IMPLEMENTASI METODE PRAPROSES DATA MENGGUNAKAN METODE EXANTE DALAM PEMBUATAN APLIKASI PENGGALIAN POLA ASOSIASI

TUGAS AKHIR

Disusun Oleh :

MUHAMMAD ANSHAR
NRP. 5201 100 022

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

IMPLEMENTASI METODE PRAPROSES DATA MENGGUNAKAN METODE EXANTE DALAM PEMBUATAN APLIKASI PENGGALIAN POLA ASOSIASI

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

Prof. Dr. Ir. Arif Djunaidy, MSc

NIP. 131 633 403

SURABAYA

JANUARI 2006

Abstrak

ExAnte merupakan sebuah metode praproses yang mengombinasikan dua buah pembatas dalam penambangan pola asosiasi, yaitu *antimonotone* dan *monotone*. Antimonoton merupakan sifat dari penambangan pola dimana apabila nilai support dari sebuah itemset berada dibawah nilai yang telah ditentukan oleh pengguna, maka superset dari itemset tersebut juga tidak akan memenuhi nilai support yang diinginkan. Sifat antimonoton ini berguna untuk memperkecil ruang pencarian yang ada, karena itemset yang tidak memenuhi syarat pada level awal (pola kecil), juga tidak akan memenuhi pada level berikutnya. Kebalikan dari batasan antimonoton disebut batasan monoton, yang dapat diperoleh dengan mencari itemset yang memiliki nilai monotonitas lebih besar dari syarat yang diberikan oleh pengguna.

Dalam Tugas Akhir ini dibuat sebuah aplikasi pencarian pola yang *frequent* dengan mengimplementasikan metode praproses data *ExAnte*. *ExAnte* bekerja dengan menyinergikan kedua batasan diatas. Sinergi antar dua batasan ini dilakukan dalam dua tahapan proses. Tahapan pertama adalah proses μ-reduction yang pada dasarnya melakukan penghapusan transaksi-transaksi yang tidak lolos batasan monoton dari dataset transaksi. Tahapan kedua adalah proses α-reduction yang berkaitan dengan proses penghapusan item-item non *frequent* dari dataset. Dengan adanya penyelarasan dari kedua batasan tersebut, maka keuntungan yang didapatkan dapat lebih besar dari jumlah keuntungan antara keduanya secara individu.

Pada aplikasi yang telah berhasil dibuat, dilakukan uji coba terhadap beberapa jenis data dengan karakteristik yang berbeda. Hasil uji coba menunjukkan bahwa penerapan metode *ExAnte* secara signifikan mampu mereduksi ukuran dari dataset. Semakin besar nilai batasan baik itu batasan monoton maupun antimonon, persentase reduksi data yang dihasilkan akan ikut meningkat. Dengan kata lain, semakin selektif batasan yang diberikan akan menghasilkan persentase reduksi yang semakin baik. Efek dari reduksi data ini adalah meningkatnya kinerja proses penggalian pola terhadap dataset dengan tanpa kehilangan pola-pola yang menarik.

***Kata kunci**: penggalian pola, batasan monoton, batasan antimonoton, reduksi ruang pencarian.*

Kata Pengantar

Segala puji bagi Allah Swt yang Maha Pengasih lagi Maha Penyayang. Atas rahmat, taufik dan hidayah-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul:

IMPLEMENTASI METODE PRAPROSES DATA

MENGGUNAKAN METODE EXANTE

DALAM PEMBUATAN APLIKASI

PENGGALIAN POLA ASOSIASI

Tugas Akhir ini merupakan syarat akademis bagi para mahasiswa dalam rangka menyelesaikan studi kesarjanaan pada Jurusan Sistem Informasi, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Tugas Akhir yang penulis kembangkan ini bertujuan untuk mengimplemenatasikan metode ExAnte dalam sebuah apliakasi penggalian pola sehubungan dengan proses reduksi data. Semoga Tugas Akhir ini dapat bermanfaat bagi perkembangan ilmu pengetahuan, khususnya di bidang data mining.

Surabaya, Februari 2006

Penulis

Ucapan Terima Kasih

Segala puji bagi Allah Swt yang Maha Pengasih lagi Maha Penyayang. Pada kesempatan ini, penulis menyampaikan penghargaan dan rasa terima kasih yang sebesarnya-besarnya atas bimbingan, dukungan, doa, serta bantuan yang diberikan kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini kepada:

1. Papa dan mama yang senantiasa memberi semangat, dukungan, dan doa, sehingga penulis dapat menyelesaikan tugas akhir ini.
2. Bapak Dr. Ir. Arif Djunaidy, M.Sc selaku dosen pembimbing, yang telah memberikan banyak masukan dan bimbingan.
3. Vetrylla Prima Z, yang telah meluangkan banyak waktu, tenaga dan pikiran bagi penulis, makasih banyak ta.
4. Kakak Indah, Mas Andre, Yusar, Bagus. Terima kasih buat doa dan dukungannya.
5. Teman-teman senasib seperjuangan, Wawa, Toes, Angga, Didit, Risa. Terima kasih dan sukses selalu.
6. Teman-teman SI'01 , Poleng, Boss, Soni, Beni, Ayi', Irfan, Ghofur, Dimas, Kindi, Ryan, Pipu', Arie, Prisa, Cemit, Dephta, Icha, Pangestu, Sofyan, Bagus, Mirza, Dudut, Gesti, Diyah, Eka, Tante, Arin, Ratih, Yeni, Mbet, Reza, Mo, Miko, Mamang, Adhi, Yudi. Terima kasih untuk kebersamaannya.
7. Semua warga plus pengurus lab SI, Te & Zaza' ☺. Terima kasih bantuan dan dukungannya.
8. Anak-anak B-1, Terima kasih atas dorongan semangat serta kebersamaannya.
9. Seluruh warga Sistem Informasi, Terima kasih.
10. Semua pihak yang telah mendukung penulis hingga Tugas Akhir ini selesai.

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

Daftar Isi

Daftar Gambar

Daftar Tabel

Daftar Segmen program

BAB I

PENDAHULUAN

BAB I

PENDAHULUAN

1.1. Latar belakang

Dalam kehidupan sehari-hari, informasi mengalir terus menerus. Banyak penelitian penggalian data telah dilakukan untuk mengolah informasi-informasi tersebut menjadi pengetahuan yang berguna. Salah satunya adalah penggalian pola asosiasi. Yaitu pencarian beberapa itemset (kumpulan item) yang sering muncul secara bersamaan dari sebuah basis data transaksi (frequent itemset). Contoh pemanfaatan dari penggalian ini adalah analisis *market basket*. Seiring dengan waktu, informasi yang mengalir tidak berkurang melainkan terus bertambah. Permasalahan yang muncul adalah bagaimana kita mencari pola atau itemset yang benar-benar berguna diantara tumpukan informasi yang terus berkembang itu?

Kebutuhan akan penyaringan informasi dan teknik pemfokusan terhadap informasi yang menarik menjadi sangat penting. Secara umum, pengetahuan/informasi yang dicari tersembunyi pada bagian kecil pola diantara sejumlah besar kumpulan pola yang tidak berhubungan yang dibangkitkan dari data berjumlah sangat besar. Keadaan yang seperti ini (terdapat data yang tidak relevan), tentu saja akan merugikan. Terdapat dua alasan utama mengapa hal tersebut dianggap merugikan. Yang pertama adalah menurunnya kinerja, membuat teknologi penambangan menjadi tidak efisien atau bahkan tidak dapat digunakan. Sedangkan alasan kedua adalah kesulitan dalam mencari pola yang dapat dianggap menarik atau berguna diantara pola-pola yang tidak berguna.

Tujuan utama dari praproses data adalah mengolah kumpulan data berukuran besar sehingga menghasilkan kumpulan data yang berukuran lebih kecil dengan tanpa kehilangan pola-pola yang berguna atau menarik. Salah satu pendekatan yang paling menjanjikan adalah *constraint-based data mining* (penambangan data berdasarkan batasan), dimana pengguna menjalankan proses

penambangan dengan menetapkan batasan-batasan agar pengetahuan yang dihasilkan telah memenuhi batasan, serta dikategorikan sebagai berguna atau menarik. Batasan merupakan pertahanan pengguna terhadap membanjirnya aliran infomasi. Sebab dengan adanya batasan, maka besarnya ruang pencarian serta data masukan dapat diperkecil. Dengan demikian, waktu serta sumber daya yang dibutuhkan juga ikut berkurang. Selain itu, batasan juga mengakibatkan pemusatan perhatian pada pengetahuan yang berguna, sehingga jumlah pola yang dihasilkan akan berkurang pada pola-pola yang berpotensi saja.

Metode praproses ExAnte merupakan salah satu metode sederhana yang mengimplementasikan penggunaan batasan dalam mengurangi besarnya ruang pencarian serta data masukan seperti yang dijelaskan sebelumnya. ExAnte memanfaatkan batasan untuk mengurangi ukuran data yang akan diproses menjadi dataset berisi data yang menarik saja. Pengurangan data ini, selanjutnya akan mengakibatkan pengurangan jumlah kandidat pola secara dramatis. Hal ini tentu saja mengakibatkan meningkatnya kinerja pada proses penambangan selanjutnya.

1.2. Tujuan

Tujuan dari tugas akhir ini adalah mendesain dan mengimplementasikan metode ExAnte sebagai metode praproses data dalam sebuah sistem aplikasi penggalian pola asosiasi serta membandingkan kinerja aplikasi dengan dan tanpa menggunakan ExAnte.

1.3. Permasalahan

Permasalahan yang diangkat dalam tugas akhir ini adalah:

- Bagaimana cara mendesain dan mengimplementasikan metode ExAnte Sebagai metode praproses data?
- Bagaimana cara mengintegrasikan metode ExAnte dengan algoritma pencarian pola yang sudah ada ke dalam sebuah sistem aplikasi penggalian pola asosiasi?

- Bagaimana cara mengevaluasi kinerja aplikasi penggalian pola asosiasi dengan dan tanpa menggunakan metode ExAnte?

1.4. Batasan Masalah

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

- Tugas akhir akan difokuskan pada penerapan metode ExAnte pada proses penggalian pola asosiasi
- Untuk evaluasi, uji coba akan dilakukan dengan menggunakan beberapa algoritma penambangan pola sebagai pembanding
- Pengembangan aplikasi akan dititikberatkan pada implementasi metode ExAnte sebagai metode praproses data
- Pengembangan aplikasi akan menggunakan sistem operasi Windows, bahasa pemrograman Java serta database Oracle

1.5. Metodologi

Pembuatan tugas akhir ini terbagi menjadi beberapa tahap pengerjaan yang tertera sebagai berikut :

1.5.1. Studi kepustakaan

Studi kepustakaan dilakukan untuk mencari informasi mengenai metode serta algoritma yang akan digunakan dalam tugas akhir. Pada tahap ini, penulis mencari dan merangkum kepustakaan apa saja yang dapat menunjang pengerjaan tugas akhir ini. Diantaranya yaitu pencarian informasi mengenai metode ExAnte serta penerapannya pada algoritma penggalian pola asosiasi.

1.5.2. Desain aplikasi penggalian pola

Pada tahapan ini, akan dibuat rancangan dari sistem yang ingin dibuat nantinya. Desain sistem akan memfokuskan pada implementasi metode ExAnte

pada aplikasi, bagaimana mengolah data mentah serta bagaimana proses komputerisasinya.

Pada masukan, didesain supaya dapat memenuhi semua kebutuhan aplikasi. Baik itu atribut yang akan digunakan maupun parameter-parameter yang akan berpengaruh seperti nilai minimum support serta batasan lain. Sedangkan pada bagian proses aplikasi, didesain supaya dapat mengimplementasikan metode ExAnte dengan baik. Luarannya didesain agar dapat digunakan serta dimanfaatkan pada proses penambangan selanjutnya.

1.5.3. Implementasi aplikasi

Pada tahapan ini, difokuskan pada pembuatan aplikasi sesuai dengan desain yang ada. Untuk data masukan, akan didapatkan dengan cara menghubungkan aplikasi dengan database transaksi. Data transaksi merupakan data sintentis hasil pembangkitan serta data yang didapatkan dari internet. Pemrosesan, dilakukan oleh aplikasi yang dibuat dengan bahasa pemrograman java, dan berjalan pada sistem operasi windows. Keluaran dari aplikasi merupakan sebuah kumpulan data transaksi baru yang kemudian akan diolah pada proses penambangan selanjutnya.

1.5.4. Uji Coba dan Evaluasi

Pada tahapan ini akan dilakukan uji coba terhadap aplikasi yang telah dibuat. Dan akan ditelusuri semua fitur yang dimiliki oleh aplikasi ini. Parameter keberhasilan yang dipakai :

- Ketepatan, perbandingan luaran yang dihasilkan oleh sistem dengan hasil perhitungan secara manual dengan menggunakan tool lain.
- Kecepatan, untuk mendapatkan waktu proses yang seminimal mungkin
- Efisiensi, Perbandingan sumber daya yang dibutuhkan oleh aplikasi.
- Fleksibilitas, kemampuan respon aplikasi terhadap berbagai macam alternatif masukan yang ada

1.5.5. Penyusunan Buku Tugas Akhir

Pembuatan dokumentasi dari keseluruhan isi Tugas Akhir. Pendokumentasian ini diharapkan akan berguna untuk pengkajian lebih lanjut dalam pengembangan sistem.

1.6. Sistematika Penulisan

Penyusunan laporan tugas akhir ini dapat dikelompokkan sebagai berikut. Bab I membahas mengenai Pendahuluan. Dalam bab ini dibahas hal-hal yang berhubungan dengan rencana pengerjaan tugas akhir ini. Berisi latar belakang, tujuan dan perumusan masalah, pembatasan dan asumsi masalah, dan sistematika penulisan.

Bab II membahas mengenai Dasar Teori. Bab ini membahas teori-teori yang menunjang dalam pengerjaan tugas akhir ini ditinjau dari sisi keilmuan penambangan data yang berupa pencarian pola frequent. Berisi hasil studi literatur mengenai algoritma ExAnte sebagai metode praproses data, serta beberapa algoritma penambangan pola yang digunakan dalam aplikasi.

Bab III membahas Desain Aplikasi. Bab ini membahas tentang desain sistem penunjang keputusan dengan berbagai model dan alur kerja yang memungkinkan suatu sistem dapat diimplementasikan secara optimal. Berisi diagram *use case*, diagram aktivitas, diagram sekuensi, diagram *View Of Participated Class* (VOPC), dan diagram kelas yang merepresentasikan keseluruhan desain dari aplikasi.

Bab IV membahas Implementasi Aplikasi. Bab ini membahas tentang bagaimana cara mengimplementasikan desain yang telah dibangun menjadi sebuah aplikasi. Berisi implementasi proses dari use case, implementasi kelas-kelas pembangun aplikasi serta implementasi antarmuka aplikasi.

Bab V membahas mengenai Uji Coba dan Evaluasi. Bab ini berisi penjelasan tentang pelaksanaan uji coba terhadap sistem. Berdasarkan uji coba yang dilaksanakan, dilakukan analisis dan evaluasi terhadap hasil yang diperoleh.

Bab VI sebagai Penutup. Berisi simpulan yang dapat diambil dari pelaksanaan Tugas Akhir beserta saran untuk pengembangan selanjutnya

BAB II

DASAR TEORI

BAB II

DASAR TEORI

Tugas Akhir ini bertujuan untuk mendesain dan mengimplementasikan metode ExAnte sebagai metode praproses data dalam sebuah aplikasi penggalian pola asosiasi serta membandingkan kinerja aplikasi dengan dan tanpa menggunakan ExAnte. Oleh karena itu, pada bab ini, selain dijelaskan mengenai metode ExAnte sebagai sebuah metode praproses data, juga dijelaskan mengenai penambangan data secara garis besar, konsep pencarian pola *frequent* dan kaidah asosiasi serta pembangkitan dataset transaksi sintetis yang digunakan dalam Tugas Akhir ini.

2.1. Data Mining

Data mining (penambangan data) merupakan sebuah proses ekstraksi informasi yang potensial, implisit, dan tidak diketahui sebelumnya (misalnya aturan-aturan, batasan-batasan, dan regularitas) dari sekumpulan data pada basis data besar [MSC-96]. Penambangan data muncul sebagai jawaban atas permasalahan pengambilan keputusan yang dihadapi oleh organisasi-organisasi retail. Dengan penambangan data, manajemen perusahaan dapat memperoleh informasi tentang aturan atau pola transaksi yang dilakukan pelanggan sehingga perusahaan dapat menerapkan strategi manajemen yang tepat, misalnya menambah stok barang tertentu, mengatur tata letak barang berdasarkan pola beli konsumen, membuat katalog produk serta melakukan segmentasi pasar.

Ada beberapa teknik penambangan data yang berkembang saat ini, diantaranya: pencarian aturan asosiasi, pencarian pola sekuensial, klasifikasi data dan klasterisasi data. Pada sub-bab selanjutnya akan disajikan uraian mengenai pencarian aturan asosiasi dan metode ExAnte sebagai metode praproses data pada pencarian *frequent pattern* (pola yang sering muncul). Sedangkan teknik yang lain tidak dibahas karena aplikasi yang dibuat khusus untuk keperluan dua topik tersebut.

2.1.1. Pencarian Aturan Asosiasi

Aturan asosiasi mendeskripsikan hubungan asosiasi antar item dalam suatu basis data transaksi dimana jika beberapa item dibeli pada sebuah transaksi, maka item-item lain juga dibeli [AGR-93]. Untuk menemukan aturan asosiasi, pertama harus dicari itemset yang cukup sering terjadi dalam basis data transaksi dimana itemset adalah kumpulan item yang dibeli secara bersamaan. Setelah menemukan itemset tersebut, aturan asosiasi dapat diperoleh sebagai berikut: Jika itemset yang ditemukan adalah Y{Y = I1,I2,...,Ik | k ≥ 2}, maka aturan asosiasi item-item dari himpunan { I1,I2,...,Ik } dapat dicari. Anteseden dari aturan ini adalah X, yang merupakan subset dari Y, dan konsekuennya adalah Y-X. Aturan asosiasi X→ Y-X dimiliki oleh basis data transaksi D dengan *confidence factor* c, jika terdapat paling tidak c% transaksi di D yang mengandung X juga mengandung Y-X. Contoh dari aturan asosiasi tersebut adalah sebagai berikut: 95% dari transaksi dimana kopi dan gula dibeli, susu dibeli juga. Bentuk dari aturan asosiasi ini adalah " kopi, gula → susu ". Anteseden dari aturan ini adalah kopi dan gula sedangkan konsekuennya adalah susu. Persentase sebesar 95% merupakan *confidence factor* dari aturan tersebut.

Sebuah transaksi dikatakan mendukung sebuah itemset Z, jika Z terdapat pada transaksi tersebut [AGR-93]. Nilai support untuk sebuah itemset didefinisikan sebagai rasio dari jumlah transaksi yang memiliki itemset ini terhadap jumlah seluruh transaksi di D. Dengan demikian permasalahan utama dalam pencarian aturan asosiasi adalah untuk menemukan semua itemset yang memenuhi *minimum support* yang ditentukan oleh pengguna. Tiap itemset semacam itu disebut sebagai *frequent itemset*.

2.1.2. Definisi

Diasumsikan bahwa items: {X1, X2, ..., Xn} adalah sebuah set dari anggota yang berbeda, dan sebuah itemset X adalah sebuah subset/himpunan bagian tidak kosong dari items. Apabila k = |X|, maka X bisa disebut juga dengan k-itemset. Sebuah transaksi adalah pasangan (tID,X) dimana TID adalah identitas transaksi

sedangkan X adalah itemset atau isi dari transaksi tersebut. Sebuah itemset X dikatakan terkandung dalam transaksi (TID,Y) apabila $X \subseteq Y$. Diberikan sebuah basis data transaksi TDB himpunan bagian/subset dari transaksi yang mengandung itemset X dinotasikan dengan TDB[X]. Support dari sebuah itemset X, ditulis dengan SuppTDB(X) adalah kardinalitas dari TDB[X]. Apabila diberikan sebuah *minimum support* δ (didefinisikan oleh pengguna), sebuah itemset X dapat dikatakan *frequent* dalam TDB apabila $\text{SuppTDB}(X) \geq \delta$. Aturan asosiasi $X \rightarrow Y\text{-}X$ dimiliki oleh basis data transaksi TDB dengan *confidence factor* c jika terdapat paling tidak c% transaksi di TDB yang mengandung X juga mengandung Y-X.

2.1.3. Penambangan Frequent Pattern

Seperti yang telah dijelaskan sebelumnya, *frequent pattern* adalah kumpulan itemset yang memenuhi *minimum support* yang ditentukan oleh pengguna. Proses penambangan *frequent pattern* telah menjadi topik penelitian yang paling hangat dalam penambangan data akhir-akhir ini. Hal tersebut dapat dilihat dari banyaknya penelitian dan algoritma yang telah dihasilkan.

Pada sub-bab ini dijelaskan mengenai algoritma-algoritma yang dipakai dalam tugas akhir ini. Algoritma yang dimaksud adalah algoritma-algoritma pencarian pola asosiasi yang memiliki pengaruh besar dalam kemajuan teknologi penambangan *frequent pattern*. Tugas akhir ini menggunakan tiga macam algoritma, yaitu Apriori sebagai algoritma dasar dalam penambangan *frequent pattern*, FP-Growth yang merupakan algoritma berbasis *tree* dan terakhir adalah Eclat yang merupakan varian dari Apriori.

2.1.3.1. Algoritma Apriori

Algoritma Apriori merupakan algoritma yang memegang peranan penting dalam penambangan *frequent pattern*. Nama dari algoritma ini didasarkan pada cara kerja algoritma yang menggunakan pengetahuan prior dari properti *frequent* itemset. Apriori melakukan pendekatan secara iteratif yang dikenal dengan pendekatan *level-wise*, dimana k-itemset digunakan untuk menelusuri (k+1)-

itemset [AGR-93]. Pertama, dicari set *frequent* 1-itemset. Set ini dinotasikan dengan L1. L1 digunakan untuk mencari L2, yaitu set *frequent* 2-itemset. Set ini digunakan untuk mencari L3, dan seterusnya, hingga tidak ada lagi *frequent* k-itemset yang dapat ditemukan. Pencarian dari tiap Lk membutuhkan satu kali pemindaian basis data secara utuh. Untuk meningkatkan efisiensi dari pembangkitan level-wise dari *frequent* itemset, digunakan sebuah properti penting yang dinamakan properti apriori. Properti tersebut digunakan untuk mengurangi ruang pencarian yang penjelasannya adalah sebagai berikut:

Apriori property: Untuk semua subset tidak kosong dari frequent itemset, pasti juga akan frequent [AGR-93]. Penjelasannya yakni apabila sebuah itemset I tidak memenuhi batasan minimum support, min_sup, maka I tidak frequent, SuppTDB(I) < min_supp. Apabila sebuah item A ditambahkan ke dalam itemset I, maka I U A juga tidak frequent, SuppTDB(I U A) < min_sup.

Properti ini termasuk ke dalam kategori spesial dari properti yang disebut dengan anti-monotonitas. Yaitu, apabila sebuah set tidak lolos tes, semua supersetnya juga akan gagal. Algoritma Apriori berjalan melalui iterasi dari dua langkah berikut ini:

Join Step:

Untuk menemukan Lk, sebuah set kandidat k-itemset dibangkitkan dari menggabungkan Lk-1 itu sendiri. Set kandidat ini dapat dilambangkan dengan Ck. Diasumsikan bahwa I1 dan I2 adalah itemset pada Lk-1, maka Lk yang dihasilkan dari kedua itemset tersebut adalah hasil pengkombinasian anggota-anggota dari kedua itemset tersebut.

Prune Step:

Ck adalah superset dari Lk, yang mana anggota Lk merupakan atau tidak merupakan *frequent* itemset, sedangkan semua *frequent* k-itemset termasuk didalam Ck. Pemindaian basis data untuk menentukan *count* dari tiap kandidat pada Ck akan menentukan anggota dari Lk (semua kandidat yang memiliki *count* tidak kurang dari *minimum support* adalah *frequent*, sehingga menjadi anggota

dari Lk). Ck, berkemungkinan untuk memiliki ukuran yang sangat besar, sehingga akan mengakibatkan proses komputasi yang berat. Untuk mengurangi ukuran dari Ck, maka digunakan properti apriori sebagai berikut: Semua (k-1)-itemset yang tidak *frequent* tidak mungkin menjadi subset dari *frequent* k-itemset. Oleh karena itu, apabila terdapat (k-1)-subset dari kandidat k-itemset tidak terdapat pada Lk-1, maka kandidat tidak akan mungkin *frequent* sehingga dapat dihilangkan dari Ck.

2.1.3.2. Algoritma FP-Growth

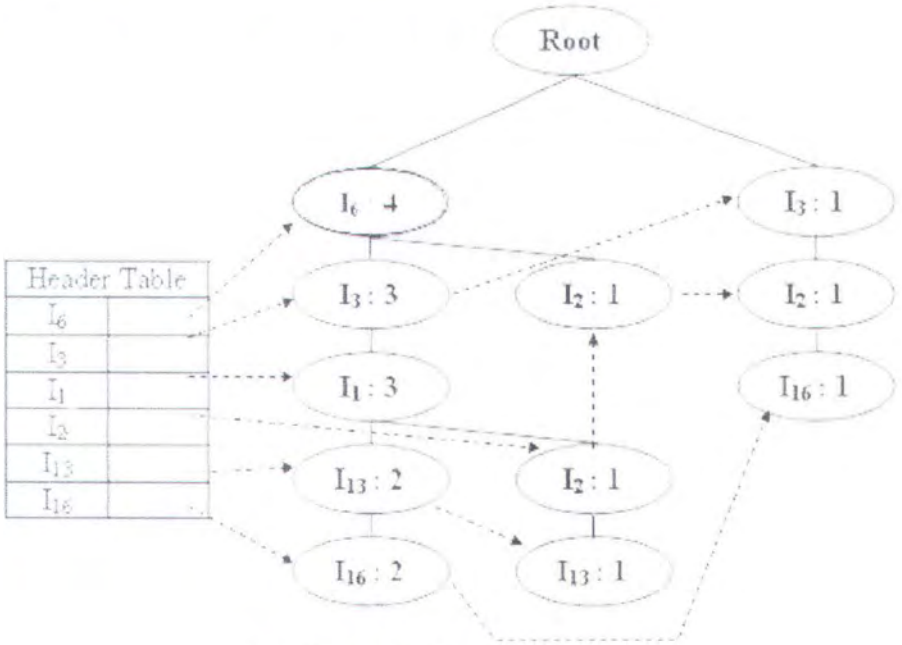
Algoritma FP-Growth menggunakan struktur *frequent pattern* tree (FP-Tree). FP-Tree adalah pengembangan struktur trie dimana tiap set item disimpan sebagai *node* dalam pohon bersama dengan frekuensi kemunculannya [JWH-00]. Pada tiap *node* dalam pohon tersimpan item, *count* dan *next*. Item-item yang terdapat pada jalur dari *root* pohon sampai *node* melambangkan set item tersebut. Penghubung *node* adalah *next*, yang merupakan petunjuk ke *node* berikutnya yang memiliki item yang sama pada FP-Tree. FP-Tree hanya menyimpan item-item yang *frequent* atau memenuhi batasan minimum/*support count*. *Root* dari pohon adalah null, dan item dimasukkan ke dalam pohon dengan mengurutkan item berdasarkan frekuensinya. Tabel 2.1 menunjukkan contoh transaksi beserta item-item *frequent* yang telah diurutkan. Support *count* δ yang diberikan adalah 3.

Tabel 2. 1 Contoh tabel transaksi

Transaksi	Frequent Item Terurut
T1 = {I_5, I_1, I_3, I_4, I_7, I_{12}, I_{13}, I_{16}}	{I_6, I_3, I_1, I_{13}, I_{16}}
T2 = {I_1, I_2, I_3, I_6, I_{12}, I_{13}, I_{15}}	{I_6, I_3, I_1, I_2, I_{13}}
T3 = {I_2, I_6, I_8, I_{10}, I_{15}}	{I_6, I_2}
T4 = {I_2, I_3, I_{11}, I_{19}, I_{16}}	{I_3, I_2, I_{16}}
T5 = {I_1, I_6, I_3, I_5, I_{12}, I_{16}, I_{13}, I_{14}}	{I_6, I_3, I_1, I_{13}, I_{16}}

Gambar 2.1 menunjukkan FP-Tree yang telah dibangun dari contoh transaksi pada tabel 2.1. Dapat dilihat bahwa FP-Tree memiliki informasi lengkap yang dibutuhkan untuk penambangan pola dan dalam bentuk yang ringkas. Yang

dimaksud dengan header table pada sebelah kiri gambar pohon adalah tabel daftar item yang telah diurutkan berdasarkan jumlah kemunculannya.



Gambar 2. 1 Struktur FP-Tree

Kelebihan yang dimiliki oleh algoritma FP-Growth antara lain :

- Kecepatan dalam pencarian frequent item
- Penggunaan yang efektif dengan struktur yang ringkas
- Hanya melalui 2 tahapan penelusuran basis data

FP-Growth mengubah masalah penambangan pola frekuensi dengan panjang k menjadi sebuah urutan dari k-*frequent* dari 1-set item melalui sebuah set *conditional pattern base*. Dengan masukan sebuah struktur FP-Tree, algoritma akan mencari seluruh *frequent pattern* yang ada. *Conditional pattern base* adalah sebuah representasi yang ringkas dari turunan basis data dimana hanya a1 dan lintasan *prefix*nya dalam FP-Tree yang ditampilkan [JWH-00]. Sebagai contoh, lintasan <I6:4, I3:3, I1:3, I13:2, I16:2> dari contoh FP-Tree diatas. Untuk penambangan pola termasuk item I13 dalam lintasan ini yang perlu diperhatikan hanyalah lintasan *prefix* dari I13, sebab *node-node* setelah I13 akan ditambang pada proses yang lain (dalam hal ini hanya I16).

Dengan melihat lintasan awal <I6:4, I3:3, I1:3> , setiap pola yang mengandung I13 akan memiliki frekuensi sama dengan frekuensi I13. Oleh karena itu, frekuensi pada lintasan *prefix* dapat disesuaikan menjadi <I6:2, I3:2, I1:2> yang biasa disebut *transformed prefix path* (lintasan *prefix* yang telah berubah bentuk). Sebuah set *transformed prefix path* dari ai membentuk basis data kecil berisi pola yang berhubungan dengan ai dan memiliki informasi lengkap yang dibutuhkan untuk penambangan pola yang mengandung ai. Oleh karena itu, penambangan *conditional pattern base* untuk semua ai secara rekursif, adalah ekuivalen dengan penambangan seluruh tree.

2.1.3.3. Algoritma Eclat

Algoritma Eclat merupakan algoritma pencarian pola dengan prinsip kerja yang hampir sama dengan Apriori [MZK-97]. Kedua algoritma ini, menggunakan pengurutan penuh pada kumpulan item P(A){A1, A2, A3, ...,An}. Item-item pada A diurutkan dari kecil ke besar dan berkorespondensi satu-satu dengan pemetaan O. Pemetaan ini (*item coding*), dilakukan untuk mendapatkan sebuah himpunan bilangan integer yang berurutan. Jika itemset X, Y merupakan elemen atau bagian dari P(A), maka *prefix* dideskripsikan dengan :

$$prefix(X,Y) := \{\{x \in X \mid x \le z\} \mid maximal\ z \in X \cap Y : \{x \in X \mid x \le z\} = \{y \in Y \mid y \le z\}\} \quad \ldots(2.1)$$

Dari persamaan 2.1 diatas, nilai dari *prefix* (X,Y) ditentukan dari anggota dari X dan Y. Dimana jumlah anggota yang sama-sama dimiliki oleh kedua himpunan tersebut, harus kurang dari atau sama dengan maximal z. Yaitu, irisan kedua himpunan.

Eclat dimulai dengan *prefix* kosong, dan matrik item-transaksi *incidence* CØ yang untuk selanjutnya disebut dengan *incidence matrix* dan disimpan sebagai daftar item yang mencakup $C_\emptyset := \{(x, T(\{x\})) \mid x \in A\}$. *Incidence matrix* ini kemudian disaring dengan syarat seperti yang ditunjukkan pada persamaan 2.2, sehingga hanya mengandung *frequent* item saja.

$$freq(C) := \{(x, T_x) \mid (x, T_x) \in C, |T_x| \geq \min \sup\} \quad \ldots(2.2)$$

Persamaan diatas, merepresentasikan *frequent* 1-item-extension pada *prefix*. Untuk sembarang *prefix* $p \in P(A)$ serta *incidence matrix* C dari *frequent* 1-item-extension dari p, dapat dihitung *incidence matrix* Cx terhadap 1-item-extension dari p gabungan dengan {x} dengan menyilangkan baris seperti ditunjukkan dalam persamaan 2.3 di bawah ini :

$$C_x := \{(y, T_x \cap T_y) \mid (y, T_y) \in C, y > x\} \quad \ldots(2.3)$$

Pada persamaan di atas, $(x, T_x) \in C$ adalah baris yang merepresentasikan $p \cup \{x\}$. Cx kemudian disaring untuk mendapatkan semua 1-item-extension yang *frequent* dari $p \cup \{x\}$. Prosedur ini diulang terus hingga hasil dari *incidence matrix* Cx adalah kosong, yang mengindikasikan bahwa tidak ada lagi 1-item-extension yang *frequent* pada *prefix*.

Seperti yang ditunjukkan pada pseudocode algoritma pada gambar 2.2 dibawah ini, algoritma dimulai dari pembuatan *incidence matrix* $C_O := \{(x, T(\{x\})) \mid x \in A\}$ yang beranggotakan item-item pada A. Setelah itu matriks ini disaring berdasarkan frekuensinya seperti pada persamaan 2.2. Proses ini menghasilkan matriks yang hanya berisi item-item *frequent* saja. Proses selanjutnya adalah memasukkan matrik awal ini kedalam sebuah iterasi dimana tiap baris pada Cx disilangkan dengan baris yang lainnya. Apabila jumlah kolom dari baris yang dihasilkan memenuhi syarat frekuensi diatas, maka baris hasil penyilangan ini akan menjadi anggota dari matriks hasil. Apabila matriks yang dihasilkan dari proses ini masih memiliki anggota atau tidak null, maka iterasi akan dimulai kembali, sampai matriks hasil tidak memiliki anggota.

input : alphabet A with ordering $\leq$,
multiset $T \subseteq P(A)$ of sets of items,
minimum support value minsup $\in N$.
output : set F of frequent itemsets and their support counts.
$F := \{(\emptyset, |T|)\}$.
$C_\emptyset := \{(x, T(\{x\})) \mid x \in A\}$.
$C'_\emptyset := freq(C_\emptyset) := \{(x, T_x) \mid (x, T_x) \in C_\emptyset, |T_x| \geq \min\sup\}$.
$F := \{\emptyset\}$.
addFrequentSupersets$(\emptyset, C'_\emptyset)$
function addFrequentSupersets
input : frequent itemset $p \in P(A)$ called prefix,
incidence matrix C of frequent 1 - item - extensions of p.
output : add all frequent extensions of p to global variable F.
for $(x, T_x) \in C$ do
$q := p \cup \{x\}$.
$C_q := \{(y, T_x \cap T_y) \mid (y, T_y) \in C,\ y > x\}$
$C'_q := freq(C_q) := \{(y, T_y) \mid (y, T_y) \in C_q, |T_y| \geq \min\sup\}$
if $C'_q \neq \emptyset$ then
addFrequentSupersets(q, C_q)
end if
$F = F \cup \{(q, |T_x|)\}$.
end for

Gambar 2. 2 Pseudocode Algoritma Eclat

Perbedaan yang paling menonjol antara Eclat dan Apriori terletak pada bagaimana cara penelusuran *prefix* tree, serta penentuan *support* dari itemset, misalnya jumlah transaksi dimana itemset tersebut berada [MZK-97]. Apriori menentukan *support* dari itemset baik dengan melakukan pengecekan setiap kandidat itemset dimana transaksi itu berada, ataupun dengan menelusuri semua subset transaksi yang telah dihitung ukurannya, kemudian menambahkan itemset yang terkait ke dalam *counter*.

Eclat melakukan penelusuran *prefix* tree dengan urutan *depth first*, yang memperluas *prefix* dari itemset hingga mencapai batas antara *frequent* dan *infrequent itemset* lalu kembali menelusuri *prefix* selanjutnya (backtrack) dengan urutan *lexiographic* [MZK-97]. Eclat menentukan *support* dari sembarang itemset

dengan jalan membentuk daftar pengenal transaksi yang mengandung itemset dengan jalan menyilangkan (*intersect*) dua daftar pengenal transaksi dari dua itemset yang dibedakan oleh satu item, dan kemudian secara bersama-sama memproses itemset tersebut.

2.1.3.4. Pembangkitan Data Sintetis

Pembangkitan data transaksi sintetis dilakukan untuk mengevaluasi kinerja dari algoritma pada berbagai macam karakteristik data. Transaksi-transaksi ini dibuat menyerupai bentuk transaksi pada lingkungan perusahaan retail. Model dari sebuah transaksi merupakan gambaran dari dunia nyata ketika seorang pembeli melakukan pembelian sekumpulan item secara bersamaan. Tiap set dari transaksi-transaksi tersebut berpotensi menjadi *maximal large itemset* atau itemset besar maksimal [AGR-93]. Contoh dari bentuk itemset ini adalah set yang mengandung kopi, susu dan gula. Akan tetapi, beberapa orang bisa saja hanya membeli beberapa jenis item dari set tersebut. Sebagai contoh, beberapa orang membeli kopi dan gula, dan beberapa mungkin hanya membeli gula saja. Sebuah transaksi dimungkinkan untuk mengandung lebih dari satu *large itemset*. Sebagai contoh, seorang pembeli mungkin memesan sabun dan pasta gigi ketika memesan kopi dan gula, dimana sabun dan pasta gigi membentuk *large itemset* yang lain. Ukuran transaksi biasanya berkisar pada *mean* atau rataan dan sedikit transaksi yang memiliki banyak item. Begitu pula dengan *large itemset* yang juga berkisar pada *mean* atau rataan dengan sedikit *large itemset* yang memiliki jumlah item banyak. Untuk menciptakan sebuah dataset, aplikasi pembangkitan data sintetis memiliki parameter seperti yang ditunjukkan pada tabel 2.2

Tabel 2. 2 Parameter pembangkitan data sintetis

D	Jumlah transaksi
I	Ukuran rata-rata large itemset potensial
MI	Ukuran maksimum large itemset potensial
L	Jumlah large itemset
T	Ukuran rata-rata transaksi
MT	Ukuran maksimum transaksi
N	Jumlah item

Sedangkan metode pembuatan transaksi tiruan ini adalah sebagai berikut: pertama–tama dibuat himpunan *large itemset* potensial L, dan kemudian dipilih sebuah *large itemset* dari L untuk ditempatkan pada sebuah transaksi [ADT-02]. Ukuran tiap–tiap *large itemset* potensial berkisar antara 1 dan |MI| dimana peluang munculnya *large itemset* dengan ukuran 1, 2, ..., dan |MI| diperoleh dari distribusi Poisson dengan mean sama dengan |I|. Peluang–peluang tersebut kemudian dinormalkan sehingga jumlah peluang–peluang tersebut sama dengan 1. Sebagai contoh, misalkan ukuran rata–rata |I| dari large itemset adalah 3 dan ukuran maksimum |MI| dari *large itemset* adalah 5, maka menurut distribusi Poisson dengan mean |I|, peluang untuk ukuran 1, 2, 3, 4, dan 5 adalah 0.17, 0.26, 0.26, 0.19 dan 0.12. Setelah proses normalisasi, peluang–peluang tersebut kemudian diakumulasi sehingga tiap ukuran berada pada suatu interval tertentu seperti yang ditunjukkan pada tabel 2.3. Untuk tiap *large itemset* potensial, dihasilkan sebuah angka random real antara 0 dan 1 untuk menentukan ukuran *large itemset* potensial.

