ANALISIS DATA TRANSAKSI PENJUALAN DENGAN METODE ASSOCIATION RULE MENGGUNAKAN ALGORITMA PINCER SEARCH

SKRIPSI

oleh:

SITRA RIVIANA LATIFA

0410960055-96

PROGRAM STUDI ILMU KOMPUTER

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS BRAWIJAYA

MALANG

2009

UNIVERSITAS BRAWIJAYA

ANALISIS DATA TRANSAKSI PENJUALAN DENGAN METODE *ASSOCIATION RULE* MENGGUNAKAN ALGORITMA *PINCER SEARCH*

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer
dalam bidang Ilmu Komputer

oleh:
SITRA RIVIANA LATIFA
0410960055-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2009

UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

ANALISIS DATA TRANSAKSI PENJUALAN DENGAN METODE *ASSOCIATION RULE* MENGGUNAKAN ALGORITMA *PINCER SEARCH*

oleh:
SITRA RIVIANA LATIFA
0410960055-96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 16 Oktober 2009
dan dinyatakan memenuhi syarat untuk memperoleh
gelar Sarjana Komputer dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Wayan Firdaus M., SSi., MT	**Agus Wahyu Widodo, ST**
NIP. 197209191997021001	**NIP. 197408052001121001**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Agus Suryanto, MSc
NIP. 196908071994121001

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama **:** Sitra Riviana Latifa
NIM **:** 0410960055-96
Jurusan **:** Matematika
Program Studi **:** Ilmu Komputer
Penulis skripsi berjudul **:** Analisis Data Transaksi Penjualan dengan Metode *Association Rule* Menggunakan Algortima *Pincer Search*

Dengan ini menyatakan bahwa :

1. Isi dari skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam skripsi ini.
2. Apabila dikemudian hari ternyata skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 16 Oktober 2009
Yang menyatakan,

(Sitra Riviana Latifa)
NIM. 0410960055

ANALISIS DATA TRANSAKSI PENJUALAN DENGAN METODE *ASSOCIATION RULE* MENGGUNAKAN ALGORITMA *PINCER SEARCH*

ABSTRAK

Association rule digunakan untuk menemukan keterkaitan antar *item* dari sekumpulan data yang besar. Biasanya digunakan dalam menganalisis data transaksi penjualan untuk menemukan pola-pola dan perilaku konsumen dalam melakukan suatu transaksi. *Association rule* terdiri dari dua proses, yaitu dengan menemukan semua *frequent itemset* dan dilanjutkan dengan membangkitkan aturan asosiasi yang kuat dari *frequent itemset* tersebut. Untuk menemukan *frequent itemset* menggunakan algoritma *Pincer Search*. Algoritma ini menggunakan dua arah pencarian sekaligus untuk menemukan *frequent itemset*. Dengan pencarian dua arah, maka dapat mengurangi jumlah kandidat yang akan dicari sehingga seluruh *frequent itemset* akan lebih cepat ditemukan.

Tujuan dari penelitian ini adalah menerapkan algoritma *Pincer Search* pada data transaksi penjualan. Perancangan pengujian sistem dilakukan untuk membandingkan kecepatan proses dan jumlah L_k dan C_k setiap iterasi antara agoritma *Pincer Search* dengan algoritma *Apriori*. Parameter yang digunakan dalam pengujian sistem adalah periode transaksi, *minimum support* dan *minimum confidence*. Pengujian ini bertujuan untuk menganalisis apakah ada hubungan antara kecepatan proses yang dihasilkan antara algoritma *Pincer Search* dan algoritma *Apriori* dengan memberikan masukan *minimum support* dan *confidence* yang berbeda-beda.

Hasil pengujian menunjukkan bahwa kecepatan proses algoritma *Pincer Search* lebih cepat dibandingkan dengan algoritma *Apriori*. Pada minimum support yang kecil, jumlah iterasi pada algoritma *Pincer Search* lebih pendek dari algoritma *Apriori*. Besarnya *minimum support* dan *minimum confidence* berbanding terbalik dengan banyaknya aturan yang dihasilkan.


UNIVERSITAS BRAWIJAYA

ANALYSIS DATA SALES TRANSACTION WITH ASSOCIATION RULE METHOD USE PINCER SEARCH ALGORITHM

ABSTRACT

Rule Association is used to find relationship between items of group of big data. It can be used in analyzing sales transaction data to find patterns and behavior of consumer in doing a transaction. Association rule consist of two step of process, that are finding all frequent itemset and continued by awakening strong association from that of frequent itemset itself. To find frequent itemset it is use algorithm of Pincer Search. This Algorithm is using two seeking direction at the same time to find the frequent itemset. With seeking two directions, it can lessen the amount of candidate to look for, so that all frequent itemset will be more quickly to be found.

The objectives of this research is to apply Pincer Search algorithm at sales transaction data. Scheme of examination system is done to compare the speed process and amount of L_k and C_k each iteration among Pincer Search algorithm with Apriori algorithm. Parameter which is used in system examination are period of transaction, minimum support and minimum confidence. This examination aimed to analyze is that any relation between speed of yielded process among algorithm of Pincer Search and algorithm Apriori by giving minimum support and the different confidence.

The results of examination indicate that, the speed of process algorithm of Pincer Search is quicker when it is compared with algorithm Apriori. At small support minimum, the amount of iteration at algorithm of Pincer Search is shorter than algorithm Apriori. The level of minimum support and minimum confidence inversely proportional with many rules that yielded order

KATA PENGANTAR

Puji syukur Penulis panjatkan kepada Allah SWT atas segala limpahan rahmat dan hidayah-Nya, sehingga skripsi yang berjudul "Analisis Data Transaksi Penjualan dengan Metode *Association Rule* Menggunakan Algoritma *Pincer Search*" ini dapat diselesaikan. Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Ilmu Komputer, Jurusan Matematika, Fakultas MIPA, Universitas Brawijaya.

Pada penyusunan skripsi ini, Penulis ingin mengucapkan terima kasih kepada :

1. Wayan Firdaus Mahmudi, SSi, MT, selaku pembimbing utama atas arahan serta bimbingannya dalam penyusunan skripsi ini.
2. Agus Wahyu Widodo, ST, selaku pembimbing pendamping atas arahan serta bimbingannya dalam penyusunan skripsi ini.
3. Dr. Agus Suryanto, MSc, selaku Ketua Jurusan Matematika Fakultas MIPA Universitas Brawijaya Malang.
4. Drs. Marji MT, selaku Ketua Program Studi Ilmu Komputer, Jurusan Matematika, Fakultas MIPA Universitas Brawijaya.
5. Candra Dewi, S.Kom, MSc, selaku pembimbing akademik.
6. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada Penulis selama menempuh pendidikan di Program Studi Ilmu Komputer Jurusan Matematika FMIPA Universitas Brawijaya.
7. Segenap staf dan karyawan di Jurusan Matematika FMIPA Universitas Brawijaya yang telah banyak membantu Penulis dalam pelaksanaan penyusunan skripsi ini.
8. Orang tua dan saudara penulis atas dukungan materi dan doa restunya kepada penulis.
9. Hyuamail, stripebluefish, yyuta, indiego_13, shevphila, mas boy thanks for your helpful, supports, advice for me. Enkyu…

10. Eltharine, thanks for being my best friend in the last 5 years and being my 'teman seperjuangan' in finishing my thesis. Thanks for take me and turn me back in my home safely.
11. BIP, atas bantuannya mendapatkan tanda tangan.
12. Rekan-rekan di Program Studi Ilmu Komputer FMIPA Universitas Brawijaya yang telah banyak memberikan bantuannya demi kelancaran pelaksanaan penyusunan skripsi ini.
13. Dan semua pihak yang telah membantu dalam penyusunan skripsi ini yang tidak dapat penulis sebutkan satu per satu.

Penulis sadari bahwa masih banyak kekurangan dalam penyusunan skripsi ini, oleh karena itu Penulis sangat menghargai saran dan kritik yang sifatnya membangun demi perbaikan penulisan dan mutu isi skripsi ini untuk kelanjutan penelitian serupa di masa mendatang. Penulis berharap skripsi ini dapat bermanfaat bagi saya sebagai penulis dan kepada para pembaca skripsi ini.

Malang, Oktober 2009

Penulis

DAFTAR ISI

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

BAB I
PENDAHULUAN

1.1. Latar Belakang

Association rule merupakan salah satu proses *data mining* yang mampu menemukan pola dan aturan (*rule*) untuk menemukan asosiasi, korelasi, dan struktur kausal antar item dari sekumpulan data yang besar. Hal-hal tersebut dapat merepresentasikan informasi penting yang ingin diketahui pada data yang diberikan. Penggalian *association rule* ini biasanya sering digunakan dalam menganalisis data transaksi untuk menemukan pola-pola dan perilaku konsumen dalam melakukan suatu transaksi dimana pola-pola ini merupakan kumpulan item (*itemset*) yang sering dibeli oleh konsumen.

Association rule terdiri dari dua langkah proses, yaitu dengan menemukan semua *frequent itemset* dan dilanjutkan dengan membangkitkan aturan asosiasi yang kuat dari *frequent itemset* tersebut (Han, 2001). Proses pencarian item-item yang sering dibeli atau sering muncul dalam transaksi (*frequent itemset*) pada data yang besar membutuhkan waktu yang sangat lama, dan hal ini merupakan syarat untuk membentuk *association rule*. Dari kondisi tersebut, diperlukan suatu algoritma yang bisa meminimalisasi pembacaan basis data, sehingga bisa mengoptimasi waktu yang dibutuhkan.

Ada beberapa algoritma yang digunakan untuk penggalian aturan asosiasi. Algoritma yang digunakan dalam penelitian ini adalah algoritma *Pincer Search*. Algoritma *Pincer Search* adalah suatu algoritma yang menggabungkan pencarian dari dua arah yaitu, *top-down* dan *bottom-up* (Lin, 1998). Arah pencarian utamanya sama dengan algoritma *Apriori* yaitu dengan pendekatan b*ottom-up*, bedanya pada *Pincer Search* secara bersamaan dilakukan pencarian secara t*op-down*, yang menghasilkan *output* berupa *Maximum Frequent Set (MFS)*. *Maximum Frequent Set* adalah sekumpulan *maximal itemset - itemset* yang tergolong *frequent*.

Dengan menggunakan dua arah pencarian sekaligus, maka dapat mengurangi jumlah kandidat yang akan dicari sehingga *Maximal Frequent Itemset* akan lebih cepat ditemukan.

Berdasarkan latar belakang yang telah dipaparkan, maka judul yang diambil dalam Tugas Akhir ini adalah "**Analisis**

Data Transaksi Penjualan dengan Metode *Association Rule* menggunakan Algoritma Pincer Search".

1.2. Rumusan Masalah

Berdasarkan latar belakang di atas maka rumusan masalah dalam penelitian ini adalah bagaimana menerapkan metode *Association Rule* dengan algoritma *Pincer Search* pada data transaksi penjualan.

1.3. Batasan Masalah

Agar dalam pemecahan masalah tidak menyimpang dari tujuan semula dan menghindari kemungkinan meluasnya pembahasan, perlu kiranya dilakukan batasan-batasan permasalahan sebagai berikut:

1. Algoritma yang diimplementasikan adalah algoritma *Apriori* dan algoritma *Pincer Search*.
2. Data set yang digunakan adalah transaksi penjualan untuk mengetahui pola keterkaitan *item* barang yang dibeli oleh konsumen dan tidak membahas jumlah stok setiap *item*.
3. Input berisi data transaksi penjualan yang didalamnya terdapat kode transaksi dan kode barang. Output yang dihasilkan adalah *rules-rules* yang didapat dari proses *association rule*, yang ditampilkan dalam bentuk tabel.

1.4. Tujuan Penelitian

Tujuan penelitian yang ingin dicapai dalam penulisan skripsi ini adalah untuk menerapkan algoritma *Pincer Search* pada data transaksi penjualan.

1.5. Manfaat Penelitian

Manfaat dari penelitian ini diharapkan dapat menghasilkan sebuah sistem yang dapat menghasilkan informasi tentang pola asosiasi antar *item* yang berbeda, dimana nantinya dapat diperoleh keuntungan dari pola tersebut.

1.6. Sistematika Penulisan

Tugas akhir ini disusun berdasarkan sistematika penulisan sebagai berikut:

1. BAB I PENDAHULUAN
   Berisi latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.
2. BAB II TINJAUAN PUSTAKA
   Menguraikan tentang pengertian *Data Mining*, pengertian *Market Basket Analysis*, pengertian *Association* Rule, algoritma A*priori*, algoritma *Pincer Search*.
3. BAB III METODE PENELITIAN
   Pada bab ini akan dijelaskan mengenai metode-metode yang digunakan dan langkah-langkah yang harus dilakukan dalam penelitian ini.
4. BAB IV HASIL DAN PEMBAHASAN
   Pada bab ini akan dijelaskan mengenai proses penerapan sistem, pengujian, dan evaluasi sistem perangkat lunak yang dibangun.
5. BAB V KESIMPULAN DAN SARAN
   Berisi kesimpulan dari seluruh rangkaian penelitian dan saran yang diharapkan bermanfaat untuk pengem bangan tugas akhir ini selanjutnya.


UNIVERSITAS BRAWIJAYA

BAB II
TINJAUAN PUSTAKA

2.1 Data Mining

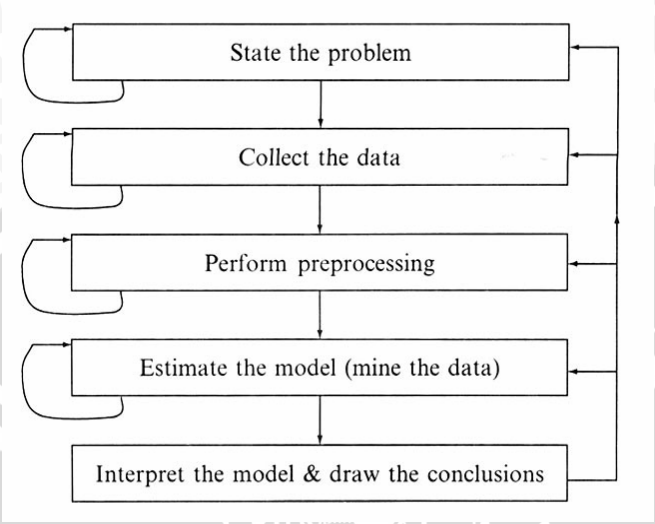
Menurut Gartner Group, *data mining* adalah proses untuk menemukan korelasi baru, pola dan trend dengan seleksi pada sejumlah data yang besar dengan menggunakan teknologi pengenalan pola, statistik dan matematika (Larose, 2005).

Sedangkan menurut Edelstein, *data mining* adalah suatu proses analisa data untuk menemukan pola dan hubungan di dalam data yang kemungkinan dapat digunakan untuk melakukan suatu prediksi yang valid (Edelstein, 1999).

