Institut Teknologi Sepuluh Nopember

TESIS - CI2541

PEMBANGKITAN KAIDAH ASOSIASI DARI TOP-K FREQUENT CLOSED ITEMSET YANG DIDASARKAN PADA STRUKTUR DATA BERBASIS LATTICE

Dian Puspita Hapsari
NRP : 5104 201 023

DOSEN PEMBIMBING
Prof. Dr. Ir. Arif Djunaidy, M.Sc.

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2008

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M. Kom)
di
Institut Teknologi Sepuluh Nopember

oleh :
Dian Puspita Hapsari
Nrp. 5104 201 023

Tanggal Ujian : 04 Agustus 2008
Periode Wisuda : Oktober 2008

Disetujui oleh:

1. **Prof. Dr. Ir. Arif Djunaidy, M.Sc.** **(Pembimbing)**
   NIP. 131 633 403

2. **Ir. F.X. Arunanto, M.Sc.** **(Penguji)**
   NIP. 131 285 253

3. **Ir. Esther Hanaja, M.Sc.** **(Penguji)**
   NIP. 130 816 212

4. **Yudhi Purwananto, M.Kom.** **(Penguji)**
   NIP. 132 172 210

Direktur Program Pascasarjana,

DEPARTEMEN PENDIDIKAN NASIONAL INSTITUT TEKNOLOGI SEPULUH NOPEMBER PROGRAM PASCASARJANA

Prof. Ir. Suparno, MSIE., PhD
NIP. 130 532 035

PEMBANGKITAN KAIDAH ASOSIASI DARI TOP-K FREQUENT CLOSED ITEMSET BERBASIS STRUKTUR DATA LATTICE

Nama Mahasiswa : Dian Puspita Hapsari
NRP : 5104 201 023
Pembimbing : Prof. Dr. Ir. Arif Djunaidy, M.Sc.

ABSTRAK

Proses penggalian kaidah asosiasi pada dasarnya terdiri dari dua tahapan utama, yaitu pembangkitan item-item yang sering muncul dan pembangkitan kaidah-kaidah asosiasi. Dalam penelitian sebelumnya, terdapat dua algoritma utama untuk proses pembangkitan item-item yang sering muncul, yaitu algoritma yang berbasis struktur data lattice seperti Algoritma CHARM-L (*Closed Association Rule Mining-Lattice*) dan algoritma yang berbasis struktur data FP-Tree seperti Algoritma TFP (*Top Frequent Pattern*). Dari penelitian yang berkaitan dengan penggalian *frequent closed itemsets*, belum pernah dikembangkan algoritma berbasis struktur data lattice menggunakan batasan top-k sebagai pengganti penetapan nilai *minimum support*, sehingga terdapat peluang dilakukannya penelitian untuk mendesain dan mengimplementasikan algoritma penggalian top-k *frequent closed itemset* berbasis struktur data *lattice*.

Dalam penelitian ini dikembangkan algoritma pembangkitan kaidah asosiasi dari top-k *frequent closed itemset* berbasis struktur data *lattice*. Dalam penelitian ini, nilai top-k digunakan sebagai pengganti nilai batasan *minimum support* sebagai acuan derajat kemunculan sebuah itemset dari *frequent itemset* yang paling dicari atau paling diinginkan oleh pengguna. Untuk ini, pengguna hanya diminta memasukkan nilai bilangan bulat postif *k* untuk membangkitkan semua kaidah asosiasi dari top-k frequent closed itemset yang dapat dibangkitkan sesuai dengan nilai minimum confidence atau tingkat korelasi yang ditentukan oleh pengguna. Algoritma yang berhail dikembangkan dalam penelitian ini diimplementasikan dalam lingkungan sistem operasi Windows dengan menggunakan bahasa pemrograman C. Kinerja waktu komputasi dalam membangkitkan top-k frequent closed itemset dari algoritma yang dikembangkan dibandingkan dengan algoritma pembanding sejenis yang membangkitkan top-k frequent closed itemset berbasis struktur data frequent-pattern tree (algoritma TFP).

Hasil uji coba menunjukkan bahwa algoritma yang telah berhasil diimplementasikan mampu menghasilkan semua kaidah asosiasi yang diingikan oleh pengguna dari top-k frequent closed itemset yang dibangkitkan sesuai dengan nilai *k* yang diberikan. Walaupun kinerja waktu komputasi pembangkitan top-k frequent closed itemset dari algoritma yang dikembangkan sedikit lebih lambat dibandingkan dengan algoritma pembandingnya, tetapi struktur data lattice yang dihasilkan dapat memudahkan penentuan hubungan subset dan superset antar itemset dalam proses pembangkitan kaidah asosiasi. Kelebihan ini terutama berguna untuk mengurangi ruang pencarian kandidat itemset yang dianggap infrequent, sehingga dapat mempercepat proses pembangkitan kaidah asosiasi.

Kata kunci: top-k frequent closed itemset, struktur data lattice, kaidah asosiasi, data mining.

MINING ASSOCIATION RULES FROM A TOP-K FREQUENT CLOSED ITEMSET BASED-ON THE LATTICE DATA STRUCTURE

By : Dian Puspita Hapsari
Student Identity Number : 5104 201 023
Supervisor : Prof. Dr. Ir. Arif Djunaidy, M.Sc.

ABSTRACT

Basically, the mining process of association rules can be divided into two major stages; i.e., the frequent itemset generation and association rules generation. Two primary algorithms for generating frequent itemsets have been developed in the earlier research works; these are, the algorithm which is based on the lattice data structure, such as CHARM-L (Closed Association Rule Mining - Lattice) algorithm, and the algorithm which is based on FP-Tree data structure, such as TFP (Top Frequent Pattern) algorithm. Among the research works that deal with the frequent closed itemsets mining, no research work has been done for developing an algorithm that is based on the lattice based data structure that uses the top-k as a parameter substituting the minimum support value requirement. Therefore, a research opportunity exists for designing and implementing an algorithm for mining a top-k frequent closed itemset based on the lattice data structure.

In this research, an algorithm for mining association rules from a top-k frequent closed itemset based on lattice data structure is developed. The top-k value is used as a substitution of the minimum support value as a level of appearance for filtering the most interesting frequent itemset required by users. The user is required to give a positive integer *k* as an input to generate all possible association rules from a top-k frequent closed itemset according to the minimum confidence value or the corelation rank ditermined by the user. The algorithm that has been succesfully developed and implementated using C programming language within the Windows operating system. The run time for generating the top-k frequent closed itemsets based on the lattice data structure of the algorithm developed in this research is compared to the similar algorithm for mining top-k frequent closed itemsets based on the frequent-pattern tree (i.e., TFP algorithm).

Experimental results show that the algorithm that has been succesfully implementated is capable of producing all the association rules needed by the user from a top-k frequent closed itemset based on lattice data structure according to the given *k* value. The run time of the algorithm developed in this research is slightly slower than that of the algorithm used for the comparison. However, the lattice data structure produced by the algorithm is useful for determining subset and superset relationship among items during the process of association rules generation. This strong poin is particularly useful to reduce the search space for finding candidate of infrequent itemset, and therefore it can faster the process of mining association rules.

Keywords : top-k frequnet closed itemset, lattice data structure, mining association rule, data mining.

KATA PENGANTAR

Segala puji-pujian hanya kepada Allah SWT yang telah memberikan rahmat dan hidayah-Nya atas selesaikannya tesis ini. Sholawat dan salam kepada junjungan kita Nabi Muhammad SAW yang telah menghantarkan kita kepada jalan yang benar.

Rasa terima kasih yang mendalam saya ucapkan pula kepada :

1. Ibu dan Bapak, terutama Ibu yang selalu mengiringi kami dengan do'a.
2. Bapak Prof. Dr. Ir. Arif Djunaidy, M.Sc selaku dosen pembimbing yang telah banyak memberikan waktu dan pikiran dalam membimbing dan membukakan pintu ilmu-ilmu baru yang sebelumnya tidak pernah kami dapatkan.
3. Teman-teman Pasca Sarjana Teknik Informatika angkatan 2004 yang selalu kompak memberikan dukungan moril dan spirituil sehingga selesainya tesis ini.
4. Suami, Alit Arswendo yang tercinta dan putri tersayang, Zaha Kalisha Azra, yang selalu setia mendampingi dan selalu memberikan ketenangan hati.
5. Rasa terimakasih juga tak lupa kami ucapkan kepada para dosen penguji dan para dosen pengajar di Jurusan Teknik Informatika Fakultas Teknologi Informasi, yang telah memberi bimbingan selama saya menempuh studi Pascasarjana.

Penulis berharap agar buku ini dapat memberikan sumbangan ilmu pengetahuan, utamanya dibidang data mining dengan teknik asosiasi. Buku ini disusun dengan segala kemampuan dan keterbatasannya, seperti dalam pepatah, "tiada gading yang tak retak", maka penulis mengharapkan perbaikan-perbaikan dari pembaca mengenai segala kekurangan yang ada.

Terima kasih,

Penulis

Surabaya, Agustus 2008

DAFTAR ISI

Halaman

Halaman

DAFTAR GAMBAR

Halaman

HALAMAN INI SENGAJA DIKOSONGKAN

DAFTAR TABEL

Halaman

DAFTAR SIMBOL & ISTILAH

Simbol	Istilah	Definisi	Hal
$\mathcal{T}$	*Transaksi*	Sebuah basisdata transaksi yang memiliki transaksi identifier yang unik dan berisi sekelompok item-item.	9
$\mathcal{A}$	*Itemset*	Sebuah himpunan yang beranggotakan item-item.	9
	k-Itemset	Sebuah himpunan yang memiliki anggota sejumlah k item.	9
$\sigma(X)$	*Support dari sebuah itemset X (Support Count)*	Menyatakan jumlah transaksi yang muncul dan mengandung *itemset* X dalam suatu basis data.	9
	Confidence	Seberapa frequent item – item dalam *Y* muncul dalam transaksi yang berisi *X*.	10
Min_conf	*Minimum Confidence*	Nilai batas terendah untuk confidence dari sebuah itemset X yang ditentukan oleh pengguna.	11
Min_sup	*Minimum Support*	Nilai batas terendah untuk support dari sebuah itemset X yang ditentukan oleh pengguna.	11
	Top-k	Tingkat keseringan muncul senilai k	
	Frequent	Itemset dianggap frequent jika nilai support itemset tersebut lebih besar dari nilai batas minimum support yang ditentukan.	13
	Frequent itemset disebut *maximal*	Jika tidak ada satupun *superset itemset* terdekatnya yang *frequent*.	17
	Frequent itemset disebut *closed*	Jika tidak ada satupun *superset* $Y \supset X$ dengan $\sigma(X) = \sigma(Y)$.	18

BAB 1
PENDAHULUAN

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Dalam data mining, analisa asosiasi digunakan untuk mencari relasi-relasi yang direpresentasikan pada kaidah-kaidah asosiasi atau sekelompok item-item yang sering muncul. Strategi umum yang banyak digunakan dalam algoritma–algoritma pencarian kaidah asosiasi atau *association rule mining* dibagi menjadi dua tahap utama. Tahap pertama, pencarian semua itemset-itemset yang sering muncul dan memenuhi batasan minimum support atau sering disebut sebagai *frequent itemset generation*. Dan tahap kedua, pembangkitan kaidah asosiasi yang menghasilkan kaidah-kaidah dengan nilai confidence yang tinggi atau disebut sebagai *rule generation*.

Pada tahap pertama dalam algoritma pencarian kaidah asosiasi, yaitu pencarian itemset-itemset yang sering muncul pada umumnya menggunakan sebuah batasan nilai *minsup* untuk menghasilkan kumpulan itemset–itemset yang sering muncul secara lengkap dan benar. Hal tersebut didasari oleh algoritma yang sangat populer, yaitu algoritma *Apriori* (Agrawal, 1996). Keunggulan algoritma *Apriori* adalah mampu mengurangi ruang pencarian atau *search space* dari itemset-itemset yang sering muncul (*frequent itemset*) dengan *Prinsip Apriori*, (bahwa setiap subpola dari sebuah pola yang sering muncul pasti bersifat sering muncul pula). Namun algoritma Apriori akan mengalami penurunan performa pada dataset yang rapat (yaitu dataset yang memiliki pola-pola yang panjang).

Kelemahan algoritma Apriori diperbaiki pada algoritma CHARM-L (Zaki, Hsiao, 2005) yang mampu menghasilkan *frequent closed itemset lattice*. Pencarian itemset-itemset yang sering muncul dengan menggunakan *frequent closed itemset* akan membantu dalam pembangkitan set-set kaidah-kaidah yang tidak berulang atau *nonredundant rules sets*. Dan dengan menggunakan *lattice* akan meningkatkan efisiensi waktu pencarian itemset-itemset yang sering muncul dan memudahkan pengguna untuk mengetahui hubungan subset-superset yang sangat berguna dalam melakukan proses selanjutnya yaitu tahap pembangkitan

kaidah asosiasi. Namun algoritma CHARM-L juga memiliki kendala dalam menentukan batasan *minimum_support* (*minsup*) yang sesuai. Dalam menentukannya dibutuhkan pengetahuan yang rinci tentang *mining query* dan tugas spesifikasi data, serta kemampuan untuk estimasi, tanpa harus melakukan pencarian kaidah asosiasi. Dalam menentukan nilai batasan *minimum_support* pada pencarian *frequent itemset*, bila ditetapkan terlalu kecil akan membawa pada pembangkitan ribuan itemset–itemset, dan bila terlalu besar sering tidak menghasilkan jawaban (itemset–itemset yang sesuai dengan batasan *minimum_support*) (Han, Pei, and Y. Yin, 2000).

Untuk mengatasi kendala tersebut beberapa peneliti mengusulkan sebuah model pencarian dengan Top-k yang digunakan untuk menentukan ruang pencarian dengan derajat tertentu (nilai derajat k, yang ditentukan oleh pengguna. Dimana k adalah sejumlah angka yang menentukan derajat kemunculan dari *frequent itemset* yang paling dicari atau yang diinginkan oleh pengguna). Dalam hal ini, menentukan nilai k dianggap lebih mudah dilakukan dibandingkan dengan menentukan batasan *minimum_support.* (Wang, Han, Lu and Tzvetkov, 2005).

1.2 Perumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan sebelumnya, perumusan masalah dalam penelitian ini berkaitan dengan upaya untuk mengembangkan algoritma pembangkitan kaidah asosiasi dari top-k frequent closed itemset lattice. Pada algoritma yang dikembangkan ini, diharapkan akan memberi solusi terhadap kendala yang ada pada algoritma CHARM_L yaitu; Sulitnya menentukan nilai batasan *minsup* yang tepat bagi pengguna pada pencarian itemset-itemset yang sering muncul, digantikan dengan menentukan nilai k untuk Top-k yang dinginkan oleh pengguna, maka ada peluang penelitian untuk melakukan perbaikan terhadap algoritma yang ada. Berdasar pada kendala-kendala tersebut maka pertanyaan penelitian (*research question*) dari penelitian ini adalah bagaimana merancang dan mengimplementasikan algoritma pembangkitan kaidah asosiasi dari top-k frequent closed itemsets berbasis struktur data lattice.

1.3 Tujuan dan Manfaat Penelitian

Tujuan penelitian ini adalah mendesain serta mengimplementasikan algoritma pembangkitan kaidah asosiasi dari top-k frequent closed itemset berbasis struktur data lattice.

Manfaat penelitian ini adalah memungkinkan pengguna dengan mudah untuk mendapatkan kaidah asosiasi dari top-k frequent closed itemset berbasis struktur data lattice, tanpa memberikan nilai minimum support.

1.4 Batasan Penelitian

Berdasarkan perumusan masalah yang ada dalam penelitian ini, maka ditentukan batasan penelitian sebagai berikut;

a. Algoritma yang dibuat hanya untuk dataset transaksi keranjang belanja atau *market basket*. Dan dataset transaksi yang digunakan berupa data biner.
b. Dataset transaksi yang digunakan berupa data dengan format horisontal untuk pencarian top-k frequent closed itemset lattice.
c. Pembentukan struktur data dari dataset yang ada akan disesuaikan dengan besar memory yang tersedia.
d. Penelitian ini hanya difokuskan pada kegiatan *Frequent Itemsets Mining* atau penggalian itemset-itemset yang sering muncul.

1.5 Sistematika Penulisan

Sistematika penulisan dalam penelitian ini diawali dengan Bab 1, membahas tentang latar belakang penelitian, perumusan masalah, tujuan penelitian, batasan penelitian, kontribusi dan manfaat penelitian.

Selanjutnya Bab 2, membahas tentang teori-teori dan konsep yang melandasi penelitian ini. Teori-teori dan konsep yang digunakan diambil dari pustaka atau tulisan tentang penelitian-penelitian yang pernah dilakukan sebelumnya, berkaitan dengan pembangkitan kaidah asosiasi dari top-k frequent closed itemset lattice.

Bab 3, membahas tentang desain algoritma pembangkitan kaidah asosiasi dari top-k frequent closed itemset yang didasarkan pada struktur data berbasis

lattice. Dalam bab ini akan dijelaskan algoritma pencarian *frequent closed itemset lattice* berdasar nilai Top-k dan algoritma pembangkitan kaidah asosiasi.

Bab 4, membahas tentang implementasi algoritma pembangkitan kaidah asosiasi dari frequent closed itemset lattice dengan pendekatan Top-k. Dan akan diulas proses ujicoba yang diawali dengan ulasan tentang lingkungan ujicoba. Dan kemudian penjelasan tentang data ujicoba yang digunakan serta skenario ujicoba yang akan dilakukan dilanjutkan dengan penjelasan pelaksanaan ujicoba diakhiri dengan analisis hasil ujicoba.

Sebagai penutup, Bab 5 membahas tentang simpulan dari analisis hasil uji coba penelitian dan saran pengembangan penelitian yang dapat dilakukan selanjutnya.

BAB 2
KAJIAN PUSTAKA

BAB 2
KAJIAN PUSTAKA

Berikut ini penjelasan tentang tinjauan kepustakaan yang digunakan dalam penelitian. Pembahasannya meliputi tentang konsep pencarian kaidah asosiasi secara umum pada sub bab 2.1, dilanjutkan dengan sub bab 2.2 yang membahas tentang pencarian frequent itemset, pada sub bab 2.3 akan dijelaskan tentang pencarian frequent closed itemsets, pada sub bab 2.4 dijelaskan tentang *Formal Concept Analysis* sebagai dasar teori lattice, pada sub bab 2.5 dijelaskan tentang pembangkitan frequent closed itemset lattice, pada sub bab 2.6 dijelaskan tentang pembangkitan Top-k frequent closed itemsets lattice.

2.1 Pencarian Kaidah Asosiasi secara Umum

Akhir–akhir ini, kemampuan sistem komputer dalam menghasilkan dan mengumpulkan data meningkat dengan pesat. Terlihat dari semakin banyaknya komputerisasi pada setiap transaksi bisnis dan pemerintahan, dan tersedianya perangkat keras penyimpan basis data yang dapat menyimpan data yang sangat besar sekali. Berjuta–juta basis data dihasilkan pada manajemen bisnis, administrasi pemerintahan, dan pada banyak aplikasi lainnya. Pesatnya perkembangan ukuran basis data dapat disebabkan karena kemampuan dari sistem basis datanya. Kondisi ini menimbulkan kebutuhan baru yang penting, yaitu : teknik baru yang melakukan proses transformasi dari basis data transaksional yang besar tersebut untuk mendapatkan informasi penting yang dibutuhkan. Sehingga Data Mining menjadi bahan riset yang penting sekarang ini.

