PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK DATA MINING UNTUK PENGGALIAN POLA ASOSIASI DENGAN MENGGUNAKAN ALGORITMA VIPER

TUGAS AKHIR

OLEH :

TONY ARIES WARDIANTO

NRP : 5197100013

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

2003

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK DATA MINING UNTUK PENGGALIAN POLA ASOSIASI DENGAN MENGGUNAKAN ALGORITMA VIPER

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui
Dosen Pembimbing

Dr. Ir. Arif Djunaidy, M.Sc.
NIP. 131 633 403

SURABAYA
FEBRUARI , 2003

kupersembahkan
sebagai wujud baktiku kepada ibu dan bapak
serta kedua kakakku mbak Titien dan mas Tanto

ABSTRAK

Perangkat lunak data mining dapat memberikan kontribusi yang penting dalam dunia bisnis. Pola-pola asosiasi yang merepresentasikan perilaku pelanggan yang dihasilkan dari sebuah perangkat lunak data mining dapat digunakan sebagai bahan pertimbangan dalam pengambilan keputusan dalam suatu perusahaan. Berbagai algoritma telah pernah dikembangkan untuk mendapatkan pola-pola asosiasi dengan mempertimbangkan aspek efektifitas dan efisiensi baik dalam sumber daya yang diperlukan, akurasi hasil, kehandalan dan waktu proses.

Dalam tugas akhir ini dirancang dan diimplementasikan satu aplikasi data mining untuk mendapatkan pola asosiasi dengan menerapkan algoritma yang menggunakan layout data vertikal yaitu Vertical Itemset Partitioning for Efficient Rule-extraction (VIPER). Dalam metode ini, pencarian pola asosiasi dilakukan dengan menggunakan tiga tahap proses utama. Tahap pertama merupakan pembacaan basis data dan pencarian 1-itemset utama. Tahap kedua merupakan tahap untuk mendapatkan 2-itemset utama dengan melakukan enumerasi pada 1-itemset utama. Tahap ketiga merupakan iterasi dengan proses utama, yaitu Fully Organized Candidate-generation (FORC) untuk mendapatkan itemset kandidat dan Fast ANding Graph for Snakes (FANGS) untuk perhitungan support. Beberapa optimasi dilakukan pada kedua proses itu, yaitu optimasi pencarian subset dari itemset kandidat secara simultan dan lazy snake writes untuk pemilihan itemset-itemset yang terlibat pada iterasi selanjutnya.

Perangkat lunak yang telah dibangun diuji coba untuk menggali pola asosiasi pada sejumlah data transaksi sintetis yang dibangkitkan dengan menggunakan data generator. Hasil uji coba perangkat lunak yang dilakukan menunjukkan bahwa algoritma ini dapat mencari pola asosiasi dan mampu melakukan proses mining pada obyek data transaksi dengan kapasitas hampir dua juta record. Sedangkan untuk waktu yang diperlukan dalam proses mining menunjukkan bahwa algoritma ini cenderung lebih cepat jika dibandingkan dengan algoritma Hybrid yang dibahas pada tugas akhir sebelumnya.

KATA PENGANTAR

Alhamdulillah, segala puji bagi Allah SWT atas segala nikmat, rahmat dan karunia-Nya. Semua tidak terlepas dari pertolongan dan kekuatan yang diberikan kepada penulis sehingga penulis dapat menyelesaikan tugas akhir ini dengan baik.

Laporan Tugas Akhir ini dengan judul "**Perancangan dan Pembuatan Perangkat Lunak Data Mining untuk Penggalian Pola Asosiasi dengan Menggunakan Algoritma VIPER**" disusun sebagai dokumentasi sekaligus merupakan penutup rangkaian pelaksanaan Tugas Akhir.

Besar harapan penulis bahwa Laporan Tugas Akhir ini dapat memberikan suatu kontribusi tersendiri bagi wahana ilmu pengetahuan dalam bidang Teknologi Informasi secara umum dan bidang *Information Retrieval* pada khususnya.

Penulis menyadari bahwa masih banyak kekurangan dalam penyusunan Tugas Akhir ini. Untuk itu kritik dan saran yang membangun akan diterima dengan terbuka.

Akhir kata, penulis mengucapkan selamat membaca dan semoga karya Tugas Akhir ini dapat memberikan manfaat yang sebesar-besarnya bagi generasi sekarang dan yang akan datang.

Surabaya, Januari 2003

Penulis

UCAPAN TERIMA KASIH

Penulis dalam kesempatan ini mengucapkan terima kasih dan penghargaan yang sebesar-besarnya bagi semua pihak yang telah membantu dalam pelaksanaan Tugas Akhir, baik secara langsung maupun tidak langsung. Secara khusus penulis mengucapkan terima kasih kepada :

1. Kedua orang tua tercinta penulis, Bapak Suwardi dan Ibu Mudjiati. Terima kasih yang tak terhingga atas kasih sayang, arahan, dukungan, didikan, dan segala bantuan yang telah bapak dan ibu berikan. Hanya Allah SWT yang mampu membalasnya.
2. Saudara-saudara penulis, Mbak Titien, Mas Tanto, Mas Ir yang selalu mendukung dengan sabar serta segala bantuan dan semangat yang telah diberikan hingga penulis selesai kuliah.
3. Bapak Dr. Ir. Arif Djunaidy, MSc selaku dosen pembimbing dan dosen wali selama masa perkuliahan. Terima kasih atas dorongan semangat, bimbingan dan saran serta dukungan yang telah banyak diberikan.
4. Bapak Agus Zaenal Arifin, SKom MKom, selaku ketua Jurusan Teknik Informatika.
5. Seluruh staf pengajaran dan tata usaha Jurusan Teknik Informatika, Pak Yudi, Pak Soleh, Mbak Erna, Mbak Eva, Mas Sugeng, Mas Hudan, Mas Hermono serta para staf lainnya.
6. Bapak Drs. EC. Ir. Riyanarto Sarno MSc. PhD, selaku Direktur Utama PT. Limaxindo Intersistem.
7. Bapak Luthfi Hidayat, ST., S.Sos. selaku manager Product Development PT. Limaxindo Intersistem.

8. Seluruh LIMAXers. Terima kasih atas segala dukungan yang diberikan selama ini. Tetap solid!
9. Teman-teman sesama pejuang TA, mas Reza, mas Heri, mas Puji, Susane, Ochin, Vicka, Sinyo Hengky dan Sinyo Rezin (xie xie ni), Rahardhono, Bimo. Sukses selalu!
10. Maimun, Aris, Kurnix, Arif, Ajun, Agus, Asmunin, Badrus, Dees, Daning, Hera, Tita dan semua teman-teman C-0D yang tidak bisa penulis sebutkan satu-persatu.
11. Teman-teman kampung kos GL-10 baik kubu tua maupun kubu muda. Terima kasih atas segala provokasinya yang membantu penulis untuk lebih semangat menyelesaikan Tugas Akhir ini.

Akhir kata, penulis memohon maaf yang sebesar-besarnya apabila penulis tidak sengaja pernah melakukan atau mengatakan hal-hal yang tidak berkenan.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dewasa ini aplikasi basis data telah berkembang dan digunakan secara luas dalam berbagai bidang, misalnya ilmu pengetahuan, pemerintahan, bisnis dan lain-lain. Perkembangan ini menyebabkan semakin besarnya kapasitas data yang disimpan. Kumpulan data ini merupakan aset yang berharga sehingga muncul pemikiran untuk memanfaatkan dan mengolahnya menjadi informasi yang lebih berarti.

Data mining merupakan teknik yang digunakan untuk menggali informasi berupa pola dan aturan yang memiliki makna yang secara implisit terdapat dalam suatu basis data dengan kapasitas yang sangat besar.

Sejak dimunculkan teknik data mining khususnya penggalian pola asosiasi pada data transaksi pembelian, mulailah berkembang berbagai algoritma untuk penyelesaian masalah ini. Diantara algoritma yang muncul memberikan perbaikan maupun alternatif penyelesaian dengan memperhitungkan aspek-aspek penting dalam proses *mining*. Hal penting yang harus diperhatikan dalam penerapan aplikasi data mining antara lain keakuratan hasil yang diperoleh, kehandalan dalam mengatasi obyek mining dengan kapasitas yang besar, efektifitas serta efisiensi algoritma yang digunakan.

Algoritma-algoritma untuk penggalian kaidah asosiasi sebelumnya menggunakan obyek basis data dengan *layout* horisontal. Pada *layout* ini setiap baris data merepresentasikan transaksi pembeli dan item yang dibeli pada transaksi tersebut. Beberapa tahun terakhir ini muncul suatu pendekatan

alternatif terhadap obyek basis data yang digali, yaitu basis data dengan *layout* vertikal dimana setiap kolom data merepresentasikan ada tidaknya suatu item dalam suatu transaksi. Dari beberapa algoritma dinyatakan bahwa basis data dengan *layout* vertikal lebih menguntungkan daripada *layout* horisontal.

Salah satu algoritma yang efektif dan menggunakan obyek basis data dengan *layout* vertikal adalah algoritma **V**ertical **I**temset **P**artitioning for **E**fficient **R**ule-extraction (**VIPER**) yang digunakan dalam Tugas Akhir ini.

1.2 Permasalahan

Adapun beberapa rumusan permasalahan yang ada adalah sebagai berikut:

- Bagaimana mendapatkan data input yang sesuai dengan kebutuhan perangkat lunak.
- Bagaimana merancang struktur data yang tepat sehingga kinerja perangkat lunak dengan menerapkan algoritma VIPER menjadi optimal.
- Bagaimana mendesain antar muka yang baik sehingga mudah digunakan oleh pengguna.

1.3 Batasan Masalah

Batasan masalah dalam Tugas Akhir ini antara lain:

- Data transaksi yang akan digunakan sebagai obyek *mining* merupakan data hasil pembangkitan perangkat lunak *data generator* [ADH-02] dan telah digunakan sebagai obyek mining pada Tugas Akhir sebelumnya.
- Untuk uji perbandingan dengan algoritma data mining lainnya digunakan hasil uji coba pada metode *Hybrid* [DAN-02].

1.4 Tujuan dan Manfaat

Tujuan pembuatan Tugas Akhir ini adalah:

- Merancang dan membangun perangkat lunak data mining untuk menemukan pola asosiasi dengan mengimplementasikan algoritma VIPER.
- Melakukan pemilihan struktur data, perancangan proses dan keluaran yang tepat sehingga diperoleh efektifitas dan efisiensi dalam penggunaan memori, waktu dan aspek lainnya.

Manfaat yang dapat diperoleh adalah:

- Memperoleh informasi seluruh pola asosiasi yang terbentuk atau yang sesuai dengan permintaan pengguna.
- Prototip penggalian pola asosiasi yang dibangun dapat dikembangkan dalam membangun aplikasi yang lebih spesifik dengan menggunakan data riil.

1.5 Metodologi

Pengerjaan perangkat lunak ini dibagi dalam beberapa tahap yang masing-masing akan dijelaskan sebagai berikut:

- Studi literatur

  Pada tahap ini dilakukan pemahaman literatur beberapa referensi jurnal data mining berkaitan dengan penggunaan algoritma VIPER, pemahaman setiap tahap yang dilakukan, alur proses yang berlangsung serta optimasi yang dilakukan selama proses *mining*. Selain itu juga dipelajari algoritma pada beberapa Tugas Akhir yang berbeda sebagai bahan perbandingan.

- Perancangan perangkat lunak

  Berdasarkan hasil studi literatur, pada tahap ini dilakukan analisa dan perancangan struktur data yang sesuai untuk mengimplementasikan algoritma VIPER sehingga kinerjanya dapat optimal dan perancangan proses dengan pendekatan fungsional yang divisualisasikan dengan menggunakan Diagram Aliran Data (DAD). Dalam DAD ini digambarkan alur data yang diberikan oleh pengguna, proses-proses yang berlangsung dan keluaran yang akan diperoleh.

- Pembuatan perangkat lunak

  Pada tahap ini dilakukan implementasi dengan membangun satu perangkat lunak berdasarkan perancangan yang telah dibuat dan menggunakan *compiler* bahasa pemrograman Java. Dalam tahap ini juga dijelaskan struktur dan penggunaan perangkat lunak yang dibangun.

- Uji coba dan evaluasi

  Pada tahap ini program yang telah dibuat diuji pada beberapa kriteria pengujian yang berbeda dan menggunakan data uji yang dibangkitkan dengan menggunakan *data generator* [ADH-02] dan telah digunakan sebagai obyek *mining* pada Tugas Akhir sebelumnya. Dari hasil uji coba ini dilakukan evaluasi beberapa hubungan antara obyek data transaksi, jumlah pola asosiasi, waktu proses serta parameter *mining* yang digunakan. Selain itu juga dilakukan perbandingan waktu proses *mining* dengan aplikasi pada Tugas Akhir sebelumnya.

- Penyusunan buku Tugas Akhir

  Pada tahap terakhir ini dilakukan penyusunan buku sebagai laporan hasil dokumentasi pelaksanaan Tugas Akhir.

1.6 Sistematika Penulisan

Dalam penyusunannya, laporan Tugas Akhir ini dibagi dalam enam bab yang membahas bagian-bagian sebagai berikut:

Bab I Pendahuluan

Dalam bab ini dijelaskan beberapa hal pokok tentang Tugas Akhir yang dilakukan yaitu latar belakang, permasalahan dan batasannya, tujuan dan manfaat, metodologi serta sistematika penulisan buku Tugas Akhir.

Bab II Data Mining

Bab ini berisi konsep dasar data mining dan contoh penggunaan aplikasinya dalam kehidupan, hubungan antara data *warehouse*, OLAP dan data *mining* serta cara kerja dan jenis-jenis teknik data *mining*.

Bab III Pencarian Pola Asosiasi Menggunakan Algoritma VIPER

Dalam bab ini dijelaskan setiap tahap yang dilakukan dalam algoritma VIPER untuk proses pencarian pola asosiasi. Untuk penjelasan ini digunakan satu contoh data transaksi sederhana.

Bab IV Perancangan dan Pembuatan Perangkat Lunak

Menguraikan perancangan data, perancangan proses dengan menggunakan DAD dan perancangan antar muka dari perangkat lunak yang dibuat. Pada sub bab selanjutnya dijelaskan mengenai implementasi struktur data, *pseudocode* untuk pengimplementasian perancangan proses dan tampilan antar muka serta fungsi-fungsi pada perangkat lunak tersebut.

Bab V Uji Coba dan Evaluasi

Bab ini berisi penjelasan proses uji coba dan evaluasi perangkat lunak dengan menggunakan beberapa kriteria pengujian yang berbeda dan dilakukan pada lingkungan pengujian tertentu terhadap obyek data transaksi sintetis.

Bab VI Penutup

Bab ini merupakan bagian terakhir dari buku Tugas Akhir yang berisi kesimpulan berdasarkan uji coba perangkat lunak dan kemungkinan pengembangan lebih lanjut yang dapat dilakukan.

BAB II

DATA MINING

BAB II

DATA MINING

Dalam bab ini diuraikan mengenai hal mendasar berhubungan dengan data mining, pengertian, penerapannya dalam kehidupan, hubungannya dengan data *warehouse* dan OLAP, cara kerja serta jenis-jenis teknik dalam data mining.

2.1 Pengertian Data Mining

Data mining adalah eksplorasi dan analisa dari data dalam ukuran besar untuk menemukan pola dan aturan yang mempunyai makna [LIN-97].

Dengan menggunakan teknik data mining maka pola dan aturan yang tersirat dalam suatu basis data dengan kapasitas yang sangat besar dapat diperoleh. Pola-pola yang diperoleh tersebut dapat berupa aturan asosiasi, pola sequensial dan lain-lain.

Mengingat kapasitas obyek basis data yang sangat besar maka dalam teknik data mining sangat diperhatikan efektifitas, efisiensi dan fisibelitas algoritma yang digunakan dalam penggalian pola dan aturan tersebut.

Salah satu pola dan aturan dalam basis data yang akan digali dalam Tugas Akhir ini adalah pola asosiasi. Pola asosiasi menggambarkan hubungan antar item dimana ketika suatu item dibeli pada suatu transaksi maka item yang lain juga dibeli.

2.2 Perkembangan Teknik Data Mining

Sejak awal diperkenalkannya teknik data mining, ilmu ini mengalami perkembangan yang pesat. Berbagai macam perangkat lunak yang menerapkan teknik ini banyak ditawarkan untuk membantu penyediaan informasi dalam bidang-bidang tertentu. Ada beberapa aspek yang mendukung pesatnya perkembangan ilmu ini antara lain:

- Saat ini semakin meluasnya institusi-institusi baik perusahaan, institusi pemerintahan maupun institusi pendidikan yang menggunakan aplikasi basis data. Penggunaan aplikasi ini akan menghasilkan akumulasi data dalam jumlah yang besar. Data merupakan sumber data potensial yang dapat digali untuk mendapatkan informasi yang lebih bernilai.
- Informasi yang diperoleh dari proses data mining lebih lanjut sangat berguna untuk membantu para pembuat keputusan dalam menentukan langkah-langkah strategis yang akan diambil.
- Perkembangan perangkat keras komputer yang pesat juga mendukung perkembangan ilmu ini mengingat bahwa sumber data yang menjadi obyek mining juga semakin besar sehingga memerlukan perangkat keras yang baik untuk memproses hal itu.
- Orang-orang yang bergelut dalam teknik data mining selalu melakukan riset untuk mendapatkan algoritma data mining yang paling efektif dan efisien. Efektif berarti suatu algoritma menggali informasi dari sumber data yang sangat besar dengan tetap mempertimbangkan keterbatasan kemampuan perangkat keras dan efisien berarti waktu yang diperlukan dalam proses mining seminimal mungkin.

2.3 Penggunaan Data Mining dalam Kehidupan

Aplikasi data mining sudah mulai banyak diterapkan dalam bidang kehidupan. Umumnya dalam dunia bisnis ilmu ini dikembangkan untuk membantu para pembuat keputusan dalam penyediaan informasi yang mendukung dalam penentuan keputusan. Berikut adalah contoh keuntungan yang diperoleh dari penggunaan aplikasi data mining, yaitu:

- Menarik pelanggan baru

  Perusahaan yang memiliki *data warehouse* dapat menggunakan aplikasi data mining untuk menganalisa pelanggan yang ada, mendapatkan karakteristik konsumen yang menggunakan produk maupun jasa perusahaan. Dari karakteristik tersebut dapat dicari calon pelanggan yang potensial dan memiliki karakteristik yang serupa dalam model sehingga dapat ditentukan target dan cara menarik pelanggan baru.

- Memperkuat hubungan dengan pelanggan yang ada

  Dengan menggunakan data transaksi oleh pelanggan dapat diperoleh informasi mengenai produk baru maupun jasa yang memiliki prospek yang baik. Dari data tersebut kemudian ditentukan kriteria pelanggan yang memiliki kecenderungan untuk memilih produk baru tersebut. Dengan demikian dapat diketahui pelanggan yang dapat memberikan profit dari produk baru yang dikeluarkan.

- Pengurangan kecurangan (fraud)

  Data mining dapat digunakan untuk identifikasi kecurangan penggunaan kartu kredit, menemukan pola tingkah laku dari pelanggan yang cenderung beresiko serta menentukan penerbitan kartu kredit dengan melihat rasio kemampuan pembayaran tagihan terhadap suatu pelanggan.