Dalam prakteknya, terdapat dua sasaran utama data mining (Kantardzic, 2003), yaitu prediksi dan deskripsi.

- Prediksi. Melibatkan beberapa variabel data untuk meramalkan nilai yang akan datang dan bertujuan untuk menghasilkan suatu model yang dapat digunakan untuk klasifikasi, prediksi, estimasi dll.
- Deskripsi. Terpusat untuk menemukan suatu pola yang dapat ditafsirkan oleh manusia dan bertujuan untuk memperoleh suatu pemahaman sistem yang dianalisa dengan cara pencarian pola dan relasi pada data yang besar.

Kantardzic (2003) menggambarkan proses *data mining* seperti pada Gambar 2.1. Dalam proses tersebut terdapat beberapa prosedur umum eksperimental dengan menggunakan *data mining*. Yang pertama ialah merumuskan permasalahan, dimana pada tahap ini ditetapkan sebuah rumusan masalah serta variabel-variabel yang terlibat. Prosedur yang kedua ialah mengumpulkan data. Pada prosedur ini, konsentrasi ditujukan mengenai proses pembuatan atau pengumpulan data. Prosedur yang ketiga yaitu *preprocessing data*, yaitu prosedur untuk menyeleksi data yang akan digunakan dalam proses. Prosedur selanjutnya ialah estimasi model yang dapat disebut sebagai proses utama pada prosedur ini, sebab implementasi dari teknik *data mining* dilakukan pada prosedur ini. Dan yang terakhir ialah menafsirkan informasi yang dihasilkan dari proses sebelumnya.



Gambar 2.1 Proses data mining

Ada beberapa teknik yang dimiliki *data mining*. Menurut Larose (2005), teknik *data mining* yang utama ialah *description, estimation, prediction, classification, clustering,* dan *association rule*. Dari seluruh teknik yang disebutkan, dalam penelitian ini hanya akan memberikan uraian mengenai *association rule* yang akan dijelaskan pada sub bab selanjutnya.

2.2 Association Rule

Association rule adalah suatu prosedur untuk mencari hubungan antar *item* dalam suatu data set yang ditentukan (Han, Kamber, 2001).

Association rule ini meliputi dua tahap (Kantardzic, 2003), yaitu :

1. Mencari kombinasi yang paling sering terjadi dari suatu *itemset*
2. Meng-*generate association rule* dari *frequent itemset* yang telah dibuat sebelumnya

Dalam menentukan suatu *association rule*, terdapat suatu ukuran kepercayaan yang didapatkan dari hasil pengolahan data dengan perhitungan tertentu. Umumnya ada dua ukuran, yaitu *support* dan *confidence* (Han dan Kamber, 2001). *Support* adalah suatu ukuran yang menunjukkan seberapa besar tingkat dominasi suatu *item/itemset* dari keseluruhan transaksi. Sedangkan *confidence* adalah suatu ukuran yang menunjukkan hubungan antar 2 atau lebih

item secara *conditional*. Secara sederhana perhitungan untuk mendapatkan *support* dan *confidence* dapat dijelaskan sebagai berikut

- *support* ($A \Rightarrow B$) = jumlah transaksi yang berisi A dan B (2.1)

- *confidence* ($A \Rightarrow B$) $= \frac{P(A \cap B)}{P(A)}$

$$= \frac{\text{jumlah transaksi yang terdiri A dan B}}{\text{jumlah transaksi A}} \quad (2.2)$$

Dengan menggunakan Persamaan 2.1 dan 2.2 dapat menghasilkan nilai *support* dan *confidence* dimana kedua ukuran ini nantinya berguna dalam menentukan *interesting rules*, yaitu untuk dibandingkan dengan batasan (*threshold*) yang ditentukan oleh *user*. Batasan tersebut umumnya terdiri dari *minimum support* dan *minimum confidence*. Jika nilai *support*-nya ≥ *min_support* dan *confidence*-nya ≥ *min_confidence*, maka *rule* tersebut bisa dikatakan sebagai *interesting rule*. *Minimum support* adalah parameter yang digunakan sebagai batasan *item/itemset* yang harus dipenuhi dalam suatu kelompok data untuk dapat dijadikan aturan, sedangkan *minimum confidence* adalah parameter yang mendefinisikan *minimum level* dari *confidence* (kepercayaan) yang harus dipenuhi oleh aturan yang berkualitas.

Sebagai contoh, dari suatu himpunan data transaksi, seseorang mungkin menemukan suatu hubungan berikut, yaitu jika konsumen yang biasanya membeli roti tawar juga membeli keju dalam satu transaksi yang sama ditunjukkan sebagai berikut :

Roti tawar → keju [*Support* 2%, *confidence* 60%]

Yang artinya : "60% dari transaksi di *database* yang memuat *item* roti tawar juga memuat *item* keju. Sedangkan 2% dari seluruh transaksi yang ada di di database memuat kedua item itu".

Support dan *confidence* dituliskan dengan nilai antara 0%-100%. Sebuah set dari *item* disebut juga *itemset*. Sebuah *itemset* yang mengandung k *item* adalah k-*itemset*. Set {roti tawar, keju} adalah 2-*itemset*. Jumlah kejadian munculnya *itemset* adalah jumlah transaksi yang mengandung *itemset* tersebut. Jika suatu *itemset* dimana *support*-nya lebih besar atau sama dengan *min_support* yang merupakan *threshold* yang diberikan oleh *user*, maka *itemset* tersebut disebut *juga frequent itemset*. Sebaliknya, jika *support*-nya kurang

dari *min_support* yang telah ditentukan oleh *user* maka disebut *infrequent itemset*.

2.3 Market Basket Analysis

Menurut Han Jiawei (2001), fungsi *Association Rules* seringkali disebut dengan "*Market Basket Analysis"*. *Market basket analysis* adalah salah satu cara yang digunakan untuk menganalisis data penjualan dari suatu perusahaan. Proses ini menganalisis kebiasaan belanja konsumen dengan menemukan asosiasi antar *item-item* yang berbeda yang diletakkan konsumen dalam keranjang belanja. Hasil yang telah didapatkan ini nantinya dapat dimanfaatkan oleh perusahaan retail seperti toko atau swalayan untuk mengembangkan strategi pemasaran dengan melihat *item-item* mana saja yang sering dibeli secara bersamaan oleh konsumen.

Association rule yang didefinisikan pada basket data, digunakan untuk keperluan promosi, desain katalog, segmentasi konsumen dan target pemasaran. Secara tradisional, *association rule* digunakan untuk menemukan trend bisnis dengan menganalisa transaksi konsumen.

2.4 *Bottom-up* dan *Top-down*

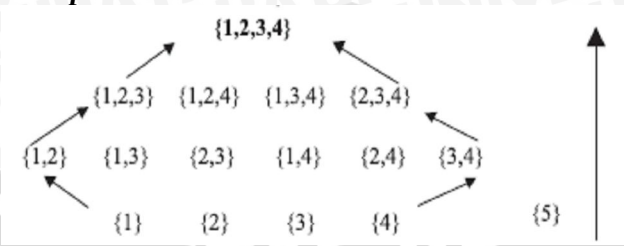
Pada umumnya untuk menemukan *frequent itemset* yang merupakan tahap awal dari *association rule,* dapat dilakukan dengan dua cara pendekatan, yaitu *bottom-up* dan *top-down search* [Lin, 1998].

Tabel 2.1 Tabel Contoh Data

Transaksi	Itemset
100	1,2,3,4,5
200	1,3
300	1,2
400	1,2,3,4

Berdasarkan pada Table 2.1 akan diberikan gambaran mengenai cara kerja *Bottom-Up* dan *Top-Bottom* dengan nilai *min_support*=2.

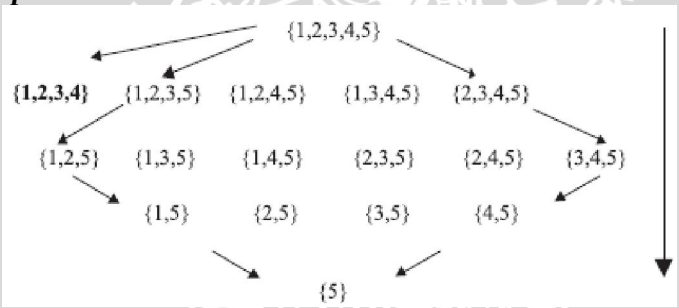
Bottom-Up Search



Gambar 2.2 *Bottom-Up Search*

Metode *bottom-up* dimulai dari 1-*itemset* sampai dengan ke n-*itemset*. Metode ini akan menemukan *itemset* yang memenuhi *min_support* dan tidak meneruskan *itemset* yang tidak memenuhi *min_support*. Pada transaksi tersebut telah ditemukan *frequent itemset*nya adalah : {1}, {2}, {3}, {4}, {1,2}, {1,3}, {1,4}, {2,3}, {2,4}, {3,4}, {1,2,3}, {1,2,4}, {1,3,4}, {2,3,4}, {1,2,3,4}. Metode ini efektif apabila maksimal *itemset*nya relatif pendek.

Top-Down Search



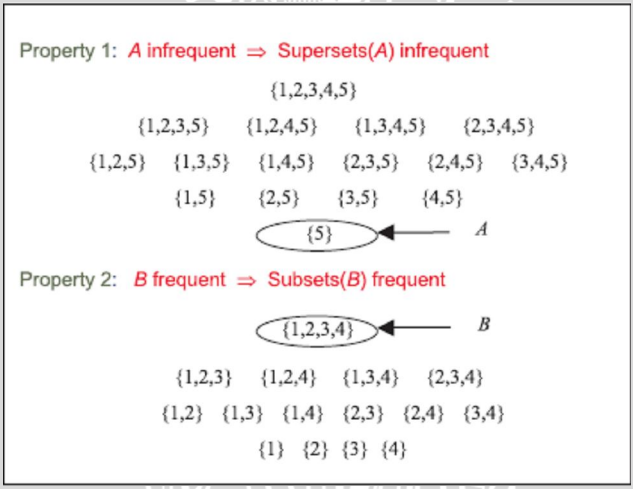
Gambar 2.3 *Top-Down Search*

Metode *top-down* ini melakukan pencarian mulai dari n-*itemset* sampai ke 1-*itemset*. Ketika k-itemset ditemukan sebagai *infrequent*, maka semua *subset* (k-1) akan di-*generate* pada langkah selanjutnya. Sebaliknya, jika *k-itemset frequen*t, maka semua *subset* ini adalah *frequent* dan tidak perlu di-*generate* lagi sesuai dengan *property* 2. Kebalikan dari metode *bottom-up*, metode ini efektif dalam pencarian dari suatu maksimal *itemset* yang panjang. Dengan metode ini dapat menuruni banyak level hanya dengan sekali jalan. Baik *bottom-up* maupun *top-down search* merupakan metode *one-way search*.

2.5 *Downward Closure Property* dan *Upward Closure Property*

Kedua properti ini digunakan untuk melakukan *pruning candidate set* sehingga dapat mengurangi waktu penghitungan. Pendekatan untuk melakukan *prune* dengan cara seperti ini adalah sebagai berikut [Lin, 1998]:

1. *Upward Closure Property* : Jika semua *infrequent itemset* ditemukan di *Bottom Up*, hasilnya dapat digunakan untuk mengeliminasi kandidat yang ditemukan pada arah *Top Down* yang telah ditemukan sejauh ini. Semua *superset* dari *itemset* ini pastilah *infrequent*.
2. *Downward Closure Property* : Jika semua *Maximal Frequent Itemset* ditemukan pada arah *Top Down*, maka *itemset* tersebut dapat digunakan untuk mengeliminasi banyak kandidat dari arah *Bottom Up*. Semua *subset* dari *itemset* ini pastilah *frequent*.



Gambar 2.4 Dua Properti Closure

Berdasarkan pada Tabel 2.1 dan Gambar 2.4 dapat dijelaskan sebagai berikut. Jika *itemset* {5} *infrequent*, maka *itemset* {2,5} juga *infrequent* karena nilai *support itemset* {2,5} sama dengan atau kurang dari *support itemset* {5}. Dan sebaliknya, jika *itemset* {1,2,3,4} *frequent*, maka *itemset* {1,2,3} juga *frequent*

2.6 Algoritma Apriori

Algoritma *apriori* [Agrawal, 1994] merupakan tipikal dari pendekatan *bottom up*. Algoritma ini menggunakan pendekatan iteratif, yang dikenal dengan *level-wise search*, dimana k-kelompok produk digunakan untuk mengeksplorasi (k+1)-kelompok produk atau (k+1)-*itemset*. Algoritma ini berdasarkan pada prinsip *Apriori*, yaitu jika suatu *itemset* merupakan *frequent itemset*, maka semua *subset*-nya akan berupa *frequent itemset*. Pembentukan *frequent itemset* dilakukan dengan mencari semua kombinasi item-item yang memiliki *support* di atas *min_support* yang telah ditentukan.

Proses pada algoritma ini membangkitkan *frequent itemset* per level, dimulai dari level 1-*itemset* sampai ke *longest itemset*, kandidat level yang baru dibentuk dari *frequent itemset* yang ditemukan di level sebelumnya lalu menentukan nilai *support*-nya.

Dibawah ini merupakan prosedur-prosedur yang digunakan dalam algoritma *Apriori*:

1. Algoritma Apriori-gen

1) Memanggil prosedur *join* untuk *generate* kandidat set
2) Memanggil prosedur *prune* untuk mendapatkan hasil akhir kandidat set

2. Join Prosedur

1) **For** i from 1 to $|L_k - 1|$
2) **For** j from i+1 to $|L_k|$
3) **if** L_k.itemset$_i$ and L_k.itemset$_j$ have the same (k-1)-prefix
4) $C_{k+1} := C_{k+1} \cup (L_k.\text{itemset}_i \cup L_k.\text{itemset}_j)$
5) **else**
6) **break**

Pada langkah ini dilakukan kombinasi dua *frequent k-itemset* dimana mempunyai *(k-1)prefix* yang sama untuk meng-*generate (k+1)-itemset* sebagai kandidat yang baru

3. Prune Prosedur

1) **For all** itemsets c in C_{k+1}
2) **For all** k-subsets s of c
3) **If** $s \notin L_k$
4) **Delete** c from C_{k+1}

Pada langkah ini membuang *itemset* c dalam C_{k+1} yang beberapa *subset* ukuran *k* yang tidak berada dalam L_k. Dengan kata lain, *superset* dari *infrequent* itemset dihilangkan.

