25027/H/06

MILIK PERPUSTAKAAN
ITS

IMPLEMENTASI PENGGALIAN KAIDAH ASOSIASI TANPA AMBANG BATAS SUPPORT DENGAN MENGGUNAKAN ALGORITMA BOMO

TUGAS AKHIR

RSSI
005.1
Per
i-1
2006

PERPUSTAKAAN
ITS

Tgl. Terima	16-2-06
Terima Dari	H
No. Agenda Prp.	224110

Disusun Oleh :

FATWA PERDANA
5201.100.026

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2006

IMPLEMENTASI PENGGALIAN KAIDAH ASOSIASI TANPA AMBANG BATAS SUPPORT DENGAN MENGGUNAKAN ALGORITMA BOMO

TUGAS AKHIR

Diajukan untuk Memenuhi Sebagian Persyaratan

Memperoleh Gelar Sarjana Komputer

Pada

Program Studi Sistem Informasi

Fakultas Teknologi Informasi

Institut Teknologi Sepuluh Nopember

Surabaya

Mengetahui/Menyetujui

Dosen Pembimbing I

DEPARTEMEN PENDIDIKAN NASIONAL
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI INFORMASI
PROGRAM STUDI SISTEM INFORMASI

Prof. Dr. Ir. Arif Djunaidy, M.Sc.
NIP. 131.633.403

SURABAYA

JANUARI 2006

Untuk Ibuku dan Ayahku
atas pengorbanan dan
kasih sayangnya

ABSTRAK

ABSTRAK

Pada penggalian pola kaidah assosiasi, biasanya nilai batas ambang dukungan minimum telah ditentukan untuk mencari *frequent itemsets*. Dalam prakteknya, penentuan nilai batas ambang dukungan minimum yang tepat sulit dilakukan karena nilai batas ambang dukungan minimum akan berbeda untuk setiap ukuran dan jenis data.

Berdasarkan permasalahan tersebut, dalam tugas akhir ini diimplementasikan algoritma *BOMO (Build-Once and Mine-Once)* dan pendekatan *loopback* untuk mencari *frequent itemsets* tanpa menentukan nilai ambang batas dukungan minimum. Dalam konteks ini, proses pencarian *frequent itemset* diupayakan menjadi proses pencarian *n* buah *k-itemset* paling menarik dari sebuah *FP-tree* data transaksi yang dibangkitkan. Adapun proses pencarian dapat dimulai dari penelusuran dari awal *FP-tree* (*top down*), penelusuran dari bawah *FP-tree* (*bottom up*), penelusuran dari tengah *FP-tree* (*middle*) atau pendekatan *loopback*. Untuk mencari *n* buah *k-itemset* paling menarik, maka kedua algoritma memiliki tahapan yang meliputi: pengurutan item berdasarkan jumlah dukungan, pembangkitan *FP-tree*, proses pencarian *n* buah *k-itemset* paling menarik dan proses pencarian batas ambang dukungan minimum untuk setiap *itemset*

Aplikasi yang telah berhasil dibuat dilakukan uji coba terhadap data transaksi buatan dengan berbagai parameter yang berbeda. Hasil uji coba terhadap data transaksi tiruan menunjukkan bahwa nilai ambang batas dukungan yang dihasilkan akan berbeda pada setiap data transaksi. Selain itu dapat disimpulkan pula bahwa waktu komputasi proses penelusuran dari tengah *FP-tree* lebih cepat dibandingkan penelusuran dari atas *FP-tree*, penelusuran dari bawah *FP-tree* atau pendekatan *loopback*. Dari aspek penggunaan memori dapat disimpulkan bahwa pendekatan *loopback* membutuhkan memori yang lebih kecil untuk menyimpan *FP-tree* dibandingkan algoritma *BOMO*.

Kata Kunci : *pola kaidah asosiasi, n* buah *k-itemset paling menarik, FP-tree, BOMO, pendekatan loopback*

KATA PENGANTAR

KATA PENGANTAR

Tiada lain puja dan sembah syukur hanya untuk Sang Pencipta, Allah SWT yang selalu memberikan bimbingan dan petunjuk sehingga akhir dari awal perjuangan ini dapat dilalui dengan indah dan manis. Tidak dapat dibayangkan apabila perjalanan melelahkan ini dijalani tanpa kasih dan sayang-Nya. Keajaiban-Nya sangat terasa dan barangkali tidak akan terulang kembali.

Perjalanan ini diakhiri dengan dibuatnya Tugas Akhir yang berjudul

"IMPLEMENTASI PENGGALIAN KAIDAH ASOSIASI TANPA AMBANG BATAS SUPPORT DENGAN MENGGUNAKAN ALGORITMA BOMO"

sebagai salah satu persyaratan dalam meraih gelar semu disisi-Nya di Program Sudi Sistem Informasi Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember Surabaya. Tugas Akhir ini dipersembahkan khusus untuk Ayahanda tercinta yang saat ini semoga saat ini masih tetap berada di sisi-Nya.

Terima kasih penulis sampaikan kepada :

- Ibunda tercinta **Ismiyati** dan Ayahda tercinta **Nahdodin**, atas semua doa, cucuran keringat maupun keikhlasan hati beliau.
- Saudara kandung Ipnu rasionalita dan Huda lizaman, atas semua dukungan dan semangatnya.
- Bpk. Prof. Dr. Ir. Arif Djunaidy, M.Sc selaku Dosen Pembimbing yang selalu memberi saran dan arahan sebelum dan sesudah Tugas Akhir
- Bpk. Febriliyan Samopa S.Kom selaku Dosen Wali yang selalu memberi saran dan arahan selama masa studi
- Bpk. Ir. Khakim Ghozali selaku Ketua Program Studi Sistem Informasi ITS
- Bpk. Rully Soelaiman, S.Kom, M.Kom selaku Sekretaris Program Studi Sistem Informasi ITS
- Ibu Wiwik Anggraeni S.Si, M.Kom selaku Koordinator Tugas Akhir, atas perhatiannya kepada para pengambil TA

- Dosen-dosen Sistem Informasi: Bpk. Ir. Aris Tjahyanto, M.Kom, Bpk. Arif Djunaidy, MSc. PhD., Bpk. Febriliyan Samopa, S.Kom, M.Kom, Bpk. Mudjahidin, ST, MT, Bpk. Faizal Johan A, S.Kom, Bpk. Edwin Riksakomara, S.Kom, MT, Bpk. Bambang Setiawan, S.Kom, M.Kom, Ibu Erma Suryani, S.Kom, MT, Bpk. Rully Agus, S.Kom, dan Ibu Nur Aini, S.Kom, atas segala bimbingan dan arahan selama ini
- Mas Bambang selaku administrator lab SI, terima kasih atas hak akses yang luar biasa
- Mas Kadir dan Mbak Anita, serta Pak Jarwo, yang selalu dan sering direpoti oleh penulis dan rekan-rekan
- Karyawan Sistem Informasi pada khususnya dan FTIF pada umumnya: Pak Yudi, Pak Narno, Bu Tutik, pak Cucuk, Mas Hermono, Mas Hari, Pak Muin, Pak Karmono, Pak Bagyo, Pak Soleh, Mas Sugeng, Mbak Eva, Wawan dkk.
- Keluarga penulis: Keluarga Mbak UM, keluarga Bulek IS, Keluarga Om Soha, Keluarga Bude semarang dan Keluarga Bude Wonosobo.
- Semua Bapak dan Ibu Guru kami yang tercinta.
- Para sahabat dan latih tanding: Mo Risca dan "keluarga", Om angga serta Samuel samsul.
- Para sahabat lain: Sopia Assuari, Mang bambang, bukan malaikat priza, cados Icha dan keluarga, Mirza "mabok", Mbet pitt.
- Para anak bimbing sedarah biru: Mbah titus, Cong anshar, Dit didit, Risa "Dora" dan Om Angga. Kapan kita dapat ekspresi "matanya harus bicara yang maut".
- Para yang membantu: Bagus atas koding yang luar biasa, Unta goft atas cara penulisan buku ini. Dan ucapan yang luar biasa buat sony "item" atas segalah baceman, bantuan maupun hutang "duit" yang tidak akan penulis bayar.
- Para pejuang Lab SI Poleng, Yeny, Dudut, Beny, Icha, Acong, Yudi, Diyah, Tante Indah, Titus. Kapan bisa ketemu dan berjuang bareng lagi

- Para pejuang-pejuang lain : Arin, Depta, Cemiit, Adhi Njoss, Ari, Om Angga, Kindi, Risa "Dora", Gesti, Mo KISI, Eka Anis Rhofita, Didit, Ratih "blukutuk", Mbet, Phipu, semoga Tuhan memberkati.
- Saudara 2001 yang laen Irfan, Njoss Sofyan, Gofs, Miko, Boz, Bagus, Reza, Bambang Mamang, Priza, Dimas, Ryan, Mirza, Pangestu. Wish You All the Best. We love 2001.
- Para angkatan 2002: Papa didit atas koding databasenya, actor bang joni atas kodingnya dan rekan 2002 lainnya.
- Para angkatan 2003: para admin zaza', TT harist, pretty dan rekan 2003 lainya
- Para pujangga cinta kost yang selalu "stress" Samuel samsul, Daniel pasarella, bondan si lumba-lumba dan habib, semoga kalian diterima disisi makhluk halus yang menjadi selomate.

Dan pihak-pihak yang mempunyai peranan penting dalam kehidupan penulis, yang tidak dapat disebutkan satu-persatu.

Tugas akhir ini membawa banyak harapan dan cita-cita. Dan semoga harapan tersebut dapat terwujud. Namun dengan tugas akhir ini penulis semakin menyadari keterbatasan sebagai seorang manusia biasa yang memiliki banyak kesalahan. Oleh karena itu penulis sangat beterima kasih jika ada saran maupun kritik yang membangun.

Surabaya Januari 2006

Penulis

DAFTAR ISI

DAFTAR ISI

HALAMAN JUDUL i
LEMBAR PENGESAHAN ii
ABSTRAK iii
KATA PENGANTAR iv
DAFTAR ISI v
DAFTAR GAMBAR x
DAFTAR TABEL xiii
DAFTAR PSEUDOCODE xv
DAFTAR SEGMEN PROGRAM xvi
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Tujuan dan manfaat 1
1.3 Permasalahan 2
1.4 Batasan masalah 2
1.5 Metodologi 2
1.6 Sistematika Laporan 3
BAB II TEORI PENUNJANG 5
2.1 Penggalian Data 5
2.2 Penggalian kaidah asosiasi 8
2.3 Pencarian *Frequent Itemset* 10
2.3.1 Algoritma *Apriori* 11
2.3.2 Algoritma *FP-growth* 14
2.3.2.1 Pembangkitan *FP-tree* 14
2.3.2.2 Pencarian *frequent itemset* menggunakan algoritma *FP-growth* 19
BAB III ALGORITMA BOMO 23
3.1 Pembangkitan Algoritma *BOMO* 23
3.2 Hasil Pola Kaidah Asosiasi 35
3.3 Metode Pencarian *n* k-*itemset* Paling Menarik 36
3.3.1 Urutan Pembangkitan dari *Conditional FP-tree* 37

3.3.2 Pendekatan *Loopback* 38
3.4 Peningkatan Kinerja Pencarian *n* k-*itemset* Paling Menarik 40
3.4.1 Pemangkasan *Conditional FP-Tree* 40
3.4.2 Pencarian Ukuran Batas Ambang Dukungan Minimum 41
BAB IV DESAIN & IMPLEMENTASI APLIKASI 42
4.1 Desain Aplikasi 42
4.1.1 Analisis *Use Case* 42
4.1.2 Realisasi *Use Case* 57
4.1.2.1 Kelas Utilitis 58
4.1.2.2 Diagram Sekuensi & *VOPC* 58
4.1.3 Diagram Kelas 70
4.1.4 Desain Antarmuka 70
4.2 Implementasi Aplikasi 75
4.2.1 Implementasi Basis Data 75
4.2.2 Implementasi Kelas 76
4.2.3 Implementasi Antarmuka 95
BAB V UJI COBA & ANALISIS 100
5.1 Lingkungan Uji Coba 100
5.2 Data Uji Coba 100
5.2.1 Karakteristik data 101
5.2.2 Pembuatan Data Uji Coba 102
5.3 Skenario uji coba 103
5.3.1 Uji Kebenaran Aplikasi 103
5.3.2 Uji Kinerja Aplikasi 103
5.4 Pelaksanaan Uji Coba dan Analisis Hasil 103
5.4.1 Uji Kebenaran Aplikasi 103
5.4.2 Uji Kinerja Aplikasi 107
BAB VI SIMPULAN 123
DAFTAR PUSTAKA 125
LAMPIRAN A DESKRIPSI PERANCANGAN DAN PEMBUATAN APLIKASI A-1

LAMPIRAN B SEGMEN PROGRAM B-1
LAMPIRAN C HASIL 3 BUAH 3-ITEMSET PALING MENARIK DARI APLIKASI UNTUK UJI KEBENARAN C-1
LAMPIRAN D HASIL POLA KAIDAH ASOSIASI DARI APLIKASI UNTUK UJI KEBENARAN D-1
LAMPIRAN E UJI KINERJA APLIKASI E-1

DAFTAR GAMBAR

DAFTAR GAMBAR

Gambar 2.1 Proses penggalian data 6
Gambar 2.2 Bentuk umum kaidah asosiasi 9
Gambar 2.3 Perhitungan dukungan dan keterpercayaan untuk $\{ I_1 \} \rightarrow \{ I_3 \}$ 10
Gambar 2.4 Dua transaksi dengan *prefix path* yang sama. 15
Gambar 2.5 Struktur item *prefix-subtree*. 15
Gambar 2.6 Proses pembangkitan *FP-tree* 18
Gambar 2.7 Hasil pembangkitan *FP-tree* dengan batas ambang dukungan minimum adalah 3 18
Gambar 2.8 conditional *FP-tree* item I_1 20
Gambar 2.9 *Conditional FP-tree* item $I_1(FP-tree_{I1})$ 21
Gambar 3.1 Inisialisasi *FP-tree* 27
Gambar 3.2 Proses pembangkitan *FP-tree* 29
Gambar 3.3 Hasil pembangkitan *FP-tree* 29
Gambar 3.4 Metode penelusuran dari akhir *FP-tree* 31
Gambar 3.5 *Conditional FP-Tree* untuk I_5 31
Gambar 3.6 *Conditional FP-Tree* untuk I_7 32
Gambar 3.7 *Conditional FP-Tree* item pada tabel *header* I_5 33
Gambar 3.8 *Conditional FP-Tree* untuk I_7 34
Gambar 3.9 Metode penelusuran dari awal *FP-tree* 37
Gambar 3.10 Metode penelusuran dari awal *FP-tree* 38
Gambar 3.11 Metode penelusuran dari tengah *FP-tree* 38
Gambar 3.12 *FP-tree* dengan $\xi = 4$ 38
Gambar 3.13 *FP-tree* dengan $\xi = 3$ 40
Gambar 4.1 Diagram *use case* aplikasi 44
Gambar 4.2 Diagram aktifitas lakukan uji koneksi basis data 47
Gambar 4.3 Diagram aktifitas lakukan pencarian *n* k-*itemset* paling menarik 50
Gambar 4.4 Diagram aktifitas lakukan pencarian pola kaidah asosiasi 53
Gambar 4.5 Diagram aktifitas lakukan pemantauan memori 54

Gambar 4.6 Diagram aktifitas lakukan penyimpanan data 56
Gambar 4.7 Diagram aktifitas lakukan permintaan informasi 57
Gambar 4.8 Diagram sekuensi lakukan uji koneksi 59
Gambar 4.9 Diagram *VOPC* lakukan uji koneksi. 59
Gambar 4.10 Diagram sekuensi lakukan pencarian *n* k-*itemset* paling menarik .. 61
Gambar 4.11 Diagram aktifitas dapatkan_jumlah_dan_item_terurut () 62
Gambar 4.12 Diagram aktifitas pembangkitan *FP-tree* 62
Gambar 4.13 Diagram aktifitas proses *top-down*(), *middle*() atau *bottom-up*() 63
Gambar 4.14 Diagram aktifitas method bangkitkan_FP-tree() 63
Gambar 4.15 Diagram aktifitas untuk method simpan_hasil() 64
Gambar 4.16 Diagram *VOPC* lakukan pencarian *n* k-*itemset* paling menarik..... 65
Gambar 4.17 Diagram sekuensi lakukan pencarian pola kaidah asosiasi 66
Gambar 4.18 Diagram *VOPC* lakukan pencarian pola kaidah asosiasi..... 67
Gambar 4.19 Diagram sekuensi lakukan pemantauan memori 67
Gambar 4.20 Diagram *VOPC* lakukan pemantauan memori 68
Gambar 4.21 Diagram sekuensi lakukan penyimpanan data 68
Gambar 4.22 Diagram *VOPC* lakukan penyimpanan data..... 69
Gambar 4.23 Diagram sekuensi lakukan permintaan informasi 69
Gambar 4.24 Diagram *VOPC* lakukan permintaan informasi..... 69
Gambar 4.25 Diagram kelas aplikasi 70
Gambar 4.26 Desain antarmuka aplikasi..... 71
Gambar 4.27 Desain antarmuka uji koneksi 72
Gambar 4.28 Desain antarmuka penggalian data..... 73
Gambar 4.29 Desain antarmuka pola kaidah asosiasi 74
Gambar 4.30 Desain antarmuka simpan data..... 75
Gambar 4.31 Antarmuka kerangka utama..... 95
Gambar 4.32 Menu navigasi aplikasi 95
Gambar 4.33 Antarmuka *use case* lakukan pemantauan memori. 96
Gambar 4.34 Antarmuka *use case* lakukan penyimpanan data..... 97
Gambar 4.35 Antarmuka *use case* lakukan uji koneksi 97
Gambar 4.36 Antarmuka *use case* pencarian *n* k-*itemset* paling menarik 98

Gambar 4.37 Antarmuka *use case* pencarian pola kaidah asosiasi 99
Gambar 4.38 Antarmuka *use case* lakukan permintaan informasi..... 99
Gambar 5.1 Pola kaidah asosiasi untuk hasil 3 3-*itemset* paling menarik menggunakan metode penelusuran dari akhir *FP-tree*..... 106
Gambar 5.2 Waktu komputasi proses 10 buah paling menarik pada T20I10D100K dengan urutan pembangkitan *conditional FP-tree*..... 108
Gambar 5.3 Waktu komputasi proses 10-*itemset* pada T20I10D100K dengan urutan pembangkitan *conditional FP-tree* 108
Gambar 5.4 Waktu komputasi proses 10 buah paling menarik pada T20I10D100K dengan pendekatan *loopback*..... 109
Gambar 5.5 Waktu komputasi proses 10-*itemset* pada T20I10D100K dengan pendekatan *loopback*..... 109
Gambar 5.6 Memori yang digunakan pendekatan *loopback* untuk menyimpan *FP-tree* dataset T20I10D100K 110
Gambar 5.7 Memori yang digunakan urutan pembangkitan *conditional FP-tree* untuk menyimpan *FP-tree* dataset T20I10D100K..... 110
Gambar 5.8 Waktu proses untuk berbagai nilai T..... 111
Gambar 5.9 Penggunaan memori untuk berbagai nilai T..... 112
Gambar 5.10 Waktu proses untuk berbagai nilai I..... 113
Gambar 5.11 Penggunaan memori untuk berbagai nilai I..... 114
Gambar 5.12 Waktu proses untuk berbagai nilai L..... 115
Gambar 5.13 Penggunaan memori untuk berbagai nilai L..... 116
Gambar 5.14 Waktu proses untuk berbagai nilai M..... 117
Gambar 5.15 Penggunaan memori untuk berbagai nilai M..... 118
Gambar 5.16 Waktu proses untuk berbagai nilai C..... 119
Gambar 5.17 Penggunaan memori untuk berbagai nilai C 120
Gambar 5.18 Waktu proses untuk berbagai nilai D 121
Gambar 5.19 Penggunaan memori untuk berbagai nilai D 122

DAFTAR TABEL

DAFTAR TABEL

Tabel 2.1 Data item 9
Tabel 2.2 Data transaksi 10
Tabel 2.3 *Frequent* 1-*itemset* (L_1) 11
Tabel 2.4 Pembangkitan kandidat 2-*itemset* (C_2) 12
Tabel 2.5 *Frequent* 2-*itemset* (L_2) 12
Tabel 2.6 Pembangkitan kandidat 3-*itemset* (C3) 13
Tabel 2.7 *Frequent* 3-*itemset* (L_3) 13
Tabel 2.8 Pembangkitan kandidat 4-*itemset* (C_4) 13
Tabel 2.9 *Frequent* 4-*itemset* (L_4) 13
Tabel 2.10 Item dan jumlah dukungan pada basis data 17
Tabel 2.11 Daftar *frequent* item yang terurut 17
Tabel 2.12 Himpunan item terurut pada transaksi 18
Tabel 2.13 *Frequent itemset* untuk item I_1 21
Tabel 2.14 *Conditional pattern base* dan *conditional FP-tree* setiap item 22
Tabel 2.15 *Frequent itemset* menggunakan *FP-growth* 22
Tabel 3.1 Data transaksi 25
Tabel 3.2 1-*itemset* dengan jumlah dukungan 26
Tabel 3.3 1-*itemset* terurut 26
Tabel 3.4 Himpunan item terurut pada setiap transaksi 27
Tabel 3.5 Hasil 3 3-*itemset* paling menarik sementara 32
Tabel 3.6 Hasil 3 3-*itemset* paling menarik pada item I_5 33
Tabel 3.7 Hasil 3 3-*itemset* paling menarik pada item I_7 34
Tabel 3.8 Hasil pencarian 3 3-*itemset* paling menarik 35
Tabel 3.9 ξ_K untuk $1 \leq k \leq 3$ 35
Tabel 3.10 Hasil pola kaidah asosiasi 35
Tabel 3.11 Hasil 2 2-*itemsets* paling menarik dengan $\xi = 4$. 39
Tabel 3.12 Hasil 2 2-*itemsets* paling menarik dengan $\xi = 3$. 39
Tabel 4.1 Daftar proses 43

MILIK PERPUSTAKAAN
ITS

Tabel 4.2 Tabel dokumentasi naratif *use case* lakukan uji koneksi 45
Tabel 4.3 Tabel dokumentasi naratif *use case* lakukan pencarian *n* k-*itemset* paling menarik 47
Tabel 4.4 Tabel dokumentasi naratif *use case* lakukan pencarian pola kaidah asosiasi 52
Tabel 4.5 Tabel dokumentasi naratif *use case* lakukan pemantauan memori 53
Tabel 4.6 Tabel dokumentasi naratif *use case* lakukan penyimpanan data 54
Tabel 4.7 Tabel dokumentasi naratif *use case* lakukan permintaan informasi 56
Tabel 4.8 Kelas utilitis beserta fungsinya 58
Tabel 4.9 Penjelasan method kelas `JFrame`*BOMO* 77
Tabel 4.10 Penjelasan method kelas connectDB 81
Tabel 4.11 Penjelasan method kelas `BOMO` 83
Tabel 4.12 Penjelasan method kelas `FP3` 90
Tabel 4.13 Keterangan atribut pada simpul 94
Tabel 4.14 Penjelasan Method kelas `Kitemsets`. 94
Tabel 4.15 Penjelasan *navigation menu* aplikasi 96
Tabel 5.1 Contoh data transaksi 101
Tabel 5.2 Rata-rata Ukuran *Itemset* 102
Tabel 5.3 Data 1 102
Tabel 5.4 Data transaksi dalam format "id_key #TID #Item" 104
Tabel 5.5 Hasil pencarian 3 3-*itemsets* paling menarik dengan aplikasi 105
Tabel 5.6 Hasil pencarian 3 3-*itemsets* paling menarik 105
Tabel 5.7 Hasil analisis terhadap berbagai nilai T 111
Tabel 5.8 Hasil analisis terhadap berbagai nilai I 113
Tabel 5.9 Hasil analisis terhadap berbagai nilai L 115
Tabel 5.10 Hasil analisis terhadap berbagai nilai M 117
Tabel 5.11 Hasil Analisis terhadap berbagai nilai C 119
Tabel 5.12 Hasil Analisis terhadap berbagai nilai D 121

DAFTAR PSEUDOCODE

DAFTAR PSEUDOCODE

Pseudocode 2.1 Algoritma pembangkitan *FP-tree* 16
Pseudocode 2.2 Algoritma *FP-growth* 19
Pseudocode 3.1 Algoritma *BOMO*. 25
Pseudocode 3.2 Tahap pembangkitan *FP-tree* 28
Pseudocode 3.3 Tahap penggalian algoritma *BOMO* 30

DAFTAR SEGMEN PROGRAM

DAFTAR SEGMEN PROGRAM

Segmen program 4.1 Method `JButtonTestConnectActionPerformed()` 78
Segmen program 4.2 Kelas `LoadProcess` 79
Segmen program 4.3 Kelas `assocProcess` 80
Segmen program 4.4 Method `JButtonTestConnect1MouseClicked()` 80
Segmen program 4.5 Method `connect()` 82
Segmen program 4.6 Method `connectOracle()` 82
Segmen program 4.7 Method `connectMysql()` 82
Segmen program 4.8 Method `countItemOccurrences()` 84
Segmen program 4.9 Method `ScanRec()` 85
Segmen program 4.10 Method `eksekusi(String,double)` 86
Segmen program 4.11 Method `eksekusi(String, String)` 87
Segmen program 4.12 Method `associationRule()` 88
Segmen program 4.13 Method `top_down()` 91
Segmen program 4.14 Method `middle()` 92
Segmen program 4.15 Method `Bottom_up()` 92
Segmen program 4.16 Method `setSupportAssociation()` 92

BAB I
PENDAHULUAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pada pola kaidah assosiasi sebuah batas ambang dukungan minimum biasanya diasumsikan telah diberi nilai sehingga mudah untuk mencari kumpulan item yang memiliki jumlah minimum sama dengan batas ambang dukungan minimum (*frequent itemset*). Dalam prakteknya sulit memberikan nilai yang tepat untuk batas ambang dukungan yang sesuai dengan hasil yang dinginkan. Hal ini dikarenakan apabila nilai terlalu besar maka kemungkinan akan mendapatkan sedikit hasil ataupun tidak sama sekali. Dan apabila nilai terlalu kecil akan mendapatkan banyak hasil yang tidak memiliki arti dan biaya *komputasional* yang mahal.

Penentuan batas ambang dukungan minimum akan berbeda untuk setiap ukuran dan jenis data yang berbeda pula, sehingga penggunaan nilai batas ambang dukungan tidak boleh sama (*unifrom*). Dimana nilai batas ambang dukungan untuk data berukuran besar memiliki kecenderungan nilai lebih kecil daripada nilai batas ambang dukungan untuk data berukuran kecil.

Berbagai masalah tersebut menyebabkan kesulitan untuk mencari *frequent itemsets*. Berbagai algoritma banyak diterapkan untuk mencari *frequent itemsets* tanpa memberikan nilai batas ambang dukungan, salah satu algoritma yang diimplementasikan yaitu algoritma *BOMO* (*Build-Once and Mine-Once*).

1.2 Tujuan dan manfaat

Tujuan tugas akhir ini adalah diimplementasikan algoritma *BOMO* dan melakukan uji coba terhadap beberapa data transaksi buatan. Sehingga adapun manfaat yang dapat diperoleh yaitu didapatkan *itemset* yang diinginkan tanpa memberikan nilai batas ambang dukungan minimum serta pola kaidah asosiasi dari *itemset* tersebut.

1.3 Permasalahan

a. Bagaimana cara mengimplementasikan algoritma *BOMO* kedalam sebuah aplikasi yang mampu mencari *frequent itemsets* tanpa memberikan nilai batas ambang dukungan minimum?

b. Bagaimana membangkitkan data uji coba dan menggunakan didalam uji coba aplikasi?

1.4 Batasan masalah

Dari permasalahan yang telah disebutkan di atas, maka batasan-batasan dalam tugas akhir ini adalah:

a. Data percobaan merupakan data buatan hasil pembangkitan.

b. Implementasi akan dilakukan dalam lingkungan sistem operasi berbasis windows dengan menggunakan bahasa pemograman berbasis java.

1.5 Metodologi

Penyusunan tugas akhir ini dibagi menjadi beberapa tahapan sebagai berikut :

(a) Studi kepustakaan

Aktivitas yang dilakukan pada tahapan ini adalah mengumpulkan dan mempelajari berbagai literatur yang terkait dengan proses penyusunan sistem ini, antara lain yang berkenaan dengan algoritma *BOMO, FP-Tree*, pola kaidah asosiasi dan berbagai literatur lain yang masih dalam lingkup permasalahan tugas akhir ini.

(b) Desain aplikasi

Aktivitas yang dilakukan pada tahapan ini adalah menyusun rincian dari pembuatan aplikasi untuk mengimplementasikan algoritma *BOMO*. Serta merencanakan kebutuhan pengguna dan kebutuhan sistem yang dibutuhkan dalam membuat aplikasi.

(c) Pembuatan aplikasi

Aktivitas yang dilakukan pada tahapan ini adalah membangun sebuah aplikasi yang mengimplementasikan algoritma *BOMO* untuk mencari *frequent itemsets* tanpa memberikan nilai batas ambang dukungan minimum dengan bahasa pemograman java

(d) Uji coba dan evaluasi

Aktivitas yang dilakukan pada tahapan ini adalah menguji dan mengevaluasi aplikasi yang telah dibuat. Proses uji coba aplikasi menggunakan data kecil dan data besar.

(e) Penyusunan buku tugas akhir

Aktivitas yang dilakukan pada tahapan ini adalah menyusun dokumentasi yang menjelaskan dasar teori dan metode yang digunakan serta hasil yang telah dilakukan. Dokumen ini diharapkan dapat berguna bagi pembaca yang memiliki keinginan untuk mengkaji lebih lanjut atau untuk keperluan pengembangan aplikasi yang lebih baik.

1.6 Sistematika Laporan

Sistematika penulisan laporan tugas akhir akan dibagi menjadi enam bab. Pada bab pertama pendahuluan dijelaskan tentang latar belakang, tujuan, permasalahan, batasan masalah serta metodologi penyusunan buku tugas akhir.

Pada bab kedua teori penunjang dijelaskan tentang teori-teori yang mendukung pembuatan tugas akhir ini. Bab ini berisi pengertian tentang konsep dasar penggalian data, kaidah asosiasi dan algoritma lain yang digunakan dalam pencarian *frequent itemset* untuk mendapatkan pola kaidah asosiasi.

Bab ketiga algoritma *BOMO* dijelaskan secara khusus tentang algoritma ini. Bab ini berisi tahapan untuk pencarian *itemset* menggunakan algoritma *BOMO* berdasarkan deskripsi dari paper Ying-Ling Cheung dan Ada Wai-Chee Fu [YLC-04]. Bab ini juga menunjukkan beberapa metode untuk meningkatkan kinerja algoritma *BOMO*.

Bab keempat desain dan implementasi aplikasi dijelaskan rancangan aplikasi yang dikembangkan. Pada desain berisi penjelasan proses yang terjadi pada aplikasi, sedangkan implementasi berisi program dari aplikasi dan tampilan antarmuka.

Bab kelima uji coba dan analisis dijelaskan tentang uji kebenaran dan kinerja dari aplikasi yang dikembangka. Pada bab ini berisi proses uji coba mulai dari pembuatan dataset, skenario uji coba, pelaksanaan uji coba dan analisis dari pelaksanaan uji coba.

Pada bab akhir simpulan dan saran berisi kesimpulan yang dapat diambil dari pelaksanaan tugas akhir. Pada bab ini diharapkan didapatkan saran untuk pengembangan selanjutnya

BAB II
TEORI PENUNJANG

BAB II

TEORI PENUNJANG

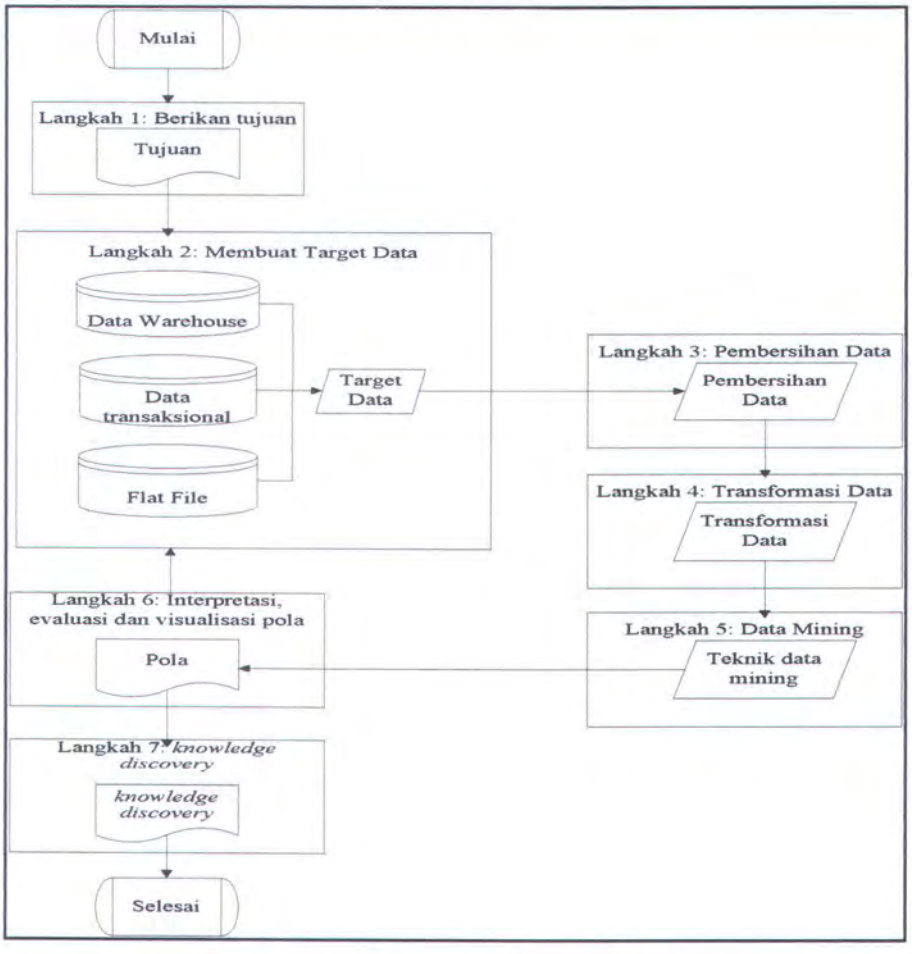
Istilah penggalian data digunakan secara khusus dalam proses pencarian pengetahuan pada basis data, yang selanjutnya disebut sebagai *knowledge discovery*. Berbagai latar belakang dibutuhkannya penggalian data, seperti berbagai teknik tradisional yang tidak fisibel untuk mengolah data mentah, meledaknya volume data yang dihimpun atau disimpan dalam gudang data serta adanya proses komputasi yang dapat diupayakan. Dalam penggalian data terdapat beberapa teknik berbeda yang digunakan untuk tujuan berbeda pula. Oleh karena itu, pada bab ini, selain dijelaskan mengenai penggalian data secara garis besar juga akan dijelaskan mengenai beberapa teknik penggalian data secara umum.

2.1 Penggalian Data

Penggalian data (*data mining*) merupakan sebuah proses ekstrasi informasi yang potensial, implisit, dan tidak diketahui sebelumnya dengan tujuan mendapatkan pengetahuan pada basis data, sebagai contoh aturan-aturan, batasan-batasan dan regularitas dari sekumpulan data pada basis data besar [CHE-96]. Untuk mendapatkan pengetahuan pada basis data maka diperlukan proses seperti terlihat pada gambar 2.1.

Proses penggalian data seperti ditunjukkan gambar 2.1, diawali dengan memberikan tujuan proses penggalian data dengan memahami domain dari aplikasi yang berupa pengetahuan awal dan sasaran pengguna. Pada tahap kedua, membuat target data berupa pemilihan data dan fokus pada beberapa bagian data tertentu. Pada tahap ketiga dan keempat, dilakukan pembersihan dan transformasi data, berupa penghilangan data derau, *outliers* dan data tidak lengkap. Pada tahap kelima, proses penggalian data menggunakan teknik-teknik penggalian data seperti pencarian kaidah asosiasi (*association rule mining*), pencarian pola sekuensial (*sequences rule mining*), klassifikasi dan regresi (*classification and prediction*), klasterisasi (*clustering*) dan lain-lain. Pada tahap keenam didapatkan

interpretasi, evaluasi dan visualisasi pola sebagai hasil proses data mining dan pada tahap ketujuh digunakan sebagai pengetahuan basis data.



Gambar 2.1 Proses penggalian data

Pada tahap kelima terdapat beberapa teknik yang dapat digunakan dalam penggalian data. Berikut ini beberapa teknik penggalian data, antara lain:

(a) Pencarian kaidah asosiasi

Mendeteksi kumpulan-kumpulan attribut yang muncul bersamaan (*co-occur*) dalam frekuensi yang sering, dan membentuk sejumlah kaidah dari kumpulan-kumpulan tersebut. Sebagai contoh 90% dari orang yang berbelanja di suatu supermarket yang membeli roti juga membeli selai, dan 60% dari semua orang yang berbelanja membeli keduanya.

(b) Pencarian pola sekuensial

Mencari urutan sejumlah kejadian yang secara umum terjadi bersama–sama dalam satu waktu tertentu. Sebagai contoh dalam satu kumpulan urutan-urutan DNA, ACGTC diikuti oleh GTCA setelah sebuah celah dengan probabilitas sebesar 30%.

(c) Klassifikasi dan Regresi

Menentukan sebuah baris data baru ke salah satu dari beberapa kategori atau kelas yang telah didefinisikan sebelumnya. Regresi berkaitan dengan prediksi kolom yang bernilai nyata. Keduanya disebut juga sebagai *supervised learning*.

(d) Klasterisasi

Membagi sekelompok data menjadi beberapa bagian (*subset*) atau beberapa kelompok data yang sedemikian rupa, sehingga elemen-elemen dari suatu kelompok tertentu memiliki bagian properti yang digunakan bersama. Pembagian sekelompok data memiliki tingkat similaritas yang tinggi dalam satu kelompok dan tingkat similaritas antar kelompok yang rendah. Klasterisasi disebut juga sebagai *unsupervised learning*.

(e) Pelacakan Similaritas

Untuk suatu basis data dari sejumlah objek atau sebuah *query* terhadap objek yang diberikan, maka didapatkan objek-objek yang berada dalam jarak yang ditentukan pengguna. Sehingga akan dicari kesamaan objek berdasarkan jarak yang ditentukan pengguna.

(f) Deteksi deviasi

Pencarian baris yang yang paling berbeda dari baris lainnya atau dengan kata lain disebut sebagai *outliers*. *Outliers* ini dapat diabaikan sebagai data yang salah atau mungkin merupakan informasi yang "menarik" yang dapat digunakan bagi pengguna.

Berdasarkan penjelasan diatas maka penggalian data dapat dibedakan menjadi dua bagian, yaitu: *supervised* dan *unsupervised data mining*. *Supervised data mining* berarti kemungkinan hasil dari proses penggalian data telah ditentukan. Sebaliknya *unsupervised data mining* berarti kemungkinan hasil dari proses penggalian data tidak ditentukan. Contoh s*upervised data mining* yaitu klassifikasi dan regresi. Sedangkan *unsupervised* data mining terdiri atas: penggalian kaidah asosiasi dan klasterisasi. Penggalian data yang digunakan untuk menyusun tugas akhir ini adalah penggalian kaidah asosiasi karena bertujuan. menghasilkan pola kaidah asosiasi.

2.2 Penggalian kaidah asosiasi

Definisi dari penggalian kaidah asosiasi adalah menemukan sekumpulan item yang dimiliki bersama-sama oleh sejumlah objek dalam sebuah basis data. Pencarian kaidah asosiasi mengunakan dua buah parameter nilai yaitu dukungan (*support*) dan keterpercayaan (*confidence*) yang memiliki nilai antara 0% - 100 %. Berikut sedikit penjelasan mengenai dukungan dan keterpercayaan.

Sebagai contoh terdapat relasi I berisi sejumlah kumpulan item yang kemudian dikatakan sebagai *itemset*, dimana masing–masing *itemset* terdiri dari sekumpulan atribute bertipe boolean $I_1, I_2, \ldots, I_n$. Dan basis data transaksi D yang berisi transaksi T, adalah himpunan dari I atau $T \subseteq I$ .Dimana transaksi T pada basis data transaksi D memiliki sebuah atribut yang unik yang dinotasikan dengan TID. Dalam konteks ini, A dan B merupakan *itemset* dari transaksi T, jika dan hanya jika $A \subseteq T$ dan $B \subseteq T$. Sehingga jumlah A dinotasikan $\sigma(A)$ merupakan jumlah dukungan (*support count*) *itemset* A yang terdapat pada basis data transaksi D.

Kaidah asosiasi A→B, jika dan hanya jika $A \subset I$, $B \subset I$ dan $A \cap B \neq 0$. Sehingga A→B memiliki dukungan S pada transaksi T, dimana S merupakan persentase *itemset* $A \cup B$ pada basis data transaksi D. Dan A→B memiliki keterpercayaan C pada transaksi T, dimana C merupakan persentase jumlah *itemset* A yang terdapat pada relasi I, yang diikuti *itemset* B . Berikut persamaan

2.1 untuk menentukan dukungan A→B dan persamaan 2.2 untuk menentukan keterpercayaan A→B pada kaidah asosiasi.

$$\text{Dukungan}(A \rightarrow B) = P(A \cup B) \tag{2.1}$$

$$\text{Keterpercayaan}\ (A \rightarrow B) = P(A \mid B) \tag{2.2}$$

Berdasarkan persamaan 2.1 dan 2.2 maka persentase dukungan didapatkan dengan membagi jumlah dukungan untuk A dan B dengan jumlah transaksi ($\sigma(A \cup B) / |T|$), Sedangkan persentase keterpercayaan didapatkan dengan membagi jumlah dukungan untuk A dan B dengan jumlah dukungan untuk A ($\sigma(A \cup B) / \sigma(A)$).

Kaidah asosiasi berbentuk A→B maka kaidah asosiasi tersebut terdiri dari dua bagian, yaitu : *antecedent* dan *consequent*. Bentuk umum dari kaidah asosiasi itu adalah sebagai berikut

If *antecedent* **then** *consequent*

atau

If A **then** B (A **dan** B **adalah** *frequent itemset*)

Gambar 2.2 Bentuk umum kaidah asosiasi

Dibawah ini adalah contoh yang dapat digunakan untuk menjelaskan dukungan dan keterpercayaan pada kaidah asosiasi. Tabel 2.1 merupakan tabel data item dengan kode untuk masing–masing jenis obat pada apotik. Tabel 2.2 berisikan data transaksi yang terjadi pada suatu apotik.

Tabel 2.1 Data item

Nama Item	Kode
AbsorbAid	I_1
aCELLeration Gel	I_2
Acetyl-L-Carnitine	I_3
Acid Ampules	I_4
Acid Defense	I_5
Acid Eye Creme	I_6
Acid Hydrating	I_7
Acid Serum	I_8

MILIK PERPUSTAKAAN ITS

Tabel 2.2 Data transaksi

TID	Item
1	I_4, I_1, I_3, I_6, I_8
2	I_1, I_2, I_3, I_4, I_6
3	I_2, I_4, I_5, I_7
4	I_2, I_3, I_7, I_8
5	I_1, I_4, I_3, I_5, I_8, I_6

Bila { I_1 }→{ I_3 } adalah kaidah asosiasi, maka dapat dalam konteks ini membeli obat (x, "I_1") → membeli obat (x, "I_3") [S = 40%, C = 67%]. . Gambar 2.2 merupakan hasil perhitungan dukungan dan keterpercayaan untuk { I_1 }→{ I_3 }.

$$Support(I_1 \rightarrow I_3) = \frac{\sigma(I_1, I_3)}{|T|} \qquad Confidence(I_1 \rightarrow I_3) = \frac{\sigma(I_1, I_3)}{\sigma(I_1)}$$

$$Support(I_1 \rightarrow I_3) = \frac{2}{5} = 40\% \qquad Confidence(I_1 \rightarrow I_3) = \frac{2}{3} = 67\%$$

Gambar 2.3 Perhitungan dukungan dan keterpercayaan untuk { I_1 }→{ I_3 }.

Tujuan dari kaidah asosiasi adalah menemukan aturan relasi *itemset* yaitu *frequent itemset*. Kaidah asosiasi yang valid memiliki nilai dukungan lebih besar sama dengan nilai batas ambang dukungan minimum (*minimum support threshold* atau *minsupport threshold)* dan nilai keterpercayaan lebih besar sama dengan nilai batas ambang keterpercayaan minimum (*minimum confidence threshold* atau *minconfidence threshold)*

Dikatakan *frequent itemset* jika *itemset* memiliki jumlah dukungan yang sama atau lebih besar sama dengan nilai batas ambang dukungan minimum. Untuk menghasilkan pola kaidah asosiasi maka harus dicari terlebih dahulu *frequent itemset* dari data transaksi.

2.3 Pencarian *Frequent Itemset*

Pencarian *frequent itemset* dapat menggunakan berbagai algoritma diantaranya algoritma *Apriori* dan algoritma *FP-growth*. Kedua algoritma ini akan dibahas pada sub-bab berikut ini.