- Pengawasan internet

  Perusahaan periklanan di internet maupun organisasi lain yang memilki aplikasi berbasis web dapat menggunakan "cookies" untuk mengumpulkan data para pengunjung web mereka. *Cookies* tersebut dapat digunakan untuk membuat profil pengguna sehingga memenuhi target iklan yang lebih baik. Informasi dikumpulkan dari alamat IP, penentuan lokasi geografis pengguna dan *site* yang dikunjungi.

Sebagai salah satu contoh aplikasi yang menerapkan teknik data mining yaitu website **amazon.com** yang menggali informasi pada *market basket data* dengan mempelajari karakteristik pelanggan dari jenis barang yang dibeli. Pada gambar 2.1 dibawah dapat dilihat bahwa ketika seorang pengunjung web amazon memasukkan satu buah CD audio yang dipilih dalam *shopping cart,* sistem web amazon secara otomatis memberikan informasi tambahan pada bagian bawah yang menunjukkan beberapa CD audio dari penyanyi yang berbeda yang pada umumnya juga dibeli oleh pengunjung lain yang membeli CD yang dipiih tersebut. Pada contoh ini dapat dilihat penggunaan pola asosiasi dalam data transaksi pembelian yang menunjukkan hubungan antar itemset, misalnya asosiasi untuk barang A dan B, ketika barang A ada dalam transaksi maka sekian persen dari jumlah transaksi menyatakan bahwa barang B juga ada dalam transaksi tersebut.

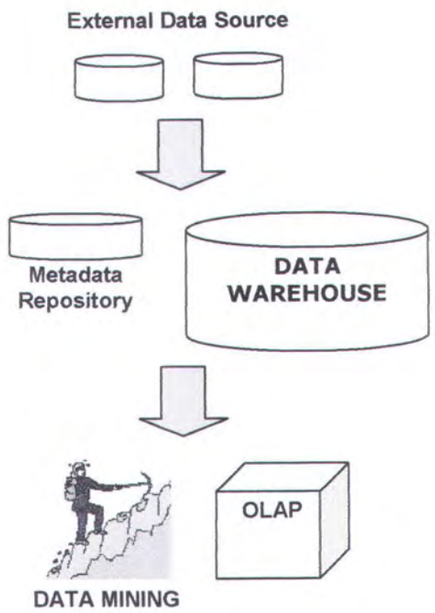
Permasalahan utama dalam *data warehouse* adalah:

- *Semantic Integration*, menghilangkan ketidaksesuaian data misalnya akibat perbedaan mata uang maupun skema.
- *Heterogenous Sources*, pengaksesan data dari berbagai format sumber data maupun repositori, disini dapat memanfaatkan hasil replikasi data.
- *Load, Refresh and Purge*, berkaitan dengan pengambilan, refresh dan penghapusan data yang sudah tidak valid.
- *Metadata Management*, menjaga informasi mengenai seluruh data dalam warehouse.

OLAP (*On-Line Analytical Processing*) berkaitan dengan teknik analisa dataset yang digunakan untuk menganalisa data pada *data warehouse*. *Data warehouse* yang menyimpan data yang besar membentuk suatu multidimensional data dan disebut sebagai *multidimensional database* atau *data cube*. Operasi-operasi dalam OLAP merupakan operasi untuk berinteraksi dengan *data cube* untuk analisa data multidimensional. Operasi tersebut pada beberapa sistem memasukkan fungsi untuk analisa statistik, misalnya analisa trend, rasio dan peringkat, analisa iteratif, analisa time-series dan lain-lain. [HAN-97]

Data Mining menyangkut analisa data dan eksplorasi lebih dalam terhadap sumber data tersebut. Data mining berkaitan dengan ilmu *machine learning*, pemrograman matematis dan ilmu statistika serta dilakukan terhadap obyek berupa dataset yang sangat besar.

Gambar berikut menunjukkan hubungan antara data warehouse, OLAP dan data mining.



Gambar 2.2 Data Warehouse-OLAP-Data Mining

2.6 Jenis-Jenis Teknik Data Mining

Software data mining akan menganalisa hubungan dan pola yang tersimpan dalam data transaksi. Secara umum, ada empat teknik dan tipe relasi yang dapat digali, yaitu:

- *Classes*, relasi ini digunakan untuk mendapatkan suatu data dari data transaksi yang disimpan dan telah dimasukkan sebelumnya dalam kelompok tertentu. Sebagai contoh pada kasus restoran yang mendapatkan data kapan dan apa yang dipesan oleh pelanggan sehingga dari data tersebut dapat ditentukan menu spesial pada saat-saat tertentu.
- *Clusters*, item data dikelompokkan berdasarkan hubungan logikal. Item data yang memiliki kemiripan dikelompokkan dalam kluster yang sama sedangkan yang berbeda dikelompokkan dalam kluster lainnya.

- *Associations*, pola yang menunjukkan hubungan antara dua item atau lebih dimana ketika suatu item dibeli maka item yang lain juga memiliki kecenderungan dibeli dalam transaksi tersebut.
- *Sequential Patterns*, pola ini hampir sama dengan association namun dengan memperhatikan urutan item (*sequence*).

BAB III

PENDEKATAN ALGORITMA PADA PROSES DATA MINING

BAB III

PENDEKATAN ALGORITMA VIPER

PADA PROSES DATA MINING

Dalam bab ini diuraikan mengenai pola asosiasi, keuntungan, fitur dan tahap-tahap dalam algoritma VIPER untuk mendapatkan pola asosiasi dengan menggunakan satu contoh data transaksi sederhana serta analisa kompleksitas algoritmanya.

3.1 Pengertian Pola Asosiasi

Pola asosiasi merupakan satu tipe relasi yang digali dari satu data transaksi (*market basket data*) dengan memasukkan parameter penggalian yang diinginkan oleh pengguna. Pola ini menunjuk pada hubungan antar item data dimana ketika suatu item ada dalam transaksi maka item yang lain juga ada dalam transaksi tersebut. Contoh penggunaan pola asosiasi ini dalam satu aplikasi telah dijelaskan pada sub bab 2.3, yaitu pada kasus pembelian item barang di web amazon.com.

3.2 Istilah Penting

Dalam proses pencarian pola asosiasi terdapat beberapa istilah dan notasi yang harus diketahui sehingga dapat memudahkan pemahaman pada pembacaan bab-bab selanjutnya.

Tabel 3.1 Istilah dan notasi pada data mining

Istilah	Notasi	Keterangan
Itemset	I	Himpunan item dalam suatu data transaksi
Database	D	Basis data yang menyimpan data transaksi pembelian
Minimum Support	$minsup$	Nilai yang ditentukan oleh pengguna yang menunjuk pada jumlah atau prosentase minimum kemunculan suatu item dalam data transaksi untuk dapat memenuhi *frequent*.
Confidence Support	$confsup$	Nilai yang ditentukan oleh pengguna yang menunjuk pada probabilitas minimum hubungan antar item untuk dapat memenuhi kaidah asosiasi.
Support	$\sigma(X)$	Jumlah transaksi dimana item X berada sebagai subset
Frequent	F	Himpunan item dalam D yang memenuhi *minsup*
Frequent k-Itemset	F_k	Himpunan k-itemset dalam D yang memenuhi *minsup*
Candidate k-Itemset	C_k	Himpunan k-item yang menjadi kandidat dalam D untuk menjadi *frequent* itemset. C_k belum melalui proses perhitungan *support*
	S_k	C_k atau F_k
Transaction ID	TID	ID Transaksi
Item ID	IID	ID Item

- *tid-vector* merupakan format penyimpanan data dalam bentuk vektor yang berisi bit 0 dan 1 yang menunjukkan ada tidaknya itemset dalam suatu transaksi.
- *tid-list* merupakan format penyimpanan data dalam bentuk vektor yang berisi ID transaksi (TID) dimana suatu itemset ada dalam transaksi tersebut.
- *snake* merupakan data bitset untuk setiap itemset yang berisi bit 0 dan 1 yang merepresentasikan ada tidaknya item tersebut dalam suatu transaksi.

Pada tabel berikut ini terdapat satu contoh item-item data yang akan digunakan dalam tabel transaksi sederhana yang akan dijadikan sebagai obyek *mining* dalam pembahasan secara bertahap pencarian pola asosiasi dengan menggunakan algoritma VIPER.

Tabel 3.2 Contoh item data dan kodenya

Nama Buah	Kode
Anggur	A
Belimbing	B
Mangga	M
Rambutan	R
Semangka	S

Tabel 3.3 dibawah ini merupakan contoh data transaksi yang berisi item-item data yang disebutkan pada tabel 3.2 di atas.

Tabel 3.3 Contoh data transaksi

TID	ITEM(s)
1	A B R S
2	B M S
3	A B R S
4	A B M S
5	A B M R S
6	B M R

Dengan mengacu pada tabel diatas dan diasumsikan nilai *minsup* adalah 3 maka 1-itemset utama (frequent itemset) yang memenuhi *frequent* (memenuhi *minsup*) adalah:

A → support = 4 M → support = 4 S → support = 5

B → support = 6 R → support = 4

Support untuk item A adalah empat karena jumlah kemunculan dalam seluruh transaksi ada empat yaitu pada TID=1, 3, 4 dan 5. Perhitungan yang sama juga

dilakukan terhadap item yang lain sehingga hasil 1-itemset utama dinotasikan sebagai F_1 = { A, B, M, R, S }.

3.3 Keuntungan Menggunakan Algoritma VIPER

Data transaksi pembelian secara umum direpresentasikan dalam matrik dua dimensi dengan baris menunjukkan TID dan kolom menunjukkan IID. Data transaksi pada tabel 3.3 diatas merupakan satu contoh bentuk umum *layout* obyek *mining*.

Pada dasarnya representasi data transaksi dapat disimpan dalam beberapa *layout* antara lain:

- Horisontal Item Vektor (HIV)

  Basis data dikelompokkan dalam baris sehingga tiap baris akan menyimpan TID dan bit 0 dan 1 yang menunjukkan ada tidaknya suatu item pada TID tersebut.

- Horisontal Item List (HIL)

  Basis data dikelompokkan dalam baris sehingga tiap baris akan menyimpan TID dan daftar IID yang dibeli pada TID tersebut.

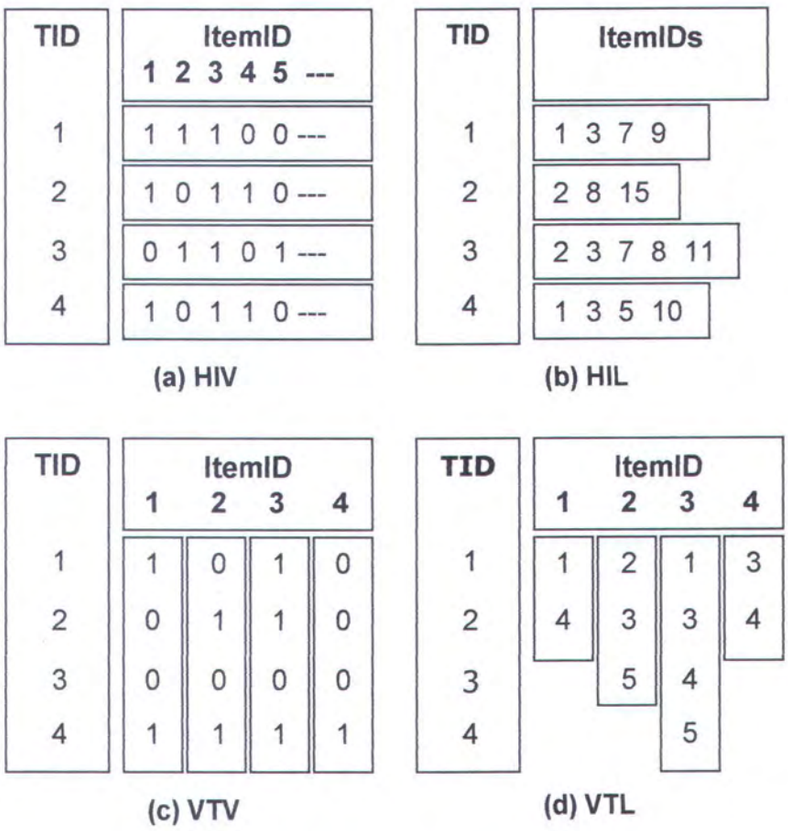
- Vertikal TID Vektor (VTV)

  Basis data dikelompokkan dalam kolom sehingga tiap kolom akan menyimpan IID dan bit 0 dan 1 yang menunjukkan ada tidaknya suatu item pada setiap TID.

- Vertikal TID List (VTL)

  Basis data dikelompokkan dalam kolom sehingga tiap kolom akan menyimpan IID dan daftar TID dimana IID tersebut dibeli.

Untuk lebih jelas, contoh gambar untuk masing-masing *layout* data transaksi diatas adalah sebagai berikut:



Gambar 3.1 Perbandingan layout data transaksi

Algoritma VIPER yang digunakan dalam Tugas Akhir ini melibatkan penyimpanan data item dengan layout VTV. Penyimpanan data dalam bentuk himpunan bit ini memungkinkan penggunaan memori secara hemat.

Ada beberapa keuntungan yang dapat diperoleh pada proses mining dengan menggunakan obyek basis data dalam *layout* vertikal, antara lain:

- Dengan menggunakan *layout* vertikal maka memudahkan proses perhitungan *support* karena dapat diperoleh hanya dengan melakukan interseksi pada *tid list*. Hal ini lebih sederhana dibandingkan penggunaan *hashtree* yang komplek seperti yang digunakan pada algoritma apriori.

- Reduksi terhadap data item dilakukan sebelum proses pembacaan sehingga hanya himpunan item yang berhubungan dengan proses *mining* selanjutnya yang akan dibaca dari memori.
- Memungkinkan komputasi yang asinkron pada itemset utama karena itemset pada level yang lebih tinggi dapat dihitung meskipun seluruh item pada level sebelumnya belum seluruhnya selesai dihitung. Sebagai contoh adalah suatu data item A, B dan C. Setelah *support* A dan B diketahui maka perhitungan support AB dapat dilakukan walaupun perhitungan support C belum selesai.

3.4 Fitur efisiensi pada Algoritma VIPER

Algoritma VIPER menyimpan data bitset untuk setiap data item yang berguna dalam proses mining selanjutnya. Pada setiap iterasi terjadi pergantian dan reduksi pada data item ini sehingga alokasi memori untuk penyimpanan data item berkurang akan tetapi bertambah untuk penyimpanan pola asosiasi yang terbentuk. Selain itu juga dilakukan beberapa optimasi pada beberapa bagian algoritma yang dapat mempercepat proses *mining* ini. Optimasi yang dilakukan antara lain:

- Optimasi saat pembangkitan itemset kandidat (*candidate* itemset) dengan menggunakan FORC yang mengacu pada metode *equivalence class clustering* sehingga dengan menggunakan data S_k dapat diperoleh C_{k+1} hingga C_{2k}. Selain itu pada proses FORC juga terdapat optimasi dalam pengecekan subset dari tiap itemset kandidat.
- Optimasi reduksi *snake* yang akan disimpan dalam memori dengan melakukan pemilihan *snake* yang memiliki potensi untuk digunakan pada proses mining berikutnya.

- Optimasi proses perhitungan *support* dengan hanya melakukan interseksi yang konkuren pada pasangan *snake* dengan menggunakan strategi *pipeline*. Hal ini memungkinkan perhitungan asinkron yang lebih cepat.

3.5 Algoritma VIPER

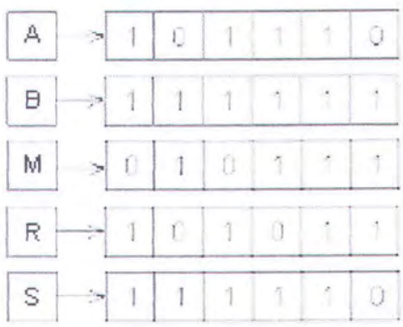
Berikut ini akan dijelaskan secara bertahap proses pencarian k-itemset utama dengan menggunakan algoritma VIPER. Secara garis besar algoritma VIPER terbagi dalam tiga tahap, yaitu tahap pertama, tahap kedua dan tahap subsequent. Dalam penjelasan ini digunakan data transaksi pada tabel 3.3 sebagai obyek *mining* dengan asumsi *minsup*=50% dan *confsup*=100%.

3.5.1 Tahap Pertama

Pada tahap ini proses yang dilakukan adalah:

- Pembacaan dan konversi data dari tabel dalam basis data menjadi format VTV.
- Perhitungan *support* tiap item dilakukan bersamaan pada saat pembacaan data sehingga pada akhir tahap ini akan diperoleh 1-itemset utama(F_1).
- Data tiap item dalam layout VTV (snake) yang memenuhi minimum support tetap disimpan dalam memori.

Data tiap item dalam format VTV, seperti pada gambar 3.2, tersimpan sebagai himpunan bit 0 dan 1 yang akan digunakan untuk proses mining selanjutnya.



Gambar 3.2 Data snake untuk tiap item data

Gambar 3.3 menunjukkan *pseudocode* untuk tahap pertama. Kompleksitas waktu untuk proses ini adalah O(n) karena terjadi perulangan pada baris 2 hingga 18 sebanyak n dimana n adalah jumlah record yang dibaca dari tabel transaksi.

```
1. rs = query("select TID,ITEM from table order by ITEM,TID")
2. while (rec = ambil_record_dari_rs) {
3.   cTID = ambil_nilai(TID)_dari_rec
4.   cItem = ambil_nilai(ITEM)_dari_rec
5.   if ((cItem = curItem))||(rec_pada_baris_awal)) {
6.     curItem = cItem
7.     set_bit_di_VTV_pada_index(cTID)
8.     supp = supp + 1
9.   } else if (cItem <> curItem) {
10.    if (supp>=minsup) {
11.      F1 = F1 U data(curItem,supp)
12.      hGenCover = hGenCover U data(curItem,VTV.clone)
13.    }
14.    VTV.andNot(VTV)
15.    set_bit_di_VTV_pada_index_cTID
16.    supp = 1
17.  }
18.}
19.if (supp>=minsup) {
20.  F1 = F1 U data(curItem,supp)
21.  hGenCover = hGenCover U data(curItem,VTV)
22.}
23.F = F U F1

/* keterangan:
VTV = bitset tiap itemset
hGenCover = hashtable untuk seluruh generator cover
F1  = vektor frequent 1-itemset
F   = obyek berisi seluruh data frequent itemset
```

Gambar 3.3 *Pseudocode* tahap pertama

Pada akhir tahap ini *frequent 2-itemset* (F_2) yang diperoleh adalah:

F2 = {AB, AR, AS, BM, BR, BS, MS, RS}.

