BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Penelitian yang dilakukan oleh (Rahayu, 2014), menerapkan teknik data mining dengan metode algoritma *Apriori* untuk menganalisis penjualan tiket pesawat pada perusahaan jumbo travel medan. Peneliti berfokus pada mencari kecenderungan tiket pesawat yang dijual secara bersamaan. Dalam penelitian tersebut disebutkan bahwa *data mining* sangat berguna untuk mengetahui pola frekuensi penjualan tiket pesawat. Sehingga data mining merupakan teknologi yang sangat berguna bagi perusahaan untuk menemukan informasi yang penting dari gudang data.

Sama halnya seperti penelitian yang dilakukan oleh (Yanto & Khoiriah, 2015), teknik *data mining* dengan metode algoritma *Apriori* diterapkan untuk menentukan pola pembelian obat. Peneliti menemukan bahwa pola pembelian obat dapat dihasilkan dengan menggunakan teknik *data mining*. Pola pembelian ditentukan dengan melihat hasil kecenderungan konsumen membeli obat berdasarkan kombinasi 2 *itemset.* Hasil dari penelitian dapat digunakan untuk membantu pengaturan tata letak obat.

Dalam penelitian yang dilakukan oleh (Haryanto, 2011), algoritma *Apriori* dimanfaatkan untuk menganalisa penjualan suku cadang sepeda motor yang disebut juga dengan *Market Basket Analysis*. Penelitian ini menghasilkan bahwa hubungan relasional tiap produk yang dibeli dapat dirumuskan dengan *Association Rules Mining.* Hasil dari *Market Basket Analysis* digunakan untuk penawaran pada konsumen.

(Nurdin, 2015) membangun sebuah sistem yang digunakan untuk menganalisis penjualan barang pada *Supermarket* Sejahtera Lhokseumawe. Penelitian tersebut menerapkan *data mining* dengan menggunakan metode algoritma *Apriori.* Berdasarkan penelitiannya, algoritma *Apriori* dapat menghasilkan *frequent itemset* serta *association rules.* Semakin kecil batas transaksi serta *minimum confidence* yang ditentukan akan berpengaruh pada *rules* yang dihasilkan, akan tetapi waktu pemrosesan akan bertambah lebih lama.

Akan tetapi penelitian yang dilakukan oleh (Audrint, Atastina, & Dayawati, 2011) menemukan bahwa apabila nilai minimum support kecil maka hasil akan mencapai nilai optimal. Dalam hal ini Algoritma *AprioriTid* mengungguli performa Algoritma *Apriori.*

Berdasarkan penelitian tentang perbandingan algoritma yang dilakukan oleh (Kurana & Sharma, 2013), Algoritma *Apriori* dan *AprioriTid* memiliki kelebihan dan kekurangan masing-masing. *Apriori* memiliki waktu untuk *initial phase* lebih cepat dibandingkan dengan *AprioriTid*, namun pada fase berikutnya performa menurun. Berbanding terbalik dengan *AprioriTid* yang memiliki initial phase yang lebih lama, namun akan lebih cepat pada fase berikutnya. Untuk mengatasi masalah tersebut, Algoritma *AprioriHybrid* didesain (Agrawal & Srikant, 1994). Algoritma *AprioriHybrid* mengungguli performa dari Algoritma *Apriori* serta *AprioriHybrid*.

Dalam penelitian ini, algoritma *AprioriHybrid* akan diimplementasikan untuk menganalisa transaksi penjualan. Transaksi penjualan berasal dari data penjualan perusahaan AmigoGroup. Hasil dari analisa tersebut merupakan aturan asosiasi antar item.

2.2 Landasan Teori

2.2.1 Data Mining

Data mining merupakan teknik yang digunakan untuk mendapatkan informasi dari kumpulan data dalam skala yang sangat besar (Han & Kamber, 2011). Proses tersebut dinamakan dengan *information retrieval*. Informasi yang didapatkan kemudian digunakan untuk keperluan dalam dunia nyata. Misalnya digunakan sebagai bahan pertimbangan dalam pengambilan keputusan perusahaan. Proses penerimaan informasi harus dapat dilakukan dalam waktu yang singkat. Oleh karena itu, kecepatan proses merupakan sebuah aspek yang penting dalam *Data Mining*. Hal tersebut dikarenakan skala data yang besar dan terus berkembang. Selain efisiensi pemrosesan, sistem penyimpanan data merupakan aspek yang penting. Oleh karena itu proses *information retrieval* dalam *data mining* mengkombinasikan prinsip pengenalan pola, prinsip statistik, serta *rule* yang dihasilkan dari *machine learning*.

Data mining juga disebut dengan *KDD* (*Knowledge Discovery from Data*) (Han & Kamber, 2011). *KDD* merupakan sebuah proses iteratif yang terdiri dari :

1. *Preprocessing (Data Cleaning & Data Integration)*

   Data cleaning merupakan tahap di mana data dibersihan dari *noise* yang dapat berupa tipografi atau duplikasi.

2. *Data Selection*

   Tahap ini menyeleksi data dari database yang memiliki relevansi dengan proses *data mining.*

3. *Data Transformation*

   Dalam tahap ini, data diolah dengan operasi tertenu sehingga sesuai untuk diproses ke dalam tahapan data mining. Operasi yang dilakukan antara lain operasi agregat, ataupun statistik.

4. *Data Mining*

   Dalam tahap ini data diolah dengan menggunakan metode tertentu untuk mendapatkan informasi /pola yang dicari. Metode tersebut dapat bervariasi sesuai dengan informasi /pola yang dicari.

5. *Pattern Evaluation*

   Informasi /pola yang didapatkan kemudian di evaluasi apakah sesuai dengan fakta / hipotesis atau tidak. Pengujian/evaluasi dilakukan berdasarkan *interestingness measures*.

6. *Knowledge Presentation*

   Pola /Informasi yang didapatkan kemudian harus divisualisasikan ke dalam bentuk yang mudah dimengerti oleh semua pihak.

2.2.2 Data Warehouse

Data warehouse merupakan basis data relasional yang digunakan untuk kepentingan analisis. Basis data tersebut tidak ditujukan untuk kepentingan transaksional seperti basis data pada umumnya. Data yang ada berasal dari kumpulan data-data transaksional dari berbagai sumber. *Data warehouse* mengalami proses *data cleaning*, *data integration*, *data transformation*, *data loading*, dan *periodic data refreshing* (Han & Kamber, 2011). *Data warehouse* memiliki 4 karakteristik yaitu *Subject Oriented*, *Integrated*, *Nonvolatile*, dan *Time Variant*. *Subject oriented* berarti *data warehouse* didesain untuk membantu menganalisa masalah tertenu. Misalnya untuk menggali informasi dari penjualan sebuah perusahaan. Penjualan dalam konteks ini merupakan subjek yang di teliti. Karena sumber data dalam data warehouse dapat berasal dari eksternal. Maka data warehouse harus mampu mengolah data tersebut ke dalam format yang konsisten. Oleh karena itu, data warehouse harus terintegrasi /*integrated*. Data yang ada dalam data warehouse memiliki sifat *nonvoaltile*. Hal tersebut berarti data yang masuk tidak akan pernah berubah seiring berjalannya waktu. Untuk dapat

menganalis trend berdasarkan data, dibutuhkan data dalam skala yang besar. Untuk itu data warehouse harus terfokus pada perubahan data, oleh karena itu data warehouse bersifat *time variant*. Proses yang dilakukan dalam data warehouse terdiri dari *ETL* , *Dimensional Model* , *Dimensional Table* serta *OLAP*. *Dimensional model* digunakan untuk membentuk struktur data dan membentuk hasil return dari query. OLAP merupakan *Online Analytical Processing*. OLAP digunakan untuk melakukan *query* terhadap data. OLAP memiliki peran penting karena harus dapat mengembalikan data query dengan cepat. Salah satu infrastruktur perangkat lunak yang dapat digunakan adalah *Hadoop*. *Hadoop* merupakan projek *open source* yang dikelola oleh *apache foundation*. *Hadoop* terdiri dari dua buah komponen. HDFS (*Hadoop File System*) serta *Map Reduce*.

2.2.3 Big Data

Big data merupakan kumpulan data yang sangat besar terdiri dari berbagai macam data yang bertambah semakin cepat seiring berjalannya waktu. Diperlukan teknik khusus untuk mengekstrak informasi dari *big data* (Stubbs, 2014).

2.2.4 Association Rule Mining

Association rule mining merupakan alat yang digunakan untuk menemukan aturan asosiasi dari kombinasi dari item yang sering paling muncul (Brown, 2014). *Association rule mining* juga disebut dengan *Market Basket Analysis* (Kusrini & Taufiq, 2009), yaitu metode untuk menemukan pola yang sama dalam jumlah yang yang banyak, serta menemukan asosiasi, korelasi atau struktur yang umum antar data satu dengan yang lainnya dalam suatu kumpulan data. Aplikasi association rules mining dapat juga berupa *cross-marketing catalog design*, *loss-leader analysis*, *clustering*, *classification*, dan lain-lain. Dalam *association rule* terdapat dua buah *constraint* yang disebut dengan *support* dan *confidence*. *Support* merupakan rasio munculnya kumpulan item dalam sebuah *item set* dibandingkan dengan jumlah kumpulan *item* dalam *item set* tersebut.

Misalkan kumpulan item {susu, madu, telur} memiliki kemunculan 2 kali dari jumlah total kumpulan item 20. Maka nilai *support* dari {susu, madu, telur} adalah 2/20 = 0.1. *Confidence* merupakan persentase kejadian dimana sebuah transaksi item X juga mengandung item Y. Rumus dari *confidence* dan *support* adalah sebagai berikut :

$$support(X \Longrightarrow Y) = P(X \cup Y)$$

$$confidence(X \Longrightarrow Y) = P(X|Y)$$

Terdapat beberapa algoritma yang dapat digunakan dalam *associatio rule* misalnya *FP-Growth*, *AIS*, *SETM*, *Apriori*, *AprioriTid*, *Eclat*, *Partition*, *AprioriHybrid*, dan lain-lain.

2.2.5 Algoritma Apriori Hybrid

Algoritma AprioriHybrid merupakan algoritma penggabungan dari Algoritma *Apriori* dan *AprioriTID* (Agrawal & Srikant, 1994). Kedua algoritma tersebut memiliki karakteristik yang sama, yaitu meng-*generate candidate* dan meghitung *itemsets*. Perbedaan dari kedua algoritma tersebut adalah performa yang dihasilkan. *Apriori* memiliki waktu untuk melakukan fase awal yang lebih cepat, namun akan menurun pada fase-fase berikutnya. *AprioriTid* memiliki waktu untuk melakukan fase awal yang lebih lama, namun pada fase berikutnya performa akan meningkat (Agrawal & Srikant, 1994). Pada dasarnya cara kerja algoritma *AprioriHybrid* adalah menjalankan Apriori pada fase awal kemudian berpindah menjalankan AprioriTid pada fase berikutnya apabila *set* kombinasi dapat dimasukkan dalam *memory* (Agrawal & Srikant, 1994).

2.2.6 Algoritma Apriori

Algoritma *Apriori* merupakan algoritma yang ditemukan oleh R. Agrawal dan R. Srikant pada tahun 1994 untuk mencari pola yang sering muncul. Nama *Apriori* sendiri dikarenakan algoritma tersebut menggunakan *prior knowledge* pada setiap properti *frequent itemset* (Han & Kamber, 2011) .Algoritma *Apriori* merupakan algoritma *association rule* yang dapat digunakan untuk memecahkan masalah dalam dunia nyata yang berhubungan dengan pengelompokan data. Misalnya kombinasi parcel yang tepat, penempatan barang dalam swalayan, pengambilan kombinasi mata kuliah, pembagian murid berdasarkan hobi, rekomendasi barang pada sistem penjualan online, dan lain sebagainya. Algoritma ini memiliki 2 buah *constraint/parameter* wajib diketahui nilainya untuk dapat menemukan *output.* Parameter tersebut adalah *support* dan *confidence.* Nilai *support* didapatkan dari membandingkan frekuensi kemunculan kombinasi item dengan seluruh kombinasi yang ada. Nilai *confidence* adalah nilai yang mendefinisikan kuat atau tidaknya korelasi antar item yang dikombinasikan. Item yang dikombinasikan dinamakan dengan *candidate generation.* Langkah yang dilakukan dalam algoritma untuk menemukan *frequent itemset* adalah penggabungan (*joint step*) dan pembuangan (*prune step*).

Langkah pertama adalah penggabungan (*the joint step*) yang terdiri dari :

1. Kandidat pertama merupakan seluruh item tunggal yang ada dalam kumpulan item.

2.Mencari *candidate generation* dengan cara mengkombinasikan item tunggal dengan item lainnya berdasarkan nilai. Tahap ini kemudian juga disebut dengan *apriori-gen.*

Langkah kedua merupakan pembuangan (*prune action*) terdiri dari :

1. Penghitungan nilai *support* dari masing-masing *candidate itemset* yang dihasilkan dari langkah sebelumnya.

$$Support = \frac{frekuensi\ kemunculan\ candidate\ itemset}{jumlah\ seluruh\ transaksi} x\ 100$$

2. *Candidate itemset* yang memiliki nilai support lebih kecil daripada *min support* akan dibuang.

Apabila semua kombinasi item yang tersisa memiliki nilai *support* yang sama dengan *min support*, maka proses penggabungan dan pembuangan dihentikan. Setelah proses tersebut selesai, kemudian nilai *confidence* dihitung. Pasangan yang memiliki nilai *confidence* 100%, maka dinyatakan memenuhi kriteria. Langkah dalam algoritma *Apriori* dituliskan ke dalam *pseudocode* sebagai berikut :

1) L_1 = {Large 1-itemsets}
2) **for** (k=2; L_{k-1}≠0; k++) **do begin**
3) C_k = apriori-gen(L_{k-1});
4) **forall** transactions l Є D **do begin**
5) C_t = subset (C_k,t)
6) **forall** candidates c Є C_t **do**
7) c.count++
8) **end**
9) L_k = {c Є C_k}c.count>-minsup}
10) **end**
11) Answer = U_k L_k;

Dikutip dari : (Agrawal & Srikant, 1994) Fast Algorithms for Mining Association Rules

Pseuocode proses *candidate generation* adalah sebagai berikut:

1) **insert into** C_k //join step
2) **select** $p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}$
3) **from** L_{k-1} p, L_{k-1} q
4) **where** $p.item_1 = q.item_1, ..., p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
5) **forall** itemsets c C_k do // prune step
6) **forall** (k-1)-siubsets s of c **do**
7) **if**(s is not element of L_{k-1}) **then**
8) **delete** c from C_k

Dikutip dari : (Agrawal & Srikant, 1994) Fast Algorithms for Mining Association Rules

2.2.7 Algoritma AprioriTid

Algoritma *AprioriTid* merupakan algoritma perbaikan dari *Apriori*. Algoritma juga menggunakan proses *apriori-gen* untuk mendapatkan *candidate itemsets* sebelum proses dimulai (Agrawal & Srikant, 1994). *AprioriTid* tidak menggunakan database lagi untuk menghitung nilai *support* setelah fase pertama, namun menggunakan itemset yang telah dibuat pada fase sebelumnya. Pseudocode dari *AprioriTid* adalah sebagai berikut :

1) L_1 ={large 1-itemsets};
2) C_1 = database D;
3) **for** (k_2; $L_{k-1} \neq 0$;k++) **do begin**
4) C_k = apriori-gen(L_{k-1});
5) A_k = 0;
6) **forall** entries t $\in$ A_{k-1} **do begin**
7) C_t = {c $\in$ C_k | c – c[k]} $\in$ t.set-of-itemsets $\wedge$ (c-c[k-1]) $\in$ t.se-of-itemsets}

8) **forall** candidates c Є Ct **do**

9) c.count++;

10) **If** ($C_t \neq 0$) **then** A_k += < t.TID,C_t >;

11) End

12) $L_k = \{c \in C_k \mid c.count \geq minsup\}$

13) End

14) Answer = $U_k L_k$;

Dikutip dari : (Agrawal & Srikant, 1994) Fast Algorithms for Mining Association Rules

BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Kebutuhan

3.1.1 Kebutuhan Fungsional

Sistem yang dibuat digunakan untuk menganalisis data transaksi penjualan tunai Amigo Group. Analisis yang dilakukan berupa pencarian *association rules* / kombinasi dari item-item yang dibeli. Untuk dapat melakukan hal tersebut sistem harus mampu melakukan fungsi-fungsi sebagai berikut

1. Sistem dapat melakukan fungsi ETL

   Proses *Extract Transformation Load* dilakukan oleh penulis untuk memindahkan data dari database transaksional ke dalam *datawarehouse*. Tahapan *cleaning data* dilakukan pada saat ETL. Proses ETL dilakukan menggunakan software *Pentaho Data Integrator*

2. Sistem dapat melakukan data *filtering.*Proses pemfilteran data digunakan untuk menyeleksi data transaksi pada periode tertentu

3. Sistem mampu mencari kombinasi item yang paling sering terjadi atau *frequent itemst generation.* Dalam proses ini, sistem dapat mencari kombinasi item yang memiliki nilai support melebihi batas minimum.

4. Sistem mampu menghasilkan aturan asosiasi (*strong rules*) yang kuat. Aturan asosiasi dihasilkan dari *frequent itemset* yang ditemukan sebelumnya. Aturan asosiasi inilah yang menjadi hasil akhir dari sistem

3.1.2 Kebutuhan Non Fungsional

Kombinasi item tidak dapat berupa sebuah barang saja. Oleh karena itu dalam pencarian *association rules*, banyaknya *item* yang ada dalam sebuah transaksi haruslah berjumlah lebih dari satu. Data transaksi penjualan yang akan digunakan dalam sistem hanyalah yang memiliki item minimal dua buah.

Sistem yang dibuat haruslah dapat berjalan di semua jenis sistem operasi, oleh karena itu penulis membangun sistem berbasiskan web. Bahasa pemrograman yang digunakan untuk pengembangan adalah *Java Server Pages* (JSP)

3.2 *Use Case*

3.2.1 Model *Use Case*

System	*Amigo Association Rules Generator*
ID	UC-01
Title	Admin *login* ke sistem
Actor	Admin
Precondition	Admin terdaftar dalam sistem
Postcondition	Admin masuk ke dalam sistem
MSS	1. Admin membuka sistem /aplikasi 2. Sistem menampilkan halaman login berisikan username dan password 3. Admin memasukkan username dan password 4. Admin menekan tombol *login* 5. Sistem melakukan verifikasi *login* 6. Admin masuk ke dalam sistem
Extensions	5a. Username atau password pengguna salah 5a.1. Sistem menampilkan halaman login, admin diminta

	memasukkan username dan password kembali 5a.2. Admin memasukkan username dan password Use case dilanjutkan ke poin ke 5

System	*Amigo Association Rules Generator*
ID	UC-02
Title	Admin melihat statistik penjualan dari periode waktu tertentu
Actor	Admin
Precondition	Admin masuk ke dalam sistem (*logged in*)
Postcondition	Sistem menampilkan informasi statistik penjualan dari periode waktu tertentu
MSS	1. Admin membuka halaman statistik 2. Sistem meminta admin memilih periode waktu 3. Admin memilih periode waktu yang disediakan sistem 4. Sistem menampilkan halaman *loading* 5. Sistem memproses informasi dari periode waktu inputan user 6. Sistem menampilkan informasi statistik dari periode waktu tertentu
Extensions	2a. Admin memasukkan periode waktu baru 2a.1. Sistem menampilkan *form* periode baru berisikan tanggal awal periode dan tanggal akhir periode 2a.2. Admin memasukkan periode waktu 2a.3. Admin menekan tombol submit 2a.4. Sistem menyimpan periode baru, ditambahakan kemudian dalam daftar periode waktu Use case dilanjutkan ke poin ke 3

System	Amigo Association Rules Generator
ID	UC-03
Title	Admin melakukan analisis asosiasi
Actor	Admin
Precondition	Admin masuk ke dalam sistem (*logged in*)
Postcondition	Sistem menampilkan asosiasi antar strip pada periode waktu tertentu
MSS	1. Admin membuka halaman asosiasi 2. Sistem meminta admin memasukkan nilai *minimum support*, *minimum confidence* serta memilih periode waktu 3. Admin memasukkan nilai *mininum support*, *minimum confidence* serta memilih periode waktu yang disediakan sistem 4. Admin menekan tombol *run* 5. Sistem menampilkan halaman *loading* 6. Sistem memproses informasi dari periode waktu inputan user 7. Sistem menampilkan asosiasi antar strip pada periode waktu sesuai inputan user
Extensions	4a. Admin tidak memasukkan nilai *minimum support* atau *minimum confidence* atau tidak memilih periode waktu 4a.1. Sistem menampilkan *modal box* yang berisikan peringatan bahwa user belum memasukkan semua *input* yang diperlukan 7a. Sistem tidak menemukan aturan yang kuat pada periode waktu inputan user dengan nilai *minimum support* dan

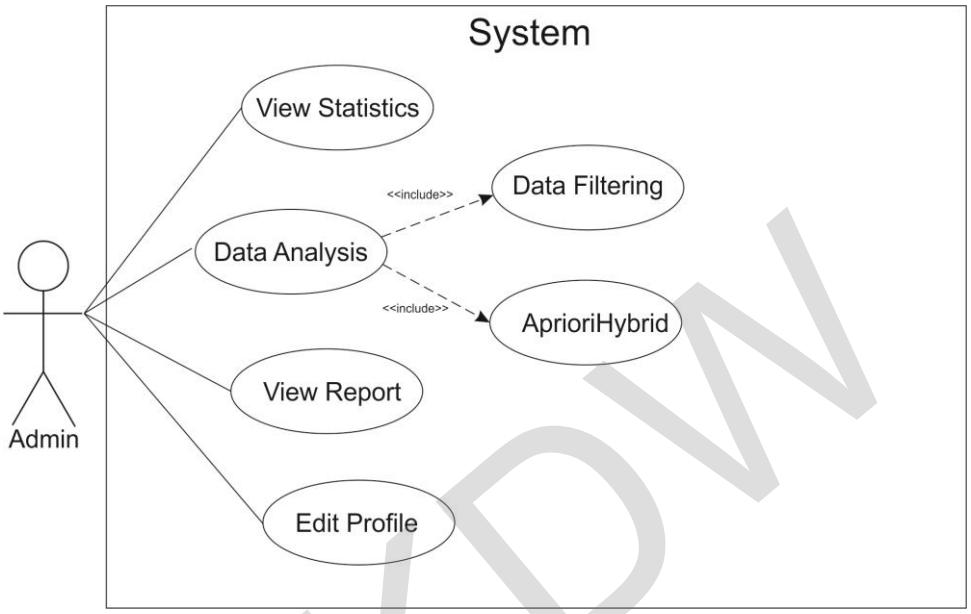
	minimum confidence yang telah ditentukan 7a.1. Sistem menampilkan pesan informasi bahwa pada periode tersebut tidak ditemukan aturan asosiasi yang kuat dengan nilai *minimum support* dan *minimum confidence* yang dimasukkan admin

System	Amigo Association Rules Generator
ID	UC-04
Title	Admin melihat hasil analisis aturan asosiasi
Actor	Admin
Precondition	Admin masuk ke dalam sistem (*logged in*)
Postcondition	Sistem menampilkan informasi statistik penjualan dari periode waktu tertentu
MSS	1. Admin membuka halaman *association result* 2. Sistem meminta admin memilih periode waktu 3. Admin memilih periode waktu yang disediakan sistem 4. Sistem menampilkan halaman *loading* 5.Sistem mengambil hasil analisis asosiasi sesuai dengan periode input user dari *database* 6. Sistem menampilkan informasi hasil analisis asosiasi
Extensions	-

System	Amigo Association Rules Generator
ID	UC-05
Title	Admin mengganti *username* atau *password*
Actor	Admin
Precondition	Admin masuk ke dalam sistem (*logged in*)
Postcondition	Sistem menampilkan informasi statistik penjualan dari periode waktu tertentu
MSS	1. Admin membuka halaman pengaturan profil 2. Sistem menampilkan *username*, *password* dan *email* admin 3. Admin melakukan perubahan *username email*, atau *password* 4. Sistem menampilkan dialog box perubahan profil 5. Admin memasukkan profil baru 6. Admin menekan tombol *save* 7. Sistem menyimpan perubahan profil ke database 8. Sistem menampilkan pesan sukses
Extensions	6a. Admin membatalkan perubahan 6a.1. Sistem tidak mengubah profil dalam database

3.2.2 Use Case Diagram

Sistem yang akan dibangun tidak menerapkan *multiuser* sehingga hanya satu pengguna saja yang memiliki akses penuh ke sistem. Pengguna tersebut selanjutnya disebut admin. Diagram *use case* dapat dilihat pada gambar 3.1.