Ketersediaan data sudah bukan hal yang sulit diperoleh lagi dewasa ini apalagi ditunjang dengan banyaknya kegiatan yang sudah dilakukan secara komputerisasi. Namun data ini seringkali diperlakukan hanya sebagai rekaman tanpa pengolahan lebih lanjut sehingga tidak mempunyai nilai guna lebih untuk keperluan masa mendatang. Analisa dari tiap koleksi data tersebut akan menghasilkan pengetahuan atau informasi, misalnya berupa pola dan kaidah asosiasi yang terjadi pada data. Pola dan kaidah asosiasi bisa terjadi pada berbagai jenis data baik data ekonomi, keuangan, kesehatan dan lain-lain. Penggalian

kaidah asosiasi mempunyai peranan penting dalam proses pengambilan keputusan. Tahapan besar dari proses *Data Mining* adalah mengidentifikasikan *frequent itemset* dan membentuk kaidah asosiasi dari itemset tersebut. Kaidah asosiasi digunakan untuk menggambarkan hubungan antar item pada tabel data transaksional.

Tapi semakin berkembangnya teknologi komputer di dunia industri, semakin pesat pula perkembangan ukuran data tabel transaksional yang dihasilkan. Dan pada data tabel transaksional yang besar (VLDB, Very Large Database) tersebut, proses pencarian frequent itemset sangatlah sulit. Dari kondisi tersebut, sudah banyak algoritma yang dibentuk untuk mencari kaidah asosiasi. Tetapi keterbatasan tetap saja ada. Keterbatasan yang paling mencolok adalah diperlukannya pembacaan basis data secara berulang yang mengurangi kinerja algoritma tersebut. Sehingga diperlukan suatu algoritma yang sangat efisien yang bisa meminimalisasi pembacaan basis data, sehingga bisa mengoptimasi waktu yang dibutuhkan.

Analisis kaidah asosiasi atau *association analysis* merupakan salah satu teknik dalam data mining, digunakan untuk mengungkapkan hubungan yang menarik (*interesting relationship*), yang tersembunyi dalam sekelompok data transaksional dalam jumlah yang sangat besar. Sebuah hubungan yang belum terungkap dapat direpresentasikan dengan sebuah kaidah asosiasi atau sekelompok itemset yang sering muncul. Analisis kaidah asosiasi dipelajari secara intensif untuk banyak kegunaan seperti, dibidang system rekomender, data bioinformatik, diaknosa pendukung keputusan, telekomunikasi, deteksi instruksi, web mining untuk analisis dokumen dan pengungkapan hubungan asosiasi pada data dalam jumlah besar juga berguna dalam bidang pemasaran. Dengan pengungkapan hubungan asosiasi yang menarik pada record transaksi bisnis, dapat membantu dalam realisasi desain katalog, promosi pemasaran, manajemen gudang, analisa keputusan dan manajemen bisnis.

Sebagai contoh yang paling sederhana pada kegiatan analisis kaidah asosiasi adalah analisis keranjang belanja atau *market basket analysis*, yang mempelajari kebiasaan-kebiasaan pembelian dari konsumen, dengan cara mencari kelompok-kelompok barang yang sering dibeli dalam sekali transaksi belanja.

Sebagai contoh; {Mie instan}→{Telur}, yang memiliki hubungan implikasi yang kuat antara penjualan mie instan dan telur. Yang mempunyai arti, banyak pelanggan yang membeli mie instan juga membeli telur. Informasi tersebut dapat meningkatkan penjualan bagi pihak penjual antara lain dengan strategi tata letak barang dalam supermarket atau strategi pemasaran silang produk dan kombinasi barang yang akan dikenakan potongan harga.

Analisis kaidah asosiasi sendiri dapat diklasifikasikan kedalam beberapa kategori berdasarkan pada kriteria yang berbeda. Berdasarkan pada tipe dari nilai yang dikandung dari sebuah kaidah, asosiasi dapat diklasifikasikan kedalam asosiasi *boolean* dan asosiasi *quantitative*. Sebuah asosiasi *boolean* dapat menunjukkan hubungan antara obyek-obyek yang bersifat kategorikal atau diskrit. Sebagai contoh; {Komputer}→{Software}, dan sebuah asosiasi *quantitative* merupakan sebuah asosiasi multidimensional yang melibatkan atribut-atribut numerik yang didiskritkan secara dinamis (pada contoh, umur dan pendapatan) dan dimungkinkan juga melibatkan atribut-atribut kategorikal. Sebagai contoh; umur{X, "30-34"} ∩ pendapatan{X."5jt-10jt"} → membeli{X, "LCD TV"}.

Dalam analisis asosiasi, akan dilakukan pencarian kaidah asosiasi atau *association rule mining* yang terdiri dari dua aktifitas utama. Yang pertama yaitu, menemukan *frequent itemset,* mencari kelompok-kelompok barang atau disebut dengan itemset yang sering muncul dan memenuhi sebuah batasan *minimum support (min_sup)* untuk seluruh transaksi-transaksi belanja yang terjadi. Dan aktifitas yang kedua, akan dicari sebuah kaidah asosiasi yang kuat (*strong rule*) dan memenuhi batasan *minimum confidence (min_conf)*.

Untuk lebih memudahkan pemahaman tentang analisis kaidah asosiasi, perlu diketahui tentang istilah-istilah dasar yang ada. Pada subbab berikut ini akan diulas mengenai hal tersebut.

2.1.1 Istilah Dasar dalam Analisis Kaidah Asosiasi

Dalam analisis kaidah asosiasi, kegiatan analisis akan diterapkan pada sebuah dataset transaksi. Ada beberapa cara untuk menyajikan sebuah dataset transaksi. Pilihan representasi dataset tersebut berdampak pada biaya I/O yang terlibat ketika mencari nilai *support count* dari kandidat item-item yang

dibangkitkan. Pada Tabel 2.1 terdapat dua cara representasi dataset yang berbeda untuk merepresentasikan transaksi-transaksi keranjang belanja atau *market basket transaction.* Dimana a,b,c,d,e merupakan item-item yang ada dalam sebuah transaksi, dan angka 1 sampai dengan 10 merupakan identifikasi transaksi yang terjadi. Penyajian dataset transaksi pada sisi sebelah kiri pada Tabel 2.1 merupakan *horizontal data layout,* yang diadopsi oleh banyak algoritma penggalian kaidah asosiasi, termasuk algoritma *Apriori.* Cara yang lain, adalah penyajian dataset transaksi yang dikenal dengan *vertical data layout* dengan menyimpan daftar *transaction-identifier* (TID-list) untuk setiap itemnya.

Support untuk tiap kandidat itemset pada *vertical data layout* didapatkan dengan mengambil irisan dari TID-list dengan tiap subset iitem-item tersebut. Panjang dari TID-list akan berkurang untuk ukuran itemset yang besar. Bagaimanapun juga, terdapat kendala pada pendekatan ini bahwa initial set dari TID-list kemungkinan terlalu besar untuk disesuaikan pada memory, sehingga memerlukan teknik tersendiri untuk melakukan kompresi TID-list.

Tabel 2.1 Sebuah format dataset Horizontal dan Vertical

TID	Items
1	a,b,e
2	b,c,d
3	c,e
4	a,c,d
5	a,b,c,d
6	a,e
7	a,b
8	a,b,c
9	a,c,d
10	b

a	b	c	d	e
1	1	2	2	1
4	2	3	4	3
5	5	4	5	6
6	7	8	9	
7	8	9		
8	10			
9				

Sumber: Kumar, 2000

Data dalam keranjang belanja atau *market basket dataset* dapat ditampilkan atau direpresentasikan kedalam bentuk format biner seperti tampak pada Tabel 2.2, dimana setiap baris pada tabel berkorepondensi dengan sebuah transaksi dan tiap kolom pada tabel berkorespondensi dengan sebuah item. Sebuah item dapat perlakuan sebagai variabel biner yang nilainya = 1, artinya jika

item tersebut ada dalam sebuah transaksi. Dan bila variabel biner nilainya = 0, artinya item tersebut tidak ada dalam transaksi. Karena kemunculan sebuah item didalam transaksi lebih penting, maka nilai variabel biner dari item tersebut bersifat *asymmetric*.

Tabel 2.2. Sebuah representasi biner 0/1 dari data market basket.

TID	Bread	Milk	Diapers	Beer	Eggs	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

Sumber: Kumar, 2000

Itemset dan *Support Count*. Bila $I = \{i_1, i_2, i_3, \ldots, i_d\}$ merupakan set dari semua item- item dalam data market basket dan $T = \{t_1, t_2, t_3, \ldots, t_N\}$ merupakan set dari semua transaksi–transaksi. Setiap transaksi t_i, berisi sebuah subset dari item–item yang dipilih dari I. Dalam analisis kaidah asosiasi, sebuah koleksi dari nol atau lebih item–item dianggap sebuah itemset. Jika sebuah itemset yang berisi k item–item, maka item tersebut disebut sebagai k-itemset. Sebagai contoh, {Mie instan, Telur, Minyak} merupakan contoh sebuah 3-itemset. Null itemset atau itemset kosong adalah bila sebuah itemset tersebut tidak memiliki item.

Panjang sebuah transaksi atau *transaction width* didefinisikan sebagai jumlah item – item yang ada dalam sebuah transaksi. Sebuah transaksi t_j dikatakan berisi sebuah itemset X jika X adalah sebuah subset dari t_j. Bagian yang paling penting dari sebuah itemset adalah *support count* (frekuensi kemunculan), yang dijadikan sebagai acuan jumlah transaksi yang berisi sebuah itemset yang ada. Secara matematis *support count* disimbolkan dengan $\sigma(X)$, untuk sebuah itemset X dapat dinyatakan sebagai berikut; $\sigma(X) = |\{ t_i \mid X \subset t_i, t_i \in T\}|$, dimana simbol $|\,.\,|$ adalah lambang jumlah elemen–elemen dalam sebuah set. *Support count* untuk {Mie instan, Telur, Minyak }= 2 artinya, hanya terdapat 2 transaksi yang berisi ketiga item tersebut.

Kaidah asosiasi merupakan sebuah ekspresi implikasi dari $X \rightarrow Y$, dimana X dan Y itemset yang disjoint, $X \cap Y = 0$. Kekuatan dari sebuah kaidah asosiasi dapat diukur dengan *support* dan *confidence*. Dimana *support* menjelaskan seberapa sering sebuah kaidah atau rule dapat diterapkan pada dataset, sementara *confidence* merupakan penjelasan seberapa frequent item – item dalam *Y* muncul dalam transaksi yang berisi *X*. Definisi formal dari pengukuran tersebut adalah sebagai berikut; *Support*, $s\ (X \rightarrow Y) = \sigma(X \cup Y)/N$;

Confidence, $c\ (X \rightarrow Y) = \sigma(X \cup Y)/\sigma(X)$.

Sebagai contoh kaidah {Milk, Diapers}→{Beer}, dengan sebuah tabel transaksi, seperti yang tampak pada tabel 2.3 berikut ini;

Tabel 2.3. Transaksi – transaksi pada analisis keranjang belanja.

TID	*Items*
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}

Sumber (Kumar, 2000)

Maka untuk menghitung support sebagai berikut;

Support, $s\ (X \rightarrow Y) = \sigma(X \cup Y)/N$

Support, $s\ (X \rightarrow Y)$ = {Milk, Diapers, Beer}/ N (jumlah transaksi)

Support, $s\ (X \rightarrow Y) = 2/5 = 0{,}4$

Dan untuk menghitung confidence sebagai berikut;

Confidence, $c\ (X \rightarrow Y) = \sigma(X \cup Y)/\sigma(X)$

Confidence, $c\ (X \rightarrow Y)$ = {Milk, Diapers, Beer}/{Milk, Diapers}

Confidence, $c\ (X \rightarrow Y) = 2/3 = 0{,}67$

2.1.2 Penggunaan Support dan Confidence

Mengapa menggunakan *support* dan *confidence*? *Support* merupakan pengukuran yang sangat penting, karena sebuah rule yang memiliki *support* yang rendah kemungkinan kemunculannya dianggap sebagai kebetulan. *Support* selain

digunakan untuk menghilangkan rule–rule yang tidak menarik, juga dianggap sebagai bagian yang mampu mengeksploitasi efisiensi pengungkapan kaidah asiosiasi. Sedangkan *Confidence* merupakan pengukuran atas *reliability* (hal yang tahan uji atau dapat dipercaya) dari keterkaitan hubungan yang ada pada sebuah kaidah. Untuk $X \rightarrow Y$, dengan nilai *confidence* yang tinggi, maka pola tersebut lebih kuat keberadaannya dalam transaksi yang berisi X. *Confidence* juga menyediakan sebuah estimasi kondisi probabilitas dari Y terhadap X.

Untuk pengungkapan kaidah asosiasi pada sebuah set dari transaksi–transaksi *T*, pencarian kaidah dilakukan dengan membatasi pada set yang memiliki *support* $\geq$ *min_support* dan *confidence* $\geq$ *min_confidence*. Dengan menggunakan pendekatan secara *brute-force*, pencarian kaidah–kaidah asosiasi dilakukan dengan menghitung *support* dan *confidence* untuk seluruh transaksi yang ada. Pendekatan ini dianggap terlalu mahal, karena kaidah yang diekstrak dari dataset dapat berkembang secara eksponensial. Proses ini akan lebih efektif jika dilakukan pemangkasan/pengurangan kaidah (*pruning rule*) diawal proses, sebelum dilakukan perhitungan nilai *support* dan *confidence*. Sebagai ilustrasi, sebuah itemset pada Tabel 2.3 {Beer, Diaper, Milk}, maka dihasilkan kaidah yang akan memiliki *support* yang sama. Dan menghasilkan 6 buah kaidah-kaidah asosiasi sebagai berikut;

{Beer, Diaper}→{Milk}; {Diaper, Milk}→{Beer}; {Milk}→{ Beer, Diaper };
{Beer, Milk}→{Diaper}; {Beer }→{ Diaper, Milk}; {Diaper}→{Beer, Milk};

Untuk melakukan pemangkasan/pengurangan kaidah (*pruning rule*), diberlakukan sebuah aturan sebagai berikut; Jika sebuah itemset tidak sering muncul (*infrequent*) maka kandidat kaidah asosiasi (pada contoh terdapat enam kandidat) tersebut akan dipangkas secara cepat tanpa harus menghitung nilai *confidence* dari kaidah tersebut.

Dari pentingnya nilai *support*, seperti yang telah dijelaskan sebelumnya dapat disimpulkan bahwa kegiatan penggalian itemset yang sering muncul menjadi kegiatan yang sangat menentukan keberhasilan dari kegiatan analisis kaidah asosiasi. Pada subbab berikut ini akan dijelaskan tentang kegiatan *Frequent Itemset Mining*.

2.2 Pencarian Frequent Itemset

Algoritma penggalian pola yang sering muncul atau *frequent itemset mining algorithm* dalam analisa kaidah asosiasi secara umum menggunakan sebuah batasan *minimum support* untuk dapat menghasilkan set dari itemset–itemset yang sering muncul secara lengkap dan benar. Hal tersebut didasari oleh algoritma yang sangat populer, yaitu algoritma *Apriori*. Formulasi permasalahan pada penggalian kaidah asosiasi (*mining association rule*) diawali dengan langkah formal yang harus dilalui sebagai berikut; pada sebuah set *T* yang berisi transaksi – transaksi, akan dicari semua kaidah–kaidah yang memiliki *support* (frekuensi kemunculan) ≥ nilai batasan *minimum support* (*min_sup*) dan *confidence* (pola yang kuat) ≥ nilai batasan *minimum confidence* (*min_conf*), dimana *minimum support* serta *minimum confidence* berhubungan dengan nilai batasan *support* dan *confidence* yang ditentukan oleh pengguna.

Dengan menggunakan pendekatan *brute-force* untuk penggalian kaidah asosiasi dengan menghitung support dan confidence untuk setiap kaidah yang mungkin muncul, dianggap terlalu mahal karena kaidah yang muncul terlalu banyak atau kemunculannya secara eksponensial pada saat diekstrak dari sebuah data set. Dan bila telah dikenakan batasan support–confidence maka akan menghasilkan sedikit kaidah atau *rule*, tahap sebelumnya dianggap melakukan hal yang sia–sia. Maka diambil keputusan untuk memangkas kaidah–kaidah yang dibangkitkan sebelumnya, sebelum dilakukan perhitungan nilai support–confidence dari itemset tersebut.

Namun terdapat kendala pada kinerja algoritma apriori, kinerjanya akan turun drastis bila diaplikasikan pada data set yang rapat dan memiliki pola – pola yang panjang. Dan untuk menentukan batasan *minimum support* yang sesuai, dibutuhkan pengetahuan yang rinci tentang *mining query* dan tugas spesifikasi data, serta kemampuan untuk estimasi, tanpa harus melakukan pencarian, berapa banyak itemset–itemset yang akan dibangkitkan dengan penetapan batasan saat ini. Menentukan nilai batasan *minimum support* bila terlalu kecil akan membawa pada pembangkitan ribuan itemset–itemset, dan bila terlalu besar sering tidak menghasilkan jawaban (itemset–itemset yang sesuai dengan batasan *minimum support*).

Berawal dari sebuah algoritma bernama algoritma Apriori inilah, merupakan algoritma pertama untuk pencarian itemset yang sering muncul pada *boolean association rules*. Nama algoritma tersebut berasal dari kata dasar "prior" yang artinya "lebih dahulu" atau "sebelumnya". Bahwa algoritma tersebut menggunakan pengetahuan yang sebelumnya atau *prior knowledge* dari properti itemset yang sering muncul atau *frequent itemset*. Algoritma Apriori menggunakan metode pemangkasan berdasar nilai *support* yang secara sistematis untuk mengontrol pertumbuhan secara eksponensial kandidat-kandidat itemset. Algoritma Apriori menggunakan sebuah pendekatan iteratif yang dikenal dengan *level-wise search*, dimana k-itemset digunakan untuk mencari (k+1)-itemset. Apabila set dari 1-itemset ditemukan, akan dikenali sebagai C_1. C_1 tersebut digunakan untuk mencari 2-itemset atau disebut dengan C_2 dan seterusnya sampai tidak ditemukan lagi k-itemset yang sering muncul. Dan untuk mencari setiap C_k membutuhkan satu kali penuh pembacaan seluruh transaksi pada basis data.

Untuk meningkatkan efisiensi pembangkitan *level-wise* dari itemset yang sering muncul, algoritma Apriori menggunakan sebuah property penting yang disebut *Apriori Property* yaitu, 'setiap subpola dari sebuah pola yang sering muncul, pasti bersifat sering muncul pula' (biasa disebut dengan *Downward Closure Property*). Hal ini didasarkan pada observasi berikut ini, jika sebuah itemset *I* tidak memenuhi batasan nilai *minimum_support*, maka *I* tidak sering muncul. Sifat ini digunakan untuk mengurangi ruang pencarian atau *search space* kandidat itemset yang sering muncul.

Berikut ini adalah penjelasan Algoritma Apriori dengan menggunakan sebuah contoh basisdata transaksi yang berisi sebuah set transaksi *T*={*T1, T2, T3, T4*}dan itemset *I*={*I1, I2, I3, I4, I5*}. Dimana pada transaksi T1={*I1, I2, I5*}, T2={*I2, I3, I4*}, T3={*I3, I4*}, dan T4={*I1, I2, I3, I4*}. Dan mengacu pada Algoritma Apriori, berikut ini terdapat dua variabel yang digunakan yaitu variabel Ck dan Fk. Dimana Ck (*Candidate*) adalah variabel untuk set dari kandidat k-itemset dan Fk (*Frequent*) adalah variabel untuk k-itemset yang sering muncul. Algoritma tersebut diawali dengan inisialisasi dataset untuk memberikan nilai support untuk setiap item. Kemudian, set semua 1-itemset, F1 akan diketahui (pada langkah 1 dan 2).