Tabel 2. 3 Probabilitas Ukuran-ukuran Itemset

Size	Range
1	0 ~ 0,17
2	0,18 ~ 0,43
3	0,44 ~ 0,69
4	0,70 ~ 0,88
5	0,89 ~ 1

Selanjutnya jumlah *large itemset* potensial di L diset sejumlah |L|. Item–item pada *large itemset* yang pertama dipilih secara random. Beberapa item pada *large itemset* selanjutnya dipilih dari *large itemset* yang telah dibuat sebelumnya. Cara pemilihan item-item tersebut adalah sebagai berikut: untuk tiap item pada *large itemset* sebelumnya, dilemparkan sebuah mata uang untuk menentukan apakah item tersebut tetap dipakai pada *large itemset* yang sedang dibuat atau tidak. Jika jumlah item pada *large item* yang sedang dibuat belum memenuhi ukuran *large itemset* itu, maka sisa item pada *large itemset* tersebut diambil secara random dari kumpulan item. Setelah himpunan *large itemset* L selesai dibuat,

lagkah selanjutnya adalah membuat transaksi pada basisdata. Ukuran masing–masing transaksi diambil dari distribusi Poisson dengan *mean* sama dengan |T| dimana ukurannya berkisar antara 1 dan |MT|. Metode untuk menentukan ukuran transaksi sama dengan metode untuk menentukan ukuran sebuah *large itemset*. Untuk sebuah transaksi, dipilih secara random sebuah *large itemset* dari L dengan ukuran yang tidak melebihi ukuran transaksi tersebut dan menempatkannya pada transaksi tersebut. Sisa item pada transaksi yang pertama dipilih secara random. Beberapa item pada transaksi berikutnya diambil dari transaksi yang telah dibuat sebelumnya. Untuk tiap item pada transaksi sebelumnya, dilemparkan sebuah koin untuk menentukan apakah item tersebut tetap ada pada transaksi tersebut atau tidak. Setelah mengambil item–item dari sebuah *large itemset* dan dari transaksi sebelumnya, sisa item pada transaksi tersebut ditentukan secara random. Sedangkan untuk nilai *price* yang akan dipakai dalam batasan monoton, ditentukan dengan menggunakan distribusi uniform [FBC-03B].

2.2. Praproses Data

Diantara permasalahan penambangan data, penambangan pola asosiasi telah menjadi topik pembahasan yang paling populer. Bersamaan dengan popularitas tersebut, maka kebutuhan akan aplikasi penambangan pola asosiasi yang lebih interaktif semakin berkembang. Dengan tujuan untuk memenuhi kebutuhan tersebut, maka tugas akhir ini mengangkat *Dimension Reduction Method* (DRM) atau metode pengurangan dimensi, yang mana merupakan sebuah metode praproses untuk meningkatkan interaksi penambangan pola asosiasi dengan batasan item.

Teknik penambangan pola asosiasi yang menggunakan batasan telah terbukti efektif dalam mengurangi ruang pencarian pada penambangan pola, dan juga meningkatkan efisiensi. Jenis batasan yang paling banyak dipelajari adalah batasan frekuensi, dimana antimonotonitas digunakan untuk memperkecil ruang pencarian secara eksponensial dari masalah. Eksploitasi antimonotonitas dengan batasan frekuensi dikenal dengan sebutan trik Apriori [AGR-93]. Hal tersebut secara dramatis dapat memperkecil ruang pencarian sehingga membuat proses

komputasi pencarian *frequent* itemset menjadi mungkin atau *feasible*. Frekuensi tidak hanya efektif secara proses komputasional, namun juga penting secara semantik karena frekuensi menyediakan support bagi berbagai pencarian informasi. Untuk alasan tersebut, frekuensi merupakan batasan dasar dari apa yang biasa disebut sebagai *frequent* itemset mining.

Namun, banyak batasan lain yang dapat memfasilitasi eksplorasi serta kontrol dengan pemfokusan pada pengguna, yang juga bertujuan untuk mengurangi proses komputasi. Sebagai contoh, pengguna mungkin saja tertarik pada penambangan semua itemset yang *frequent* serta dengan total harga yang lebih besar dari nilai yang diberikan oleh pengguna, atau bisa juga dicari itemset yang mengandung paling tidak dua produk yang diinginkan oleh pengguna.

Dalam penambangan pola asosiasi, penerapan batasan monoton bersama frekuensi bertujuan untuk mengurangi data masukan serta ruang pencarian. Oleh karena itu, dalam tugas akhir ini penulis mengangkat tentang metode ExAnte, sebuah metode praproses data yang memiliki kemampuan dalam menerapkan keselarasan antara batasan antimonoton dengan monoton. Dengan adanya penyelarasan dari kedua batasan tersebut, maka keuntungan yang didapatkan bisa lebih besar dari jumlah keuntungan antara keduanya secara individu.

2.2.1. Batasan Antimonoton

Batasan antimonoton merupakan sifat dari penambangan pola dimana apabila nilai *support* dari sebuah itemset berada dibawah nilai yang telah ditentukan oleh pengguna, maka subset dari itemset tersebut juga tidak akan memenuhi nilai *support* yang diinginkan [FBC-05]. Antimomoton berguna untuk memperkecil ruang pencarian yang ada. Sebab itemset yang tidak memenuhi syarat pada level awal (pola kecil), maka untuk seterusnya juga tidak akan memenuhi (pola yang lebih besar).

Batasan antimonoton adalah batasan yang paling efektif dan mudah untuk digunakan dalam mengurangi ruang pencarian. Oleh karena, setiap turunan dari batasan antimonoton pasti juga akan berupa antimonoton pula. Dengan begitu, trik

apriori yang berdasarkan antimonotonitas dapat digunakan untuk mengeksploitasi pengurangan secara lengkap pada turunan tersebut. Semakin banyak batasan antimonoton yang diberikan, maka trik apriori akan semakin selektif.

Berikut ini adalah definisi dari batasan frekuensi C_{freq}[TDB]: Apabila itemset X adalah *frequent*, dapat dinotasikan dengan C_{freq}[TDB](X) atau secara lebih sederhana menjadi C_{freq}(X). Jika diberikan sebuah itemset X dengan batasan C_{Am}. Apabila C_{Am} memenuhi pada X maka akan memenuhi pada semua subset dari X. Batasan frekuensi adalah batasan antimonoton, keadaan ini digunakan oleh algoritma Apriori dengan penjelasan sebagai berikut :

Apabila itemset X tidak memenuhi C_{freq}, maka tidak ada superset dari X yang dapat memenuhi C_{freq}, oleh karena itu X dapat dihilangkan dari ruang pencarian. Penghilangan ini dapat mempengaruhi sebagian besar dari ruang pencarian. Oleh karena itu, algoritma Apriori beroperasi pada tingkatan per level dan bergerak secara *bottom up*. Tiap kali ditemui itemset yang tidak *frequent* maka seluruh supersetnya dapat dihilangkan.

2.2.2. Batasan Monoton

Berbeda dengan batasan antimonoton, batasan monoton bergerak dari tingkatan itemset lengkap atau satu transaksi. Yang dikatakan dengan batasan monoton adalah apabila sebuah itemset tidak memenuhi batasan monoton yang diberikan oleh pengguna maka turunan dari itemset tersebut juga tidak akan memenuhi batasan monoton. Sebagai contoh, diberikan sebuah batasan monoton minSum (jumlah minimal total harga) dari itemset sebesar $40. Apabila terdapat itemset yang tidak memenuhi batasan tersebut, maka itemset tersebut tidak akan berguna dalam proses selanjutnya. Hal ini disebabkan karena kemungkinan yang dapat terjadi pada itemset tersebut hanyalah jumlah anggotanya berkurang atau tetap. Atau dengan kata lain total harga yang dimiliki tidak akan pernah bertambah dan memenuhi batasan monoton yang diberikan. Contoh batasan monoton ditunjukkan pada tabel 2.4 dibawah ini.

Tabel 2. 4 Contoh batasan monoton

Monotone constraints	Formula
Cardinality	$\|X\| \geq n$
Price sum	$sum(X\ price) \geq n$
Maximum price	$max(X\ price) \geq n$
Minimum price	$min(X\ price) \leq n$
Price range	$Range(X\ price) \geq n$

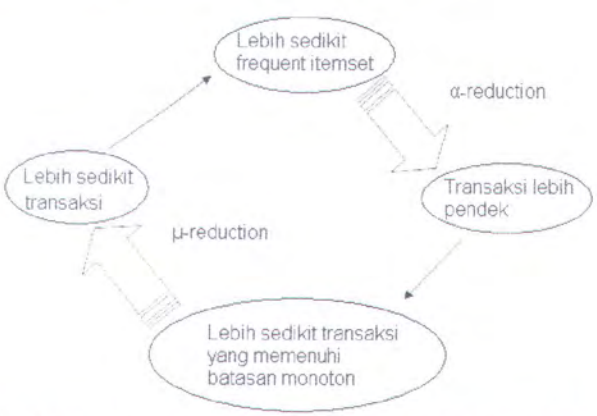
Pada tabel 2.4, kelima batasan tersebut adalah monoton, sebab apabila sebuah itemset tidak memenuhi batasan monoton di atas maka turunan dari itemset tersebut juga tidak akan memenuhi. Diberikan sebuah itemset X serta batasan C_m adalah monoton. Apabila C_m dipenuhi oleh X, maka akan dipenuhi oleh segala superset dari X, dengan catatan bahwa batasan monoton dibuat terpisah dari masukan basis data transaksi yang diberikan. Hal ini dibutuhkan karena batasan monoton sederhana dengan batasan monoton global seperti batasan *nonfrequent* ($SuppTDB(X) \leq \delta$) perlu dipisahkan. Batasan ini masih merupakan batasan monoton namun memiliki properti yang berbeda karena bergantung pada dataset dan membutuhkan pemindaian dataset secara menyeluruh dalam proses komputasi.

2.2.3. Algoritma ExAnte

ExAnte bekerja dengan tujuan menyinergikan kedua batasan yang telah dijelaskan sebelumnya, yaitu batasan monoton dan antimonoton. Batasan monoton diterapkan pada basis data masukan, karena sebuah transaksi juga merupakan sebuah itemset. Transaksi yang tidak memenuhi batasan monoton dapat dibuang. Sebab, transaksi tersebut tidak akan ikut dalam solusi support *count*. Transaksi tersebut tidak akan berguna karena tidak akan mengandung pola yang menarik. Oleh karena itu, transaksi tidak akan menambah support *count* itemset yang dapat memecahkan masalah yang diberikan. Penghapusan transaksi-transaksi ini tidak akan mengurangi nilai *support* dari pola-pola yang menarik. Hal ini disebut juga pengurangan transaksi berdasarkan monoton *μ-reduction* [FBC-03A].

Pengurangan basis data masukan seperti diatas dapat mengakibatkan menurunnya *support* dari itemset yang gagal memenuhi batasan monoton. Dengan demikian, itemset yang tidak memenuhi syarat frekuensi (antimonoton) akan semakin banyak. Batasan monoton diatas dapat memperkuat pengurangan berdasarkan antimonoton. Yaitu penghapusan items-items yang tidak memenuhi syarat frekuensi. Hal ini disebut dengan *α-reduction* [FBC-03A].

Penghilangan item-item dari transaksi di atas, mengakibatkan berkurangnya ukuran dari transaksi yang memenuhi batasan monoton. Hal tersebut mengakibatkan transaksi tersebut memiliki kemungkinan untuk tidak lagi memenuhi batasan monoton [FBC-05]. Sebagai contoh, batasan monoton berdasarkan total harga minimum, maka sebuah transaksi yang awalnya memenuhi batasan (total harga diatas syarat yang diberikan) mungkin tidak lagi memenuhi batasan ketika item *nonfrequent* dihilangkan. Sehingga terjadi peningkatan jumlah transaksi yang gagal memenuhi batasan monoton. Dengan kata lain terjadi loop antara dua buah parameter pengurangan data (α dan μ) yang bekerja bergantian untuk mengurangi ruang pencarian dan set data masukan. Kedua parameter tersebut akan saling menguatkan satu sama lain sampai tidak ada lagi kemungkinan pengurangan yang dapat dilakukan. Secara keseluruhan, jalannya metode ExAnte dapat dilihat pada gambar 2.3.



Gambar 2. 3 Lingkaran pengurangan ruang pencarian dan data masukan

Algoritma ExAnte diatas dapat menggunakan pseudocode seperti yang ditunjukkan pada gambar 2.4 dibawah ini. Proses TDB := $\alpha[TDB]C_{freq}$ berfungsi untuk menerapkan batasan frekuensi pada basis data dan proses TDB :=

$\mu[TDB]C_M$ menerapkan batasan monoton terhadap basis data. Hal tersebut terus berlanjut selama masih ada item-item yang dapat dihilangkan dari basis data (| I | < old_number_interesting_items).

```
Prosedur : ExAnte(TDB,C_M, minSupp)
1. I := 0;
2. forall itemset t pada TDB do
3.       if C_M(t) then forall items I in t do
4.                i.count++; if i.count ≥ minSupp then I :=I U i;
5. old_number_interesting_items := | Items |;
6. while | I | < old_number_interesting_items do
7.       TDB := α[TDB]C_freq;
8.       TDB := μ[TDB]C_M;
9.       old_number_interesting_items := | I |;
10.      I := 0
11.      forall itemset t pada TDB do
12.               forall items i pada t do
13.                        i.count++;
14.                        if i.count ≥ minSupp then I := I U i;
15. end while
```

Gambar 2. 4 Pseudocode algritma ExAnte

Metode seperti di atas akan mengakibatkan terjadinya reduksi jumlah besar pada basis data transaksi yang akan melalui proses pencarian pola. Atau bahkan mungkin dataset tersebut tidak perlu lagi melalui proses pencarian pola karena adanya kemungkinan bahwa itemset yang tersisa merupakan pola-pola yang dicari.

BAB III

DESAIN APLIKASI

BAB III

DESAIN APLIKASI

Bab ini menjelaskan bagaimana desain aplikasi dilakukan agar sesuai dengan tujuan sistem. Tujuan utama dari aplikasi ini adalah mempermudah pengguna dalam melakukan penambangan pola, serta memberikan informasi yang jelas tentang kinerja aplikasi dengan berbagai variasi parameter masukan dari pengguna.

Metode *ExAnte* merupakan metode praproses data. Metode ini bekerja sebelum proses penambangan pola dilakukan. Oleh karena itu, aplikasi ini didesain selain untuk melakukan proses penambangan pola secara normal (tanpa praproses), juga memiliki kemampuan untuk memproses data masukan terlebih dahulu sebelum melalui proses penambangan.

Aplikasi ini dibuat dengan menggunakan sistem UML (*Unified Modelling Language*). UML merupakan sebuah bahasa pemodelan, yang dibuat dengan notasi yang spesifik dalam pembangunan sebuah model dari aplikasi atau software. UML mendukung berbagai macam elemen notasi grafik, yang menyediakan keuntungan besar bagi pengembang aplikasi. Yakni, dengan membantu membangun model yang bervariasi, dapat ditelusuri serta dapat dikelola. Dengan kata lain, UML mendukung daur hidup pengembangan aplikasi.

Untuk membuat suatu model, UML memiliki diagram grafis sebagai berikut

- Use Case diagram
- Class diagram
- Behaviour diagram :
    - Statechart diagram
    - Activity diagram
- Interaction diagram :

- Sequence diagram
- Collaboration diagram

➢ Implementation diagram :

- Component diagram
- Deployment diagram

Dari daftar diatas, beberapa diagram yang digunakan dalam desain proses ini adalah: diagram *use case*, diagram sekuensi, diagram aktivitas serta diagram kelas. Untuk penjelasan mengenai pengertian diagram-diagram ini serta penggunaanya dalam desain aplikasi akan dibahas pada sub bab berikut:

3.1. Analisis Use Case

Analisis *use case* menggambarkan sisi fungsionalitas dari sistem yang baru. *Use case* merepresentasikan interaksi yang dapat terjadi antara user (pengguna atau mesin lain) dengan sistem. Sebuah *use case* adalah satu unit kerja yang memiliki pengaruh pada sistem. Contoh dari *use case* sederhana antara lain, menciptakan sebuah kereta, memodifikasi sebuah kereta atau membuat pemesanan.

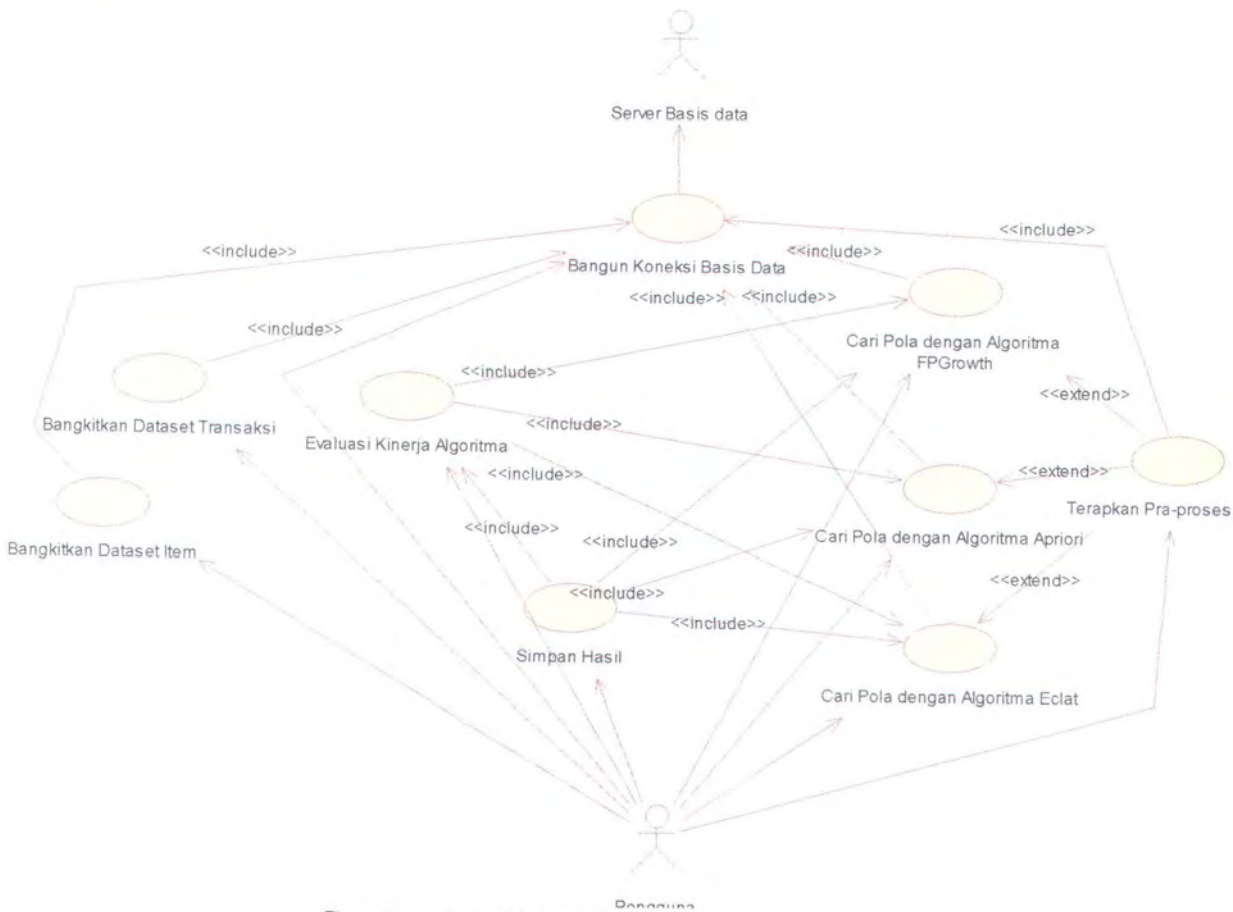
Tiap-tiap *use case* memiliki sebuah deskripsi yang menjelaskan fungsi yang akan dibangun pada sistem yang baru. Sebuah *use case* dapat berupa perluasan (*extend*) dan mengandung (*include*) fungsi dari *use case* yang lain dengan properti mereka sendiri. *Use case* pada dasarnya berhubungan dengan "aktor". Aktor adalah pengguna atau mesin yang berinteraksi dengan sistem dalam melakukan kerja.

Beberapa *use case* yang dimiliki oleh aplikasi antara lain :

- Bangun Koneksi Basis Data
- Bangkitkan Dataset Transaksi
- Bangkitkan Dataset Item

- Terapkan Praproses *ExAnte*
- Cari Pola dengan Algoritma Apriori
- Cari Pola dengan Algoritma Fpgrowth
- Cari Pola dengan Algoritma Eclat
- Evaluasi Kinerja Algoritma
- Simpan Hasil

Use case dari aplikasi ini dapat digambarkan pada diagram *use case* pada gambar 3.1 di bawah ini:



Gambar 3. 1 Diagram Use Case

Deskripsi dari masing – masing *use case* diatas dapat dijelaskan sebagai berikut

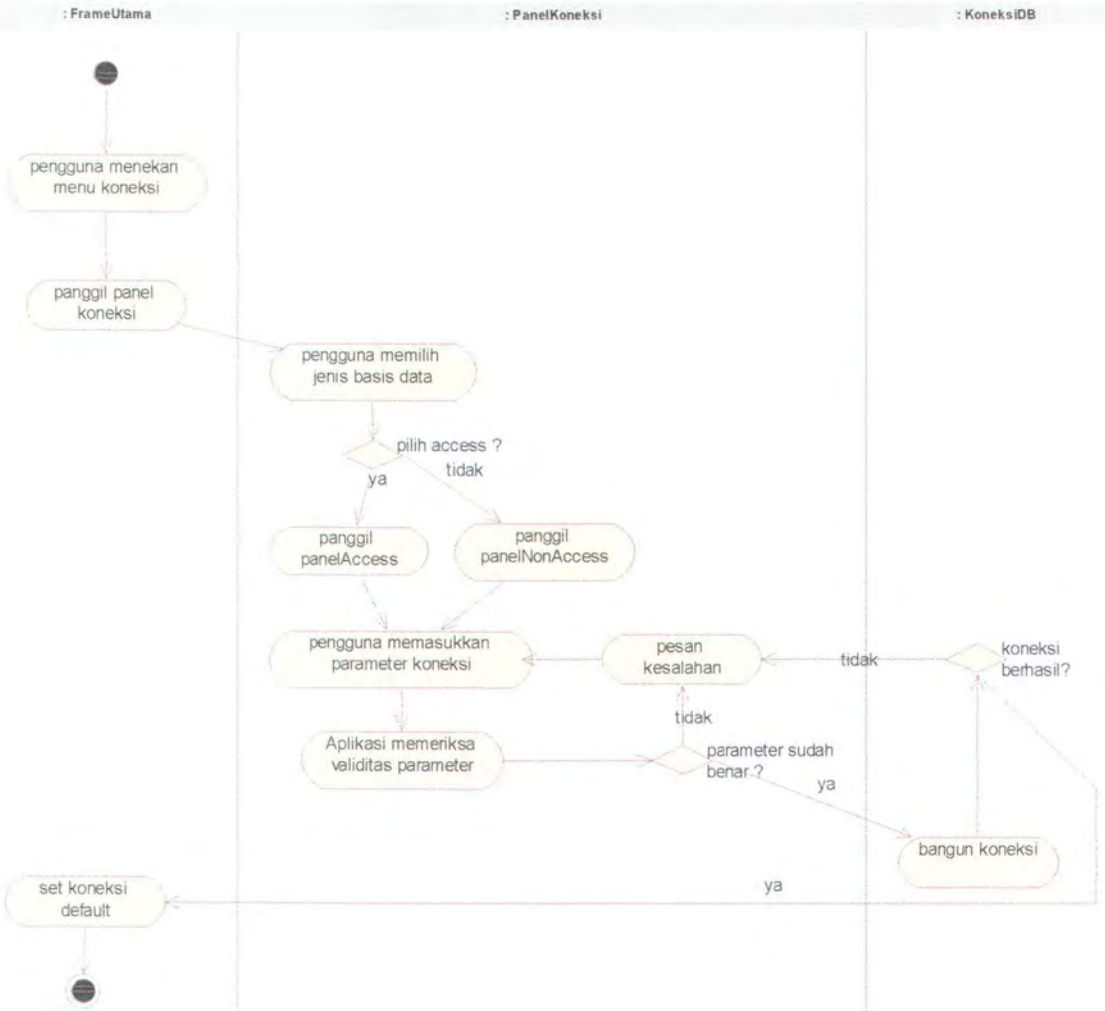
3.1.2. Bangun Koneksi Basis Data

Use case ini menggambarkan proses pembangunan hubungan antara aplikasi dengan server basis data. Spesifikasi *use case* ini dapat dilihat pada tabel 3.1.

Tabel 3. 1 Spesifikasi Use Case "Bangun Koneksi Basis Data"

Spesifikasi	Penjelasan
Aktor	➢ Aktor ➢ Server Basis Data
Tujuan	➢ Membangun koneksi dengan server basis data untuk digunakan dalam proses aplikasi yang lain
Ringkasan	➢ Use case ini pembangunan koneksi dengan server basis data berdasarkan masukan dari pengguna
Masukan	➢ Jenis basis data ➢ Nama basis data ➢ Alamat server ➢ Port yang digunakan server ➢ Username ➢ Password
Luaran	➢ Koneksi
Kondisi Awal	➢ Aktor berada pada tampilan awal, tidak ada koneksi ➢ Koneksi telah terbangun, Aktor ingin mengubah koneksi
Kondisi Akhir	➢ Perubahan status koneksi pada FrameUtama
Aliran Aksi normal	➢ Pengguna menekan menu "Koneksi" ➢ Pengguna memilih jenis basis data ➢ Pengguna memasukkan parameter nama basis data, host, port, username dan password. ➢ Pengguna menekan tombol "Connect" untuk membangun koneksi ➢ Aplikasi memeriksa validitas masukan pengguna ➢ Aplikasi membangun koneksi yang diinginkan ➢ Aplikasi mengubah status default koneksi pada frame utama

Seluruh aktivitas dalam *use case* ini dapat disederhanakan dalam bentuk diagram aktivitas pada gambar 3.2.



Gambar 3. 2 Diagram Aktivitas untuk Use Case "Bangun Koneksi Basis Data"

3.1.3. Bangkitkan Dataset Transaksi

Use case ini menggambarkan proses pembangkitan dataset transaksi sintetis sesuai dengan parameter yang diinginkan oleh pengguna. Spesifikasi *use case* ini dapat dilihat pada tabel 3.2.

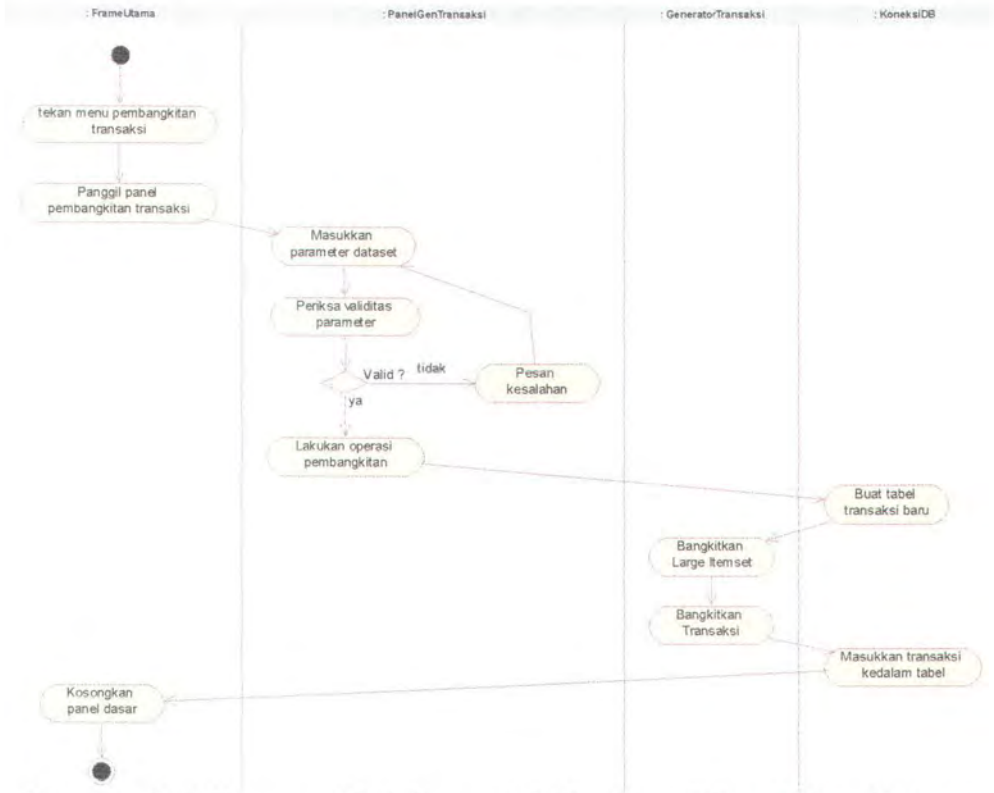
Tabel 3. 2 Spesifikasi Use Case "Bangkitkan Dataset Transaksi"

Spesifikasi	Penjelasan
Aktor	➢ Pengguna
Tujuan	➢ Membangkitkan dataset transaksi sintetis
Ringkasan	➢ Aktor memasukkan parameter dataset yang menunjukkan karakteristik dataset, aplikasi melakukan pembangkitan data

Tabel 3.2 Spesifikasi Use Case "Bangkitkan Dataset Transaksi" (lanjutan)

Spesifikasi	Penjelasan
Luaran	➢ Tabel transaksi pada server basis data
Kondisi Awal	➢ Aktor berada pada tampilan panel pembangkitan transaksi
Kondisi Akhir	➢ Tabel transaksi berhasil dibuat
Aliran Aksi normal	➢ Pengguna memilih menu "Pembangkitan Transaksi" ➢ Aplikasi memanggil panel pembangkitan transaksi ➢ Pengguna memasukkan parameter yang dibutuhkan ➢ Pengguna memasukkan nama tabel yang akan digunakan ➢ Pengguna menekan tombol "Generate" ➢ Aplikasi memeriksa validitas parameter ➢ Aplikasi membangkitkan dataset ➢ Aplikasi mengirim hasil pembangkitan ke server basis data

Seluruh aktivitas dalam *use case* ini dapat disederhanakan dalam bentuk diagram aktivitas pada gambar 3.3.



Gambar 3. 3 Diagram Aktivitas untuk Use Case "Bangkitkan Dataset Transaksi"

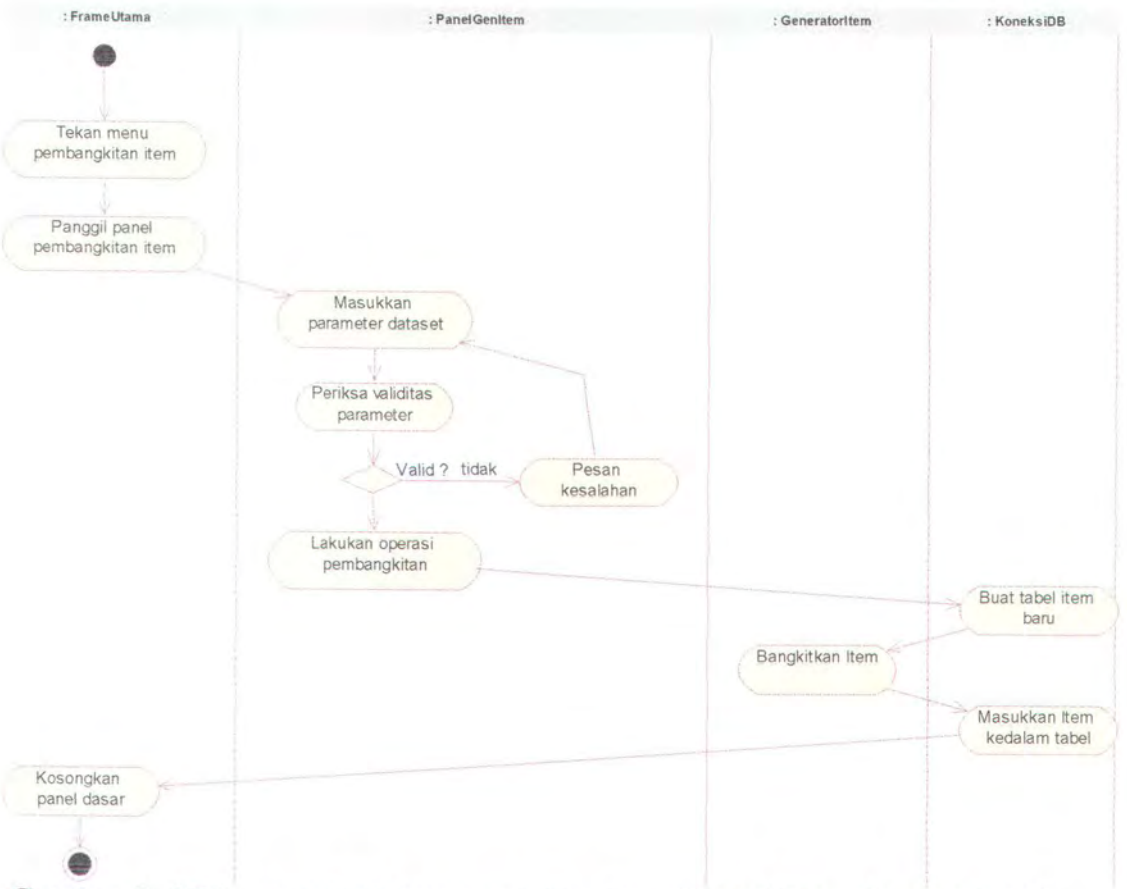
3.1.4. Bangkitkan Dataset Item

Use case ini menggambarkan proses pembangkitan dataset sintetis untuk item beserta nilai *price* per item sesuai dengan parameter yang diinginkan oleh pengguna. Spesifikasi *use case* ini dapat dilihat pada tabel 3.3.

Tabel 3. 3 Spesifikasi Use Case "Bangkitkan Dataset Item"

Spesifikasi	Penjelasan
Aktor	➢ Pengguna
Tujuan	➢ Membangkitkan dataset item sintetis
Ringkasan	➢ Aktor memasukkan parameter dataset yang menunjukkan karakteristik dataset, aplikasi melakukan pembangkitan data
Masukan	➢ Jumlah Item (NP) ➢ Minimum Price (MIP) ➢ Maximum Price (MAP) ➢ Nama tabel item
Luaran	➢ Tabel item pada server basis data
Kondisi Awal	➢ Aktor berada pada tampilan panel pembangkitan item
Kondisi Akhir	➢ Tabel item berhasil dibuat
Aliran Aksi normal	➢ Pengguna memilih menu "Pembangkitan Item" ➢ Aplikasi memanggil panel pembangkitan item ➢ Pengguna memasukkan parameter yang dibutuhkan ➢ Pengguna memasukkan nama tabel yang akan digunakan ➢ Pengguna menekan tombol "Generate" ➢ Aplikasi memeriksa validitas parameter ➢ Aplikasi membangkitkan dataset ➢ Aplikasi mengirim hasil pembangkitan ke server basis data

Seluruh aktivitas dalam *use case* ini dapat disederhanakan dalam bentuk diagram aktivitas pada gambar 3.4.



Gambar 3. 4 Diagram Aktivitas untuk Use Case "Bangkitkan Dataset Item"

3.1.5. Terapkan Praproses *ExAnte*

Use case ini menggambarkan proses reduksi dataset sebelum dilakukan penambangan pola. Spesifikasi *use case* ini dapat dilihat pada tabel berikut ini :

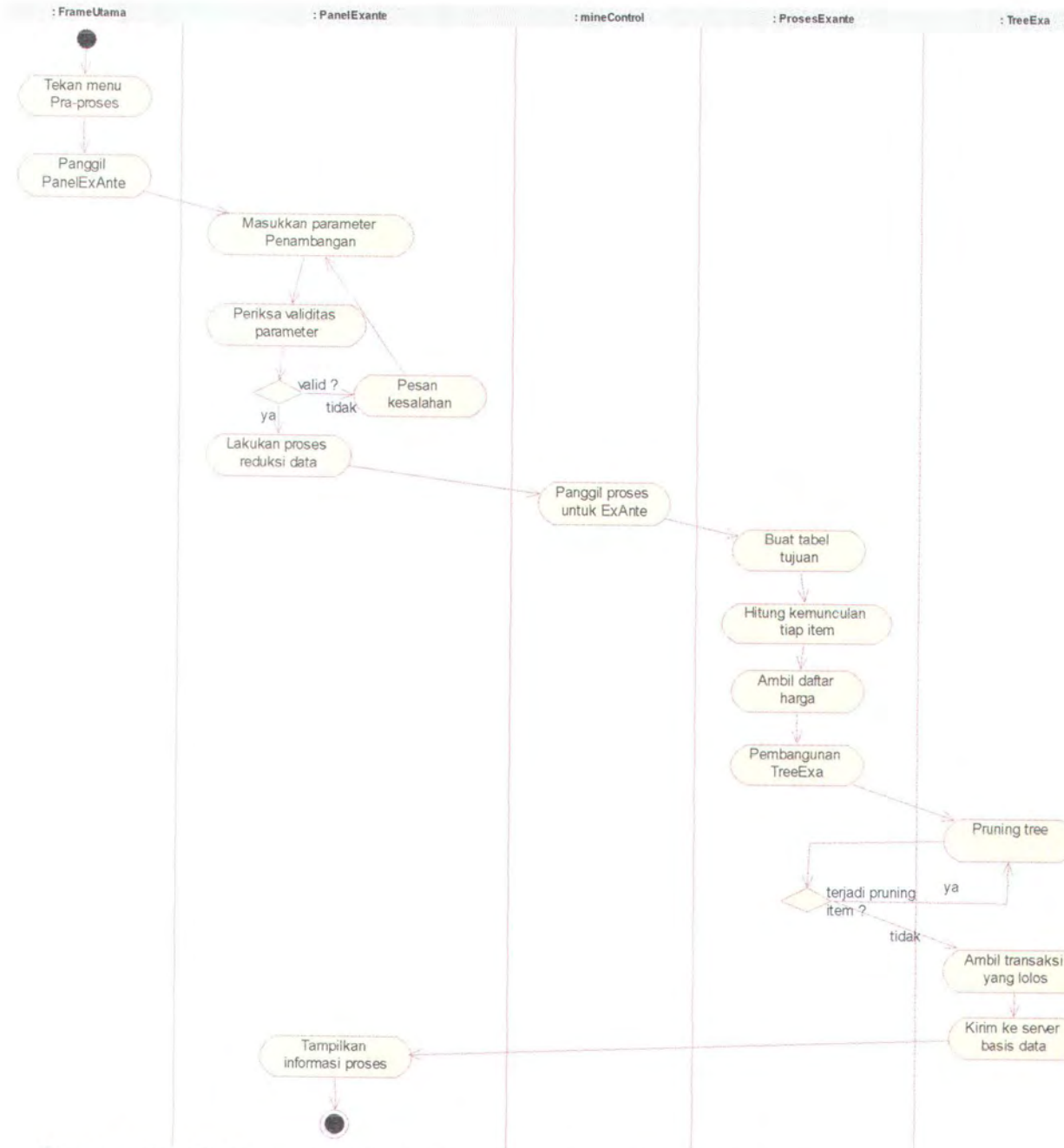
Tabel 3. 4 Spesifikasi Use Case "Terapkan Praproses *ExAnte*"

Spesifikasi	Penjelasan
Aktor	➢ Pengguna
Tujuan	➢ Mereduksi ukuran dataset transaksi dengan menerapkan metode *ExAnte* pada dataset transaksi
Ringkasan	➢ Use case ini merupakan implementasi dari metode praproses *ExAnte*. Proses yang terjadi berupa penerapan metode *ExAnte* dan dilanjutkan proses pencarian pola.
Masukan	➢ Nama Tabel Transaksi ➢ Nama Tabel Item ➢ Minimum Support

Tabel 3.5. Spesifikasi Use Case "Terapkan Praproses *ExAnte*" (lanjutan)

Spesifikasi	Penjelasan
	➢ Batasan Monoton ➢ Niliai Batasan Monoton ➢ Nama tabel tujuan
Luaran	➢ Dataset transaksi yang ukurannya telah tereduksi pada tabel tujuan ➢ Informasi proses reduksi
Kondisi Awal	➢ Aktor berada pada tampilan panel praproses
Kondisi Akhir	➢ Terdapat tabel hasil pada server basis data yang merupakan hasil dari proses *ExAnte*
Aliran Aksi normal	➢ Pengguna memilih menu "Praproses" ➢ Aplikasi memanggil Panel*ExAnte* ➢ Pengguna memasukkan parameter-parameter yang diinginkan ➢ Pengguna menekan tombol "Start" ➢ Aplikasi memerikasa validitas parameter ➢ Aplikasi melakukan proses *ExAnte* ➢ Aplikasi menghasilkan tabel hasil reduksi data

Seluruh aktivitas dalam *use case* ini dapat disederhanakan dalam bentuk diagram aktivitas sebagaimana dapat dilihat pada gambar 3.5 :



Gambar 3. 5 Diagram Aktivitas untuk Use Case "Terapkan Praproses *ExAnte*"

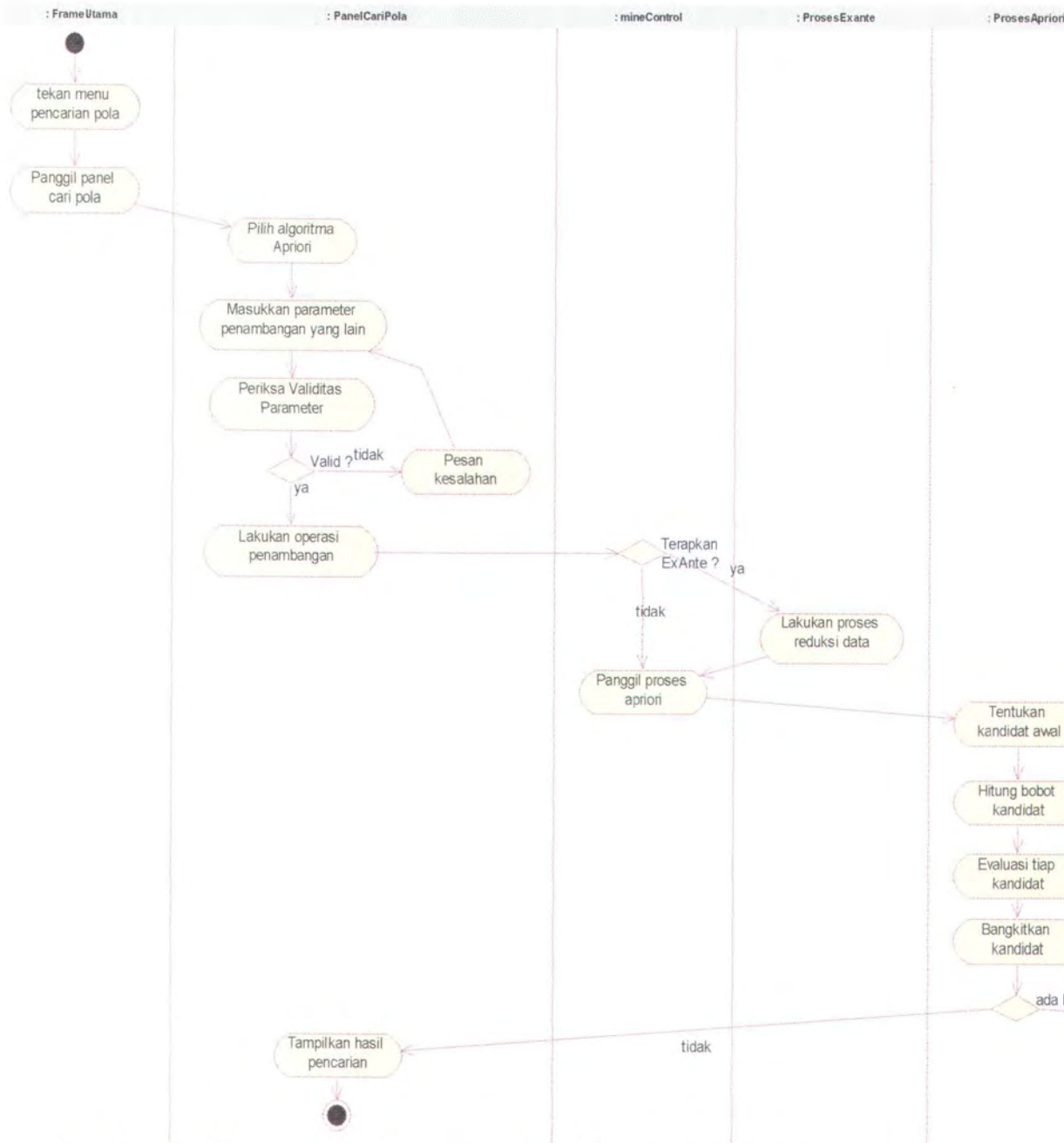
3.1.6. Cari Pola dengan Algoritma Apriori

Use case ini menggambarkan proses pencarian pola *frequent* dengan menggunakan algoritma Apriori. Spesifikasi *use case* ini dapat dilihat pada tabel 3.5.