2.3.1 Algoritma *Apriori*

Algoritma *apriori* merupakan salah satu algoritma yang terbukti efektif untuk pencarian *frequent itemset*. Hal ini terbukti dengan banyaknya variasi algoritma yang dikembangkan berdasarkan konsep awal dari algoritma *apriori* Algoritma *apriori* untuk mencari *frequent itemset* dengan membangkitkan semua kemungkinan *itemset* atau pembangkitan kandidat (*candidate generation*).. Berikut ini sedikit penjelasan tentang proses algoritma *apriori* berdasarkan pada Agrawal dan Srikant [AGS-93].

Dikatakan k-*itemset* jika *itemset* memiliki himpunan item dengan jumlah k, sehingga notasi L_k merupakan *frequent* k-*itemset* dan notasi C_k merupakan pembangkitan kandidat untuk k-*itemset*. Nilai C_k merupakan hasil penggabungan $L_{(k-1)}$ dengan $L_{(k-1)}$ ($L_{(k-1)} \cup L_{(k-1)}$). Didapatkan L_k jika C_k memiliki jumlah lebih besar sama dengan nilai batas ambang dukungan minimum.

Dibawah ini adalah contoh untuk menjelaskan algoritma *apriori*. Misalnya ditentukan nilai batas ambang dukungan minimum data transaksi pada tabel 2.2 adalah 3, maka hasil L_1 adalah 1-*itemset* yang memiliki jumlah lebih besar sama dengan 3. Hasil *frequent* 1-*itemset* ditunjukkan pada table 2.3.

Tabel 2.3 *Frequent* 1-*itemset* (L_1)

1-*itemset*	Jumlah Dukungan
I_3	4
I_4	4
I_1	3
I_2	3
I_6	3
I_8	3

Tahap selanjutnya dilakukan pembangkitan kandidat untuk 2-*itemset* (C_2) yaitu dengan menggabungkan 1-*itemset* yang *frequent* dengan 1-*itemset* yang *frequent* ($L_1 \cup L_1$). Hasil pembangkitan kandidat untuk 2-*itemset* (C_2) seperti yang ditunjukkan pada tabel 2.4, sedangkan kolom jumlah dukungan merupakan jumlah 2-*itemset* dalam data transaksi tabel 2.2.

Tabel 2.4 Pembangkitan kandidat 2-*itemset* (C_2)

2-*itemset*	Jumlah Dukungan
I_3, I_4	3
I_3, I_1	3
I_3, I_2	2
I_3, I_6	3
I_3, I_8	2
I_4, I_1	3
I_4, I_2	2
I_4, I_6	3
I_4, I_8	3
I_1, I_2	1
I_1, I_6	3
I_1, I_8	2
I_2, I_6	1
I_2, I_8	1
I_6, I_8	2

Pada tahap selanjutnya dicari *frequent* 2-*itemset* berdasarkan hasil pembangkitan kandidat 2-*itemset* (C_2) pada tabel 2.4. *Frequent* 2-*itemset* merupakan C_2 yang memiliki jumlah lebih besar dari nilai batas ambang dukungan minimum ($C_2 \geq 3$). *Frequent* 2-*itemset* ditunjukkan pada tabel 2.5.

Tabel 2.5 *Frequent* 2-*itemset* (L_2)

2-*itemset*	Jumlah Dukungan
I_3, I_4	3
I_3, I_1	3
I_3, I_6	3
I_4, I_1	3
I_4, I_6	3
I_4, I_8	3
I_1, I_6	3

Setelah beberapa interasi dengan menggunakan proses yang sama seperti pada pencarian *frequent* 2-*itemset*, maka pada tahap selanjutnya dihasilkan *frequent* 3-*itemset* (L_3) dan *frequent* 4-*itemset* (L_4). Untuk lebih memperjelas proses yang terjadi, maka pembangkitan kandidat 3-*itemset* (C_3) ditunjukkan pada tabel 2.6, dan *frequent* 3-*itemset* ditunjukkan pada tabel 2.7. Pada tabel 2.8

merupakan pembangkitan kandidat 4-*itemset* yang merupakan hasil penggabungan L_3 dengan L_3, dan *frequent* 4-*itemset* (L_4) ditunjukkan pada table 2.9.

Tabel 2.6 Pembangkitan kandidat 3-*itemset* (C3)

3-*itemset*	Jumlah Dukungan
I_3, I_4, I_1	3
I_3, I_4, I_6	3
I_3, I_4, I_8	2
I_3, I_1, I_6	3
I_4, I_1, I_6	3
I_4, I_1, I_8	2
I_4, I_6, I_8	2

Tabel 2.7 *Frequent* 3-*itemset* (L_3)

3-*itemset*	Jumlah Dukungan
I_3, I_4, I_1	3
I_3, I_4, I_6	3
I_3, I_1, I_6	3
I_4, I_1, I_6	3

Tabel 2.8 Pembangkitan kandidat 4-*itemset* (C_4)

4-*itemset*	Jumlah Dukungan
I_3, I_4, I_1, I_6	3

Tabel 2.9 *Frequent* 4-*itemset* (L_4)

4-*itemset*	Jumlah Dukungan
I_3, I_4, I_1, I_6	3

Proses interasi pencarian *frequent itemset* berhenti setelah terbentuk L_4 karena pembentukan pembangkitan kandidat 5-*itemset* tidak menghasilkan nilai ($C_5=\phi$). Sehingga L_1, L_2, L_3 dan L_4 merupakan *frequent itemset* yang ditemukan dengan menggunakan algoritma *apriori*.

Penggunaan pembangkitan kandidat pada algoritma *apriori* menyebabkan pembacaan berulang kali terhadap basis data serta pembangkitan kandidat dapat memiliki ukuran yang sangat besar. Berdasarkan latar belakang tersebut

dibutuhkan algoritma pencarian *frequent itemset* yang lebih baik, salah satu algoritma yang dikembangkan adalah algoritma *FP-growth.*

2.3.2 Algoritma *FP-growth*

Algoritma *Frequent Pattern-growth* (*FP-growth*) merupakan algoritma untuk mendapatkan *frequent itemset* tanpa menggunakan pembangkitan kandidat. Pada algoritma *FP-growth* ini penggalian *frequent itemset* dilakukan dengan membangkitan struktur data pohon yang disebut *Frequent Pattern tree* (*FP-tree*). Dengan menggunakan pembangkitan *FP-tree*, maka urutan data transaksi dapat dicari tanpa melakukan kueri terhadap basis data, hal ini yang menyebabkan algoritma *FP-growth* tidak melakukan pembacaan berurang kali terhadap basis data. Untuk lebih memudahkan pemahaman, maka pembangkitan *FP-tree* akan dijelaskan terlebih dahulu, kemudian pada sub-bab berikutnya akan dijelaskan mengenai proses pencarian *frequent itemset* menggunakan algoritma *FP-growth* berdasarkan pada Han et all [JWH-00].

2.3.2.1 Pembangkitan *FP-tree*

Frequent pattern tree (*FP-tree*) merupakan variasi struktur data pohon, dimana menggunakan struktur *prefix-tree* untuk menyimpan informasi *frequent pattern* yang penting dan padat [YLC-04]. *FP-tree* memiliki sebuah simpul akar (*root node*) dengan nilai "null", beberapa kumpulan item *prefix-subtree* sebagai simpul anak (*child node*) dari simpul akar dan sebuah tabel *header* untuk menyimpan item yang *frequent*.

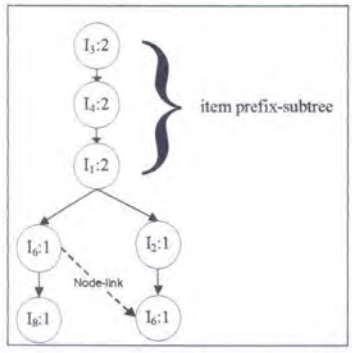
Struktur item *prefix-subtree* merupakan bagian pohon (*subtree*) yang menjadi *prefix path* (himpunan item terurut yang mengawali k-*itemsets* dan dimulai dari simpul akar), dari *suffix pattern* (k-*itemsets*). Sebagai contoh I_4, I_1, I_3, I_6, I_8 merupakan item terurut dari besar ke kecil berdasarkan jumlah dukungan, jika I_8 merupakan *suffix pattern* maka dapat dikatakan I_4, I_1, I_3 , I_6 merupakan *prefix paths* dari simpul I_8.

Sebagai contoh struktur item *prefix-subtree* apabila dua data transaksi (ditunjukkan pada gambar 2.4) memiliki *prefix path* yang sama yaitu I_3, I_4, I_1, I_6,

maka himpunan item tersebut akan digunakan sebagai I_3, I_4, I_1, I_6 *prefix-subtree* (*item-item prefix-subtree*) dengan mencatat jumlah dukungan setiap item adalah 2 untuk memperjelas mengenai struktur item *prefix-subtree* seperti ditunjukkan pada gambar 2.5.

TID:1 → I_3, I_4, I_1, I_6, I_8
TID:2 → I_3, I_4, I_1, I_6, I_2

Gambar 2.4 Dua transaksi dengan *prefix path* yang sama.



Gambar 2.5 Struktur item *prefix-subtree*.

Setiap simpul pada pohon memiliki tiga atribut yaitu item (*item_name*), jumlah (*count*) dan kaitan simpul (*node-link*). Item merupakan nama item yang berada pada simpul tersebut, jumlah menyatakan jumlah item pada simpul tersebut sedangkan kaitan simpul digunakan untuk mengetahui letak item yang sama pada pohon atau bernilai "null" apabila tidak ada, hal ini karena item-item yang sama tersebar pada seluruh pohon. Sebuah tabel *header* dua komponen yaitu item yang *frequent* dan simpul kepala dari kaitan simpul (*head of node-link*) pada pohon.

Beberapa keuntungan dalam penggunaan struktur *FP-tree* antara lain menyimpan informasi lengkap untuk proses penggalian *frequent itemset* yang dilakukan dengan membagi pada bagian yang lebih kecil (*divide-and-conguer methodology*) dan juga menghindar pembacaan berulang kali terhadap basis data. *FP-tree* memiliki ukuran yang tidak pernah lebih besar dari basis data transaksi karena item-item yang tidak *frequent* telah dibuang, sedangkan item-item yang *frequent* disimpan dalam urutan yang dapat dipakai oleh banyak transaksi dalam struktur item *prefix-subtree*.

Seperti ditunjukkan gambar 2.5 algoritma pembangkitan *FP-Tree* [JWH-00] melibatkan dua tahapan utama, yaitu (1) cari daftar *frequent* item yang terurut, (2) pembangkitan *FP-Tree*. Sehingga pembacaan terhadap basis data hanya dilakukan dua kali.

```
Algoritma Pembangkitan FP-tree
Masukkan: Data transaksi dan batas ambang dukungan
minimum
Keluaran : FP-tree dan frequent pattern tree

Method : Pembangkitan FP-tree
(1) Baca basis data satu kali untuk mendapatkan
    frequent item F beserta jumlah yang dimiliki.
    Urutkan F dari besar ke kecil berdasarkan jumlah
    yang dimiliki, simpan sebagai Flist (Daftar
    frequent item yang terurut)
(2) Buat simpul root dari FP-tree T dengan nilai
    "null", Untuk setiap transaksi Trans pada basis
    data lakukan:
    Dapatkan item yang frequent pada Trans dan urutkan
    berdasarkan Flist kemudian notasikan dengan [p|P],
    dimana p adalah elemen pertama sedangkan P adalah
    urutan berikutnya. maka masukkan [p|P] kedalam FP-
    tree T dengan menggunakan Insert_NFPtree([p|P], T)

Insert_NFPtree([p|P], T)
(1) Jika FP-tree T memiliki simpul child N dimana
    N.nama_item = p. nama_item maka jumlah item N
    ditambah 1.
    jika tidak maka buat simpul baru untuk item N
    dengan jumlah 1, dimana memiliki simpul item N
    memiliki hubungan simpul parent terhadap T dan
    memiliki hubungan simpul terhadap simpul item N
    pada T (struktur node link)
(2) jika itemset P belum kosong maka lakukan
    Insert_NFPtree(P, N) secara rekrusif.
```

Pseudocode 2.1 Algoritma pembangkitan *FP-tree*

Berikut ini adalah contoh yang dapat digunakan untuk menjelaskan pembangkitan *FP-tree*. Misalnya ditentukan batas ambang dukungan minimum untuk tabel 2.2 adalah 3. Pada tahap (1) cari daftar *frequent* item yang terurut diawali dengan pembacaan pada basis data transaksi untuk mendapatkan item beserta jumlah dukungan. Hasil proses ini ditunjukkan pada tabel 2.10.

Tabel 2.10 Item dan jumlah dukungan pada basis data

Item	Jumlah Dukungan
I_1	3
I_2	3
I_3	4
I_4	4
I_5	2
I_6	3
I_7	2
I_8	3

Kemudian dilakukan pengurutan item-item dari besar ke kecil berdasarkan jumlah dukungan yang dimiliki, dan menghapus item-item yang memiliki jumlah dukungan kurang dari 3 (item yang tidak *frequent*). Hasil tahap cari daftar *frequent* item yang terurut seperti terlihat pada table 2.11.

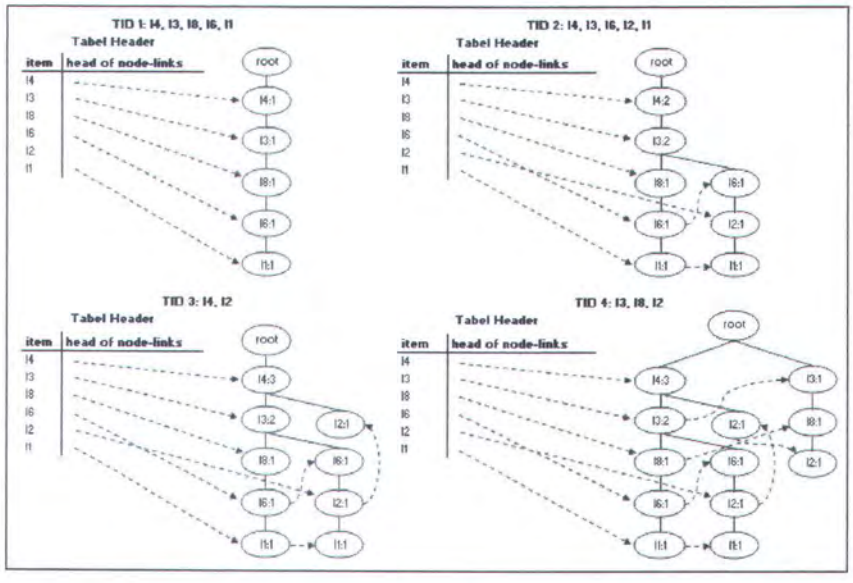
Tabel 2.11 Daftar *frequent* item yang terurut

Item	Jumlah Dukungan
I_4	4
I_3	4
I_8	3
I_6	3
I_2	3
I_1	3

Pada tahap (2) pembangkitan *FP-tree* diawali dengan melakukan pembacaan terhadap basis data untuk setiap transaksi yang ada. Kemudian pada setiap transaksi, item-item yang memiliki jumlah lebih kecil dari nilai batas ambang dukungan minimum dihapus atau tidak dimasukkan kedalam pohon yang telah diinisialisasi, sedangkan item-item yang *frequent* diurutkan dari besar ke kecil berdasarkan jumlah yang dimiliki. Himpunan item setiap transaksi yang telah terurut dari besar ke kecil (ditunjukkan pada tabel 2.12) akan dimasukan ke dalam pohon, proses ini ditunjukkan pada gambar 2.6 dan gambar 2.7.

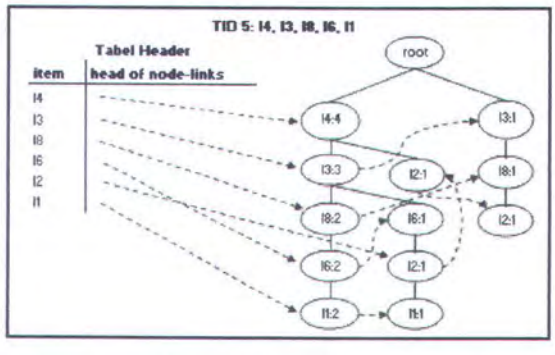
Tabel 2.12 Himpunan item terurut pada transaksi

TID	Item terurut
1	I_3, I_4, I_1, I_6, I_8
2	I_3, I_4, I_1, I_2, I_6
3	I_4, I_2
4	I_3, I_2, I_8
5	I_3, I_4, I_1, I_6, I_8



Gambar 2.6 Proses pembangkitan *FP-tree*

Secara keseluruhan proses pembangkitan *FP-tree* akan menghasilkan sebuah pohon dan sebuah tabel *header* yang memiliki kaitan simpul. Hasil akhir proses pembangkitan *FP-tree* seperti yang ditunjukkan pada gambar 2.7, dimana *FP-tree* dengan nilai batas ambang dukungan minimum sama dengan 3.



Gambar 2.7 Hasil pembangkitan *FP-tree* dengan batas ambang dukungan minimum adalah 3

2.3.2.2 Pencarian *frequent itemset* menggunakan algoritma *FP-growth*

FP-growth merupakan algoritma untuk mendapatkan *frequent itemset* tanpa menggunakan pembangkitan kandidat yang dikembangkan oleh Jiawei Han [JWH-00]. Algoritma ini dikembangkan untuk menghilangkan biaya komputasional pembangkitan kandidat. Untuk lebih jelasnya proses algoritma *FP-growth* ditunjukkan pada pseudocode 2.2.

```
Algoritma FP-growth
Masukkan : FP-tree dan batas ambang dukungan minimum
Keluaran : itemset yang frequent
Method : Panggil fungsi FP-growth(FP-tree,null)

Procedure FP-growth(FP-tree,frequent itemset  ){
(1) jika pohon memiliki satu cabang P
(2)     maka untuk setiap kombinasi(dinotasikan β)
        dari simpul pada P lakukan:
(3)         dapatkan urutan item β∪α dengan dukungan
            = dukungan minimum simpul di β
(4) jika tidak untuk setiap α_i pada tabel header
    lakukan {
(5)     dapatkan urutan item β=α_i ∪α dengan dukungan =
        α_i .dukungan
(6)     dapatkan conditional pattern base untuk β dan
        bangkitkan conditional FP-tree untuk
        β (FP-tree_β);
(7)     jika FP-tree_β ≠ φ
(8)         Panggil fungsi FP-growth(FP-tree_β, β)
}
```

Pseudocode 2.2 Algoritma *FP-growth*

Berikut adalah contoh yang dapat digunakan untuk menjelaskan algoritma *FP-growth*. Misalnya digunakan *FP-tree* yang dibangkitkan untuk tabel 2.2 dimana nilai batas ambang dukungan minimum sama adalah 3. Seperti yang terlihat pada gambar 2.7, pohon yang dibangkitkan memiliki cabang, sehingga dilakukan proses iterasi untuk setiap item pada tabel *header* (pseudocode 2.2 baris 4). Dimisalkan nilai β adalah I_1 memiliki jumlah dukungan 3, karena nilai $\alpha_i = I_1$ dan $\alpha =$ null (pseudocode 2.2 baris 5), karena awal proses *FP-growth* (*FP-tree,null*) sehingga α adalah null. Proses selanjutnya dibangkitkan *conditional*

pattern base dan *conditional FP-tree* untuk β. Berikut ini penjelasan kedua proses tersebut:

- **Pembangkitan conditional pattern base**

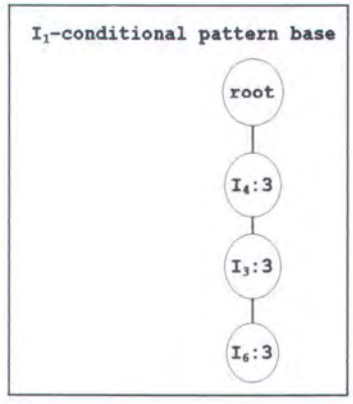
Conditional pattern base merupakan subbasis data yang berisikan kumpulan *prefix paths* dari *FP-tree* yang dibangkitkan untuk sebuah basis data transaksi. *Prefix paths* dibangun berdasarkan struktur kaitan simpul untuk sebuah item yang sama pada *FP-tree*. Hasil pembangkitan *conditional pattern base* untuk item I_1 seperti yang ditunjukkan pada gambar 2.10. Setiap himpunan item yang berada didalam kurung merupakan *prefix paths* dari item I_1, kemudian diikuti dengan jumlah dukungan *prefix paths.*

$\{(I_4I_3I_8I_6{:}2)\ ,\ (I_4I_3I_6I_2{:}1)\}$

Gambar 2.8 conditional *FP-tree* item I_1

- **Pembangkitan conditional FP-Tree**

Pada tahap ini, dihitung jumlah setiap item pada *conditional pattern base*, sehingga jumlah item I_4=3, I_3=3, I_8=2, I_6=3 dan I_2=1. Kemudian item yang memiliki jumlah lebih besar sama dengan nilai batas ambang dukungan minimum akan dibangkitkan menjadi sebuah pohon yang disebut sebagai *conditional FP-Tree*. $FP-tree_{I1}$ yang merupakan hasil pembangkitan *conditional FP-Tree* untuk item I_1 seperti yang ditunjukkan pada gambar 2.9. Simpul pada $FP-tree_{I1}$ adalah item I_4, I_3, I_6 karena memiliki jumlah lebih besar sama dengan 3. Hasil conditional *FP-Tree* untuk $FP-tree_{I1}$ dapat dituliskan sebagai $\{(I_4 = 3, I_3 = 3, I_6 = 3)\}|I_1$, dimana setiap item yang berada dalam kurung merupakan simpul dari *suffix pattern* item yang berada diluar garis vertical.



Gambar 2.9 ***Conditional FP-tree*** **item I_1(** $FP-tree_{I1}$ **)**

Pada tahap ini, karena pembangkitan *conditional FP-Tree* menghasilkan $FP-tree_{I1}$ ($FP-tree_{I1} \neq \phi$) maka dilakukan penggalian *frequent itemset* untuk $FP-tree_{I1}$ dengan menjalankan *FP-growth* ($FP-tree_{I1}$,I_1) (pseudocode 2.2 baris 8). Berdasarkan kondisi $FP-tree_{I1}$ yang memiliki satu cabang (pseudocode 2.2 baris 1) maka dilakukan kombinasi item I_1 dengan himpunan item pada $FP-tree_{I1}$ (pseudocode 2.2 baris 2-3). *Frequent itemset* yang didapatkan dari hasil kombinasi I_1 dengan himpunan pada $FP-tree_{I1}$ ditunjukkan pada tabel 2.13

Tabel 2.13 ***Frequent itemset*** **untuk item I_1**

Frequent itemset	Jumlah
I_4I_1	3
I_3I_1	3
I_6I_1	3
$I_4I_3I_1$	3
$I_4I_6I_1$	3
$I_3I_6I_1$	3
$I_4I_3I_6I_1$	3

Setelah interasi setiap item pada tabel *header* (pseudocode 2.2 baris 4), dengan menggunakan proses yang sama seperti pada penggalian *frequent itemset* untuk item I_1, maka *conditional pattern base* dan *conditional FP-tree* setiap item ditunjukkan pada tabel 2.14 dan hasil *frequent itemset* menggunakan algoritma *FP-growth* ditunjukkan pada tabel 2.15.

Tabel 2.14 ***Conditional pattern base*** **dan** ***conditional FP-tree*** **setiap item**

Item	*Conditional Pattern Base*	*Conditional FP-tree*
I_4	null	Null
I_3	$\{(I_4:3)\}$	$\{(I_4=3)\}\|I_3$
I_8	$\{(I_4I_3:2), (I_3:1)\}$	$\{(I_3=3)\}\|I_8$
I_6	$\{(I_4I_3I_8:2), (I_4I_3:1)\}$	$\{(I_4=3, I_3=3)\}\|I_6$
I_2	$\{(I_4I_3I_6:1), (I_4:1), (I_3I_8:1)\}$	Null
I_1	$\{(I_4I_3I_8I_6:2), (I_4I_3I_6I_2:1)\}$	$\{(I_4=3, I_3=3, I_6=3)\}\|I_1$

Tabel 2.15 ***Frequent itemset*** **menggunakan** ***FP-growth***

k	*Frequent k-itemset*
1	$I_4, I_3, I_8, I_6, I_2, I_1$
2	$I_3I_6, I_4I_6, I_1I_6, I_3I_4, I_3I_1, I_4I_1, I_3I_8$
3	$I_3I_4I_6, I_4I_1I_6, I_3I_1I_6, I_3I_4I_1$
4	$I_3I_4I_1I_6$

BAB III
ALGORITMA BOMO

BAB III

ALGORITMA BOMO

BOMO (*Build Once Mine Once*) merupakan algoritma pencarian *frequent itemset* tanpa ditentukan nilai batas ambang dukungan minimum. Permasalahan ini tidak dapat diselesaikan dengan menggunakan algoritma *apriori* maupun *FP-growth* didalam pencarian *frequent itemset* yang telah dijelaskan pada bab 2.

Ide pengembangan *BOMO* berdasarkan permasalahan awal bahwa sulit memberikan nilai batas ambang dukungan minimum yang tepat untuk pencarian *frequent itemset* sesuai hasil yang dinginkan. Hal ini dikarenakan apabila nilai batas ambang dukungan minimum terlalu besar, maka kemungkinan akan mendapatkan hasil yang sedikit atau tidak mendapatkan hasil sama sekali, sebaliknya apabila nilai batas ambang dukungan minimum terlalu kecil, maka akan mendapatkan hasil yang mungkin tidak bermanfaat bagi pengguna tetapi memiliki waktu komputasi proses yang lama. Selain itu, penentuan nilai batas ambang dukungan minimum akan berbeda untuk setiap ukuran dan jenis data yang digunakan, sehingga penggunaan nilai batas ambang dukungan minimum tidak boleh sama (*uniform*) untuk setiap ukuran dan jenis data. Dimana nilai batas ambang dukungan minimum untuk data berukuran besar memiliki kecenderungan nilai lebih kecil daripada nilai batas ambang dukungan minimum untuk data berukuran kecil.

3.1 Pembangkitan Algoritma *BOMO*

Berdasarkan permasalahan tersebut maka gagasan dasar dari algoritma *BOMO* adalah dengan dibangkitkan *FP-tree* untuk semua item I pada basis data transaksi D, kemudian melakukan proses pencarian *frequent itemset* yang diupayakan menjadi proses pencarian *n* k-*itemset* paling menarik (*n-most interesting k-itemsets*). Proses pencarian berdasarkan maksimal k-*itemset* (batas atas k_{max}) dan *n* yang paling menarik dari *n* k-*itemset* paling menarik.

Nilai batas ambang dukungan minimum dinotasikan ξ pada algoritma *BOMO* tidak ditentukan, akan tetapi setiap k-*itemset* memiliki nilai. Nilai batas ambang dukungan minimum setiap k-*itemset* dinotasikan dengan ξ_K. Dalam hal ini ξ_K merupakan jumlah dukungan *itemset* terkecil pada setiap k-*itemset*, dimana diawal pencarian *n* k-*itemset* paling menarik nilai ξ_K adalah 0 pada setiap $1 \leq k \leq k_{max}$. Tetapi nilai ξ_K akan meningkat pada saat proses penggalian data. Meskipun nilai ξ pada algoritma *BOMO* tidak ditentukan, tetapi dapat dihitung karena merupakan nilai terkecil dari nilai ξ_K. Berikut persamaan 3.1 untuk mendapatkan nilai ξ pada algoritma *BOMO*

$$\xi = \min(\xi_2, \xi_3, \ldots\ldots, \xi_{K\max}) \qquad 3.1$$

Berdasarkan pembahasan sebelumnya maka dapat dikatakan algoritma *BOMO* memiliki algoritma pencarian *frequent itemset* yang sama seperti algoritma *FP-growth.* Dalam konteks ini algoritma *BOMO* membangun *FP-tree* untuk melakukan pencarian *frequent itemset* sama seperti algoritma *FP-growth.* Akan tetapi algoritma *BOMO* menggunakan *FP-tree* yang dibangkitkan untuk semua item I pada basis data transaksi D dan proses pencarian *frequent itemset* hanya terbatas pada *n* k-*itemset* paling menarik. Untuk lebih jelasnya maka pseudocode algoritma *BOMO* [YLC-04] seperti yang ditunjukkan pada Pseudocode 3.1

Seperti yang ditunjukkan pada pseudocode 3.1, pencarian *n* k-*itemset* paling menarik menggunakan algoritma *BOMO* akan melibatkan tujuh tahapan utama. Pada tahap pertama inisialisasi $result_K$, tahap kedua dapatkan jumlah dukungan untuk setiap 1-*itemset*, tahap ketiga pembangkitan 1-*itemset* yang terurut dan pemberian nilai ξ_1, tahap keempat pemberian nilai awal untuk $\xi_2 = \xi_3 = \ldots = \xi_{k\max} = \xi = 0$, tahap kelima inisialisasi *FP-tree,* dan tahap keenam tahappembangkitan *FP-tree,* serta tahap ketujuh tahap penggalian data menggunakan algoritma *BOMO.*

MILIK PERPUSTAKAAN ITS

```
Masukkan : basis data D, maksimal k, nilai n
Keluaran : n k-itemset paling menarik
(1) Inisialisasi result_K.
(2) Dapatkan jumlah dukungan untuk setiap 1-itemset.
(3) Pembangkitan 1-itemset terurut dan pemberian nilai awal ξ1.
(4) Pemberian nilai awal untuk ξ2 = ξ3 = ... = ξk max = ξ = 0 .
(5) Inisialisasi FP-tree.
(6) Tahap pembangkitan FP-tree.
(7) Tahap penggalian data.
```

Pseudocode 3.1 Algoritma *BOMO*.

Pada sub-bab berikut dijelaskan secara rinci setiap tahapan proses algoritma *BOMO*, dimana setiap sub-bab akan diberikan contoh untuk menjelaskan proses algoritma *BOMO*. Misalnya ditentukan $n = 3$ dan maksimal k-*itemset* = 3 untuk data transaksi tabel 2.2, agar lebih mempermudah penjelasan maka data transaksi tabel 2.2 ditulis kembali pada tabel 3.1

Tabel 3.1 Data transaksi

TID	Item
1	I_4, I_1, I_3, I_6, I_8
2	I_1, I_2, I_3, I_4, I_6
3	I_2, I_4, I_5, I_7
4	I_2, I_3, I_7, I_8
5	$I_1, I_4, I_3, I_5, I_8, I_6$

(a) Inisialisasi *resultK*

Pada tahap ini, memberikan inisialisasi $result_K$ yang merupakan hasil n k-*itemset* paling menarik. Pemberian nilai awal $result_K$ adalah null. Kemudian pada tahap tahap penggalian data nilai $result_K$ akan didapatkan sesuai dengan k-*itemset* yang ditemukan, untuk $1 \leq k \leq k_{max}$. Inisialisasi *resultK* yang merupakan hasil 3 3-*itemset* paling menarik dengan nilai "*null*"

(b) Dapatkan jumlah dukungan untuk setiap 1-*itemset*

Pada tahap ini jumlah dukungan setiap item pada basis data dicari, pada tabel 3.2 akan ditunjukkan 1-*itemset* dengan jumlah dukungan untuk data transaksi pada tabel 3.1.

Tabel 3.2 1-*itemset* dengan jumlah dukungan

Item	Jumlah Dukungan
I_1	3
I_2	3
I_3	4
I_4	4
I_5	2
I_6	3
I_7	2
I_8	3

(c) Pembangkitan 1-*itemset* terurut dan pemberian nilai awal ξ_1

Pada tahap ini item diurutkan dari besar ke kecil berdasarkan jumlah dukungan item yang telah didapatkan dalam tahap sebelumnya. Pada tahap ini didapatkan nilai ξ_1 adalah 3 yang merupakan jumlah terkecil 3 1-*itemset* paling menarik. Hasil 1-*itemset* terurut dari besar ke kecil berdasarkan jumlah dukungan ditunjukkan pada tabel 3.3

Tabel 3.3 1-*itemset* terurut

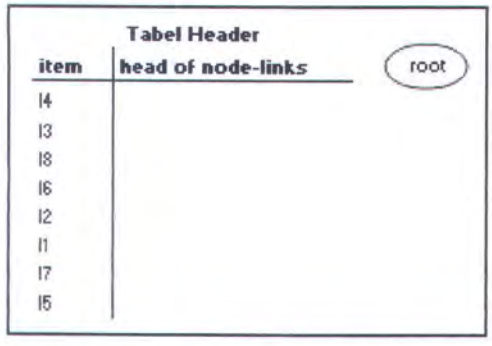
Item	Jumlah Dukungan
I_4	4
I_3	4
I_8	3
I_6	3
I_2	3
I_1	3
I_7	2
I_5	2

(d) Pemberian nilai awal untuk $\xi_2 = \xi_3 = ... = \xi_{k\max} = \xi = 0$.

Pemberian nilai awal untuk $\xi_2 = \xi_3 = ... = \xi_{k\max} = \xi = 0$, akan tetapi pada tahap penggalian data nilai tersebut meningkat untuk setiap k pada $2 \leq k \leq k_{max}$. Peningkatan ξ_K sesuai dengan peningkatan jumlah dukungan terkecil setiap *n* k-*itemset* paling menarik yang didapatkan. Dalam konteks ini lakukan inisialisasi $\xi_2 = \xi_3 = 0$.

(e) Inisialisasi *FP-tree*

Pada tahap ini diinisialisasi *FP-tree* yang terdiri dari sebuah tabel *header* dan sebuah pohon dengan simpul akar bernilai "*null*". Dimana tabel *header* yang dibangkitkan merupakan item terurut sesuai dengan tabel 3.3. Inisialisasi *FP-tree* ditunjukkan pada gambar 3.1.



Gambar 3.1 Inisialisasi *FP-tree*

(f) Tahap pembangkitan *FP-tree*

Pada tahap ini dibangkitkan *FP-tree* hasil ekstraksi secara lengkap data transaksi pada tabel 3.1. Himpunan item pada setiap transaksi diurutkan berdasarkan item terurut pada tabel 3.3, sebelum dimasukkan pada *FP-tree* yang dibangkitkan. Himpunan item terurut setiap transaksi ditunjukkan pada tabel 3.4.

Tabel 3.4 Himpunan item terurut pada setiap transaksi

TID	Data transaksi
1	I_4, I_3, I_8, I_6, I_1
2	I_4, I_3, I_6, I_2, I_1
3	I_4, I_2, I_7, I_5
4	I_3, I_8, I_2, I_7
5	$I_4, I_3, I_8, I_6, I_1, I_5$

Himpunan item terurut pada setiap transaksi seperti yang ditunjukkan pada tabel 3.4, selanjutnya dimasukkan dalam *FP-tree* yang dibangkitkan sebelumnya. Setiap item pada data transaksi yang dimasukan kedalam pohon pada *FP-tree* akan menjadi sebuah simpul baru apabila belum memiliki urutan simpul dalam pohon, sebaliknya apabila item pada data transaksi yang dimasukkan kedalam

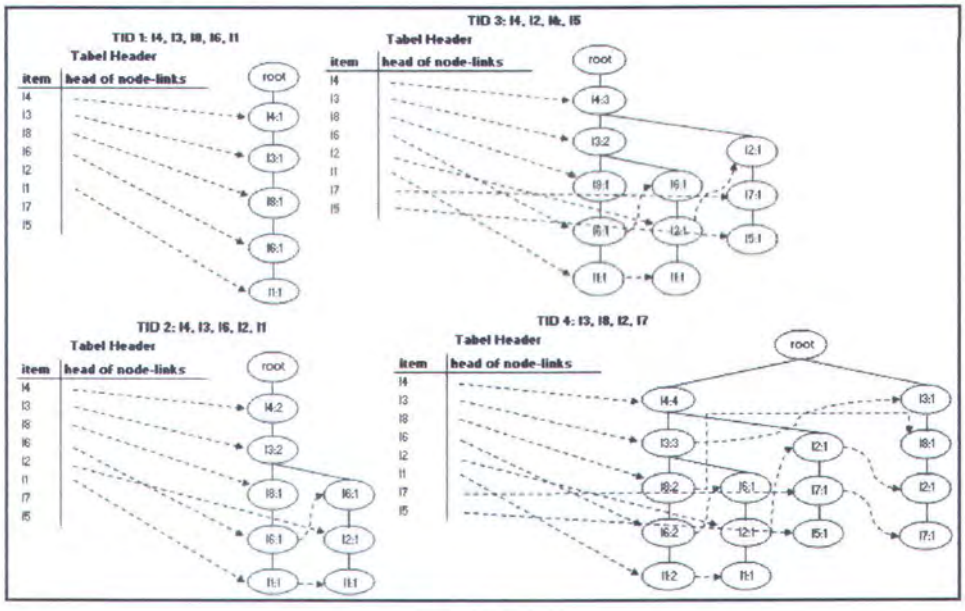
pohon telah memiliki urutan simpul, maka hanya akan menambah jumlah item yang tersimpan pada simpul. Mengenai struktur *FP-tree* telah dijelaskan secara lengkap dalam bab 2 (sub-bab pembangkitan *FP-tree*). Pseudocode 3.3 merupakan tahap pembangkitan *FP-tree*, yang dapat memberikan penjelasan pada tahap ini.

```
NFP-build(FP-tree T, Database D, 1-itemsets terurut, ξ)
(1) Bentuk 1-itemsets terurut dari besar ke kecil berdasarkan
    jumlah 1-itemset yang dimiliki, dimana jumlah 1-itemset ≥
    ξ, simpan sebagai selected-list.
(2) Lakukan update FP-tree, untuk setiap data transaksi pada D
    Jika data transaksi berisi item pada selected-list
        lalu urutkan 1-itemset data transaksi D sesuai dengan
    selected-list, jika [i|I] adalah frequent itemset pada data
    transaksi D dimana i adalah 1-itemset pertama pada data
    transaksi D dan I merupakan 1-itemset sisanya, maka masukkan
    [i|I] kedalam FP-tree T dengan menggunakan
    Insert_NFPtree([i|I], T)

Insert_NFPtree([i|I], T)
(3) Jika FP-tree T memiliki simpul child C dimana C.nama_1-
    itemset = i.nama_1-itemset
    maka jumlah 1-itemset C = jumlah 1-itemset C + jumlah 1-
    itemset i
        jika tidak maka buat simpul baru untuk 1-itemset C
    dengan jumlah yang dimiliki 1-itemset C, dimana memiliki
    simpul 1-itemset C memiliki hubungan parent terhadap T dan
    memiliki hubungan simpul terhadap simpul C. nama_1-itemset
    yang sama pada FP-tree T (structure node link)
(4) jika itemset I belum kosong maka lakukan Insert_NFPtree(I, T)
    secara rekrusif.
```

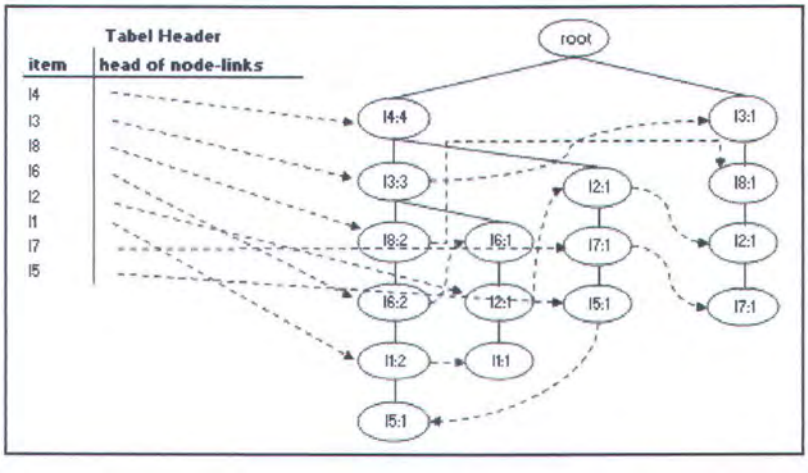
Pseudocode 3.2 Tahap pembangkitan *FP-tree*

Semua data transaksi pada tabel 3.4 dimasukkan kedalam pohon, dengan menggunakan *Insert_NFPtree([i|I], T)* sesuai pseudocode tahap pembangkitan *FP-tree*. Data transaksi yang dimasukkan pada pohon berdasarkan urutan TID, yaitu dimulai dari TID yang memiliki nilai terkecil sampai TID yang memiliki nilai terbesar. Proses pembangunan *FP-tree* untuk data transaksi dengan TID 1-4 pada tabel 3.4 ditunjukkan pada gambar 3.2.



Gambar 3.2 Proses pembangkitan *FP-tree*

Secara keseluruhan tahap pembangkitan *FP-tree* menghasilkan struktur data tabel *header* dan pohon dengan struktur *prefix-subtree.* Struktur data tabel *header* menyimpan item beserta jumlahn dukungan sesuai tabel 3.3 dan simpul kepala untuk struktur kaitan simpul. Struktur data pohon yang dibangkitkan memiliki simpul akar yang bernilai "null" dan beberapa simpul anak yang menyimpan item beserta jumlah dukungan yang ada pada simpul itu. Setiap item yang sama memiliki keterhubungan didalam *FP-tree*. Hasil tahap pembangunan *FP-tree* untuk data transaksi pada tabel 3.3 seperti yang ditunjukkan pada gambar 3.3, dimana merupakan *FP-tree* yang dibangkitkan pada saat TID adalah 5.



Gambar 3.3 Hasil pembangkitan *FP-tree*

(g) Tahap Penggalian Data

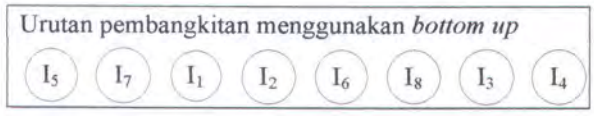
Pada tahap ini, *FP-tree* yang dibangkitkan pada gambar 3.5 dicari 3 3-*itemset* paling menarik. Penggalian data untuk mendapatkan *n* k-*itemset* paling menarik memiliki dua kondisi yaitu jika pohon memiliki prefix path dalam satu cabang (*single path)* atau pohon memiliki banyak cabang (*multi path).*

Tahap penggalian ini memiliki proses sama dengan *FP-growth* pada bab 2. Dalam konteks ini algoritma *BOMO* juga menggunakan pembangkitan *FP-tree* untuk melakukan pencarian *frequent itemset* sama seperti algoritma *FP-growth.* Akan tetapi algoritma *BOMO* menggunakan *FP-tree* yang dibangkitkan untuk semua item pada basis data transaksi dan proses pencarian *frequent itemset* hanya terbatas pada *n* k-*itemset* paling menarik. Tahap penggalian juga dilakukan peningkatan nilai ξ_K, jika jumlah k-*itemset* lebih besar sama dengan ξ_K. Proses tahap penggalian [YLC-04] ditunjukkan pada pseudocode 3.3.

```
NFP-mine(Tree, pefix path α , ξ,ξ1,ξ2,....,ξkmax ) {
   1. jika tree memiliki prefix path α merupakan single path P
      maka setiap kombinasi item β pada P maka
        (a) Dapatkan itemset β∪α dengan  jumlah = jumlah minimal
            item pada β.
        (b) jika  ξ|β∪α| ≤ jumlah itemset β∪α
  maka masukan itemset β∪α kedalam result|β∪α|;
          tingkatkan nilai ξ|β∪α|;
            tingkatkan nilai ξ, jika perlu ;
                    jika result|β∪α| berisi itemset x dengan jumlah ≤
                          ξ|β∪α| maka hapus x dari result|β∪α|;
   2. jika tidak untuk setiap αi di header table dari tree maka
        (c)  Dapatkan itemset β=αi∪α  dengan jumlah = jumlah
             dari itemset αi
        (d) Bangkitkan conditional pattern base β dengan ξ lalu
             Bangkitkan conditional FP-Tree untuk β atau Treeβ
        (e) jika Treeβ ≠ null jalankan
            NFP-mine(Tree, β , ξ,ξ1,ξ2,....,ξkmax ) }
```

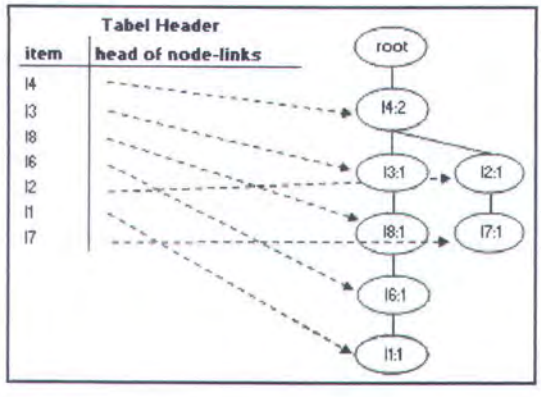
Pseudocode 3.3 Tahap penggalian algoritma *BOMO*

Misalnya penggalian data untuk mendapatkan 3 3-*itemset* paling menarik menggunakan metode penelusuran dari akhir *FP-tree*, metode ini akan dijelaskan pada sub-bab selanjutnya. Pada metode ini, maka urutan pencarian dimulai dari item yang terletak paling bawah (item yang memiliki jumlah terkecil) pada tabel *header* dan menuju ke item yang terletak paling atas (item yang memiliki jumlah terbesar). Metode penelusuran dari akhir *FP-tree* ditunjukkan pada gambar 3.4



Gambar 3.4 Metode penelusuran dari akhir *FP-tree*

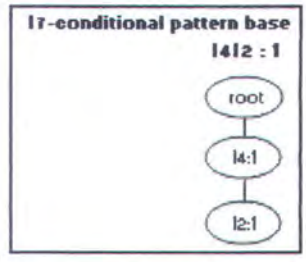
Pencarian 3 3-*itemset* paling menarik dimulai dari item I_5, pertama dicari *conditional pattern base* untuk item I_5, sehingga didapatkan *conditional pattern base* untuk I_5 adalah $I_4I_3I_8I_6I_1$:1 dan $I_4I_2I_7$:1. Tahap berikutnya dibangkitkan *conditional FP-tree* untuk I_5 dengan menggunakan semua item pada *conditional pattern base* I_5, karena awal proses penggalian data sehingga nilai batas ambang dukungan adalah 0. Hasil pembangkitan *conditional FP-tree* untuk I_5 seperti yang ditunjukkan pada gambar 3.5.



