ANALISA KECEPATAN ALGORITMA PENELUSURAN, ALGORITMA APRIORI DAN ALGORITMA APRIORITID UNTUK MEMPEROLEH FREQUENT ITEMSET PADA ASSOCIATION RULE

SKRIPSI

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

Oleh :
JEFRI YUDI TUMYWA
0310963014-96

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

2008


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

ANALISA KECEPATAN ALGORITMA PENELUSURAN, ALGORITMA APRIORI DAN ALGORITMA APRIORITID UNTUK MEMPEROLEH FREQUENT ITEMSET PADA ASSOCIATION RULE

Oleh:
JEFRI YUDI TUMYWA
0310963014-96

Setelah dipertahankan di depan Majelis Penguji
Pada tanggal 8 Januari 2008
dan dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Wayan Firdaus M.,Ssi.,MT	**Dian Eka R,Ssi.,M.Kom**
NIP. 132 158 724	**NIP. 132 300 224**

Mengetahui,
a.n. Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya
Sekretaris,

Dra. Ani Budi Astuti,M.Si
NIP. 131 993 385


UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	**:**	Jefri Yudi Tumywa
NIM	**:**	0310963014
Jurusan	**:**	Matematika
Penulis skripsi berjudul	**:**	Analisa Kecepatan Algoritma Penelusuran, Algoritma Apriori dan Algoritma AprioriTid Untuk Memperoleh Frequent Itemset Pada Association Rule.

Dengan ini menyatakan bahwa :

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

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 8 Januari 2008
Yang menyatakan,

(Jefri Yudi Tumywa)
NIM. 0310963014

ANALISA KECEPATAN ALGORITMA PENELUSURAN, ALGORITMA APRIORI DAN ALGORITMA APRIORITID UNTUK MEMPEROLEH FREQUENT ITEMSET PADA ASSOCIATION RULE

ABSTRAK

Association rule adalah suatu teknik dalam data minning yang digunakan untuk mencari hubungan keterkaitan antara satu item atau beberapa item dengan item yang lain. Teknik ini memiliki dua tahap yaitu mencari pola frekuensi item atau *frequent itemset* dan kemudian mencari *association rule* dari itemset tersebut. Algoritma apriori dan AprioriTid adalah dua jenis algoritma yang sering digunakan untuk mempercepat proses pencarian *frequent itemset*. Penelitian ini akan melakukan uji coba waktu proses tiga jenis algoritma, yaitu algoritma penelusuran, Apriori dan Aprioritid yang di terapkan dalam bahasa *query* untuk melakukan pencarian *frequent itemset* dengan minsup yang berbeda-beda terhadap empat jenis data. Penelitian ini ditujukan untuk menganalisa waktu proses pencarian *Association rule* yang menggunakan algoritma penelusuran, Apriori dan AprioriTid dalam mencari pola frekuensi item. Dari pengujian yang dilakukan secara berulang-ulang didapatkan hasil bahwa penggunaan AprioriTid dapat mengoptimasi kecepatan proses pencarian *association rule* hingga rata-rata 76 % lebih cepat dari pada menggunakan algoritma penelusuran sedangkan algoritma Apriori rata-rata 55% lebih cepat dari pada algoritma penelusuran. Rata-rata selisih waktu proses antar minsup pada algoritma penelusuran, Apriori dan Aprioritid berturut-turut ialah 2,4 detik, 18,55 detik dan 10,23 detik. Hal ini menunjukkan bahwa algoritma Apriori memiliki waktu proses antar minsup yang paling tidak stabil karena memiliki selisih waktu antar minsup yang paling signifikan
.

SPEED ANALYSIS OF NON APRIORI ALGORTHM, APRIORI ALGORITHM AND APRIORITID ALGORITHM TO GET THE FREQUENT ITEMSET ON ASSOCIATION RULE

ABSTRACT

Association Rule is a data mining technique used to find the relation between an item with another. This technique has two steps, finding the item frequency pattern or frequent itemset and then searching the association rule for the itemset. Apriori and AprioriTid are two kinds of algorithm that is frequently used to speed up the frequent itemset seeking process. This research will analysis the time of the three different algorithms, non apriori, apriori and aprioritid algorithm, applied in query language to find frequent itemset with different minsup against four kinds of data. This research is aimed to analyze the association rule's processing time using non apriori, apriori and aprioritid to find item frequency pattern. Repeated tests show that the usage of aprioritid can optimize the association rule's processing time up to 76% faster than non apriori algorithm, while apriori algorithm is 55% faster than non apriori algorithm. The mean difference between minsup in non apriori, apriori and aprioritid algorithm is 2.4 seconds, 18.55 seconds and 10.23 seconds. This shows that apriori algorithm has the most unstable processing time between minsup because it has the most significant time difference between minsup.

Kata Pengantar

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan limpahan hidayahNya, penulis masih dapat belajar dan mengerjakan Tugas Akhir yang berjudul "Analisa Kecepatan Algoritma Penelusuran, Algoritma Apriori dan Algoritma AprioriTid untuk Memperoleh Frequent Itemset pada Association Rule". Tugas Akhir ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Ilmu Komputer, jurusan Matematika, fakultas MIPA, universitas Brawijaya.

Semoga Allah melimpahkan rahmat atas Nabi Muhammad SAW, makhluk paling mulia yang senantiasa memberikan cahaya petunjuk, dan atas keluarganya dan sahabat-sahabatnya..

Dalam penyelesaian tugas akhir ini, penulis telah mendapat begitu banyak bantuan baik moral maupun materiil dari banyak pihak. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Wayan Firdaus Mahmudy, SSi, MT sebagai Pembimbing I dan ketua Program Studi Ilmu Komputer Unibraw Malang dan Dian Eka Ratnawati, Ssi, M.Kom selaku pembimbing II. Terima kasih atas semua waktu dan bimbingan yang telah diberikan.
2. Drs. Nurul Hidayat, Spd. , Msc selaku Penasihat Akademik.
3. Segenap bapak dan ibu dosen yang telah mendidik dan mengamalkan ilmunya kepada penulis.
4. Segenap staf dan karyawan di Jurusan Matematika FMIPA Universitas Brawijaya
5. Ayah, Ibu, Kakak dan Adik. Terima kasih atas cinta, kasih sayang, doa, dukungan dan semangat yang tiada henti.
6. Sahabat- sahabat ilkomers `03 dan seluruh penghuni ilkom.
7. Pihak lain yang telah membantu terselesaikannnya Tugas Akhir ini yang tidak bisa penulis sebutkan satu-persatu.

Penulis sadari bahwa masih banyak kekurangan dalam laporan ini disebabkan oleh keterbatasan kemampuan dan pengalaman. Oleh karena itu Penulis sangat menghargai saran dan kritik yang sifatnya membangun demi perbaikan penulisan dan mutu isi tugas akhir ini untuk kelanjutan penelitian serupa di masa mendatang.

Penulis berharap semoga tugas akhir ini dapat memberikan manfaat kepada pembaca dan bisa diambil manfaatnya, baik oleh Penulis selaku mahasiswa maupun pihak-pihak lain yang tertarik untuk menekuni pengembangan aplikasi dengan menggunakan data mining.

Malang, 31 Desember 2007

Penulis

DAFTAR ISI

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pemilik usaha atau manager pemasaran suatu bidang usaha seperti departemen store atau supermarket, tentunya harus mengetahui permintaan pasar dan berusaha untuk memenuhinya. Informasi tentang item apa saja yang sedang popular di pasaran dan bagaimana hubungan item tersebut dengan item yang lain tentunya menjadi sangat penting. Sebagai contoh misalnya seberapa besar kecenderungan seorang pembeli roti untuk membeli mentega.

Untuk mencari hubungan antar item tersebut dibutuhkan teknik dalam *data mining* yang bernama *Association rule*. *Association rule* adalah suatu metode dalam *data mining* untuk mendapatkan informasi yang berharga dari data-data yang jumlahnya sangat banyak. Teknik *association rule* melakukan pembacaan data pada daftar transaksi untuk menemukan hubungan antar item

Permasalahannya ialah makin banyak transaksi harian yang terjadi, maka makin besar pula jumlah data transaksi yang disimpan. Untuk melakukan pembacaan sebanyak itu, dengan menggunakan penelusuran biasa, komputer akan membutuhkan waktu yang sangat lama.

Salah satu cara mengatasi permasalahan di atas adalah dengan menyertakan algoritma Apriori atau AprioriTid ke dalam proses *association rule*. Algoritma Apriori dan AprioriTid adalah dua algoritma paling terkenal dan sering digunakan untuk mencari frekuensi kemunculan suatu item atau sekelompok item.

Sebelumnya telah ada penelitian yang membandingkan kecepatan waktu pada algoritma AIS, SETM, Apriori, AprioriTid dan AprioriHybrid. Penelitian tersebut dilakukan pertama kali oleh Rakesh Agrawal dan Ramakrishnan Srikant pada tahun 1994 dengan menerapkannya pada DBMS DB2/6000 dengan jumlah transaksi sebanyak 100.000 transaksi. Hasil dari penelitian diatas ditulisakan dalam jurnal yang berjudul *Fast Algorithms for mining Association Rule.* Salah satu Analisa yang didapat dari membandingkan kecepatan algoritma Apriori dan AprioriTid ialah bahwa algoritma AprioriTid mempunyai kinerja yang lebih baik atau lebih cepat dari pada algoritma Apriori ketika daftar transaksi yang baru memiliki jumlah data yang lebih kecil daripada daftar transaksi sebelumnya.

Tugas akhir ini akan mencoba untuk menganalisa kecepatan kinerja algoritma penulusuran, algoritma Apriori dan algoritma AprioriTid untuk mencari frekuensi kemunculan sekelompok item dan mencari hubungan item tersebut dengan item yang lain. Algoritma-algoritma di atas akan diterapkan dalam bahasa *query* SQL. Penelitian ini nantinya akan dilakukan dengan jumlah data, jenis parameter data, dan parameter ambang batas yang berbeda dengan penelitian sebelumnya. Hasil yang diperoleh dari penelitain ini berupa analisa kecepatan ketiga algoritma dalam bentuk grafik waktu.

Karena ketiga algoritma diterapkan dalam bahasa *query* maka *Database Manajemen System* akan mempunyai peranan yang penting. *Database Management Systems* (DBMS) adalah perangkat lunak untuk mendefinisikan, menciptakan, mengelola dan mengendalikan pengaksesan basis data(Nugroho,2004). Dengan menggunakan fitur dalam DBMS, pengaturan data jauh lebih efisien dibandingkan dengan penyimpanan data pada berkas (*file system based*).

Berdasarkan latar belakang yang telah dipaparkan di atas maka pada penulisan ini penulis akan mengambil judul "**Analisa Kecepatan Algoritma Penelusuran, Algoritma Apriori dan Algoritma AprioriTid untuk Memperoleh Frequent Itemset pada Association Rule".**

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan di atas maka rumusan masalah yang dikerjakan sebagai berkut.

1. Bagaimana menerapkan algoritma penelusuran,algoritma Apriori dan Algoritma AprioriTid untuk mencari pola frekuensi item dengan menggunakan bahasa *query*?
2. Bagaimana analisa waktu kinerja pencarian *association rule* yang menggunakan algoritma penelusuran, Apriori dan AprioriTid dalam mencari pola frekuensi item?

1.3 Batasan Masalah

Pembatasan masalah pada penulisan tugas akhir ini adalah :

1. Algoritma penelusuran, algoritma Apriori dan algoritma AprioriTid diterapkan pada bentuk bahasa *query* atau SQL.

2. Algoritma Apriori dan AprioriTid yang digunakan sesuai dengan algoritma yang dituliskan pada jurnal "Fast Algorithm for Mining Association Rule".
3. Struktur tabel yang digunakan adalah struktur tabel yang menyediakan satu kolom untuk satu kombinasi item atau multikolom.
4. Hubungan item yang dicari sebanyak 4 kombinasi item.
5. Database manajemen system yang digunakan adalah SQL server 2000.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai pada penelitian tugas akhir ini adalah :

1. Menerapkan algoritma penelusuran, algoritma Apriori dan Algoritma AprioriTid untuk mencari pola frekuensi item dengan menggunakan bahasa *query*.
2. Menganalisa waktu kinerja pencarian *Association rule* yang menggunakan algoitma penelusuran, Apriori dan AprioriTid dalam mencari pola frekuensi item.

1.5 Manfaat Penelitian

Manfaat yang dapat diambil pada penelitian ini adalah dapat membantu memberikan gambaran seberapa cepat pencarian association rule dengan menggunakan algoritma Apriori atau aprioriTid dibanding dengan menggunakan algoritma penelusuran.

1.6 Sistematika Penulisan

Pembuatan tugas akhir ini dilakukan dengan pembagian bab sebagai berikut.

BAB I : PENDAHULUAN

Bab ini membahas mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika penulisan tugas akhir.

BAB II : TINJAUAN PUSTAKA

Bab ini membahas dan menjelaskan mengenai konsep dasar basis data, SQL, *stored procedure*, *Association Rule*, algoritma penelusuran , algoritma Apriori, dan algoritma AprioriTid serta dilengkapi dengan referensi baik dari buku maupun internet untuk melengkapi dasar – dasar teori yang ada.

BAB III : METODOLOGI DAN PERANCANGAN

Bab ini menerangkan beberapa hal mengenai jenis-jenis data yang akan diuji, rancangan basis data dan tabel, spesifikasi perangkat lunak dan perangkat keras yang digunakan dalam penelitian, langkah-langkah dalam melakukan penelitian, dan gambaran penerapan masing-masing algoritma dalam bahasa *query*.

BAB IV : HASIL DAN PEMBAHASAN

Bab ini menerangkan proses implementasi dari rancangan penelitian yang dijelaskan pada bab III. Implementasi yang dijelaskan terdiri dari implementasi basis data, implementasi *query* dan implementasi program. Bab ini juga menjelaskan penerapan pada aplikasi dan analisa yang diperoleh dari data dan grafik waktu yang di dapat dari pengujian yang dilakukan.

BAB V : KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari pembahasan dan saran yang diharapkan bermanfaat untuk pengembangan tugas akhir ini selanjutnya.

BAB II
DASAR TEORI

2.1 Basis Data

2.1.1 Definisi

Konsep dasar dari basis data adalah koleksi dari data-data yang terorganisir dengan cara sedemikian rupa sehingga data mudah disimpan dan dimanipulasi(diperbaharui, dicari ,diolah dengan perhitungan-perhitungan tertentu dan dihapus)(Silberschatz,2002). Secara teoritis, basis data tidak selalu berhubungan dengan komputer.

Pengertian basis data jika dipandang dari ilmu komputer adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diolah menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut(Nugroho,2004).

Dari definisi di atas, terdapat tiga hal yang berhubungan dengan basis data, yaitu :

1. data itu sendiri yang diorganisasikan dalam bentuk basis data (*database*).
2. media penyimpanan (*storage*) untuk menyimpan basis data tersebut. Media ini merupakan bagian dari tehnologi perangkat keras yang digunakan di sistem informasi. Media penyimpanan yang umumnya digunakan berupa *harddisk*.
3. perangkat lunak untuk memanipulasi basis data. Perangkat lunak ini dapat dibuat sendiri dengan menggunakan bahasa pemrograman komputer atau didapat dalam bentuk suatu paket. Banyak paket perangkat lunak yang disediakan untuk memanipulasi basis data. Paket perangkat lunak ini disebut dengan DBMS (*Database Management Systems*).

2.1.2 SQL

Perintah yang digunakan untuk mengolah data dan tabel yang tersimpan dalam basis data ialah berupa bahasa *query*. *Structured Query Language*(SQL) adalah bahasa *query* standar, yang digunakan pada sebagian besar aplikasi basis data. SQL sering digunakan oleh programmer untuk membuat aplikasi pengolah *database* dan biasanya di gunakan oleh administrator *database* untuk membuat atau memelihara *database*(Nugroho,2004).

Secara umum, bahasa SQL memiliki beberapa bagian, yaitu : (Nugroho,2004)

1. DDL (*Data Definition Language*)
   Kelompok perintah SQL yang berfungsi untuk mendefinisikan struktur suatu *database* atau tabel,contoh : CREATE, ALTER, DROP.
2. DML (*Data Manipulation Language*)
   Kelompok perintah SQL yang berfungsi untuk memanipulasi data yang terdapat pada *database* atau tabel, Contoh : SELECT, INSERT, UPDATE, DELETE.
3. *View Definition*
   SQL memuat perintah-perintah untuk mendefinisikan tampilan-tampilan (*view*)yang dikehendaki pengguna.
4. *Transaction Cotrol*
   Kelompok perintah SQL yang berfungsi untuk menspesifikasi awal dan akhir suatu transaksi, contoh: COMMIT, ROLLBACK.
5. *Embeded SQL* dan *Dynamic SQL*
   Terminologi ini mencangkup kemampuan SQL untuk disisipkan pada beberapa bahasa pemrograman misalnya pada Visual Basic,Delphi,Java dan sebagainya.
6. *Integrity*
   SQL DDL mencangkup perintah-perintah untuk menspesifikasi batasan-batasan integritas.
7. *Authorization*
   SQL DDL mencangkup perintah-perintah untuk membatasi akses pada basis data demi alasan keamanan.

