BAB III
METODOLOGI DAN PERANCANGAN SISTEM

3.1 Metodologi

Pada bab ini akan dibahas metode dan langkah-langkah yang digunakan untuk melakukan sequential pattern mining pada data transaksi penjualan menggunakan algoritma SPADE.

Penelitian dilakukan dengan langkah-langkah sebagai berikut:

1. Melakukan studi literatur yang berkaitan dengan data mining, *sequential pattern mining*, dan algoritma SPADE.
2. Merancang sistem untuk sequential pattern mining pada data transaksi.
3. Implementasi sistem berdasarkan analisis dan perancangan yang dilakukan.
4. Melakukan pengujian terhadap sistem.
5. Melakukan evaluasi tingkat keberhasilan sistem dan analisis hasil pengujian.

3.2 Analisa Sistem

3.2.1 Deskripsi Sistem

Sistem ini dibangun untuk membantu minimarket dalam mengambil kebijakan terkait tentang produk yang dijual. Tujuan dari sistem ini adalah untuk mencari hubungan pembelian produk antara 1 produk dengan produk yang lain berdasarkan urutan pembeliannya. Dengan hasil dari sistem ini, diharapkan dapat membantu minimarket dalam mengambil kebijakan terkait produk yang dijual, seperti promosi, penentuan stok dan lain sebagainya.

Hasil dari algoritma SPADE ini kemudian diuji dengan parameter uji yang digunakan berdasarkan pada support dan confidence.

Sistem ini menghasilkan rule dengan membentuk keterkaitan antar produk dengan mempertimbangkan urutan pembelian. Data yang digunakan untuk menentukan hasil dari sistem ini adalah:

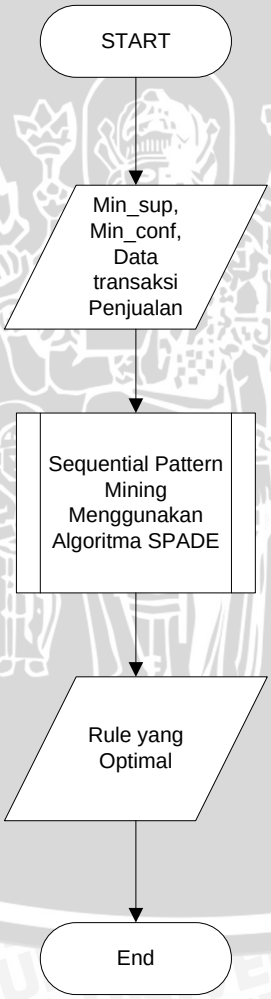
1. Pelanggan
2. Waktu transaksi
3. Produk

3.2.2 Deskripsi Data

Pada sistem ini, data yang akan digunakan adalah dataset transaksi penjualan toko elektronik yang didapatkan dari website www.customers-dna.com. Dari dataset ini, atribut yang akan digunakan dalam sistem ini adalah pelanggan, waktu transaksi dan produk yang dibeli. Jumlah data yang digunakan adalah 4000 data yang dibagi menjadi beberapa bagian dalam berbagai macam uji coba.

3.3 Perancangan Proses

Pada bagian ini akan dijelaskan tentang proses membangun sistem analisa transaksi penjualan. Proses analisa dilakukan menggunakan algoritma SPADE. Flowchart deskripsi sistem digambarkan pada Gambar 3.1.



Gambar 3.1 Flowchart deskripsi sistem

Untuk penjelasan flowchart di atas adalah sebagai berikut:

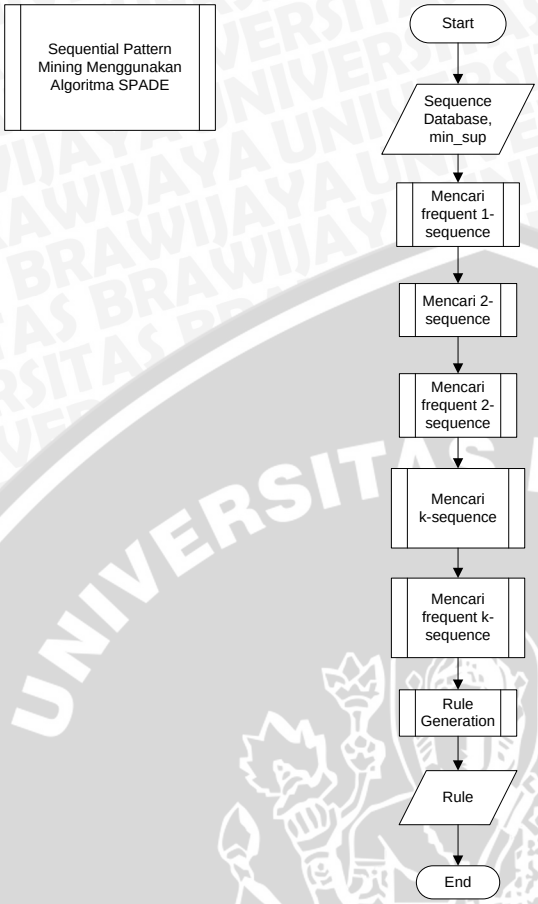
1. Input yang digunakan dalam sistem ini adalah min_sup yaitu support minimal, min_conf yaitu confidence minimal, dan data transaksi penjualan
2. Algoritma yang digunakan adalah SPADE. Algoritma ini mencari frequent sequence atau urutan pembelian yang sering muncul dari data transaksi penjualan menggunakan min_sup. Setelah semua frequent sequence ditemukan, akan dibentuk rule dari sequence tersebut menggunakan min_conf dan lift rasio.
3. Output dari program ini adalah rule – rule yang optimal berdasarkan lift rasio.

