BAB 2

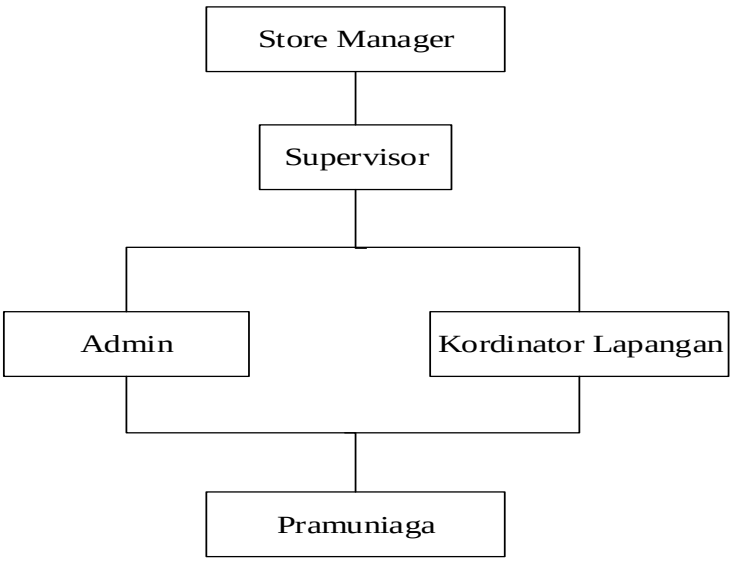
TINJAUAN PUSTAKA

2.1 Tentang Perusahaan

Summit The Boutique Outlet adalah sebuah tempat perbelanjaan yang berada di Jl. R.E. Marthadinata Bandung yang berdiri sejak tahun 2001 dan salah satu bidang usaha yang dikelola oleh PT. The Big Price Cut Group yang berada di Jl. Cipaganti Kota Bandung. Ada banyak jenis barang yang dijual di Summit antara lain fashion, aksesoris, juga oleh – oleh khas Bandung. Dari barang yang tersedia, jenis barang tersebut dibedakan menjadi beberapa kategori barang untuk membedakan jenis barang tersebut.

2.1.1 Struktur Organisasi Perusahaan

Struktur organisasi perusahaan adalah gambar suatu bagan yang menerangkan posisi atau jabatan dan struktur kerja pegawai dalam suatu perusahaan. Berikut ini adalah struktur organisasi di Summit The Boutique Outlet:



Gambar 2. 1 Struktur Organisasi di Summit The Boutique Outlet

2.1.2 Deskripsi Kerja

Setiap perusahaan tentu memiliki karyawan – karyawan yang diperkerjakan untuk membuat perusahaan maju dan keryawan – karywan memiliki tugasnya masing – masing. Berikut deskripsi kerja dari masing – masing karyawan yang bekerja di Summit the boutique outlet :

1. *Store Manager*

   Store Maneger bertugas untuk mengawasi kerja para karyawan, pengambil keputusan dan penanggung jawab semua aktifitas yang berada di Summit.

2. *Supervis*or

   Supervisor bertugas untuk Mengawasi semua aktifitas dan membatu aktifitas di Summit.

3. Admin

   Admin bertugas untuk mengelola semua data yang ada di Summit seperti data penjualan, data gudang, data barang dan lain – lain.

4. Kordinator Lapangan

   Mengevaluasi absensi karyawan kepada *Store manager.*

5. Pramuniaga

   Bertugas untuk melayani customer yang akan membeli barang di Summit, merapihkan barang, juga menyimpan letak barang untuk dipajang.

2.2 Landasan Teori

Landasan teori adalah membahas tentang materi atau teori apa saja yang digunakan sebagai bahan acuan dalam membuat tugas akhir ini. Landasan teori yang digunakan merupakan hasil studi literature baik dari buku atau situs internet.

2.2.1 Data

Data adalah sesuatu yang belum mempunyai arti bagi penerimanya dan masih memerlukan adanya suatu pengolahan. Data bisa berujut suatu keadaan, gambar, suara, huruf, angka, matematika, bahasa ataupun simbol-simbol lainnya

yang bisa kita gunakan sebagai bahan untuk melihat lingkungan, obyek, kejadian ataupun suatu konsep [4]. Dalam pendekatan basis data tidak hanya berisi basis data itu sendiri tetapi juga termasuk definisi atau deskripsi dari data yang disimpan. Definisi data disimpan dalam sistem katalog, yang berisi informasi tentang struktur tiap berkas, tipe dan format penyimpanan tiap *item* data, dan berbagai konstrin dari data. Semua informasi yang disimpan dalam katalog ini biasa disebut *meta-data* [5].

2.2.2 Basis Data

Basis data adalah mekanisme yang digunakan untuk menyimpan informasi atau data. Informasi adalah sesuatu yang kita gunakan sehari-hari untuk berbagai alasan. Dengan basis data, pengguna dapat menyimpan data secara terorganisasi. Setelah data disimpan, informasi harus mudah diambil. Kriteria dapat digunakan untuk mengambil informasi. Cara data disimpan dalam basis data menentukan seberapa mudah mencari informasi berdasarkan banyak kriteria. Data pun harus mudah ditambahkan ke dalam basis data, dimodifikasi, dan dihapus [6].

basis data sendiri dapat didefinisikan dalam sejumlah sudut pandang seperti:

1. Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.
2. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (*redudansi*) yang tidak perlu, untuk memenuhi berbagai kebutuhan.
3. Kumpulan *file*/ tabel/ arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronis

