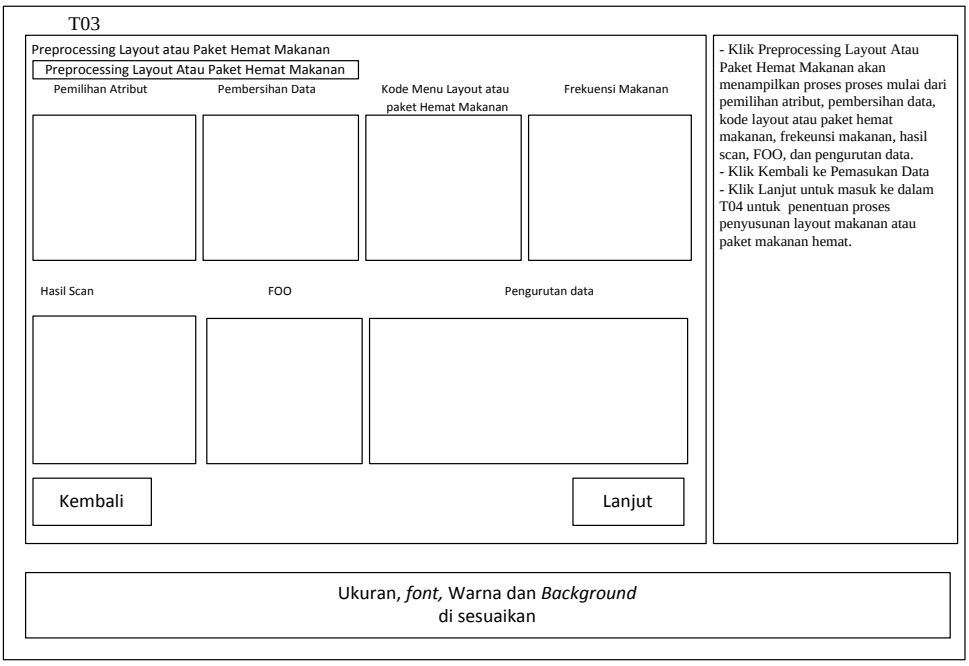
Perancangan antar muka berfungsi untuk memberi gambaran tentang pernagkat lunak yang akan dibangun, supaya lebih memudahkan dalam pembuatan perangkat lunak nantinya. Adapun perancangan antar muka untuk perangakt lunak yang akan dibangun akan dijelaskan berikut ini :



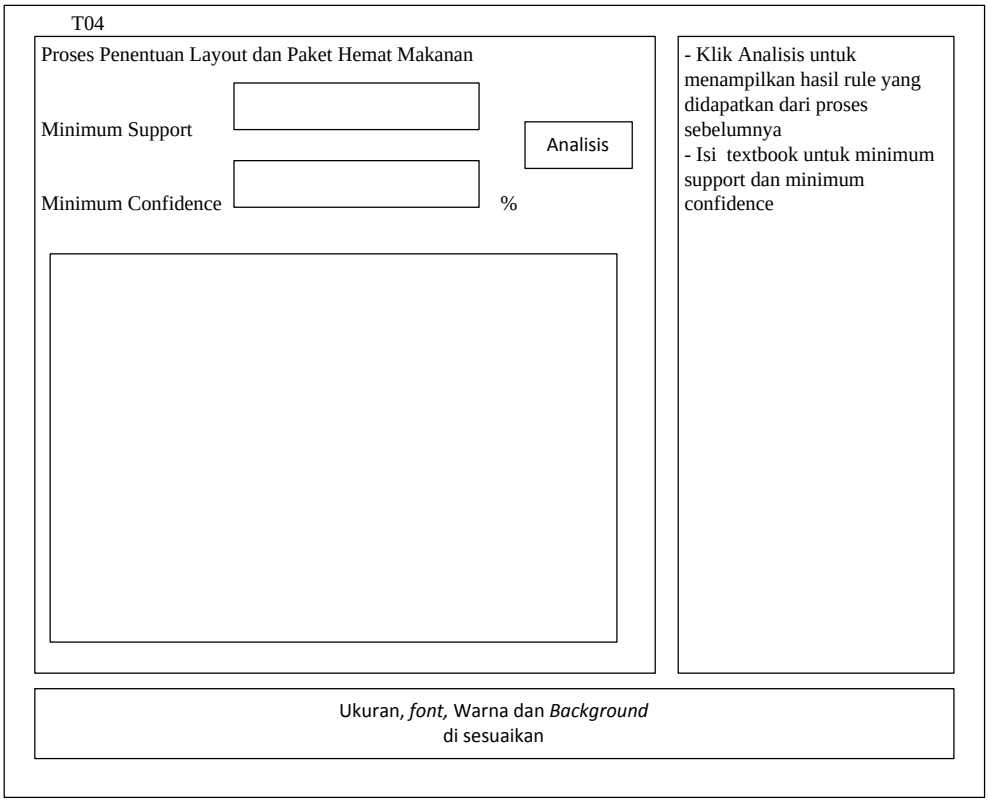
Gambar 3. 136 Perancangan Antar Muka Menu Utama