Gambar 3. 1 Use Case Diagram

Dari diagram *use case* pada gambar 3.1 hal-hal yang dapat dilakukan oleh admin adalah sebagai berikut :

1. Admin dapat melihat statistik dari data yang telah melalui proses ETL. Statistik data dapat dipilih sesuai dengan periode tertentu
2. Admin dapat melakukan proses analisis aturan asosiasi dengan memasukkan periode penjualan, nilai *minimum support* dan *minimum confidence*. Hasil dari analisis ini adalah aturan asosiai pada periode waktu penjualan yang bersangkutan
3. Admin dapat melihat hasil analisis aturan asosiasi yang telah tersimpan dalam database.
4. Admin dapat melakukan pengubahan profil yang berupa *username*, *password* dan *email*.

3.3 Rancangan *Environment*

Sistem yang dibangun terbagi menjadi dua, yaitu client dan server. Implementasi sistem akan dilakukan di sisi *client. Server* digunakan untuk penyimpanan data. Spesifikasi perangkat keras dari server adalah sebagai berikut :

1. Processor Intel Xeon 5300 series
2. 6GB DDR2 RAM

Dari sisi client, spesifikasi hardware yang akan digunakan untuk pengujian adalah sebagai berikut :

1. Processor Intel i5 3230M
2. 8GB DDR3 RAM
3. 1TB Harddisk, dengan ruang bebas kurang lebih 200GB
4. Keyboard
5. Mouse
6. Monitor

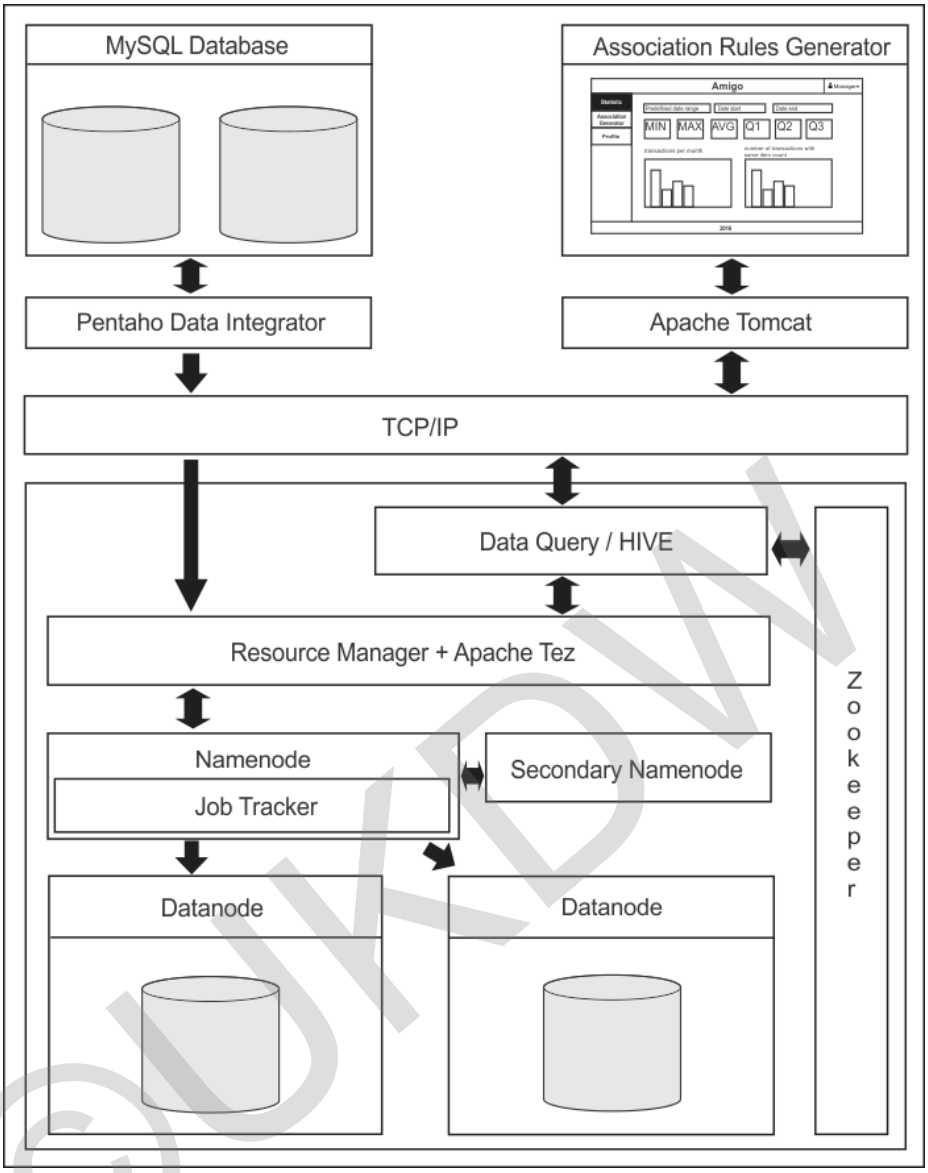
Penyimpanan data pada server menggunakan eksosistem *Hadoop*. Konsep penyimpanan data menggunakan sistem data warehouse. Database yang digunakan bersifat *analytical*. Komponen ekosistem *hadoop* yang dipasang pada *server* adalah sebagai berikut :

1. *Hadoop*
2. *Hive*
3. *Hbase*
4. *Zookeeper*

Sistem operasi yang digunakan untuk membangun sistem adalah *Linux Mint* versi 17.1 64Bit. Sedangkan sistem operasi yang digunakan untuk server merupakan *Fedora Server* 23.

3.4 Arsitektur Sistem

Server menggunakan arsitektur *data warehouse* dengan memanfaatkan ekosistem *Hadoop.* Ekosistem *Hadoop* dengan konfigursai *multicluster* membutuhkan *server* berjumlah lebih dari satu. Komponen *hadoop* yang digunakan adalah *Hadoop* (*HDFS* dan *MapReduce*), *Hive, Hbase,* dan *Zookeper*. HDFS berada pada lapisan paling bawah yang berhubungan langsung dengan penyimpanan. Semua perintah yang ditujukan kepada HDFS dikontrol oleh *MapReduce*. *Hive* digunakan untuk operasi CRUD data. *Hive* merupakan komponen dari ekosistem *Hadoop* yang digunakan untuk melakukan operasi database dalam hadoop. Perintah yang digunakan dalam Hive berupa perintah *SQL*. Dalam melakukan proses database, *Hive* membutuhkan *Zookeper*. *Hive* dan *Zookeper* berada di atas lapisan *Resource Manager* / *MapReduce*. Semua komponen ekosistem *hadoop* yang dibutuhkan akan dipasang pada semua komputer *server*.



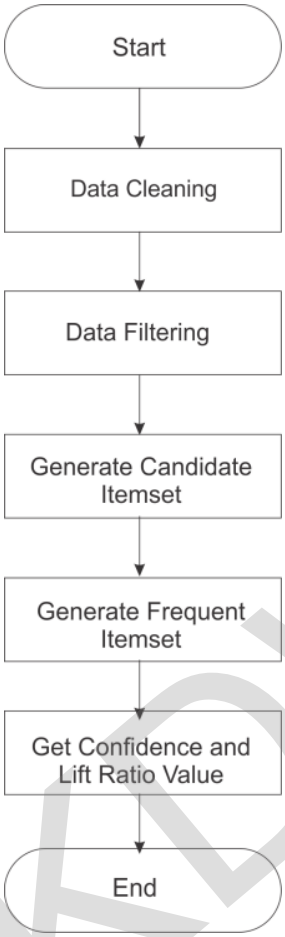
Gambar 3. 2 Rancangan Arsitektur Sistem

Sistem yang dibangun menggunakan bahasa pemrograman *Java Server Pages* (JSP). Untuk dapat menjalankan web berbasis JSP, dibutuhkan program *Apache Tomcat*. JSP dan *Tomcat* membutuhkan *Java Virtual Machine* yang berada pada paket *Java Development Kit. Apache Tomcat* dan *JDK* akan dipasang pada komputer *client*.

3.5 Rancangan Proses

Proses pencarian *association rules* secara garis besar terdiri dari *data cleaning, data filtering, frequent itemset generation,* dan *rules generation*. *Data cleaning* digunakan untuk membersihkan data yang tidak valid. Setelah itu, data yang akan dianalisis dipilih dalam *data filtering.* Data transaksi kemudian dianalisis dalam *frequent itemset generation*. Dari proses ini, aka didapatkan kombinasi item yang sifatnya *frequent,* atau paling sering muncul. Dari kombinasi item tersebut akan dicari kombinasi yang memiliki kekuataan asosiasi paling besar dalam proses *Rules Generation*.

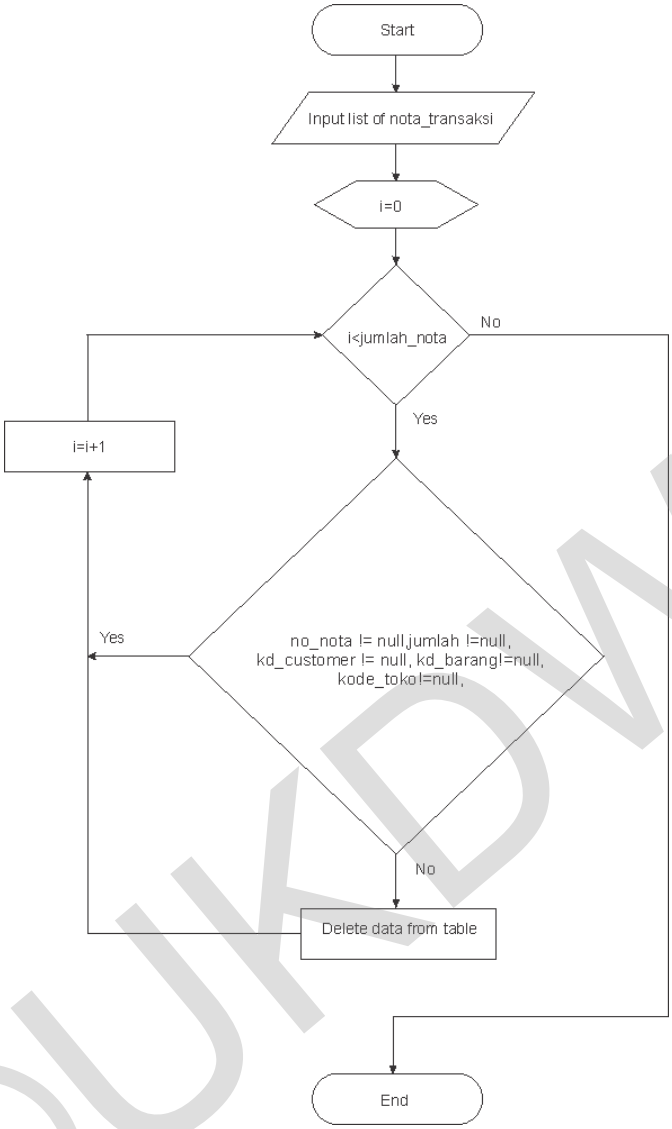
Proses ETL pada penelitian ini dilakukan pada tahapan data cleaning. Pembentukan tabel fakta yang bersumber dari beberapa tabel transaksional masuk ke dalam tahapan *extract.* Hasil dari esktraksi tersebut kemudian ditransformasikan ke dalam format yang akan digunakan untuk proses pembentukan rules. Data penjualan yang ada dalam tabel transaksional masih dalam bentuk kode barang. Dalam penelitian ini, penulis hanya menganalisis pada level strip barang. Oleh karena itu perlu pengubahan dari kode barang ke dalam bentuk strip. Tahapan ini merupakan bagian transformasi dari proses ETL. Setelah data ditransformasikan, kemudian data dimasukkan ke dalam HDFS. Tahapan ini merupakan bagian *Load* dari proses *ETL*. Secara garis besar, alur kerja sistem digambarkan pada gambar 3.3 .



Gambar 3. 3 Garis besar alur kerja sistem

3.5.1 *Flowchart Data Cleaning*

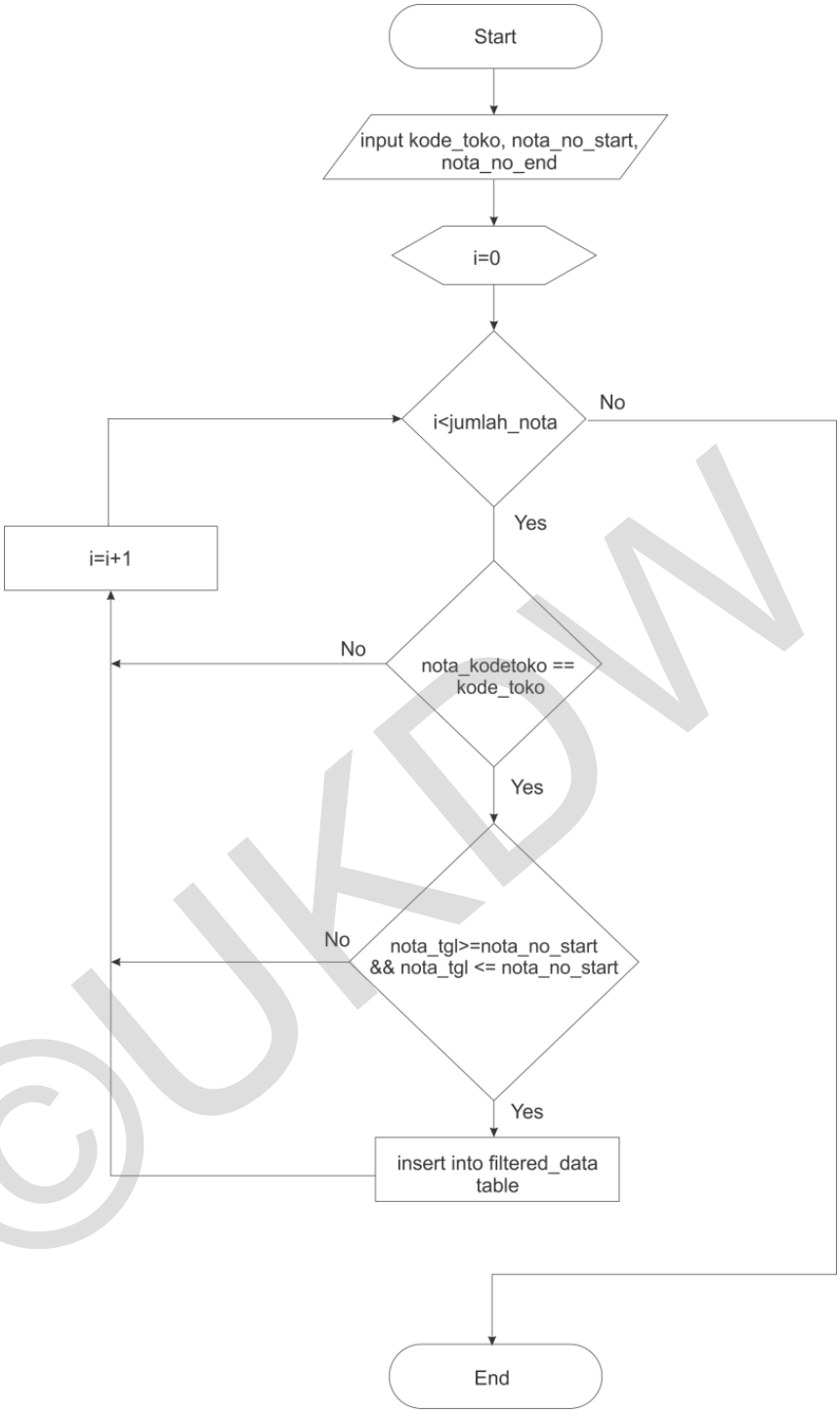
Tidak semua data transaksi yang ada dapat digunakan secara langsung. Oleh karena itu diperlukan proses *data cleaning* terlebih dahulu. Dalam proses ini, data yang tidak valid akan dibuang. Kriteria data yang tidak valid adalah apabila memiliki *field* yang bernilai *null*. Karena data yang digunakan adalah data transaksi penjualan barang, maka apabila terdapat f*ield* yang bernilai 0 atau *null*, data tersebut juga akan dibuang.



Gambar 3. 4 Flowchart Data Cleaning

3.5.2 *Flowchart Data Filtering*

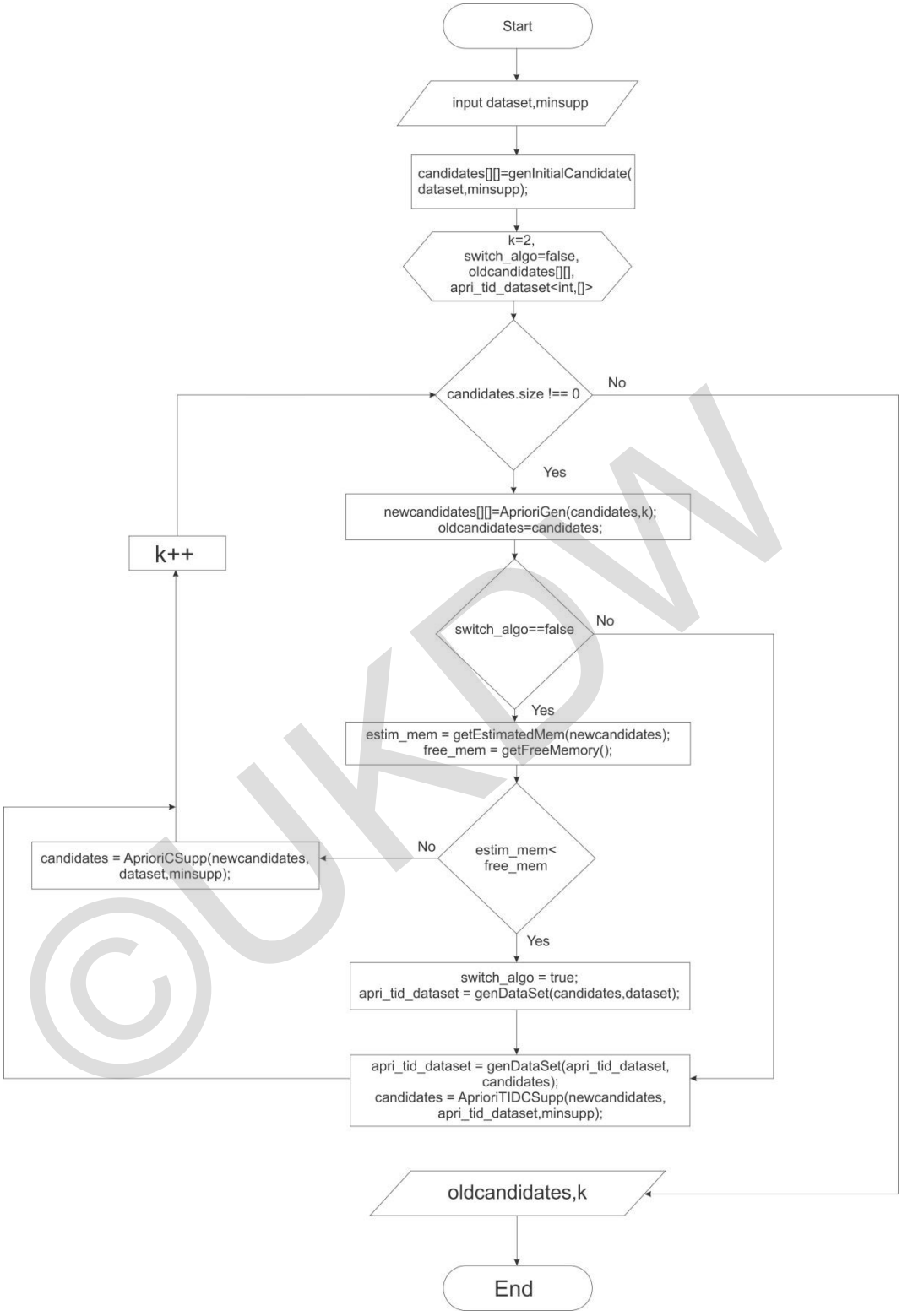
Pengguna dapat memilih data yang akan dianalisis. Proses tersebut dilakukan pada tahap data filtering. Batasan data dapat berupa *range* waktu. Pengguna menginputkan batas awal tanggal nota transaksi dan batas akhir tanggal nota transaksi tersebut, kemudian sistem akan mengambil data nota transaksi yang terjadi diantara tanggal tersebut. Alur kerja proses data filtering dapat dilihat pada gambar 3.5 berikut ini,