2.2.2.1 Operasi Dasar Basis Data

Didalam sebuah disk, basis data dapat diciptakan dan dapat pula ditiadakan. Didalam sebuah disk, kita dapat pula menempatkan beberapa (lebih dari satu) basisdata. Sementara dalam sebuah basis data, kita dapat menempatkan satu atau lebih *file*/tabel. Pada *file*/tabel inilah sesungguhnya data disimpan/ditempatkan. Setiap basis data umumnya dibuat untuk mewakili sebuah semesta data yang spesifik. Misalnya, ada basis data kepegawaian, basis data akademik, basis data inventori (pergudangan), dan sebagainya. Sementara dalam basis data akademik, misalnya, kita dapat menempatkan *file* mahasiswa, *file* mata_kuliah, *file* dosen, *file* jadwal, *file* kehadiran, *file* nilai, dan seterusnya. Karena itu, operasi-operasi dasar yang dapat kita lakukan berkenaan dengan basis data dapat meliputi [6]:

1. Pembuatan basis data baru (*create database*), yang identik dengan pembuatan lemari arsip yang baru.
2. Penghapusan basis data (*drop database*), yang identik dengan perusakan lemari arsip (sekaligus beserta isinya, jika ada).
3. Pembuatan *file*/tabel dari suatu basis data (*create table*), yang identik dengan penambahan map arsip baru ke sebuah lemari sarsip yang telah ada.
4. Penghapusan *file*/tabel dari suatu basis data (*drop table*), yang identik dengan perusakan map arsip lama yang ada disebuah lemari arsip.
5. Penambahan/pengisian data baru ke sebuah *file*/tabel disebuah basis data (*insert*), yang identik dengan penambahan ke lemari arsip ke sebuah map arsip.
6. Pengambilan data dari sebuah *file*/tabel (*retrieve/search*) yang identik dengan pencarian lembaran arsip dari sebuah map arsip.
7. Pengubahan data dari sebuah *file*/tabel (*update*), yang identik dengan perbaikanisi lembaran arsip yang ada di sebuah map arsip.
8. Penghapusan data dari sebuah *file*/tabel (*delete*), yang identik dengan penghapusan sebuah lembaran arsip yang ada di sebuah map arsip.

Operasi yang berkenaan dengan pembuatan objek (basis data dan tabel) merupakan operasi awal yang hanya dilakukan sekali dan berlaku seterusnya. Sedang operasi-operasi yang berkaitan dengan isi tabel (data) merupakan operasi rutin yang akan berlangsung berulang-ulang dan karena itu operasi-operasi inilah yang lebih tepat mewakili aktivitas pengelolaan (*management*) dan pengolahan (*processing*) data dalam basis data [6].

2.2.2.2 Objektif Basis Data

Telah disebutkan diawal bahwa tujuan awal dan utama dalam pengelolaan data dalam sebuah basis data adalah agar kita dapat memperoleh/menemukan kembali data (yang kita cari) dengan mudah dan cepat. Di samping itu, pemanfaatan basis data untuk pengelolaan data, juga memiliki tujuan-tujuan lain [6]. Secara lebih lengkap, pemanfaatan basis data dilakukan untuk memenuhi sejumlah tujuan (objektif) seperti buku ini [6]:

1. Kecepatan dan Kemudahan (*Speed*)

   Pemanfaatan basis data memungkinkan kita untuk dapat menyimpan data atau melakukan perubahan/manipulasi terhadap data atau menampilkan kembali data tersebut dengan lebih cepat dan mudah, daripada jika kita menyimpan data secara manual (*non elektronis*) atau secara *elektronis* (tetapi tidak dalam bentuk penerapan basis data, misalnya dalam bentuk *spread sheet* atau dokumen teks biasa).

2. Efisiensi Ruang Penyimpanan (*Space*)

   Karena keterkaitan yang erat antar kelompok data dalam sebuah basis data, maka redundansi (pengulangan) data pasti akan selalu ada. Banyaknya redundansi ini tentu akan memperbesar ruang penyimpanan (baik di memori utama maupun memori sekunder) yang harus disediakan. Dengan basis data, efisiensi/optimalisasi penggunaan ruang penyimpanan dapat dilakukan, karena kita dapat melakukan penekanan jumlah redundansi data, baik dengan menerapkan sejumlah pengkodean atau dengan membuat relasi-relasi (dalam bentuk *file*) antar kelompok data yang saling berhubungan.

3. Keakuratan (*Accuracy*)

   Pemanfaatan pengkodean atau pembentukan relasi antar data bersama dengan penerapan aturan/batasan (*constraint*) tipe data, domain data, keunikan data, dan sebagainya, yang seara ketat dapat diterapkan dalam sebuah basis data, sangat berguna untuk menekan ketidakakuratan pemasukan/penyimpanan data.

4. Ketersediaan (*Availability*)

   Pertumbuhan data (baik dari sisi jumlah maupun jenisnya) sejalan dengan waktu akan semakin membutuhkan ruang penyimpanan yang besar. Padahal tidak semua data itu selalu kita gunakan/butuhkan. Karena itu kita dapat memilah adanya data utama/master/referensi, data transaksi, data histori hingga data kadaluarsa. Data yang sudah jarang atau bahkan tidak pernah lagi kita gunakan, dapat kita atur untuk dilepaskan dari sistem basis data yang sedang aktif (menjadi *off-line*) baik dengan cara penghapusan atau dengan memindahkannya ke media penyimpanan *off-line* (seperti *removable disk* atau *tape*). Di sisi lain, karena kepentingan pemakaian data, sebuah basis data dapat memiliki data yang disebar di banyak lokasi geografis. Data nasabah sebuah bank, misalnya, dipisah-pisah dan disimpan dilokasi yang sesuai dengan keberadaan nasabah. Dengan pemanfaatan teknologi jaringan komputer, data yang berada di suatu lokasi/cabang, dapat juga diakses (menjadi tersedia/*available*) bagi lokasi/cabang lain.