3.3.1 Sequential Pattern Mining Menggunakan Algoritma SPADE

Tahap proses algoritma SPADE adalah sebagai berikut:

a. Input awal adalah database sequence dan min_sup. Kemudian dicari frequent 1 sequence dari sequence database yang memenuhi min_sup
b. Setelah ditemukan semua frequent 1-sequence, dicari frequent 2-sequence dengan cara melakukan join pada semua frequent 1 sequence dan dicari yang memenuhi syarat min_sup.
c. Langkah berikutnya adalah mencari frequent k-sequence. Tahap ini akan dijelaskan lebih lanjut.
d. Setelah ditemukan semua frequent sequence, kemudian dibentuk rule dari frequent sequence tersebut.
e. Output program adalah rule.

Untuk lebih jelasnya dapat dilihat pada Gambar 3.2 berikut.



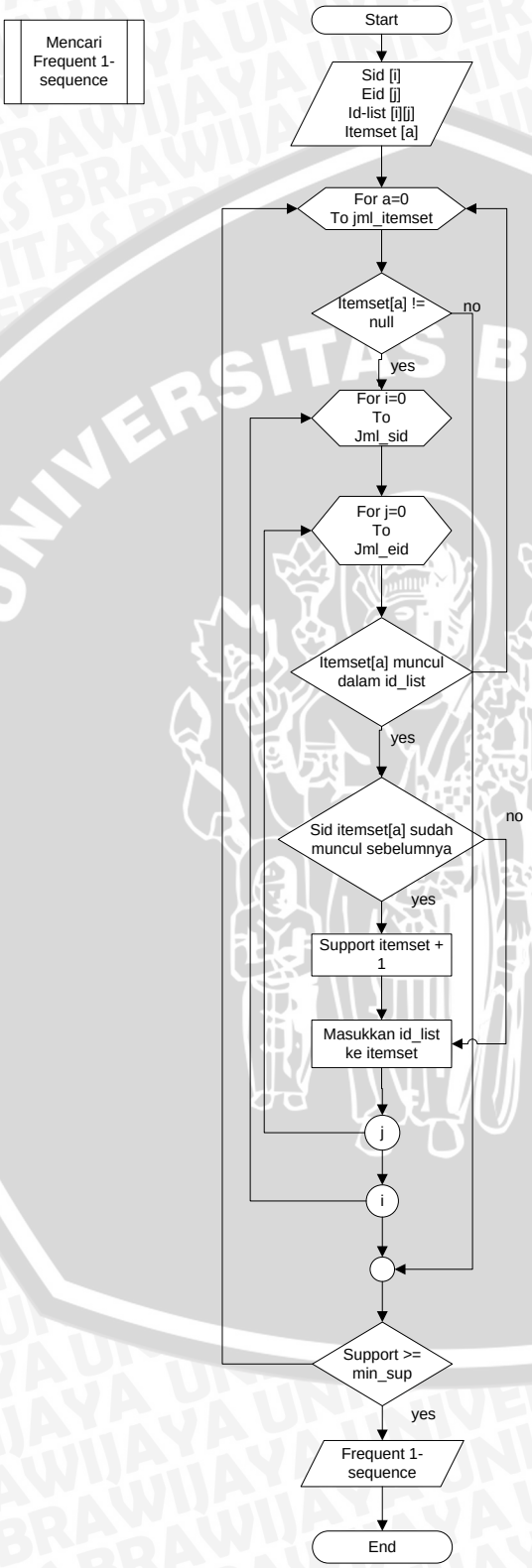
Gambar 3.2 Flowchart algoritma SPADE

3.3.1.1 Mencari Frequent 1-sequence

Langkah dalam mencari frequent 1-sequence adalah sebagai berikut

1. Dimasukkan inputan database sequence yang terdiri dari sid, eid, id-list (pasangan sid, eid) dan itemset.
2. Dilakukan perulangan untuk setiap itemset, dicek pada setiap sid dan eid.
3. Dilakukan pengecekan apakah itemset muncul dalam pasangan id-list.
4. Apabila muncul maka id-list tersebut dimasukkan ke dalam id-list itemset tersebut dan membentuk 1-sequence.
5. Dilakukan pengecekan apakah sid yang dimasukkan sudah muncul sebelumnya, apabila belum maka nilai support ditambah 1. Apabila sudah ada maka id-list hanya dimasukkan saja, tapi nilai support tidak ditambahkan.
6. Dilakukan pengecekan jumlah support untuk masing – masing 1-sequence, apabila lebih dari min_sup, maka diterima sebagai frequent 1-sequence.

Untuk lebih jelasnya, proses mencari 1-sequence digambarkan dalam Gambar 3.3 berikut.