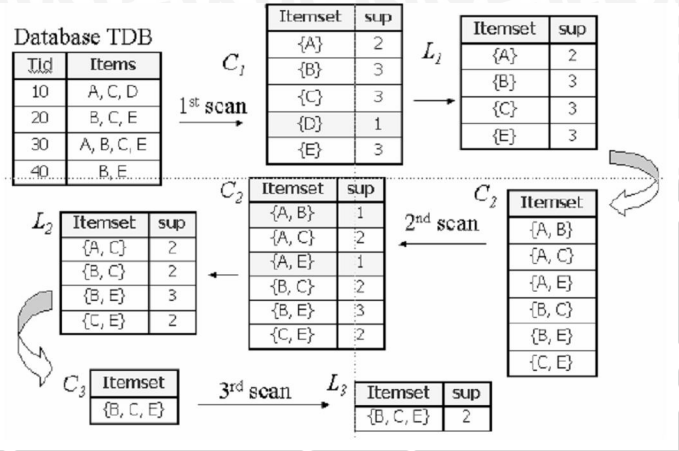
4. Algoritma Apriori

C_k : kandidat *itemset* dengan ukuran *k*
L_k : *frequent itemset* dengan ukuran *k*
D : data seluruh transaksi

1) $L_0 := \varnothing$; k:= 1;
2) $C_1 := \{\{i\} \mid i \in I\}$
3) Answer := $\varnothing$
4) while $C_k \neq \varnothing$
5) read database and count supports for C_k
6) L_k := {frequent itemsets in C_k}
7) C_{k+1} := Apriori-gen(L_k)
8) k := k+1
9) Answer := Answer $\cup$ L_k
10) return Answer

Berdasarkan algoritma di atas, yang dilakukan pertama kali yaitu menelusuri seluruh *record* di basis data transaksi dan menghitung *support* dari tiap *item*, yang merupakan kandidat *1-itemset*, C_1. Setelah itu L_1 dibangun dengan menyaring C_1 dengan *support count* yang lebih besar sama dengan *min_support* untuk dimasukkan ke dalam L_1. Untuk membangun L_2, algoritma *apriori* menggunakan prosedur *join* untuk menghasilkan C_2. Dari C_2, 2-*itemset* yang memiliki *support* yang lebih besar sama dengan *min_support* akan disimpan ke dalam L_2. Proses ini diulang sampai tidak ada lagi kemungkinan *k-itemset*.

Contoh tahapan pembangkitan C_1, L_1, C_2, L_2, C_3, L_3 ditunjukkan pada Gambar 2.5.



Gambar 2.5 Ilustrasi Algoritma Apriori

Kelemahan dari algoritma *apriori* adalah proses *generate* kandidat yang sangat besar sekali sehingga membutuhkan waktu komputasi yang lama. Misal untuk ukuran 100 item data $\{a_1,a_2\ldots,a_{100}\}$, perlu dibangun $2^{100} \approx 10^{30}$ kandidat.

2.7 Algoritma Pincer Search

Algoritma *Pincer-Search* dikenal juga sebagai algoritma T*wo-Way Search* (Lin, 1997). Algoritma ini dinamakan *Two-Way Search* karena memakai 2 cara pendekatan, yaitu *Top Down* serta *Bottom Up*. Dalam prosesnya, arah pencarian utama dari *Pincer search* adalah *Bottom Up*, seperti Algoritma *Apriori* untuk menemukan *Frequent Itemset*, bedanya pada *Pincer search* secara bersamaan dilakukan pencarian secara *Top Down*, yaitu *Maximum Frequent Candidat Set* (MFCS) yang menghasilkan output berupa *Maximum Frequent Set (MFS)*. *Maximum Frequent Candidat Set* (MFCS) adalah itemset yang subset-subsetnya belum diketahui frequent atau infrequent. Sedangkan *Maximum Frequent Set* adalah sekumpulan *maximal itemset-itemset* yang tergolong *frequent*. Guna dari *Maximum Frequent Set* adalah untuk mengurangi *(prunning)* jumlah kandidat *Frequent Itemset* yang perlu diperiksa secara *Bottom Up*.

Selain menggabungkan 2 pendekatan *Bottom Up* dan *Top Down*, *Pincer Search* juga menggunakan dua properti khusus, yakni *Downward Closure Property* dan *Upward Closure Property* yang telah dijelaskan pada subbab 2.5. Dengan cara pendekatan ini dapat

meningkatkan pencarian MFS. Selain meng*generate* 1-*itemset* menggunakan algoritma *apriori,* tetapi juga menggunakan top down untuk melakukan *prune item-item frequent* sebagai kandidat-kandidat untuk digenerasikan selanjutnya dalam setiap tahapannya dengan menggunakan bantuan MFCS. Penghitungan *support* tidak hanya dilakukan pada *bottom up* saja, tetapi juga secara *top down*, yaitu menghitung *support* dari MFCS.

Dibawah ini adalah prosedur-prosedur yang digunakan dalam algoritma *Pincer Search* [Lin, 1998] :

1. Join Procedure

Menggunakan prosedur join milik *apriori*

2. Algoritma MFCS gen

Input: Old MFCS and the infrequent set S_k found in pass k
Output: New MFCS

1. **for** all itemsets s $\in$ S_k
2. **for** all itemsets m $\in$ MFCS
3. **if** s is a subset of m
4. MFCS := MFCS \ {m}
5. **for** all items e $\in$ itemset s
6. **if** m \ {e} is not a subset of any itemset in the MFCS
7. MFCS := MFCS $\cup$ {m \ {e}}
8. **return** MFCS

Misalkan {1,2,3,4,5,6} merupakan MFCS yang lama, dan 2 *itemset* baru yang *infrequent* ditemukan {1,6},{3,6}. Maka pertama-tama s yang digunakan untuk melakukan eliminasi adalah {1,6}. Jika {1,2,3,4,5,6} mengandung {1,6} MFCS akan ter-*update* menjadi {2,3,4,5,6} dan {1,2,3,4,5}. Kemudian menuju ke s berikutnya yaitu {3,6}. {2,3,4,5,6} mengandung {3,6}, maka akan dipecah menjadi {2,4,5,6} dan {2,3,4,5}. Karena {2,3,4,5} merupakan *subset* dari {1,2,3,4,5} maka tidak akan digabung menjadi MFSC. Jadi MFCS saat ini adalah {1,2,3,4,5} dan {2,4,5,6}.

3. Recovery Procedure

Input: C_{k+1} from join procedure, L_k, and current MFS
Output: a complete candidate set C_{k+1}
1. **for** all itemsets l in L_k
2. **for** all itemsets m in MFS
3. **if** the first k-1 items in l are also in m
4. /* suppose $m.item_j = l.item_{k-1}$ */
5. **for** i from j+1 to |m|
6. $C_{k+1} := C_{k+1} \cup \{\{l.item1, l.item2,..., l.item_k, m.item_i\}\}$

Misalnya pada *pass* berikutnya table L_3 ditemukan {{1,2,3}, {1,2,4}, {1,2,5}, {1,3,4}, {1,3,5}, {1,4,5}, {2,3,4}, {2,3,5}, {2,4,5}, {2,4,6}, {2,5,6}, {3,4,5}, {4,5,6}}dan dari MFCS {{1,2,3,4,5}, {2,4,5,6}} ditemukan {1,2,3,4,5} sebagai MFS, maka semua *subset* MFS dalam L_3 akan dihilangkan sehingga table L_3 hanya akan berisi {{2,4,6}, {2,5,6}, {4,5,6}. Saat L kehilangan banyak anggotanya karena eliminasi dari MFS, maka *recovery procedure* akan berguna untuk mengembalikan kemungkinan kandidat C_4 yang hilang. Dalam hal ini ditemukan bahwa {2,4,5} merupakan *subset* dari MFS yang memiliki k-1 *prefix* sama dengan {2,4,6}. Sehingga dikembalikan anggota L menjadi {2,4,5},{2,4,6},{2,5,6},{4,5,6}. Dan C_4 yang terbentuk setelah *join* adalah {2,4,5,6}.

4. New Prune Procedure

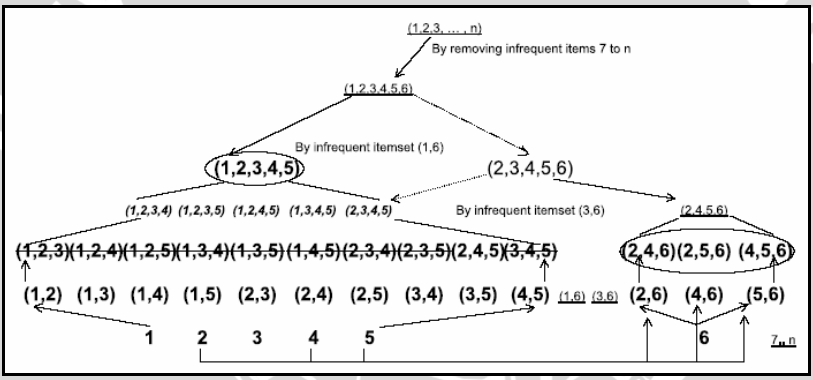
Input: current MFCS and C_{k+1} after join and recovery procedures
Output: final candidate set C_{k+1}
1. **for** all itemsets c in C_{k+1}
2. **if** c is not a subset of any itemset in the current MFCS
3. **delete** c from C_{k+1}

5. Pincer Search Algoritma

Input: a database and a user-defined minimum support
Output: MFS which contains all maximal frequent itemsets
1. $L_0 := \varnothing$; k := 1; $C_1 := \{\{i\} \mid i \in I\}$
2. MFCS := {{1, 2,…,n}}; MFS := $\varnothing$
3. **while** $C_k \neq \varnothing$
4. read database and count supports for C_k and MFCS
5. remove frequent itemsets from MFCS and add them to MFS
6. L_k := {frequent itemsets in C_k} \ {subsets of MFS}

7. S_k := {infrequent itemsets in C_k}
8. call the *MFCS-gen* algorithm if $S_k \neq \varnothing$
9. call the *join* procedure to generate C_{k+1}
10. if any frequent itemset in C_k is removed in line 6
11. call *recovery* procedure to recover candidates to C_{k+1}
12. call new *prune* procedure to prune candidates in C_{k+1}
13. k := k + 1
14. **end-while**
15. **return** MFS

Gambar 2.6 adalah gambaran cara kerja dari Algoritma *Pincer Search*:



Gambar 2.6 Ilustrasi Algoritma *Pincer Search*

Keterangan gambar:

- Tulisan bercetak tebal adalah kandidat yang *frequent*
- Tulisan bergaris bawah adalah kandidat yang *infrequent*
- Tulisan bercetak miring adalah kandidat yang dihilangkan (*prune*) dengan MFCS
- Tulisan dengan coretan ditengahnya mewakili kandidat yang *frequent* tapi tidak perlu digunakan untuk membuat (*generate*) kandidat *itemset* yang baru.
- *Itemset* dengan garis yang melingkarinya mewakili *Maximum Frequent Itemset*

BAB III
METODE PENELITIAN

Pada bab ini akan dibahas tentang metode dan tahap perancangan dalam analisis data transaksi penjualan dengan metode *Association Rule* menggunakan algoritma *Pincer Search*.

Adapun tahapan pembuatannya adalah sebagai berikut:

1. Melakukan studi literatur mengenai algoritma *Apriori* dan algoritma *Pincer Search* dalam permasalahan *Association Rule.*
2. Menganalisa sistem dan melakukan perancangan sistem.
3. Mengimplementasikan sistem.
4. Melakukan uji coba sistem dengan memasukkan data transaksi penjualan ke dalam sistem.
5. Melakukan evaluasi (analisis) hasil uji coba sistem.

3.1 Deskripsi Umum Sistem

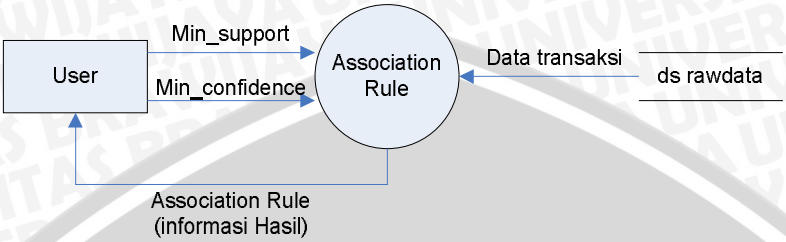
Sistem ini bertujuan untuk menemukan asosiasi antar *item* dari data transaksi penjualan pada swalayan. Dari asosiasi yang dihasilkan dapat diketahui keterkaitan *item* yang terdapat dalam transaksi penjualan. Pada sistem ini akan menghasilkan *frequent itemset*, yaitu *itemset* yang memenuhi *minimum support*. Dari *frequent itemset* yang telah dihasilkan akan digunakan untuk menemukan *association rule*. *Association rule* yang terbentuk adalah *frequent itemset* yang memenuhi *minimum confidence*.

3.2 Data Penelitian

Data utama yang diperlukan untuk penelitian ini adalah data transaksi penjualan yang diambil dari swalayan. Data ini diperlukan untuk mengetahui tingkat kedekatan antar *item* berdasarkan data transaksi penjualan selama periode tertentu.

3.3 Deskripsi Sistem

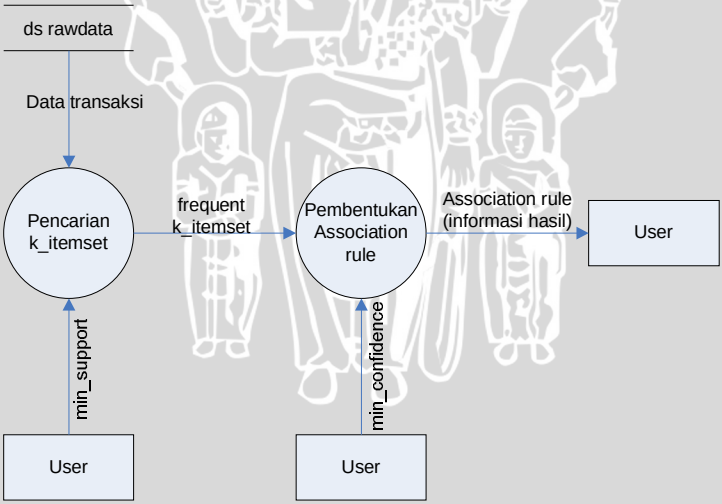
Pada subbab ini akan dibahas tentang langkah-langkah dalam proses mencari *association rule*. Gambar 3.1 adalah gambaran sistem secara garis besar.



Gambar 3.1 *Context Diagram*

Berdasarkan Gambar 3.1, dapat terlihat bahwa proses *association rule* membutuhkan 3 data masukan, yaitu data transaksi, *min_support* dan *min_confidence*. Data transaksi didapatkan dari *database* yang telah melalui proses denormalisasi, sedangkan data *min_support* dan *min_confidence* dimasukkan oleh *user*. Dan data keluaran dari proses tersebut adalah informasi hasil yang datanya berupa *association rule*, yang diberikan ke *user*.

Conteks diagram pada Gambar 3.1 dapat dijabarkan secara lebih detail pada *Data Flow Diagram* (DFD) level 1 seperti pada Gambar 3.2.