5. Kelengkapan (*Completenes*)

   Lengkap/tidaknya data yang kita kelola dalam sebuah basis data bersifat relatif (baik terhadap kebutuhan pemakai maupun terhadap waktu). Bila seorang pemakai sudah menganggap bahwa data yang dipelihara sudah lengkap, maka pemakai yang lain belum tentu berpendapat sama. Atau, yang sekarang dianggap sudah lengkap, belum tentu di masa yang akan datang juga demikian. Dalam sebuah basis data, di samping data kita juga harus menyimpan struktur (baik yang mendefinisikan objek-objek dalam basis data maupun definisi

dari tiap objek, seperti struktur *file*/tabel atau indeks). Untuk mengakomodasi kebutuhan kelengkapan data yang semakin berkembang, maka kita tidak hanya dapat menambah *record-record* data, tetapi juga dapat melakukan perubahan struktur dalam basis data, baik dalam bentuk penambahan objek baru (tabel) atau dengan penambahan *field-field* baru pada suatu tabel.

6. Keamanan (*Security*)

   Memang ada sejumlah (aplikasi) pengelola basis data yang tidak menerapkan aspek keamanan dalam penggunaan basis data. Tetapi untuk sistem yang besar dan serius, aspek keamanan juga dapat diterapkan dengan ketat. Dengan begitu kita dapat menentukan siapa-siapa (pemakai) yang boleh menggunakan basis data beserta objek-objek di dalamnya dan menentukan jenis-jenis operasi apa saja yang boleh dilakukannya.

7. Kebersamaan Pemakaian (*Sharability*)

   Pemakai basis data seringkali tidak terbatas pada satu pemakai saja, atau di satu lokasi saja atau oleh satu sistem/aplikasi saja. Data pegawai dalam basis data kepegawaian, misalnya, dapat digunakan oleh banyak pemakai, dari sejumlah departemen dalam perusahaan atau oleh banyak sistem (sistem penggajian, sistem akuntansi, sistem inventori, dan sebagainya). Basis data yang dikelola oleh sistem (aplikasi) yang mendukung lingkungan multiuser, akan dapat memenuhi kebutuhan ini, tetapi tetap dengan menjaga/menghindari (karena data yang sama diubah oleh banyak pemakai pada saat yang bersamaan) atau kondisi *deadlock* (karena ada banyak pemakai yang saling menunggu untuk menggunakan data).

2.2.3 Database Management System

Kumpulan atau gabungan *database* dengan perangkat lunak aplikasi yang berbasis *database* tersebut dinamakan *Database Management System* (DBMS). DBMS merupakan koleksi terpadu dari *database* dan program–program komputer (utilitas) yang digunakan untuk mengakses dan memelihara *database*. Program-program tersebut menyediakan berbagai fasilitas operasi untuk memasukan, melacak, dan memodifikasi data kedalam *database*, mendefinisikan data baru, serta mengolah data menjadi informasi yang dibutuhkan (DBMS = *Database* + Program Utilitas) [4]. Perangkat lunak yang termasuk DBMS seperti *dBase*, *FoxBase*, *Rbase*, *Microsoft-Access* (sering juga disingkat *Ms-Access*) dan *Borland Pradox* (untuk DBMS yang sederhana) atau *Borland-Interbase*, *MS-Sql*, *Sever*, *Oracle Database*, *IBM*, *DB2*, *Informix*, *Sybase*, *MySql*, *PostgreSQL* (untuk DBMS yang lebih kompleks dan lengkap) [6].

2.2.4 Data Mining

Data mining, sering juga disebut *knowledge discovery in database* (KDD), adalah kegiatan yang meliputi pengumpulan, pemakaian data historis untuk menemukan keteraturan, pola atau hubungan dalam set data berukuran besar. Keluaran dari *data mining* ini bisa dipakai untuk memperbaiki pengambilan keputusan di masa depan. Sehingga istilah *pattern recognition* sekarang jarang digunakan karena ia termasuk bagian dari *data mining* [7]. *Data Mining* adalah suatu istilah yang digunakan untuk menguraikan penemuan pengetahuan didalam database. *Data Mining* adalah proses yang menggunakan teknik statistik, matematika, kecerdasan buatan, dan *machine learning* untuk mengektrasi dan mengidentifikasi informasi yang bermanfaat dan pengetahuan yang terkait dari berbagai database besar (Turban, dkk. 2005) [8].

2.2.4.1 Tahapan *Data Mining*

Berikut adalah tahapan dalam *data mining* :

1. *Data Selection*

Seleksi data dari sekumpulan data operasional perlu dilakukan sebelum tahap penggalian informasi dalam KDD dimulai. Data hasil seleksi yang akan digunakan untuk proses *data mining*, disimpan dalam suatu berkas, terpisah dari baris data operasional.

2. *Pre-Processing/cleaning*

   Sebelum proses *data mining* dapat dilakukan, perlu dilakukan dahulu proses *cleaning* pada data yang menjadi fokus KDD, *Knowledge Discovery Databases*. Proses *cleaning* mencakup antara lain membuang duplikasi data, memeriksa data yang inkonsisten, dan memeperbaiki kesalahan pada data. Juga dilakukan proses *enrichment*, yaitu proses "meperkaya" data yang sudah ada dengan data atau informasi lain yang relevan dan diperlukan untuk KDD. *Pre-processing* data adalah hal yang harus dilakukan dalam proses *data mining* karena tidak semua data atau atribut data dalam data digunakan dalam proses data *mining*. Proses ini dilakukan agar data yang akan digunakan sesuai dengan kebutuhan. Adapun langkah-langkah *pre-processing* adalah sebagai berikut :

   a. Pemilihan atribut (*atribut selection*)

      Pemilihan atribut adalah proses pemilihan atribut data yang akan digunakan sehingga dapat kita olah untuk proses *mining.*

   b. Pembersihan data (*data cleaning*)

      Proses menghilangkan *noise* dan menghilangkan data yang tidak *relevan*.