Gambar 3. 137 Perancangan Antar Muka Pemasukan Data



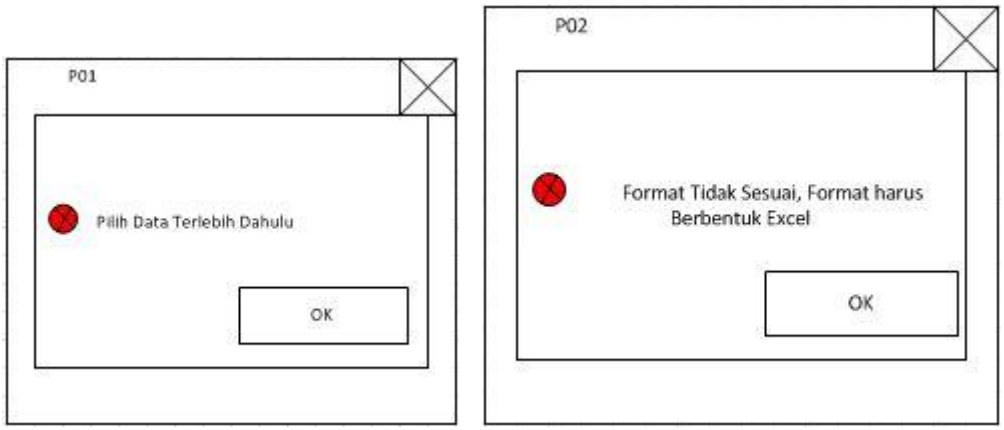
Gambar 3. 138 Perancangan Antar Muka Preprocessing