Gambar 3.2 DFD level 1

Pada DFD level 1, seperti yang terlihat pada Gambar 3.2, proses penggalian *association rule* dijabarkan menjadi 2 subproses, yaitu:

a. Proses pencarian *frequent* k-itemset, dan
b. Proses pembentukan *association rule*.

Proses pencarian *frequent* k-*itemset* seperti yang terlihat pada Gambar 3.2 membutuhkan 2 data masukan, yaitu data transaksi dan *min_support*. Sedangkan data keluaran dari proses tersebut adalah semua *frequent* k-*itemset*. Data ini nantinya akan digunakan sebagai data masukan untuk proses selanjutnya, yaitu proses pembentukan *association rule*. Jadi, proses pembentukan *association rule* membutuhkan 2 data masukan, yaitu k-*itemset* yang *frequent* (data keluaran dari proses pencarian k-*itemset*), dan data *min_confidence*. D keluaran dari proses pembentukan *association rule* adalah semua data aturan asosiasi yang memenuhi *min_confidence*.

3.4 Perancangan Sistem

Pada subbab perancangan sistem, akan dijelaskan mengenai berbagai perancangan yang terjadi dalam membangun sistem ini, antara lain perancangan basis data, perancangan proses dan perancangan *interface*.

3.4.1 Perancangan Struktur Data Sistem

Pada sistem ini, data yang digunakan disimpan dalam *database*. *Database* ini berupa kumpulan tabel yang berfungsi untuk tempat menyimpan dan mendapatkan data yang diperlukan. Berikut ini akan dijelaskan mengenai tabel-tabel yang dipergunakan.

a. Tabel Barang
   Tabel barang berisi informasi mengenai data barang yang diperlukan. Berisi data mengenai kode barang dan nama barang dan ditunjukkan pada Tabel 3.1

Tabel 3.1 Atribut tabel barang

Field	Type	Deskripsi
Kd_barang	Varchar	Kode barang
Nama_barang	Varchar	Nama barang

b. Tabel Jual
   Tabel jual berisi data semua transaksi penjualan. Tabel jual terdiri dari field kd_jual dan tanggal dan ditunjukkan pada Tabel 3.2

Tabel 3.2 Atribut tabel jual

Field	Type	Deskripsi
Kd_jual	Integer	Kode jual
Tanggal	Datetime	Tanggal transaksi

c. Tabel Detil

Tabel detil merupakan relasi antara tabel barang dengan tabel jual. Terdiri dari field kd_transaksi dan kd_barang dan ditunjukkan pada Tabel 3.3.

Tabel 3.3 Atribut tabel detil

Field	Type	Deskripsi
Kd_jual	Integer	Kode jual
Kd_barang	Varchar	Kode barang

d. Tabel Rawdata

Tabel Rawdata merupakan relasi antara tabel jual dan tabel detil. Tabel inilah yang nantinya akan digunakan dalam pencarian association rule. Terdiri dari field tanggal, kd_tansaksi, kd_barang dan ditunjukkan pada Tabel 3.4.

Tabel 3.4 Atribut tabel rawdata

Field	Type	Deskripsi
Tanggal	DateTime	Tanggal transaksi
Kd_jual	Integer	Kode jual
Kd_barang	Varchar	Kode barang

e. Tabel Asupport dan PSupport

Tabel Asupport dan PSupport merupakan tabel yang terbentuk setelah analisa *support* dijalankan. Tabel ini akan berisi *frequent itemset* dari 1-*itemset* hingga k-*itemset* yang *support*nya memenuhi *min_support* yang telah ditentukan oleh *user* dan ditunjukkan pada Tabel 3.5. Tabel ASupport untuk menyimpan hasil *frequent itemset* dengan menggunakan algoritma *apriori*, sedangkan tabel PSupport untuk menyimpan hasil *frequent itemset* dengan menggunakan algoritma *pincer search*.

Tabel 3.5 Atribut tabel ASupport dan PSupport

Field	Type	Deskripsi
Kombinasi	Integer	Jumlah kombinasi
Itsets	Varchar	Itemset
Support	Integer	Nilai support itemset

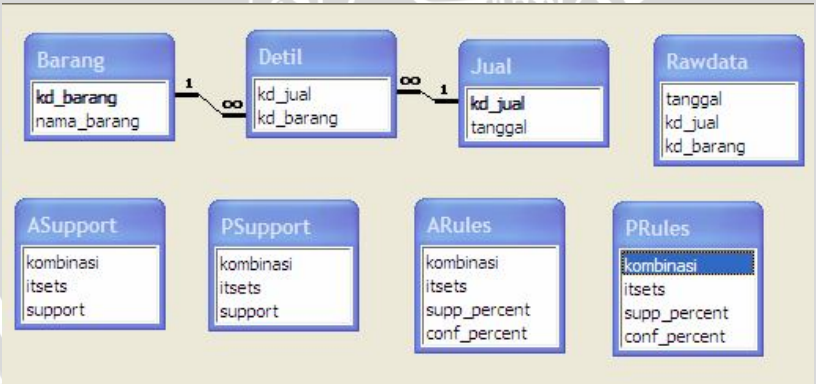
f. Tabel ARules dan PRules

Tabel ARules dan PRules berisi *rules* hasil dari *association rule* dan ditunjukkan pada Tabel 3.6. Tabel Arules untuk menyimpan hasil *rules* dengan menggunakan algoritma *apriori* dan Prules untuk menyimpan hasil *rules* dengan menggunakan algoritma *pincer search*.

Tabel 3.6 Atribut tabel Arules dan PRules

Field	Type	Deskripsi
Kombinasi	Integer	Jumlah kombinasi
Itsets	Varchar	Itemset
Support_percent	Integer	Nilai support dalam persentase
Confidence_percent	Integer	Nilai confidence dalam persentase

Struktur dan relasi antar tabel digambarkan pada gambar 3.3.



Gambar 3.3 Struktur dan Relasi antar Tabel

Tipe data yang digunakan dalam sistem ini adalah array. Tipe data ini digunakan untuk menyimpan itemset yang akan digunakan di setiap prosedur.

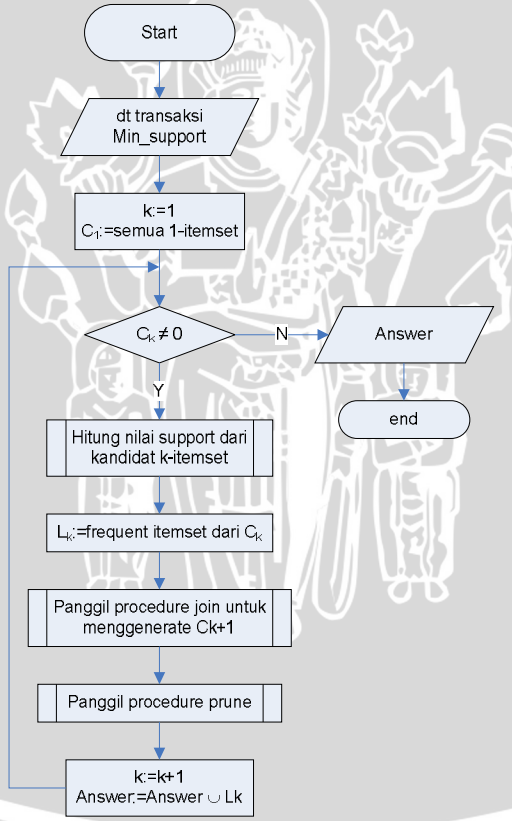
3.4.2 Perancangan Proses

Perancangan proses dalam *association rule* menggunakan algoritma *apriori* maupun algoritma *pincer search* dijelaskan pada subbab berikut ini:

1. Proses Association Rule dengan Algortima Apriori

Berikut ini merupakan prosedur-prosedur pada algoritma *apriori* yang digunakan dalam pencarian *frequent itemset.*

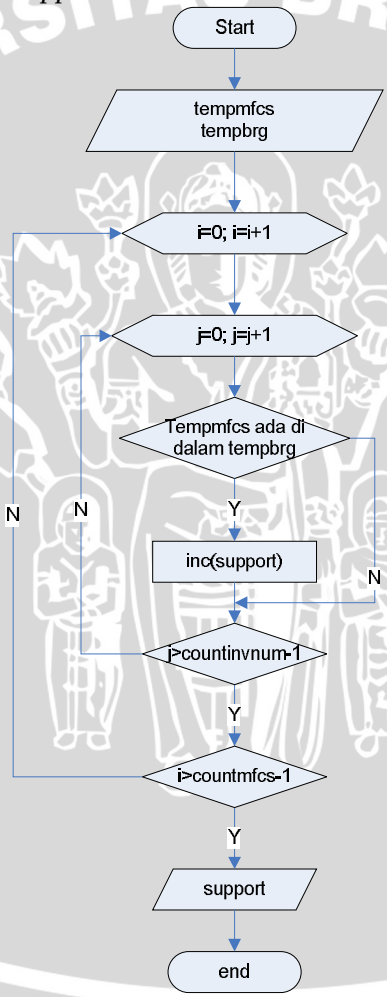
a) Prosedur *Apriori*



Gambar 3.4 *Flowchart* prosedur *apriori*

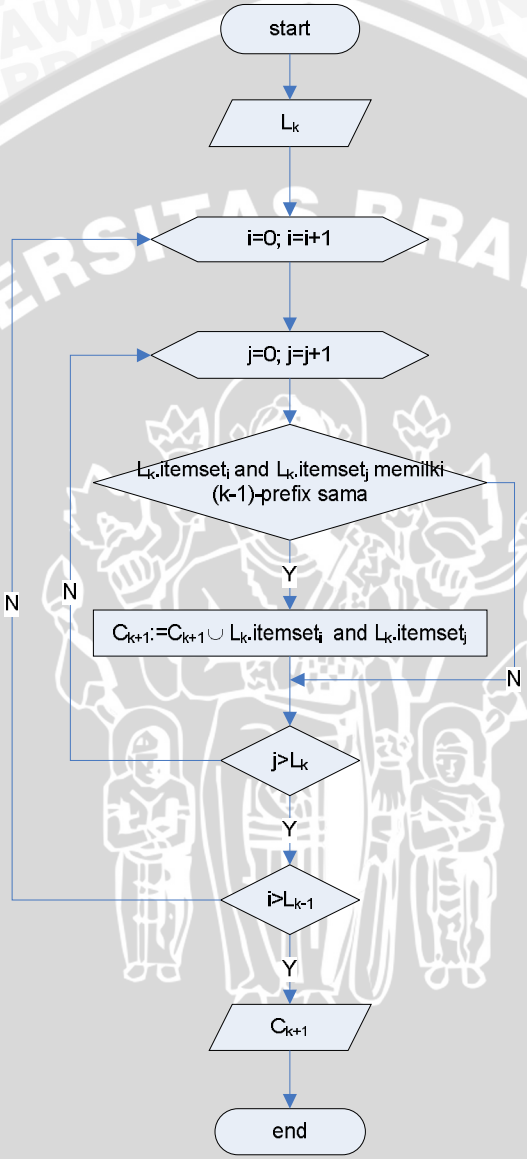
Prosedur ini merupakan prosedur utama dari algoritma *apriori*. Yang dilakukan pertama kali yaitu menghitung nilai *support* dari kandidat k-*itemset*. Jika nilai *support*nya lebih besar sama dengan nilai *min_support*, maka *itemset* tersebut akan ditambahkan ke dalam L_k. Setelah itu akan dipanggil prosedur *join* untuk meng*generate* C_{k+1}. Selanjutnya memanggil prosedur *prune* untuk melakukan *prunning*. Hal ini akan dilakukan berulang sampai nilai $C_k = 0$.

b) Prosedur *Count Support*



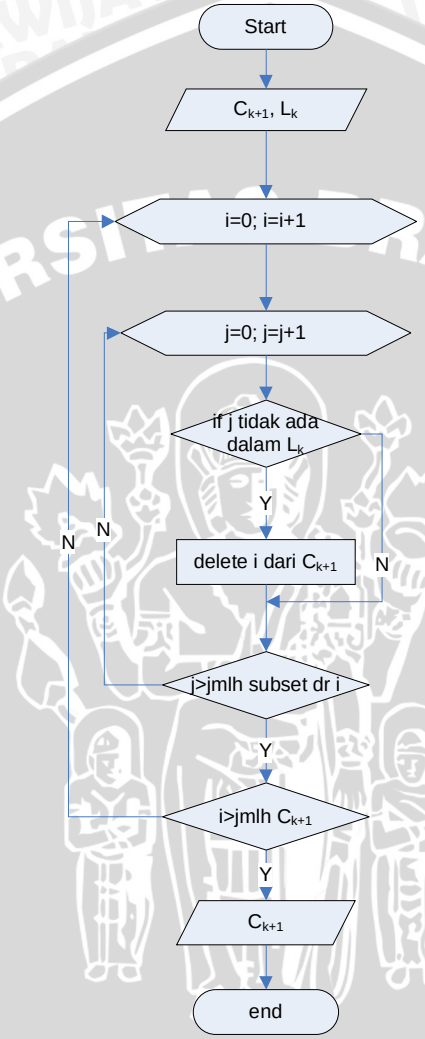
Gambar 3.5 *Flowchart* prosedur *Count Support*

c) Prosedur *Join*



Gambar 3.6 *Flowchat* prosedur *join*

d) Procedure *Prune*

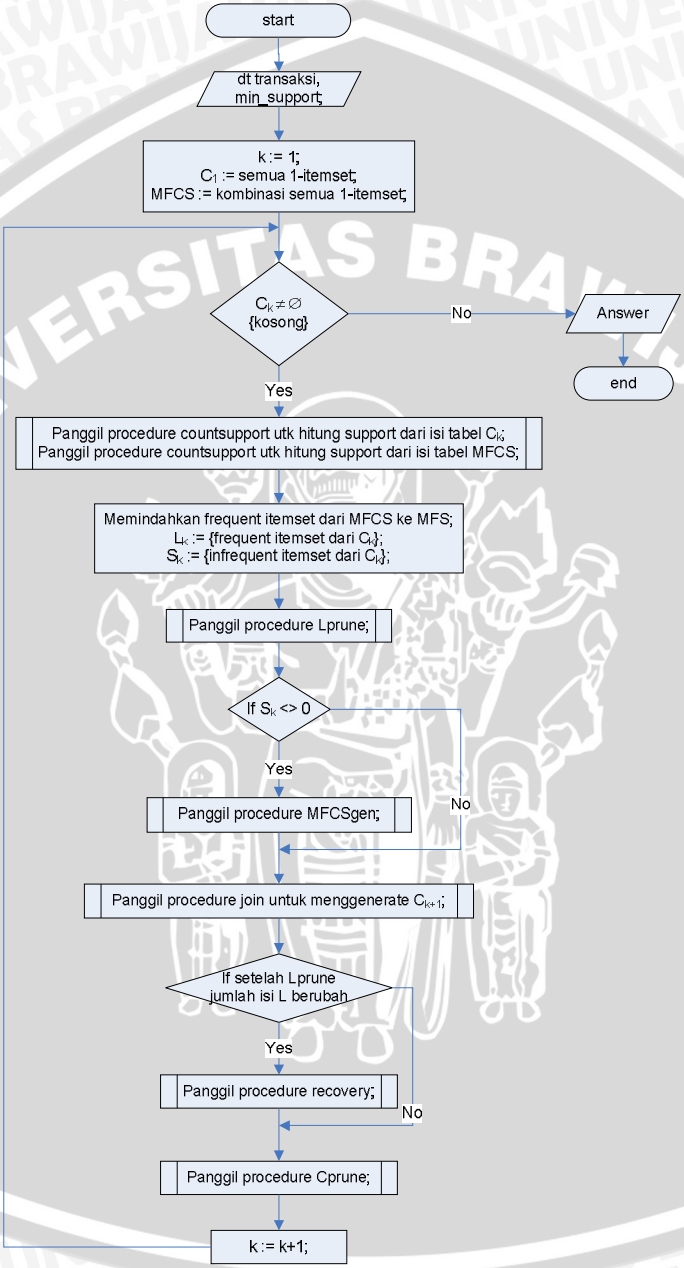


Gambar 3.7 *Flowchart* prosedur *prune*

2. Proses Association Rule dengan Algortima Pincer Search

Berikut ini merupakan prosedur-prosedur pada algoritma *pincer search* yang digunakan dalam pencarian *frequent itemset*. Untuk prosedur CountSupport dan prosedur Join sama seperti pada Gambar 3.5 dan 3.6.