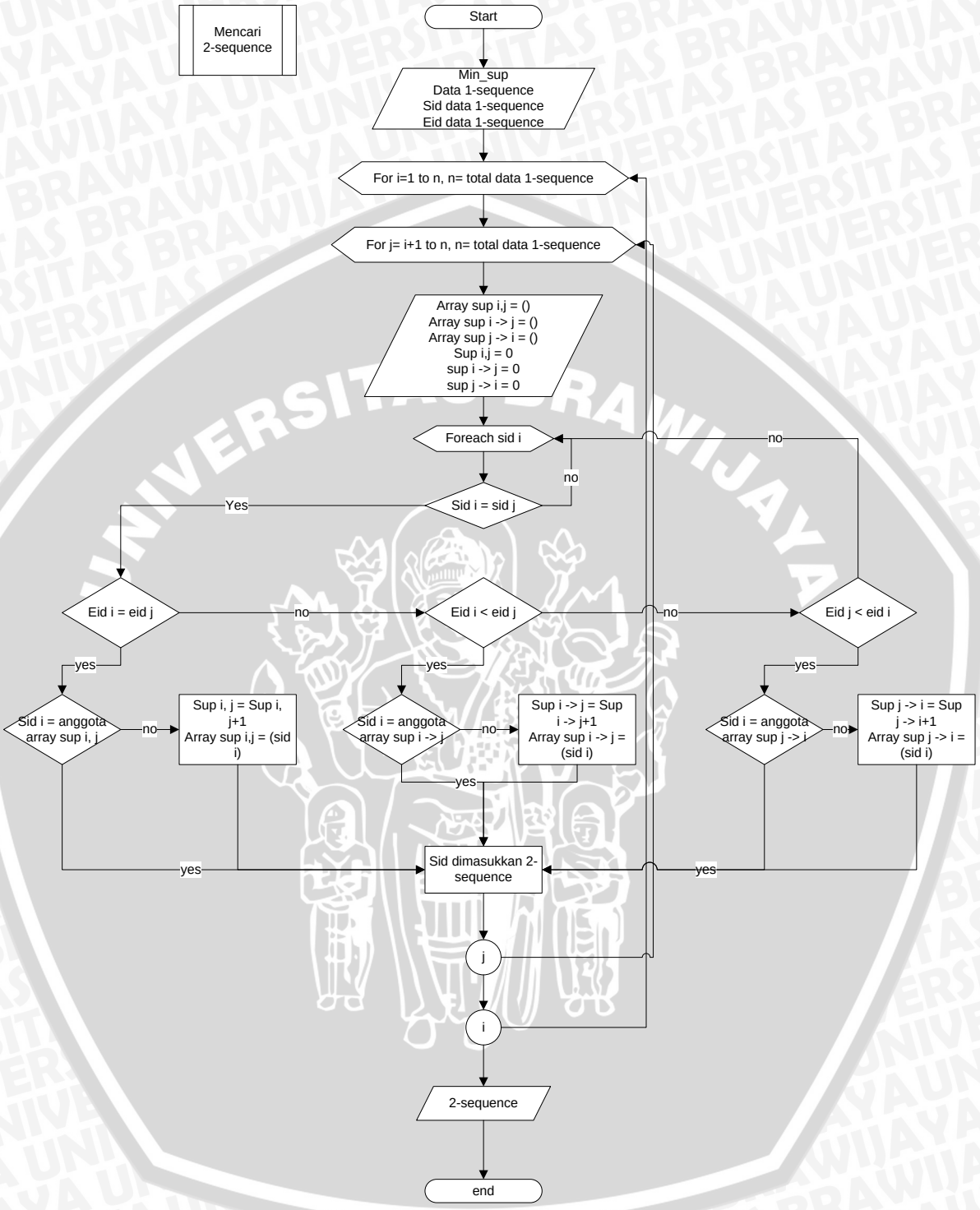
Gambar 3.3 Mencari Frequent 1-sequence

3.3.1.2 Mencari 2-sequence

Tahap proses mencari 2-sequence adalah sebagai berikut:

1. Melakukan input min_sup, dan data 1-sequence yang setiap data 1-sequence tersebut terdiri dari sid dan eid.
2. Dilakukan perulangan untuk setiap item 1-sequence dan item 1-sequence setelahnya.
3. Untuk setiap sid dilakukan pengecekan apakah sid pada item 1-sequence ke i sama dengan item ke j.
4. Apabila sama, maka dilakukan apakah eid ke i sama dengan, kurang dari atau lebih dari eid item ke j.
5. Jika eid i sama dengan eid j maka membentuk 2-sequence i,j dimana i dan j muncul bersamaan. Jika eid j lebih dari eid i maka membentuk 2-sequence i → j, dimana item j terjadi setelah item i, sebaliknya jika eid i lebih dari eid j makan membentuk 2-sequence j → i.
6. Untuk masing – masing kemungkinan dilakukan pengecekan apakah sid sudah pernah muncul sebelumnya, jika belum maka nilai support ditambah 1.
7. Apabila sid sudah pernah muncul ataupun belum, 2-sequence yang terbentuk dimasukkan ke dalam daftar 2-sequence.

Untuk lebih jelas dapat dilihat pada Gambar 3.4.

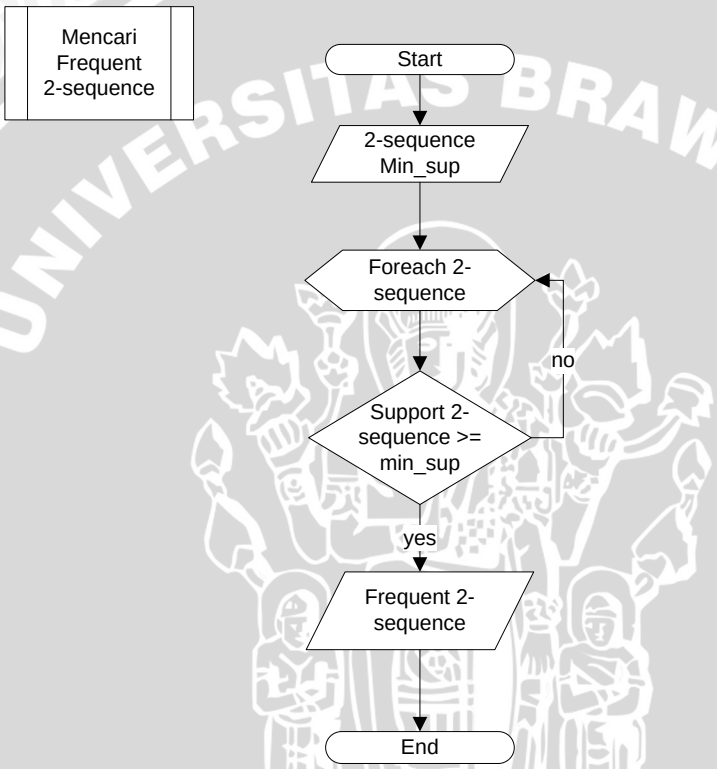


Gambar 3.4 Mencari 2-sequence

3.3.1.3 Mencari Frequent 2-sequence

Tahap mencari frequent 2-sequence adalah sebagai berikut:

1. Dilakukan input min_sup dan data 2-sequence.
2. Dilakukan pengecekan untuk setiap anggota 2-sequence apakah supportnya memenuhi syarat lebih dari atau sama dengan min_sup. Apabila memenuhi maka dimasukkan dalam data frequent 2-sequence.