Berikut ini adalah contoh bahasa *query* untuk DDL pada pembuatan tabel gudang, dan *query* DML standar(*select, from, where*) untuk mendapatkan nama item pada table gudang yang memilki id = ’24’.

```
create Gudang
   (  Item_id integer,
      Item_name char(50)  );
select Item_name
from Gudang
where Item_id = 24;
```

2.1.3 Stored procedure

Stored procedure ialah fungsi-fungi atau prosedur yang didefinisikan oleh administrator *database* untuk memudahkan mengolah data dalam tabel(Suhendric,2001). *Stored procedure* ini

disimpan dalam *database* sehingga proses eksekusi nya terdapat pada server *database*. Keberadaan *stored procedure* ini sangat berguna bagi seorang pembuat aplikasi *database* karena dengan adanya *stored procedure*, maka kode program yang dibuat pada aplikasi jauh lebih sederhana. Selain itu keberadaan *stored procedure* pada server *database* juga mampu mengurangi beban traffic pada jaringan komputer(Santoso,2006).

Server *database* meng-compile *stored procedure* hanya sekali yaitu pada saat *stored procedure* tersebut dibentuk pertama kali. Hasil dari proses meng-*compile* tersebut ialah berupa *execution plan* (rencana eksekusi) yang memiliki *cost query* yang paling minimum(Silberschatz,2002). *Execution plan* ialah kumpulan proses yang akan di eksekusi secara berurutan untuk menghasilkan data dari query yang dicompile. Rencana eksekusi tersebut disimpan dalam memori dan dapat digunakan kembali jika *stored procedure* yang sama di jalankan lagi.

Stored procedure di buat dalam bahasa SQL dan hanya mampu dibuat,dirubah atau dihapus oleh administrator *database*. Berikut ini adalah contoh pembuatan *stored procedure*

```
Create Procedure sp_barang(@kode as varchar(10) )
AS
  Select * from tb_barang where kode_barang=@kode
Go
```

Stored procedure diatas jika di jalankan akan menampilkan identitas barang tertentu sesuai dengan yang yang diinginkan. Perintah untuk menjalankan *stored procedure* diatas ialah

```
Exec sp_barang '11-1';
```

2.2 Data Mining

2.2.1 Definisi

Berdasarkan pengertian yang tercantum dalam buku "*Advances in Knowledge Discovery and Data mining*" *Data mining* atau *Knowledge Discovery in Database* (KDD) adalah keseluruhan proses non-trivial untuk mencari dan mengidentifikasi pola (*pattern*) dalam data, dimana pola yang ditemukan bersifat sah (*valid*), baru (*novel*), dapat bermanfaat (*potentially usefull*), dan dapat dimengerti (*ultimately understandable*)(William,2001).

Istilah *data mining* dan *knowledge discovery in databases* (KDD) sering kali digunakan secara bergantian untuk menjelaskan

proses penggalian informasi tersembunyi dalam suatu basis data yang besar. Sebenarnya kedua istilah tersebut memiliki konsep yang berbeda akan tetapi berkaitan satu sama lain, dimana salah satu tahapan dalam keseluruhan proses KDD adalah *data mining*.

Dari pengertian diatas, data *mining* seharusnya dipahami sebagai suatu proses, yang memiliki tahapan-tahapan tertentu dan juga ada umpan balik dari setiap tahapan ke tahapan sebelumnya. Pada umumnya proses data *mining* berjalan interaktif karena tidak jarang hasil *data mining* pada awalnya tidak sesuai dengan harapan analisnya sehingga perlu dilakukan desain ulang prosesnya(Hand,2001).

2.2.2 Tahap-Tahap Data Mining

Sebagai suatu rangkaian proses, *data mining* dapat dibagi menjadi beberapa tahap. Tahap-tahap tersebut bersifat interaktif di mana pemakai terlibat langsung atau dengan perantaraan *knowledge base*. Berikut ini adalah tahap-tahap dilakukan dalam proses *data mining*(Pramudiono,2006).

1. Pembersihan data

Data yang memiliki isian-isian yang tidak sempurna seperti data yang hilang, data yang tidak valid atau juga hanya sekedar salah ketik bisa dikatakan sebagai data yang tidak relevan. Terdapat juga atribut-atribut data yang tidak relevan dengan hipotesis *data mining* yang kita miliki. Data-data yang tidak relevan lebih baik dibuang karena keberadaannya bisa mengurangi akurasi dari hasil *data mining*. Pembersihan data juga akan mempengaruhi kinerja dari sistem *data mining* karena data yang ditangani akan berkurang jumlah dan kompleksitasnya.

2. Integrasi data / penggabungan data

   Tidak jarang data yang diperlukan untuk *data mining* tidak hanya berasal dari satu *database* tetapi juga berasal dari beberapa *database* atau file teks. Integrasi data dilakukan pada atribut-aribut yang mengidentifikasikan entitas-entitas yang unik seperti atribut nama, jenis produk, dan nomor pelanggan. Integrasi data perlu dilakukan secara cermat karena kesalahan pada integrasi data bisa menghasilkan hasil yang menyimpang dan bahkan menyesatkan pengambilan aksi nantinya. Sebagai contoh bila integrasi data berdasarkan jenis produk ternyata menggabungkan produk dari kategori yang berbeda maka akan didapatkan

korelasi antar produk yang sebenarnya tidak ada. Dalam integrasi data ini juga perlu dilakukan transformasi dan pembersihan data karena seringkali data dari dua *database* berbeda tidak sama cara penulisannya atau bahkan data yang ada di satu *database* ternyata tidak ada di *database* lainnya.

3. Transformasi data
   Beberapa teknik *data mining* membutuhkan format data yang khusus sebelum bisa diaplikasikan. Sebagai contoh beberapa teknik standar seperti analisis asosiasi dan *clustering* hanya bisa menerima input data kategorikal. Karenanya data berupa angka numerik yang berlanjut perlu dibagi-bagi menjadi beberapa interval. Proses ini sering disebut binning.
4. Aplikasi teknik *data mining*
   Tahap ini sendiri hanya merupakan salah satu bagian dari proses *data mining*. Ada beberapa teknik *data mining* yang sudah umum dipakai diantaranya *clustering, Association Rule, Decision Tree, Memory Base Reasoning*(MBR).
5. Evaluasi pola yang ditemukan
   Dalam tahap ini hasil dari teknik *data mining* berupa pola-pola yang khas maupun model prediksi dievaluasi untuk menilai apakah hipotesis yang ada memang tercapai. Bila ternyata hasil yang diperoleh tidak sesuai hipotesis ada beberapa alternatif yang dapat diambil seperti : menjadikannya umpan balik untuk memperbaiki proses *data mining*, mencoba teknik *data mining* lain yang lebih sesuai, atau menerima hasil ini sebagai suatu hasil yang di luar dugaan yang mungkin bermanfaat.
6. Presentasi pola yang ditemukan untuk menghasilkan aksi
   Tahap terakhir dari proses *data mining* adalah bagaimana memformulasikan keputusan atau aksi dari hasil analisis yang didapat. Ada kalanya hal ini harus melibatkan orang-orang yang tidak memahami *data mining*. Karenanya presentasi hasil *data mining* dalam bentuk pengetahuan yang bisa dipahami semua orang adalah satu tahapan yang diperlukan dalam proses *data mining*. Dalam presentasi ini, visualisasi juga bisa membantu mengkomunikasikan hasil *data mining*.

2.3 Association Rule

2.3.1 Definisi

Association rule adalah teknik *data mining* untuk menemukan aturan assosiatif antara suatu kombinasi item dengan kombinasi item yang lain. Contoh dari aturan assosiatif dari analisis pembelian di suatu pasar swalayan adalah seberapa besar kemungkinan seorang pelanggan membeli roti bersamaan dengan membeli susu. Dengan pengetahuan tersebut, pemilik pasar swalayan dapat mengatur penempatan barangnya atau merancang kampanye pemasaran dengan memakai kupon diskon untuk kumpulan barang tertentu. *Association rule* juga sering disebut dengan istilah *basket analysis*.

Beberapa istilah yang sering digunakan dalam *association rule* adalah *itemset*, *support* dan *confidence*. *Itemset* adalah himpunan dari sekelompok item yang memiliki jumlah kombinasi item yang sama. *Itemset* dengan jumlah kombinasi item sebanyak k item disebut dengan k-itemset. Masing-masing item atau kumpulan item dalam k-*itemset* memiliki persentasi support diatas nilai *support* yang telah ditentukan sebelumnya atau yang biasa disebut dengan *minimum support* (minsup). *Support* ialah jumlah frekuensi kemunculan suatu item atau sejumlah item secara bersama-sama dalam satu transaksi. Sedagkan *confidence* ialah suatu nilai yang menunjukkan seberapa erat hubungan item atau sekelompok item dengan kelompok item yang lain. Jika *confidence* antar item tersebut memiliki nilai diatas nilai *minimum confidence* yang ditentukan, maka bisa dikatakan bahwa hubungan item-item tersebut tergolong sebagai *strong association rule*.

Menganalisa *associaton rule* didefinisikan sebagai suatu proses untuk menemukan semua aturan assosiatif yang memenuhi syarat minimum untuk *support* (*minimum support*) dan syarat minimum untuk *confidence* (*minimum confidence*). *Associaton rule* dikenal juga sebagai salah satu teknik *data mining* yang menjadi dasar dari berbagai teknik *data mining* lainnya. Salah satu nya ialah tahap dari analisa asosiasi itu sendiri yang disebut analisa pola frequensi tinggi (*frequent pattern mining*). Banyak pakar dibidang *data mining* yang berlomba-lomba untuk mencari algoritma yang efisien untuk mencari pola frekuensi tinggi.

2.3.2 Bentuk Dasar Aturan Assosiatif

Penting tidaknya suatu aturan assosiatif dapat diketahui dengan dua parameter, *support* dan *confidence.* Aturan assosiatif biasanya dinyatakan dalam bentuk :

{roti, mentega} -> {susu} (support = 40%, confidence = 50%)

Arti dari pernyataan diatas ialah bahwa 40% dari seluruh transaksi, memuat item roti, mentega dan juga susu. Dari seluruh transaksi yang memuat item roti dan mentega, 50 % diantaranya juga memuat item susu. Pernyataan diatas dapat juga diartikan : "Seorang konsumen yang membeli roti dan mentega punya kemungkinan 50% untuk juga membeli susu. Aturan ini cukup dikuatkan dengan adanya 40% dari seluruh jumlah transaksi, memuat ketiga item tersebut".

2.3.3 Metodologi Analisis Asosiasi

Langkah-langkah yang harus dlakukan untuk mencari aturan assosatif terbagi menjadi dua tahap(Pramudiono,2007). Tahapan tersebut ialah :

1. Analisa pola frekuensi tinggi (*Frequent Itemset*)
2. Pembentukan aturan assosiatif

2.3.3.1 Analisis Pola Frekuensi Tinggi (*Frequent Itemset*)

Tahap pertama ialah bertujuan untuk ini mencari kombinasi item yang memenuhi syarat minimum dari nilai *support* dalam daftar transaksi. Dibawah ini adalah contoh tabel transaksi pasar swalayan

Tabel 2.1 Contoh Tabel Transaksi

No transaksi	Item Terjual
1	Pena,roti,mentega
2	roti,mentega,telur,susu
3	buncis,telur,susu
4	roti,mentega
5	roti,mentega,kecap,telur,susu

Diasumsikan bahwa syarat minimum dari nilai *support* untuk pola frekuensi tinggi dalam contoh ini adalah 30%. Berikut ini adalah rumus untuk menghitung persentase support.

$$\text{Support}(\%) = \frac{\sum \text{frekuensi k-}itemset}{\sum \text{transaksi}} \times 100 \qquad (2.1)$$

(Prasetyo,2006)

Berdasarkan persamaan diatas maka dapat diketahui bahwa jumlah transaksi yang memuat {roti,mentega} ada 4 (*support* 80%), sedangkan jumlah transaksi yang memuat {roti,mentega,susu} ada 2 (*support* 40%), transaksi yang memuat {buncis} hanya 1 (*support* 20%).

Kombinasi item yang memiliki support diatas minimum support terdapat dalam tabel 2.2. Kombinasi item-item inilah yang disebut item yang memiliki pola frekuensi tinggi atau biasa disebut *frequent itemset*.

Tabel 2.2 Tabel Frequent Itemset

Kombinasi Item	Support
{roti}	80%
{mentega}	80%
{telur}	60%
{susu}	60%
{roti,mentega}	80%
{telur,susu}	60%
{roti,susu}	40%
{mentega,susu}	40%
{roti,telur}	40%
{mentega,telur}	40%
{roti,mentega,susu}	40%
{roti,mentega,telur,susu}	40%

2.3.3.2 Pembentukan Aturan Assosiatif

Setelah semua pola frekuensi tinggi ditemukan, barulah dicari aturan assosiatif yang memenuhi syarat minimum untuk *confidence.* Nilai *confidence* aturan assosiatif A->B dapat dihitung dengan persamaan 2.2.

$$Confidence(A -> B) = \frac{Support(A \cup B)}{Support(A)} \qquad (2.2)$$

(Prasetyo,2006)

Bila syarat minimum untuk *confidence* dari contoh diatas adalah 50%, maka salah satu aturan assosiatif yang dapat ditemukan adalah

{telur,susu} → {roti,mentega}

dengan nilai confidence 66.6% yang didapat dengan cara

$$\{telur,susu\} \rightarrow \{roti,mentega\} = \frac{support(\{roti,mentega,telur,susu\})}{support(\{telur,susu\})}$$

$$= \frac{40\%}{60\%} = 66.6\%$$

berdasarkan contoh kasus diatas, aturan assosiatif lain yang dapat ditemukan pada tabel 2.3.

Tabel 2.3 Tabel aturan assosiatif

Aturan Assosiatif	Support	Confidence
{telur,susu} → {roti,mentega}	40%	66.6%
{roti,mentega} → {susu}	40%	50%
{mentega,susu} →{roti}	40%	100%

Perlu dicatat bahwa tahap pertama untuk mencari pola frekuensi tinggi biasanya paling banyak menghabiskan waktu(Agrawal,1994). Oleh karena itu banyak peneliti berusaha mengembangkan algoritma yang efisien. Salah satu algoritma yang sering digunakan untuk menyelesaikan problem frekuensi tinggi ialah dengan menggunakan algoritma apriori.

2.4 Algoritma Penelusuran

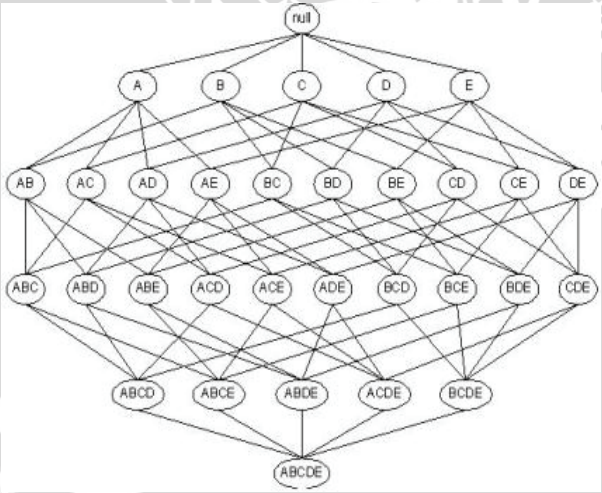
Algoritma penelusuran merupakan algoritma sederhana yang dapat diterapkan untuk pencarian pola frekuensi tingi atau *frequent itemset*(Prasetyo,2006). Langkah-langkah dalam algoritma ini adalah:

1. Mendata Item.
   Membaca data detail transaksi item dan mendapatkan jenis-jenis item yang terdaftar. Item-item tersebut disimpan pada daftar item. Data detail transkaksi merupakan data transaksi yang memuat nomor transaksi beserta item-item yang terbeli.
2. Pembentukan 1-itemset
   Menyaring item-item pada daftar item yang memiliki nilai kemunculan atau support diatas nilai minimum support yang terlah ditentukan

3. Membuat daftar kombinasi *k* item
   Mengkombinasikan item-item yang terdaftar pada daftar item. Pembentukan kombinasi ini tidak bergantung pada item-item yang terdaftar pada himpunan *k*-itemset yang telah terbentuk sebelumnya. Jumlah kombinasi yang terbentuk dapat di ketahui dengan persamaan kombinasi C_k^n, yang mana *k* adalah jumlah item dalam kombinasi dan *n* adalah jumlah item yang terdaftar dalam daftar item.

4. Pembentukan *k*-itemset ($k \geq 2$)
   Masing-masing item pada kombinasi yang terbentuk sebelumnya akan di cocokkan pada data detail transaksi. Jika semua item pada suatu kombinasi terdapat pada transaksi yang sama, maka nilai support pada kombinasi tersebut di tambahkan dengan 1. Kombinasi-kombinasi item yang memiliki support diatas nilai minimum support yang ditentukan sebelumnya akan di daftarkan pada himpunan *k*-itemset.