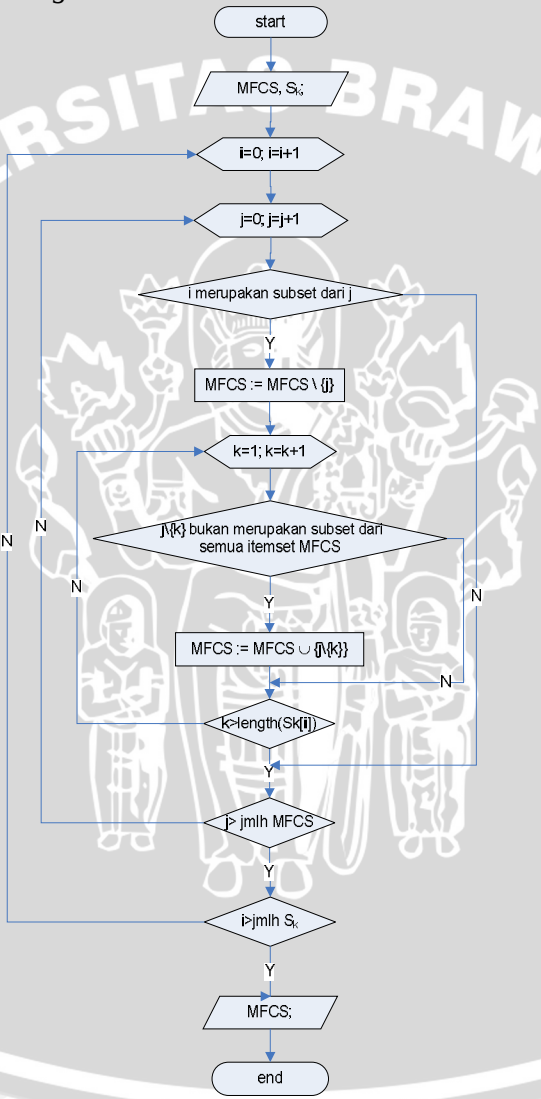
a) Prosedur *Pincer Search*



Gambar 3.8 *Flowchart* Prosedur *Pincer Search*

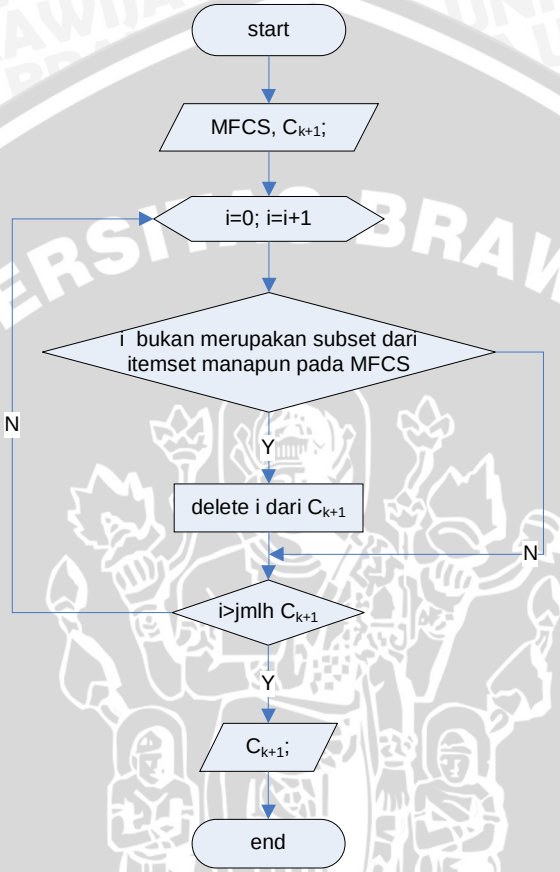
Prosedur ini merupakan prosedur utama dari algoritma *pincer search*. Prosedur ini akan memanggil prosedur-prosedur yang lain sesuai dengan keperluannya masing-masing dalam algoritma ini. Prosedur-prosedur yang dipanggil, antara lain:

b) Prosedur *MFCSgen*



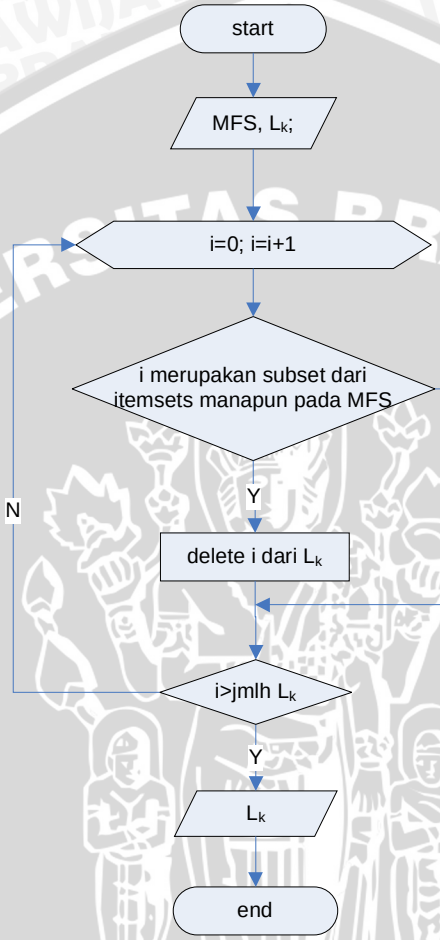
Gambar 3.9 *Flowchart* Prosedur *MFCSgen*

c) Prosedur *CPrune*



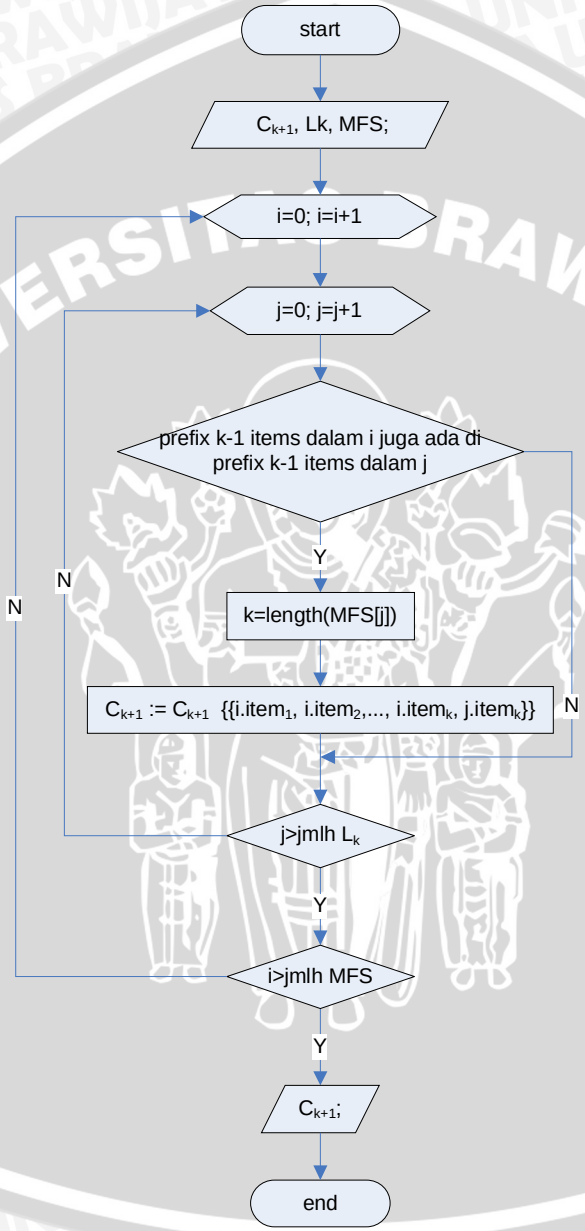
Gambar 3.10 *Flowchart* Prosedur *CPrune*

d) Prosedur *LPrune*



Gambar 3.11 *Flowchart* Prosedur *LPrune*

e) Prosedur *Recovery*



Gambar 3.12 *Flowchart* Prosedur *Recovery*

3.4.3 Perancangan Interface

Form Association Rule (Gambar 3.14) terdiri dari beberapa bagian, antara lain :

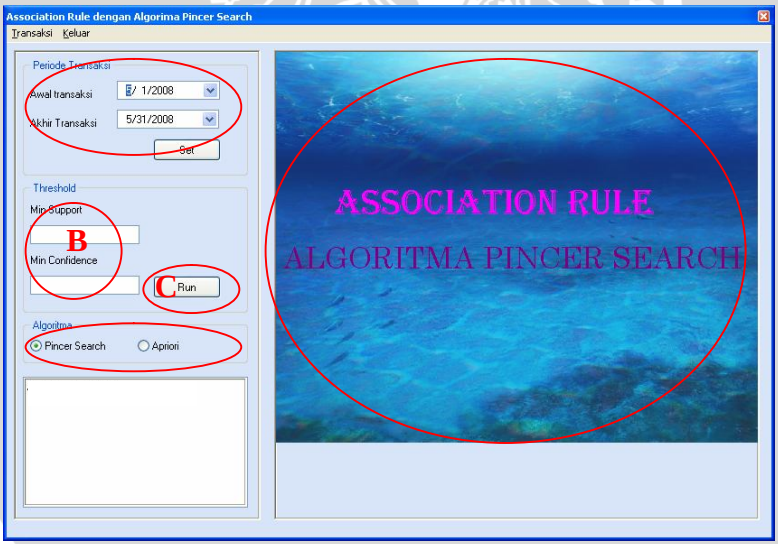
A : periode transaksi, bagian ini digunakan untuk menentukan tanggal awal dan tanggal akhir transaksi yang ingin diolah.

B : *treshold*, bagian ini user menentukan berapa *min_support* dan *min_confidence* yang diinginkan

C : *Run*, bagian ini digunakan untuk mengeksekusi program.

D : *Algoritma*, bagian ini digunakan untuk memilih antara algoritma *apriori* atau algoritma *pincer search*

E : *view result*, bagian ini akan terdiri dari dua hasil, yaitu hasil *association rule* menggunakan algoritma *apriori* dan algoritma *pincer search*. Hasil yang akan ditampilkan yaitu langkah pencarian dan hasil *rule* akan ditampilkan dalam bentuk tabel.



Gambar 3.13 Rancangan *form association rule*

3.5 Contoh Penghitungan Manual

Pujari (2001) memberikan contoh ilustrasi transaksi pada Tabel 3.5. Dimana notasi-notasi yang digunakan antara lain:
L_k = *frequent* k-*itemset*
C_k = *candidate k*-itemsets (yang potensial untuk menjadi *frequent itemsets*), dimana k menunjukkan jumlah pasangan item
S_k = *infrequent* k-*itemset*

Tabel 3.7 Contoh Tabel Transaksi

Transaksi	Itemsets
1	A1, A2, A3, A4, A5
2	A1, A3, A4, A5
3	A1, A2, A3, A4
4	A1, A3, A5
5	A4, A5

Data tabel 3.7 dalam *database* transaksional direpresentasikan seperti tampak pada Tabel 3.8:

Tabel 3.8 Representasi Data Transaksi dalam *Database* Transaksional

ID_Transaksi	Item
1	A1
1	A2
1	A3
1	A4
1	A5
2	A1
2	A3
2	A4
2	A5
.	.
.	.
.	.