3. *Transformation*

   Coding adalah proses transformasi pada ata yang telah dipilih, sehingga data tersebut sesuai untuk proses *data mining.* Proses *coding* dalam KDD merupakan proses kreatif dan sangat tergantung pada jenis atau pola informasi yang akan dicari dalam basis data.

4. *Data mining*

   Data mining adalah proses mencari pola atau informasi menarik dalam data terpilih dengan menggunakan teknik atau metode tertentu.

Teknik, metode, atau algortima yang tepat sangat bergantung pada tujuan dan proses KDD secara keseluruhan.

5. *Interpretation/evaluate on*

   Pola informasi yang dihasilkan dari proses *data mining* perlu ditampilkan dalam bentuk yang mudah dimengerti oleh pihak yang berkepentingan. Tahap ini merupakan bagian dari proses KDD yang disebut *interpretation*. Tahap ini mencakup pemeriksaan apakah pola atau informasi yang disebut bertentangan dengan fakta atau hipotesis yang ada sebelumnya.

2.2.4.2 Metode *Data Mining*

Secara garis besar, Han dalam bukunya menjelaskan bahwa metode *data mining* dapat dilihat dari dua sudut pandang pendekatan yang berbeda, yaitu pendekatan deskriptif dan pendekatan prediktif [9].

Pendekatan deskriptif adalah pendekatan dengan cara mendeskripsikan data inputan. Metode yang termasuk kedalam pendekatan ini adalah :

1. Metode deskripsi konsep/kelas, yaitu data dapat diasosiasikan dengan kelas atau konsep. Ada tiga macam pendeskripsian yaitu (1) karakteristik data, dengan membuat *summary* karakter umum atau fitur data suatu kelas target, (2) diskriminasi data, dengan membandingkan *class* target dengan satu atau sekelompok kelas pembanding, (3) gabungkan antara karakterisasi dan diskriminasi.
2. Metode *association rule*, yaitu menemukan aturan asosiatif atau pola kombinasi dari suatu *item* yang sering terjadi dalam sebuah data.

Pendekatan kedua adalah pendekatan prediktif, yaitu pendekatan yang dapat digunakan untuk memprediksi, dengan hasil berupa kelas atau *cluster*. Metode yang termasuk dalam pendekatan ini adalah :

1. Metode klasifikasi dan prediksi, yaitu metode analisis data yang digunakan untuk membentuk model yang mendeskripsikan kelas data yang penting, atau model yang memprediksikan trend data. Klasifikasi digunakan untuk memprediksi kelas data yang bersifat kategorial,

sedangkan prediksi untuk memodelkan fungsi yang mempunyai nilai kontinu.

2. Metode *clustering*, mengelompokan data untuk membentuk kelas-kelas baru atau sering disebut *cluster*. Metode *clustering* bertujuan untuk memaksimalkan persamaan dalam satu *cluster* dan meminimalkan perbedaan antar *cluster*.

2.2.5 Association Rule

Association Rule atau Aturan Asosiasi adalah teknik *Data Mining* untuk menemukan aturan asosiatif atau pola kombinasi dari suatu item. Bila kita mengambil contoh aturan asosiatif dalam suatu transaksi pembelian barang di suatu minimarket adalah kita dapat mengetahui berapa besar kemungkinan seorang konsumen membeli suatu item bersamaan dengan item lainnya (membeli roti bersama dengan selai). Karena awalnya berasal dari studi tentang database transaksi pelanggan untuk menentukan kebiasaan suatu produk dibeli bersama apa, maka association rule sering juga dinamakan *market basket* analysis [7].

Association Rule adalah bentuk jika "kejadian sebelumnya" kemudian "konsekuensinya" (*If antecedent, then consequent*), yang diikuti dengan perhitungan aturan *support* dan *confidence*. Bentuk umum dari *association rule* adalah *Antecedent -> Consequent*. Bila kita ambil contoh dalam sebuah transaksi pembelian barang di sebuah minimarket didapat bentuk *association rule* roti -> selai, artinya bahwa pelanggan yang membeli roti ada kemungkinan pelanggan tersebut juga akan membeli selai, dimana tidak ada batasan dalam jumlah item-item pada bagian *antecedent* ataupun *consequent* dalam sebuah rule.

Association rule memiliki dua tahap pengerjaan, yaitu [10]:

1. Mencari kombinasi yang paling sering terjadi dari suatu *itemset*.
2. Mendefinisikan *Condition* dan *Result* (*Generate Rule*)

Dalam menentukan suatu *association rule*, terdapat suatu *interestingness measure* (ukuran kepercayaan) yang didapat dari hasil pengolahan data dengan perhitungan tertentu. Umumnya ada dua ukuran, yaitu :

1. *Support* : suatu ukuran yang menunjukkan seberapa besar tingkat dominasi suatu item/itemset dari keseluruhan transaksi. Ukuran ini menentukan apakah suatu item/itemset layak untuk dicari *confidence*-nya (misal, dari keseluruhan transaksi yang ada, seberapa besar tingkat dominasi suatu item yang menunjukkan bahwa item A dan item B dibeli bersamaan).
2. *Confidence* : suatu ukuran yang menunjukkan hubungan antara 2 item secara conditional (misal, menghitung kemungkinan seberapa sering item B dibeli oleh pelanggan jika pelanggan tersebut membeli sebuah item A).

Kedua ukuran ini nantinya berguna dalam menentukan kekuatan suatu pola dengan membandingkan pola tersebut dengan nilai minimum kedua parameter tersebut yang ditentukan oleh pengguna. Bila suatu pola memenuhi kedua nilai minimum parameter yang sudah ditentukan sebelumnya, maka pola tersebut dapat disebut sebagai *interesting rule* atau *strong rule*.