5. Lakukan Iterasi
   Mengulangi langkah 3 dan 4 dengan nilai $k=k+1$ hingga tidak ditemukan kombinasi pada *k*-itemset.



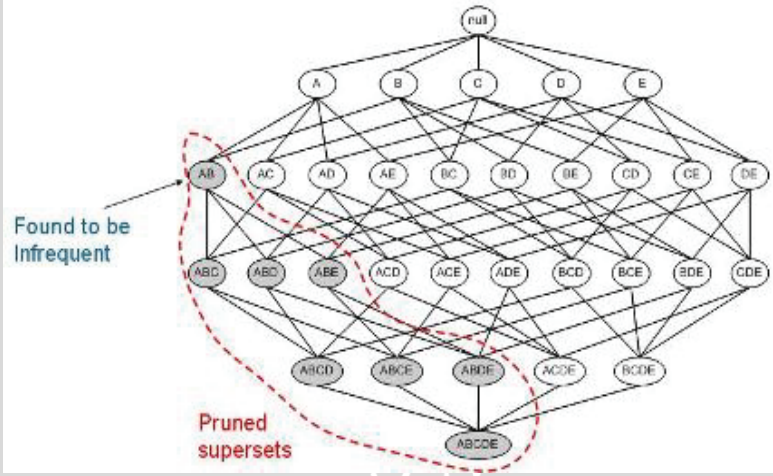
Gambar 2.1 Penggunaan algoritma penelusuran (Prasetyo,2006)

2.5 Algoritma Apriori

2.5.1 Definisi

Algoritma Apriori adalah algoritma paling terkenal untuk menemukan pola frekuensi tinggi. Pola frekuensi tinggi adalah pola-pola item di dalam suatu kumpulan data yang memiliki kemunculan atau *support* di atas minimum *support*(Prasetyo,2006). Suatu item atau kumplan item (*itemset*) yang memiliki nilai *support* lebih besar daripada minimum *support* yang ditentukan biasa disebut dengan *frequent itemset*, sebaliknya jika *support* dari item atau *itemset* memiliki nilai lebih kecil dari pada minimum *support* maka disebut *itemset infrequent*.

Apriori adalah salah satu pendekatan yang sering digunakan pada *frequent Itemset mining*. Prinsip dasar Apriori adalah jika terdapat sebuah *itemset* yang *infrequent*, maka *itemset* yang tersebut tidak perlu lagi di cari *superset*-nya sehingga jumlah kandidat yang harus diperiksa menjadi berkurang(Goethals,2000). Gambar 2.2 adalah ilustrasi dari penerapan apriori dalam mencari *frequent itemset.*



Gambar 2.2 Penggunaan algoritma Apriori
(Prasetyo,2006)

Walaupun akhir-akhir ini dikembangkan banyak algoritma yang lebih efisien dari Apriori seperti FP-growth dan LCM, tetapi algoritma apriori tetap menjadi algoritma yang paling banyak diimplementasikan dalam produk komersial untuk *data mining*

karena dianggap algoritma yang paling mapan.dan mudah dalam penerapan.

2.5.2 Proses Apriori

Secara umum, algoritma Apriori mempunyai langkah-langkah sebagai berikut :

1. Pembentukan 1-itemset
   Mendata item-item yang memiliki jumlah kemunculan pada data detail transaksi diatas minimum *support* yang telah ditentukan. Data detail transkaksi merupakan data transaksi yang memuat nomor transaksi beserta item-item yang terbeli.
2. Pembentukan kandidat k-item atau C_k ($k \geq 2$)
   Kandidat *k*-item diperoleh dengan menyusun kombinasi–kombinasi *k*-item yang item-item nya terdaftar pada data (*k*-1) itemset.
3. Memangkas kandidat(Prune step)
   Menghapus kombinasi item dalam daftar kandidat yang sudah terbentuk yang mana *subset* dari kombinasi item tersebut, tidak terdapat pada itemset sebelumnya. *Subset* ialah kombinasi (*k*-1) item dari *k*-itemset. Misal jika himpunan 3-itemset nya ialah {1,2,3}, {1,2,4}, {1,3,4}, {1,3,5}, {2,3,4} maka kandidat untuk 4-itemset hanya {1,2,3,4}. *Itemset* {1,3,4,5} tidak lolos pada tahap pemangkasan ini karena subset {1,4,5} tidak terdapat pada himpunan 3-itemset. Gambar 2.3 adalah listing untuk pemangkasan kandidat.

```
1  forall itemsets c ∈ Ck do
2     forall (k-1)-subsets s of c do
3        if (s ∉ Lk-1) then
4           delete c from Ck;
```

Gambar 2.3. Listing Prune step kandidat Apriori (Agrawal,1994)

4. Pencocokan data transaksi dengan data kandidat k-item (*Subset*)
   Pada langkah ini, item-item pada tiap transaksi disusun membentuk kombinasi *k*-item. Kombinasi-kombinasi tersebut dicocokkan dengan data kombinasi yang ada pada daftar kandidat *k*-item. Jika terdapat kombinasi yang sama, maka nilai *support* atau nilai kemunculan pada kombinasi tersebut

ditambahkan dengan 1 (*increment*). Langkah ini terus dilakukan hingga pembacaan data transaksi terakhir.

5. Pembentukan *k*-itemset
   Kombinasi-kombinasi item yang terdaftar pada himpunan *k*-itemset adalah kombinasi-kombinasi item yang terdaftar pada kandidat *k*-item dan memiliki nilai kemunculan diatas minimum support.
6. Lakukan Iterasi
   Mengulangi langkah 2 hingga 5 dengan nilai $k=k+1$ hingga tidak diketemukan kombinasi pada *k*-itemset

Bentuk listing dari langkah-langkah algoritma apriori dapat dilihat pada Gambar 2.4.

```
1   L1 = {large 1-itemsets};
2   for ( k = 2; Lk-1 ≠ Ø ; k++ ) do begin
3     Ck = apriori-gen(Lk-1); // New candidates
4       For all transactions t ∈ D do begin
5           // Candidates contained in t
6           Ct = subset(Ck , t);
7           forall candidates c ∈ Ct do
8               c.count++;
9       end
10      Lk = {c ∈ Ck | c.count ≥ minsup };
11    end
12    Answer = ⋃k Lk;
```

Gambar 2.4. Listing algoritma Apriori (Agrawal,1994)

Gambar 2.5 berikut adalah contoh dari pengguaan algoritma apriori pada daftar transaksi

Database

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

L_1

Itemset	Support
{1}	2
{2}	3
{3}	3
{5}	3

C_2

Itemset	Support
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

Itemset	Support
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

Itemset	Support
{2 3 5}	2

L_3

Itemset	Support
{2 3 5}	2

Gambar 2.5 Contoh algoritma Apriori

2.5.3 Proses AprioriTid

Algoritma AprioriTid adalah suatu algoritma yang merupakan pengembangan dari algoritma Apriori(Agrawal,1994). Secara garis besar proses yang dilakukan oleh kedua algoritma ini adalah sama, namun dalam hal pembacaan data pada daftar detail transaksi, algoritma ini melakukannya dengan efektif. Algoritma ini hanya cukup membaca satu kali data-data yang ada pada daftar detail transaksi untuk mendapatkan *k*-itemset. Hal ini berbeda dengan algoritma Apriori sebelumnya yang sangat bergantung pada data detail transaksi untuk mencari setiap kandidat *itemset.* Tahapan dalam algoritma Apriori adalah sebagai berikut :

1. Pembentukan 1-itemset
   Mendata item-item pada data detail transaksi yang memiliki jumlah kemunculan melebihi batas *support* atau minimum *support* yang telah ditentukan. Data detail transkaksi merupakan data transaksi yang memuat nomor transaksi beserta item-item yang terbeli.
2. Inisialisasi data newtrans-1 (NT_1)
   Pada tahap awal ini, data newtrans diinisialisasi dengan isian yang sama dengan data yang ada pada data transaksi.
3. Pembentukan kandidat *k*-item atau C_k ($k \geq 2$)
   Kandidat *k*-item diperoleh dengan menyusun kombinasi – kombinasi *k*-item yang item-item nya terdaftar pada data (*k*-1) -itemset.

4. Memangkas Kandidat(Prune step)
   Proses pada langkah ini tidak berbeda dengan langkah pada algoritma Apriori. *Itemset-itemset* pada kandidat yang sebelumnya telah terbentuk akan dihapus jika *subset* dari *itemset* tersebut, tidak terdapat pada himpunan *itemset* sebelumnya. *Subset* ialah kombinasi (*k*-1) item dari himpunan *k*-itemset. Misal jika himpunan 3-itemset nya ialah {1,2,3}, {1,2,4}, {1,3,4} , {1,3,5}, {2,3,4} maka kandidat untuk 4-itemset hanya {1,2,3,4}. *Itemset* {1,3,4,5} tidak lolos pada tahap pemangkasan ini karena *subset* {1,4,5} tidak terdapat pada himpunan 3-itemset. Bentuk algoritma dari pemangkasan kandidat dapat dilihat pada gambar 2.3.
5. Mengisi data newtrans-*k* (NT_k)
   Isi pada data newtrans (NT_k) didapat dengan cara membaca sekelompok itemset dari tiap transaksi yang terdaftar pada newtrans sebelumnya (NT_{k-1}). Sekelompok itemset tersebut di cocokan dengan kombinasi item yang terdaftar pada data kandidat (C_k). Pencocokan dilakukan dengan memecah tiap-tiap kombinasi yang ada pada daftar kandidat menjadi beberapa subset, misal jika kombinasi atau itemset {2 3 4} maka subsetnya ialah {2 3}, {2 4} dan {3 4}. Jika semua subset dari suatu itemset terdapat pula pada sekelompok itemset pada daftar newtrans sebelumnya (NT_{k-1}) dan semuanya memiliki no transaksi yang sama, maka itemset tersebut di ditambahkan pada data newtrans (NT_k) beserta nomor transakinya.
6. Mencatat nilai *support*
   Pada setiap penambahan item pada transaksi baru pada daftar newtrans (NT_k) maka nilai *support* pada kombinasi (C_k) yang sama dengan yang terdaftar pada newtrans tersebut ditambahkan dengan angka 1(*increment*).
7. Pembentukan *k-itemset*
   Kombinasi-kombinasi item yang terdaftar pada himpunan *k*-itemset adalah kombinasi-kombinasi item yang terdaftar pada kandidat *k*-item dan memiliki nilai kemunculan diatas minimum *support.*
8. Lakukan iterasi
   Mengulangi langkah 3 hingga 7 dengan nilai $k=k+1$ hingga tidak diketemukan kombinasi pada *k*-itemset.

Bentuk listing dari langkah-langkah algoritma apriori dapat dilihat pada Gambar 2.6.

```
1   L1 = {large 1-itemsets};
2   NT1 = database D;
3   for ( k = 2; Lk-1 ≠ Ø ; k++ ) do begin
4     Ck = apriori-gen(Lk-1); //new kandidat;
5     NTk = Ø;
6      For all entries t ∈ NT(k-1) do begin
7          //determine candidate itemsets in Ck contained
8          // in the transaction with id t.TID
9          Ct = {c ∈ Ck | (c - c[k]) ∈ t.set-of-itemsets ∧
10                        (c - c[k-1]) ∈ t.set-of-itemsets};
11
12         For all candidates c  ∈ Ct do
13              c.count++;
14         if (Ct ≠ Ø) then NTk += < t.TID,Ct >;
15     end
16     Lk = {c ∈ Ck | c.count ≥ minsup }
17  End
18  Answer = ⋃k Lk;
```

Gambar 2.6 Listing Algoritma AprioriTid (Agrawal,1994)

Database

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

NT_1

TID	Set-of-Itemsets
100	{ {1} , {3} , {4} }
200	{ {2} , {3} , {5} }
300	{ {1} , {2} , {3} , {5} }
400	{ {2} , {5} }

L_1

Itemset	Support
{1}	2
{2}	3
{3}	3
{5}	3

C_2

Itemset	Support
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

NT_2

TID	Set-of-Itemsets
100	{ {1 3} }
200	{ {2 3} , {2 5} , {3 5} }
300	{ {1 2} , {1 3} , {1 5} , {2 3} , {2 5} , {3 5} }
400	{ {2 5} }

L_2

Itemset	Support
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

Itemset	Support
{2 3 5}	2

NT_3

TID	Set-of-itemset
200	{ {2 3 5} }
300	{ {2 3 5} }

L_3

Itemset	Support
{2 3 5}	2

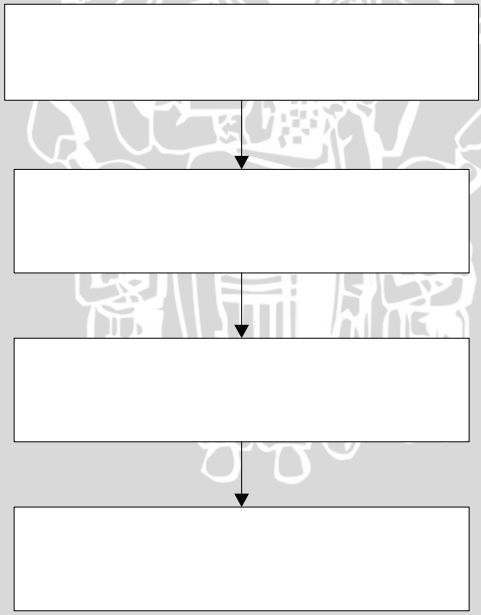
Gambar 2.7 Contoh algoritma AprioriTid


UNIVERSITAS BRAWIJAYA

BAB III
METODOLOGI DAN PERANCANGAN

Pada bab ini akan dibahas mengenai metode dan tahap-tahap yang digunakan dalam pembuatan perangkat lunak pengukur kecepatan pembentukan *Association rule.* Adapun tahapan pembuatannya adalah sebagai berikut:

1. Melakukan studi literatur mengenai algoritma penelusuran, algoritma Apriori, algoritma AprioriTid dan Association rule.
2. Membuat query untuk algoritma penelusuran, Apriori dan AprioriTid.
3. Merancang perangkat lunak untuk membentuk association rule dari tabel-tabel yang telah di bentuk oleh query pencari frequent itemset.
4. Melakukan uji coba pada data.
5. Melakukan evaluasi dan analisa hasil kecepatan waktu yang diperoleh dari uji coba perangkat lunak terhadap data.



Gambar 3.1 Flowchart pembuatan perangkat lunak

Membu
masing-m

3.1 Pola Penelitian

Pada penelitian ini dianailisis kecepatan kinerja tiga algoritma yaitu Penelusuran, Apriori dan AprioriTid dalam menemukan pola frekuensi tinggi untuk mencari *association rule*. Masing-masing algoritma akan diuji untuk mencari kombinasi item hingga 4-itemset. Pencarian hanya dilakukan hingga 4-itemset disebabkan karena data-data yang digunakan, maksimal hanya mampu membentuk 4 kombinasi item yang memiliki frekuensi kemunculan diatas minsup. Masing-masing kombinasi item akan disimpan dalam tabel pada database.

Data yang akan digunakan untuk membantu pengujian adalah data transaksi penjualan. Data tersebut diperoleh dari situs *www.fimi.cs.helsinki.fi/data/* yang filenya bernama retail.zip.

File retail.zip memiliki data yang dibutuhkan dalam mencari association rule yaitu data nomor transaksi dan data item-item yang terjual dalam transaksi.

Untuk menambah keragaman data dalam penelitian, maka data akan di bagi menjadi 4 jenis data yang memiliki beberapa parameter yang berbeda-beda. Parameter-paremeter pada data ialah jumlah transaksi, jumlah *record* transaksi, jumlah item, dan rata-rata jumlah item pada satu transaksi.

Untuk memperoleh hasil waktu yang akurat, maka masing masing algoritma akan melakukan pengujian pada tiap-tiap data sebanyak 12 kali. Hasil waktu proses pada tiap data akan diperoleh dengan cara mencari nilai rata-rata dari hasil pengujian pada data. Untuk mempermudah menganalisis hasil waktu pada tiap-tiap data, maka hasil rata-rata akan direpresentasian dalam bentuk grafik.

3.2 Jenis Data

Data yang diperoleh pada situs *www.fimi.cs.helsinki.fi/data/* merupakan data penjualan 10.000 jenis item barang yang terjual dalam 80.000 transaksi. Format file yang diperoleh adalah format file text yang ber ekstensikan .txt. Data transaksi tersebut akan dibagi menjadi 4 jenis data berdasarkan parameter-parameter yang di inginkan. Untuk memudahkan proses dalam penelitian ini maka, data text tersebut dipindahkan ke dalam tabel pada DBMS SQL Server 2000 dengan bantuan program aplikasi yang dibuat sendiri. Tabel 3.1 adalah keterangan dari empat jenis data yang akan di uji pada penelitian ini.

Tabel 3.1 Tabel Data

Nama Data	Jumlah Transaksi	Jumlah Item	Jumlah Record	Rata-rata jml item
T10I5V5	10.000	5000	49042	5
T10I5V8	10.000	5000	79853	8
T30I8V5	30.000	8000	153348	5
T30I8V8	30.000	8000	237675	8

3.3 Perancangan Tabel

Penelitian ini memerlukan suatu tabel untuk menampung data-data transaksi penjualan yang didapat dari mem-*parsing* file text. Sehingga diperlukan suatu tabel yang terdiri dari 2 *field*. Tabel inilah yang nantinya menjadi tabel pokok atau tabel dasar dari tabel-tabel yang lain. Tabel ini disebut tabel pokok karena data-data pada tabel yang lain merupakan hasil dari pembacaan data pada tabel 3.2.