Dan bila dalam bentuk tabular, data transaksi tampak seperti pada Tabel 3.9

Tabel 3.9 Format Tabular Data Transaksi

Transaksi	A1	A2	A3	A4	A5
1	1	1	1	1	1
2	1	0	1	1	1
3	1	1	1	1	0
4	1	0	1	0	1
5	0	0	0	1	1

Min_support = 2

3.5.1 Algoritma Apriori

Perhitungan manual dengan menggunakan algoritma *apriori* sebagai berikut:

Langkah 1, pada saat k=1, dilakukan penghitungan *support* 1-*itemset* sebagai berikut
{1}→4, {2}→2 , {3}→4 , {4}→ 4 , {5}→4
Berdasarkan penghitungan *support* 1-*itemset* diatas, sehingga
L_1 := {{1}, {2}, {3}, {4}, {5}}

Meng-*generate candidate set*
C_2 := {{1,2}, {1,3}, {1,4}, {1,5}, {2,3}, {2,4}, {2,5}, {3,4}, {3,5}, {4,5}}

Langkah 2, dimana k=2, dilakukan penghitungan *support* 2-*itemset* sebagai berikut
{1,2}→2, {1,3}→2, {1,4}→3, {1,5}→2, {2,3}→2, {2,4}→2, {2,5}→1, {3,4}→2, {3,5}→3, {4,5}→2
Sehingga L_2 menjadi:
L_2 := {{1,2}, {1,3}, {1,4}, {1,5}, {2,3}, {2,4}, {3,4}, {3,5}, {4,5}}

Meng-*generate candidate set*, proses kandidat *generate* dilakukan dengan,

- Menggunakan {1,2} dan {1,3} sehingga menjadi {1,2,3}
- Menggunakan {1,2} dan {1,4} sehingga menjadi {1,2,4}
- Menggunakan {1,2} dan {1,5} sehingga menjadi {1,2,5}
- Menggunakan {1,3} dan {1,4} sehingga menjadi {1,3,4}
- Menggunakan {1,3} dan {1,5} sehingga menjadi {1,3,5}
- Menggunakan {1,4} dan {1,5} sehingga menjadi {1,4,5}
- Menggunakan {2,3} dan {2,4} sehingga menjadi {2,3,4}

- Menggunakan {3,4} dan {3,5} sehingga menjadi {3,4,5}

Sehingga
C_3 := {{1,2,3}, {1,2,4}, {1,2,5}, {1,3,4}, {1,3,5}, {1,4,5}, {2,3,4}, {3,4,5}}
Pruning step menghilangkan {1,2,5}, karena *subset* 2-*itemset* {2,5} tidak berada di L_2.
Sehingga pada langkah ketiga, dimana k=3, kandidat set C_3 menjadi,
C_3 := {{1,2,3}, {1,2,4}, {1,3,4}, {1,3,5}, {1,4,5}, {2,3,4}, {3,4,5}}

Langkah 3, dimana k=3, dilakukan penghitungan *support* 3-*itemset* sebagai berikut
{1,2,3}→2, {1,2,4}→2, {1,3,4}→3, {1,3,5}→2, {1,4,5}→2, {2,3,4}→2, {3,4,5}→2
Sehingga L_3 menjadi:
L_3 := {{1,2,3}, {1,2,4}, {1,3,4}, {1,3,5}, {1,4,5}, {2,3,4}, {3,4,5}}

Meng-*generate candidate set*, proses kandidat *generate* dilakukan dengan,

- Menggunakan {1,2,3} dan {1,2,4} sehingga menjadi {1,2,3,4}
- Menggunakan {1,3,4} dan {1,3,5} sehingga menjadi {1,3,4,5}

Sehingga
C_4 := {{1,2,3,4}, {1,3,4,5}}

Langkah 4, dimana k=4, dilakukan penghitungan *support* 4-*itemset* sebagai berikut,
{1,2,3,4}→2, {1,3,4,5}→2
Sehingga L_4 menjadi:
L_4 := {{1,2,3,4}, {1,3,4,5}}

Meng-*generate candidate set*
Karena pada proses join tidak ada kandidat *itemset* yang dihasilkan, maka
$C_5 := \varnothing$
Karena $C_5 := \varnothing$, maka proses berhenti.
Jadi, total *frequent set* adalah $L := L_1 \cup L_2 \cup L_3 \cup L_4$

3.5.2 Algoritma Pincer Search

Perhitungan manual dengan menggunakan algoritma pincer search sebagai berikut:

$L_o := \emptyset$; k := 1;

C_1 := {{1}, {2}, {3}, {4}, {5}}

MFCS := {1,2,3,4,5}

MFS := ∅;

Langkah 1 : menghitung jumlah *support* masing-masing pada *database*

{1}→4, {2}→2 , {3}→4 , {4}→ 4 , {5}→4

MFCS := {1,2,3,4,5} → 1 dan MFS := ∅;

L_1 := {{1}, {2}, {3}, {4}, {5}}

$S_1 := \emptyset$

Meng-*generate candidate itemset*

C_2 := {{1,2}, {1,3}, {1,4}, {1,5}, {2,3}, {2,4}, {2,5}, {3,4}, {3,5}, {4,5}}

Langkah 2 : menghitung jumlah *support* masing-masing di C_2 pada *database*

{1,2}→2, {1,3}→2, {1,4}→3, {1,5}→2, {2,3}→2, {2,4}→2, {2,5}→1, {3,4}→2, {3,5}→3, {4,5}→2

MFCS := {1,2,3,4,5} → 1;

MFS := ∅;

L_2 := {{1,2}, {1,3}, {1,4}, {1,5}, {2,3}, {2,4}, {3,4}, {3,5}, {4,5}}

S_2 := {2,5}

Untuk {2,5} di S_2 dan untuk {1,2,3,4,5} di MFCS, mendapatkan elemen MFCS yang baru yaitu {1,3,4,5} dan {1,2,3,4}

Meng-*generate candidate set*

C_3 := {{1,2,3}, {1,2,4}, {1,2,5}, {1,3,4}, {1,3,5}, {1,4,5}, {2,3,4}, {3,4,5}}

Karena {1,2,5} bukan merupakan subset dari MFCS, maka dihilangkan dari C_3, sehingga C_3 menjadi,

C_3 := {{1,2,3}, {1,2,4}, {1,3,4}, {1,3,5}, {1,4,5}, {2,3,4}, {3,4,5}}

Langkah 3 : menghitung jumlah *support* masing-masing di C_3 pada *database*

{1,2,3}→2, {1,2,4}→2, {1,3,4}→3, {1,3,5}→2, {1,4,5}→2, {2,3,4}→2, {3,4,5}→2

MFCS := {1,3,4,5}→2, {1,2,3,4}→2
MFS := {1,3,4,5}, {1,2,3,4}
L_3 := {{1,2,3}, {1,2,4}, {1,3,4}, {1,3,5}, {1,4,5}, {2,3,4}, {3,4,5}}
Karena *itemsets* {1,2,3}, {1,2,4}, {1,3,4}, {1,3,5}, {1,4,5}, {2,3,4}, {3,4,5} merupakan *subset* dari MFS, maka dihilangkan dari L_3, tetapi tidak dilupakan, sehingga L_3 menjadi,
$L_3 := \varnothing$
$S_3 := \varnothing$

Karena tidak ada lagi kandidat yang dapat di*generate*, maka algoritma berhenti. Total *frequent set* adalah L := $L_1 \cup L_2 \cup$MFS$\cup$*subset* dari MFS yang tidak terdapat dalam L_1 dan L_2.

Kedua algoritma ini menemukan *frequent itemset* yang sama, yaitu: {{1}, {2}, {3}, {4}, {5}, {1,2}, {1,3}, {1,4}, {1,5}, {2,3}, {2,4}, {3,4}, {3,5}, {4,5}, {1,2,3}, {1,2,4}, {1,3,4}, {1,3,5}, {1,4,5}, {2,3,4}, {3,4,5}, {1,2,3,4} dan {1,3,4,5}}

Setelah semua *frequent itemset* ditemukan, hal selanjutnya yaitu membentuk *association rule*. Maka *association rule* yang dihasilkan antara lain sebagai berikut:

- *Rule* 1 : A1 ⇒ A2, mempunyai tingkat *confidence* = 2/4 = 50%
- *Rule* 2 : A2 ⇒ A1, mempunyai tingkat *confidence* = 2/2 = 100%
- *Rule* 3 : A1 ^ A2 ⇒ A3, mempunyai tingkat *confidence* = 2/2 = 100%
- *Rule* 4 : A1 ^ A3 ⇒ A2, mempunyai tingkat *confidence* = 2/2 = 100%
- *Rule* 5 : A2 ^ A3 ⇒ A1, mempunyai tingkat *confidence* = 2/2 = 100%
- *Rule* 6 : A3 ⇒ A1 ^ A2, mempunyai tingkat *confidence* = 2/4 = 50%
- *Rule* 7 : A2 ⇒ A1 ^ A3, mempunyai tingkat *confidence* = 2/2 = 100%
- *Rule* 8 : A1 ⇒ A2 ^ A3, mempunyai tingkat *confidence* = 2/4 = 50%, dst

Berdasarkan penghitungan manual dengan kedua algoritma tersebut, maka dapat dilihat algoritma *pincer search* dapat menemukan *frequent itemset* dengan satu langkah yang lebih pendek daripada algoritma *apriori* dengan hasil *rule* yang sama. Tetapi untuk penghitungan waktu akan dilakukan pengujian seperti pada subbab berikut.

3.6 Perancangan Pengujian

Pengujian dilakukan untuk membandingkan algoritma *Apriori* dan *Pincer Search* pada data transaksi penjualan. Pengujian dilakukan dengan menggunakan tabel data yang sama dengan parameter yang berbeda. Parameter yang digunakan dalam pengujian adalah nilai *min_support* dan *min_confidence.* Uji coba pertama akan mencatat hasil keluaran selain jumlah iterasi juga waktu prosesnya. Hasil dari uji coba disimpan pada Tabel 3.10.

Tabel 3.10 Pengujian Perbandingan Algoritma *Apriori* dan *Pincer Search*

Min_ Supp	***Min_ Conf***	Jmlh rule	***Apriori***		***Pincer Search***	
			Iterasi ke-	**Wkt**	***Iterasi ke-***	**Wkt**

Sedangkan uji coba kedua akan mencatat jumlah kandidat itemset dan large itemset tiap-tiap iterasi. Hasil uji coba disimpan pada tabel 3.11

Tabel 3.11 Pengujian tiap iterasi

Min_ Supp	***Iterasi ke-***	***Pincer Search***		***Apriori***	
		Lk	***Ck***	***Lk***	***Ck***

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dijelaskan pada sub bab ini adalah lingkungan perangkat keras dan lingkungan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam sistem analisis data transaksi penjulan dengan *association rule* dengan algoritma *Pincer Search* adalah

1. Prosesor Intel Celeron CPU 1.60GHz
2. RAM 1,99 GB
3. Harddisk dengan kapasitas 80 GB
4. Monitor
5. Keyboard
6. Mouse

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan adalah

1. Sistem operasi Microsoft Windows XP S*ervisePack 2*
2. *Borland Delphi* 7.0 sebagai *software development* dalam mengembangkan aplikasi.
3. Microsoft SQL Server 2000 untuk pengaksesan *database*.

4.2 Implementasi Perangkat Lunak

Sebagaimana yang telah dijelaskan pada Bab 3, aplikasi untuk *association rule* dengan algoritma *Pincer Search* ini terdiri dari dua tahap utama, yaitu tahap pembentukan *large itemset* dan tahap pembentukan *rule*. Pada saat pembentukan *large itemset* inilah algoritma *Pincer Search* digunakan.

Pada tampilan utama, *user* diminta untuk menentukan batasan tanggal yang akan diproses, *min_ support* dan *min_conf* ke dalam sistem. Proses dimulai saat *user* menekan tombol *run*.

4.2.1 Tahap Pembentukan *Large Itemset*

Pada proses ini dilakukan beberapa iterasi sampai tidak ada lagi kandidat *itemse*t yang memenuhi *minimum support*. Pada proses ini terdapat beberapa prosedur yang digunakan, yaitu:

Prosedur *Count Support*

Prosedur *count support* digunakan untuk menghitung nilai *support* kandidat *itemset* dengan bantuan array tempmfcs, tempcsmfcs dan tempbrg. Array tempmfcs digunakan untuk menyimpan *itemset* yang akan dihitung nilai *support*nya. Array tempbrg digunakan untuk menyimpan data transaksi. Isi array tempmfcs ini akan dibandingkan dengan isi array dari tempbrg. Jika jumlah variabel cloc=jumlah kombinasi n, maka *itemsets* tersebut ada dalam data transaksi, kemudian nilai variabel call akan ditambah 1. Nantinya nilai variabel call akan disimpan dalam array tempcsmfcs yang akan menyimpan semua hasil penghitungan *support* tiap *itemsets* dari tempmfcs. Baris kode yang digunakan untuk penghitungan *suppor*t ditunjukkan pada Gambar 4.1:

```
for i:=0 to countmfcs-1 do
 begin
   call:=0;
   for j:=0 to countinvnum-1 do
   begin
     cloc:=0;
       n:=1;
       for k:=1 to length(tempmfcs[i])do
       begin
         if tempmfcs[i][k]<>',' then
           tempeachitem:=tempeachitem+tempmfcs[i][k]
         else
         begin
           for l:=0 to ttempbrg[j]-1 do
           if tempbrg[j,l]=tempeachitem then
           begin
             cloc:=cloc+1;
             break;
           end;
         tempeachitem:='';
           inc(n);
         end;
       end;
       for l:=0 to ttempbrg[j]-1 do
       if tempbrg[j,l]=tempeachitem then
       begin
         cloc:=cloc+1;
```

```
          break;
        end;
      tempeachitem:='';
      if cloc=n then
        call:=call+1;
    end;
    tempcsmfcs[i]:=inttostr(call);
```

Gambar 4.1 Mengitung nilai support

Prosedur LPrune

Prosedur lprune digunakan untuk melakukan pemangkasan pada *large itemset*. Jika pada Lk ada yang merupakan *subset* dari MFS, maka *itemset* tersebut akan dihilangkan dari Lk, sehingga tidak diproses dalam iterasi selanjutnya. *Itemset* dari MFS akan disimpan terlebih dulu ke dalam array teachmfcs. Baris kode yang menunjukkan proses penyimpanan ke array teachmfcs ditunjukkan pada gambar 4.2

```
  for i:=0 to SGMFS.RowCount-2 do
  begin
    tcount:=0;
    for j:=1 to length(SGMFS.Cells[1,i]) do
      if SGMFS.Cells[1,i][j]<>',' then

teachmfcs[i][tcount]:=teachmfcs[i][tcount]+SGMFS.Cells[1,i][j]
      else
        tcount:=tcount+1;
    tlmfcs[i]:=tcount+1;
  end;
```

Gambar 4.2 Menyimpan dalam array teachmfcs

Selanjutnya, isi array teachmfcs dibandingkan dengan dengan *itemset* yang ada pada Lk. Nilai variabel countsame akan ditambah 1 ketika *item* di array sama dengan *item* Lk. Jika nilai variable countsame=nilai variable citem maka *itemsets* Lk merupakan *subset* dari MFS. Baris kode yang menunjukkan proses tersebut ditunjukkan pada gambar 4.3

```
  for i:=0 to SGLk.RowCount-2 do
  begin
    subset:=false;
    for j:=0 to cr do
    begin
      countsame:=0;
      citem:=1;
      for k:=1 to length(SGLk.Cells[1,i]) do
      begin
        if SGLk.Cells[1,i][k]<>',' then
```

```
          tempeachitem:=tempeachitem+SGLk.Cells[1,i][k]
        else
        begin
          for l:=0 to tlmfcs[j]-1 do
            if teachmfcs[j][l]=tempeachitem then
            begin
              inc(countsame);
              break;
            end;
          tempeachitem:='';
          inc(citem);
        end;
      end;
      for l:=0 to tlmfcs[j]-1 do
        if teachmfcs[j][l]=tempeachitem then
        begin
          inc(countsame);
          break;
        end;
      tempeachitem:='';
      if countsame=citem then
      begin
        subset:=true;
        break;
      end;
    end;
```

Gambar 4.3 Membandingkan array teachmfcs dengan itemset L_k

Procedure MFCSgen

Proses selanjutnya, yaitu meng*generate* MFCS jika dalam Sk terdapat *infrequent itemset*, yaitu *itemset* yang tidak memenuhi *minimum support*. Pada proses ini *itemset* yang *infrequent* pada Sk akan dihilangkan dari *itemset* MFCS sehingga akhirnya isi dari MFCS benar-benar tidak terdapat *infrequent itemset*. *Itemset* MFCS akan disimpan terlebih dahulu ke dalam array teachmfcs. Baris kode yang menunjukkan proses penyimpanan ke array teachmfcs ditunjukkan pada gambar 4.4

```
    for j:=0 to SGMFCS.RowCount-2 do
    begin
      tcount:=0;
      for k:=1 to length(SGMFCS.Cells[1,j]) do
        if SGMFCS.Cells[1,j][k]<>',' then

teachmfcs[j][tcount]:=teachmfcs[j][tcount]+SGMFCS.Cells[1,j][k]
        else
          tcount:=tcount+1;
      tlmfcs[j]:=tcount+1;
    end;
```