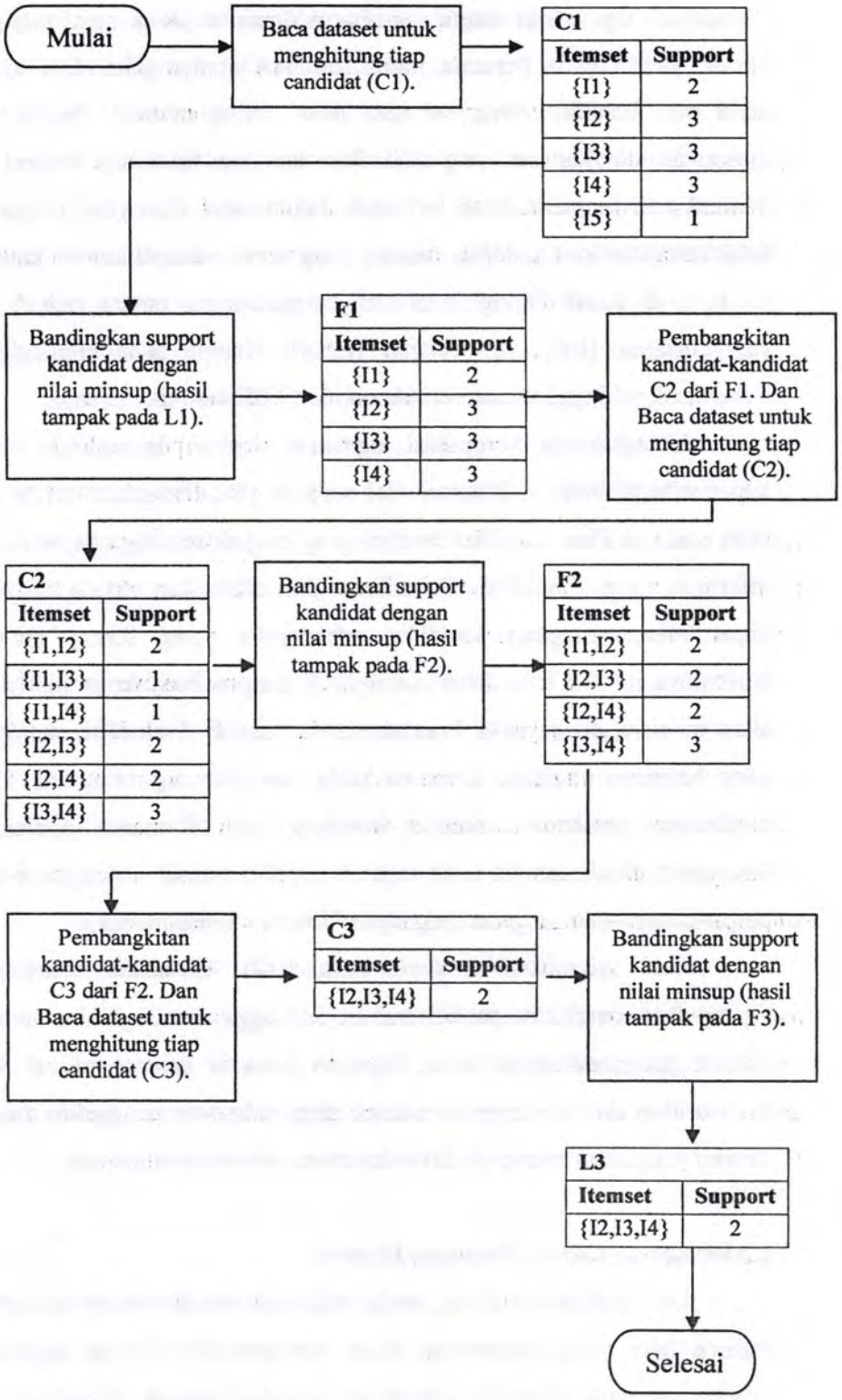
Algoritma Apriori : untuk pembangkitan frequent itemset.

```
1:  k = 1.
2:  Fk = {i | i ∈ I ∩ σ ({i}) ≥ N x minsup}. //cari semua 1-itemset yang sering muncul
3:  repeat
4:    k = k+1.
5:    Ck = Apriori-gen(Fk-1).     // pembangkitan candidat itemset
6:    for tiap transaksi t ∈ T do
7:    Ct = subset(Ck, t).         // identifikasi semua kandidat yang ada pada t
8:       for tiap kandidat itemset c ∈ Ct do
9:          (c) = σ (c) + 1.     // menambah support count
10:      end for
11:   end for
12:      Fk = {c | c ∈ Ck ∩ σ (c) ≥ N x minsup}. //ekstrak k-itemset yang sering muncul
13:   until Fk = 0
14:   Result = U Fk.
```

Sumber : Kumar (2000)

Algoritma tersebut akan berjalan secara iteratif untuk membangkitkan kandidat baru k-itemset menggunakan *frequent* (k-1)-itemset, yang ditemukan pada iterasi sebelumnya (pada langkah 5). Untuk menghitung support dari kandidat-kandidat yang ada algoritma ini melakukan perulangan (pada langkah 6-11). Fungsi subset digunakan untuk menyebutkan semua kadidat itemset pada Ck yang termasuk dalam tiap transaksi t. Setelah menghitung supportnya, algoritma ini akan menghapus semua kandidat itemset yang jumlah supportnya kurang dari nilai minsup yang ditetapkan sebelumnya. (pada langkah 12). Algoritma ini aka berhenti ketika tidak ada lagi itemset yang sering muncul yang dapat dibangkitkan. Fk = 0. (pada langkah 13). Untuk memudahkan pemahaman dapat dibantu dengan Gambar 2.1.

Pada algoritma ini, untuk tahap pembangkitan kandidat itemset terdapat dua proses penting antara lain proses pembangkitan itu sendiri dan proses pemangkasan. Untuk proses pembangkitan, pada tahap ini akan dibangkitkan k-itemset berdasarkan pada (k-1)-itemset yang sering muncul yang telah ditemukan sebelumnya. Dan untuk proses pemangkasan, pada tahap ini akan dihapus kandidat-kandidat k-itemset dengan strategi berdasarkan nilai support atau *support-based strategy*. Maka dapat disimpulkan bahwa hanya k-itemset yang memiliki nilai support besar, dianggap sering muncul. Dan ini dianggap sangat efektif untuk mengurangi jumlah kandidat itemset yang sering muncul.



Gambar 2.1 Diagram alir algoritma Apriori.

Secara prinsip ada banyak cara untuk pembangkitan kandidat itemset. Namun ada tiga syarat utama yang harus dipenuhi untuk pembangkitan kandidat itemset yang efektif. Pertama, harus dihindari pembangkitan kandidat yang tidak perlu atau bersifat *infrequent* atau tidak sering muncul. Kedua, harus dapat dipastikan kandidat set yang dihasilkan lengkap, tidak ada itemset yang sering muncul atau *frequent*, tidak termasuk didalamnya. Dan yang ketiga, seharusnya tidak menghasilkan kandidat itemset yang sama, sebagai contoh kandidat itemset {a, b, c, d} dapat dibangkitkan oleh penggabungan antara {a,b,c} dengan {d}, {a,c} dengan {b,d}, {c} dengan {a,b,d}. Karena akan meningkatkan waktu komputasi, sehingga tujuan semula untuk efektifitas tidak tercapai.

Kompleksitas komputasi algoritma Apriori dipengaruhi oleh beberapa faktor sebagai berikut. Batasan nilai support, jika ditetapkan terlalu rendah maka akan menghasilkan kandidat itemset yang banyak sehingga dapat menghilangkan informasi yang diinginkan. Sebaliknya jika ditetapkan terlalu tinggi maka tidak dapat mengungkapkan kandidat sebenarnya yang sering muncul. Faktor berikutnya jumlah item (*dimensionality*), dengan banyaknya jumlah item maka akan meningkatkan biaya komputasinya. Jumlah Transaksi, merupakan faktor yang berpengaruh juga, terutama pada *run time* algoritma ini. Karena harus melakukan pembacaan dataset berulang kali. Rata-rata panjang transaksi, khususnya untuk dataset yang rapat (memiliki transaksi dengan pola-pola yang panjang) dan akan sangat mempengaruhi waktu komputasinya.

Dari Algoritma Apriori yang telah dijelaskan sebelumnya, dapat disimpulkan masih terdapat kelemahan. Sehingga memungkinkan untuk dilakukan metode penggembangan untuk kegiatan *frequent itemset mining*. Pada subbab berikut akan diulas mengenai metode pengembangan penggalian *frequent closed itemset* yang akan memperbaiki kelemahan metode sebelumnya.

2.3 Pencarian Closed Frequent Itemset

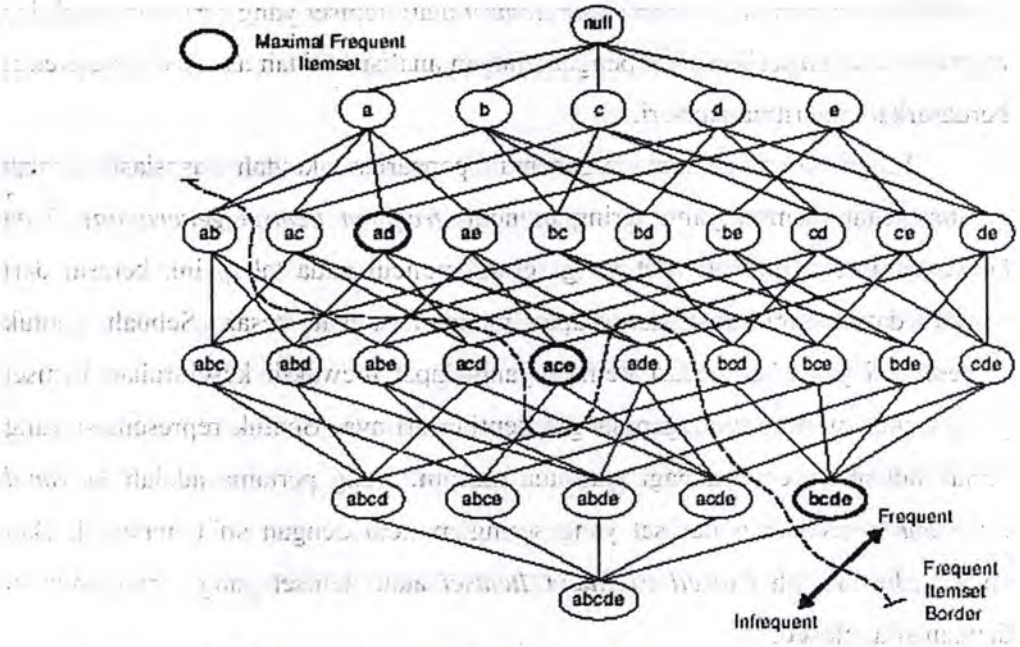
Dari penjelasan diatas, maka sejumlah peneliti mengembangkan metode-metode baru yang diharapkan dapat memperbaiki kinerja algoritma Apriori. Difokuskan pada tahap pembangkitan frequent itemset, dianggap sebagai tahap paling menentukan efektifitas dari sebuah analisis kaidah asosiasi. Berikut ini

dijelaskan mengenai *closed frequent itemset* atau itemset yang sering muncul dan memiliki sifat *closed*, sebagai pengembangan analisis kaidah asosiasi yang efektif berdasarkan algoritma Apriori.

Tahap awal dalam algoritma pencarian kaidah asosiasi adalah pembangkitan itemset yang sering muncul (*frequent itemset generation*). Pada kenyataannya, jumlah itemset yang sering muncul pada tahap ini, berasal dari sebuah dataset transaksi dan dapat menjadi sangat besar. Sebuah bentuk representasi yang kecil, dari itemset yang dapat mewakili keseluruhan itemset yang sering muncul dianggap sangat penting artinya. Bentuk representasi yang dimaksudkan tersebut terbagi atas dua macam. Yang pertama adalah *Maximal Frequent Itemset* atau itemset yang sering muncul dengan sifat maximal. Dan yang kedua adalah *Closed Frequent Itemset* atau itemset yang sering muncul dengan sifat closed.

Yang pertama, *Maximal Frequent Itemsets*, didefinisikan sebagai jika tidak ada satupun superset itemset terdekatnya yang frequent.. Maximal fréquent itemsets secara efektif dalam menyediakan representasi yang padat dari *frequent itemset*, dengan kata lain, *maximal frequent* adalah set terkecil dari itemset yang sering muncul. Pada Gambar 2.2. mengilustrasikan *maximal frequent itemset* pada struktur lattice. Item–item pada lattice dibagi menjadi dua kelompok; yang pertama adalah kelompok yang sering muncul dan kedua adalah kelompok yang *infrequent*. Sebuah batas untuk itemset yang sering muncul, diatas garis putus–putus adalah kelompok itemset yang sering muncul dan untuk dibawah garis putus-putus adalah kelompok untuk *infrequent* itemset dan tampak {a,d}, {a,c,e}, {b,c,d,e}merupakan itemset yang *maximal fréquent* karena superset yang mereka miliki bersifat *infrequent*. Sebaliknya, {a,c}bukan maximal frequent karena satu dari superset,{a,c,e}sering muncul.

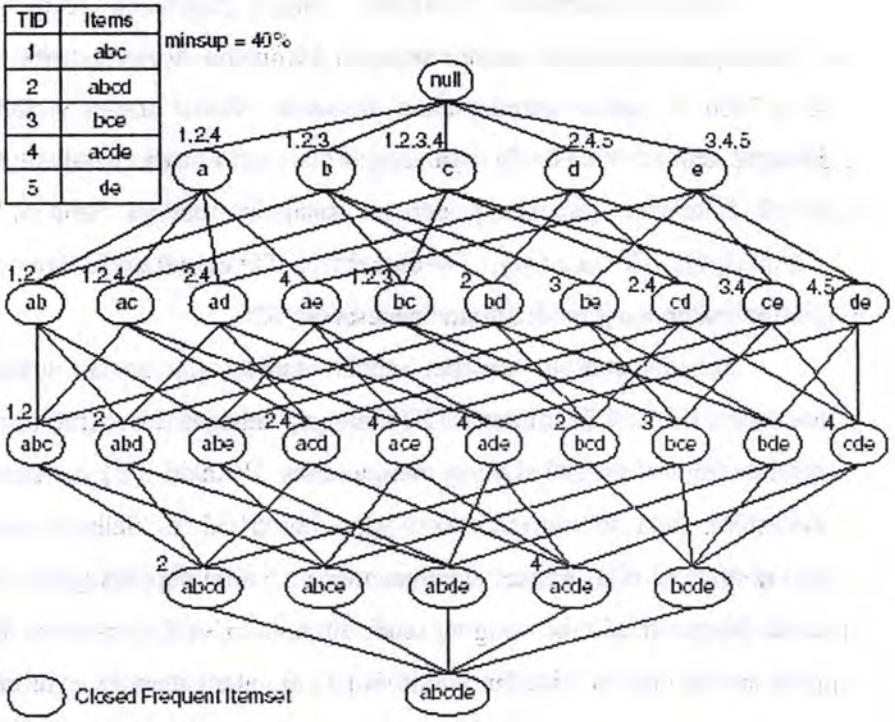
Maximal frequent itemset, dianggap sebagai representasi efektif untuk itemset yang sering muncul. Dengan kata lain, *maximal frequent itemset* adalah himpunan terkecil yang mewakili seluruh itemset yang sering muncul. Namun dalam representasi dengan bentuk *maximal frequent itemset* ini, tidak berisi informasi support dari subsetnya.



Gambar 2.2. Maximal Frequent Itemset pada Struktur Lattice

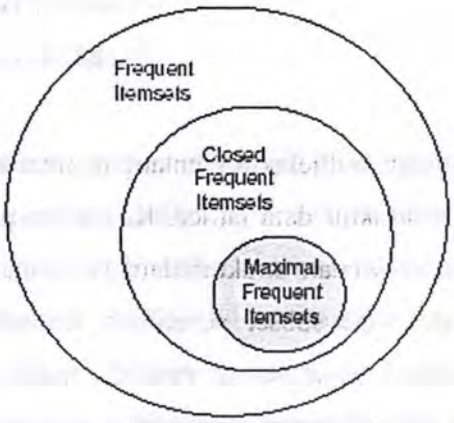
Sehingga dibutuhkan sebuah representasi yang lebih baik selain dapat mewakili seluruh itemset yang sering muncul, juga dapat memberikan informasi support dari subsetnya. Hal ini dapat dipenuhi oleh bentuk representasi *Closed Frequent Itemset*. Berawal dari *closed itemset* yang menyediakan sebuah representasi minimal dari itemset – itemset tanpa kehilangan informasi support itemset tersebut. Definisi *closed itemset* adalah sebuah itemset X dianggap *closed* jika tidak ada *superset–superset* saat ini yang memiliki *support count* yang tepat sama dengan X. Dan diilustrasikan pada Gambar 2.3.

Maka *Closed Frequent Itemset* dapat didefinisikan sebagai sebuah itemset X yang sering muncul bersifat *closed* dan support-nya lebih besar atau sama dengan minimum support. *Closed frequent itemset* sangat berguna untuk menghilangkan kaidah asosiasi yang ganda atau *redundant*. Dan digunakan untuk proses *rule generation* atau pembangkitan kaidah. *Maximal frequent itemset* pasti bersifat *closed* karena tidak memiliki support count yang sama dengan supersetnya saat ini.



Gambar 2.3. Closed Frequent Itemset pada Struktur Lattice.

Hubungan antara *frequent itemset, maximal dan closed frequent itemset* dapat dilihat pada Gambar 2.4 berikut ini;



Gambar 2.4 Hubungan antara *frequent itemset*, *maximal frequent itemset* dan *closed frequent itemset*.

Dalam Algoritma CHARM, sebagai algoritma pengembangan yang didasarkan pada algoritma sebelumnya (Algoritma Apriori) memiliki sebuah cara yang efisien untuk mendapatkan *frequent closed itemset*. Sebagai ilustrasi, terdapat sebuah tabel basis data yang berisi enam buah transaksi. Masing-masing berisi 5-itemset {a,c,d,t,w} dengan susunan sebagai berikut. T1={a,c,t,w}, T2={c,d,w}, T3={a,c,t,w}, T4={a,c,d,w}, T5={a,c,d,t,w}, T6={c,d,t}. Dengan nilai minsupp yang telah ditentukan sebesar 50%.

Sebagai contoh, terdapat sebuah itemset {a,c,w}dan sebuah *transaction identifier*, tid {2,4,5}, dimana t(X) dianggap sebagai tidset (merupakan himpunan seluruh tid dari transaksi yang mengandung X). Dan i(Y) merupakan himpunan item-item pada seluruh transaksi yang berisi tid Y. Sebuah itemset dianggap sering muncul bila itemset tersebut memiliki nilai support yang lebih besar atau sama dengan nilai minsup yang telah ditentukan, $\sigma(X) \geq$ minsup. Sebuah itemset yang sering muncul bersifat maximal jika frequent itemset tersebut bukan subset dari frequent itemset manapun. Dan sebuah frequent itemset bersifat closed jika keberadaanya bukan *proper superset* $Y \supset X$ dengan $\sigma(X) = \sigma(Y)$.

$$
\begin{aligned}
c(a,c,w) &= i(t(a,c,w)) \\
&= i(t(a) \cap t(c) \cap t(w)) \\
&= 1345 \cap 123456 \cap 12345 \\
&= i(1345) = acw \ (\textit{closed})
\end{aligned}
$$

Berikutnya akan dijelaskan tentang pembangkitan *frequent closed itemset* berdasarkan pada struktur data lattice. Karena pada tahap pembangkitan kaidah asosiasi dibutuhkan sebuah struktur data yang mampu mengambarkan dengan jelas hubungan superset-subset sebuah itemset. Selama ini algoritma untuk pencarian itemset yang sering muncul, hanya menghasilkan keluaran dalam bentuk daftar yang dianggap menyulitkan pengguna untuk menentukan hubungan superset-subset sebuah itemset. Namun sebelumnya akan diulas dasar-dasar konsep struktur data lattice dari FCA (*Formal Concept Analysis*) yang digunakan untuk menggambarkan relasi-relasi yang ada dalam kaidah-kaidah asosiasi.