Tabel 3.2 Tabel tb_trans

Field	Tipe	Keterangan
No_trans	Varchar(5)	*Primary Key*
Kode_item	Int(4)	*Primary Key*

Keterangan
No_trans : No transaksi penjualan
kode_item : Kode jenis barang

Tabel 3.2 akan menampung puluhan hingga ratusan ribu *record*, karena tiap baris transaksi yang terdapat dari file text dapat menghasilkan puluhan *record* pada tabel ini. Field No_trans berisi nomor urut penjulan yang dimulai dengan no 1.

Selain tabel tb_trans, penelitian ini juga membutuhkan beberapa tabel yang digunakan untuk menampung data hasil pembacaan data pada tabel tb_trans. Keberadaan tabel-tabel 3.7 hingga 3.12 tidak-lah harus ada, melainkan bisa digantikan dengan tabel virtual. Namun penggunaan tabel virtual dapat memperlambat proses pembacaan data, karena pemakaian tabel virtual membutuhkan memori yang besar pada saat proses pengujian. Berdasarkan fungsinya, tabel-tabel tersebut di bagi menjadi 3 jenis.

1. Tabel Itemset

Tabel ini merupakan tabel yang menyimpan data-data kombinasi item dari pencarian tiap-tiap *itemset*. Bentuk dari tabel-tabel ini disebut tabel multi kolom karena tiap-tiap item ditempatkan pada field yang berbeda-beda. Model tabel multi kolom merupakan model tabel yang paling efisien untuk mencari *frequent itemset* daripada menggunakan model tabel *single* kolom(Rajamani,2001). Tabel inilah yang nantinya akan langsung digunakan untuk mencari aturan assosiatif. Sesuai dengan jumlah *itemset* yang dicari pada penelitian ini, maka tabel ini terdiri dari 4 tabel.

Tabel 3.3 Tabel tb_itemset1

Field	Tipe	Default	Keterangan
Kode_item	Int(4)	-	*Primary Key*
Support	Int(4)	0	

Tabel 3.4 Tabel tb_itemset2

Field	Tipe	Default	Keterangan
Kode1	Int(4)	-	*Primary Key*
Kode2	Int(4)	-	*Primary Key*
Support	Int(4)	0	

Tabel 3.5 Tabel tb_itemset3

Field	Tipe	Default	Keterangan
Kode1	Int(4)		*Primary Key*
Kode2	Int(4)		*Primary Key*
Kode3	Int(4)		*Primary Key*
Support	Int(4)	0	

Tabel 3.6 Tabel tb_itemset4

Field	Tipe	Default	Keterangan
Kode1	Int(4)		*Primary Key*
Kode2	Int(4)		*Primary Key*
Kode3	Int(4)		*Primary Key*
Kode4	Int(4)		*Primary Key*
Support	Int(4)	0	

Keterangan
kode_item : kode jenis barang
Kode1 : kode item barang kombinasi pertama
Kode2 : kode item barang kombinasi ke-2
Kode3 : kode item barang kombinasi ke-3
Kode4 : kode item barang kombinasi ke-4
Support : banyaknya kemunculan suatu kombinasi

2. Tabel Kandidat

Tabel kandidat merupakan tabel yang menyimpan kombinasi-kombinasi item yang yang akan dicari nilai *support*-nya. Tabel ini bersifat optional karena keberadaanya dapat menyederhanakan dan mempercepat pencarian *itemset*.

Tabel 3.7 Tabel tb_kandidat2

Field	Tipe	Keterangan
Kode1	Int(4)	*Primary Key*
Kode2	Int(4)	*Primary Key*

Tabel 3.8 Tabel tb_kandidat3

Field	Tipe	Keterangan
Kode1	Int(4)	*Primary Key*
Kode2	Int(4)	*Primary Key*
Kode3	Int(4)	*Primary Key*

Tabel 3.9 Tabel tb_kandidat4

Field	Tipe	Keterangan
Kode1	Int(4)	*Primary Key*
Kode2	Int(4)	*Primary Key*
Kode3	Int(4)	*Primary Key*
Kode4	Int(4)	*Primary Key*

Keterangan
Kode1 : Kode item barang kombinasi pertama
Kode2 : Kode item barang kombinasi ke-2
Kode3 : Kode item barang kombinasi ke-3
Kode4 : Kode item barang kombinasi ke-4

3. Tabel Newtrans

Tabel Newtrans digunakan oleh algoritma AprioriTid untuk menyimpan suatu daftar transaksi. Daftar transaksi yang disimpan ialah daftar transaksi yang item-itemya tergolong dalam item yang memiliki *support* di atas minimum *support*. Dengan adanya tabel-tabel ini, diharapkan banyaknya *record* yang perlu dibaca untuk mendapatkan *itemset* berikutnya menjadi berkurang. Tabel ini nantinya akan menggantikan peran tabel tb_trans dalam menyediakan data.

Tabel 3.10 Tabel tb_newtrans2

Field	Tipe	Keterangan
No_trans	Varchar(5)	*Primary Key*
Kode1	Int(4)	*Primary Key*
Kode2	Int(4)	*Primary Key*

Tabel 3.11 Tabel tb_newtrans3

Field	Tipe	Keterangan
No_trans	Varchar(5)	*Primary Key*
Kode1	Int(4)	*Primary Key*
Kode2	Int(4)	*Primary Key*
Kode3	Int(4)	*Primary Key*

Tabel 3.12 Tabel tb_newtrans4

Field	Tipe	Keterangan
No_trans	Varchar(5)	*Primary Key*
Kode1	Int(4)	*Primary Key*
Kode2	Int(4)	*Primary Key*
Kode3	Int(4)	*Primary Key*
Kode4	Int(4)	*Primary Key*

Keterangan

No_trans : nomor transaksi penjualan
Kode1 : kode item barang kombinasi pertama
Kode2 : kode item barang kombinasi ke-2
Kode3 : kode item barang kombinasi ke-3
Kode4 : kode item barang kombinasi ke-4

Untuk mempermudah dalam membuat aplikasinya, maka isian pada kolom kode1, kode2, kode3 dan kode4 pada tabel-tabel 3.3 hingga 3.12 memiliki beberapa aturan. Aturan-aturan tersebut diantaranya :

1. Nilai kode1 hingga kode4 tidak boleh ada kesamaan antara satu dengan yang lain.
2. Nilai kode diurutkan secara *lexicographic*, artinya bahwa nlai kode1 < kode2 dan kode2 < kode3 dan seterusnya.

3.4 Peralatan Penelitian

Perangkat lunak yang digunakan pada penelitian ini ialah :

1. Sistem operasi Microsoft Windows XP Professional Edition
2. Borland Delphi 6.0 sebagai software development dalam mengembangkan aplikasi untuk menguji waktu proses ketiga metode untuk menemukan Association Rule.
3. Microsoft SQL Server 2000 Enterprise Edition sebagai DBMS *(Database Managemen System)*.

Perangkat keras yang digunakan pada penelitian ini adalah sebuah *PC(Personal Computer)* dengan spesifikasi sebagai berikut :

1. Processor Pentium 4 1.5 GHz
2. 384 MB RAM
3. 80 GB 7200 rpm HDD

3.5 Langkah Penelitian

Seperti yang telah dijelaskan, bahwa penelitian ini akan mengkaji 3 jenis algoritma untuk mencari pola frekuensi tinggi atau *frequent itemset* hingga diketemukan association rule. Berikut adalah langkah-langkah yang dilakukan dalam penelitian ini.

1. Mencari *association rule* dengan masing-masing algoritma pada salah satu jenis data dengan minsup 0,2 %.
2. Mengulang langkah pertama dengan nilai minsup mulai dari 0,4% hingga 2 % dengan pertambahab 0,2%
3. Mencatat waktu proses untuk tiap-tiap minsup pada masing-masing algoritma.
4. Melakukan langkah 1 sampai 3 sebanyak 12 kali.
5. Mencari nilai rata-rata dari data waktu yang terkumpul dan membuat grafik.
6. Melakukan langkah 1 sampai 5 untuk 3 jenis data yang lain

3.6 Penerapan Teknik Penelusuran

Pencarian pola frekuensi tinggi dengan algoritma penelusuran sangat bergantung pada tabel tb_trans yang merupakan tabel pokok. Langkah pertama untuk pencarian association pada algoritma ini ialah mencari nilai *support* pada masing-masing jenis item yang terdaftar pada tabel tb_trans. Nilai *support* diperoleh dengan cara menghitung seberapa banyak kemunculan jenis item tersebut pada tabel tb_trans. Jenis item yang memiliki nilai *support* lebih besar dari pada nilai minimum *suppport* akan disimpan pada tabel tb_itemset1.

Tabel tb_itemset1 ini nantinya tidak akan berperan dalam mencari *itemset* selanjutnya. Untuk pencarian 2-itemset dan seterusnya, algoritma ini membutuhkan daftar kombinasi item-item yang terdaftar pada tabel tb_trans. Pencarian semacam ini akan dilakukan secara terus menerus hingga mencapai *itemset* yang di inginkan.

Banyaknya kombinasi item yang diperoleh pada tiap-tiap pencarian *itemset* dapat diketahui tanpa mengetahui item-item pada itemset sebelumnya. Persamaan 3.1 adalah rumus kombinasi yang dapat digunakan untuk mengetahui jumlah kombinasi.

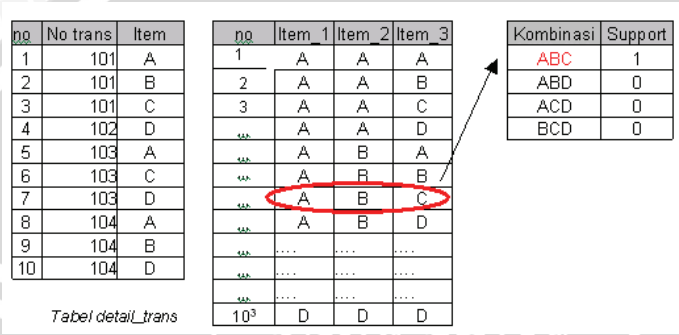
$$\text{Jumlah kombinasi} = C_k^n \qquad (3.1)$$

Variabel n merupakan jumlah jenis item yang terdapat pada tabel tb_trans. Sedangkan nilai k ialah jumlah item dalam kombinasi dari *itemset* yang ingin dicari. Misal, jika diketahui jml item = 50 dan jumlah item dalam kombinasi ialah 2 maka jumlah kombinasi yang diperoleh ialah

$$C_2^{50} = \frac{50!}{(50-2)!\ 2!} = \frac{50 \times 49 \times 48!}{48! \times 2!} = \frac{2450}{2} = 1225$$

Untuk mengetahui besarnya support masing-masing kombinasi pada tabel *k*-itemset, maka langkah berikutnya ialah mecocokkan masing-masing kombinasi tersebut kedalam gabungan tabel *k* tb_trans (*k*>1). Pencocokkan ini dilakukan dengan mambaca per *record* pada *record* gabungan transaksi.

Pada pembacaan *record* gabungan tersebut, jika ada kombinasi yang sama dan masing-masing item dalam kombinasi tersebut memiliki no transaksi atau no penjualan yang sama, maka nilai *support* pada kombinasi tersebut akan ditambahkan 1. Kombinasi *k*-item akan disimpan dalam tabel *k*-itemset jika nilai *support*-nya lebih dari nilai minimum *support* yang telah ditetapkan sebelumnya.



Gambar 3.2 Contoh penambahan nilai *Support*

Pada contoh gamabr 3.1 tampak bahwa pada pembacaan *record* gabungan item mencapai kombinasi ACA, hanya kombinasi item ABC saja yang *support*-nya ditambah 1, sedangkan kombinasi ABD tidak. Itu disebabkan karena item {A,B} memiliki No_trans yang berbeda dengan item {D}.

3.8 Penerapan Algoritma Apriori

Langkah pertama dalam mencari pola frekuensi tinggi atau *frequent itemet* dengan algoritma Apriori selalu diawali dengan mencari himpunan 1-itemset. Langkah ini dimulai dari mencari besarnya frekuensi penjualan masing-masing item berdasarkan tabel tb_trans. Untuk mendapatkan daftar 1-itemset langkah-langkahnya tidak berbeda dengan algoritma penelusuran, yaitu dengan memilih item-item yang yang memiliki *support* di atas minimum *support*.

Langkah berikutnya ialah melakukan pencarian kandidat *k*-itemset ($k \geq 2$). Pencarian ini dilakukan dengan cara mencari kombinasi dari item-item yang terdaftar dalam tabel (*k*-1)-itemset. Hasil kombinasi tersebut akan disimpan dalam tabel kandidat. Gambar 3.2 adalah bentuk query dari pengisian tabel kandidat (tb_kandidat).

```
1  Insert  into kandidat
2  Select  p.item1,p.item2,...,p.itemk-1,q.itemk-1
3  From    Lk-1 as p, Lk-1 as q
4  Where   p.item1 = q.item1,...p.itemk-2=q.itemk-2,
5          p.itemk-1<q.itemk-1
```

Gambar 3.3 Query pengisian tabel kandidat

Banyaknya jumlah kombinasi yang didapatkan ialah sebesar $C_k^{n(k-1)}$ yang mana *n(k-1)* adalah jumlah item pada tabel (*k*-1)-*itemset* dan *k* adalah banyaknya item pada suatu kombinasi. Misalnya jika jumlah item yang di jual pada super market ialah 50 item, sedangkan jumlah item yang terdaftar pada tabel 1-itemset ialah 30 maka jumlah kombinasi pada tabel kandidat 2-itemset ialah

$$C_2^{30} = \frac{30!}{(30-2)!\,2!}$$
$$= \frac{30 \times 39 \times 38!}{38! \times 2!}$$
$$= \frac{870}{2} = 435$$

Untuk meminimalkan proses pencarian support pada kandidat yang telah terbentuk, maka perlu dilakukan pemangkasan untuk kandidat yang subset nya tidak ada pada daftar itemset sebelumnya ((*k*-1)-itemset). Pemangkasan ini berlaku hanya untuk *k*-itemset dimana $k \geq 3$, karena dapat dipastikan bahwa kandidat-kandidat pada pencarian 2-itemset semua subset-nya terdaftar pada tabel 1-itemset.

```
1  delete from k-kandidat
2  where subset not in
3  (select kode1,kode2,..kodek-1 from (k-1)-itemset)
```

Gambar 3.4 Query pemangkasan kandidat

Sama seperti penerapan algoritma penelusuran bahwa untuk mengetahui frekuensi kemunculan masing-masing kombinasi pada kandidat *k*-itemset, maka diperlukan penggabungan *k* tabel tb_trans. Penggabungan kedua tabel tb_trans tersebut akan menghasilkan sebuah tabel yang memiliki jumlah *record* sama dengan

record_size^k yang mana *record_size* ialah banyaknya *record* pada tabel tb_trans dan *k* ialah jumlah tabel tb_trans yang digabungkan.

Frekuensi kemunculan atau *support* dari suatu *k*-kombinasi didapatkan dengan cara mencocokkan masing-masing kombinasi yang tersimpan pada tabel kandidat dengan kombinasi *k* item yang ada pada gabungan *k* tabel tb_trans. Jika kombinasi tersebut ada yang sesuai atau cocok dan *k* item tersebut memiliki nomor transaksi yang sama, maka nilai *support* pada kombinasi tersebut dijumlahkan dengan satu. kombinasi item yang memiliki nilai *support* di atas minimum *support* akan disimpan dalam tabel *k*-itemset dan item-item tersebut akan di ikut sertakan dalam pencarian *itemset* berikutnya

Tabel tb_trans

no	No trans	Item
1	101	A
2	101	B
3	101	C
4	102	E
5	102	F
6	103	A
7	103	B
8	103	C
9	103	F
10	104	A
11	104	B
12	104	F
13	105	B
14	105	C
15	106	A
16	106	C
17	106	D

Minsup=2

Tabel tb_itemset1

Item	support
A	4
B	4
C	4
F	3

Tabel tb_kandidat2

Item1	item2
A	B
A	C
A	F
B	C
B	F
C	F

Tabel tb_itemset2

Item1	item2	support
A	B	3
A	C	3
B	C	3

Gambar 3.5 Penerapan algoritma Apriori

3.8 Penerapan Algoritma AprioriTid

Algoritma AprioriTid adalah pengembangan dari algoritma Apriori. Secara garis besar kedua algoritma ini adalah sama, namum sedikit berbeda dalam hal pembacaan *record* pada tabel tb_trans. Algoritma AprioriTid hanya cukup membaca satu kali *record-record* yang ada pada tabel tb_trans untuk mendapatkan *k*-itemset. Hal ini berbeda dengan algoritma Apriori sebelumnya yang sangat bergantung pada tabel tb_trans untuk mencari setiap *itemset*.

Pada langkah pertama diawali dengan mencari 1-itemset yang caranya sama dengan algoritma Apriori. Agar tidak bergantung pada tabel tb_trans, maka untuk mencari tiap *itemset* algoritma ini membutuhkan tabel tb_newtrans. Tabel ini digunakan untuk menggantikan tabel tb_trans, yang isinya merupakan transaksi penjualan item-item yang kodenya terdaftar pada kandidat *itemset*. Untuk isian tabel newtrans yang pertama, isi *record*-nya sama dengan tabel tb_trans.