Gambar 4.4 Menyimpan itemset MFCS dalam array teachmfcs

Selanjutnya, isi dari array teachmfcs akan dibandingkan dengan *itemset* yang terdapat pada Sk untuk mengetahui apakah *itemset* Sk merupakan *subset* dari MFCS. Baris kode yang menunjukkan proses ini ditunjukkan pada gambar 4.5

```
countsame:=0;
for k:=1 to length(SGSk.Cells[1,i]) do
begin
  if SGSk.Cells[1,i][k]<>',' then
    tempeachitem:=tempeachitem+SGSk.Cells[1,i][k]
  else
  begin
    for l:=0 to tlmfcs[j]-1 do
      if teachmfcs[j][l]=tempeachitem then
      begin
        inc(countsame);
        break;
      end;
    tempeachitem:='';
  end;
end;

for l:=0 to tlmfcs[j]-1 do
  if teachmfcs[j][l]=tempeachitem then
  begin
    inc(countsame);
    break;
  end;
tempeachitem:='';
```

Gambar 4.5 Membandingkan array teachmfcs dengan itemset S_k

Jika S_k merupakan *subset* dari MFCS maka array tempmfcs akan dipindahkan ke array eachmfcs dan *item* yang sama dengan *item* di Sk akan diberi tanda ‘x’. Baris kode yang menunjukkan proses ini ditunjukkan pada gambar 4.6

```
sameloc:=false;
for k:=1 to length(SGSk.Cells[1,i]) do
begin
  if SGSk.Cells[1,i][k]<>',' then
    tempeachitem:=tempeachitem+SGSk.Cells[1,i][k]
  else
  begin
    for l:=0 to tlmfcs[j]-1 do
      if teachmfcs[j][l]=tempeachitem then
      begin
        sameloc:=true;
        break;
      end;
    if sameloc=true then
    begin
      r:=r+1;
```

```
            for l:=0 to tlmfcs[j]-1 do
              if teachmfcs[j][l]=tempeachitem then
                eachmfcs[r][l]:='x'
              else
                eachmfcs[r][l]:=teachmfcs[j][l];
            lmfcs[r]:=tlmfcs[j];
          end;
          tempeachitem:='';
```

Gambar 4.6 Memberi tanda ‘x’ jika subset dari MFCS

Selanjutnya, isi array eachmfcs yang berisi tanda ‘x’ untuk *item* yang dihilangkan akan akan dipindahkan lagi ke dalam array eachmfcs2 tanpa ada tanda ‘x’. Setelah itu isi array eachmfcs2 akan dicek apakah ada yang sama. Baris kode yang menunjukkan proses ini ditunjukkan pada gambar 4.7

```
  for j:=0 to r do
begin
  tcount:=0;
  for k:=0 to lmfcs[j]-1 do
    if eachmfcs[j][k]<>'x' then
    begin
      eachmfcs2[j][tcount]:=eachmfcs[j][k];
      tcount:=tcount+1;
    end;
  lmfcs2[j]:=tcount;
end;

for j:=1 to r-1 do
  for k:=0 to j-1 do
  begin
    countsame:=0;
    for l:=0 to lmfcs2[j]-1 do
      for m:=0 to lmfcs2[k]-1 do
        if eachmfcs2[j][l]=eachmfcs2[k][m] then
        begin
          countsame:=countsame+1;
          break;
        end;
    if countsame>=lmfcs2[k] then
      lmfcs2[j]:=forb;
  end;
```

Gambar 4.7 Menghilangkan tanda ‘x’

Jika isi array eachmfcs2 telah dicek, maka disimpan kembali ke dalam tabel MFCS. Baris kode yang menunjukkan proses ini ditunjukkan pada gambar 4.8

```
    for j:=0 to r do
      if lmfcs2[j]<>forb then
      begin
        for k:=0 to lmfcs2[j]-1 do
          SGMFCS.Cells[2,SGMFCS.RowCount-
1]:=SGMFCS.Cells[2,SGMFCS.rowcount-1]+eachmfcs2[j][k]+',';
        SGMFCS.Cells[2,SGMFCS.RowCount-
1]:=copy(SGMFCS.Cells[2,SGMFCS.RowCount-
1],1,length(SGMFCS.Cells[2,SGMFCS.RowCount-1])-1);
        SGMFCS.RowCount:=SGMFCS.RowCount+1;
      end;
```

Gambar 4.8 Menyimpan kembali ke MFCS

Prosedur Join

Prosedur join digunakan untuk menggabungkan *itemset* Lk dengan dirinya sendiri tetapi hanya jika memiliki k-1 *prefix* yang sama. Baris kode yang menunjukkan proses join jika kombinasi=1 ditunjukkan pada gambar 4.9

```
       for i:=0 to SGLk.RowCount-3 do
        for j:=i+1 to SGLk.RowCount-2 do
        begin
          SGCk.Cells[1,SGCk.RowCount-
1]:=SGLk.Cells[1,i]+','+SGLk.Cells[1,j];
          SGCk.RowCount:=SGCk.RowCount+1;
        end
```

Gambar 4.9 Join 1 kombinasi

Jika kombinasi>=2, Baris kode yang menunjukkan proses ini ditunjukkan pada gambar 4.10

```
      for i:=0 to SGLk.RowCount-3 do
        for j:=i+1 to SGLk.RowCount-2 do
        begin
          count1:=0;
          count2:=0;
          for k:=length(SGLk.Cells[1,i]) downto 1 do
            if SGLk.Cells[1,i][k]<>',' then
              inc(count1)
            else
              break;
          for k:=length(SGLk.Cells[1,j]) downto 1 do
            if SGLk.Cells[1,j][k]<>',' then
              inc(count2)
            else
              break;
          if copy(SGLk.Cells[1,i],1,length(SGLk.Cells[1,i])-
count1)=copy(SGLk.Cells[1,j],1,length(SGLk.Cells[1,j])-count2)
then
          begin
```

```
            SGCk.Cells[1,SGCk.RowCount-
1]:=SGLk.Cells[1,i]+','+copy(SGLk.Cells[1,j],length(SGLk.Cells[
1,j])-count2+1,length(SGLk.Cells[1,j]));
            SGCk.RowCount:=SGCk.RowCount+1;
          end;
        end;
```

Gambar 4.10 Join lebih dari satu kombinasi

Procedure Recovery

Prosedur recovery digunakan untuk mengembalikan *large itemset* yang mungkin hilang karena lprune yang dilakukan terhadap Lk. Pada prosedur ini yang dicek merupakan *subset* yaitu k-1 prefix *itemset* Lk dengan *itemset* MFS. Jika merupakan *subset* dari MFS, maka *itemset* ke k pada MFS akan digabungkan dengan k-1 *prefix itemset* Lk. *Itemset* MFS akan disimpan terlebih dahulu ke dalam array teachmfcs. Baris kode yang menunjukkan proses ini ditunjukkan pada gambar 4.11

```
  for i:=0 to SGMFS.RowCount-2 do
  begin
    tcount:=0;
    for j:=1 to length(SGMFS.Cells[1,i]) do
      if SGMFS.Cells[1,i][j]<>',' then

teachmfcs[i][tcount]:=teachmfcs[i][tcount]+SGMFS.Cells[1,i][j]
      else
        tcount:=tcount+1;
    tlmfcs[i]:=tcount+1;
  end;
```

Gambar 4.11 Menyimpan itemset MFS dalam array teachmfcs

Selanjutnya, isi array teachmfcs dibandingkan dengan dengan *itemset* Lk. Nilai variabel countsame akan ditambah 1 ketika *item* di array sama dengan *item* Lk. Jika nilai variabel countsame=nilai variabel citem maka *itemsets* Lk merupakan *subset* dari *itemset* MFS dan k-1 *prefix* pada Lk akan digabungkan dengan *item* ke k dari *itemset* MFS . Baris kode yang menunjukkan proses tersebut ditunjukkan pada gambar 4.12

```
      for k:=1 to Length(SGLk.Cells[1,i])-count1 do
      begin
        if SGLk.Cells[1,i][k]<>',' then
          tempeachitem:=tempeachitem+SGLk.Cells[1,i][k]
        else
        begin
          for l:=0 to tlmfcs[j]-2 do
            if teachmfcs[j][l]=tempeachitem then
            begin
              inc(countsame);
```

```
            break;
          end;
        tempeachitem:='';
        inc(citem);
      end;
    end;
    if countsame=citem then
    begin
      subset:=true;
      lostitem:=teachmfcs[j][tlmfcs[j]-1];
      break;
    end;
  end;
```

Gambar 4.12 Membandingkan array teachmfcs dengan itemset L_k

Prosedur CPrune

Prosedur ini digunakan untuk melakukan pemangkasan pada *candidat itemset*. Jika pada Ck terdapat *itemset* yang bukan merupakan *subset* dari MFCS, maka *itemset* tersebut akan dihapus dari Ck, sehingga tidak diproses dalam iterasi selanjutnya. *Itemset* MFCS akan disimpan terlebih dulu ke dalam array teachmfcs. Baris kode yang menunjukkan proses penyimpanan ke array teachmfcs ditunjukkan pada gambar 4.13

```
for i:=0 to SGMFCS.RowCount-2 do
  begin
    tcount:=0;
    for j:=1 to length(SGMFCS.Cells[1,i]) do
      if SGMFCS.Cells[1,i][j]<>',' then

teachmfcs[i][tcount]:=teachmfcs[i][tcount]+SGMFCS.Cells[1,i][j]
      else
        tcount:=tcount+1;
    tlmfcs[i]:=tcount+1;
  end;
```

Gambar 4.13 Menyimpan itemset MFCS dalam array teachmfcs

Selanjutnya, isi array teachmfcs dibandingkan dengan dengan *itemset* pada Ck. Nilai variabel countsame akan ditambah 1 ketika *item* di array sama dengan *itemset* Ck. Jika nilai variabel countsame=nilai variabel citem maka *itemsets* Ck merupakan *subset* dari MFCS. Baris kode yang menunjukkan proses tersebut ditunjukkan pada gambar 4.14

```
    for j:=0 to cr do
   begin
     countsame:=0;
     citem:=1;
     for k:=1 to length(SGCk.Cells[1,i]) do
     begin
       if SGCk.Cells[1,i][k]<>',' then
         tempeachitem:=tempeachitem+SGCk.Cells[1,i][k]
       else
       begin
         for l:=0 to tlmfcs[j]-1 do
           if teachmfcs[j][l]=tempeachitem then
           begin
             inc(countsame);
             break;
           end;
         tempeachitem:='';
         inc(citem);
       end;
     end;
     for l:=0 to tlmfcs[j]-1 do
       if teachmfcs[j][l]=tempeachitem then
       begin
         inc(countsame);
         break;
       end;
     tempeachitem:='';
     if countsame=citem then
     begin
       subset:=true;
       break;
     end;
  end;
```

Gambar 4.14 Membandingkan array teachmfcs dengan itemset C_k

Procedure MFStol

Prosedur MFStol digunakan untuk menjabarkan anggota MFS menjadi *subset-subset*nya untuk disimpan dalam Lk jika *itemset* tersebut sebelumnya belum disimpan dalam tabel Lk. *Itemset* yang akan dijabarkan disimpan terlebih dahulu ke dalam translatelist, variabel n merupakan nilai kombinasi dan variabel countlast digunakan untuk menyimpan nilai akhir kombinasi yang akan dijalankan. Baris code yang menunjukkan proses penjabaran anggota MFS ditunjukkan pada gambar 4.15

```
translatelist.commatext:=SGTemp.Cells[1,0];
n:=strtoint(SGTemp.Cells[0,0]);
p:=n-1;

for r:=p downto countlast do
begin
  tcomb:=StrToInt(SGTemp.Cells[0,0]);

  count:=n;
  for m:=2 to r do
    count:=count*(n-(m-1)) div m;
  setlength(x,r);
  for m:=0 to r-1 do
    x[m]:=m+1;
  addlistentry;

  for m:=2 to count do
  Begin
    incpos:=r-1;
    while x[incpos]>=n-r+incpos+1 do
      dec(incpos);
    inc(x[incpos]);
    for j:=incpos+1 to r-1 do
      x[j]:=x[j-1]+1;
    addlistentry;
  end;
end;
```

Gambar 4.15 Menjabarkan itemset MFS menjadi subset-subsetnya

4.2.2 Tahap Pembentukan Rule

Dari *large itemset* yang telah terbentuk, langkah selanjutnya adalah pembentukan *association rule* yang memenuhi *minimum support* dan *minimum confidence* yang telah ditentukan. Pada tahap ini juga dilakukan penjabaran terhadap *itemset*. *Itemset* akan dijabarkan sampai dengan Lk=1. Setelah itu, *itemset* yang telah dijabarkan akan diurutkan secara terbalik. *Itemset* yang pertama dan *itemset* yang telah diurutkan akan digabung sehingga membentuk suatu *rule.*

Baris kode untuk penjabaran *itemset* sampai pada *subset* kombinasi =1 ditunjukkan pada gambar 4.16.

```
    translatelist.commatext:=StringGrid2.Cells[1,0];
    n:=strtoint(StringGrid2.Cells[0,0]);
    p:=n-1;
    for r:=p downto 1 do
    begin
      count:=1;
      actualcount:=0;
      for m:=1 to r do
        count:=count*(n-(m-1));
      setlength(x,r);
      for m:=0 to r-1 do
        x[m]:=m+1;
      addlistentry;

      for m:=2 to r do
        count:=count div m;
      for m:=2 to count do
      Begin
        incpos:=r-1;
        while x[incpos]>=n-r+incpos+1 do
          dec(incpos);
        inc(x[incpos]);
        for j:=incpos+1 to r-1 do
          x[j]:=x[j-1]+1;
        addlistentry;
      end;
    end;
    for j:=StringGrid2.RowCount-2 downto 0 do
    begin
      StringGrid3.Cells[1,StringGrid3.RowCount-
1]:=StringGrid2.Cells[1,j];
      StringGrid3.RowCount:=StringGrid3.RowCount+1;
    end;
```

Gambar 4.16 Mencari kombinasi MFS sampai dengan 1 kombinasi

Selanjutnya, *itemset* yang telah diurutkan dengan *itemset* sebelumnya akan digabungkan sehingga membentuk suatu aturan. Setelah itu dilakukan penghitungan untuk menghitung nilai *supp_percent* dan *conf_percent*. Baris kode untuk membentuk aturan *association rule* ditunjukkan pada gambar 4.17 :

```
for i:=0 to StringGrid2.RowCount-2 do
      begin
        StringGrid4.Cells[0,i]:=IntToStr(i+1);
        StringGrid4.Cells[1,i]:=supersets;
        StringGrid4.Cells[2,i]:=StringGrid2.Cells[1,i]+'-
   >'+StringGrid3.Cells[1,i];
        tc:=round(supersupp*100/tottransaction);
        StringGrid4.Cells[3,i]:=inttostr(tc);
      ic:=round(supersupp*100/StrToInt(StringGrid2.Cells[2,i]));
        StringGrid4.Cells[4,i]:=intToStr(ic);
      end;
```

Gambar 4.17 Membentuk *association rule*

4.3 Penerapan Aplikasi

Aplikasi diterapkan dengan memasukkan data sesuai dengan keinginan *user*. Langkah awal yang dilakukan ialah menentukan tanggal data transaksi yang akan dianalisa pada tabel transaksi. Penentuan data awal dan data akhir transaksi yang digunakan ditentukan dengan memasukkan batasan tanggal transaksi atau periode tanggal transaksi terjadi. Batasan tanggal digunakan untuk menganalisa data pada periode tertentu saja dari keseluruhan transaksi.