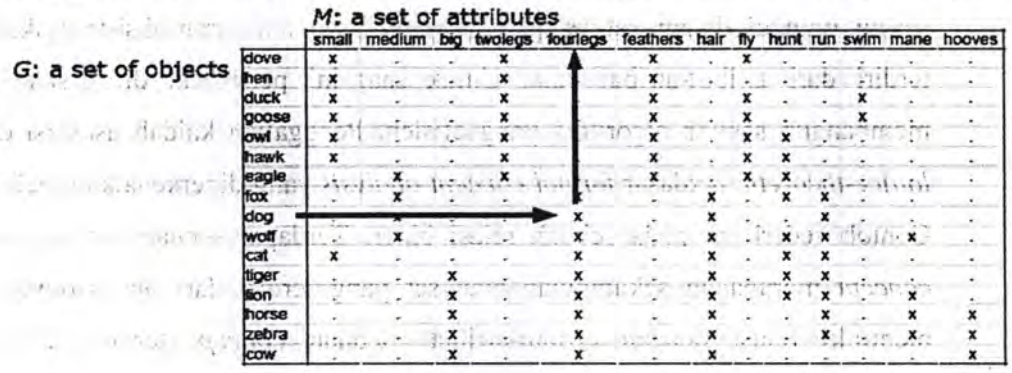
2.4 Formal Concept Analisis

Pencarian kaidah asosiasi, merupakan sebuah tugas penggalian informasi yang penting, dimana kegiatan tersebut untuk mengungkap semua pola-pola yang sering muncul dalam set-set (*Frequent Itemset*) atau transaksi-transaksi yang terdiri dari atribut-atribut data. Untuk saat ini penelitian difokuskan untuk membangun sebuah algoritma yang efisien. Penggalian kaidah asosiasi dengan *lattice-theoretic* berdasar *formal concept analysis* yang diperkenalkan oleh Wille. Contoh sederhana pada sebuah relasi biner. Terdapat sebuah *concept* dimana *concept* merupakan sekumpulan atau set yang terdiri dari obyek-obyek yang memiliki relasi dengan atribut-atribut. Sebuah *concept* terdiri dari *extent* (transaksi-transaksi) dan sebuah *intent* (atribut-atribut), sehingga semua obyek-obyek dalam *extent*/transaksi dapat berbagi *intent*/atribut dan *vice versa*. Dengan menunjukkan semua *frequent itemsets* secara unik akan dibentuk sebuah *set frequent concepts*. *Concept lattice* selain dapat menghasilkan algoritma yang efisien juga dapat membantu visualisasi dalam pencarian kaidah asosiasi yang kuat dan dapat memberikan *framework* untuk *reasoning* tentang asosiasi, dan dapat digunakan untuk teknik *classification* dan *clustering* pada *concept learning*.

Gambar 2.5 Ilustrasi sebuah *Concept* dalam *Formal Concept Analysis*

Dalam *concept* yang terdiri dari sekelompok obyek dan atribut akan terkait dalam *context*-nya, seperti tampak pada Gambar 2.5. Kelompok-kelompok ini dan relasi-relasi yang menghubungkan kelompok tersebut menjadi dasar dari simpulan-simpulan yang kita buat. Semua *concept* yang kita buat dan implikasi apapun yang kita buat semua berdasarkan pada *context*. Mengubah *context* akan mengubah *concept-concept* dan juga akan mengubah strukturnya. Formal Context merupakan sebuah group yang terdiri dari (*G, M, I*) dimana *G* mewakili sebuah

himpunan dari obyek-obyek, dan *M* mewakili sebuah himpunan dari atribut-atribut dan sebuah relasi diwakili oleh *I*. Ada banyak cara untuk mengubah data yang rumit kedalam bentuk *formal context*. Contoh pada Gambar 2.6 berikut ini.



Sumber: Bastian Wormuth 2004

Gambar 2.6 Matrix untuk mengubah data dalam bentuk formal context

Pada Gambar 2.6 merupakan sebuah matrix sebagai salah satu cara untuk mengubah data kedalam bentuk *formal context*. Mengubah matrik dengan cara mentransformasi bentuknya akan menghasilkan implikasi-implikasi dan relasi-relasi yang sama. Sekarang akan diulas tentang definisi *Formal Concept* yang berkaitan dengan himpunan-himpunan data atau *data sets*. Secara matematika sebuah *Formal Concepts* akan disimbolkan dengan operator-operator (contoh : operator '), sehingga *Formal Concepts* didefinisikan sebagai berikut; Untuk sebuah himpunan obyek-obyek A, A' didefinisikan sebagai A' = (semua atribut-atribut dalam *M* dipakai bersama oleh himpunan dari obyek-obyek dari A). Untuk sebuah himpunan obyek-obyek B, B' didefinisikan sebagai B' = (semua himpunan dari obyek-obyek dalam *G* memiliki semua atribut-atribut yang dimiliki oleh B).

Sebuah pasangan dari kelompok-kelompok obyek dan atribut (A,B) yang memenuhi kondisi A'= B dan B'= A disebut sebagai sepasang *Formal Concept*. Bagaimana menghasilkan sebuah Formal Concept, berikut ini akan diulas menggunakan matrik;

	small	medium	big	twolegs	fourlegs	feathers	hair	fly	hunt	run	swim	mane	hooves
dove	x	.	.	x	.	x	.	x	.	.	.	.	.
hen	x	.	.	x	.	x	.	.	.	.	.	.	.
duck	x	.	.	x	.	x	.	x	.	.	x	.	.
goose	x	.	.	x	.	x	.	x	.	.	x	.	.
owl	x	.	.	x	.	x	.	x	x	.	.	.	.
hawk	x	.	.	x	.	x	.	x	x	.	.	.	.
eagle	.	x	.	x	.	x	.	x	x	.	.	.	.
fox	.	x	.	.	x	.	x	.	x	x	.	.	.
dog	.	x	.	.	x	.	x	.	.	x	.	.	.
wolf	.	x	.	.	x	.	x	.	x	x	.	x	.
cat	x	.	.	.	x	.	x	.	x	x	.	.	.
tiger	.	.	x	.	x	.	x	.	x	x	.	.	.
lion	.	.	x	.	x	.	x	.	x	x	.	x	.
horse	.	.	x	.	x	.	x	.	.	x	.	x	x
zebra	.	.	x	.	x	.	x	.	.	x	.	x	x
cow	.	.	x	.	x	.	x	.	.	.	.	.	x

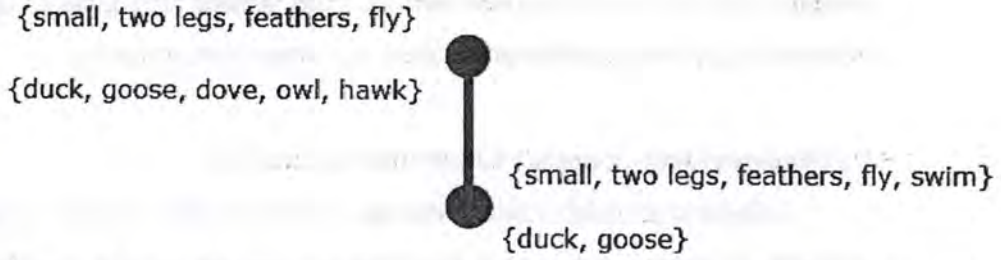
Langkah pertama: pilih kelompok manapun dari obyek-obyek yang ada pada A, misalnya a={duck}.

Langkah kedua: tentukan atribut untuk A' = {small, two legs, feathers, fly, swim}.

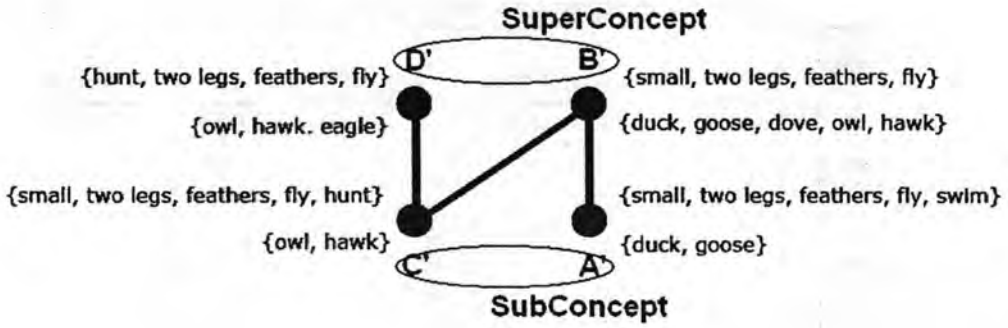
Langkah ketiga: tentukan (A')' = {small, two legs, feathers, fly, swim}' = {duck, goose}.

Langkah keempat: menentukan sebuah formal concept (A", A') = ({duck, goose}, {small, two legs, feathers, fly, swim}).

Untuk mengGambarkan formal concept dari (A", A') = ({duck, goose}, {small, two legs, feathers, fly, swim}) dengan sebuah diagram garis dan titik atau graph akan tampak sebagai berikut:



Dan untuk bentuk formal concept yang lain (B", B') = ({duck, goose, dove, owl, hawk},{small, two legs, feathers, fly}). Dan dapat disimpulkan (A",A') sebagai *subconcept* dari (B",B') dan (B",B') disebut sebagai *superconcept* dari (A",A'). Hubungan *subconcept* dan *superconcept* mendefinisikan sebuah kelompok B yang terurut dari semua *formal concepts* dari sebuah *formal context*. *Formal concept analysis* inilah yang mendasari teori lattice pada pencarian kaidah asosiasi.



Dari sekelompok *frequent concepts* inilah yang akan memunculkan semua frequent itemsets. Lattice dari *frequent concepts* dapat juga digunakan untuk menghasilkan sekelompok kaidah dari semua asosiasi yang ditentukan. Dalam usaha pencarian *frequent concept* atau *concept* yang sering muncul terdapat dua permasalahan yang harus diselesaikan oleh sebuah algoritma pencarian *frequent concept*. Permasalahan pertama, bagaimana sebuah algoritma dapat menghasilkan semua *concept* yang memungkinkan (kuantitatif) dan persoalan yang kedua adalah bagaimana sebuah algoritma dapat meminimalkan perulangan *concept* yang dihasilkan (kualitatif).

Dari penjelasan mengenai *formal concept analysis* sebagai dasar teori struktur data lattice ini, akan digunakan sebagai acuan perancangan algoritma penggalian top-k frequent closed itemset. Pada subbab berikut akan diulas lebih lanjut tentang proses pembangkitan frequent closed itemset lattice.

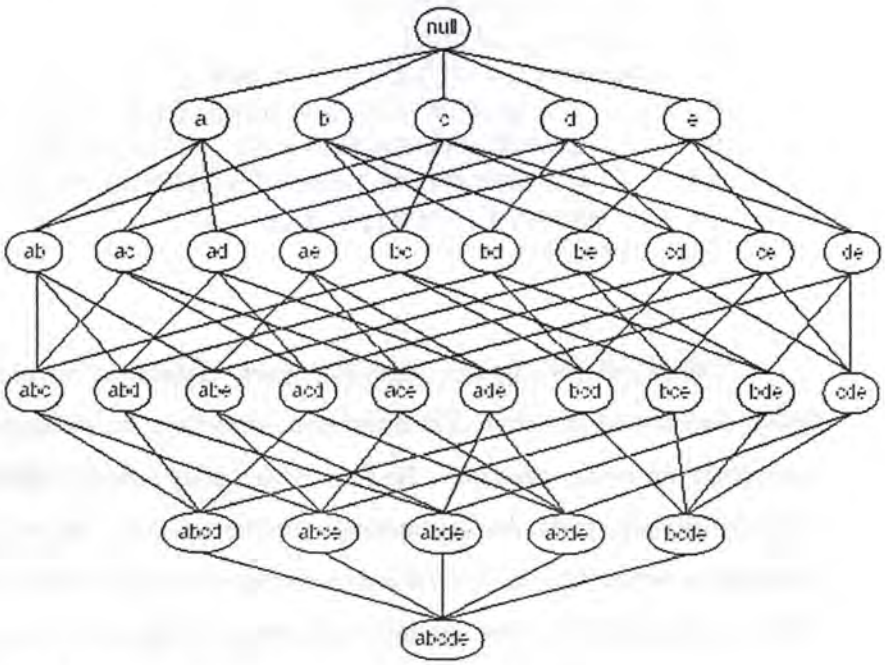
2.5 Pembangkitan Frequent Closed Itemset Lattice

Sebelumnya telah diulas tentang *closed frequent itemset* yang mampu mengurangi ruang pencarian kandidat itemset yang sering muncul. Dan dianggap sebagai bentuk representasi *frequent itemset* yang paling efisien, karena dapat sekaligus memuat informasi support kandidat frequent itemset. Untuk sub bab ini, akan diulas tentang bentuk keluaran yang dihasilkan pada tahap pembangkitan frequent itemset, yaitu dalam bentuk lattice.

Semua algoritma pencarin *closed frequent itemset* yang ada saat ini tidak menghasilkan keluaran berupa lattice secara eksplisit. Keluaran algoritma-algoritma tersebut hanya berupa daftar semua *closed frequent itemset* yang

ditemukan. Hal ini dianggap menyulitkan pengguna dalam pemahaman hubungan subset-superset dari itemset yang ditemukan, pada tahap pembangkitan kaidah asosiasi sangatlah penting artinya untuk mengetahui bentuk lattice dari *closed frequent itemset* yang ditemukan. Karena pada tahap ini akan ditentukan nilai confidence dari sebuah itemset yang akan dengan mudah dipahami dengan sebuah lattice. Juga pada proses pemangkasan kandidat, dengan menggunakan *downward closure property*, akan sangat mudah dipahami dengan menggunakan lattice. Sebuah bentuk struktur lattice dapat dilihat pada Gambar 2.7.

Pada kenyataannya, membangun sebuah lattice dianggap terlalu melambat untuk sejumlah besar *closed frequent itemset*. Maka dibutuhkan strategi baru, dengan menghitung langsung lattice secara bersamaan dengan proses pencarian *closed frequent itemset*. Dasar gagasannya adalah ketika suatu *closed frequent itemset* ditemukan, maka secara efisien dapat secara langsung ditentukan semua *closed frequent itemset* yang merupakan supersets dari sebuah itemset. Pendekatan ini membawa ke arah sebuah algoritma yang sangat efisien.



Gambar 2.7 Sebuah itemset dalam struktur lattice.

Untuk visualisasi *closed frequent itemset* yang dihasilkan oleh algoritma sebelumnya (CHARM), dibutuhkan algoritma yang mampu membangun sebuah struktur lattice dari *closed frequent itemset* yang ditemukan. Algoritma yang dimaksud adalah Algoritma CHARM-L (Zaki, Hsiao, 2005), dimana huruf L = Lattice. Pada algoritma ini mampu memberikan solusi yang efisien, dengan cara menggabungkan dua proses yaitu proses pencarian *closed frequent itemset* dengan proses pembentukan lattice. Pada saat ditemukan sebuah *closed frequent itemset*, maka akan ditentukan secara langsung posisinya dalam lattice. Untuk lebih rinci akan dijelaskan cara kerja dari algoritma ini, dan untuk lebih memudahkan pemahaman akan diberikan sebuah contoh.

Algoritma CHARM-Lattice:

CHARM-L ($\mathcal{D}$, *min_sup*):
1. $[\emptyset] = \{l_i \times t(l_i) : l_i \in \mathcal{I} \wedge \sigma(l_i) \geq min\_sup\}$
2. CHARM-L-EXTEND ($[\emptyset]$, $\mathcal{L}_r = \{\emptyset\}$)
3. **return** $\mathcal{L}$ //lattice of closed sets

CHARM-L-EXTEND ($[P]$, $\mathcal{L}_c$):
4. **for each** $l_i \times t(l_i), \mathbb{C}(l_i)$ in $[P]$
5. $P_i = P \cup l_i$ and $[P_i] = \emptyset$
6. UPDATE-$\mathbb{C}$ (l_i, $[P]$)
7. **for each** $l_j \times t(l_j), \mathbb{C}(l_j)$ in $[P]$, with $j > i$
8. $X = l_j$ and $Y = t(l_i) \cap t(l_j)$ and $\mathbb{C}(X) = \mathbb{C}(l_i) \cap \mathbb{C}(l_j)$
9. CHARM-PROPERTY($X \times Y, l_i, l_j, P_i, [P_i], [P]$)
10. $\mathcal{L}_n$ = SUBSUMPTION-CHECK-LATTICE-GEN($\mathcal{L}_c, P_i, \mathbb{C}(P_i)$)
11. CHARM-L-EXTEND ($[P_i]$, $\mathcal{L}_n$)
12. delete $[P_i]$

CHARM-L merupakan algoritma yang efisien dalam menggali struktur lattice dari semua frequent closed itemset. Algoritma ini mengenumerasi closed itemset dalam ruang pencarian IT-Tree yang dapat dipecah sehingga pencarian dapat dilakukan pada masing-masing subtree secara *independent*. Pencarian dilakukan dengan metode *Hybird Search* didasarkan pada sifat pasangan itemset-tidset sehingga dapat memotong level pencarian. Penggunaan format data vertikal dengan diffset digunakan untuk mengurangi penggunaan memori. Pendekatan yang dipilih untuk memangkas *non-closed itemset* yang ditemukan selama proses enumerasi dilakukan dengan operasi irisan himpunan yang sederhana.