Berikutnya ialah membuat daftar kandidat *k*-itemset ($k \geq 2$) yang mana itemnya hanya item yang terdaftar pada tabel (k-1)-itemset. Kombinasi *k* item tersebut disimpan dalam tabel kandidat. Seperti pada Apriori, daftar kandidat *k*-itemset ($k \geq 3$) pada algoritma ini perlu juga untuk dilakukan pemangkasan guna meminimalkan kandidat.

Isi pada tabel transaksi baru atau tb_newtrans diperoleh dari tabel transaksi sebelumnya yang item-itemnya terdaftar pada tabel kandidat. Apabila ada *k* transaksi item pada tabel transaksi sebelumnya yang memiliki nomor transaksi yang sama dan item-item tersebut terdaftar dalam tabel kandidat, maka item-item tersebut beserta nomor transaksinya akan di *input*-kan pada tabel tb_newtrans dalam satu *record* $\{no\_trans, item_1, item_2, ... item_n\}$. Gmabr 3.5 adalah contoh bentuk query untuk mengisi tabel tb_newtrans2.

```
1  Insert into tb_newtrans2
2  Select Trans1.no_trans,kandidat.kode1,
3         kandidat.kode2
4  From   tb_trans as Trans1,tb_trans as Trans2,
5         tb_kandidat  as kandidat
6  Where  Trans1.no_trans  = Trans2.no_trans,
7         Trans1.kode_item = kandidat.kode1,
8         Trnas2.kode_item = kandidat.kode2
```

Gambar 3.6 Query pengisian tabel tb_newtrans2

Untuk mencari *k*-itemset ($k \geq 2$) maka algoritma ini hanya membaca *record-record* yang ada pada tabel transaksi baru atau tabel tb_newtrans. Hal inilah yang membedakan algoritma Apriori dengan AprioriTid karena algoritma ini tidak lagi menggunakan tabel tb_trans dalam mencari *itemset*.

Jumlah *record* yang dibaca pada tabel tb_newtrans tidak selalu lebih sedikit dari pada transaksi sebelumnya. Hal ini dipengaruhi oleh banyaknya item yang terdaftar pada (*k*-1)-itemset. Makin banyak item-item yang terdaftar pada tabel (*k*-1)-itemset, maka ada kemungkinan tabel transaksi yang baru memiliki jumlah *record* yang lebih banyak dari tabel transaksi sebelumya. Itu sebabnya algoritma AprioriTid tidak selalu lebih unggul dari pada algoritma pendahulunya (algoritma Apriori) (Agrawal,1994).

Tabel tb_trans

no	No trans	Item
1	101	A
2	101	B
3	101	C
4	102	E
5	102	F
6	103	A
7	103	B
8	103	C
9	103	F
10	104	A
11	104	B
12	104	F
13	105	B

Minsup=2

Tabel tb_itemset1

item	support
A	5
B	4
C	3
F	4

Tabel tb_kandidat2

item1	item2
A	B
A	C
A	F
B	C
B	F
C	F

14	105	C
15	106	A
16	106	C
17	106	D
18	107	A
19	108	D
20	108	F

Tabel tb_newtrans2

no	No trans	item1	item2
1	101	A	B
2	101	A	C
3	101	B	C
4	103	A	B
5	103	A	C
6	103	A	F
7	103	B	C
8	103	B	F
9	103	C	F
10	104	A	B
11	104	A	F
12	104	B	F
13	105	B	C
14	106	A	C

Tabel tb_itemset2

item1	item2	support
A	B	3
A	C	2
A	F	3
B	C	3
B	F	2

Tabel tb_kandidat3

item1	item2	item3
A	B	C
A	B	F
A	C	F
B	C	F

Tabel tb_newtrans3

no	No trans	item1	item2	item3
1	101	A	B	C
2	103	A	B	C
3	103	A	B	F
4	103	A	C	F
5	103	B	C	F
6	104	A	B	F

Tabel tb_itemset3

Item1	item2	item3	support
A	B	C	2
A	B	F	2

Gambar 3.7 Penerapan Algoritma AprioriTid


UNIVERSITAS BRAWIJAYA

BAB IV
HASIL DAN PEMBAHASAN

4.1 Implementasi

4.1.1 Implementasi Basis Data

Tabel-tabel multi kolom yang digunakan pada penelitian ini, dibuat hanya sekali yaitu tabel tb_itemset1, tb_itemset2, tb_itemset3, tb_itemset4, tb_kandidat2, tb_kandidat3, tb_kandidat4, tb_newtrans2, tb_newtrans3, tb_newtrans4. Pada saat proses berjalan tidak ada pembuatan tabel lagi. Hal ini dimaksudkan untuk meminimalkan waktu proses.

4.1.2 Implementasi Query

Masing-masing algoritma pencarian *frequent itemset*, yaitu algoritma penelusuran, Apriori dan AprioriTid dituliskan dalam bentuk bahasa *query* dan disimpan dalam *strored procedure* yang berbeda. *Stored procedure* sp_penelusuran untuk algoritma penelusuran. *Stored procedure* sp_apriori untuk algoritma apriori. *Stored procedure* sp_aprioritid untuk algoritma aprioriTid. Pengimplementasian *query* dalam bentuk *stored procedure* ini bertujuan untuk memperoleh waktu komputasi yang seminimal mungkin karena pencarian *frequent itemset* diserahkan sepenuhnya pada DBMS yang mana dalam hal ini ialah SQL Server 2000.

```
1   create procedure sp_penelusuran(@minsup int) as
2   ------Mengisi tabel 2-itemset
3   insert into tb_itemset1
4   select kode_item,count(*)
5   from tb_trans
6   group by kode_item
7   having count(*)> @minsup
8   ------Mengisi tabel 2-itemset
9   insert into tb_itemset2
10  select Trans1.kode_item, Trans2.kode_item, count(*)
11  from tb_trans as Trans1, tb_trans as Trans2
12  where Trans1.no_trans  = Trans2.no_trans and
13        Trans1.kode_item <  Trans2.kode_item
14  group by Trans1.kode_item, Trans2.kode_item
15  having count(*) > @minsup
    .....
```

Gambar 4.1 *Stored procedure* sp_penelusuran

```
1   create procedure sp_apriori(@minsup int) as
2   ------Mengisi tabel 1-itemset
3   insert into tb_itemset1
4   select kode_item, count(*)
5   from tb_trans
6   group by kode_item
7   having count(*) > @minsup
8
9   ------Mengisi tabel Kandidat 2
10  insert into tb_kandidat2
11  select Item1.kode_item, Item2.kode_item
12  from tb_itemset1 as Item1, tb_itemset1 as Item2
13  where Item1.kode_item < Item2.kode_item
14
15  ------Mengisi tabel 2-itemset
16  insert into tb_itemset2
17  select Trans1.kode_item, Trans2.kode_item, count(*)
18  from tb_trans as Trans1, tb_trans as Trans2,
19       tb_kandidat2 as Candidat
20  where Trans1.no_trans = Trans2.no_trans and
21        Candidat.kode1  = Trans1.kode_item and
22        Candidat.kode2  = Trans2.kode_item
23  group by Trans1.kode_item, Trans2.kode_item
24  having count(*)> @minsup
    .....
```

Gambar 4.2 *Stored procedure* sp_apriori

```
1   create procedure sp_apriori_tid(@minsup int) as
2   ------Mengisi tabel 1-itemset
3   insert into tb_itemset1
4   select kode_item,count(*) from tb_trans
5   group by kode_item
6   having count(*)> @minsup
7
8   ------Mengisi tabel Kandidat 2
9   insert into tb_kandidat2
10  select Itemset1.kode_item, Itemset2.kode_item
11  from tb_itemset1 as Itemset1,tb_itemset1 as Itemset2
12  where Itemset1.kode_item < Itemset2.kode_item
13
14  ------Mengisi tabel Transaksi baru 2
15  insert into tb_newtrans2
16  select rans1.no_trans,Trans1.kode_item,
17         Trans2.kode_item
18  from tb_trans as trans1,tb_trans as trans2,
19       tb_kandidat2 as Candidat
20  where Trans1.no_trans = Trans2.no_trans and
21       Candidat.kode1 = Trans1.kode_item and
```

```
22          Candidat.kode2 = Trans2.kode_item
23      group by Trans1.no_trans, Trans1.kode_item,
24              Trans2.kode_item
25
26      ------Mengisi tabel 2-itemset
27      insert into tb_itemset2
28      select kode1,kode2,count(*)
29      from tb_newtrans2
30      group by kode1,kode2
31      having count(*) > @minsup
        .....
```

Gambar 4.3 *Stored procedure* sp_apriori_tid

Masing-masing *stored procedure* pada gambar 4.1, 4.2, dan 4.3 melakukan pencarian *frequent itemset* dari 1-itemset hingga 4-itemset secara berurutan sesuai dengan algoritma yang digunakan. Listing lengkap stored procedure sp_penelusuran, sp_apriori dan sp_apriori_tid ada pada lampiran 3.

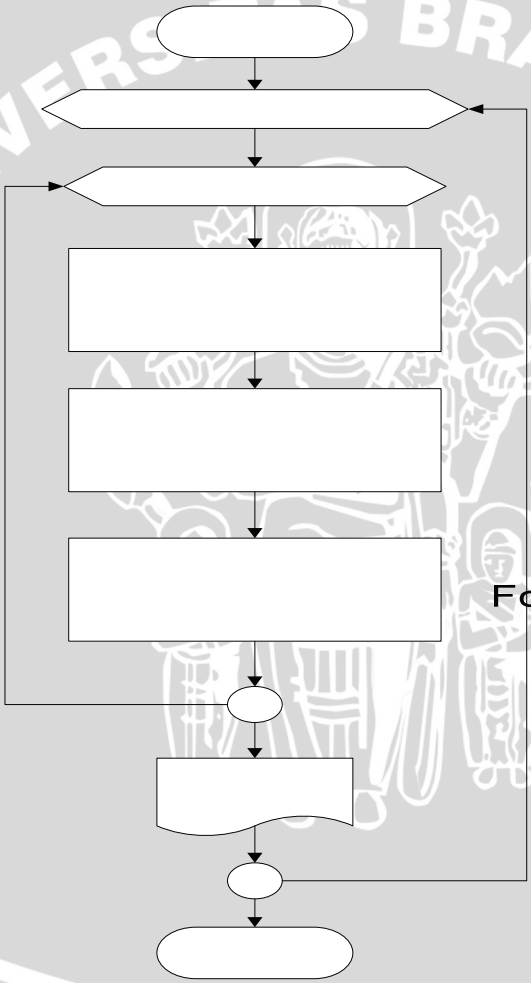
Selain *stored procedure* sp_penelusuran, sp_apriori dan sp_apriori_tid, diperlukan sebuah *stored procedure* sp_delete_table. *Stored procedure* ini digunakan untuk menghapus seluruh record yang ada pada semua tabel kecuali tabel tb_trans. *Stored procedure* sp_delete_table di eksekusi sebelum mengeksekusi salah satu dari ketiga *stored procedure* diatas. Isi dari *stored procedure* sp_delete_table dapat dilihat pada gambar 4.4.

```
1   create procedure sp_delete_table as
2   -- Menghapus tabel itemset
3   delete from tb_itemset4
4   delete from tb_itemset3
5   delete from tb_itemset2
6   delete from tb_itemset1
7
8   -- Menghapus tabel kandidat
9   delete from tb_kandidat4
10  delete from tb_kandidat3
11  delete from tb_kandidat2
12
13  -- Menghapus tabel newtrans
14  delete from tb_newtrans4
15  delete from tb_newtrans3
16  delete from tb_newtrans2
```

Gambar 4.4 *Stored procedure* sp_delete_table

4.1.3 Implementasi Program

Secara umum program ini memiliki tiga prosedur yang mengeksekusi tiga *stored procedure* pencari *frequent itemset* dan satu prosedur untuk mencari *association rule*. Prosedur untuk masing- masing algoritma dijalankan secara berurutan yang dimulai dari prosedur penelusuran, prosedur apriori dan prosedur aprioritid.



Start

For A=1 to banya

For i=1 to banya

Timer=
exec sp_dele
Timer
Exec sp_penelusu
proc assocation_r
time[1][i]=

Timer=

Gambar 4.5 *Flowchart* aplikasi

```
1   for i:=1 to nsup do
2         begin
3               penelusuran(minsup[i]);
4               apriori(minsup[i]);
5               aprioritid(minsup[i]);
6         end;
```

Gambar 4.6 Sintak pemanggilan prosedur untuk masing-masing algoritma

4.1.3.1 Prosedur Pencari Frequent Itemset

Proses pada masing-masing prosedur pencari *frequent itemset* memiliki struktur yang hampir sama, namun ketiga prosedur tersebut mengakses *stored procedure* yang berbeda. Prosedur Penelusuran akan mengeksekusi *stored procedure* sp_penelusuran, prosedur apriori akan mengeksekusi *stored procedure* sp_apriori, dan prosedur aprioritid akan mengeksekusi *stored procedure* sp_apriori_tid. Karena tabel-tabel yang digunakan oleh ketiga *stored procedure* adalah sama, maka tiap-tiap prosedur akan mereset isian tabel terlebih dahulu dengan mengakses *stored procedure* sp_delete_table. Sebagai contoh, prosedure penelusuran dituliskan pada gambar 4.7

Waktu yang diperlukan untuk mencari 4-itemset hingga diketemukan *association rule* dicatat pada variabel selisih. Dari variabel selisih nantinya akan di peroleh nilai waktu dalam satuan detik yang disimpan dalam variabel array dua dimensi yang bernama hasil. Pencatatan waktu tersebut dapat dilihat pada gambar 4.7 baris ke 12 hingga 16.

```
1   procedure TFmain.penelusuran(minsup:integer);
2   var starttime,selisih : TDateTime;
3       Hour,Min,Sec,MSec : Word;
4   Begin
5   //--mereset tabel
6   sp_delete.ExecProc;
7
8   sp_penelusuran.Parameters[1].Value:=minsup;
9   starttime:=now;
10  sp_penelusuran.ExecProc;
11  association();
12  selisih:=Now-starttime;
13  DecodeTime(selisih,Hour,min,sec,Msec);
14  if Min<>0 then
15     Sec:=Sec+(Min*60);
16  hasil[x,1]:=format('%0d.%0d',[Sec,Msec]);
17  end;
```

Gambar 4.7 Sintaks prosedur penelusuran

4.1.3.2 Prosedur Association Rule

Prosedur ini berfungsi untuk menghasilkan *association rule* dari 4 kombinasi item yang tersimpan dalam tabel tb_itemset4. Langkah pertama ialah memperoleh item-item yang yang terdapat pada tabel tb_itemset4 beserta nilai *support*-nya. Item tersebut disimpan dalam variable array kode[1] untuk menyimpan item pertama, kode[2] untuk menyimpan kode kedua, kode[3] untuk menyimpan kode ketiga dan kode[4] untuk menyimpan kode keempat. Nilai *support* untuk 4 kombinasi item tersebut disimpan dalam variable support.

Langkah berikutnya ialah mendapatkan kombinasi-kombinasi dari item yang terdapat pada variabel array kode[] beserta nilai *support*-nya. Kombinasi tersebut dimulai dari 4 kombinasi 1 hingga 4 kombinasi 3. Kombinasi-kombinasi tersebut beserta *support*-nya disimpan dalam variable array kombinasi[] dan variabel array sup[]. Perintah pencarian kombinasi beserta *support*-nya dapat dilihat pada gambar 4.9 baris ke 15 hingga 52.

Setelah kombinasi dan nilai *support*-nya didapatkan maka langkah berikutnya ialah mencari nilai *confidence* dari pasangan kombinasi tersebut. Caranya ialah dengan memasangkan kombinasi yang berisi satu item dengan kombinasi yang berisi tiga item, kemudian memasangkan kombinasi yang berisi 2 item dengan kombinasi 2 item yang lain, dan yag terakhir ialah memasangkan kombinasi yang berisi tiga item dengan kombinasi yang berisi satu item. Pemasagan tersebut dilakukan secara berurutan dengan aturan bahwa tidak boleh ada item yang sama dalam pasangan kombinasi tersebut. Gambar 4.8 adalah contoh perolehan kombinasi dari item {A,B,C,D} beserta pemasangan kombinasinya.

4-itemset : {A,B,C,D}
Kode[1] =A Kode[3] = C
Kode[2] =B Kode[4] = D

Jenis Kombinasi :

1. Kombinasi[1] = A	8. Kombinasi[8] = BC
2. Kombinasi[2] = B	9. Kombinasi[9] = BD
3. Kombinasi[3] = C	10. Kombinasi[10] = CD
4. Kombinasi[4] = D	11. Kombinasi[11] = ABC
5. Kombinasi[5] = AB	12. Kombinasi[12] = ABD
6. Kombinasi[6] = AC	13. Kombinasi[13] = ACD
7. Kombinasi[7] = AD	14. Kombinasi[14] = BCD

Pemasangan kombinasi			
1. A → BCD 2. B → ACD 3. C → ABD 4. D → BCD	5. AB → CD 6. AC → BD 7. AD → BC	8. BC → AD 9. BD → AC 10.CD → AB	11. ABC → D 12. ABD → C 13. ACD → B 14. BCD → A

Gambar 4.8 Kombinasi dari 4 jenis item

Setelah pasangan kombinasi didapatkan, maka langkah selanjutnya ialah menghitung nilai *confidence* dari masing-masing pasangan kombinasi. Pasangan kombinasi pada gambar 4.8 diatas tampak bahwa kombinasi[1] dipasangkan dengan kombinasi[14], kombinasi[2] dipasangkan dengan kombinasi[13] dan kombinasi[*n*] dipasangkan dengan kombinasi[15-*n*].