Gambar 3. 5 Flowchart Data Filtering

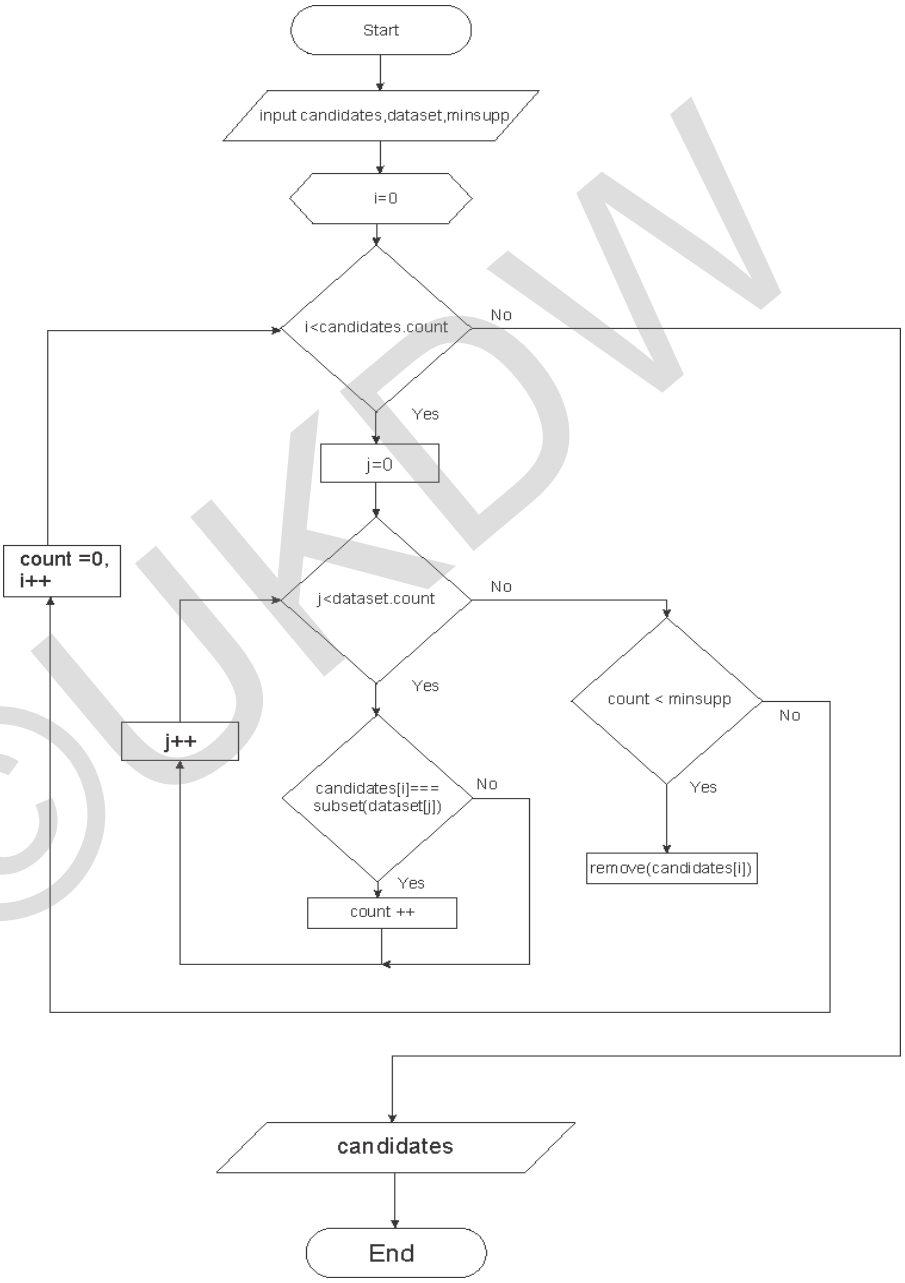
3.5.3 *Flowchart AprioriHybrid*

AprioriHybrid menggabungkan dua buah algoritma untuk mencari *frequent itemset*, yaitu *Apriori* dan *AprioriTID*. Sebelum melakukan pencarian *frequent itemset*, dilakukan pencarian element dari transasksi yang memiliki nilai *support* lebih dari minimum sebagai kandidat inisial. Kemudian dilakukan pembentukan *candidate itemset* dengan menjalankan fungsi *AprioriGen*. Pada tahap awal atau k=2, algoritma yang dipakai adalah *Apriori*. Kemudian proses akan menghitung estimasi memori yang akan terpakai apabila dataset dimasukkan dalam memori. Apabila memori mencukupi, maka algoritma yang dipakai adalah *AprioriTID*. Proses tidak lagi melakukan scanning pada *database*, melainkan menyimpan id transaksi yang menjadi support dari setiap *candidates* dalam memori. Id yang disimpan digunakan untuk menghitung nilai *support*.



Gambar 3. 6 Flowchart AprioriHybrid

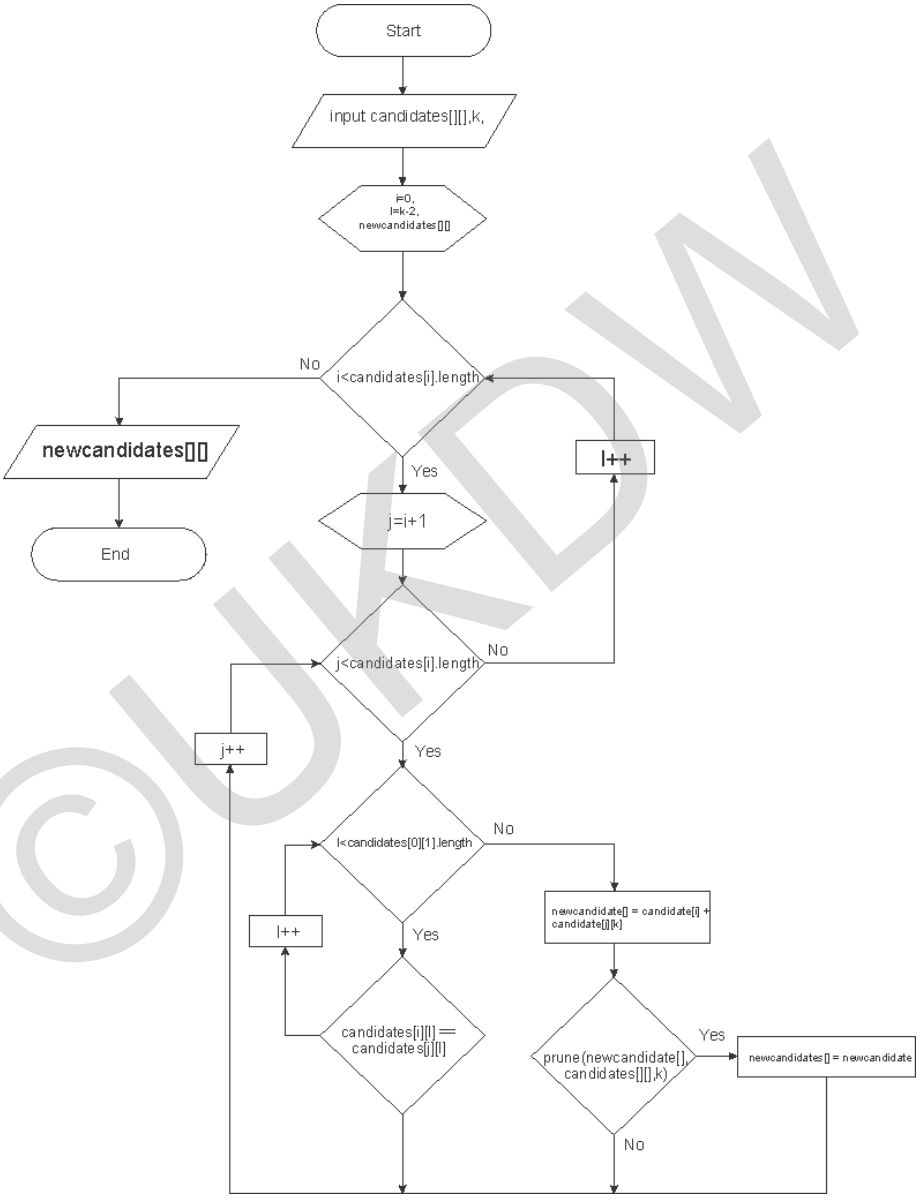
Proses penghitungan nilai support pada fungsi AprioriCSupp dan AprioriTIDCSupp adalah dengan cara menghitung banyaknya transaksi yang memiliki subset yang sama dengan *candidate itemset*. Apabila jumlah transaksi lebih dari mininum support, maka candidate itemset tersebut akan menjadi *frequent itemset.* Apabila tidak *candidate itemset* tersebut tidak akan digunakan atau dibuang.



Gambar 3. 7 Flowchart AprioriCountSupport

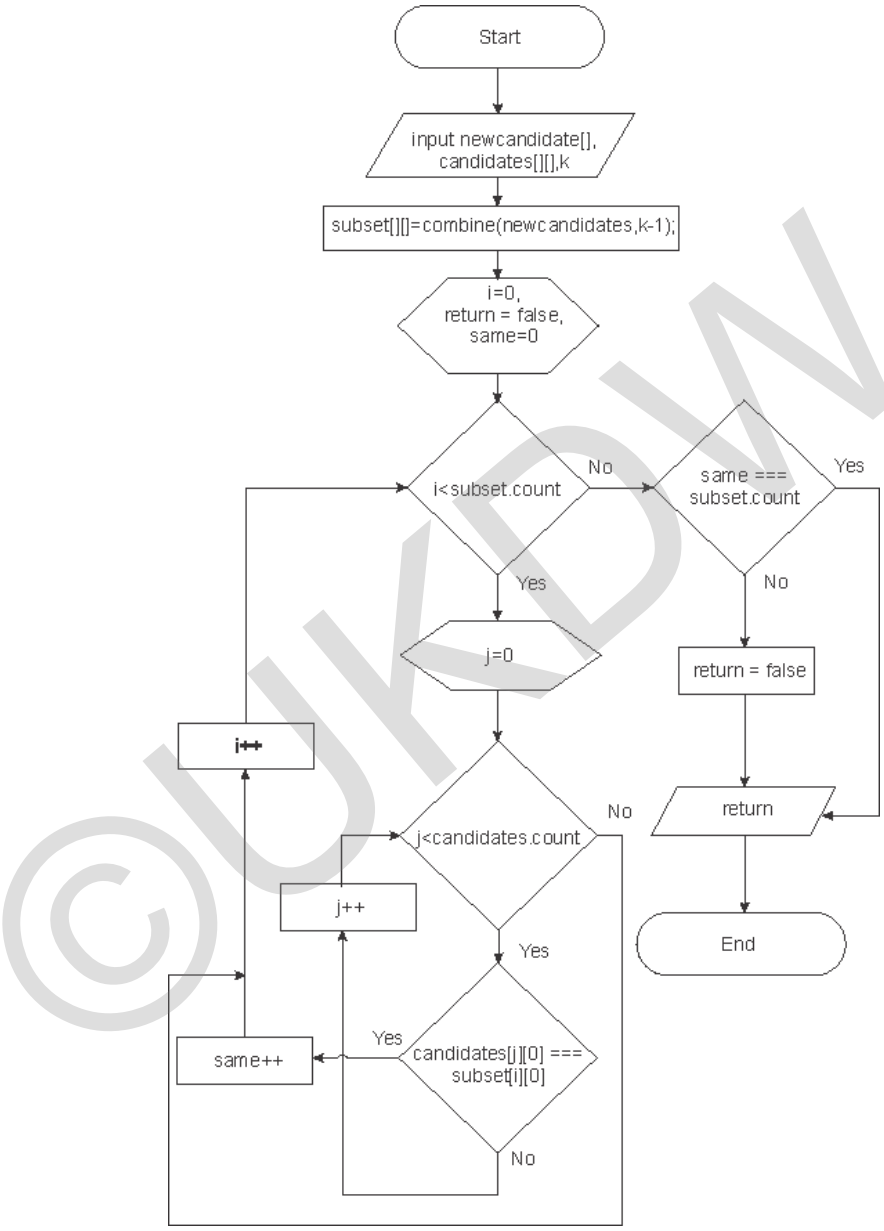
3.5.4 *Flowchart AprioriGen*

Proses pembentukan kandidat dilakukan dengan cara menggabungkan semua kandidat (*candidates*) dengan dirinya sendiri (*self join*). Proses penggabungan pada tahap *k* memiliki aturan di mana. Nilai pertama sampai nilai k-1 dari dua buah kandidat yang akan digabungkan haruslah sama.



Gambar 3. 8Flowchart AprioriGen

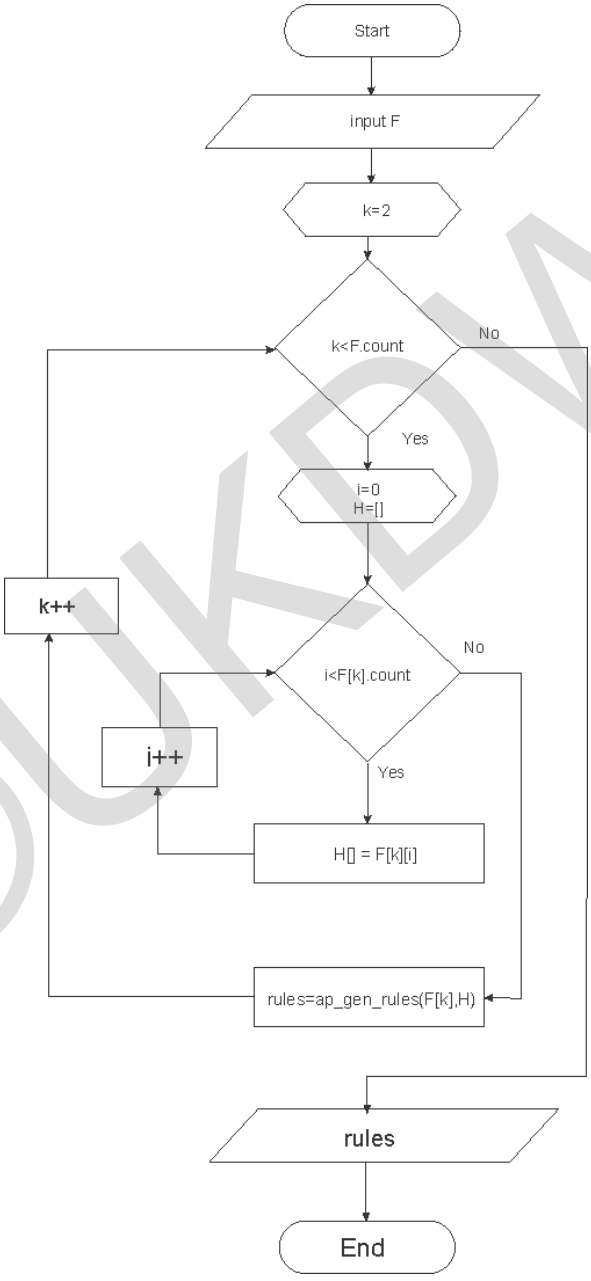
Setelah dilakukan penggabungan, dilakukan tahap *pruning*. Dimana kandidat baru dicari subset k-1. Apabila setiap subset k-1 dari kandidat baru tidak ada dalam daftar kandidat lama. Maka kandidat baru tersebut tidak akan digunakan atau dibuang.



Gambar 3. 9Flowchart AprioriGen Pruning

3.5.5 *Flowchart Rules Generation*

Setelah *frequent itemset* didapatkan, untuk mendapatkan *rules* yang valid dilakukan proses *Rules Generation*. Dalam proses ini setiap *frequent itemset* akan dicari subset mulai dari hubungan 1 *item* dengan k-1 *item* sampai dengan k-1 *item* dengan 1 *item*.



Gambar 3. 10 Flowchart Rules Generation

3.6 Rancangan *Database*

Pemodelan basis data dilakukan dengan menganut konsep *Datawarehouse*. Terdapat sebuah tabel fakta yang terhubung dengan dua atau lebih tabel dimensi. Skema yang digunakan adalah bintang / *star schema.* Analisis yang dilakukan hanya dilakukan pada transaksi penjualan tunai saja, oleh karena itu dalam skema yang digunakan hanya akan terdapat satu tabel fakta.

Analisis *association rules* membutuhkan data yang rinci. Oleh karena itu tabel fakta yang dibut tidak akan melakukan agregrasi jumlah barang dalam setiap transaksi karena akan menghilangkan detail. Setiap barang dalam sebuah transaksi akan menjadi sebuah *row* dalam tabel fakta. Tabel fakta berisi :

1. No nota transaksi
2. Id tanggal
3. Id lokasi
4. Id Strip
5. Id Kedalaman produk
6. Id Merk
7. Jumlah
8. Harga
9. Total harga

Dimensi yang akan digunakan dalam pemodelan data adalah dimensi lokasi, dimensi waktu, dimensi strip, dimensi kedalaman produk, dan dimensi merk. Dimensi lokasi berisikan :

1. Id lokasi
2. Nama lokasi

Strip merupakan kelompok barang secara umum. Misalnya celana pendek dewasa, atau baju berkerah dewasa. Isi dari tabel strip adalah :

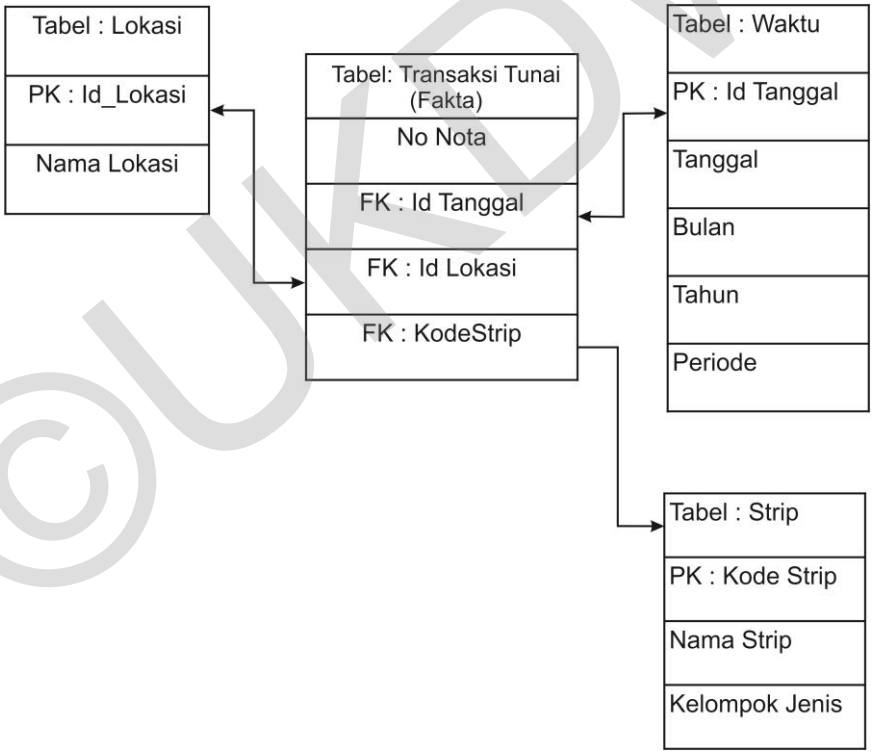
1. Kode strip
2. Nama strip

3. Kelompok jenis

Dimensi waktu digunakan untuk memberikan keterangan kapan terjadinya transaksi. Dalam tabel dimensi waktu berisi *field* sebagai berikut :

1. Id tanggal
2. Tanggal
3. Bulan
4. Tahun
5. Periode

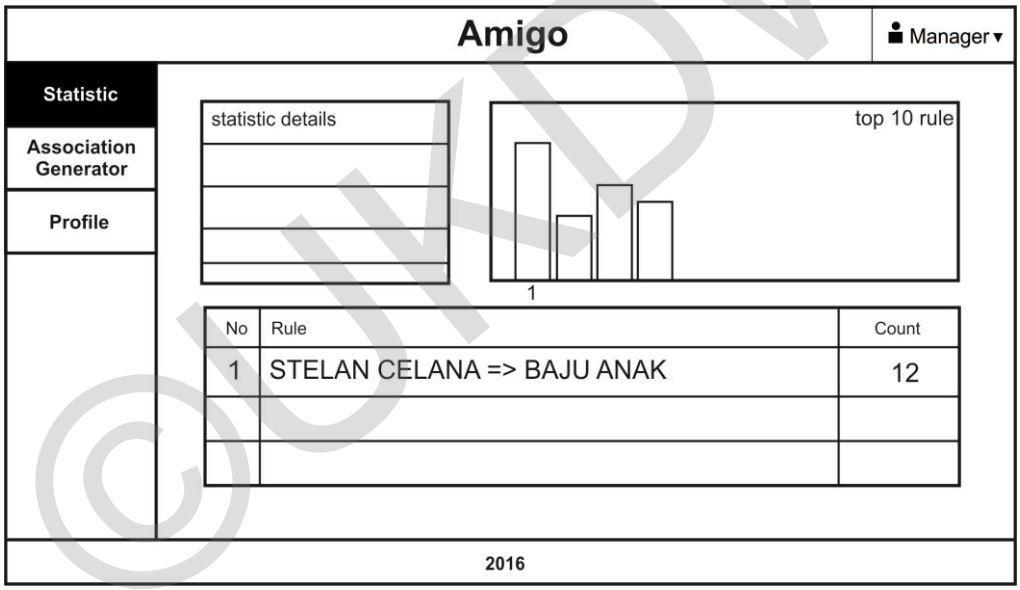
Rancangan skema yang digunakan digambarkan sebagai berikut :



Gambar 3. 11 Rancangan pemodelan basis data dengan skema star

3.7 Rancangan Antar Muka

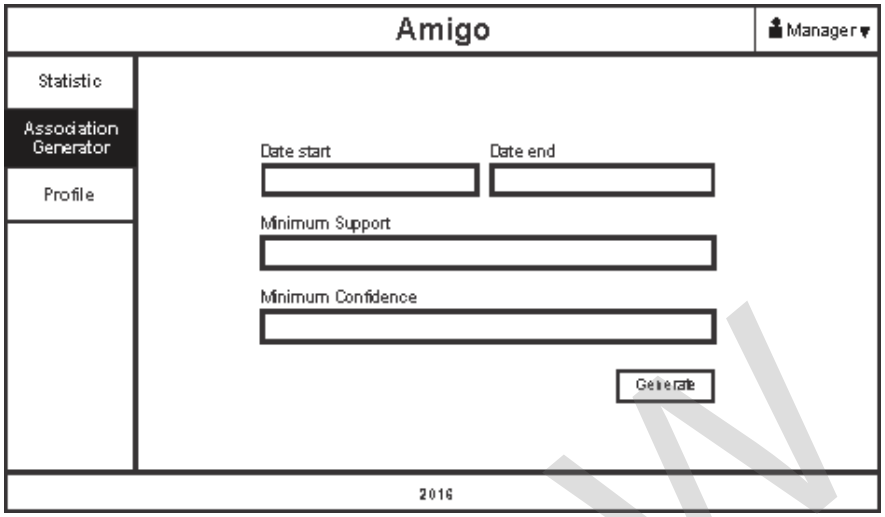
Antarmuka sistem yang dibangun untuk analisis *association rules* akan berbentuk *dashboard.* Terdapat sub-sub menu yang digunakan untuk melihat atau mengontrol bagian-bagian khusus. Komponen utama sistem akan dibagi menjadi 3 bagian. Yang pertama merupakan *panel* statistik data. Panel ini berupa gambaran statistik data pada rentang waktu tertentu. Statistik yang ditampilkan meliputi nilai rata-rata, nilai minimum, nilai maksimum, nilai kuartil satu, nilai kuartil dua dan nilai kuartil tiga dari banyaknya item dalam sebuah transaksi. Selain itu pada panel ini juga menampilkan data banyaknya transaksi per bulan dalam periode tertentu. Diharapakan dari statistik yang ditampilkan user dapat menetapkan nilai minimum support dan confidence berdasarkan statistik data yang akan dianalisis.



