BAB 2
TINJAUAN PUSTAKA

2.1 Profil Instansi

2.1.1 Sejarah Mini Market HD Mart

Mini Market HD Mart didirikan oleh seorang wirausahawan yang bernama Bapak Haji Darjat. Mini Market tersebut berdiri sejak tahun 2015 di Jl. Raya Cijurey Sukabumi. Tujuan pendirian usaha Mini Market HD Mart ini dilihat dari potensi masyarakat yang bersifat konsumtif, selain itu dilihat dari faktor lain belum adanya mini market dan toko ritel pada tempat tersebut, bapak Haji Darjat akhirnya mendirikan suatu Mini Market. Tidak Hanya dilihat dari masyarakat saja pendirian usaha tersebut difungsikan untuk memenuhi kebutuhan keluarga bapak Haji Darjat. Akhirnya sampai saat ini Mini Market HD Mart tersebut dapat berkembang pesat dalam kurun waktu kurang dari satu tahun lebih dengan kiat-kiat seorang wirausaha yang dimiliki oleh bapak Haji derajat yaitu percaya diri, optimis, dan tekad yang besar untuk menggapai suatu impian dalam berwirausaha. Ditambah etos kerja dan ulet dalam mengerjakan usaha yang digelutinya.

2.1.2 Logo

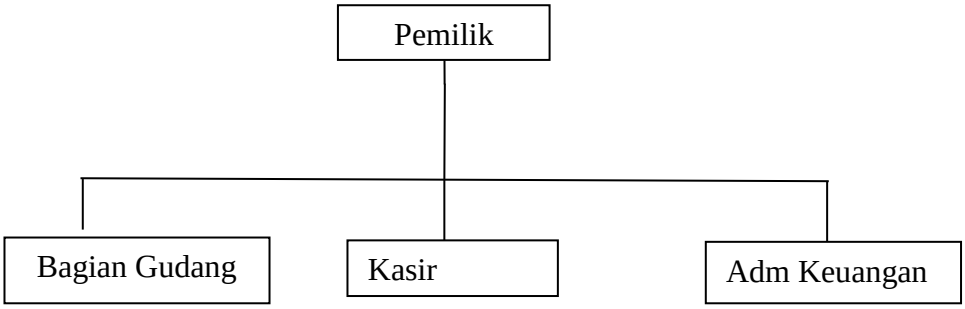
Berikut ini adalah logo dari Mini Market HD Mart yang dapat dilihat pada gambar 2.1 di bawah ini:



Gambar 2. 1 Logo HD Mart

2.1.3 Struktur Organisasi

Berikut ini adalah struktur organisasi Mini Market HD Mart yang dapat dilihat pada gambar 2.2 di bawah ini:



Gambar 2. 2 Struktur HD Mart

Job description:

1. Pemilik bertanggung jawab penuh terhadap Mini Market HD Mart, tugas dan tanggung jawabnya sebagai berikut:
   a. Membuat perencanaan, strategi dan kebijakan yang menyangkut operasi Mini Market HD Mart
   b. Menyusun anggaran kebutuhan persediaan bahan baku.
   c. Melakukan kontrol secara keseluruhan di Mini Market HD Mart.
   d. Memegang kendali atas keputusan penting yang bersifat umum berkaitan dengan keuangan.
2. Bagian Gudang memiliki tugas dan tanggung jawab sebagai berikut:
   a. Mengecek ketersediaan stok barang
   b. Membeli persediaan stok barang.
3. Kasir memiliki tugas dan tanggung jawab sebagai berikut:
   a. Melayani konsumen di kasir.
   b. Mengisi barang di tempat pajangan.
4. Bagian Adm keuangan bertugas terhadap transaksi pelanggan, tugas dan tanggung jawab sebagai berikut:
   a. Bertanggung jawab terhadap hal-hal yang menyangkut keuangan.
   b. Menghitung pemasukan dan pengeluaran setiap bulannya.

2.1.4 Visi dan Misi

Visi dari Mini Market HD Mart ini adalah menjadikan perusahaan Mini Market, Menjadi jaringan distribusi ritel terkemuka yang dimiliki oleh masyarakat luas, berorientasi kepada pemberdayaan pengusaha kecil, pemenuhan kebutuhan dan harapan konsumen, serta mampu bersaing secara *Global*. Sedangkan Misinya adalah memberi kepuasan kepada konsumen dengan berfokus pada barang dan pelayanan yang berkualitas unggul.

2.2 Landasan Teori

Landasan teori yang berkaitan dengan materi atau teori yang digunakan sebagai acuan melakukan penelitian. Landasan teori yang diuraikan merupakan hasil studi literatur, buku-buku, maupun situs internet.