Berdasarkan persamaan (2.2) pada bab II, nilai *confidence* pada pasangan kombinasi ke-*n* dapat diperoleh dengan membagi suport{A,B,C,D} dengan sup[*n*]. Penerapan persamaan (2.1) dapat dilihat pada gambar 4.9 baris ke 58.

Pasangan kombinasi yang memenuhi aturan *association rule* ialah pasangan yang memiliki nilai *confidence* diatas nilai minimum *confidence* yang telah ditentukan.

```
1   procedure TFmain.association;
2   var query,query1,query2,query3:string;
3       . . .
4   begin
5    query:='select * from tb_itemset4';
6    ADOquery1.SQL.Text:=query;
7    ADOQuery1.Active:=true;
8    ADOQuery1.First;
9    while ADOQuery1.Eof=false do
10   begin
11    kode[1]:=ADOQuery1.FieldByName('kode1').AsString;
12    . . .
13    suport:=ADOQuery1.FieldByName('support').AsFloat;
14    //---itemset 1
15    query1:='select kode_item,support from tb_itemset1
16             where kode_item in ';
17    query1:=query1+'('+kode[1]+','+kode[2]+','+kode[3]+
18             ','+kode[4]+') ';
19    query1:=query1+' order by cast(kode_item as int)';
20    . . .
```

```
21   while ADOQuery2.Eof=false do
22   begin
23     i:=i+1;
24     sup[i]:=ADOQuery2.FieldByName('support').AsInteger
25     kombinasi[i]:= DOQuery2.FieldByName('kode_item').
26                    AsString;
27     ADOQuery2.Next;
28   end;
29   . . .
30   //---itemset 2
31   query2:='select kode1,kode2,support from tb_itemset2
32            where ';
33   query2:=query2+'kode1='+kode[1]+' and
34            kode2='+kode[2]+' or ';
35   query2:=query2+'kode1='+kode[1]+' and
36            kode2='+kode[3]+' or ';
37   query2:=query2+'kode1='+kode[1]+' and
38            kode2='+kode[4]+' or ';
39   . . .
40   while ADOQuery2.Eof=false do
41   begin
42    i:=i+1;
43    sup[i]:=ADOQuery2.FieldByName('support').AsInteger;
44     . . .
45  //---itemset 3
46   query3:='select kode1,kode2,kode3,support support
47            from tb_itemset3 where ';
48   query3:=query3+'kode1='+kode[1]+'and
49            kode2='+kode[2] +' and kode3='+kode[3]+'
50            or ';
51   query3:=query3+'kode1='+kode[1]+' and
52     . . .
53  //---Count Confidence
54  for i:=1 to 14 do
55   begin
56     if sup[i]<1 then
57        break;
58     conf:=suport / sup[i];
59     {if conf>0.2 then
60         re1.Lines.Add(kombinasi[i]+' ->
61         '+kombinasi[15-i]+'  = '+FloatToStr(conf));}
62   end;
63    ADOQuery1.Next;
64    end;
65  end;
```

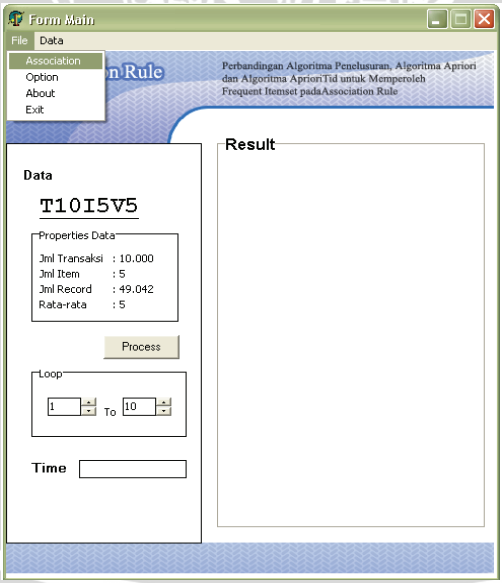
Gambar 4.9 Sintak prosedur association

4.2 Penerapan Aplikasi

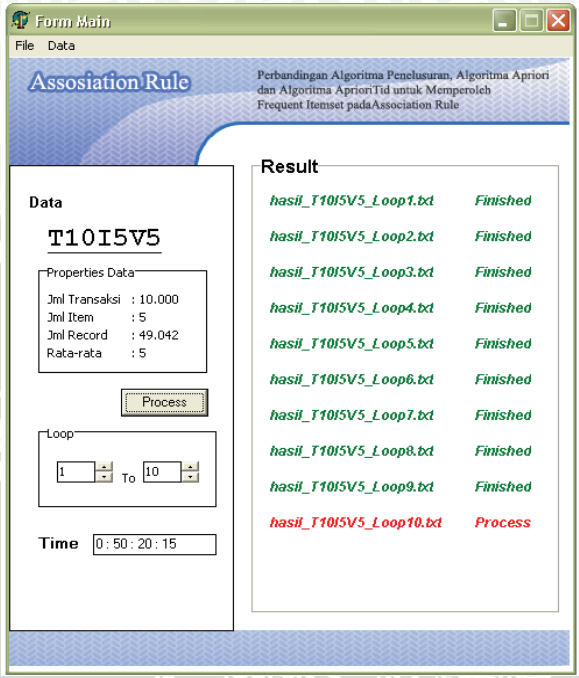
Aplikasi yang sudah dibuat digunakan untuk mendapatkan waktu proses yang dibutuhkan dalam menemukan *association rule* dengan ke-3 jenis algoritma pencari *frequent itemset* seperti yang sudah dijelaskan pada bab tiga. Pengujian satu jenis data dilakukan sebanyak 12 hingga 15 kali dan masing-masing hasil uji di simpan dalam file teks yang memiliki nama sesuai dengan nama data dan no urut pengujian. Misal file T10I5V5loop2.txt berarti bahwa isi pada file tersebut ialah hasil pengujian ke-2 pada data T10I5V5.

4.2.1 Form Main

Gambar 4.10 ialah tampilan awal dari aplikasi. Properties data menjelaskan keterangan dari data yang akan diuji. Keterangan tersebut berupa jumlah transaksi, jumlah item, jumlah record dan rata-rata kemunculan item dalam satu transaksi. Untuk merubah data yang akan diuji, user dapat memilih menu Data yang ada sebelah kanan menu File. Isian yang ada pada Loop digunakan untuk mendefinisikan berapa kali pengujian akan dilakukan. Tombol Process digunakan untuk mulai melakukan proses pengujian.



Gambar 4.10 Tampilan awal form pengujian



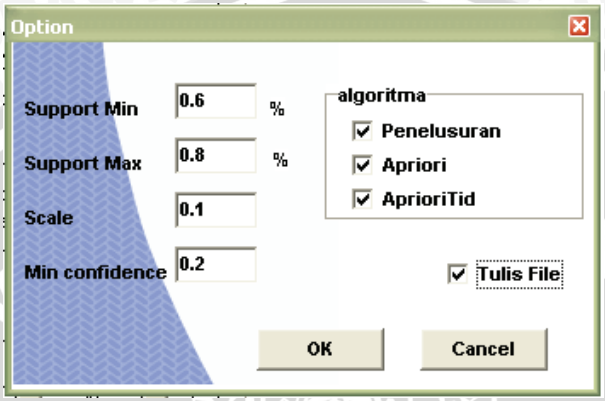
Gambar 4.11 Tampilan aplikasi sedang menjalankan proses

Pada saat aplikasi sedang menjalankan proses, form akan menampilkan nama file teks yang sedang diproses dengan warna hijau dan merah pada kolom result. Nama file yang berwarna hijau menandakan bahwa file teks tersebut telah diproses, sedangkan yang berwarna merah menandakan bahwa file teks tersebut masih dalam pembuatan.

4.2.2 Form Option

Untuk menampilkan form Option, user harus dapat memilih menu Option dalam menu File. Form ini berisi kolom isian yang nantinya digunakan sebagai parameter pada pencarian *frequent itemset* dan *association rule*. Isian Support min dan Support max digunakan untuk menentukan batas bawah dan batas atas persentase minimum *support*. Kolom isian Scale digunakan untuk menetukan besarnya kenaikan persentase *support* dari *support* min hingga *support* max. Isian Min confidence digunakan untuk menentukan

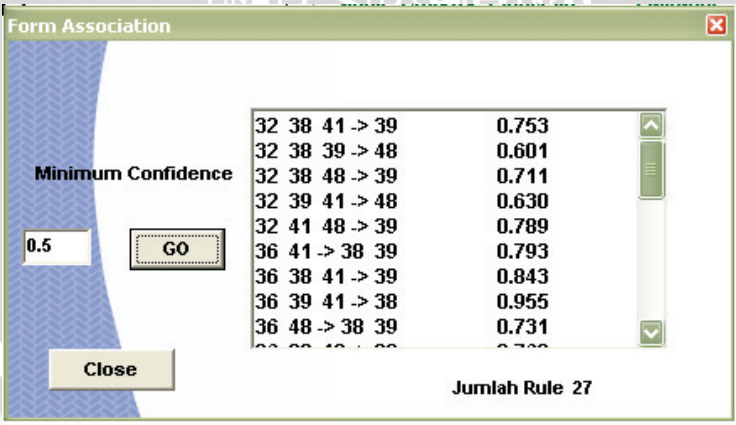
minimum confidence pada saat pencarian *association rule.* Gambar 4.12 adalah tampilan dari form Option.



Gambar 4.12 Tampilan form Option

4.2.3 Form Association

Hasil *association* yang terbentuk dapat dilihat pada form *Association*. Form ini ditampilkan dengan memilih menu Association dalam menu File. Menu tersebut akan aktif jika proses pada menu utama telah dijalankan. Gambar 4.13 adalah tampilan dari dari form Association.



Gambar 4.13 Tampilan form Association

4.2.4 Hasil Proses Aplikasi

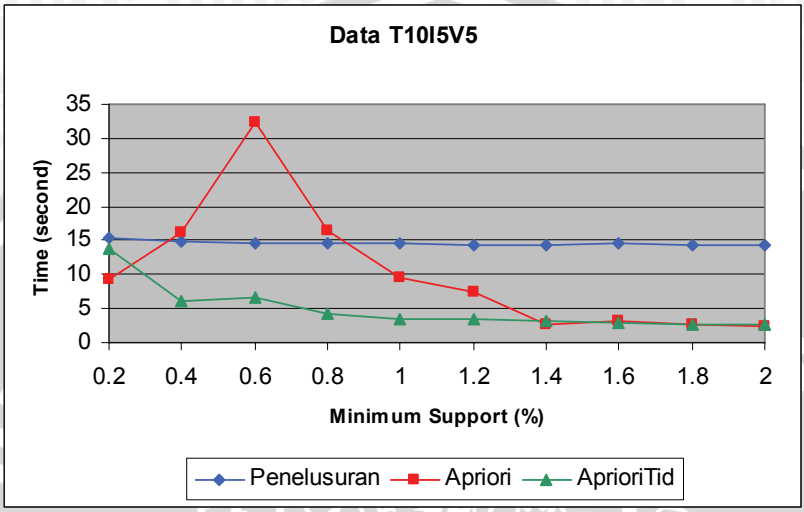
Seperti yang telah dijelaskan diatas, bahwa hasil dari aplikasi ini ialah file teks yang berisi waktu proses pencarian *association rule* yang dimulai dari pencarian *frequent itemset* dari 1 hingga 4 kombinasi. Gambar 4.14 adalah isi dari salah satu file teks yang terbentuk dari hasil proses aplikasi. Baris pertama pada file hasil menunjukkan besarnya minimum *support* yang diperoleh dari persentase minsup dikalikan dengan jumlah transaksi. Sedangkan baris ke-2 dan seterusnya berturut-turut ialah hasil waktu proes dari pencarian *association rule* dengan algoritma penelusuran, algoritma Apriori, AprioriTid.

20	40	60	80	100	120	140	160
18.453	15.328	14.547	14.532	14.578	19.703	17.844	16.0
10.891	4.484	32.843	16.657	10.375	7.187	2.594	3.141
14.593	6.141	5.922	3.828	3.688	3.390	2.969	2.813

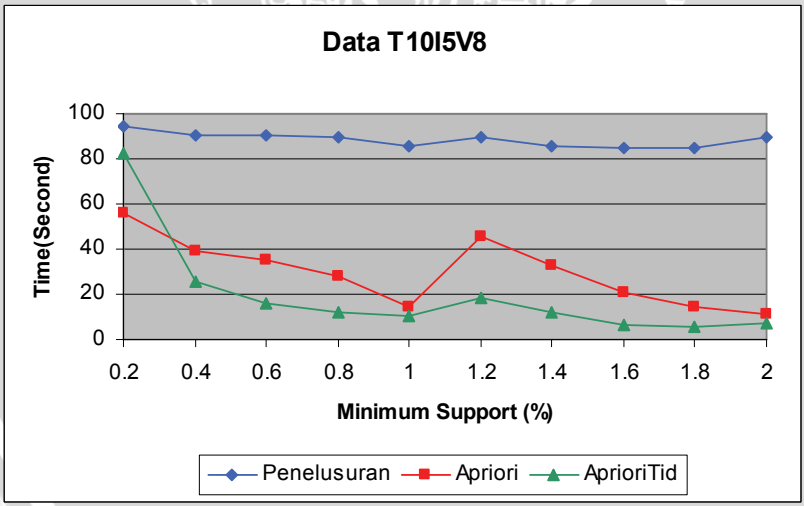
Gambar 4.14 Isi dari file text

4.3 Analisis Waktu Algoritma Penelusuran, Algoritma Apriori dan Algoritma AprioriTid

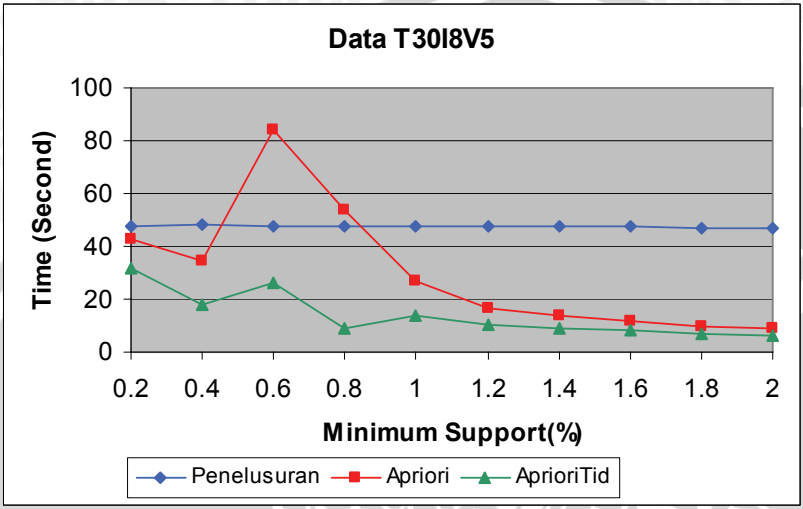
Untuk mendapatkan perbandingan waktu ke-3 algoritma tersebut, maka semua file text yang terbentuk dari jenis data yang sama dihitung rata-ratanya. Gambar 4.15, 4.16, 4.17 dan gambar 4.18 adalah grafik rata-rata waktu proses ke-3 jenis algoritma dalam mencari *association rule* untuk 4 jenis data yaitu data T10I5V5, T10I5V8, T30I8V5 dan T30I8V8.



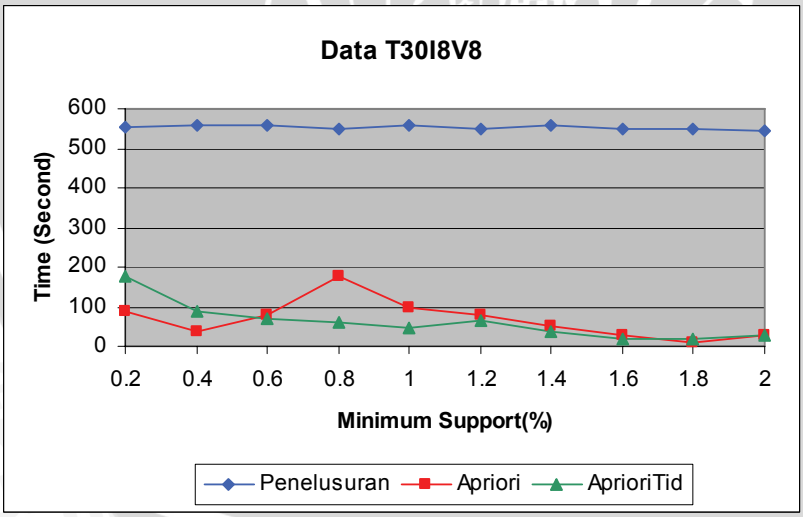
Gambar 4.15 Grafik hasil pengujian pada Data T10I5V5



Gambar 4.16 Grafik hasil pengujian pada Data T10I5V8



Gambar 4.17 Grafik hasil pengujian pada Data T30I8V5



Gambar 4.18 Grafik hasil pengujian pada Data T30I8V8

Berdasarkan empat gambar grafik, tampak bahwa algoritma penelusuran memiliki waktu proses yang lebih lama jika dibandingkan dengan waktu proses algoritma Apriori dan AprioriTid. Perubahan waktu proses dari suatu nilai minsup ke nilai minsup yang lebih besar cenderung stabil. Hal ini dapat dilihat pada tabel 4.1 yang mana algoritma penelusuran memiliki rata-rata perubahan kecepatan pada tiap minsup nya ialah sebesar 2.174 detik.