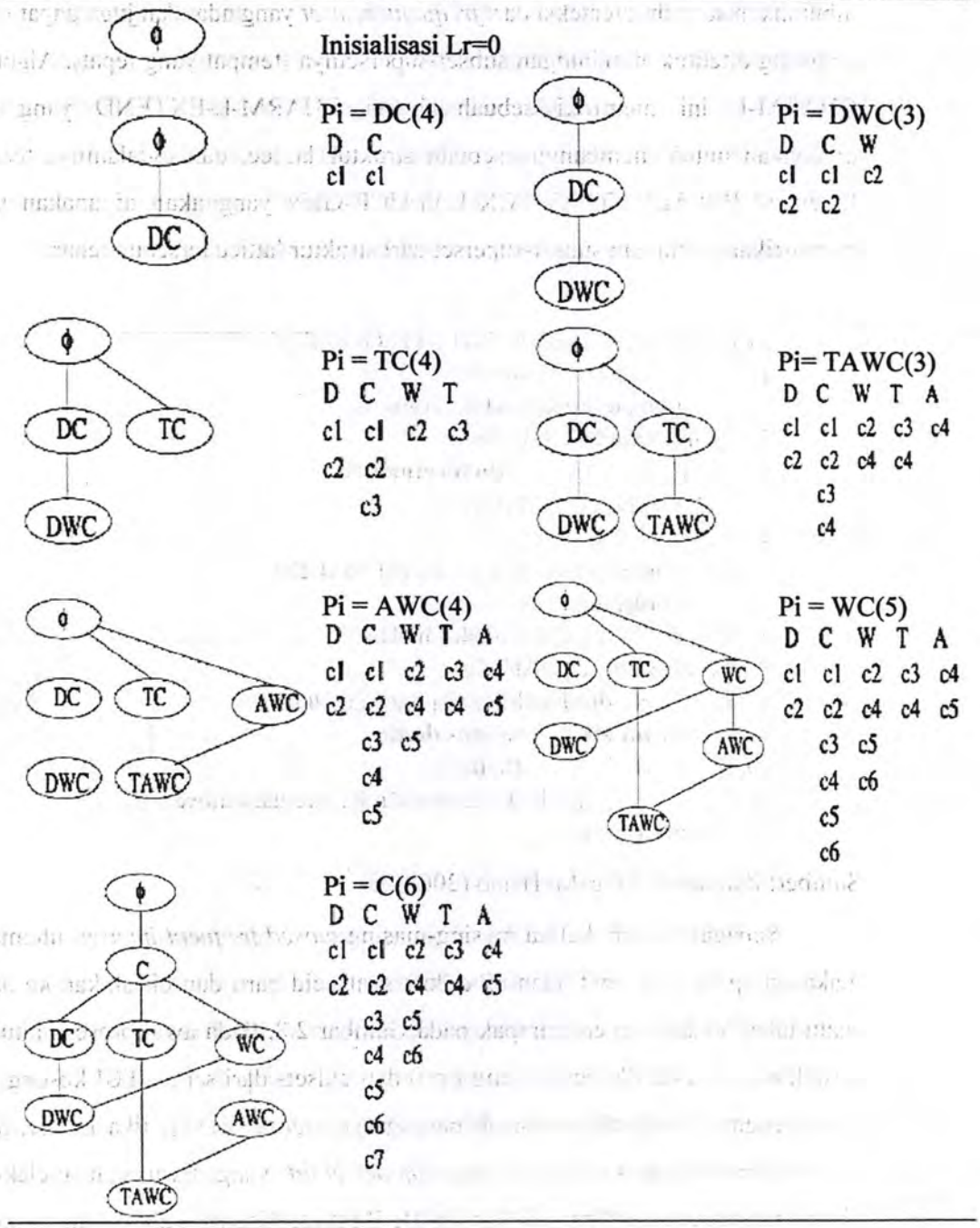
Algoritma CHARM-L ini prosesnya diawali dengan tahap inisialisasi, *root node* dari lattice = 0, kemudian dibutuh sebuah inisialisasi *parent class* untuk item yang sering muncul. Jika telah ditemukan sebuah *frequent itemset,* maka dapat diberikan sebuah *frequent itemset identifier* (cid). Untuk langkah selanjutnya dibutuhkan sebuah interseksi dari *frequent itemset* yang ada, dan juga dapat secara langsung ditentukan hubungan subset-supersetnya (tempat yang tepat). Algoritma CHARM-L ini memiliki sebuah fungsi CHARM-L-EXTEND yang akan digunakan untuk membangun sebuah struktur lattice, dan didalamnya terdapat fungsi SUBSUMPTION-CHECK-LATTICE-GEN yang akan digunakan untuk memastikan hubungan subset-superset dari struktur lattice tersebut benar.

SUBSUMPTION-CHECK-LATTICE-GEN($\mathcal{L}_i$, X, $\mathbb{C}(X)$):
1. $\mathcal{S} = \{Z \in \mathcal{C} | Z.\text{cid} \in \mathbb{C}(X)\}$
   //eliminate subsumed itemsets
2. **for each** $Z \in \mathcal{S}$ **do**
3. **if** $\sigma(X) = \sigma(Z)$ **then return** $\mathcal{L}_i$.
   //Insert X as child of $\mathcal{L}_i$
4. $\mathcal{L}_n = X$
5. $\mathcal{L}_i$.children.add($\mathcal{L}_n$), $\mathcal{L}_n$.parents.add($\mathcal{L}_i$)
   //Adjust Lattice
6. $\mathcal{S}^{\min} = \{Z \in \mathcal{S} | Z \text{ is Minimal}\}$
7. **for all** $Z \in \mathcal{S}^{\min}$ **do**
8. $\mathcal{L}_n$.children.add(Z), Z.parents.add($\mathcal{L}_n$)
9. **for all** $Z_p \in Z$.parents **do**
10. **if** $Z_p \subset \mathcal{L}_n$ **then**
11. Z_p.children.remove(Z), Z.parents.remove(Z_p)
12. **return** $\mathcal{L}_n$

Sumber: Zaki and Ching-Jui Hsiao (2005)

Sebagai contoh, ketika masing-masing *closed frequent itemset* ditemukan, maka setiap FCI tersebut akan diberikan suatu cid baru dan disisipkan ke dalam suatu tabel *lookup*, seperti tampak pada Gambar 2.8. Pada awal proses yaitu saat inisialisasi, root dari lattice kosong Lr=0 dan cidsets dari semua FCI kosong. FCI yang pertama ditambahkan adalah pasangan *closet set* {DC}, jika Lc=Lr, maka kita menambahkan Ln=DC sebagai *node child* yang baru dan melakukan perubahan pada cidset dari C dan D. Berikutnya, kita menambahkan pasangan *closet set* {DWC}, dengan Lc=DC. Dan DWC ditambahkan sebagai node baru Ln

dan menjadi *node child* dari DC dan cidsets juga dibaharui. Proses selanjutnya berlanjut dan setiap node yang baru akan menjadi *node child* dari node saat ini (sesuai nilai Lc saat ini).



Gambar 2.8 Ilustrasi algoritma CHARM-L.

Satu kasus menarik adalah ketika kita menambahkan pasangan closet set {WC}, yang mana memerlukan menyesuaian pointer induk dari lattice node yang ada. Dimana untuk C(WC) = C(C) ∩ C(W) = {c1, c2, c3, c4, c5} ∩ {c2, c4, c5} = {c2, c4, c5}.

Sebagai catatan, bahwa cidset yang dijelaskan sebelumnya hanya untuk item-item tunggal, namun pada algoritma ini menghitung cidset dari semua itemset dengan cara melakukan interseksi. Dimana S={DWC, TAWC, AWC}. Maka {WC} dengan *support count*= 5, tidak memiliki *closed superset* yang sesuai sehingga kita harus menambahkan Ln=WC sebagai *node child* dari Lc=Lr dan melakukan perhitungan untuk element minimalnya, Smin={DWC, AWC}. Maka tiap elemen minimalnya akan menjadi *node child* dari WC. Kemudian, akan diamati apakah terdapat *parent* dari sebuah set dalam Smin yang merupakan subset dari WC, hal ini akan mengubah bentuk lattice yang ada.

Pada awalnya Lr=0, parent dari AWC (sebelum ditambah WC), adalah subset dari WC, maka kita akan menghilangkan *parent–child link* antara Lr dan AWC. Pada akhirnya *closed set* terakhir yang akan ditambahkan pada lattice adalah C dan kita akan mendapatkan closed itemset lattice yang penuh. Pada lattice yang terakhir terbentuk akan dapat dilihat *parent–child link* yang sesuai dengan melihat nilai *support count* dari masing-masing node yang terbentuk. Bahwa C dengan nilai *support count*= 6 sebagai *parent* dari itemset yang lain.

Dari penjelasan diatas, dapat disimpulkan bahwa untuk membangun sebuah struktur lattice memerlukan waktu komputasi yang cukup besar. Hal ini dianggap mengurangi nilai efisiensi dari algoritma pencarian *frequent closed itemset*, disisi lain visualisasi dalam bentuk lattice sangat membantu pengguna untuk melanjutkan pada proses pembangkitan kaidah asosiasi. Sehingga penentuan jumlah *frequent closed itemset* yang ditemukan, dianggap sebagai sesuatu yang penting. Maka timbul ide untuk menggunakan *top-k frequent closed itemset*, dimana nilai top-k adalah batasan yang ditentukan oleh pengguna. Dan nilainya dapat ditentukan misalnya, sebesar 20 atau 100. Dengan batasan top-k ini, diharapkan menambah nilai efisiensi dari algoritma pencarian *frequent closed itemset* yang sudah ada. Pada subbab berikut ini akan dijelaskan mengenai pembangkitan *top-k frequent closed itemset*.

2.6 Pembangkitan Top-k Frequent Closed Itemset

Untuk pembangkitan top-k frequent closed itemset, pencarian itemset yang sering muncul masih bergantung pada nilai *min_sup* yang ditentukan oleh pengguna. Penentuan nilai *min_sup* ini masih dianggap menyulitkan pengguna, maka dalam penggalian *top-k frequent closed itemset* digunakan minimum length (*min_l*) serta menentukan k, yang dianggap lebih mudah ditentukan oleh pengguna. Penggalian itemset yang sering muncul dengan top-k mengacu pada k itemset yang bersifat *frequent closed*, dan *min_l* dimana *min_l* >= 0 merupakan panjang minimal dari itemset-itemset yang bersifat closed. Bagaimanapun, ada beberapa perbedaan pokok menyangkut permasalahan pementuan *min_l*, sebagai berikut;

- Pertama, *min_l* pada umumnya suatu angka kecil (seperti 2 sampai dengan 20), yang mana dapat ditetapkan oleh para pemakai dengan aplikasi yang mereka pikirkan, sedangkan *min_sup* adalah sering bersifat *data-dependent* dan boleh diperlukan yang disesuaikan atas suatu cakupan luas.
- Kedua, *min_l* batasan memberi kita kebebasan untuk menggali hanya dengan itemsets tanpa pertama dengan menemukan banyak itemset yang lebih pendek, atau menambang hanya itemsets yang panjangnya spesifik.
- Ketiga, sekalipun orang tidak ingin menetapkan batasan *min_l*, pekerjaan penggalian top-k masih menyediakan suatu alat untuk pekerjaan penggalian yang efisien tanpa menetapkan batasan *minimum support* dengan cara pengaturan *min_l* sama dengan 0.

Namun ada sedikit perbedaan antara algoritma CHARM dengan algoritma TFP (*Top Frequent Pattern*). Untuk algoritma TFP ini, pencarian *frequent closed itemset* didasarkan pada algoritma *frequent pattern-growth* atau disebut dengan FP-Growth. Sehingga algoritma TFP memiliki keunggulan sebagai berikut; Pertama transaksi manapun yang lebih pendek dibandingkan dengan *min_l* tidak akan tercakup dalam kandidat itemset yang dicari. Kedua, min_support dapat ditingkatkan secara dinamis dan dibangun dalam bentuk struktur data FP-TREE, yang mana hal ini akan membantu dalam proses pemangkasan cabang yang tidak perpotensi pada pembangkitan *frequent closed itemset* dari pohon sebelumnya. Ketiga, cabang pohon yang paling potensial dapat dicari terlebih dahulu untuk

menaikkan *min_sup* lebih lanjut, dan *min_sup* yang diangkat kemudian secara efektif. Keempat, satu set metoda *search space pruning* dan *closure cheking scheme* untuk itemset-itemset, yang efisien, digunakan untuk meningkatkan kecepatan dalam proses penggalian *closed itemsets*.

Berikut mekanisme pembangkitan *Top-k Frequent Closed Itemset*:

1. Membangun FP-Tree dari input DB (database).
   a. Jika jumlah dari frequent itemset dalam transaksi T tidak kurang dari min_*l*, maka T akan dimasukkan kedalam FP-Tree.
   b. Pada saat yang sama, digunakan metode *closed_node_count* untuk meningkatkan min_sup yang telah dinaikkan untuk memangkas *infrequent itemset* dari FP-Tree.
2. Pruning yang dilakukan pada FP-Tree.
   - Dengan menggunakan metode descendant_sum ditujukan untuk meningkatkan nilai min_sup untuk melakukan pemangkasan pada FP-tree.
3. Mining Proses (Top-k frequent closed itemset).
   a. Penelusuran secara *top-down* pada tiap item yang ada dalam tabel global header.
   b. Jika terdapat item yang memiliki *l_count* tidak kurang dari *min_sup* saat ini, maka item tersebut dianggap diperlukan sebagai sebuah prefix frequent itemset.
   c. Bangun FP-tree dan panggil subroutine Mine_count_FP-tree() untuk mencari frequent closed itemset.
   d. Telusuri dengan strategi bottom-up dan disesuaikan dengan *support descending order* untuk menghasilkan keluaran atau output berupa *top-k frequent closed itemset.*

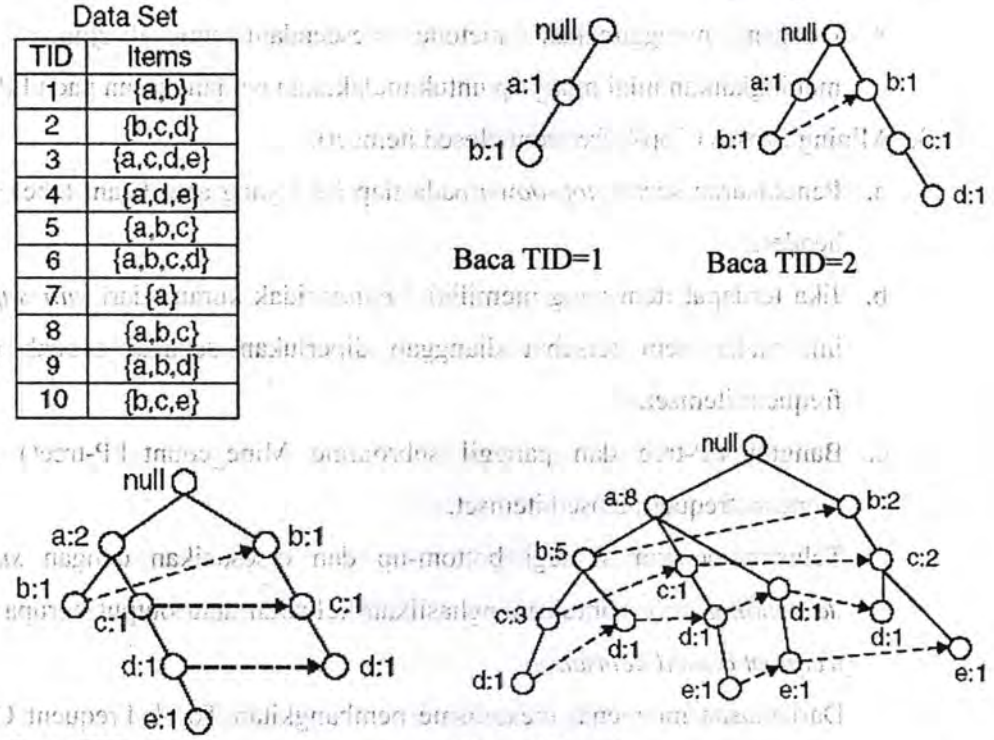
Dari ulasan mengenai mekanisme pembangkitan Top-k Frequent Closed Itemset ini akan dilanjutkan dengan pembahasan tentang mekanisme pencarian pola yang sering muncul dengan FP-Growth, yang akan dibahas pada subbab berikut ini.

2.6.1 Pencarian Pola dengan FP-Growth

Algoritma FP-growth ini memiliki pendekatan yang berbeda dengan algoritma Apriori dalam menghasilkan itemset yang sering muncul. Dengan menggunakan sebuah struktur data yang disebut dengan FP-tree algoritma ini secara langsung dapat menentukan itemset yang sering muncul. Sebuah FP-tree merupakan representasi input data yang dimampatkan, dibangun dengan membaca satu transaksi dalam satu waktu pada dataset dan memetakannya pada FP-tree. Pada Gambar 2.7 dapat dilihat ilustrasi pembentukan sebuah FP-tree dari sebuah transaksi dataset yang berisi sepuluh buah transaksi, yang masing-masing transaksi memiliki nomor identifikasi transakasi. Dan setiap transaksi berisi sebuah itemset.

Transaction Data Set

TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}



Gambar 2.9 Ilustrasi konstruksi sebuah FP-tree.

Untuk keadaan yang terbaik pada algoritma FP-growth ini, jika semua transaksi memiliki kumpulan item yang sama, maka FP-tree hanya memiliki satu

cabang saja. Untuk keadaan yang terburuk, jika setiap transaksi memiliki kumpulan itemset yang berbeda satu dengan yang lain. Dan ukuran dari FP-tree secara efektif sama dengan ukuran data yang sebenarnya. Bagaimanapun juga kebutuhan simpanan fisik untuk FP-tree sangatlah tinggi karena membutuhkan ruang tambahan untuk menyimpan pointer antara node yang satu dengan yang lain dan counter untuk setiap item yang ada. Ukuran FP-tree juga bergantung pada urutan item-item tersebut. FP-growth adalah algoritma yang menghasilkan itemset yang sering muncul dari sebuah FP-tree secara *bottom-up*. Strategi ini mencari itemset yang sering muncul dengan pendekatan *suffix-based* dan mengimplementasikan strategi divide-and-conquer yang akan mengubah permasalahan kedalam sub-sub masalah. Bila kondisi dataset yang ada menghasilkan FP-tree dengan kondisi prefix tree penuh, maka kinerja algoritma ini akan menurun karena akan menghasilkan sub-sub masalah yang banyak dan menggabungkan hasilnya untuk setiap submasalah yang ada.

Waktu komputasi sebuah algoritma FP-growth bergantung pada faktor pemampatan urutan itemset yang sering muncul, urutan yang berbeda untuk item yang sering muncul akan menghasilkan ukuran tree yang berbeda pula, hal ini akan berpengaruh terhadap ruang serta waktu pencarian. Dalam algoritma FP-growth ini dibutuhkan kapasitas memory yang besar untuk menyimpan nilai kemunculan sebuah item dalam data transaksi yang besar. Dan dilakukan proses pembacaan setiap data transaksi yang ada untuk mengatahui itemset yang sering muncul.

Berdasarkan algoritma TFP ini, diharapkan akan tercapai keinginan untuk membuat algoritma pencarian *frequent closed itemset* yang efisien. Serta keuntungan bagi pengguna adalah dimudahkan dengan hanya menentukan nilai k dan *min_l* tanpa menentukan nilai *minimum_support* yang sesuai.