2.2.1 Data

Menurut Lia Kuswayatno, data adalah kumpulan kejadian atau peristiwa yang terjadi di dunia nyata yang berupa angka-angka, huruf-huruf, simbol-simbol khusus, atau gabungan dari semuanya. Dalam data terdapat himpunan data (*data-set)* yang merupakan kumpulan dari objek dan atributnya. Atribut merupakan sifat atau karakteristik suatu objek yang biasanya dikenal sebagai *variable, field,* karakteristik atau fitur.

Salah satu himpunan data (*data-set*) adalah *record* data, yaitu data yang terdiri dari sekumpulan *record,* yang masing-masing terdiri dari satu set atribut yang tetap. Salah satu yang termasuk dalam tipe data *record* yaitu data transaksi. Data transaksi merupakan sebuah tipe khusus dari *record* data, di mana tiap *record* (transaksi) meliputi satu *set item* [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].

Menurut Fathansyah [7], 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 Data pada Basis Data dan Hubungannya

Ada 3 jenis data pada sistem *database*, yaitu [8]:

1. Data operasional dari suatu organisasi, berupa *data* yang tersimpan dalam basis data
2. Data masukan (*input data*), data dari luar sistem yang di masukan melalui peralatan *input* (*keyboard*), yang dapat merubah data operasional.
3. Data keluaran (*output data*), berupa laporan melalui peralatan *output* sebagai hasil dari dalam sistem yang mengakses data operasional.

2.2.2.2 Keuntungan dan Kerugian Pemakaian Sistem *Database*

Keuntungan [8]:

1. Terpeliharanya keselarasan *data*
2. *Data* dapat dipakai secara bersama-sama.
3. Memudahkan penerapan standarisasi dan batas-batas pengamanan.
4. Terpeliharanya keseimbangan atas perbedaan kebutuhan data dari setiap aplikasi.
5. Program /data *independent*.

Kerugian:

1. Mahal dalam implementasinya.
2. Rumit
3. Penanganan proses *recovery backup* sulit.
4. Kerusakan pada sistem basis data dapat mempengaruhi.

2.2.3 *Database Mangement 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 memasukkan, melacak, dan memodifikasi data ke dalam *database*, mendefinisikan data baru, serta mengolah data menjadi informasi yang dibutuhkan (*DBMS* = *Database* + Program Utilitas) [9].

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* sudah tidak digunakan lagi [10].

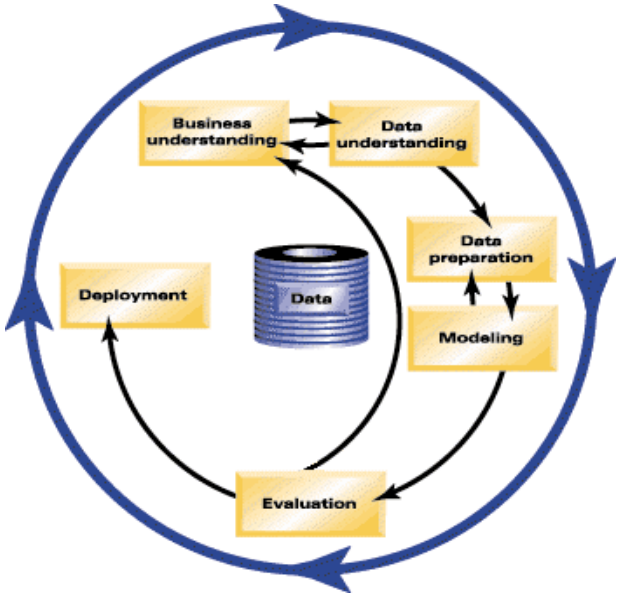
Secara umum, definisi *Data Mining* dapat diartikan sebagai berikut [11]:

1. Proses penemuan pola yang menarik dari data yang tersimpan dalam jumlah besar.
2. Ekstraksi dari suatu informasi yang berguna atau menarik (non-trivial, implisit, sebelumnya belum diketahui potensi kegunaannya) pola atau pengetahuan dari data yang di simpan dalam jumlah besar.
3. Eksplorasi dari analisa secara otomatis atau semi otomatis terhadap data dalam jumlah besar untuk mencari pola dan aturan yang berarti.