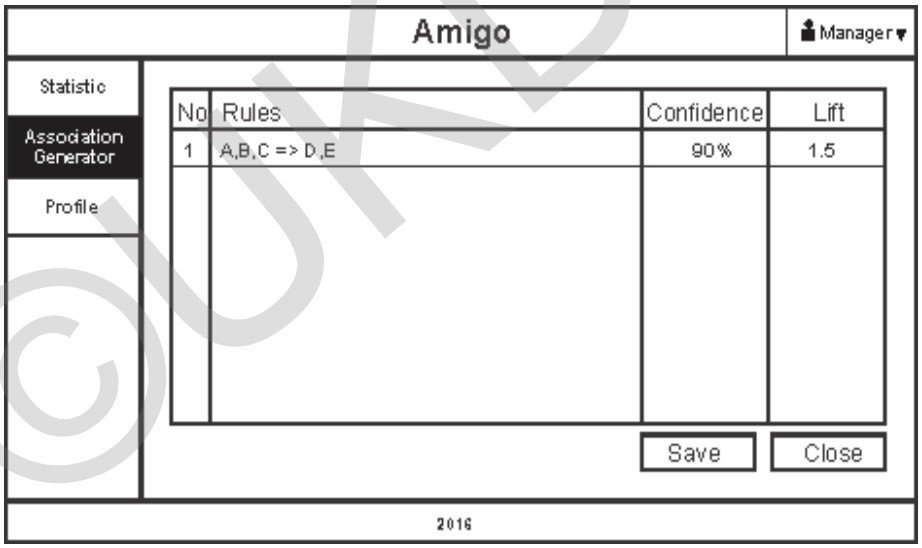
Gambar 3. 12 Rancangan antarmuka panel statistik

Bagian kedua merupakan panel yang digunakan untuk mencari kombinasi item / *association rules*. Pada bagian ini, pengguna diminta untuk memasukan rentang waktu, nilai *minimum support*, dan *minimum confidence*. Rentang waktu digunakan untuk membatasi data yang akan dianalisis. Hanya data transaksi yang masuk dalam rentang waktu sesuai inputan pengguna yang akan dianalisis. Setelah proses analisis sistem akan menampilkan tabel berupa kombinasi item,

nilai support, nilai confidence dan nilai lift. Kombinasi yang ditampilkan merupakan *strong rules* yang dihasilkan.

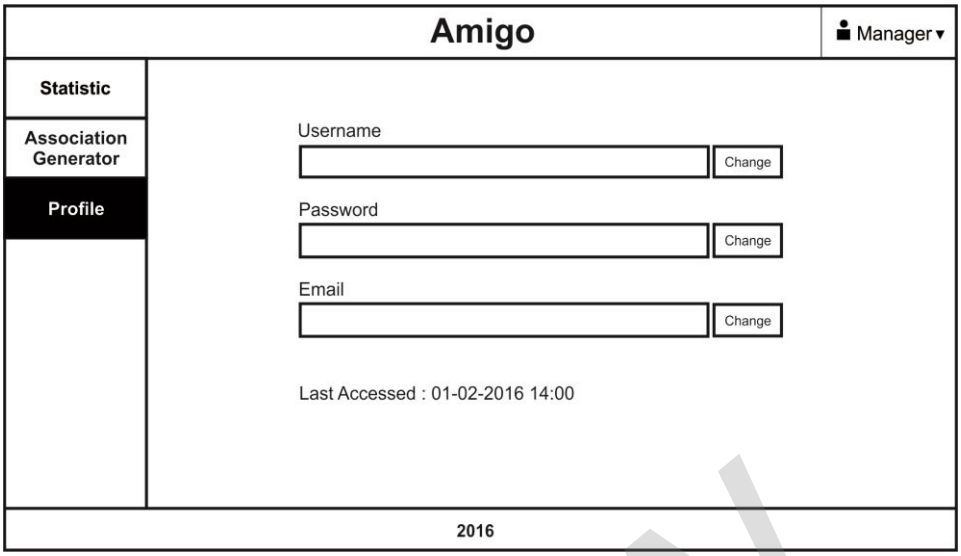


Gambar 3. 13 Rancangan antarmuka panel association generator



Gambar 3. 14 Rancangan antarmuka panel association report

Bagian ketiga merupakan pengaturan profil pengguna. Dari panel ini, pengguna dapat mengganti data pribadi yang meliputi *username*, *password* dan *email*.



Gambar 3. 15 Rancangan antarmuka panel pengaturan profil

3.8 Rancangan Pengujian

Pengujian sistem dilakukan dengan membandingkan aturan yang ditemukan pada periode yang berbeda dengan nilai *minimum support* dan *minimum confidence* yang sama. Misalnya aturan asosiasi kuartal pertama tahun 2011 dibandngkan dengan kuartal pertama 2012 dengan nilai minimum support 5% dan minimum confidence 50%.

Dari hasil aturan asosiasi yang ditemukan kemudian dicari aturan yang sama. Apabila terdapat aturan asosiasi yang sama maka aturan yang sama berlaku pada kedua periode tersebut. *Trend* pembelian antara periode dapat dikatakan memiliki kesamaan.

Apabila diantara dua periode atau lebih tidak ditemukan asosiasi yang sama, atau memiliki kesamaan asosiasi namun tidak memiliki nilai *confidence* dan *lift* yang tidak kuat. Dapat dikatakan bahwa dari kedua periode tersebut tidak memiliki trend pembelian yang sama. Profil data / statistik dari kedua periode tersebut kemudian dibandingkan.

No Transaksi	Strip Barang
01-001	AA, EA,AB,ED,BB
01-002	LL,EL,AS,AB,AC,AD
01-003	AA, DD,CC,BB,ED,BC
01-004	AA, ED, AB,BB
01-006	ED,AA,BB
01-007	AA,EA,BB
01-008	AA,ED,EA,BB,DD
01-009	DD,EB
01-010	AA,EA,BB,EB

Tabel 3. 1 Contoh transaksi penjualan kuartal pertama 2011

No Transaksi	Strip Barang
02-001	EA,AA,BB,ED
02-002	EA,ED,AS,AA,AB,AC,BB
02-003	AA, DD,BC,BB,ED
02-004	AA, EC, AB,BC
02-006	ED,AA,BB,BC
02-007	AA,EA,BB ,BC
02-008	AB,ED,EA,BB,DD
02-009	BB,BC,ED,EA
02-010	AA,EA

Tabel 3. 2 Contoh transaksi penjualan kuartal pertama 2012

Kedua data penjualan pada periode yang berbeda tersebut akan dicari aturan asosiasi menggunakan parameter *minimum confidence* dan *minimum support* sebagai berikut :

Jenis	Nilai
Minimum Support	20% atau 20% x jumlah transaksi = 2
Mininum Confidence	70%

Setelah dianalisis dengang menggunakan algoritma AprioriHybrid ditemukan aturan yang memenuhi kriteria sebagai berikut :

No	Rules	Support	Confidence	Lift
4	ED -> AA,BB,EA	20%	100%	2.5
10	EA,ED -> AA,BB	20%	100%	1.6
11	BB,EA,ED -> AA	20%	100%	1.6
12	AA,EA,ED -> BB	20%	100%	3.3

Tabel 3. 3 Penghitungan nilai minimum confidence dan lift transaksi penjualan kuartal pertama 2011

No	Rules	Support	Confidence	Lift
4	ED -> AA,BB,EA	20%	100%	1.6
10	EA,ED -> AA,BB	20%	100%	2.5
11	BB,EA,ED -> AA	20%	100%	1.6
12	AA,EA,ED -> BB	20%	100%	3.3

Tabel 3.4 Penghitungan nilai minimum confidence dan lift transaksi penjualan kuartal pertama 2012

Dari tabel 3.3 dan 3.4 diketahui bahwa aturan yang dimiliki kedua periode mempunyai kesamaan 100%. Aturan asosiasi yang sama adalah :

1. ED -> AA,BB,EA
2. EA,ED -> AA,BB
3. BB,EA,ED -> AA
4. AA,EA,ED -> BB

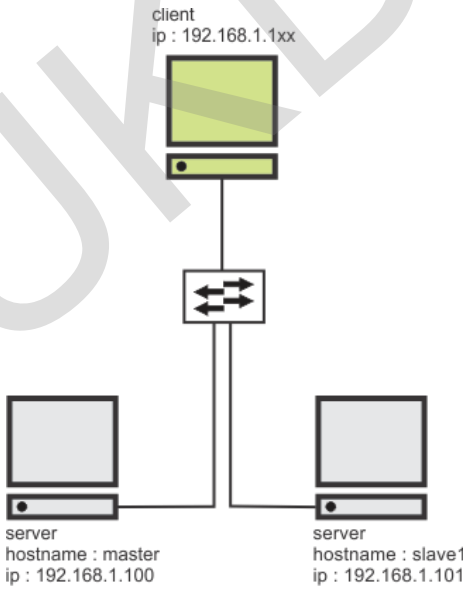
Oleh karena itu dapat dikatakan bahwa trend asosiasi pada periode penjualan kuartal pertama 2011 dan 2012 memiliki trend yang sama.

BAB IV
IMPLEMENTASI DAN ANALISIS

4.1 Implementasi Sistem

Bagian implementasi sistem berisi mengenai penerapan rancangan sistem yang telah dibuat pada bab sebelumnya. Implementasi sistem terdiri dari konfigurasi *server*, implementasi ETL, implementasi *data cleaning*, implementasi *data filtering*, implementasi *apriorihybrid*, i*mplementasi apriorigen*, implementasi *apgenrules*, serta implementasi antarmuka.

4.1.1 Konfigurasi Server

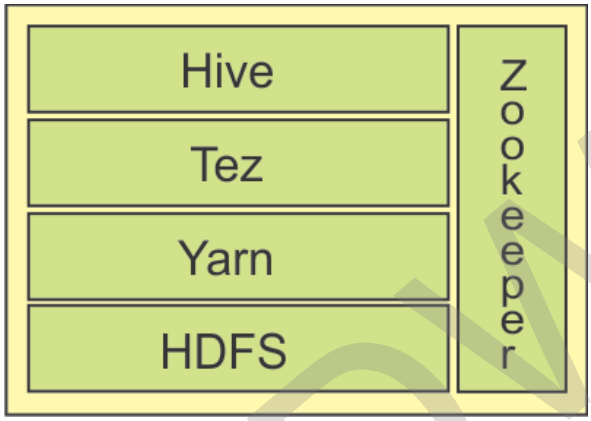


Gambar 4. 1 Topologi arsitektur client-server

Server terdiri dari dua buah komputer yang masing-masing terhubung dengan sebuah switch. Salah satu server berperan sebagai induk yang mengatur semua proses permintaan data dari client. Komputer server dan client saling

terhubung melalui sebuah switch. Topologi yang digunakan adalah topologi star. Topologi dari arsitektur client server dapat dilihat pada gambar 4.1

Setiap komputer server diinstall ekosistem hadoop yang terdiri dari HDFS, Yarn ResourceManager, Tez, Zookeeper serta Hive. Struktur dari ekosistem Hadoop dapat dilihat pada gambar 4.2 sebagai berikut :



Gambar 4. 2 Struktur Hadoop Ecosystem

Konfigurasi server terdiri dari konfigurasi *hostname*, konfigurasi IP, konfigurasi *Hadoop*, konfigurasi *Hive*, konfigurasi *Tez*, serta konfigurasi *Zookeeper*.

4.1.1.1 Konfigurasi *hostname*

Hostname merupakan nama dari sebuah komputer. Pemberian nama pada masing-masing komputer disesuaikan dengan perananan masing-masing komputer. Komputer yang digunakan sebagai induk akan dinamakan *master*, sedangkan komputer lain dinamakan *slave*X. X merupakan urutan dari komputer. Misalnya komputer non induk pertama maka *hostname* nya adalah *slave1*. Pengubahan nama *hostname* dilakukan dengan mengubah *file /etc/hostname*. Pada *file* tersebut diisikan nama komputer. Penulis menggunakan hostname *master* pada komputer induk dan *slave1* pada komputer non-induk.

4.1.1.2 Konfigurasi *IP Address*

Berdasarkan skema yang dibuat, Ip blok yang digunakan untuk server adalah 192.168.1.99/24. Ip pertama (192.168.1.99) digunakan untuk default

gateway. Ip selanjutnya direservasi khusus untuk server. 192.168.1.100 digunakan untuk master dan 192.168.1.101 untuk slave1. Pengaturan reservasi IP dilakukan pada router, dengan cara menambahkan *pair* atau pasangan berupa alamat *mac* port ethernet masing-masing komputer serta alamat ip yang digunakan. Pengaturan yang dilakukan penulis dapat dilihat pada gambar berikut :

4.1.1.3 Konfigurasi *Hadoop*

Pada bagian menjelaskan tentang konfigurasi Hadoop yang akan digunakan sebagai tempat penyimpanan data. Dalam penelitian ini, penulis menggunakan *Hadoop 2.7.2* dengan konfigurasi *multi cluster*. Sebelum mengkonfigurasi *Hadoop,* terdapat beberapa persiapan yang harus dilakukan agar *Hadoop* dapat berfungsi, yaitu:

1. *Java Development Kit* (JDK)

   Hadoop membutuhkan *java* dengan versi 5 keatas . Dengan demikian, sebelum mengkonfigurasi *Hadoop*, *java* perlu dikonfigurasi terlebih dahulu. Versi j*ava* yang sudah *terinstall* pada sistem operasi *linux* dapat diketahui dengan perintah: `java -version`

2. *Secure Socket Shell* (SSH)

   Hadoop membutuhkan *SSH* untuk mengakses *nodes* yang dimiliki. Pada konfigurasi *multi cluster*, *SSH* dibutuhkan untuk melakukan *remote* terhadap *node* yang berperan sebagai *datanode.* Ketika *Hadoop* dijalankan dari komputer yang berperan sebagai *namenode* (komputer induk), secara otomatis proses *Hadoop* pada komputer *datanode* akan juga berjalan. Hal ini dikarenakan *namenode* melakukan perintah secara *remote*. Untuk dapat melakukan perintah tersebut komputer induk perlu melakukan *login* terlebih dahulu ke setiap komputer *datanode*. Agar tidak memerlukan input *password* diperlukanlah sebuah kunci yang dikenali bersama oleh setiap komputer yaitu kunci RSA. RSA *key* dapat dibuat dengan perintah: `ssh-keygen -t rsa -P ""`. Perintah tersebut akan menghasilkan file berupa id rsa yang berada di direktori

~/.ssh/id_rsa.pub selanjutnya isi dari file tersebut akan ditambahkan ke dalam file *~/.ssh/authorized_keys* ke setiap komputer *datanode*. Untuk melakukan verifikasi bahwa komputer induk dapat melakukan *ssh*, dilakukan pengujian dengan cara mengetikan perintah *ssh username@ip_komputer_datanode* di terminal, misalnya *ssh root@192.168.1.100*. Apabila benar maka komputer namenode dapat membuka terminal/shell sebagai *root* dari *datanode* tanpa perlu memasukkan password.

Setelah JDK dan SSH sudah dikonfigurasi dengan benar. Paket *Hadoop* diekstraksi pada direktori */usr/local/hadoop*. Selanjutnya, *Hadoop Environment Variables* perlu dimasukkan ke dalam file `bashrc` yang terdapat pada direktori `~`. Mengedit bashrc dapat dilakukan dengan perintah `nano ~/.bashrc` kemudian tambahkan beberapa baris berikut:

```
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_INSTALL=$HADOOP_HOME
```

Perubahan tersebut dapat di-*apply* dengan perintah `source ~/.bashrc`.

Selanjutnya, agar *Hadoop* dapat berjalan di atas java, maka *path* untuk direktori java pada Hadoop perlu dikonfigurasi dengan cara menambahkan `JAVA_HOME` pada file `hadoop-env.sh` yang terdapat pada direktori `$HADOOP_HOME/etc/hadoop`. Kemudian, terdapat 4 buah *file* pada direktori `$HADOOP_HOME/conf` yang diubah untuk konfigurasi *Hadoop*, yaitu:

1. core-site.xml

   core-site.xml akan berisi informasi tentang port yang digunakan *instance* Hadoop, jumlah memory yang dialokasikan untuk sistem, batas *memory* yang diberikan, dan juga besarnya *buffer*. Berikut konfigurasi core-site.xml:

```
<configuration>
<property>
    <name>hadoop.tmp.dir</name>
    <value>/tmp/hadoop</value>
</property>

<property>
    <name>fs.default.name</name>
    <value>hdfs://master:9000</value>
</property>
</configuration>
```

2. hdfs-site.xml

   hdfs-site.xml berisi informasi tentang jumlah replikasi data yang akan dibuat, direktori data *namenode* serta *datanode*. Pada konfigurasi *multi cluster*, nilai replikasi data adalah 1. Berikut konfigurasi hdfs-site.xml:

```
<configuration>
<property>
    <name>dfs.name.dir</name>
    <value>file:///usr/local/hadoop/hdfs/name</value>
</property>
<property>
    <name>dfs.data.dir</name>
    <value>file:///usr/local/hadoop/hdfs/data</value>
</property>
<property>
    <name>dfs.replication</name>
    <value>1</value>
</property>
</configuration>
```

3. yarn-site.xml

```
<configuration>
<property>
    <name>yarn.nodemanager.aux-service</name>
    <value>mapreduce_shuffle</value>
</property>
```

```
</configuration>
```

4. mapred-site.xml

mapred-site.xml berisi informasi mengenai *framework Mapreduce* yang akan digunakan. Berikut konfigurasi mapred-site.xml:

```
<configuration>
<property>
    <name>mapred.job.tracker</name>
    <value>master:9001</value>
</property>
<property>
    <name>yarn.resourcemanager.hostname</name>
     <value>master</value>
</property>
</configuration>
```

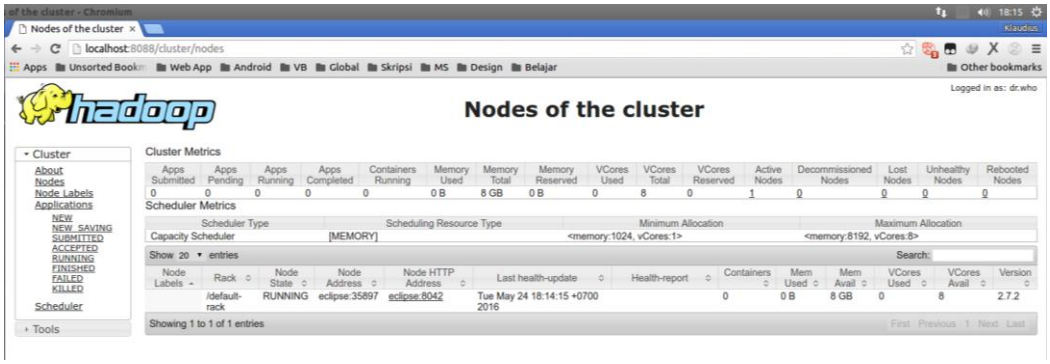
Sesuai dengan konfigurasi pada core-site.xml, direktori `/app/tmp/hadoop` harus dibuat terlebih dahulu sebelum melakukan format namenode.

```
su mkdir -p /tmp/hadoop
su chown hduser:hadoop /tmp/hadoop
su chmod 750 /tmp/hadoop
```

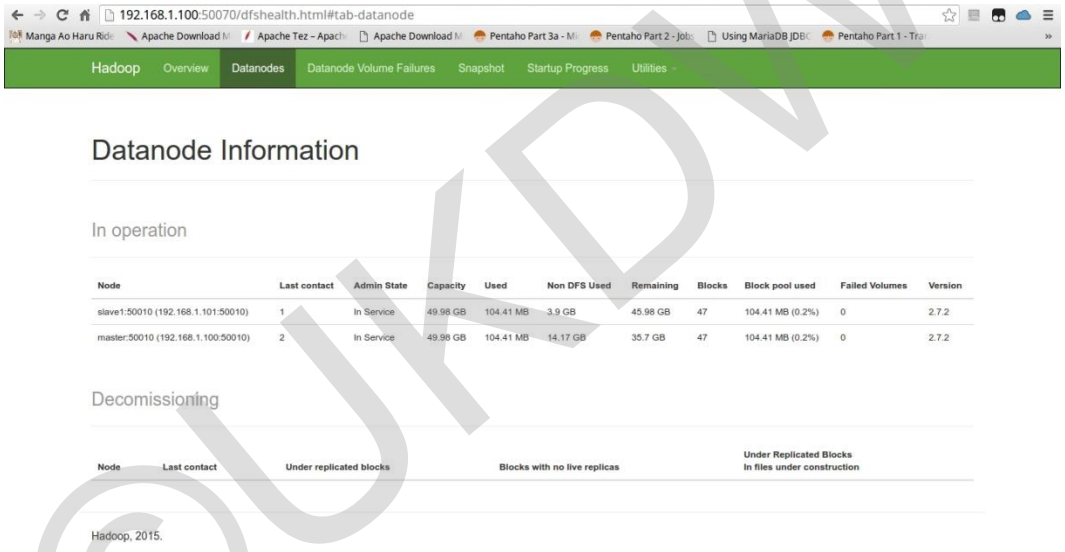
Sebelum menggunakan *Hadoop*, *Hadoop File System* (HDFS) perlu diformat terlebih dahulu dengan perintah:

```
$HADOOP_HOME/bin/hdfs dfs namenode -format
$HADOOP_HOME/bin/hdfs dfs datanode -format
```

Setelah melakukan format, maka *Hadoop* dapat dijalankan dengan perintah `$HADOOP_HOME/sbin/start-all.sh` dan untuk menghentikan dengan perintah `$HADOOP_HOME/sbin/stop-all.sh`. Informasi *Hadoop cluster* yang sudah dikonfigurasi dapat dilihat dengan mengakses port 50070 melalu *browser*. Tampilan *Hadoop* yang telah dikonfigurasi dapat dilihat pada Gambar.4.3 dan Gambar.4.4.