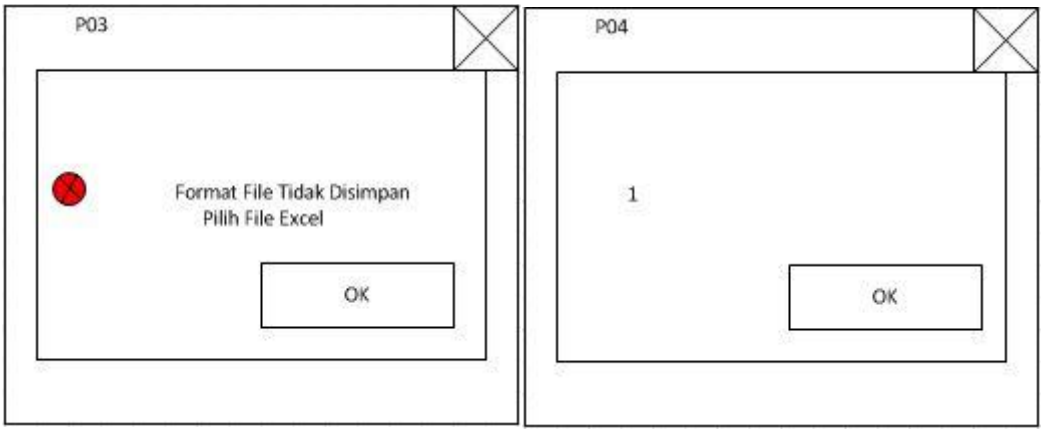


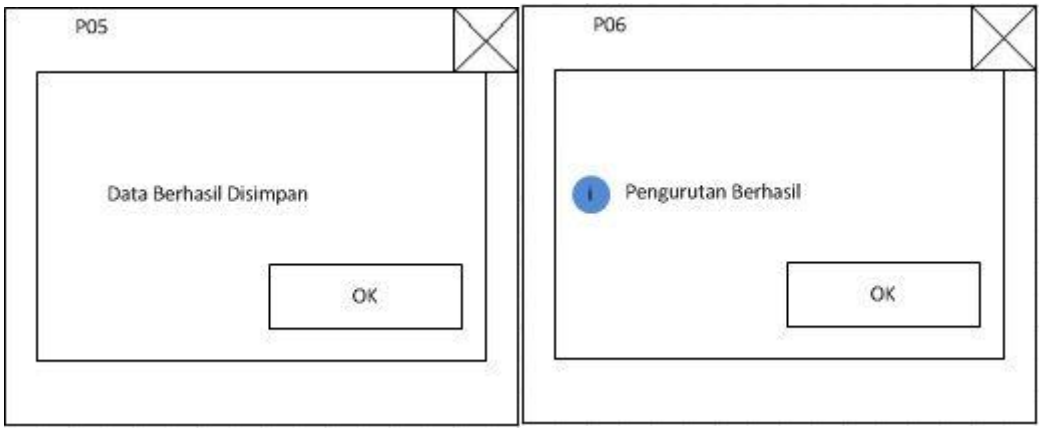
Gambar 3. 139 Perancangan Antar Muka Aturan Asosiasi

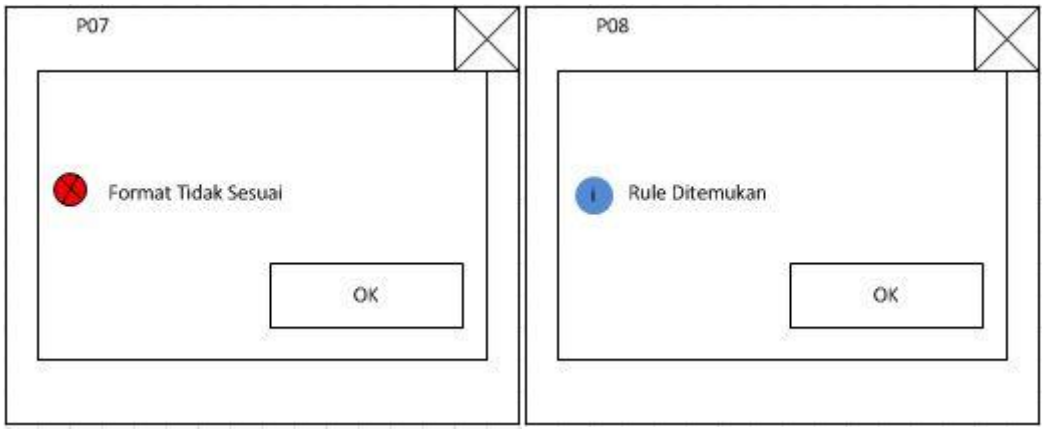
3.2.4 Perancangan Pesan

Adapun perancangan pesan dapat dilihat pada gambar dibawah ini :





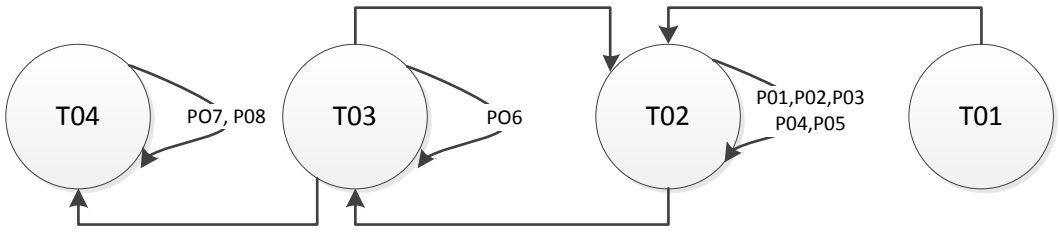




Gambar 3. 140 Perancangan Pesan

3.2.5 Jaringan 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. 141 Jaringan Semantik