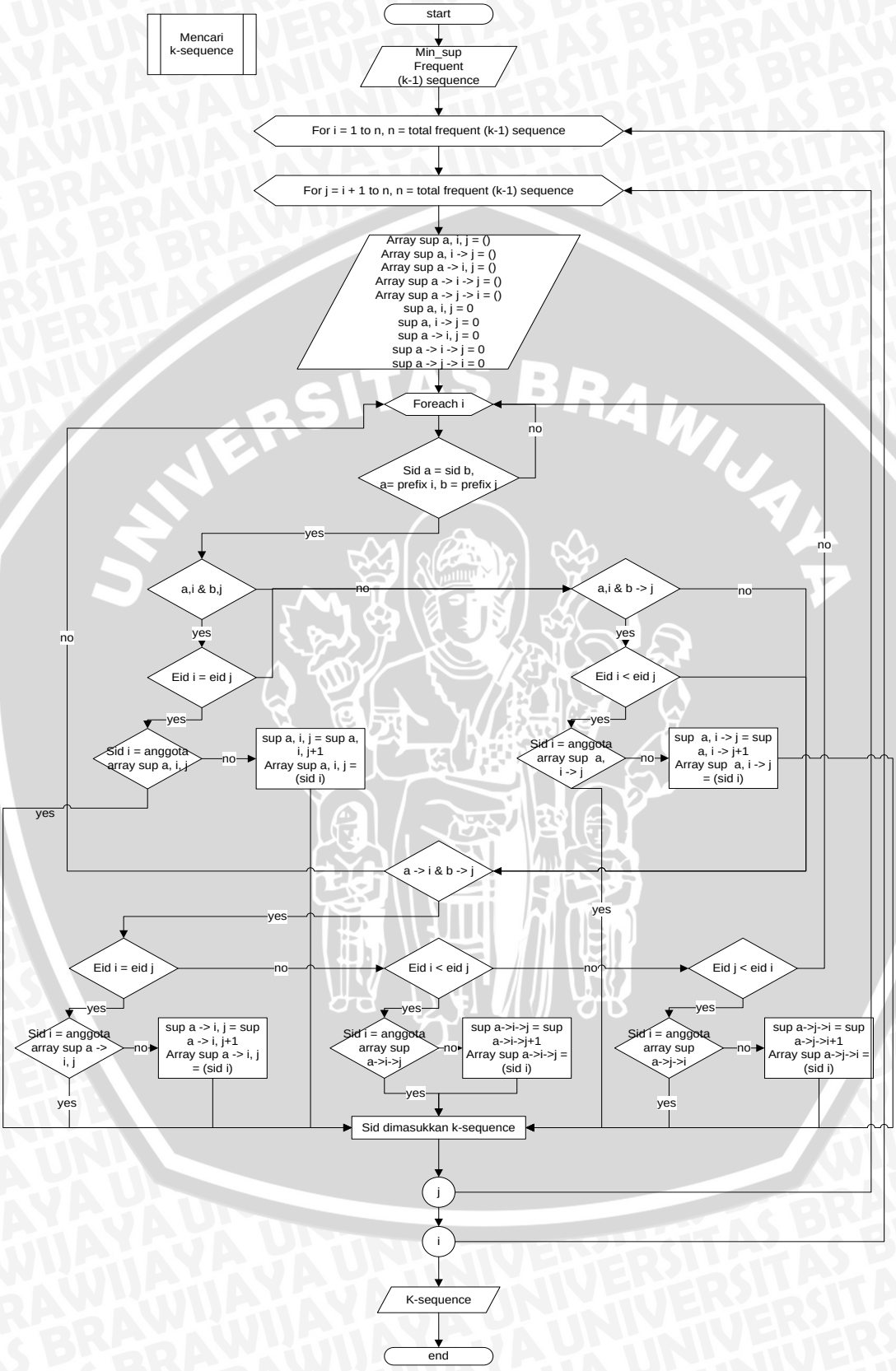
Gambar 3.5 Mencari frequent 2-sequence

3.3.1.4 Mencari k-sequence

Tahap mencari k-sequence adalah sebagai berikut:

1. Diinputkan min_sup dan frequent (k-1)-sequence.
2. Untuk setiap anggota (k-1)-sequence dilakukan pengecekan apakah prefix item ke-i (a) dan prefix item setelah i atau ke-j (b) sama atau tidak.
3. Jika memiliki prefix yang sama, kemudian dicek hubungan item i dan j:
   a. Jika a,i dan a,j maka dicek apakah eid i = eid j, jika memenuhi maka terbentuk k-sequence a,i,j.

b. Jika a,i dan a→j maka dicek apakah eid j lebih besar dari eid i, jika benar maka terbentuk k-sequence a,i→j
c. Jika a→i dan a→j maka terdapat 3 kemungkinan yaitu:
   I. Jika eid i sama dengan eid j maka membentuk k-sequence a→i,j.
   II. Jika eid j lebih besar dari eid i maka membentuk k-sequence a→i→j.
   III. Jika eid i lebih besar dari eid j maka membentuk k-sequence a→j→i
4. Untuk masing – masing kemungkinan k-sequence dilakukan pengecekan apakah sid sudah pernah muncul sebelumnya, jika belum maka nilai support ditambah 1.
5. Apabila sid sudah pernah muncul ataupun belum, k-sequence yang terbentuk dimasukkan ke dalam daftar k-sequence.