Gambar 4. 3 Daftar aplikasi yang berjalan pada cluster



Gambar 4. 4 Tampilan Dashboard Hadoop

Gambar 4.3 menunjukkan seluruh aplikasi yang berjalan pada *cluster* yaitu dengan mengakses `master:8088` sementara gambar 4.4 menunjukkan informasi tentang *Hadoop* yang telah dikonfigurasi termasuk *datanode* yang aktif dan juga *filesystem* yang berisi data-data misalnya tabel *hive*.

4.1.1.4 Konfigurasi *Hive*

Hive merupakan salah satu komponen infrastruktur data *warehouse* yang mempunyai fasilitas *query* dan pengaturan *dataset* di dalam HDFS. Hive akan berjalan di atas *MapReduce*/*Tez* dan digunakan untuk melakukan *query* terhadap

data yang sudah tersimpan di *Hadoop*. Pada dasarnya Hive bersifat *read-based* sehingga tidak mendukung operasi transaksional seperti *update*, dan *delete*.

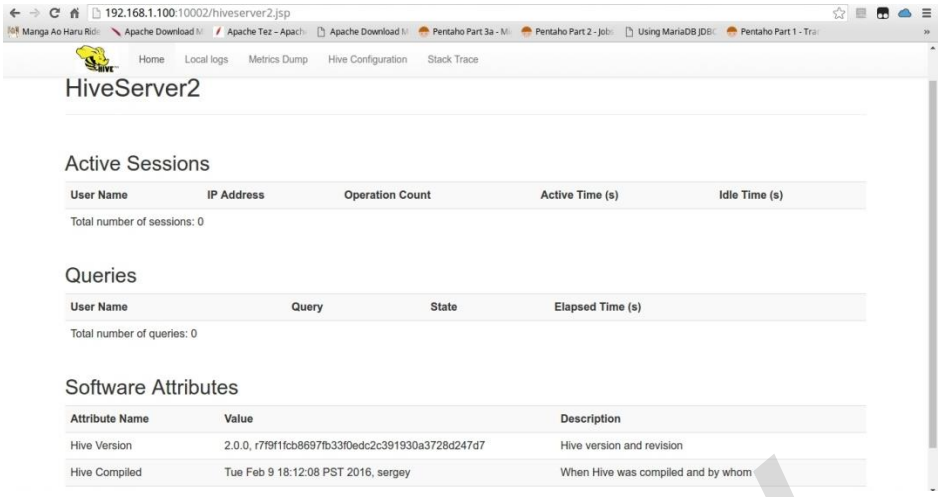
Untuk menjalankan *Hive*, Paket hive diekstrak ke dalam direktori `/usr/local/hive`. Kemudian menambahkan *hive environment variables* ke `~/.bashrc` seperti pada konfigurasi Hadoop. Berikut *hive environment variables* yang ditambahkan:

```
export HIVE_HOME=/usr/local/hive
export PATH=$PATH:$HIVE_HOME/bin
export CLASSPATH=$CLASSPATH:/usr/local/Hadoop/lib/*:.
export CLASSPATH=$CLASSPATH:/usr/local/hive/lib/*:.
```

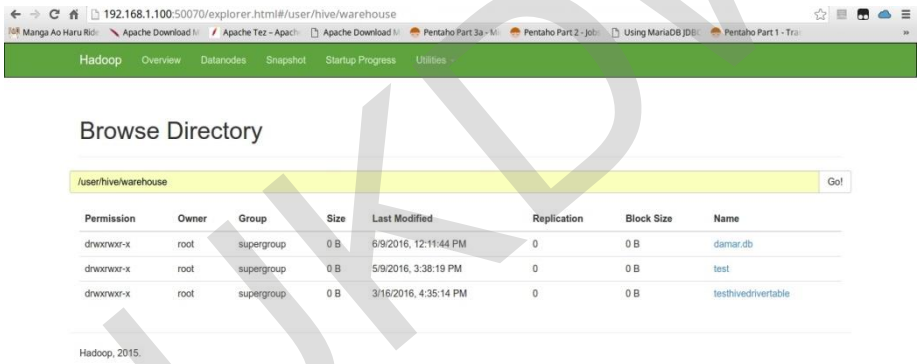
Agar *Hive* terintegrasi dan berjalan di atas *Hadoop*, maka lokasi installasi hadoop harus dikenali *Hive* yaitu dengan menambahkan `export HADOOP_HOME=/usr/local/hadoop` ke dalam `hive-env.sh` yang terdapat pada direktori `$HIVE_HOME/conf`. Selain itu, sebelum menjalankan *Hive*, direktori *Hive* perlu dibuat di dalam HDFS dengan perintah:

```
$HADOOP_HOME/bin/hdfs dfs -mkdir -p /tmp
$HADOOP_HOME/bin/hdfs dfs -mkdir -p /user/hive/warehouse
$HADOOP_HOME/bin/hdfs dfs -chmod g+w /tmp
$HADOOP_HOME/bin/hdfs dfs -chmod g+w /user/hive/warehouse
```

Hive memiliki database bawaan dengan nama *default* yang menempati /user/hive/warehouse. Semua tabel dan database memiliki *metadata (hivemetastore)* yang disimpan dalam *database metastore*. Dalam implementasi yang dilakukan, jenis *database* yang digunakan adalah *derby*. Untuk itu maka *hivemetastore* perlu diformat terlebih dahulu. *Hivemetastore* dapat diformat dengan cara masuk ke direktori `$HIVE_HOME/bin` lalu jalankan perintah `./schematool -dbType derby -initSchema`. Selanjutnya, *hive* dapat dijalankan dengan perintah `$HIVE_HOME/bin/hiveserver2`. *Hive server* yang sudah dikonfigurasi dapat dilihat pada gambar 4.3, sementara *database hive* yang disimpan pada *HDFS* dapat dilihat pada gambar 4.4.



Gambar 4. 5 Hiveserver2



Gambar 4. 6 Hive dalam HDFS

4.1.1.5 Konfigurasi *Zookeeper*

Zookeeper berfungsi untuk mengatur *task*, distribusi, dan pemrosesan data khususnya pada system yang bersifat *distributed mode*. Peran *zookeeper* sebenarnya tidak terlalu dirasakan oleh *end-user* tetapi sangat penting untuk optimalisasi kinerja komponen penyusun *datawarehouse*. Konfigurasi *zookeeper* cukup sederhana yaitu hanya dengan menambahkan *zookeeper environment variables* ke `.bashrc` dan mengatur file `zoo.cfg` yang merupakan file

konfigurasi *zookeeper*. *Environment variables* yang dimasukkan ke `~/.bashrc` adalah `export ZK_HOME=/usr/local/zookeeper`. Sementara itu, pada file konfigurasi `zoo.cfg` yang terdapat pada direktori `$ZK_HOME/conf` ditambahkan konfigurasi direktori tempat *zookeeper* menyimpan data, misalnya `dataDir=/usr/local/zoo_data`. Perintah `$ZK_HOME/bin/zkServer.sh start` digunakan untuk menjalankan *zookeeper* sementara `$ZK_HOME/bin/zkServer.sh stop` digunakan untuk menghentikan service *zookeeper.*

4.1.1.6 Konfigurasi *Tez*

Apache Tez merupakan *framework* yang memungkinkan untuk melakukan pemrosesan data secara kompleks di ekosistem *Hadoop*. *Tez* dapat digunakan untuk menggantikan *MapReduce* yang merupakan *framework* bawaan dari *Hadoop. Tez* berjalan di atas YARN *Resource manager* sama seperti *MapReduce*. *Tez* tidak menyediakan *file binary* sehingga perlu dilakukan kompilasi terlebih dahulu *source code* dari *apache tez* dengan perintah `mvn clean package -DskipTests=true -Dmaven.javadoc.skip=true`. Kompilasi akan menghasilkan *package tez* pada direktori `tez-dist/target` dengan nama tez-x.y.z-SNAPSHOT.tar.gz. Pada implementasi yang dilakukan, penulis menggunakan versi Tez 0.9.0, sehingga *file* yang dihasilkan bernama tez-0.9.0-SNAPSHOT.tar.gz. Selanjutany file tersebut dimasukkan ke dalam HDFS dalam folder /apps/tez-0.9.0-SNAPSHOT dengan perintah `hadoop dfs -mkdir /apps/tez-0.9.0-SNAPSHOT hadoop dfs -copyFromLocal tez-dist/target/tez-0.9.0-SNAPSHOT.tar.gz /apps/tez-0.9.0-SNAPSHOT/ .` Kemudian package tez diekstrak ke dalam folder /usr/local/tez-0.9.0-SNAPSHOT.

Langkah selanjutnya adalah menambahkan environment variables ke dalam file .bashrc. Baris yang ditambahkan adalah sebagai berikut :

```
export ZK_HOME=/usr/local/zookeeper
export TEZ_HOME=/usr/local/tez-0.9.0-SNAPSHOT
export TEZ_JARS=/usr/local/tez-0.9.0-SNAPSHOT
export TEZ_CONF_DIR=/usr/local/tez-0.9.0-SNAPSHOT/conf
exportPATH= $PATH:$HADOOP_HOME/bin:
```

```
$HADOOP_HOME/sbin$HIVE_HOME/bin:$HBASE_HOME/bin:
$CATALINA_HOME/bin:$ZK_HOME/bin

exportCLASSPATH=$CLASSPATH:$HADOOP_HOME/lib/native/*:.
export CLASSPATH=$CLASSPATH:$CATALINA_HOME/lib/*:.
export CLASSPATH=$CLASSPATH:$HIVE_HOME/lib/*:.
export CLASSPATH=$CLASSPATH:$HBASE_HOME/lib/*:.
export CLASSPATH=$CLASSPATH:$TEZ_CONF_DIR/*:.
export CLASSPATH=$CLASSPATH:$TEZ_JARS/*:.
export CLASSPATH=$CLASSPATH:$TEZ_JARS/lib/*:.
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH
:${TEZ_CONF_DIR}:${TEZ_JARS}/*:${TEZ_JARS}/lib/*:.
```

Kemudian file konfigurasi tez dibuat dalam direktori $TEZ_HOME/conf dengan nama tez-site.xml. Berikut merupakan konfigurasi tez-site.xml

```
<configuration>
<property>
<name>tez.lib.uris</name>
<value>${fs.defaultFS}/apps/tez-0.9.0-SNAPSHOT/tez-0.9.0-
SNAPSHOT.tar.gz</value>
</property>
<property>
    <name>tez.am.resource.memory.mb</name>
    <value>4096</value>
</property>
    <property>
      <name>tez.task.launch.cluster-default.cmd-opts</name>
      <value>-server -Djava.net.preferIPv4Stack=true</value>
    </property>
<property>
      <name>tez.am.container.reuse.enabled</name>
      <value>true</value>
    </property>
    <property>
      <name>tez.session.am.dag.submit.timeout.secs</name>
      <value>300</value>
    </property>
    <property>
      <name>tez.session.client.timeout.secs</name>
      <value>-1</value>
    </property>
    <property>
      <name>tez.staging-dir</name>
      <value>/tmp/${user.name}/staging</value>
    </property>
    <property>
      <name>tez.use.cluster.hadoop-libs</name>
      <value>false</value>
    </property>
</configuration>
```

Untuk dapat menggunakan tez sebagai framework utama, maka konfigurasi mapred-site.xml pada direktori $HADOOP_HOME/etc/hadoop juga

harus disesuaikan. Ditambahkan dengan property mapreduce.framework.name seperti berikut :

```
<property>
<name>mapreduce.framework.name</name>
<value>yarn-tez</value>
</property>
```

Selain konfigurasi yarn, konfigurasi yang perlu disesuaikan adalah konfigurasi *Hive* yang terletak di $HIVE_HOME/conf. Pada file hive-site.xml ditambahkan dengan property hive.execution.engine dengan nilai tez seperti berikut :

```
<property>
<name>hive.execution.engine</name>
<value>tez</value></property>
```

4.1.2 Proses Extract Transformation Load (ETL)

Sebelum proses ETL dilakukan penulis melakukan data *profiling* terlebih dahulu untuk melihat karakteristik data yang ada. Profiling dilakukan untuk menentukan data yang tidak normal/*outlier*. Data yang tidak normal tersebut akan dibuang dan tidak akan digunakan pada saat pencarian asoiasi. Sumber data penulis merupakan data transaksi penjualan Amigo selama 3 tahun mulai dari tahun 2010-2012. Data yang diolah merupakan jumlah strip dalam setiap transaksi. Apabila ada 5 buah strip yang berbeda dalam setiap transaksi, maka satu transaksi tersebut memiliki nilai 5. Batasan jumlah strip dalam setiap transaksi yang diterapkan oleh penulis adalah minimal 2 buah. Sehingga transaksi dengan 1 buah strip diabaikan.

Tabel 4. 1 Statistik frekuensi jumlah pembelian barang setiap transaksi

No	Jumlah Strip	Frekuensi		
		2010	2011	2013
1	2	8277	8283	8584
2	3	4021	3896	4601
3	4	2382	2071	2500
4	5	1338	1156	1313

No	Jumlah Strip	Frekuensi		
		2010	2011	2013
5	6	815	699	812
6	7	505	436	525
7	8	316	282	318
8	9	195	191	227
9	10	152	131	155
10	11	111	112	113
11	12	62	58	64
12	13	61	49	56
13	14	36	31	38
14	15	34	26	24
15	16	28	13	19
16	17	14	17	21
17	18	16	9	14
18	19	13	5	12
19	20	5	6	6
20	21	8	2	4
21	22	6	4	5
22	23	8	2	4
23	24	1	1	2
24	25	2	4	5
25	26	3	1	3
26	27	2	0	3
27	28	0	5	2
28	29	1	2	1
29	30	2	1	2
30	31	1	1	1
31	32	0	1	0
32	33	0	0	1

No	Jumlah Strip	Frekuensi		
		2010	2011	2013
33	34	0	0	1
34	35	2	1	0
35	36	1	0	0
36	39	1	0	0
37	42	1	0	0
38	45	1	0	0
39	46	1	0	1
40	49	0	0	1
41	52	1	0	0
42	57	1	0	0
43	97	0	0	1
44	322	0	1	0
Total		18424	17497	19439

Frekuensi kemunculan jumlah strip dapat dilihat pada tabel 4.1 di atas. Frekuensi paling banyak terdapat pada 2 strip setiap transaksi. Setiap tahun memiliki nilai maksimal yang bervariasi yang berbeda-beda. Pada tahun 2010 misalnya terdapat pembeli yang melakukan transaksi dengan jumlah strip 522. Hal tersebut menunjukan adanya *outlier* atau data yang tidak normal. Oleh karena itu penulis menggunakan pendekatan IQR untuk menentukan batasan data yang dikategorikan sebagai data normal. Batas yang digunakan penulis adalah batas dalam (*lower inner fence & upper inner fence*). Rumus penghitungan *low inner fence* dan *upper inner fence* adalah

lower inner fence : Q1-(1.5*(Q3-Q1))

upper inner fence : Q3+(1.5*(Q3-Q1))

Tabel 4. 2 Statistik data penjulan

	2010	2011	2012
MIN	2	2	2
MAX	57	322	97
Q1	2	2	2
MEDIAN	3	3	3
Q3	4	4	4
IQR (Q3-Q1)	2	2	2
Lower inner fence	-1	-1	-1
Upper inner fence	7	7	7

Dari tabel 4.2 dapat diketahui bahwa data dari setiap tahun memiliki nilai Q1, Q2, dan Q3 yang sama. Oleh karena itu maka dapat disimpulkan bahwa batas nilai untuk pasti sama. Batas paling bawah yaitu -1 sedangkan batas paling atas adalah 7, sehingga transaksi yang memiliki lebih dari 7 buah strip tidak digunakan. Dikarenakan nilai paling kecil (*MIN*) dari data setiap tahun adalah 2, maka batas paling bawah tidak digunakan karena nilai paling kecil berada di atas batas paling bawah.

Sebelum dimasukkan ke dalam HDFS, data yang ada terlebih dahulu ditransformasikan sesuai format yang digunakan dalam penelitian ini. Data transaksi yang ada dalam tabel *d_nota_tunai* merupakan data transaksi nota dengan kode barang, bukan strip. Oleh karena itu penulis mengubah dari kode barang menjadi strip terlebih dahulu. Apabila terdapat beberapa barang yang masih memiliki strip yang sama maka akan dianggap satu strip. Penulis membuat tabel baru dalam database amigo dengan nama *d_nota_unclean* yang memiliki struktur sebagai berikut :

Tabel 4. 3 Struktur tabel d_nota_unclean

Nama Kolom	Jenis	Deskripsi
id	Varchar (100)	Nomor nota
Strip	char(4)	Kode strip

Penulis kemudian melakukan *query* untuk memasukan data ke dalam tabel *d_nota_unclean. Query* dilakukan dengan aplikasi *phpmyadmin.* Query yang digunakan adalah sebagai berikut : *INSERT INTO `d_nota_unclean` SELECT no_nota, SUBSTRING(kd_barang,1,2) AS strip FROM `d_nota_tunai` GROUP BY no_nota,strip.* Tabel d_nota_unclean nantinya akan digunakan sebagai sumber data dari tabel fakta penjualan. Contoh hasil dari transformasi dapat dilihat pada gambar 4.3 .

id	strip
01-0110-00001	FA
01-0110-00002	JA
01-0110-00003	DC
01-0110-00004	BJ
01-0110-00004	EC
01-0110-00005	FA
01-0110-00006	BQ
01-0110-00007	BK
01-0110-00008	AM
01-0110-00008	JF
01-0110-00009	AF
01-0110-00009	KA
01-0110-00010	LD
01-0110-00011	AQ

Gambar 4.7 Hasil transformasi kode barang menjadi strip

Proses ETL dilakukan menggunakan *Pentaho Data Integrator* (PDI). . Terdapat 2 jenis proses dalam PDI yaitu *transformasi* dan *job. Job* terdiri dari beberapa transformasi yang digabungkan menjadi satu. Transformasi yang digunakan meliputi *table_input* dan *hadoop_file_ouput.* Sumber data dalam proses load berasal dari tabel-tabel database amigo. Dalam proses ini struktur tabel ditransformasikan sesuai dengan struktur tabel yang ada dalam HDFS.

Penulis menggunakan skema *star* dalam merancang *datawarehouse* yang akan digunakan. Oleh karena itu terdapat tabel fakta dan tabel dimensi. Penulis

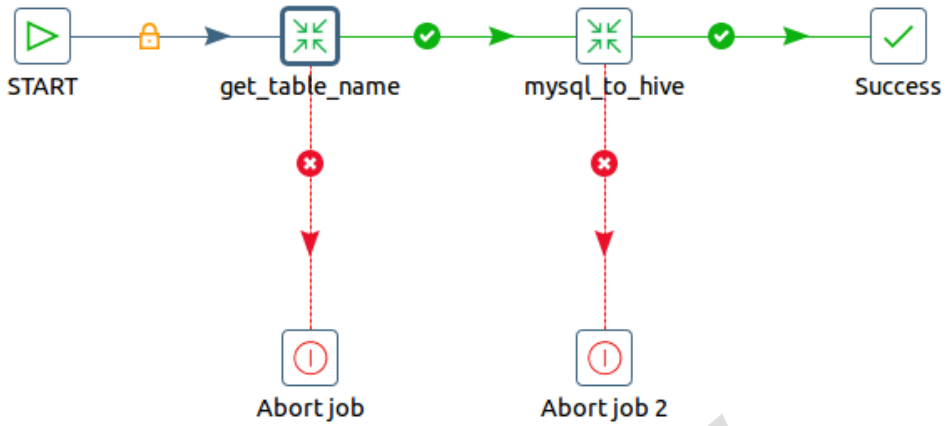
membuat 3 tabel dimensi dan 1 tabel fakta yaitu dimensi waktu, dimensi strip, dimensi lokasi, dan fakta penjualan (*sales_fact*). Setiap primary key dalam tabel dimensi akan menjadi foreign key dalam tabel fakta.

4.1.2.1 ETL Tabel Dimensi

Seperti yang telah disebutkan sebelumnya, model dimensi yang dibentuk dalam penelitian ini terdiri dari 3 tabel dimensi. Setiap tabel dimensi akan menyimpan informasi khusus seperti waktu penjualan, strip, dan juga lokasi penjualan barang.

1. Dimensi Waktu

   Dimensi_waktu akan menyimpan informasi waktu penjualan yang diambil dari informasi waktu nota dari tabel m_nota_tunai. Data waktu yang tersimpan dalam database berupa datetime dengan format YYYY-MM-DD H:i:s. Sementara itu, sistem membutuhkan informasi waktu, yang kemudian disimpan dalam tabel dimensi_waktu, berupa tahun, bulan, tanggal, kuarter, dan season. Untuk mencapai hal tersebut, maka dilakukan proses ETL dengan sumber data dari tabel m_nota_tunai yang menyimpan informasi tanggal nota. Proses ETL dimensi waktu dapat dilihat pada Gambar. 4.5.



Gambar 4. 5 Proses ETL untuk membentuk model dimensional

Gambar 4.5 merupakan proses ETL berupa *job* yang dilakukan di Pentaho untuk membentuk tabel dimensi waktu – dan akan dipakai juga untuk membentuk tabel dimensi lainnya bahkan tabel fakta. *Job* pada Gambar 4.5 dibentuk dari 2 buah *transformation* yaitu `get_table_name` seperti pada Gambar 4.6 dan `mysql_to_hive` seperti pada Gambar 4.7.

Gambar 4. 6 Transformasi get_table_name

Gambar 4. 7 Transformasi mysql_to_hive

Untuk mengambil data dari *MySQL*, pertama-tama dibutuhkan daftar semua tabel yang ada sebagaimana ditunjukkan *transformation* get_table_name pada Gambar 4.6. Pada bagian Get table names, pentaho

melakukan koneksi melalu mysql connector. Selanjutnya, semua *row* yang muncul akan di-*copy* pada bagian Copy rows to result. *Transformation* ini selanjutnya akan digunakan sebagai salah satu step dalam *job* pada Gambar 4.5.