Gambar 3.5 *Conditional FP-Tree* untuk I_5

Pembangkitan *conditional FP-tree* untuk I_5 memiliki cabang maka dicari *conditional FP-tree* setiap item pada tabel *header FP-tree* untuk I_5 (pseudocode 3.3 langkah 2). Dalam hal ini, proses pembangkitan *conditional FP-tree* diawali untuk item I_7 karena merupakan item yang terletak paling bawah pada tabel *header FP-tree* untuk item I_5 (gambar 3.5) serta simpan *itemset* I_7I_5 hasil

gabungan $I_7 \cup I_5$ (pseudocode 3.3 langkah c) . Hasil pembangkitan *conditional FP-tree* untuk I_7 seperti yang ditunjukkan pada gambar 3.6.



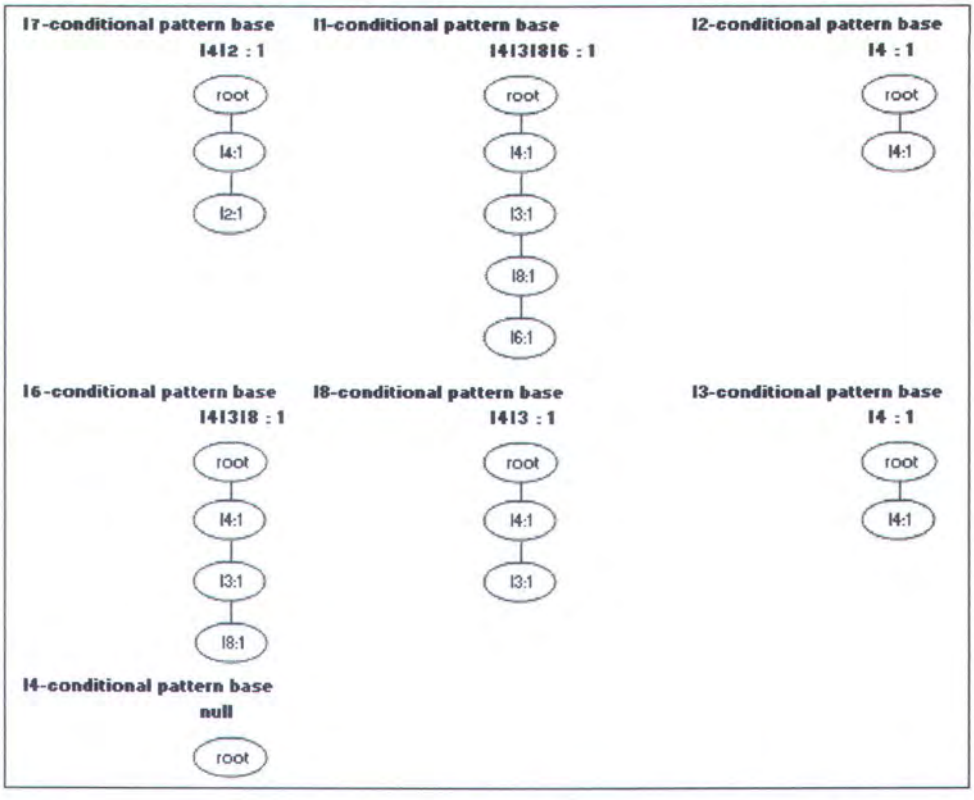
Gambar 3.6 *Conditional FP-Tree* untuk I_7

Karena hasil pembangkitan *conditional FP-Tree* untuk item I_7 memiliki satu cabang (gambar 3.8), sehingga dilakukan kombinasi I_7I_5 dengan setiap item pada *conditional FP-tree* untuk I_7 (pseudocode 3.3 langkah a) untuk mendapatkan 3 3-*itemset* paling menarik. Sedangkan jumlah setiap *itemset* hasil kombinasi merupakan jumlah terkecil dari item pembentuknya (pseudocode 3.3 langkah b). Hasil 3 3-*itemset* paling menarik dihasilkan ditunjukkan pada table 3.5.

Tabel 3.5 Hasil 3 3-*itemset* paling menarik sementara

k	k-itemsets	Jumlah Dukungan
1	I_5	2
	I_7	1
	-	-
2	I_7, I_5	1
	-	-
	-	-
3	I_4, I_7, I_5	1
	I_2, I_7, I_5	1
	-	-

Berdasarkan persamaan 3.1 maka nilai ξ=0, karena jumlah terkecil k-*itemset* ditemukan adalah 0. Pada tahap selanjutnya dilakukan pembangkitan *conditional FP-tree* item lainnya pada tabel *header conditional FP-Tree* untuk I_5 (gambar 3.5), Seluruh hasil pembangkitan *conditional FP-tree* untuk item lain pada tabel header I_5 ditunjukkan pada gambar 3.7, dan hasil 3 3-*itemset* paling menarik yang didapatkan ditunjukkan pada tabel 3.6.



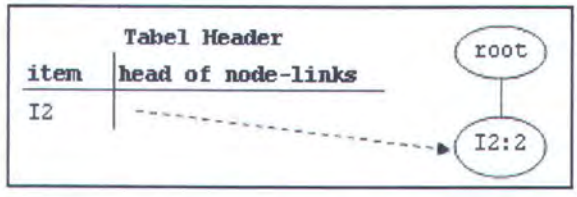
Gambar 3.7 *Conditional FP-Tree* item pada tabel *header* I_5

Tabel 3.6 Hasil 3 3-*itemset* paling menarik pada item I_5

k	k-itemsets	Jumlah Dukungan
1	I_5	2
	I_7	1
	I_1	1
2	I_7, I_5	1
	I_1, I_5	1
	I_2, I_5	1
3	I_4, I_7, I_5	1
	I_2, I_7, I_5	1
	I_6, I_1, I_5	1

Berdasarkan pada gambar 3.4, maka penggalian data dilanjutkan untuk item I_7. Tahap pertama dicari *conditional pattern base* untuk item I_7, sehingga didapatkan *conditional pattern base* untuk I_7 adalah I_4I_2:1 dan $I_3I_8I_2$:1. Tahap berikutnya dibangkitkan *conditional FP-tree* untuk I_7 dengan menggunakan item I_2, karena setelah proses penggalian item I_5 nilai batas ambang dukungan adalah 1

(pseudocode 3.3 langkah b), nilai ini berdasarkan tabel 3.6. Hasil pembangkitan *conditional FP-tree* untuk I_7 seperti yang ditunjukkan pada gambar 3.8



Gambar 3.8 *Conditional FP-Tree* untuk I_7

Karena hasil pembangkitan *conditional FP-Tree* untuk item I_7 memiliki satu cabang (gambar 3.10), sehingga dilakukan kombinasi I_2 dengan I_7 (pseudocode 3.3 langkah a). Sedangkan jumlah I_2I_7 adalah 2, karena jumlah terkecil dari item pembentuknya adalah 2 (pseudocode 3.3 langkah b). Hasil 3 3-*itemset* paling menarik dihasilkan ditunjukkan pada table 3.7

Tabel 3.7 Hasil 3 3-*itemset* paling menarik pada item I_7

k	k-itemsets	Jumlah
1	I_5	2
	I_7	2
	I_2	2
2	I_2, I_7	2
	I_1, I_5	1
	I_2, I_5	1
3	I_4, I_7, I_5	1
	I_2, I_7, I_5	1
	I_6, I_1, I_5	1

Proses penggalian data untuk mendapatkan 3 3-*itemset* paling menarik berdasarkan gambar 3.4, dimulai dari dari I_5 sampai I_4. sehingga proses pencarian akan berhenti jika tahap penggalian untuk I_4 telah selesai. Tahap penggalian yang terjadi memiliki proses yang sama dengan tahap penggalian I_7 dan I_5. Apabila keseluruhan tahap penggalian telah selesai maka akan dihasilkan *3-most interesting 3-itemsets*. Hasil *3-most interesting 3-itemsets* untuk data transaksi pada tabel 3.1 ditunjukkan pada tabel 3.8 dan nilai batas ambang dukungan setiap k-*itemset* atau ξ_K untuk $1 \leq k \leq 3$ ditunjukkan pada tabel 3.9

Tabel 3.8 Hasil pencarian 3 3-*itemset* paling menarik

k	k-itemsets	Jumlah Dukungan
1	I_4	4
	I_3	4
	I_8	3
2	I_1, I_4	3
	I_1, I_3	3
	I_1, I_8	3
3	I_1, I_4, I_6	3
	I_1, I_3, I_4	3
	I_1, I_3, I_6	3

Tabel 3.9 ξ_K untuk $1 \leq k \leq 3$

k	Nilai batas ambang dukungan
1	$\xi_1 = 3$
2	$\xi_2 = 3$
3	$\xi_3 = 3$

3.2 Hasil Pola Kaidah Asosiasi

Setelah semua *n-itemset* paling menarik ditemukan, maka langkah selanjutnya adalah mencari pola kaidah asosiasi. Seperti yang telah dijelaskan dalam bab 2. Semua kaidah asosiasi yang memiliki nilai dukungan lebih besar daripada nilai batas ambang dukungan minimum merupakan kaidah asosiasi yang benar. Daftar semua kaidah asosiasi yang benar untuk *n* k-*itemset* paling menarik pada tabel 3.8 terdapat pada tabel 3.10.

Tabel 3.10 Hasil pola kaidah asosiasi

NO	Kaidah Asosiasi	$\frac{\sigma(A \cup B)}{T}$	$\frac{\sigma(A \cup B)}{\sigma(A)}$	Dukungan	Keterpercayaan
1	$I_3 \rightarrow I_4$	3/5	3/4	60%	75%
2	$I_4 \rightarrow I_3$	3/5	3/4	60%	75%
3	$I_4 \rightarrow I_6$	3/5	3/4	60%	75%
4	$I_6 \rightarrow I_4$	3/5	3/3	60%	100%
5	$I_3 \rightarrow I_8$	3/5	3/4	60%	75%
6	$I_8 \rightarrow I_3$	3/5	3/3	60%	100%
7	$I_3 \rightarrow I_4, I_6$	3/5	3/4	60%	75%

Tabel 3.10 Hasil pola kaidah asosiasi (lanjutan)

NO	Kaidah Asosiasi	$\frac{\sigma(A \cup B)}{T}$	$\frac{\sigma(A \cup B)}{\sigma(A)}$	Dukungan	Keterpercayaan
8	$I_3, I_4 \rightarrow I_6$	3/5	3/3	60%	100%
9	$I_4 \rightarrow I_6, I_3$	3/5	3/4	60%	75%
10	$I_4, I_6 \rightarrow I_3$	3/5	3/3	60%	100%
11	$I_6 \rightarrow I_3, I_4$	3/5	3/3	60%	100%
12	$I_6, I_3 \rightarrow I_4$	3/5	3/3	60%	100%
13	$I_1 \rightarrow I_3, I_4$	3/5	3/3	60%	100%
14	$I_1, I_3 \rightarrow I_4$	3/5	3/3	60%	100%
15	$I_3 \rightarrow I_4, I_1$	3/5	3/4	60%	75%
16	$I_3, I_4 \rightarrow I_1$	3/5	3/3	60%	100%
17	$I_4 \rightarrow I_1, I_3$	3/5	3/4	60%	75%
18	$I_4, I_1 \rightarrow I_3$	3/5	3/3	60%	100%
19	$I_1 \rightarrow I_4, I_6$	3/5	3/3	60%	100%
20	$I_1, I_4 \rightarrow I_6$	3/5	3/3	60%	100%
21	$I_4 \rightarrow I_6, I_1$	3/5	3/4	60%	75%
22	$I_4, I_6 \rightarrow I_1$	3/5	3/3	60%	100%
23	$I_6 \rightarrow I_1, I_4$	3/5	3/3	60%	100%
24	$I_6, I_1 \rightarrow I_4$	3/5	3/3	60%	100%

Pola kaidah asosiasi yang dihasilkan digunakan sebagai bahan pengambilan keputusan. Sebagai contoh pola kaidah asosiasi pada tabel 3.10 nomor 1 adalah : 60% (persentase dukungan) dari kastamer yang membeli item I_3 juga membeli item I_4. Maka kaidah asosiasi yang terbentuk adalah **If** I_3 = *yes* **then** I_4 = *yes*. Berdasarkan data ini para pelaku bisnis dapat menganalisa bisnis mereka. Setelah mereka menganalisa, maka mereka dapat mengambil keputusan bisnis yang penting. Aplikasi yang lain seperti: *market basket analysis*, *add-on sales*, *Cross-marketing*, dan segmentasi *customer* juga sering menggunakan kaidah asosiasi.

3.3 Metode Pencarian *n* k-*itemset* Paling Menarik

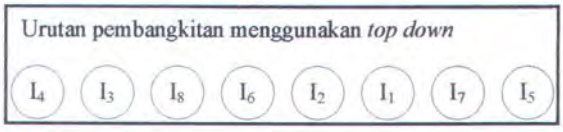
Pada sub-bab ini akan dijelaskan metode untuk memperoleh *n* k-itemset paling menarik yaitu menggunakan urutan pembangkitan *conditional FP-tree* (untuk algoritma *BOMO*) dan pendekatan *loopback*. Berikut penjelasan kedua metode tersebut berdasarkan Yin-Ling Cheung and Ada Wai-Chee Fu [YLC-04] :

3.3.1 Urutan Pembangkitan dari *Conditional FP-tree*

Urutan pembangkitan *conditional FP-tree* merupakan merupakan posisi awal (*contruction order*) pada tabel *header* untuk proses pencarian *frequent itemset*. Pada pembahasan selanjutnya dikatakan sebagai urutan pembangkitan untuk mempermudah penulisan. Perbedaan posisi awal akan berpengaruh pada ξ, ξ_k untuk $1 \leq k \leq k_{max}$, sehingga akan berpengaruh pada lama waktu komputasi proses. Berikut ketiga metode urutan pembangkitan:

(a) Metode penelusuran dari awal *FP-tree* (*top-down*)

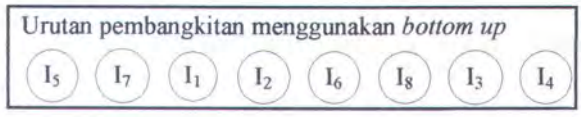
Metode ini memulai penggalian data menggunakan posisi awal pada awal item tabel *header* dan menuju ke akhir untuk membentuk *conditional pattern base*, sehingga *n* k-*itemset* paling menarik yang memiliki jumlah besar akan ditemukan terlebih dahulu. Akan tetapi *n* k-*itemset* paling menarik dengan k besar terdistribusi merata pada *Fp-tree* yang dibangkitkan, sehingga untuk ξ_k dengan nilai k besar akan meningkat secara lambat daripada untuk k kecil, yang pada akhirnya akan berpengaruh pada pembentukan nilai ξ berdasarkan persamaan 3.1 dan pemangkasan *conditional FP-Tree*. Urutan pembangkitan *conditional FP-tree* menggunakan metode penelusuran dari awal *FP-tree* untuk tabel *header* gambar 3.1 ditunjukkan pada gambar 3.9.



Gambar 3.9 Metode penelusuran dari awal *FP-tree*

(b) Metode penelusuran dari akhir *FP-tree* (*bottom up*)

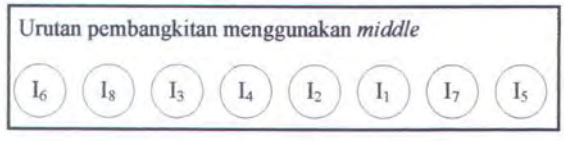
Metode ini menggunakan posisi awal pada akhir item tabel *header* dan menuju awal untuk membentuk *conditional pattern base*, sehingga k-*itemset* paling menarik yang memiliki jumlah kecil akan ditemukan terlebih akan tetapi berpengaruh lambat pada pembentukan *pruning conditional FP-tree*. Urutan pembangkitan *conditional FP-tree* menggunakan penelusuran dari akhir *FP-tree* untuk tabel *header* gambar 3.1 ditunjukkan pada gambar 3.10.



Gambar 3.10 Metode penelusuran dari awal *FP-tree*

(c) Metode penelusuran dari tengah *FP-tree* (*middle*)

Metode ini menggunakan posisi awal pada tengah item tabel *header*, dimana pembentukan *conditional pattern base* dimulai dari tengah menuju ke awal dan dilanjutkan dari tengah + 1 item pada tabel *header* menuju kebawah. Urutan pembangkitan *conditional FP-tree* menggunakan penelusuran dari tengah *FP-tree* untuk tabel *header* gambar 3.1 ditunjukkan pada gambar 3.11.

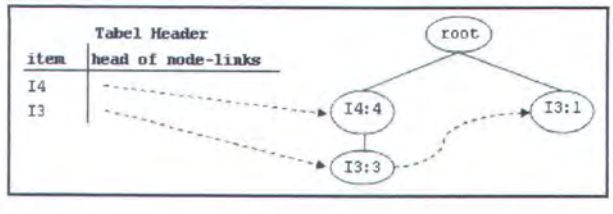


Gambar 3.11 Metode penelusuran dari tengah *FP-tree*

3.3.2 Pendekatan *Loopback*

Metode ini melakukan pencarian *n* k-*itemsets* paling menarik dengan *FP-tree* yang dibangkitkan dari sebagian item pada data transaksi secara berulang (*loopback*). Dalam konteks ini, *FP-tree* yang dibangkitkan hanya berisi item yang memiliki jumlah lebih besar sama dengan nilai batas ambang dukungan minimum, dimana nilai ini merupakan jumlah dukungan untuk item pada *n* paling menarik. Berikut penjelasan singkat pendekatan *loopback*.

Misalnya dilakukan penggalian data untuk menghasilkan 2 2-*itemsets* paling menarik untuk data transaksi pada tabel 3.1. Sehingga nilai batas ambang dukungan minimum untuk *FP-tree* sebagian adalah 4, karena jumlah dukungan pada *n* paling menarik adalah 4 (I_3 = 4). *FP-tree* sebagian yang dibangkitkan ditunjukkan pada gambar 3.12.



Gambar 3.12 *FP-tree* dengan ξ = 4.

FP-tree yang dibangkitkan kemudian dilakukan tahap penggalian, hasil pencarian 2 2-*itemsets* paling menarik untuk pendekatan *loopback* dengan $\xi = 4$ ditunjukan pada tabel 3.11.

Tabel 3.11 Hasil 2 2-*itemsets* paling menarik dengan $\xi = 4$.

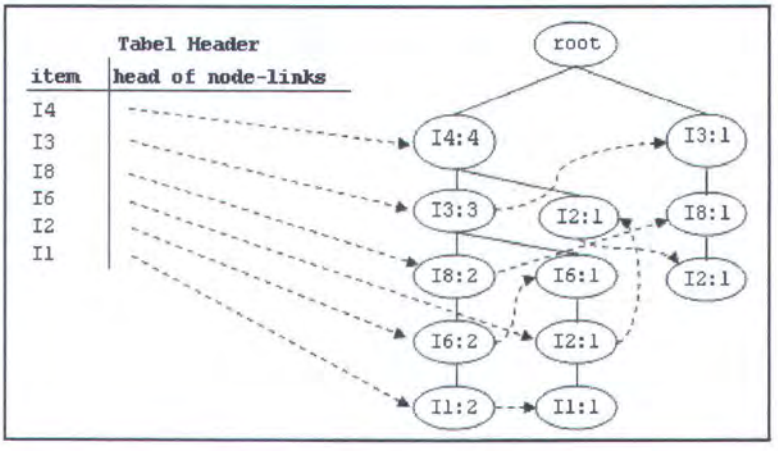
k	k-*itemsets*	Jumlah Dukungan
1	I_4	4
	I_3	4
2	-	-
	-	-

Pada kondisi tertentu mungkin 2 2-*itemsets* paling menarik telah didapatkan, akan tetapi pada kondisi lain belum tentu didapatkan, seperti pada contoh diatas. Apabila *n* k-*itemsets* paling menarik belum didapatkan maka proses pencarian dilakukan sekali lagi dengan menurunkan nilai batas ambang dukungan minimum yang ditemukan. Dalam hal ini, nilai batas ambang dukungan minimum pada *FP-tree* yang telah dibangkitkan (gambar3.12) adalah 4, sehingga nilai batas ambang dukungan minimum diturunkan menjadi 4 * f untuk $0 < f < 1$ dimana f sebagai notasi faktor penurunan (*decrease factor*). Kemudian dilakukan proses pembangkitan *FP-tree* berdasarkan nilai batas ambang dukungan minimum yang baru tersebut. Misalnya ditentukan faktor penurunan untuk *FP-tree* tersebut adalah 0.8, sehingga nilai batas ambang minimum yang baru adalah 3. Maka hasil pembangkitan *FP-tree* dengan nilai batas ambang dukungan minimum 3 ditunjukkan pada gambar 3.13 dan hasil akhir 2 2-*itemsets* paling menarik dengan nilai ambang batas minimum 3 diberikan pada tabel 3.12.

Tabel 3.12 Hasil 2 2-*itemsets* paling menarik dengan $\xi = 3$.

k	k-itemsets	jumlah
1	I_3	4
	I_4	4
2	I_4, I_3	3
	I_3, I_8	3

MILIK PERPUSTAKAAN ITS



Gambar 3.13 ***FP-tree*** **dengan** $\xi = 3$.

Proses penggalian data menggunakan pendekatan *loopback* melakukan pembacaan data transaksi berulang kali untuk membangkitkan sebuah *FP-tree* baru. Setiap *FP-tree* yang dibangkitkan dilakukan tahap penggalian untuk mencari *n* k-*itemsets* paling menarik. Meskipun tahap penggalian dilakukan berulang kali, tetapi hasil *n* k-*itemsets* paling menarik yang telah didapatkan tidak perlu dicari ulang. Misalnya *Fp-tree* pada gambar 3.12 nilai 1-*itemset* yang didapat I_4, I_3 (pada tabel 3.11), sehingga proses penggalian berikutnya untuk *Fp-tree* pada gambar 3.13 didapatkan 2-*itemset* yaitu (I_4, I_3) dan (I_3, I_8) (pada tabel 3.12).

3.4 Peningkatan Kinerja Pencarian *n* k-*itemset* Paling Menarik

Beberapa metode telah dikembangkan untuk meningkatkan kinerja pencarian *n* k-*itemset* paling menarik sehingga waktu komputasi proses akan berlangsung lebih cepat. Metode yang telah dikembangkan [YLC-04] antara lain pemangkasan *conditional FP-tree* dan pencarian ukuran batas ambang dukungan minimum. Keempat metode tersebut akan dijelaskan pada berikut ini:

3.4.1 Pemangkasan *Conditional FP-Tree*

Pembangkitan *conditional pattern FP-Tree* hanya dapat dilakukan apabila item pada tabel *header* memiliki jumlah lebih besar dari nilai batas ambang dukungan minimum. Sehingga proses pembangkitan *conditional pattern FP-Tree*

dapat dihentikan untuk item $a_1, a_2, a_3, \ldots, a_n$ pada tabel *header*, dimana item a_1 - a_n memiliki jumlah lebih kecil daripada ξ.

3.4.2 Pencarian Ukuran Batas Ambang Dukungan Minimum

Pemberian nilai awal ξ dan ξ_K dimana $2 \leq k \leq k_{max}$ akan memberikan proses pemangkasan *conditional FP-tree* (*pruning conditional FP-tree*) yang lebih cepat, sehingga penggalian data berlangsung lebih cepat.

Berikut penjelasan singkat metode pencarian ukuran batas ambang dukungan minimum Dikatakan C_k merupakan *array* dengan ukuran N untuk ξ_K dimana $2 \leq k \leq k_{max}$. Untuk setiap lintasan (*path*) dengan panjang k dari simpul akar, sehingga simpul k merupakan simpul terakhir dari lintasan tersebut. Simpan nilai terbesar batas ambang dukungan minimum setiap simpul k ke dalam C_k.

Dimisalkan k = 3 dan N = 1 untuk *FP-tree* pada gambar 3.5. Maka akan didapatkan dua *path root(null)* → (I_4:4) dan *root(null)* → (I_3:1) untuk k=1, tiga path *root(null)* → (I_4:4) → (I_3:3), *root(null)* → (I_4:4) → (I_2:1) dan *root(null)* → (I_3:1) → (I_8:1) untuk k=2 dan empat *path root(null)* → (I_4:4) → (I_3:3) → (I_8:2), *root(null)* → (I_4:4) → (I_3:3) → (I_6:1), *root(null)* → (I_4:4) → (I_2:1) → (I_7:1) dan *root(null)* → (I_3:1) → (I_8:1) → (I_2:1) untuk k=3. sehingga jumlah dukungan N terbesar pada C_1, C_2 dan C_3 adalah 4, 3 dan 3 kemudian dapat dikatakan nilai ξ_1, ξ_2, ξ_3 adalah 4, 3 dan 3 dan berdasarkan persamaan 3.1 maka nilai ξ adalah 3.

BAB IV
DESAIN & IMPLEMENTASI APLIKASI

BAB IV

DESAIN & IMPLEMENTASI APLIKASI

Dalam bab ini dijelaskan dua tahapan penting yaitu desain aplikasi dan tahap implementasi aplikasi untuk algoritma *BOMO*. Untuk menjelaskan desain aplikasi akan digunakan penggambaran konsep *Unified Modelling Languange* (*UML*) yang dimiliki oleh aplikasi Rational Rose Enterprise Edition 2002, sedangkan tahap implementasi meliputi implementasi basis data, implementasi kelas dan implementasi antarmuka dari aplikasi. Gambaran secara umum deskripsi perancangan dan pembuatan aplikasi terdapat pada lampiran A.

4.1 Desain Aplikasi

Pada bagian ini dijelaskan desain aplikasi untuk algoritma *BOMO* yang meliputi empat aspek penting yaitu desain *use case*, analisis *use case*, realisasi *use case*, diagram kelas dan desain antarmuka. Untuk tahap desain aplikasi digunakan empat macam diagram yang mereprentasikan setiap proses dalam aplikasi. Diagram–diagram tersebut antara lain adalah :

- Diagram *use case*: untuk menjelaskan aktor yang ada dalam sistem dan kegiatan yang bisa dilakukan oleh aktor tersebut
- Diagram aktifitas (*activity diagram*): untuk menggambarkan urutan aktifitas atau proses yang ada dalam tiap kegiatan yang dilakukan oleh aktor sistem aplikasi ini.
- Diagram sekuensi (*sequence diagram*): untuk menggambarkan urutan kejadian antar objek yang berinteraksi dalam sistem aplikasi ini.
- Diagram kelas (*class diagram*): diagram kelas menggambarkan keseluruhan kelas dari aplikasi yang mengimplementasikan desain diagram *use case*.

4.1.1 Analisis *Use Case*

Suatu *use case* dapat berhubungan dengan *use case* yang lain. Hubungan antar *use case* menunjukan keterkaitan proses pada kedua *use case* tersebut.

Secara umum, hubungan antara dua *use case* dapat dibedakan menjadi dua jenis, yaitu:

- Hubungan *include*

  Hubungan *include* terjadi bila suatu *use case* membutuhkan *use case* yang lainnya agar *use case* tersebut dapat berjalan. Sebagai contoh, *use case* ambil mata kuliah membutuhkan *use case* tawarkan mata kuliah

- Hubungan *extend*

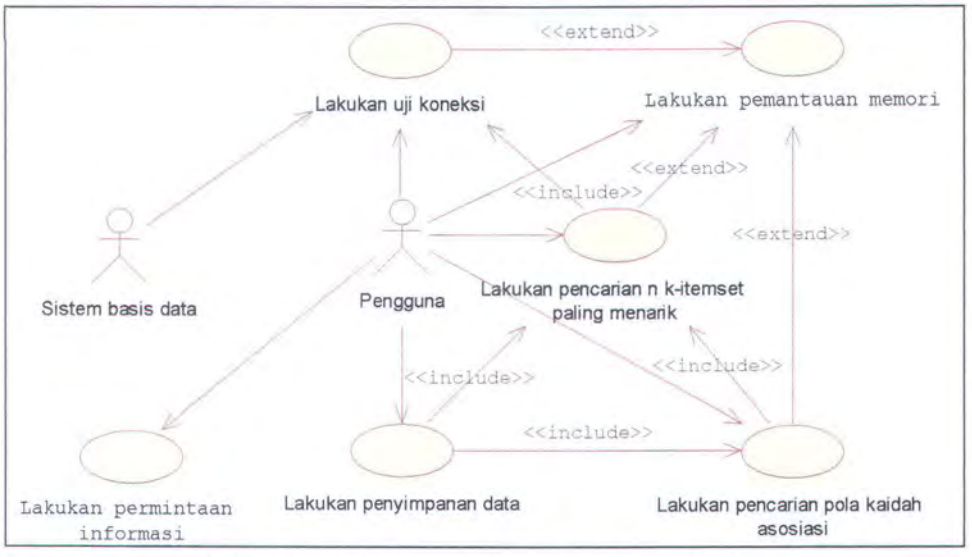
  Hubungan *extend* terjadi bila suatu *use case* dapat disertai dengan suatu *use case* lain atau tidak disertai dengan suatu *use case* lain tersebut. Sebagai contoh, *use case* bayar uang kuliah dan *use case* ambil mata kuliah. Seorang mahasiswa yang telah membayar uang kuliah dapat mengambil mata kuliah atau tidak mengambil mata kuliah.

Untuk membuat diagram *use case*, proses-proses yang terdapat pada aplikasi harus diidentifikasi terlebih dahulu. Detil dari keterangan proses-proses yang terdapat dalam aplikasi dijelaskan dalam tabel 4.1.

Tabel 4.1 Daftar proses

No	Nama proses	Keterangan
1	Lakukan uji koneksi	Proses ini merupakan proses awal yang sangat vital. Proses ini untuk mendapatkan koneksi terhadap basis data sesuai parameter yang telah ditentukan oleh aktor pengguna.
2	Lakukan pencarian *n* k-*itemset* paling menari	Proses ini untuk mencari *n* k-*itemset* paling menarik dari data transaksi.
3	Lakukan pencarian pola kaidah asosiasi	Proses ini mencari pola kaidah asosiasi dari hasil *n* k-*itemset* paling menarik
4	Lakukan permintaan informasi	Proses ini menampilkan informasi sistem melalui antarmuka aplikasi.
5	Lakukan pemantauan memori	Proses ini menampilkan informasi penggunaan memori saat aplikasi dijalankan
6	Lakukan penyimpanan data	Proses ini digunakan untuk menyimpan data hasil pencarian *n* k-*itemset* paling menarik atau hasil pola kaidah asosiasi

Berdasarkan proses–proses tersebut, maka dapat ditentukan diagram *use case* aplikasi yang dapat ditunjukkan dalam gambar 4.1. Pada diagram *use case* tersebut, terdapat dua aktor yaitu aktor pengguna dan aktor sistem basis data, serta memiliki enam *use case* berdasarkan proses pada table 4.1



Gambar 4.1 Diagram *use case* aplikasi

Pada diagram *use case* tersebut, terdapat dua aktor yaitu aktor pengguna dan aktor sistem basis data. Berikut penjelasan mengenai kedua aktor tersebut :

- Aktor Pengguna

  Merupakan pengguna yang memakai aplikasi untuk melakukan proses–proses seperti yang dijelaskan pada tabel 4.1

- Aktor sistem basis data

  Merupakan sistem yang menyediakan data transaksi dalam proses pencarian *n* k-*itemset* paling menarik.

Untuk memperjelas detail desain proses setiap *use case* digunakan tabel dokumentasi naratif *use case* [LDB-04]. Berikut ini merupakan detail proses setiap *use case* pada aplikasi yang diimplementasikan dalam tugas akhir ini.

(a) Lakukan uji koneksi

Pada sub-bab ini dijelaskan mengenai proses yang terjadi didalam *use case* lakukan uji koneksi basis data. Tabel dokumentasi naratif *use case* lakukan uji koneksi ditunjukkan pada tabel 4.2.

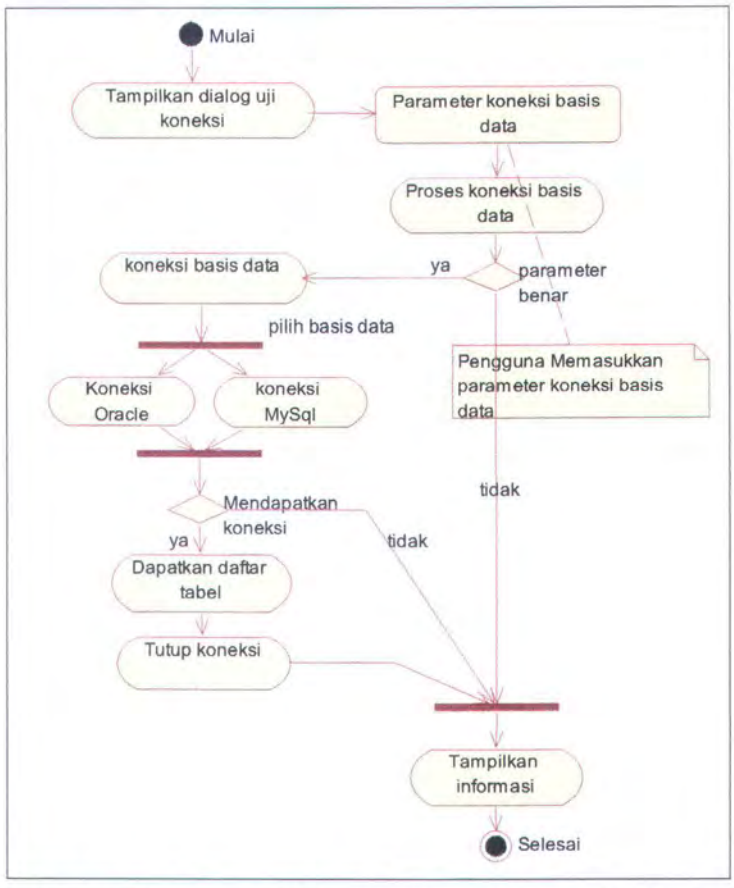
Tabel 4.2 Tabel dokumentasi naratif *use case* lakukan uji koneksi

Nama *Use case*	lakukan uji koneksi basis data	**Desain Sistem**
Pelaku Utama	- Aktor pengguna - Aktor sistem basis data	
Pelaku Partisipan Lain	Tidak ada	
Deskripsi	*Use case* ini digunakan oleh aktor pengguna untuk melakukan uji koneksi basis data sesuai parameter yang telah ditentukan.	
Prakondisi	Aplikasi dapat dijalankan.	
Pemicu	*Use case* ini diawali saat aktor pengguna meminta uji koneksi basis data	
Aliran Kejadian Normal	**Kegiatan Aktor Pengguna**	**Respon Aplikasi**
	Langkah 1: Aktor pengguna memilih menu untuk menampilkan antarmuka uji koneksi basis data. **Langkah 3:** Aktor pengguna memasukkan parameter dan eksekusi uji koneksi basis data.	**Langkah 2**: Aplikasi merespon dengan menampilkan antarmuka uji koneksi basis data **Langkah 4**: Aplikasi melakukan validasi parameter koneksi basis data. **Langkah 5**: Aplikasi melakukan koneksi data sesuai parameter.
		Sub Aliran Kejadian Normal
		* Jika basis data Oracle dipilih, maka eksekusi koneksi Oracle. * Jika basis data mySql dipilih, maka eksekusi koneksi mySql. **Langkah 6**: Aplikasi me lakukan kueri nama tabel basis data. **Langkah 7**: Aplikasi menampilkan nama tabel dan informasi koneksi basis data. **Langkah 8**: Aplikasi menutup koneksi
Aliran Alternatif	**Alt-Langkah 4**: Jika parameter uji koneksi basis data salah, aplikasi menampilkan informasi kesalahan parameter	
	Alt-Langkah 5: Jika aplikasi koneksi basis data gagal, maka ditampilkan informasi kegagalan koneksi.	
Pascakondisi	Aplikasi menampilkan nama tabel pada basis data	
Aturan *Use case*	Parameter yang digunakan untuk uji koneksi basis data adalah: (1) tipe basis data dan (2) nama pengguna dan (3) kata sandi pengguna dan (4) komputer *server* dan (5) *port* dan (6) nama basis data	
Batasan dan Spesifikasi Implementasi	1. *Use case* ini hanya dapat digunakan untuk basis data mySql dan Oracle. 2. Aplikasi hanya dapat melakukan koneksi untuk satu basis data	
Asumsi	Terdapat basis data.	

Pengguna dapat menjalankan proses ini dengan memilih menu "Koneksi Basis Data" pada kerangka utama (tabel 4.2 langkah 1). Kemudian aplikasi merespon dengan menampilkan dialog antarmuka uji koneksi basis data, sehingga aktor pengguna dapat memasukkan parameter koneksi basis data (tabel 4.2 langkah 2). Proses uji koneksi basis data dimulai saat pengguna menentukan parameter koneksi terhadap basis data. Beberapa parameter yang dimasukkan yaitu tipe basis data, nama pengguna, kata sandi pengguna, komputer *server*, *port* dan nama basis data (tabel 4.2 langkah 3). Koneksi yang dilakukan aplikasi bergantung tipe basis data yang dipilih aktor pengguna, dimana aplikasi menyediakan dua buah tipe basis data yaitu *mySql* dan *Oracle* (tabel 4.2 langkah 5). Aplikasi hanya dapat melakukan koneksi untuk satu basis data yaitu apabila memilih basis data *Oracle* maka aplikasi menjalankan koneksi *Oracle*, jika memilih basis data *mySql* maka aplikasi menjalankan koneksi *mySql*.

Proses ini sangat penting karena memungkinkan aplikasi untuk mendapatkan data masukan berupa nama tabel data transaksi untuk proses penggalian data (tabel 4.2 langkah 6). Setiap informasi koneksi akan bergantung pada tipe basis data yang ditentukan oleh pengguna. Aplikasi akan menyediakan informasi keberhasilan maupun kegagalan koneksi (tabel 4.2 langkah 7). Setelah aplikasi mendapatkan nama tabel untuk pencarian *n* k-*itemset* paling menarik maka secara otomatis aplikasi akan menutup koneksi (tabel 4.2 langkah 8).

Pada proses selanjutnya, aplikasi secara otomatis dapat membuat koneksi atau melakukan kueri basis data berdasarkan parameter tersebut. Sehingga aktor pengguna dapat memilih data transaksi sesuai dengan nama tabel basis data yang dipilih. Aplikasi hanya dapat melakukan koneksi untuk satu basis data, sehingga apabila aktor pengguna ingin menggunakan data transaksi pada basis data lain (dalam hal ini *Oracle* atau *Mysql*) harus mengisikan parameter lagi dan melakukan proses uji koneksi. Diagram aktifitas lakukan uji koneksi basis data seperti yang ditunjukkan pada gambar 4.2



Gambar 4.2 Diagram aktifitas lakukan uji koneksi basis data

(b) Lakukan pencarian *n* k-itemset paling menarik

Pada sub-bab ini dijelaskan mengenai proses yang terjadi didalam *use case* lakukan pencarian *n* k-*itemset* paling menarik. Tabel dokumentasi naratif *use case* lakukan pencarian *n* k-*itemset* paling menarik ditunjukkan pada tabel 4.3.

Tabel 4.3 Tabel dokumentasi naratif *use case* lakukan pencarian *n* k-*itemset* paling menarik

Nama *Use case*	lakukan pencarian *n* k-*itemset* paling menarik	**Desain Sistem**
Pelaku Utama	- Aktor pengguna - Aktor sistem basis data	
Pelaku Partisipan Lain	Tidak ada	

Tabel 4.3 Tabel dokumentasi naratif *use case* lakukan pencarian *n* k-*itemset* paling menarik (lanjutan)

Deskripsi	*Use case* ini digunakan aktor pengguna untuk pencarian *n* k-*itemset* yang paling menarik dari data transaksi sesuai dengan parameter yang ditentukan.	
Prakondisi	Aplikasi harus memiliki koneksi basis data.	
Pemicu	*Use case* ini diawali saat aktor pengguna meminta hasil *n* k-*itemset* paling menarik dari data transaksi.	
Aliran Kejadian Normal	**Kegiatan Aktor Pengguna**	**Respon Aplikasi**
	Langkah 1: Aktor pengguna memilih menu untuk menampilkan antarmuka penggalian data. **Langkah 3:** Aktor pengguna memasukkan parameter dan eksekusi penggalian data.	**Langkah 2**: Aplikasi merespon dengan menampilkan antarmuka penggalian data. **Langkah 4**: Aplikasi melakukan validasi parameter penggalian data. **Langkah 5**: Eksekusi penggalian data menggunakan urutan pembangkitan. **Langkah 6**: Aplikasi mencari jumlah item dan item terurut **Langkah 7**: Aplikasi membangkitan *FP-tree*. **Langkah 8**: Aplikasi mencari ukuran nilai batas ambang dukungan minimum. **Langkah 9**: Aplikasi melakukan penggalian data
		Sub Aliran Kejadian Normal
		* Jika metode *top down* dipilih, maka eksekusi proses *top down*. * Jika metode *middle* dipilih, maka eksekusi proses *middle*. * Jika metode *bottom up* dipilih, maka eksekusi proses b*ottom up*. **Langkah 10**: Aplikasi menampilkan *n* k-*itemset* paling menarik.
Aliran Alternatif	**Alt-Langkah 4**: Jika parameter penggalian data salah, aplikasi menampilkan informasi kesalahan parameter. **Alt-Langkah 5**: Jika eksekusi penggalian data menggunakan pendekatan *loopback* maka aplikasi melakukan: - Mencari jumlah item dan item terurut - Eksekusi pendekatan *loopback* - Jika *FP-tree* ada, penggalian data menggunakan metode *top_down* berulang kali sampai ditemukan *n* k-*itemset* paling menarik Menampilkan *n* k-*itemset* paling menarik (langsung langkah 10)	

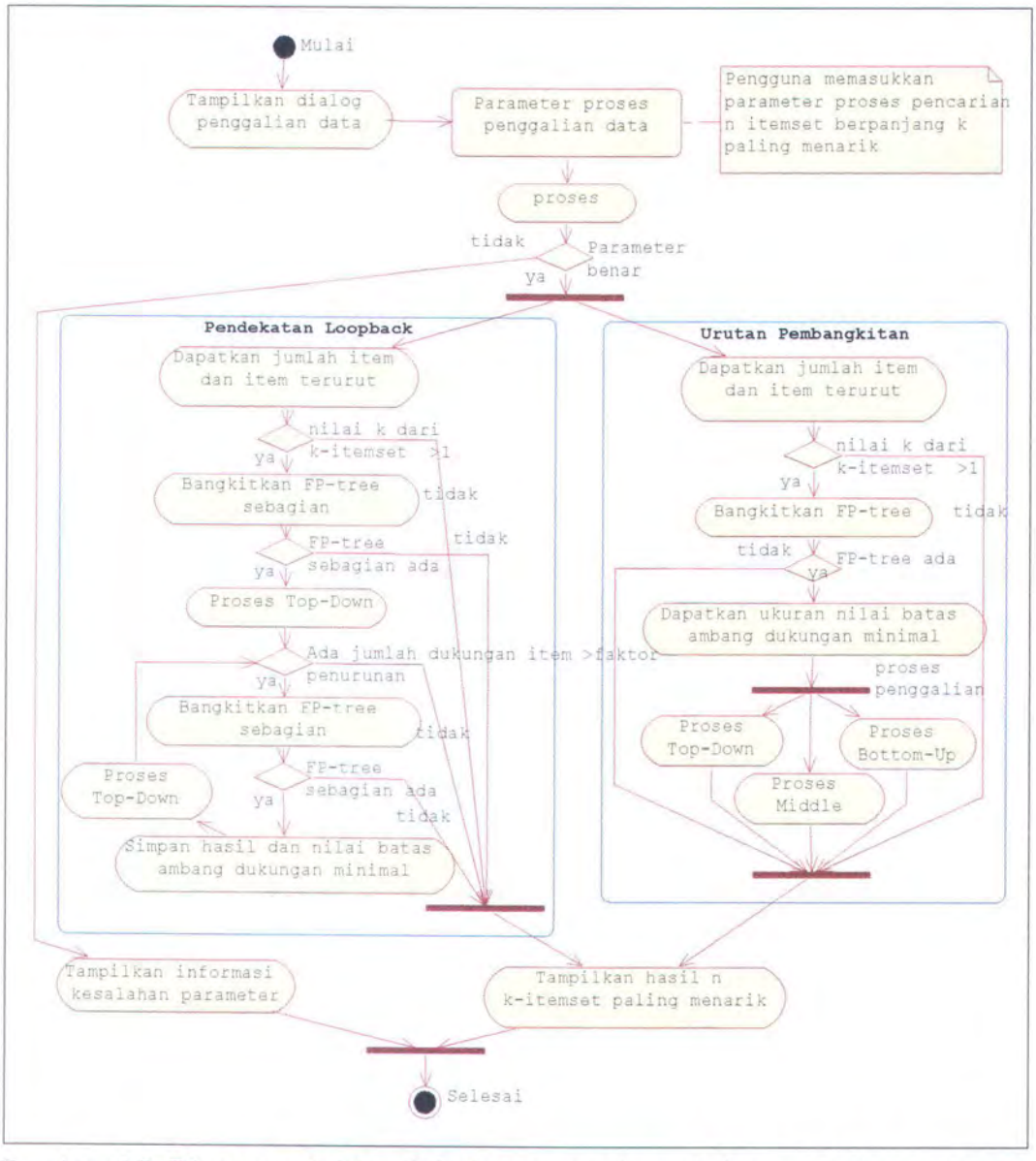
Tabel 4.3 Tabel dokumentasi naratif *use case* lakukan pencarian *n* k-*itemset* paling menarik (lanjutan)

Aliran Alternatif	**Alt-Langkah 7**: Jika parameter maksimal k adalah 1, maka aplikasi menampilkan *n* k-*itemset* paling menarik (langsung langkah 10). **Alt-Langkah 8**: Jika *FP-tree* tidak ada, maka aplikasi menampilkan *n* k-*itemset* paling menarik (langsung langkah 10).
Pascakondisi	Aplikasi menampilkan hasil *n* k-*itemset* paling menarik dari data transaksi
Aturan *Use case*	Parameter yang digunakan pada penggalian data adalah (1) nama tabel data transaksi dan (2) metode penggalian data dan (3) maksimal k dan (4) *n* paling menarik * jika aktor pengguna memilih urutan pembangkitan *conditional FP-tree*, maka pilih metode *top down, middle* atau *bottom up* * jika aktor pengguna memilih pendekatan *loopback*, maka tentukan faktor penurunan
Batasan dan Spesifikasi Implementasi	*Use case* ini dilakukan untuk data transaksi yang disimpan pada tabel dengan atribut : id_key (*number*) dan TID (*number*) dan Item (*number*)
Asumsi	Setiap basis data memiliki tabel data transaksi.