Gambar 3.6 Mencari k-sequence

3.3.1.5 Mencari Frequent k-sequence

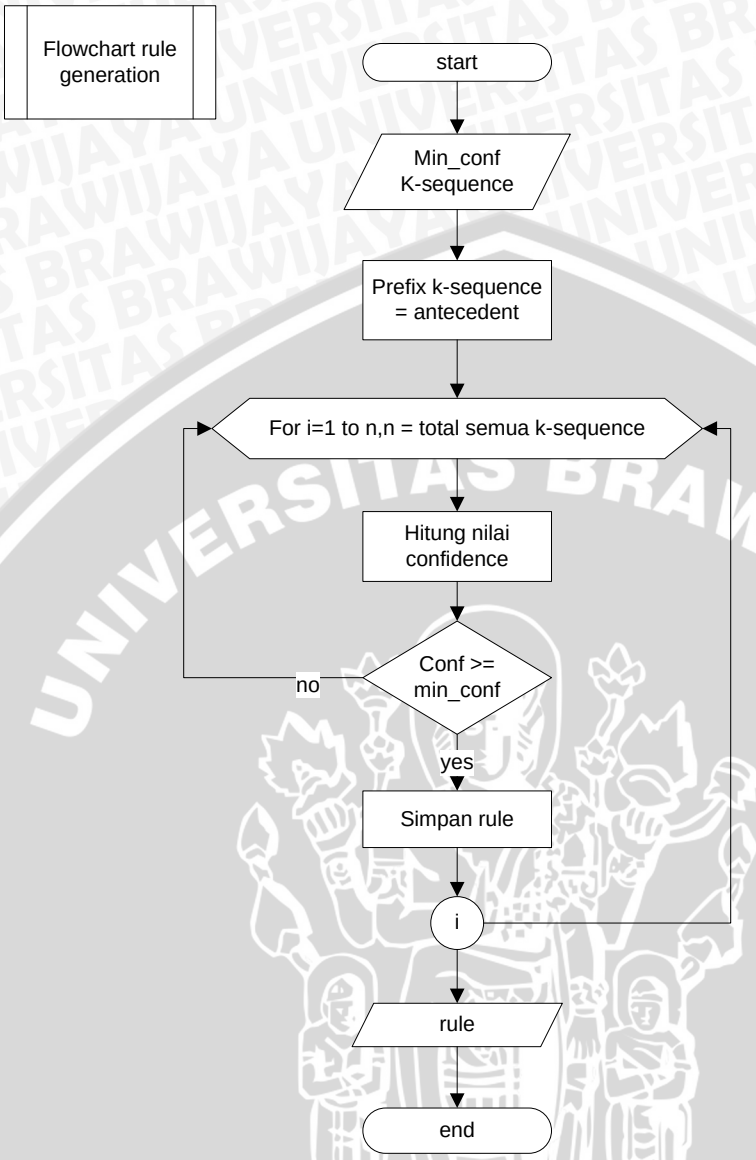
Untuk mencari frequent k-sequence, digunakan proses yang sama seperti dalam mencari frequent 2-sequence. Tetapi input yang digunakan adalah k-sequence.

3.3.1.6 Rule Generation

Tahap dalam pembentukan rule adalah sebagai berikut:

1. Input data adalah min_conf dan semua k-sequence, yang terdiri dari semua frequent sequence kecuali 1-sequence.
2. Dibentuk rule untuk setiap sequence, prefixnya merupakan antecedent pada rule.
3. Dilakukan perulangan untuk setiap rule dihitung nilai confidencenya, kemudian dilakukan pengecekan apakah memenuhi min_conf.
4. Jika memenuhi maka rule disimpan.

Untuk lebih jelasnya dapat dilihat pada Gambar 3.7



Gambar 3.7 Rule Generation

3.4 Contoh Perhitungan Manual

Dari data transaksi yang digunakan, diambil data transaksi dari 4 pelanggan untuk digunakan dalam perhitungan manual. Transaksi ini terjadi sepanjang tahun 2001. Data tersebut dapat ditunjukkan dalam Tabel 3.1 – 3.4 berikut.

Tabel 3.1 Tabel transaksi pelanggan 102084120

No. Pelanggan	Tanggal	Item
102084120	2001-03	34
102084120	2001-04	1
102084120	2001-04	50
102084120	2001-04	96
102084120	2001-04	110
102084120	2001-05	80
102084120	2001-05	84
102084120	2001-06	41
102084120	2001-06	92
102084120	2001-07	35
102084120	2001-07	36
102084120	2001-08	18
102084120	2001-08	36
102084120	2001-08	99
102084120	2001-09	87
102084120	2001-10	38
102084120	2001-10	112
102084120	2001-11	75
102084120	2001-11	102
102084120	2001-12	17
102084120	2001-12	76
102084120	2001-12	113

Tabel 3.2 Tabel transaksi pelanggan 105074329

No. Pelanggan	Tanggal	Item
105074329	2001-01	60
105074329	2001-01	80
105074329	2001-01	438
105074329	2001-07	70

105074329	2001-07	168
105074329	2001-07	191
105074329	2001-08	18
105074329	2001-08	54
105074329	2001-09	23
105074329	2001-09	61
105074329	2001-09	623
105074329	2001-12	141
105074329	2001-12	154
105074329	2001-12	627
105074329	2001-12	1