2.2.5.1 Metodologi Dasar Analisis Asosiasi

Metodologi dasar *Association Rule* terbagi menjadi dua tahap, yaitu [10]:

a. Analisa pola frekuensi tinggi

   Tahap ini mencari pola item yang memenuhi syarat minimum dari nilai *support* dalam database. Menurut Larose, kita bebas menentukan nilai *minimum support* (minsup) dan *minimum confidence* (mincof) sesuai kebutuhan [11]. Sebagai contoh, bila ingin menemukan data-data yang memiliki hubungan asosiasi yang kuat, minsup dan mincof-nya bisa diberi nilai yang tinggi. Sebaliknya, bila ingin melihat banyaknya variasi data tanpa terlalu mempedulikan kuat atau tidaknya hubungan asosiasi antara item-nya, nilai minsup dan mincofnya dapat diisi rendah. Untuk rekomendasi dalam menentukan *minimum support* dapat diambil dari perhitungan rata-rata 1 jenis produk pada data yang digunakan, seperti rumus berikut :

$$\text{Support}(A) = \frac{\text{Jumlah Transaksi Yang Mengandung A}}{\text{Jumlah Total Transaksi}} \quad \text{(Persamaan 2-1)}$$

Persamaan 1 menjelaskan bahwa nilai *support* didapat dengan cara membagi jumlah transaksi yang mengandung *item* A (satu *item*) dengan jumlah total seluruh transaksi. Sedangkan untuk mencari nilai *support* dari 2 *item* menggunakan rumus berikut :

$$\text{Support}(A,B) = P(A \cap B) = \frac{\text{Jumlah Transaksi Yang Mengandung Item A dan B}}{\text{Jumlah Total Transaksi}} \quad \text{(Persamaan 2-2)}$$

Persamaan 2 menjelaskan bahwa nilai *support* 2-*itemsets* didapat dengan cara membagi jumlah transaksi yang mengandung *item* A dan *item* B (*item* pertama bersamaan dengan *item* yang lain) dengan jumlah total seluruh transaksi.

1. Pembentukan Aturan Assosiatif

Setelah semua pola frekuensi tinggi d*item*ukan, kemudian mencari aturan asosiatif yang memenuhi syarat minimum untuk *confidence* dengan menghitung *confidence* aturan assosiatif A -> B dari *support* pola frekuensi tinggi A dan B, menggunakan rumus :

$$\textit{Confidence} = (A \rightarrow B) = \frac{\text{Jumlah Transaksi Yang Mengandung A dan B}}{\text{Jumlah Transaksi Yang Mengandung A}} \quad \text{(Persamaan 2-3)}$$

Persamaan 3 menjelaskan bahwa nilai *confidence* diperoleh dengan cara membagi jumlah transaksi yang mengandung *item* A dan *item* B (*item* pertama bersamaan dengan *item* yang lain) dengan jumlah transaksi yang mengandung *item* A (*item* Pertama atau *item* yang ada disebelah kiri).

2.2.6 Algoritma CT-Pro

Algoritma ini merupakan pengembangan dari algoritma *FP-GROWTH* dengan melakukan modifikasi pada *tree* yang digunakan. Algoritma ini menggunakan struktur *Compressed FP-Tree* (*CFP-Tree*) dimana informasi dari sebuah *FP-Tree* diringkas dengan struktur yang lebih kecil atau ringan, sehingga

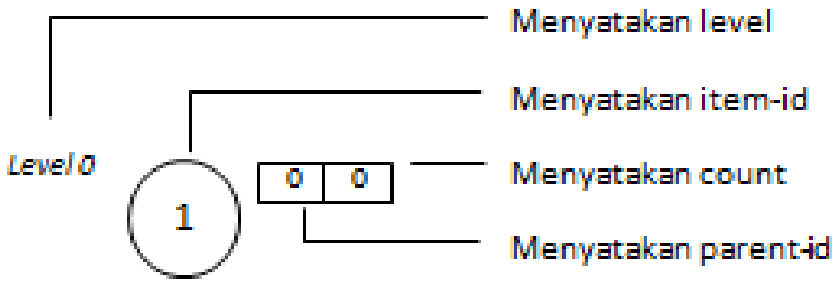
baik pembentukan *tree* maupun *frequent itemset mining* yang dilakukan menjadi lebih cepat[12].

Langkah-langkah algoritma *CT-PRO* adalah sebagai berikut [13]:

1. Mencari *Frequent Item*, pada tahap ini terjadi proses-proses sebagai berikut:

   Data-data yang telah dikumpulkan, diseleksi dan pilih data yang relevan (data yang lengkap).Data-data yang ada, kemudian dilakukan transformasi data. Kemudian masing masing data diseleksi berdasarkan *minimum support* yang telah ditentukan, kemudian didapat *item* yang frekuen .Masing-masing *item* dihitung frekuensi kemunculannya sehingga dihasilkan *GlobalitemTable*.