2.2.4.1 *CRISP-DM*

Cross-Indutry Standard Prosess for Data Mining (*CRISP-DM*) yang dikembangkan tahun 1996 oleh analis dari beberapa industri seperti Daimler Chrysler, SPSS dan NCR. *CRISP-DM* menyediakan standar proses *Data Mining* sebagai strategi pemecahan masalah secara umum dari bisnis atau unit penelitian. Dalam *CRISP-DM*, sebuah proyek *Data Mining* memiliki siklus hidup yang terbagi dalam enam fase. Keseluruhan fase berurutan yang ada tersebut bersifat adaptif. Fase berikutnya dalam urutan bergantung kepada keluaran dari fase sebelumnya. Hubungan antar fase digambarkan dengan panah. Sebagai contoh, jika proses

berada pada fase *modelling*. Berdasar pada perilaku dan karakteristik model, proses mungkin harus kembali kepada fase *data preparation* untuk perbaikan lebih lanjut terhadap data atau berpindah maju kepada fase *evaluation*. Enam fase yang ada dalam metode *CRISP-DM* tersebut dapat dilihat pada Gambar 2.3:



Gambar 2. 3 CRISP – DM

Fase-fase dari *CRISP-DM* [12]:

1. *Business understanding*
    a. Penentuan tujuan objek dan kebutuhan secara detail dalam lingkup bisnis atau unit penelitian secara keseluruhan.
    b. Menerjemahkan tujuan dan batasan menjadi formula dari permasalahan *Data Mining*.
2. *Data understanding*
    a. . Mengumpulkan data.
    b. Menggunakan analisis penyelidikan data untuk mengenali lebih lanjut data dan pencarian pengetahuan awal.
    c. Mengevaluasi kualitas data.
    d. Jika diinginkan, pilih sebagian kecil grup data yang mungkin mengandung pola dari permasalahan.

3. *Data preparation*
    a. Siapkan dari data awal, kumpulan data yang akan digunakan untuk keseluruhan fase berikutnya. Fase ini merupakan pekerjaan berat yang perlu dilaksanakan secara intensif.
    b. Pilih kasus dan variabel yang ingin dianalisis dan yang sesuai dengan analisis yang akan dilakukan.
    c. Lakukan perubahan pada beberapa variabel jika dibutuhkan.
    d. Sipakan data awal sehingga siap untuk perangkat pemodelan.
4. *Modelling*
    a. Pilihan dan aplikasikan teknik pemodelan yang sesuai.
    b. Membangun model
    c. Menilai model
    d. Jika diperlukan, proses dapat kembali ke fase pengolahan data untuk menjadikan data ke dalam bentuk yang sesuai dengan spesifikasi kebutuhan teknik *Data Mining* tertentu.
5. *Evaluation*
    a. Mengevaluasi satu atau lebih model yang digunakan dalam fase pemodelan untuk mendapatkan kualitas dan efektivitas sebelum disebarkan untuk digunakan.
    b. Menetapkan apakah terdapat model yang memenuhi tujuan pada fase awal.
    c. Menentukan apakah terdapat permasalahan penting dari bisnis atau penelitian yang tidak tertangani dengan baik.
    d. Mengambil keputusan berkaitan dengan penggunaan hasil dari *Data Mining*.
6. *Deployment*
    a. Menggunakan model yang dihasilkan. Terbentuknya model tidak menandakan telah terselesaikannya proyek.
    b. Contoh sederhana penyebaran: pembuatan laporan.
    c. Contoh kompleks penyebaran: penerapan proses *Data Mining* secara paralel pada departemen lain.

2.2.5 *Association Rule*

Association rule adalah salah satu teknik utama atau prosedur dalam *Market Basket Analysis* untuk mencari hubungan antar-*item* dalam suatu *dataset* dan menampilkan bentuk *association rule* [13]. *Association rule* (aturan asosiasi) akan menemukan pola tertentu untuk mengasosiasikan data yang satu dengan data yang lain. Untuk mencari *association rule* dari suatu kumpulan data, tahap pertama yang harus dilakukan adalah mencari *frequent itemset* terlebih dahulu. *Frequent itemset* adalah sekumpulan *item* yang sering muncul secara bersamaan. Setelah semua pola *frequent itemset* ditemukan, barulah mencari aturan assosiatif atau aturan keterkaitan yang memenuhi syarat yang telah ditentukan.