Tabel 3.3 Tabel transaksi pelanggan 102447799

No. Pelanggan	Tanggal	Item
102447799	2001-01	96
102447799	2001-01	540
102447799	2001-01	582
102447799	2001-02	1
102447799	2001-07	17
102447799	2001-07	120
102447799	2001-07	182
102447799	2001-07	141
102447799	2001-08	143
102447799	2001-08	16
102447799	2001-08	188
102447799	2001-09	80
102447799	2001-10	129
102447799	2001-10	190
102447799	2001-10	191
102447799	2001-10	139
102447799	2001-10	623

102447799	2001-11	17
102447799	2001-11	38
102447799	2001-11	58
102447799	2001-11	62
102447799	2001-11	66
102447799	2001-11	138

Tabel 3.4 Tabel transaksi pelanggan 100226412

No. Pelanggan	Tanggal	Item
100226412	2001-01	33
100226412	2001-01	789
100226412	2001-04	14
100226412	2001-06	542
100226412	2001-07	66
100226412	2001-07	69
100226412	2001-07	58
100226412	2001-07	176
100226412	2001-08	35
100226412	2001-08	41
100226412	2001-08	92
100226412	2001-08	168
100226412	2001-08	754
100226412	2001-09	138
100226412	2001-09	60
100226412	2001-09	617
100226412	2001-11	197
100226412	2001-12	86
100226412	2001-12	154

Dari contoh data transaksi tersebut, kita bentuk tabel sequence vertikal. Tabel sequence vertikal dibuat dengan cara mengelompokkan masing – masing

pelanggan yang diberi label sid atau sequence id, kemudian mengelompokkan item yang dibeli pada bulan yang sama yang diberi label itemset, kemudian diberi no ururtan eid atau event id. Contohnya pelanggan 102084120, pada bulan ke 3 membeli item 34, kemudian pada bulan ke 4 membeli item 1, 50, 96, 110. Tabel sequence vertikal dapat dilihat pada Tabel 3.5 berikut.

Tabel 3.5 Tabel sequence vertikal.

sid	eid	itemset
102084120	1	34
102084120	2	1, 50, 96, 110
102084120	3	80, 84
102084120	4	41, 92
102084120	5	35, 36
102084120	6	18, 36, 99
102084120	7	87
102084120	8	38, 112
102084120	9	75, 102
102084120	10	17, 76, 113
105074329	1	60, 80, 438
105074329	2	70, 168, 191
105074329	3	18, 54
105074329	4	23, 61, 623
105074329	5	141, 154, 627, 1
102447799	1	96, 540, 582
102447799	2	1
102447799	3	17, 120, 182, 141
102447799	4	143, 16, 188
102447799	5	80
102447799	6	129, 190, 191, 139, 623
102447799	7	17, 38, 58, 62, 66, 138
100226412	1	33, 789

100226412	2	14
100226412	3	542
100226412	4	66, 69, 58, 176
100226412	5	35, 41, 92, 168, 754
100226412	6	138, 60, 617
100226412	7	197
100226412	8	86, 154

Setelah dibentuk tabel sequence vertikal, kemudian ditentukan min_sup yang akan digunakan untuk menetukan frequent sequences. Dalam perhitungan manual ini digunakan min_sup sebesar 50% atau 0.5. Sehingga itemset dianggap frequent apabila muncul 50% dari keseluruhan pelanggan, dalam manual ini berarti minimal 2 kali. Untuk menentukan 1-sequence, dilakukan scan pada tabel sequence vertikal, kemudian dicari itemset yang muncul lebih banyak dari min_sup. Kemudian dibentuk tabel itemset tersebut lengkap dengan id_listnya, yaitu pasangan sid dan eidnya. Itemset yang memiliki id-list lebih dari min_sup, dalam hal ini 50%, dianggap memenuhi syarat. Apabila suatu itemset muncul lebih dari 1 kali dalam 1 pelanggan (sid), id-listnya tetap dicatat namun supportnya tetap dihitung 1. Contohnya itemset 17, muncul pada sid 102084120 eid 10, dan pada sid 102447799 eid 3 dan 7. Pada id-list dicatat ketiganya,namun itemset ini dianggap hanya memiliki support 2, karena hanya muncul pada 2 pelanggan (sid). Berikut adalah tabel daftar frequent 1-sequence.

Tabel 3.6 Tabel Frequent 1-sequence

1		17		18	
sid	eid	sid	eid	sid	eid
102084120	2	102084120	10	102084120	6
105074329	5	102447799	3	105074329	3
102447799	2	102447799	7		

35		38		41	
sid	eid	sid	eid	sid	eid
102084120	5	102084120	8	102084120	4
100226412	5	102447799	7	100226412	5

58		60		66	
sid	eid	sid	eid	sid	eid
102447799	7	105074329	1	102447799	7
100226412	4	100226412	6	100226412	4

80		92		96	
sid	eid	sid	eid	sid	eid
102084120	3	102084120	4	102084120	2
105074329	1	100226412	5	102447799	1
102447799	5				

138		141		154	
sid	eid	sid	eid	sid	eid
102447799	7	105074329	5	105074329	5
100226412	6	102447799	3	100226412	8

168		191		623	
sid	eid	sid	eid	sid	eid
105074329	2	105074329	2	105074329	4
100226412	5	102447799	6	102447799	6

Dari daftar 1-sequence tersebut, kemudian digunakan untuk membentuk 2-sequence. Caranya adalah melakukan join untuk masing – masing frequent 1-sequence tersebut termasuk dengan dirinya sendiri. Kemudian dicek apakah id-listnya memenuhi syarat min_sup. Contoh proses join adalah sebagai berikut, apabila itemset 1 dijoin dengan itemset 17, maka kemungkinan hasilnya adalah

sequence 1,17 yaitu itemset 1 dan 17 muncul bersamaan, sehingga sequence ini sama dengan 17,1 karena muncul bersamaan, sehingga hanya perlu dicek salah satu saja. Sebaliknya hasil lain dari join itu adalah sequence 1→17 dimana sequence ini berbeda dengan sequence 17→1. Sequence 1→17 menunjukkan itemset 17 muncul setelah itemset 1, sedangkan 17→1 menunjukkan itemset 1 muncul setelah itemset 17. Karena dianggap sequence berbeda, maka keduanya harus dicek apakah id-listnya memenuhi min_sup.