Gambar 3.5 di bawah menunjukkan *pseudocode* untuk tahap kedua. Kompleksitas untuk baris1 adalah O(n*log(n)), baris 2-7 adalah O(nm), baris 8-9 adalah $O((n-1)^2)$, baris 10-13 adalah O(m) dan baris 14-18 adalah O(m) dimana n merupakan jumlah elemen F1 dan m merupakan jumlah TID. Kompleksitas total *pseudocode* ini adalah $O(n*\log(n)+nm+(n-1)^2+2m)$.

```
1. iSort = sort_element(F1)
2. for each element mX in iSort {
3.    tmpB = ambil_data_bitset_pada_hGenCover_dengan_key(mX)
4.    for (j=0;j<jumlah_TID;j++)
5.      if (bit_ke-j_pada_bitset_tmpB = 1)
6.        ctSrc[j] = ctSrc[j] U iSort[i]
7. }
8. while (combV = generate_pasangan(F1))
9.    F2 = F2 U data(combV,0)
10.for each element xCt in ctSrc {
11.   while (combV = kombinasi(xCt))
12.     tambah_nilai_support_element_dengan_key(combV)pada_F2
13.}
14.for each keys xStr in F2 {
15.   supp = ambil_support_element_ dengan_key(xStr)_pada_F2
16.   if (supp<minsup)
17.     F2 = F2 - data(xStr)
18.}
19.F = F U F2;

/* keterangan:
F2    = hashtable untuk triangle array 2-itemset utama
ctSrc = array horisontal tuple
F     = obyek berisi seluruh data frequent itemset
```

Gambar 3.5 *Pseudocode* tahap kedua

3.5.3 Tahap Subsequent

Pada tahap ini dilakukan proses pencarian k-itemset utama dimana k lebih besar dari dua. Proses utamanya adalah pembangkitan itemset kandidat dengan menggunakan metode FORC dan perhitungan *support* dengan menggunakan metode FANGS. Pada proses-proses ini dilakukan beberapa optimasi untuk meningkatkan efisiensi kinerja algoritma.

3.5.3.1 Pembangkitan itemset kandidat dengan metode FORC

Fully ORganized Candidate-generation (FORC) merupakan metode pembangkitan itemset kandidat yang mengacu pada *equivalence class clustering*. Hal yang menarik pada metode ini adalah dengan menggunakan masukan yaitu data S_k (F_k atau C_k), maka dapat diperoleh itemset kandidat untuk C_{k+1} hingga C_{2k}. Adapun alur proses FORC adalah sebagai berikut:

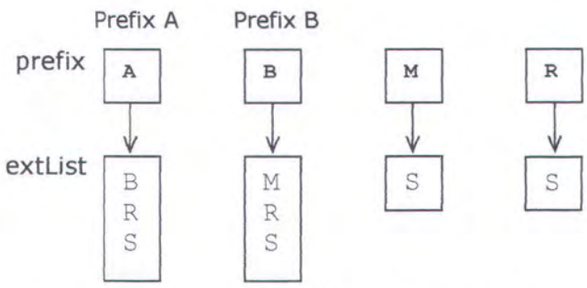
- Data S_k diklasifikasikan dalam kelas yang disebut *"equivalence class"*. Kriteria pengelompokan itemset ditentukan dari itemset yang memiliki *prefix* k-1 itemset yang sama. Sebagai contoh, untuk itemset ABCD maka *prefix* = ABC dan element terakhirnya adalah D.
- Untuk setiap kelas, *prefix* disimpan dalam *hashtable* dan elemen yang terakhir dari itemset disimpan dalam daftar item yang diurutkan secara *ascending* yang dinamakan *extList*.
- Melakukan proses penggabungan (UNION) antara *prefix* dengan seluruh pasangan yang dapat dibentuk dari extList yang ada di kelas tersebut.
- Memeriksa keberadaan subset setiap itemset kandidat yang terbentuk. Jika seluruh subset ada maka itemset tersebut termasuk dalam itemset kandidat . Penggunaan *prefix* sebagai kunci pada *hashtable* memudahkan pencarian subset tiap itemset dan pada pemeriksaan ini dilakukan optimasi sehingga subset untuk beberapa itemset dapat diperiksa secara simultan.

Hingga saat ini itemset utama yang diperoleh adalah F_1 dan F_2. Langkah selanjutnya adalah mendapatkan itemset kandidat dengan menggunakan masukan yaitu data S_k. Data F_2 menjadi masukan untuk mendapatkan itemset kandidat C_3 dan C_4 dan prosesnya adalah sebagai berikut:

- $C_3 = FORC(F_2)$

  F_2: AB AR AS BM BR BS MS RS

  AB, AR dan AS termasuk dalam satu kelas karena memiliki *prefix* yang sama yaitu A dan *extList* untuk kelas ini adalah B, R dan S. Sedangkan BM, BR dan BS termasuk dalam kelas dengan *prefix* B dan pada *extList* diisi dengan item M,R dan S. Berikut adalah gambar *equivalence class* yang terbentuk untuk penentuan C_3:



Gambar 3.6 Equivalence Class untuk C_3

Untuk masing-masing kelas dilakukan penggabungan itemset antara *prefix* dengan senua pasangan item yang dibentuk dari *extList*. Dari hasil ini dilakukan pemeriksaan keberadaan subset untuk masing-masing itemset kandidat yang terbentuk. Jika subset suatu itemset tidak ditemukan dalam kelas manapun maka itemset kandidat tersebut dihapus. Berikut ini adalah pemeriksaan subset untuk C_3:

C_3 = ABR : periksa subset BR, R ada di prefix B → extList

ABS : periksa subset BS, S ada di prefix B → extList

ARS : periksa subset RS, S ada di prefix R → extList

BMR : periksa subset MR, R tidak ada di prefix M → extList

BMS : periksa subset MS, S ada di prefix M → extList

BRS : periksa subset RS, S ada di prefix R → extList

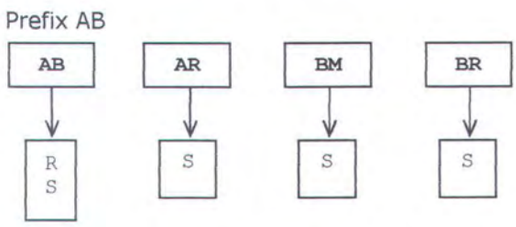
Karena itemset MR sebagai subset dari itemset kandidat BMR tidak ditemukan maka itemset BMR dihapus dari daftar itemset kandidat sehingga C_3 = ABM, ABS, ARS, BMS dan BRS.

- $C_4 = FORC(C_3)$

Berbeda dengan C_3, C_4 diperoleh dengan menggunakan masukan data itemset kandidat yaitu C_3. Namun cara mendapatkan hasil kandidat-nya sama dengan cara mendapatkan C_3.

C_3: ABR ABS ARS BMS BRS

ABR dan ABS termasuk dalam satu kelas karena memiliki *prefix* yang sama yaitu AB dan *extList* untuk kelas ini adalah R dan S. Berikut adalah gambar *equivalence class* yang terbentuk untuk penentuan C_4:



Gambar 3.7 Equivalence Class untuk C_4

Dengan menggabungkan *prefix* dengan pasangan item yang dapat dibentuk pada *extList* maka diperoleh:

C_3 = ABRS : periksa subset ARS, BRS

ARS : S ada di prefix AR → extList

BRS : S ada di prefix BR → extList

Sehingga C_4 = ABRS.

Pseudocode untuk proses ini dapat dilihat pada gambar 3.8. Kompleksitas bagian algoritma untuk baris 1-4, baris 5-14, baris 7-14, baris 10-11 dan baris 12-

13 secara terurut adalah O(n), O(m), O(m-1), O(k) dan O(m-2). Kompleksitas total *pseudocode* ini adalah O(n+m((m-1)(k+(m-2)))) dimana n merupakan jumlah itemset dalam S_k, m merupakan jumlah prefix dan k merupakan jumlah subset dari itemset kandidat.

```
1.  for each itemset i in Sk {
2.    insert(i.prefix) into hPrefix
3.    insert(i.lastElement) into i.prefix -> extList
4.  }
5.  for each prefix vect in hPrefix {
6.    tmpremList = vect -> extList
7.    for each element t in tmpRemList {
8.      newP = vect U t
9.      remList = {i| i ∈ tmpRemList dan i>t}
10.     for each (k-1) subset subP of newP
11.       remList = remList ∩ (subP -> extList)
12.     for each element q in remList
13.       newCand = vect U q
14.   }
15. }

/* keterangan:
Sk = Fk atau Ck
hPrefix = hashtable prefix -> extList
newCand = item kandidat baru yang diperoleh
```

Gambar 3.8 *Pseudocode* proses FORC

3.5.3.2 Optimasi pencarian simultan pada FORC

Optimasi pencarian simultan dilakukan pada saat pencarian subset dari suatu itemset kandidat yang telah diidentifikasi. Struktur untuk FORC seperti telah dijelaskan pada subbab sebelumnya menggunakan *equivalence class* yang memungkinkan untuk pencarian subset dari beberapa itemset kandidat secara simultan.

Sebagai contoh dari hasil proses awal FORC diperoleh C_3 = {ABR, ABS, ARS, BMS, BRS}. Kemudian diperiksa subset untuk itemset kandidat yang berasosiasi dengan item AB yaitu ABR dan ABS, sedangkan untuk itemset yang lain dilakukan proses yang sama. Dalam hal ini subset yang harus diperiksa

adalah A_i dan B_i dimana i adalah R atau S. Subset A_i tidak perlu diperiksa karena ABR dan ABS dihasilkan dari *prefix* A sehingga sudah jelas seluruh subset Ai ada.

Untuk memeriksa subset BR maka kita mengakses kelas dengan *prefix* B dan mencari apakah item S ada pada *extList – prefix* B. Pada posisi kelas saat ini dilakukan juga pencarian terhadap item S pada *extList* karena item tersebut berhubungan dengan subset untuk itemset kandidat ABS.

Dengan cara demikian penentuan keberadaan subset untuk beberapa itemset kandidat dapat dilakukan dengan sekali akses pada suatu kelas.

3.5.3.3 Proses perhitungan support dengan metode FANGS

Perhitungan *support* pada metode *Fast ANding Graph for Snakes* (FANGS) dilakukan pada data itemset kandidat yang telah disimpan dalam struktur DAG yang dihasilkan dari proses FORC. Keuntungan dari metode FANGS adalah perhitungan *support* secara simultan dalam sekali waktu untuk itemset i+1 hingga 2i.

3.5.3.3.1 Struktur DAG

Setiap itemset kandidat yang terbentuk dari proses FORC digabungkan dalam struktur DAG. Dalam struktur DAG, itemset *leaf* merupakan itemset utama pada level i (F_i) sedangkan pada ketinggian r terdapat itemset kandidat untuk itemset i+r dan ditunjuk oleh pasangan yang menjadi subsetnya pada level r-1. Setiap *snake* kandidat memiliki dua variabel yaitu "latest TID" (LTID) dan "current count" (CCNT). Nilai awal variabel LTID di-set -1 dan variable CCNT di-set 0. Variabel ini akan digunakan pada saat dilakukan perhitungan *support* dengan

menggunakan prosedur FANGS. Ketinggian struktur DAG maksimum untuk menyimpan data itemset kandidat adalah 2i/2 karena prosedur FANGS untuk perhitungan *support* mampu untuk mendapatkan itemset utama F_{i+1} hingga F_{2i}. Seperti terlihat pada gambar 3.10, itemset *leaf* pada struktur DAG merupakan 2-itemset utama (F_2) yang digunakan untuk menghitung *support* 3-itemset kandidat (C_3) dan 4-itemset kandidat (C_4). Item A, B, M, R dan S merupakan data *snake* yang dibaca dari memori hasil dari tahap *mining* sebelumnya. Data *tid-list* pada tiap itemset *leaf* merupakan hasil penggabungan pasangan *1-snake*. Itemset AB, AR, AS, BM, BR, BS, MS dan RS merupakan itemset *leaf*. Pada level diatasnya terdapat 3-itemset kandidat yang masing-masing ditunjuk oleh pasangan itemset yang menjadi subsetnya. Sebagai contoh itemset ABR ditunjuk oleh itemset AB dan AR. Itemset ABRS pada level 2 juga ditunjuk oleh pasangan subsetnya pada level 1 yaitu ABR dan ABS. Gambar berikut menunjukkan pseudocode untuk proses inisialisasi struktur DAG.

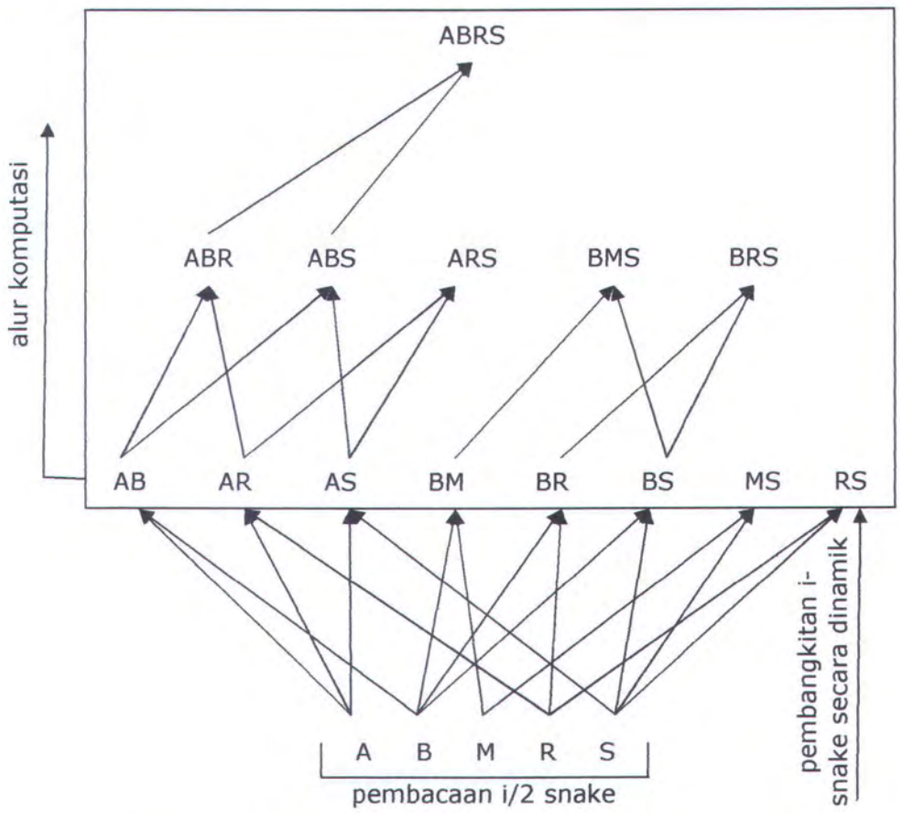
```
1.  hLeaf = ambil_data_frequent_terakhir_pada(F)
2.  for each element xStr in hLeaf {
3.    bs1 = ambil_data_bitset(getHalf(0,xStr))_pada_ hGenCover
4.    bs2 = ambil_data_bitset(getHalf(1,xStr))_pada_hGenCover
5.    bRes = bs1 and bs2
6.    hSnake = hSnake U data(xStr, new Leaf_VIPER(bRes))
7.  }
8.  tambah_element_hSnake_pada(candDAG)

/* keterangan:
hSnake = hashtable untuk himpunan snake
hLeaf  = hashtable berisi data terakhir frequent itemset
F      = obyek berisi seluruh data frequent itemset
bs1, bs2 = bitset diperoleh dari generator cover
bRes     = TID list dihasilkan dari ANDing bitset
candDAG  = obyek menyimpan itemset dalam struktur DAG
```

Gambar 3.9 *Pseudocode* proses inisialisasi DAG

Kompleksitas waktu untuk proses ini adalah O(n). Perulangan terjadi pada baris 2 hingga 7 dimana n adalah jumlah elemen dalam himpunan *frequent* terakhir.

Gambar berikut ini merupakan visualisasai struktur DAG hasil proses FORC dengan menggunakan data transaksi pada tabel 3.3.



Gambar 3.10 Struktur DAG untuk penghitungan *support*

3.5.3.3.2 Proses perhitungan *support*

Dengan menggunakan struktur DAG maka proses perhitungan *support* menjadi lebih sederhana karena dilakukan dengan interseksi TID antara pasangan itemset yang menjadi subset dari suatu itemset kandidat. Adapun proses penghitungan *support* dengan metode FANG berlangsung sebagai berikut:

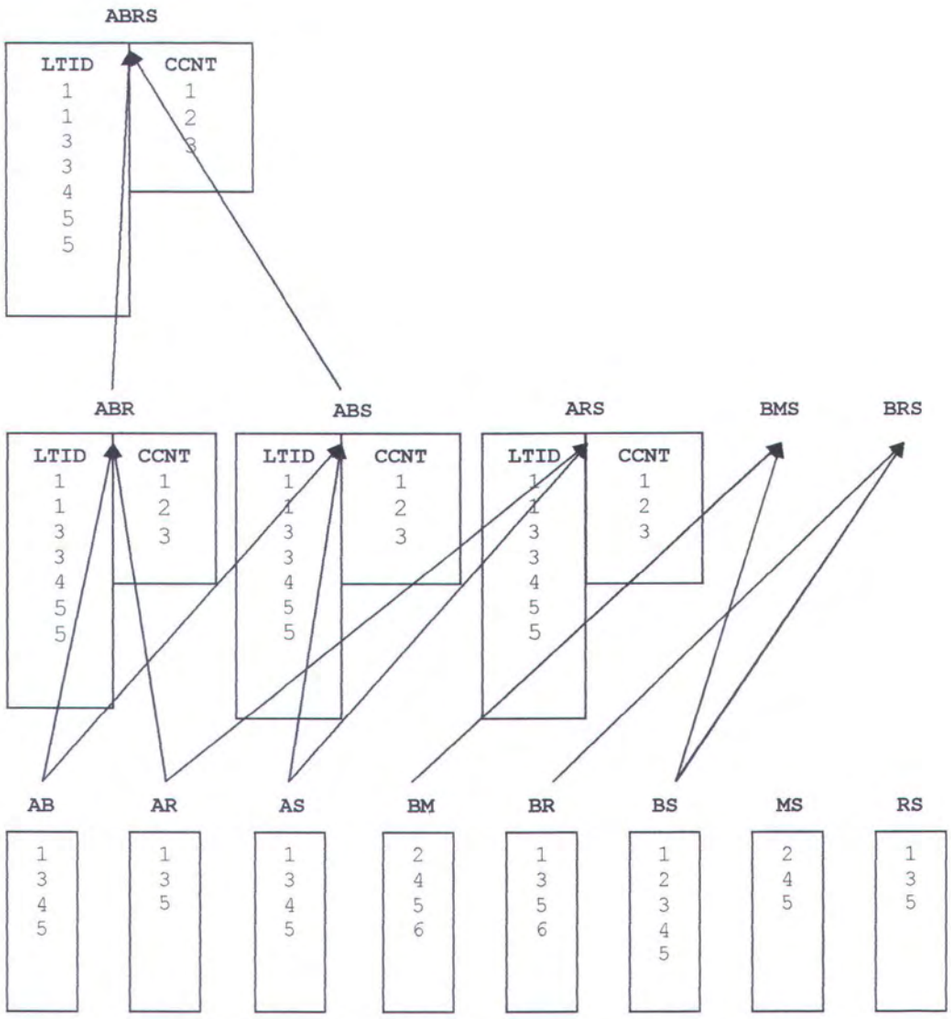
- Dengan menggunakan data *snake* yang dibaca dari memori dibentuk *tid vector* untuk setiap itemset *leaf* yang kemudian dikonversi ke dalam bentuk

tid list. Setiap itemset *leaf* akan memiliki data *tid list* yang diurutkan secara *ascending*.

- Dengan menggunakan *tid list* tersebut proses dilakukan satu TID pada satu saat sehingga setiap TID diproses secara sekuensial.
- Perhitungan dimulai dari *tid list* pada setiap itemset *leaf*. Ketika TID dibaca dari itemset *leaf*, nilai variabel LTID dibandingkan dengan TID ini. Jika nilai TID lebih besar dari nilai variabel LTID maka nilai variabel LTID diubah menjadi nilai TID. Sedangkan jika nilainya sama maka nilai variabel CCNT ditambah 1. Perubahan nilai variabel CCNT akan menyebabkan perubahan nilai variabel LTID maupun CCNT secara rekursif pada itemset di level yang lebih tinggi yang berhubungan dengan itemset yang mengalami perubahan tersebut.