Selanjutnya, pada *transformation* mysql_to_hive bagian Table Input, akan dilakukan koneksi ke tabel yang berisi data mentah dengan mysql connector. Selanjutnya, query dilakukan untuk mengambil data sekaligus membentu data tersebut sesuai kebutuhan seperti terlihat pada Gambar 4.8. Informasi tanggal nota yang diperoleh dari tabel `m_nota_tunai` kemudian diolah sehingga membentuk informasi tahun, bulan, tanggal, dan season. Informasi *season* berupa beberapa hari besar di Indonesia yaitu Imlek, Idul Fitri, dan Natal yang telah disesuaikan dengan kalender nasional Indonesia. Setelah semua informasi yang dibutuhkan sudah dibentuk, data di-*load* ke dalam HDFS melalui bagian Hadoop File Output seperti pada Gambar 4.7. Pada bagian ini, dibutuhkan koneksi ke Hadoop Cluster yaitu dengan mengkonfigurasi hadoop plugin yang terdapat pada direktori `data-integration/plugins/pentaho-big-data-plugin/hadoop-configurations` pada pentaho. Data tersebut dapat langsung di-*load* ke dalam database hive yang ada di dalam Hadoop filesystem. Contoh data yang sudah di-*load* ke dalam hive dapat dilihat pada Gambar 4.4.

```
SELECT
DATE_FORMAT(m.tgl_nota,'%Y-%m-%d') AS id_waktu,
substring(m.tgl_nota,1,4) as tahun,
substring(m.tgl_nota,6,2) as bulan ,
substring(m.tgl_nota,9,2) as tanggal,
(CASE
WHEN substring(m.tgl_nota,1,10) >= '2010-09-09'
  and substring(m.tgl_nota,1,10) <= '2010-09-13' THEN 'idulfitri'
WHEN substring(m.tgl_nota,1,10) >= '2011-08-29'
  and substring(m.tgl_nota,1,10) <= '2011-09-02' THEN 'idulfitri'
WHEN substring(m.tgl_nota,1,10) >= '2012-08-19'
  and substring(m.tgl_nota,1,10) <= '2012-08-26' THEN 'idulfitri'
WHEN substring(m.tgl_nota,1,10) >= '2013-09-05'
  and substring(m.tgl_nota,1,10) <= '2012-09-09' THEN 'idulfitri'
WHEN substring(m.tgl_nota,1,10) = '2010-02-14' THEN 'imlek'
WHEN substring(m.tgl_nota,1,10) = '2011-02-03' THEN 'imlek'
WHEN substring(m.tgl_nota,1,10) = '2012-01-23' THEN 'imlek'
WHEN substring(m.tgl_nota,1,10) = '2013-02-18' THEN 'imlek'
WHEN substring(m.tgl_nota,6,5) = '12-25' THEN 'natal'
ELSE 'biasa'
END) as season,
(CASE
WHEN substring(m.tgl_nota,6,2) >= '01' and substring(m.tgl_nota,6,2) <= '03' THEN 'Q1'
WHEN substring(m.tgl_nota,6,2) >= '04' and substring(m.tgl_nota,6,2) <= '06' THEN 'Q2'
WHEN substring(m.tgl_nota,6,2) >= '07' and substring(m.tgl_nota,6,2) <= '09' THEN 'Q3'
WHEN substring(m.tgl_nota,6,2) >= '10' and substring(m.tgl_nota,6,2) <= '12' THEN 'Q4'
END) as kuartal
FROM m_nota_tunai m GROUP BY id_waktu
```

Gambar 4. 8 Query tabel dimensi_waktu

2. Dimensi Strip

Tabel dimensi_merk akan menyimpan informasi strip barang berupa kode_strip, nama_strip, dan kelompok jenis yang di-*extract* dan di-*transform* dari tabel strip. Proses ETL tabel dimensi_strip hampir sama dengan proses ETL dimensi waktu yaitu dengan menggunakan *job* seperti pada Gambar 4.5. Query untuk memodelkan tabel dimensi_merk dapat dilihat pada Gambar 4.9.

```
SELECT
kd_strip,nama_strip,kel_jns
FROM strip
```

Gambar 4. 9 Query tabel dimensi_merk

3. Dimensi Lokasi

Tabel dimensi_lokasi menunjukkan lokasi toko tempat barang-barang dijual. Tabel ini memiliki 2 buah kolom yaitu id lokasi dan nama lokasi. Amigo Group memiliki 10 gerai, sementara penelitian ini akan mengambil data dari toko yang berada di Klaten, Jawa Tengah (kode toko: 03). Proses ETL tabel dimensi_lokasi hampir sama dengan proses ETL dimensi waktu yaitu dengan menggunakan *job* seperti pada Gambar 4.5.

4.1.2.2 ETL Tabel Fakta

Secara umum, proses ETL pada tabel fakta *sales_fact* hampir sama dengan proses ETL pada tabel-tabel dimensi seperti penjelasan sebelumnya. Proses ETL menggunakan *job* dan *transformation* yang sama dengan ETL tabel dimensi. Salah satu perbedaan terletak pada tabel-tabel yang membentuk tabel fakta yang terdiri dari beberapa tabel sekaligus yaitu tabel `m_nota_tunai`, `d_nota_tunai`, dan tabel `d_nota_unclean`. Tabel fakta juga perlu dibentuk agar menyimpan *foreign key* berupa *primary key* dari tabel-tabel dimensi yang telah

dibentuk sebelumnya. Dari data *profiling* yang telah dilakukan sebelumnya telah ditentukan batasan minimal dan maksimal jumlah strip dalam setiap transaksi yang akan dimasukkan dalam datawarehouse. Oleh karena itu dalam proses ini transaksi yang akan diambil dari tabel *d_nota_unclean* adalah data transaksi yang memiliki jumlah minimal strip 2 dan jumlah maksimal strip 7 .Query untuk membentuk tabel fakta dapat dilihat pada Gambar 4.13.

```
SELECT DATE_FORMAT(m.tgl_nota,'%Y-%m-%d') AS tanggal,
d.no_nota,t.id AS id_toko,
SUBSTRING(kd_barang,1,2) AS strip,
s.kel_jns FROM `d_nota_tunai` d, `m_nota_tunai` m, `strip` s, `toko`t
WHERE d.no_nota IN (
        SELECT id FROM `d_nota_unclean` GROUP BY id HAVING
      COUNT(id)>1 AND COUNT(id)<8)
 AND d.no_nota=m.no_nota
AND SUBSTRING(kd_barang,1,2)=s.kd_strip
 GROUP BY no_nota,strip
```

Gambar 4. 10 Query tabel fakta

4.1.3 Implementasi Proses *Cleaning*

Implementasi data cleaning dilakukan pada tahapan ETL. Proses cleaning dilakukan pada saat data dimasukkan ke dalam *datawarehouse*. Hal tersebut dilakukan untuk mengurangi beban kerja sistem, sehingga pada saat melakukan analisis asosiasi sistem tidak perlu melakukan *cleaning* berulang kali. Kriteria data yang valid adalah transaksi yang memiliki jumlah transaksi lebih dari satu dan kurang dari 8 sesuai dengan batasan yang ditentukan pada saat *profiling data*. Selain itu, data dikatakan valid apabila nilai *tanggal* tidak *null*.

List 4.1 Pseudocode implementasi proses data cleaning

1	BEGIN
2	SELECT DATE_FORMAT(*m.tgl_nota*,'%Y-%m-%d') AS *tanggal*, *d.no_nota*,*t.id* AS *id_toko*,SUBSTRING(*kd_barang*,1,2) AS *strip*, *s.kel_jns* FROM `*d_nota_tunai*` *d*, `*m_nota_tunai*` *m*, `*strip*` *s*, `*toko*` *t* WHERE *d.no_nota* IN (SELECT *id* FROM `*d_nota_unclean*` GROUP BY *id* HAVING COUNT(*id*)>1 AND COUNT(*id*)<8) AND *d.no_nota=m.no_nota* AND SUBSTRING(*kd_barang*,1,2)=*s.kd_strip* GROUP BY *no_nota*,*strip*
3	FOR EACH *tanggal, no_nota,id_toko, strip,kel_jns*DO INSERT INTO *sales_fact* table END FOR
4	END

4.1.4 Implementasi Proses Data *Filtering*

Data yang melewati proses cleaning kemudian akan difilter berdasarkan periode waktu serta lokasi toko sesuai dengan inputan user. Proses filtering dilakukan untuk mendapatkan *candidate* awal yang berupa strip. Strip yang dipilih adalah yang memiliki frekuensi lebih dari atau dengan minium support. Hasil dari proses filtering ini adalah data yang sudah siap untuk diproses sistem dengan menggunakan algoritma *AprioriHybrid.*

List 4. 2 Pseudocode implementasi proses data filtering

1	BEGIN
2	GET *start_date*
3	GET *end_date*
4	GET *lokasi*
5	GET *min_supp*
6	SET C_1
7	SELECT *strip*,COUNT(*strip*) AS *jumlah* FROM *sales_fact* WHERE *tanggal* BETWEEN *start_date* AND *end_date* AND *id_lokasi=lokasi* GROUP BY *strip* HAVING COUNT(*strip)* >=
8	FOR EACH *strip* DO INSERT *strip* INTO C_1 END FOR
9	END

4.1.5 Implementasi Algoritma *AprioriHybrid*

Algoritma *AprioriHybrid* merupakan penggabungan dari Algoritma *Apriori* dan *AprioriTID*. Perbedaan yang mendasar adalah cara penghitungan nilai *support* dari setiap *itemset*. Data yang diolah dalam proses ini adalah data yang telah lolos tahap *filtering*. Hasil dari proses ini adalah *frequent itemset* yang berupa kombinasi dari *strip* yang berbeda-beda.

List 4. 3 Pseudocode Implementasi Algoritma AprioriHybrid

```
1   BEGIN
2   GET C1
3   SET loop = TRUE
4   SET switchalgo = FALSE
5   SET aprioritifdatasetsize = generateDataSetSize
6   SET TBar = NULL
    WHILE loop IS TRUE DO
      Ck = AprioriGen(Ck-1)
      IF Ck IS NOT NULL THEN
          SET freememory = getfreememory
          IF switchalgo IS FALSE AND aprioritifdatasetsize <
          freememory  THEN
              Tbar = generateTBar
              SET switchalgo AS TRUE
          ENDIF
          IF switchalgo IS TRUE
              FOREACH Ck AS candidate
               SET count = 0
               FOREACH TBar AS transaction
                 IF transaction size < k THEN
                     remove transaction from Tbar
                    CONTINUE
                 END IF
                 IF candidate is contain in transaction THEN
                  SET count = count +1
```

7	END IF ENDFOR IF *count* < *min_supp* THEN remove *candidate* from C_k END IF END FOR ELSE FOREACH *Ck* AS *candidate* SET *count = SELECT COUNT(b.no_nota) FROM* *(SELECT DISTINCT no_nota FROM sales_fact* *where strip=candidate[0] …. STRIP=candidate[k]* *HAVING COUNT(no_nota)=k) b;* IF *count* < *min_supp* THEN remove *candidate* from C_k END IF END FOR END IF IF *Ck* IS NULL *loop*=FALSE END IF ELSE *loop*=FALSE END IF END WHILE
8	END

4.1.6 Implementasi *AprioriGen*

Fungsi *AprioriGen* digunakan untuk menggabungkan *candidate itemset* pada saat iterasi ke *k*. Dalam proses ini penggabungan dilakukan pada *candidate itemset* yang memilki urutan elemen pertama sampai dengan *k*-1 yang sama. Apabila sama, kedua *candidate itemset* tersebut akan digabungkan. Hasil penggabungan kemudian dicari *subset* dengan jumlah elemen k-1. Apabila *subset* yang dihasilkan tidak ada dalam C_{k-1} maka kombinasi tersebut akan diabaikan.

List 4. 4 Pseudocode Implementasi AprioriGen

No	Pseudocode
1	BEGIN
2	GET *Ck*
3	GET *k*
4	SET C_{k+1} IF *k == 2* THEN FOR *i = 0* TO C_k*.count*-1 DO FOR *j = i+1* TO C_k*.count*-1 DO *candidate* add *Ck[i]* *candidate* add *Ck[j]* add *candidate* to C_{k+1} END FOR END FOR ELSE FOR *i = 0* TO C_k*.count*-1 DO FOR *j = i+1* TO C_k*.count*-1 DO SET *count* = 0 FOR l=0 TO k-1 DO

5	IF $C_k[i][l]$ == $C_k[i][l]$ *candidate* add $C_k[i][l]$ *count = count +1* ELSE BREAK END IF END FOR IF *count == k-1* *candidate* add $C_k[j][l]$ *subsetk-1[]* = generate subsets from *candidate* SET *qualify* = TRUE FOREACH *subsetk-1[]* AS subset DO IF subset not contain in C_k SET *qualify* = FALSE BREAK END IF END FOR IF *qualify* THEN C_{k+1} add *candidate* END IF END IF END FOR END FOR END IF
6	END

4.1.7 Implementasi Rules Generation

Proses ini menghasilkan rules dari frequent itemset terakhir yang ditemukan. Dalam pemilihan rules, hal yang dipertimbangkan adalah kekuatan lift dan confidence. Apabila sebuah aturan asosiasi memiliki nilai confidence melebihi nilai minimum dan lift lebih dari satu, maka aturan tersebut akan dimasukkan sebagai aturan yang kuat. Apabila tidak memenuhi syarat maka aturan akan diabaikan.

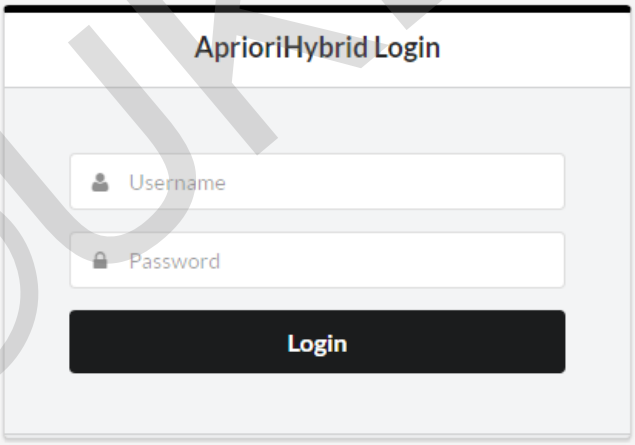
List 4. 5 Pseudocode Implementasi Rules Generation

1	BEGIN
2	GET *C*
3	GET *k*
4	GET *min_conf*
5	GET *transaction_count*
6	SET strong*rules[]*
7	FOR EACH C_k AS *candidate* *subsetpair* = generate pairs from *candidate* FOR EACH *subsetpair* AS *pair* SET *supportfirstelement* = count support first element from C_k SET *supportsecondelement* = count support second element from C_k SET *supportpair = candidate.support* IF 100 * (*supportpair/supportfirstelement*) >= *min_conf* AND ((*supportpair/supportfirstelement*)/(*supportsecondelement/transaction_count*))>=1 THEN *rules* add *pair* *strongrules* add *rules*

	END IF END FOR END FOR
8	END

4.1.8 Implementasi Antarmuka

4.1.8.1 Halaman Login



Gambar 4. 11 Halaman Login

Halaman login digunakan untuk autentikasi pengguna. Sistem yang dibuat hanya bisa digunakan oleh pengguna yang memiliki otoritas. Hal tersebut dikarenakan data yang dianalisis adalah data vital perusahaan yang tidak sembarang orang dapat mengakses dengan bebas. Sistem yang dibuat tidak menerapkan *multiuser,* sehingga pengguna yang ada hanya satu. Pengguna dapat melakukan *login* dengan menginputkan *username* dan *password.* Pengguna yang mencoba mengakses halaman lain tanpa login akan selalu diarahkan ke halaman login oleh sistem.

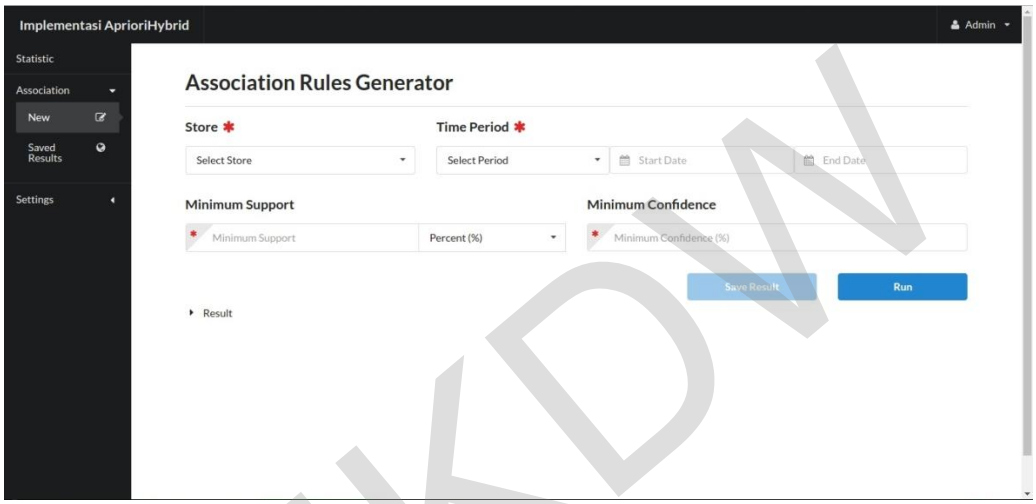
4.1.8.2 Halaman Statistik

Gambar 4. 12 Halaman Statistik

Setelah pengguna melakukan autentikasi dengan benar, maka sistem akan mengarahkan ke halaman statistik. Pada halaman ini pengguna dapat melihat statistik pola asosiasi pada periode waktu yang ditentukan. Pengguna dapat menentukan parameter *min support* dan *min confi*dence secara manual dengan memilih opsi dari dropdown pearameter. Statistik yang ditunjukkan adalah jumlah

pola yang sering muncul.. Halaman statistik dibuat oleh penulis dengan harapan pengguna dapat melihat pola asosiasi yang paling kuat dengan minium support dan minimum confidence yang telah ditentukan sebelumnya.

4.1.8.3 Halaman Association Rules Generator



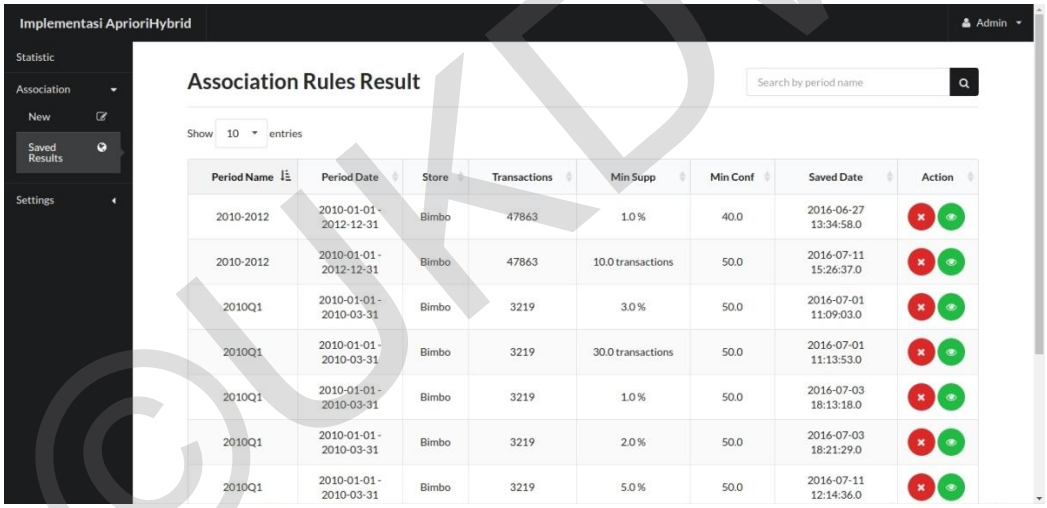
Gambar 4. 13 Halaman Association Rules Generator

Antarmuka pada gambar 4.1.4.3 digunakan untuk mencari asosiasi antar barang pada kurun waktu tertentu. Pada halaman ini pengguna diminta untuk memilih periode waktu yang telah dibuat sebelumnya, minimum support serta minimum confidence. Minimum support merupakan jumlah minimal frekuensi kejadian dari kombinasi itemset dari seluruh transaksi yang ada. Apabila nilai minimum support 10 % dari 1000 transaksi, maka itemset yang dipilih adalah yang memiliki frekuensi 10/100 x 1000 = 100 kejadian. Minimum confidence merupakan nilai minimum kekuatan sebuah aturan yang dibuat dari frequent itemset. Apabila pengguna tidak memasukkan parameter yang diminta, sistem akan menampilkan pesan peringatan. Dalam mencari asosiasi terkadang sistem tidak menemukan aturan yang kuat, apabila demikian sistem akan menampilkan pesan melalui dialog box yang menginformasikan bahwa tidak ada asosiasi yang

kuat pada periode waktu tersebut. Apabila terdapat asosiasi yang kuat, maka sistem akan menyimpan hasil dari analisis ke dalam database.

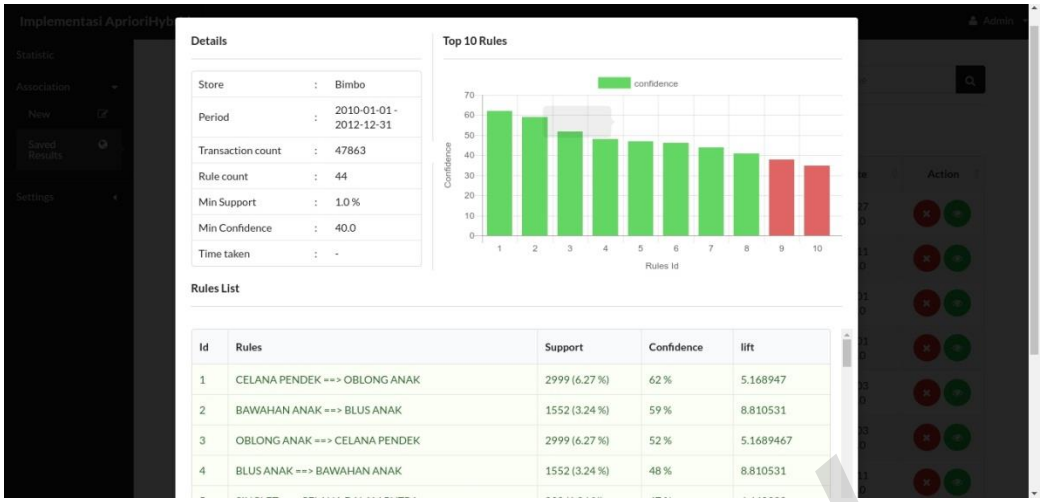
Pengguna kemungkinan menjalankan analisis lebih dari satu kali pada periode waktu, minimum support dan minimum confidence yang sama. Oleh karena itu apabila dalam periode waktu tersebut pengguna pernah melakukan analisis , sistem akan menampilkan dialog box yang menginformasikan bahwa analisis pernah dilakukan, kemudian sistem akan menampilkan hasil yang telah diperoleh sebelumnya.