Pengguna dapat menjalankan proses ini dengan memilih menu "Proses" → submenu "Penggalian Data" pada kerangka utama (tabel 4.3 langkah 1). Kemudian aplikasi akan menampilkan dialog untuk proses penggalian data. Disini pengguna terlebih dahulu memilih tabel basis data yang tersedia dalam *combo box*, daftar pilihan tabel tersebut ada apabila koneksi basis data berhasil. Kemudian dilanjutkan dengan mengisi beberapa parameter lain seperti urutan pembangkitan, faktor penurunan nilai batas ambang, nilai *n* dan maksimal k-*itemset* dari *n* k-*itemset* paling menarik (tabel 4.3 langkah 3).

Penggalian data dengan aplikasi ini dibagi menjadi 2 cara yaitu menggunakan urutan pembangkitan *conditional FP-tree* atau pendekatan *loopback* (tabel 4.3 langkah 5). Apabila aktor pengguna memilih urutan pembangkitan *conditional FP-tree* maka proses penggalian dengan *top down, middle* atau *bottom up* dieksekusi sesuai metode yang ditentukan (tabel 4.3 langkah 9), jika memilih pendekatan *loopback* maka proses penggalian dengan pendekatan *loopback* dieksekusi sesuai faktor penurunan nilai ambang batas yang ditentukan. Pada tahap akhir ditampilkan *n* k-*itemset* paling menarik (tabel 4.3 langkah 10). Untuk lebih jelasnya mengenai proses pada *use case* ini, maka

diagram aktifitas lakukan pencarian *n* k-*itemset* paling menarik ditunjukkan pada gambar 4.3.



Gambar 4.3 Diagram aktifitas lakukan pencarian *n* k-*itemset* paling menarik

Secara umum proses penggalian data untuk mencari *n* k-*itemset* yang paling menarik memiliki tiga tahapan utama seperti yang terlihat pada tabel 4.3, berikut penjelasan ketiga tahapan tersebut:

(1) Dapatkan jumlah item dan item terurut (tabel 4.3 langkah 6 & Alt-langkah 5)

Pada tahapan ini aplikasi mencari jumlah dukungan setiap item pada data transaksi. Kemudian item pada data transaksi diurutkan dari besar ke kecil

berdasarkan jumlah dukungan. Tahap ini merupakan tahap perhitungan jumlah untuk setiap 1-*itemset* (pseudocode 3.1 baris 2) dan pembangkitan 1-*itemset* yang terurut dan pemberian nilai awal untuk ξ_1 (pseudocode 3.1 baris 3) pada algoritma *BOMO*.

(2) Pembangkitan *FP-tree* (tabel 4.3 langkah 7 & Alt-langkah 5)

Pada tahap ini akan dibangkitkan sebuah *FP-tree*. Tahap ini meliputi inisialisasi $result_K$ pemberian nilai awal untuk $\xi_2 = \xi_3 = ... = \xi_{k\max} = \xi = 0$, inisialisasi *FP-tree* dan tahap pembangkitan *FP-tree*. Ketiga tahap tersebut merupakan tahap pertama, keempat, kelima dan keenam dari algoritma *BOMO* (pseudocode 3.1). *FP-tree* yang dibangkitkan bergantung pada metode penggalian data yang ditentukan aktor pengguna. Aktor pengguna dapat memilih urutan pembangkitan *conditional FP-tree* atau pendekatan *loopback*. Dimana untuk urutan pembangkitan *conditional FP-tree* akan dibangkitkan *FP-tree* dari semua item yang ada, apabila menggunakan pendekatan *loopback* maka akan dibangkitkan *FP-tree* yang hanya menggunakan beberapa item tertentu saja yang memiliki jumlah dukungan lebih besar sama dengan jumlah dukungan item ke *n* paling menarik.

(3) Pencarian *n* k-*itemset* paling menarik(tabel 4.3 langkah 9 & Alt-langkah 5)

Pada tahapan ini atau tahap ketujuh pada algoritma *BOMO* yaitu tahap penggalian data. Proses pencarian *n* k-*itemset* paling menarik dapat menggunakan urutan pembangkitan *conditional FP-tree* dengan metode *top down, middle* atau *bottom up* atau pendekatan *loopback* dengan menentukan nilai faktor penurunan (f) dari batas ambang dukungan minimum.

Dengan selesainya proses penggalian data maka hasil *n* k-*itemset* paling menarik akan ditampilkan oleh aplikasi(tabel 4.3 langkah 10).

(c) Lakukan pencarian pola kaidah asosiasi

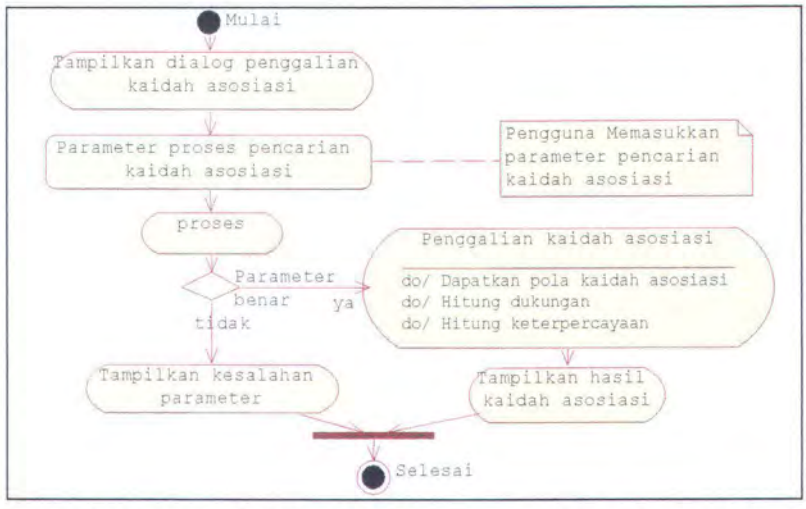
Pada sub-bab ini dijelaskan mengenai proses yang terjadi didalam *use case* lakukan pencarian pola kaidah asosiasi. Tabel dokumentasi naratif *use case* lakukan pencarian pola kaidah asosiasi ditunjukkan pada tabel 4.3.

Tabel 4.4 Tabel dokumentasi naratif *use case* lakukan pencarian pola kaidah asosiasi

Nama *Use case*	lakukan pencarian pola kai-dah asosiasi	**Desain Sistem**
Pelaku Utama	Aktor pengguna	
Pelaku Partisipan Lain	Tidak ada	
Deskripsi	*Use case* ini digunakan aktor pengguna untuk proses pencarian pola kai-dah asosiasi sesuai dengan parameter yang ditentukan.	
Prakondisi	Aplikasi menampilkan hasil *n* k-*itemset* paling menarik	
Pemicu	*Use case* ini diawali saat aktor pengguna meminta pola kaidah asosiasi dari *n* k-*itemset* paling menarik.	
Aliran Kejadian Normal	**Kegiatan aktor pengguna**	**Respon aplikasi**
	Langkah 1: Aktor pengguna memilih menu untuk menampil-kan antarmuka kaidah asosiasi. **Langkah 3:** Aktor pengguna memasukkan parameter dan eksekusi kaidah asosiasi.	**Langkah 2**: Aplikasi merespon dengan menampilkan antarmuka pencarian pola kaidah asosiasi **Langkah 4**: Aplikasi melakukan validasi parameter **Langkah 5**: Aplikasi mencari pola kaidah asosiasi. **Langkah 6**: Aplikasi menampil-kan hasil pencarian pola kaidah asosiasi.
Aliran Alternatif	**Alt-Langkah 4**: Jika parameter penggalian data salah, aplikasi menampilkan informasi kesalahan parameter.	
Pascakondisi	Aplikasi menampilkan hasil pola kaidah asosiasi	
Aturan *Use case*	Parameter yang digunakan pada pola kaidah asosiasi adalah (1) nilai minimum dan maksimal k dari k-*itemset* dan (2) nilai minimum dukungan dan (3) nilai minimum keterpercayaan	
Batasan dan Spesifikasi Implementasi	*Use case* dapat ini dilakukan jika nama item dari kaidah asosiasi disimpan pada tabel product dengan atribut : id_key (*number*) dan Item (*number*)	
Asumsi	Tidak ada	

Aktor pengguna dapat menjalankan proses ini dengan memilih menu "Proses" → submenu "Kaidah Asosiasi" pada kerangka utama (tabel 4.4 langkah 1). Disini aktor pengguna terlebih dahulu menentukan parameter untuk proses pencarian pola kaidah asosiasi antara lain minimum k dan maksimal k-*itemset* dari k-*itemset* serta menentukan nilai minimum dari dukungan (*support*) dan keterpercayaan (*confidence*) dari pola kaidah asosiasi yang akan dicari (tabel 4.4 langkah 3). Proses dalam *use case* ini mencari pola kaidah asosiasi, nama item,

nilai dukungan serta keterpercayaan. Diagram aktifitas untuk proses lakukan pencarian pola kaidah asosiasi ditunjukkan pada gambar 4.4.



Gambar 4.4 Diagram aktifitas lakukan pencarian pola kaidah asosiasi

(d) Lakukan pemantauan memori

Pada sub-bab ini dijelaskan mengenai proses yang terjadi didalam *use case* lakukan pemantauan memori. Tabel dokumentasi naratif *use case* lakukan pemantauan memori ditunjukkan pada tabel 4.5.

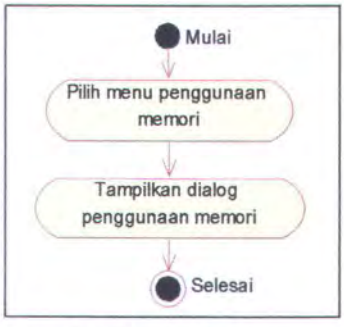
Tabel 4.5 Tabel dokumentasi naratif *use case* lakukan pemantauan memori

Nama *Use case*	Lakukan pemantauan memori	**Desain Sistem**
Pelaku Utama	Aktor Pengguna	
Pelaku Partisipan Lain	Tidak ada	
Deskripsi	*Use case* ini digunakan aktor pengguna untuk menampilkan informasi penggunaan memori saat aplikasi dijalankan.	
Prakondisi	Aplikasi dapat dijalankan.	
Pemicu	*Use case* ini diawali saat aktor pengguna meminta tampilan informasi penggunaan memori.	
Aliran Kejadian Normal	**Kegiatan Aktor Pelaku**	**Respon Sistem**
	Langkah 1: Aktor pengguna memilih menu untuk menampil-kan antarmuka penggunaan me-mori.	**Langkah 2**: Aplikasi menampilkan antarmuka penggu-naan memori
Aliran Alternatif	Tidak ada	
Pascakondisi	Aplikasi menampilkan antarmuka informasi penggunaan memori saat aplikasi dijalankan.	
Aturan *Use case*	Tidak ada	

Tabel 4.5 Tabel dokumentasi naratif *use case* lakukan pemantauan memori (lanjutan)

Batasan dan Spesifikasi Implementasi	1.Aplikasi hanya dapat menampilkan satu informasi saja. 2.Antarmuka penggunaan memori ditampilkan terpisah dari aplikasi yang dikembangkan.
Asumsi	Tidak ada

Ketika pengguna memilih menu "File" → submenu "Memori" (tabel 4.5 langkah 1) maka aplikasi menampilkan antarmuka penggunaan memori secara terpisah. Antarmuka penggunaan memori tersebut berasal dari Sun Microsystems, Inc [SUN-99]. Diagram aktifitas Lakukan pemantauan memori ditunjukkan pada gambar 4.5.



Gambar 4.5 Diagram aktifitas lakukan pemantauan memori

(e) Lakukan penyimpanan data

Pada sub-bab ini dijelaskan mengenai proses yang terjadi didalam *use case* lakukan penyimpanan data. Tabel dokumentasi naratif *use case* lakukan penyimpanan data seperti ditunjukkan pada tabel 4.6.

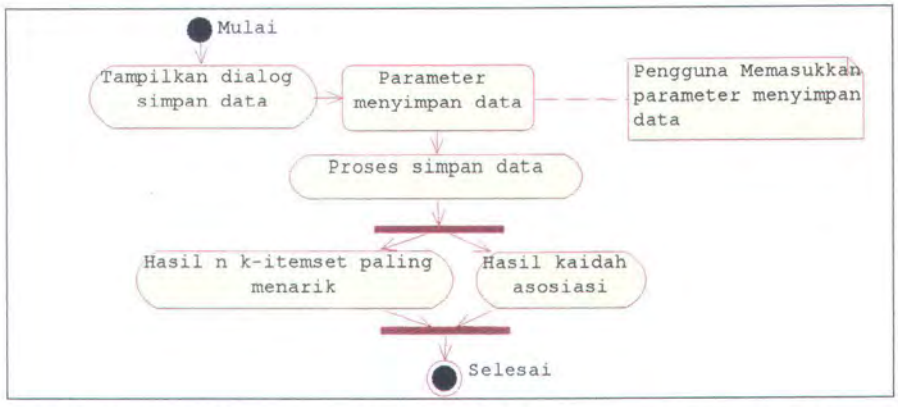
Tabel 4.6 Tabel dokumentasi naratif *use case* lakukan penyimpanan data

Nama *Use case*	Lakukan penyimpanan data	**Desain Sistem**
Pelaku Utama	Aktor Pengguna	
Pelaku Partisipan Lain	Tidak ada	
Deskripsi	*Use case* ini digunakan oleh aktor pengguna untuk menyimpan data hasil pencarian *n* k-*itemset* paling menarik dan pencarian pola kaidah asosiasi	
Prakondisi	Aplikasi menampilkan hasil *n* k-*itemset* paling menarik atau hasil pola kaidah asosiasi	
Pemicu	*Use case* ini diawali saat aktor pengguna menyimpan data hasil *n* k-*itemset* paling menarik atau hasil pola kaidah asosiasi	

Tabel 4.6 Tabel dokumentasi naratif *use case* lakukan penyimpanan data (lanjutan)

Aliran Kejadian Normal	**Kegiatan Aktor Pengguna**	**Respon Aplikasi**
	Langkah 1: Aktor pengguna memilih menu untuk menampilkan antarmuka menyimpan data **Langkah 3:** Aktor pengguna memasukkan parameter dan eksekusi menyimpan data.	**Langkah 2**: Aplikasi merespon dengan menampilkan antarmuka menyimpan data. **Langkah 4**: Aplikasi melakukan eksekusi menyimpan data
		Sub Aliran Kejadian Normal
		* Jika aktor pengguna memilih data hasil *use case* penggalian data maka simpan data *n* k-*itemset* paling menarik * Jika aktor pengguna memilih data hasil *use case* pola kaidah asosiasi maka aplikasi simpan data pola kaidah asosiasi.
Aliran Alternatif	Tidak ada	
Pascakondisi	Data hasil pencarian tersimpan dalam disk	
Aturan *Use case*	1. Parameter yang digunakan pada menyimpan data adalah (a) nama file (b) data hasil yang disimpan 2. Nama file yang disimpan sudah benar	
Batasan dan Spesifikasi Implementasi	Tidak ada	
Asumsi	Nama file yang sehingga aplikasi tidak melakukan validasi terhadap parameter *use case* lakukan penyimpanan data	

Ketika pengguna memilih memilih menu "File" → submenu "Simpan" (tabel 4.6 langkah 1) maka aplikasi menampilkan dialog antarmuka menyimpan data (tabel 4.6 langkah 2). Apabila parameter menyimpan data telah ditentukan, maka pengguna dapat menjalankan proses dengan menekan tombol "Simpan Data" (tabel 4.6 langkah 3). Diagram aktifitas proses menyimpan data ditunjukkan pada gambar 4.6.



Gambar 4.6 Diagram aktifitas lakukan penyimpanan data

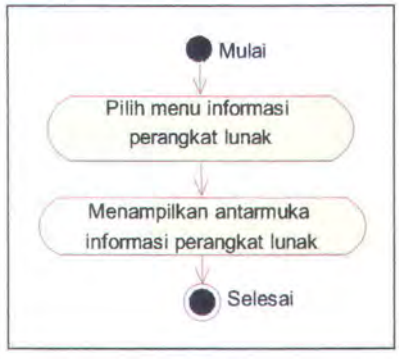
(f) Lakukan permintaan informasi

Pada sub-bab ini dijelaskan mengenai proses yang terjadi didalam *use case* lakukan permintaan informasi. Tabel dokumentasi naratif *use case* lakukan permintaan informasi ditunjukkan pada tabel 4.7.

Tabel 4.7 Tabel dokumentasi naratif *use case* lakukan permintaan informasi

Nama *Use case*	Lakukan permintaan informasi	**Desain Sistem**
Pelaku Utama	Aktor pengguna	
Pelaku Partisipan Lain	Tidak ada	
Deskripsi	*Use case* ini digunakan aktor pengguna untuk menampilkan informasi aplikasi.	
Pra Kondisi	Aplikasi dapat dijalankan.	
Pemicu	*Use case* ini diawali saat aktor pengguna meminta tampilan informasi aplikasi	
Aliran Kejadian Normal	**Kegiatan Aktor Pengguna**	**Respon Aplikasi**
	Langkah 1: Aktor pengguna meminta informasi aplikasi	**Langkah 2**: Aplikasi menampilkan antarmuka informasi aplikasi
Aliran Alternatif	Tidak ada	
Pascakondisi	Aplikasi menampilkan antarmuka informasi aplikasi	
Aturan *Use case*	Tidak ada	
Batasan dan Spesifikasi Implementasi	Aplikasi hanya dapat menampilkan satu Informasi.	
Asumsi	Tidak ada	

Pengguna dapat menjalankan proses ini dengan memilih menu "Informasi" pada kerangka utama (tabel 4.7 langkah 1). Kemudian aplikasi akan menampilkan antarmuka informasi aplikasi. Informasi aplikasi yang ditampilkan antara lain, algoritma yang diimplementasikan, *software* pembuat aplikasi dan sistem basis data yang dapat digunakan oleh penggunan aplikasi. Diagram aktifitas lakukan permintaan informasi ditunjukkan pada gambar 4.7.



Gambar 4.7 Diagram aktifitas lakukan permintaan informasi

4.1.2 Realisasi *Use Case*

Pada sub-bab ini diuraikan detail urutan kejadian antar objek yang berinteraksi dalam setiap *use case* dengan menggunakan diagram sekuensi dan hubungan antar obyek tersebut dengan menggunakan diagram *VOPC* (*View of Participated Class*). Sehingga pada tahap sebelumnya harus diidentifikasi obyek-obyek yang dapat digunakan berulang kali dalam kelas utilitis.

Diagram sekuensi menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) dalam sebuah pesan (*message*) terhadap waktu. Diagram sekuensi terdiri antar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Diagram sekuensi biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah kejadian (*event*) untuk menghasilkan keluaran tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan keluaran apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. Pesan digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada tahap

MILIK PERPUSTAKAAN ITS

desain berikutnya, pesan dipetakan menjadi operasi/metoda dari diagram *VOPC*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah pesan. Objek tersebut memiliki sifat khusus yang dapat didefinisikan menjadi objek *boundary, controller* dan *persistent entity.*

4.1.2.1 Kelas Utilitis

Kelas utilitis merupakan kelas yang digunakan berulang kali pada setiap proses. Detail setiap kelas utilitis aplikasi ditunjukkan pada tabel 4.8.

Tabel 4.8 Kelas utilitis beserta fungsinya

Kelas utilitis	Fungsi
Kontrol Utama	Kelas kontrol untuk melakukan eksekusi pencarian *n* k-*itemset* paling menarik dan pencarian pola kaidah asosiasi.
Kontrol Koneksi	Kelas kontrol untuk melakukan koneksi basis data
Item	Kelas entity untuk menyimpan item dan jumlah dukungan
FP-tree	Kelas kontrol untuk pembangkitan *FP-tree*
Tabel Header	Kelas entity untuk menyimpan tabel header dari *FP-tree*
Pohon	Kelas entity untuk menyimpan ekstraksi data transaksi
Itemset	Kelas entity untuk menyimpan k-*itemset* dan jumlah dukungan

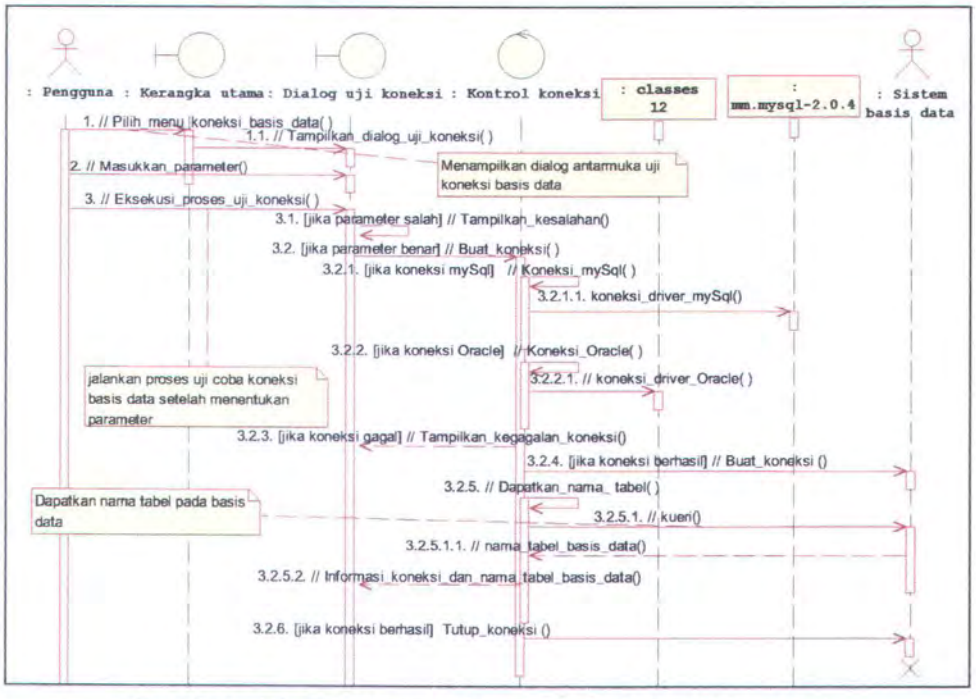
4.1.2.2 Diagram Sekuensi & *VOPC*

Berikut ini merupakan diagram sekuensi dan *VOPC* pada aplikasi yang diimplementasikan dalam tugas akhir ini.

(a) Lakukan uji koneksi

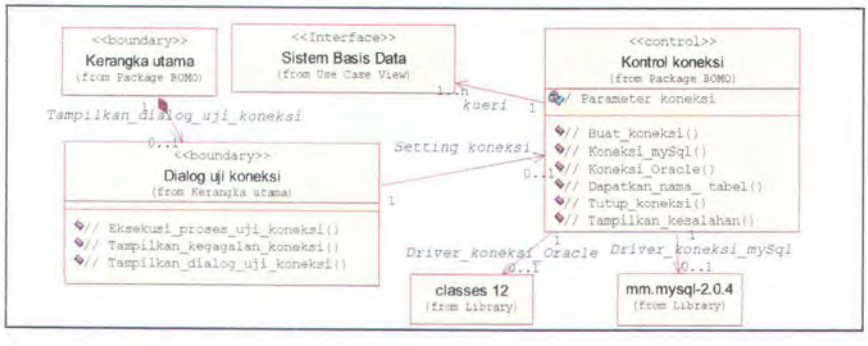
Proses dimulai saat aktor pengguna memilih menu "koneksi basis data" pada kerangka utama untuk menampilkan antarmuka uji koneksi. Ketika parameter koneksi dieksekusi, maka aplikasi akan melakukan validasi parameter melalui antarmuka uji koneksi. Jika parameter valid, maka aplikasi akan membuat koneksi sesuai dengan tipe basis data yang ditentukan. Apabila koneksi berhasil maka dipanggil method untuk mendapatkan nama tabel data transaksi. Informasi keberhasilan koneksi dan daftar nama tabel akan ditampilkan kepada aktor pengguna. Jika parameter salah maupun koneksi gagal maka aplikasi akan

menampilkan informasi tersebut. Diagram sekuensi lakukan uji koneksi ditunjukkan pada gambar 4.8.



Gambar 4.8 Diagram sekuensi lakukan uji koneksi

Seperti yang terlihat pada diagram sekuensi uji koneksi (gambar 4.8), terdapat tiga kelas yang terlibat dalam proses ini yaitu kerangka utama sebagai antarmuka awal dari aplikasi, dialog uji koneksi sehingga aktor pengguna dapat menentukan parameter koneksi dan kontrol koneksi bertanggung jawab terhadap koneksi basis data. Untuk menggambarkan hubungan ketiga kelas ini maka diagram vopc ditunjukkan pada gambar 4.9



Gambar 4.9 Diagram *VOPC* lakukan uji koneksi.

(b) Lakukan pencarian *n* k-*itemset* paling menarik

Proses dimulai ketika aktor pengguna meminta *n* k-*itemset* paling menarik dari suatu data. Pada saat dieksekusi, aplikasi akan menentukan validasi parameter melalui kelas antarmuka *n* k-*itemset* paling menarik. Jika parameter valid maka kelas kontrol utama memulai penggalian data. Kelas ini mencari jumlah item dan item terurut serta pembangkitan *FP-tree* dari data transaksi. Sedangkan pencarian *n* k-*itemset* dilakukan pada kelas *FP-tree*.

Seperti yang ditunjukkan diagram aktifitas pada gambar 4.3, pencarian *n* k-*itemset* paling menarik memiliki 2 proses yang dapat berjalan terpisah sesuai parameter yang ditentukan oleh pengguna yaitu:

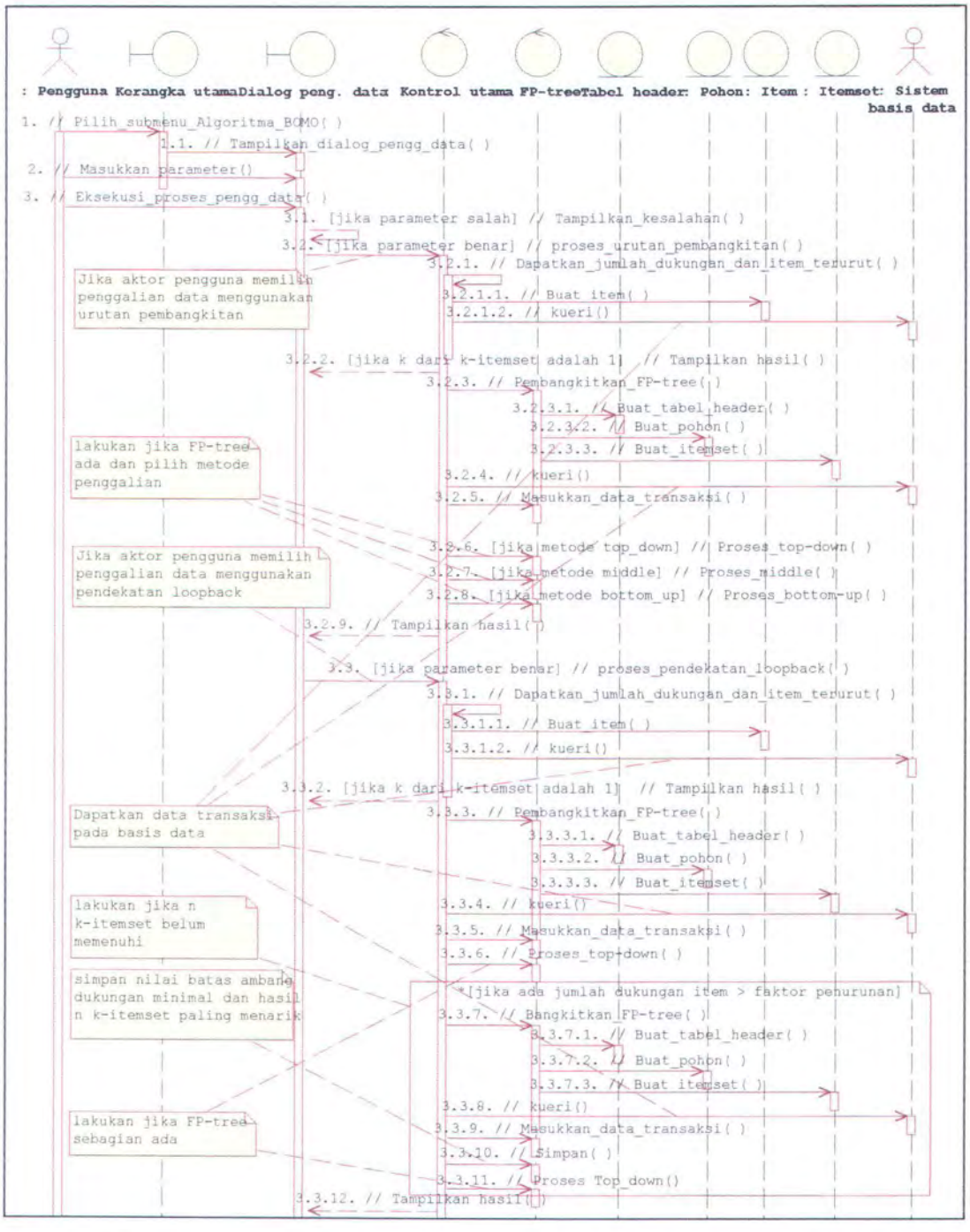
- Penggalian data menggunakan urutan pembangkitan

  Apabila parameter maksimal k-*itemset* lebih dari satu maka dibangkitkan sebuah *FP-tree* yang dibangun dari seluruh item pada data transaksi. Melalui *FP-tree* yang dibangkitkan dicari kemungkinan nilai batas ambang dukungan minimum, kemudian dilakukan penggalian data menggunakan metode *top-down, middle* atau *bottom up.*.

- Penggalian data menggunakan pendekatan *loopback*

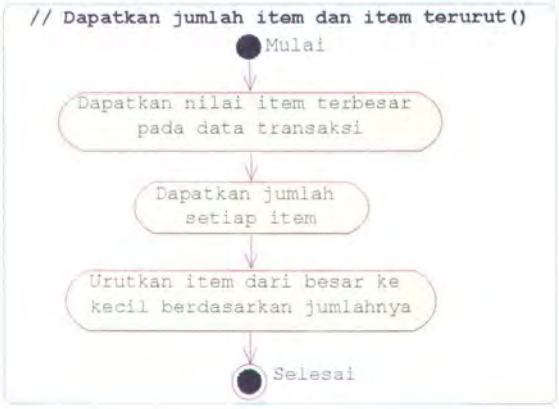
  Apabila parameter maksimal k-*itemset* pada k-*itemset* lebih dari satu, maka *FP-tree* dibangkitkan dengan beberapa item tertentu. Proses penggalian data menggunakan metode *top-down* dilakukan secara berulang sampai ditemukan *n* k-*itemset* paling menarik atau jika nilai batas ambang dukungan lebih besar dari faktor penurunan. Sehingga proses penggalian data menggunakan pendekatan *loopback* melakukan pembacaan terhadap basis data beberapa kali.

Untuk lebih menjelaskan aliran kejadian pada *use case* ini, sebuah diagram sekuensi lakukan pencarian *n* k-*itemset* paling menarik ditunjukkan gambar 4.10.



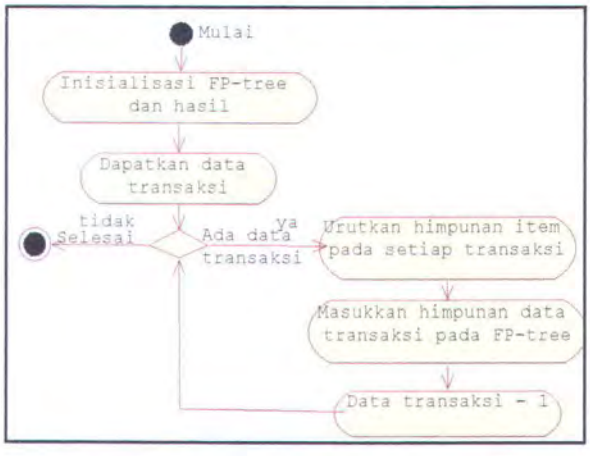
Gambar 4.10 Diagram sekuensi lakukan pencarian *n* k-*itemset* paling menarik

Untuk mencari jumlah dukungan item dan item terurut maka dilakukan pembacaan basis data satu kali. Kemudian setiap item diurutkan dari besar ke kecil berdasarkan jumlah dukungan. Diagram aktifitas untuk mendapatkan jumlah item dan item terurut ditunjukkan pada gambar 4.11



Gambar 4.11 Diagram aktifitas dapatkan_jumlah_dan_item_terurut ()

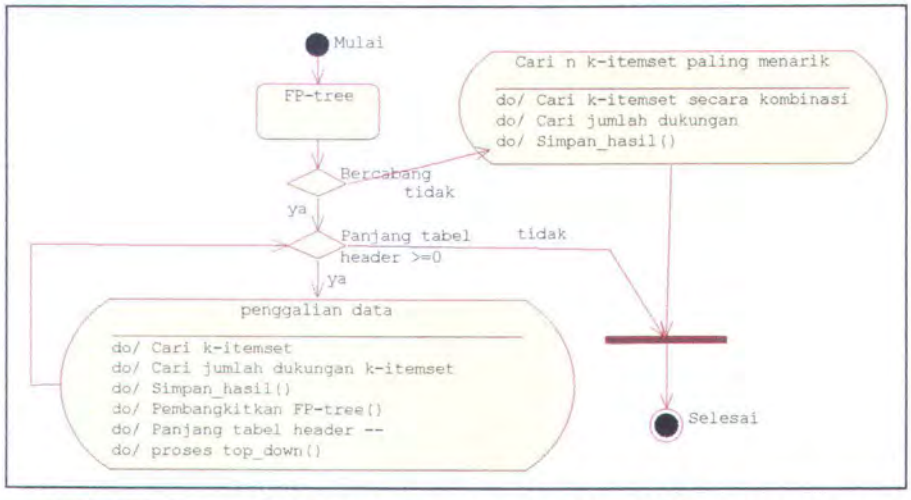
Pembangkitan *FP-tree* merupakan ekstraksi data transaksi menjadi *FP-tree*. Pembangkitan *FP-tree* dimulai dengan inisialisai *FP-tree*. Kemudian setiap transaksi yang telah terurut dimasukkan kedalam *FP-tree* yang diinisialisasi. Diagram aktifitas pembangkitan *FP-tree* ditunjukkan pada gambar 4.12



Gambar 4.12 Diagram aktifitas pembangkitan *FP-tree*

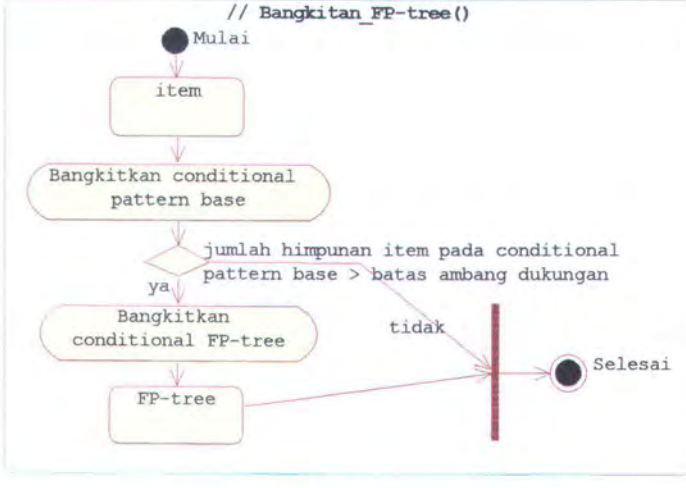
Proses pencarian *n* k-*itemset* paling menarik dapat menggunakan metode *top-down, middle* atau *bottom-up*. Proses pencarian menggunakan ketiga metode melihat kondisi pohon yang dibangun, apabila kondisi pohon tidak bercabang maka akan dilakukan kombinasi k-*itemset* dengan item pada tabel *header* dan hasil kombinasi k-*itemset* akan disimpan dengan menggunakan method `simpan_hasil()`. Jika kondisi pohon bercabang maka akan dibangun *FP-tree* baru dengan menggunakan method `Bangkitkan_FP-tree()`, dan apabila *FP-tree* baru yang dibangkitkan ada, maka dilakukan proses pencarian *n* k-*itemset* paling

menarik kembali. Pada method `simpan_hasil()` nilai batas ambang dukungan minimum dan hasil akan diperbaharui. Diagram aktifitas metode *top-down, middle* atau *bottom-up* disajikan dalam gambar 4.13.



Gambar 4.13 Diagram aktifitas proses *top-down*(), *middle*() atau *bottom-up*()

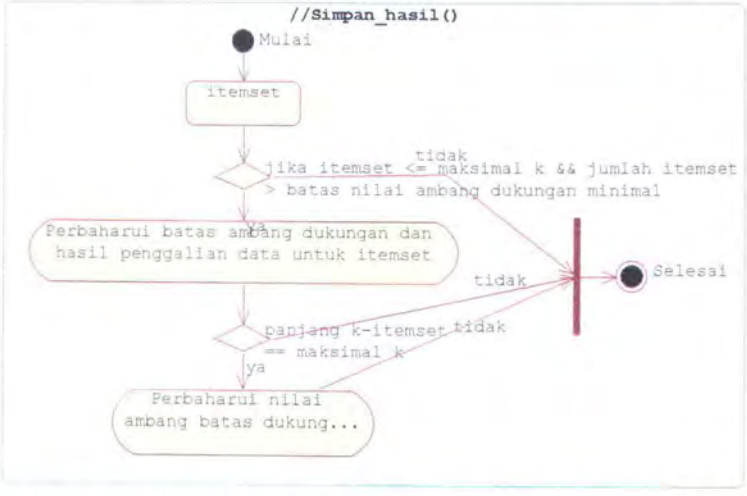
Method `Bangkitkan_FP-tree()` digunakan untuk membuat *FP-tree* baru dari *FP-tree* yang telah dibangkitkan. Tahap method `Bangkitkan_FP-tree()` meliputi pembangkitan *conditional pattern base* dan *conditional FP-tree.* Diagram aktifitas method `Bangkitkan_FP-tree()` ditunjukkan gambar 4.14.



Gambar 4.14 Diagram aktifitas method bangkitkan_FP-tree()

Method `simpan_hasil()` digunakan untuk memperbaharui hasil *n* k-*itemset* paling menarik dan nilai batas ambang dukungan minimum setiap k-*itemset.* Proses pada method ini dimulai ketika k-*itemset* dicek apakah kondisi jumlah

dukungan item pada k-*itemset* lebih kecil sama dengan maksimal k-*itemset* dan k-*itemset* memiliki jumlah lebih besar dari ξ_K, jika benar maka perbaharui ξ_K dan hasil *n* k-*itemset* paling menarik untuk k-*itemset*. Nilai ξ diperbaharui jika maksimal k-*itemset* sama dengan jumlah item pada k-*itemset*. Diagram aktifitas untuk method `simpan_hasil()` ditunjukkan gambar 4.15.



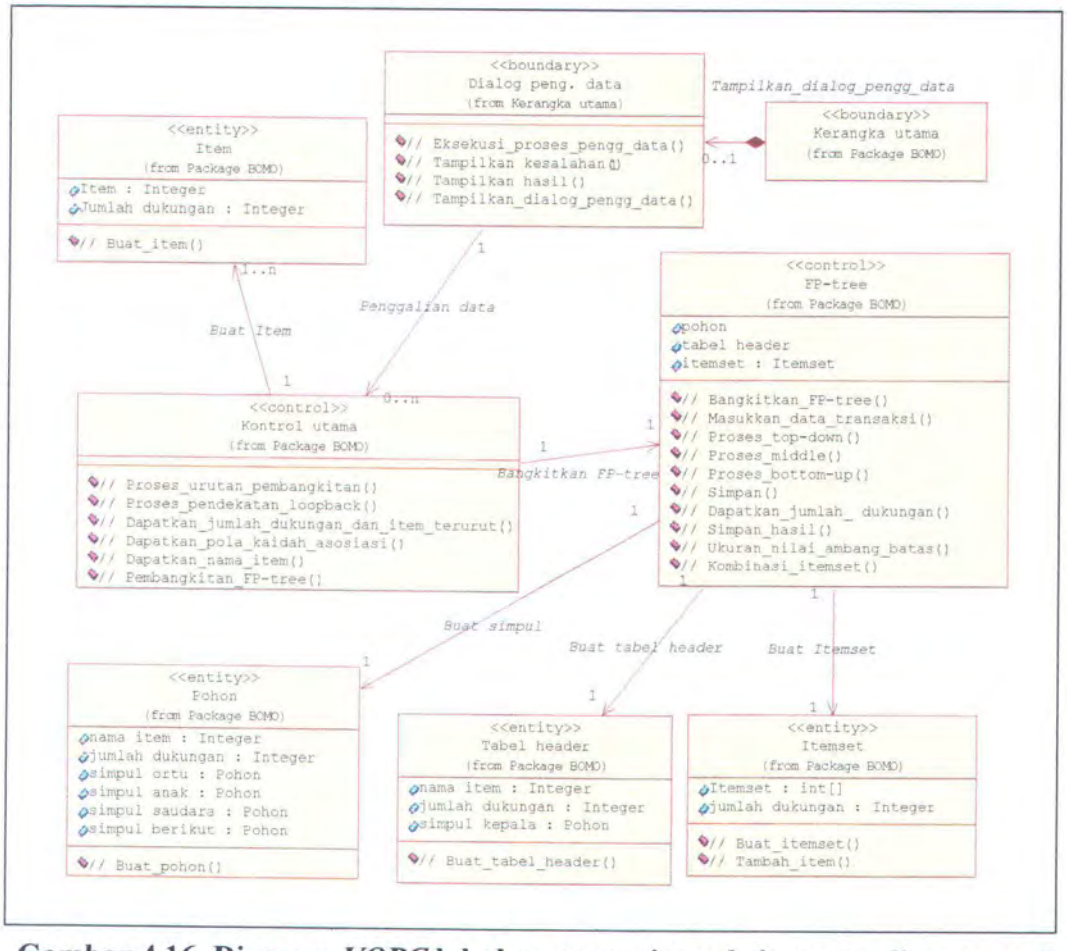
Gambar 4.15 Diagram aktifitas untuk method simpan_hasil()

Seperti yang terlihat pada diagram sekuensi lakukan pencarian *n* k-*itemset* paling menarik, terdapat delapan kelas yang terlibat dalam proses ini antara lain

- Kerangka utama sebagai antarmuka awal dari aplikasi.
- Dialog penggalian data sehingga aktor pengguna dapat menentukan parameter penggalian data.
- Kontrol utama bertanggung jawab terhadap penggalian data untuk mencari jumlah item dan item terurut serta pembangkitan *FP-tree* dari data transaksi.
- Item merupakan kelas entity untuk menyimpan data item terurut beserta jumlah dukungan yang dimiliki. Sehingga himpunan item pada transaksi dapat diurutkan dari besar ke kecil.
- *Itemset* merupakan kelas entity untuk menyimpan hasil *n* k-*itemset* yang paling menarik. Pada kelas ini terdapat method `tambah_item()` yang memungkinkan menyimpan *itemset* hasil kombinasi.

- *FP-tree* merupakan kelas kontrol untuk melakukan pencarian *n* k-*itemset* paling menarik dari data transaksi. Pada kelas ini terdapat method `proses_top_down()`, `proses_middle()` dan `proses_bottom_up()` sehingga aktor pengguna dapat memilih pencaraian sesuai metode yang dinginkan.
- Tabel header merupakan kelas entity untuk menyimpan tabel header dari *FP-tree* yang dibangkitkan.
- Pohon merupakan kelas entity untuk menyimpan ekstraksi data transaksi dalam struktur data pohon.