Contoh perhitungan *support* dapat dilihat pada gambar 3.11. *TID list* untuk itemset AB, AR, AS, BM, BR, BS, MS dan RS dibentuk dari data *1-snake* yang dibaca dari memori. Sesuai urutan TID pada data transaksi dilakukan pembacaan *tid list* tiap itemset *leaf*. Prosedur perhitungan *support* itemset dilakukan jika terdapat TID saat ini pada *tid list* suatu itemset. Pada gambar tersebut dapat dilihat bahwa ketika TID 1 dibaca dari *tid list* itemset AB, nilai variabel LTID setiap itemset kandidat yang ditunjuk oleh itemset AB yaitu itemset ABR dan ABS diubah menjadi 1. Selanjutnya TID 1 dibaca dari *tid list* AR dan mengubah nilai variabel LTID untuk itemset ABR dan ARS. Karena nilai variabel LTID untuk ABR bernilai sama, yaitu 1 maka nilai variabel CCNT itemset ABR ditambah 1. Proses ini dilakukan secara rekursif dan diteruskan hingga

perhitungan support untuk C_4. Berikut adalah struktur perhitungan *support* dengan metode FANGS:



Gambar 3.11 Proses perhitungan support

Proses FANGS untuk perhitungan *support* berlangsung secara rekursif pada struktur DAG. *Pseudocode* untuk proses ini dapat dilihat pada gambar berikut:

```
1.  for (i=0;i<jumlah_TID;i++) {
2.    for each keys x in hSnake {
3.      lVip = ambil_data_leaf_dengan_key(x)_pada_hSnake
4.      lVip.selfCheck(TID)
5.    }
6.  }

/* keterangan:
vTID  = vektor daftar seluruh TID dalam transaksi
lVIP  = obyek bertipe Leaf_VIPER
```

Gambar 3.12 *Pseudocode* proses FANGS

Kompleksitas waktu untuk *pseudocode* ini adalah O(nm) dimana n adalah jumlah TID dan m adalah jumlah elemen dalam data *leaf*.

3.5.3.3.3 Optimasi penulisan *snake* dengan *Lazy Snake Writes*

Lazy snake writes merupakan optimasi yang digunakan untuk mereduksi *snake* yang akan disimpan dalam memori sehingga hanya *snake* yang berguna untuk proses komputasi berikutnya yang disimpan.

Pada saat proses perhitungan *support* untuk itemset kandidat pada struktur DAG dengan menggunakan *i-snake* kita tidak dapat mengetahui itemset kandidat mana pada level 2i yang akan memenuhi *minsup* sehingga tidak perlu menyimpan seluruh *snake* pada level 2i ke memori. Setelah melalui proses optimasi pemilihan *snake*, *i-snake* disimpan dalam memori. *i-snake* inilah yang pada subsequent selanjutnya digunakan sebagai *generator cover* untuk itemset *leaf*. *Generator cover* merupakan pasangan *snake* yang digunakan untuk membangkitkan data *tid list* itemset hasil penggabungan pasangan *snake* ini. Itemset hasil penggabungan pasangan *snake* menjadi salah satu itemset *leaf* pada struktur DAG.

Jika kembali melihat gambar 3.10 maka *i-snake* untuk itemset *leaf* dibangkitkan dari interseksi pasangan i/2 *snake* yang dibaca dari memori yang

telah disimpan pada tahap *mining* sebelumnya. Data *tid list* itemset AB dibangkitkan dari pasangan *snake* itemset A dan B.

Ada beberapa metode yang diterapkan dalam penerapan optimasi untuk pemilihan *generator cover* antara lain:

- Penentuan *snake* yang dijadikan *generator cover* dilakukan sebelum proses perhitungan support dengan cara melakukan interseksi antara pasangan *snake* secara rekursif hingga menghasilkan suatu itemset kandidat pada level 2i.
- Ada suatu kemungkinan bahwa beberapa pasangan *generator cover* akan menghasilkan satu itemset kandidat pada level 2i. Sebagai contoh pada gambar 3.10 terlihat bahwa pasangan itemset (AB, AR) dan (AB, AS) merupakan *generator cover* untuk itemset kandidat ABRS. Pada kondisi demikian diusahakan untuk menggunakan *generator cover* yang telah teridentifikasi untuk itemset sebelumnya. Pada contoh diatas itemset AB dipilih lebih dari sekali sehingga lebih optimal dan dapat mengurangi jumlah *snake* yang harus disimpan.

Optimasi pemilihan *generator cover* untuk setiap itemset kandidat pada level 2i berdasarkan urutan secara *descending* pada nilai *support*-nya. Dalam proses ini diutamakan penggunaan *generator cover* yang memiliki nilai support lebih tinggi. Dasar pemikiran yang digunakan adalah bahwa itemset *leaf* dengan nilai *support* yang lebih tinggi akan berpeluang menjadi *generator cover* dari beberapa itemset kandidat. Gambar 3.13 menunjukkan psedocode untuk metode tersebut. Kompleksitas waktu untuk metode ini adalah O(n) karena hanya terdapat satu perulangan yaitu pada baris 1 hingga 5 dimana n = 2.

```
1.  for (int i=0;i<2;i++) {
2.    xStr = ambil_half_itemset(i)
3.    if (tidak_ada_element_dengan_key(xStr)_pada_hGenCover)
4.      hGenCover = hGenCover U ambil_data_bitset_dengan_key(xStr)
5.  }

/* keterangan:
xStr    = setengah itemset awal/akhir
hGenCover = hashtable untuk himpunan generator cover
bs      = bitset untuk meyimpan data VTV
xvTID   = vector untuk tiap item yang berisi TID
```

Gambar 3.13 *Pseudocode* metode Lazy Snake Write

3.5.4 Pencarian pola asosiasi

Pola asosiasi merupakan aturan yang menunjukkan hubungan antara dua itemset, misalnya itemset1 dan itemset2 yang menyatakan bahwa jika itemset1 ada maka itemset2 juga ada dalam suatu transaksi.

Secara matematis kaidah asosiasi A→B merupakan perbandingan antara jumlah *support* itemset AUB dengan jumlah support item A dan dapat diformulakan sebagai:

$$A \rightarrow B = \frac{\sigma(AUB)}{\sigma A}$$

Dengan demikian, untuk dapat mencari kaidah asosiasi dari suatu data transaksi maka data yang diperlukan adalah daftar seluruh itemset utama dan nilai *confidence support*.

Data itemset utama yang diperoleh dari proses *mining* sebelumnya dapat dikategorikan berdasarkan prosentase *support*nya sebagai berikut:

Tabel 3.4 Itemset utama berdasarkan prosentase *support*

Prosentase Support	Daftar Itemset
100% (^)	B
83% (5)	S, BS
67% (4)	A, M, R, AB, AS, BM, BR, ABS
50% (3)	AR, MS, RS, ABR, ARS, BMS, BRS, ABRS

Sebagai contoh untuk mencari kaidah asosiasi A → B maka:

$$A \rightarrow B = \frac{\sigma(AUB)}{\sigma A}$$

$$= \frac{4}{4} \times 100\ \% = 100\ \%$$

A → B merupakan salah satu kaidah asosiasi karena memenuhi *confidence support* yang telah ditentukan yaitu 100%. Dengan menggunakan proses yang sama pada data *support* diatas maka hasil seluruh kaidah asosiasi yang diperoleh adalah sebagai berikut:

A → B	(4/4)	AB → A	(4/4)	RS → B	(3/3)
A → S	(4/4)	AR → B	(3/3)	AR → BS	(3/3)
A → BS	(4/4)	AR → S	(3/3)	RS → AB	(3/3)
M → B	(4/4)	AS → B	(4/4)	ABR → S	(3/3)
R → B	(4/4)	MS → B	(3/3)	ARS → B	(3/3)
S → B	(4/4)	RS → A	(3/3)	BRS → A	(3/3)

Ada dua parameter lagi yang sifatnya pilihan (optional) untuk mendapatkan pola asosiasi yang diharapkan, yaitu *antecedent* dan *consequent. Antecedent* merupakan itemset pada pola asosiasi yang berada pada sisi sebelah kiri dan *Consequent* merupakan itemset pada pola asosiasi yang berada pada sisi sebelah kanan. Jika tidak ditentukan, nilai kedua variabel ini diset sebagai asteris (*).

Pseudocode untuk proses pencarian pola asosasi dapat dilihat pada gambar 3.14. Kompleksitas waktu untuk pseudocode ini adalah O(nm) dimana n adalah jumlah semua itemset utama yang diperoleh dan m adalah jumlah subset tiap item.

```
1. item = (semua k-itemset yang muncul)
2. for ( Ai ∈ item) {
3.   UpSupp = nilai support Ai
4.   Subset = {Semua subset Ai}
5.   for (Si ∈ Subset){
6.     DownSupp = nilai support Si
7.     Conf = (upSupp/downSupp)*100
8.     if (Conf>=minconf)&(equal(antecedent dan consequent))
9.       Rule = {Ai/Si → Si}
10.  }
11. }

/* keterangan:
UpSupp   = support pembilang
DownSupp = support penyebut
```

Gambar 3.14 *Pseudocode* pembentukan pola asosiasi

BAB IV

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

BAB IV

PERANCANGAN DAN PEMBUATAN

PERANGKAT LUNAK

Dalam bab ini diuraikan mengenai perancangan data, perancangan proses dengan pendekatan fungsional yang divisualisasikan dalam Diagram Aliran Data (DAD) dan perancangan antar muka perangkat lunak yang akan dibuat berikut implementasinya dalam basis data dan bahasa pemrograman yang digunakan.

4.1 Perancangan Perangkat Lunak

Dalam sub bab ini akan dibahas mengenai perancangan data, baik data masukan, data proses dan data keluaran, perancangan proses serta perancangan antar muka dan fungsinya.

4.1.1 Perancangan Data

4.2.1.1 Data Masukan

Data masukan yang digunakan sebagai obyek *mining* dalam perangkat lunak ini adalah data transaksi sintetis yang dibangkitkan dari *data generator* [ADH-02] dan beberapa parameter yang diperlukan dalam proses *mining*. Tabel transaksi sintetis yang akan digunakan memiliki *field* utama yaitu ID transaksi dengan tipe data *number* dan ID item dengan tipe data *varchar(10)*. Sedangkan parameter yang diperlukan adalah *minimum support* yang digunakan untuk mendapatkan k-itemset utama dan *confidence support* yang diperlukan untuk mendapatkan pola asosiasi. *Minimum support* dan *confidence support*

merupakan prosentase kemunculan yang dikehendaki dari jumlah transaksi yang ada.

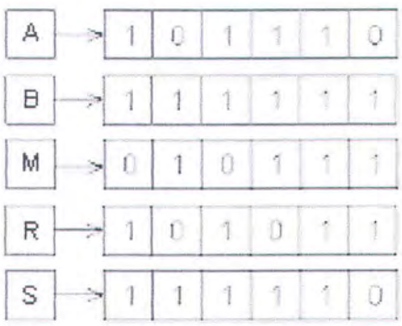
Untuk memperoleh pola asosiasi terdapat pilihan untuk menentukan permintaan pengguna dengan menset *antecedent* dan *consequent*. Jika permintaan pengguna ini tidak diisi maka secara default *antecedent* dan *consequent* diset sebagai asteris (*).

4.2.1.2 Data Proses

Pada saat proses mining terdapat beberapa jenis data yang dirancang dan disesuaikan untuk mengimplementasikan algoritma VIPER. Jenis data tersebut antara lain sebagai berikut:

- Data *snake*

  Algoritma VIPER melakukan sekali pembacaan pada basis data dan memindah data tersebut ke dalam memori dalam format VTV. Data yang disimpan dalam memori ini adalah data yang memenuhi nilai *minimum support* dan potensial digunakan pada proses mining selanjutnya. Setiap item memiliki data berupa himpunan bit yang bernilai '0' dan '1'. Panjang himpunan bit sama dengan jumlah transaksi yang ada dan secara default seluruh bit pada himpunan bit tersebut diset sebagai '0'. Jika item muncul pada suatu ID transaksi maka himpunan bitnya diset '1' pada index transaksi tersebut. Berikut adalah gambar data himpunan bit untuk setiap item:



Gambar 4.1 Data *snake* untuk tiap item

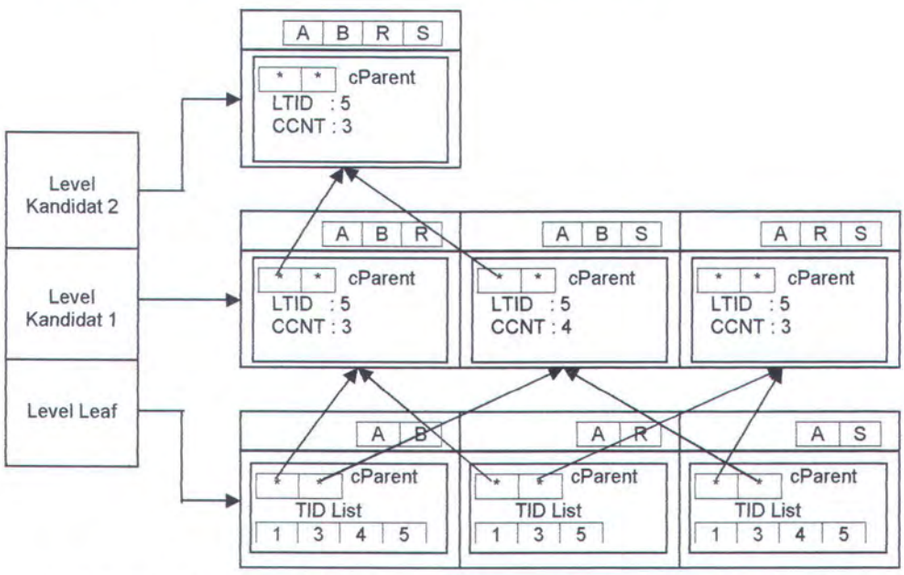
- Data item l*eaf*

  Data item leaf dan data item kandidat yang akan dijelaskan berikutnya dirancang untuk membangun struktur DAG dan digunakan saat perhitungan *support*. Data ini digunakan untuk mendapatkan item kandidat dan pengisian *support* untuk kandidat-kandidat item tersebut secara rekursif sehingga data ini berada pada level paling dasar pada struktur DAG. Ada dua obyek didalamnya yaitu cParent dan TID List. cParent bertipe vektor yang berisi dan berelasi dengan obyek item kandidat yang dihasilkan dari obyek item leaf tersebut. Sedangkan obyek TID List digunakan untuk menyimpan data daftar TID dimana itemset tersebut ada. Gambar data item leaf dapat dilihat pada gambar 4.2.

- Data item kandidat

  Data item kandidat digunakan untuk mendapatkan item kandidat lainnya pada level yang lebih tinggi dan perhitungan support untuk item kandidat tersebut dan item kandidat lainnya yang berelasi dengan item kandidat ini. Data ini memiliki tiga obyek yaitu cParent, LTID dan CCNT. Obyek cParent bertipe vektor yang berisi dan berelasi dengan obyek item kandidat yang dihasilkan dari obyek item kandidat tersebut. Obyek yang kedua yaitu LTID

bertipe number dan berisi ID transaksi terakhir yang diperiksa pada item kandidat tersebut. Sedangkan obyek CCNT bertipe number dan berisi nilai *support* untuk tiap item kandidat. Gambar berikut menunjukkan data item leaf dan data item kandidat dalam struktur DAG. Data item leaf berada pada level leaf dan data item kandidat berada pada level kandidat.



Gambar 4.2 Data item leaf dan data item kandidat dalam struktur DAG

4.2.1.3 Data Keluaran

Data keluaran yang dihasilkan dari perangkat lunak selama proses *mining* akan disimpan dalam bentuk file teks. Ada empat file yang akan dibuat dalam satu proses mining antara lain:

1. File proses, menyimpan informasi proses yang berlangsung saat dilakukan proses untuk mendapatkan k-itemset utama.
2. File *frequent*, menyimpan informasi berupa daftar k-itemset utama yang diperoleh dari proses mining.

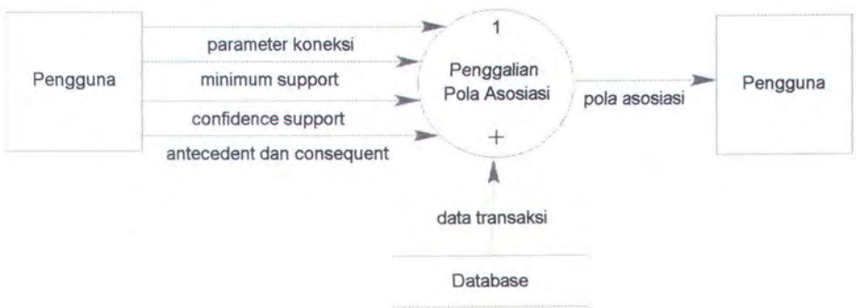
3. File *rules*, menyimpan informasi berupa daftar pola asosiasi yang diperoleh dari proses mining.
4. File *summary*, menyimpan laporan hasil proses *mining* dari data masukan hingga data keluaran. Informasi yang disimpan tersebut meliputi:
   - Nama data tabel transaksional
   - Jumlah record, jumlah transaksi dan jumlah item yang ada di data tabel transaksi
   - Nilai *minimum support* dan *confidence support*
   - Waktu koneksi basis data, waktu inisialisasi dan waktu proses pencarian k-itemset utama.
   - Jumlah data itemset utama yang diperoleh
   - Nilai *antecedent* dan *consequent*
   - Jumlah data pola asosiasi yang berhasil dibentuk
   - Waktu pembentukan pola asosiasi

4.1.2 Perancangan Proses

Perancangan proses perangkat lunak ini menggunakan pendekatan fungsional yang divisualisasikan dalam Diagram Aliran Data (DAD). Pembuatan DAD tersebut menggunakan perangkat lunak Power Designer versi 6.0. Diagram aliran data ini akan menjelaskan proses data mining yang dilakukan dari level 0 hingga level 2.

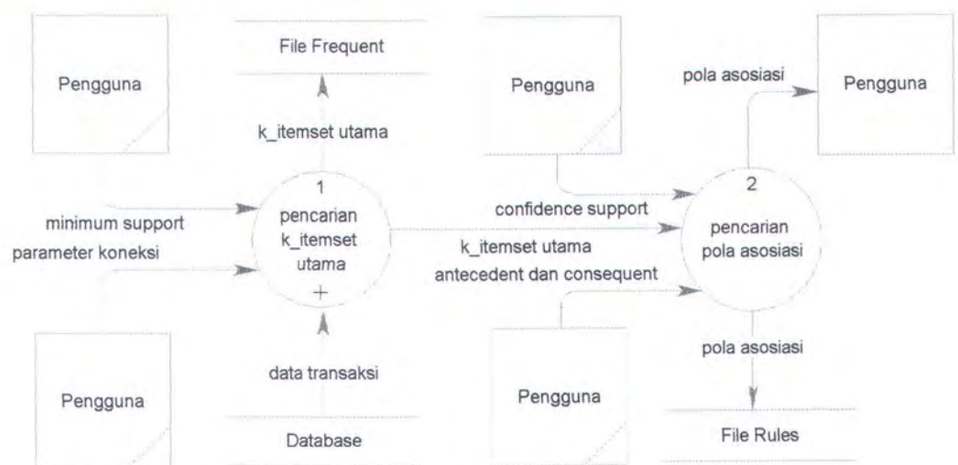
Proses dimulai dari pengguna sebagai entitas eksternal yang melakukan permintaan untuk mendapatkan hasil berupa pola asosiasi. Diagram untuk pencarian pola asosiasi pada level 0 dapat dilihat pada gambar 4.3. Pengguna memberikan data masukan yang diperlukan untuk menjalankan proses *mining* yaitu parameter koneksi ke basis data termasuk tabel transaksi yang digunakan

sebagai obyek mining, *minimum support* dan *confidence support* serta nilai *antecedent* dan *consequent* yang sifatnya pilihan(optional) untuk mendapatkan pola asosiasi. Dalam proses ini data transaksi sesuai dengan konfigurasi dari pengguna dibaca dari basis data dan keluaran yang akan diperoleh adalah pola asosiasi yang diharapkan.