Tabel 3. 5 Spesifikasi Use Case "Cari Pola dengan Algoritma Apriori"

Spesifikasi	Penjelasan
Aktor	➢ Pengguna
Tujuan	➢ Mencari pola-pola yang memenuhi batasan pegguna dengan menggunakann algoritma Apriori
Ringkasan	➢ Use case ini menerapkan algoritma Apriori dalam melakukan penambangan pola.
Masukan	➢ Nama Tabel Transaksi ➢ Nama Tabel Item ➢ Minimum Support ➢ Batasan Monoton ➢ Niliai Batasan Monoton
Luaran	➢ Tabel pola frequent ➢ Informasi proses penambangan
Kondisi Awal	➢ Aktor berada pada tampilan panel cari pola
Kondisi Akhir	➢ Algoritma penambangan selesai melakukan proses ➢ Panel cari pola dilengkapi hasil proses
Aliran Aksi normal	➢ Pengguna memilih menu "Pencarian Pola" ➢ Aplikasi memanggil PanelCariPola ➢ Pengguna memilih algoritma Apriori pada Combo Box algoritma ➢ Pengguna memasukkan parameter-parameter yang diinginkan ➢ Pengguna menekan tombol "Cari Pola" ➢ Aplikasi memerikasa validitas parameter ➢ Aplikasi memanggil proses Apriori ➢ Pemanggilan kandidat awal ➢ Pemberian bobot pada kandidat ➢ Evaluasi kandidat ➢ Pembangkitan kandidat baru ➢ Aplikasi menampilkan hasil pencarian pada panel cari pola

Seluruh aktivitas dalam *use case* ini dapat disederhanakan dalam bentuk diagram aktivitas pada gambar 3.6.



Gambar 3. 6 Diagram Aktivitas untuk Use Case "Cari Pola dengan Algoritma Apriori"

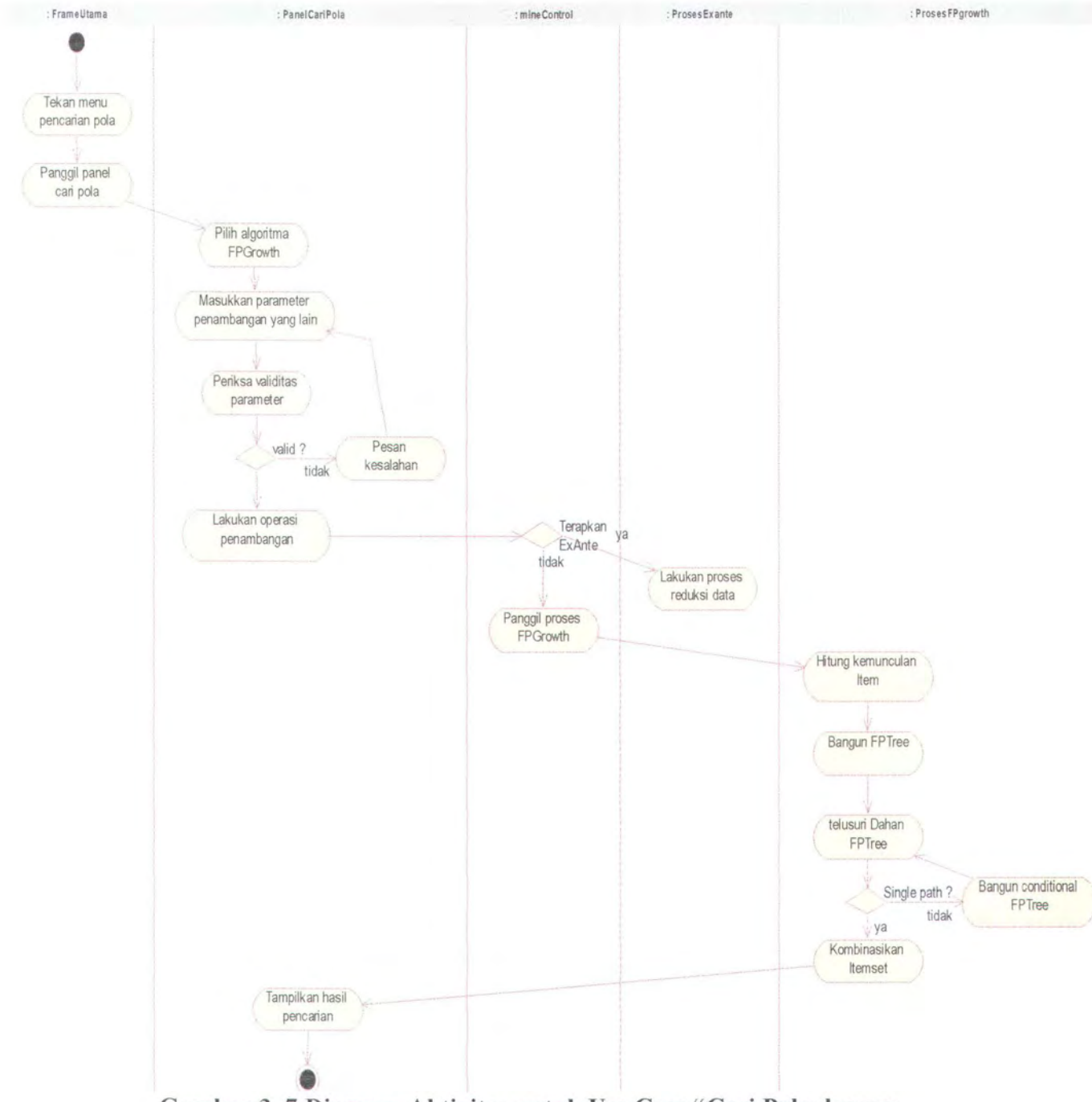
3.1.7. Cari Pola dengan Algoritma FPGrowth

Use case ini menggambarkan proses pencarian pola *frequent* dengan menggunakan algoritma FPGrowth. Spesifikasi *use case* ini dapat dilihat pada tabel 3.6.

Tabel 3. 6 Spesifikasi Use Case "Cari Pola dengan Algoritma FPGrowth"

Spesifikasi	Penjelasan
Aktor	➢ Pengguna
Tujuan	➢ Mencari pola-pola yang memenuhi batasan pegguna dengan menggunakann algoritma FPGrowth
Ringkasan	➢ Use case ini menerapkan algoritma FPGrowth dalam melakukan penambangan pola.
Masukan	➢ Nama Tabel Transaksi ➢ Nama Tabel Item ➢ Minimum Support ➢ Batasan Monoton ➢ Nilai Batasan Monoton
Luaran	➢ Tabel pola frequent ➢ Informasi proses penambangan
Kondisi Awal	➢ Aktor berada pada tampilan panel cari pola
Kondisi Akhir	➢ Algoritma penambangan selesai melakukan proses ➢ Panel cari pola dilengkapi hasil proses
Aliran Aksi normal	➢ Pengguna memilih menu "Pencarian Pola" ➢ Aplikasi memanggil PanelCariPola ➢ Pengguna memilih algoritma FPGrowth pada Combo Box algoritma ➢ Pengguna memasukkan parameter-parameter yang diinginkan ➢ Pengguna menekan tombol "Cari Pola" ➢ Aplikasi memerikasa validitas parameter ➢ Aplikasi memanggil proses fpgrowth ➢ Penghitungan jumlah kemunculan ➢ Pembangunan FPTree ➢ Penelusuran FPTree ➢ Pencarian pola dari dahan FPTree ➢ Aplikasi menampilkan hasil pencarian pada panel cari pola

Seluruh aktivitas dalam *use case* ini dapat disederhanakan dalam bentuk diagram aktivitas pada gambar 3.7 :



Gambar 3. 7 Diagram Aktivitas untuk Use Case "Cari Pola dengan Algoritma FPGrowth"

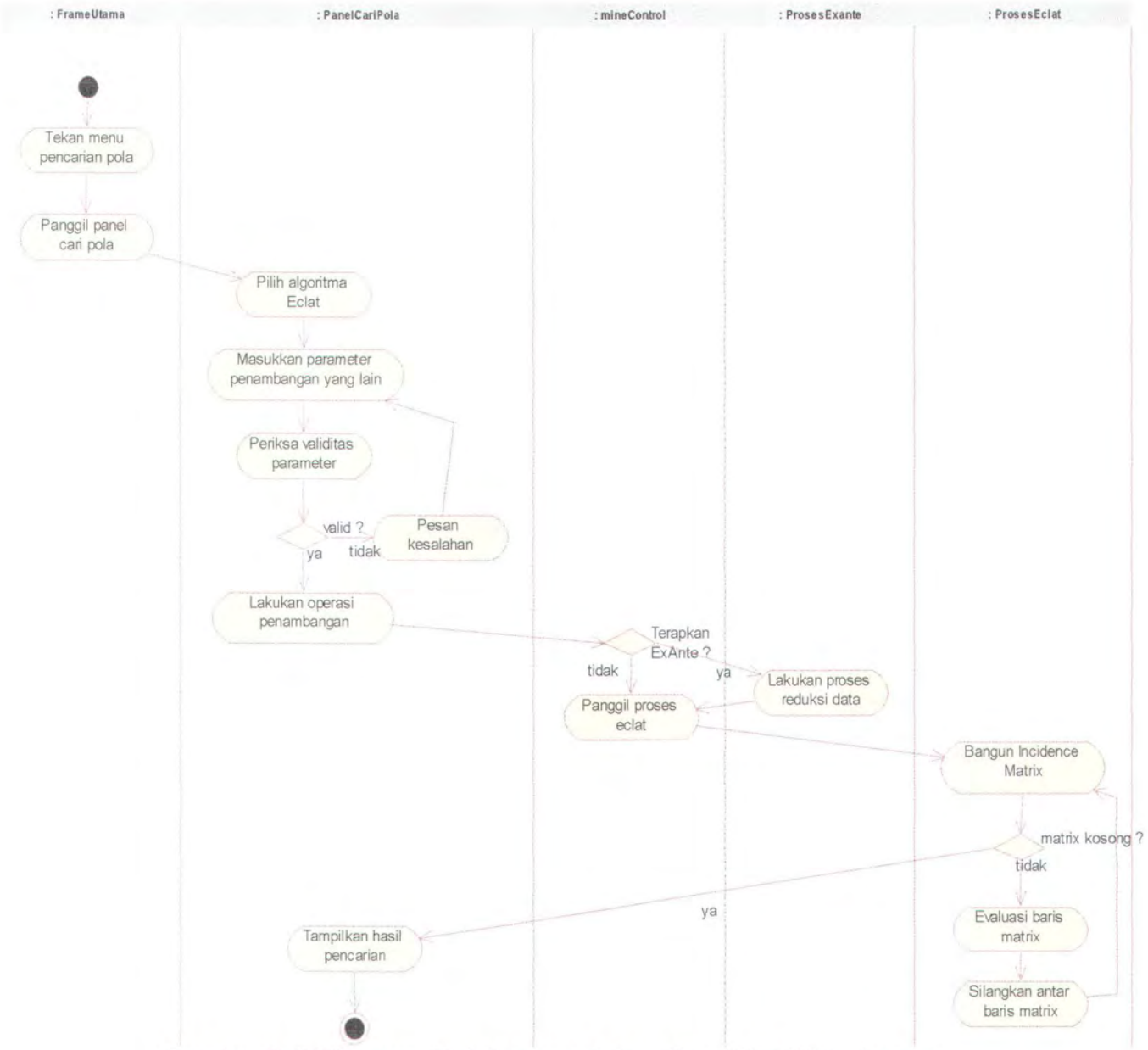
3.1.8. Cari Pola dengan Algoritma Eclat

Use case ini menggambarkan proses pencarian pola *frequent* dengan menggunakan algoritma Eclat. Spesifikasi *use case* ini dapat dilihat pada tabel 3.7.

Tabel 3. 7 Spesifikasi Use Case "Cari Pola dengan Algoritma Eclat"

Spesifikasi	Penjelasan
Aktor	➢ Pengguna
Tujuan	➢ Mencari pola-pola yang memenuhi batasan pegguna dengan menggunakan algoritma Eclat
Ringkasan	➢ Use case ini menerapkan algoritma Eclat dalam melakukan penambangan pola.
Masukan	➢ Nama Tabel Transaksi ➢ Nama Tabel Item ➢ Minimum Support ➢ Batasan Monoton ➢ Niliai Batasan Monoton
Luaran	➢ Tabel pola frequent ➢ Informasi proses penambangan
Kondisi Awal	➢ Aktor berada pada tampilan panel cari pola
Kondisi Akhir	➢ Algoritma penambangan selesai melakukan proses ➢ Panel cari pola dilengkapi hasil proses
Aliran Aksi normal	➢ Pengguna memilih menu "Pencarian Pola" ➢ Aplikasi memanggil PanelCariPola ➢ Pengguna memilih algoritma Eclat pada Combo Box algoritma ➢ Pengguna memasukkan parameter-parameter yang diinginkan ➢ Pengguna menekan tombol "Cari Pola" ➢ Aplikasi memerikasa validitas parameter ➢ Aplikasi memanggil proses eclat ➢ Pembangunan incidence matrix ➢ Pemangkasan matrix ➢ Penyilangan antar baris matrix ➢ Aplikasi menampilkan hasil pencarian pada panel cari pola

Seluruh aktivitas dalam *use case* ini dapat disederhanakan dalam bentuk diagram aktivitas di bawah ini :



Gambar 3. 8 Diagram Aktivitas untuk Use Case "Cari Pola dengan Algoritma Eclat"

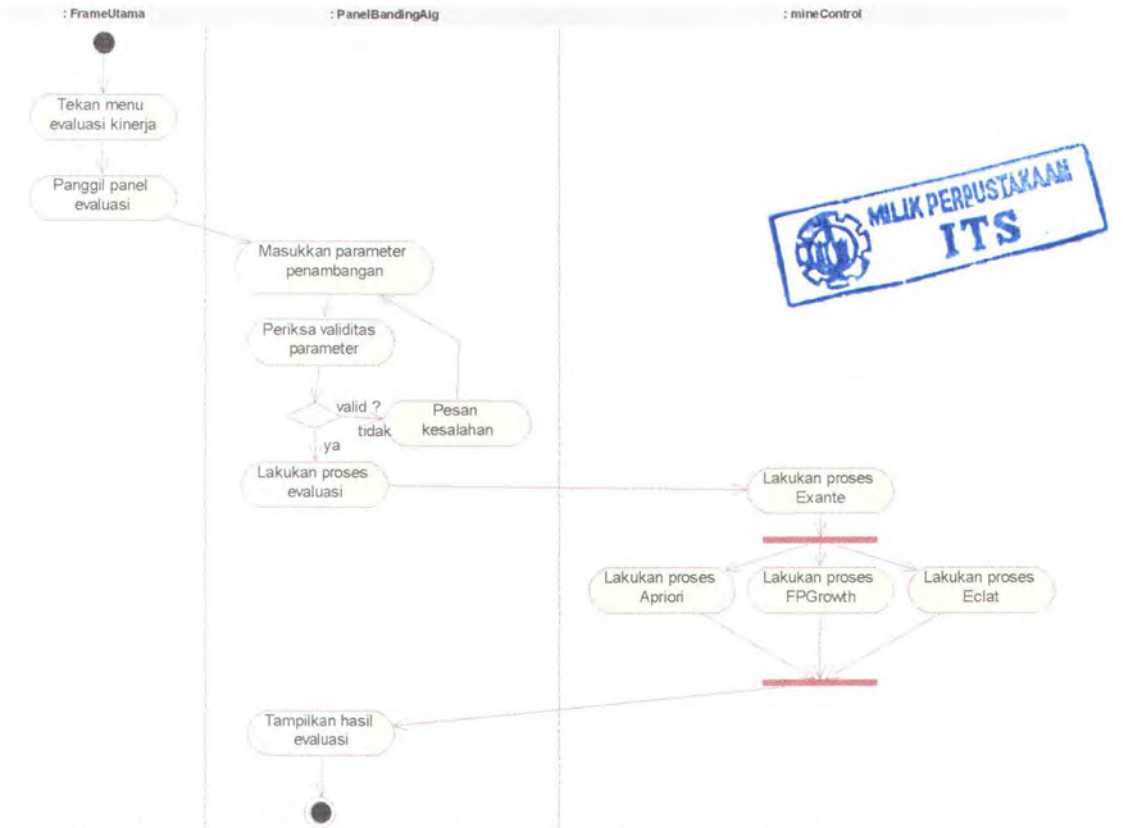
3.1.9. Evaluasi Kinerja Algoritma

Use case ini melakukan sebuah rangkaian proses penambangan pola dengan menerapkan seluruh pilihan algoritma yang ada, baik menggunakan metode *ExAnte* ataupun tidak. Hal ini bertujuan untuk membandingkan hasil dan kinerja masing-masing proses. Spesifikasi *use case* ini dapat dilihat pada tabel 3.8.

Tabel 3. 8 Spesifikasi Use Case "Evaluasi Kinerja Algoritma"

Spesifikasi	Penjelasan
Aktor	➢ Pengguna
Tujuan	➢ Membandingkan kebutuhan sumber daya dari tiap algoritma, baik itu menggunakan metode *ExAnte* atau tidak
Ringkasan	➢ Proses pada use case ini berupa pemanggilan kombinasi proses algoritma yang ada secara berurutan. Kombinasi berupa ExAnte-Appriori, ExAnte-FPGrowth, ExAnte-Eclat, Apriori, FPGrowth dan Eclat.
Masukan	➢ Nama Tabel Transaksi ➢ Nama Tabel Item ➢ Minimum Support ➢ Batasan Monoton ➢ Niliai Batasan Monoton
Luaran	➢ Informasi proses penambangan masing-masing kombinasi.
Kondisi Awal	➢ Aktor berada pada tampilan panel evaluasi kinerja
Kondisi Akhir	➢ Tabel informasi berisi informasi proses masing-masing kombinasi
Aliran Aksi normal	➢ Pengguna memilih menu "Evaluasi Kinerja" ➢ Aplikasi memanggil PanelEvaluasiKinerja ➢ Pengguna memasukkan parameter-parameter yang diinginkan ➢ Pengguna menekan tombol "Cari Pola" ➢ Aplikasi memerikasa validitas parameter ➢ Pencarian pola dengan ExAnte-Appriori ➢ Pencarian pola dengan ExAnte-FPGrowth ➢ Pencarian pola dengan ExAnte-Eclat ➢ Pencarian pola dengan Apriori ➢ Pencarian pola dengan FPGrowth ➢ Pencarian pola dengan Eclat. ➢ Aplikasi mencatat informasi proses ke tabel informasi proses

Seluruh aktivitas dalam *use case* ini dapat disederhanakan dalam bentuk diagram aktivitas pada gambar 3.9.



Gambar 3. 9 Diagram Aktivitas untuk Use Case "Evaluasi kinerja algoritma"

3.1.10. Simpan Hasil

Use case ini menggambarkan proses penyimpanan hasil penambangan kedalam bentuk file dengan nama file sesuai dengan masukan pengguna. Spesifikasi *use case* ini dapat dilihat pada tabel berikut ini :

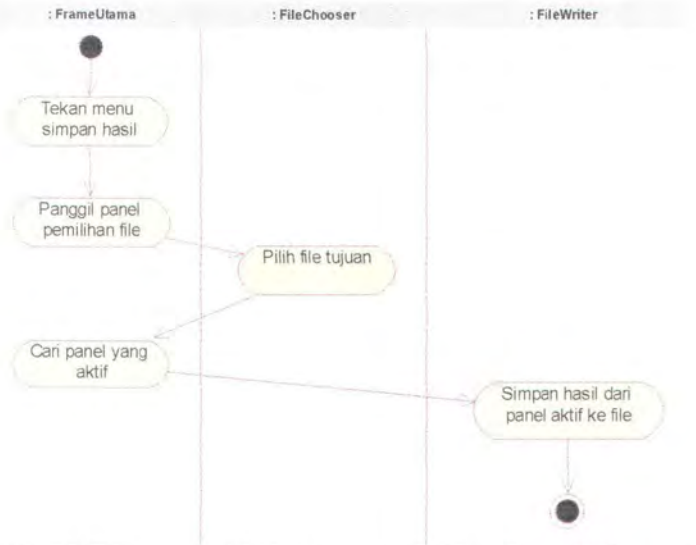
Tabel 3. 9 Spesifikasi Use Case "Simpan Hasil"

Spesifikasi	Penjelasan
Aktor	➢ Pengguna
Tujuan	➢ Menyimpan hasil proses yang ditampilkan pada file pilihan pengguna
Ringkasan	➢ Use case ini dilakukan apabila pengguna ingin menyimpan hasil penambangan kedalam bentuk file.
Masukan	➢ Keluaran proses pencarian pola atau perbandingan algoritma ➢ Nama file
Luaran	➢ File hasil

Tabel 3.9. Spesifikasi Use Case "Simpan Hasil" (lanjutan)

Spesifikasi	Penjelasan
Kondisi Awal	➢ Aktor berada pada tampilan panel pencarian pola ➢ Aktor berada pada tampilan panel evaluasi kinerja
Kondisi Akhir	➢ File hasil berisi seluruh hasil proses
Aliran Aksi normal	➢ Pengguna menekan menu "Simpan Hasil" pada FrameUtama ➢ Aplikasi memanggil PanelSimpan ➢ Pengguna memilih direktori tempat penyimpanan beserta nama filenya ➢ Pengguna menekan tombol "Save" ➢ Aplikasi menyimpan informasi yang ada pada panel hasil ke file pilihan pengguna

Seluruh aktivitas dalam *use case* ini dapat disederhanakan dalam bentuk diagram aktivitas di bawah ini :



Gambar 3. 10 Diagram Aktivitas untuk Use Case "Simpan Hasil"

3.2. Realisasi Use Case

Realisasi *use case* dapat digambarkan dengan sebuah diagram sekuensi. Diagram sekuensi menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang

digambarkan 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 *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*.

Activation bar menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary*, *controller* dan *persistent entity*.

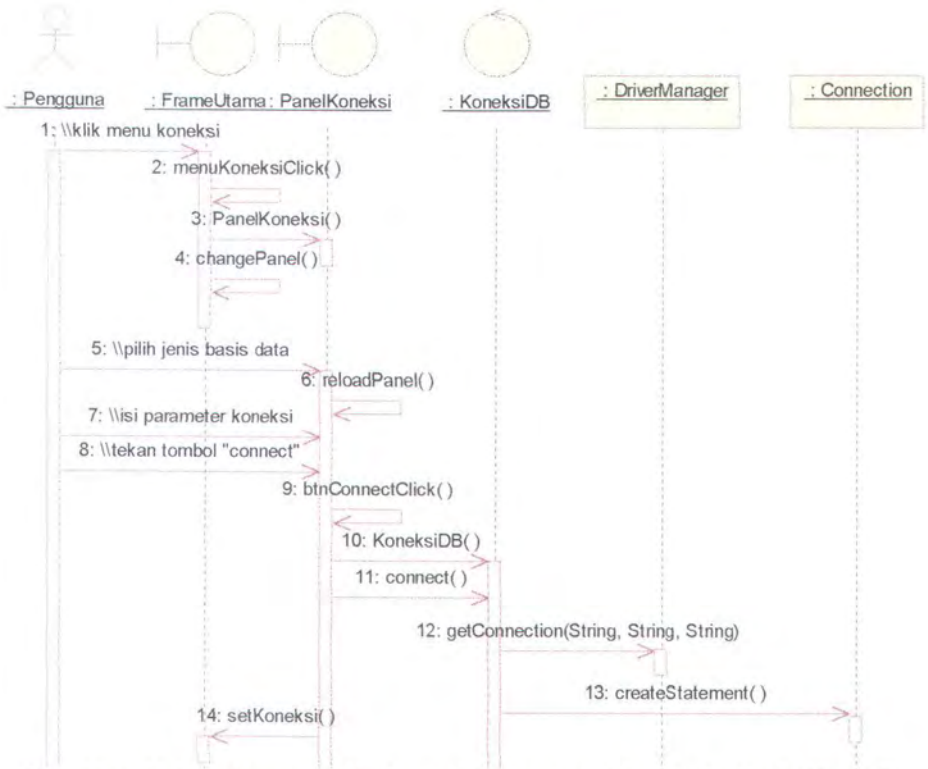
3.2.1. Realisasi Use Case "Bangun Koneksi Basis Data"

Dalam *use case* ini, dilakukan pembuatan hubungan antara aplikasi dengan server basis data. Untuk melakukannya, aplikasi memanfaatkan dua buah kelas utilitas yang berhubungan dengan pembangunan koneksi ini. Kelas utilitas tersebut adalah :

- DriverManager
    - Kelas ini digunakan untuk membangun koneksi berdasarkan parameter url, username dan password yang dimasukkan oleh pengguna
- Connection
    - Kelas ini digunakan sebagai penghubung dengan server basis data serta melakukan berbagai operasi kueri SQL.

Proses dalam use case ini dilakukan oleh sebuah kelas yang bernama KoneksiDB. Kelas ini merupakan kelas kontrol yang berfungsi menghubungkan

aplikasi dengan server basis data. Kelas ini terlibat hampir dalam semua proses use case dalam aplikasi. Baik itu berupa pembangkitan dataset maupun pada pencarian pola. Kelas lain yang terlibat dalam use case ini adalah kelas PanelKoneksi. Kelas ini merupakan kelas yang menyediakan antarmuka untuk kebutuhan pembangunan koneksi dengan server basis data. Aliran kejadian dari *use case* ini dapat digambarkan dengan diagram sekuensi pada gambar dibawah ini :



Gambar 3. 11 Diagram Sekuensi "Bangun Koneksi Basis Data"

3.2.1.2. Aktivitas

1) \\ klik menu koneksi

   - Dilakukan oleh pengguna untuk memanggil panel koneksi dari frame utama.

2) menuKoneksiClick( )

   - Merupakan operasi yang dilakukan oleh frame utama untuk menampilkan panel koneksi di frame utama

3) PanelKoneksi( )

- Inisialisasi PanelKoneksi dengan komponen-komponen pembangun antarmuka.

4) changePanel()

- Pengosongan panel dasar frame utama dan menampilkan panel yang diinginkan

5) \\ pilih jenis basis data

- Pengguna dapat memilih antara beberapa macam basis data yang dapat dihubungkan dengan aplikasi

6) reloadPanel()

- Operasi ini berfungsi untuk memanggil panel untuk masukan parameter koneksi yang sesuai dengan pilihan jenis basis data dari pengguna

7) \\ isikan parameter koneksi

- Pengisian parameter-parameter yang diperlukan dalam pembangunan koneksi

8) \\ tekan tombol "Connect"

- Penekanan tombol untuk memulai proses pembangunan koneksi

9) btnConnectClick()

- Operasi memulai proses koneksi pada panel koneksi

10) KoneksiDB()

- Panel koneksi memanggil kelas KoneksiDB. Kelas ini berfungsi sebagai kelas pembangun koneksi

11) connect( )

- KoneksiDB membangun koneksi dengan server basis data berdasarkan parameter pengguna

12) getConnection(String, String, String)

- Pembangunan koneksi

13) createStatement( )

- Menghubungkan dengan server basis data dalam melakukan operasi SQL

14) setKoneksi(KoneksiDB)

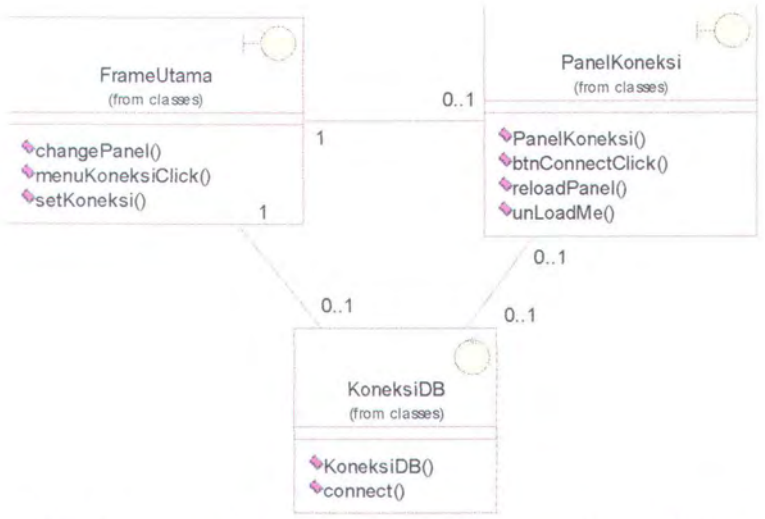
- Setelah koneksi berhasil dibangun, koneksi memberikan koneksi tersebut ke frame utama untuk digunakan pada proses-proses yang lain.

3.2.1.3. Gambaran Partisipasi Objek

Proses dalam *use case* "Bangun Koneksi Basis Data" dibangun dengan melibatkan beberapa objek sebagai berikut :

- FrameUtama
- PanelKoneksi
- KoneksiDB

Hubungan partisipasi objek-objek diatas dapat digambarkan dengan menggunakan diagram VOPC (*View of Participated Class*) di bawah ini :



Gambar 3. 12 VOPC "Bangun Koneksi Basis Data"

3.2.2. Realisasi Use Case "Bangkitkan Dataset Transaksi"

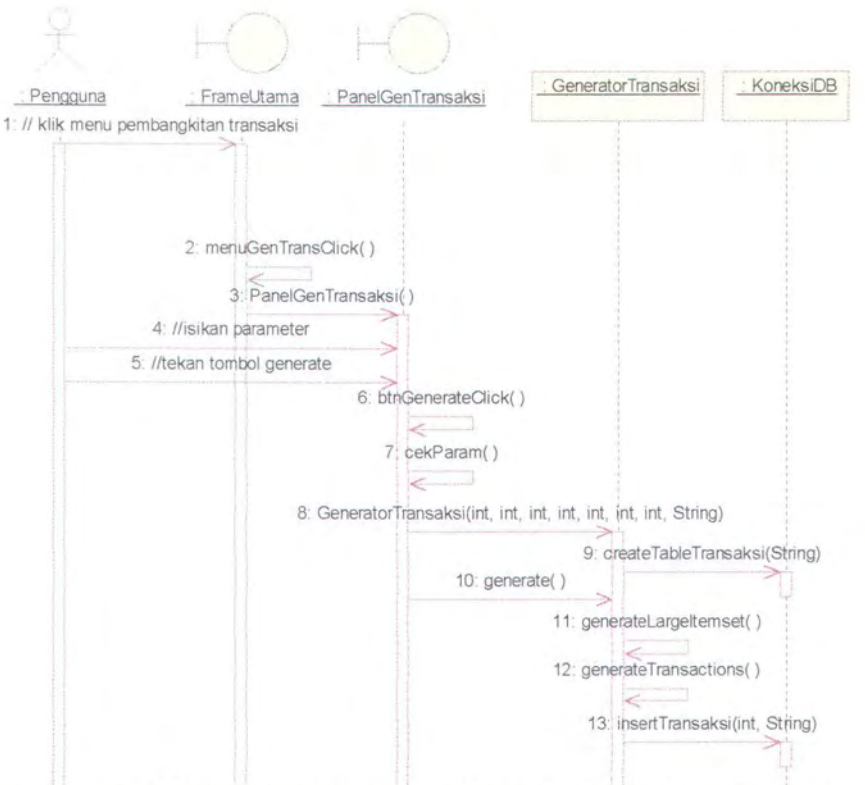
Dalam *use case* ini, dilakukan pembuatan dataset transaksi sintetis berdasarkan parameter pengguna. Untuk melakukannya, aplikasi memanfaatkan 3 buah kelas utilitas. Kelas utilitas tersebut adalah :

- GeneratorTransaksi
  - Kelas ini digunakan untuk membangkitkan large itemset dan transaksi

- Poisson
  - Kelas ini berfungsi untuk menghasilkan bilangan berdasarkan distribusi poisson

- RandomNumberGenerator
  - Kelas ini berfungsi untuk menghasilkan angka acak dengan berbagai pilihan distribusi

Aliran kejadian dari *use case* ini dapat digambarkan dengan diagram sekuensi pada gambar 3.13.



Gambar 3. 13 Diagram Sekuensi "Bangkitkan Dataset Transaksi"

3.2.2.2. Aktivitas

1) // klik menu pembangkitan transaksi

   - Untuk memulai proses pembangkitan, pengguna harus memanggil terlebih dahulu panel untuk pembangkitan transaksi

2) menuGenTransClick( )

   - Operasi ini dilakukan ketika pengguna menekan menu pembangkitan transaksi

3) PanelGenTransaksi( )

   - Merupakan operasi inisialisasi komponen-komponen visual dari panel pembangkitan transaksi

4) //isikan parameter

- Untuk pembangkitan transaksi, pengguna harus memasukkan parameter yang menentukan karakteristik dataset tersebut.

5) //tekan tombol generate

- Setelah selesai mengisikan parameter, pengguna harus menekan tombol "Generate" untuk memulai proses pembangkitan

6) btnGenerateClick( )

- Operasi ini dipanggil ketika terjadi penekanan tombol "Generate" sebagai tanda dimulainya proses pembangkitan

7) cekParam( )

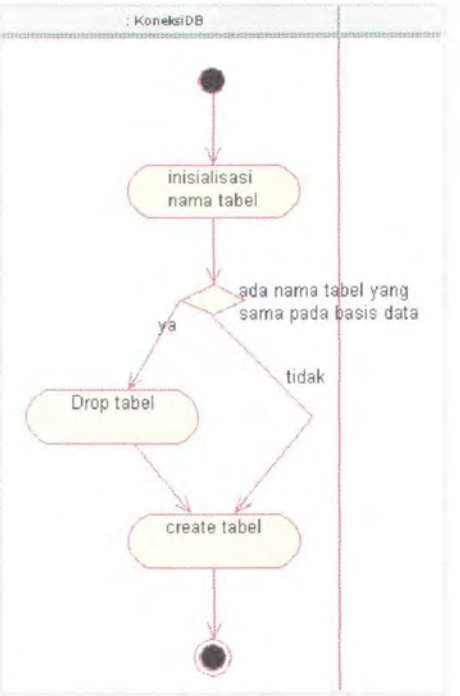
- Operasi ini dilakukan untuk melakukan pengecekan prameter apakah sudah lengkap dan benar atau tidak

8) GeneratorTransaksi(int, int, int, int, int, int, int, String)

- Inisialisasi kelas GeneratorTransaksi serta pelemparan parameter masukan dari pengguna

9) createTableTransaksi(String)

- Langkah pertama dalam proses pemabngkitan transaksi adalah pembuatan terlebih dahulu tabel transaksi tersebut. Prinsip kerja dari pembuatan tabel ini dapat digambarkan dengan diagram aktivitas sebagaimana gambar 3.14.



Gambar 3. 14 Diagram Aktivitas createTableTransaksi(String)

10) generate( )

- Operasi ini berfungsi sebagai pemanggil operasi generateLargeItemset( ) dan generateTransactions( ). Kedua operasi ini merupakan operasi yang melakukan pembangkitan dataset.

11) generateLargeItemset( )

- Operasi ini berfungsi untk membangkitkan sekumpulan Large Itemset dari item-item yang ada. Ukuran dari large itemset ini didapatkan dari parameter yang didapatkan dari pengguna. Selanjutnya, large itemset berguna sebagai calon anggota dari transaksi.

12) generateTransactions( )

- Operasi ini melakukan pembangkitan transaksi dan mengirimkannya ke server basis data

13) insertTransaksi(int, String)

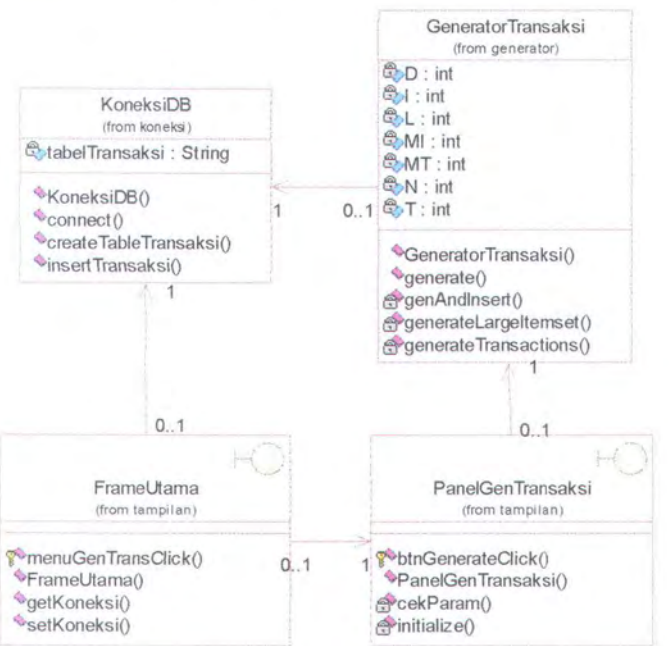
- Operasi ini berfungsi untuk mengirimkan transaksi-transaksi yang telah dibangkitkan ke server basis data untuk disimpan dalam tabel transaksi yang telah dibuat.