Langkah selanjutnya yaitu *user* menentukan batasan untuk nilai *min_supp* dan *min_conf*. Nilai *min_support* yang ditentukan akan berpengaruh pada banyak atau sedikit serta lama atau cepatnya proses di dalam aplikasi. Sedangkan nilai *min_conf* akan berpengaruh pada jumlah aturan yang dihasilkan.

Setelah batasan *min support* dan *min confidence* ditentukan, *user* dapat langsung memulai proses dengan menekan tombol *Run*.

4.4 Analisa Hasil

Untuk menguji kekonsistenan aplikasi, dalam penelitian ini dilakukan pengujian terhadap aplikasi. Aplikasi diterapkan dengan memasukkan parameter-parameter yang sesuai dengan keinginan *user* terhadap data transaksi yang dianalisa sehingga dapat memperoleh hasil analisa yang lebih spesifik.

Berdasarkan skenario uji coba yang telah dijelaskan pada subbab 3.6, pada pengujian pertama, dilakukan tiga kali uji coba dengan jumlah transaksi yang berbeda.

Pada pengujian pertama, data yang digunakan adalah 477 transaksi dengan 2479 jumlah *record*. Pengujian dilakukan dengan *minimum support* dan *minimum confidence* yang berbeda-beda. Hasil pengujian ditunjukkan pada tabel 4.1.

Tabel 4.1 Tabel pengujian 477 transaksi

Min_ Supp	*Min_ Conf*	Jmlh rule	*Apriori*		*Pincer Search*	
			Iterasi ke-	Wkt	*Iterasi ke-*	Wkt
5	50%	867	7	0:1:27	4	0:1:23
	75%	663				
	100%	559				
7	50%	39	5	0:0:35	4	0:0:34
	75%	14				
	100%	3				
9	50%	18	4	0:0:18	4	0:0:18
	75%	5				
	100%	0				

Pada pengujian kedua, data yang digunakan adalah 1459 transaksi dengan 12441 jumlah *record*. Pengujian dilakukan dengan *minimum support* dan *minimum confidence* yang berbeda-beda. Hasil pengujian ditunjukkan pada tabel 4.2

Tabel 4.2 Tabel pengujian 1459 transaksi

Min_ Supp	*Min_ Conf*	Jmlh rule	*Apriori*		*Pincer Search*	
			Iterasi ke-	Wkt	*Iterasi ke-*	Wkt
7	50%	22	5	0:8:40	4	0:7:08
	75%	9				
	100%	2				
8	50%	8	4	08:34	4	0:6:50
	75%	3				
	100%	1				
9	50%	5	4	0:8:21	4	0:6:46
	75%	2				
	100%	0				

Pada pengujian ketiga, data yang digunakan adalah 5627 transaksi dengan 48243 jumlah *record*. Pengujian dilakukan dengan *minimum support* dan *minimum confidence* yang berbeda-beda. Hasil pengujian ditunjukkan pada tabel 4.3

Tabel 4.3 Tabel pengujian 5627 transaksi

Min_ Supp	*Min_ Conf*	Jmlh rule	*Apriori*		*Pincer Search*	
			Iterasi ke-	Wkt	*Iterasi ke-*	Wkt
15	50%	7	4	0:32:03	4	0:28:32
	75%	3				
	100%	0				
20	50%	2	4	0:32:03	4	0:28:32
	75%	1				
	100%	0				
25	25%	1	4	0:31:44	4	0:28:31
	50%	0				
	100%	0				
30	25%	1	3	0:31:28	3	0:28:15
	50%	0				
	100%	0				

Bentuk pengujian yang kedua yaitu menghitung jumlah kandidat dan *frequent itemset* tiap-tiap iterasi. Hasil dari pengujian 477 transaksi ditunjukkan pada tabel 4.4

Tabel 4.4 Tabel Pengujian tiap iterasi

Min_ Supp	*Iterasi ke-*	*Pincer Search*		*Apriori*	
		Lk	Ck	*Lk*	*Ck*
5	1	204	20706	204	20706
	2	66	50	66	50
	3	1	0	43	20
	4	-	-	30	12
	5	-	-	12	2
	6	-	-	2	0
7	1	134	8911	134	8911
	2	21	7	21	7
	3	1	0	5	1
	4	-	-	1	0
9	1	95	4465	95	4465
	2	10	2	10	2
	3	2	0	2	0

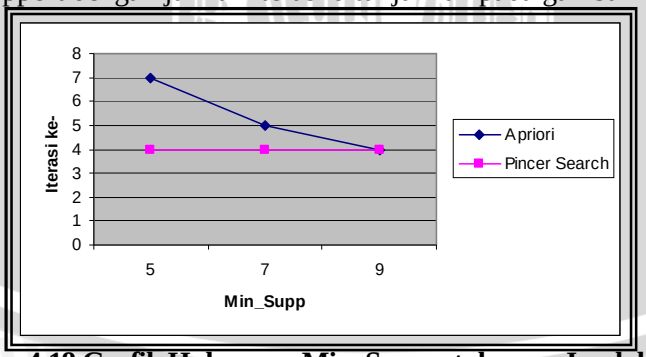
Berdasarkan data pengujian pada tabel 4.1, 4.2 dan 4.3, didapatkan bahwa kecepatan program dalam melakukan penghitungan *support* dan pencarian *rules* tergantung pada data yang diproses, dalam hal ini transaksi dan jenis barang yang terdapat di dalam transaksi serta *minimum support* dan *minimum confidence* yang dimasukkan oleh *user*.

Besar kecilnya *minimum support* akan berpengaruh terhadap waktu yang diperlukan untuk melakukan proses. Nilai *minimum support* berbanding terbalik dengan waktu yang diperlukan. Semakin kecil nilai *minimum support*, maka waktu yang diperlukan juga semakin lama. Hal ini dikarenakan, dengan *nilai minimum* yang kecil maka semakin banyak *itemset* yang memenuhi *minimum support* sehingga semakin banyak pula *itemset* yang akan diproses.

Banyaknya *minimum confidence* juga mempengaruhi jumlah aturan yang dihasilkan. Semakin kecil nilai *minimum confidence*, maka menghasilkan *rule* yang lebih banyak. Sebaliknya, *minimum confidence* yang besar akan menghasilkan *rule* yang sedikit.

Berdasarkan pada tabel 4.4, jumlah iterasi pada algoritma Pincer Search dengan min_support yang kecil, lebih pendek dibandingkan dengan algoritma Apriori. Pada saat min_support=5 iterasi ke-3, jumlah Ck dan Lk pada Pincer Seach mengalami perubahan dari algoritma Apriori. Dengan jumlah Ck dan Lk yang kecil, maka waktu proses juga akan lebih cepat. Hal ini dkarenakan adanya prosedur Lprune, dimana itemset Lk akan dihilangkan jika itemset tersebut merupakan subset dari MFS, seperti yang telah dibahas pada perancangan proses pada bab 3.

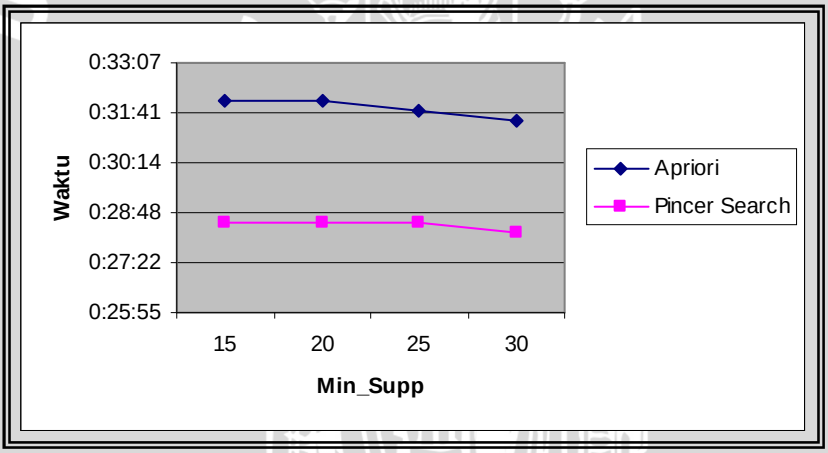
Berdasarkan pada table 4.1, maka hasil perbandingan min_support dengan jumlah iterasi ditunjukkan pada gambar 4.18



Gambar 4.18 Grafik Hubungan Min_Support dengan Jumlah iterasi

Pada Gambar 4.18 dapat disimpulkan bahwa dengan *min_support* yang lebih kecil algoritma *Pincer Search* dapat menemukan *large itemset* dengan langkah yang lebih pendek. Misal, dengan *min_support*=5, terlihat bahwa algoritma *Pincer Search* menemukan *large itemset* pada iterasi ke-4, sedangkan algoritma *Apriori* menemukan *large itemset* pada iterasi ke-7. Algoritma *Pincer Search* menemukan *large itemset* lebih pendek 3 iterasi dibandingkan dengan algoritma *Apriori*. Hal ini disebabkan karena pada algoritma *Pincer Search* proses pencarian dilakukan secara dua arah yaitu *Bottom Up* dan *Top Down* seperti yang telah dijelaskan pada subbab 2.4 yaitu jika *k-itemset frequent*, maka semua *subset* dari *frequent* ini adalah *frequent* dan tidak perlu di*generate* lagi.

Sedangkan berdasarkan pada tabel 4.3 maka hasil perbandingan min_support dengan waktu yang diperlukan ditunjukkan pada Gambar 4.19



Gambar 4.19 Grafik Hubungan Min_Support dengan Waktu yang diperlukan

Pada Gambar 4.2 dapat disimpulkan bahwa waktu yang diperlukan pada algortima *Pincer Search* lebih cepat dibandingkan dengan algoritma *Apriori*. Hal ini juga berkaitan dengan pengujian tabel 4.4. Dimana dengan jumlah itemset yang lebih kecil, maka waktu yang diperlukan juga akan lebih cepat.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil analisis skripsi ini didapatkan kesimpulan antara lain:

1. Algoritma *Apriori* dan algoritma *Pincer Search* menghasilkan *large itemset* yang sama.
2. Dengan *min_support* yang kecil, algoritma *Pincer Search* dapat menemukan *large itemset* dengan langkah yang lebih pendek dibandingkan dengan algoritma *Apriori*. Karena dengan *min_support* yang kecil, *maximal frequent set* pada *Pincer Search* lebih cepat ditemukan.
3. Waktu proses pada *algoritma Pincer Search* lebih cepat dibandingkan dengan algoritma *Apriori*. Terlihat pada tabel 4.4 pada saat *min_support*=5 iterasi ke-3, terjadi penurunan jumlah C_k dan L_k pada algoritma Pincer Search. Dengan jumlah itemset yang kecil, maka waktu prosesnya akan lebih cepat dibandingkan dengan waktu proses saat itemset besar.
4. Besarnya *min_support* berbanding terbalik dengan waktu yang diperlukan untuk menemukan *large itemset*.
5. Besarnya *min_confidence* berbanding terbalik dengan jumlah aturan yang dihasilkan.
6. Waktu yang diperlukan untuk pemrosesan tergantung pada jumlah transaksi dan jumlah barang.

5.2 Saran

Dari penelitian yang telah dilakukan dapat direkomendasikan saran-saran sebagai berikut:

1. Pada pengembangan yang mendatang, dapat ditambahkan fasilitas untuk melakukan *mining* berdasarkan jenis/kategori barang tertentu.
2. Selain dalam bentuk tabel, hasil dari *association rule* dapat ditampilkan dalam bentuk grafis sehingga lebih memperjelas *rules-rules* yang didapatkan dari proses pencarian *rule*.

DAFTAR PUSTAKA

A.K. Pujari. 2001. *Data Mining Techniques*. Orient Blackswan

Budhi, Gregorius Satia, Leo W. Santoso dan Edward Susanto. 2006. Metode *Market Basket Analysis* menggunakan Algoritma *Pincer Search* untuk Sistem Pembantu Pengambilan Keputusan. *1st Seminar on Application and Research in Industrial Technology, SMART* 2006. Jurusan Teknik Informatika Universitas Kristen Petra. Surabaya

Edelstein, H.A. 1999. *Introduction to Data Mining and Knowledge Discovery (3rd Edition)*. Two Crows Corp. Potomac, MD.

Han, Jiawei, Micheline Kamber. 2001. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, California.

Kadir, Abdul. 2004. *Pemograman Database dengan Delphi 7 Menggunakan Access dan ADO*. Andi Offset. Yogyakarta.

Kantardzic, Mehmed. 2003. *Data Mining: Concepts, Models, Methods, and Algorithms*. John Wiley & Sons. New Jersey. Subbab Data-Mining Process.

Larose, Daniel T. 2005. *Discovering Knowledge in Data: An Introduction to Data Mining*. John Wiley & Sons, Inc. New Jersey.

Lin, Dao-I, Zvi M. Kedem. 1997. *Pincer-Search: A New Algorithm for Discovering the Maximum Frequent Sets*. Proceeding Of The 6th International Conference On Extending Database Technology (EDBT).

Lin, Dao-I. 1998. *Fast Algorithms for Discovering the Maximum Frequent Set*. Dissertation, Department of Computer Science, New York University.

Malreddy, Kranthi K. 2004. *Mining Frequent Using Advanced Partition Approach*. A Thesis in Computer Science, Faculty of Texas Tech University

R. Agrawal and R. Srikant. 1994. "*Fast Algorithm for Mining Association Rules*". In Proceedings of the International Conference on Very Large Data Bases.

Yulita, Marsela dan Veronica S. Moertini. 2002. *Analisis Keranjang Pasar dengan Algoritma Hash-Based pada transaksi Penjualan di Apotek*. Integral Majalah Ilmiah Matematika dan Ilmu Pengetahuan Alam, Vol 9, No 3 (2002). Jurusan Ilmu Komputer Universitas Katolik Parahyangan. Bandung.