BAB 3
PERANCANGAN ALGORITMA

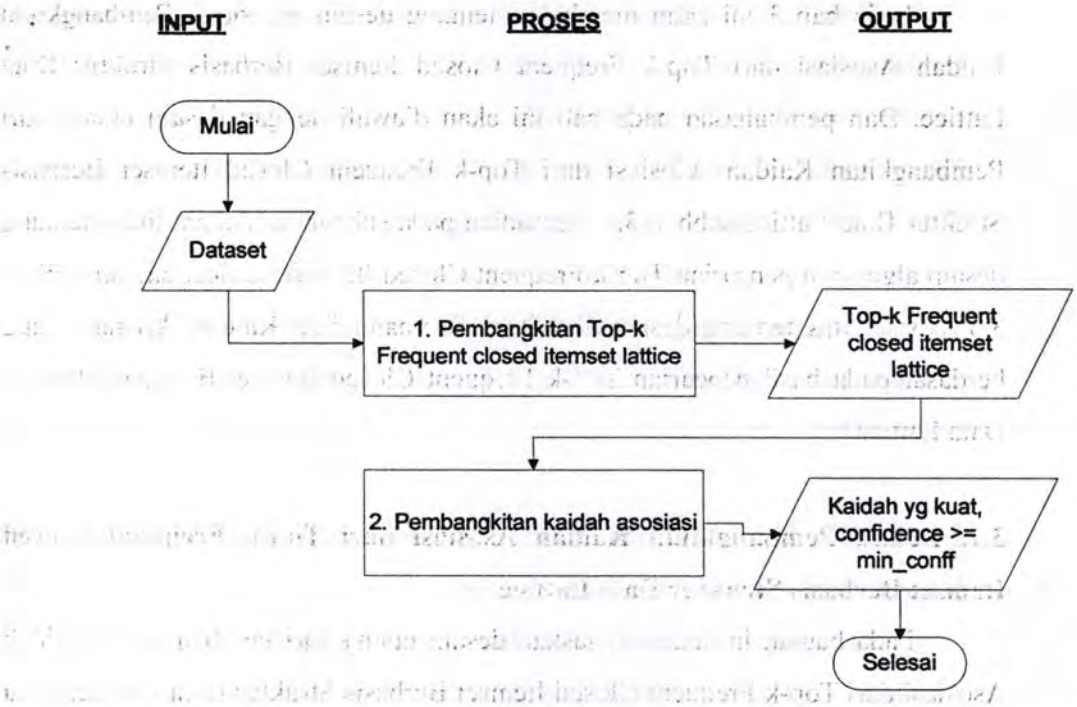
BAB 3
DESAIN ALGORITMA

Pada bab 3 ini akan membahas tentang desain algoritma Pembangkitan Kaidah Asosiasi dari Top-k Frequent Closed Itemset Berbasis Struktur Data Lattice. Dan pembahasan pada bab ini akan diawali dengan desain utama dari Pembangkitan Kaidah Asosiasi dari Top-k Frequent Closed Itemset Berbasis Struktur Data Lattice subbab 3.1, kemudian pada subbab 3.2. akan diulas tentang desain algoritma pencarian Top-k Frequent Closed Itemset Lattice, bagian subbab 3.3 akan diulas tentang desain algoritma Pembangkitan Kaidah Asosiasi yang berdasar pada hasil pencarian Top-k Frequent Closed Itemset Berbasis Struktur Data Lattice.

3.1. Desain Pembangkitan Kaidah Asosiasi dari Top-k Frequent Closed Itemset Berbasis Struktur Data Lattice

Pada bagian ini akan dijelaskan desain utama dari Pembangkitan Kaidah Asosiasi dari Top-k Frequent Closed Itemset Berbasis Struktur Data Lattice. Pada analisis kaidah asosiasi pada penggalian data atau data mining terdapat strategi umum yang terdiri dari dua tahapan, sebagai berikut; (1) Pembangkitan itemset yang sering muncul atau *Frequent Itemset Generation*, merupakan strategi yang secara obyektif mencari semua itemset – itemset yang tepat dengan menggunakan batasan *minimum support*. Itemset ini disebut dengan *frequent itemset* atau itemset yang sering muncul. Kemudian diikuti tahap yang ke-(2) Pembangkitan Kaidah Asosiasi atau *Rule Generation*, merupakan strategi umum yang secara obyektif mengekstrak semua rule yang memiliki nilai *confidence* yang tinggi dari itemset - itemset yang sering muncul, yang ditemukan pada tahap sebelumnya. Maka tahap ini akan menghasilkan sebuah kaidah-kaidah asosiasi dan kaidah-kaidah asosiasi ini disebut dengan kaidah yang kuat atau *strong rules*. Dari penjelasan tersebut akan menjadi dasar dari desain utama Pembangkitan Kaidah Asosiasi berdasarkan Top-k Frequent Closed Itemset Berbasis Struktur Data Lattice, yang dapat digambarkan dengan Blok Diagram Desain Utama Pembangkitan Kaidah Asosiasi

berdasarkan Top-k Frequent Closed Itemset Berbasis Struktur Data Lattice yang ditunjukkan pada Gambar 3.1 berikut ini.

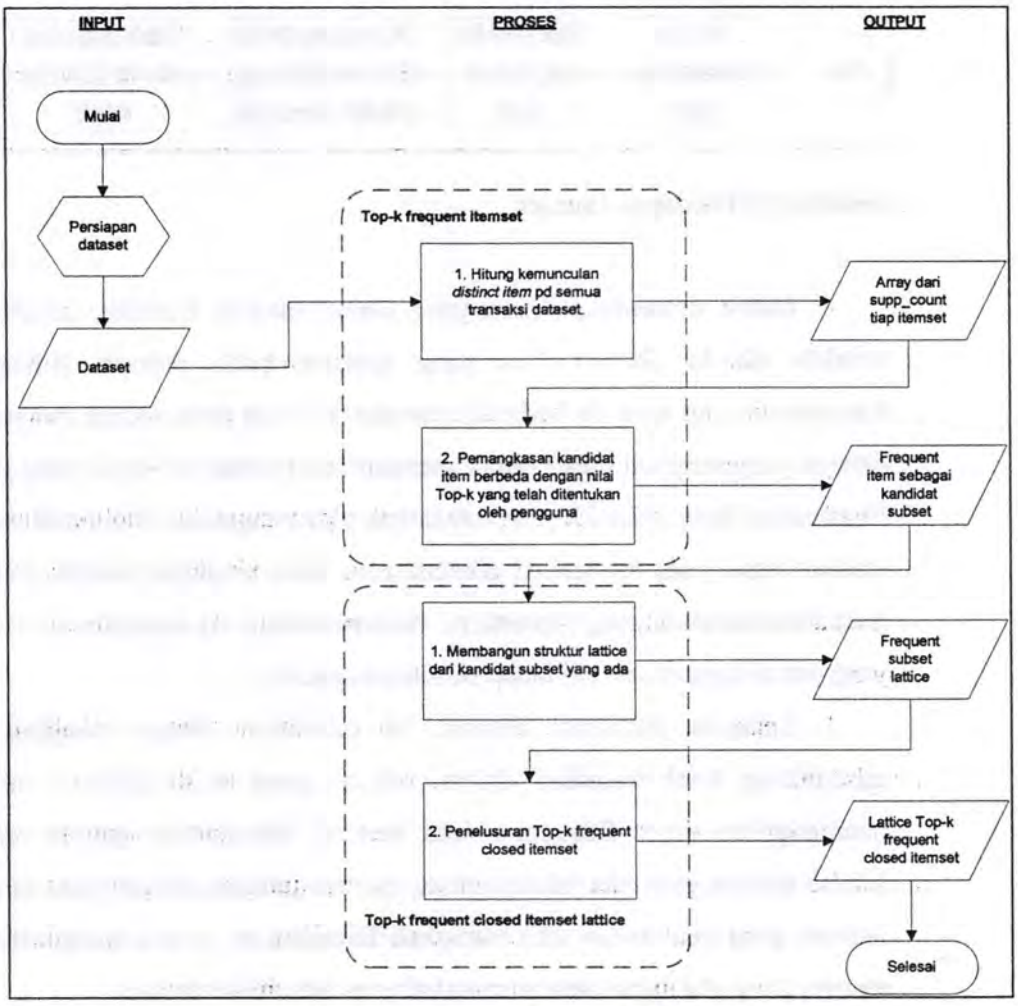


Gambar 3.1 Desain Utama Pembangkitan Kaidah Asosiasi dari Top-k Frequent Closed Itemset Berbasis Struktur Data Lattice.

Dari blok diagram diatas dapat terlihat dengan jelas proses-proses utama yang ada pada Desain Utama Pembangkitan Kaidah Asosiasi dari Top-k Frequent Closed Itemset Berbasis Struktur Data Lattice. Pada diagram alir terlihat sebuah dataset transaksi yang terlebih dahulu melalui tahap persiapan dataset untuk mengetahui jumlah item yang berbeda disetiap record dan rata-rata panjang record, secara detail akan dijelaskan pada subbab 3.2 yang mengulas tentang proses Pembangkitan Top-k Frequent Closed Itemset berbasis struktur data lattice dan hasil dari proses pertama ini akan menjadi masukan untuk tahap berikutnya, yaitu proses Pembangkitan Kaidah Asosiasi dari Top-k Frequent Closed Itemset Berbasis Struktur Data Lattice yang akan diulas pada subbab 3.3.

3.2. Pembangkitan Top-k Frequent Closed Itemset Lattice

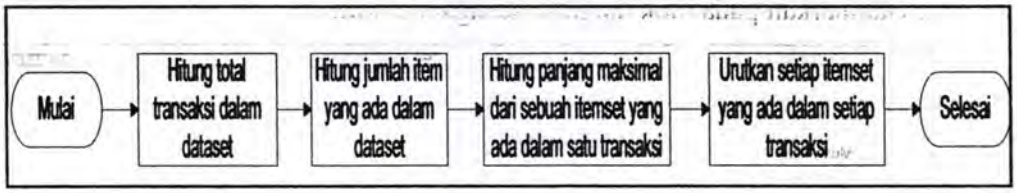
Pada tahap Pembangkitan Top-k Frequent Closed Itemset Lattice diGambarkan pada blok diagram sebagai berikut;



Gambar 3.2 Pembangkitan Top-k Frequent Closed Itemset Lattice

Diawali dengan proses persiapan data yang akan menghasilkan dataset. Proses persiapan data yang dilakukan, dianggap sebagai proses yang bukan termasuk proses utama. Untuk langkah-langkah persiapan dataset atau *preparing dataset*, dapat dilihat pada blok diagram Gambar 3.3. Dengan dataset yang telah disiapkan tersebut maka proses pembangkitan Top-k Frequent closed itemset lattice sebagai proses utama dibagi menjadi dua sub proses. Yang pertama adalah

proses pencarian top-k frequent itemset dan proses yang kedua adalah proses pencarian frequent closed itemset.



Gambar 3.3 Persiapan Dataset

Untuk menentukan item yang sering muncul tersebut, harus diketahui terlebih dahulu *distinct item* yang terdapat pada dataset. Hitung derajat kemunculan tiap item yg berbeda atau *distinct item* pada semua transaksi dalam dataset yang menjadi dasar untuk menentukan peringkat Top-k yang jumlahnya disesuaikan dengan nilai k yang ditentukan oleh pengguna. Dan untuk setiap top-k *distinct* item yang ditentukan dibatasi oleh nilai maximal jumlah distinct item pada data transaksi yang digunakan.. Pada penelitian ini, menghitung *distinct item* yang ada dianggap sebagai tahap persiapan dataset.

Tahapan persiapan dataset ini dilakukan dengan langkah pertama menghitung total transaksi dalam dataset yang akan dipakai untuk tahap pembangkitan top-k frequent closed itemset. Dilanjutkan dengan menghitung jumlah itemset yang ada dalam dataset, dan menghitung panjang maksimal sebuah itemset yang ada dalam satu transaksi. Dilanjutkan proses mengurutkan setiap itemset yang ada dalam setiap transaksi yang ada dalam dataset.

Juga perlu menghitung total item berbeda yang ada dalam dataset sebagai batasan penentuan nilai k tidak lebih besar dari total item berbeda yang ada dalam dataset, nilai k yang diberikan pengguna sebagai acuan penentuan peringkat *frequent distinct item* dataset yang dianggap sebagai parameter masukan pada tahap pertama Pembangkitan *frequent closed itemset lattice*. Pada tahap Pembangkitan Top-k frequent closed itemset lattice terdiri dari tahap (1) Top-k frequent itemset dan (2) Top-k frequent closed itemset lattice yang akan dibahas dalam subbab-subbab berikut.

3.2.1 Top-k Frequent Item

Sebuah struktur lattice dapat digunakan untuk menyebutkan satu persatu semua itemset yang mungkin. Secara umum, sebuah dataset yang berisi k items secara potensial akan menghasilkan sejumlah 2^k-1 *frequent itemsets*, tidak termasuk null set. Karena nilai k dimungkinkan bernilai besar dalam kenyataannya maka ruang pencarian dari itemsets dibutuhkan pencarian sebesar angka eksponensial. Terdapat beberapa cara untuk mengurangi kompleksitas komputasi dari kegiatan pencarian itamset yang sering muncul antara lain dengan mengurangi jumlah kandidat itemsets.

Pada desain algoritma ini, untuk tahap persiapan data akan ditetapkan nilai minimal length, maksimal length dan nilai Top-k yang semuanya dapat dengan mudah ditentukan oleh pengguna. Diharapkan dapat mengurangi jumlah kandidat itemsets. Diawali dengan penentuan nilai minimal length dan maksimal length, jika sebuah itemset berisi k items, maka disebut sebagai sebuah k-itemset. Sebagai contoh {susu, telur, gula, tepung}adalah 4-itemset. Maka jika ditetapkan minimal length=2 dan maximal length=10 kita hanya akan memproses data transaksi dengan min 2-itemset dan max 10-itemset. Untuk nilai Top-k, nilai k adalah derajat keseringan muncul atau rangking derajat kemunculan *distinc item* yang ada pada data transaksi. Penentuan nilai k oleh pengguna dibatsi oleh batas nilai maximal jumlah *distinc item* data transaksi yang digunakan.

Dari penetapan minimal length, maksimal length dan nilai Top-k, akan menghasilkan *frequent itemset* yang akan digunakan pada tahap pembangkitan *frequent closed itemset lattice*. Hal ini didasari dari prinsip apriori; "*Jika sebuah itemset sering muncul maka semua subsetsnya akan sering muncul juga.*" Dari *frequent itemset* yang dihasilkan akan menjadi kandidat itemset pada proses pencarian *closed frequent itemset*. Jika terdapat sebuah itemset dengan nilai k lebih kecil dari nilai k yang ditetapkan oleh pengguna, maka itemset tersebut bukan kandidat itemset pada proses pencarian *closed frequent itemset*. Sehingga dapat disimpulkan dengan nilai k ini, dapat memangkas itemset-itemset yang dianggap tidak sering muncul (*infrequent itemset*).

Pada tahap berikut ini akan digunakan sebuah dataset transaksi, seperti yang tampak pada Tabel 3.1, Untuk proses pembangkitan Top-k frequent itemset

akan diawali dengan langkah (1) Menghitung kemunculan tiap item pada seluruh transaksi dalam basisdata, seperti yang tampak pada kolom empat pada Tabel 3.1, dan dari proses tersebut yang akan menghasilkan sebuah array dari nilai kemunculan tiap item untuk seluruh transaksi yang ada pada basisdata. Array ini akan digunakan untuk proses selanjutnya yaitu proses (2) Pemangkasan kandidat item-item yang berbeda dengan nilai k yang telah ditentukan oleh pengguna.

Sehingga dapat ditentukan peringkat atau derajat kemunculan itam-item yang sering muncul dalam transaksi dataset. Sebagai contoh nilai Top-k yang diberikan oleh pengguna adalah 5 (Top-5), minimal length = 1, dan maksimal length = 5, maka dapat diartikan pengguna menginginkan 5 peringkat itemset berbeda yang paling sering muncul pada dataset transaksi.

Dari ke-lima peringkat tersebut dijadikan dasar untuk menentukan itemset-itemset yang closed. Kandidat itemset yang memenuhi top-k = 5 adalah: a=8; b=7; c=6; d=4; dan yang kelima adalah e=3. Nilai minimal length=1 artinya transaksi dengan minimal 1-itemset yang akan diproses dan maximal length=5 artinya transaksi dengan maximal 5-itemset yang akan diproses, max_length ini juga akan mempengaruhi kedalaman atau level dari lattice yang dibentuk. Sehingga nilai max_length=5 maka level lattice yang terbentuk nantinya 5 level.

Tabel 3.1. Tabel Dataset Transaksi

1	2	3	4
TID	**Items**	**Item yang telah terurut**	**Jumlah kemunculan**
100	d, a, e, g	a, d, e, g	a = 8
200	b, a	a, b	b = 7
300	h, a, c, b, e	a, b, c, e, h	c = 6
400	a, b, c, d	a, b, c, d	d = 4
500	a, c, b, f, d, i	a, b, c, d, f, i	e = 3
600	b, a, c	a, b, c	f = 2
700	f, a, b, i, c, d	a, b, c, d, f, i	g = 2
800	a, e, c, b	a, b, c, e	h = 1
900	J	j	i = 1
			j = 1

Dari proses pemangkasan kandidat itemset dengan nilai Top-k, akan dihasilkan frequent itemset sebagai kandidat subset frequent itemset lattice yang

akan dibangun dan digunakan pada proses berikutnya yaitu membangun Top-k frequent closed itemset lattice.

Algoritma tersebut akan diawali dengan menghitung nilai *support count distinct item* yang ada dalam dataset yang ada kemudian membangun array untuk setiap item pada semua transaksi yang ada dalam dataset, baris kedua dan ketiga pada algoritma tersebut. Array ini nantinya akan berisi nilai (*support count distinct item*). Sebagai berikut, Count [0] = 8 untuk a, Count [1] = 7 untuk b, Count [2] = 6 untuk c, Count [3] = 4 untuk d, dan terakhir Count [4] = 3 untuk e.

Algoritma Top-k Frequent itemset

Masukan: sebuah dataset.

Keluaran: array itemset dengan nilai support count dan array kemunculan item x (t(X)) pada semua transaksi.

```
1.  For1 T = 1 to jumlah transaksi
2.      Hitung support_count_item
3.      Bangun_array_item_pada_semua_transaksi
4.      Tulis_transaksi_yang_mengandung_X
5.  End for1
6.
7.  For2 I = 1 to jumlah item
8.    If Hitung_support_count_item[i] > (k-1) then
9.        Frequent_item[i] = support_count[i]
10. End for2
```

Pada algoritma Top-k Frequent Item ini, juga akan diproses sebuah array *tid* yang akan berisi nilai kemunculan tiap *distinct item* pada setiap transaksi yang ada pada dataset, baris ke empat pada algoritma diatas. Nilai ini nantinya akan digunakan sebagai indikator apakah sebuah item tersebut sering muncul (*frequent*) atau tidak (*infrequent*), mengacu pada aksioma pada penelitian sebelumnya.
Theorema1. Bahwa sebuah itemset yang (*frequent*) atau sering muncul, dibangun dari item-item yang bersifat (*frequent*) atau sering muncul juga.

Dari aksioma tersebut maka array *tid* yang akan berisi nilai kemunculan tiap *distinct item* pada setiap transaksi yang ada pada dataset, seperti tampak pada tabel 3.2. Dari tabel tersebut akan digunakan sebagai kandidat subset, yang akan

menghasilkan *frequent subset lattice*. Untuk mendapatkan *frequent item*, nilai *support count* dari kandidat subset akan dibandingkan dengan (k-1), baris kedelapan sampai kesembilan algoritma diatas. Maka kandidat subset yang sering muncul atau *frequent* ini, akan digunakan pada proses selanjutnya. Yaitu tahap pembangunan Top-k Frequent Closed Itemset Lattice, yang akan diulas pada subbab berikutnya.