Gambar 4.3 DAD level 0

Proses penggalian pola asosiasi dapat didekomposisi dalam DAD level 1 dan dapat dilihat pada gambar berikut:

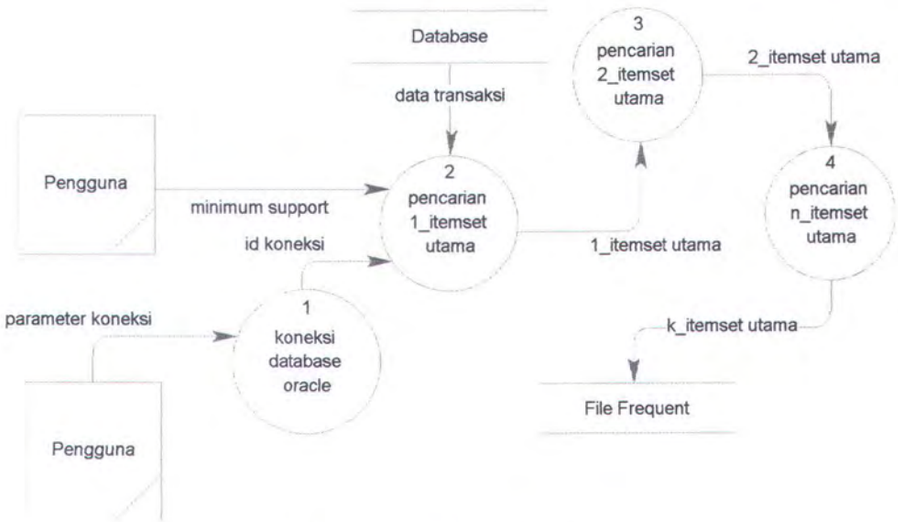


Gambar 4.4 DAD level 1

Proses penggalian pola asosiasi terdiri dari dua sub proses yaitu proses pencarian k-itemset utama dan proses pencarian pola asosiasi. Proses pencarian k-itemset utama akan membaca data transaksi dari basis data dan mendapatkan

k-itemset yang memenuhi syarat *minimum support* yang ditentukan oleh pengguna. Hasil dari proses ini berupa daftar seluruh k-itemset utama yang menjadi masukan untuk proses pencarian pola asosiasi dan juga akan disimpan dalam file frequent. Dari hasil k-itemset utama tersebut dilakukan proses pencarian seluruh pola asosiasi dengan mengikuti batasan *antecedent* dan *consequent* serta memenuhi *minimum confidence* yang telah ditentukan. Hasil seluruh pola asosiasi yang diperoleh akan ditampilkan kepada pengguna dan juga akan disimpan dalam file *rules*. Proses pencarian k-itemset utama dapat didekomposisi dalam DAD level 2 dan dapat dilihat pada gambar 4.5.

Ada empat sub proses yang ada dalam proses pencarian k-itemset utama yaitu proses koneksi database, proses pencarian 1-itemset utama, proses pencarian 2-itemset utama dan proses pencarian n-itemset utama dimana n>2. Proses koneksi database merupakan proses awal untuk mengakses sumber data dalam database. Selanjutnya berdasarkan konfigurasi dan parameter *mining*, dilakukan proses pencarian 1-itemset utama. Hasil dari proses ini disimpan dan digunakan untuk mendapatkan 2-itemset utama. Tahap terakhir yang dilakukan adalah iterasi proses pencarian n-itemset utama dimana n>2 hingga itemset kandidat tidak dapat diperoleh lagi. Hasil seluruh itemset utama yang diperoleh akan digunakan untuk proses pencarian pola asosiasi dan juga disimpan dalam file frequent.

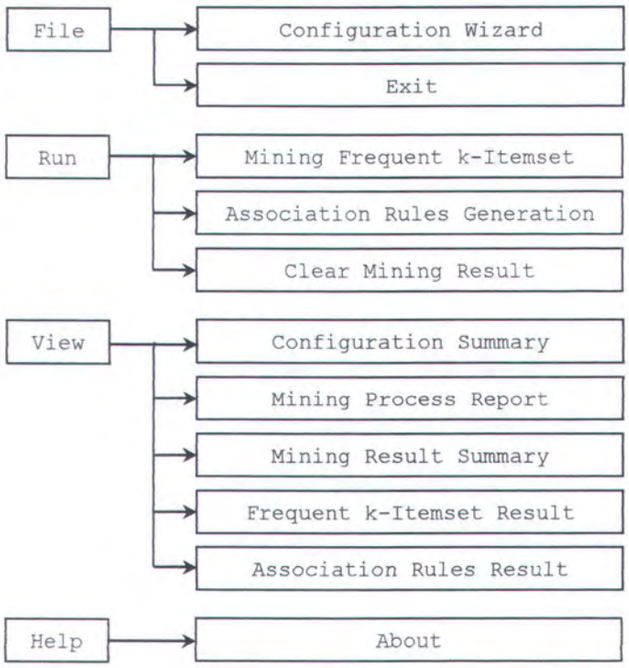


Gambar 4.5 DAD level 2

4.1.3 Perancangan Antar Muka

Perangkat lunak dibangun dalam bentuk *Single Document Interface* (SDI). Untuk melakukan konfigurasi, penggalian pola asosiasi maupun mendapatkan laporan dari proses *mining* pengguna dapat mengakses menu maupun toolbar yang disediakan. Gambar 4.6 menunjukkan rancangan struktur menu yang ada pada perangkat lunak yang dibangun. Berikut ini adalah penjelasan fungsi dari tiap menu yang disediakan.

- Menu File
  - Configuration Wizard, digunakan untuk mengatur koneksi ke basis data di Oracle, pemilihan tabel data transaksi sebagai obyek *mining*, penentuan parameter-parameter *minimum support* dan *confidence support* yang diperlukan untuk mendapatkan seluruh k-itemset utama dan pola asosiasi serta penentuan metode proses yang digunakan.
  - Exit, digunakan untuk keluar dari aplikasi.



Gambar 4.6 Struktur menu aplikasi

- Menu Run
  - Mining Frequent k-Itemset, digunakan untuk menjalankan proses pencarian seluruh data k-itemset utama.
  - Association Rules Generation, digunakan untuk proses pencarian pola asosiasi dengan menggunakan data k-itemset utama yang telah diperoleh. Pada awal pemilihan menu ini akan muncul satu window untuk menentukan nilai *antecedent* dan *consequent* pola asosiasi. Pengisian *antecedent* dan *consequent* ini bersifat pilihan (optional) dan secara default diset dengan asteris (*).
  - Clear Mining Result, digunakan untuk menghapus seluruh data hasil proses mining yang tersimpan dalam memori.

- Menu View
  - Configuration Summary, digunakan untuk mengetahui nilai-nilai maupun obyek hasil konfigurasi dari pengguna.
  - Mining Process Report, digunakan untuk menampilkan laporan proses-proses yang dilakukan selama proses mining berlangsung.
  - Mining Result Summary, digunakan untuk menampilkan laporan ringkas hasil proses mining. Isi laporan ini sesuai dengan rancangan data keluaran untuk file *summary*.
  - Frequent k-Itemset Result, digunakan untuk menampilkan laporan seluruh data *frequent* itemset yang dihasilkan.
  - Association Rules Result, digunakan untuk menampilkan laporan seluruh data pola asosiasi yang dihasilkan.
- Menu Help
  - About, digunakan untuk menampilkan identitas pembuat aplikasi.

4.2 Implementasi Perangkat Lunak

Pada sub bagian ini akan dijelaskan implementasi dari perancangan data, proses, desain antar muka dan alur penggunaannya yang dibangun dengan menggunakan bahasa pemrograman Java.

4.2.1 Implementasi Data

4.2.1.1 Data Masukan

Data transaksi yang digunakan sebagai obyek *mining* disimpan dalam database di RDBMS Oracle. Nama *field* yang digunakan pada data transaksi adalah TID, menunjukkan ID transaksi dan ITEM, menunjukkan nama item. Struktur tabelnya dapat dilihat pada tabel dibawah ini.

Tabel 4.1 Struktur tabel data transaksi

Nama Field	Tipe Data
TID	Number
ITEM	Varchar(10)

Pembuatan satu tabel transaksi di Oracle dengan spesifikasi seperti pada tabel 4.1 dapat dilakukan dengan menggunakan perintah DDL sebagai berikut:

```
Create table nama_table {
   TID number,
   ITEM varchar(10)
}
```

Kemudian pengisian data transaksi dapat dibangkitkan dengan menggunakan *data generator* [ADH-02].

4.2.1.2 Data Proses

Dengan menggunakan bahasa pemrograman Java, data *snake* untuk tiap itemset direpresentasikan menggunakan kelas BitSet karena nilai yang akan disimpan hanya bernilai '0' atau '1'. Keuntungan dari kelas ini adalah memerlukan alokasi memori yang hemat karena data disimpan dalam bit sehingga untuk menyimpan data *snake* suatu itemset dalam n buah transaksi diperlukan kelas BitSet dengan panjang n bit pula.

Implementasi data item kandidat direpresentasikan dengan menggunakan kelas hashtable dimana obyek item digunakan sebagai kunci, kelas vektor digunakan untuk obyek yang menyimpan daftar item kandidat yang dibangkitkan dari itemset tersebut (cParent) dan tipe data integer untuk obyek LTID dan CCNT. Pengambilan data yang disimpan dalam hashtable berdasarkan kunci yang ditentukan. Kelas CItemset_VIPER dibuat untuk menyimpan satu item kandidat dan didesain untuk memenuhi kebutuhan saat proses penghitungan support. Atribut cParent pada kelas ini berisi obyek yang menunjukkan relasi

dengan obyek lain yang juga bertipe Citemset_VIPER sehingga membentuk suatu graph. Berikut adalah abstraksi kelas CItemset_VIPER:

```
Class CItemset_VIPER {
  int LTID, CCNT;
  Vector cParent;

  CItemset_VIPER()
  // konstruktor
  void addParent(CItemset_VIPER cv)
  // setting untuk obyek yang dituju (directed graph)
  void setLTID(int t)
  // menset nilai LTID
  int getCCNT()
  // mengambil nilai LTID
}
```

Gambar 4.7 Abstraksi kelas CItemset_VIPER

Implementasi data item *leaf* juga direpresentasikan dengan menggunakan kelas hastable dimana obyek item digunakan sebagai kunci dan kelas vektor untuk menyimpan obyek cParent dan bitset untuk obyek *tid-list*. Kelas Leaf_VIPER dibuat untuk menyimpan satu data item leaf dan berada pada level dasar di struktur DAG. Sama halnya dengan kelas Citemset_VIPER, atribut cParent pada kelas ini berisi obyek yang menunjukkan relasi dengan obyek lain yang bertipe Citemset_VIPER. Sedangkan atribut LeafTID merupakan data ID transaksi dimana itemset tersebut ada. Setiap data ID transaksi ini akan diperiksa untuk mengisi support itemset kandidat. Berikut adalah abstraksi kelas Leaf_VIPER:

dikirimkan serta operasi pemindahan kursor. Struktur kelas ClsDB_VIPER dapat dilihat pada gambar berikut:

```
Class ClsDB_VIPER {
  Connection conn
  ResultSet rs
  Statement stmt

  DbCls_VIPER()
  // konstruktor
  boolean isConnected()
  // mendapatkan status koneksi ke database
  boolean connect (String h, int p, String s, String u, String w)
  // melakukan koneksi ke datbase
  boolean execSQL(String q)
  // menjalankan perintah SQL
  boolean isRSFirst()
  // mengetahui apakah kursor berada pada posisi pertama (awal)
  int numRows()
  // mendapatkan jumlah record yang dihasilkan
  Object fetchRow(String fname)
  // mendapatkan nilai field pada satu record
  boolean moveNext()
  // memindah posisi kursor ke record selanjutnya
  void rsClose()
  // menutup resultset
}
```

Gambar 4.9 Abstraksi kelas ClsDB_VIPER

- **ClsInfo_VIPER**

  Kelas ini dibuat untuk menyimpan dan mendapatkan segala informasi yang berkaitan dengan hasil konfigurasi dan parameter *mining* yang ditentukan oleh pengguna serta informasi data transaksi yang dipilih, dalam hal ini adalah jumlah record, jumlah transaksi dan jumlah item. Berikut adalah struktur kelas ClsInfo_VIPER:

```
Class ClsInfo_VIPER {
  String hostname, username, password, sidName, defTable,
         tidName, itemName
  int ms, cs, nms, port, nTID, nItem, nRow
  Vector vAnt, vCon

  ClsInfo_VIPER()
  // konstruktor
  void setIdentity(String h, int p, String s, String u, String w)
  // mengisi informasi koneksi
  void setObject(String d, String t, String i)
  // mengisi informasi table yang menjadi obyek mining
  void setParam(int m, int c)
  // mengisi informasi minimum support dan confidence support
  void fillInfo()
  // mengisi informasi jumlah record, transaksi dan item

  void setAntecedent(Vector v)
  void setConsequent(Vector v)
  Vector getAntecedent()
  Vector getConsequent()
  String getHost()
  int getPort()
  String getUser()
  String getSID()
  String getDefTable()
  int getMinSup()
  int getNMinSup()
  int getConfSup()
  int getNumTID()
  int getNumItem()
  int getNumRow()

  // menset dan mendapatkan properti class
}
```

Gambar 4.10 Abstraksi kelas ClsDB_VIPER

- **Association_VIPER**

  Kelas ini dibuat dan berkaitan dengan proses pencarian pola asosiasi. Operasi-operasi yang ada memungkinkan perangkat lunak untuk mendapatkan pola asosiasi sesuai permintaan pengguna dengan mengisikan nilai antecedent dan *consequent*. Berikut adalah struktur kelas Association_VIPER:

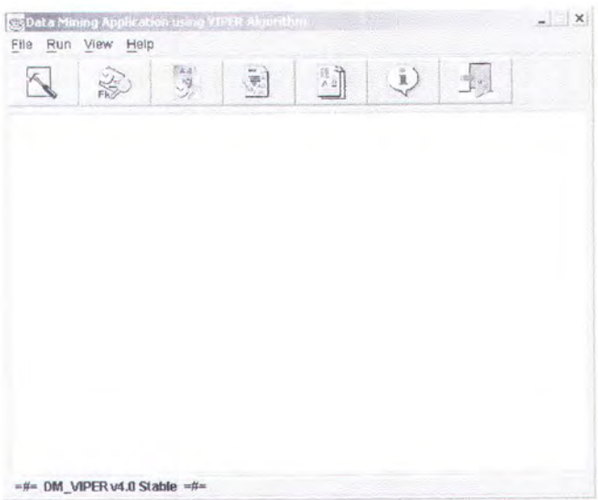
```
Class Association_VIPER {
  static final String fileAR = "association.txt";

  Association_VIPER()
  // konstruktor
  void checkInsert(int ch, String srcS, String inpS)
  // pengecekan kevalidan itemset
  void fillPossibleRule(int ch, String sX)
  // pengisian asteris dengan itemset yang mungkin
  int findSupport(String s)
  // mencari support suatu itemset
  String removeElementFrom(String s, int pos)
  // menghapus sebagian item dari itemset
  String printRemaining(String Source,String Substractor)
  // mengisi item sisa dari itemset yang disebutkan
  String unionSort(String s1,String s2)
  // pengurutan item pada itemset
  void getAR(String s1, String s2)
  // pencarian pola asosiasi dimana ant. dan con. non *
  void generateAR_woAC()
  // pencarian pola asosiasi dimana ant. dan con. = *
  void generateAR_wAC()
  // pencarian pola asosiasi dimana ant./con. berisi *
```

Gambar 4.11 Abstraksi kelas Association_VIPER

4.2.3 Implementasi Antar Muka

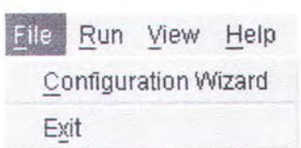
Berdasarkan perancangan antar muka pada bab sebelumnya, dilakukan implementasi sesuai dengan menu-menu yang ada. Gambar 4.13 adalah form utama aplikasi data mining dengan menggunakan algoritma VIPER. Pada bagian atas terdapat menubar untuk mengakses fungsi-fungsi yang disediakan perangkat lunak. Di bawah menubar terdapat dan toolbar yang menjadi alternatif pengaksesan fungsi-fungsi tersebut secara cepat. Bagian tengah merupakan area teks untuk menampilkan seluruh hasil yang diperoleh dari proses mining baik konfigurasi maupun laporan hasil.



Gambar 4.12 Form utama aplikasi VIPER

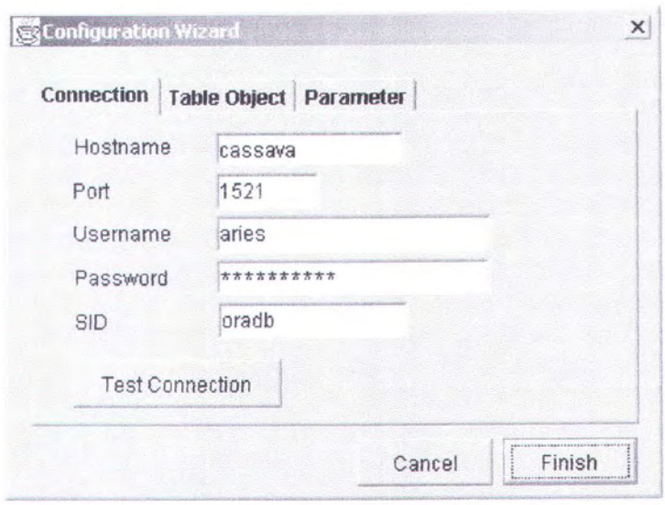
a. Menu File

Menu *File* terdiri dari dua sub menu yaitu *Configuration Wizard* dan *Exit* yang dapat dilihat pada gambar berikut:

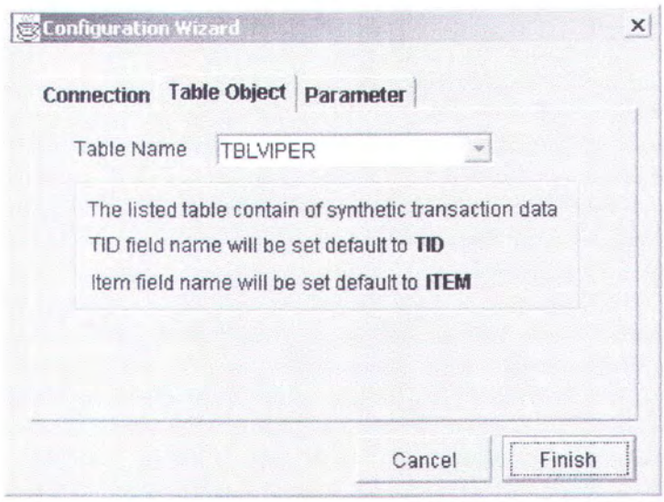


Gambar 4.13 Menu File

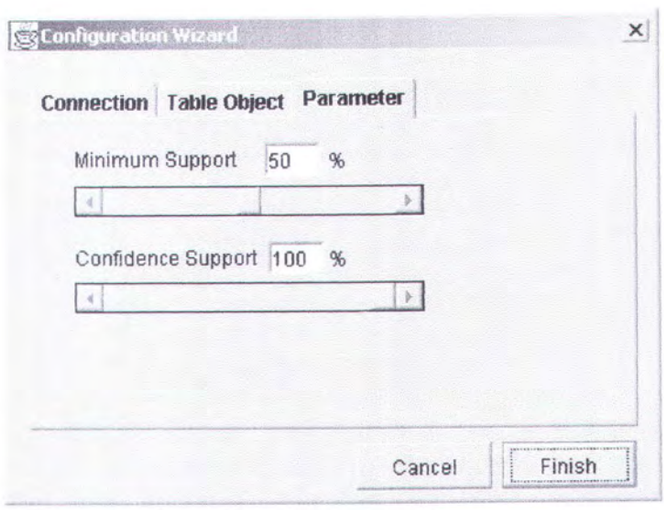
Fungsi untuk sub menu ini telah dijelaskan pada bab perancangan antar muka. Bila memilih sub menu *Configuration Wizard* maka akan tampil form konfigurasi. Form ini memiliki empat tabulasi antara lain tabulasi *Connection* untuk pengisian parameter koneksi, tabulasi *Table Object* untuk penentuan tabel yang digunakan dan tabulasi *Parameter* untuk pengisian variabel *minsup* dan *confsup*. Gambar untuk keempat tabulasi ini dapat dilihat masing-masing pada gambar 4.14, 4.15 dan 4.16.