2. Membangun *CFP-Tree* , pada tahap ini terjadi proses-proses sebagai berikut:
   1. *Frequent item* yang telah didapatkan, diurutkan berdasarkan *Global Item* tabel yang ada secara menurun (diurutkan mulai dari *item* berfrekuensi terbesar hingga terkecil).
   2. Dengan *frequent item* yang telah terurut ini dibentuk *Global CFP-Tree* , aturan pembentukan *Global CFP-Tree* sebagai berikut:
      a. *CFP-Tree* terdiri dari *tree* yang memiliki *root* yang mewakili indeks dari *item* dengan tingkat kemunculan tertinggi dan kumpulan *subtree* sebagai anak dari *root*.
      b. Jika I = {i1,i2, …, ik} adalah kumpulan dari *frequent item* dalam transaksi, *item* dalam transaksi akan dimasukkan ke dalam *CFP-Tree* dimulai dari *root subtree* yang merupakan i1 dalam *item* tabel.
      c. *Root* dari *CFP-Tree* merupakan *level*-0 dari *tree*.
      d. Setiap *node* dalam *CFP-Tree* memiliki empat *field* utama yakni *item-id*, *parent-id*, *count* yang merupakan jumlah *item* pada *node* tersebut, dan *level* yang menunjukkan struktur data *tree* pada *node* tersebut dimulai dari *item* yang terdapat pada *item* tabel dengan *level* yang terdapat pada *CFP-Tree.*



Gambar 2. 2 Struktur CFP-Tree

Dalam pencarian *item frequent* dan membangun CFP *tree* digunakan Algoritma sebagai berikut :

```
input  Database D, Support Threshold σ output CFP-Tree
1 begin
2  // Step 1: Identify frequent items
3  for each transaction t ∈ D
4     for each item i ∈ t
5        if i ∈ ItemTable
6           Increment count of i
7        else
8           Insert i into GlobalItemTable with count = 1
9        end if
10     end for
11  end for
12  Sort GlobalItemTable in      frequency descending order
13  Assign an index for each frequent item in the    GlobalItemTable
14  // Step 2: Construct CFP-Tree
15  Construct the left most branch of the tree
16  for each transaction t ∈ D
17   Initialize mappedTrans
18     for each frequent item i ∈ t
19      mappedTrans = mappedTrans ∪ GetIndex(i)
20     end for
21   Sort mappedTrans in ascending order of item ids
22   InsertToCFPTree(mappedTrans)
23  end for
24 end
25 Procedure InsertToCFPTree(mappedTrans)
26  firstItem := mappedTrans[1]
27  currNode := root of subtree pointed by      ItemTable[firstItem]
28  for each subsequent item i ∈ mappedTrans
29   if currNode has child represent i
30    Increment count[firstItem-1] of the child node
31   else
32    Create child node and set its      count[firstItem-1] to 1
 33    Link the node to its respective node-link
34   end if
35  end for
36 end
```

Gambar 2. 3 Algorithm 1 CT-PRO Algorithm: Step 1 and Step 2[13]

3. Melakukan penggalian *frequent patterns*, pada tahap ini terjadi proses-proses sebagai berikut :
    a. Pada tahap *mining* ini, algoritma *CT-Pro* bekerja dengan melakukan *bottom-up mining* sehingga *Global Item* tabel diurutkan mulai dari *item* berfrekuensi terkecil hingga terbesar.
    b. Untuk setiap *item* yang terdaftar pada *Global Item* tabel yang telah diurutkan, dilakukan pencarian *node* yang berkaitan dengan *item* tersebut pada *Global CFP-Tree* . Dari semua *node* yang d*item*ukan untuk setiap *item* inilah yang disebut dengan *Local Frequent item* dan digunakan untuk membuat *Local Item* Tabel.
    c. Pada pembuatan *Local item* tabel ini juga dilakukan berdasarkan jumlah minimum *support* yang telah ditentukan.
    d. Setelah itu, dibuat *Local CFP-Tree* berdasarkan *Local Item* Tabel yang terbentuk. Aturan pembentukan *Local CFP-Tree* sama dengan pembentukan *Global CFP-Tree* , yang membedakan adalah pada *Global CFP-Tree* yang digunakan dalam pembentukan *tree*-nya adalah *Global Item* tabel yang terbentuk dari *Global Item* tabel data sedangkan pada *Local CFP-Tree* yang digunakan dalam pembentukan *tree*-nya adalah *Local Item* tabel yang terbentuk dari *Local Frequent item.*
    e. Dari *Local CFP-Tree* dibentuk *frequent pattern* sesuai dengan *item* yang di-*mining*.
    f. Dari *frequent pattern* dihitung masing-masing *item* yang memenuhi dihitung *confidencenya*. Apabila memenuhi minimum *confidence* maka masing-masing *item* yang bersangkutan dijadikan sebagai *knowledge*.

Berikut merupakan algoritma dalam penggalian *frequent Patterns:*

```
input  CFP-Tree output Frequent Itemsets FP
1 Procedure Mining
2    for each frequent item i ∈ GlobalItemTable
     from the least to the most frequent
3  Initialize LocalFrequentPatternTree  with i as
the root
4  ConstructLocalItemTable(x)
5  for each frequent item j ∈ LocalItemTable
6   Attach i as a child of x
7 end for
8  ConstructLocalCFPTree(x)
9  RecMine(x)
10  Traverse the LocalFrequentPatternTree
     to print the frequent itemsets
11  end for
12 end
 Procedure ConstructLocalItemTable(i)
14  for each occurrence of node i in the CFP-Tree
15   for each item j in the path to the root
16        if j ∈ LocalItemTable
17          Increment count of j
18        else
19          Insert j into LocalItemTable with count =
1
20        end if
21   end for
22  end for
23 end
```

```
Procedure ConstructLocalCFPTree(i)
25  for each occurrence of node i in the CFP-Tree
 26  Initialize mappedTrans
27   for each frequent item j ∈ LocalItemTable
in the path to the root
28      mappedTrans = mappedTrans ∪ GetIndex(j)
 29     end for
30  Sort mappedTrans in ascending order of item
ids
31  InsertToCFPTree(mappedTrans)
32  end for
33 end
34 Procedure RecMine(x)
35  for each child i of x
36  Set all counts in LocalItemTable to 0
37  for each occurrence of node i in     the
LocalCFPTree 38    for each item j in the path to
the root
39          Increment count of j in LocalCFPTree
40   end for
 41  end for
42  for each frequent item k ∈ LocalItemTable
43   Attach k as a child of i
44  end for
45  RecMine(i)
46  end for
47 end
```

Gambar 2. 4 Algoritma 1 CT-PRO Penggalian *Frequent Patterns*[13]

Berikut ini merupakan contoh kasus penerapan metode *association rule* menggunakan CFP-*Tree:*

Misalkan terdapat *itemset* seperti gambar di bawah dan ingin kita dapatkan *frequent Pattern*-nya. Min_support untuk *itemset* di bawah ini adalah 2 (20%).