Tabel 3.2. Tabel Kemunculan item pada tiap transaksi

	Kemunculan item pada transaksi
t(a)	{1, 2, 3, 4, 5, 6, 7, 8}
t(b)	{2, 3, 4, 5, 6, 7, 8}
t(c)	{3, 4, 5, 6, 7, 8}
t(d)	{1, 4, 5, 7}
t(e)	{1, 3, 6}

3.2.2 Top-k Frequent Closed Itemset Lattice

Pada tahap proses membangun Top-k Frequent Closed Itemset Lattice Diawali dengan (1) Membangun struktur lattice dari kandidat subset yang ada yang akan menghasilkan frequent subset lattice, frequent subset lattice ini merupakan lattice untuk kandidat itemset yang dihasilkan dari proses sebelumnya. Dan Proses kedua adalah (2) Pemangkasan atau *pruning* dengan nilai Top-k yang disertai dengan min_*l* dan penelusuran Top-k frequent closed itemset. Dan dari proses yang kedua ini akan dihasilkan lattice Top-k closed frequent itemset. Algoritma Frequent Closed Itemset Lattice dapat dijelaskan sebagai berikut;

Algoritma ini diawali dengan menganggap array item yang sering muncul, `Frequent_item[i]` sebagai hasil akhir proses sebelumnya. Pembangkitan Top-k Frequent Closed Itemset Lattice adalah `parent` atau kandidat *closed frequent itemset*. Pada algoritma ini juga terdapat sebuah *routine* `bangun child untuk masing-masing parent()` yang akan membangun lattice dari kandidat *closed frequent itemset*. Dan proses pemangkasan atau *pruning*, dilakukan dengan membandingkan nilai *support count* dari anak atau *child* dengan nilai (k-1). Hal

ini dilakukan untuk mengurangi ruang pencarian dan berdampak pada waktu komputasi. Lattice dari itemset yang sering muncul tampak seperti pada Gambar 3.4. Node-node yang lebih gelap latarbelakangnya mengGambarkan itemset yang sering muncul, hal ini dapat terlihat dari nilai *support count*-nya.

Algoritma Frequent closed itemset lattice

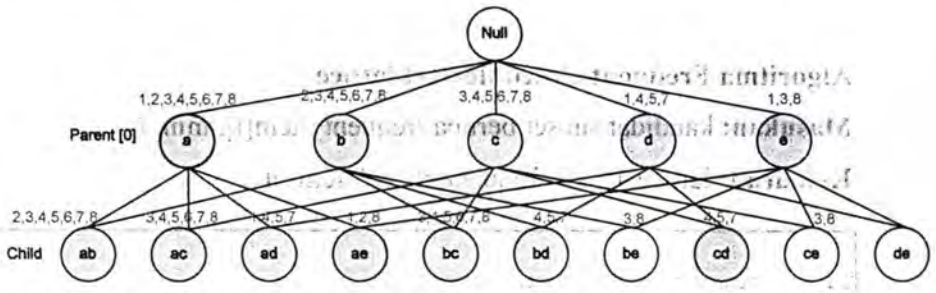
Masukan: kandidat subset berupa frequent_item[i], min_*l*

Keluaran: lattice Top-k frequent closed itemset

```
1.    parent = Frequent_item[i]
2.
3.    Repeat
4.    call bangun child untuk masing-masing parent()
5.    scan dan hitung support_count_child
6.
7.    if support_count_child >= (k-1) then
8.      bangun kandidat_closed_frequent_itemset
9.      parent = kandidat_closed_frequent_itemset
10.     call bangun child untuk masing-masing parent()
11.
12.   //downward closure property
13.   if support_count_child ∩ support_count_parent then
14.     frequent closed itemset = support_count_child
15.   else
16.     frequent closed itemset = support_count_parent
17.
18.   Until panjang maksimal child =(frequent_distinct_item -1)
```

Pada pembentukan *closed frequent itemset* diasumsikan bahwa semua subset adalah *closed*, jika subset tersebut memenuhi syarat; nilai *support_count* subset tersebut > dari (k-1) dan panjang itemset > panjang minimal itemset, min_*l*. Nilai *Support_count* subset tersebut dicari dengan melakukan mengiriskan array transaksi dari item *t*(X) pembentuknya. Dan Top-k closed frequent itemset akan dihasilkan dengan membandingkan nilai *support count* dari anak atau *child* dengan nilai *support count* dari *parent-nya*. Hal ini didasarkan pada definisi

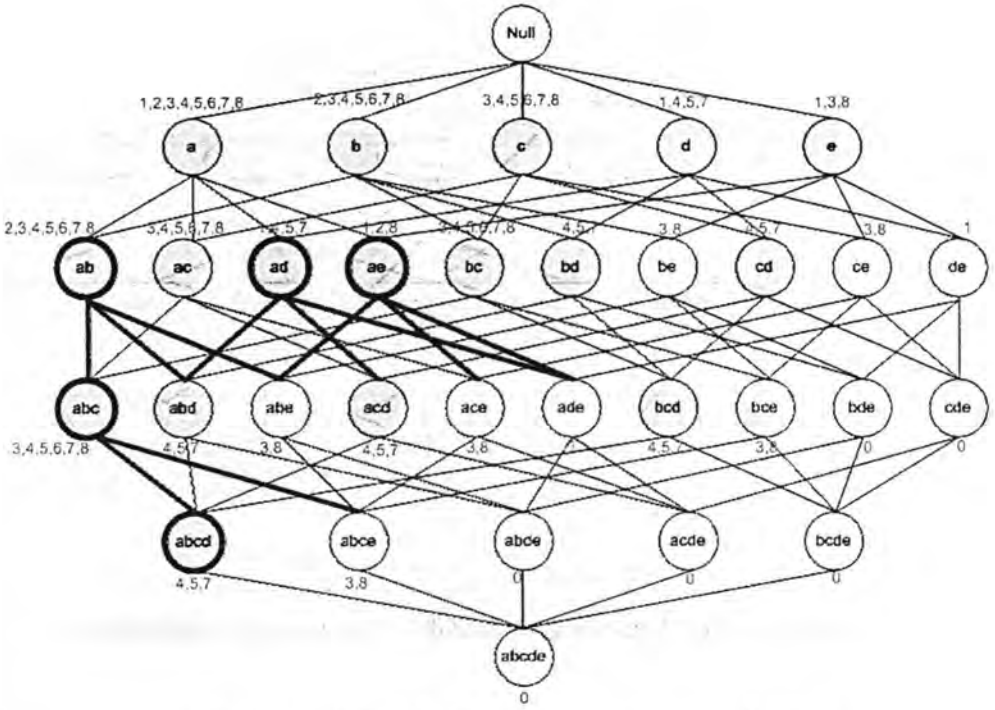
sebuah frequent itemset yang *closed,* jika tidak ada superset–superset saat ini yang memiliki *support count* yang tepat sama dengan subsetnya. Sebagai contoh ab=7; abc=6; ad=4; ae=3; dan abcd=3;



Gambar 3.4. Frequent itemset lattice

merupakan top-k closed frequent itemset dan Top-k Closed frequent itemset ini akan sangat berguna untuk menghilangkan kaidah-kaidah asosiasi yang ganda atau *redundant*, pada tahap Pembangkitan Kaidah Asosiasi. Berdasarkan definisi dari Kumar; "Closed Itemset adalah sebuah itemset dikatakan closed jika tidak mempunyai superset langsung yang memiliki support count yang tepat sama dengannya." Dari definisi ini maka dari contoh yang dibuat dihasilkan closed itemset ({a,b};{a,d};{a,e};{a,b,c};{a,b,c,d}) seperti pada Gambar 3.5. Top-k Closed Frequent Itemset lattice.

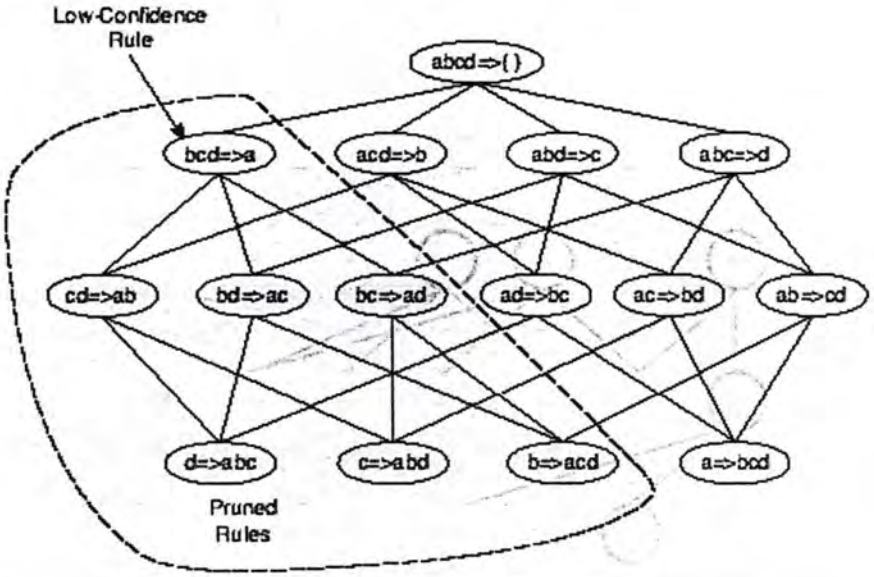
Dengan demikian top-k closed frequent itemset lattice merupakan hasil akhir dari tahap pertama dari desain utama, yaitu Pembangkitan Top-k Closed Frequent Itemset Lattice. Tahap kedua dari desain utama, yaitu Pembangkitan Kaidah Asosiasi akan menggunakan top-k closed frequent itemset lattice sebagai masukan dan akan dihitung nilai confidence dari masing-masing top-k closed frequent itemset yang ada dan dibandingkan dengan nilai *min_confidence* yang telah ditentukan oleh pengguna sebelumnya.



Gambar 3.5. Top-k Closed Frequent Itemset lattice

Pada tahap kedua dari analisis asosiasi ini adalah pembangkitan kaidah asosiasi, menggunakan itemset yang sering muncul sebagai kandidat-kandidat kaidah asosiasi yang baru. Pada Gambar 3.5 menggambarkan sebuah struktur lattice untuk kaidah asossiasi yang dibangkitkan dari *frequent itemset* {a, b, c, d}. Jika beberapa node dalam lattice memiliki confidence yang rendah, kemudian berdasarkan pada teorema diatas maka seluruh subgrup-nya dapat dipangkas dengan segera. Terdapat perbedaan pada tahap pembangkitan kaidah asosiasi dibandingkan dengan tahap sebelumnya, pada tahap ini tidak perlu melewati dataset berulang kali untuk menghitung *confidence* tiap kaidah dengan menggunakan *support count* dihitung pada saat pembangkitan *frequent itemset*.

Bagaimana sistem data mining memberikan informasi kaidah–kaidah atau rules yang menarik (*interesting*) bagi pengguna? Berawal dari pertanyaan tersebut kebanyakan algoritma–algoritma pencarian kaidah asosiasi menggunakan *support–confidence* sebagai ukuran untuk kaidah tersebut dapat dianggap menarik atau tidak bagi pengguna.



Gambar 3.6. Pemangkasan kaidah asosiasi dengan confidence.

Disisi lain penggunaan *minimum_support* dan batasan *confidence* membantu untuk mengurangi eksplorasi kaidah yang tidak menarik (*uninteresting rule*) karena dianggap sebagai kegiatan yang sia–sia. Namun pengukuran dengan menggunakan *support–confidence* masih dianggap memiliki keterbatasan. Dimana banyak pola–pola kaidah yang memiliki potensi bersifat menarik atau *interesting* namun memiliki nilai *support* yang rendah pada saat dikenakan batasan *support* dan tidak memenuhi maka pola atau kaidah tersebut akan dihapus. Serta kemungkinan masih terdapat kaidah yang tidak menarik yang dihasilkan oleh pengukuran *support-confidence* tersebut.

BAB 4
UJI COBA DAN ANALISIS HASIL

BAB 4
UJI COBA DAN ANALISIS HASIL UJICOBA

Pada bab 4 berikut ini akan dijelaskan mengenai uji coba dan analisis hasil uji coba aplikasi pembangkitan kaidah asosiasi dari top-k frequent closed itemset berdasarkan struktur data lattice. Pada bab 4 ini akan dibagi menjadi beberapa sub bab sebagai berikut, sub bab 4.1 akan mengulas tentang Lingkungan uji coba. Sub bab 4.2 akan mengulas tentang Data yang digunakan dalam uji coba. Sub bab 4.3 akan mengulas tentang Skenario uji coba. Sub bab 4.4 akan mengulas tentang Pelaksanaan uji coba. Sub bab 4.5 akan mengulas tentang Hasil uji coba. Dan Analisis hasil uji coba akan dijelaskan pada subbab 4.6.

4.1 Lingkungan Uji Coba

Pada sub bab ini akan dijelaskan tentang mengenai lingkungan uji coba aplikasi pembangkitan kaidah asosiasi dari top-k frequent closed itemset berdasarkan struktur data lattice, yang diimplementasikan dengan menggunakan bahasa pemrograman C. Serta menggunakan komputer dengan processor Intel Celeron 1.8 GHz, RAM 2 GB, dan sistem operasi Windows XP.

4.2 Data Uji Coba

Dataset yang digunakan pada penelitian ini adalah dataset nyata, yang memiliki karakteristik sangat rapat (terdiri dari banyak itemset yang sering muncul meskipun dikenakan nilai support yang tinggi). Dalam uji coba ini terdapat beberapa dataset nyata yang digunakan adalah dataset *gazelle* dan *chess, connect, pumsb*. Dataset *gazelle* digunakan untuk mewakili *sparse database* atau basis data yang renggang dan dataset *chess, connect,* dan *pumsb* digunakan dalam ujicoba ini untuk mewakili *dense database* atau basis data yang rapat.

Untuk dataset *gazelle*, memiliki 498 item, dengan rata-rata panjang item pada tiap transaksi 2.5 dan berisis 59.601 record. Dataset *gazelle* ini merupakan data *click-stream* dari Gazelle.com. dan dapat diambil dari *Blue Martini Software*. Dataset *connect* memiliki 130 item, dengan rata-rata panjang item pada tiap transaksi 43 dan berisis 67.557 record. Dataset ini merupakan data tahapan dalam

sebuah game dan dapat diperoleh di *UC Irvine Machine Learning Database Repository*. Begitu juga dataset *chess* merupakan langkah-langkah pada game catur dengan 76 item dan rata-rata panjang transaksi 37 dari 3196 transaksi. Sedangkan dataset *pumsb* merupakan data sensus dengan jumlah item sebanyak 7117 buah, rata-rata panjangnya 74 dari 49.046 transaksi yang ada.

Tabel 4.1 Karrakteristik Data Uji Coba

Data set	Jml Item	Rerata Panj	Jml Record
Chess	76	37	3.196
Connect	130	43	67.557
Pumsb	7.117	74	49.046
Gazelle	498	267	59.601

Tabel 4.1 menunjukkan karakteristik dataset yang digunakan dalam uji coba. Kolom 1 menyatakan nama basis data, kolom 2 menyatakan jumlah item yang digunakan dalam basis data. Kolom 3 menunjukkan rerata jumlah itemset dalam setiap transaksi, dan kolom 4 menunjukkan jumlah *record* atau transaksi pada tiap basis data.

4.3 Skenario Uji Coba

Untuk mengetahui kinerja algoritma ini maka terdapat beberapa skenario ujicoba yang diterapkan sebagai berikut;

a. Membandingkan kecepatan waktu proses dari algoritma modifikasi dengan algoritma TFP dan menentukan nilai min_l = 0.
b. Melihat karakteristik kecepatan komputasi algoritma yang dikembangkan pada beberapa dataset yang rapat dan dataset yang renggang.

Berdasarkan kedua skenario ini maka uji coba yang dilakukan bertujuan untuk mengetahui karakteristik kecepatan komputasi algoritma yang dikembangkan.

4.4 Pelaksanaan Uji Coba

Pelaksanaan uji coba untuk skenario yang pertama, pembangkitan Top-k Frequent Closed Itemset, membandingkan jumlah frequent closed itemset yang dihasilkan dan kecepatan komputasi algoritma pengembangan Top-k Frequent

Closed Itemset lattice dengan algoritma TFP. Menjalankan algoritma Top-k Frequent Closed Itemset lattice dengan memberikan inputan data set yang ada dan mencatat waktu komputasinya serta jumlah frequent closed itemset yang dihasilkan, kemudian menjalankan algoritma TFP dengan memberikan inputan data set yang ada dan mencatat waktu komputasinya serta jumlah frequent closed itemset yang dihasilkan. Dari catatan waktu komputasi dan jumlah frequent closed itemset yang dihasilkan kedua algoritma tersebut akan dibandingkan. Dari pelaksanaan ujicoba ini, telah ditentukan nilai min_l = 0 dan memberikan nilai Top-k yang beragam untuk setiap dataset yang ada.

Pelaksanaan uji coba untuk skenario yang kedua, menjalankan algoritma Top-k Frequent Closed Itemset lattice pada beberapa dataset yang ada dengan memberikan nilai min_l yang beragam dan nilai top-k yang ditentukan sesuai jumlah transaksi yang ada pada dataset yang ada. Dan dilakukan pencatatan waktu komputasi dari min_l yang beragam untuk setiap dataset.

Sebelumnya dilakukan persiapan dataset, dataset disimpan dalam bentuk file text, dengan format kolom-baris yang diawali dari kolom untuk jumlah item disetiap transaksi seperti yang tampak pada tabel berikut:

Tabel 4.2 Tabel dataset untuk ujicoba

Jumlah item tiap transaksi	Item											
	Item1	Item2	Item3	Item4	Item5	Item6	Item7	Item8	Item9	...		
3	4	5	72									
1	88											
3	119	121	238									
13	5	48	78	118	119	120	121	149	151			
5	0	80	82	83	164							
4	80	84	144	148								
1	148											
...												

Kemudian akan dibuat data spesifikasi untuk setiap dataset yang digunakan. Data spesifikasi ini berisi informasi file dataset yang digunakan, informasi jumlah item dan informasi jumlah transaksi dataset.