4.1.8.4 Halaman Rules Result



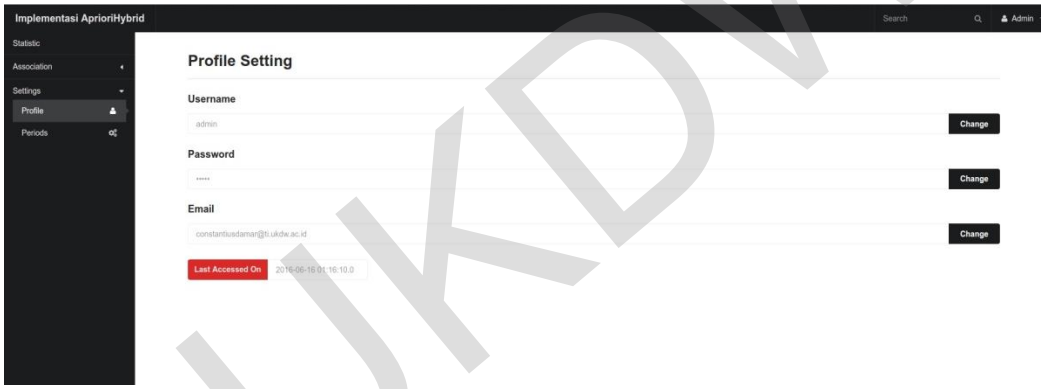
Gambar 4. 14 Halaman Rules Result

Halaman ini berisikan hasil-hasil dari analisis asosiasi yang pernah dilakukan oleh pengguna. Setiap periode ditampilkan berdasarkan nilai minimum support dan minimum confidence. Informasi yang ditampilkan adalah periode transaksi, nilain minimum support, nilai minimum confidence, aturan yang didapatkan beserta nilai confidence dan nilai lift. Apabila pengguna ingin melihat hasil aturan dari analisis yang pernah dilakukan, maka pengguna harus menekan tombol berwarna hijau baris yang diinginkan.



Gambar 4. 15 Detail analisis asosiasi

4.1.8.5 Halaman Edit Profile



Gambar 4. 16 Halaman Edit Profile

Pengguna dapat mengubah username dan password yang dimiliki melalui halaman edit profile. Pengubahan username dapat dilakukan dengan cara menekan tombol “change”. Hal tersebut berlaku juga apabila pengguna ingin mengubah password. Pada saat pengguna menekan tombol change maka sistem akan menampilkan dialog box yang berisikan inputan username atau password yang baru. Pengguna dapat menyimpan password atau username yang baru dengan menekan tombol “save” pada dialog box. Selain mengganti username dan password, pengguna juga dapat mengganti alamat email dengan cara yang sama . Pada antarmuka ini sistem juga menampilkan akses terakhir pengguna ke sistem. Hal tersebut bertujuan untuk memberikan informasi login pengguna.

4.2 Analisis Sistem

Analisis sistem akan menguraikan hasil dari implementasi sistem untuk mencari asosiasi dari data yang ada.

Untuk menganalisis hasil keluaran sistem dalam menghasilkan aturan asosiasi, maka dilakukan pengujian sebanyak 12 kali dengan menggunakan data transaksi penjualan Amigo Group, khususnya toko Bimbo pada penjualan periode 1 Januari 2010 – 31 Desember 2012. Pengujian dilakukan dengan menggunakan jumlah data yang berbeda-beda namun parameter *minimum support* dan *minimum confidence* yang konstan. Setiap pengujian rentang waktu penjualan yang digunakan adalah 3 bulan (1 *quarter*).

Tabel 4. 4 Rentang waktu tiap periode

Periode	Rentang Waktu
Q1	1 Januari – 31 Maret
Q2	1 April – 30 Juni
Q3	1 July – 30 September
Q4	1 Oktober – 31 Desember

Penentuan minimum support yang digunakan untuk analisis dilakukan dengan cara menguji 30% data (tahun 2010) menggunakan *minimum support* mulai dari 1% sampai dengan 5%. Apabila pada *minimum support* ke *n*, hasil asosiasi tidak mengalami perubahan dari minimum support sebelumnya. Maka minimum support ke *n* tersebut yang digunakan untuk menganalisis data. Dari pengujian pada tabel 4.5 dan 4.6 didapatkan bahwa pada *minimum support* 3% ,4% dan 5% sebagian besar jumlah kombinasi item, jumlah aturan yang kuat serta jumlah semua aturan tidak mengalami perubahan. Oleh karena itu nilai minimum support yang digunakan penulis untuk menguji data yaitu 3%.

Tabel 4. 5 Pengujian min support 2010Q1 dan 2010Q2

	1					2				
Periode	2010Q1					2010Q2				
Jumlah Transaksi	3219					3065				
Jumlah Item	90					90				
Minimum Confidence	50 %					50 %				
Minimum Support (%)	1	2	3	4	5	1	2	3	4	5
Strong Rule	4	2	2	2	2	4	4	3	3	2
Total Rule	12	16	4	4	2	12	18	4	4	2
Item count	3	2	2	2	2	3	2	2	2	2
Max Support (%)	1.49	9.94	9.94	9.94	9.94	1.79	9.10	9.10	9.10	9.10

Tabel 4. 6 Pengujian min support 2010Q3 dan 20104

	3					4				
Periode	2010Q3					2010Q4				
Jumlah Transaksi	7151					2414				
Jumlah Item	90					90				
Minimum Confidence	2010Q3					2010Q4				
Minimum Support (%)	1	2	3	4	5	1	2	3	4	5
Strong Rule	2	4	4	4	4	5	5	2	2	2
Total Rule	6	10	8	6	4	34	10	4	4	2
Item count	3	2	2	2	2	2	2	2	2	2
Max Support (%)	1.03	6.24	6.24	6.24	6.24	5.63	5.63	5.63	5.63	5.63

Tabel 4. 7 Hasil Pengujian Tahun 2010

	1	2	3	4
Periode	2010Q1	2010Q2	2010Q3	2010Q4
Jumlah Transaksi	3219	3065	7151	2414
Jumlah Item	2	2	2	2
Minimum Support	3%	3%	3%	3%
Minimum Confidence	50 %	50 %	50 %	50 %
Strong Rule	2	3	4	2
Total Rule	4	4	8	4
Max Support	9,94%	9,01%	6,24%	5,63%

Tabel 4. 8 Hasil Pengujian Tahun 2011

	1	2	3	4
Periode	2011Q1	2011Q2	2011Q3	2011Q4
Jumlah Transaksi	3126	2728	6766	2594
Jumlah Item	2	2	2	2
Minimum Support	3%	3%	3%	3%
Minimum Confidence	50 %	50 %	50 %	50 %
Strong Rule	2	2	3	2
Total Rule	4	4	8	4
Max Support	6,53%	5,65%	4,64%	7,29%

Tabel 4. 9 Hasil Pengujian Tahun 2012

	1	2	3	4
Periode	2012Q1	2012Q2	2012Q3	2012Q4
Jumlah Transaksi	3431	3246	6799	3324
Jumlah Item	2	2	2	2
Minimum Support	3%	3%	3%	3%
Minimum Confidence	50 %	50 %	50 %	50 %
Strong Rule	2	2	3	2
Total Rule	4	4	8	4
Max Support	8,74%	6,99%	4,29%	6,29%

Berdasarkan hasil pengujian tersebur dapat diketahui sebaran jumlah transaksi, jumlah item /strip, *strong rule* atau aturan asosiasi yang memenuhi kriteria di atas nilai batas *confidence*, jumlah total *rule* yang dihasilkan (termasuk dengan aturan yang tidak memenuhi nilai batas *confidence*) serta nilai maksimum support yang dihasilkan dari kombinasi item. Dari data pengujian dapat dilihat bahwa periode yang paling banyak menghasilkan aturan terdapat pada kuartal ketiga dari tahun 2010-2012. Selain kuartal ketiga, jumlah total aturan yang dihasilkan dari masing-masing periode memiliki kemiripan atau dapat dikatkan sama. Banyaknya aturan yang dihasilkan pada kuartal ketiga juga dipengaruhi dengan naiknya jumlah transaksi pada periode tersebut. Pada periode selain kuartal ketiga, total transaksi yang terjadi berkisar antara 2000-3500 transaksi, namun pada kuartal ketiga jumlah transaksi berada di atas 5000.

Hasil asosiasi pada setiap kuartal dapat dilihat pada tabel 4.10. Dari tabel tersebut dapat dilihat *trend* pembelian dari setiap kuartal. Kuartal pertama, kedua,

dan keempat setiap tahun mulai 2010-2012 pada umumnya memiliki kesamaan pola, yaitu barang yang dibeli merupakan celana pendek dan oblong anak. Sedangkan kuartal ketiga setiap tahun juga memiliki kesamaan trend. Barang yang dibeli pada kuartal tersebut adalah blus anak, bawahan anak, celana pendek serta oblong anak. Perbedaan antara kuartal satu, dua , empat dengan tiga dikarenakan *margin* jumlah transaksi yang cukup besar. Semakin banyak data transaksi, maka semakin banyak pula aturan asosiasi yang akan dihasilkan menggunakan parameter *minium support* dan *minimum confidence* yang konstan

Tabel 4. 10 Hasil Aturan Asosiasi Pada Setiap Kuartal

No	Periode	Aturan	% perulangan pola
1	2010Q1	1. CELANA PENDEK => OBLONG ANAK 2. OBLONG ANAK => CELANA PENDEK	-
2	2010Q2	1. CELANA PENDEK => OBLONG ANAK 2. OBLONG ANAK => CELANA PENDEK 3. BH REMAJA => CELANA DALAM PUTRI	-
3	2010Q3	1. BAWAHAN ANAK => BLUS ANAK 2. CELANA PENDEK => OBLONG ANAK 3. BLUS ANAK => BAWAHAN ANAK 4. OBLONG ANAK => CELANA PENDEK	-
4	2010Q4	1. CELANA PENDEK => OBLONG ANAK 2. OBLONG ANAK => CELANA PENDEK	-
5	2011Q1	1. CELANA PENDEK => OBLONG ANAK 2. OBLONG ANAK => CELANA PENDEK	100
6	2011Q2	1. CELANA PENDEK => OBLONG ANAK 2. OBLONG ANAK => CELANA PENDEK	66

No	Periode	Aturan	% perulangan pola
7	2011Q3	1. BAWAHAN ANAK => BLUS ANAK 2. CELANA PENDEK => OBLONG ANAK 3. BLUS ANAK => BAWAHAN ANAK	75
8	2011Q4	1. CELANA PENDEK => OBLONG ANAK 2. OBLONG ANAK => CELANA PENDEK	100
9	2012Q1	1. CELANA PENDEK => OBLONG ANAK 2. OBLONG ANAK => CELANA PENDEK	100
10	2012Q2	1. CELANA PENDEK => OBLONG ANAK 2. OBLONG ANAK => CELANA PENDEK 3. BH REMAJA => CELANA DALAM PUTRI	100
11	2012Q3	1. BAWAHAN ANAK => BLUS ANAK 2. BLUS ANAK => BAWAHAN ANAK 3. CELANA PENDEK => OBLONG ANAK	75
12	2012Q4	1. CELANA PENDEK => OBLONG ANAK 2. OBLONG ANAK => CELANA PENDEK	100

Hasil pengujian pada tabel 4.10 menunjukkan bahwa pada kuartal pertama terdapat pola asosiasi yang berulang setiap tahunnya. Presentase pola yang berulang yaitu 100%, dikarenakan jumlah pola dan hubungan antar barang yang sama persis pada tiap tahun. Setiap kuartal pertama pola asosiasi yang terbentuk merupakan kombinasi dari 2 barang yaitu celana pendek dan oblong anak. Pola asosiasi pada kuartal 2 juga berulang setiap tahun. Presentase pola yang berulang pada tahun 2011 hanyalah 66% dibandingkan dengan tahun 2010. Sedangkan pada tahun 2012 presentase pola yang berulang berjumlah 100%. Hal tersebut dikarenakan pola yang terbentuk pada kuartal kedua tahun 2012 sama persis dengan tahun 2010. Pada kuartal ketiga presentase pola yang berulang hanya 75% pada tahun 2011 dan 2012 apabila dibandingakan dengan tahun 2010. Kuartal

ketiga tahun 2011 dan 2012 menghasilkan pola dengan susunan kombinasi yang masing-masing memiliki 2 buah item penyusun. Pada kuartal keempat keberulangan pola memiliki presentasi nilai 100%. Dari tahun 2010 sampai dengan tahun 2012, pola yang terbentuk memiliki kesamaan jumlah dan susunan kombinasi. Dari tabel 4.10 juga dapat disimpulkan bahwa susunan kombinasi item yang paling sering muncul pada setiap kuartal merupakan celana pendek dan oblong anak. Presentase kemunculan kombinasi barang tersebut bernilai 100%. Pengujian pada setiap kuartal selalu menghasilkan kombinasi barang tersebut.

Hasil pengujian dengan menggunakan minimum support 3% selalu menghasilkan kombinasi strip yang berjumlah 2. Percobaan selanjutnya dilakukan dengan menurunkan nilai *minimum support* menjadi 2% dan 1%. *Minimum Confidence* tidak berubah dari pengujian sebelumnya, nilai yang digunakan yaitu 50%.

Tabel 4. 11 Hasil pengujian tahun 2010 dengan minimum support 2 %

	1	2	3	4
Periode	2010Q1	2010Q2	2010Q3	2010Q4
Jumlah Transaksi	3219	3065	7151	2414
Jumlah Item	2	2	2	2
Minimum Support	2%	2%	2%	2%
Minimum Confidence	50 %	50 %	50 %	50 %
Strong Rule	2	4	4	5
Total Rule	16	18	10	10
Max Support	9,94%	9,10%	6,24%	5,63%

Tabel 4. 12 Hasil pengujian tahun 2011 dengan minimum support 2 %

	1	2	3	4
Periode	2011Q1	2011Q2	2011Q3	2011Q4
Jumlah Transaksi	3126	2728	6766	2594
Jumlah Item	2	2	2	2
Minimum Support	2%	2%	2%	2%
Minimum Confidence	50 %	50 %	50 %	50 %
Strong Rule	2	5	3	2
Total Rule	10	14	14	16
Max Support	6,53%	5,65%	4,64%	7,29%

Tabel 4. 13 Hasil pengujian tahun 2012 dengan minimum support 2 %

	1	2	3	4
Periode	2012Q1	2012Q2	2012Q3	2012Q4
Jumlah Transaksi	3431	3246	6799	3324
Jumlah Item	2	2	2	2
Minimum Support	2%	2%	2%	2%
Minimum Confidence	50 %	50 %	50 %	50 %
Strong Rule	2	3	3	2
Total Rule	8	16	16	14
Max Support	8,74%	6,99%	4,29%	6,29%

Tabel 4. 14 Hasil pengujian tahun 2010 dengan minimum support 1 %

	1	2	3	4
Periode	2010Q1	2010Q2	2010Q3	2010Q4
Jumlah Transaksi	3219	3065	7151	2414
Jumlah Item	3	3	3	2
Minimum Support	1%	1%	1%	1%
Minimum Confidence	50 %	50 %	50 %	50 %
Strong Rule	4	4	2	5
Total Rule	12	12	6	34
Max Support	1,49%	1,79%	1,03%	5,63%

Tabel 4. 15 Hasil pengujian tahun 2011 dengan minimum support 1 %

	1	2	3	4
Periode	2011Q1	2011Q2	2011Q3	2011Q4
Jumlah Transaksi	3126	2728	6766	2594
Jumlah Item	2	2	2	3
Minimum Support	1%	1%	1%	1%
Minimum Confidence	50 %	50 %	50 %	50 %
Strong Rule	3	7	3	2
Total Rule	48	50	52	6
Max Support	6,53%	5,65%	4,37%	1,31%

Tabel 4. 16 Hasil pengujian tahun 2012 dengan minimum support 1 %

	1	2	3	4
Periode	2012Q1	2012Q2	2012Q3	2012Q4
Jumlah Transaksi	3431	3246	6799	3324
Jumlah Item	3	3	2	3
Minimum Support	1%	1%	1%	1%
Minimum Confidence	50 %	50 %	50 %	50 %
Strong Rule	1	1	3	2
Total Rule	6	6	56	6
Max Support	1,17%	1,11%	4,29%	1,08%

Pengujian dengan menggunakan *minimum support* 1% dan 2% memperlihatkan bahwa semakin kecil parameter *minimum support* maka *rule* atau aturan yang dihasilkan akan semakin banyak. Hal tersebut dikarenakan syarat frekuensi kemunculan kecil. Walaupun demikian, namun *strong rule* yang dihasilkan tidak banyak dikarenakan nilai *confidence* yang dimiliki tidak berada di atas *threshold.* Oleh karena itu, nilai *minimum support* tidak berpengaruh langsung pada nilai *confidence*. Nilai *minimum support* yang kecil juga menghasilkan aturan dengan jumlah elemen penyusun yang berbeda-beda. Hal tersebut membuat pola kemunculan aturan pada periode berikutnya memiliki kemungkinan yang sangat kecil.

Berdasarkan beberapa hasil pengujian di atas, pola asosiasi yang dihasilkan oleh sistem dengan menggunakan parameter *minimum support* 3% dan *minimum confidence* 50% mempunyai kemungkinan untuk berulang pada tahun berikutnya. Oleh karena itu, dapat dikatakan bahwa hasil keluaran sistem tepat

dan dapat digunakan untuk pendukung pengambilan keputusan strategis oleh pengguna.