Dari contoh tadi kita cek apakah id-listnya memenuhi min_sup. Sequence 1,17 atau 17,1 tidak memenuhi syarat, karena tidak ada id-listnya dari 1 dengan 17 yang muncul bersamaan sehingga supportnya 0. Sedangkan sequence 1→17 memenuhi syarat karena pada sid 102084120, itemset 1 muncul pada eid 2 dan itemset 17 muncul pada eid 10, sehingga itemset 17 terjadi setelah itemset 1. Kemudian sequence ini juga muncul pada sid 102447799 pada eid 2 kemudian 3 dan eid 2 kemudian 7. Perhatikan walaupun pada sid 102447799 muncul sequence ini 2 kali namun supportnya tetap dihitung 1 karena masih dalam 1 sid. Sehingga sequence 1→17 dianggap frequent karena muncul 2 kali dari total 4 pelanggan atau memiliki support 50%. Sedangkan sequence 17→1 dianggap tidak frequent karena berdasarkan id-list 17 dan 1, tidak ada eid 1 yang terjadi setelah 17.

Berikut adalah daftar lengkap dari frequent 2-sequence setelah dilakukan join pada semua masing – masing frequent 1-sequence.

Tabel 3.7 Tabel Frequent 2-sequence

1 → 17			1 → 38		
sid	eid(1)	eid(17)	sid	eid(1)	eid(38)
102084120	2	10	102084120	2	8
102447799	2	3	102447799	2	7
102447799	2	7			

1 → 80			41, 92		
sid	eid(1)	eid(80)	sid	eid(41)	eid(92)
102084120	2	3	102084120	4	4
102447799	2	5	100226412	5	5

58, 66			60 → 154		
sid	eid(58)	eid(66)	sid	eid(60)	eid(154)
102447799	7	7	105074329	1	5
100226412	4	4	100226412	6	8

80 → 17			80 → 18		
sid	eid(80)	eid(17)	sid	eid(80)	eid(18)
102084120	3	10	102084120	3	6
102447799	5	7	105074329	1	3

80 → 38			80 → 191		
sid	eid(80)	eid(38)	sid	eid(80)	eid(191)
102084120	3	8	105074329	1	2
102447799	5	7	102447799	5	6

80 → 623			96 → 17		
sid	eid(80)	eid(623)	sid	eid(96)	eid(17)
105074329	1	4	102084120	2	10
102447799	5	6	102447799	1	3
			102447799	1	7

96 → 38			96 → 80		
sid	eid(96)	eid(38)	sid	eid(96)	eid(80)
102084120	2	8	102084120	2	3
102447799	1	7	102447799	1	5

168 → 154		
sid	eid(168)	eid(154)
105074329	2	5
100226412	5	8

Langkah berikutnya adalah kita menentukan frequent k-sequence menggunakan join pada semua (k-1) sequence yang memiliki prefix yang sama. Proses join sequence dengan prefiks yang sama adalah sebagai berikut. Misalkan sequence [B → A] dengan set atom [B → AB, B → AD, B → A → A, B → A → D, B → A → F]. Jika kita misalkan P mewakili prefiks B → A, maka kelas tersebut dapat kita tulis ulang untuk mendapatkan [P] = [PB, PD, P → A, P → D, P → F]. Dapat dilihat bahwa kelas ini memiliki 2 jenis atom: atom event [PB, PD], dan atom sequence [P → A, P → D, P → F]. Untuk mengembangkan kelas ini perlu dilakukan join pada semua pasangan atom. Namun tergantung pada pasangan atom yang dijoin, dapat muncul hingga 3 kemungkinan frequent sequence hasil:

1. Atom event dengan atom event: Jika kita menggabungkan PB dengan PD, maka kemungkinan hasilnya hanya atom event baru PBD.
2. Atom event dengan atom sequence: Jika gabungkan PB dengan P → A, maka kemungkinan hasilnya hanya atom sequence baru PB → A.
3. Atom sequence dengan atom sequence: Jika kita menggabungkan P → A dengan P → F, maka ada 3 kemungkinan hasil: atom event baru P → AF, dan 2 atom sequence baru P → A → F dan P → F → A. Kasus khusus muncul apabila kita menggabungkan P → A dengan dirinya sendiri, yang akan menghasilkan hanya atom sequence baru P → A → A.

Dari proses tersebut, misalkan digunakan sequence dari perhitungan manual kita lakukan join pada sequence 96 → 38 dengan sequence 96 → 80. Kemungkinan hasilnya adalah [96 → 38,80], [96 → 38 → 80] dan [96 → 80 → 38]. Dari hasil tersebut yang id-listnya memenuhi syarat min_sup hanya [96 → 80 → 38] yaitu muncul pada sid 102084120 dengan eid 2, 3 dan 8, dan pada sid

102447799 pada eid 1, 5 dan 7. Berikut adalah daftar frequent 3 sequence yang berhasil dibentuk.

Tabel 3.8 Tabel Frequent 3-sequence

96 → 80 → 38			
sid	eid(96)	eid(80)	eid(38)
102084120	2	3	8
102447799	1	5	7

Langkah berikutnya adalah mencari frequent 4-sequence, namun karena hanya ditemukan 1 frequent 3-sequence, maka kemungkinanya hanya sequence tersebut digabungkan dengan sequence itu sendiri. Yang menghasilkan 96 → 80 → 38 → 38, dimana sequence ini tidak memenuhi syarat min_sup. Karena tidak ditemukan frequent sequence lagi dan tidak dapat dibentuk sequence baru lagi maka proses pencarian frequent sequence dihentikan.