Gambar 4.14 Form konfigurasi - tabulasi Connection



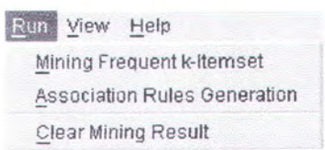
Gambar 4.15 Form konfigurasi – tabulasi Table Object



Gambar 4.16 Form konfigurasi – tabulasi Parameter

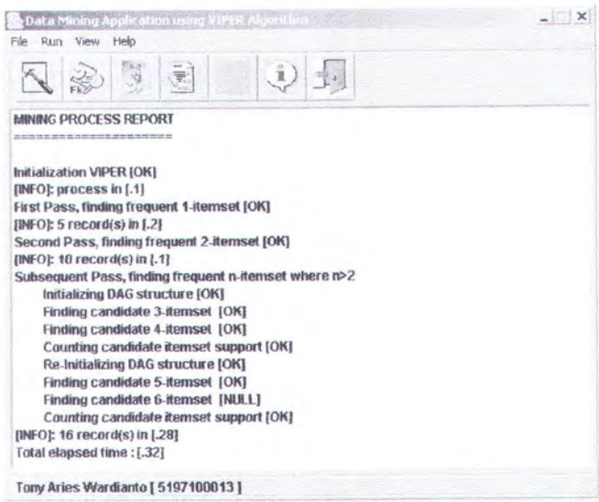
b. Menu Run

Menu *Run* untuk eksekusi proses-proses mining terdiri dari tiga sub menu yang dapat dilihat pada gambar berikut:



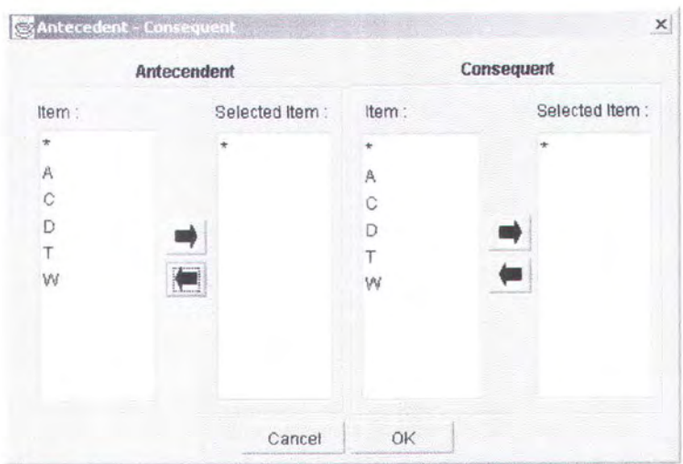
Gambar 4.17 Menu Run

Laporan hasil proses untuk pemilihan sub menu *Mining Frequent k-Itemset* dan *Association Rules Generation* ditampilkan di area teks pada form utama. Berikut ini adalah gambar form utama yang menampilkan laporan proses yang berlangsung selama pencarian k-itemset utama:



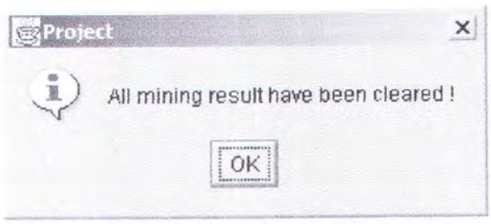
Gambar 4.18 Form utama yang menampilkan informasi proses mining

Ketika pengguna memilih sub menu *Association Rules Generation* maka akan muncul satu form yang berfungsi untuk pengisian nilai variabel *antecedent* dan *consequent* yang diperlukan untuk proses pencarian pola asosiasi. Form tersebut tampak seperti gambar berikut:



Gambar 4.19 Form antecedent – consequent

Bila ingin menghapus seluruh output dan me*refresh* memori dari hasil proses mining, pilih menu *Clear Mining Result*. Setelah proses penghapusan selesai maka akan muncul pesan informasi seperti gambar berikut:



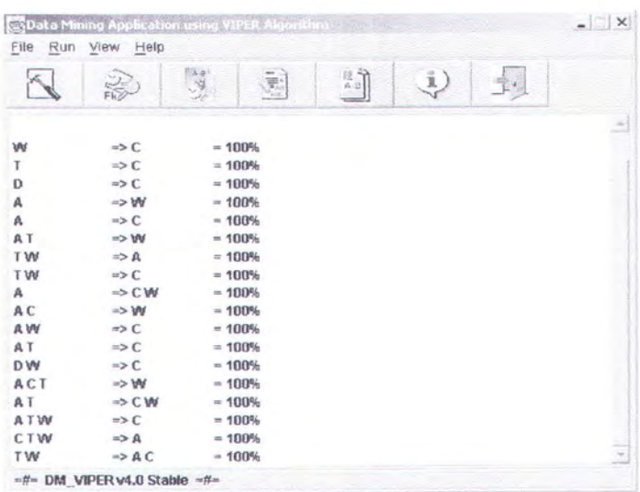
Gambar 4.20 Pesan penghapusan hasil proses mining

c. Menu View

Menu *View u*ntuk menampilkan laporan hasil konfigurasi dan hasil proses mining terdiri dari lima sub menu yang dapat dilihat pada gambar berikut:

Gambar 4.21 Menu View

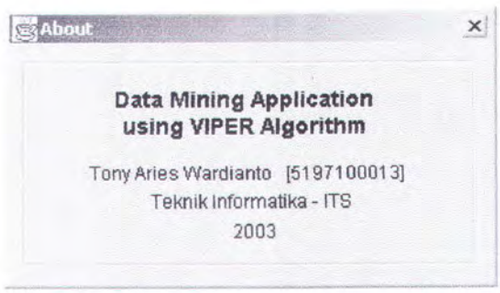
Seluruh laporan akan ditampilkan di area teks pada form utama. Berikut ini adalah gambar form utama yang menampilkan laporan pola asosiasi yang diperoleh:



Gambar 4.22 Form utama yang menampilkan informasi hasil pola asosiasi

d. Menu Help

Pada menu ini hanya terdapat satu submenu yaitu About yang menampilkan form yang berisi informasi pembuat aplikasi. Form tersebut tampak seperti gambar berikut:



Gambar 4.23 Form informasi pembuat aplikasi

Dalam perangkat lunak ini disediakan *toolbar* sebagai alternatif pengaksesan fungsi-fungsi dalam perangkat lunak selain dari sub menu dalam *menubar*. Toolbar terdapat pada bagian atas form utama dan tampak seperti gambar berikut:

Gambar 4.24 Toolbar aplikasi VIPER

Dengan penjelasan urut dari kiri ke kanan, fungsi tombol-tombol tersebut adalah:

- Tombol 1 memiliki fungsi yang sama dengan pemilihan menu **File → Configuration Wizard**
- Tombol 2 memiliki fungsi yang sama dengan pemilihan menu **Run → Mining Frequent k-Itemset**
- Tombol 3 memiliki fungsi yang sama dengan pemilihan menu **Run → Association Rules Generation**.

- Tombol 4 memiliki fungsi yang sama dengan pemilihan menu **View** → **Frequent k-Itemset Result**.
- Tombol 5 memiliki fungsi yang sama dengan pemilihan menu **View** → **Association Rules Result**.
- Tombol 6 memiliki fungsi yang sama dengan pemilihan menu **Help** → **About**.
- Tombol 7 memiliki fungsi yang sama dengan pemilihan menu **File** → **Exit**.

BAB V

UJI COBA DAN EVALUASI

BAB V

UJI COBA DAN EVALUASI

Dalam bab ini diuraikan mengenai hasil uji coba dari perangkat lunak yang telah dibuat. Pengujian ini dilakukan pada beberapa kriteria uji dengan menggunakan data transaksi sintetis yang dibangkitkan dari *data generator*.

5.1 Lingkungan Uji Coba

Pembuatan perangkat lunak ini dilakukan dengan menggunakan bahasa pemrograman Java, IDE Oracle Jdeveloper versi 3.2 pada sistem operasi Windows 2000 Professional. Data transaksi disimpan dalam RDBMS Oracle8i Enterprise versi 8.1.7. Perangkat keras yang digunakan adalah komputer dengan prosesor Intel Pentium III 533 MHz dan memori sebesar 256 Mb. Koneksi ke basis data dilakukan dengan menggunakan *driver* JDBC Thin Driver karena memiliki keuntungan sebagai berikut:

- Diterapkan secara keseluruhan dengan Java
- Dapat di-*download* dari server ke browser
- Tingkat portabilitas-nya tinggi
- Mempergunakan protokol TCP/IP

5.2 Data Uji Coba

Data transaksi yang digunakan sebagai obyek pengujian perangkat lunak data mining ini merupakan data sintetis yang juga digunakan pada pengujian perangkat lunak data mining pada tugas akhir yang lain. Spesifikasi dari setiap

data yang digunakan pada uji coba kinerja perangkat lunak dapat dillihat pada tabel 5.1. Untuk spesifikasi dari setiap data yang digunakan pada uji coba perangkat lunak untuk permintaan berbagai tipe dapat dillihat pada tabel 5.2. Sedangkan untuk spesifikasi dari setiap data yang digunakan pada uji coba kehandalan perangkat lunak dapat dillihat pada tabel 5.3.

Tabel 5.1 Spesifikasi Data Uji Coba Kinerja

	NAMA ITEM	JUMLAH RECORD	JUMLAH TRANSAKSI	JUMLAH ITEM
Data1	Item25	17486	3561	25
Data2	Item30	17974	3561	30
Data3	Item50	70665	10064	50
Data4	Item75	87192	11001	75
Data5	Item100	107426	13588	100

Tabel 5.2 Spesifikasi Data Uji Coba Permintaan Berbagai Tipe

	NAMA ITEM	JUMLAH RECORD	JUMLAH TRANSAKSI	JUMLAH ITEM
Data6	DataBaru1	24627	5000	25
Data7	DataBaru2	66622	7500	50
Data8	DataBaru3	35968	6000	30

Tabel 5.3 Spesifikasi Data Uji Coba Kehandalan

	NAMA ITEM	JUMLAH RECORD	JUMLAH TRANSAKSI	JUMLAH ITEM
Data9	BigData1	247035	50000	1000
Data10	BigData2	499830	50000	59
Data11	BigData3	1228275	50000	99
Data12	BigData4	1998705	100000	1000

5.3 Pelaksanaan Uji Coba

Dalam sub bab ini dijelaskan hasil uji coba perangkat lunak dengan menggunakan data sintetis yang telah disebutkan pada sub bab sebelumnya. Pengujian yang dilakukan meliputi pengujian kebenaran hasil pola asosiasi yang diperoleh, pengujian pencarian pola sesuai permintaan pengguna, pengujian kinerja dan pengujian kehandalan.

5.3.1 Uji coba kebenaran

Skenario uji coba yang dilakukan adalah pencarian pola asosiasi dengan menggunakan obyek tabel sederhana sesuai dengan contoh data transaksi yang digunakan dalam pembahasan algoritma VIPER dalam buku ini. Adapun spesifikasi tabel tersebut adalah sebagai berikut:

Tabel 5.4 Spesifikasi Data Uji Kebenaran

KRITERIA	NILAI
Jumlah Record	23
Jumlah Transaksi	5
Jumlah Item	9

Parameter mining yang digunakan adalah *minsup* = 50%, *confsup* = 100%, *antecedent* dan *consequent* di-set default (*). Hasil yang diperoleh dari proses mining ini sesuai dengan yang tertulis dalam buku. Laporan yang diberikan oleh perangkat lunak adalah sebagai berikut:

```
Frequent 1-itemset number      = 5
Frequent 2-itemset number      = 8
Frequent 3-itemset number      = 5
Frequent 4-itemset number      = 1
Total frequent itemset number = 19
Association rules number       = 18
```

Uji coba kebenaran selanjutnya dilakukan dengan membandingkan hasil mining algoritma VIPER dan hasil yang diperoleh pada algoritma Hybrid [DAN-02] dengan menggunakan data1. Tabel 5.5 dan 5.6 menunjukkan hasil mining kedua algoritma tersebut. Hasil numerik waktu proses mining baik waktu pencarian seluruh itemset utama maupun waktu pembentukan pola asosiasi yang disebutkan pada kedua tabel tersebut disajikan dalam format *detik,milidetik*. Dari hasil ini dapat diketahui bahwa jumlah itemset utama dan jumlah pola asosiasi yang dihasilkan oleh kedua algoritma sama sehingga dapat disimpulkan bahwa hasil yang diperoleh dari algoritma VIPER adalah valid.

Tabel 5.5 Hasil yang diperoleh algoritma HYBRID

Kriteria	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Jml itemset utama	931	392	221	146	112	85	68	59	52	39
Waktu pencarian itemset utama	9,59	4,66	3,85	3,65	3,61	3,50	3,46	3,41	3,37	3,35
Jml pola asosiasi	743	308	175	113	96	74	58	53	51	50
Waktu pembentukan pola asosiasi	0,72	0,15	0,10	0,04	0,05	0,04	0,02	0,01	0,03	0,02

Tabel 5.6 Hasil yang diperoleh algoritma VIPER

Kriteria	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Jml itemset utama	931	392	221	146	112	85	68	59	52	39
Waktu pencarian itemset utama	6,22	4,24	4,07	3,82	3,75	3,35	3,66	3,37	3,61	3,15
Jml pola asosiasi	743	308	175	113	96	74	58	53	51	50
Waktu pembentukan pola asosiasi	0,72	0,51	0,17	0,07	0,05	0,04	0,02	0,03	0,02	0,02

5.3.2 Uji coba permintaan berbagai tipe

Dalam uji coba ini digunakan tiga tipe permintaan pengguna yaitu permintaan tipe I, tipe II dan tipe III. Sedangkan data transaksi yang digunakan adalah data6, data7 dan data8. Adapun tipe permintaan pengguna tersebut adalah sebagai berikut:

- **Permintaan tipe I**

  Untuk permintaan tipe I baik *antecedent* maupun *consequent* terdiri dari item-item yang tidak berisi tanda asterisk(*). Pencarian pola asosiasi dilakukan terhadap item-item yang memenuhi *minsup* dan disebutkan dalam *antecedent* dan *consequent*. Bila asosiasi yang ada memenuhi *minconf* maka asosiasi tersebut termasuk dalam pola asosiasi. Hasil uji coba untuk permintaan I dengan menggunakan data8 dapat dilihat pada tabel 5.7.

- **Permintaan tipe II**

  Tipe II baik *antecedent* atau *consequent* terdiri dari item-item yang berisi tanda asterisk(*). Baik *antecedent* dan *consequent* dapat berisi tanda asteris

saja, item saja atau kombinasi antara keduanya. Sebelum dilakukan pencarian pola asosiasi, dicari kombinasi untuk mengisi nilai *antecedent* atau *consequent* yang mengandung tanda asteris dengan itemset-itemset yang memungkinkan untuk membentuk satu asosiasi. Setiap asosiasi hasil kombinasi diperiksa nilai *confidence*-nya. Bila asosiasi yang diperiksa memenuhi *minconf* maka asosiasi tersebut termasuk dalam pola asosiasi. Hasil uji coba untuk permintaan II dengan beberapa kombinasi asteris pada *antecedent* atau *consequent* dengan menggunakan data7 dapat dilihat pada tabel 5.8.

- **Permintaan tipe III**

  Untuk uji coba permintaan tipe III *antecedent* dan *consequent* diisi dengan tanda asterisk (*). Pencarian asosiasi untuk permintaan tipe ini dilakukan terhadap seluruh itemset yang memenuhi *minsup* dan seluruh kombinasi yang mungkin dan memenuhi nilai *minconf*. Asosiasi yang memenuhi nilai *minconf* termasuk dalam pola asosiasi. Hasil uji coba untuk permintaan II dengan menggunakan data6, data7 dan data8 dapat dilihat pada tabel 5.9.

Dari ketiga tabel hasil uji coba permintaan berbagai tipe dapat diketahui bahwa perangkat lunak yang dibangun dapat memperoleh pola asosiasi yang sesuai dengan permintaan pengguna dengan menentukan batasan-batasan pola asosiasi yang akan dicari melalui parameter *antecedent* dan *consequent*. Berikut ini disajikan hasil uji coba permintaan berbagai tipe dimana hasil numerik waktu proses yang diberikan merupakan waktu yang diperlukan untuk mendapatkan pola asosiasi (tidak termasuk waktu untuk mendapatkan seluruh itemset utama) dan format yang digunakan adalah *detik,milidetik*.