3.2.2.3. Gambaran Partisipasi Objek

Proses dalam *use case* "Bangkitkan Dataset Transaksi" dibangun dengan melibatkan beberapa objek sebagai berikut :

- FrameUtama
- KoneksiDB
- PanelGenTransaksi
- GeneratorTransaksi

Hubungan partisipasi objek-objek di atas dapat digambarkan dengan menggunakan diagram VOPC (*View of Participated Class*) di bawah ini :



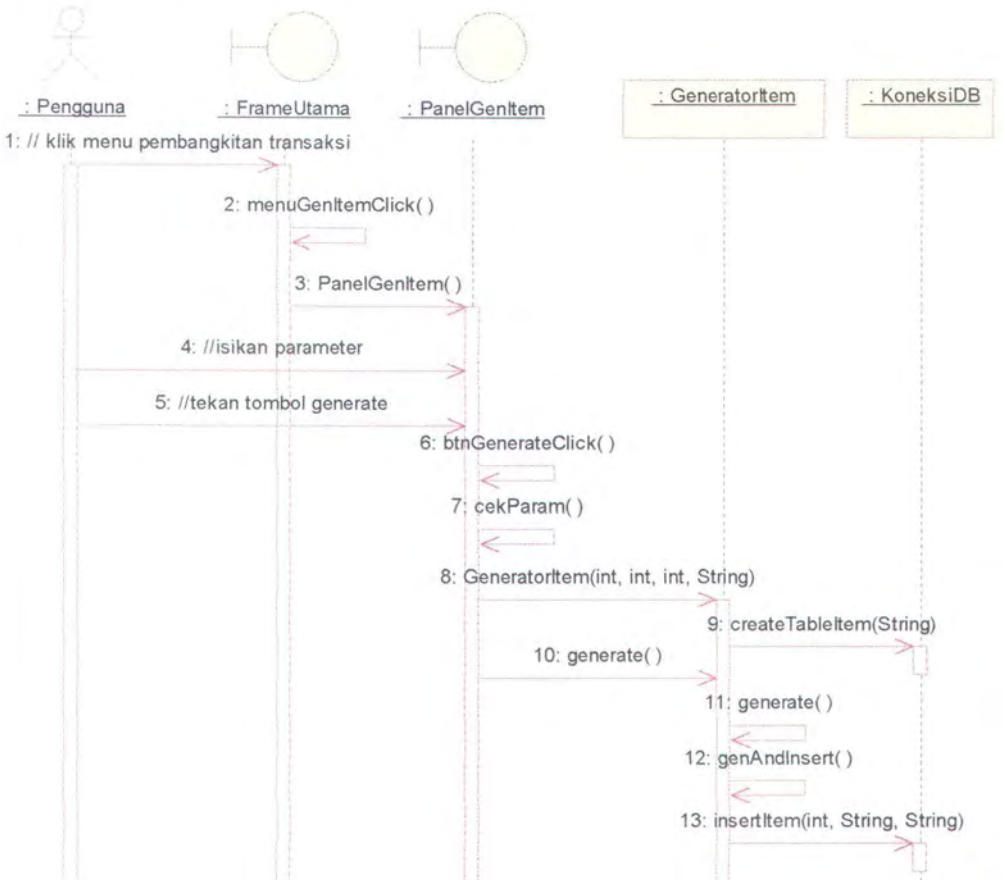
Gambar 3. 15 VOPC "Bangkitkan Dataset Transaksi"

3.2.3. Realisasi Use Case "Bangkitkan Dataset Item"

Dalam *use case* ini, dilakukan pembuatan dataset item sintetis berdasarkan parameter pengguna. Untuk melakukannya, aplikasi memanfaatkan dua buah kelas utilitas. Kelas utilitas tersebut adalah :

- GeneratorItem
  - Kelas ini digunakan untuk membangkitkan large itemset dan transaksi
- RandomNumberGenerator
  - Kelas ini berfungsi untuk menghasilkan angka acak dengan berbagai pilihan distribusi

Aliran kejadian dari *use case* ini dapat digambarkan dengan diagram sekuensi pada gambar dibawah ini :



Gambar 3. 16 Diagram Sekuensi "Bangkitkan Dataset Item"

3.2.3.1. Aktivitas

1) // klik menu pembangkitan item

   - Untuk memulai proses pembangkitan, pengguna harus memanggil terlebih dahulu panel untuk pembangkitan item

2) menuGenItemClick( )

   - Operasi ini dilakukan ketika pengguna menekan menu "Pembangkitan Item"

3) PanelGenItem( )

   - Merupakan operasi inisialisasi komponen-komponen visual dari panel pembangkitan item

4) //isikan parameter

   - Untuk pembangkitan item, pengguna harus memasukkan parameter yang menentukan karakteristik dataset tersebut.

5) //tekan tombol generate

   - Setelah selesai mengisikan parameter, pengguna harus menekan tombol "Generate" untuk memulai proses pembangkitan

6) btnGenerateClick( )

   - Operasi ini dipanggil ketika terjadi penekanan tombol "Generate" sebagai tanda dimulainya proses pembangkitan

7) cekParam( )

   - Operasi ini dilakukan untuk melakukan pengecekan prameter apakah sudah lengkap dan benar atau tidak

8) GeneratorItem(int, int, int, String)

   - Inisialisasi kelas GeneratorItem serta pelemparan parameter masukan dari pengguna

9) createTableItem(String)

- Langkah pertama dalam proses pembangkitan item, adalah pembuatan terlebih dahulu tabel item tersebut.

10) generate( )

- Operasi ini berfungsi sebagai pemanggil operasi genAndInsert( ).

11) genAndInsert( )

- Operasi ini melakukan pembangkitan item dan mengirimkannya ke server basis data

12) insertItem(int, String, String)

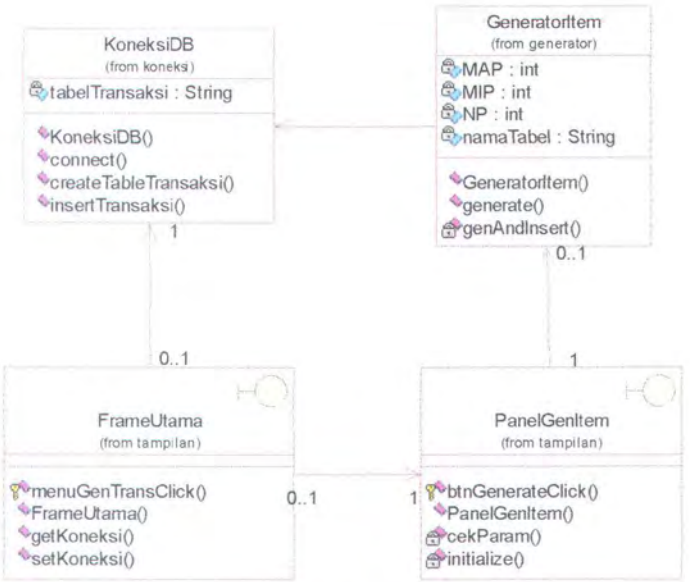
- Operasi ini berfungsi untuk mengirimkan item-item yang telah dibangkitkan ke server basis data untuk disimpan dalam tabel item yang telah dibuat.

3.2.3.2. Gambaran Partisipasi Objek

Proses dalam *use case* "Bangkitkan Dataset Item" dibangun dengan melibatkan beberapa objek sebagai berikut :

- FrameUtama
- PanelGenItem
- GeneratorItem
- KoneksiDB

Hubungan partisipasi objek-objek di atas dapat digambarkan dengan menggunakan diagram VOPC (*View of Participated Class*)pada gambar 3.17.



Gambar 3. 17 VOPC "Bangkitkan Dataset Item"

3.2.4. Realisasi Use Case "Terapkan Praproses ExAnte"

Dalam *use case* ini, dilakukan praproses terhadap dataset transaksi dengan tujuan untuk mereduksi ukuran dataset tersebut. Proses dalam use case ini menggunakan beberapa kelas dalam melakukan prosesnya.

Kelas mineControl merupakan kelas kontrol yang pertama kali dipanggil setelah pengguna menekan tombol start. Kelas mineControl inilah yang akan menentukan operasi-operasi mana saja yang harus dilakukan untuk memenuhi parameter dari pengguna.

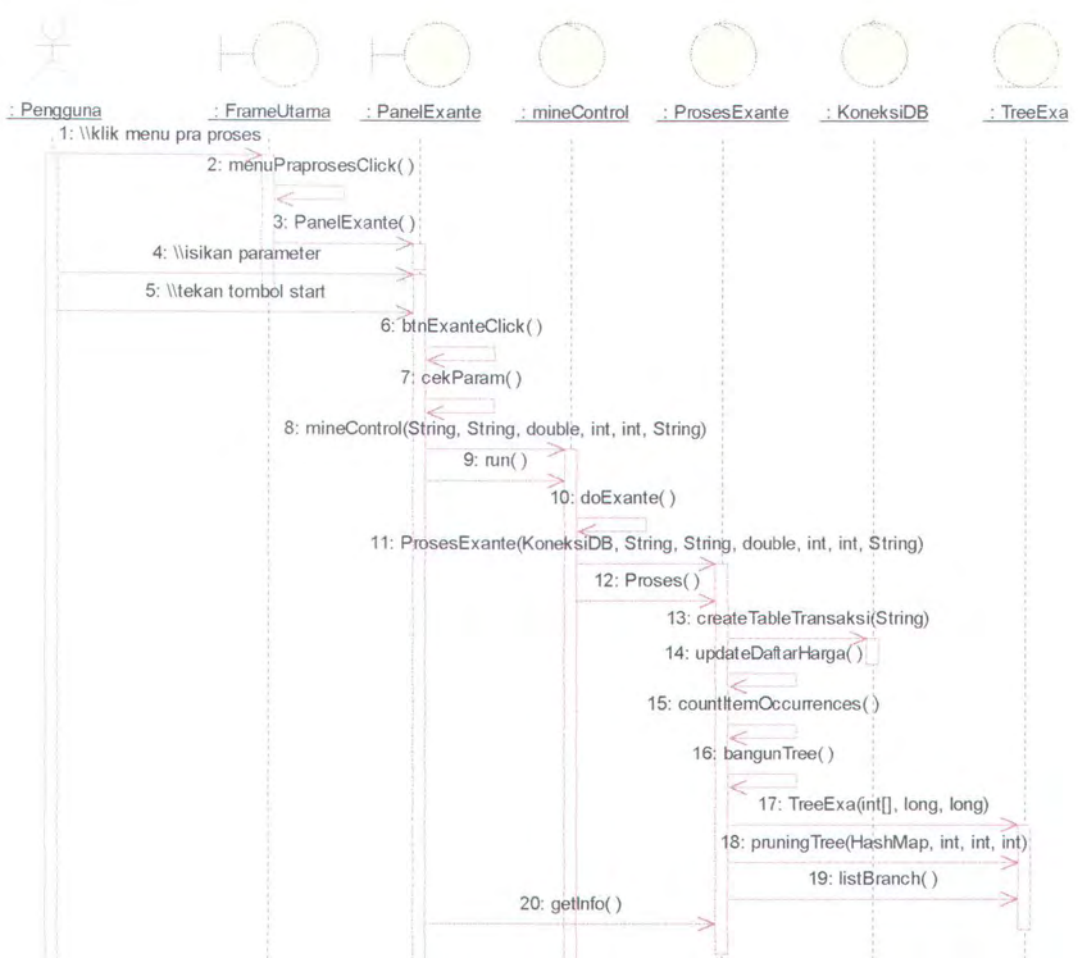
Proses reduksi data dilakukan oleh kelas ProsesExante. Kelas ini merupakan kelas control yang berfungsi sebagai pengatur proses apa saja yang akan dilakukan. Untuk melakukan fungsinya, kelas ini memanfaatkan kelas ExaTree. Yaitu, struktur data berbentuk pohon yang merepresentasikan isi dari table transaksi awal.

Selain 2 buah kelas kontrol di atas, dalam melakukan proses reduksi data dalam use case ini juga digunakan kelas KoneksiDB yang telah dijelaskan pada sub bab sebelumnya. Kelas ini dipanggil untuk melakukan pembuatan tabel transaksi serta mengirimkan transaksi-transaksi yang lolos proses reduksi.

Kelas utilitas dari use case ini antara lain :

- TreeExa
  - Seperti dijelaskan di atas, kelas ini berfungsi untuk menyimpan transaksi dalam bentuk pohojn
- ExaNode
  - Kelas ini menjelaskan node-node yang ada pada ExaTree
- HeaderEntry
  - Kelas ini berfungsi menyimpan item beserta frequensi kemunculannya

Aliran kejadian dari *use case* ini digambarkan dengan diagram sekuensi pada gambar 3.18



Gambar 3. 18 Diagram Sekuensi "Terapkan Praproses ExAnte"

3.2.4.2. Aktivitas

1) // klik menu praproses

   - Langkah pertama dalam melakukan praproses adalah dengan cara memanggil terlebih dahulu panel ExAnte dengan menekan menu "Praproses"

2) menuPraprosesClick( )

   - Ketika menu "praproses" ditekan, frame utama akan melaksanakan operasi ini dan memanggil panelExante.

3) PanelExante( )

   - Inisialisasi PanelExante dengan komponen-komponen pembangun antarmuka.

4) // isikan parameter

   - Pengisian parameter-parameter yang diperlukan dalam Praproses data.

5) //Tekan tombol start

   - Untuk melakukan proses, tekan tombol start.

6) btnExanteClick( )

   - Operasi ini menandakan dimulainya proses reduksi

7) cekParam( )

   - Operasi ini memeriksa apakah nama tabel transaksi dan item serta nilai minimum support dan batasan monoton telah terisi. Jika semua parameter telah terisi dan valid, maka operasi mengembalikan nilai "*true*", dan apabila sebaliknya, akan mengembalikan nilai "*false*".

8) mineControl(String, String, boolean, int, double, int, int,String)

   - Operasi ini menginisialisasikan penambangan dengan mengirimkan parameter dan memanggil algoritma sesuai pilihan pengguna.

9) Run()

- Operasi dimulainya *thread* proses reduksi

10) doExAnte( )

- Kelas MineControl memanggil kelas ExAnte sekaligus melemparkan parameter penambangan dari pengguna

11) ProsesExante(KoneksiDB, String, String, double, int, int,String)

- Inisialisasi porses

12) Proses()

- Operasi ini berisi kumpulan tahapan operasi-operasi yang harus dijalankan.

13) createTableTransaksi(String)

- Metode ExAnte tidak mengubah tabel dataset awal, melainkan dengan memindahkan isi tabel tersebut ke sebuah tabel sementara dan selanjutnya mereduksi ukuran tabel tersebut. Operasi ini digunakan untuk membuat tabel sementara tersebut.

14) updateDaftarHarga( )

- Operasi ini dilakukan untuk menyimpan jumlah kemunculan masing-masing item pada dataset beserta nilai *price* nya.

15) countItemOccurrences( )

- Operasi ini berfungsi untuk menghitung jumlah kemunculan dari tiap item

16) bangunTree( )

- Pembangunan TreeExa, yaitu pohon yang merepresentasikan dataset transaksi yang nantinya akan mengalami proses reduksi.

17) TreeExa(int[], long, long)

- Inisialisasi TreeExa dengan parameter berupa daftar item, jumlah transaksi dan minimum count.

18) pruningTree(HashMap, int, int, int)

- Pemangkasan pohon TreeExa, pemangkasan pertama berupa pemangkasan dahan yang tidak lolos batasan monoton, dan yang kedua adalah pengilangan nilai count item-item non frequent.

19) listBranch( )

- penelusuran akhir pohon untuk mencari transaksi-transaksi yang lolos batasan dan mengirimkannya ke table hasil

20) getInfo( )

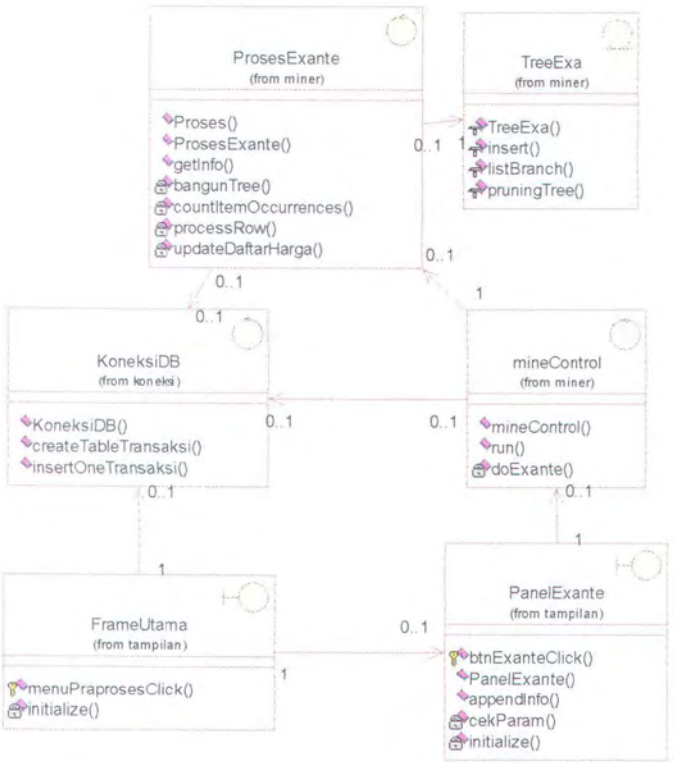
- Menampilkan informasi dari proses yang telah dilakukan.

3.2.4.3. Gambaran partisipasi objek

Proses dalam *use case* "Terapkan Praproses ExAnte" dibangun dengan melibatkan beberapa objek sebagai berikut :

- FrameUtama
- PanelExante
- ProsesExAnte
- MineControl
- KoneksiDB
- TreeExa

Hubungan partisipasi objek-objek di atas dapat digambarkan dengan menggunakan diagram VOPC (*View of Participated Class*) di bawah ini :



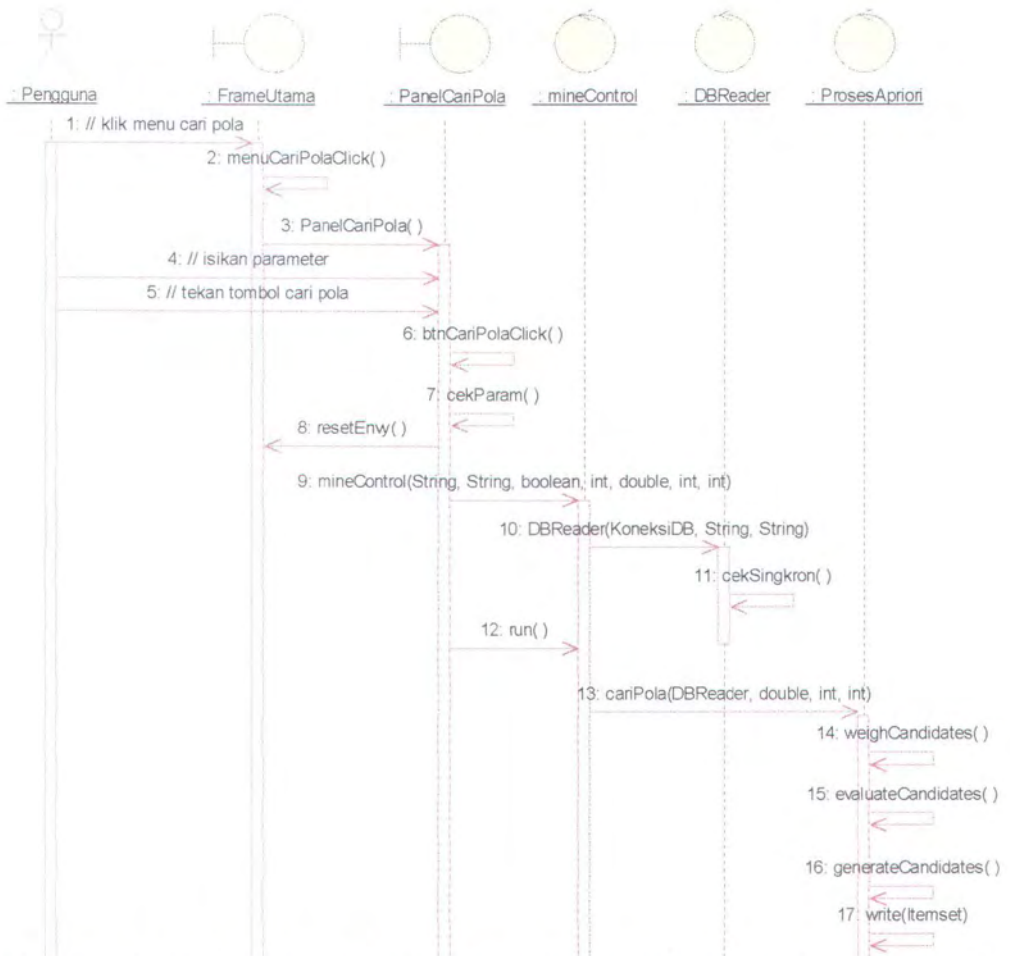
Gambar 3. 19 VOPC "Terapkan Praproses ExAnte"

3.2.5. Realisasi Use Case "Cari Pola dengan Algoritma Apriori"

Dalam *use case* ini, dilakukan pencarian pola dengan mengikuti prinsip kerja algoritma Apriori berdasarkan batasan dari pengguna. Untuk melakukannya, aplikasi memanfaatkan sebuah kelas utilitas yaitu :

- HashTree
  - Kelas ini digunakan untuk membuat index item dalam bentuk sebuah *tree*

Aliran kejadian dari *use case* ini dapat digambarkan dengan diagram sekuensi pada gambar 3.20



Gambar 3. 20 Diagram Sekuensi "Cari Pola dengan Algoritma Apriori"

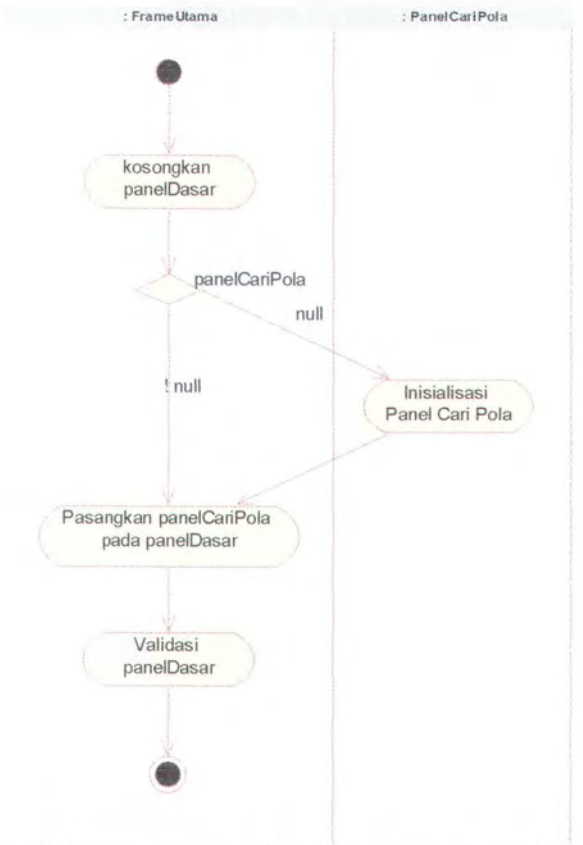
3.2.5.2. Aktivitas

1) // Klik menu cari pola

- Langkah pertama dalam melakukan penambangan adalah dengan cara memanggil terlebih dahulu panel pencarian pola dengan menekan menu "Cari Pola"

2) menuCariPolaClick( )

- Ketika menu "Cari Pola" ditekan, frame utama akan memanggil operasi ini. Diagram aktivitas untuk operasi ini dapat ditunjukkan pada gambar 3.21.



Gambar 3. 21 Diagram Aktivitas menuCariPolaClick()

3) PanelCariPola( )

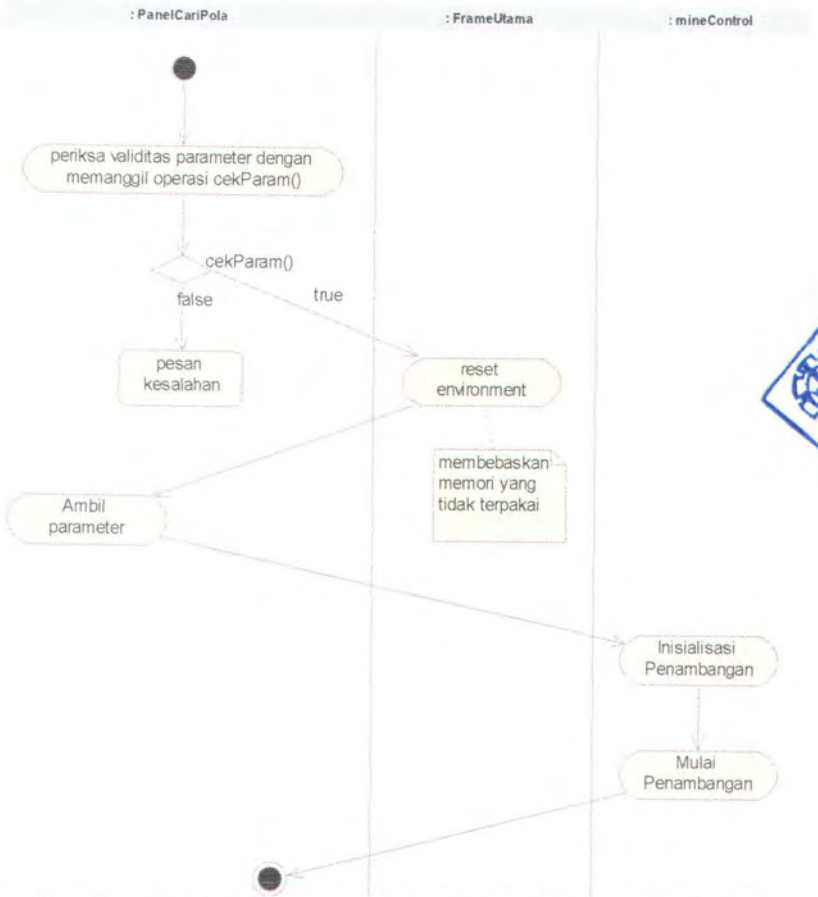
- Inisialisasi PanelCariPola dengan komponen-komponen pembangun antarmuka.

4) // isikan parameter

- Pengisian parameter-parameter yang diperlukan dalam penambangan pola.
- Parameter tersebut antara lain :
  - Nama tabel transaksi, (text field)
  - Nama tabel item, (text field)
  - Penerapan ExAnte, (radio button)
  - Algoritma, (combo box)
  - Nilai minimum support, (text field)

- Jenis batasan monoton, (combo box)
- Nilai batasan monoton, (text field)
- // tekan tombol cari pola
- btnCariPolaClick( )

➢ ketika tombol "Cari Pola" ditekan, PanelCariPola akan memanggil operasi ini. Diagram aktivitas untuk operasi ini dapat ditunjukkan pada gambar di bawah ini :



Gambar 3. 22 Diagram Aktivitas btnCariPolaClick( )

5) cekParam( )

➢ Operasi ini memeriksa apakah nama tabel transaksi dan item serta nilai minimum support dan batasan monoton telah terisi. Jika semua parameter telah terisi dan valid, maka operasi mengembalikan nilai "true", dan apabila sebaliknya, akan mengembalikan nilai "false".

6) resetEnvy( )

- Operasi ini mengembalikan nilai memori maksimal ke nol, serta membebaskan memori yang tak terpakai melalui garbage colector.

7) mineControl(String, String, boolean, int, double, int, int)

- Operasi ini menginisialisasikan penambangan dengan mengirimkan parameter dan memanggil algoritma sesuai pilihan pengguna.

8) DBReader(KoneksiDB, String, String)

- Pemanggilan kelas DBReader yang digunakan untuk pembacaan tabel pilihan pengguna.

9) cekSingkron( )

- Pemeriksaan sinkronisasi antara tabel transaksi dengan tabel item. Apabila pada tabel transaksi terdapat item yang tidak tercantum pada tabel item, maka muncul pesan kesalahan.

10) run( )

- Operasi untuk menandakan dimulainya proses penambangan.

11) cariPola(DBReader, double, int, int) (Pengguna memilih algoritma Apriori)

- MineControl memanggil kelas Apriori untuk melakukan proses penambangan
- Proses penambangan yang dilakukan oleh operasi ini dapat digambarkan pada diagram aktivitas gambar 3.23

Gambar 3. 23 Operasi cariPola() algoritma Apriori

12) weighCandidates( )

- Operasi ini memberikan bobot kepada tiap kandidat. Bobot digunakan untuk menentukan apakah kandidat tersebut termasuk frequent atau tidak

13) evaluateCandidates( )

- Operasi ini mengevaluasi tiap-tiap kandidat apakah frequent atau tidak

14) generateCandidates( )

- Operasi ini membangkitkan kandidat-kandidat itemset baru dari itemset yang lolos proses evaluasi

15) write(Itemset)

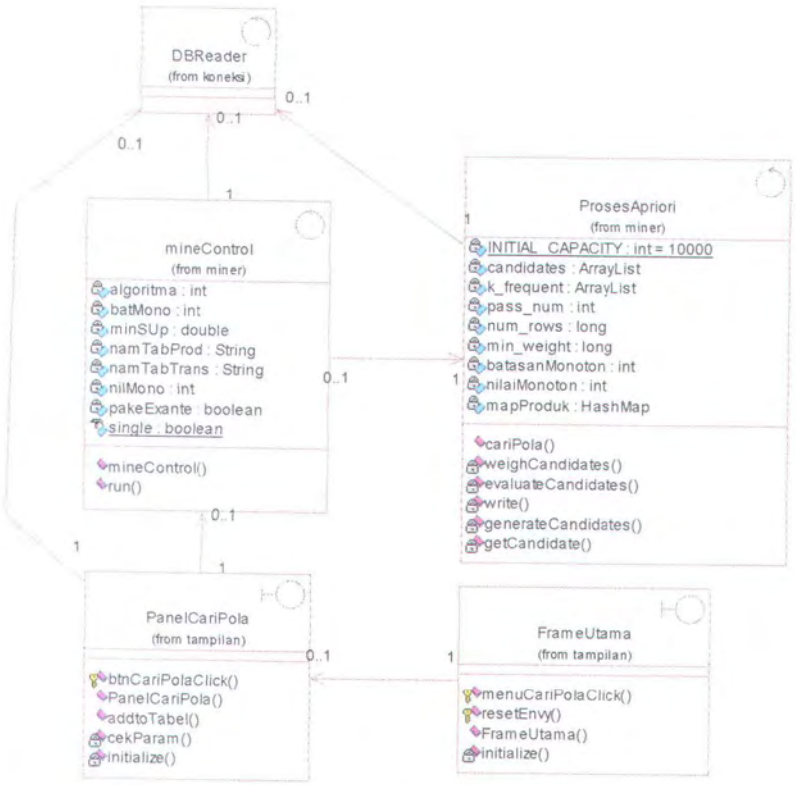
- Operasi ini menuliskan pola yang lolos batasan ke tabel hasil pada PanelCariPola

3.2.5.3. Gambaran Partisipasi Objek

Proses dalam *use case* "Cari Pola dengan Algoritma Apriori" dibangun dengan melibatkan beberapa objek sebagai berikut :

- FrameUtama
- PanelCariPola
- MineControl
- ProsesApriori
- DBReader

Hubungan partisipasi objek-objek di atas dapat digambarkan dengan menggunakan diagram VOPC (*View of Participated Class*) di bawah ini :



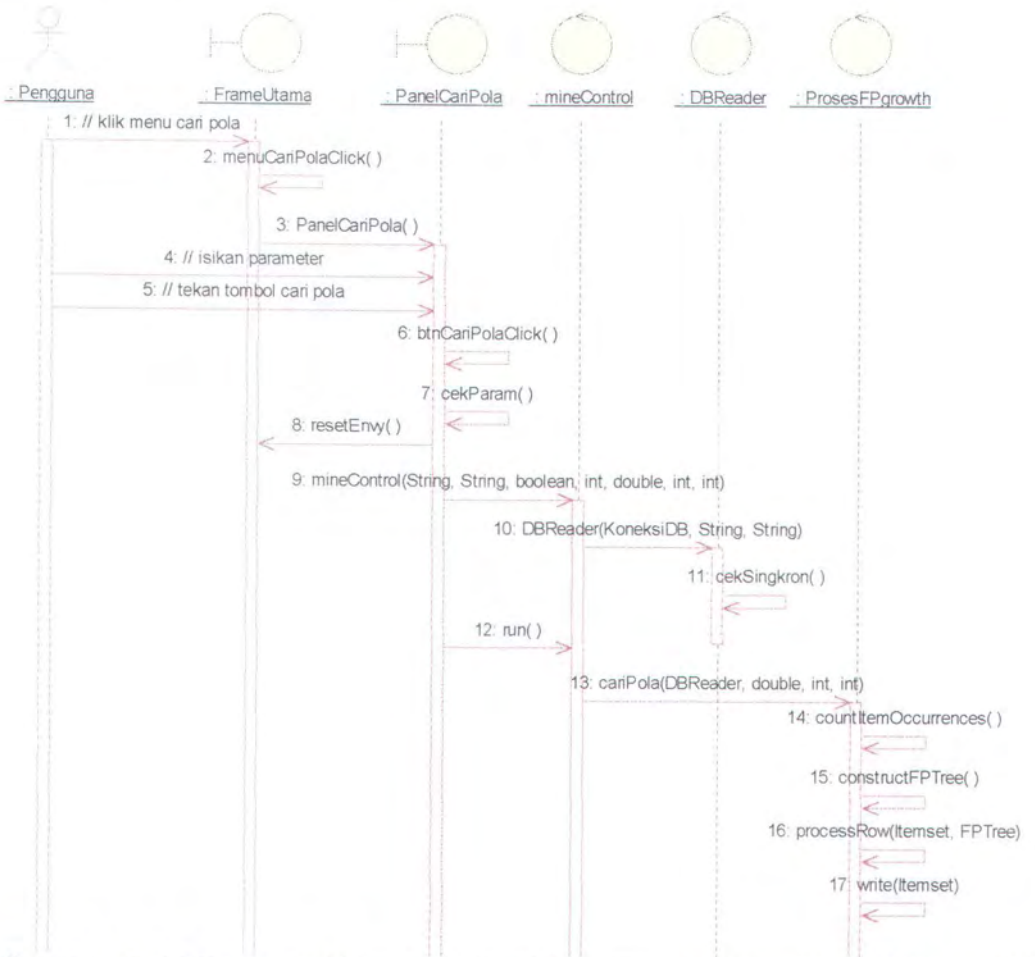
Gambar 3. 24 VOPC "Cari Pola dengan Algoritma Apriori"

3.2.6. Realisasi Use Case "Cari Pola dengan Algoritma FPGrowth"

Dalam *use case* ini, dilakukan pencarian pola dengan mengikuti prinsip kerja algoritma FPGrowth berdasarkan batasan dari pengguna. Untuk melakukannya, aplikasi memanfaatkan sebuah kelas utilitas yaitu :

- FPTree
  - Kelas ini digunakan untuk membuat representasi dataset dalam bentuk *tree*

Aliran kejadian dari *use case* ini dapat digambarkan dengan diagram sekuensi pada gambar dibawah ini :



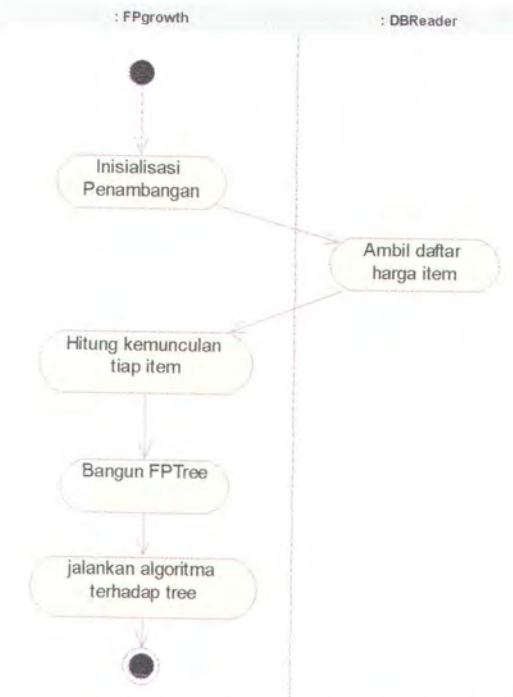
Gambar 3. 25 Diagram Sekuensi "Cari Pola dengan Algoritma FPGrowth"

3.2.6.2. Aktivitas

Untuk aktivitas nomor 1) sampai dengan 12) dari sekuensi model FPGrowth ini adalah serupa dengan aktivitas pada sekuensi model Apriori pada sub bab 3.2.5 di atas. Oleh karena itu yang perlu dijelaskan pada sub bab ini adalah aktivitas selanjutnya yaitu :

13) cariPola(DBReader, double, int, int)) (Pengguna memilih algoritma FPGrowth)

- MineControl memanggil kelas FPGrowth untuk melakukan proses penambangan
- Proses penambangan yang dilakukan oleh operasi ini dapat digambarkan pada diagram aktivitas dibawah ini :



Gambar 3. 26 Operasi cariPola() algoritma FPGrowth

14) countItemOccurrences( )

- Penghitungan jumlah kemunculan untuk tiap distinct item yang muncul pada tabel transaksi

15) constructFPTree( )

- Pembangunan tree dengan cara menelusuri tiap transaksi dan memasukkannya ke dalam tree berdasarkan urutan kemunculannya

16) processRow(Itemset, FPTree)

- Pencarian pola dengan cara menelusuri dahan-dahan pada FPTree

17) write(Itemset)

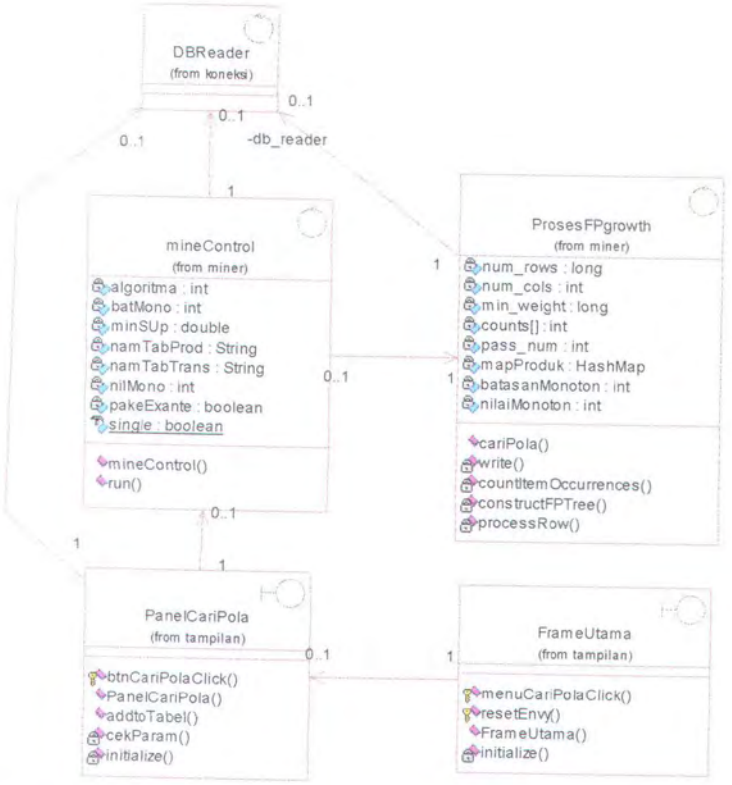
- Operasi ini menuliskan pola yang lolos batasan ke tabel hasil pada PanelCariPola

3.2.6.3. Gambaran Partisipasi Objek

Proses dalam *use case* "Cari Pola dengan Algoritma FPGrowth" dibangun dengan melibatkan beberapa objek sebagai berikut :

- FrameUtama
- PanelCariPola
- MineControl
- ProsesFPGrowth
- DBReader

Hubungan partisipasi objek-objek di atas dapat digambarkan dengan menggunakan diagram VOPC (*View of Participated Class*) di bawah ini :



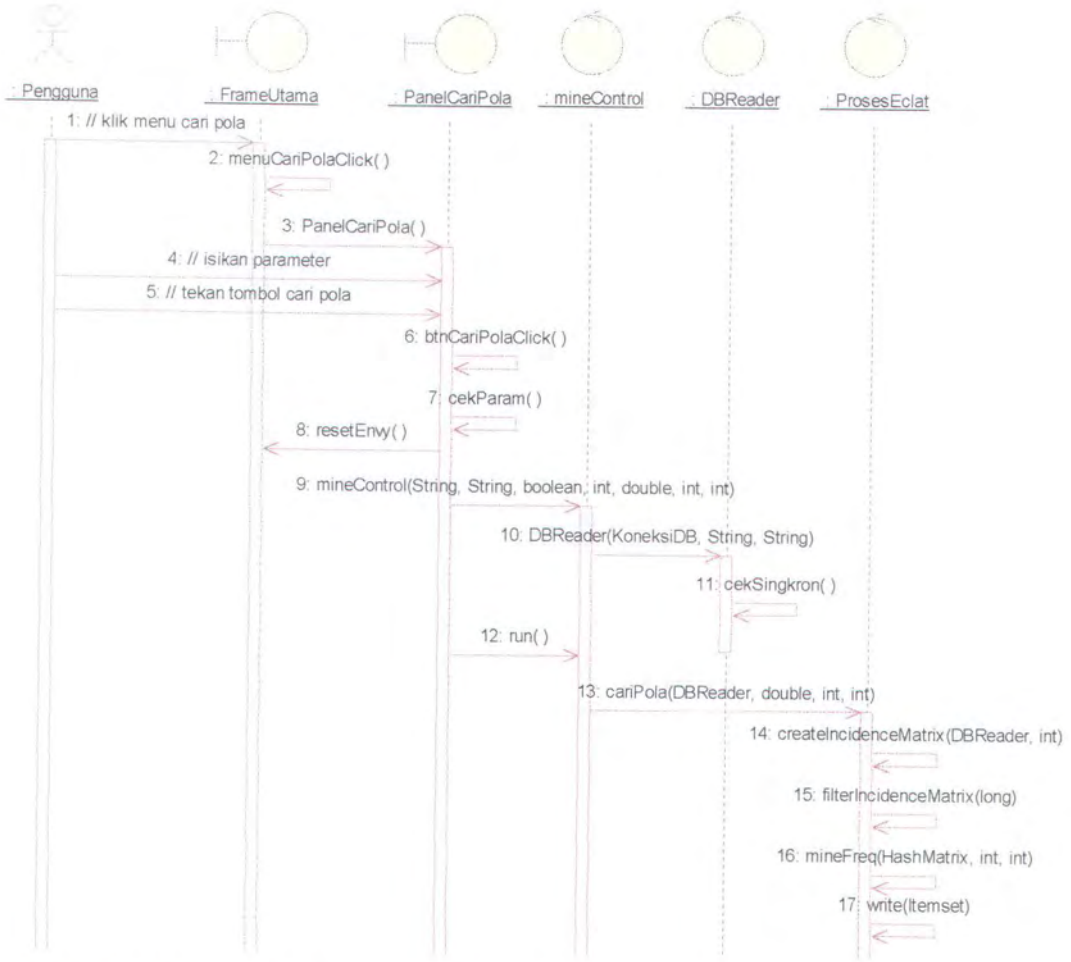
Gambar 3. 27 VOPC "Cari Pola dengan Algoritma FPGrowth"

3.2.7. Realisasi Use Case "Cari Pola dengan Algoritma Eclat"

Dalam *use case* ini, dilakukan pencarian pola dengan mengikuti prinsip kerja algoritma Eclat berdasarkan batasan dari pengguna. Untuk melakukannya, aplikasi memanfaatkan sebuah kelas utilitas yaitu :

- HashMatrix
  - Kelas ini digunakan untuk merepresentasikan dataset ke dalam bentuk sebuah matrik

Aliran kejadian dari *use case* ini dapat digambarkan dengan diagram sekuensi pada gambar 3.28



Gambar 3. 28 Diagram Sekuensi "Cari Pola dengan Algoritma Eclat"

3.2.7.2. Aktivitas

Untuk aktivitas nomor 1) sampai dengan 12) dari sekuensi model Eclat ini adalah serupa dengan aktivitas pada sekuensi model Apriori pada sub bab III.2.5 di atas. Oleh karena itu yang perlu dijelaskan pada sub bab ini adalah aktivitas selanjutnya.