Tabel 4.1 Tabel rata-rata perubahan kecepatan proses antar minsup

	Time (Second)				
Algoritma	T10I5V5	T10I5V8	T30I8V5	T30I8V8	Rata-rata
Penelusuran	0.127	2.857	0.217	6.406	2.174
Apriori	5.992	11.963	14.659	41.618	18.558
AprioriTid	1.356	10.614	5.723	23.242	10.234

Berbeda dengan algoritma penelusuran, waktu proses algoritma apriori rata-rata lebih cepat. Hal ini disebabkan karena jumlah item pada daftar kandidat yang dilibatkan pada pencarian itemset lebih sedikit dari pada algoritma penelusuran. Jumlah record pada hasil algoritma apriori dapat dilihat pada lampiran I.

Hampir pada semua hasil grafik yang didapat, tampak terdapat perubahan waktu proses yang yang melambat secara signifikan pada minsup tertentu. Perubahan waktu proses yang seperti itu disebabkan karena perubahan *Execution Plan* (rencana eksekusi) dalam meng-eksekusi *stored procedure* sp_apriori.

Seperti yang telah dijelaskan pada bab sebelumnya bahwa *Stored procedure* pada MS SQL Server 2000 cukup di-*compile* sekali pada awal pembuatan. Hasil *compile* tersebut berupa *execution plan* yang memiliki *cost I/O* dan *cost CPU* paling minimum dari pada alternatif *execution plan* yang lain. Aturan tersebut tidak berlaku untuk *stored procedure* yang mana tabel yang dijadikan acuan mengalami banyak perubahan karena faktor *insert* (memasukkan *record* baru) maupun *delete* (menghapus *record*). Tabel 4.2 menunjukkan persentase beban proses eksekusi sp_apriori untuk data T10I5V5 yang diperoleh dari *Query* Analyser yang terdapat pada SQL Server 2000.

Tabel 4.2 Tabel beban proses dan cost

Langkah/ step pada sp_apriori	Minsup					
	40		60		80	
	Beban proses(%)	Cost	Beban proses(%)	Cost	Beban proses(%)	Cost
itemset1	1.190	0.498	5.090	0.497	7.700	0.498
kandidat2	0.150	0.061	1.130	0.110	0.990	0.069
itemset2	25.500	10.600	75.330	7.360	65.880	4.580
kandidat3	0.150	0.061	0.620	0.061	0.880	0.061
prune3	0.580	0.240	1.330	0.130	1.870	0.130
itemset3	49.360	20.600	6.430	0.628	9.050	0.638
kandidat4	0.150	0.061	0.620	0.061	0.880	0.061
prune4	0.950	0.397	2.870	0.280	4.040	0.280
itemset4	21.980	9.180	6.570	0.641	9.240	0.640

Nilai kolom cost didapat dari nilai *subtree cost* yang nampak pada diagram *execution plan* pada tiap langkah. Sedangkan nilai pada kolom persentase beban proses didapat dari persentase *query cost* pada masing-masing langkah pada *execution plan*. *Subtree cost* ialah nilai total cost dari suatu operasi pada *execution plan* dan semua operasi yang tergolong pada *subtree* yang sama.

Pada tabel 4.2 terlihat adanya perubahan beban proses pada beberapa langkah yang sama pada minsup yang berbeda namun perubahan *execution plan* tersebut tetap memiliki cost eksekusi yang paling minimum. Hal ini terlihat dari jumlah cost I/O dan CPU yang cenderung semakin menurun dari minsup yang kecil ke minsup yang lebih besar. Jumlah cost yang menurun tersebut sesuai dengan jumlah record yang terlibat pada pembentukan *itemset*. Makin besar minsup makin sedikit record yang ikut dalam proses. Penurunan jumlah record tersebut dapat dilihat pada tabel jumlah record pada lampiran I.

Pada penelitian ini, perubahan waktu proses algoritma AprioriTid dalam mencari *itemset* cenderung lebih stabil daripada algoritma Apriori pada tiap-tiap minsup-nya. Hal itu dapat dilihat pada tabel 4.1 yang mana selisih waktu pada apriori sebesar 18,558 detik sedangkan pada selisih waktu pada aprioritid sebesar 10,234 detik.

Melambatnya proses yang dialami algoritma ini juga karena perubahan *execution plan* pada *stored procedure* sp_aprioriTid.

Dalam segi kecepatan proses, algoritma AprioriTid rata-rata dapat melakukannya lebih cepat dari pada algoritma Penelusuran dan apriori.

Pada grafik data T30I8V8 untuk minsup 0,2 dan 0,4, waktu proses algoritma Aporiori dapat mengungguli AprioriTid. Hal ini disebabkan karena jumlah record pada tabel tb_newtrans2 pada minsup 0,2 dan 0,4 berurut-turut ialah 754904 record dan 505567 record. Jumlah ini jauh lebih besar dari pada jumlah *record* pada tabel tb_trans yang besarnya 237675 record. Jumlah *record* tersebut dapat dilihat pada tabel jumlah record pada lampiran I.

Berdasarkan data yang telah diperoleh dari hasil percobaan, diperoleh data kecepatan waktu algoritma Apriori dan AproiriTid relatif terhadap algoritma penelusuran. Pada data T10I5V5 algoritma apriori 30,22% lebih cepat dari algoritma Penelusuran sedangkan algoritma AprioriTid 66,6% lebih cepat dari pada algoritma Penelusuran. Untuk jenis data yang lain, ditunjukkan pada tabel 4.3.

Tabel 4.3 Presentase kecepatan algoritma Apriori dan AprioriTid terhadap algoritma penelusuran

Algoritma	T10I5V5	T10I5V8	T30I8V5	T30I8V8	Rata-Rata
Apriori	30,22 %	66,60 %	36,28 %	87,75 %	55.21 %
AprioriTid	66,60 %	78,55 %	70,90 %	89,10 %	76.29%

.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Beberapa kesimpulan yang dapat diambil dari hasil penelitian ini ialah :

1. Pada pengujian ini, algoritma penelusuran,algoritma Apriori dan Algoritma AprioriTid untuk mencari *frequent itemset* telah berhasil diimplementaskan dengan menggunakan bahasa *query*.
2. Penggunaan aprioriTid dapat mengoptimasi kecepatan proses pencarian *association rule* hingga rata-rata 76 % lebih cepat dari pada menggunakan aloritma penelusuran sedangkan algoritma apriori rata-rata 55% lebih cepat dari pada algoritma penelusuran.
3. Rata-rata selisih waktu proses antar minsup pada algoritma penelusuran, apriori dan aprioritid berturut-turut ialah 2,4 detik, 18,55 detik dan 10,23 detik. Hal ini menunjukkan bahwa penerapan algoritma apriori dan aprioritid dalam bahasa *query* yang dijalankan pada DBMS SQL Server 2000 memiliki waktu proses pencarian *association rule* yang tidak stabil jika dibandingkan dengan penerapan algoritma penelusuran.

5.2 Saran

Untuk pengembangan lebih lanjut, disarankan menggunakan tool DBMS SQL server 2000 Service Pack 2 atau SQL server 2005 karena kedua DBMS tersebut telah memiki suatu fasilitas atau fungsi untuk mencegah DBMS meng-*compile* ulang suatu *stored procedure*. Dengan pencegahan tersebut ada kemungkinan bahwa waktu proses dua jenis algoritma tersebut berjalan lebih stabil pada tiap perubahan minsup-nya.

Penelitian selanjutnya juga disarankan untuk menganalisis kinerja *query* tidak hanya dari sisi waktu proes eksekusi melainkan juga dari sisi kompleksitas *query*.

Daftar Pustaka

Agrawal, Rakesh dan Ramakrishnan Srikant. Fast Algorithm for Mining Association Rules.1994 http://www.sigmod.org/vldb/conf/1994/P487.PDF Diakses tanggal 24 Maret 2007

Anonymous. Association Rules. http://www.aicomponents.com Diakses tanggal 1 April 2007

Anonymous. Conditions for Stored Procedure Recompilation. http://www.microsoft.com/en/us/default.aspx Diakses tanggal 2 November 2007

Anonymous. Dasar Analisis Asosiasi. http://www.intro-dm.pbwiki.com. Diakses tanggal 1 April 2007

Anonymous. Execution Plan Caching and Reuse. http://www.msdn2.microsoft.com/en-us/library/ Diakses tanggal 21 Oktober 2007

Anonymous. Single SQL Statement Processing http://www.msdn2.microsoft.com/en-us/library/ Diakses tanggal 22 Oktober 2007

Anonymous. Stored Procedure http://www.msdn2.microsoft.com/en-us/library/ Diakses tanggal 22 Oktober 2007

Ganesha, Ramesh dan Mohammed J, Zaki. Indexing and Data Access Methods for Database Mining. 2002

Goethals. 2000 Bart. Survey On Frequent Pattern Mining.

Hand, David dkk. 2001. Principle of Data Mining.

Hegland, Markus, 2003. Data Mining- Challenge, Models, Methods and Algorithms.

Margaret, H.Dunham dkk. 2000.A Survey of Association Rules.

Nugroho, Adi. 2004. *Konsep Pengembangan Sistem Basis Data*. Informatika, Bandung

Pramudiono, Iko. 2007. Algoritma Apriori. http://datamining.japati.net/forum/index.php?topic=10.0. Diakses tanggal 1 April 2007

Prasetyo, Philips Kokoh. Aprioi. 2006 http://www.philips.wordpress.com. Diakses tanggal 1 April 2007

Purdom, W. Paul dan Gutch Van Derk. 1999. Averages Case Performance of the Apriori Algorithm.

Rajamani,Karthick.dkk. Efficient Mining for Association Rules with Relational Database Systems. 2001 http://tech.groups.yahoo.com/group/indo-dm/files/ Diakses tanggal 2 Agutus 2007

Rakesh Agrawal dkk. 1997. Mining Associoation Rules Between Sets of Items in Large Database.

Santoso,Adi. Keunggulan Stored Procedure dibandingkan adhoc query. 2006 http://forum.uinsgd.ac.id/showthread.php?t=96 Diakses tanggal 22 oktober 2007

Silberschatz dkk. 2002. *Data Base Concept, Fourth Edition*. Mc.Graw-Hill Higher Education. New York.

Suhendric, Dejan dan Woodhead Tom. 200. SQL Server Stored Procedure Programming.

William, Graham.Data Mining Desktop Survival Guide.2001 http://www.datamining.togaware.com". Diakses tanggal 15 April 2007

LAMPIRAN

Lampiran 1

1. Table Record Algoritma Penelusuran

Nama data : T10I5V5
Jml record Tabel transaksi : 49042

Minsup	1-Itemset	2-Itemset	3-Itemset	4-Itemset
20	284	340	141	33
40	121	121	54	13
60	69	65	34	8
80	41	48	24	6
100	32	37	17	1
120	25	30	13	1
140	21	21	10	1
160	19	18	9	1
180	15	17	9	1
200	14	16	9	1

Nama data : T10I5V8
Jml record Tabel transaksi : 79853

Minsup	1-Itemset	2-Itemset	3-Itemset	4-Itemset
20	696	957	513	127
40	254	363	188	39
60	156	196	96	21
80	99	126	59	13
100	67	79	73	11
120	51	61	35	8
140	35	50	27	6
160	30	42	24	6
180	27	34	20	6
200	24	31	19	4

Nama data : T30I8V5
Jml record Tabel transaksi : 153348

Minsup	1-Itemset	2-Itemset	3-Itemset	4-Itemset
60	1201	1272	530	99
120	479	480	202	39
180	274	285	115	20
240	191	200	74	13
300	142	142	51	10
360	109	102	43	9
420	83	75	38	8
480	70	65	31	6
500	64	54	24	6
600	54	51	23	4

Nama data : T30I8V8
Jml record Tabel transaksi : 237675

Minsup	1-Itemset	2-Itemset	3-Itemset	4-Itemset
60	2225	3603	2023	530
120	1067	1342	675	156
180	610	714	361	75
240	394	474	233	45
300	281	349	151	30
360	213	252	112	25
420	181	212	87	21
480	148	172	70	14
500	122	132	54	11
600	103	110	47	10

2. Tabel Recod Algoritma Apriori

Nama data : T10I5V5
Jml record Tabel transaksi : 49042

Minsup	1-Itemset	Kandidat 2	2-Itemset	Kandidat 3	3-Itemset	Kandidat 4	4-Itemset
20	284	40186	340	14079	141	1385	33
40	121	7260	121	1735	54	180	13
60	69	2346	65	405	34	59	8
80	41	820	48	210	24	22	6
100	32	496	37	124	17	14	1
120	25	300	30	86	13	8	1
140	21	210	21	50	10	7	1
160	19	171	18	32	9	7	1
180	15	105	17	27	9	7	1
200	14	91	16	23	9	7	1

Nama data : T10I5V8
Jml record Tabel transaksi : 79853

Minsup	1-Itemset	Kandidat 2	2-Itemset	Kandidat 3	3-Itemset	Kandidat 4	4-Itemset
20	696	241860	957	116746	513	20910	127
40	254	32131	363	16566	188	2732	39
60	156	12090	196	4665	96	668	21
80	99	4851	126	1942	59	235	13
100	67	2211	79	700	73	110	11
120	51	1275	61	403	35	66	8
140	35	595	50	291	27	36	6
160	30	435	42	174	24	28	6
180	27	351	34	117	20	16	6
200	24	276	31	94	19	13	4

Nama data : T30I8V5

Jml record Tabel transaksi : 153348

Minsup	1-Itemset	Kandidat 2	2-Itemset	Kandidat 3	3-Itemset	Kandidat 4	4-Itemset
60	1201	720600	1272	220631	530	23331	99
120	479	114481	480	30169	202	3241	39
180	274	37401	285	9919	115	1091	20
240	191	18145	200	4560	74	419	13
300	142	10011	142	2441	51	179	10
360	109	5886	102	1351	43	117	9
420	83	3403	75	686	38	92	8
480	70	2415	65	497	31	60	6
500	64	2016	54	335	24	27	6
600	54	1431	51	293	23	23	4

Nama data : T30I8V8

Jml record Tabel transaksi : 237675

Minsup	1-Itemset	Kandidat 2	2-Itemset	Kandidat 3	3-Itemset	Kandidat 4	4-Itemset
60	2225	2474200	3603	1529857	2023	300318	530
120	1067	568711	1342	234849	675	35341	156
180	610	185745	714	64351	361	1139	75
240	394	77421	474	27754	233	4798	45
300	281	39340	349	14745	151	1971	30
360	213	22578	252	7979	112	1061	25
420	181	16290	212	5470	87	563	21
480	148	10878	172	3675	70	353	14
500	122	7381	132	2146	54	211	11
600	103	5253	110	1405	47	158	10

3. Tabel Recod Algoritma AprioriTid

Nama data : T10I5V5

Jml record Tabel transaksi : 49042

Minsup	1-Itemset	Kandidat 2	Newtrans 2	2-Itemset	Kandidat 3	Newtrans 3	3-Itemset	Kandidat 4	Newtrans 4	4-Itemset
20	284	40186	45412	340	14079	19430	141	1385	3364	33
40	121	7260	32616	121	1735	11279	54	180	1922	13
60	69	2346	26052	65	405	8157	34	59	1428	8
80	41	820	21823	48	210	7165	24	22	1063	6
100	32	496	20186	37	124	6482	17	14	943	1
120	25	300	18384	30	86	5829	13	8	651	1
140	21	210	17225	21	50	5165	10	7	645	1
160	19	171	16600	18	32	4707	9	7	645	1
180	15	105	15296	17	27	4572	9	7	645	1
200	14	91	14946	16	23	4415	9	7	645	1

Nama data : T10I5V8

Jml record Tabel transaksi : 79853

Minsup	1-Itemset	Kandidat 2	Newtrans 2	2-Itemset	Kandidat 3	Newtrans 3	3-Itemset	Kandidat 4	Newtrans 4	4-Itemset
20	696	241860	154195	957	116746	90635	513	20910	23100	127
40	254	32131	91938	363	16566	48120	188	2732	10698	39
60	156	12090	71929	196	4665	32540	96	668	6432	21
80	99	4851	57228	126	1942	24630	59	235	4556	13
100	67	2211	47279	79	700	18552	73	110	3419	11
120	51	1275	41870	61	403	16071	35	66	2965	8
140	35	595	35829	50	291	14694	27	36	2505	6
160	30	435	33837	42	174	12966	24	28	2021	6
180	27	351	32515	34	117	11831	20	16	1728	6
200	24	276	30831	31	94	11328	19	13	1702	4