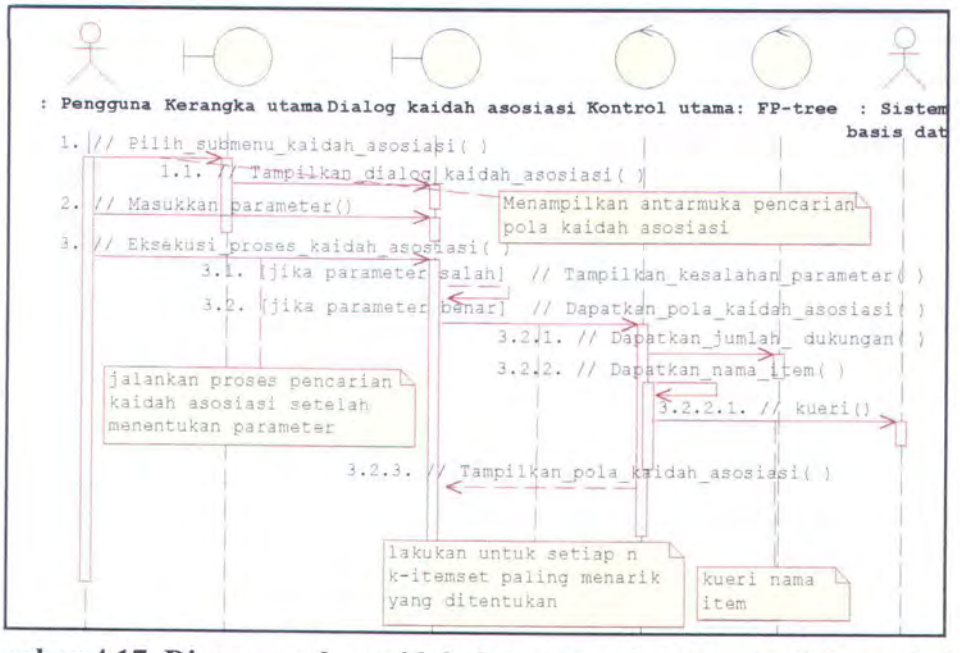
Untuk menggambarkan hubungan delapan kelas ini maka diagram VOPC ditunjukkan pada gambar 4.16



Gambar 4.16 Diagram *VOPC* lakukan pencarian *n* k-*itemset* paling menarik

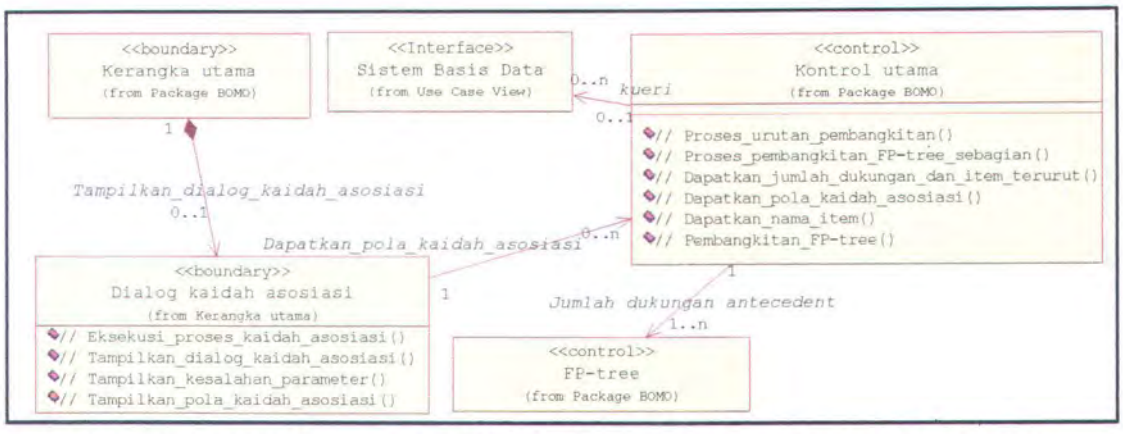
(c) Lakukan pencarian pola kaidah asosiasi

Proses dimulai ketika aktor pengguna meminta pola kaidah asosiasi berdasarkan parameter yang telah ditentukan. Pada saat dieksekusi, aplikasi akan menentukan validasi parameter melalui kelas dialog kaidah asosiasi. Jika parameter valid maka kelas kontrol utama mencari pola kaidah asosiasi. Pada kelas ini dicari nama item, nilai dukungan serta keterpercayaan untuk pola kaidah asosiasi. Sedangkan jumlah dukungan *antecedent* pada pola kaidah asosiasi berdasarkan *FP-tree* yang dibangkitkan. Untuk lebih menjelaskan aliran kejadian pada *use case* ini, sebuah diagram sekuensi lakukan pencarian pola kaidah asosiasi ditunjukkan pada gambar 4.17..



Gambar 4.17 Diagram sekuensi lakukan pencarian pola kaidah asosiasi

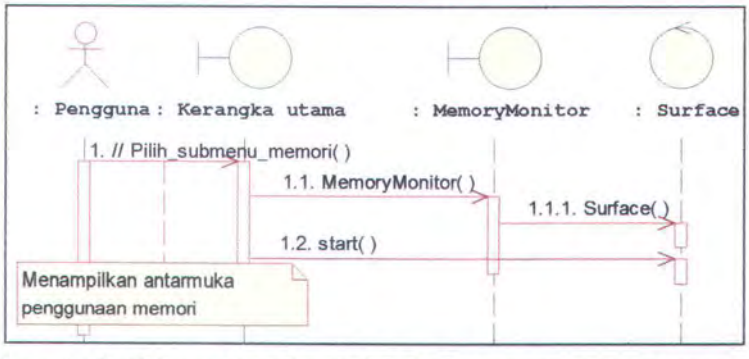
Seperti yang terlihat pada diagram sekuensi lakukan pencarian pola kaidah asosiasi, terdapat empat kelas yang terlibat dalam proses ini yaitu kerangka utama sebagai antarmuka awal dari aplikasi, dialog kaidah asosiasi sehingga aktor pengguna dapat menentukan parameter pencarian pola kaidah asosiasi, kontrol utama bertanggung jawab terhadap pencarian pola kaidah asosiasi dan *FP-tree* untuk mendapatkan nilai jumlah dukungan *antecedent itemset* pada kaidah asosiasi. Untuk menggambarkan hubungan keempat kelas ini maka diagram *VOPC* ditunjukkan pada gambar 4.18.



Gambar 4.18 Diagram *VOPC* lakukan pencarian pola kaidah asosiasi

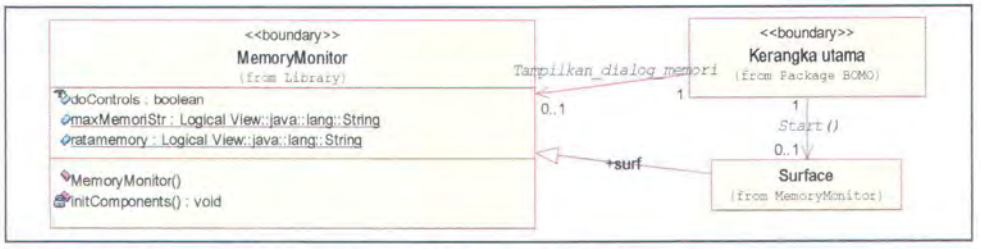
(d) Lakukan pemantauan memori

Ketika pengguna memilih menu "File" → submenu "Memori" maka aplikasi menampilkan antarmuka penggunaan memori. Antarmuka penggunaan memori ditangani khusus oleh kelas `MemoryMonitor`. Kode sumber kelas tersebut berasal dari Sun Microsystems, Inc. Diagram sekuensi lakukan pemantauan memori ditunjukkan pada gambar 4.19.



Gambar 4.19 Diagram sekuensi lakukan pemantauan memori

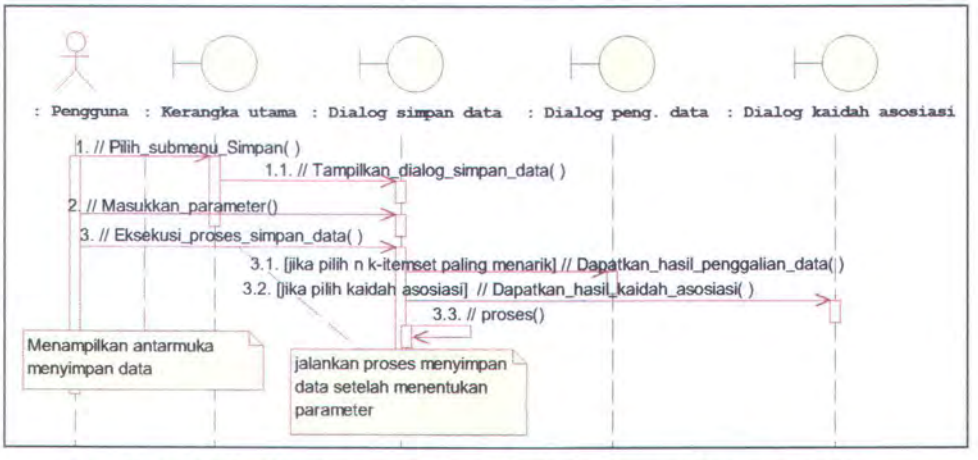
Seperti yang terlihat pada diagram sekuensi lakukan pemantauan memori terdapat tiga kelas yang terlibat dalam proses ini yaitu kerangka utama sebagai antarmuka awal dari aplikasi, `MemoryMonitor` untuk menampilkan antarmuka penggunaan memori dan `surface` merupakan kelas kontrol untuk mendapatkan besar memori yang digunakan. Untuk menggambarkan hubungan ketiga kelas ini maka diagram *VOPC* ditunjukkan pada gambar 4.20.



Gambar 4.20 Diagram *VOPC* lakukan pemantauan memori

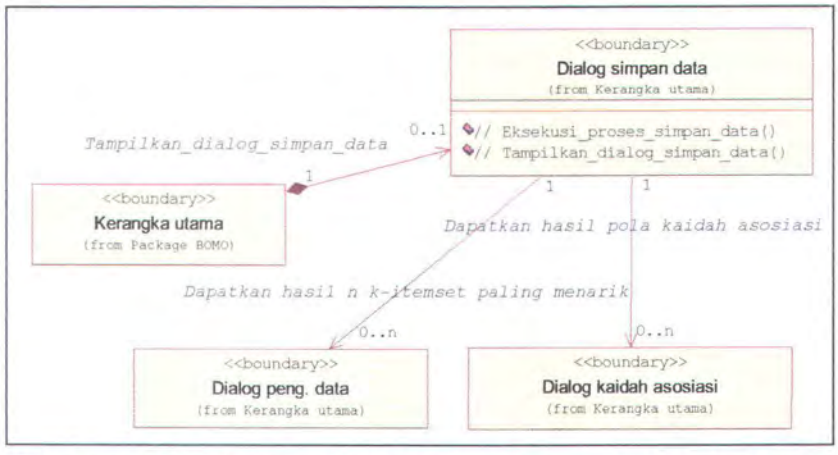
(e) Lakukan penyimpanan data

Proses dimulai saat aktor pengguna memilih menu "File" → submenu "Simpan" pada kerangka utama untuk menampilkan antarmuka simpan data. Ketika *use case* lakukan penyimpanan data dieksekusi, maka aplikasi akan menyimpan data hasil *n* k-*itemset* paling menarik dari dialog penggalian data atau hasil pola kaidah asosiasi dari dialog kaidah asosiasi. Diagram sekuensi menyimpan data ditunjukkan pada gambar 4.21.



Gambar 4.21 Diagram sekuensi lakukan penyimpanan data

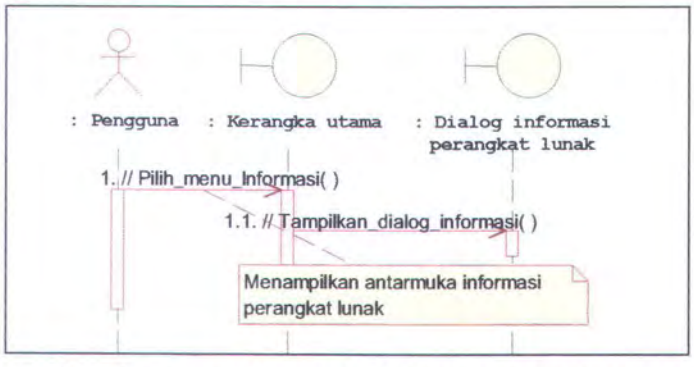
Seperti yang terlihat pada diagram sekuensi lakukan penyimpanan data terdapat empat kelas yang terlibat dalam proses ini yaitu kerangka utama sebagai antarmuka awal dari aplikasi, dialog simpan data sehingga aktor pengguna dapat menentukan parameter simpan data dan bertanggung jawab proses menyimpan data, dialog penggalian data untuk mendapatkan data hasil *n* k-*itemset* paling menarik dan dialog kaidah asosiasi untuk mendapatkan data hasil pola kaidah asosiasi. Untuk menggambarkan hubungan keempat kelas ini maka diagram *VOPC* ditunjukkan pada gambar 4.22.



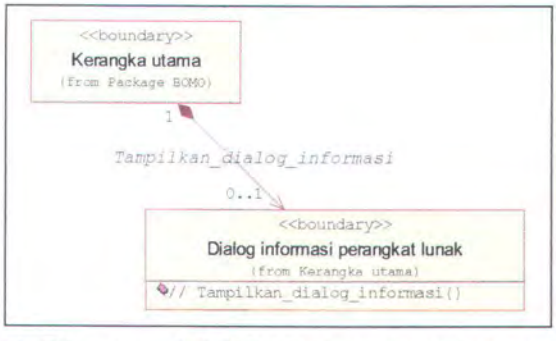
Gambar 4.22 Diagram *VOPC* lakukan penyimpanan data

(f) Lakukan permintaan informasi

Ketika pengguna memilih menu "Informasi" maka aplikasi menampilkan antarmuka informasi aplikasi. Diagram sekuensi Lakukan permintaan informasi ditunjukkan pada gambar 4.23, sedangkan diagram VOPC lakukan permintaan informasi ditunjukkan pada gambar 4.24



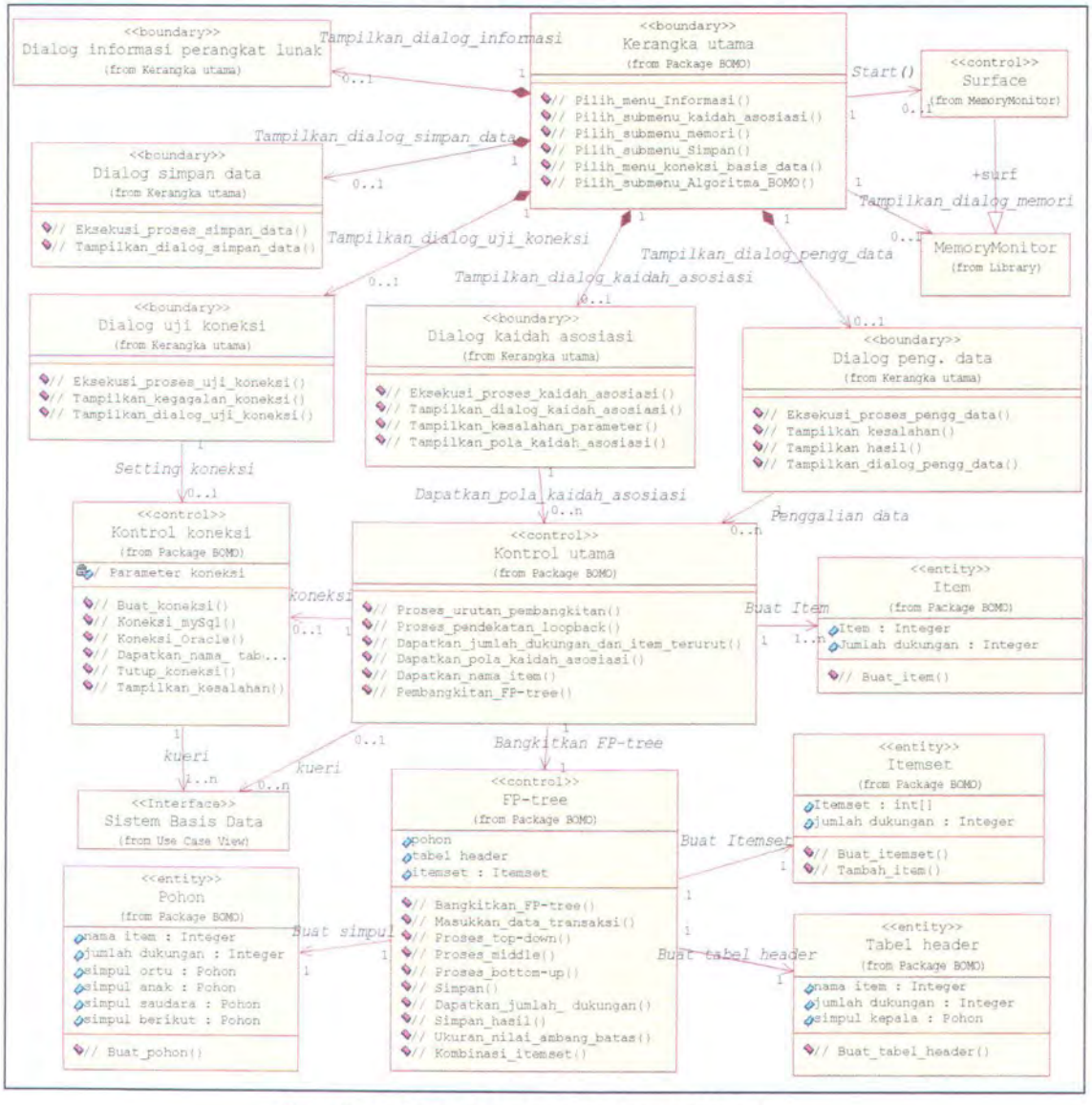
Gambar 4.23 Diagram sekuensi lakukan permintaan informasi



Gambar 4.24 Diagram *VOPC* lakukan permintaan informasi

4.1.3 Diagram Kelas

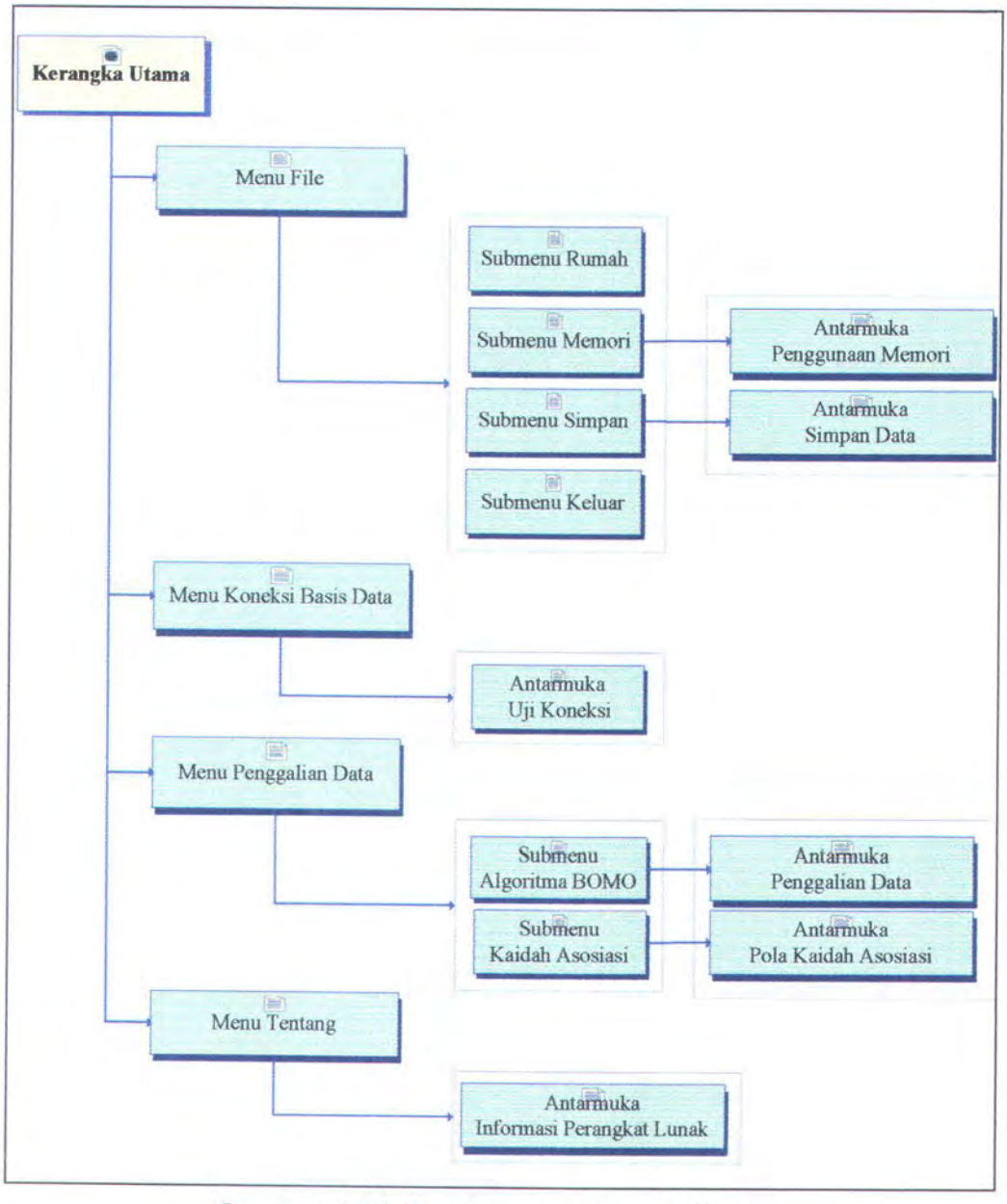
Diagram kelas menggambarkan keseluruhan kelas dari aplikasi yang mengimplementasikan desain diagram *use case*. Terdapat sembilan kelas untuk aplikasi yang dikembangkan seperti yang ditunjukkan pada gambar 4.25.



Gambar 4.25 Diagram kelas aplikasi

4.1.4 Desain Antarmuka

Antarmuka merupakan komponen yang menghubungkan aplikasi dengan aktor pengguna. Desain antarmuka aplikasi ditunjukkan pada gambar 4.26.



Gambar 4.26 Desain antarmuka aplikasi

Berdasarkan desain antarmuka aplikasi pada gambar 4.26 terdapat enam antarmuka aplikasi. Berikut detail keenam antarmuka aplikasi:

(a) Antarmuka kerangka utama

Antarmuka kerangka utama merupakan antarmuka awal aplikasi dijalankan. Antarmuka kerangka utama memiliki menu atau submenu untuk menampilkan dialog antarmuka setiap *use case*.

(b) Antarmuka uji koneksi

Antarmuka uji koneksi merupakan dialog antarmuka *use case* lakukan uji koneksi basis data. Antarmuka ini menangani data masukkan koneksi yaitu:

- Tipe basis data (*String*).

  Merupakan tipe basis data yang digunakan untuk menyimpan data transaksi tiruan, dimana aplikasi hanya menyediakan dua tipe basis data yang dapat dipilih pengguna yaitu *mySql* dan *Oracle*.
- Nama pengguna (*String*).

  Nilai yang dimasukkan merupakan pengguna yang memiliki akses atau koneksi terhadap basis data yang menyimpan data transaksi.
- Kata sandi pengguna (*String*).

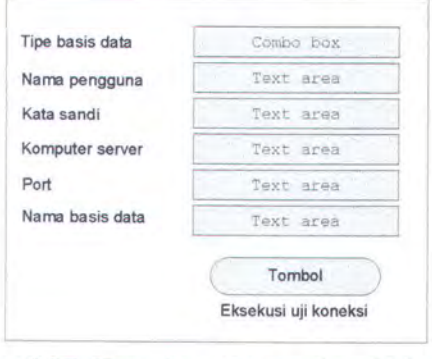
  Merupakan nilai yang menunjang nama pengguna sehingga dapat melakukan koneksi atau akses basis data.
- Komputer server (*String*).

  Nama komputer tempat basis data disimpan. Apabila basis data tersimpan dalam satu komputer maka nilai ini dapat diisikan dengan "*localhost*".
- Port (*Integer*)

  Merupakan jalur koneksi terhadap basis data yang digunakan.
- Nama basis data (*String*).

  Merupakan nama basis data yang digunakan untuk menyimpan tabel sumber pada proses penggalian data.

Berdasarkan kebutuhan data masukkan *use case* lakukan uji koneksi, maka desain antarmuka uji koneksi ditunjukkan pada gambar 4.27



Gambar 4.27 Desain antarmuka uji koneksi

(c) Antarmuka penggalian data

Antarmuka penggalian data merupakan dialog antarmuka *use case* lakukan pencarian *n* k-*itemset* paling menarik. Antarmuka ini menangani data masukkan yaitu:

- Tabel sumber.

  Merupakan nama tabel sumber data transaksi untuk proses penggalian data.
- Pilih metode urutan pembangkitan.

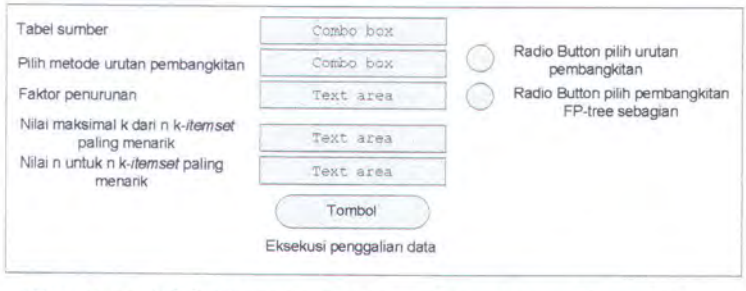
  Penggalian data menggunakan urutan pembangkitan *conditional FP-tree*. Aktor pengguna dapat memlih metode urutan pembangkitan *top down*, *bottom up* atau *middle*.
- Faktor penurunan (*float*).

  Nilai faktor penurunan proses penggalian data menggunakan pendekatan *loopback*.
- Nilai *n* paling menarik (*Integer*).

  Merupakan parameter nilai *n* dari *n* k-*itemset* paling menarik. .
- Nilai maksimal k-*itemset* (*Integer*).

  Merupakan parameter nilai *n* dari *n* k-*itemset* paling menarik.

Berdasarkan kebutuhan data masukkan *use case* lakukan pencarian *n* k-*itemset* paling menarik, maka desain antarmuka penggalian data ditunjukkan pada gambar 4.28.



Gambar 4.28 Desain antarmuka penggalian data

(d) Antarmuka pola kaidah asosiasi

Antarmuka pola kaidah asosiasi merupakan dialog antarmuka *use case* lakukan pencarian pola kaidah asosiasi. Antarmuka ini menangani data masukkan yaitu:

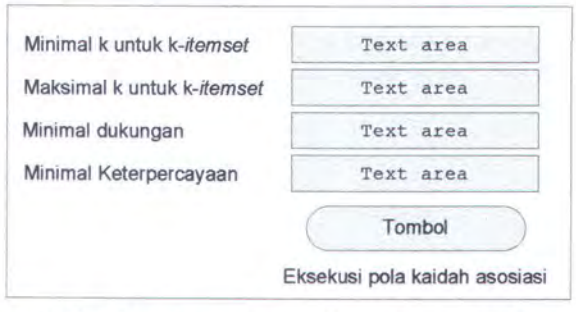
- Nilai k dari *itemset* berpanjang k (*Integer*)

  Merupakan parameter nilai minimum dan maksimal k-*itemset* dari *n* k-*itemset* paling menarik pada pencarian pola kaidah asosiasi.
- Nilai minimum dukungan (*Integer*)

  Memberikan nilai minimum dari dukungan pola kaidah asosiasi yang harus dicari oleh aplikasi.
- Nilai minimum keterpercayaan (*Integer*)

  Memberikan nilai minimum dari keterpercayaan pola kaidah asosiasi yang harus dicari oleh aplikasi

Berdasarkan kebutuhan data masukkan *use case* lakukan pencarian pola kaidah asosiasi, maka desain antarmuka pola kaidah asosiasi ditunjukkan gambar 4.29.



Gambar 4.29 Desain antarmuka pola kaidah asosiasi

(e) Antarmuka penggunaan memori

Antarmuka penggunaan memori merupakan dialog antarmuka *use case* lakukan pemantauan memori. Pada antarmuka ini hanya menampilkan informasi penggunaan memori secara grafik.

(f) Antarmuka simpan data

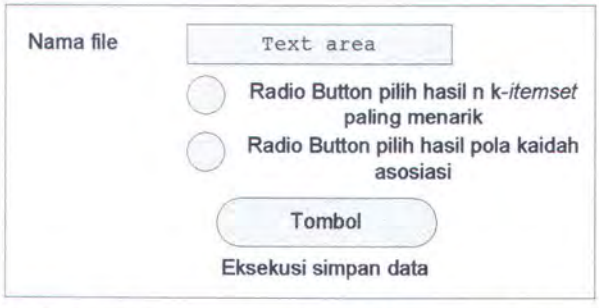
Antarmuka simpan data merupakan dialog antarmuka *use case* lakukan penyimpanan data. Antarmuka ini menangani data masukkan yaitu:

- Nama file (*String*)

  Merupakan nama file untuk menyimpan data hasil.
- Data

  Data hasil pencarian *n* k-*itemset* paling menarik atau pola kaidah asosiasi yang akan disimpan.

Berdasarkan kebutuhan data masukkan *use case* lakukan penyimpanan data, maka desain antarmuka simpan data ditunjukkan pada gambar 4.30.



Gambar 4.30 Desain antarmuka simpan data

(g) Antarmuka informasi aplikasi

Antarmuka informasi aplikasi merupakan dialog antarmuka *use case* lakukan permintaan informasi yang menampilkan informasi aplikasi.

4.2 Implementasi Aplikasi

Sebagaimana desain dilakukan, implementasi aplikasi ini juga dibagi menjadi tiga bagian, yakni implementasi basis data, implementasi kelas, dan implementasi antarmuka. Bahasa pemrograman yang digunakan adalah Java dengan kompiler jdk1.5.0. Implementasi aplikasi ini disarankan dijalankan pada lingkungan yang telah diujicobakan seperti dijelaskan dalam bab 5 (sub-bab 5.1 lingkungan uji coba)

4.2.1 Implementasi Basis Data

Implementasi basis data untuk aplikasi yang dikembangkan membutuhkan format khusus untuk memudahkan proses penggalian data. Secara umum basis data akan memiliki beberapa tabel data transaksi untuk menyimpan data transaksi dan sebuah tabel nama produk untuk nama item. Berikut format tabel basis data untuk aplikasi yang dikembangkan.

(a) Tabel data transaksi

Tabel data transaksi terdiri atas tiga kolom yaitu, kolom *id_key* yang berisikan identitas tiap baris pada tabel, *TID* yang berisikan identitas tiap

transaksi, serta kolom *item* yang berisikan sebuah item yang terkait dalam transaksi. Ketiga kolom tersebut memiliki tipe data *number*. Demi penyederhanaan, nilai dalam *TID* dan *item* akan terurut dari kecil ke besar.

(b) Tabel nama produk

Tabel nama produk terdiri atas dua kolom yaitu, kolom *id_key* yang berisikan identitas item serta kolom *item* yang berisikan nama sebuah item. Kolom *id_key* memiliki tipe data *number*, sedangkan kolom *item* memiliki tipe data *varchar*.

4.2.2 Implementasi Kelas

Diagram kelas menggambarkan keseluruhan kelas dari aplikasi yang mengimplementasikan desain diagram *use case*. Terdapat sembilan kelas untuk aplikasi yang dikembangkan, antara lain:

(a) Kelas Kerangka Utama

Secara umum kelas ini hanya menangani dialog antarmuka aplikasi menggunakan `javax.swing`, akan tetapi terdapat dua kelas internal yang menangani threads sehingga aplikasi dapat berjalan dengan baik. Kedua internal kelas tersebut yaitu

- Kelas `LoadProcess extends Thread`

  Kelas ini berfungsi agar dialog antarmuka dapat berjalan baik saat perngkat lunak melakukan eksekusi pencarian *n itemset* berpanjang k yang paling menarik.

- Kelas `assocProcess extends Thread`

  Kelas ini berfungsi agar dialog antarmuka dapat berjalan baik saat perngkat lunak melakukan eksekusi pencarian pola kaidah asosiasi.

Pada segmen implementasi aplikasi kelas ini akan disebut sebagai `JFrame`*`BOMO`*. Selanjutnya setiap method yang ada pada kelas `JFrame`*`BOMO`* seperti yang ditunjukkan pada tabel 4.9

Tabel 4.9 Penjelasan method kelas `JFrameBOMO`.

Method	Keterangan
`JFrameBOMO()`	Konstruktor pada kelas `JFrameBOMO`
`initComponents()`	Inisialisasi komponen aplikasi untuk setiap dialog antarmuka pada *use case*.
`JMenuItemHome()`	Menampilkan kerangka utama
`setJInternal-FrameConnect()`	Menampilkan dialog antarmuka *use case* lakukan uji koneksi basis data
`setBOMOAlgoritma()`	Menampilkan dialog antarmuka *use case* lakukan pencarian *n* k-*itemset* paling menarik
`setJPanesAssosiation()`	Menampilkan dialog antarmuka *use case* lakukan pencarian pola kaidah asosiasi.
`JmenuItemSave()`	Menampilkan dialog antarmuka *use case* lakukan penyimpanan data.
`visibleMemoryUse()`	Menampilkan dialog antarmuka *use case* lakukan pemantauan memori
`setMenuAbout()`	Menampilkan dialog antarmuka *use case* lakukan permintaan informasi.
`JMenuItemExit()`	Menjalankan fungsi untuk keluar dari aplikasi
`JButtonTestConnect ActionPerformed()`	Menjalankan eksekusi *use case* lakukan uji koneksi basis data
`JButtonEksekusiBOMO ActionPerformed()`	Menjalankan eksekusi *use case* lakukan pencarian *n* k-*itemset* paling menarik
`AssociationRule()`	Menjalankan eksekusi *use case* lakukan pencarian pola kaidah asosiasi
`JButtonTestConnect1 MouseClicked()`	Menjalankan eksekusi *use case* lakukan penyimpanan data.

Berdasarkan tabel 4.9 kelas `JFrameBOMO` terdapat empat method proses eksekusi yaitu method `JButtonTestConnectActionPerformed()`, `JbuttonEk-sekusiBOMOActionPerformed()`, `AssociationRule()` dan `JbuttonTest-Connect1MouseClicked()`. Berikut detail penjelasan dari keempat method tersebut :

- **Method `JButtonTestConnectActionPerformed()`**

  Implementasi ini digunakan oleh aktor pengguna untuk eksekusi uji koneksi basis data. Aktor pengguna terlebih dahulu menentukan parameter koneksi basis data. Parameter koneksi akan disimpan pada kelas connectDB, sehingga untuk proses selanjutnya aplikasi dapat memiliki koneksi yang sama. Jika

koneksi berhasil, maka aplikasi akan menutup koneksi tersebut. Kode sumber method `JButtonTestConnectActionPerformed()` yang digunakan untuk eksekusi *use case* lakukan uji koneksi basis data seperti yang ditunjukkan pada segmen program 4.1.

```
(1) private void JButtonTestConnectActionPerformed
    (java.awt.event.ActionEvent evt){
(2) try {
(3)     this.JComboBoxSource.removeAllItems();
(4)     int datType=JComboDatType.getSelectedIndex();
(5)     String username = JTextFieldUserName.getText();
(6)     String password = JPassFieldPassword.getText();
(7)     String server = JTextFieldServerName.getText();
(8)     String database = JTextFieldDatabase.getText();
(9)     int port=Integer.parseInt(JTextFieldPort.getText());
(10) connDB= new connectDB(username, password, server,
    database, port, datType);
(11) connDB.closeDB();
(12) }catch(Throwable e){
(13) this.JTextAreaConnect.setText(e.getMessage());
(14) }}
```

Segmen program 4.1 Method `JButtonTestConnectActionPerformed()`

- **Method `JButtonEksekusiBOMOActionPerformed()`**

Implementasi ini digunakan oleh aktor pengguna untuk eksekusi pencarian *n* k-*itemset* paling menarik. Ketika method `JButtonEksekusiBOMOAction-Performed()` dijalankan maka akan memanggil kelas `LoadProcess`. Aktor pengguna terlebih dahulu menentukan parameter pencarian *n* k-*itemset* paling menarik (segmen program 4.2 baris 4-7). Proses pencarian *n* k-*itemset* paling menarik berdasarkan parameter yang ditentukan, jika aktor pengguna memilih urutan pembangkitan *conditional FP-tree* maka dijalankan `bomo.eksekusi(string, string)` (segmen program 4.2 baris 4-7), sedangkan jika memilih pendekatan *loopback* maka dijalankan `bomo.eksekusi (string, double)` (segmen program 4.2 baris 4-7). Informasi kesalahan parameter akan ditampilkan oleh antarmuka aplikasi (segmen program 4.2 baris 4-7). Kode sumber kelas `LoadProcess` ditunjukkan pada segmen program 4.2

```
(1) class LoadProcess extends Thread{
(2) public void run() {
(3) try {demo.surf.setmaxMemori();
(4)  int Kmax = Integer.parseInt(JTextFieldK_max.getText());
(5)  int N_Value=Integer.parseInt(JTextFieldN_Value.getText());
(6)  String ConsOrder =
  String.valueOf(JComboBoxMining.getSelectedItem());
(7)  String BasisData =
  String.valueOf(JComboBoxSource.getSelectedItem());
(8)    if(Kmax==0||N_Value==0)JTextAreaResult.setText("Kmax and
  n-most intresting must >0");
(9)    else {
(10)  if(JComboBoxSource.getSelectedItem()!=null){
(11)   bomo.setBOMO(Kmax, N_Value,connDB);
(12) if(JRadioButtonLowerBound.isSelected()==true){
(13)      bomo.eksekusi(ConsOrder,BasisData);
(14)   }else {
(15)      double faktor_f =
  Double.parseDouble(JTextFieldLoopback.getText());
(16)      if(faktor_f>0&&faktor_f<=1){
(17)         bomo.eksekusi(BasisData,faktor_f);
(18)   }else
(19)      JTextAreaResult.setText("Faktor f for Loopback must 0
  < Faktor f < 1");
(20)     }}else JTextAreaResult.append("Please Get Table
  Source\n");
(21) }} catch (Throwable e) {
(22)    System.out.println("LoadProcess failed.....");
(23)    e.printStackTrace();}
(24) JButtonEksekusiBOMO.setEnabled(true);
(25) loadProcess = null;
(26) }}
```

Segmen program 4.2 Kelas `LoadProcess`

- **Method `AssociationRule()`**

Implementasi ini digunakan oleh aktor pengguna untuk eksekusi pencarian pola kaidah asosiasi Ketika method `AssociationRule()` dijalankan maka akan memanggil kelas assocProcess. Aktor pengguna terlebih dahulu menentukan parameter pencarian pola kaidah asosiasi (segmen program 4.3 baris 4-7). Eksekusi proses ini akan dijalankan `bomo.associationRule`(int, int, int, int) (segmen program 4.3 baris 10). Informasi kesalahan parameter akan ditampilkan oleh antarmuka aplikasi. Kode sumber kelas assocProcess ditunjukkan pada segmen program 4.3

```
(1) class assocProcess extends Thread{
(2) public void run(){
(3) try {
(4)    int
   k_item_awal=Integer.parseInt(JTextFieldKitem.getText());
(5)    int
   k_item_akhir=Integer.parseInt(JTextFieldKitem1.getText());
(6)    int support=Integer.parseInt(JTextFieldSupport.getText());
(7)  int
   confidence=Integer.parseInt(JTextFieldConfidence.getText());
(8) ProgressBar1.setValue(0);JButtonEksekusiBOMO1.setEnabled(fals
   e);
(9)    if(k_item_awal<=k_item_akhir)
(10)       bomo.associationRule(k_item_awal, k_item_akhir,
   support, confidence);
(11)      JButtonEksekusiBOMO1.setEnabled(true);
(12) }catch (Throwable e) {
(13)      System.out.println("assocProcess failed.....");
(14)      e.printStackTrace();}
(15) JButtonEksekusiBOMO1.setEnabled(true);assocProcess=null;}}
```

Segmen program 4.3 Kelas `assocProcess`

- **Method `JButtonTestConnect1MouseClicked()`**

Implementasi ini digunakan oleh aktor pengguna untuk menyimpan data . Aktor pengguna dapat menyimpan hasil *n* k-*itemset* paling menarik (segmen program 4.4 baris 5) atau pola kaidah asosiasi (segmen program 4.4 baris 7). Selain menentukan data yang disimpan, maka aktor pengguna harus mengisikan nama file (segmen program 4.4 baris 8). Kode sumber method `JButtonTestConnect1MouseClicked()` untuk eksekusi *use case* lakukan penyimpanan data seperti yang ditunjukkan pada segmen program 4.4

```
(1) private void
  JButtonTestConnect1MouseClicked(java.awt.event.MouseEvent
  evt) {
(2)     try{
(3)         String save;
(4)         if(this.JRadioButtonMining.isSelected())
(5)             save = JFrameBOMO.JTextAreaResult.getText();
(6)         else
(7)             save
  =JFrameBOMO.JTextAreaAssociation.getText();;
(8)         String nameFile=this.JTextFieldSave.getText();
(9)         FileWriter input = new FileWriter(new
  File(nameFile));
```

Segmen program 4.4 Method `JButtonTestConnect1MouseClicked()`

MILIK PERPUSTAKAAN ITS

```
(10) System.gc();
(11) Input.write("")+save);
(12) input.flush();
(13)
(14)     }catch(Exception e){
(15)         }
(16)     }
```

Segmen program 4.4 Method `JButtonTestConnect1MouseClicked()` (lanjutan)

(b) Kelas Kontrol Koneksi

Sebuah kelas yang menyediakan fasilitas koneksi basis data. Secara default basis data yang digunakan adalah Oracle, akan tetapi pengguna dapat memilih basis data mySql. Untuk melakukan koneksi terhadap *Oracle*, aplikasi menggunakan koneksi yang dibangun dengan komponen *JDBC* yang membutuhkan package *classes 12* dari java. Sedangkan untuk melakukan koneksi mySql menggunakan driver *org.gjt.mm.mysql.Driver* yang membutuhkan *package mm.mysql-2.0.4* dari java.

Pada implementasi aplikasi kelas ini akan disebut sebagai `connectDB`. Untuk lebih jelasnya maka fungsi dari method pada kelas `connectDB` ditunjukkan pada tabel 4.10.

Tabel 4.10 Penjelasan method kelas `connectDB`.

Method	Keterangan
`connectDB()`	Method konstruktor pada kelas `connectDB`
`connect()`	Method untuk memilih koneksi berdasarkan tipe basis data.
`connectMysql()`	Melakukan koneksi terhadap basis data mySql
`connectOracle()`	Melakukan koneksi terhadap basis data Oracle
`closeDB()`	Menutup koneksi basis data.
`getTable()`	Mendapatkan nama tabel pada basis data

Pengguna menjalankan aplikasi terlebih dahulu, kemudian membuka dialog uji coba koneksi basis data. Setelah parameter koneksi basis data ditentukan, maka pengguna dapat melakukan eksekusi. Pada saat eksekusi, kelas `JFrameBOMO` akan memanggil method `connectDB()` pada kelas `connectDB`. `Method tersebut` bertanggung jawab terhadap parameter koneksi basis data. Method `connectDB()`

akan menjalankan method `connect()` yang digunakan untuk menentukan basis data menggunakan Oracle atau mySql. Apabila koneksi basis data berhasil maka aplikasi akan menutup koneksi dengan memanggil method `closeDB()`. Untuk lebih jelasnya, kode sumber method `connect()` seperti yang ditunjukkan pada segmen program 4.5

```
(1) public void connect(){
(2)         if(typeDB==0)this.connectMysql();
(3)         else this.connectOracle();
(4) }
```

Segmen program 4.5 Method `connect()`

Method `connect()` untuk menentukan koneksi terhadap *mySql* akan memanggil method `connectMysql()` sedangkan untuk koneksi terhadap *Oracle* akan memanggil method `connectOracle()`. Kode sumber untuk method `connectOracle()` ditunjukkan pada segmen program 4.6 dan kode sumber untuk method `connectMysql()` ditunjukkan pada segmen program 4.7. Informasi koneksi akan ditampilkan dengan menggunakan `javax.swing.JTextArea`. berdasarkan variabel `infoConn`.

```
(1) private void connectMysql() {
(2) String infoConn = null;
(3) String url= "jdbc:oracle:thin:@//"+server+":"+jalur+"/"+basis
    data;
(4) try{
(5)    DriverManager.registerDriver (new
    oracle.jdbc.driver.OracleDriver());
(6)    con=DriverManager.getConnection(url,penggunaname,
    password);
(7) ................
(8) }
```

Segmen program 4.6 Method `connectOracle()`

```
(1) private void connectOracle() {
(2) String infoConn = null;
(3) String url = "jdbc:mysql://"+server+":"+jalur+"/"+basis
    data+"";
(4) try {Class.forName(class_driver);
(5)    con = DriverManager.getConnection(url, penggunaname,
    password);
(6) .............
(7) }
```

Segmen program 4.7 Method `connectMysql()`

(c) Kelas Kontrol Utama

Merupakan kelas yang digunakan untuk menangani aktifitas pencarian *n* k-*itemset* paling menarik dan pola kaidah asosiasi. Pada implementasi aplikasi kelas ini disebut kelas `Bomo`. Kelas *BOMO* menghubungkan kelas `JFrame`*BOMO* dengan kelas–kelas lain pada aplikasi.. Untuk lebih jelasnya maka fungsi setiap method pada kelas *BOMO* ditunjukkan tabel 4.11.