13) cariPola(DBReader, double, int, int)) (Apabila pengguna memilih algoritma Eclat)

- MineControl memanggil kelas Eclat untuk melakukan proses penambangan
- Proses penambangan yang dilakukan oleh operasi ini dapat digambarkan pada diagram aktivitas pada gambar 3.29:

Gambar 3. 29 Operasi cariPola() algoritma Eclat

14) createIncidenceMatrix(DBReader, int)

- Pembangunan incidence matrix dengan cara menelusuri tiap transaksi pada tabel transaksi

15) filterIncidenceMatrix(long)

- Pembuangan baris-baris yang berukuran kurang dari minimum support

16) mineFreq(HashMatrix, int, int)

- Penambangan matrik dengan cara mengawinkan tiap baris matrik.

17) write(Itemset)

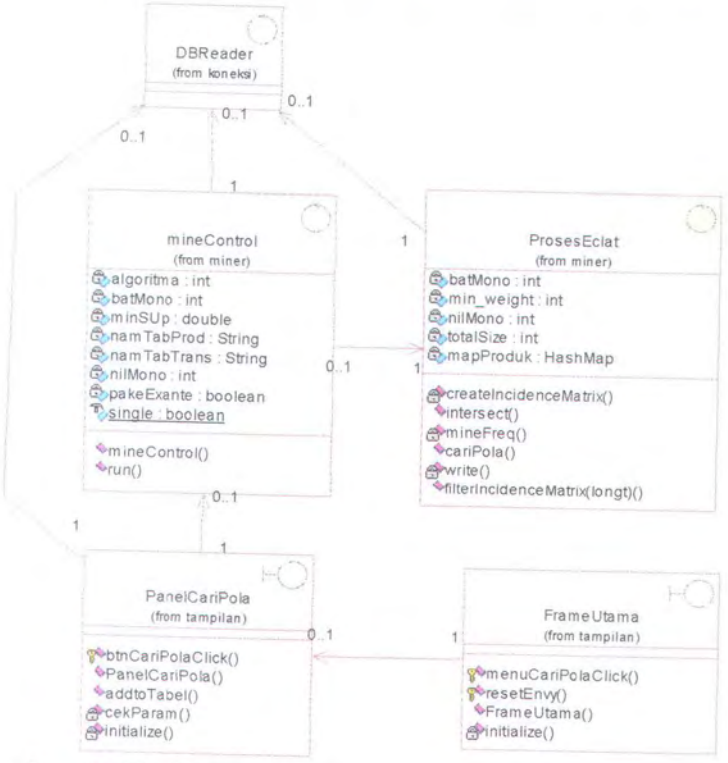
- Operasi ini menuliskan pola yang lolos batasan ke tabel hasil pada PanelCariPola

3.2.7.3. Gambaran Partisipasi Objek

Proses dalam *use case* "Cari Pola dengan Algoritma Eclat" dibangun dengan melibatkan beberapa objek sebagai berikut :

- FrameUtama
- PanelCariPola
- MineControl
- ProsesEclat
- DBReader

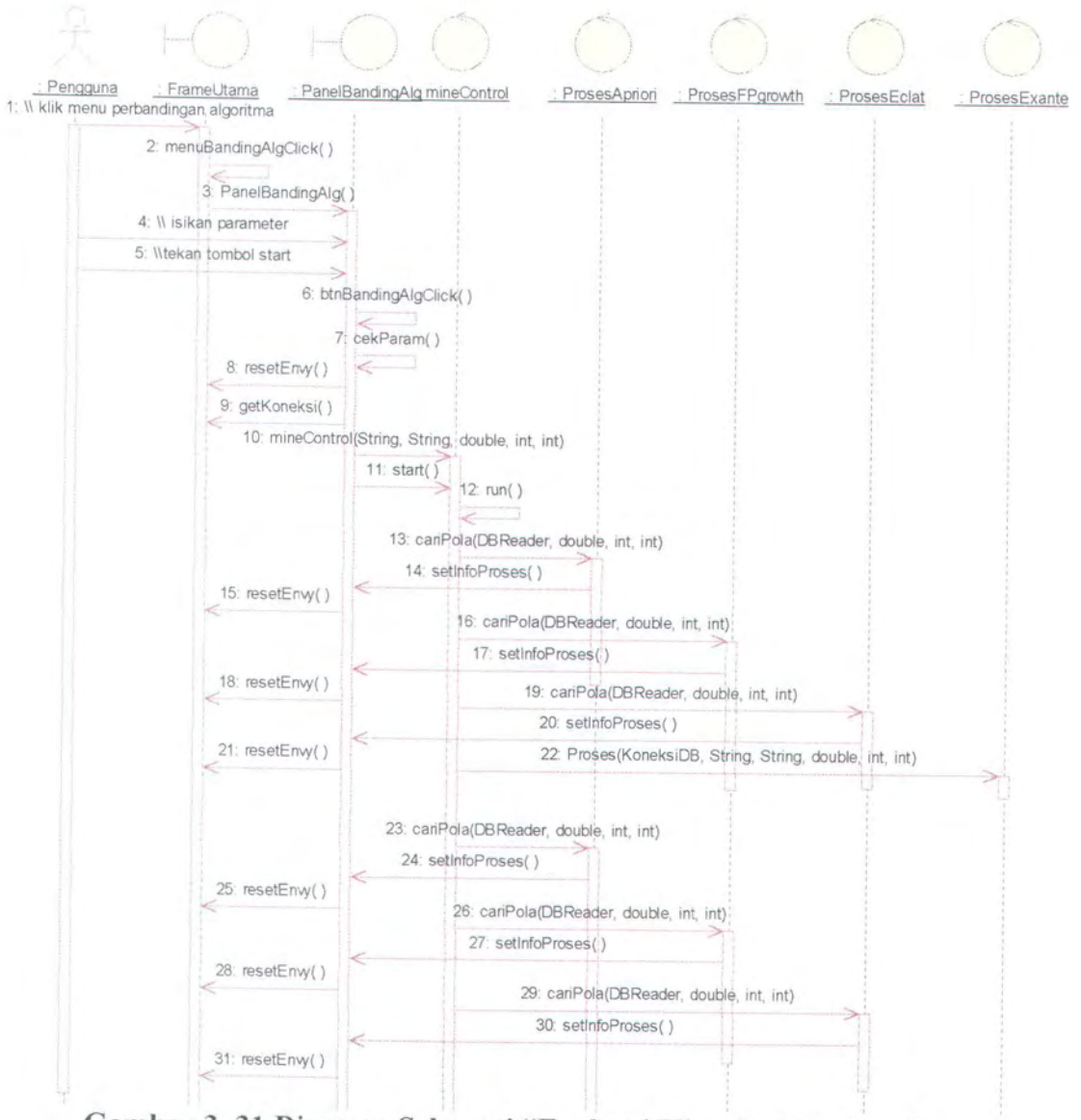
Hubungan partisipasi objek-objek di atas dapat digambarkan dengan menggunakan diagram VOPC (*View of Participated Class*) di bawah ini :



Gambar 3. 30 VOPC "Cari Pola dengan Algoritma Eclat"

3.2.8. Realisasi Use Case "Evaluasi Kinerja Algoritma"

Dalam *use case* ini, dilakukan pembandingan kinerja algoritma-algoritma penambangan. Aliran kejadian dari *use case* ini dapat digambarkan dengan diagram sekuensi pada gambar dibawah ini :



Gambar 3. 31 Diagram Sekuensi "Evaluasi Kinerja Algoritma"

3.2.8.2. Aktivitas

1) \\ klik menu perbandingan algoritma

   - Langkah pertama dalam melakukan proses perbandingan adalah dengan cara memanggil terlebih dahulu panel perbandingan algoritma dengan menekan menu "Cari Pola"

2) menuBandingAlgClick( )

   - Operasi ini dilakukan ketika pengguna menekan menu "Perbandingan Algoritma"

3) PanelBandingAlg( )

   - Inisialisasi PanelBandingAlg dengan komponen-komponen pembangun antarmuka.

4) \\ isikan parameter

   - Pengisian parameter-parameter yang diperlukan dalam penambangan pola.

5) \\tekan tombol start

   - Setelah selesai mengisikan parameter, pengguna harus menekan tombol "Start" untuk memulai proses pembandingan

6) btnBandingAlgClick( )

   - Operasi ini dipanggil ketika terjadi penekanan tombol "Start" sebagai tanda dimulainya proses pembandingan

7) cekParam( )

   - Operasi ini dilakukan untuk melakukan pengecekan prameter apakah sudah lengkap dan benar atau tidak

8) resetEnvy( )

   - Operasi ini mengembalikan nilai memori maksimal ke nol, serta membebaskan memori yang tak terpakai melalui garbage colector.

9) getKoneksi( )

- Operasi ini memanggil nilai koneksi dari FrameUtama untuk digunakan dalam proses penambangan

10) mineControl(String, String, double, int, int)

- Operasi ini menginisialisasikan penambangan dengan mengirimkan parameter dan melakukan proses penambangan berdasarkan kombinasi algoritma yang mungkin.

11) start( )

- Operasi ini dipanggil untuk menandakan dimulainya proses perbandingan

12) Run()

- Merupakan method untuk memulai sebuah thread baru penambangan.

13) cariPola(DBReader, double, int, int)

- Operasi ini dimiliki oleh tiap algoritma penambangan. Tiap algoritma tersebut menjalankan operasi ini berdasarkan definisi prosesnya masing-masing.

14) setInfoProses( )

- Operasi ini berfungsi untuk melemparkan kembali informasi proses yang telah dilakukan oleh algoritma untuk ditampilkan pada PanelBandingAlg

15) Proses(KoneksiDB, String, String, double, int, int)

- Operasi ini berfungsi untuk menjalankan metode ExAnte pada dataset transaksi

16) ResetEnvy()

- Operasi ini melakukan pembersihan lingkungan berupa pembebasan memori yang tidak digunakan.

17) CariPola()

- Operasi ini melakukan proses penambangan pola dengan algoritma Apriori

18) setInfoProses()

- Operasi ini mengembalikan informasi proses penambangan yang telah dilakukan untuk ditampilkan kepada pengguna

19) ResetEnvy()

- Operasi ini melakukan pembersihan lingkungan berupa pembebasan memori yang tidak digunakan.

20) SetInfoProses()

- Operasi ini mengembalikan informasi proses penambangan yang telah dilakukan untuk ditampilkan kepada pengguna

21) ResetEnvy()

- Operasi ini melakukan pembersihan lingkungan berupa pembebasan memori yang tidak digunakan.

22) Proses()

- Operasi untuk memulai proses reduksi data dengan menggunakan metode ExAnte

23) cariPola()

- Operasi untuk melakukan penambangan dngan menggunakan algoritma apriori lagi namun dengan dataset awal berupa dataset yang telah direduksi dengan metode ExAnte.

24) setInfoProses()

- Operasi ini mengembalikan informasi proses penambangan yang telah dilakukan untuk ditampilkan kepada pengguna

25) ResetEnvy()

- Operasi ini melakukan pembersihan lingkungan berupa pembebasan memori yang tidak digunakan.

26) cariPola()

- Operasi untuk melakukan penambangan dngan menggunakan algoritma FPGrowth lagi namun dengan dataset awal berupa dataset yang telah direduksi dengan metode ExAnte.

27) setInfoProses()

- Operasi ini mengembalikan informasi proses penambangan yang telah dilakukan untuk ditampilkan kepada pengguna

28) ResetEnvy()

- Operasi ini melakukan pembersihan lingkungan berupa pembebasan memori yang tidak digunakan.

29) cariPola()

- Operasi untuk melakukan penambangan dngan menggunakan algoritma Eclat lagi namun dengan dataset awal berupa dataset yang telah direduksi dengan metode ExAnte.

30) setInfoProses()

- Operasi ini mengembalikan informasi proses penambangan yang telah dilakukan untuk ditampilkan kepada pengguna

31) ResetEnvy()

- Operasi ini melakukan pembersihan lingkungan berupa pembebasan memori yang tidak digunakan.

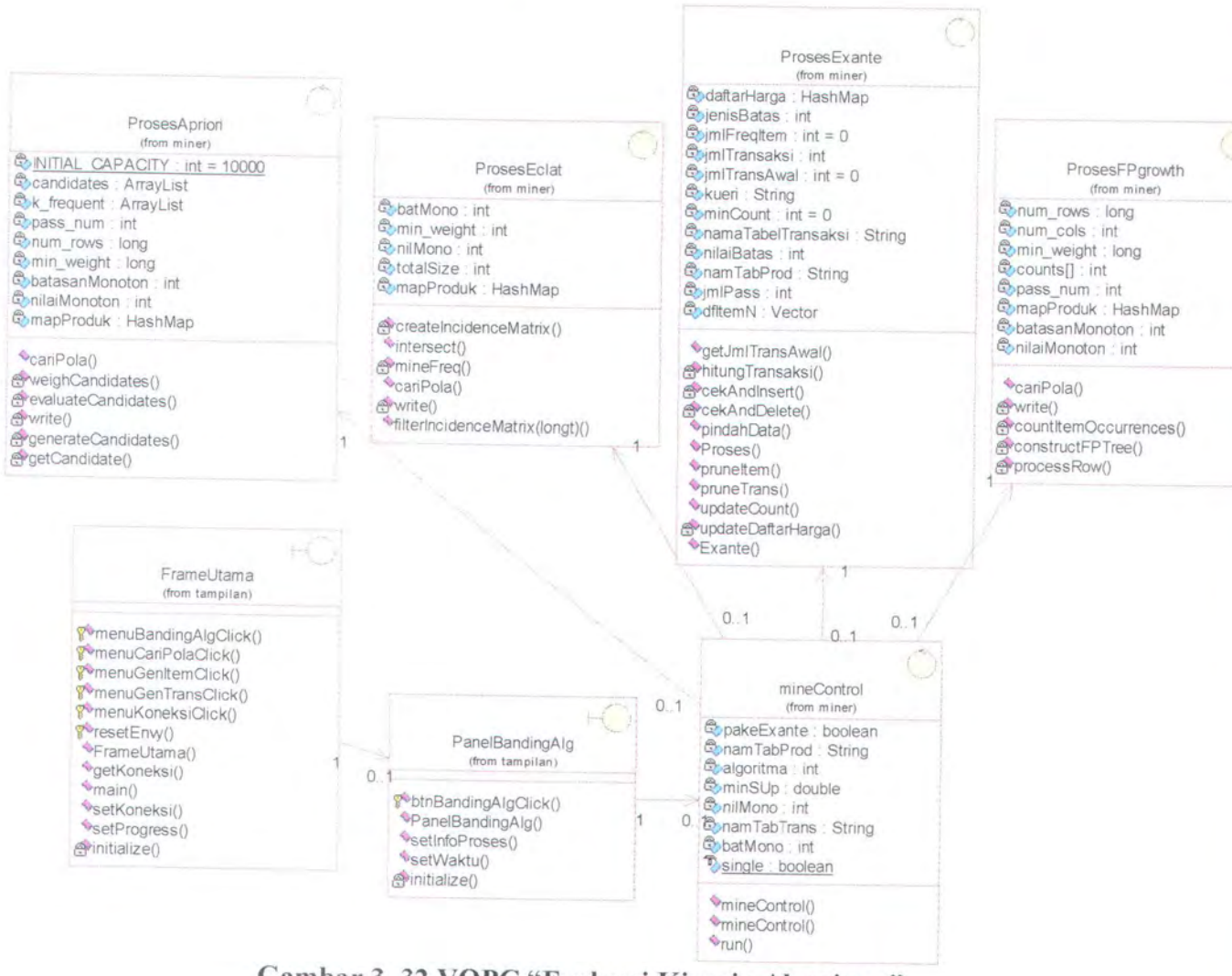
3.2.8.3. Gambaran Partisipasi Objek

Proses dalam *use case* "Evaluasi Kinerja Algoritma" dibangun dengan melibatkan beberapa objek sebagai berikut :

- FrameUtama

- PanelCariPola
- MineControl
- ProsesExAnte
- ProsesApriori
- ProsesFPGrowth
- ProsesEclat

Hubungan partisipasi dari objek-objek yang terlibat pada *use case* ini, dapat digambarkan dengan menggunakan diagram VOPC (*View of Participated Class*), sebgaimana ditampilkan pada gambar 3.32



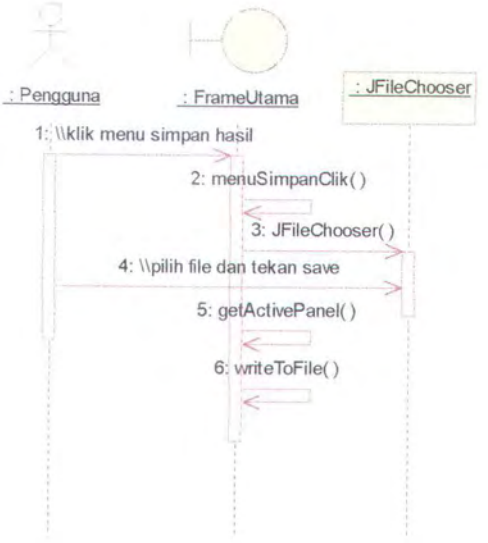
Gambar 3. 32 VOPC "Evaluasi Kinerja Algoritma"

3.2.9. Realisasi Use Case "Simpan Hasil"

Dalam *use case* ini, dilakukan penyimpanan hasil proses yang telah dilakukan kedalam bentuk sebuah file. Untuk melakukannya, aplikasi memanfaatkan dua buah kelas utilitas yaitu :

- JFileChooser
  - Kelas ini digunakan untuk memilih file yang akan digunakan dalam penyimpanan
- FileWriter
  - Kelas ini digunakan untuk menuliskan informasi ke dalam file

Aliran kejadian dari *use case* ini dapat digambarkan dengan diagram sekuensi pada gambar dibawah ini :



Gambar 3. 33 Diagram Sekuensi "Simpan Hasil"

3.2.9.2. Aktivitas

1) \\klik menu simpan hasil

- Langkah pertama dalam melakukan proses penyimpanan hasil adalah dengan cara memanggil terlebih dahulu panel penyimpanan dengan menekan menu "Simpan"

2) menuSimpanClik( )

- Operasi ini dilakukan ketika pengguna menekan menu "Simpan"

3) JFileChooser( )

- Operasi ini digunakan untuk memanggil kelas JFileChooser yang berfungsi sebagai tempat pengguna menentukan file tujuan

4) \\pilih file dan tekan save

- Setelah pengguna menentukan file tujuan, pengguna menekan tombol "Save" sebagai tanda bagi aplikasi untuk memulai proses penyimpanan

5) getActivePanel( )

- Operasi ini berfungsi untuk menentukan hasil proses mana yang ingin disimpan oleh pengguna. Penentuan hasil proses ini dilakukan dengan cara melihat panel manakah yang aktif pada saat pengguna menekan menu "Simpan"

6) writeToFile( )

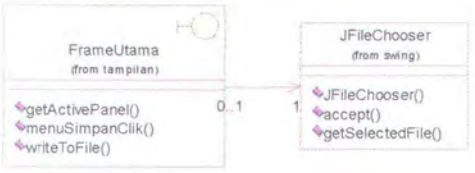
- Operasi ini berfungsi untuk menuliskan hasil ke file tujuan

3.2.9.3. Gambaran partisipasi objek

Proses dalam *use case* "Simpan Hasil" dibangun dengan melibatkan beberapa objek sebagai berikut :

- FrameUtama
- JFileChooser

Hubungan partisipasi objek-objek di atas dapat digambarkan dengan menggunakan diagram VOPC (*View of Participated Class*) di bawah ini :



Gambar 3. 34 VOPC "Simpan Hasil"

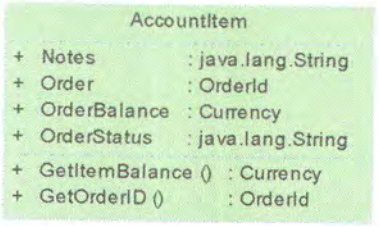
3.3. Analisa Kelas Global

Kelas adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Kelas menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). Diagram kelas menggambarkan struktur dan deskripsi kelas, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Kelas memiliki tiga area pokok :

1. Nama (dan *stereotype*)
2. Atribut
3. Metoda

Gambar di bawah ini merupakan contoh dari sebuah kelas:



Gambar 3. 35 Contoh Kelas

Atribut dan metoda dapat memiliki salah satu sifat berikut :

- *Private*, tidak dapat dipanggil dari luar kelas yang bersangkutan.
- Protected, hanya dapat dipanggil oleh kelas yang bersangkutan dan anak-anak yang mewarisinya.
- Public, dapat dipanggil oleh siapa saja.

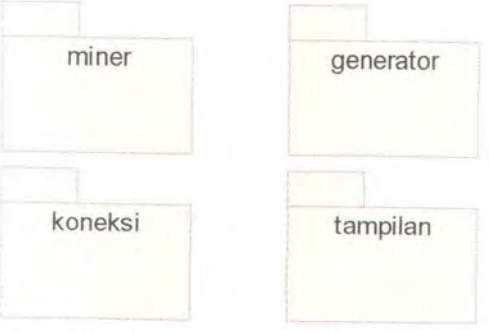
Kelas dapat merupakan implementasi dari sebuah interface, yaitu kelas abstrak yang hanya memiliki metoda. Interface tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan terlebih dahulu menjadi sebuah kelas. Dengan demikian interface mendukung resolusi metoda pada saat run-time.

Sesuai dengan perkembangan model kelas, kelas dapat dikelompokkan menjadi *package*. Kita juga dapat membuat diagram yang terdiri atas *package*.

Hubungan Antar Kelas

- Asosiasi, yaitu hubungan statis antar kelas. Umumnya menggambarkan kelas yang memiliki atribut berupa kelas lain, atau kelas yang harus mengetahui eksistensi kelas lain. Panah navigability menunjukkan arah query antar-kelas.
- Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas..").
- Pewarisan, yaitu hubungan hirarkis antar kelas. Kelas dapat diturunkan dari kelas lain dan mewarisi semua atribut dan metoda kelas asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari kelas yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
- Hubungan dinamis, yaitu rangkaian pesan (message) yang di-passing dari satu kelas kepada kelas lain. Hubungan dinamis dapat digambarkan dengan menggunakan diagram sekuensi.

Sehubungan dengan desain proses aplikasi ini, aplikasi dibagi menjadi 4 package utama. Tiap-tiap package ini, dibedakan berdasarkan fungsionalitas dari masing-masing package. Package-package ini ditunjukkan pada gambar 3.36.



Gambar 3. 36 Package Aplikasi Penambangan Pola

Tiap-tiap pakage diatas berisi kelas-kelas yang memiliki fungsi sesuai dengan fungsionalitas package. Penjelasan dari tiap package adalah sebagai berikut :

3.3.2. Package Tampilan

Package ini mengandung beberapa kelas yang mengimplementasikan antarmuka pengguna dengan menggunakan Javax.swing. Secara umum package ini tidak memiliki fungsi penambangan apapun, namun hanya menyediakan berbagai tampilan yang dapat memberikan nilai usabilitas yang baik. Beberapa kelas yang dikandung oleh pakage ini dapat dijabarkan tabel di bawah ini :

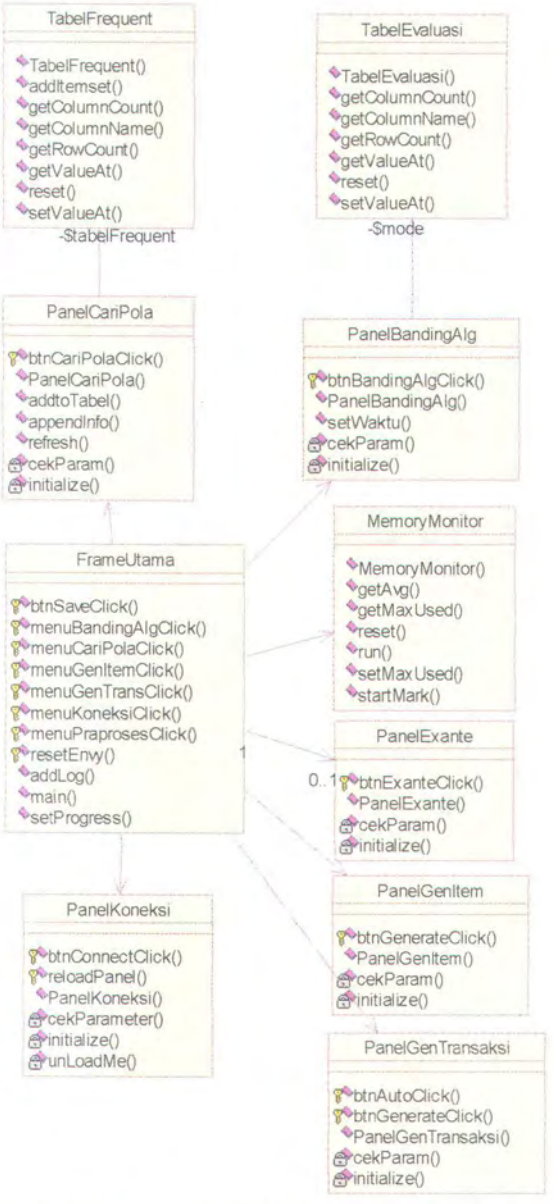
Tabel 3. 10 Kelas pada Package Tampilan

Nama kelas	Definisi
FrameUtama	Merupakan kelas yang berfungsi sebagai penghubung aplikasi dengan pengguna. Berisi menu-menu yang menghubungkan pengguna dengan fitur-fitur aplikasi. Kelas ini juga berfungsi sebagai penampung luaran dari kelas lain seperti informasi memori dari kelas MemoryMonitor serta koneksi dari kelas KoneksiDB.
PanelKoneksi	Merupakan kelas yang berfungsi untuk menghubungkan pengguna dengan kelas KoneksiDB yaitu membuat koneksi dengan basis data.
PanelGenTransaksi	Merupakan kelas penghubung dengan kelas generator. Bertugas mengirimkan parameter pembangkitan transaksi sintetis dari pengguna kepada kelas GeneratorTransaksi.
PanelGenItem	Merupakan kelas yang menghubungkan dengan kelas GeneratorItem. Bertugas mengirimkan parameter pembangkitan item sintetis dari pengguna kepada kelas GeneratorItem.
PanelExante	Merupakan kelas yang berfungsi untuk memanggil fungsi Praproses berdasarkan metode Exante dengan parameter yang telah ditentukan oleh pengguna.
PanelCariPola	Merupakan kelas yang berfungsi untuk memanggil fungsi penambangan berdasarkan algoritma dan parameter lain yang telah ditentukan oleh pengguna.
PanelBandingAlg	Merupakan kelas yang berfungsi untuk memanggil fungsi penambangan dengan menerapkan seluruh variasi algoritma yang ada berdasarkan parameter yang telah ditentukan oleh pengguna.

Tabel 3.10. Kelas pada Package Tampilan (lanjutan)

Nama kelas	Definisi
MemoryMonitor	Merupakan kelas yang berfungsi untuk menampilkan informasi memory yang digunakan oleh aplikasi kepada pengguna. Kelas ini akan terus mengirimkan informasi memori kepada pengguna selama aplikasi aktif.

Hubungan antar kelas di atas dapat digambarkan pada diagram kelas gambar 3.37.



Gambar 3. 37 Diagram Kelas Package Tampilan

3.3.3. Package koneksi

Package koneksi merupakan package yang berisi kelas-kelas yang berfungsi menghubungkan aplikasi dengan basis data transaksi. Kelas yang ada dalam package ini juga berfungsi untuk menerjemahkan struktur data masukan menjadi struktur data proses yang dapat dimengerti oleh aplikasi. Dalam package ini terdapat dua buah kelas yang dapat dijabarkan dengan tabel 3.11.

Tabel 3. 11 Kelas pada Package Koneksi

Nama kelas	Definisi
KoneksiDB	Berfungsi untuk menghubungkan aplikasi dengan basis data serta melakukan berbagai macam operasi kueri pada basis data.
DBReader	Berfungsi untuk mengkonversi bentuk transaksi menjadi struktur Itemset serta mengambil parameter-parameter yang dibutuhkan dalam proses penambangan.

Hubungan antarkelas di atas dapat digambarkan dengan diagram kelas di bawah ini :

Gambar 3. 38 Diagram Kelas Package Koneksi

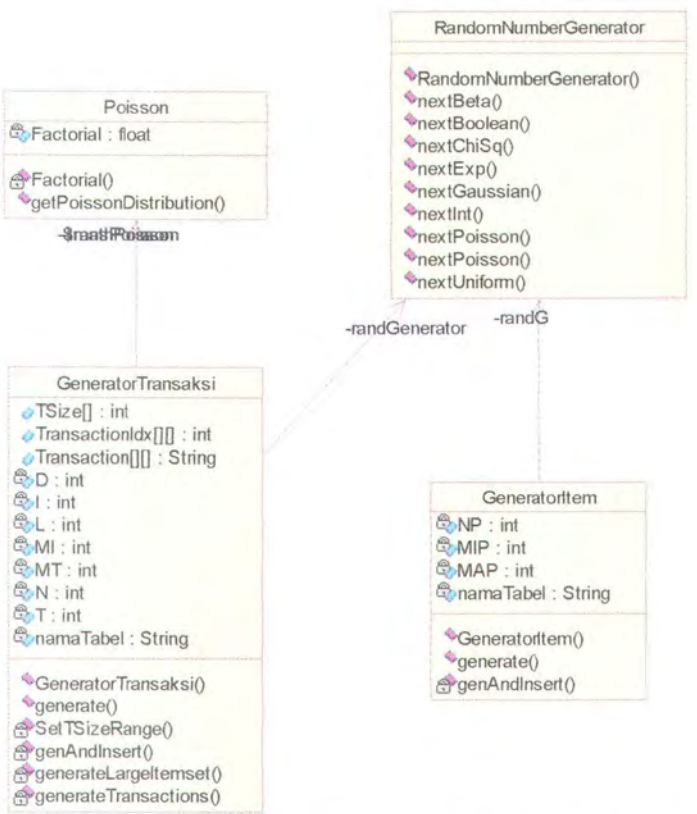
3.3.4. Package Generator

Package generator merupakan package kelas-kelas yang berfungsi untuk membangkitkan data sintetis. Pembangkitan data yang digunakan dalam aplikasi ada dua, yaitu pembangkitan data transaksi dan pembangkitan data item. Kelas-kelas yang berperan dalam proses ini dapat dijabarkan dengan tabel 3.12.

Tabel 3. 12 Kelas pada Package Generator

Nama kelas	Definisi
GeneratorTransaksi	Merupakan kelas yang bertanggung jawab untuk membangkitkan transaksi berdasarkan masukan parameter pengguna dan mengirimkannya ke basis data
GeneratorItem	Merupakan kelas yang bertanggung jawab untuk membangkitkan item berdasarkan masukan parameter pengguna dan mengirimkannya ke basis data
Poisson	Kelas ini berfungsi untuk menghasilkan nilai distribusi poisson sesuai dengan parameter yang dimasukkan.
RandomNumberGenerator	Kelas ini berfungsi untuk menghasilkan angka-angka acak dengan berbagai pilihan distribusi. Angka-angka acak inilah yang digunakan dalam membangkitkan dataset sintetis.

Hubungan antarkelas di atas dapat digambarkan dengan diagram kelas di bawah ini :



Gambar 3. 39 Diagram Kelas Package Generator

3.3.5. Package Miner

Package miner berperan dalam menjalankan proses penambangan pola. Dalam package ini terdapat kelas yang menunjang algoritma-algoritma penambangan. Kelas- kelas tersebut dapat dijabarkan dengan menggunakan tabel di bawah ini :

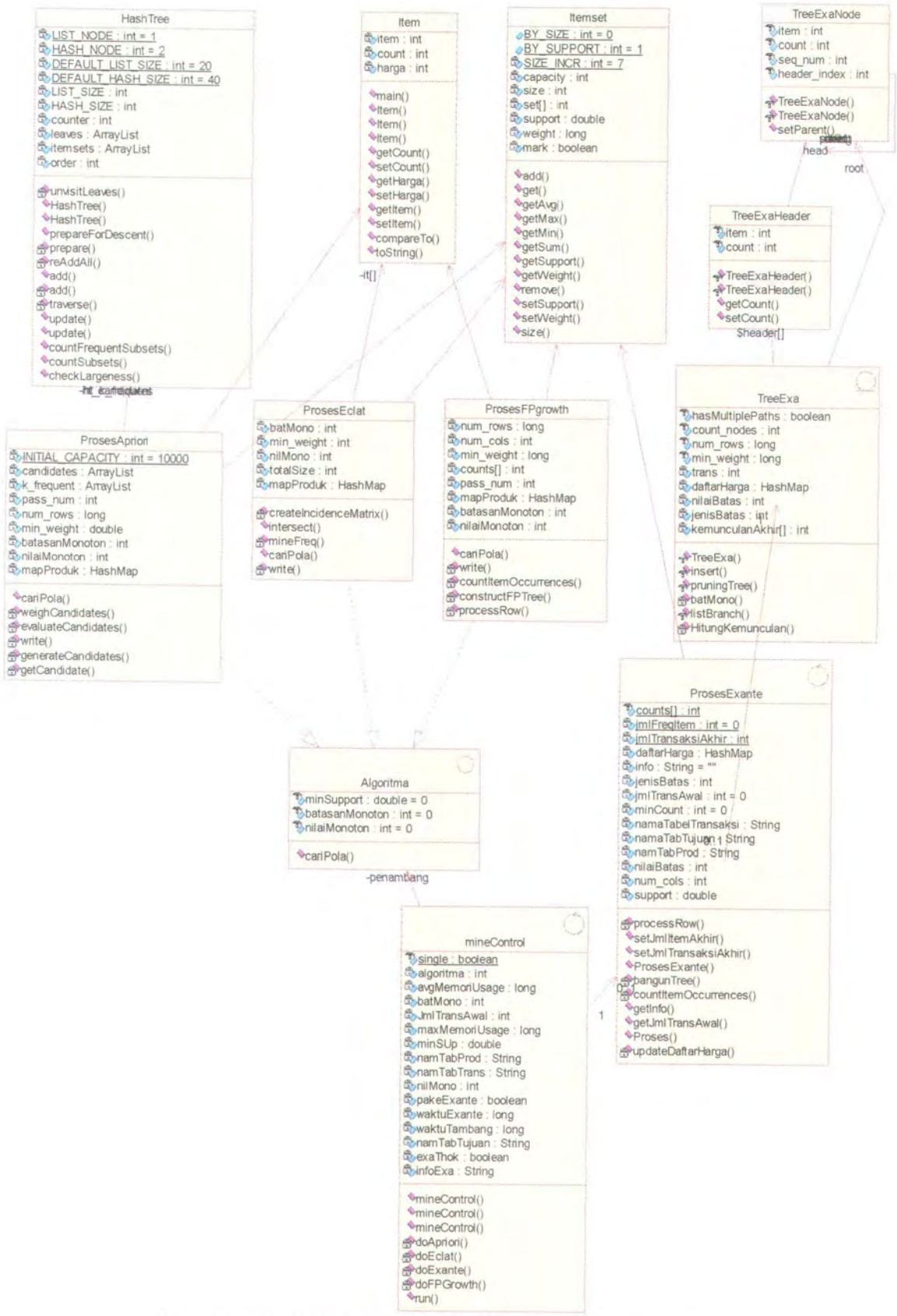
Tabel 3. 13 Kelas pada Package Miner

Nama kelas	Definisi
MineControl	Kelas ini merupakan kelas yang pertama kali dipanggil apabila pengguna ingin melakukan pencarian pola. Kelas inilah yang menciptakan sebuah Thread serta memanggil kelas-kelas lain yang dibutuhkan untuk penambangan berdasarkan keinginan pengguna.
Algoritma	Kelas ini berupa sebuah Interface yang memiliki sifat cari pola. Sifat kelas ini diwariskan ke kelas Apriori, FPGrowth dan Eclat.
ProsesApriori	Merupakan kelas yang menerapkan algoritma Apriori. Kelas ini dipanggil untuk mencari semua pola yang telah memenuhi batasan yang telah ditentukan. Untuk mempercepat proses, kelas ini memakai struktur data HashTree untuk membuat index dari transaksi.
HashTree	Kelas HashTree merupakan kelas penunjang dari Apriori. Menggunakan struktur tree untuk membuat index transaksi.
ProsesFPGrowth	Merupakan kelas yang menerapkan algoritma FPGrowth. Kelas ini dipanggil untuk mencari semua pola yang telah memenuhi batasan yang telah ditentukan. Kelas ini memiliki inner class yang berupa FPTree, yaitu struktur data yang merepresentasikan dataset transaksi kedalam bentuk pohon.
ProsesEclat	Merupakan kelas yang menerapkan algoritma Eclat. Kelas ini dipanggil untuk mencari semua pola yang telah memenuhi batasan yang telah ditentukan. Kelas ini memiliki inner class yang berupa HashMatrix, yaitu struktur data yang merepresentasikan dataset transaksi kedalam bentuk matrik.
ProsesExAnte	Merupakan kelas yang menerapkan algoritma ExAnte yang berfungsi untuk mereduksi ukuran dataset berdasarkan batasan yang telah ditentukan.