4.5 Hasil Uji Coba

Berdasarkan skenario yang dijelaskan pada sub bab 4.3 sebelumnya maka hasil uji coba yang dilakukan sebagai berikut;

Hasil ujicoba yang pertama akan menggunakan dataset chess, connect, pumsb yang mewakili data *real* dengan *type dense* atau rapat, dan dataset gazelle yang mewakili *type sparse*. Dengan memberikan nilai min_l = 0 dan nilai top-k yang beragam maka hasil pencatatan waktu komputasi untuk kedua algoritma dapat dilihat pada tabel 4.3 dan tabel 4.4 berikut ini;

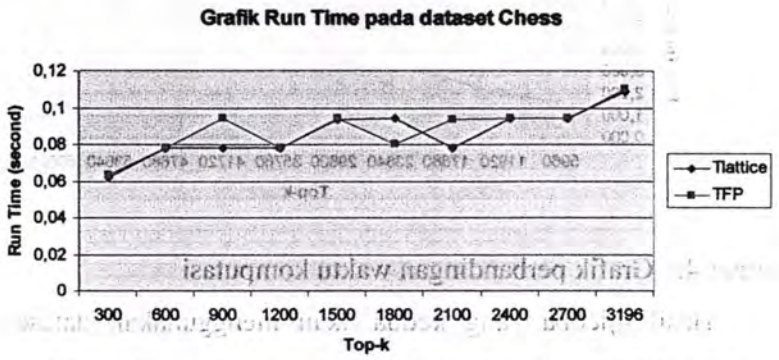
Tabel 4.3 Tabel hasil uji coba skenario pertama dataset chess dan connect

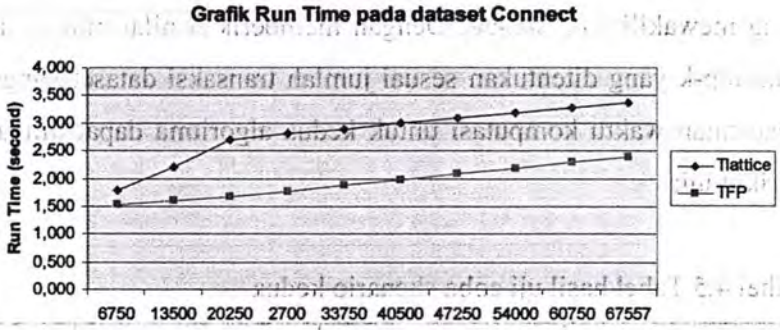
Chess			Connect		
Nilai Top-k	Tlattice (waktu)	TFP (waktu)	Nilai Top-k	Tlattice (waktu)	TFP (waktu)
300	0,062	0,063	6750	1,778	1,515
600	0,078	0,078	13500	2,210	1,594
900	0,078	0,094	20250	2,687	1,672
1200	0,078	0,078	2700	2,797	1,766
1500	0,093	0,094	33750	2,875	1,875
1800	0,094	0,080	40500	2,984	1,969
2100	0,078	0,093	47250	3,078	2,079
2400	0,094	0,094	54000	3,188	2,172
2700	0,094	0,094	60750	3,281	2,282
3196	0,109	0,110	67557	3,375	2,390

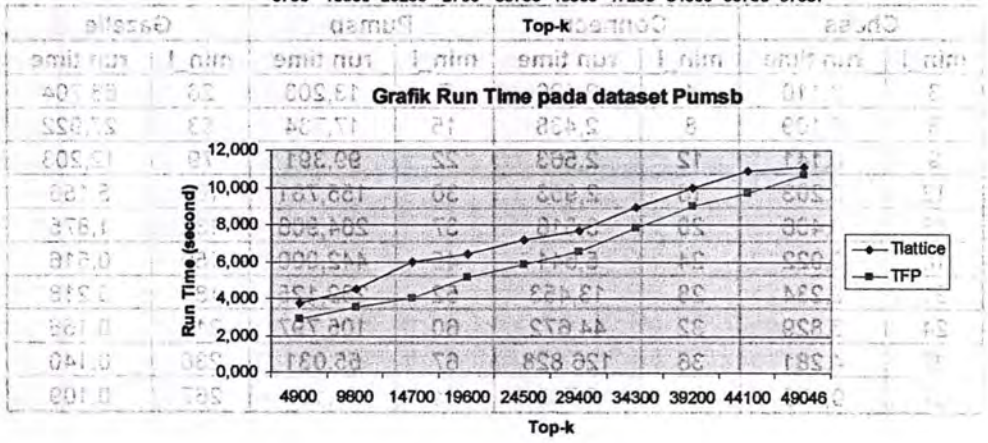
Tabel 4.4 Tabel hasil uji coba skenario pertama dataset pumsb dan gazelle

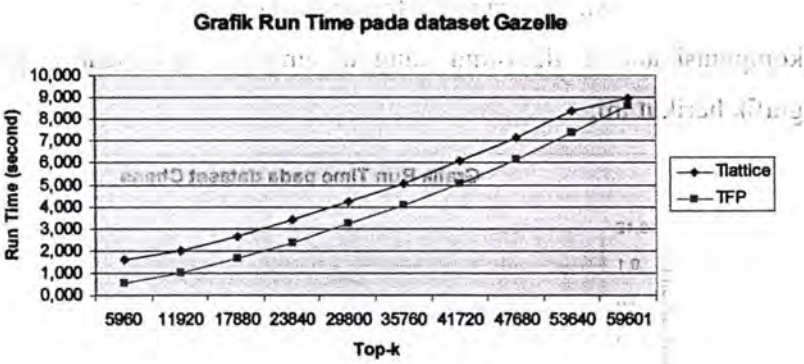
Pumsb			Gazelle		
Nilai Top-k	Tlattice (waktu)	TFP (waktu)	Nilai Top-k	Tlattice (waktu)	TFP (waktu)
4900	3,753	2,859	5960	1,585	0,516
9800	4,500	3,500	11920	2,000	1,000
14700	5,969	4,031	17880	2,616	1,610
19600	6,411	5,125	23840	3,383	2,343
24500	7,182	5,797	29800	4,230	3,219
29400	7,625	6,547	35760	5,056	4,047
34300	8,907	7,782	41720	6,060	5,047
39200	9,953	8,984	47680	7,136	6,125
44100	10,844	9,672	53640	8,379	7,359
49046	11,087	10,672	59601	8,944	8,609

Dari hasil pencatatan tersebut dapat dibandingkan kecepatan waktu komputasi antara algoritma yang dikembangkan dengan algoritma TFP pada grafik berikut ini;









Gambar 4.1 Grafik perbandingan waktu komputasi

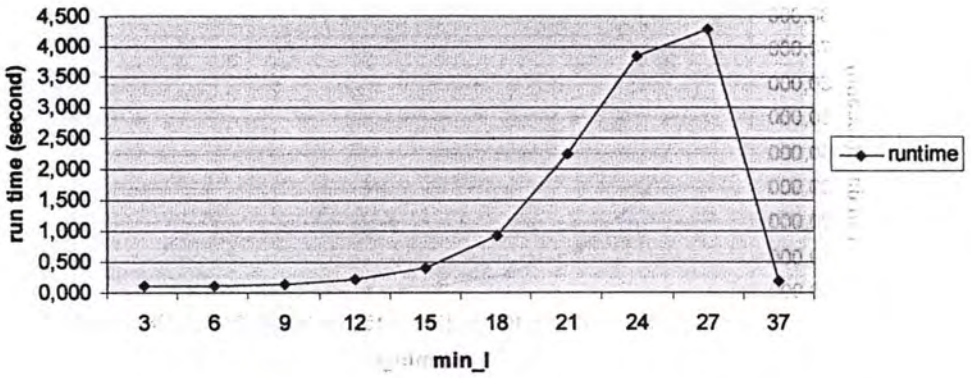
Hasil ujicoba yang kedua akan menggunakan dataset chess, connect, pumsb yang mewakili data *real* dengan *type dense* atau rapat, dan dataset gazelle yang mewakili *type sparse*. Dengan memberikan nilai min_l yang beragam dan nilai top-k yang ditentukan sesuai jumlah transaksi dataset yang ada, maka hasil pencatatan waktu komputasi untuk kedua algoritma dapat dilihat pada tabel 4.5 berikut ini;

Tabel 4.5 Tabel hasil uji coba skenario kedua

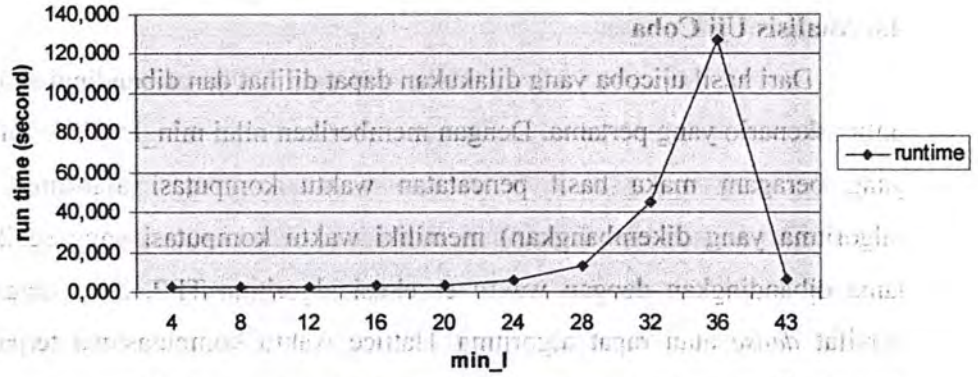
Chess		Connect		Pumsb		Gazelle	
min_l	run time	min_l	run time	min_l	run time	min_l	run time
3	0,110	4	2,406	7	13,203	26	68,704
6	0,109	8	2,438	15	17,734	53	27,922
9	0,141	12	2,563	22	99,391	79	12,203
12	0,203	16	2,953	30	155,781	106	5,156
15	0,406	20	3,516	37	264,906	132	1,875
18	0,922	24	5,641	45	442,000	159	0,516
21	2,234	28	13,453	52	282,125	185	0,218
24	3,829	32	44,672	60	105,797	212	0,156
27	4,281	36	126,828	67	55,031	238	0,140
37	0,172	43	6,703	74	12,141	267	0,109

Dari hasil pencatatan tersebut dapat dibandingkan kecepatan waktu komputasi antara algoritma yang dikembangkan dengan algoritma TFP dengan menentukan nilai min_l yang beragam untuk setiap dataset yang digunakan. Dan hasil pencatatan tersebut dapat dilihat pada grafik berikut ini;

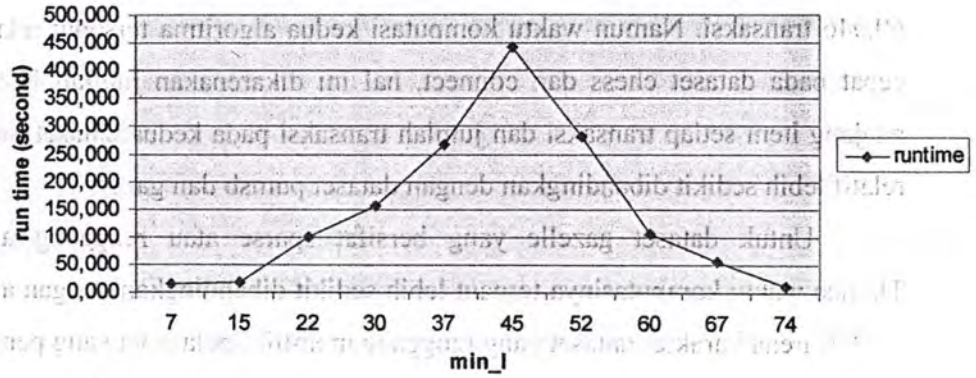
Grafik Kinerja pada dataset Chess

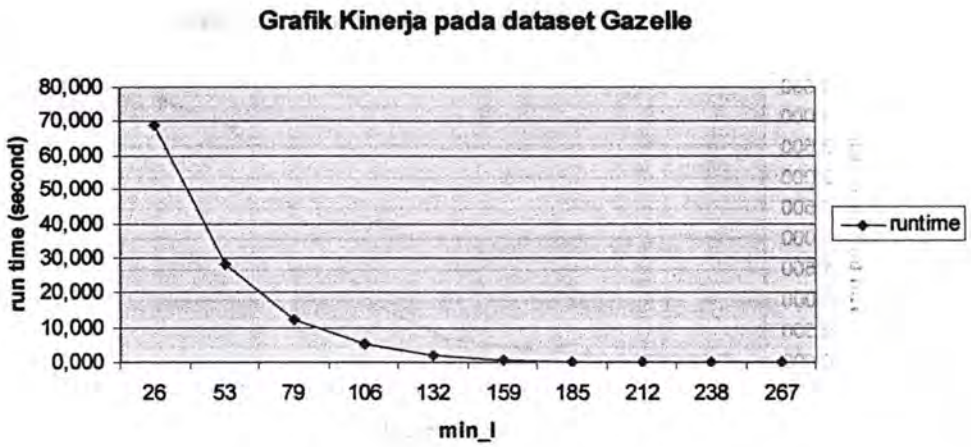


Grafik Kinerja pada dataset Connect



Grafik Kinerja pada dataset Pumsb





Gambar 4.2 Grafik kinerja algoritma Tlattice

4.6 Analisis Uji Coba

Dari hasil ujicoba yang dilakukan dapat dilihat dan dibandingkan hasilnya, untuk skenario yang pertama. Dengan memberikan nilai min_l = 0 dan nilai top-k yang beragam maka hasil pencatatan waktu komputasi algoritma Tlatiice (algoritma yang dikembangkan) memiliki waktu komputasi yang sedikit lebih lama dibandingkan dengan waktu eksekusi algoritma TFP. Pada dataset yang bersifat *dense* atau rapat algoritma Tlattice waktu komputasinya terpaut lebih banyak dibandingkan dengan algoritma TFP, hal ini diakibatkan proses baca dataset pada tahap awal untuk menemukan item-item berbeda yang sering muncul, terutama pada dataset connect dengan jumlah transaksi terbesar yaitu 69.046 transaksi. Namun waktu komputasi kedua algoritma tersebut relatif lebih cepat pada dataset chess dan connect, hal ini dikarenakan jumlah item, rerata panjang item setiap transaksi dan jumlah transaksi pada kedua dataset jumlahnya relatif lebih sedikit dibandingkan dengan dataset pumsb dan gazelle.

Untuk dataset gazelle yang bersifat sparse atau renggang algoritma Tlattice waktu komputasinya terpaut lebih sedikit dibandingkan dengan algoritma TFP. Karena karakter dataset yang renggang memiliki pola-pola yang pendek. Hal ini menyebabkan kandidat frequent item yang dicari menjadi lebih banyak. Dapat dilihat bahwa waktu komputasi akan bertambah seiring bertambahnya nilai k yang diberikan. Dan waktu komputasi kedua algoritma untuk memproses dataset pumsb

dan gazelle diatas 3 detik dan akan terus bertambah seiring bertambahnya nilai k tang diberikan.

Untuk skenario yang kedua, dengan memberikan nilai min_l yang beragam dan nilai top-k yang ditentukan sesuai jumlah transaksi dataset yang ada, maka hasil pencatatan waktu komputasi algoritma Tlattice dapat dilihat pada Gambar 4.2. Dari grafik tersebut padat dilihat, pada dataset yang rapat bentuk grafik waktu komputasinya memiliki puncak. Puncak ini menandakan bahwa waktu komputasi yang paling lama untuk dataset chess (4,281 detik), connect (126,828 detik), dan pumsb (442,000 detik). Pada dataset pumsb mencapai 442,000 detik disebabkan pumsb memiliki jumlah item berbeda terbanyak dibandingkan dengan dataset yang lain.

Untuk dataset gazelle, waktu komputasi algoritma Tlattice membentuk grafik yang berbeda dengan dataset chess, connect dan pumsb. Dataset gazelle yang memiliki sifat yang renggang dengan pola-pola yang pendek, pada nilai min_l yang rendah memerlukan waktu komputasi yang besar 68,704 detik dan waktu komputasi yang cenderung menurun pada nilai top-k yang besar. Hal ini disebabkan karena min_l yang rendah sama artinya dengan memproses banyak kandidat dari frequent itemset.

BAB 5
PENUTUP

BAB 5
PENUTUP

Bab 5 ini menjelaskan mengenai simpulan yang dapat diambil dari analisis hasil ujicoba yang telah dilakukan dalam penelitian pembangkitan kaidah asosiasi dari top-k frequent closed itemset berdasarkan struktur data lattice. Selain itu juga diberikan saran untuk kemungkinan pengembangan lebih lanjut.

5.1 Simpulan

Berdasarkan hasil uji coba pembangkitan kaidah asosiasi dari top-k frequent closed itemset berdasarkan struktur data lattice yang telah dilakukan, dapat diambil beberapa simpulan seperti berikut:

a. Penelitian ini telah berhasil mendesain dan mengimplementsikan algoritma pembangkitan kaidah asosiasi dari top-k frequent closed itemset berdasarkan struktur data lattice. Pengguna dapat membangkitkan kaidah asosiasi dari top-k frequent closed itemset berdasarkan struktur data lattice tanpa harus memberikan nilai *minimum support* (*min_sup*).
b. Dari hasil perbandingan kecepatan waktu komputasi, untuk algoritma top-k frequent closed itemset berdasarkan struktur data lattice sedikit lebih lambat dibandingkan dengan algoritma pembandingnya, yaitu Algoritma TFP. Hal ini dikarenakan algoritma yang dikembangkan harus melakukan proses pembangkitan itemset yang frequent terlebih dahulu, yang kemudian dilanjutkan dengan proses pembangkitan kandidat frequent closed itemset dari struktur lattice yang terbentuk. Di lain pihak, algoritma TFP yang menggunakan FP-tree mampu mengekstrak frequent closed itemset langsung dari struktur tree yang dibangun, sehingga mempercepat waktu komputasinya. Namun algoritma yang dilakukan dalam penelitian ini memiliki kelebihan struktur data berbasis lattice yang dibentuk. Dengan menggunakan struktur data lattice akan memudahkan penentuan hubungan subset dan superset yang berguna untuk pemangkasan kandidat itemset yang dianggap *infrequent*. Hal ini berdampak pada pengurangan ruang pencarian dalam proses pembentukan kaidah asosiasi.

5.2 Kemungkinan Pengembangan Lebih Lanjut

Seperti dijelaskan dalam simpulan, bahwa algoritma top-k frequent closed itemset berdasarkan struktur data lattice memiliki waktu komputasi sedikit lebih lambat dibandingkan dengan algoritma pembandingnya yaitu algoritma TFP. Hal ini dikarenakan algoritma yang dikembangkan dalam penelitian ini harus melakukan proses pembangkitan kandidat frequent closed itemset dari struktur lattice yang terbentuk. Pengembangan lebih lanjut dapat difokuskan pada upaya untuk memperbaiki kecepatan komputasi pada proses pembangkitan kandidat frequent closed itemset dari struktur lattice yang terbentuk.

Daftar Pustaka

Agrawal, R., Imielinski, T., Swami, A. (1993), *Mining association rules between sets of items in large databases*. In: Proc. of SIGMOD. 207–216.

Brin, S., Motwani, R., Silverstein, C. (1997), *Beyond market basket: Generalizing association rules to correlations*. In: Proc. of SIGMOD. 265–276.

Han J., J. Pei, and Y. Yin, (2000), *Mining Frequent Patterns without Candidate Generation*, Proc. 2000 ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD '00), pp. 1-12.

Han J., Kamber M., (2000), *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publishers.

Lu Ying, and Tzvetkov Petre, Han J., (2005), *TFP: An Efficient Algorithm for Mining Top-K Frequent Closed Itemsets. Proc. Sixth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining.*

N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, (1999), *Discovering Frequent Closed Itemsets for Association Rules*, Proc. Seventh Int'l Conf. Database Theory.

S. Brin, R. Motwani, and C. Silverstein, (1997), *Beyond Market Basket: Generalizing Association Rules to Correlations*, Proc. 1997 ACMSIGMOD Int'l Conf. Management of Data (SIGMOD '97), pp. 265-276.

Tan Pan–Ning., Kumar V. and Steinbach M., (2005), *Introduction to Data Mining.* Addison Wesley, USA.

Tan, Pan-Ning., Kumar, V. (2000), *Interestingness measures for association patterns: A perspective*. In: Proc. of Workshop on Postprocessing in Machine Learning and Data Mining.

Zaki, M,J., Mitsunori Ogihara (1998), *Theoretical Foundations of Association Rules*, Computer Science Department, University of Rochester, Rochester NY 14627.

Zaki, M,J. (2000), *Generating Non-Redundant Association Rules*, Proc. Sixth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining.

Zaki M.J. and Hsiao C.-J., (2005), *Efficient Algorithms for Mining Closed Itemsets and Their Lattice Structure*. Technical Report 99-10, Computer Science Dept., Rensselaer Polytechnic Inst.

BIODATA PENULIS

Penulis memiliki nama lengkap **Dian Puspita Hapsari** dilahirkan diSurabaya, Propinsi Jawa Timur 29 Mei tahun 1978. Pendidikan formal SMA diselesaikan penulis pada tahun 1996. Setamat SMA melanjutkan pendidikan S-1 di Jurusan Manajemen Informatika disebuah Sekolah Tinggi Manajemen Informatika dan Teknik Komputer (STIKOM) Surabaya, menyelesaikan studi strata satu pada tahun 2000. Setelah menyelesaikan pendidikan S-1, penulis bekerja sebagai programmer lepas, hingga tahun 2002 menjadi karyawan tetap di Eastjava.com sebagai Designer Web. Pertengahan tahun 2003 bulan Mei diangkat sebagai Dosen Tetap Universitas Narotama hingga sekarang. Pada tahun 2004 mendapat kesempatan tugas belajar dari Universitas Narotama untuk melanjutkan jenjang pendidikan S-2 di Jurusan Teknik Informatika ITS Surabaya dan berhasil lulus pada tahun 2008.