Setelah semua frequent sequence ditemukan, kita bentuk rule dari frequent sequence tersebut. Awalnya ditentukan min_conf terlebih dahulu. Pada perhitungan ini kita gunakan min_conf 0.6. Cara menghitung conf dari suatu rule misalkan A => B adalah support dari A → B dibagi dengan support A. Kemudian rule yang diambil adalah yang memenuhi min_conf. Berikut adalah daftar rule dengan confidencenya.

Tabel 3.9 Tabel rule dengan nilai confidence

Rule	Confidence
1 => 17	0.66666667
1 => 38	0.66666667
1 => 80	0.66666667
41 => 92	1
58 => 66	1
60 => 154	1
80 => 17	0.66666667

80 => 18	0.66666667
80 => 38	0.66666667
80 => 191	0.66666667
80 => 623	0.66666667
96 => 17	1
96 => 38	1
96 => 80	1
168 => 154	1
96 → 80 => 38	1

Setelah ditentukan rule yang memenuhi batas min_conf, kemudian kita hitung lift rasio untuk mengukur kekuatan rule yang dibentuk. Untuk menghitung lift dari rule A => B, kita gunakan rumus 2.4. Namun dari rumus tersebut dapat kita turunkan rumusnya hingga didapat untuk menghitung lift dengan membagi confidence dari rule (A => B) dengan support dari B. Nilai lift rasio dikatakan baik apabila lebih dari 1, dan semakin tinggi semakin baik. Berikut adalah daftar rule dengan confidence dan liftnya.

Tabel 3.10 Tabel rule dengan confidence dan lift rasio

Rule	Confidence	Lift Ratio
1 => 17	0.66666667	1.333333
1 => 38	0.66666667	1.333333
1 => 80	0.66666667	0.888889
41 => 92	1	2
58 => 66	1	2
60 => 154	1	2
80 => 17	0.66666667	1.333333
80 => 18	0.66666667	1.333333
80 => 38	0.66666667	1.333333
80 => 191	0.66666667	1.333333
80 => 623	0.66666667	1.333333

96 => 17	1	2
96 => 38	1	2
96 => 80	1	1.333333
168 => 154	1	2
96 → 80 => 38	1	2

3.5 Perancangan Uji Coba

Perancangan uji coba digunakan untuk membuktikan nilai kekuatan dari rule yang dihasilkan oleh algoritma SPADE. Untuk menentukan nilai kekuatan rule yang dihasilkan digunakan nilai lift rasio. Jika nilai lift rasio lebih dari atau sama dengan 1 maka rule sering muncul bersamaan dan antecedent berpengaruh positif pada consequent, begitu pula sebaliknya apabila nilai lift rasio kurang dari 1.

Tabel 3.11 Tabel Rule uji coba

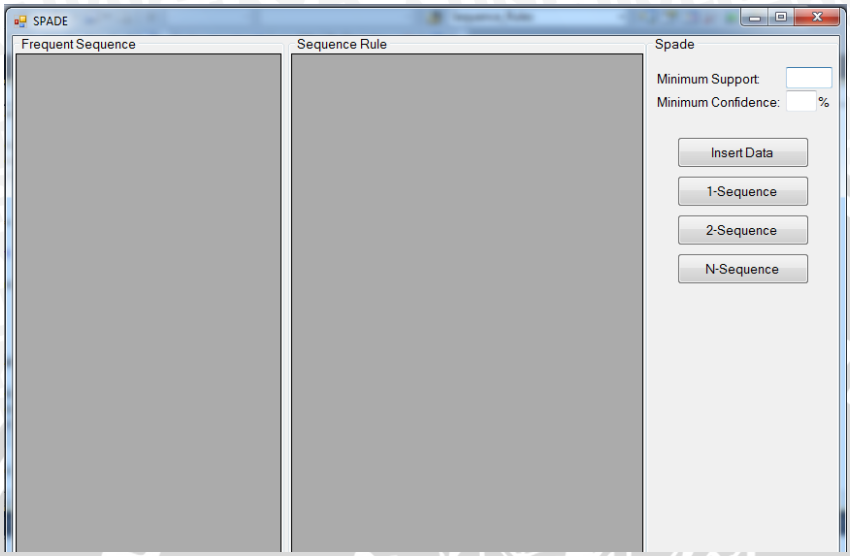
Sequence	Confident	Lift Ratio

Dari tabel rule dapat dilihat rule mana saja yang kurang dari atau lebih dari dan sama dengan 1. Jika terdapat rule yang memiliki lift dari 1, maka dapat ditarik kesimpulan bahwa antecedent pada rule dapat mempengaruhi consequent.

Prose uji coba dilakukan 3 kali dengan parameter yang berbeda untuk mengecek kekuatan rule, uji coba pertama dengan memasukkan nilai parameter minimum support 2, 3 dan 4 dengan nilai minimum confidence dan jumlah data tetap, yaitu confidence 70% dan jumlah data 3000. Uji coba berikutnya dengan mengganti nilai parameter minimum confidence 60%, 70% dan 80% dengan nilai minimum support dan jumlah data tetap, yaitu support 3 dan jumlah data 3000. Uji coba terakhir dengan parameter jumlah data transaksi 2000, 3000 dan 4000 dengan nilai minimum support dan minimum confidence tetap, yaitu support 3 dan confidence 70%.

3.6 Perancangan Antarmuka

Berikut ini adalah rancangan antarmuka yang akan digunakan dalam sistem ini.



Gambar 3.8 Rancangan Antarmuka Sistem.