Tabel 5.7 Hasil uji coba permintaan tipe I

	1	2	3	4	5	6	7	8	9	10
Antecedent	Item8	Item23	Item10 Item17	Item18 Item8	Item8	Item24 Item25	Item20	Item20 Item25	Item13	Item13 Item19
Consequent	Item28	Item24	Item8	Item17 Item28	Item17	Item10 Item28	Item25	Item28 Item29	Item19	Item20 Item24
Support	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Confidence	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%
Jumlah Rule	1	0	1	1	1	0	0	0	0	0
Waktu Proses	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1

Tabel 5.8 Hasil uji coba permintaan tipe II

	1	2	3	4	5	6	7	8	9	10
Antecedent	*	Item22	Item28 *	*	Item22 * Item36	*	Item5 *	Item22 Item35	Item48 *	Item45
Consequent	Item22	*	*	Item27 *	*	Item27 Item28 *	Item27 *	*	Item27*	Item27 Item28 *
Support	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Confidence	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%
Jumlah Rule	17	14	35	3	5	9	4	3	1	0
Waktu Proses	0,2	0,1	0,2	0,2	0,1	0,1	0,3	0,1	0,1	0,1

Tabel 5.9 Hasil uji coba permintaan tipe III

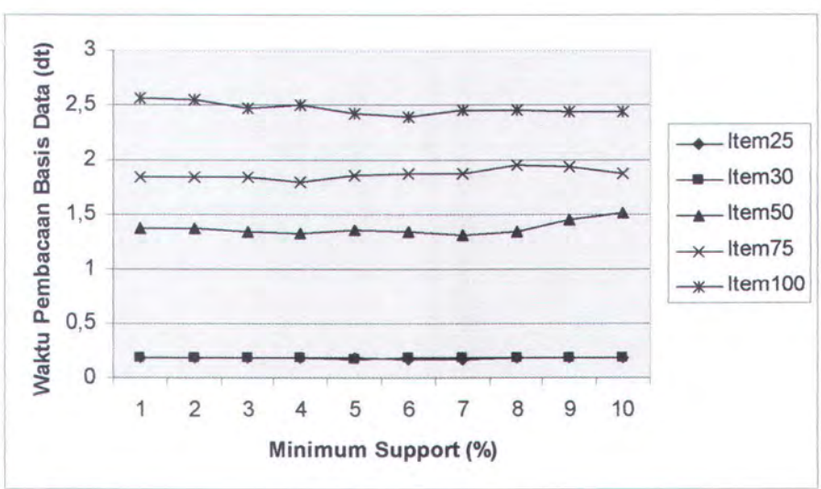
	Data6	Data7	Data8
Antecedent	*	*	*
Consequent	*	*	*
Support	10%	10%	10%
Confidence	25%	25%	25%
Jumlah Rule	89	186	205
Waktu Proses	0,10	0,16	0,14

5.3.3 Uji coba kinerja

Dalam bagian ini diuraikan hasil uji coba kinerja dari perangkat lunak yang telah dibuat dengan melakukan perbandingan nilai minimum support dengan total itemset utama yang ditemukan, jumlah pola asosiasi yang terbentuk, total waktu komputasi, waktu pembacaan basis data, waktu komputasi, dan waktu pembentukan pola asosiasi. Pada setiap perbandingan diberikan grafik hasil uji coba kinerja algoritma. Obyek tabel yang digunakan adalah data1 hingga data5 dengan mengubah parameter *minimum support.*

- **Perbandingan minimum support dengan waktu pembacaan basis data**

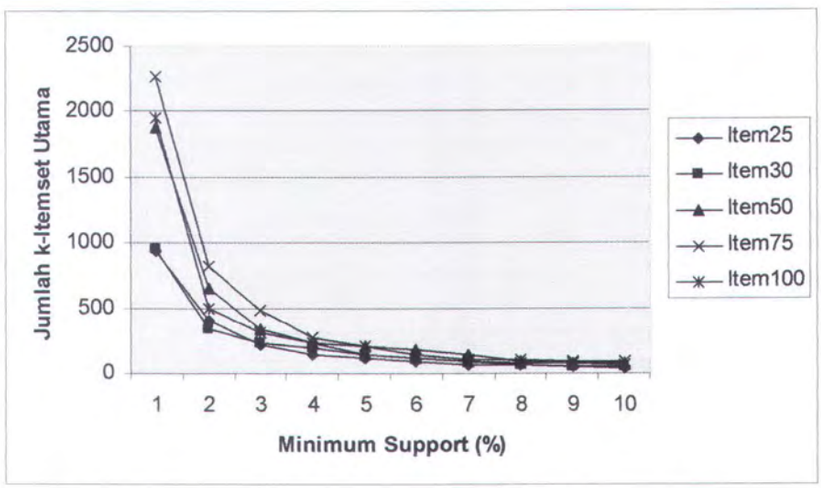
  Waktu pembacaan basis data merupakan waktu untuk mendapatkan seluruh record dari data transaksi. Hasil uji coba perangkat lunak menunjukkan bahwa waktu pembacaan basis data cenderung tetap dan tidak dipengaruhi oleh *minsup* sehingga waktu pembacaan untuk satu obyek tabel cenderung tetap walaupun dilakukan perubahan parameter *minsup*. Grafik pada gambar 5.1 menunjukkan perbandingan nilai *minsup* dengan waktu pembacaan basis data.



Gambar 5.1 Perbandingan *minsup* – waktu pembacaan basis data

- **Perbandingan minimum support dengan total itemset utama yang diperoleh**

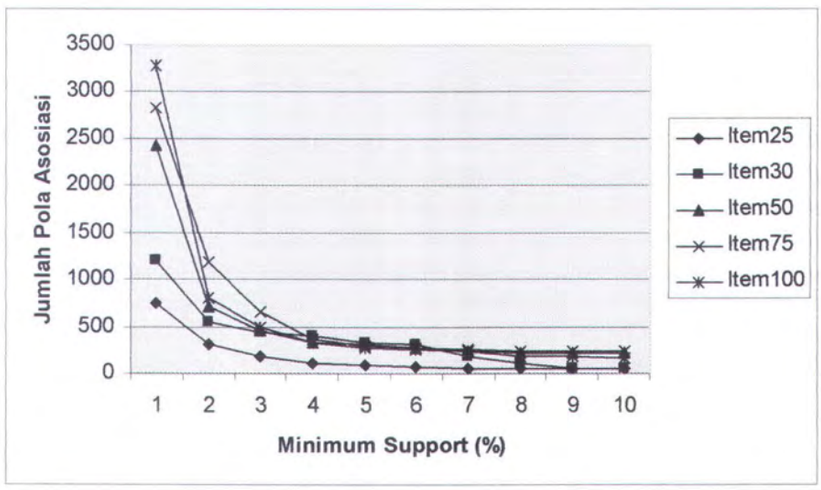
  Hasil pengujian perangkat lunak ini menunjukkan bahwa semakin besar nilai *minsup* maka makin kecil total itemset utama yang diperoleh. Grafik perbandingan nilai *minsup* dengan total itemset utama yang diperoleh dapat dilihat pada gambar berikut:



Gambar 5.2 Perbandingan *minsup* – total itemset utama

- **Perbandingan minimum support dengan jumlah pola asosiasi yang terbentuk**

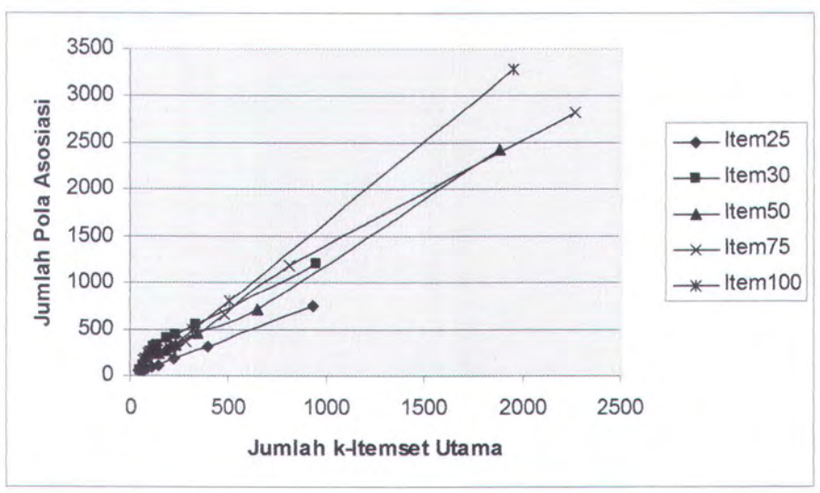
  Hasil pengujian perangkat lunak ini menunjukkan bahwa semakin besar nilai *minsup* maka semakin kecil jumlah pola asosiasi yang terbentuk. Hal ini berkaitan dengan semakin sedikitnya jumlah itemset utama yang dapat dilibatkan dalam pembentukan pola asosiasi. Grafik perbandingan nilai *minsup* dengan jumlah pola asosiasi yang terbentuk dapat dilihat pada gambar 5.3.



Gambar 5.3 Perbandingan *minsup* – jumlah pola asosiasi

- **Perbandingan jumlah itemset utama dengan jumlah pola asosiasi yang terbentuk**

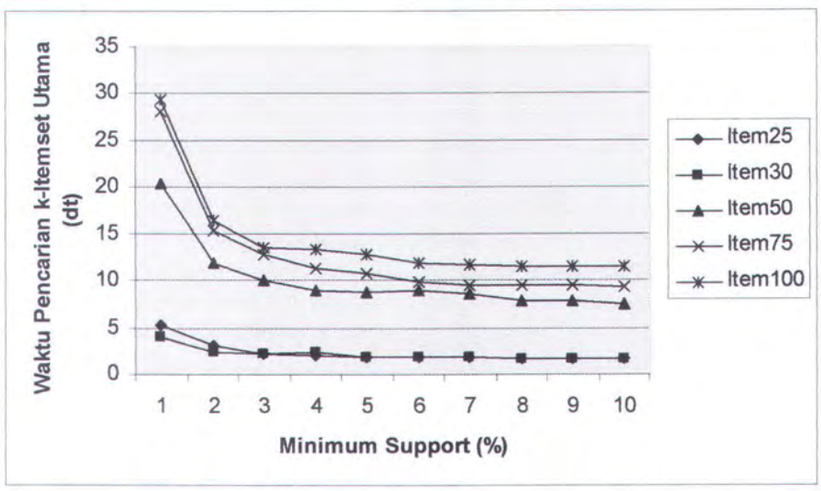
  Hasil pengujian perangkat lunak ini menunjukkan bahwa makin besar jumlah itemset utama maka makin besar pula jumlah pola asosiasi yang terbentuk karena itemset yang terlibat dalam pembentukan asosiasi lebih besar. Grafik perbandingan jumlah itemset utama dengan jumlah pola asosiasi yang terbentuk dapat dilihat pada gambar berikut:



Gambar 5.4 Perbandingan jumlah itemset utama – jumlah pola asosiasi

- **Perbandingan minimum support dengan waktu pencarian k-itemset utama**

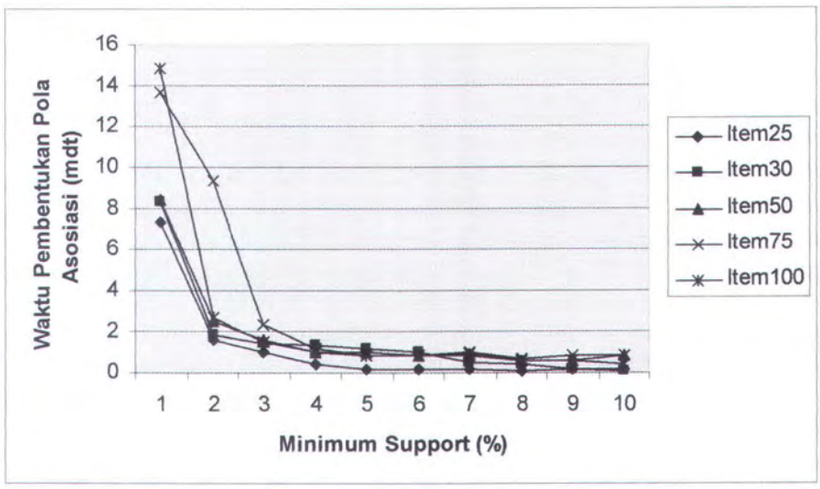
  Hasil pengujian perangkat lunak ini menunjukkan bahwa semakin besar nilai *minsup* maka waktu pencarian k-itemset utama semakin kecil. Hal ini dikarenakan semakin besar nilai *minsup* menyebabkan semakin sedikit pula jumlah itemset utama yang diperoleh dan yang akan diproses untuk tahap mining selanjutnya. Grafik perbandingan nilai *minsup* dengan waktu pencarian k-itemset utama dapat dilihat pada gambar berikut:



Gambar 5.5 Perbandingan *minsup* – waktu pencarian k-itemset utama

- **Perbandingan minimum support dengan waktu pembentukan pola asosiasi**

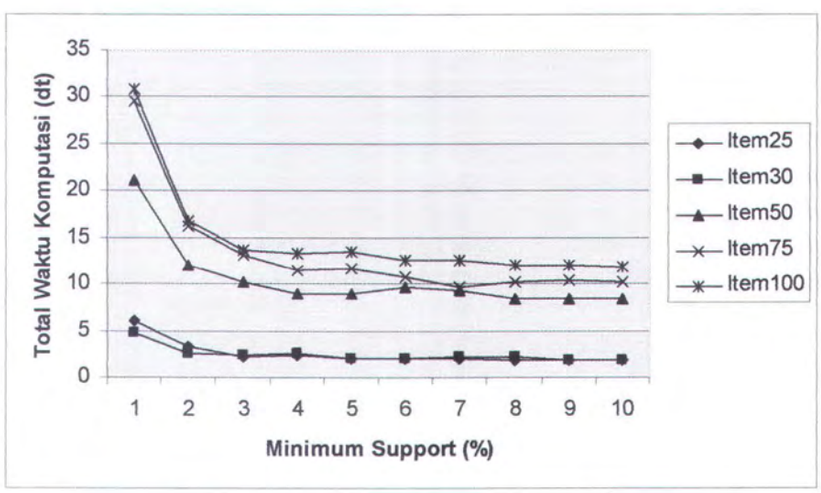
  Hasil pengujian perangkat lunak ini menunjukkan bahwa semakin besar nilai *minsup* maka waktu pembentukan pola asosiasi semakin kecil. Hal ini seiring dengan semakin sedikitnya itemset utama yang diperoleh sehingga asosiasi yang dapat dibentuk dan diperiksa *confidence*-nya juga makin sedikit. Grafik perbandingan nilai *minsup* dengan waktu pembentukan pola asosiasi dapat dilihat pada gambar 5.6.



Gambar 5.6 Perbandingan minsup – waktu pembentukan pola asosiasi

- **Perbandingan minimum support dengan total waktu komputasi**

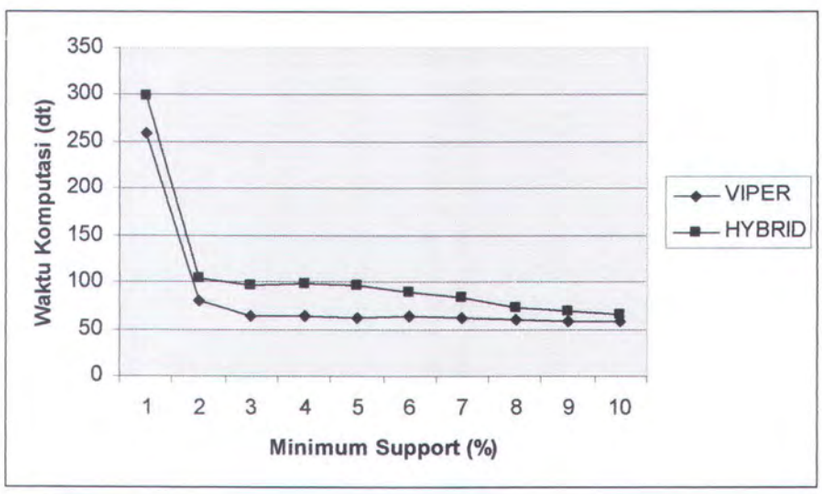
Total waktu komputasi merupakan hasil penjumlahan waktu pencarian itemset utama dan waktu pembentukan pola asosiasi. Hasil pengujian perangkat lunak ini menunjukkan bahwa semakin besar nilai *minsup* maka makin kecil total waktu komputasi yang diperlukan. Ini berkaitan dengan kecenderungan yang sama untuk hubungan *minsup* dengan waktu pencarian itemset utama dan waktu pembentukan pola asosiasi. Grafik perbandingan nilai *minsup* dengan total waktu komputasi dapat dilihat pada gambar 5.7.



Gambar 5.7 Perbandingan *minsup* – total waktu komputasi

- **Perbandingan dengan perangkat lunak lain**

  Pada uji coba ini perangkat lunak dibandingkan dengan perangkat lunak lain yang dibangun dengan menggunakan algoritma Hybrid [DAN-02]. Adapun data yang digunakan adalah data 10 dengan spesifikasi jumlah item 59, jumlah transaksi 50.000 dan jumlah record 499.830. Dengan mengubah parameter *minsup* dari 1% hingga 10% perbandingan dilakukan dengan mencatat waktu yang diperlukan untuk mendapatkan seluruh itemset utama yang memenuhi *minsup* yang ditentukan. Hasil uji coba menunjukkan bahwa kinerja perangkat lunak yang dibangun lebih cepat dibandingkan dengan perangkat lunak lain yang dibangun dengan menggunakan algoritma Hybrid. Berikut adalah grafik yang menunjukkan perbandingan kinerja kedua perangkat lunak ini:



Gambar 5.8 Perbandingan dengan perangkat lunak lain

5.3.4 Uji coba kehandalan

Pada bagian ini dilakukan uji coba kinerja dari perangkat lunak yang dibuat dengan melakukan proses mining terhadap data uji dengan jumlah

transaksi medium dan besar dengan spesifikasi yang disebutkan pada tabel 5.3. Pengujian kehandalan juga dibandingkan dengan hasil yang diperoleh dari perangkat lunak yang menggunakan algoritma Hybrid [DAN-02]. Pengujian dilakukan dengan membandingkan waktu proses mining secara keseluruhan hingga mendapatkan pola asosiasi dan parameter *mining* yang digunakan adalah *minconf* = 25%, *antecedent* = [*], *consequent* = [*].

Tabel 5.10 dan 5.11 menunjukkan hasil uji coba kehandalan dan perbandingan untuk data9 dan data10. Dari hasil ini dapat diketahui bahwa pada data medium waktu proses algoritma VIPER lebih cepat daripada Hybrid. Tabel 5.12 menunjukkan hasil uji coba kehandalan dan perbandingan untuk data11. Hasil uji coba pada data ini menunjukkan bahwa kinerja algoritma VIPER cenderung lebih cepat akan tetapi tidak mampu mendapatkan hasil pada minimum support < 7% karena keterbatasan memori sedangkan untuk algoritma Hybrid masih dapat memperoleh hasil pada minimum support tersebut. Tabel 5.13 menunjukkan hasil uji coba kehandalan dan perbandingan untuk data12. Dari hasil ini dapat diketahui bahwa kinerja algoritma VIPER cenderung lebih cepat dibandingkan dengan algoritma Hybrid. Selain data-data tersebut ada satu data transaksi yang ikut disertakan dalam uji coba kehandalan. Data transaksi dengan nama *jumbo3* memiliki spesifikasi jumlah item 1.000, jumlah transaksi 60.000 dan jumlah record 1.943.087. Tabel 5.14 menunjukkan hasil uji coba kehandalan dan perbandingan untuk *jumbo3*. Dari hasil ini dapat diketahui bahwa kinerja algoritma VIPER cenderung lebih cepat dibandingkan dengan algoritma Hybrid. Hasil numerik berupa waktu proses mining pada tabel berikut disajikan dalam format *jam:menit:detik,milidetik*.