Nama data : T30I8V5
Jml record Tabel transaksi : 153348

Minsup	1-Itemset	Kandidat 2	Newtrans 2	2-Itemset	Kandidat 3	Newtrans 3	3-Itemset	Kandidat 4	Newtrans 4	4-Itemset
60	1201	720600	222779	1272	220631	96460	530	23331	18091	99
120	479	114481	149866	480	30169	56646	202	3241	9830	39
180	274	37401	118751	285	9919	43493	115	1091	6852	20
240	191	18145	102641	200	4560	35454	74	419	5058	13
300	142	10011	91189	142	2441	30184	51	179	3943	10
360	109	5886	82027	102	1351	25319	43	117	3605	9
420	83	3403	73476	75	686	21429	38	92	3169	8
480	70	2415	68705	65	497	19967	31	60	2682	6
500	64	2016	66272	54	335	18464	24	27	2229	6
600	54	1431	62385	51	293	17923	23	23	2212	4

Nama data : T30I8V8
Jml record Tabel transaksi : 237675

Minsup	1-Itemset	Kandidat 2	Newtrans 2	2-Itemset	Kandidat 3	Newtrans 3	3-Itemset	Kandidat 4	Newtrans 4	4-Itemset
60	2225	2474200	754904	3603	1529857	548621	2023	300318	160546	530
120	1067	568711	505567	1342	234849	279947	675	35341	68547	156
180	610	185745	363830	714	64351	178969	361	1139	41705	75
240	394	77421	284006	474	27754	133053	233	4798	28801	45
300	281	39340	235500	349	14745	106947	151	1971	19779	30
360	213	22578	203914	252	7979	86919	112	1061	15754	25
420	181	16290	186775	212	5470	77321	87	563	12786	21
480	148	10878	167023	172	3675	66544	70	353	10706	14
500	122	7381	150846	132	2146	55941	54	211	8847	11
600	103	5253	138142	110	1405	48690	47	158	7843	10

Lampiran II

Tabel Waktu Proses dan Tabel Selisih Waktu Dalam Satuan Detik (second)

1. Data T10I5V5

Tabel Waktu

Algoritma	Minsup (%)									
	0.2	**0.4**	**0.6**	**0.8**	**1**	**1.2**	**1.4**	**1.6**	**1.8**	**2**
Penelusuran	15.352	14.761	14.705	14.499	14.466	14.401	14.384	14.451	14.348	14.345
Apriori	9.259	16.229	32.228	16.443	9.648	7.419	2.605	3.206	2.761	2.475
AprioriTid	13.694	6.215	6.696	4.201	3.550	3.573	3.064	2.847	2.638	2.784

Tabel Selisih Waktu

Algoritma	Minsup (%)									
	0.4 - 0.2	0.6 - 0.4	0.8 - 0.6	1 - 0.8	1.2 -1	1.4 - 1.2	1.6 - 1.4	1.8 - 1.6	2 - 1.8	Rata-rata
Penelusuran	0.591	0.056	0.206	0.033	0.064	0.018	0.067	0.103	0.002	0.127
Apriori	6.970	15.999	15.785	6.795	2.229	4.814	0.601	0.445	0.286	5.992
AprioriTid	7.479	0.481	2.495	0.651	0.023	0.509	0.217	0.209	0.145	1.356

2. Data T10I5V8

Tabel Waktu

Algoritma	Minsup (%)									
	0.2	**0.4**	**0.6**	**0.8**	**1**	**1.2**	**1.4**	**1.6**	**1.8**	**2**
Penelusuran	552.899	558.321	556.751	550.021	559.437	547.664	557.844	549.023	547.790	545.281
Apriori	87.024	39.055	79.400	178.430	97.414	77.063	52.977	27.813	9.860	28.509
AprioriTid	175.249	90.172	68.133	58.180	45.977	66.492	37.196	18.750	16.789	26.477

Tabel Selisih Waktu

Algoritma	Minsup (%)									
	0.4 - 0.2	0.6 - 0.4	0.8 - 0.6	1 - 0.8	1.2 -1	1.4 - 1.2	1.6 - 1.4	1.8 - 1.6	2 - 1.8	Rata-rata
Penelusuran	4.176	0.510	1.039	4.312	4.730	4.795	0.210	0.589	5.352	2.857
Apriori	16.470	4.052	7.216	13.931	31.511	12.968	12.343	6.174	3.005	11.963
AprioriTid	56.977	9.251	4.337	1.696	8.198	6.092	6.022	0.782	2.169	10.614

3. Data T30I8V5

Tabel Waktu

Algoritma	Minsup (%)									
	0.2	**0.4**	**0.6**	**0.8**	**1**	**1.2**	**1.4**	**1.6**	**1.8**	**2**
Penelusuran	47.707	48.176	47.349	47.435	47.446	47.288	47.265	47.427	47.217	47.214
Apriori	42.459	34.427	83.808	53.883	27.107	16.683	13.737	11.744	9.564	9.287
AprioriTid	31.749	17.784	25.987	8.868	13.658	10.264	8.935	8.244	6.582	6.230

Tabel Selisih Waktu

Algoritma	Minsup (%)									
	0.4 - 0.2	0.6 - 0.4	0.8 - 0.6	1 - 0.8	1.2 -1	1.4 - 1.2	1.6 - 1.4	1.8 - 1.6	2 - 1.8	Rata-rata
Penelusuran	0.469	0.827	0.085	0.011	0.158	0.023	0.162	0.210	0.003	0.217
Apriori	8.031	49.381	29.926	26.776	10.423	2.946	1.993	2.180	0.277	14.659
AprioriTid	13.965	8.203	17.119	4.790	3.394	1.329	0.690	1.662	0.352	5.723

4. Data T30I8V8

Tabel Waktu

Algoritma	Minsup (%)									
	0.2	**0.4**	**0.6**	**0.8**	**1**	**1.2**	**1.4**	**1.6**	**1.8**	**2**
Penelusuran	552.899	558.321	556.751	550.021	559.437	547.664	557.844	549.023	547.790	545.281
Apriori	87.024	39.055	79.400	178.430	97.414	77.063	52.977	27.813	9.860	28.509
AprioriTid	175.249	90.172	68.133	58.180	45.977	66.492	37.196	18.750	16.789	26.477

Tabel Selisih Waktu

Algoritma	Minsup (%)									
	0.4 - 0.2	0.6 - 0.4	0.8 - 0.6	1 - 0.8	1.2 -1	1.4 - 1.2	1.6 - 1.4	1.8 - 1.6	2 - 1.8	Rata-rata
Penelusuran	5.422	1.571	6.730	9.416	11.773	10.180	8.821	1.233	2.509	6.406
Apriori	47.969	40.345	99.030	81.016	20.352	24.086	25.164	17.953	18.649	41.618
AprioriTid	85.077	22.040	9.953	12.203	20.516	29.297	18.446	1.961	9.688	23.242

Lampiran III
Listing Stored Procedure pencari frequent Itemset

1. Stored Procedure sp_Apriori

```
Create procedure sp_penelusuran(@minsup int) as
--//Mengisi tabel 1-itemset
insert into tb_itemset1
select kode_item,count(*)
from tb_trans
group by kode_item
having count(*)> @minsup

--//Mengisi tabel 2-itemset
insert into tb_itemset2
select Trans1.kode_item, Trans2.kode_item, count(*)
from tb_trans as Trans1, tb_trans as Trans2
where Trans1.no_trans  = Trans2.no_trans and
      Trans1.kode_item <  Trans2.kode_item
group by Trans1.kode_item, Trans2.kode_item
having count(*) > @minsup

--//Mengisi tabel 3-itemset
insert into tb_itemset3
select Trans1.kode_item, Trans2.kode_item,
       Trans3.kode_item, count(*)
from tb_trans as Trans1, tb_trans as Trans2,
     tb_trans as Trans3
where Trans1.no_trans  = Trans2.no_trans and
      Trans1.kode_item < Trans2.kode_item and
      Trans2.no_trans  = Trans3.no_trans and
      Trans2.kode_item < Trans3.kode_item
group by Trans1.kode_item, Trans2.kode_item,
         Trans3.kode_item
having count(*) > @minsup

--//Mengisi tabel 4-itemset
insert into tb_itemset4
select Trans1.kode_item, Trans2.kode_item,
       Trans3.kode_item, Trans4.kode_item, count(*)
from tb_trans as Trans1, tb_trans as Trans2,
     tb_trans as Trans3, tb_trans as Trans4
where Trans1.no_trans = Trans2.no_trans and
      Trans1.kode_item < Trans2.kode_item and
      Trans2.no_trans = Trans3.no_trans and
      Trans2.kode_item < Trans3.kode_item and
      Trans3.no_trans = Trans4.no_trans and
      Trans3.kode_item < Trans4.kode_item
```

```
group by Trans1.kode_item,Trans2.kode_item,
         Trans3.kode_item, Trans4.kode_item
having count(*) > @minsup

GO
```

2. Stored Procedure sp_Apriori

```
CREATE procedure sp_apriori(@minsup int) as
--//Mengisi table 1-itemset
Insert into tb_itemset1
Select kode_item, count(*)
From   tb_trans
Group by kode_item
Having count(*) > @minsup

--//Mengisi table kandidat 2-itemset
Insert into tb_kandidat2
Select Item1.kode_item, Item2.kode_item
From tb_itemset1 as Item1, tb_itemset1 as Item2
Where Item1.kode_item < Item2.kode_item

--//Mengisi table 2-itemset
Insert into tb_itemset2
Select Trans1.kode_item, Trans2.kode_item, count(*)
From tb_trans as Trans1, tb_trans as Trans2,
     tb_kandidat2 as Candidat
Where Trans1.no_trans = Trans2.no_trans and
      Candidat.kode1  = Trans1.kode_item and
      Candidat.kode2  = Trans2.kode_item
Group by Trans1.kode_item, Trans2.kode_item
Having count(*)> @minsup

--//Mengisi table  kandidat 3-itemset
Insert into tb_kandidat3
Select Item1.kode1, Item1.kode2, Item2.kode2
From tb_itemset2 as Item1,tb_itemset2 as Item2
Where  Item1.kode1 = Item2.kode1 and
       Item1.kode2 < Item2.kode2

--//prune step3
Delete from tb_kandidat3
Where kode2 not in(select kode1 from tb_itemset2) and
      kode3 not in(select kode2 from tb_itemset2)

--//Mengisi table 3-itemset
Insert into tb_itemset3
Select Trans1.kode_item, Trans2.kode_item,
       Trans3.kode_item, count(*)
From tb_trans as Trans1, tb_trans as Trans2,
     tb_trans as Trans3, tb_kandidat3 as Candidat
Where Trans1.no_trans = Trans2.no_trans and
      Trans2.no_trans = Trans3.no_trans and
      Candidat.kode1  = Trans1.kode_item and
      Candidat.kode2  = Trans2.kode_item and
```

```
      Candidat.kode3  = Trans3.kode_item
Group by Trans1.kode_item,Trans2.kode_item,Trans3.kode_item
Having count(*) > @minsup

--//Mengisi table  kandidat 4-itemset
Insert into tb_kandidat4
Select Item1.kode1, Item1.kode2, Item1.kode3, Item2.kode3
From tb_itemset3 as Item1,tb_itemset3 as Item2
Where Item1.kode1 = Item2.kode1 and
      Item1.kode2 = Item2.kode2 and
      Item1.kode3 < Item2.kode3

--//prune step4
Delete from tb_kandidat4
Where (kode1 not in(select kode1 from tb_itemset2) and
       kode3 not in(select kode2 from tb_itemset2) and
       kode4 not in(select kode3 from tb_itemset2))
   or (kode2 not in(select kode1 from tb_itemset2) and
       kode3 not in(select kode2 from tb_itemset2) and
       kode4 not in(select kode3 from tb_itemset2))

--//Mengisi table  4-itemset
Insert into tb_itemset4
Select Trans1.kode_item,Trans2.kode_item,Trans3.kode_item,
       Trans4.kode_item, count(*)
From tb_trans as Trans1, tb_trans as Trans2, tb_trans as
     Trans3, tb_trans as Trans4, tb_kandidat4 as Candidat
Where Trans1.no_trans = Trans2.no_trans and
      Trans2.no_trans = Trans3.no_trans and
      Trans3.no_trans = Trans4.no_trans and
      Candidat.kode1  = Trans1.kode_item and
      Candidat.kode2  = Trans2.kode_item and
      Candidat.kode3  = Trans3.kode_item and
      candidat.kode4  = Trans4.kode_item
Group by Trans1.kode_item,Trans2.kode_item,Trans3.kode_item,
         Trans4.kode_item
Having count(*)> @minsup

GO
```

3. Stored Procedure sp_Apriori_Tid

```
CREATE procedure sp_apriori_tid(@minsup int) as
--//Mengisi table  1-itemset
Insert into tb_itemset1
Select kode_item,count(*)
From tb_trans
Group by kode_item
Having count(*)> @minsup

--//Mengisi table  kandidat 2-itemset
Insert into tb_kandidat2
Select Itemset1.kode_item, Itemset2.kode_item
From tb_itemset1 as Itemset1,tb_itemset1 as Itemset2
Where Itemset1.kode_item < Itemset2.kode_item

--//Mengisi table transaksi baru  2-itemset
Insert into tb_newtrans2
Select Trans1.no_trans,Trans1.kode_item,
       Trans2.kode_item
From tb_trans as trans1,tb_trans as trans2,
     tb_kandidat2 as Candidat
Where Trans1.no_trans = Trans2.no_trans and
      Candidat.kode1 = Trans1.kode_item and
      Candidat.kode2 = Trans2.kode_item
Group by Trans1.no_trans, Trans1.kode_item,
         Trans2.kode_item

--Mengisi table  2-itemset
Insert into tb_itemset2
Select kode1,kode2,count(*)
From tb_newtrans2
Group by kode1,kode2
Having count(*) > @minsup

--Mengisi table  kandidat 3-itemset
Insert into tb_kandidat3
Select Itemset1.kode1, Itemset1.kode2, Itemset2.kode2
From tb_itemset2 as Itemset1, tb_itemset2 as Itemset2
Where  Itemset1.kode1 = Itemset2.kode1 and
       Itemset1.kode2 < Itemset2.kode2

--//prune step3
Delete from tb_kandidat3
Where kode2 not in(select kode1 from tb_itemset2) and
      kode3 not in(select kode2 from tb_itemset2)
```

```
--//Mengisi table transaksi baru  3-itemset
Insert into tb_newtrans3
Select Trans1.no_trans, Trans1.kode1, Trans1.kode2,
       Trans2.kode2
From tb_newtrans2 as Trans1, tb_newtrans2 as Trans2,
     tb_kandidat3 as Candidat
Where Trans1.no_trans = Trans2.no_trans  and
      Candidat.kode1  = Trans1.kode1 and
      Candidat.kode2  = Trans1.kode2 and
      Candidat.kode3  = Trans2.kode2
Group by Trans1.no_trans, Trans1.kode1, Trans1.kode2,
         Trans2.kode2

--//Mengisi table  3-itemset
Insert into tb_itemset3
Select kode1, kode2, kode3, count(*)
From   tb_newtrans3
Group by kode1,kode2,kode3
Having count(*) > @minsup

--//Mengisi table  kandidat 4-itemset
Insert into tb_kandidat4
Select itemset1.kode1,itemset1.kode2,itemset1.kode3,
       itemset2.kode3
From tb_itemset3 as itemset1,tb_itemset3 as itemset2
Where  itemset1.kode1 = itemset2.kode1 and
       itemset1.kode2 = itemset2.kode2 and
       itemset1.kode3 < itemset2.kode3

--//prune step4
Delete from tb_kandidat4
Where (kode1 not in(select kode1 from tb_itemset2) and
       kode3 not in(select kode2 from tb_itemset2) and
       kode4 not in(select kode3 from tb_itemset2))
   or (kode2 not in(select kode1 from tb_itemset2) and
       kode3 not in(select kode2 from tb_itemset2) and
       kode4 not in(select kode3 from tb_itemset2))

--//Mengisi table transaksi baru  4-itemset
Insert into tb_newtrans4
Select Trans1.no_trans, Trans1.kode1, Trans1.kode2,
       Trans1.kode3, Trans2.kode3
From tb_newtrans3 as Trans1, tb_newtrans3 as Trans2,
     tb_kandidat4 as Candidat
Where Trans1.no_trans = Trans2.no_trans  and
      Candidat.kode1  = Trans1.kode1 and
      Candidat.kode2  = Trans1.kode2 and
      Candidat.kode3  = Trans1.kode3 and
      Candidat.kode4  = Trans2.kode3
```

```
Group by Trans1.no_trans, Trans1.kode1, Trans1.kode2,
         Trans1.kode3, Trans2.kode3

--//Mengisi table  4-itemset
Insert into tb_itemset4
Select kode1, kode2, kode3, kode4, count(*)
From tb_newtrans4
Group by kode1, kode2, kode3, kode4
Having count(*) > @minsup

GO
```