Tabel 3.13. Kelas pada Package Miner (lanjutan)

Nama kelas	Definisi
TreeExa	Kelas TreeExa merupakan kelas penunjang dari *ExAnte*. Menggunakan struktur tree untuk membuat representasi dataset yang ingin direduksi.
Itemset	Merupakan kelas yang digunakan untuk menampung item-item pada transaksi beserta atribut transaksi yang menunjang dalam penambangan pola seperti support, weight dan size.
Item	Merupakan kelas yang melambangkan distinct item yang ada pada dataset. Kelas ini menyimpan informasi tentang jumlah kemunculan item pada dataset serta informasi price dari item yang bersangkutan.

Hubungan antarkelas di atas dapat digambarkan dengan diagram kelas pada gambar 3.40.



Gambar 3. 40 Diagram Kelas Package Miner

3.4. Desain Antarmuka

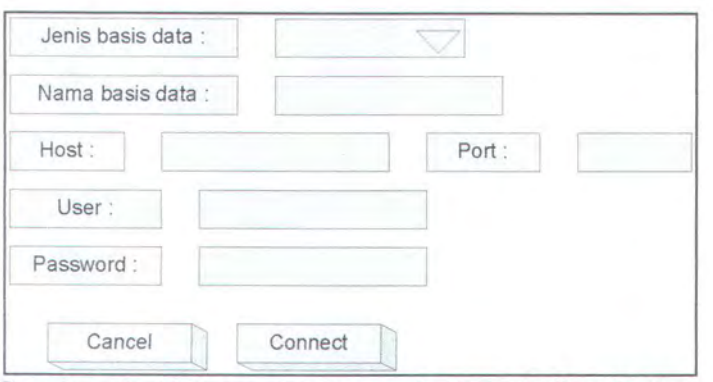
Antarmuka merupakan penghubung antara proses aplikasi dengan pengguna. Desain antarmuka aplikasi ini dibuat untuk memberikan gambaran layout antarmuka aplikasi berdasarkan masing-masing fitur realisasi usecase.

3.4.1. Antarmuka Koneksi basis data

Komponen-komponen yang dibutuhkan dalam menerima masukan parameter dari pengguna untuk fitur ini antara lain :

- 1 combo box yang berisi pilihan jenis basis data yang ingin digunakan
- 1 text field untuk masukan nama basis data
- 1 text field untuk masukan nama alamat server basis data
- 1 text field untuk masukan port yang digunakan
- 1 text field untuk masukan nama pengguna
- 1 password field untuk masukan password pengguna
- 1 text field untuk masukan nama basis data apabila pengguna menggunakan file Access

Desain layout untuk menampung komponen–komponen antarmuka di atas adalah sebagai berikut :



Gambar 3. 41 Layout antarmuka koneksi basis data

3.4.2. Antarmuka Pembangkitan transaksi

Komponen-komponen yang dibutuhkan dalam menerima masukan parameter dari pengguna untuk fitur ini antara lain :

- 1 text field untuk masukan Jumlah item (N)
- 1 text field untuk masukan Rata-rata ukuran Large Itemset (I)
- 1 text field untuk masukan Ukuran maksimum Large Itemset (MI)
- 1 text field untuk masukan Jumlah Large Itemset (L)
- 1 text field untuk masukan Rata-rata ukuran Transaksi (T)
- 1 text field untuk masukan Ukuran maksimum Transaksi (MT)
- 1 text field untuk masukan Jumlah Transaksi (D)
- 1 text field untuk masukan Nama Tabel Transaksi

Desain layout untuk menampung komponen–komponen antarmuka di atas adalah sebagai berikut :

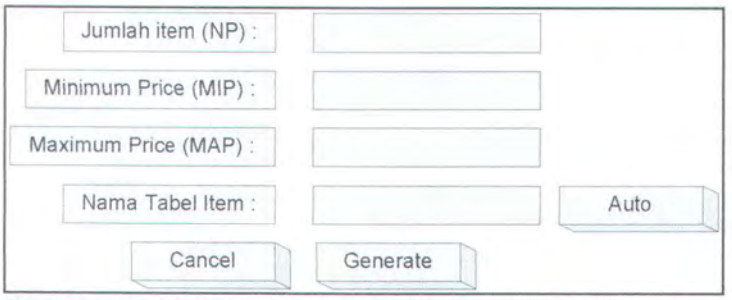
Gambar 3. 42 Layout antarmuka pembangkitan transaksi

3.4.3. Antarmuka Pembangkitan item

Komponen-komponen yang dibutuhkan dalam menerima masukan parameter dari pengguna untuk fitur ini antara lain :

- 1 text field untuk masukan Jumlah Item (NP)
- 1 text field untuk masukan Minimum Price (MIP)
- 1 text field untuk masukan Maximum Price (MAP)
- 1 text field untuk masukan Nama tabel item

Desain layout untuk menampung komponen–komponen antarmuka di atas adalah sebagai berikut :



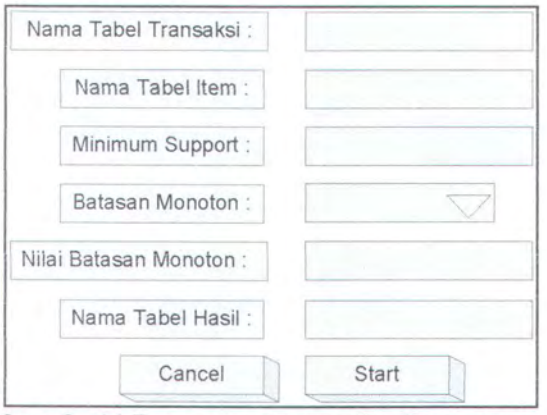
Gambar 3. 43 Layout antarmuka pembangkitan item

3.4.4. Antarmuka Praproses data

Komponen-komponen yang dibutuhkan dalam menerima masukan parameter dari pengguna untuk fitur ini antara lain :

- 1 text field untuk masukan Nama Tabel Transaksi
- 1 text field untuk masukan Nama Tabel Item
- 1 text field untuk masukan Minimum Support
- 1 combo box yang berisi pilihan Batasan Monoton
- 1 text field untuk masukan Niliai Batasan Monoton
- 1 text field untuk masukan Nama table tujuan

Desain layout untuk menampung komponen–komponen antarmuka di atas adalah sebagai berikut :



Gambar 3. 44 Layout antarmuka Praproses data

3.4.5. Antarmuka Pencarian pola

Komponen-komponen yang dibutuhkan dalam menerima masukan parameter dari pengguna untuk fitur ini antara lain :

- 1 text field untuk masukan Nama Tabel Transaksi
- 1 text field untuk masukan Nama Tabel Item
- 2 radio button untuk pilihan penerapan *ExAnte*
- 1 combo box yang berisi pilihan Jenis Algoritma
- 1 text field untuk masukan Minimum Support
- 1 combo bCox yang berisi pilihan Batasan Monoton
- 1 text field untuk masukan Nilai Batasan Monoton

Desain layout untuk menampung komponen–komponen antarmuka di atas adalah sebagai berikut :

Gambar 3. 45 Layout antarmuka pencarian pola

3.4.6. Antarmuka Evaluasi kinerja

Komponen-komponen yang dibutuhkan dalam menerima masukan parameter dari pengguna untuk fitur ini antara lain :

- 1 text field untuk masukan Nama Tabel Transaksi
- 1 text field untuk masukan Nama Tabel Item
- 1 text field untuk masukan Minimum Support
- 1 combo bCox yang berisi pilihan Batasan Monoton
- 1 text field untuk masukan Nilai Batasan Monoton

Desain layout untuk menampung komponen–komponen antarmuka di atas ditunjukkan pada gambar 3.46

Gambar 3. 46 Layout antarmuka Evaluasi kinerja

BAB IV

IMPLEMENTASI APLIKASI

BAB IV

IMPLEMENTASI APLIKASI

Pada bab implementasi ini, dijelaskan mengenai bagaimana cara mengimlementasikan desain yang telah dibahas pada bab III. Implementasi meliputi proses-proses apa saja yang dilakukan oleh aplikasi, langkah-langkah bagaimana aplikasi melakukannya serta komponen-komponen pendukung dalam menjalankan proses tersebut.

4.1. Implementasi Use Case Bangun Koneksi Basis Data

Proses pembangunan koneksi basis data merupakan proses yang menghubungkan aplikasi dengan basis data transaksi. Koneksi basis data menggunakan JDBC API sebagai penghubung dengan database. JDBC API adalah kumpulan kelas dan antarmuka yang ditulis dalam bahasa Java yang menyediakan API (*Application Programming Interface*) standar sebagai alat bantu bagi pembuat program (*developer*) dan memberikan kemungkinan untuk menulis aplikasi database dengan menggunakan semua Java API. JDBC API memudahkan untuk mengirim *statement* SQL ke sistem database relasional dan mendukung bermacam-macam dialek SQL.

Setelah pengguna mengisikan parameter koneksi dan menekan tombol "Connect", aplikasi akan memanggil method `btnConnectClick()` seperti ditunjukkan di bawah ini :

```
1.    protected void btnConnectClick(int i) {
2.     If (cekParameter(i)) {
3.      String jenis = (String) comboJenis
4.       .getSelectedItem();
5.      if (i == 1) {
6.       String nama = tfNama.getText();
7.       String host = tfHost.getText();
8.       String port = tfPort.getText();
9.       String user = tfUser.getText();
10.      String pass = String.valueOf(tfPass
11.       .getPassword());
```

```
12.       koneksi = new KoneksiDB(jenis, nama, host,
13.        port, user, pass);
14.      } else {
15.       String name = tfAccessFile.getText();
16.       String user = tfUserAccess.getText();
17.       String pass = String.valueOf(tfPassAccess
18.        .getPassword());
19.       koneksi = new KoneksiDB(name, user, pass);
20.      }
21.      try {
22.       koneksi.connect();
23.       FrameUtama.setKoneksi(koneksi);
24.       unLoadMe();
25.      } catch (ClassNotFoundException e) {
26.       JOptionPane.showMessageDialog(null, e
27.        .getMessage());
28.       e.printStackTrace();
29.      } catch (SQLException e) {
30.       e.printStackTrace();
31.       JOptionPane.showMessageDialog(null, e
32.        .getMessage());
33.      }
34.     }
35.   }
```

Segmen program 4. 1 Method btnConnectClick()

Method di atas melakukan pengecekan parameter yang dimasukkan oleh pengguna (baris 2). Proses pengecekan ini memeriksa apakah pengguna telah memasukkan parameter dengan lengkap dan valid. Setelah melalui proses pengecekan parameter, proses berikutnya adalah menginisialisasi kelas `KoneksiDB` (baris 19). Kelas ini bertanggung jawab atas pembangunan koneksi dan eksekusi kueri. Setelah itu, method memamnggil method `connect()` pada kelas tersebut.

Komponen yang dibutuhkan dalam koneksi dengan database adalah *driver*. *Driver* berfungsi untuk menghubungkan aplikasi dengan jenis database yang sesuai dengan *driver* tersebut. Oleh karena itu, langkah pertama dalam proses ini adalah mendaftarkan *driver* database ke sistem. Setelah *driver* terdaftar, proses selanjutnya adalah membuka koneksi dengan database tersebut. Caranya dengan menggunakan objek `Connection` dari `java.sql.Connection`. Pembangunan koneksi ini memerlukan penyertaan URL, username dan password.

URL merupakan alamat tempat server database berada. Setelah koneksi berhasil dibangun, berikutnya adalah pembuatan *statement*. Objek `statement` digunakan untuk mengirimkan *statement* SQL ke database. Segmen program 4.2 menunjukkan kode pembuatan koneksi dengan server basis data.

```
1.    public void connect()
2.      throws ClassNotFoundException, SQLException {
3.      if (jenis.equals("Access")) {
4.       Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
5.       this.url = "jdbc:odbc:Driver={" +
6.            "Microsoft Access Driver (*.mdb)};DBQ=";
7.       this.url += fileName.trim() + ";DriverID=22}";
8.      } else if (jenis.equals("MySql")) {
9.       Class.forName("org.gjt.mm.mysql.Driver");
10.      this.url = "jdbc:mysql://" + host + ":" + port
11.       + "/" + nama;
12.     } else if (jenis.equals("Postgre SQL")) {
13.      Class.forName("org.postgresql.Driver");
14.      this.url = "jdbc:postgresql://" + host + ":"
15.       + port + "/" + nama;
16.     } else if (jenis.equals("SQL Server")) {
17.      Class
18.       .forName("com.microsoft.jdbc.sqlserver." +
19.              "SQLServerDriver");
20.      this.url = "jdbc:microsoft:sqlserver://"
21.       + host + ":" + port + ";user=" + user
22.       + ";password=" + pass;
23.      this.conn = DriverManager.getConnection(url);
24.     } else if (jenis.equals("Oracle")) {
25.      Class
26.       .forName("oracle.jdbc.driver.OracleDriver");
27.      this.url = "jdbc:oracle:thin:@" + host + ":"
28.       + port + ":" + nama;
29.     }
30.     this.conn = DriverManager.getConnection(url,
31.      user, pass);
32.     FrameUtama.addLog("Connected to :" + url);
33.     this.stmt = conn.createStatement(
34.      ResultSet.TYPE_SCROLL_INSENSITIVE,
35.      ResultSet.CONCUR_UPDATABLE);
36.     this.stmt2 = conn.createStatement();
37.   }
```

Segmen program 4. 2 Method connect()

Operasi pada method di atas, akan menentukan server basis data manakah yang diinginkan oleh pengguna. Setelah pembangunan koneksi diatas berhasil,

maka aplikasi akan menggunakan koneksi ini sebagai koneksi *default* sampai pengguna merubahnya kembali.

4.2. Implementasi Use case Bangkitkan Dataset Transaksi

Proses pembangkitan transaksi dalam aplikasi ini merupakan proses pembuatan dataset transaksi sintetis dengan menggunakan beberapa parameter yang menandakan karakteristik dari dataset transaksi yang akan dibangkitkan nantinya. Implementasi dari *use case* ini menggunakan acuan dari tugas akhir Adita P [ADT-02]. Pengimplementasiannya berupa modifikasi kelas dari tugas akhir tersebut. Parameter yang digunakan dalam pembangkitan transaksi ini adalah adalah :

- Jumlah item (N)
- Rata-rata ukuran Large Itemset (I)
- Ukuran maksimal Large Itemset (MI)
- Jumlah Large Itemset (L)
- Rata-rata ukuran transaksi (T)
- Ukuran maksimal transaksi (MT)
- Jumlah transaksi (D)

Proses dimulai dari masukan pengguna yang harus mengisi parameter-parameter diatas. Untuk nama tabel tempat menyimpan dataset hasil pembangkitan pengguna dapat memberi nama tersendiri atau memilih nama default yang menunjukkan nilai-nilai parameter dari dataset. Proses penentuan parameter ini ditangani oleh sebuah kelas yaitu `PanelGenTransaksi`. Kelas ini merupakan *extends* dari objek `javax.swing.JPanel`. Kelas ini berfungsi sebagai antarmuka dengan pengguna dalam proses pembangkitan data ini. Proses pembangkitan dilakukan oleh kelas `GeneratorTransaksi`. Pemanggilan kelas tersebut dari `PanelGenTransaksi` dilakukan oleh method `btnGenerateClick()` di bawah ini :

```
1.    protected void btnGenerateClick() {
2.     if (cekParam()) {
3.      int N = Integer.parseInt(tfN.getText());
4.      int I = Integer.parseInt(tfI.getText());
5.      int MI = Integer.parseInt(tfMI.getText());
6.      int L = Integer.parseInt(tfL.getText());
7.      int T = Integer.parseInt(tfT.getText());
8.      int MT = Integer.parseInt(tfMT.getText());
9.      int D = Integer.parseInt(tfD.getText());
10.     String tabName = tfNamaTabel.getText();
11.     genTransaksi = new GeneratorTransaksi(N, I,
12.      MI, L, T, MT, D, tabName);
13.     try {
14.      genTransaksi.generate();
15.     } catch (SQLException e) {
16.      JOptionPane.showMessageDialog(null, e
17.       .getMessage());
18.      e.printStackTrace();
19.     }
20.    }
21.   }
```

Segmen program 4. 3 Method btnGenerateClick() pembangkitan transaksi

Pada method `btnGenerateClick()` di atas, langkah pertama adalah pengecekan parameter dari pengguna. Apabila parameter yang dimasukkan pengguna sudah benar, maka parameter tersebut digunakan dalam pemanggilan kelas `GeneratorTransaksi` (baris 11). Untuk memulai proses pembangkitan, ditandai dengan method `generate()` Method ini memanggil dua buah method, method pertama untuk membangkitkan large itemset dan method kedua untuk membangkitkan transaksi. Pemanggilan kedua method ini dapat ditunjukkan pada gambar dibawah ini :

```
1.    public void generate() throws SQLException {
2.     FrameUtama
3.      .addLog("Pembangkitan Transaksi dimulai");
4.     con = FrameUtama.getKoneksi();
5.     FrameUtama.addLog("Pembuatan tabel transaksi");
6.     con.createTableTransaksi(namaTabel);
7.     FrameUtama.addLog("Pembangkitan large itemset");
8.     generateLargeItemset();
9.     FrameUtama.addLog("Pembangkitan Transaksi");
10.    generateTransactions();
```

```
11.     FrameUtama
12.      .addLog("Pembangkitan Transaksi selesai");
13.   }
```

Segmen program 4. 4 Method generate() pada GeneratorTransaksi

4.3. Implementasi Use Case Bangkitkan Dataset Item

Proses pembangkitan produk dalam aplikasi ini merupakan proses pembuatan dataset produk sintetis dengan menggunakan beberapa parameter yang menandakan karakteristik dari dataset transaksi yang akan dibangkitkan nantinya. Parameter tersebut adalah :

- Jumlah item (NP)
- Harga Minimum (MIP)
- Harga Maksimum (MAP)

Proses dimulai dari masukan pengguna yang harus mengisi parameter-parameter diatas. Untuk nama tabel tempat menyimpan dataset hasil pembangkitan pengguna dapat memberi nama tersendiri atau memilih nama default yang menunjukkan nilai-nilai parameter dari dataset. Proses penentuan parameter ini ditangani oleh sebuah kelas yaitu `PanelGenItem`. Kelas ini merupakan *extends* dari objek `javax.swing.JPanel`. Kelas ini berfungsi sebagai antarmuka dengan pengguna dalam proses pembangkitan data. Proses pembangkitan dilakukan oleh kelas `GeneratorItem`. Pemanggilan kelas ini dari `PanelGenItem` ditunjukkan pada segmen program di bawah ini :

```
1.    protected void btnGenerateClick()
2.     throws SQLException {
3.     if (cekParam()) {
4.      int NP = Integer.parseInt(tfNP.getText());
5.      int MIP = Integer.parseInt(tfMIP.getText());
6.      int MAP = Integer.parseInt(tfMAP.getText());
7.      String tabName = tfNamaTabel.getText();
8.      genItem = new GeneratorItem(NP, MIP, MAP,
9.       tabName);
10.     genItem.generate();
11.    }
12.   }
```

Segmen program 4. 5 Method btnGenerateClick()

Operasi di atas akan memanggil method `generate()` pada kelas `GeneratorItem`. Method ini melakukan proses pembangkitan dengan cara memanggil angka acak berdasarkan distribusi uniform.

```
1.    public void generate() {
2.     FrameUtama.addLog("Pembangkitan Item dimulai");
3.     con = FrameUtama.getKoneksi();
4.     FrameUtama.addLog("Pembuatan tabel item baru");
5.     con.createTableItem(namaTabel);
6.     FrameUtama.addLog("Pembangkitan item");
7.     try {
8.      genAndInsert();
9.     } catch (SQLException e) {
10.     FrameUtama.addLog("Pembangkitan item :"
11.      + e.getMessage());
12.     e.printStackTrace();
13.    }
14.   }
15.
16.   private void genAndInsert() throws SQLException {
17.    for (int i = 0; i < NP; i++) {
18.     double k = randG.nextUniform(MIP, MAP);
19.     int Price = (int) k;
20.     con.insertItem(i + 1, "item" + (i + 1), ""
21.      + Price);
22.    }
23.    FrameUtama.addLog("Pembangkitan Item selesai");
24.   }
```

Segmen program 4. 6 Method generate() pada GeneratorItem

4.4. Implementasi Use Case Terapkan Praproses Exante

Metode ExAnte merupakan sebuah metode untuk mengolah dataset transaksi sebelum dataset tersebut diproses dalam pencarian *frequent pattern*. Parameter-parameter yang dibutuhkan dalam proses ini antara lain :

- Nama tabel transaksi
- Nama tabel item
- Nilai minimum support
- Jenis batasan monoton
- Nilai batasan monoton

- Nama tabel hasil

Langkah-langkah yang dilakukan dalam proses ini antara lain :

- **Pembuatan tabel sementara.**

Tahapan ini dilakukan karena metode ExAnte akan menghilangkan item-item nonfrequent serta transaksi-transaksi yang tidak memenuhi batasan monoton. Agar dataset asli tidak berubah isinya dan dapat digunakan dalam uji coba dengan algoritma lain, maka pengolahan dataset dialihkan ke database sementara. Tabel transaksi sementara ini memiliki struktur yang sama dengan struktur dari tabel transaksi asli. Tahapan ini dilakukan dengan memanggil method `createTabelTransaksi` pada class KoneksiDB. Method ini ditunjukkan pada gambar berikut ini :

```
1.    public void createTableTransaksi(String namaTabel) {
2.     try {
3.      execute("drop table " + namaTabel);
4.     } catch (Exception e) {
5.      e.getMessage();
6.     }
7.     try {
8.      execute("create table " + namaTabel
9.       + "(TID varchar (10), PID varchar(10))");
10.     tabelTransaksi = namaTabel;
11.    } catch (Exception e) {
12.     e.printStackTrace();
13.    }
14.   }
```


Segmen program 4. 7 Method createTableTransaksi()

Secara default, tabel sementara yang akan dibuat dengan pemanggilan method di atas, diberi nama dengan tabel "temp".

- **Pengambilan atribut price dari basis data item**

Metode *ExAnte* memerlukan atribut *price* dari item-item yang terlibat dalam transaksi untuk digunakan dalam penyaringan dengan batasan monoton. Pengambilan atribut ini dilakukan oleh method `getDaftarItems()` yang dimiliki oleh class `DBReader`. Kembalian dari method ini berupa `Vector` yang berisi `Item`. Kemudian, item-item tersebut dipindahkan ke dalam bentuk

HashMap dengan ID Item sebagai pengenal untuk mempermudah pencarian item tersebut. Dalam operasi ini juga dilakukan proses penghiungan kemunculan item. Penghitungan ini dilakukan pada tabel asli. Nilai dari penghitungan ini, disimpan pada atribut *array* yang bernama counts. Tujuan dari operasi ini adalah untuk mengurutkan item berdasarkan jumlah kemunculannya. Proses ini dilakukan dalam method updateDaftarHarga() pada segmen program di bawah ini :

```
1.   private void updateDaftarHarga()
2.    throws Exception {
3.    Vector dfItem = dbre.getDaftarItems();
4.    counts = new int[num_cols + 1];
5.    daftarHarga = new HashMap();
6.    jmlItemAwal = dfItem.size();
7.    info += "\n Jumlah Item awal : " + jmlItemAwal;
8.    for (int i = 0; i < jmlItemAwal; i++) {
9.     Item it = (Item) dfItem.get(i);
10.    counts[it.getItem()] = it.getCount();
11.    daftarHarga.put(new Integer(it.getItem()), it);
12.   }
13.   dfItem = null;
14.  }
```

Segmen program 4. 8 Method updateDaftarHarga()

- **Pemindahan data dari database asli ke TreeExa (Pembangunan pohon)**

TreeExa merupakan sebuah struktur data berbentuk pohon yang merepresentasikan isi dataset transaksi. Pembangunan pohon ini didasarkan pada pembuatan FPTree pada algoritma FPGrowth.

Proses pemindahan ini dilakukan dengan cara menelusuri tabel transaksi per satu transaksi. Method getFirstRow() dan getNextRow() mengembalikan itemset-itemset yang mewakili sebuah transaksi. Pembangunan pohon dilakukan dengan tujuan untuk mengurangi akses basis data oleh aplikasi. Usaha reduksi data, cukup dilakukan pada pohon yang dibentuk oleh proses ini. Proses pembangunan pohon ini dilakukan oleh operasi bangunTree() di bawah ini :

```
1.    private TreeExa bangunTree() throws SQLException {
2.     if (counts.length == 0)
3.      return null;
4.     Item[] item_objs = new Item[num_cols];
5.     for (int i = 1, j = 0; i < num_cols + 1; i++) {
6.      item_objs[j++] = new Item(i, counts[i]);
7.     }
8.     Arrays.sort(item_objs);
9.     int[] items = new int[item_objs.length];
10.    for (int i = 0; i < item_objs.length; i++)
11.     items[i] = item_objs[item_objs.length - i - 1]
12.      .getItem();
13.    TreeExa fpt = new TreeExa(konek, items,
14.     jmlTransAwal);
15.    Itemset row = dbre.getFirstRow();
16.    processRow(row, fpt);
17.    while ((row = dbre.getNextRow()) != null) {
18.     processRow(row, fpt);
19.    }
20.    return fpt;
21.   }
```

Segmen program 4. 9 Method bangunTree()

Pada method di atas, tiap itemset transaksi akan dimasukkan ke dalam pohon `TreeExa`. Proses pemindahan itemset ini akan berlangsung terus sampai seluruh Itemset yang ada pada tabel transaksi awal selesai diperiksa. Proses pemindahan dilakukan oleh method `processRow` di bawah ini :

```
1.    private static void processRow(Itemset row,
2.     TreeExa fpt) {
3.     int i, j, item;
4.     int[] items;
5.     Item[] item_objs;
6.     int rowSize = row.size();
7.     if (rowSize > 0) {
8.      item_objs = new Item[rowSize];
9.      for (i = 0, j = 0; i < rowSize; i++) {
10.      item = row.get(i);
11.      item_objs[j++] = new Item(item, counts[item]);
12.     }
13.     Arrays.sort(item_objs);
14.     items = new int[item_objs.length];
15.     for (i = 0; i < item_objs.length; i++) {
16.      items[i] = item_objs[item_objs.length - i - 1]
```

```
17.         .getItem();
18.       }
19.       fpt.insert(items, 1);
20.     }
21.   }
```

Segmen program 4. 10 Method processRow()

- **Pemangkasan pohon**

Setelah pemindahan data ke `TreeExa` selesai dilakukan, langkah selanjutnya adalah melakukan pemangkasan pohon tersebut berdasarkan batasan monoton dan antimonotonnya. Pemangkasan ini dilakukan untuk membuang transaksi-transaksi yang tidak lolos batasan monoton. Tahapan ini dilakukan oleh method `pruningTree` pada class `TreeExa`. Tahapan ini ditunjukkan dengan gambar di bawah ini :

```
1.    boolean pruningTree(HashMap a, int minCount,
2.     int jenisBatas, int nilaiBatas) {
3.     this.min_weight = minCount;
4.     this.daftarHarga = a;
5.     this.nilaiBatas = nilaiBatas;
6.     this.jenisBatas = jenisBatas;
7.     for (int i = header.length - 1; i >= 0; i--) {
8.      if (header[i].count >= min_weight) {
9.       for (TreeExaNode side_walker = header[i].head;
10.      side_walker != null; side_walker = side_walker.next) {
11.       int counts = 0;
12.       int countPucuk = 0;
13.       if (side_walker.child != null) {
14.        for (TreeExaNode anak = side_walker.child;
15.        anak != null; anak = anak.sibling) {
16.         counts = counts + anak.count;
17.        }
18.        if (side_walker.count > counts) {
19.         countPucuk = side_walker.count - counts;
20.        }
21.       }
22.       if (side_walker.child == null) {
23.        countPucuk = side_walker.count - counts;
24.       }
25.       if (countPucuk > 0) {
26.        Itemset is = new Itemset();
27.        for (TreeExaNode up_walker = side_walker;
```

```
28.          up_walker != root; up_walker = up_walker.parent) {
29.           is.add(up_walker.item);
30.          }
31.          if (!is.lolosBatMono(daftarHarga,
32.           jenisBatas, nilaiBatas)) {
33.           for (TreeExaNode up_walker = side_walker;
34.           up_walker != root; up_walker = up_walker.parent) {
35.            up_walker.count = up_walker.count
36.              - countPucuk;
37.           }
38.          }
39.         }
40.        }
41.       }
42.      }
43.      return pruneItem();
44.     }
```

Segmen program 4. 11 Method updateCount()

- **Pruning item**

Setelah semua dahan pohon selesai melalui proses penghitungan kemunculan, langkah berikutnya adalah penghilangan item non *frequent* dari pohon. Dengan adanya penghilangan item ini, tabel sementara hanya akan berisi item-item yang termasuk *frequent* saja. Proses ini dilakukan pada operasi di bawah ini :

```
1.     boolean pruneItem() {
2.      for (int i = header.length - 1; i >= 0; i--) {
3.       int countitem = 0;
4.       for (TreeExaNode side_walker = header[i].head;
5.       side_walker != null; side_walker = side_walker.next) {
6.        countitem += side_walker.count;
7.       }
8.       header[i].count = countitem;
9.      }
10.     int jml0 = 0;
11.     boolean berkurang = false;
12.     for (int i = header.length - 1; i >= 0; i--) {
13.      if (header[i].count == 0) {
14.       jml0++;
15.      }
16.      if (header[i].count < min_weight
17.       && header[i].count > 0) {
```

```
18.         jml0++;
19.         header[i].count = 0;
20.         for (TreeExaNode side_walker = header[i].head;
21.         side_walker != null; side_walker = side_walker.next) {
22.          side_walker.count = 0;
23.         }
24.        }
25.        if (jml0 > jmlNonFreq) {
26.         berkurang = true;
27.         jmlNonFreq = jml0;
28.        }
29.       }
30.       return berkurang;
31.     }
```

Segmen program 4. 12 Operasi prunItem()

- **Cek loop proses**

Tahapan terakhir dari proses *ExAnte* ini adalah pengecekan apakah masih dimungkinkan untuk melakukan pruning transaksi kembali atau tidak setelah melalui proses pruning item. Hal tersebut dapat dilihat dari apakah jumlah *frequent* item mengalami perubahan atau tidak. Jika jumlah *frequent* item berkuran, berarti masih terdapat kemungkinan untuk melakukan pruning transaksi item dan transaksi kembali.

- **Pemindahan data ke tabel tujuan**

Apabila pada pengecekan loop proses diatas tidak terjadi pengurangan item lagi, maka langkah selanjutnya adalah memindahkan itemset dari pohon ke datalam tabel tujuan. Proses ini di lakukan oleh method `listBranch()` di bawah ini :

```
1.      void listBranch() throws SQLException {
2.       HitungKemunculan();
3.       int freq = 0;
4.       for (int j = 0; j < kemunculanAkhir.length; j++) {
5.        if (kemunculanAkhir[j] >= min_weight) {
6.         freq++;
7.        }
8.       }
9.       ProsesExante.setJmlItemAkhir(freq);
10.      for (int i = header.length - 1; i >= 0; i--) {
11.       if (header[i].count != 0) {
12.        for (TreeExaNode side_walker = header[i].head;
```

```
13.       side_walker != null; side_walker = side_walker.next) {
14.        int counts = 0;
15.        if (side_walker.child != null) {
16.         for (TreeExaNode anak = side_walker.child;
17.         anak != null; anak = anak.sibling) {
18.          counts = counts + anak.count;
19.         }
20.        }
21.        if (side_walker.count > counts) {
22.         String is = "";
23.         Itemset its = new Itemset();
24.         for (TreeExaNode up_walker = side_walker;
25.         up_walker != root; up_walker = up_walker.parent) {
26.          // if (up_walker.count > 0)
27.          is += up_walker.item + " ";
28.          its.add(up_walker.item);
29.         }
30.         int countPucuk = side_walker.count - counts;
31.         if (side_walker.child == null) {
32.          countPucuk = side_walker.count;
33.         }
34.         for (int j = 0; j < its.size(); j++) {
35.          if (kemunculanAkhir[its.get(j)] < min_weight) {
36.           its.remove(its.get(j));
37.          }
38.         }
39.         if (its.lolosBatMono(daftarHarga,
40.          jenisBatas, nilaiBatas)) {
41.          for (int j = 0; j < countPucuk; j++) {
42.           trans++;
43.           konek.insertOneTransaksi("" + trans, its);
44.          }
45.         }
46.         ProsesExante.setJmlTransaksiAkhir(trans);
47.        }
48.       }
49.      }
50.     }
51.    }
```

Segmen program 4. 13 method listBranch()

Keseluruhan tahapan diatas disatukan menjadi sebuah proses utuh yang dipanggil ketika penerapan metode *ExAnte* dilakukan. Pemanggilan ini dilakukan dengan cara memanggil method proses pada gambar di bawah ini. Method proses

ini, merangkai langkah-langkah diatas menjadi sebuah rangkaian proses sekuensial.

```
1.    public void Proses() throws Exception {
2.     jmlFreqItem = 0;
3.     jmlTransaksiAkhir = 0;
4.     jmlTransAwal = dbre.getJmlTransaksi();
5.     info += "\n Jumlah Transaksi Awal : "
6.      + jmlTransAwal;
7.     num_cols = (int) dbre.getJmlItem();
8.     minCount = (int) (jmlTransAwal * support);
9.     if (minCount == 0) {
10.     minCount = 1;
11.    } else {
12.     minCount = minCount + 1;
13.    }
14.    FrameUtama.addLog("ExAnte : "
15.     + "Updating kemunculan & price item");
16.    updateDaftarHarga();
17.    FrameUtama.addLog("ExAnte : "
18.     + "Pembangunan tree");
19.    TreeExa fpt = bangunTree();
20.    boolean berkurang;
21.    FrameUtama.addLog("ExAnte : " + "Pruning tree");
22.    berkurang = fpt.pruningTree(daftarHarga,
23.     minCount, jenisBatas, nilaiBatas);
24.    int iterasi = 0;
25.    while (berkurang) {
26.     iterasi++;
27.     FrameUtama.addLog("ExAnte : "
28.      + "Iterasi ke : " + iterasi);
29.     berkurang = fpt.pruningTree(daftarHarga,
30.      minCount, jenisBatas, nilaiBatas);
31.    }
32.    FrameUtama.addLog("ExAnte : "
33.     + "Pembuatan tabel " + namaTabTujuan);
34.    konek.createTableTransaksi(namaTabTujuan);
35.    FrameUtama
36.     .addLog("ExAnte : "
37.      + "Pemindahan hasil ke tabel "
38.      + namaTabTujuan);
39.    fpt.listBranch();
40.    info += "\n Jumlah Iterasi proses Exante : "
41.     + iterasi;
42.    info += "\n Jumlah Item Akhir : " + jmlFreqItem;
```

```
43.     info += "\n Jumlah Transaksi Akhir : "
44.      + jmlTransaksiAkhir;
45.     info += "\n Persentase Reduksi Transaksi : "
46.      + (double) (jmlTransAwal - jmlTransaksiAkhir)
47.      / (double) jmlTransAwal * 100;
48.   }
```

Segmen program 4. 14 Method proses() pada class Exante

4.5. Implementasi Use Case Cari Pola dengan Algoritma Apriori

Proses penambangan pola yang digunakan dalam aplikasi ini, mengimplementasikan class `Apriori` dan `HashTree` dari package `laur.dm.ar`. Untuk memulai proses penambangan, pengguna memerlukan untuk memasukkan parameter-parameter penambangan. Parameter-parameter tersebut antara lain :

- Nama table transaksi
- Nama table item
- Pilihan penerapan *ExAnte*
- Minimum support
- Jenis batasan monoton
- Nilai batasan monoton

Dalam implementasi algoritma Apriori ini, proses penambangan dibagi menjadi tiga tahapan utama yaitu:

- **Penghitungan kemunculan dari kandidat**

Proses penambangan dengan algoritma apriori dimulai dengan himpunan 1-itemset yang ada pada tabel transaksi. Himpunan item awal inilah yang disebut dengan kandidat. Tahapan penghitungan kemunculan mencari jumlah kemunculan masing-masing item pada tabel transaksi. Tahapan ini dilakukan oleh method `weightCandidate` dengan cara membangun `HashTree` untuk mempermudah penghitungan dari item-item kandidat tersebut. Pembangunan ini dilakukan

dengan melakukan update HashTree. Proses ini ditunjukkan pada gambar di bawah ini :

```
1.    private void weighCandidates()
2.     throws SQLException {
3.     ht_candidates.prepareForDescent();
4.     Itemset row = db_reader.getFirstRow();
5.     ht_candidates.update(row);
6.     while ((row = db_reader.getNextRow()) != null) {
7.      ht_candidates.update(row);
8.     }
9.    }
```

Segmen program 4. 15 Method weightCandidates()

- **Evaluasi kandidat**

Setelah masing-masing kandidat telah diketahui jumlah kemunculannya, tahapan berikutnya adalah mengevaluasi kandidat tersebut. Tahapan evaluasi ini berupa pengujian kandidat-kandidat terhadap batasan frekuensi dan batasan monoton. Apabila kandidat tersebut termasuk dalam kandidat yang *non frequent* maka kandidat tersebut tidak diikutkan dalam proses pembangkitan kandidat berikutnya. Sedangkan apabila kandidat tersebut lolos pengujian, maka kandidat tersebut merupakan pola yang dicari dan masuk ke himpunan hasil. Proses pada tahap ini dilakukan oleh method evaluateCandidates seperti pada gambar di bawah ini :

```
1.    private void evaluateCandidates() {
2.     Itemset is;
3.     for (int i = 0; i < candidates.size(); i++)
4.      // if this is a frequent itemset
5.      if ((is = (Itemset) candidates.get(i))
6.       .getWeight() >= min_weight) {
7.       // compute support of itemset
8.       is.setSupport((double) is.getWeight()
9.        / (double) num_rows);
10.      // write itemset to the cache
11.      if (is.lolosBatMono(mapProduk,
12.       batasanMonoton, nilaiMonoton)) {
13.       // MainFrame.has.addHasil(is);
14.       write(is);
15.      }
16.      // then add it to the frequent and k_frequent collections
17.      // / frequent.add(is);
18.      k_frequent.add(is);
```

```
19.      ht_k_frequent.add(k_frequent.size() - 1);
20.     }
21.    // reinitialize candidates for next step
22.    candidates.clear();
23.    ht_candidates = new HashTree(candidates);
24.   }
```

Segmen program 4. 16 Method evaluateCandidates

- **Pembangkitan kandidat**

Tahapan pembangkitan kandidat ini menghasilkan kandidat-kandidat baru dari himpunan k-1 itemset yang telah lolos tahap evaluasi kandidat sebelumnya. Tahapan ini melakukan pengkombinasian dua buah itemset yang bisa dikombinasikan. Itemset yang dimaksud merupakan anggota dari himpunan kandidat *frequent* sebelumnya. Tahapan ini diimplementasikan seperti pada gambar di bawah ini :

```
1.    private void generateCandidates() {
2.     ht_k_frequent.prepareForDescent();
3.     if (k_frequent.size() == 0)
4.      return;
5.     for (int i = 0; i < k_frequent.size() - 1; i++)
6.      for (int j = i + 1; j < k_frequent.size(); j++)
7.       if (!getCandidate(i, j))
8.        ;
9.     // reinitialize k_frequent for next step
10.    k_frequent.clear();
11.    ht_k_frequent = new HashTree(k_frequent);
12.   }
```