Tabel 4.11 Penjelasan method kelas BOMO

Method	Keterangan
`BOMO()`	Konstruktor pada kelas *BOMO*
`getItem()`	Mendapatkan nama item pada basis data
`associationRule()`	Proses pencarian pola kaidah asosiasi beserta dukungan dan keterpercayaan
`countItemOccurrences()`	Mendapatkan jumlah dukungan dan item terurut
`ScanRec()`	Membangkitkan *FP-tree* dari ekstraksi data transaksi untuk proses penggalian data dan pola kaidah asosiasi
`ItemDescending()`	Mengurutkan himpunan item pada data transaksi berdasarkan urutan item.
`eksekusi()`	Eksekusi proses penggalian data untuk mencari *n* k-*itemset* paling menarik
`setBOMO()`	Inisialisasi atribut pada kelas `BOMO`
`home()`	Mengembalikan pada kondisi semula

Kelas *BOMO* merupakan kelas control dari aktifitas proses penggalian data dan kaidah asosiasi. Sehingga berikut ini akan diuraikan setiap aktifitas proses :

(a) Pencarian *n* k-*itemset* paling menarik.

Implementasi ini merupakan implementasi yang paling kompleks karena terdiri dari beberapa proses. Terlebih dahulu aktor pengguna menentukan parameter proses pencarian *n* k-*itemset* paling menarik yaitu menentukan nama tabel data transaksi, memilih metode penggalian data, menentukan nilai *n* dan maksimal k-*itemset* dari *n* k-*itemset*. Agar parameter proses pencarian *n* k-*itemset* paling menarik benar maka dijalankan kelas `LoadProcess extends Thread` pada `JFrame`*BOMO*.

Berdasarkan sub-bab analisis *use case* lakukan pencarian *n* k-*itemset* paling menarik, maka secara umum implementasi proses pencarian *n* k-*itemset* paling menarik memiliki 3 tahap utama yaitu

1) Dapatkan jumlah item dan item terurut

Tahap ini dilakukan untuk mendapatkan jumlah dukungan dan item terurut, yang kemudian disimpan pada tabel *header*. Tahap ini dicari dengan menggunakan method `countItemOccurrences()`. Di awal tahap, akan ditentukan banyaknya item pada basis data kemudian disimpan dalam variable `int longItem` (segmen program 4.8 baris 5). Sebuah array `countItem` dengan panjang `longItem` diinisialisasi (segmen program 4.8 baris 6), array ini digunakan untuk menyimpan jumlah dari setiap item yang ada. Kueri kedua kalinya terhadap basis data diperlukan untuk menentukan nilai `countItem` (segmen program 4.8 baris 8). Pada tahap pembangkitan *FP-tree* diperlukan item terurut dari besar ke kecil berdasarkan jumlahnya, sehingga `countItem` akan diurutkan dan disimpan dalam array first_Item (segmen program 4.8 baris 13-20). Untuk lebih jelasnya kode sumber method `countItem-Occurrences()` ditunjukkan pada segmen program 4.8.

```
(1) private void countItemOccurrences(String table){
(2)     Item[] temp=null;ResultSet rs;int longItem=0;maxItem=0;
(3)     try{ Statement stm = conn.con.createStatement();
(4)       rs =  stm.executeQuery("Select max(Item) FROM
 "+table);
(5)       while(rs.next())longItem=rs.getInt(1);
(6)       countItem = new int[longItem];
(7)       temp = new Item[longItem];
(8)       rs =  stm.executeQuery("Select Item From "+table+"");
(9)       while(rs.next())countItem[rs.getInt(1)-1]++;
(10)       rs.close();stm.close();
(11)     }catch (SQLException e){
(12)         JFrameBOMO.JTextAreaResult.append("Error first
 scanning database!!!\n"+e+"\n" );}
(13)       for(int i=longItem-1;i>=0;i--)
(14)         if(countItem[i]>0)maxItem++;
(15)           first_Item = new Item[maxItem];
(16)       for(int h=longItem-1;h>=0;h--)
(17)           temp[h]=new Item(h+1,countItem[h]);
(18)       Arrays.sort(temp);
(19)           for(int j=maxItem-1;j>=0;j--)
(20)       first_Item[maxItem-1-j]=temp[longItem-maxItem+j];}
```

Segmen program 4.8 Method `countItemOccurrences()`

2) Pembangkitan *FP-tree*

Pembangkitan *FP-tree* bertujuan untuk mengekstraksi data transaksi ke dalam sebuah pohon yang tersimpan dalam memori, sehingga dapat digunakan pada proses pencarian *n* k-*itemset* paling menarik. Proses ini dilakukan dengan menjalankan method `ScanRec()`. Di awal tahap ini, akan terlebih dahulu diinisialisasi sebuah *FP-tree* untuk item yang terurut atau `first_Item` (segmen program 4.8 baris 5), proses ini akan membangkitkan tabel *header* dan pohon dengan simpul akar berniali "*null*". Kemudian dilakukan inisialisasi hasil atau *data* $result_K$ (segmen program 4.8 baris 5). Pembacaan basis data dilakukan untuk mendapatkan data transaksi (segmen program 4.8 baris 5), kemudian himpunan item pada setiap transaksi diurutkan berdasarkan urutan item di `first_Item` dengan menggunakan method `ItemDescending()` (segmen program 4.8 baris 5). Setiap himpunan item transaksi yang telah terurut akan disimpan pada pohon dengan menggunakan method `insert()`(segmen program 4.8 baris 5). Untuk lebih jelasnya method `ScanRec()` ditunjukkan pada segmen program 4.8.

```
(1) ScanRec(){
(2) ................
(3) fpt = new FP3(item);
(4) fpt.setFP3(nInteres, kmax,first_Item);
(5) ArrayList temp = new ArrayList();
(6) int []ADescItem;ResultSet rs;
(7) try{
(8) Statement stm = conn.con.createStatement();
(9) rs =  stm.executeQuery("Select TID, Item From "+table+"
  ORDER BY TID, Item");
(10) int Tid=1, cekTid=0,temp1=0;
(11) while(rs.next()){
(12) temp1=Integer.parseInt(rs.getString(1));
(13) if(temp1 > Tid){
(14) ADescItem = new int[temp.size()];
(15) ADescItem = ItemDescending1(temp);
(16) fpt.insert(ADescItem,1);
(17) numTransaksi++;temp.clear();Tid=temp1;cekTid=0;}
(18) temp.add(cekTid++,rs.getInt(2));}
(19) numTransaksi++;ADescItem = new int[temp.size()];
(20) ADescItem = ItemDescending1(temp);
(21) fpt.insert(ADescItem,1);
(22) temp.clear();rs.close();stm.close();
(23) ................)
```

Segmen program 4.9 Method `ScanRec()`

3) Proses pencarian *n* k-*itemset* paling menarik

Sesuai desain proses penggalian data maka aktor pengguna dapat memilih pencarian *n* k-*itemset* paling menarik yaitu melalui urutan pembangkitan *conditional FP-tree* atau melalui pendekatan *loopback.* Apabila aktor pengguna memilih melalui pendekatan *loopback*, maka aplikasi akan mengeksekusi method `eksekusi(String,double)`, jika aktor pengguna memilih melalui urutan pembangkitan *conditional FP-tree* maka aplikasi akan mengeksekusi method `eksekusi(String,String)`. Kedua method tersebut berada dalam kelas *BOMO*.

Pada proses pencarian *n* k-*itemset* paling menarik menggunakan pendekatan *loopback* membangkitkan pohon dengan item tertentu (segmen program 4.10 baris 6). Pohon yang dibangkitkan dilakukan proses pencarian *n* k-*itemset* paling menarik dengan memanggil method `Top_Down` pada kelas `FP3` (segmen program 4.10 baris 7). Seperti penjelasan pada bab 3, penggalian data dengan metode ini dilakukan berulang kali sampai ditemukan *n* k-*itemset* paling menarik (segmen program 4.10 baris 11). Apabila belum ditemukan maka ditentukan nilai batas ambang dukungan baru yaitu $\xi_{baru} = \xi_{lama} \times$ faktor penurunan f (segmen program 4.10 baris 14). Dengan nilai ξ_{baru}, dibangkitkan *FP-tree* baru menggunakan method `ScanRec()` yang kemudian dilakukan penggalian lagi (segmen program 4.10 baris 16). Setiap penggalian hanya melakukan perbaharuan hasil pencarian dan nilai batas ambang dukungan, karena nilai tersebut selalu disimpan menggunakan method `setresult()` dan method `setThreshold()` pada kelas `FP3` (segmen program 4.10 baris 17-20). Untuk lebih jelasnya method `eksekusi(String,double)` ditunjukkan pada segmen program 4.10

```
(1) public void eksekusi(String table, double faktor_f){
(2) .......
(3) this.countItemOccurrences(table);
(4) .......
(5) else{
(6)     this.ScanRec(table,true);
(7)     fpt.Top_Down(new Kitemsets(),false,0);
```

Segmen program 4.10 Method `eksekusi(String,double)`

```
(8)     int thersholdnull =fpt.getThresholdnull();
(9)     Kitemsets [][]resultLoopback;int []thersholdLoopback;
(10)     if(first_Item[first_Item.length-1].count > 0&&
 first_Item.length > nInteres){
(11)        while (thersholdnull< this.border&&this.border>0
 &&first_Item[first_Item.length-1].count <=t his.border){
(12) resultLoopback=fpt.getresult();
(13)            thersholdLoopback =fpt.getThreshold();
(14)            this.border = (int)Math.ceil((double)
 this.border*faktor_f);
(15)    int index =fpt.headerTable.length;fpt=null;System.gc();
(16)            this.ScanRec(table,false);
(17)            fpt.setresult(resultLoopback);
(18)            fpt.setThreshold(thersholdLoopback);
(19)            fpt.Top_Down(new Kitemsets(),false,index);
(20)            thersholdnull =fpt.getThresholdnull();
(21) if(first_Item[first_Item.length-
 1].count==1&&this.border==1)break; }}
(22) .............
(23) }
```

Segmen program 4.10 Method `eksekusi(String,double)` (lanjutan)

Proses pencarian *n* k-*itemset* paling menarik menggunakan urutan pembangkitan, setelah *FP-tree* dibangkitkan (segmen program 4.11 baris 5) maka terlebih dahulu ditentukan ukuran awal nilai batas ambang dukungan minimum dengan menggunakan `lowerBound()`(segmen program 4.11 baris 6). Penggunaan method `lowerBound()` berdasarkan kinerja algoritma *BOMO* pada bab 3. Pencarian *n* k-*itemset* paling menarik dilakukan sesuai urutan pembangkitan *conditional FP-tree* yang ditentukan oleh aktor pengguna (segmen program 4.11 baris 7-8). Untuk lebih jelasnya method `eksekusi(String,String)` ditunjukkan pada segmen program 4.11.

```
(1) public void eksekusi(String ConsOrder,String table){
(2)     ........
(3)         this.countItemOccurrences(table);
(4)     ..........
(5) else{this.ScanRec(table);
(6) if(fpt!=null){fpt.lowerBound();
(7) if(ConsOrder.equals("Top Down"))fpt.Top_Down(new
 Kitemsets(),true,0);
(8) else if(ConsOrder.equals("Bottom Up"))fpt.Bottom_up(new
 Kitemsets(),true);
(9) else if(ConsOrder.equals("Middle"))fpt.middle(new
 Kitemsets(),true);}
(10) ......................
(11) }
```

Segmen program 4.11 Method `eksekusi(String, String)`

(b) Proses pencarian kaidah asosiasi

Implementasi ini menjelaskan bagaimana aplikasi melakukan proses pencarian pola kaidah asosiasi. Implementasi ini dapat dijalankan apabila aktor pengguna telah melakukan proses pencarian *n* k-*itemset* paling menarik, karena hasil proses tersebut digunakan sebagai data masukan untuk proses ini. Untuk mencari pola kaidah asosiasi, terlebih dahulu aktor pengguna menentukan *k-itemset,* dukungan dan keterpercayaan minimum. Agar parameter pencarian pola kaidah asosiasi benar maka dijalankan kelas `LoadProcess extends Thread` pada `JFrameBOMO`. Apabila parameter benar, maka kelas `LoadProcess` akan memanggil method `associationRule()` pada kelas `BOMO` untuk melakukan eksekusi pencarian pola kaidah asosiasi..

Pada tahap ini proses dilakukan berulang untuk setiap k-*itemset* (segmen program 4.11 baris 3) dan *n* paling menarik (segmen program 4.11 baris 6). Setiap pola kaidah asosiasi akan didapatkan dengan melakukan pencarian secara permutasi (segmen program 4.11 baris 9-21). Kemudian nilai keterpercayaan dicari dengan menggunakan method `setSupportAssociation()` pada kelas `FP3` (segmen program 4.11 baris 33), sedangkan nilai dukungan didapatkan dengan membagi jumlah yang dimiliki *itemset* dengan jumlah data transaksi (segmen program 4.11 baris 34-37). Setelah pencarian pola kaidah asosiasi selesai, maka aplikasi akan menampilkannya kepada aktor pengguna hasil pencarian dengan cara visualisasi menggunakan komponen `javax.swing` (segmen program 4.11 baris 43). Kode sumber method `associationRule()` ditunjukkan pada segmen program 4.12.

```
(1) public void associationRule(int k_item_awal,int k_item_akhir,
    int support, int confidence){
(2)  if(kmax>1&&k_item_awal>1&&kmax >= k_item_awal&&kmax >=
    k_item_akhir&&fpt!=null){
(3)   for(int kayu=k_item_awal;kayu<=k_item_akhir;kayu++){
(4)    int []ResultAssociation = new int[kayu];
(5)    int SupCountItem;
(6)    for(int i=0;i<nInteres;i++){
(7)      if(result[i][kayu-1]!=null){
(8)      SupCountItem=result[i][kayu-1].getsupportCount();
(9)      for (int j=0;j<kayu;j++)
```

Segmen program 4.12 Method `associationRule ()`

```
(10) ResultAssociation[j]=result[i][kayu-1].getItem(j);
(11)        for (int x=0;x<kayu;x++){
(12)         int temp;
(13)         if(x>=1){
(14)           temp= ResultAssociation[0];
(15)           for(int y=0;y<kayu-1;y++)
(16)             ResultAssociation[y]=ResultAssociation[y+1];
(17)             ResultAssociation[kayu-1]=temp;}
(18)      for (int h=0;h<kayu-1;h++){
(19)         Kitemsets prefixResultRule = new Kitemsets();
(20)         Kitemsets suffixResultRule = new Kitemsets();
(21)        for(int letak=h+1;letak<kayu;letak++)
(22)          suffixResultRule.add(ResultAssociation[letak]);
(23)        for(int letak=0;letak<=h;letak++)
(24)          prefixResultRule.add(ResultAssociation[letak]);
(25)        int size =prefixResultRule.size();
(26)        int SupCount=0;
(27)        int []ADescItem=new int[size];
(28)        ArrayList templist = new ArrayList();
(29)        for (int counter=0;counter<size;counter++)
(30)     templist.add(counter,prefixResultRule.getItem(counter));
(31)        ADescItem = ItemDescending1(templist);
(32)        SupCount = fpt.setSupportAssociation(ADescItem);
(33)        float floatSupCount=(float)SupCount;
(34)        float floatSupCountItem=(float)SupCountItem;
(35)        float floatnumTransaksi=(float)numTransaksi;
(36)        if(floatSupCount!=0&&floatnumTransaksi!=0){
(37)        float resultSupport =
    100*floatSupCountItem/floatnumTransaksi;
(38)        float resultConfidence =
    100*floatSupCountItem/floatSupCount;
(39)        if(resultSupport >= (float)support&&resultConfidence
    >= (float)confidence){
(40) String new_prefixResultRule=this.getItem(prefixResultRule);
(41) String new_suffixResultRule=this.getItem(suffixResultRule);
(42) JFrameBOMO.JTextAreaAssociation.append("
    "+new_prefixResultRule+" --> "+new_suffixResultRule+" [ S =
    "+resultSupport+"%, C = "+resultConfidence+"% ]\n" );
(43)   }}}}
(44) }}}
(45) System.gc();
(46) }}
```

Segmen program 4.12 Method `associationRule ()` (lanjutan)

(d) Kelas Item

Merupakan kelas untuk menyimpan nilai item dengan jumlah dukungan. Kelas ini dibutuhkan saat pengurutan item dari besar ke kecil berdasarkan jumlah dukungannya. Segmen program untuk kelas `item` terdapat pada lampiran.

(e) Kelas *FP-tree*

Kelas ini menangani pembangkitan *FP-tree* maupun pencarian *n* k-*itemset* paling menarik. Pada implementasi aplikasi kelas ini disebut sebagai `FP3`. Selanjutnya setiap method yang pada kelas `FP3` ditunjukkan tabel 4.12.

Tabel 4.12 Penjelasan method kelas `FP3`.

Method	Keterangan
`FP3()`	Konstruktor pada kelas `FP3`
`setFP3()`	Inisialisasi komponen `FP3`
`setlowerBound()`	Memberikan ukuran awal nilai batas ambang dukungan pada *FP-tree*
`getThreshold()`	Mengembalikan nilai batas ambang dukungan untuk setiap k-*itemset*.
`setThreshold()`	Memberikan nilai batas ambang dukungan untuk setiap k-*itemset* pada *FP-tree*
`getThresholdnull()`	Mengembalikan nilai batas ambang dukungan
`getlastNode()`	Mendapatkan node terpanjang dari *FP-tree*
`getresult()`	Mengembalikan nilai *n* k-*itemset* paling menarik yang ditemukan
`setresult()`	Memberikan hasil *n* k-*itemset* paling menarik pada *FP-tree* digunakan pada pendekatan *loopback*
`writeResult()`	Memperbaharui hasil *n* k-*itemset* paling menarik dan nilai batas ambang dukungan untuk setiap k-*itemset*.
`insert()`	Memasukkan data transaksi pada struktur data *FP-tree*
`Bottom_up()`	Pencarian *n* k-*itemset* paling menarik secara *bottom-up*
`middle()`	Pencarian *n* k-*itemset* paling menarik secara *middle*
`Top_Down()`	Pencarian *n* k-*itemset* paling menarik secara *top-down*
`KombinasiItem()`	Melakukan kombinasi *itemset*.
`lowerBound()`	Mencari ukuran awal nilai batas ambang dukungan minimum yang terdapat pada *FP-tree*
`NFP_Build()`	Membangkitkan *FP-tree* baru.
`ItemDescending()`	Mengurutkan item dari besar ke kecil berdasarkan jumlah dukungan.
`setSupjalurAssociation()`	Mencari jumlah dukungan *itemset* pada *FP-tree*

Proses pencarian *n* k-*itemset* paling menarik menggunakan urutan pembangkitan *conditional FP-tree*, mewajibkan aktor pengguna memilih metode urutan pembangkitan *top down*, *middle* atau *bottom up*. Kode sumber metode `Top_Down()` ditunjukkan pada segmen program 4.13, kode sumber metode `middle()` ditunjukkan pada segmen program 4.14 sedangkan kode sumber `Bottom_up()` ditunjukkan pada segmen program 4.15.

Proses pencarian *n* k-*itemset* paling menarik menggunakan ketiga metode tersebut memiliki kondisi yang sama, berdasarkan pohon yang dibangkitkan. Jika pohon yang dibangkitkan memiliki satu cabang maka dilakukan kombinasi untuk mendapatkan *n* k-*itemset* paling menarik dengan menggunakan method `KombinasiItem()` (segmen program 4.13 baris 6). Kemudian nilai batas ambang dukungan dan hasil pencarian diperbaharui dengan method `writeResult()`. Jika pohon bercabang, akan dicari *conditional pattern base* dan dibangkitkan *conditional FP-tree* untuk setiap item pada table *header* dengan menggunakan method `NFP_Build()` (segmen program 4.13 baris 13). Jika *FP-tree* yang dibangkitkan tidak kosong maka dilakukan penggalian data (segmen program 4.13 baris 14). Meskipun ketiga metode memiliki kondisi pencarian *n* k-*itemset* paling menarik yang sama tetapi memiliki urutan penggalian yang berbeda, sesuai penjelasan dalam bab 3 mengenai kinerja algoritma *BOMO*. Dimana *top_down* memulai proses pencarian *n* k-*itemset* paling menarik dari item yang memiliki jumlah dukungan terbesar pada tabel *header* (segmen program 4.13 baris 7), sedangkan *middle* memulai proses pencarian *n* k-*itemset* paling menarik dari tengah tabel *header* (segmen program 4.14 baris 3-7) dan *bottom-up* dari item yang memiliki jumlah terkecil pada tabel *header* (segmen program 4.15 baris 3).

```
(1) public void Top_Down(Kitemsets is_suffix,boolean pbar,int
    start){
(2)   if (!hasMultiplePaths){if(is_suffix.size()<kMax){
(3)         int[] items = new int[headerTable.length];
(4)         for (int i = 0; i < headerTable.length; i++)
(5)            items[headerTable.length-i-1]=headerTable[i].item;
(6)               Kombinasi2Item(items, is_suffix);}
```

Segmen program 4.13 Method `top_down()`

```
(7)   }else for (int i=start;i<=headerTable.length-1;i++){
(8)        if(headerTable[i].count>threshold[0]) {
(9)           Kitemsets is_new = new Kitemsets(is_suffix);
(10)          is_new.add(headerTable[i].item);
(11)          is_new.setsupportCount(headerTable[i].count);
(12)          writeResult(is_new);
(13)          FP3 fpt = NFP_Build(headerTable[i].item);
(14)          if (fpt != null)fpt.Top_Down(is_new, false,0);
(15)       }else break;}}
```

Segmen program 4.13 Method `top_down()` (lanjutan)

```
(1) public void middle(Kitemsets is_suffix, boolean pbar){
(2) ..............................
(3) int mid=(int)Math.ceil((double)(headerTable.length-1)/2) ;
(4) int mining=0;
(5) for (int i = 0; i <= headerTable.length - 1; i++){
(6)   if(i<=mid)mining = mid-i;
(7)   else mining =i;
(8) ..............................
(9) }}
```

Segmen program 4.14 Method `middle()`

```
(1) public void Bottom_up(Kitemsets is_suffix, boolean pbar)     {
(2) ...................
(3) for (int i = headerTable.length - 1; i >= 0; i--){
(4)   if(headerTable[i].count>threshold[0]) {
(5) .......................
(6) }}}
```

Segmen program 4.15 Method `Bottom_up()`

Untuk nilai keterpercayaan pola kaidah asosiasi A→B sesuai persamaan 2.1 pada bab 2, maka jumlah dukungan *antecedent* A dan jumlah dukungan $A \cup B$ harus diketahui. Jumlah dukungan $A \cup B$ didapatkan dari proses pencarian *n* k-*itemset* paling menarik, akan tetapi jumlah dukungan *antecedent* A belum tentu bisa didapatkan. Untuk mencari jumlah dukungan *antecedent* A, maka dijalankan method `setSupportAssociation()`. Kode sumber method `setSupportAssociation()` ditunjukkan pada segmen program 4.16.

```
(1) public int setSupportAssociation(int[] itemRule){
(2)    int entry_index= ((Integer)item2index.get(new
    Integer(itemRule[itemRule.length-1]))).intValue();
(3)    int[] counts = new int[entry_index+1];
(4)    for (TreeNode side_walker = headerTable[entry_index].head;
    side_walker != null; side_walker = side_walker.next)
```

Segmen program 4.16 Method `setSupportAssociation()`

```
(5)  for (TreeNode up_walker = side_walker; up_walker != root;
     up_walker = up_walker.parent)
(6)       counts[up_walker.header_index] += side_walker.count;
(7)       int supRule = counts[entry_index];
(8)          if (itemRule.length-2>=0)
(9)              for(int i=itemRule.length-2;i>=0;i--)
(10)               if(supRule>counts[((Integer)item2index.get(new
     Integer(itemRule[i]))).intValue()])
(11)                   supRule=counts[((Integer)item2index.get(new
     Integer(itemRule[i]))).intValue()];
(12) return supRule;
(13) }
```

Segmen program 4.16 Method `setSupportAssociation()` (lanjutan)

(f) Kelas Tabel Header

Merupakan kelas untuk menyimpan struktur data tabel *header*. Pada kelas ini disimpan tiga buah atribut yaitu item dalam objek `integer`, jumlah dukungan dalam objek *`integer`* dan simpul head dalam objek `treenode`. Pada segmen implementasi aplikasi kelas ini akan disebut sebagai `HeaderTable`. Segmen program untuk kelas `HeaderTable` terdapat pada lampiran

(g) Kelas Pohon

Merupakan kelas yang digunakan untuk menangani sebuah struktur data pohon. Pada implementasi aplikasi kelas ini akan disebut sebagai `TreeNode`. Pada kelas `TreeNode` menyimpan data transaksi yang di ekstraksi ke dalam sebuah pohon untuk proses penggalian data

Kelas `TreeNode` menyimpan atribut–atribut yang dapat membentuk keterhubungan antara item seperti simpul `parent`, `simpul child`, `simpul sibling` maupun simpul `next` untuk item yang sama. Setiap simpul akan menyimpan item dan jumlah dukungan serta atribut untuk membentuk keterhubungan antara item. Keterangan atribut–atribut pada simpul seperti pada tabel 4.13, sedangkan segmen program untuk kelas `TreeNode` terdapat pada lampiran.

Tabel 4.13 Keterangan atribut pada simpul

Atribut	Keterangan
`int item`	Nama item yang berada pada simpul itu
`int count`	Jumlah item yang berada pada simpul itu
`int seq_num`	Kedalaman simpul pada pohon
`int header_index`	Menunjukkan index item pada tabel *header*
`TreeNode parent`	Simpul orang tua yang dimiliki simpul itu
`TreeNode child`	Simpul anak yang dimiliki simpul itu
`TreeNode sibling`	Simpul saudara yang dimiliki simpul itu
`TreeNode next`	Struktur kaitan simpul yang dimiliki simpul itu

(h) Kelas Itemset

Kelas ini memiliki fungsi hampir sama dengan kelas item, akan tetapi kelas ini digunakan untuk menyimpan k-*itemset*, sehingga digunakan struktur data `array`. Pada kelas ini memiliki method `add()` untuk melakukan penambahan item pada k-*itemset* menjadi (k+1)-*itemset*. Pada implementasi aplikasi kelas ini akan disebut sebagai `Kitemsets`, untuk method yang ada pada kelas `Kitemsets` ditunjukkan pada tabel 4.14, sedangkan segmen program untuk kelas `Kitemsets` terdapat pada lampiran.

Tabel 4.14 Penjelasan Method kelas `Kitemsets`.

Method	Keterangan
`Kitemsets()`	Konstruktor pada kelas `Kitemsets`.
`getItem()`	Mendapatkan item tertentu pada k-*itemset*
`CompareItemsets()`	Membandingkan kesamaan dari k-*itemset*
`setsupportCount()`	Memberikan jumlah dukungan k-*itemset*
`getsupportCount()`	Mendapatkan jumlah dukungan k-*itemset*
`size()`	Meminta nilai k yang dimiliki k-*itemset*
`add()`	Menambahkan item pada k-*itemset* menjadi (k+1)-*itemset*
`equals()`	Mengecek apakah data inputan berupa k-*itemset*
`toString()`	Menampilkan k-*itemset* dalam bentuk string

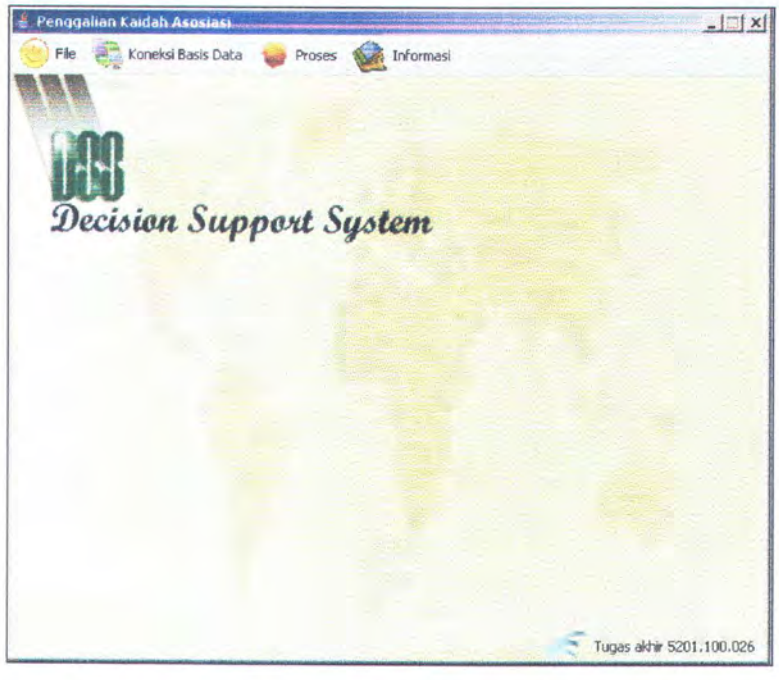
(i) Kelas `MemoryMonitor`

Kelas ini berfungsi sebagai visualisasi penggunaan memori saat aplikasi dijalankan. Kelas ini menggunakan `javax.swing` untuk menampilkan antarmuka penggunaan memori. Agar berfungsi dengan baik, kelas ini memiliki akan tetapi

terdapat kelas internal yaitu `class Surface extends JPanel implements Runnable`. Kelas `surface` digunakan untuk menggambarkan grafik penggunaan memori. Kode sumber dari kelas `MemoryMonitor` didapatkan dari Sun Microsystems, Inc[SUN-99].

4.2.3 Implementasi Antarmuka

Aplikasi ini mengimplementasikan antarmuka berbasis `java.swing`. Aplikasi memiliki satu kerangka utama yang merupakan tampilan awal dari aplikasi yang dikembangkan, seperti ditunjukkan pada gambar 4.31. Pada kerangka utama terdapat menu navigasi (*navigation menu*), seperti yang ditunjukkan pada gambar 4.32. Penjelasan detail menu navigasi diberikan pada tabel 4.15.



Gambar 4.31 Antarmuka kerangka utama

Gambar 4.32 Menu navigasi aplikasi

Tabel 4.15 Penjelasan menu navigasi aplikasi

Menu Navigasi	Keterangan
File	Menu untuk menampilkan dialog antarmuka penggunaan memori, proses menyimpan data dan keluar dari aplikasi.
Koneksi Basis Data	Menu untuk menampilkan dialog antarmuka proses uji coba koneksi basis data.
Proses	Menu untuk menampilkan dialog antarmuka proses penggalian data dan proses pencarian pola kaidah asosiasi.
Informasi	Menu untuk menampilkan informasi aplikasi

(b) Navigasi Menu File

Pada navigasi menu file memiliki empat submenu untuk menunjang kinerja dari aplikasi antara lain:

(1) Submenu Rumah

Digunakan untuk menutup semua dialog antarmuka yang ditampilkan, kecuali kerangka utama.

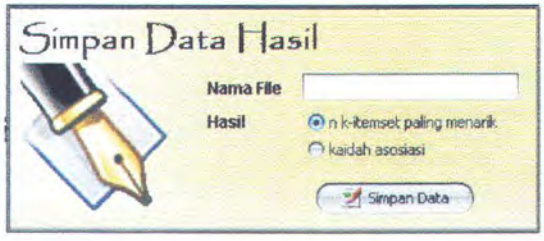
(2) Submenu Memori

Digunakan untuk menampilkan antarmuka penggunaan memori saat aplikasi dijalankan. Implementasi antarmuka *usescase* lakukan pemantauan memori ditunjukkan pada gambar 4.33.

Gambar 4.33 Antarmuka *use case* lakukan pemantauan memori.

(3) Menu Simpan

Digunakan untuk menampilkan antarmuka proses menyimpan data hasil pencarian *n* k-*itemset* paling menarik dan pola kaidah asosiasi.Implementasi antarmuka *usescase* lakukan penyimpan data ditunjukkan pada gambar 4.34.



Gambar 4.34 Antarmuka *use case* lakukan penyimpanan data.

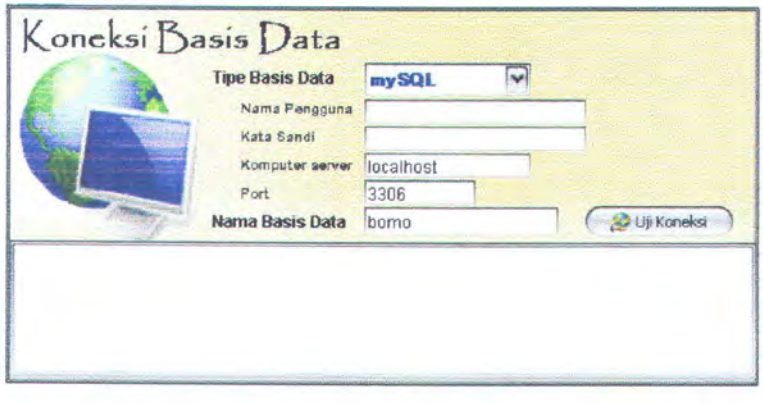
(4) Menu Keluar

Digunakan untuk keluar dari aplikasi

(c) Navigasi Menu Koneksi Basis data

Fungsi dari dialog koneksi basis data ini adalah untuk menerima data masukan yang berkaitan dengan *use case* lakukan uji koneksi dari aktor pengguna. Komponen yang terdapat pada dialog ini antara lain

- 1 *label* dan 1 *Combobox* untuk memilih tipe *basis data* yang digunakan
- 1 *label* dan 1 *text field* untuk data masukan nama pengguna
- 1 *label* dan 1 *password field* untuk data masukan *password* pengguna
- 1 *label* dan 1 *text field* untuk data masukan nama komputer basis data
- 1 *label* dan 1 *text field* untuk menentukan jalur dari komputer basis data
- 1 label dan 1 *text field* untuk data masukan nama *basis data*
- 1 *text area* untuk menampilkan laporan hasil koneksi
- 1 *button* untuk eksekusi koneksi *basis data* menggunakan data masukan yang telah dimasukkan pengguna.



Gambar 4.35 Antarmuka *use case* lakukan uji koneksi

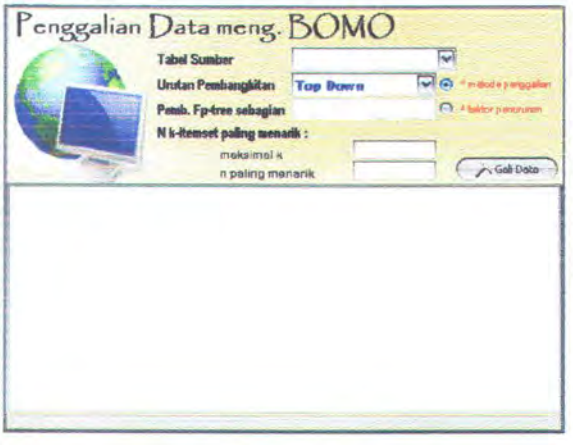
(d) Navigasi Menu Proses

Pada dialog ini merupakan inti dari aplikasi yang dikembangkan. Pengguna ditampilkan dialog untuk melakukan proses pencarian *n* k-*itemset* paling menarik dan dialog untuk melakukan pencarian pola kaidah assosiasi. Berikut penjelasannya :

(1) Submenu Penggalian Data

Fungsi dari dialog ini adalah untuk menerima data masukan yang berkaitan dengan *use case* lakukan pencarian *n* k-*itemset* paling menarik. Komponen yang terdapat pada dialog ini antara lain :

- 1 *label* dan 1 *combo box* untuk data masukkan nama tabel basis data
- 1 *radio button,* 1 *label* dan 1 *combo box* untuk data masukkan metode penggalian data untuk urutan pembangkitan *conditional FP-tree*
- 1 *radio button*, 1 *label* dan 1 *text field* untuk data masukkan factor penurunan pada penggalian data menggunakan pendekatan *loopback.*
- 1 *label* dan 1 *text field* untuk data masukan nilai *n* paling menarik
- 1 *label* dan 1 *text field* untuk data masukan nilai maksimal k-*itemset* dari k-*itemset*
- 1 t*ext area* untuk menampilkan hasil *n* k-*itemset* paling menarik menggunakan algoritma *BOMO*
- 1 *progress bar* untuk menampilkan proses yang dilakukan.
- 1 *button* untuk eksekusi proses penggalian data.



Gambar 4.36 Antarmuka *use case* pencarian *n* k-*itemset* paling menarik

(2) Submenu Kaidah Asosiasi

Fungsi dari dialog ini adalah untuk menerima data masukan yang berkaitan dengan *use case* lakukan pencarian pola kaidah asosiasi. Komponen yang terdapat pada dialog ini antara lain

- 1 *label* dan 2 *text field* untuk data masukan nilai k dari *k-itemset* yang ingin dicari kaidah asosiasinya
- 1 *label* dan 1 *text field* untuk data masukan nilai minimum dukungan
- 1 *label* dan 1 *text field* untuk data masukan nilai minimum keterpercayaan
- 1 *progress bar* untuk menampilkan proses yang dilakukan.
- 1 *button* untuk eksekusi proses pola kaidah asosiasi.

Gambar 4.37 Antarmuka *use case* pencarian pola kaidah asosiasi

(e) Navigasi Menu Informasi

Dialog ini untuk menampilkan *use case* lakukan permintaan informasi.

Gambar 4.38 Antarmuka *use case* lakukan permintaan informasi

BAB V
UJI COBA & ANALISIS

BAB V

UJI COBA & ANALISIS

Pembahasan dalam bab ini mengenai uji coba yang akan dilaksanakan. Pertama kali akan dibahas tentang lingkungan tempat uji coba aplikasi ini. Kemudian penjelasan mengenai beberapa data transaksi yang digunakan dalam uji coba dan diakhiri dengan pelaksanaan uji coba serta analisis hasil uji coba.

5.1 Lingkungan Uji Coba

Untuk mengevaluasi efektivitas dan efisiensi kinerja dari aplikasi, maka dilakukan pengujian dengan menggunakan beberapa skenario yang berbeda. Pengujian ini meliputi pengujian data kecil dan pengujian data besar. Semua eksperimen dilakukan pada sebuah Personal Computer dengan prosesor Intel ® Pentium ® 4 CPU 3.00 GHz, memori 512 MB RAM dan VGA Card Intel(R) 82865G Graphics Controller dengan AGP8X. Sistem operasi yang dijalankan adalah Microsoft Windows 2000 Professional (5.0, Build 2195). Serta menggunakan basis data *Oracle 9i Databases Release 1* (9.0.1.1.1) dan *Win MySQLadmin Ver.1.0 for Win95/Win98/NT*. Sistem aplikasi penggalian data yang akan diuji dengan menerapkan algoritma *BOMO* dan pendekatan *loopback* yang diimplementasikan dengan bahasa pemrograman Java menggunakan kompiler jdk1.5.0.

5.2 Data Uji Coba

Pembuatan data uji coba ini digunakan sebagai data transaksi buatan pada proses pencarian *n* buah *k-itemset* paling menarik. Data uji coba yang dibuat meliputi data kecil untuk uji kebenaran dan data besar untuk uji kinerja aplikasi. Berikut karakteristik data dan data uji coba yang digunakan didalam uji coba kinerja aplikasi

5.2.1 Karakteristik data

Data transaksi buatan yang digunakan menirukan karakteristik transaksi pada lingkungan *retail* dimana orang cenderung membeli beberapa item secara bersamaan pada satu kali transaksi dan banyaknya item terkelompok di sekitar nilai rata-rata [AGR-1994]. Kumpulan item semacam itulah yang disebut sebagai sebuah *large itemset*. Berikut parameter-parameter yang digunakan didalam pembentukan data transaksi buatan

- Jumlah Item (N)

  Banyaknya item yang akan digunakan dalam proses pembuatan data buatan. Misalnya N = 10 maka himpunan item yang digunakan adalah $\{I_1,I_2,I_3,...,I_{10}\}$.

- Rata-rata ukuran transaksi (T)

  Merupakan rata-rata banyaknya item pada setiap transaksi yang terjadi. Misalkan ada 5 transaksi dengan item yang dibeli adalah sebagai berikut.

Tabel 5.1 Contoh data transaksi

TID	Item
1	I_4, I_1, I_3, I_6, I_8
2	I_1, I_2, I_3, I_4, I_6
3	I_2, I_4, I_5, I_7
4	I_2, I_3, I_7, I_8
5	I_1, I_4, I_3, I_5, I_8, I_6

  Maka ukuran rata-rata transaksi adalah (5+5+4+4+6)/5 = 5 (dibulatkan)

- Jumlah Transaksi (D)

  Banyaknya transaksi yang dibuat. Pada contoh tabel 5.1 jumlah transaksi yang dibuat adalah 5.

- Rata-rata ukuran *large itemset* (I)

  Yang dimaksud dengan rata-rata ukuran *large itemset* adalah rata-rata banyaknya item pada *large itemset* yang dibuat.

Tabel 5.2 Rata-rata ukuran *itemset*

Large Itemset	Jumlah Item
1	2
2	3
3	3
4	2
5	4

Pada contoh diatas, rata-rata ukuran large *itemset* adalah (2+3+3+2+4)/5 = 3 (dibulatkan)

- Jumlah maksimal *large itemset* (L)

  Banyaknya item maksimal pada kumpulan *large itemset*. Pada tabel 5.2 diatas, ukuran maksimum *large itemset* adalah 4.

- Rata-rata korelasi antar pola(C)

  Merupakan rata-rata korelasi antar pola pada *large itemset* yang dibuat.

5.2.2 Pembuatan Data Uji Coba

Data transaksi buatan meliputi data kecil dan data besar. Data kecil digunakan untuk uji kebenaran. Bentuk data tersebut sama dengan contoh data transaksi yang diberikan dalam bab 2. Tabel berikut ini memperlihatkan data transaksi dimana data tabel transaksi yang pertama dengan nama "Data 1".

Tabel 5.3 Data 1

TID	Item
1	I_4, I_1, I_3, I_6, I_8
2	I_1, I_2, I_3, I_4, I_6
3	I_2, I_4, I_5, I_7
4	I_2, I_3, I_7, I_8
5	I_1, I_4, I_3, I_5, I_8, I_6

Pada data besar untuk uji kinerja merupakan hasil pembangkitan data dengan menggunakan aplikasi *ARMiner-Synthetic Database Generator* oleh Laurentiu Cristofor [ARM-02]. Konfigurasi dari setiap data transaksi buatan menggunakan karateristik data sesuai yang dijelaskan.

5.3 Skenario uji coba

Pada dasarnya uji coba akan selalu dilakukan terhadap beberapa kondisi yang berbeda. Sehingga didapatkan hasil untuk melakukan analisis kemampuan dari aplikasi. Berikut penjelasan skenario uji coba.

5.3.1 Uji Kebenaran Aplikasi

Pengujian data kecil digunakan untuk melihat apakah proses pencarian *n* buah *k-itemset* paling menarik pada aplikasi telah sesuai yang diinginkan. Pada uji coba ini digunakan tabel "data 1" sebagai data uji coba. Kebenaran dalam uji coba ini dibandingkan dengan hasil yang diperoleh dalam bab 3.

5.3.2 Uji Kinerja Aplikasi

Pengujian data besar dilakukan untuk menguji kehandalan kinerja aplikasi didalam mencari *n* buah *k-itemset* paling menarik. Berikut skenario uji kinerja aplikasi yang dilakukan:

a) Uji kinerja dataset T20I10D100K terhadap perubahan parameter *n* buah paling menarik dan maksimal k-*itemset*.
b) Uji kinerja terhadap perubahan rata-rata transaksi.
c) Uji kinerja terhadap perubahan rata-rata *large itemset*.
d) Uji kinerja terhadap perubahan maksimal *large itemset*.
e) Uji kinerja terhadap perubahan jumlah item.
f) Uji kinerja terhadap perubahan rata-rata korelasi antar pola.
g) Uji kinerja terhadap perubahan jumlah transaksi.

5.4 Pelaksanaan Uji Coba dan Analisis Hasil

Pada sub-bab ini akan dilakukan analisis uji coba berdasarkan skenario yang ditentukan pada sub-bab sebelumnya. Berikut penjelasan uji coba yang dilakukan sesuai skenario yang disusun.

5.4.1 Uji Kebenaran Aplikasi

Pengujian data kecil dilakukan terhadap data transaksi yang berukuran relatif kecil. Pengujian data kecil dilakukan untuk menguji kebenaran aplikasi.

Data transaksi tabel 5.3 digunakan untuk proses uji coba, sehingga diubah dulu ke dalam format standar seperti pada tabel 5.4. Format ini sesuai dengan spesifik kebutuhan sistem terhadap basis data sesuai dalam bab 4 (sub-bab implementasi basis data). Format ini juga untuk memudahkan pencarian *n* buah *k-itemset* paling menarik.

Tabel 5.4 Data transaksi dalam format "id_key #TID #Item"

id_key	TID	Item
1	1	1
2	1	3
3	1	4
4	1	6
5	1	8
6	2	1
7	2	2
8	2	3
9	2	4
10	2	6
11	3	2
12	3	4
13	3	5
14	3	7
15	4	2
16	4	3
17	4	7
18	4	8
19	5	1
20	5	3
21	5	4
22	5	5
23	5	6
24	5	8

Setelah aplikasi dijalankan dengan nilai *n* buah paling menarik dan nilai maksimal k-*itemset* adalah 3 dan proses pencarian menggunakan metode penelusuran dari awal *FP-tree*, penelusuran dari tengah *FP-tree*, penelusuran dari akhir *FP-tree* dan menggunakan pendekatan *loopback* dengan faktor penurunan antara 0.1 sampai 0.9. Maka didapatkan hasil seperti ditunjukkan pada tabel 5.5, sedangkan hasil perhitungan dalam bab 3 ditunjukkan pada tabel 5.6.