Lampiran A

SOURCE CODE

AprioriHybrid.java

```
package Struts2Example.APHybrid.algo;

import
Struts2Example.APHybrid.model.CandidateIte
msets;
import
Struts2Example.APHybrid.model.ItemSet;
import
Struts2Example.APHybrid.model.Rules;
import
Struts2Example.APHybrid.model.SingleTransa
ction;
import
Struts2Example.APHybrid.model.Transactions
Bar;
import
Struts2Example.APHybrid.util.AprioriUtil;
import
Struts2Example.Database.DBConnector;
import
Struts2Example.Database.HiveConnector;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.TreeMap;
import java.util.logging.Level;
import java.util.logging.Logger;
import jodd.util.collection.SortedArrayList;

/**
 *
 * @author constantius_damar
 */
public class AprioriHybrid {

    private double minSupp;
    private int supportReal;
    private int minConf;
    private String startDate;
    private String endDate;
    private boolean usingRealSupport;
    private int counter;
    private ArrayList<Rules> rules;
    private TreeMap<String, Integer> TreeTBar;
    private int numberOfTransaction;
    private TransactionsBar transactionsBar;
    private CandidateItemsets cm;
    private String storeId;
    private long time;
    private Map<String, Object> session;

    private final String INIT = "Initializing
process";
    private final String PROC_AT_K = "Pruning
process";
    private final String STOP = "Process
stopped";
    private final String GEN_RULES
="Generating rules";
    private final String
SWITCH_ALGO="Switching algorithm";

    public long getTime() {
        return time;
    }

    public String getStoreId() {
        return storeId;
    }

    public ArrayList<Rules> getRules() {
        return rules;
    }

    public int getCounter() {
        return counter;
    }

    public int getSupportReal() {
        return supportReal;
    }

    public int getNumberOfTransaction() {
        return numberOfTransaction;
    }

    public Map<String, Object> getSession() {
        return session;
    }
```

```
  public AprioriHybrid(double minSupp, boolean usingRealSupport, int minConf, String startDate, String endDate,String storeId) {
    this.minConf = minConf;
    this.usingRealSupport = usingRealSupport;
    this.startDate = startDate;
    this.endDate = endDate;
    this.rules = new ArrayList<>();
    this.transactionsBar = new TransactionsBar();
    this.cm = new CandidateItemsets();
    this.storeId= storeId;
  }

    public AprioriHybrid(double minSupp, boolean usingRealSupport, int minConf, String startDate, String endDate,String storeId,Map<String, Object> session) {
    this.minSupp = minSupp;
    this.minConf = minConf;
    this.usingRealSupport = usingRealSupport;
    this.startDate = startDate;
    this.endDate = endDate;
    this.rules = new ArrayList<>();
    this.transactionsBar = new TransactionsBar();
    this.cm = new CandidateItemsets();
    this.storeId= storeId;
    this.session=session;
  }

  public ArrayList getInitialCandidate() {
    ArrayList returnList = new ArrayList<>();
    try {

      Connection conn = new HiveConnector().getConn();
      String sql = "select strip,count(strip) from sales_fact_acid where tanggal between ? and ? and id_toko=? group by strip having count(strip)>=? order by strip";

      PreparedStatement ps = conn.prepareStatement(sql);
      ps.setString(1, this.startDate);
      ps.setString(2, this.endDate);
      ps.setString(3, this.storeId);
      if (usingRealSupport) {
        ps.setInt(4, this.supportReal);
      } else {
        ps.setInt(4, (int) this.minSupp);
      }
      ResultSet rs = ps.executeQuery();
      while (rs.next()) {
        returnList.add(rs.getString(1));
      }
      rs.close();
      ps.close();
      conn.close();
    } catch (Exception e) {
      System.out.println("no strip found with given frequency");
      //e.printStackTrace();
    }

    return returnList;
  }

  public TreeMap<String, ItemSet> getInitialCandidate(boolean isReturnTree) {
    TreeMap<String, ItemSet> returnTree = new TreeMap<>();
    try {

      //Connection conn = new HiveConnector("localhost", "hive", "", "default").getConn();
      Connection conn = new HiveConnector().getConn();
      String sql = "select strip,count(strip) from sales_fact_acid where tanggal between ? and ? and id_toko=? group by strip having count(strip)>=? order by strip";

      PreparedStatement ps = conn.prepareStatement(sql);
      ps.setString(1, this.startDate);
      ps.setString(2, this.endDate);
      ps.setString(3, this.storeId);
      if (usingRealSupport) {
        ps.setInt(4, this.supportReal);
      } else {
        ps.setInt(4, (int) this.minSupp);
      }
      ResultSet rs = ps.executeQuery();

      while (rs.next()) {
        ItemSet item = new ItemSet();
        item.addItem(rs.getString(1));
        item.setSupport(rs.getInt(2));
        this.counter += rs.getInt(2);
        returnTree.put(item.toString(), item);
      }
      rs.close();
      ps.close();
      conn.close();
    } catch (Exception e) {
      System.out.println("no strip found with given frequency");
      //e.printStackTrace();
    }

    return returnTree;
  }
```

```
  public void execute() throws SQLException {
    //preparation

//convert support
    //convert confidence
    //get intitial treemap
    //do loop
    //ap_gen
    //check memory
    //count support
    //ap
    //apTID
    //count confidence
    //generate all subset
    boolean state = true;
    boolean switchAlgo = false;

    Connection conn = new HiveConnector().getConn();
    PreparedStatement ps = conn.prepareStatement("select max(b.jumlah),count(b.jumlah) from (select count(no_nota) as jumlah from sales_fact_acid where tanggal BETWEEN ? AND ? AND id_toko=? group by no_nota) b");
    ps.setString(1, this.startDate);
    ps.setString(2, this.endDate);
    ps.setString(3, this.storeId);
    ResultSet rs = ps.executeQuery();
    rs.next();
    SingleTransaction estimator = new SingleTransaction("01-001-111");
    for (int i = 0; i < rs.getInt(1); i++) {
      estimator.getItemList().add("AA");
      estimator.getHashItems().add("A" + i);
    }
    this.numberOfTransaction = rs.getInt(2);
    rs.close();
    ps.close();
    conn.close();
    System.out.println("jumlah transaksi : " + rs.getInt(2));
    if (usingRealSupport) {
      double support = this.minSupp / 100;
      this.supportReal = (int) Math.ceil(rs.getInt(2) * support);
      System.out.println("nilai support : " + supportReal);
    } else {
      System.out.println("nilai support : " + this.minSupp);
    }


cm.addTreeAtIndex(getInitialCandidate(true), 1);
    System.out.println("candidate baru k=" + 1 + " =" + cm.getTreeAtIndexKey(1).size());
    long startime = System.currentTimeMillis();

    System.out.println("jumlah total transaksi dalam setiap itemset =" + counter);

    if (cm.getTreeAtIndexKey(1).size() == 0) {
      state = false;
    }
    this.session.put("progress", INIT);
    int k;
    for (k = 2; state != false; k++) {
      TreeMap<String, ItemSet> newSet = AprioriUtil.apriori_gen(cm.getTreeAtIndexKey(k - 1), k);
      System.out.println("candidate baru k=" + k + " =" + newSet.size());
      this.session.put("progress", PROC_AT_K+" "+k+" combinations");
      if (switchAlgo == false) {
        int jumlahTransaksi = getTransactionsHavingStrip(k);
        float estimateBarSize = 0;
        float freeMem = Runtime.getRuntime().freeMemory();
        try {
          //estimateBarSize = jumlahTransaksi * AprioriUtil.getDeepMemory(estimator) + this.counter * AprioriUtil.getDeepMemory(cm.getTreeAtIndexKey(k - 1).firstEntry().getValue());
          estimateBarSize = jumlahTransaksi * AprioriUtil.getDeepMemory(estimator);
        } catch (IllegalAccessException ex) {

Logger.getLogger(AprioriHybrid.class.getName()).log(Level.SEVERE, null, ex);
        }
        if (freeMem > estimateBarSize) {
          switchAlgo = true;
          //23mei2016
          transactionsBar = generateTBar(k);
          //this.session.put("progress", SWITCH_ALGO);
          System.out.println("switch algo at k =" + k);
          System.out.println("freememory :" + freeMem / (1024) + " KB");
          System.out.println("barsize :" + estimateBarSize / (1024) + " KB");
        } else {
          System.out.println("not enough memory need : " + estimateBarSize / (1024) + "KB but we have :" + freeMem / (1024) + "KB");
        }

      }
```

```
        if (newSet.size() > 0) {
          //checking switch algo
          if (!switchAlgo) {

AprioriCountSupportAndPrune(newSet);
          } else {

AprioriTIDCountSupportAndPrune(transactio
nsBar, newSet, true);
          }

          if (newSet.size() > 0) {
            //cm.addTreeAtIndex(newSet, k +
1);
            cm.addTreeAtIndex(newSet, k);
          } else {
            this.session.put("progress", STOP);
            state = false;
            System.out.println("1.stopped at k="
+ k);
            k--;

          }

        } else {

          this.session.put("progress", STOP);
          state = false;
          System.out.println("2.stopped at k=" +
k);
          k--;
        }
      }
      //System.out.println(k);
      this.session.put("progress", GEN_RULES);
      for (Map.Entry<String, ItemSet> entrySet :
cm.getTreeAtIndexKey(k - 1).entrySet()) {

entrySet.getValue().addAllRulesTo(this.rules);

      }
      HashMap<String,String> strip = new
HashMap<>();
      Connection conn1 = new
DBConnector().getConn();
      String sql = "SELECT * FROM strips";
      ResultSet rs1 =
conn1.createStatement().executeQuery(sql);
      while (rs1.next()) {
        strip.put(rs1.getString(1),
rs1.getString(2));
      }
      rs1.close();
      conn1.close();

      for (Iterator<Rules> iterator =
rules.iterator(); iterator.hasNext();) {
        Rules rulesItem = iterator.next();
        int supportPrim =
cm.getTreeAtIndexKey(rulesItem.getPrimary(
).size()).get(rulesItem.primaryToString()).getS
upport();
        int supportSec =
cm.getTreeAtIndexKey(rulesItem.getSecondar
y().size()).get(rulesItem.secondaryToString()).
getSupport();
        int supportAll =
cm.getTreeAtIndexKey(rulesItem.getSize()).ge
t(rulesItem.toString()).getSupport();
        float confidence = (float) supportAll /
supportPrim;
        int confidenceInPercent = (int)
(confidence * 100);
        float confidenceB = (float) supportSec /
this.numberOfTransaction;
        float lift = confidence / confidenceB;

rulesItem.setConfidence(confidenceInPercent)
;
        rulesItem.setLift(lift);
        rulesItem.setFullName(strip);

//        if (confidenceInPercent < this.minConf
|| lift < 1) {
//          rulesItem = null;
//          iterator.remove();
//        }
      }
      strip=null;
      long endtime =
System.currentTimeMillis();
      System.out.println("at the end of the
process : ");
      System.out.println("count of transaction :
" + this.numberOfTransaction);
      try {
        System.out.println("candidate tree
memory : " + AprioriUtil.getDeepMemory(cm)
/ 1024 + " KB");
        System.out.println("tbar memory : " +
AprioriUtil.getDeepMemory(transactionsBar)
/ (1024) + " KB");
      } catch (IllegalAccessException ex) {

Logger.getLogger(AprioriHybrid.class.getNam
e()).log(Level.SEVERE, null, ex);
      }

      if (switchAlgo) {
        transactionsBar = null;
      }

      this.time=(endtime - startime) / 1000;
      System.out.println("free memory : " +
Runtime.getRuntime().freeMemory() / (1024)
+ " KB");
      System.out.println("lama proses : " +
getTime() + "s");

System.out.println("====================
```

```
======|==|==========================");
  private String generateSQLSupport(SortedArrayList<String> list) {
    String toReturn = "SELECT COUNT(b.no_nota) FROM (SELECT DISTINCT no_nota FROM sales_fact_acid WHERE ";
    for (int i = 0; i < list.size(); i++) {
      toReturn += "strip='" + list.get(i) + "'";
      if (i != list.size() - 1) {
        toReturn += " OR ";
      }
    }
    toReturn += " AND tanggal BETWEEN '" + this.startDate + "' AND '" + this.endDate + "' AND id_toko='"+this.storeId+"' HAVING COUNT(strip)=" + list.size() + ") b";
    //System.out.println(toReturn);
    return toReturn;
  }

  public void AprioriCountSupportAndPrune(TreeMap<String, ItemSet> tm) throws SQLException {
    double minsupp;
    if (this.usingRealSupport) {
      minsupp = this.supportReal;
    } else {
      minsupp = this.minSupp;
    }
    //Connection conn = new HiveConnector("localhost", "hive", "", "default").getConn();
    Connection conn = new HiveConnector().getConn();
    Statement statement = null;
    ResultSet rs = null;
    Iterator<Map.Entry<String, ItemSet>> iter = tm.entrySet().iterator();
    while (iter.hasNext()) {
      Map.Entry<String, ItemSet> entrySet = iter.next();

//System.out.print(generateSQLSupport(entrySet.getValue().getItems())+" ");

System.out.print(entrySet.getValue().getItems() + " ");
      statement = conn.createStatement();
      rs = statement.executeQuery(generateSQLSupport(entrySet.getValue().getItems()));
      while (rs.next()) {
        System.out.print(rs.getInt(1));
        if (rs.getInt(1) < minsupp) {
          entrySet.setValue(null);
          iter.remove();
          System.out.println("--->removed");
        } else {
  }

entrySet.getValue().setSupport(rs.getInt(1));
          System.out.println(" ");
        }
      }
    }
    rs.close();
    statement.close();
    conn.close();
  }

  private TransactionsBar generateTBar(int strip) throws SQLException {
    TransactionsBar toReturn = new TransactionsBar();
    //Connection conn = new HiveConnector("localhost", "hive", "", "default").getConn();
    Connection conn = new HiveConnector().getConn();
    PreparedStatement ps = conn.prepareStatement("SELECT no_nota,collect_set(strip) FROM sales_fact_acid WHERE tanggal BETWEEN ? AND ? AND id_toko=? GROUP by no_nota HAVING COUNT(strip)>=? ORDER BY no_nota");
    ps.setString(1, this.startDate);
    ps.setString(2, this.endDate);
    ps.setString(3, this.storeId);
    ps.setInt(4, strip);
    ResultSet rs = ps.executeQuery();

    String items;
    while (rs.next()) {
      SingleTransaction st = new SingleTransaction(rs.getString(1));
      items = rs.getString(2);
      items = items.replace("\"", "").replace("[", "").replace("]", "").replace(",", "-");
      String[] itemSingles = items.split("-");
      for (String itemSingle : itemSingles) {
        st.getItemList().add(itemSingle);
        st.getHashItems().add(itemSingle);
      }

      //System.out.println(st.getItemList());

toReturn.addSingleTransaction(rs.getString(1), st);
      //System.out.println(rs.getString(1)+" "+st.getItemList());
      itemSingles = null;
    }
    items = null;

    rs.close();
    ps.close();
```

```
    conn.close();
    return toReturn;

 public void AprioriTIDCountSupportAndPrune(TransactionsBar tb, TreeMap<String, ItemSet> tm, boolean modify) {
    double minsupp;
    if (this.usingRealSupport) {
      minsupp = this.supportReal;
    } else {
      minsupp = this.minSupp;
    }
    Iterator<Map.Entry<String, ItemSet>> iter = tm.entrySet().iterator();
    while (iter.hasNext()) {
      Map.Entry<String, ItemSet> entrySetItem = iter.next();
      Iterator<Map.Entry<String, SingleTransaction>> iter2 = tb.getTransactions().entrySet().iterator();
      ////////////////////////////////
      while (iter2.hasNext()) {
        Map.Entry<String, SingleTransaction> entrySet = iter2.next();
        if (entrySet.getValue().getItemList().size() >= entrySetItem.getValue().getItems().size()) {
          boolean state = true;
          for (int i = 0; i < entrySetItem.getValue().getItems().size(); i++) {
            if (!entrySet.getValue().getHashItems().contains(entrySetItem.getValue().getItems().get(i))) {
              state = false;
              break;
            }
          }
          if (state) {

            entrySetItem.getValue().setSupport(entrySetItem.getValue().getSupport() + 1);
          }
        } else {

          entrySet.getValue().setItemList(null);
          iter2.remove();
        }
        entrySet.getValue().setSubset(null);

      }

      //System.out.print(entrySetItem.getValue().getItems() + " " + entrySetItem.getValue().getSupport() + "<>" + minsupp);
      if (entrySetItem.getValue().getSupport() < minsupp) {
        // System.out.println("--->removed");
        entrySetItem.setValue(null);
        iter.remove();
      } else {
        //System.out.println(" ");
      }
      ////////////////////////////////
    }

  }

  public void AprioriTIDCountSupportAndPrune(TransactionsBar tb, TreeMap<String, ItemSet> tm, boolean modify, boolean modify2) {
    double minsupp;
    if (this.usingRealSupport) {
      minsupp = this.supportReal;
    } else {
      minsupp = this.minSupp;
    }
    Iterator<Map.Entry<String, ItemSet>> iter;
    Map.Entry<String, ItemSet> entrySetItem;
    TransactionsBar tbNew = new TransactionsBar();
    Iterator<Map.Entry<String, SingleTransaction>> iter2 = tb.getTransactions().entrySet().iterator();
    while (iter2.hasNext()) {
      Map.Entry<String, SingleTransaction> entrySet = iter2.next();
      SingleTransaction stTemp = entrySet.getValue();
      iter = tm.entrySet().iterator();
      SingleTransaction stNew = new SingleTransaction(entrySet.getKey());
      ;
      while (iter.hasNext()) {
        entrySetItem = iter.next();
        ItemSet itemTemp = entrySetItem.getValue();

        if (stTemp.getHashItems().contains(itemTemp.getFirstSubset()) && stTemp.getHashItems().contains(itemTemp.getSecondSubset())) {

          stNew.getHashItems().add(itemTemp.toString());

          itemTemp.setSupport(itemTemp.getSupport() + 1);

        }

        if (iter2.hasNext() == false) {
```

```
            if (itemTemp.getSupport() < minsupp) {
                itemTemp = null;
            }
           }

        }

        if (stNew.getHashItems().size() != 0) {

tbNew.addSingleTransaction(stNew.getKey(), stNew);
        }

      }
      this.transactionsBar = null;
      this.transactionsBar = tbNew;
    }

    private int getTransactionsHavingStrip(int strip) {
      int toReturn = 0;
      try {
        Connection conn = new HiveConnector().getConn();
        PreparedStatement ps = conn.prepareStatement("select count(sq.no_nota) from (select no_nota from sales_fact_acid where tanggal between ? and ? and id_toko=? group by no_nota having count(strip)>=?) sq");
        ps.setString(1, this.startDate);
        ps.setString(2, this.endDate);
        ps.setString(3, this.storeId);
        ps.setInt(4, strip);
        ResultSet rs = ps.executeQuery();
        rs.next();
        toReturn = rs.getInt(1);
      } catch (Exception e) {
        e.printStackTrace();
      }
      return toReturn;
    }
}
```

AprioriUtil.java

```
package Struts2Example.APHybrid.util;


import com.javamex.classmexer.MemoryUtil;
import java.util.ArrayList;
import java.util.TreeMap;
import jodd.util.collection.SortedArrayList;
import Struts2Example.APHybrid.model.ItemSet;

/**
 *
            iter.remove();


 * @author constantius_damar
 */
public class AprioriUtil {

  public static TreeMap<String, ItemSet> apriori_gen(TreeMap<String, ItemSet> candidates, int phase, boolean modify) {
    TreeMap<String, ItemSet> newCandidates = new TreeMap<>();
    return newCandidates;
  }

  public static TreeMap<String, ItemSet> apriori_gen(TreeMap<String, ItemSet> candidates, int phase) {
    TreeMap<String, ItemSet> newCandidates = new TreeMap<>();

    ArrayList<SortedArrayList> ls = CollectionUtil.convertItemSetMapToArrayList(candidates);

    //bikin pengecualin kalau k lebih besar dari dua atau yang digabungkan cuma satu saja
    if (ls.get(0).size() == 1) {
      for (int i = 0; i < ls.size(); i++) {
        for (int j = i + 1; j < ls.size(); j++) {
          SortedArrayList<String> temp = new SortedArrayList<>();
          temp.add((String) ls.get(i).get(0));
          temp.add((String) ls.get(j).get(0));
          ItemSet item = new ItemSet(temp);
          newCandidates.put(item.toString(), item);
        }
      }
    } else {
      for (int i = 0; i < ls.size(); i++) {
        for (int j = i + 1; j < ls.size(); j++) {
          SortedArrayList<String> temp = new SortedArrayList<>();
          //int count = 0;
          boolean isMatch = true;
          for (int k = 0; k <= phase - 3; k++) {
          //for (int k = 0; k <= phase - 2; k++) {

            if (ls.get(i).get(k).equals(ls.get(j).get(k))) {

              temp.add((String) ls.get(i).get(k));
              //System.out.println(temp);
              //count++;
            }else{
              isMatch=false;
```

```
                    break;
                }
            }
            // if (count == phase - 2) {
            if (isMatch) {
                //System.out.println("break");
//                    temp.add((String)
ls.get(i).get(phase - 1));
//                    temp.add((String)
ls.get(j).get(phase - 1));
                temp.add((String)
ls.get(i).get(phase-2));
                temp.add((String)
ls.get(j).get(phase-2));
                //System.out.println(temp);
                boolean state = true;
                ArrayList<String> key =
generateSubsetKey(temp, true);
                //System.out.println(temp+"
"+key);
                for (int k = 0; k < key.size(); k++) {

//System.out.println(key.get(k));
                    if
(!candidates.containsKey(key.get(k))) {
                        state = false;
                        break;
                    }
                }
                //
                key = null;
                if (state) {
                    ItemSet item = new
ItemSet(temp);
                    if
(!newCandidates.containsKey(item.toString())
) {

newCandidates.put(item.toString(), item);
                    }
                }

            }
        }
    }

    ls = null;
    return newCandidates;
  }

  private static void genSubSet(int size, int
inputIndex, SortedArrayList<String> out,
ArrayList<String> output,
SortedArrayList<String> items) {
    if (size == 0) {
      StringBuilder sb = new StringBuilder();
      for (int i = 0; i < out.size(); i++) {
        sb.append(out.get(i));
      }
      output.add(sb.toString());
      //
      sb = null;
    } else {
      for (int i = inputIndex; i < items.size();
i++) {
        out.add(items.get(i));
        genSubSet(size - 1, i + 1, out, output,
items);
        out.remove(out.size() - 1);
      }
    }
  }

  public static ArrayList<String>
generateSubsetKey(SortedArrayList<String>
items) {
    ArrayList<String> subSet = new
ArrayList<>();
    SortedArrayList<String> out = new
SortedArrayList<>();
    AprioriUtil.genSubSet(items.size() - 1, 0,
out, subSet, items);
    //
    out = null;
    return subSet;
  }

  //modified gensubset
  public static ArrayList<String>
generateSubsetKey(SortedArrayList<String>
items, boolean modify) {
    ArrayList<String> subSet = new
ArrayList<>();

    for (int i = 0; i <= items.size() - 1; i++) {
      SortedArrayList itemstemp =
(SortedArrayList) items.clone();
      itemstemp.remove(items.size() - 1 - i);
      StringBuilder sb = new StringBuilder();
      for (int j = 0; j < itemstemp.size(); j++) {
        sb.append(itemstemp.get(j));
      }
      subSet.add(sb.toString());
    }

    return subSet;

  }

  public static long getDeepMemory(Object o)
throws IllegalAccessException {
    //return
MemoryUtil.deepMemoryUsageOf(o);
    return ObjectSizeCalculator.sizeOf(o);
  }

}
```

Kartu Konsultasi Tugas Akhir

Program Studi Teknik Informatika Fakultas Teknologi Informasi
Universitas Kristen Duta Wacana Yogyakarta
Dr. Wahidin Sudirahusada 5-25 Yogyakarta, 55224. Telp. (0274)563929

NIM : DAMAR WICAKSONO
Judul : **Implementasi Algoritma AprioriHybrid Untuk Menganalisis Association Rules Pada Penjualan Bisnis Ritel**
Dosen Pembimbing I : Budi Susanto, SKom.,M.T.

	Tanggal:	Paraf:
1		

	Tanggal:	Paraf:
2		

Gel 2

	Tanggal:	Paraf:
3	18/3/2016	

- bab 3 perlu di buat tabelah sehingga dgn materi sama yg didiskusikan.

	Tanggal:	Paraf:
4	30/3/2016	

- pengolahan database
- bab 3 di praktekkan semua dgn studi kasus penelitian

	Tanggal:	Paraf:
5	8/4/2016	

- bab 3 tolong ditambahkan rancangan pengujian, antarmuka
- cari lib. u/ big data structure

	Tanggal:	Paraf:
6	19/5/2016	

metode pengujian untuk ok!

	Tanggal:	Paraf:
7	20 juni 2016.	

- tampilan hasil rule dalam informasi bentuk manager

	Tanggal:	Paraf:
8		

Kartu Konsultasi Tugas Akhir

Program Studi Teknik Informatika Fakultas Teknologi Informasi
Universitas Kristen Duta Wacana Yogyakarta
Dr. Wahidin Sudirahusada 5-25 Yogyakarta, 55224. Telp. (0274)563929

NIM : DAMAR WICAKSONO
Judul : **Implementasi Algoritma AprioriHybrid Untuk Menganalisis Association Rules Pada Penjualan Bisnis Ritel**
Dosen Pembimbing II : Hendro Setiadi, M.Eng

	Tanggal:	Paraf:
1		

Bab I

	Tanggal:	Paraf:
2		

Bab II

	Tanggal:	Paraf:
3		

Bab III

	Tanggal:	Paraf:
4		

U.I. u/ manager
dng user.
visual diperbaiki

	Tanggal:	Paraf:
5		

	Tanggal:	Paraf:
6		

	Tanggal:	Paraf:
7		

	Tanggal:	Paraf:
8		

Program Studi Teknik Informatika
Fakultas Teknologi Informasi
Universitas Kristen Duta Wacana Yogyakarta
Dr. Wahidin Sudirahusada 5-25 Yogyakarta, 55224. Telp. (0274)563929

FORMULIR PERBAIKAN (REVISI) SKRIPSI

Strata-1 Program Studi Teknik Informatika

Yang bertanda tangan di bawah ini:

Nama	:	DAMAR WICAKSONO
N I M	:	71120125
Judul Skripsi	:	IMPLEMENTASI ALGORITMA APRIORIHYBRID UNTUK MENGANALISIS ASSOCIATION RULES PADA PENJUALAN BISNIS RITEL
Tanggal Pendadaran	:	27 Juli 2016 10:00 WIB

Telah melakukan perbaikan tugas akhir dengan lengkap.

Demikian pernyataan kami agar dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 29 Juli 2016

Dosen Pembimbing I	Dosen Pembimbing II
Budi Susanto, SKom.,M.T.	Hendro Setiadi, M.Eng

Dicetak tanggal: *29 Juli 2016 00:33 WIB*