Segmen program 4. 17 Method generateCandidates

Keseluruhan tahapan diatas disatukan menjadi sebuah proses utuh yang dipanggil ketika penambangan pola dengan menggunakan algoritma Apriori dilakukan. Pemanggilan ini dilakukan dengan cara memanggil method `cariPola` pada gambar di bawah ini. Method proses ini, merangkai langkah-langkah diatas menjadi sebuah rangkaian proses sekuensial.

```
1.    public void cariPola(DBReader dbReader,
2.     double minSupport, int batasanMonoton,
3.     int nilaiMonoton) throws Exception {
4.     db_reader = dbReader;
5.     num_rows = dbReader.getJmlTransaksi();
6.     min_weight = (num_rows * minSupport);
7.     if (min_weight == 0) {
```

```
8.       min_weight = 1;
9.      }
10.     this.batasanMonoton = batasanMonoton;
11.     this.nilaiMonoton = nilaiMonoton;
12.     candidates = new ArrayList(INITIAL_CAPACITY);
13.     k_frequent = new ArrayList(INITIAL_CAPACITY);
14.     ht_k_frequent = new HashTree(k_frequent);
15.     ht_candidates = new HashTree(candidates);
16.     Itemset is;
17.     Vector tem = db_reader.getDaftarItems();
18.     mapProduk = new HashMap();
19.     for (int i = 0; i < tem.size(); i++) {
20.      Item it = (Item) tem.get(i);
21.      mapProduk.put(new Integer(it.getItem()), it);
22.      is = new Itemset(1);
23.      is.add(it.getItem());
24.      candidates.add(is);
25.      ht_candidates.add(candidates.size() - 1);
26.     }
27.     for (pass_num = 1;; pass_num++) {
28.      weighCandidates();
29.      evaluateCandidates();
30.      if (k_frequent.size() == 0)
31.       break;
32.      if (pass_num >= tem.size())
33.       break;
34.      generateCandidates();
35.      if (candidates.size() == 0)
36.       break;
37.     }
38.    }
```

Segmen program 4. 18 Method cariPola Algoritma Apriori

4.6. Implementasi Use Case Cari Pola dengan Algoritma FPGrowth

Proses Algoritma FPGrowth yang digunakan dalam aplikasi ini, juga mengimplementasikan class FPGrowth pada package `laur.dm.ar`. Dalam implementasi algoritma FPGrowth ini, proses penambangan dibagi menjadi tiga tahapan utama yaitu:

- **Penghitungan kemunculan item**

Tahapan penghitungan kemunculan mencari jumlah kemunculan masing-masing item pada tabel transaksi. Penghitungan ini dilakukan oleh method `countItemOccurance`. Dalam method ini, dilakukan penelusuran tabel transaksi sembari melakukan update jumlah kemunculan untuk tiap item yang ditemui dalam transaksi. Method ini ditunjukkan pada segmen program di bawah ini :

```
1.    private void countItemOccurrences()
2.     throws SQLException {
3.     counts = new int[num_cols + 1];
4.     Itemset row = db_reader.getFirstRow();
5.     for (int i = 0; i < row.size(); i++)
6.      counts[row.get(i)]++;
7.     while ((row = db_reader.getNextRow()) != null) {
8.      for (int i = 0; i < row.size(); i++) {
9.       counts[row.get(i)]++;
10.     }
11.    }
12.    pass_num++;
13.   }
```

Segmen program 4. 19 Method countItemOccurance

- **Pembangunan FPTree**

Yang dilakukan pada tahapan ini adalah mengurutkan item-item terlebih dahulu berdasarkan jumlah kemunculannya. Urutannya dari item yang memiliki kemunculan paling banyak sampai ke yang paling sedikit. Dengan batasan, item-item tersebut termasuk item yang *frequent*. Sedangkan item *non frequent* tidak diperhitungkan. Setelah itu, dilakukan penelusuran tabel transaksi kembali untuk memasukkan tiap-tiap transaksi kedalam `FPTree` dengan menggunakan method `processRow` yang dimiliki oleh objek `FPTree`. Tahapan ini dilakukan oleh method `constructFPTree` seperti yang ditunjukkan pada segmen program di bawah ini :

```
1.    private FPTree constructFPTree()
2.     throws SQLException {
3.     // see how many frequent items there are in the database
4.     int num_frequent = 0;
5.     for (int i = 1; i < counts.length; i++) {
```

```
6.       if (counts[i] >= min_weight)
7.        num_frequent++;
8.      }
9.      if (num_frequent == 0)
10.      return null;
11.     // put all frequent items in an array of Items
12.     Item[] item_objs = new Item[num_frequent];
13.     for (int i = 1, j = 0; i < counts.length; i++)
14.      if (counts[i] >= min_weight)
15.       item_objs[j++] = new Item(i, counts[i]);
16.     // and sort them ascendingly according to weight
17.     Arrays.sort(item_objs);
        // then place the items in an array of ints in descending
18.  order
19.     int[] items = new int[num_frequent];
20.     for (int i = 0; i < num_frequent; i++)
21.      items[i] = item_objs[num_frequent - i - 1]
22.       .getItem();
23.     // initialize FPTree
24.     FPTree fpt = new FPTree(items, num_rows,
25.      min_weight);
26.     Itemset row = db_reader.getFirstRow();
27.     processRow(row, fpt);
28.     while ((row = db_reader.getNextRow()) != null) {
29.      processRow(row, fpt);
30.     }
31.     pass_num++;
32.     return fpt;
33.   }
```

Segmen program 4. 20 Method constructFPTree

- **Penelusuran FPTree untuk mencari frequent pattern**

Setelah `FPTree` yang merepresentasikan tabel transaksi selesai dibangun, tahapan berikutnya adalah pencarian *frequent pattern* dengan cara menelusuri dahan-dahan dari `FPTree` tersebut. Tahapan ini dilakukan oleh method `fp_growth` ditunjukkan pada gambar di bawah ini. Pada method ini, dilakukan penelusuran sesuai dengan algoritma FPGrowth, dimana dicek apakah *tree* memiliki percabangan atau tidak, jika memiliki percabangan maka akan dibangun conditional FPTree yang mana pada *tree* ini diterapkan proses penelusuran secara rekursif. Untuk dahan tanpa percabangan, dilakukan pengkombinasian item-item yang ada pada dahan tersebut untuk menghasilkan pola yang memenuhi batasan yang diberikan.

```
1.     void fp_growth(Itemset is_suffix) {
2.      // check for user-requested abort
3.      if (!hasMultiplePaths) {
        // collect items from the tree, least frequent item
4.   first!!!
5.      int[] items = new int[header.length];
6.      for (int i = 0; i < header.length; i++)
7.       items[header.length - i - 1] = header[i].item;
        // generate all item combinations of the tree's single
8.   branch
9.      combine(items, is_suffix);
10.     } else
11.     for (int i = header.length - 1; i >= 0; i--) {
12.      Itemset is_new = new Itemset(is_suffix);
13.      is_new.add(header[i].item);
14.      is_new.setWeight(header[i].count);
15.      is_new.setSupport((double) header[i].count
16.       / (double) num_rows);
17.      // write itemset to the cache
18.      if (is_new.lolosBatMono(mapProduk,
19.       batasanMonoton, nilaiMonoton)) {
20.       write(is_new);
21.      }
22.      FPTree fpt = buildConditionalFPTree(header[i].item);
23.      if (fpt != null)
24.       fpt.fp_growth(is_new);
25.     }
26.    }
```

Segmen program 4. 21 Method fp_growth()

Sama seperti algoritma Apriori, ketiga tahapan diatas diatur dan dijalankan secara berurutan melalui method `cariPola` yang dimiliki oleh class `FPGrowth`. Method ini dapat digambarkan pada gambar berikut ini :

```
1.    public void cariPola(DBReader dbReader,
2.     double minSupport, int batasanMonoton,
3.     int nilaiMonoton) throws Exception {
4.     this.db_reader = dbReader;
5.     this.batasanMonoton = batasanMonoton;
6.     this.nilaiMonoton = nilaiMonoton;
7.     num_rows = dbReader.getJmlTransaksi();
8.     num_cols = (int) db_reader.getJmlItem();
9.     min_weight = (long) (num_rows * minSupport);
10.    if (min_weight == 0) {
11.     min_weight = 1;
12.    }
13.    // tahap 1
```

```
14.    FrameUtama.setProgress(
15.     "count Item Occurrences", 10);
16.    countItemOccurrences();
17.    Vector produk = dbReader.getDaftarItems();
18.    mapProduk = new HashMap();
19.    for (int i = 0; i < produk.size(); i++) {
20.     Item it = (Item) produk.get(i);
21.     mapProduk.put(new Integer(it.getItem()), it);
22.    }
23.    // tahap 2
24.    FrameUtama
25.     .setProgress("Pembangunan FPTree", 40);
26.    FPTree fpt = constructFPTree();
27.    if (fpt != null) {
28.     // tahap 3
29.     FrameUtama.setProgress("FP Growth", 70);
30.     fpt.fp_growth(new Itemset());
31.     FrameUtama.setProgress("Finish", 100);
32.    } else
33.     FrameUtama.addLog("FPTree kosong");
34.   }
```

Segmen program 4. 22 Method cariPola Algoritma FPGrowth

4.7. Implementasi Use case Cari Pola dengan Algoritma Eclat

Proses Algoritma Eclat yang digunakan dalam aplikasi ini, mengimplementasikan algoitma Eclat. Proses penambangan dibagi menjadi tiga tahapan utama yaitu:

- **Pembangunan Incidence Matrix**

Algoritma eclat dimulai dari pembangunan *incidence matrix* yang merepresentasikan seluruh isi tabel transaksi ke dalam sebuah bentuk matrik. Pembangunan matrik ini dilakukan dengan membuat terlebih dahulu sebuah object `HashMatrix`. Serelah itu dilakukan penelusuran tabel transaksi untuk memindahkan isinya ke dalam matrik. Tahapan ini dilakukan oleh method `createIncidenceMatrix` yang dapat dilihat padda segmen program di bawah ini :

```
1.     private HashMatrix createIncidenceMatrix(
2.      DBReader trans, int numFreq)
3.      throws SQLException {
4.      int numItems = trans.getJmlItem();
```

```
5.      HashMatrix vecs = new HashMatrix(numItems);
6.      int transNum = 1;
7.      Itemset is = trans.getFirstRow();
8.      for (int i = 0; i < is.size(); i++) {
9.       Itemset item = new Itemset();
10.      item.add(is.get(i));
11.      if (vecs.getRows(item) == null) {
12.       vecs.addRow(item, new Vector());
13.      }
14.      Vector temp = vecs.getRows(item);
15.      temp.add(new Integer(transNum));
16.      vecs.updateRow(item, temp);
17.     }
18.     while ((is = trans.getNextRow()) != null) {
19.      transNum++;
20.      for (int i = 0; i < is.size(); i++) {
21.       Itemset item = new Itemset();
22.       item.add(is.get(i));
23.       if (vecs.getRows(item) == null) {
24.        vecs.addRow(item, new Vector());
25.       }
26.       Vector temp = (Vector) vecs.getRows(item);
27.       temp.add(new Integer(transNum));
28.       vecs.updateRow(item, temp);
29.      }
30.     }
31.     return vecs;
32.   }
```

Segmen program 4. 23 Method createIncidenceMatrix

- **Filtering Matrix**

Setelah *incidence matrix* selesai dibuat, dilakukan penyaringan matrik tersebut untuk menghilangkan baris-baris yang memiliki jumlah anggota kurang dari *minimum support*. Tahapan ini dilakukan oleh method `filterIncidenceMatrix` sehingga menghasilakan *incidence matrix* berisikan itemset yang memenuhi minimum support saja. Sehingga baris-baris yang lolos proses penyaringan ini menjadi anggota dari himpunan *frequent pattern*. Method ini ditunjukkan pada segmen program di bawah ini :

```
1.      public int filterIncidenceMatrix(int minSup) {
2.       for (int i = 0; i < size(); i++) {
3.        Itemset label = (Itemset) rowLabels.get(i);
4.        Vector curRow = getRows(label);
5.        if (curRow.size() < minSup) {
```

```
6.          removeRow(label);
7.          i--;
8.         }
9.        }
10.       return size();
11.     }
```

Segmen program 4. 24 Method filterIncidenceMatrix

- **Penyilangan antar baris matrik**

Untuk mencari itemset-itemset berikutnya, dilakukan penyilangan antar baris yang bisa disilangkan. Dengan begitu akan menghasilkan matrik baru yang beranggotakan itemset-itemset hasil kombinasi *matrix incidence* sebelumnya. Tahapan ini dilakukan oleh method `intersect` seperti pada gambar di bawah ini

```
1.     public static Vector intersect(Vector a, Vector b) {
2.      int item1 = 0;
3.      int item2 = 0;
4.      if (a.size() != 0 && b.size() != 0) {
5.       item1 = ((Integer) a.get(0)).intValue();
6.       item2 = ((Integer) b.get(0)).intValue();
7.      }
8.      Vector rowout = new Vector();
9.      int len1 = a.size();
10.     int len2 = b.size();
11.     int pos1 = 0, pos2 = 0, posOut = 0;
12.     while (true) {
13.      if (item1 < item2) {
14.       pos1++;
15.       if (pos1 >= len1)
16.        break;
17.       item1 = ((Integer) a.get(pos1)).intValue();
18.      } else if (item1 > item2) {
19.       pos2++;
20.       if (pos2 >= len2)
21.        break;
22.       item2 = ((Integer) b.get(pos2)).intValue();
23.      } else {
24.       rowout.add(new Integer(item1));
25.       posOut++;
26.       pos1++;
27.       if (pos1 >= len1)
28.        break;
29.       pos2++;
30.       if (pos2 >= len2)
```

```
31.          break;
32.         item1 = ((Integer) a.get(pos1)).intValue();
33.         item2 = ((Integer) b.get(pos2)).intValue();
34.       }
35.     }
36.     return rowout;
37.   }
```

Segmen program 4. 25 Method intersect()

Sama seperti algoritma apriori dan FPgrowth, ketiga tahapan diatas diatur dan dijalankan secara berurutan melalui method cariPola yang dimiliki oleh class `Eclat`. Method ini dapat digambarkan pada segmen program berikut ini :

```
1.    public void cariPola(DBReader trans,
2.     double minSUp, int batMono, int nilMono)
3.     throws Exception {
4.     this.totalSize = trans.getJmlTransaksi();
5.     this.min_weight = (int) (minSUp * totalSize);
6.     if (min_weight == 0) {
7.      min_weight = 1;
8.     }
9.     this.batMono = batMono;
10.    this.nilMono = nilMono;
11.    mapProduk = new HashMap();
12.    Vector arli = trans.getDaftarItems();
13.    int n = arli.size();
14.    it = new Item[n];
15.    for (int i = 0; i < n; i++) {
16.     it[i] = (Item) arli.get(i);
17.     mapProduk.put(new Integer(it[i].getItem()),
18.      it[i]);
19.    }
20.    Arrays.sort(it);
21.    Vector itemSortedFix = new Vector();
22.    for (int i = 0; i < it.length; i++) {
23.     Item ir = (Item) it[i];
24.     Itemset isAwal = new Itemset();
25.     isAwal.add(ir.getItem());
26.     if (ir.getCount() >= min_weight) {
27.      if (isAwal.lolosBatMono(mapProduk, batMono,
28.       nilMono)) {
29.       isAwal.setWeight(ir.getCount());
30.       isAwal.setSupport((double) ir.getCount()
31.        / (double) totalSize);
32.       write(isAwal);
33.      }
34.      itemSortedFix.add(ir);
```

```
35.       }
36.      }
37.      int numFreq = itemSortedFix.size();
38.      HashMatrix iti = createIncidenceMatrix(trans,
39.       numFreq);
40.      iti.filterIncidenceMatrix(min_weight);
41.      mineFreq(iti, 0, min_weight);
42.    }
```

Segmen program 4. 26 Method cariPola Algoritma Eclat

4.8. Implementasi Use Case Evaluasi kinerja algoritma

Proses evaluasi kinerja merupakan sebuah rangkaian dari seluruh kemungkinan penambangan yang ada pada aplikasi ini. Diantaranya penerapan metode *ExAnte*, penambangan dengan algortima Apriori, FPGrowth dan Eclat. Proses perbandingan ini diatur dan dilakukan oleh class `mineControl`. Pada class ini, untuk tiap awal penambangan dilakukan reset lingkungan kerja aplikasi. Hal tersebut dilakukan agar proses yang baru tidak terpengaruh dengan proses sebelumnya. Proses ini dapat dilihat pada bagian method pada gambar di bawah ini :

```
1.    dbread = new DBReader(kon, namTabTrans, namTabProd);
2.    JmlTransAwal = dbread.getJmlTransaksi();
3.    doApriori();
4.    doFPGrowth();
5.    doEclat();
6.    doExante();
7.    namTabTrans = "temp";
8.    dbread = new DBReader(kon, namTabTrans, namTabProd);
9.    dbread.setJmlTransaksi(JmlTransAwal);
10.   doApriori();
11.   doFPGrowth();
12.   doEclat();
13.   PanelBandingAlg.btnBandingAlg.setEnabled(true);
```

Segmen program 4. 27 Method run class mineControl bagian evaluasi

4.9. Implementasi Use Case Simpan hasil

Proses simpan hasil ini menggunakan objek `FileWriter` dari pckage `java.io`. Yang ditulis pada file hasil ini, merupakan hasil penambangan yang telah dilakukan beserta informasi-informasi yang dihasilkan. Berikut ini merupakan kode untuk penyimpanan hasil :

```
1.     protected void btnSaveClick() {
2.      boolean oke = true;
3.      String toFile = "";
4.      if (aktifPanel == 3) {
5.       toFile = panelExa.getPrintToFile();
6.      } else if (aktifPanel == 4) {
7.       toFile = panelCariPola
8.        .getPrintToFile();
9.      } else if (aktifPanel == 5) {
10.      toFile = panelBandingAlg
11.       .getPrintToFile();
12.     } else {
13.      oke = false;
14.      addLog("Panel yang aktif tidak dapat disimpan");
15.     }
16.     if (oke) {
17.      File cur = new File("./");
18.      JFileChooser jFileChooser1 = new JFileChooser();
19.      jFileChooser1
20.       .setCurrentDirectory(cur);
21.      jFileChooser1
22.       .showSaveDialog(jContentPane);
23.      File file = jFileChooser1
24.       .getSelectedFile();
25.      try {
26.       if (file != null) {
27.        FileWriter ff = new FileWriter(
28.         file);
29.        ff.write(toFile);
30.        ff.flush();
31.        addLog("File Saved to : "
32.         + file.getAbsolutePath());
33.       }
34.      } catch (FileNotFoundException e1) {
35.       e1.printStackTrace();
36.      } catch (IOException e2) {
37.       e2.printStackTrace();
38.      }
39.     }
40.    }
```

Segmen program 4. 28 Penyimpanan ke File Hasil

4.10. Implementasi antarmuka

Aplikasi memiliki antarmuka yang ditujukan sebagai penghubung antara proses yang terjadi dalam aplikasi dengan pengguna. Ketika aplikasi dijalankan,

pengguna dihadapkan pada sebuah frame yang berfungsi sebagai tampilan utama aplikasi. Frame ini berisi menu-menu yang terhubung dengan fitur-fitur lain yang dimiliki oleh aplikasi. Tampilan frame utama dapat dilihat pada gambar berikut:

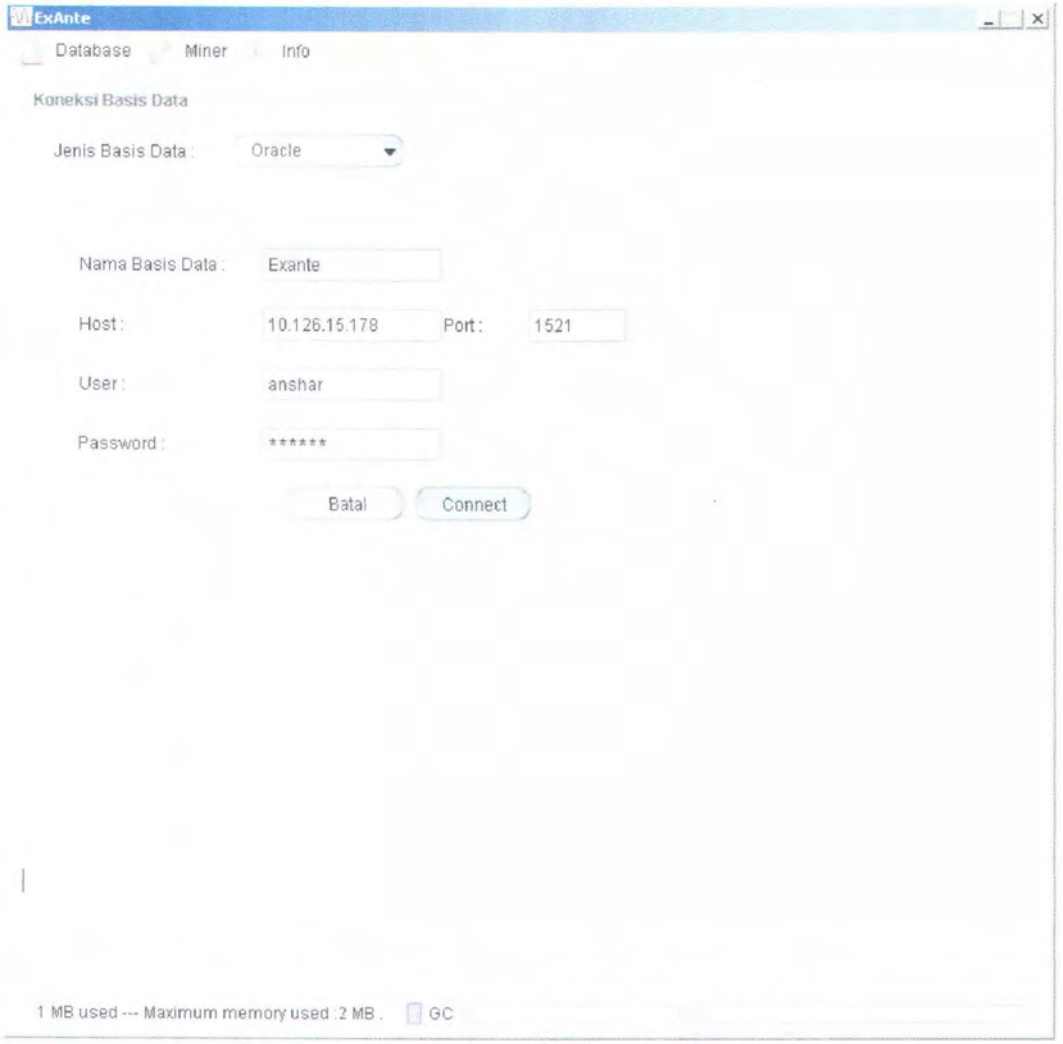
Gambar 4. 1 Antarmuka Frame Utama

Berikut ini dijelaskan antarmuka dari fitur-fitur yang dimiliki oleh aplikasi.

- **Bangun Koneksi Basis Data**

  Antarmuka untuk fitur ini adalah berupa sebuah panel yang berisi field-field yang haru diisi oleh pengguna untuk membangun

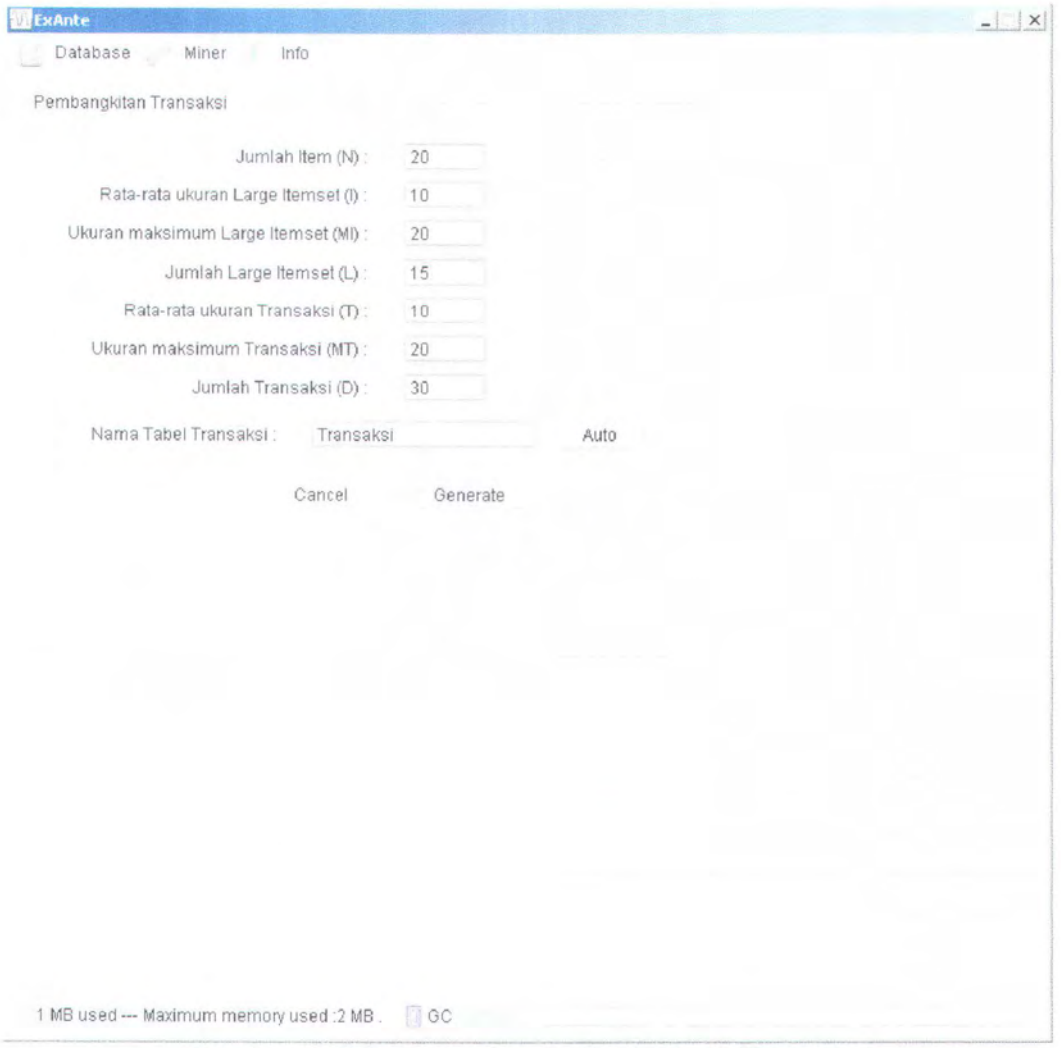
sebuah koneksi dengan server basis data. Tampilan untu fitur ini dapat ditunjukkan pada gambar 4.2



Gambar 4. 2 Antarmuka koneksi basis data

- **Bangkitkan Dataset Transaksi**

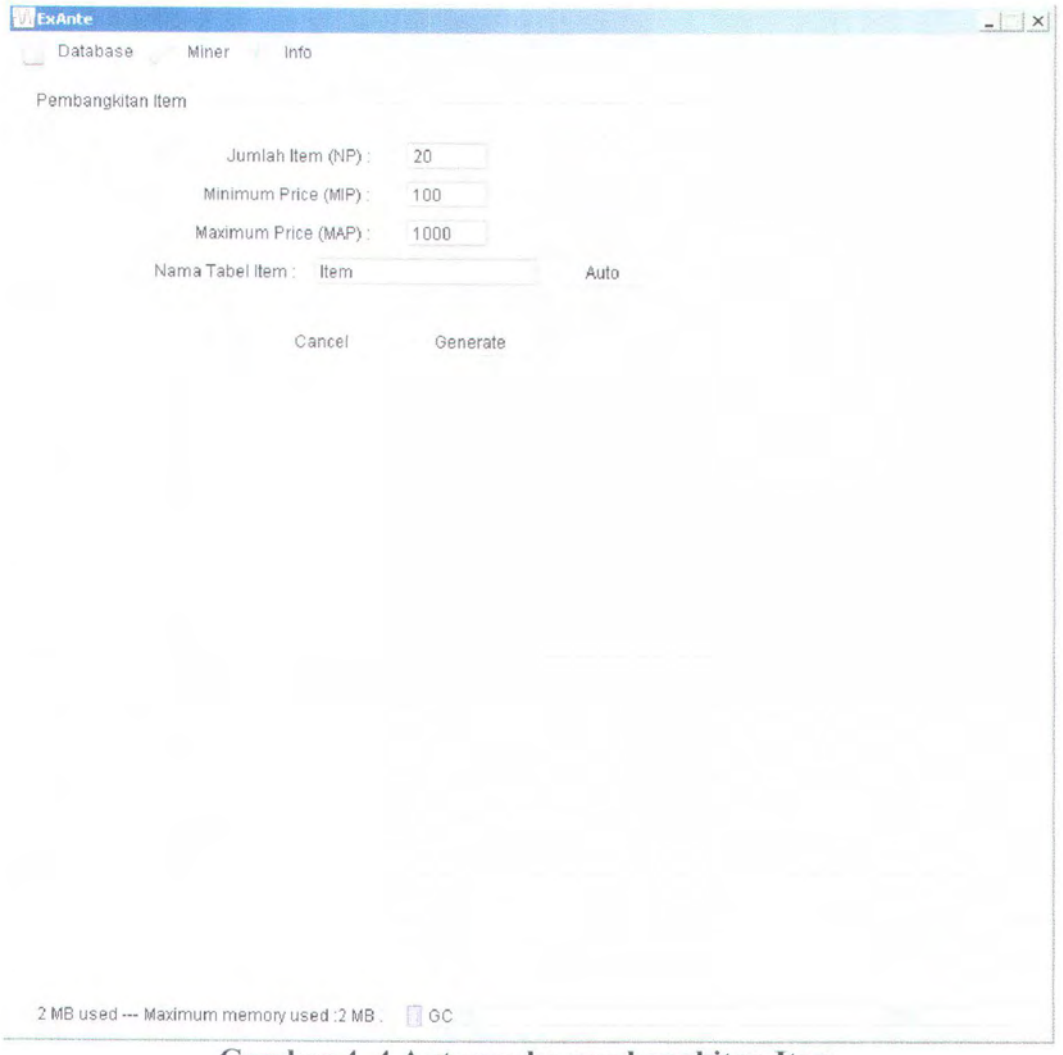
Fitur ini merupakan fitur untuk membangkitkan dataset sintetis. Antarmuka untuk fitur ini adalah berupa sebuah panel yang berisi field masukan parameter pembangkitan dataset transaksi. Tampilan untuk antarmuka fitur ini ditunjukkan pada gambar 4.3.



Gambar 4. 3 Antarmuka pembangkitan Transaksi

- **Bangkitkan Dataset Item**

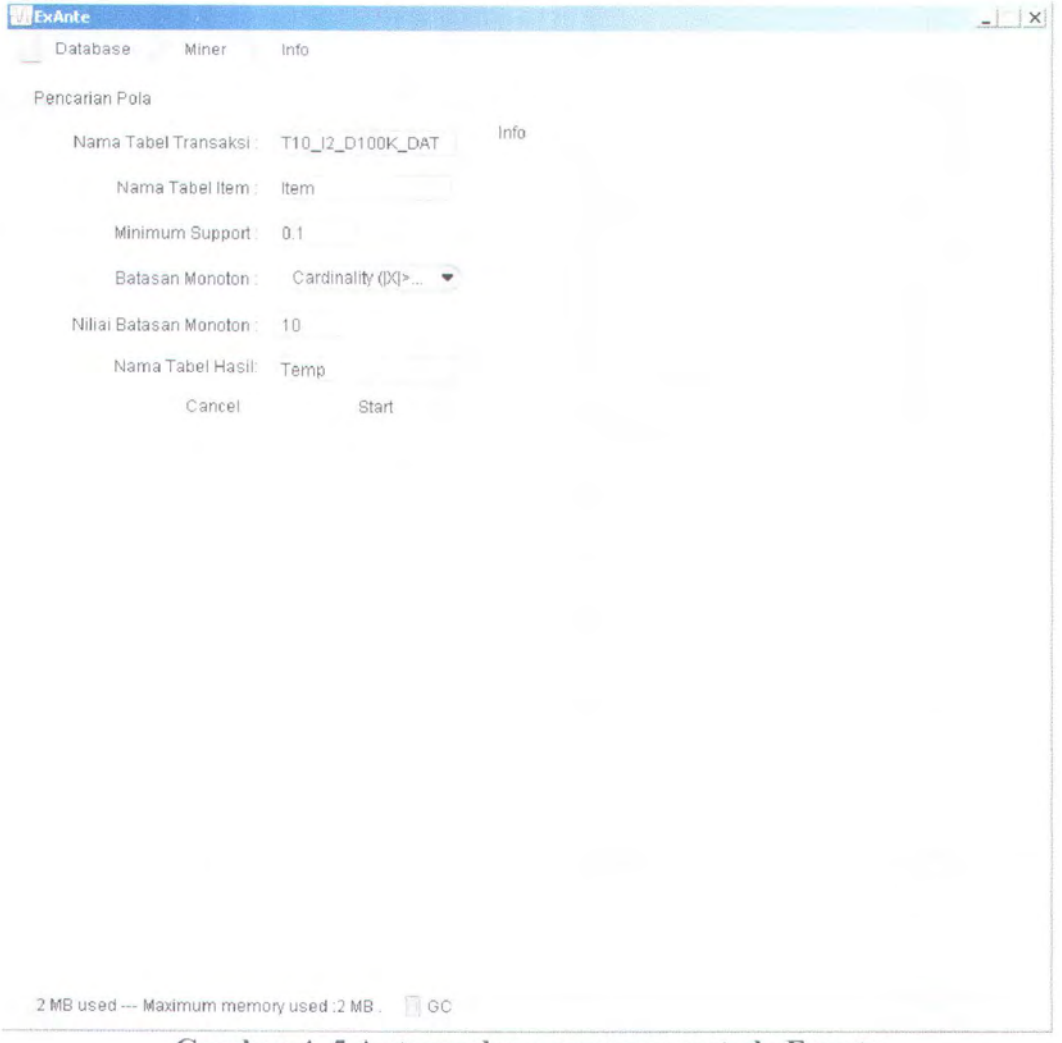
Fitur ini merupakan fitur untuk membangkitkan dataset sintetis. Antarmuka untuk fitur ini adalah berupa sebuah panel yang berisi field masukan parameter pembangkitan dataset item. Tampilan untuk antarmuka fitur ini ditunjukkan pada gambar 4.4.



Gambar 4. 4 Antarmuka pembangkitan Item

- **Terapkan Praproses ExAnte**

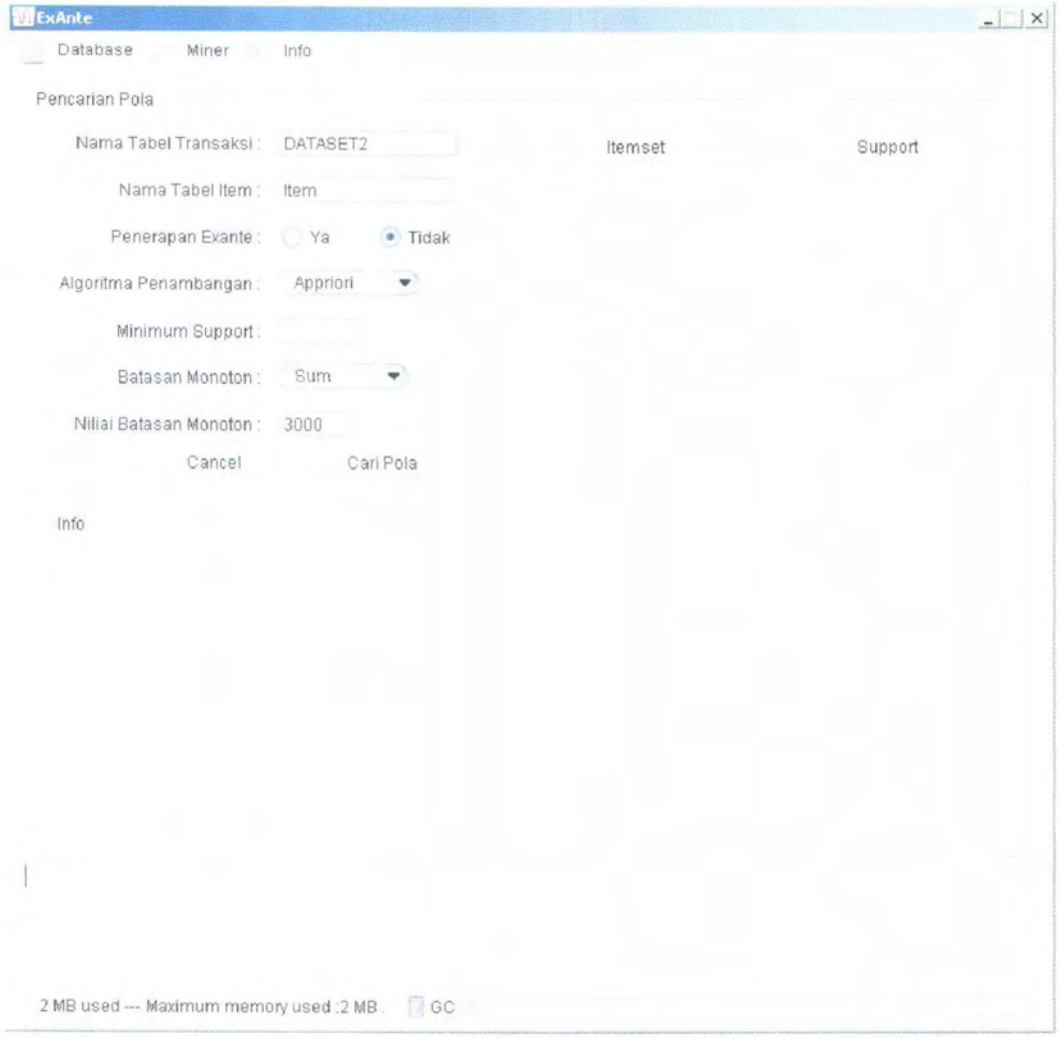
Fitur ini merupakan fitur yang menyediakan fungsi untuk melakukan proses reduksi pada dataset transaksi.dengan menerapkan metode *ExAnte* Antarmuka untuk fitur ini ditunjukkan pada gambar 4.5



Gambar 4. 5 Antarmuka penerapan metode Exante

- **Pencarian Pola dengan Algoritma**

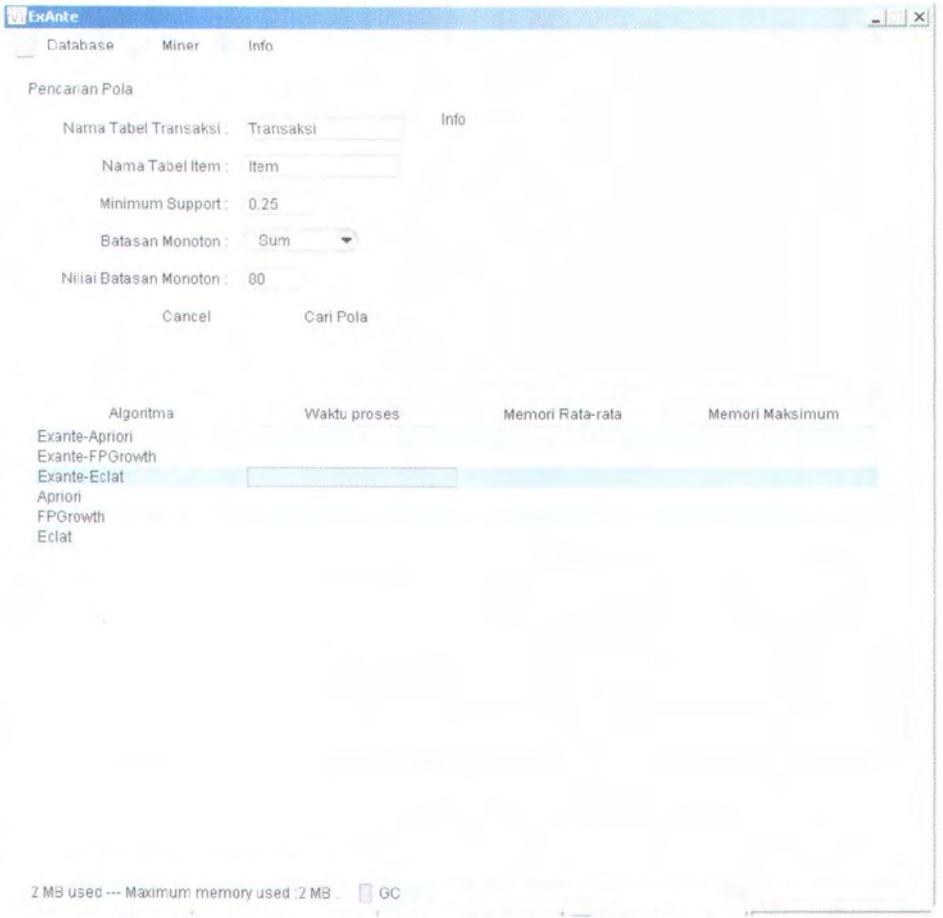
  Fitur ini berupa sebuah panel yang berisi field masukan parameter pencarian pola beserta pilihan algoritma yang ingin digunakan. Selain field masukan, panel ini juga dilengkapi dengan sebuah tabel untuk menampilkan pola-pola yang dihasilkan oleh proses pencarian pola. Antarmuka untuk fitur ini dapa dilihat pada gambar 4.6



Gambar 4. 6 Antarmuka pencarian pola

- **Evaluasi Kinerja Algoritma**

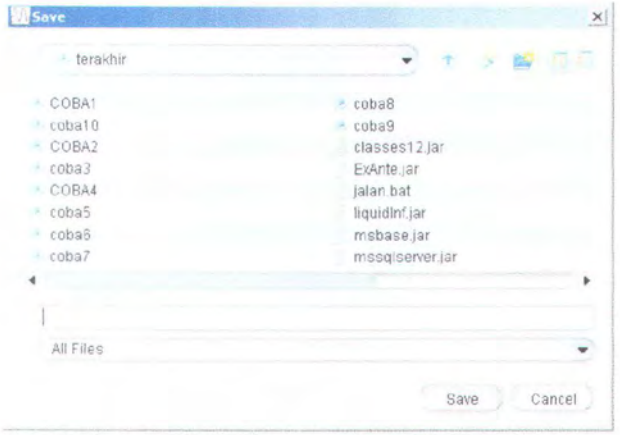
Antarmuka untuk fitur ini berupa sebuah panel yang berisi field-field masukan untuk pengguna diserta sebuah tabel yang berfungsi untuk menampilkan informasi proses masing-masing algritma. Antarmuka untuk fitur ini ditunjukkan pada gambar 4.7



Gambar 4. 7 Antarmuka Evaluasi kinerja algoritma

- **Simpan Hasil**

Fitur isimpan hasil memiliki antarmuka berupa sebuah dialog pemilihan file yang ditujukan bagi pengguna untuk memilih file tujuan penyimpanan. Antarmuka untuk fitur ini dapat ditunjukkan pada gambar berikut :



Gambar 4. 8 Antarmuka simpan hasil

BAB V

UJI COBA DAN ANALISIS HASIL

BAB V

UJI COBA DAN ANALISIS HASIL

Pada bagian ini dijelaskan tentang pelaksanaan uji coba terhadap sistem. Uji coba ini berhubungan erat dengan tujuan pembuatan aplikasi dan analisa kebutuhan. Berdasarkan uji coba yang dilaksanakan, dilakukan analisis dan evaluasi terhadap hasil yang diperoleh.