Tabel 2. 1 Sampel Transaksi

tid	*item*
1	C D E F G H
2	A C D E M
3	A B D E G K
4	A C D H
5	A C D J

Dari *itemset* di atas kita hitung jumlah kemunculan masing-masing *item* dan buang yang tidak memenuhi *min support* dan buat tabel baru dengan nama *header Table* secara menurun sehingga didapatkan *Frequent item.*berikut Hasil pencarian *item* frequent *item.*

Tabel 2. 2 Item yang Frequent

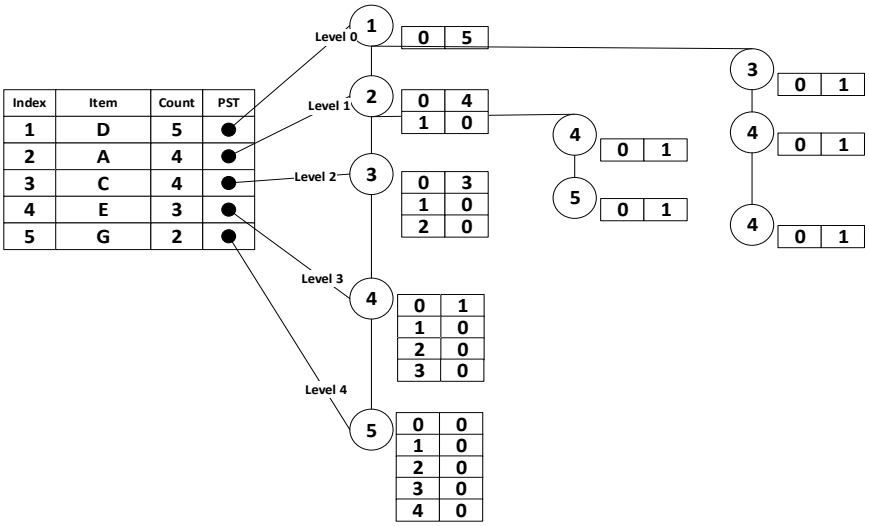
tid	*item*
1	C D E F G H
2	A C D E
3	A B D E G
4	A C D H
5	A C D

Didapat dari transaksi di atas *item* A(4),C(4),D(5),E(3),G(2) Setelah kita dapatkan *item* dengan jumlah *count* setiap *item* lakukan *mapping* pada transaksi dengan *Index* pertama yang memiliki *count* terbesar.Berikut merupakan Hasil *Mapping Node* data :

Tabel 2. 3 Mapping Transaksi

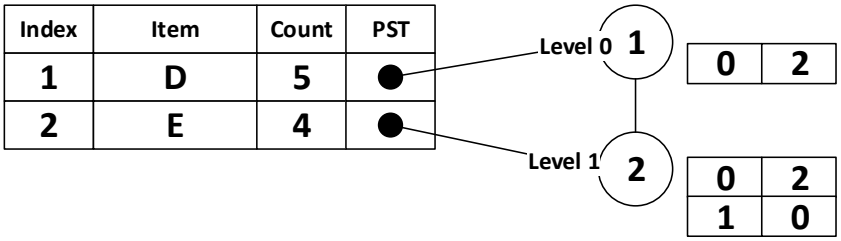
tid	*item*
1	1 2 4 5
2	1 2 3 4
3	1 3 4 5
4	1 2 3
5	1 2 3

Setelah dilakukan *mapping* data selanjutnya adalah membangun *Global CFP-Tree* dengan menggunakan data transaksi pada *Table* 2.1 berikut hasil dari pembangunan *Global* CFP – *Tree*:



Gambar 2. 5 Hasil Global CFP-Tree

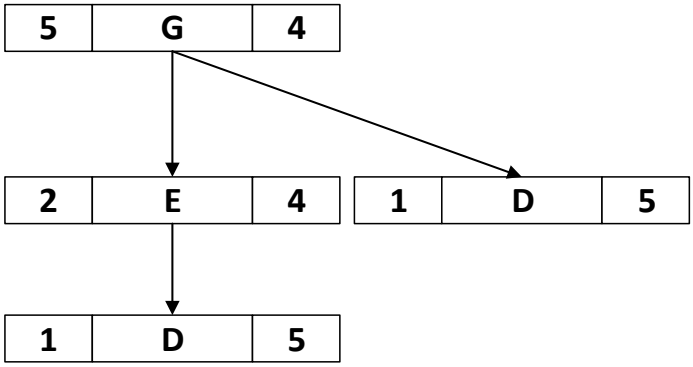
Setelah membuat *Global CFP-Tree* selanjutnya adalah membuat mencari *frequent Pattern* dari *Global*- CFP - *tree* dengan memilih *item* dengan *support count* terkecil atau memilih *node* terkahir sebagai contoh pada *Global* CFP – *Tree* pada gambar 2.3 *Index node* 5 merupakan *node* terakhir kemudian lakukan penulusuran pada *node* tersebut dengan menghitung setiap *count* dari *item*.Dari hasil di atas *item* yang memenuhi *minimum support count* adalah *item* D(2) dan E(2) selanjutnya berikan *Index* ide baru pada *local item Table*, lalu lakukan pembangunan *Local CFP-Tree* sebagai berikut :