Tabel 5.10 Hasil pengujian kehandalan untuk data9

	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
VIPER	0:0:40,16	0:0:32,91	0:0:30,38	0:0:29,52	0:0:29,33	0:0:29,21	0:0:29,27	0:0:29,28	0:0:28,68	0:0:28,52
HYBRID	0:1:56,50	0:0:57,34	0:0:45,05	0:0:40,69	0:0:39,12	0:0:38,03	0:0:37,47	0:0:37,57	0:0:36,63	0:0:36,79

Tabel 5.11 Hasil pengujian kehandalan untuk data10

	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
VIPER	0:4:22,30	0:1:28,23	0:1:13,16	0:1:11,51	0:1:11,00	0:1:9,95	0:1:9,68	0:1:7,19	0:1:6,42	0:1:6,29
HYBRID	0:5:0,96	0:1:49,30	0:1:41,22	0:1:40,85	0:1:40,72	0:1:33,42	0:1:27,26	0:1:16,66	0:1:12,84	0:1:9,63

Tabel 5.12 Hasil pengujian kehandalan untuk data11

	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
VIPER	-	-	-	-	-	-	0:5:27,46	0:3:24,09	0:3:12,04	0:3:4,52
HYBRID	2:21:44,22	0:46:42,93	0:38:0,70	0:35:21,46	0:29:42,28	0:14:54,34	0:5:55,16	0:3:29,71	0:3:20,10	0:3:17,50

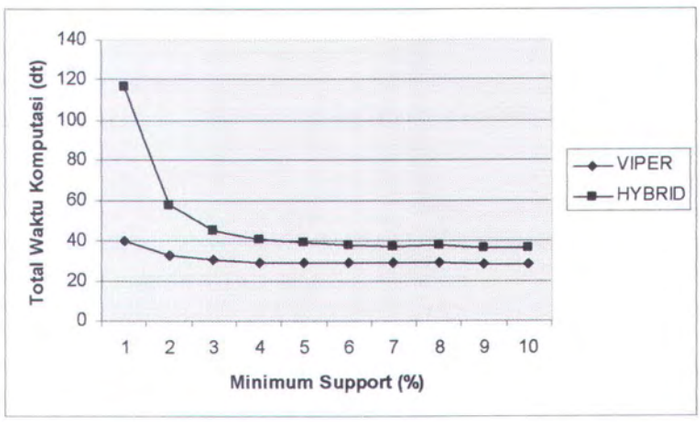
Tabel 5.13 Hasil pengujian kehandalan untuk data12

	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
VIPER	-	0:4:56,91	0:4:29,52	0:4:22,40	0:4:20,63	0:4:18,26	0:4:5,04	0:4:4,82	0:4:5,19	0:4:5,14
HYBRID	2:9:21,07	0:31:28,50	0:4:41,87	0:4:42,07	0:4:43,84	0:4:18,84	0:4:5,97	0:4:6,31	0:4:5,73	0:4:5,24

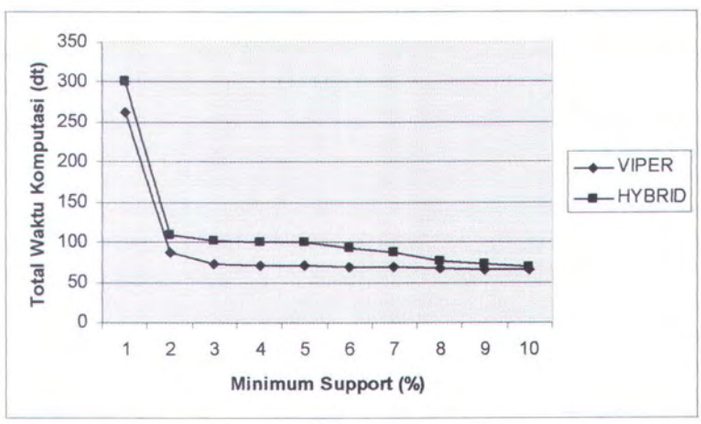
Tabel 5.14 Hasil pengujian kehandalan untuk data "jumbo3"

	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
VIPER	0:29:37,53	0:29:33,40	0:6:13,90	0:4:19,62	0:4:22,38	0:4:17,35	0:4:17,41	0:4:15,94	0:4:15,04	0:4:17,85
HYBRID	1:26:51,99	1:26:31,05	0:25:5,11	0:6:37,99	0:4:35,98	0:4:22,18	0:4:21,89	0:4:21,60	0:4:15,16	0:4:21,03

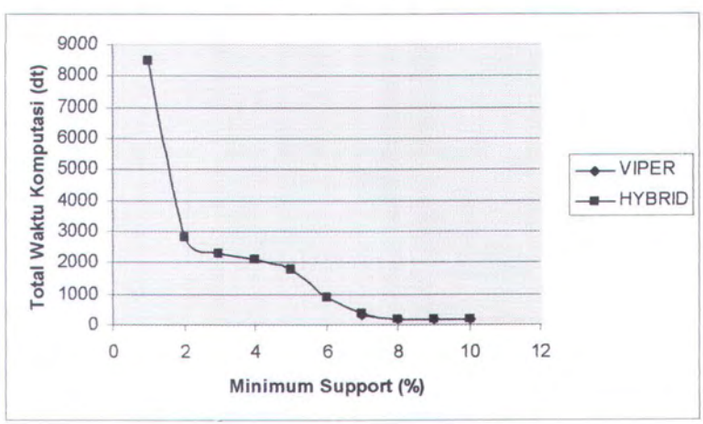
Berikut ini adalah grafik yang merepresentasikan hasil uji coba kehandalan untuk setiap tabel pada halaman 76.



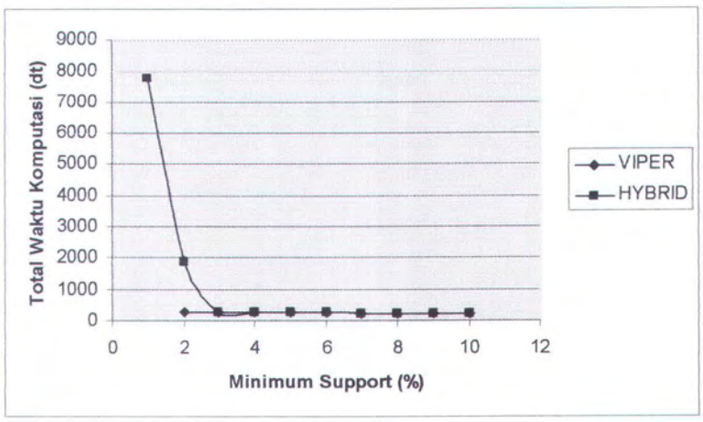
Gambar 5.9 Grafik hasil pengujian kehandalan untuk data9



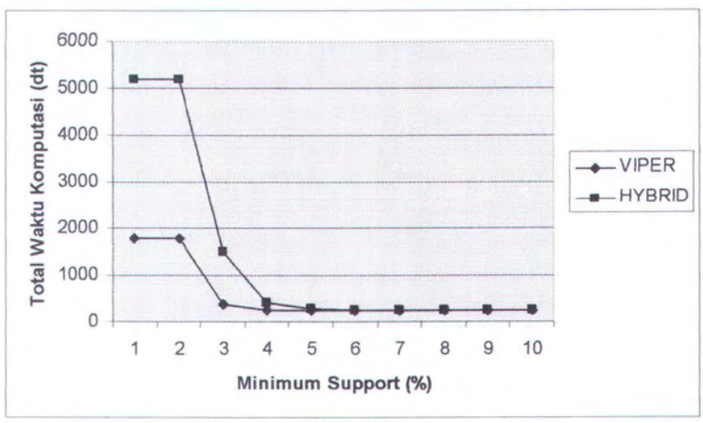
Gambar 5.10 Grafik hasil pengujian kehandalan untuk data10



Gambar 5.11 Grafik hasil pengujian kehandalan untuk data11



Gambar 5.12 Grafik hasil pengujian kehandalan untuk data12



Gambar 5.13 Grafik hasil pengujian kehandalan untuk data "jumbo3"

BAB VI

PENUTUP

BAB VI

PENUTUP

Dalam bab ini diuraikan mengenai beberapa kesimpulan dari tugas akhir yang dibuat berdasarkan uji coba yang telah dilakukan. Selain itu disertakan pula kemungkinan yang dapat dilakukan untuk pengembangan lebih lanjut dari tugas akhir ini.

6.1 Kesimpulan

Dari beberapa hasil uji coba perangkat lunak yang dilakukan, dapat diambil beberapa kesimpulan sebagai berikut:

- Hasil uji coba waktu pembacaan basis data menunjukkan bahwa waktu proses pembacaan basis data mempunyai kecenderungan yang tetap dan tidak dipengaruhi oleh besar kecilnya minimum *support*.
- Hasil uji coba kinerja dengan membandingkan minimum *support* dengan jumlah itemset utama dan pola asosiasi dapat disimpulkan bahwa semakin besar nilai minimum *support*, maka semakin kecil jumlah itemset utama dan pola asosiasi yang diperoleh.
- Hasil uji coba kinerja dengan membandingkan minimum *support* dengan waktu pencarian itemset utama dan pola asosiasi dapat disimpulkan bahwa semakin besar nilai minimum *support*, maka semakin kecil waktu yang diperlukan untuk mendapatkan seluruh itemset utama dan pola asosiasi.
- Hasil uji coba kehandalan dan perbandingan menunjukkan bahwa algoritma ini dapat menyelesaikan proses mining pada obyek data transaksi dengan

DAFTAR PUSTAKA

[SHN-00] P. Shenoy, J. Haritsa, S. Sudarshan, G. Bhalotia, M. Bawa and D. Shah. "Turbo-charging vertical mining of large databases". *In Proc. of ACM SIGMOD Intl. Conf. On Management of Data*. 2000.

[HAN-96] Han, Jiawei and Kamber, Micheline. "Data Mining : Concepts and Techniques". *http://www.cs.sfu.ca*. Intelligent Database Systems Research Lab School of Computing Science Simon Fraser University, Canada. 2000.

[HAN-97] Han, Jiawei. "OLAP Mining: An Integration of OLAP With Data Mining". Chapman and Hall. 1997.

[LIN-97] Beery, J.A and Linoff G. *Data mining techniques for marketing, sales and customer support.* New York: Wiley. 1997.

[ZAK-97] Zaki, M.J, S. Parthasarathy, M. Ogihara, and W. Li. "New Algorithms for fast discovery of association rules". *In Proc. of 3rd Intl. Conf. on Knoledge Discovery and Data Mining (KDD).* August 1997.

[AGR-93] R. Agrawal and R. Srikant. "Fast Algorithm for Mining Association Rules". *In Proceedings of the International Conference on Very Large Data Bases*, 1994.

[ADH-02] Pratiwi, Adhita. S.Kom. *Tugas Akhir Perancangan Perangkat Lunak Pembangkit Data.* Tugas Akhir. Jurusan Teknik Informatika FTIF ITS Surabaya. 2002.

[PRA-00] Ananto, Pramudyo. S.Kom. *Perancangan dan Pembuatan Perangkat Lunak Data Mining Untuk Menemukan Aturan-aturan Sesuai Dengan permintaan Pengguna.* Tugas Akhir. Jurusan Teknik Informatika FTIF ITS Surabaya. 2000.

[DAN-02] Tyaspamadya, Daning. S.Kom. *Perancangan dan Pembuatan Perangkat Lunak Data Mining untuk Penggalian Kaidah Asosiasi Menggunakan Metode Hybrid.* Tugas Akhir. Tugas Akhir. Jurusan Teknik Informatika FTIF ITS Surabaya. 2002.

[PUT-02] Darmasusila, I Putu. S.Kom. *Pengembangan Aplikasi Data Mining Untuk Pencarian Pola Asosiasi Dengan Metode Pattern Decomposition.* Tugas Akhir. Jurusan Teknik Informatika FTIF ITS Surabaya. 2002.

LAMPIRAN

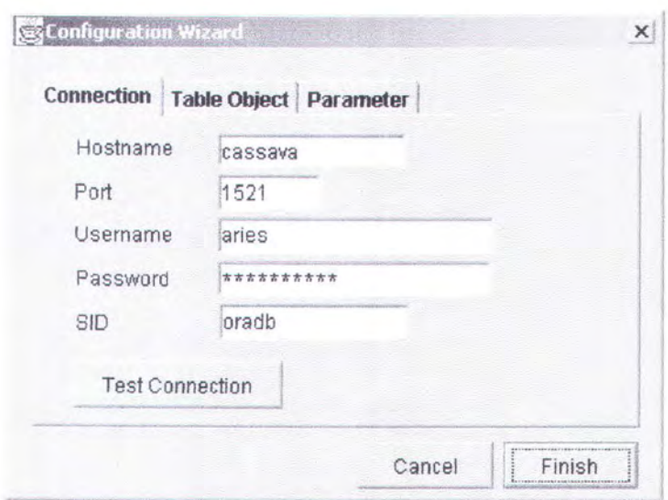
LAMPIRAN A

PETUNJUK PENGGUNAAN PROGRAM

Untuk menjalankan program, lakukan klik ganda pada file **run.bat** yang telah diubah. Jika terjadi kesalahan, periksa apakah *path* yang dimasukkan telah sesuai dengan petunjuk.

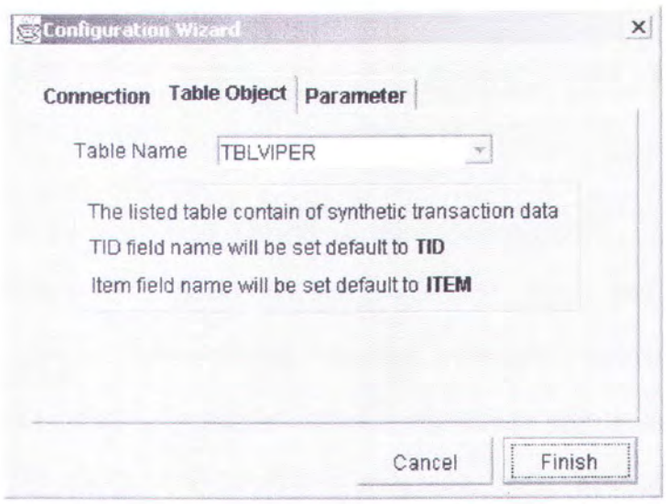
1. Koneksi basis data dan setting parameter *mining*

   Untuk melakukan koneksi basis data dan mengisi parameter *mining* pilih menu **File** kemudian pilih sub menu **Configuration Wizard** atau bisa juga dengan memilih toolbar pada index 1. Tampilan yang akan muncul tampak seperti gambar berikut:



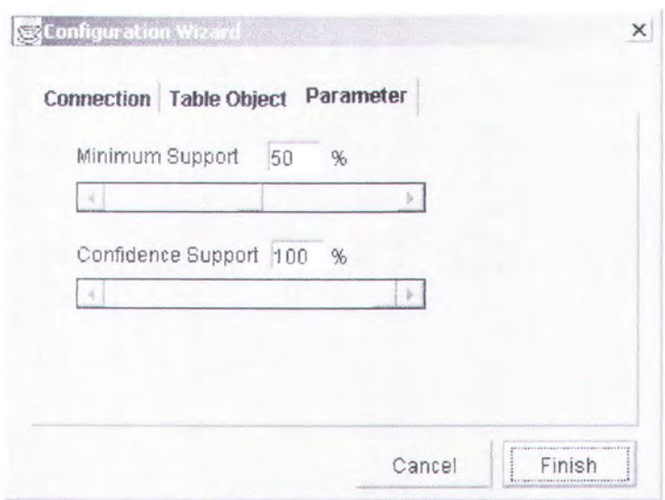
Gambar A.1 Form Configuration Wizard tab. Connection

Pada tabulasi **Connection**, pengguna dapat mengisikan parameter yang diperlukan untuk koneksi ke basis data di Oracle. Tabulasi kedua adalah **Table Object** seperti pada gambar A.2 digunakan untuk memilih tabel yang akan dijadikan sebagai obyek mining.



Gambar A.2 Form Configuration Wizard tab. Table Object

Tabulasi ketiga adalah **Parameter** seperti pada gambar A.3 digunakan untuk mengisi nilai prosentase **minimum support** dan **confidence support**.



Gambar A.3 Form Configuration Wizard tab. Parameter

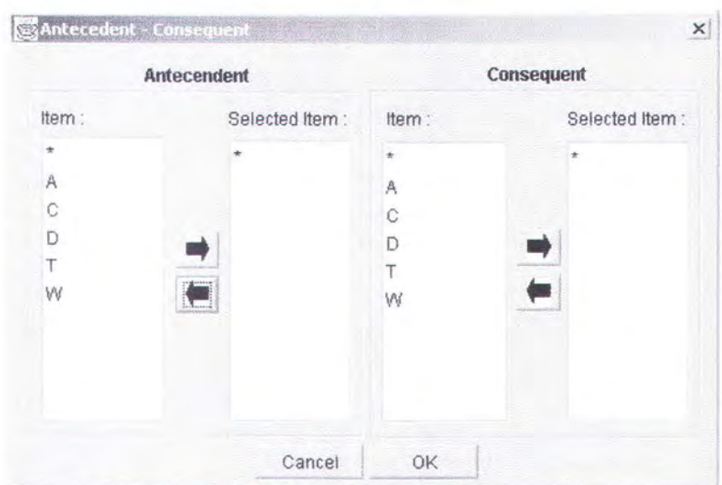
2. Mengetahui informasi koneksi dan parameter mining

   Jika ingin mengetahui informasi koneksi basis data dan parameter mining yang telah dilakukan, pilih menu **View** kemudian pilh sub menu **Configuration Summary**. Informasi baik mengenai konfigurasi mining

ataupun laporan hasil proses mining akan muncul di area teks pada form utama.

3. Menjalankan proses mining dan mendapatkan pola asosiasi

   Untuk menjalankan proses pencarian k-itemset utama pilih menu **Run** kemudian pilih sub menu **mining Frequent k-Itemset** atau dengan memilih toolbar pada index 2. Sedangkan untuk menjalankan proses pencarian pola asosiasi pilih menu **Run** kemudian pilih sub menu **Association Rules Generation** atau dengan memilih toolbar pada index 3. Untuk proses pencarian pola asosiasi ini akan muncul form seperti pada gambar A.4 untuk pengisian variabel *antecedent* dan *consequent*.

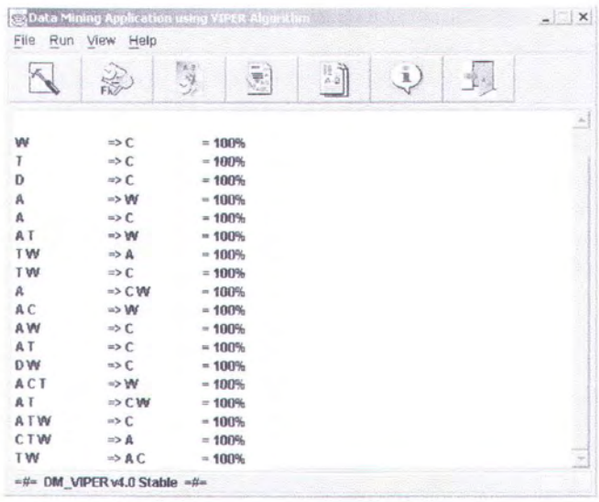


Gambar A.4 Form pengisian variabel antecedent - consequent

4. Mengetahui laporan hasil proses mining

   Untuk mendapatkan laporan hasil k-itemset utama yang diperoleh, pilih menu **View** kemudian pilih sub menu **Frequent k-Itemset Result** atau dengan memilih toolbar pada index 4. Sedangkan untuk mendapatkan laporan hasil pola asosiasi yang diperoleh, pilih menu **View** kemudian pilih sub menu **Association Rules Result** atau dengan memilih toolbar pada index 5.

Contoh laporan hasil polas asosiasi yang diperoleh dapat dilihat pada gambar berikut:



Gambar A.5 Form laporan hasil pola asosiasi

5. Menutup aplikasi

Menutup aplikasi dapat dilakukan dengan memilih menu **File** kemudian pilih sub menu **Exit** atau dapat juga dengan memilih toolbar pada index 7.