5.1. Lingkungan uji coba

Uji coba dilakukan menggunakan komputer PC dengan prosesor Intel(R) Pentium 4 3GHz dengan memori 256 MB. Sedangkan server basis data tujuan yang digunakan dalam uji coba adalah Oracle 9.2

5.2. Data uji coba

Dataset yang digunakan dalam uji coba aplikasi ini terdiri dari 2 macam. Yaitu data uji kebenaran dan data uji kinerja aplikasi. Berikut ini adalah penjelasan untuk masing-masing data uji coba :

5.2.1. Data uji kebenaran aplikasi

Uji kebenaran ini, berupa pengujian aplikasi terhadap masukan data dengan ukuran data kecil. Untuk itu, digunakan contoh dataset dari [FBC-03A]. isi dari dataset ini ditunjukkan pada table di bawah ini :

Tabel 5. 1 Tabel transaksi awal

TID	Itemset
1	b,c,d,g
2	a,b,d,e
3	b,c,d,g,h
4	a,e,g
5	c,d,f,g
6	a,b,c,d,e
7	a,b,d,f,g,h
8	b,c,d
9	b,e,f,g

Sedangkan tabel item sebagai referensi item-item diatas ditunjukkan pada tabel 5.2 :

Tabel 5. 2 Tabel Item

Item	Price
a	5
b	8
c	14
d	30
e	20
f	15
g	6
h	12

5.2.2. Data uji kinerja aplikasi

Untuk menguji kinerja aplikasi, digunakan beberapa dataset. Dataset-dataset tersebut dibangkitkan dengan mengimplementasikan aplikasi pembangkitan transaksi Ardhita P. Dataset yang digunakan antara lain :

Tabel 5. 3 Dataset uji coba

Nama Tabel Transaksional	T	I	D
T15_I10_D1K	15	10	1K
T10_I2_D100K	10	2	100K
T10_I10_D200K	10	10	200K
T20_I10_D500K	20	10	500K

Pada dasarnya uji coba akan selalu dilakukan terhadap beberapa kondisi yang berbeda. Sehingga akan didapatkan hasil yang dapat digunakan untuk menganalisa kemampuan dari perangkat lunak.

5.3. Skenario uji coba

Beberapa uji coba akan dilakukan dengan menggunakan data uji coba yang berbeda-beda. Uji coba akan menerapkan 3 buah skenario yaitu :.

5.3.1. Skenario I : Uji kebenaran luaran aplikasi

Skenario pengujian ini berupa penghitungan luaran metode *ExAnte* secara toritis dengan cara penghitungan manual menggunakan alat bantu alat bantu Microsoft excel.

5.3.2. Skenario II : Pengujian pengaruh perubahan nilai batasan terhadap reduksi ukuran dataset

Skenario pengujian ini berupa penerapan metode *ExAnte* dengan beberapa variasi batasan terhadap dataset. Tujuan dari pengujian ini adalah untuk melihat seberapa besar reduksi yang dapat dilakukan terhadap dataset baik itu reduksi item frequent maupun reduksi jumlah transaksi.

5.3.3. Skenario III : Pengujian pengaruh penerapan metode Exante terhadap proses penambangan pola

Skenario pengujian ini berupa pembandingan kinerja proses penambangan pola oleh algoritma dengan dan tanpa adanya penerapan metode *ExAnte*. Tujuan dari pengujian ini adalah untuk melihat sebearapa besar pengaruh penerapan metode *ExAnte* terhadap sumber daya yang dibutuhkan dalam proses penambangan.

5.4. Pelaksanaan dan analisis hasil uji coba

Pada bagian ini dijelaskan mengenai pelaksanaan uji coba sesuai dengan skenario-skenario yang telah dijabarkan sebelumnya. Selain itu, juga dilakukan analisis terhadap hasil uji coba tersebut.

5.4.1. Uji coba skenario I

Uji coba skenario I dilakukan untuk menguji kebenaran luaran yang dihasilkan oleh aplikasi dengan cara penghitungan manual. Untuk penghitungan secara manual dengan alat bantu Microsoft excel, maka alur prosesnya adalah sebagai berikut (minsup= 4/9 dan sum≥45):

5.4.1.1. Penghitungan manual

- Iterasi 0 (awal)

Tabel 5. 4 Pengecekan transaksi awal

TID	Itemset	sum	status
1	b,c,d,g	58	lolos
2	a,b,d,e	63	lolos
3	b,c,d,g,h	70	lolos
4	a,e,g	31	tidak lolos
5	c,d,f,g	65	lolos
6	a,b,c,d,e	77	lolos
7	a,b,d,f,g,h	76	lolos
8	b,c,d	52	lolos
9	b,e,f,g	49	lolos

Transaksi yang lolos di atas dipindah ke tabel sementara sebagai berikut :

Tabel 5. 5 Tabel transaksi yang lolos batasan monoton

TID	Itemset	sum
1	b,c,d,g	58
2	a,b,d,e	63
3	b,c,d,g,h	70
5	c,d,f,g	65
6	a,b,c,d,e	77
7	a,b,d,f,g,h	76
8	b,c,d	52
9	b,e,f,g	49

- Iterasi 1

Tabel sementara tersebut mengalami penghapusan item-item yang tidak frequent sebagai berikut :

Tabel 5. 6 Pengecekan tabel iterasi 1

TID	Itemset	sum	status
1	b,c,d,g	58	lolos
2	b,d	38	Tidak lolos
3	b,c,d,g	58	lolos
5	c,d,g	50	lolos
6	b,c,d	52	lolos
7	b,d,g	44	Tidak lolos
8	b,c,d	52	lolos
9	b,g	14	Tidak lolos

Setelah itu transaksi yang tidak lolos dihapus dari tabel menjadi berikut ini:

Tabel 5. 7 Tabel transaksi yang lolos iterasi 1

TID	Itemset	Sum
1	b,c,d,g	58
3	b,c,d,g	58
5	c,d,g	50
6	b,c,d	52
8	b,c,d	52

- Iterasi 2

Penghapusan item-item non frequent :

Tabel 5. 8 Pengecekan tabel iterasi 2

TID	Itemset	sum	status
1	b,c,d	52	lolos
3	b,c,d	52	Lolos
5	c,d	44	Tidak lolos
6	b,c,d	52	Lolos
8	b,c,d	52	Lolos

Setelah itu transaksi yang tidak lolos dihapus dari tabel menjadi berikut ini:

Tabel 5. 9 Tabel transaksi yang lolos iterasi 2

TID	Itemset	Sum
1	b,c,d	52
3	b,c,d	52
6	b,c,d	52
8	b,c,d	52

Tabel di atas adalah hasil akhir dari proses *ExAnte* sebab tidak ada lagi item maupun transaksi yang dapat dihilangkan.

5.4.1.2. Luaran Aplikasi

Untuk menguji kebenaran luaran aplikasi, dilakukan uji coba dengan cara memasukkan dataset yang sama dengan dataset diatas beserta nilai batasan yang sama pula. Tabel hasil akhir yang dihasilkan oleh aplikasi ditunjukkan pada tabel 5.10.

Tabel 5. 10 Tabel hasil akhir luaran aplikasi

TID	PID
1	2
1	3
1	4
2	2
2	3
2	4
3	2
3	3
3	4
4	2
4	3
4	4

5.4.1.3. Analisis hasil uji coba skenario I

Dari analisa proses dan tabel hasil proses aplikasi di atas, dapat dilihat bahwa hasil antara keduanya adalah sama sehingga dapat disimpulkan bahwa luaran dari aplikasi telah terbukti kebenarannya.

5.4.2. Uji coba skenario II

Skenario 1 menerapkan perubahan nilai batasan pada dataset dan melihat efeknya terhadap jumlah transaksi dan item pada hasil akhir. Pelaksanaan untuk masing-masing dataset adalah sebagai berikut :

5.4.2.1. Dataset T15_I10_D1K :

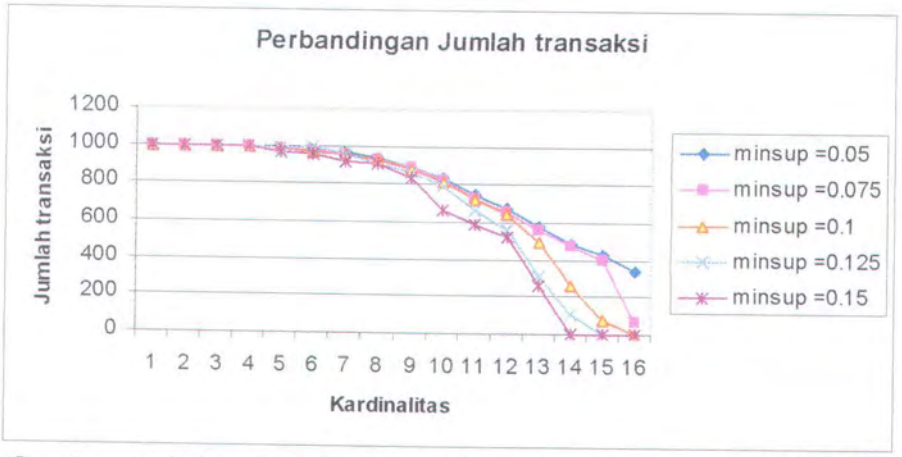
Variasi nilai minimum support :

0.05, 0.075, 0.1, 0.125, 0.15

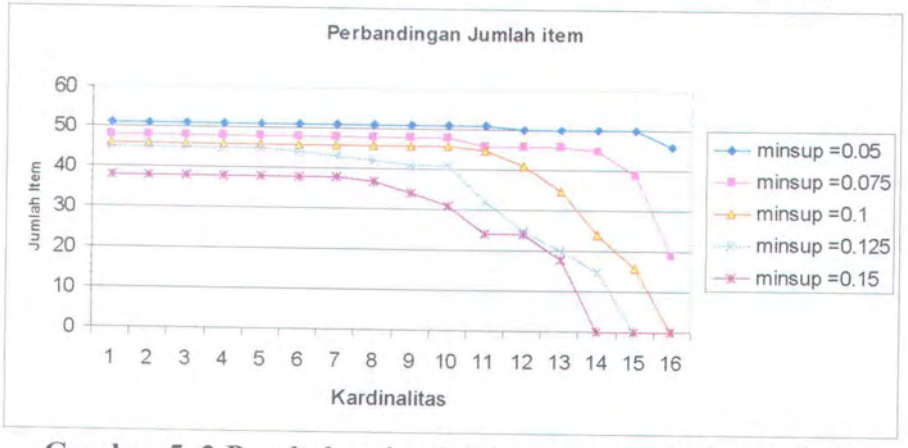
Variasi nilai Batasan monoton Cardinality($|X| \geq N$) :

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16

Setelah dilakukan uji coba dengan menggunakan aplikasi, reduksi yang terjadi dapat digambarkan pada gambar 5.1. dan 5.2



Gambar 5. 1 Perubahan jumlah transaksi dataset T15_I10_D1K



Gambar 5. 2 Perubahan jumlah item dataset T15_I10_D1K

5.4.2.2. Dataset T10_I2_D100K :

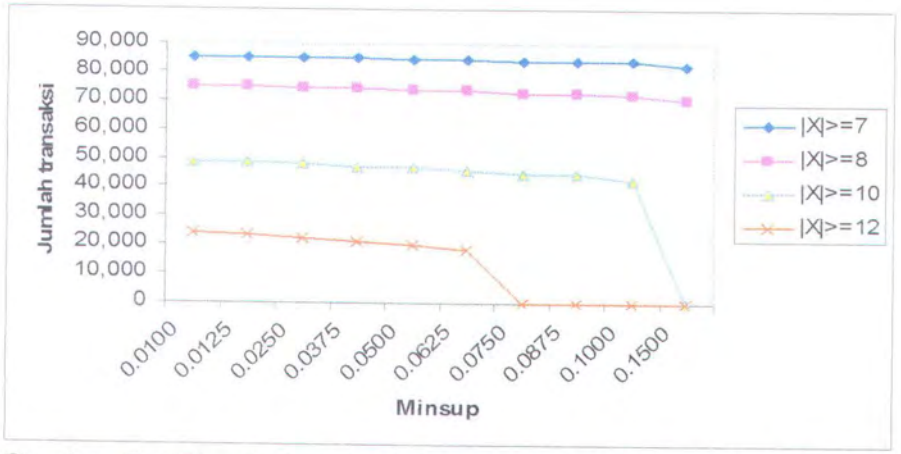
Variasi nilai minimum support :

0.0100, 0.0125, 0.0250, 0.0375, 0.0500, 0.0625, 0.0750, 0.0875, 0.1000, 0.1500

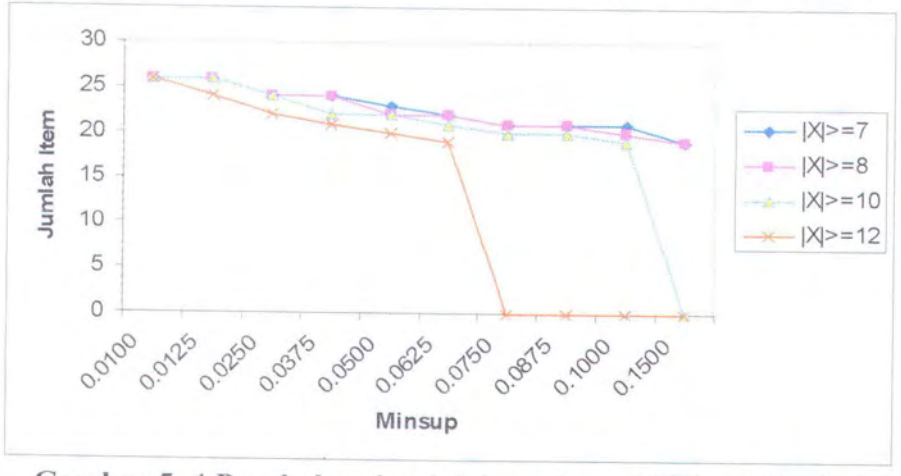
Variasi nilai Batasan monoton Cardinality($|X| \geq N$) :

7, 8, 10, 12

Setelah dilakukan uji coba dengan menggunakan aplikasi, reduksi yang terjadi dapat digambarkan pada gambar 5.3 dan 5.4.



Gambar 5. 3 Perubahan jumlah transaksi dataset T10_I2_D100K



Gambar 5. 4 Perubahan jumlah item dataset T10_I2_D100K

5.4.2.3. Dataset T10_I10_D200K:

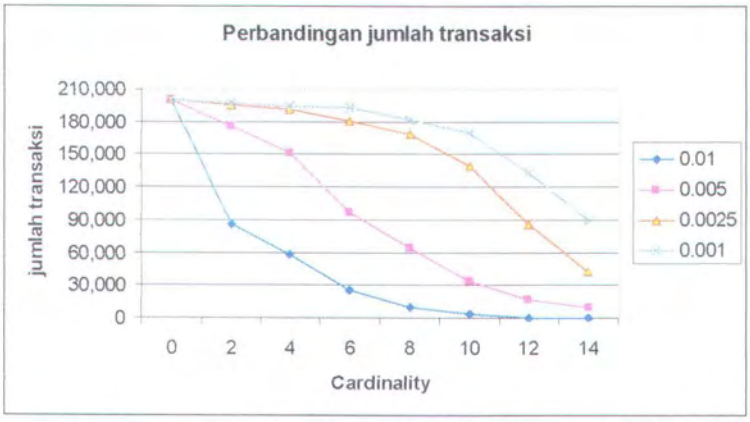
Variasi nilai minimum support :

0.01, 0.005, 0.0025, 0.001

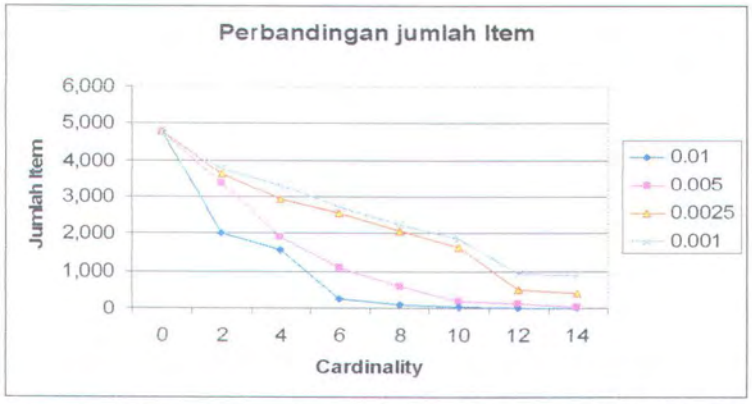
Variasi nilai Batasan monoton Cardinality($|X| \geq N$) :

0, 2, 4, 6, 8, 10, 12, 14

Setelah dilakukan uji coba dengan menggunakan aplikasi, reduksi yang terjadi dapat digambarkan pada gambar 5.5 dan 5.6.



Gambar 5. 5 Perubahan jumlah transaksi dataset T10_I10_D200K



Gambar 5. 6 Perubahan jumlah item dataset T10_I10_D200K

5.4.2.4. Dataset T20_I10_D500K:

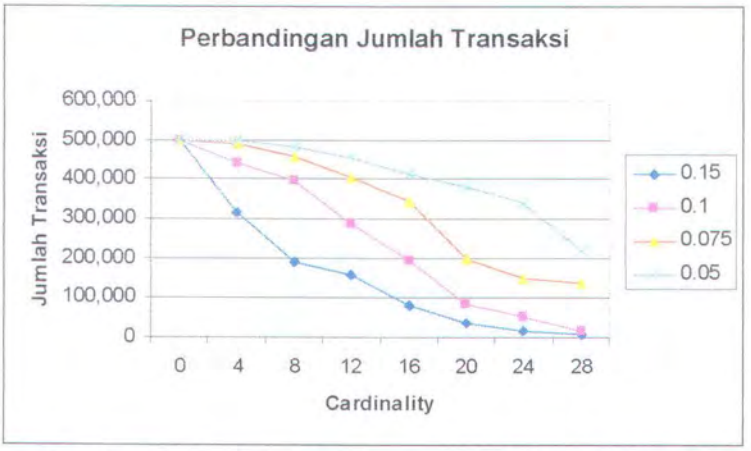
Variasi nilai minimum support :

0.15, 0.1, 0.075 , 0.05

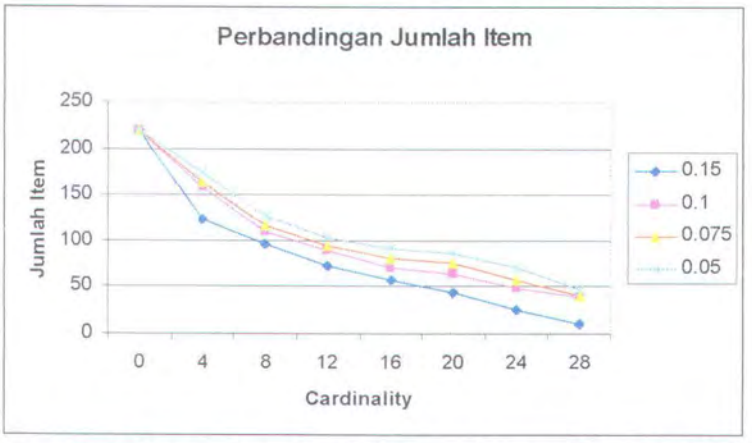
Variasi nilai Batasan monoton Cardinality(|X|≥N) :

0, 4, 8, 12, 16, 20, 24, 28

Setelah dilakukan uji coba dengan menggunakan aplikasi, reduksi yang terjadi dapat digambarkan pada gambar 5.7 dan 5.8.



Gambar 5. 7 Perubahan jumlah transaksi dataset T20_I10_D500K



Gambar 5. 8 Perubahan jumlah item dataset T20_I10_D500K

5.4.2.5. Analisis hasil uji coba skenario II

Dalam analisis perubahan nilai batasan pada dataset, didapatkan bahwa penerapan metode *ExAnte* pada sebuah dataset transaksi mengakibatkan adanya reduksi ukuran dataset. Reduksi ukuran dataset tersebut terjadi akibat adanya pengurangan transaksi serta item pada tiap iterasi yang dilakukan oleh *ExAnte*.

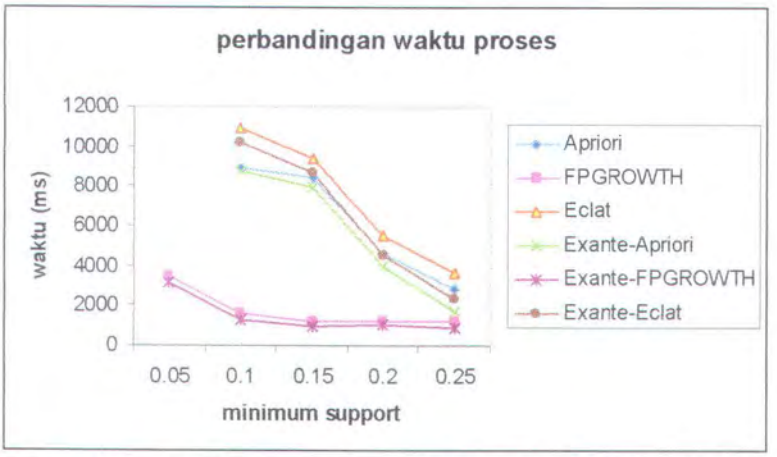
Mengacu pada hasil uji coba pada keempat dataset yang digunakan, maka hubungan antara batasan anti monoton dengan reduksi data dapat dikatakan berbanding lurus. Semakin besar nilai *minimum support* yang diberikan, maka semakin besar pula reduksi yang berakibat pada semakin berkurangnya jumlah item. Begitu pula dengan batasan monoton. Semakin selektif batasan monoton yang diberikan, misal bertambahnya nilai kardinalitas, juga akan berakibat bertambahnya persentase reduksi transaksi.

5.4.3. Uji coba skenario III

Skenario 3 berupa penerapan metode *ExAnte* pada dataset yang akan melalui proses pencarian pola. Setelah itu, informasi kinerja proses tersebut dibandingkan dengan kinerja proses apabila dataset tersebut tidak melalui praproses terlebih dahulu. Proses pencarian pola menggunakan 3 macam algoritma pencarian pola yaitu Apriori, FPGrowth dan Eclat. Yang dibandingkan dalam skenario ini adalah waktu proses, rata-rata penggunaan memori serta penggunaan memori maksimal. Untuk pelaksanaan skenario ini, cukup dilakukan terhadap 2 buah dataset yaitu dataset T15_I10_D1K dan dataset T20_I10_D500K yaitu dataset yang terkecil dan terbesar. Pelaksanaan pada dataset kecil untuk melihat pola perubahan yang terjadi secara global. Sedangkan pelaksanaan uji coba pada dataset besar untuk melihat seberapa signifikan perubahan yang terjadii. Pelaksanaan untuk masing-masing dataset adalah sebagai berikut :

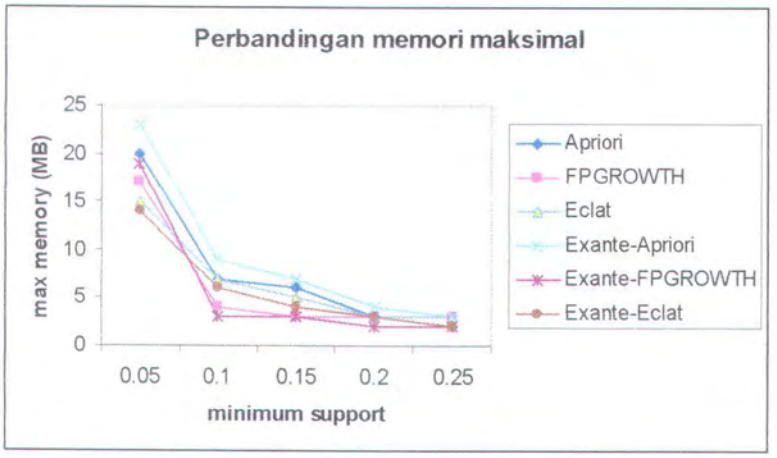
5.4.3.1. Dataset T15_I10_D1K :

Skenario uji coba untuk dataset ini menggunakan 5 nilai minimum support yang berbeda. Yaitu : 0.05, 0.1, 0.15, 0.2, dan 0.25 dengan nilai batasan monoton berupa sum(Price) $\geq$ 2000. Perbandingan waktu proses untuk tiap kombinasi algoritma ditunjukkan pada gambar berikut :

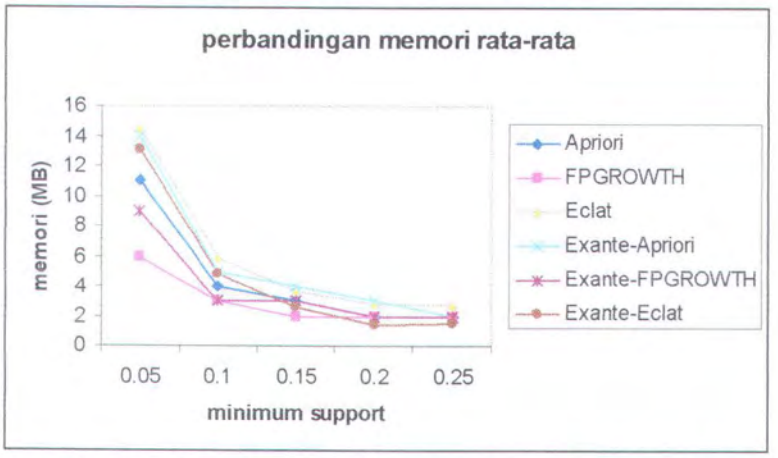


Gambar 5. 9 Perbandingan waktu proses pada dataset T15_I10_D1K

Untuk perbandingan kebutuhan memori dapat ditunjukkan pada gambar berikut :

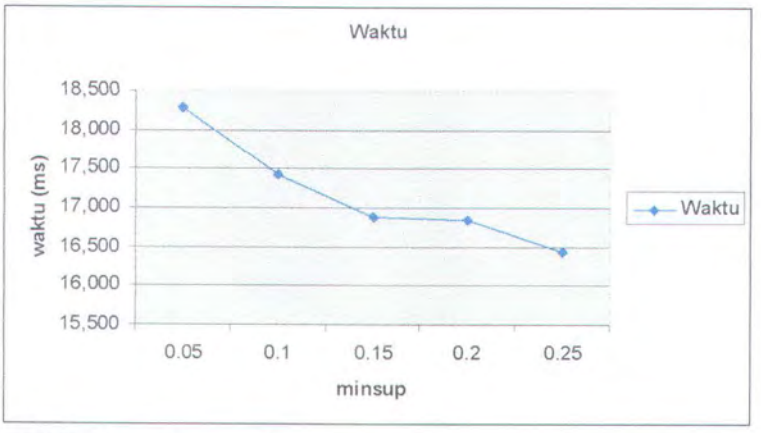


Gambar 5. 10 Perbandingan memori maksimum pada dataset T15_I10_D1K



Gambar 5. 11 Perbandingan memori rata-rata pada dataset T15_I10_D1K

Catatan waktu yang dibutuhkan untuk proses *ExAnte* ditunjukkan pada gambar 5.12. berikut ini :

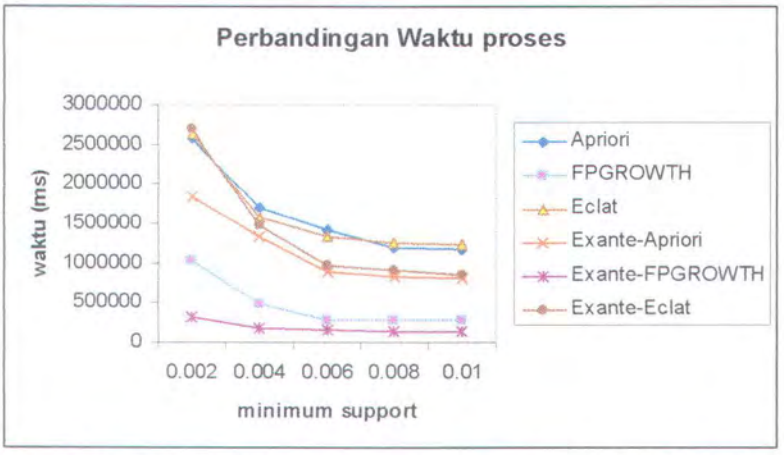


Gambar 5. 12 Kebutuhan waktu proses Exante pada dataset T15_I10_D1K

Untuk kebutuhan memori, proses *ExAnte* membutuhkan memori dengan nilai yang cenderung konstan. Untuk dataset T15_I10_D1K, kebutuan memori yang digunakan sebesar 2 MB

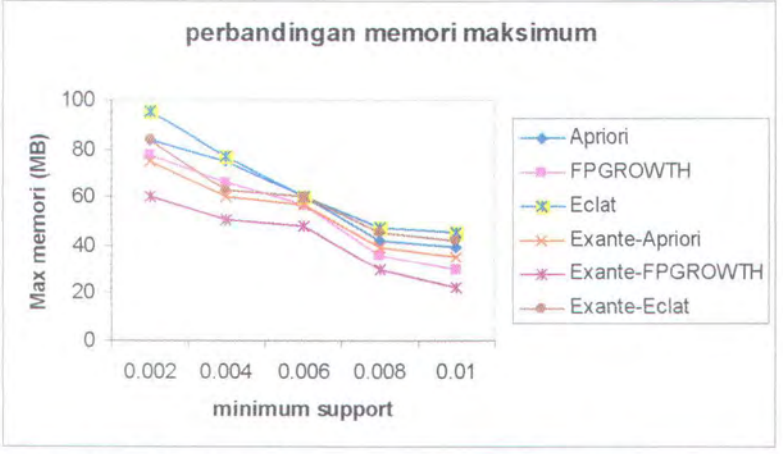
5.4.3.2. Dataset T20_I10_D500K:

Skenario uji coba untuk dataset ini menggunakan 5 nilai minimum support yang berbeda. Yaitu : 0.002, 0.004, 0.006, 0.008, dan 0.01 dengan nilai batasan monoton berupa sum(Price) $\geq$ 6000. Perbandingan waktu proses untuk tiap kombinasi algoritma ditunjukkan pada 5.12.

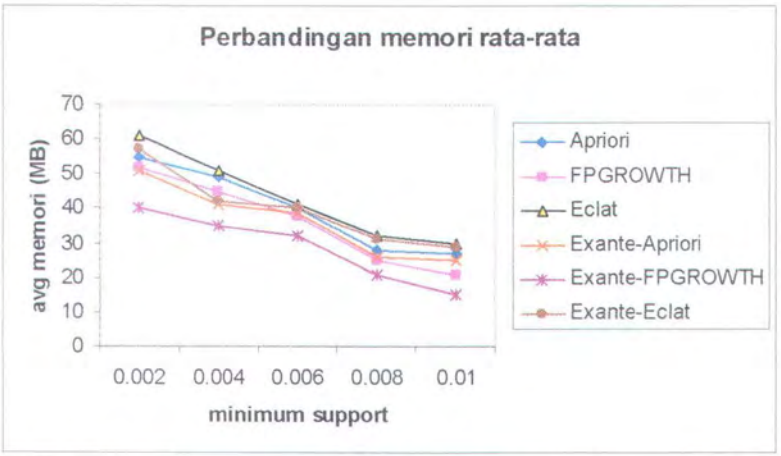


Gambar 5. 13 Perbandingan waktu proses pada dataset T20_I10_D500K

Untuk perbandingan kebutuhan memori dapat ditunjukkan pada gambar berikut :

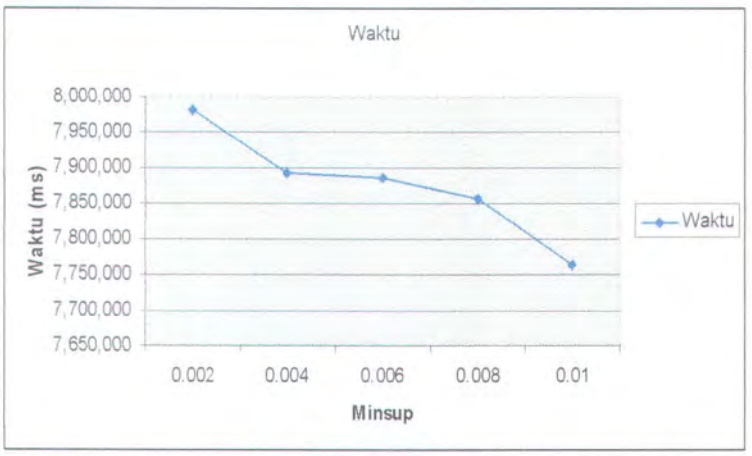


Gambar 5. 14 Perbandingan memori maksimum pada dataset T20_I10_D500K



Gambar 5. 15 Perbandingan memori rata-rata pada dataset T20_I10_D500K

Catatan waktu yang dibutuhkan untuk proses *ExAnte* ditunjukkan pada gambar 5.16 berikut ini :



Gambar 5. 16 Kebutuhan waktu proses Exante pada dataset T20_I10_D500K

Untuk kebutuhan memori, proses *ExAnte* membutuhkan memori dengan nilai yang cenderung konstan. Untuk dataset T20_I10_D500K, kebutuan memori yang digunakan sebesar 116 MB

5.4.3.3. Analisis hasil uji coba skenario III

Dalam analisis perbandingan kinerja proses dengan dan tanpa menggunakan metode *ExAnte*, didapatkan bahwa penerapan metode *ExAnte* pada sebuah dataset transaksi mengakibatkan adanya penurunan waktu proses oleh algoritma pencarian pola. Selain itu, didapatkan bahwa telah terjadi

penurunan penggunaan memori baik memori rata-rata maupun memori maksimal.

Dari uji coba yang dilakukan dapat dilihat bahwa algoritma FPGrowth memiliki kinerja yang lebih baik dibanding algoritma Apriori dan Eclat. Kinerja tersebut semakin bertambah, ketika metode *ExAnte* diterapkan sebagai praproses data.

BAB VI

SIMPULAN

BAB VI

SIMPULAN

Dalam tugas akhir ini telah berhasil diimplementasikan metode *ExAnte* sebagai metode praproses data dalam sebuah aplikasi pencarian pola asosiasi. Metode ini berfungsi untuk mereduksi ukuran dataset sehingga dapat mengurangi ruang pencarian dari agoritma penggalian pola. Pengurangan ini mengakibatkan meningkatnya kinerja dari algoritma penggalian pola tersebut.

Dari beberapa hasil uji coba aplikasi dapat ditarik dua buah simpulan seperti berikut :

a. *ExAnte* sebagai algoritma praproses data, dapat mengurangi ruang pencarian pada penambangan pola dan dapat mengurangi waktu eksekusi dalam penambangan pola, sehingga pada akhirnya dapat membuat proses komputasi pencarian frequent itemset menjadi relatif lebih cepat.

b. Dari beberapa hasil skenario uji kinerja, untuk berbagai jenis batasan yang berbeda pada beragam variasi dataset, algoritma *ExAnte* menunjukkan peningkatan kinerja yang baik dan membantu dalam memungkinkan pencarian pola yang tidak dapat dicari dengan hanya menggunakan algoritma pencarian pola biasa.

DAFTAR PUSTAKA

DAFTAR PUSTAKA

[ADT-02] Adhita Pratiwi " Perancangan dan pembuatan perangkat lunak pembangkit data yang mengikuti pola distribusi poisson untuk pencarian kaidah asosiasi dan pola sekuensial " Tugas Akhir, Jurusan Teknik Informatika FTIF ITS Surabaya. 2002.

[AGR-93] R.Agrawal, T. Imielinski, and A.N. Swami. "Mining Association Rules Between Sets of Items in Large Databases", Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, 1993.

[FBC-05] Francesco Bonchi, Fosca Giannotti, Alessio Mazzanti, and Dino Pedreschi, "ExAnte: A Preprocessing Method for Frequent-Pattern Mining", Journal of IEEE Computer Society 1541-1672, 2005.

[FBC-03A] F. Bonchi et al., "ExAnte: Anticipated Data Reduction in Constrained Pattern Mining", Proc. 7th European conf Principles and Practice of Knowledge Discovery in Databases, Springer-Verlag, 2003.

[FBC-03B] Francesco Bonchi, Fosca Giannotti, Alessio Mazzanti, and Dino Pedreschi, "ExAMiner: Optimized level-wise Frequent Pattern Mining with Monotote Constrains". Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03).2003

[JWH-01] Jiawei Han dan Micheline Kamber, "Data Mining: Concepts and Techniques", Morgan Kaufmann Publishers,2001.

[JWH-00] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," Proceedings of ACM SIGMOD Dallas, 2000.

[MSC-96] Ming-Syan Chen, Jiawei Han, Phillip S Yu, "Data Mining: An Overview from a Database Perspective", IEEE Transactions on Knowledge and Data Engineering, Vol. 8 No. 6, December 1996.

[MZK-97] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. "New Algorithms for Fast Discovery of Association Rules." Proc. 3rd Int. Conf. on Knowledge Discovery *and Data Mining (KDD'97), 283–296.* 1997.