Gambar 2. 6 *Local CFP-Tree index* 5

Setelah terbentuk *Local CFP-Tree* lakukan penelusuran kembali atau di sebut denga traversing yaitu penelusuran dari *Path sampai root* jika menghasilkan nilai

count yang telah di tentukan berati *item* tersebut memnuhi *frequent* berikut hasil projek id 5:



Gambar 2. 7 *Local CFP-Tree projection* 5

Dari *Local CFP-Tree projection* maka didapatkan *item frequent* untuk *Index* 5 dengan *item* G adalah sebagai berikut: (5,2,1:6)

2.2.7 Unifief Modelling Language (UML)

UML singkatan dari *Unified Modeling Languages* yang berarti bahasa pemodelan standar. Ketika kita membuat model menggunakan konsep UML ada aturan-aturan yang harus diikuti. Bagaimana elemen pada model-model yang kita buat berhubungan satu dengan yang lainnya harus mengikuti standar yang ada. UML bukan hanya sekedar diagram tetapi juga menceritakan konteksnya[14]. Berikut adalah beberapa model yang digunakan dalam perancangan aplikasi *Data Mining* untuk menentuan rekomendasi layout barang di Summit the Boutique Outlet untuk menggambarkan sistem dalam UML:

1. Diagram *Use Case*
2. Diagram *Activity*
3. Diagram *Sequence*
4. Diagram *Class*

2.2.7.1 *Use Case Diagram*

Diagram *Use Case* menggambarkan apa saja aktifitas yang dilakukan oleh suatu sistem. *Use Case* menggambarkan fungsi tertentu dalam suatu sistem berupa

komponen, kejadian atau kelas[14]. Komponen pembentuk *Use Case* Diagram adalah sebagai berikut :

1. Actor

   Pada dasarnya *actor* bukanlah bagian dari *use case* diagram, namun untuk dapat terciptanya suatu *use case* diagram diperlukan beberapa *actor*. *Actor* tersebut mempresentasikan seseorang atau sesuatu (seperti perangkat, sistem lain) yang berinteraksi dengan sistem. Sebuah *actor* mungkin hanya memberikan informasi inputan pada sistem, hanya menerima informasi dari sistem atau keduanya menerima, dan memberi informasi pada sistem. *Actor* hanya berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case*. *Actor* digambarkan dengan *stick man*. *Actor* dapat digambarkan secara secara umum atau spesifik, dimana untuk membedakannya kita dapat menggunakan *relationship.*

2. *Use Case*

   Use case adalah gambaran fungsionalitas dari suatu sistem, sehingga customer atau pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan dibangun.

   Catatan : *Use case* diagram adalah penggambaran sistem dari sudut pandang pengguna sistem tersebut (*user*), sehingga pembuatan *use case* lebih dititikberatkan pada fungsionalitas yang ada pada sistem, bukan berdasarkan alur atau urutan kejadian.

   Cara menentukan *Use Case* dalam suatu sistem:

   1. Pola perilaku perangkat lunak aplikasi.
   2. Gambaran tugas dari sebuah *actor*.
   3. Sistem atau “benda” yang memberikan sesuatu yang bernilai kepada *actor*.
   4. Apa yang dikerjakan oleh suatu perangkat lunak (*bukan bagaimana cara mengerjakannya).

   Ada beberapa relasi yang terdapat pada *use case* diagram:

   1. *Association*, menghubungkan link antar elemen.

2. *Generalization*, disebut juga *inheritance* (pewarisan), sebuah elemen dapat merupakan spesialisasi dari elemen lainnya.
3. *Dependency*, sebuah elemen bergantung dalam beberapa cara ke elemen lainnya.
4. A*ggregation*, bentuk *association* dimana sebuah elemen berisi elemen lainnya.

Tipe relasi/ *stereotype* yang mungkin terjadi pada *use case* diagram:

1. <<*include*>> , yaitu kelakuan yang harus terpenuhi agar sebuah *event* dapat terjadi, dimana pada kondisi ini sebuah *use case* adalah bagian dari *use case* lainnya.
2. <<*extends*>>, kelakuan yang hanya berjalan di bawah kondisi tertentu seperti menggerakkan alarm.
3. <<*communicates*>>, mungkin ditambahkan untuk asosiasi yang menunjukkan asosiasinya adalah *communicates association*. Ini merupakan pilihan selama asosiasi hanya tipe *relationship* yang dibolehkan antara *actor* dan *use case*.

2.2.7.2 *Activity Diagram*

Activity diagram memiliki pengertian yaitu lebih fokus kepada menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses. Dipakai pada business modeling untuk memperlihatkan urutan aktifitas proses bisnis. Memiliki struktur diagram yang mirip *flowchart* atau data *flow* diagram pada perancangan terstruktur. Memiliki pula manfaat yaitu apabila kita membuat diagram ini terlebih dahulu dalam memodelkan sebuah proses untuk membantu memahami proses secara keseluruhan dan *activity* dibuat berdasarkan sebuah atau beberapa *use case* pada *use case* diagram[14].

2.2.7.3 *Sequence Diagram*

Diagram sekuen menggambarkan interaksi objek pada *use case* dengan mendeksripsikan waktu hidup objek dan pesan yang dikirimkan dan di terima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus

diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode sekuen juga di butuhkan untuk melihat skenario yang ada pada *use case*[14]. *Sequence* diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *even*t untuk menghasilkan output tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya.

2.2.7.4 *Class Diagram*

Diagram kelas atau *class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan di buat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

1. Atribut merupakan variabel-variabel yang di miliki oleh suatu kelas
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas

Diagram kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron. banyak berbagai kasus, perancangan kelas yang dibuat tidak sesuai dengan kelas-kelas yang dibuat pada perangkat lunak, sehingga tidaklah ada gunanya lagi sebuah perancangan karena apa yang dirancang dan hasil jadinya tidak sesuai[14].