Tabel 5.5 Hasil pencarian *3* buah *3-itemsets* paling menarik dengan aplikasi

	Penelusuran *FP-tree*			Pendekatan *Loopback*	ξ
	Dari atas	Dari tengah	Dari bawah	Untuk f = 0.1 - 0.9	
1-itemset	I_4	I_4	I_4	I_4	4
	I_3	I_3	I_3	I_3	4
	I_8	I_8	I_8	I_8	3
2-itemset	$I_3\ I_4$	$I_4\ I_6$	$I_1\ I_4$	$I_3\ I_4$	3
	$I_4\ I_6$	$I_3\ I_6$	$I_1\ I_3$	$I_4\ I_6$	3
	$I_3\ I_8$	$I_3\ I_8$	$I_1\ I_6$	$I_3\ I_8$	3
3-itemset	$I_3\ I_4\ I_6$	$I_3\ I_4\ I_6$	$I_1\ I_3\ I_4$	$I_3\ I_4\ I_6$	3
	$I_1\ I_3\ I_4$	$I_1\ I_3\ I_4$	$I_1\ I_4\ I_6$	$I_1\ I_3\ I_4$	3
	$I_1\ I_4\ I_6$	$I_1\ I_4\ I_6$	$I_1\ I_3\ I_6$	$I_1\ I_4\ I_6$	3

Tabel 5.6 Hasil pencarian *3* buah *3-itemsets* paling menarik

k	k-*itemsets*	Jumlah Dukungan
1	I_3	4
	I_4	4
	I_1	3
2	I_3, I_4	3
	I_3, I_1	3
	I_3, I_6	3
3	I_3, I_4, I_1	3
	I_3, I_4, I_6	3
	I_3, I_1, I_6	3

Seperti terlihat pada tabel 5.5 hasil pencarian *n* buah *k-itemset* paling menarik untuk setiap metode memiliki kemungkinan hasil *itemset* yang tidak sama. Akan tetapi hasil pencarian *n* buah *k-itemset* paling menarik dengan metode penelusuran dari akhir *FP-tree* telah sesuai hasil dalam bab 3 (sub-bab pembangkitan algoritma *BOMO*), dimana proses pencarian *n* buah *k-itemset* pada dalam bab 3 menggunakan metode penelusuran dari akhir *FP-tree*.

Perbedaan hasil *n* buah *k-itemset* paling menarik disebabkan karena urutan awal pencarian keempat metode yang berbeda. Dalam konteks ini aplikasi hanya memperbaharui hasil pencarian jika jumlah dukungan *itemset* lebih besar dari nilai batas ambang dukungan minimumnya. Akan tetapi dapat dipastikan nilai ξ dan ξ_K yang dihasilkan untuk keempat metode adalah sama. Sehingga hasil pencarian

n buah *k-itemset* paling menarik menggunakan aplikasi telah seperti yang diinginkan.

Tahap berikut aplikasi mencari pola kaidah asosiasi dari *n* buah *k-itemset* paling menarik. Pola kaidah asosiasi yang ditampilkan menggunakan nama barang pada tabel 2.1. Pola kaidah asosiasi untuk hasil *n* buah *k-itemset* paling menarik menggunakan metode penelusuran dari akhir *FP-tree* ditunjukkan pada gambar 5.1, sedangkan pola kaidah asosiasi untuk hasil *n* buah *k-itemset* paling menarik menggunakan metode lain ditujukkan pada lampiran D.

```
{AbsorbAid}-->{AcidAmpules}[S=60.0%,C=100.0%]
{AcidAmpules}-->{AbsorbAid}[S=60.0%,C=75.0%]
{AbsorbAid}-->{Acetyl-L-Carnitine}[S=60.0%,C=100.0%]
{Acetyl-L-Carnitine}-->{AbsorbAid}[S=60.0%,C=75.0%]
{AbsorbAid}-->{AcidEyeCreme}[S=60.0%,C=100.0%]
{AcidEyeCreme}-->{AbsorbAid}[S=60.0%,C=100.0%]
{AbsorbAid}-->{Acetyl-L-Carnitine,AcidAmpules}[S=60.0%,C=100.0%]
{AbsorbAid,Acetyl-L-Carnitine}-->{AcidAmpules}[S=60.0%,C=100.0%]
{Acetyl-L-Carnitine}-->{AbsorbAid,AcidAmpules}[S=60.0%,C=75.0%]
{Acetyl-L-Carnitine,AcidAmpules}-->{AbsorbAid}[S=60.0%,C=100.0%]
{AcidAmpules}-->{AbsorbAid,Acetyl-L-Carnitine}[S=60.0%,C=75.0%]
{AbsorbAid,AcidAmpules}-->{Acetyl-L-Carnitine}[S=60.0%,C=100.0%]
{AbsorbAid}-->{AcidAmpules,AcidEyeCreme}[S=60.0%,C=100.0%]
{AbsorbAid,AcidAmpules}-->{AcidEyeCreme}[S=60.0%,C=100.0%]
{AcidAmpules}-->{AbsorbAid,AcidEyeCreme}[S=60.0%,C=75.0%]
{AcidAmpules,AcidEyeCreme}-->{AbsorbAid}[S=60.0%,C=100.0%]
{AcidEyeCreme}-->{AbsorbAid,AcidAmpules}[S=60.0%,C=100.0%]
{AbsorbAid,AcidEyeCreme}-->{AcidAmpules}[S=60.0%,C=100.0%]
{AbsorbAid}-->{Acetyl-L-Carnitine,AcidEyeCreme}[S=60.0%,C=100.0%]
{AbsorbAid,Acetyl-L-Carnitine}-->{AcidEyeCreme}[S=60.0%,C=100.0%]
{Acetyl-L-Carnitine}-->{AbsorbAid,AcidEyeCreme}[S=60.0%,C=75.0%]
{Acetyl-L-Carnitine,AcidEyeCreme}-->{AbsorbAid}[S=60.0%,C=100.0%]
{AcidEyeCreme}-->{AbsorbAid,Acetyl-L-Carnitine}[S=60.0%,C=100.0%]
{AbsorbAid,AcidEyeCreme}-->{Acetyl-L-Carnitine}[S=60.0%,C=100.0%]
```

Gambar 5.1 Pola kaidah asosiasi untuk hasil 3 3-*itemset* paling menarik menggunakan metode penelusuran dari akhir *FP-tree*

Hasil pola kaidah asosiasi setiap metode akan berbeda karena bergantung dari hasil *n* buah *k-itemset* paling menarik. Meskipun begitu pola kaidah asosiasi yang dihasilkan harus memiliki dukungan atau keterpercayaan lebih besar sama dengan 60%, karena nilai batas ambang dukungan minimum adalah 3. Hal ini telah sesuai dengan kaidah asosiasi yang dihasilkan oleh aplikasi dan juga telah

sesuai dengan hasil pola kaidah asosiasi dalam bab 3 (sub-bab hasil pola kaidah asosiasi).

5.4.2 Uji Kinerja Aplikasi

Uji coba ini digunakan untuk mengetahui kinerja aplikasi terhadap perubahan-perubahan data transaksi buatan dari berbagai ukuran dan distribusi data. Analisis dilakukan terhadap waktu komputasi proses yang dperlukan untuk mencari *n* buah *k-itemset* paling menarik dan penggunaan memori untuk menyimpan *FP-tree*. Hasil uji kinerja aplikasi menggunakan data transaksi buatan yang lain ditunjukkan pada lampiran E. Berikut pelaksanaan uji kinerja aplikasi:

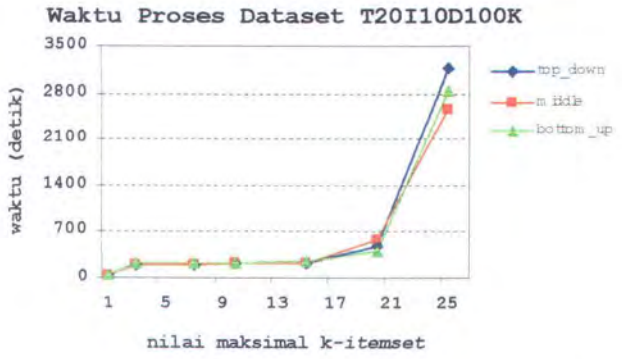
(a) Uji kinerja dataset T20I10D100K terhadap perubahan parameter *n* paling menarik dan maksimal k-*itemset*

Analisis terhadap perubahan parameter *n* buah paling menarik dan maksimal k-*itemset* dilakukan untuk mengetahui kinerja aplikasi terhadap proses pencarian *n* buah *k-itemset* paling menarik yang bervariasi. Uji coba akan menggunakan parameter nilai maksimal k-*itemset* sama dengan 1, 3 , 7, 10, 15, 20 dan 25 untuk nilai 10 buah paling menarik dan parameter nilai *n* paling menarik sama dengan 5, 10, 15, 25, 50, 100, 150 dan 200 untuk 10-*itemset* terhadap dataset T20I10D100K. Dataset T20I10D100K terdiri dari 100.000 transaksi dengan 1000 item yang berbeda. Rata-rata jumlah item dalam sebuah transaksi adalah 20 dan rata-rata jumlah panjang *large itemset* adalah 10 dengan rata-rata korelasi antar pola 0.25. Sedangkan jumlah record dataset ini adalah 1.198.073 atau 1,1 juta baris. Dalam uji kinerja ini menggunakan dua metode pencarian *n* k-*itemset* paling menarik yaitu menggunakan urutan pembangkitan *conditional FP-tree* (untuk algoritma *BOMO*) dan pendekatan *loopback*. Berikut pelaksanaan uji kinerja aplikasi terhadap dataset T20I10D100K.

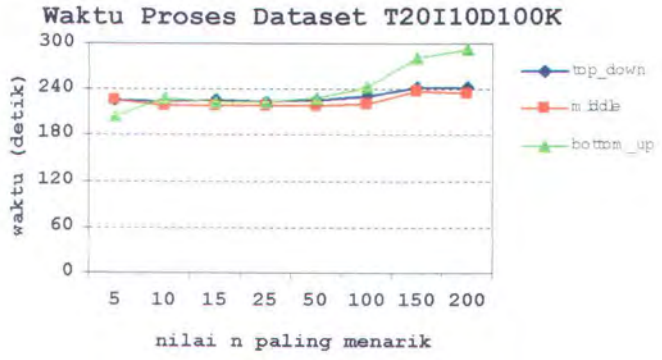
- **Urutan Pembangkitan *conditional FP-tree***

Uji coba ini mencari *n* buah k-*itemset* paling menarik yang bervariasi menggunakan metode penelusuran dari awal *FP-tree*, penelusuran dari tengah *FP-tree*, penelusuran dari akhir *FP-tree* terhadap dataset T20I10D100K. Waktu komputasi proses pencarian 10 buah k-*itemset* paling menarik ditunjukkan gambar

5.2. Sedangkan waktu komputasi proses pencarian *n* buah 10-*itemset* paling menarik ditunjukkan pada gambar 5.3.



Gambar 5.2 Waktu komputasi proses 10 buah paling menarik pada T20I10D100K dengan urutan pembangkitan *conditional FP-tree*



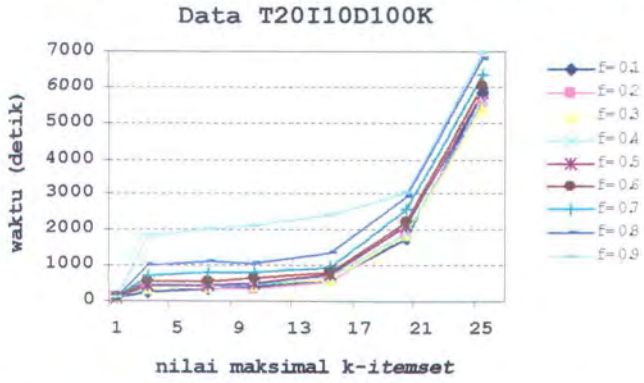
Gambar 5.3 Waktu komputasi proses 10-*itemset* pada T20I10D100K dengan urutan pembangkitan *conditional FP-tree*

Berdasarkan gambar 5.2, peningkatan maksimal k-*itemset* akan meningkatkan waktu komputasi proses. Hal ini berbeda dengan waktu komputasi proses untuk peningkatan *n* paling menarik yang cenderung tetap. Berdasarkan gambar 5.2 dan 5.3, waktu komputasi proses pencarian *n* k-*itemset* menggunakan metode penelusuran dari tengah *FP-tree* lebih cepat dibandingkan penelusuran dari awal *FP-tree* maupun penelusuran dari akhir *FP-tree*.

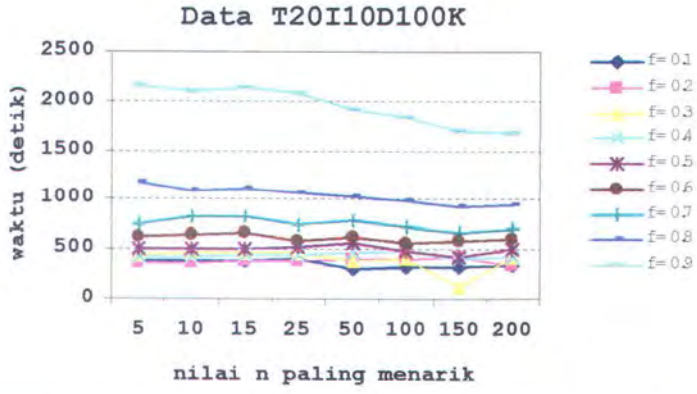
- **Pendekatan *loopback***

Uji coba ini akan menggunakan faktor penurunan dengan nilai 0.1, 0.2, ..., 0.9 terhadap dataset T20I10D100K. Waktu komputasi proses pencarian 10 k-

itemset paling menarik ditunjukkan gambar 5.4. Sedangkan waktu komputasi proses pencarian *n* 10-*itemset* paling menarik ditunjukkan pada gambar 5.5.



Gambar 5.4 Waktu komputasi proses 10 buah paling menarik pada T20I10D100K dengan pendekatan *loopback*



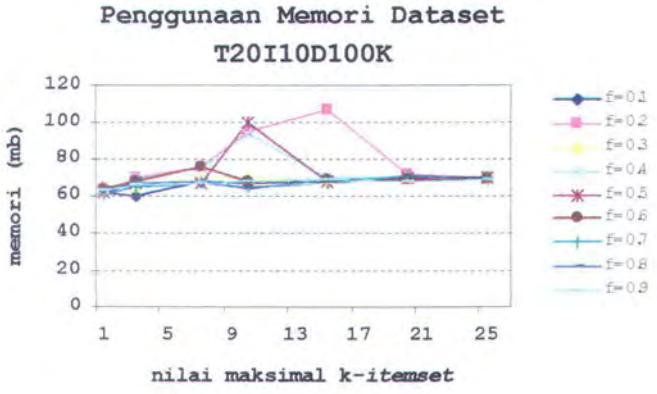
Gambar 5.5 Waktu komputasi proses 10-*itemset* pada T20I10D100K dengan pendekatan *loopback*

Hasil uji kinerja ini menunjukkan peningkatan maksimal k-*itemset* akan meningkatkan waktu komputasi proses yang diperlukan. Hal ini berbeda dengan waktu komputasi proses yang diperlukan untuk peningkatan *n* buah paling menarik yang cenderung sama. Berdasarkan gambar 5.4 dan gambar 5.5, semakin besar faktor penurunan pada pendekatan *loopback* maka semakin lama waktu komputasi proses yang diperlukan.

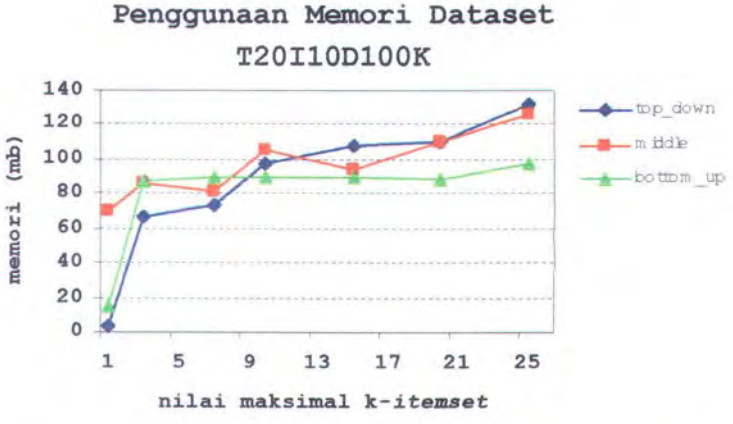
Perbandingan uji kinerja dataset T20I10D100K terhadap perubahan parameter *n* buah paling menarik dan maksimal k-*itemset* menunjukkan urutan pembangkitan *conditional FP-tree* memiliki waktu komputasi proses yang lebih cepat daripada pendekatan *loopback*. Akan tetapi pendekatan *loopback*

MILIK PERPUSTAKAAN ITS

menggunakan memori lebih kecil untuk menyimpan *FP-tree*. Hal ini disebabkan karena pendekatan *loopback* membangkitkan *FP-tree* berulang kali sehingga waktu komputasi proses lebih lama, akan tetapi *FP-tree* yang dibangkitkan untuk beberapa item saja sehingga membutuhkan memori yang lebih kecil. Memori yang dibutuhkan untuk menyimpan *FP-tree* pada pendekatan *loopback* ditunjukkan gambar 5.6, sedangkan memori yang dibutuhkan untuk menyimpan *FP-tree* pada urutan pembangkitan *conditional FP-tree* ditunjukkan gambar 5.7.



Gambar 5.6 Memori yang digunakan pendekatan *loopback* untuk menyimpan *FP-tree* dataset T20I10D100K



Gambar 5.7 Memori yang digunakan urutan pembangkitan *conditional FP-tree* untuk menyimpan *FP-tree* dataset T20I10D100K

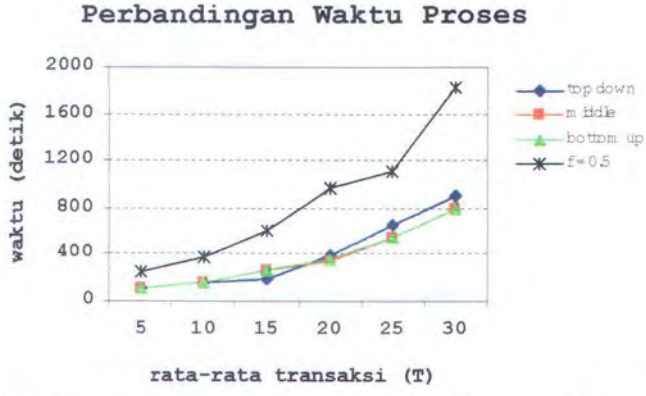
(b) Uji kinerja terhadap perubahan rata-rata transaksi

Analisis terhadap perubahan rata-rata transaksi (T) dilakukan dengan membuat beberapa dataset dengan rata-rata transaksi yang berbeda-beda.

Kemudian dilakukan pencarian 100 buah 10-*itemset* paling menarik terhadap dataset tersebut. Hasil analisis terhadap dataset dengan rata-rata transaksi yang berbeda-beda ditunjukkan pada tabel 5.7. Sedangkan grafik waktu komputasi proses yang dibutuhkan untuk setiap dataset dengan nilai rata-rata transaksi yang berbeda–beda ditunjukkan pada gambar 5.8.

Tabel 5.7 Hasil analisis terhadap berbagai nilai T

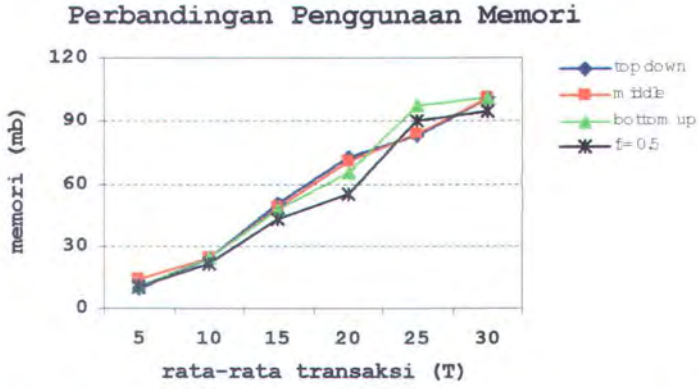
Txx.I10.D100K.L2K.M1K.C0,25							
No	T	Jumlah Record	Waktu Komputasional Proses (detik)				ξ
			Penelusuran *FP-tree*			*Loopback* (f = 0.5)	
			Dari atas	Dari tengah	Dari bawah		
1	5	949.330	104	102	102	246	284
2	10	1.198.073	152	152	152	380	336
3	15	1.558.199	189	267	269	616	381
4	20	2.003.622	385	341	352	964	336
5	25	2.483.930	651	548	551	1113	1033
6	30	2.977.354	909	804	791	1830	595



Gambar 5.8 Waktu proses untuk berbagai nilai T

Gambar 5.8 menunjukkan pencarian *n* buah k-*itemset* paling menarik menggunakan metode penelusuran dari tengah *FP-tree* lebih cepat dibandingkan penelusuran dari awal *FP-tree*, penelusuran dari akhir *FP-tree* atau pendekatan *loopback*. Waktu komputasi proses yang diperlukan pendekatan *loopback* sangat besar karena melakukan pembangkitan *FP-tree* berulang kali. Sedangkan perbandingan penggunaan memori untuk menyimpan *FP-tree* setiap dataset tersebut ditunjukkan gambar 5.9. Berdasarkan gambar 5.9 maka penggunaan

memori untuk menyimpan *FP-tree* menggunakan pendekatan *loopback* lebih kecil dibandingkan metode penelusuran dari awal *FP-tree*, penelusuran dari tengah *FP-tree*, penelusuran dari akhir *FP-tree* karena pendekatan *loopback* membagkitkan *FP-tree* untuk beberapa item tertentu.



Gambar 5.9 Penggunaan memori untuk berbagai nilai T

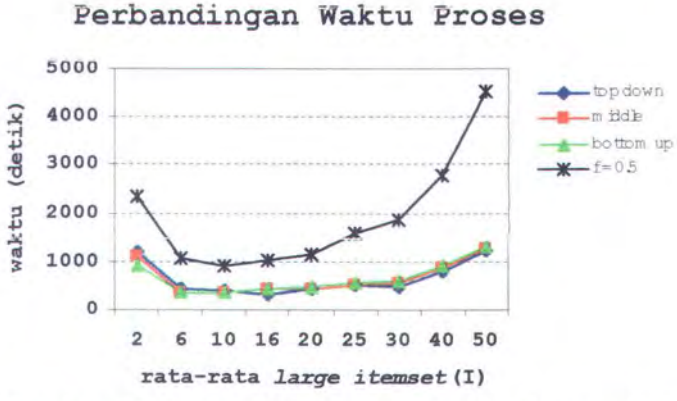
Hasil uji kinerja perubahan rata-rata transaksi (T) menunjukkan semakin besar rata-rata transaksi, maka semakin besar pula waktu komputasi prosesyang diperlukan dan semakin besar memori yang digunakan untuk menyimpan *FP-tree*. Hal ini disebabkan karena *FP-tree* yang dibangkitkan semakin besar (berdasarkan gambar 5.9) seiring meningkatnya data transaksi (tabel 5.7 kolom jumlah record), sehingga waktu komputasi proses pencarian *n* k-*itemset* paling menarik akan semakin besar.

(c) Uji kinerja terhadap perubahan rata-rata *large itemset*

Analisis terhadap perubahan rata-rata *large itemset* (I) dilakukan dengan membuat beberapa dataset dengan rata-rata *large itemset* yang berbeda-beda. Kemudian dilakukan pencarian 100 buah 10-*itemset* paling menarik terhadap dataset tersebut. Hasil analisis terhadap dataset dengan rata-rata *large itemset* yang berbeda-beda ditunjukkan pada tabel 5.8. Sedangkan grafik waktu komputasi proses yang dibutuhkan untuk setiap dataset ditunjukkan gambar 5.10.

Tabel 5.8 Hasil analisis terhadap berbagai nilai I

T20.Ixx.D100K.L2K.M1K.C0,25							
No	I	Jumlah Record	Waktu Komputasional Proses (detik)				ξ
			Penelusuran *FP-tree*			*Loopback* (f = 0.5)	
			Dari atas	Dari tengah	Dari bawah		
1	2	1.993.505	1205	1104	913	2358	6
2	6	1.992.491	451	364	360	1063	463
3	10	2.003.622	385	341	352	920	398
4	16	2.177.939	316	426	434	1035	402
5	20	2.403.566	436	428	461	1151	366
6	25	2.687.666	521	528	540	1584	340
7	30	3.036.410	491	563	613	1859	429
8	40	3.916.708	774	889	918	2761	443
9	50	4.882.510	1245	1255	1294	4527	414



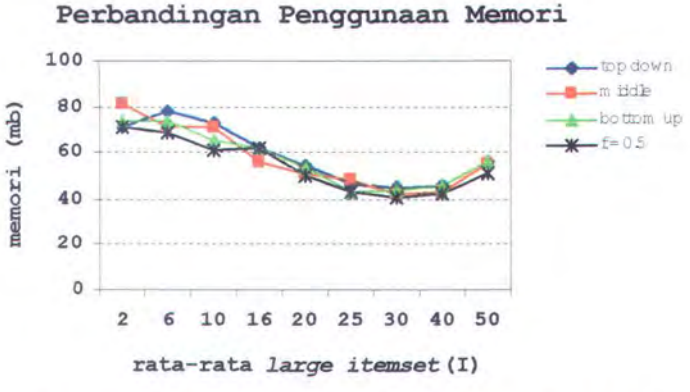
Gambar 5.10 Waktu proses untuk berbagai nilai I

Berdasarkan gambar 5.10, maka peningkatan rata-rata *large itemset* akan menurunkan waktu komputasi proses sampai nilai rata-rata *large itemset* tertentu, pada kondisi ini penggunaan metode penelusuran dari bawah *FP-tree* lebih cepat dibandingkan ketiga metode lainnya. Hal ini disebabkan *large itemset* yang lebih besar (dalam hal ini 10-*itemset*) dari rata-rata *large itemset* tersebar pada simpul-simpul terbawah *FP-tree*, sehingga waktu komputasi proses metode penelusuran dari bawah *FP-tree* lebih cepat.

Kemudian waktu komputasi proses akan meningkat seiring dengan peningkatan rata-rata *large itemset*, dalam kondisi ini waktu komputasi proses pencarian *n* buah *k-itemset* paling menarik metode penelusuran dari atas *FP-tree*

lebih cepat. Pada saat rata-rata *large itemset* semakin besar maka *FP-tree* yang dibangkitkan semakin kecil sampai nilai rata-rata *large itemset* tertentu, akan tetapi waktu komputasi proses pencarian *n* buah k-*itemset* cenderung semakin besar. Hal ini disebabkan adanya peningkatan jumlah record pada basis data transaksi sehingga waktu yang dibutuhkan untuk membangkitkan *FP-tree* semakin besar.

Sedangkan memori yang digunakan untuk menyimpan *FP-tree* akan semakin kecil sampai nilai rata-rata *large itemset* tertentu, kemudian akan semakin besar seiring peningkatan rata-rata *large itemset*. Hal ini disebabkan karena *FP-tree* menggunakan struktur item *prefix-subtree*, sehingga semakin besar data transaksi (tabel 5.7 kolom jumlah *record*) tidak selalu menyebabkan ukuran *FP-tree* semakin besar (jika tidak ada penambahan simpul baru). Dalam uji kinerja ini penggunaan memori untuk menyimpan *FP-tree* menggunakan pendekatan *loopback* lebih kecil dibandingkan ketiga metode lainnya. Perbandingan penggunaan memori untuk menyimpan *FP-tree* setiap dataset tersebut ditunjukkan pada gambar 5.11.



Gambar 5.11 Penggunaan memori untuk berbagai nilai I

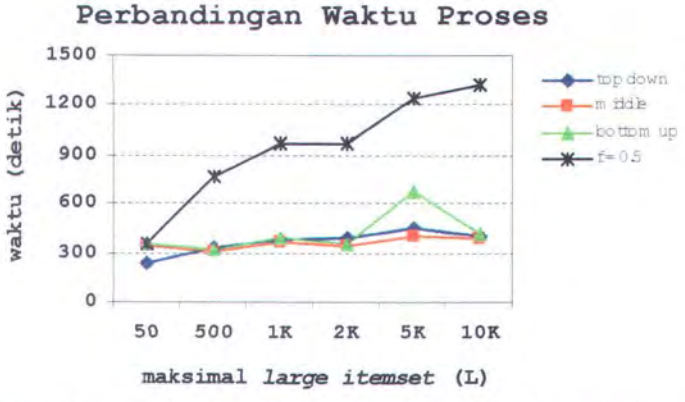
Hasil uji kinerja perubahan rata-rata *large itemset* (I) menunjukkan semakin besar rata-rata *large itemset*, maka semakin kecil waktu komputasi proses sampai nilai rata-rata *large itemset* tertentu, kemudian waktu komputasi proses akan meningkat, begitu juga penggunaan memori untuk menyimpan *FP-tree* yang dibangkitkan. Hal ini disebabkan karena rata-rata *large itemset* sangat berpengaruh terhadap *FP-tree* yang dibangkitkan.

(d) Uji kinerja terhadap perubahan maksimal *large itemset*

Analisis terhadap perubahan maksimal *large itemset* (L) dilakukan dengan membuat beberapa dataset dengan maksimal *large itemset* yang berbeda-beda. Kemudian dilakukan pencarian 100 buah 10-*itemset* paling menarik terhadap dataset tersebut. Hasil analisis terhadap dataset dengan maksimal *large itemset* yang berbeda-beda ditunjukkan pada tabel 5.9. Sedangkan grafik waktu komputasi proses yang dibutuhkan untuk setiap dataset ditunjukkan gambar 5.12.

Tabel 5.9 Hasil analisis terhadap berbagai nilai L

T20.I10.D100K.Lxx.M1K.C0,25							
No	L	Jumlah Record	Waktu Komputasional Proses (detik)				ξ
			Penelusuran *FP-tree*			*Loopback* (f = 0,5)	
			Dari atas	Dari tengah	Dari bawah		
1	50	1.976.089	231	337	355	349	9414
2	500	1.998.757	328	311	322	762	1338
3	1K	2.004.533	383	368	387	966	589
4	2K	2.003.622	385	341	352	964	398
5	5K	2.003.209	449	397	673	1238	202
6	10K	2.004.708	402	394	414	1326	119

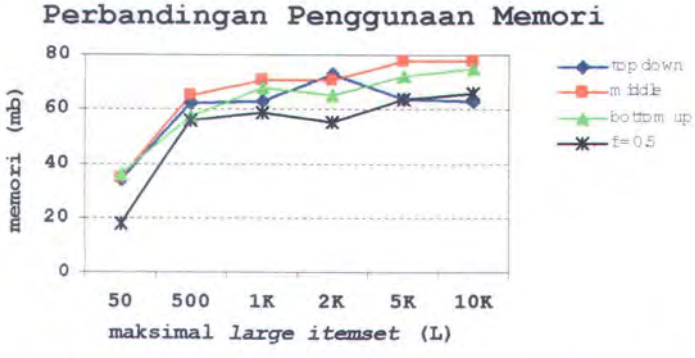


Gambar 5.12 Waktu proses untuk berbagai nilai L

Berdasarkan gambar 5.12, maka pencarian *n* k-*itemset* paling menarik menggunakan metode penelusuran dari tengah *FP-tree* lebih cepat dibandingkan ketiga metode lainnya. Sedangkan waktu komputasi proses yang diperlukan metode penelusuran dari atas *FP-tree*, metode penelusuran dari tengah *FP-tree* dan metode penelusuran dari bawah *FP-tree* cenderung sama meskipun semakin

besar maksimal *large itemset* (L). Akan tetapi semakin besar maksimal *large itemset*, maka semakin besar pula waktu komputasi proses yang diperlukan pendekatan *loopback*. Hal ini disebabkan karena semakin besar maksimal *large itemset*, maka semakin kecil nilai ambang batas dukungan minimum. Sehingga proses interasi pembangkitan *FP-tree* yang dilakukan semakin banyak, meskipun jumlah dukungan setiap item semakin kecil. Dalam konteks ini semakin kecil jumlah dukungan item, maka semakin kecil pula iterasi yang dilakukan.

Sedangkan perbandingan penggunaan memori untuk menyimpan *FP-tree* setiap dataset tersebut ditunjukkan gambar 5.13. Berdasarkan gambar 5.13 maka penggunaan memori untuk menyimpan *FP-tree* menggunakan pendekatan *loopback* lebih kecil dibandingkan ketiga metode lainnya, karena *loopback* membangkitkan *FP-tree* untuk beberapa item tertentu. Semakin besar maksimal *large itemset*, maka semakin besar pula penggunaan memori untuk menyimpan *FP-tree*. Hal ini disebabkan karena semakin besar himpunan item pada transaksi, sehingga akan terbentuk simpul-simpul baru pada *FP-tree* yang kemudian menyebabkan ukuran *FP-tree* lebih besar.



Gambar 5.13 Penggunaan memori untuk berbagai nilai L

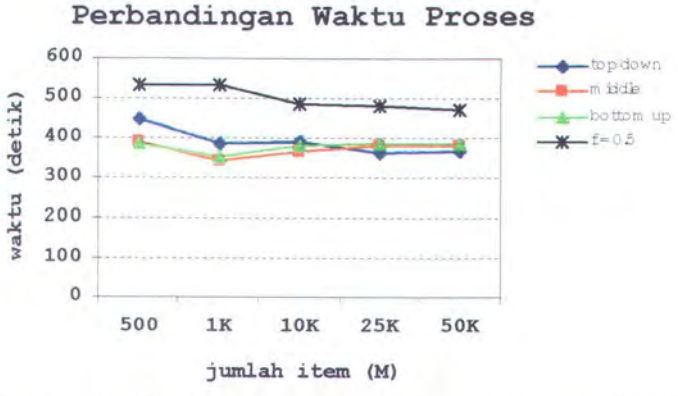
Hasil uji kinerja perubahan maksimal *large itemset* (L) menunjukkan semakin besar maksimal *large itemset*, maka waktu komputasi proses yang diperlukan untuk metode penelusuran dari awal *FP-tree*, penelusuran dari tengah *FP-tree*, penelusuran dari akhir *FP-tree* cenderung sama akan tetapi waktu komputasi proses untuk pendekatan *loopback* semakin besar, sedangkan memori yang dibutuhkan untuk menyimpan *FP-tree* semakin besar.

(e) Uji kinerja terhadap perubahan jumlah item

Analisis terhadap perubahan jumlah item (M) dilakukan dengan membuat beberapa dataset dengan jumlah item yang berbeda-beda. Kemudian dilakukan pencarian 100 buah 10-*itemset* paling menarik terhadap dataset tersebut. Hasil analisis terhadap dataset dengan jumlah item yang berbeda-beda ditunjukkan pada tabel 5.10. Sedangkan grafik waktu komputasi proses yang dibutuhkan untuk setiap dataset ditunjukkan pada gambar 5.14.

Tabel 5.10 Hasil analisis terhadap berbagai nilai M

T20.I10.D100K.L2K.Mxx.C0,25							
No	M	Jumlah Record	Waktu Komputasional Proses (detik)				ξ
			Penelusuran *FP-tree*			*Loopback* (f = 0,5)	
			Dari atas	Dari tengah	Dari bawah		
1	500	1.993.823	450	390	385	532	491
2	1K	2.003.622	385	341	352	532	336
3	10K	2.010.087	390	369	383	484	531
4	25K	2.014.342	362	383	385	482	430
5	50K	2.015.379	367	382	385	471	443

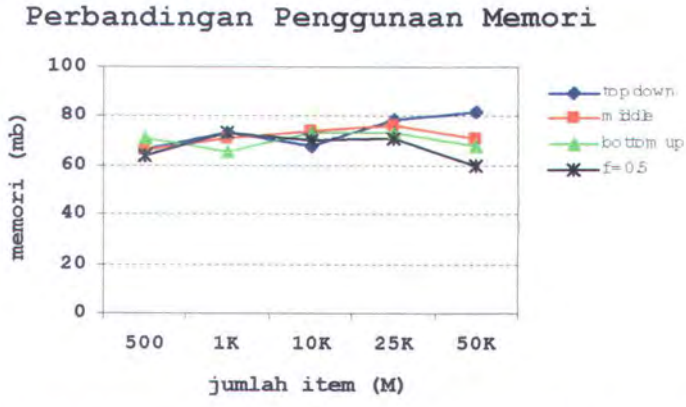


Gambar 5.14 Waktu proses untuk berbagai nilai M

Berdasarkan gambar 5.14, meskipun jumlah item semakin besar, tetapi waktu komputasi proses cenderung sama untuk metode penelusuran dari awal *FP-tree*, penelusuran dari tengah *FP-tree*, penelusuran dari akhir *FP-tree*. Akan tetapi semakin besar jumlah item, maka semakin kecil waktu komputasi proses pendekatan *loopback*. Semakin kecil waktu komputasi proses pendekatan

loopback, disebabkan karena semakin banyak variasi item pada data transaksi. Sehingga jumlah dukungan setiap item akan semakin kecil yang pada akhirnya akan mengurangi proses interasi pembangkitan *FP-tree* pada pendekatan *loopback*. Dalam konteks ini interasi dilakukan jika ada jumlah dukungan yang lebih besar dari nilai batas ambang dukungan minimum.

Sedangkan perbandingan memori yang dibutuhkan untuk menyimpan *FP-tree* ditunjukkan pada gambar 5.15. Berdasarkan gambar 5.15, meskipun jumlah item semakin besar, tetapi penggunaan memori cenderung sama untuk metode penelusuran dari awal *FP-tree*, penelusuran dari tengah *FP-tree*, penelusuran dari akhir *FP-tree*. Akan tetapi semakin besar jumlah item, maka semakin kecil penggunaan memori pendekatan *loopback*, hal ini seiring proses interasi pembangkitan *FP-tree* pada pendekatan *loopback* semakin sedikit sehingga *FP-tree* yang dibangkitkan semakin kecil.



Gambar 5.15 Penggunaan memori untuk berbagai nilai M

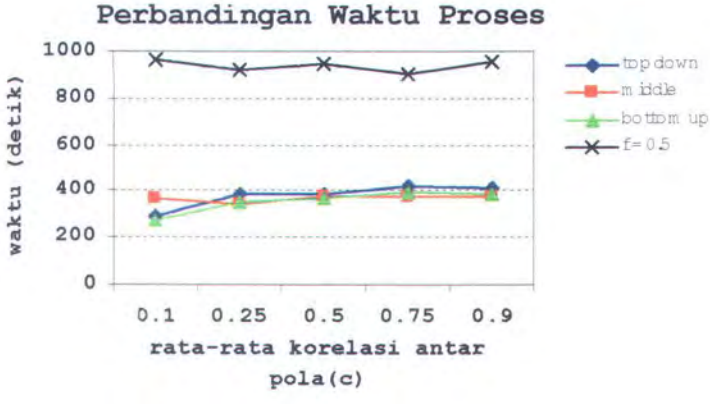
Hasil uji kinerja perubahan jumlah item (M) menunjukkan semakin besar jumlah item, maka waktu komputasi proses untuk metode penelusuran dari awal *FP-tree*, penelusuran dari tengah *FP-tree*, penelusuran dari akhir *FP-tree* cenderung sama sedangkan untuk pendekatan *loopback* semakin kecil, demikian juga memori yang dibutuhkan untuk menyimpan *FP-tree*. Waktu komputasi proses menggunakan penelusuran dari tengah *FP-tree* lebih cepat untuk M=500, M=1K dan M=10K, sedangkan untuk M=25K dan M=50K menunjukkan waktu komputasi proses menggunakan penelusuran dari awal *FP-tree* lebih cepat.

(f) Uji kinerja terhadap perubahan rata-rata korelasi antar pola

Analisis terhadap perubahan rata-rata korelasi antar pola (C) dilakukan dengan membuat beberapa dataset dengan rata-rata korelasi antar pola yang berbeda-beda. Kemudian dilakukan pencarian 100 buah 10-*itemset* paling menarik terhadap dataset tersebut. Hasil analisis terhadap dataset dengan rata-rata korelasi antar pola yang berbeda-beda ditunjukkan tabel 5.11. Sedangkan grafik waktu komputasi proses yang dibutuhkan untuk setiap dataset ditunjukkan gambar 5.16.

Tabel 5.11 Hasil Analisis terhadap berbagai nilai C

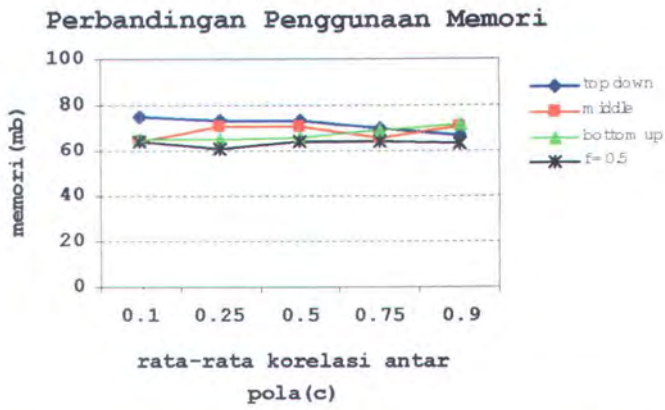
T20.I10.D100K.L2K.M1K.Cxx							
No	C	Jumlah Record	Waktu Komputasional Proses (detik)				ξ
			Penelusuran *FP-tree*			*Loopback* (f = 0,5)	
			Dari atas	Dari tengah	Dari bawah		
1	0.1	2.004.507	291	369	270	962	477
2	0.25	2.003.622	385	341	352	920	336
3	0.5	2.002.282	388	374	372	944	400
4	0.75	1.999.405	417	379	397	904	395
5	0.9	2.003.739	410	380	385	952	403



Gambar 5.16 Waktu proses untuk berbagai nilai C

Gambar 5.16 menunjukkan waktu komputasi proses pencarian *n* k-*itemset* paling menarik menggunakan metode penelusuran dari tengah *FP-tree* lebih cepat dibandingkan penelusuran dari awal *FP-tree*, penelusuran dari akhir *FP-tree* atau pendekatan *loopback*. Waktu komputasi proses pendekatan *loopback* yang sangat besar karena melakukan pembangkitan *FP-tree* berulang kali. Sedangkan

perbandingan penggunaan memori untuk menyimpan *FP-tree* setiap dataset tersebut ditunjukkan gambar 5.17. berdasarkan gambar 5.17 maka penggunaan memori untuk menyimpan *FP-tree* menggunakan pendekatan *loopback* lebih kecil dibandingkan metode penelusuran dari awal *FP-tree*, penelusuran dari tengah *FP-tree*, penelusuran dari akhir *FP-tree*, karena *loopback* membangkitkan *FP-tree* untuk beberapa item tertentu



Gambar 5.17 Penggunaan memori untuk berbagai nilai C

Hasil uji kinerja perubahan rata-rata korelasi antar pola (C) menunjukkan meskipun rata-rata korelasi antar pola yang berbeda-beda, tetapi waktu komputasi proses cenderung sama, demikian juga memori yang dibutuhkan untuk menyimpan *FP-tree*. Hal ini disebabkan karena *FP-tree* yang dibangkitkan cenderung sama (gambar 5.17 dan tabel 5.11 kolom jumlah *record*).

(g) Uji kinerja terhadap perubahan jumlah transaksi

Analisis terhadap perubahan jumlah transaksi (D) dilakukan dengan membuat beberapa dataset dengan jumlah transaksi yang berbeda-beda. Kemudian dilakukan pencarian 100 buah 10-*itemset* paling menarik terhadap dataset tersebut. Hasil analisis terhadap dataset dengan jumlah transaksi yang berbeda-beda ditunjukkan pada tabel 5.12. Sedangkan grafik waktu komputasi proses yang dibutuhkan untuk setiap dataset ditunjukkan pada gambar 5.18.

Tabel 5.12 Hasil Analisis terhadap berbagai nilai D

T20.Ixx.D100K.L2K.M1K.C0,25							
No	D	Jumlah Record	Waktu Komputasi Proses (detik)				ξ
			Penelusuran *FP-tree*			*Loopback* (f = 0,5)	
			Dari atas	Dari tengah	Dari bawah		
1	10K	200.566	13	26	25	46	64
2	50K	1.003.116	93	72	76	227	217
3	100K	2.001.909	231	221	229	1053	668
4	200K	4.001.227	660	542	522	1298	1309
5	500K	9.999.907	4175	2623	3949	6450	2041

Gambar 5.18 Waktu proses untuk berbagai nilai D

Gambar 5.8 menunjukkan waktu komputasi proses pencarian *n* k-*itemset* paling menarik menggunakan metode penelusuran dari tengah *FP-tree* lebih cepat dibandingkan penelusuran dari awal *FP-tree*, penelusuran dari akhir *FP-tree* atau pendekatan *loopback*. Waktu komputasi proses pendekatan *loopback* yang sangat besar karena melakukan pembangkitan *FP-tree* berulang kali. Sedangkan perbandingan penggunaan memori untuk menyimpan *FP-tree* setiap dataset tersebut ditunjukkan gambar 5.9. berdasarkan gambar 5.9 maka penggunaan memori untuk menyimpan *FP-tree* menggunakan pendekatan *loopback* lebih kecil dibandingkan metode penelusuran dari awal *FP-tree*, penelusuran dari tengah *FP-tree*, penelusuran dari akhir *FP-tree*, karena *loopback* membagkitkan *FP-tree* untuk beberapa item tertentu.

Gambar 5.19 Penggunaan memori untuk berbagai nilai D

Hasil uji kinerja perubahan jumlah transaksi (D) menunjukkan semakin besar jumlah transaksi, maka semakin besar pula waktu komputasi proses dan semakin besar memori yang digunakan untuk menyimpan *FP-tree*. Hal ini disebabkan karena *FP-tree* yang dibangkitkan semakin besar (berdasarkan gambar 5.19) seiring meningkatnya data transaksi (tabel 5.12 kolom jumlah *record*), sehingga waktu komputasi proses pencarian *n* buah k-*itemset* paling menarik akan semakin besar.

BAB VI
SIMPULAN

BAB VI

SIMPULAN

Dari Tugas Akhir ini, dapat disimpulkan hal-hal seperti berikut :

a) Tugas akhir ini telah berhasil mengimplementasikan algoritma *BOMO* dan *loopback*, untuk memperoleh *n* buah *k-itemset* paling menarik dari sebuah data market basket tanpa menentukan nilai batas ambang dukungan minimum.

b) *N* buah *k-itemset* paling menarik yang dihasilkan dengan metode penelusuran dari awal *FP-tree*, penelusuran dari tengah *FP-tree*, penelusuran dari akhir *FP-tree* dan pendekatan *loopback* meskipun berbeda, tetapi memiliki batas ambang dukungan minimum yang sama. Dari hasil uji coba dapat disimpulkan bahwa:

   - Pendekatan *loopback* membutuhkan memori lebih kecil untuk menyimpan *FP-tree* dibandingkan dengan metode penelusuran dari awal *FP-tree*, penelusuran dari tengah *FP-tree*, penelusuran dari akhir *FP-tree*. Akan tetapi, membutuhkan waktu komputasi proses lebih lama.
   - Pencarian *n* buah *k-itemset* paling menarik menggunakan metode penelusuran dari tengah *FP-tree* memiliki waktu komputasi proses lebih cepat dibandingkan dengan metode penelusuran dari awal *FP-tree*, penelusuran dari akhir *FP-tree* atau pendekatan *loopback*.

c) Berdasarkan hasil uji coba untuk pengaruh parameter terhadap waktu komputasi proses dan penggunaan memori dapat disimpulkan bahwa:

   - Semakin besar nilai maksimal *k-itemset*, maka semakin besar pula waktu komputasi proses yang diperlukan dan pada nilai *n* buah paling menarik yang berbeda-beda, waktu komputasi proses yang diperlukan cenderung sama. Sedangkan memori yang digunakan untuk menyimpan *FP-tree* cenderung sama.
   - Semakin besar rata-rata transaksi, maka semakin besar pula waktu komputasi proses yang diperlukan dan semakin besar pula memori yang digunakan untuk menyimpan *FP-tree*.

- Peningkatan rata-rata *large itemset* akan menurunkan waktu komputasi proses yang diperlukan sampai pada nilai rata-rata *large itemset* tertentu, kemudian waktu komputasi proses yang diperlukan akan semakin meningkat. Kecenderungan yang sama juga terjadi pada kebutuhan memori yang digunakan untuk menyimpan *FP-tree.*
- Semakin besar maksimal *large itemset*, maka waktu komputasi proses yang diperlukan untuk metode penelusuran dari awal *FP-tree*, penelusuran dari tengah *FP-tree*, penelusuran dari akhir *FP-tree* cenderung sama, akan tetapi waktu komputasi proses yang diperlukan untuk pendekatan *loopback* akan semakin besar. Sedangkan memori yang dibutuhkan untuk menyimpan *FP-tree* akan semakin besar.
- Semakin besar jumlah item, maka waktu komputasi proses yang diperlukan untuk metode penelusuran dari awal *FP-tree*, penelusuran dari tengah *FP-tree*, penelusuran dari akhir *FP-tree* cenderung sama sedangkan untuk pendekatan *loopback* semakin kecil. Kecenderungan yang sama juga terjadi pada kebutuhan memori yang digunakan untuk menyimpan *FP-tree*
- Pada rata-rata korelasi antar pola yang berbeda-beda, waktu komputasi proses yang diperlukan cenderung sama, demikian juga memori yang dibutuhkan untuk menyimpan *FP-tree.*
- Semakin besar jumlah transaksi, maka semakin besar waktu komputasi proses yang diperlukan dan semakin besar memori yang digunakan untuk menyimpan *FP-tree.*

DAFTAR PUSTAKA

DAFTAR PUSTAKA

[YLC-04] Yin-Ling Cheung and Ada Wai-Chee Fu, "*Mining Frequent Itemsets without Support Threshold: With and without Item Constraints*", IEEE Transactions on Knowledge and Data Engineering, Vol.16 No.9, September 2004.

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

[AGS-93] Agrawal, R., Imielinski, T. and Swami, A., "*Mining Association Rules Between Sets of Items in Large Databases*", Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD'93), 1993, Washington, DC, pp. 207-216.

[AGS-94] Agrawal, R. and Srikant, R., "*Fast Algorithms for Mining Association Rules*", Proc. 1994 Int'l Conf.Very Large Data Bases (VLDB'94), 1994, Santiago, Chile, pp. 487-499.

[CHE-96] Ming-Syang Chen, J. Han and Philip S Yu, "*Data Mining: An Overview from a Database Perspective*", IEEE Transactions on Knowledge and Data Engineering, Vol.8 No.6, December 1996.

[LDB-04] Jeffery L. Whitten, Lonnie D. Bentley, Kevin C. Dittman, "*Metoda Desain dan Analisis Sistem*", Edisi ke-6, Penerbit Andi, Saduran Mcgrawhill Education, 2004.

[ARM-02] *ARMinerSynthetic Database Generator written by* Laurentiu Cristofor (2002), [Online] Available from:
<http://www.cs.umb.edu/~laur/ARtool/ARtool_full_1_1_2.zip> [2002]

[SUN-99] Sun Microsystems, Inc (1999), *MemoryMonitor.java*, [Online], Available from: <www.infres.enst.fr/~elc/demos-java/Demo-Java2D/src/MemoryMonitor.java>

[ADH-02] Adhita Pratiwi, "*Perancangan dan Pembuatan Aplikasi Pembangkitan Data yang Mengikuti Pola Distribusi Poisson untuk Pencarian Kaidah Asosiasi dan Pola Sekuensial*", Tugas Akhir, Jurusan Teknik Informatika, FTIF, ITS, 2002.

LAMPIRAN A
DESKRIPSI PERANCANGAN DAN PEMBUATAN APLIKASI

Tabel Deskripsi Perancangan dan Pembuatan Aplikasi

Kebutuhan Pengguna	Kebutuhan Sistem	Diagram *Use Case*	Diagram Sekuensi
Pengguna mampu memperoleh nama tabel untuk penggalian data.	(1) Sistem menyediakan masukkan untuk koneksi basis data (2) Sistem mampu mencari nama tabel untuk penggalian data (3) Sistem mampu menampilkan nama tabel	Lakukan uji koneksi	Lakukan uji koneksi
Pengguna mampu memperoleh *n* buah *k-itemset* paling menarik dari data transaksi.	(1) Sistem menyediakan masukkan untuk penggalian data (dalam hal ini *n* buah *k-itemset* paling menarik) (2) Sistem mampu mencari *n* buah *k-itemset* paling menarik (3) Sistem mampu menampilkan *n* buah *k-itemset* paling menarik	Lakukan pencarian *n* buah *k-itemset* paling menarik	Lakukan pencarian *n* buah *k-itemset* paling menarik
Pengguna mampu memperoleh pola kaidah asosiasi dari *n* buah *k-itemset* paling menarik.	(1) Sistem menyediakan masukkan untuk pencarian pola kaidah asosisasi (2) Sistem mampu mencari pola kaidah asosisasi (3) Sistem mampu menampilkan pola kaidah asosisasi	Lakukan pencarian pola kaidah asosiasi	lakukan pencarian pola kaidah asosiasi
Pengguna mampu memperoleh informasi penggunaan memori saat aplikasi dijalankan.	(1) Sistem mampu menampilkan informasi penggunaan memori	Lakukan pemantauan memori	Lakukan pemantauan memori
Pengguna mampu memperoleh menyimpan data hasil pencarian *n* buah *k-itemset* paling menarik dan pola kaidah asosiasi	(1) Sistem menyediakan masukkan untuk menyimpan data (2) Sistem mampu menyimpan data hasil pencarian *n* buah *k-itemset* paling menarik dan pola kaidah asosiasi	Lakukan penyimpanan data	Lakukan penyimpanan data
Pengguna mampu memperoleh informasi aplikasi.	(1) Sistem mampu menampilkan informasi aplikasi	Lakukan permintaan informasi	Lakukan permintaan informasi

LAMPIRAN B
SEGMEN PROGRAM

(a) Segmen program kelas `item`

```
(1)  Package ta_bomo;
(2)  import java.util.*;
(3)  public class Item implements Comparable{
(4)      public int item;
(5)      public int count;
(6)      public Item(){ }
(7)      public Item(int i, int c) {
(8)        item = i;
(9)        count = c;
(10)     }
(11)     public int compareTo(Object o)  {
(12)       Item other = (Item)o;
(13)       return (count - other.count);
(14)     }
(15)
(16)     public String toString(){
(17)       return "<" + item + ", " + count + ">";
(18)     }
(19)   }
```

(b) Segmen program kelas `TreeNode`

```
(1)  package ta_bomo;
(2)  public class TreeNode{
(3)      public int item;
(4)      public int count;
(5)      public int seq_num;
(6)      public int header_index;
(7)      public TreeNode parent;
(8)      public TreeNode child;
(9)      public TreeNode sibling;
(10)     public TreeNode next;
(11)     public TreeNode(){ }
(12)     public TreeNode(int i, int c, int sn, int hi, TreeNode p,
     TreeNode s, TreeNode n)    {
(13)       item = i;
(14)       count = c;
(15)       seq_num = sn;
(16)       header_index = hi;
(17)       parent = p;
(18)       sibling = s;
(19)       next = n;
(20)     }
(21)   }
```

(c) Segmen program kelas `Kitemsets`

```
(1)  package ta_bomo;
(2)  import java.util.ArrayList;
(3)  public class Kitemsets implements java.io.Serializable{
(4)    private static final int increment = 5;
(5)    private int kapasitas;
(6)    private int size;
(7)    private int[] Itemset;
(8)    private int supportCount;
(9)    public Kitemsets(){ kapasitas=increment;
(10)       size=0;
(11)       Itemset=new int[kapasitas];
(12)       supportCount=0;
(13)   }
(14)   public Kitemsets(int kitemset){
(15)     if (kitemset == 0)
(16)       throw new IllegalArgumentException("null itemset");
(17)     Itemset = new int[1];
(18)     size = 1;
(19)     Itemset[0] = kitemset;
(20)    }
(21)   public Kitemsets(Kitemsets kitemset){
(22)     if (kitemset == null)
(23)       throw new IllegalArgumentException("null itemset");
(24)     kapasitas = kitemset.kapasitas;
(25)     Itemset = new int[kapasitas];
(26)     size = kitemset.size;
(27)     for (int i = 0; i < size; i++)Itemset[i] =
     kitemset.Itemset[i];
(28)     supportCount = kitemset.supportCount;
(29)   }
(30)   public int getItem(int index){
(31)   return Itemset[index];
(32)   }
(33)     public boolean CompareItemsets(Kitemsets secondItemset){
(34)    if(size!=secondItemset.size)return false;
(35)    else{
(36)         for (int i=0;i<Itemset.length;i++){
(37)             if(Itemset[i]!=secondItemset.getItem(i)){
(38)                 return false;
(39)             }}return true;
(40)       }}
(41)   public void setsupportCount(int newSupport){
(42)     if (newSupport < 0)
(43)       throw new IllegalArgumentException("newSupport must be
     >= 0");
(44)     supportCount = newSupport;
(45)   }
(46)   public int getsupportCount(){
(47)     return supportCount;
(48)   }
(49)   public int size(){
(50)     return size; }
(51)   public boolean add(int item){
(52)     if (item <= 0)
```

```
(53)        throw new IllegalArgumentException("item must be a
    positive value");
(54)      if (size == 0)
(55)        Itemset[0] = item;
(56)      else{
(57) // look for place to insert item
(58) int index;
(59) for (index = 0; index < size && item > Itemset[index];
    index++);
(60) // if item was already in itemItemset then return
(61) if (index < size && item == Itemset[index])
(62)   return false;
(63) // if Itemset is full then allocate new array
(64) if (size == kapasitas){
(65)      kapasitas = size + increment;
(66)      int[] a = new int[kapasitas];
(67)      int i;
(68)      for (i = 0; i < index; i++)
(69)        a[i] = Itemset[i];
(70)      // insert new item
(71)      a[i] = item;
(72)      for ( ; i < size; i++)
(73)        a[i + 1] = Itemset[i];
(74)      Itemset = a;}
(75) // otherwise make place and insert new item
(76) else{
(77)      int i;
(78)      for (i = size; i > index; i--)
(79)        Itemset[i] = Itemset[i - 1];
(80)      Itemset[i] = item;
(81)    }}
(82)      // update size
(83)      size++;
(84)      return true;
(85)    }
(86)   public boolean equals(Object o){
(87)      if (o == this)
(88)        return true;
(89)      if (!(o instanceof Kitemsets))
(90)        return false;
(91)      Kitemsets set = (Kitemsets)o;
(92)      if (size != set.size())
(93)        return false;
(94)      for (int i = 0; i < size; i++)
(95)        if (Itemset[i] != set.Itemset[i])
(96) return false;
(97)      return true;
(98)    }
(99)   public String toString()
(100)    {
(101)      String s = " ";
(102)      for (int i = 0; i < size; i++)
(103)          s += "I"+Itemset[i] + " ";
(104)        s += " ";
(105)      return s;
(106)    }}
```

LAMPIRAN C
HASIL 3 BUAH 3-ITEMSET PALING MENARIK DARI APLIKASI UNTUK UJI KEBENARAN

(a) Hasil *3* buah *3-itemset* paling menarik menggunakan metode penelusuran dari awal *FP-tree*

```
PENGGALIAN DATA MENGGUNAKAN ALGORITMA BOMO
Dataset : uji_kebenaran
Urutan pembangkitan conditional FP-tree : Top Down
Nilai maksimal k-itemset  = 3
Nilai n paling menarik    = 3
Batas ambang dukungan minimum = 3
Lama proses  (0 detik : 62 milidetik)

..................................................... HASIL .....................................................
Hasil 1-itemsets
Batas ambang dukungan minimum 1-itemsets = 3
  I4  S = 4
  I3  S = 4
  I8  S = 3

Hasil 2-itemsets
Batas ambang dukungan minimum 2-itemsets = 3
  I3 I4  S = 3
  I4 I6  S = 3
  I3 I8  S = 3

Hasil 3-itemsets
Batas ambang dukungan minimum 3-itemsets = 3
  I3 I4 I6  S = 3
  I1 I3 I4  S = 3
  I1 I4 I6  S = 3
```

(b) Hasil *3* buah *3-itemset* paling menarik menggunakan metode penelusuran dari tengah *FP-tree.*

```
PENGGALIAN DATA MENGGUNAKAN ALGORITMA BOMO
Dataset : uji_kebenaran
Urutan pembangkitan conditional FP-tree : Middle
Nilai maksimal k-itemset = 3
Nilai n paling menarik    = 3
Batas ambang dukungan minimum = 3
Lama proses  (0 detik : 47 milidetik)

..................................................... HASIL .....................................................
Hasil 1-itemsets
Batas ambang dukungan minimum 1-itemsets = 3
  I4  S = 4
  I3  S = 4
  I8  S = 3

Hasil 2-itemsets
Batas ambang dukungan minimum 2-itemsets = 3
```

```
I4 I6   S = 3
I3 I6   S = 3
I3 I8   S = 3

Hasil 3-itemsets
Batas ambang dukungan minimum 3-itemsets = 3
I1 I3 I4   S = 3
I1 I4 I6   S = 3
I3 I4 I6   S = 3
```

(c) Hasil *3* buah *3-itemset* paling menarik menggunakan metode penelusuran dari akhir *FP-tree*

```
PENGGALIAN DATA MENGGUNAKAN ALGORITMA BOMO
Dataset : uji_kebenaran
Urutan pembangkitan conditional FP-tree : Bottom Up
Nilai maksimal k-itemset = 3
Nilai n paling menarik     = 3
Batas ambang dukungan minimum = 3
Lama proses  (0 detik : 46 milidetik)

..................................................... HASIL .....................................................
Hasil 1-itemsets
Batas ambang dukungan minimum 1-itemsets = 3
I4   S = 4
I3   S = 4
I8   S = 3

Hasil 2-itemsets
Batas ambang dukungan minimum 2-itemsets = 3
I1 I4   S = 3
I1 I3   S = 3
I1 I6   S = 3

Hasil 3-itemsets
Batas ambang dukungan minimum 3-itemsets = 3
I1 I3 I4   S = 3
I1 I4 I6   S = 3
I1 I3 I6   S = 3
```

(d) Hasil *3* buah *3-itemset* paling menarik menggunakan pendekatan *loopback*

```
PENGGALIAN DATA MENGGUNAKAN PENDEKATAN LOOPBACK
Dataset : uji_kebenaran
Faktor penurunan           = 0.5
Nilai maksimal k-itemset   = 3
Nilai n paling menarik     = 3
Batas ambang dukungan minimum = 3
Lama proses  (0 detik : 31 milidetik)
```

MILIK PERPUSTAKAAN
ITS

..... HASIL .....

Hasil 1-itemsets
Batas ambang dukungan minimum 1-itemsets = 3
I4 S = 4
I3 S = 4
I8 S = 3

Hasil 2-itemsets
Batas ambang dukungan minimum 2-itemsets = 3
I3 I4 S = 3
I4 I6 S = 3
I3 I8 S = 3

Hasil 3-itemsets
Batas ambang dukungan minimum 3-itemsets = 3
I3 I4 I6 S = 3
I1 I3 I4 S = 3
I1 I4 I6 S = 3

LAMPIRAN D
HASIL POLA KAIDAH ASOSIASI DARI APLIKASI UNTUK UJI KEBENARAN

(a) Pola kaidah asosiasi untuk hasil *3* buah *3-itemset* paling menarik menggunakan metode penelusuran dari awal *FP-tree*

```
{Acetyl-L-Carnitine}-->{AcidAmpules}[S=60.0%,C=75.0%]
{AcidAmpules}-->{Acetyl-L-Carnitine}[S=60.0%,C=75.0%]
{AcidAmpules}-->{AcidEyeCreme}[S=60.0%,C=75.0%]
{AcidEyeCreme}-->{AcidAmpules}[S=60.0%,C=100.0%]
{Acetyl-L-Carnitine}-->{AcidSerum}[S=60.0%,C=75.0%]
{AcidSerum}-->{Acetyl-L-Carnitine}[S=60.0%,C=100.0%]
{Acetyl-L-Carnitine}-->{AcidAmpules,AcidEyeCreme}[S=60.0%,C=75.0%]
{Acetyl-L-Carnitine,AcidAmpules}-->{AcidEyeCreme}[S=60.0%,C=100.0%]
{AcidAmpules}-->{Acetyl-L-Carnitine,AcidEyeCreme}[S=60.0%,C=75.0%]
{AcidAmpules,AcidEyeCreme}-->{Acetyl-L-Carnitine}[S=60.0%,C=100.0%]
{AcidEyeCreme}-->{Acetyl-L-Carnitine,AcidAmpules}[S=60.0%,C=100.0%]
{Acetyl-L-Carnitine,AcidEyeCreme}-->{AcidAmpules}[S=60.0%,C=100.0%]
{AbsorbAid}-->{Acetyl-L-Carnitine,AcidAmpules}[S=60.0%,C=100.0%]
{AbsorbAid,Acetyl-L-Carnitine}-->{AcidAmpules}[S=60.0%,C=100.0%]
{Acetyl-L-Carnitine}-->{AbsorbAid,AcidAmpules}[S=60.0%,C=75.0%]
{Acetyl-L-Carnitine,AcidAmpules}-->{AbsorbAid}[S=60.0%,C=100.0%]
{AcidAmpules}-->{AbsorbAid,Acetyl-L-Carnitine}[S=60.0%,C=75.0%]
{AbsorbAid,AcidAmpules}-->{Acetyl-L-Carnitine}[S=60.0%,C=100.0%]
{AbsorbAid}-->{AcidAmpules,AcidEyeCreme}[S=60.0%,C=100.0%]
{AbsorbAid,AcidAmpules}-->{AcidEyeCreme}[S=60.0%,C=100.0%]
{AcidAmpules}-->{AbsorbAid,AcidEyeCreme}[S=60.0%,C=75.0%]
{AcidAmpules,AcidEyeCreme}-->{AbsorbAid}[S=60.0%,C=100.0%]
{AcidEyeCreme}-->{AbsorbAid,AcidAmpules}[S=60.0%,C=100.0%]
{AbsorbAid,AcidEyeCreme}-->{AcidAmpules}[S=60.0%,C=100.0%]
```

(b) Pola kaidah asosiasi untuk hasil *3* buah *3-itemset* paling menarik menggunakan menggunakan metode penelusuran dari tengah *FP-tree*

```
{AcidAmpules}-->{AcidEyeCreme}[S=60.0%,C=75.0%]
{AcidEyeCreme}-->{AcidAmpules}[S=60.0%,C=100.0%]
{Acetyl-L-Carnitine}-->{AcidEyeCreme}[S=60.0%,C=75.0%]
{AcidEyeCreme}-->{Acetyl-L-Carnitine}[S=60.0%,C=100.0%]
{Acetyl-L-Carnitine}-->{AcidSerum}[S=60.0%,C=75.0%]
{AcidSerum}-->{Acetyl-L-Carnitine}[S=60.0%,C=100.0%]
{AbsorbAid}-->{Acetyl-L-Carnitine,AcidAmpules}[S=60.0%,C=100.0%]
{AbsorbAid,Acetyl-L-Carnitine}-->{AcidAmpules}[S=60.0%,C=100.0%]
{Acetyl-L-Carnitine}-->{AbsorbAid,AcidAmpules}[S=60.0%,C=75.0%]
{Acetyl-L-Carnitine,AcidAmpules}-->{AbsorbAid}[S=60.0%,C=100.0%]
{AcidAmpules}-->{AbsorbAid,Acetyl-L-Carnitine}[S=60.0%,C=75.0%]
{AbsorbAid,AcidAmpules}-->{Acetyl-L-Carnitine}[S=60.0%,C=100.0%]
{AbsorbAid}-->{AcidAmpules,AcidEyeCreme}[S=60.0%,C=100.0%]
{AbsorbAid,AcidAmpules}-->{AcidEyeCreme}[S=60.0%,C=100.0%]
```

{AcidAmpules}-->{AbsorbAid,AcidEyeCreme}[S=60.0%,C=75.0%]
{AcidAmpules,AcidEyeCreme}-->{AbsorbAid}[S=60.0%,C=100.0%]
{AcidEyeCreme}-->{AbsorbAid,AcidAmpules}[S=60.0%,C=100.0%]
{AbsorbAid,AcidEyeCreme}-->{AcidAmpules}[S=60.0%,C=100.0%]
{Acetyl-L-Carnitine}-->{AcidAmpules,AcidEyeCreme}[S=60.0%,C=75.0%]
{Acetyl-L-Carnitine,AcidAmpules}-->{AcidEyeCreme}[S=60.0%,C=100.0%]
{AcidAmpules}-->{Acetyl-L-Carnitine,AcidEyeCreme}[S=60.0%,C=75.0%]
{AcidAmpules,AcidEyeCreme}-->{Acetyl-L-Carnitine}[S=60.0%,C=100.0%]
{AcidEyeCreme}-->{Acetyl-L-Carnitine,AcidAmpules}[S=60.0%,C=100.0%]
{Acetyl-L-Carnitine,AcidEyeCreme}-->{AcidAmpules}[S=60.0%,C=100.0%]

(c) Pola kaidah asosiasi untuk hasil *3* buah *3-itemset* paling menarik menggunakan pendekatan *loopback*

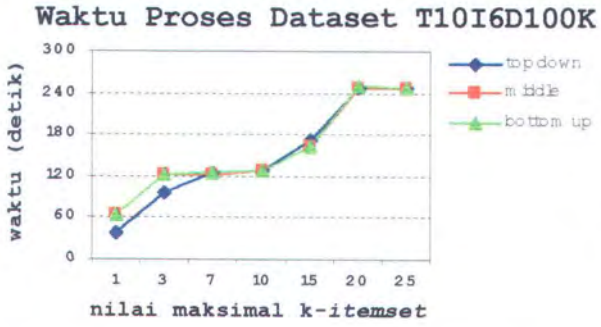
{Acetyl-L-Carnitine}-->{AcidAmpules}[S=60.0%,C=75.0%]
{AcidAmpules}-->{Acetyl-L-Carnitine}[S=60.0%,C=75.0%]
{AcidAmpules}-->{AcidEyeCreme}[S=60.0%,C=75.0%]
{AcidEyeCreme}-->{AcidAmpules}[S=60.0%,C=100.0%]
{Acetyl-L-Carnitine}-->{AcidSerum}[S=60.0%,C=75.0%]
{AcidSerum}-->{Acetyl-L-Carnitine}[S=60.0%,C=100.0%]
{Acetyl-L-Carnitine}-->{AcidAmpules,AcidEyeCreme}[S=60.0%,C=75.0%]
{Acetyl-L-Carnitine,AcidAmpules}-->{AcidEyeCreme}[S=60.0%,C=100.0%]
{AcidAmpules}-->{Acetyl-L-Carnitine,AcidEyeCreme}[S=60.0%,C=75.0%]
{AcidAmpules,AcidEyeCreme}-->{Acetyl-L-Carnitine}[S=60.0%,C=100.0%]
{AcidEyeCreme}-->{Acetyl-L-Carnitine,AcidAmpules}[S=60.0%,C=100.0%]
{Acetyl-L-Carnitine,AcidEyeCreme}-->{AcidAmpules}[S=60.0%,C=100.0%]
{AbsorbAid}-->{Acetyl-L-Carnitine,AcidAmpules}[S=60.0%,C=100.0%]
{AbsorbAid,Acetyl-L-Carnitine}-->{AcidAmpules}[S=60.0%,C=100.0%]
{Acetyl-L-Carnitine}-->{AbsorbAid,AcidAmpules}[S=60.0%,C=75.0%]
{Acetyl-L-Carnitine,AcidAmpules}-->{AbsorbAid}[S=60.0%,C=100.0%]
{AcidAmpules}-->{AbsorbAid,Acetyl-L-Carnitine}[S=60.0%,C=75.0%]
{AbsorbAid,AcidAmpules}-->{Acetyl-L-Carnitine}[S=60.0%,C=100.0%]
{AbsorbAid}-->{AcidAmpules,AcidEyeCreme}[S=60.0%,C=100.0%]
{AbsorbAid,AcidAmpules}-->{AcidEyeCreme}[S=60.0%,C=100.0%]
{AcidAmpules}-->{AbsorbAid,AcidEyeCreme}[S=60.0%,C=75.0%]
{AcidAmpules,AcidEyeCreme}-->{AbsorbAid}[S=60.0%,C=100.0%]
{AcidEyeCreme}-->{AbsorbAid,AcidAmpules}[S=60.0%,C=100.0%]
{AbsorbAid,AcidEyeCreme}-->{AcidAmpules}[S=60.0%,C=100.0%]

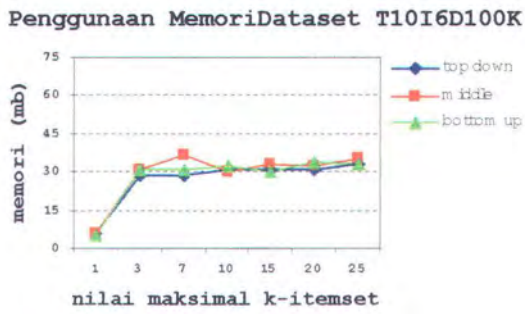
LAMPIRAN E
UJI KINERJA APLIKASI

(a) **Uji kinerja dataset T10I6D100K terhadap terhadap perubahan parameter *n* paling menarik dan maksimal k-*itemset***

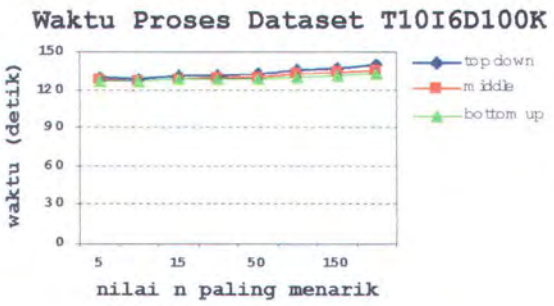
- **Urutan Pembangkitan dari *Conditional FP-tree***

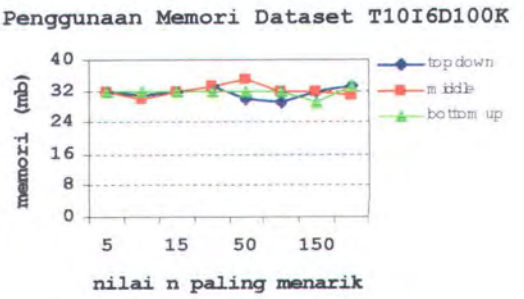
Dataset T10I6D100K					
Maksimal k-*itemset*	***n* Paling Menarik**	**Waktu Komputasi Proses (detik) Urutan Pembangkitan dari *Conditional FP-tree***			ξ
		Dari atas	**Dari tengah**	**Dari bawah**	
10	5	129	128	127	158
10	10	128	127	127	154
10	15	131	128	128	154
10	25	131	129	128	152
10	50	132	130	128	150
10	100	135	132	129	148
10	150	137	134	131	147
10	200	140	135	132	146
1	10	37	63	64	5460
3	10	97	123	123	716
7	10	125	123	124	453
10	10	128	127	127	154
15	10	171	163	162	3
20	10	247	249	250	1
25	10	247	248	247	1





Grafik waktu komputasi proses dan penggunaan memori untuk 10 buah paling menarik pada T10I6D100K

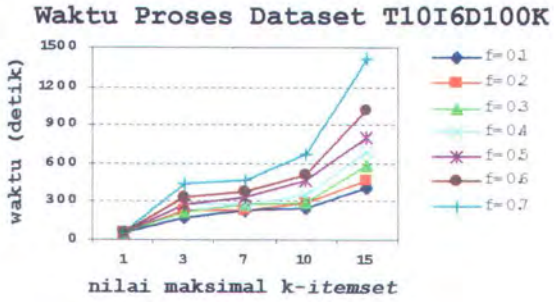


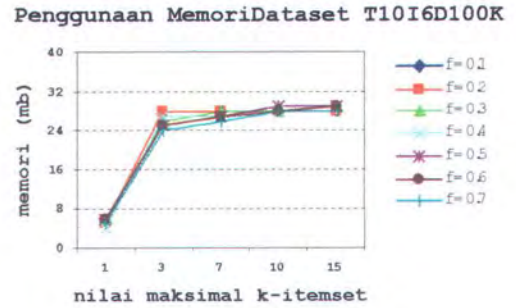


Grafik waktu komputasi proses dan penggunaan memori untuk 10-*itemset* pada T10I6D100K

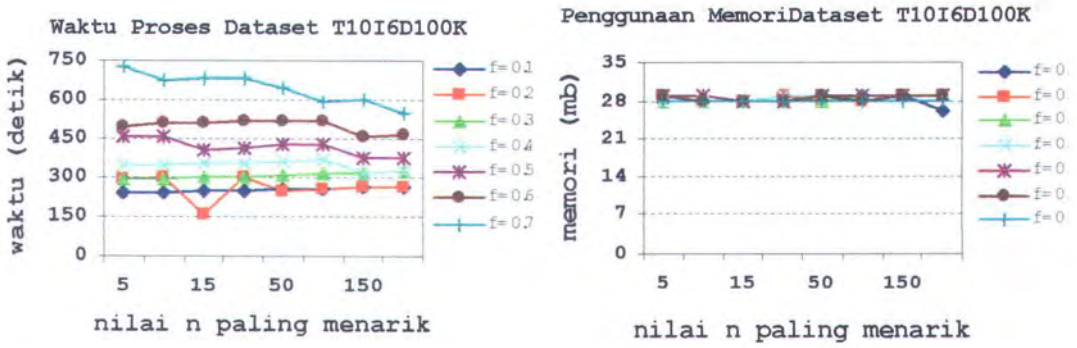
- **Pendekatan *loopback***

Dataset T10I6D100K									
Maksimal k-*itemset*	***n* Paling Menarik**	**Waktu Komputasi Proses (detik) Pendekatan Loopback**							**ξ**
		f = 0.1	f = 0.2	f = 0.3	f = 0.4	f = 0.5	f = 0.6	f = 0.7	
10	5	243	296	294	347	458	497	727	158
10	10	243	297	294	347	459	511	674	154
10	15	246	154	297	353	408	513	682	154
10	25	249	303	300	355	410	518	683	152
10	50	252	249	304	361	425	514	643	150
10	100	257	256	312	367	424	515	589	148
10	150	261	260	317	315	372	459	598	147
10	200	264	264	322	320	378	462	545	146
1	10	64	64	63	63	63	63	64	5460
3	10	171	226	225	279	276	329	434	716
7	10	229	229	283	280	332	384	467	453
10	10	243	297	294	347	459	511	674	154
15	10	411	464	577	688	800	1023	1416	3





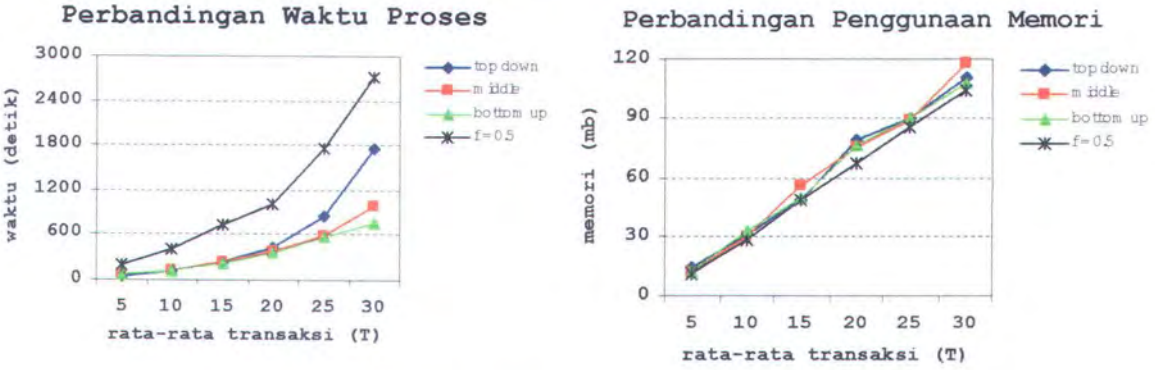
Grafik waktu komputasi proses dan penggunaan memori untuk 10 paling menarik pada T10I6D100K



Grafik waktu komputasi proses dan penggunaan memori untuk 10-*itemset* pada T10I6D100K

(b) Uji kinerja terhadap perubahan rata-rata transaksi

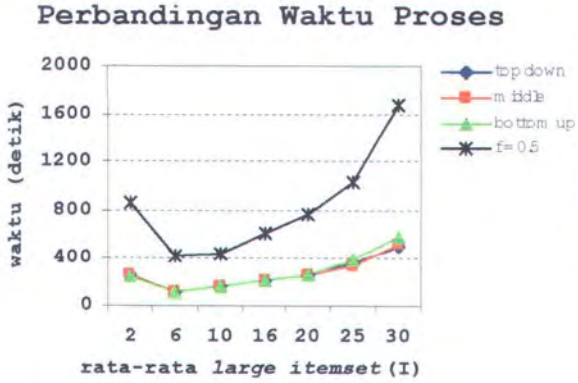
Txx.I6.D100K.L2K.M1K.C0,25							
No	T	Jumlah Record	Waktu Komputasional Proses (detik)				ξ
			Penelusuran *FP-tree*			*Loopback* (f = 0,5)	
			Dari atas	Dari tengah	Dari bawah		
1	5	674.489	58	68	68	195	157
2	10	1.033.162	117	114	113	408	148
3	15	1.498.115	237	226	218	736	211
4	20	1.992.530	426	372	352	1005	500
5	25	2.488.150	844	580	565	1758	327
6	30	2.986.370	1758	976	761	2714	374

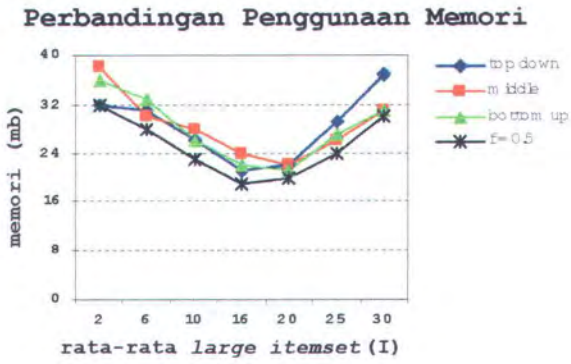


Grafik waktu komputasi proses dan penggunaan memori untuk berbagai nilai T

(c) Uji kinerja terhadap perubahan rata-rata *large itemset*

T10.Ixx.D100K.L2K.M1K.C0,25							
No	I	Jumlah Record	Waktu Komputasional Proses (detik)				ξ
			Penelusuran *FP-tree*			*Loopback* (f = 0,5)	
			Dari atas	Dari tengah	Dari bawah		
1	2	997.712	251	251	236	852	2
2	6	1.033.162	117	114	113	408	148
3	10	1.191.515	157	155	153	421	287
4	16	1.609.015	208	206	208	597	307
5	20	1.941.417	253	250	249	759	318
6	25	2.375.297	345	341	376	1024	366
7	30	2.902.559	498	508	566	1681	317

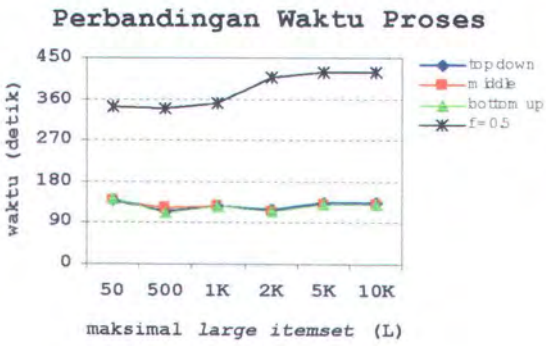


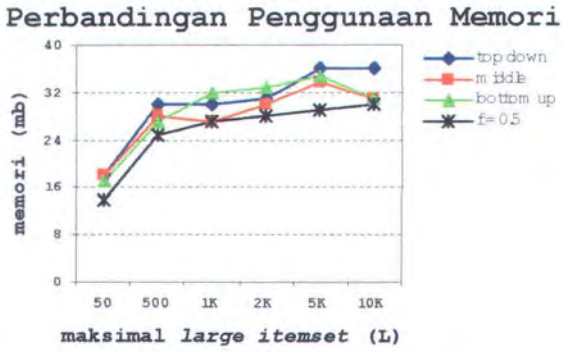


Grafik waktu komputasi proses dan penggunaan memori untuk berbagai nilai I

(d) Uji kinerja terhadap perubahan maksimal *large itemset*

T10.I6.D100K.Lxx.M1K.C0,25							
No	L	Jumlah Record	Waktu Komputasional Proses (detik)				ξ
			Penelusuran *FP-tree*			*Loopback* (f = 0,5)	
			Dari atas	Dari tengah	Dari bawah		
1	50	1.019.500	136	137	142	342	370
2	500	1.032.941	114	122	111	341	253
3	1K	1.031.928	126	125	124	350	178
4	2K	1.033.162	117	114	113	408	148
5	5K	1.037.206	134	130	129	419	83
6	10K	1.036.078	134	128	130	421	105

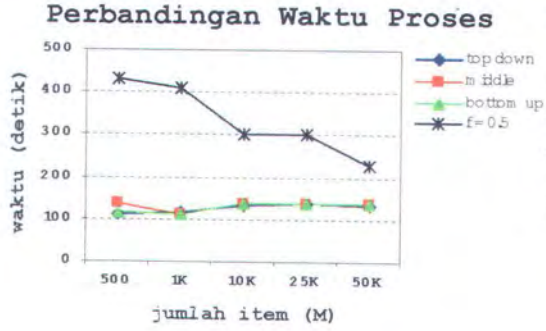


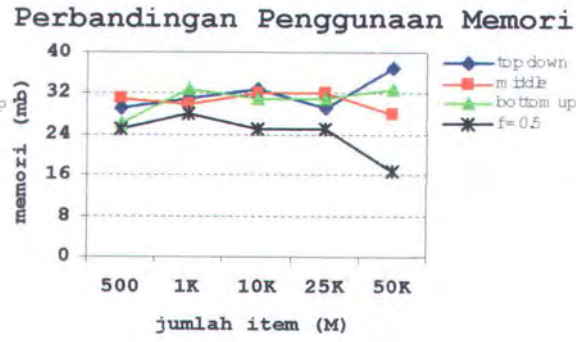


Grafik waktu komputasi proses dan penggunaan memori untuk berbagai nilai L

(e) Uji kinerja terhadap perubahan jumlah item

T10.I6.D100K.L2K.Mxx.C0,25							
No	M	Jumlah Record	Waktu Komputasional Proses (detik)				ξ
			Penelusuran *FP-tree*			*Loopback* (f = 0,5)	
			Dari atas	Dari tengah	Dari bawah		
1	500	1.032.641	111	139	115	430	150
2	1K	1.033.162	117	114	113	408	148
3	10K	1.036.494	133	136	136	301	146
4	25K	1.034.527	136	138	140	300	128
5	50K	1.038.832	134	136	136	228	215

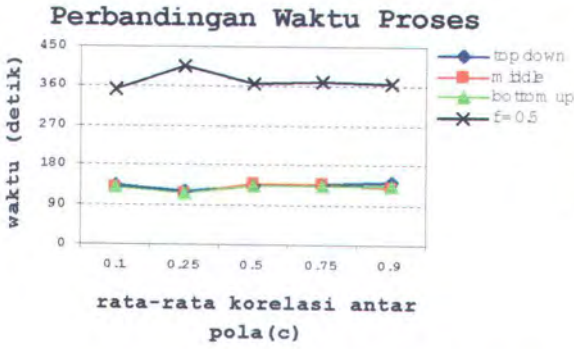


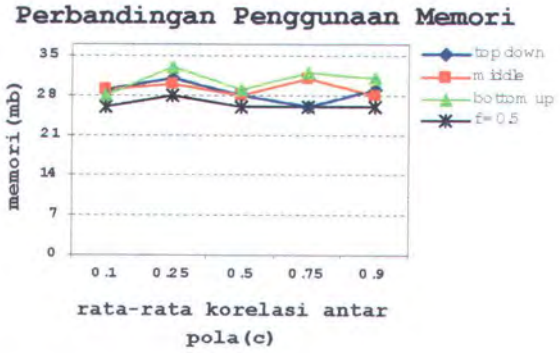


Grafik waktu komputasi proses dan penggunaan memori untuk berbagai nilai M

(f) Uji kinerja terhadap perubahan rata-rata korelasi antar pola

T10.I6.D100K.L2K.M1K.Cxx							
No	C	Jumlah Record	Waktu Komputasional Proses (detik)				ξ
			Penelusuran *FP-tree*			*Loopback* (f = 0,5)	
			Dari atas	Dari tengah	Dari bawah		
1	0.1	1.034.829	131	128	126	355	186
2	0.25	1.033.162	117	114	113	408	148
3	0.5	1.033.480	132	135	132	364	184
4	0.75	1.036.721	137	136	132	372	188
5	0.9	1.037.488	142	128	132	368	237

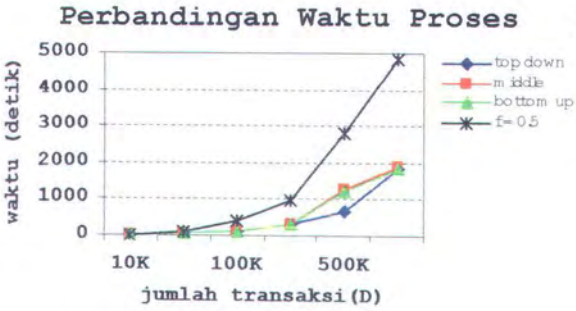


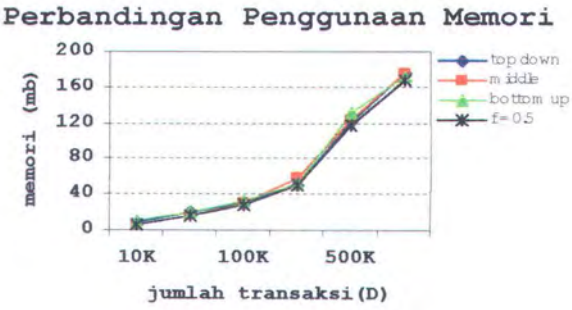


Grafik waktu komputasi proses dan penggunaan memori untuk berbagai nilai C

(g) Uji kinerja terhadap perubahan jumlah transaksi

T10.I6.D100K.Lxx.M1K.C0,25							
No	D	Jumlah Record	Waktu Komputasional Proses (detik)				ξ
			Penelusuran *FP-tree*			*Loopback* (f = 0,5)	
			Dari atas	Dari tengah	Dari bawah		
1	10K	103.822	7	9	9	24	23
2	50K	516.471	45	43	42	110	184
3	100K	1.033.162	117	114	113	408	148
4	200K	2.074.419	320	297	298	952	504
5	500K	5.167.112	682	1262	1229	2781	1139
6	750K	7.784.158	1857	1889	1845	4863	1761





Grafik waktu komputasi proses dan penggunaan memori untuk berbagai nilai D