Jika diasumsikan bahwa barang yang dijual di swalayan adalah semesta, maka setiap barang akan memiliki *boolean* variabel yang akan menunjukkan keberadaannya atau tidak barang tersebut dalam satu transaksi atau satu keranjang belanja. Pola *boolean* yang didapat digunakan untuk menganalisa barang yang dibeli secara bersamaan. Pola tersebut dirumuskan dalam sebuah *association rule*. Sebagai contoh konsumen biasanya akan membeli kopi dan susu yang ditunjukkan sebagai berikut:

Kopi → susu [*support* = 2%, *confidence* = 60%]

Association rule diperlukan suatu variabel ukuran yang ditentukan sendiri oleh *user* untuk menentukan batasan sejauh mana atau sebanyak apa *output* yang diinginkan *user*. *Support* dan *Confidence* adalah sebuah ukuran kepercayaan dan kegunaan suatu pola yang telah ditemukan. Nilai *support* 2% menunjukkan bahwa keseluruhan dari total transaksi konsumen membeli kopi dan susu secara bersamaan yaitu sebanyak 2%. Sedangkan *confidence* 60% yaitu menunjukkan bila konsumen membeli kopi dan pasti membeli susu sebesar 60%. *Association rule* memiliki dua tahap pengerjaan, yaitu:

1. Mencari kombinasi yang paling sering terjadi dari suatu *itemset.*
2. Mendefinisikan *condition* dan *result* (untuk *conditional association rule*).

Dalam menentukan suatu *associaiton 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*. Metodologi dasar analisis asosiasi terbagi menjadi dua tahap [14]:

1. Analisis pola frekuensi tinggi

Tahap ini mencari kombinasi *item* yang memenuhi syarat *minimum* dari nilai *support* dalam *database*. Nilai *support* sebuah *item* diperoleh dengan rumus berikut.

$$Support(A) = \frac{Jumlah\ transaksi \text{ yang } mengandung\ A}{Jumlah\ total\ transaksi} X100\% \qquad \text{Persamaan (2-1)}$$

Sementara itu, nilai *support* dari 2 *item* diperoleh dari rumus berikut.

$$Support(A,B) = P(A \cap B) \qquad \text{Persamaan (2-2)}$$

$$Support(A,B) = \frac{Jumlah\ transaksi \text{ yang } mengandung\ A\ dan\ B}{Jumlah\ total\ transaksi} \text{ Persamaan (2-3)}$$

2. Pembentukan aturan asosiasi

Setelah semua pola frekuensi tinggi ditemukan, 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:

$Confidence = P(B \mid A) = \frac{Jumlah\ transaksi \text{ yang } mengandung\ A\ dan\ B}{Jumlah\ transaksi\ yang\ mengandung\ A}$ Persamaan (2-4)

2.2.6 Algoritma *CT-PRO*

Gopalan dan Sucahyo (2004) menyebutkan *CT-PRO* merupakan pengembangan dari metode *FP-Growth*, di mana *CT-PRO* memiliki *tree* dengan jumlah *node* lebih sedikit dibanding *FP-Growth*. Jumlah *tree* yang lebih sedikit dikarenakan salah satu kelemahan dari algoritma *FP-Growth* adalah overhead dalam membangun banyak *FP-Tree* kondisional dalam tahap *mining*. Faktor tersebut mengurangi performa algoritma ketika pola bertambah panjang dan atau minimal *support* bertambah kecil, sehingga membutuhkan waktu yang lebih panjang dan proses pembangunan *tree* secara berulang. Dibandingkan dengan *FP-Growth*, Apriori memiliki beberapa kelemahan seperti keharusan untuk melakukan *Pattern matching* secara berulang-ulang yang menyebabkan proses *mining* memakan waktu semakin lama dan untuk data yang lebih besar akan menghasilkan kombinasi yang sangat banyak. Langkah-langkah algoritma *CT-PRO* adalah sebagai berikut:

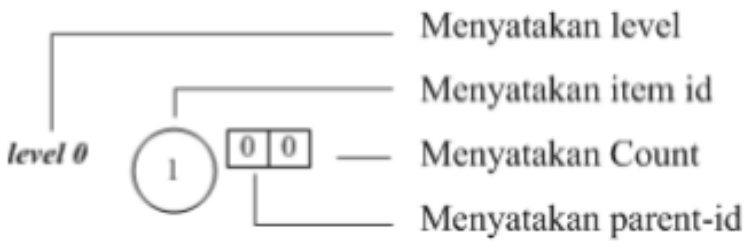
1. Menemukan *item-item* yang *frequent*

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. Membuat *CFP-Tree*

Setelah ditemukan *item-item* yang *frequent* kemudian dilakukan pembangunan *CFP-Tree*. *Frequent item* yang ada diurutkan sesuai *Global item* dari nilai yang terbesar ke terkecil. *CFP-Tree* adalah *tree* dengan properti sebagai berikut:

a. *CFP-Tree* terdiri dari *tree* yang memiliki *root* yang mewakili *Index* dari *item* dengan tingkat kemunculan tertinggi dan kumpulan sub*tree* sebagai anak dari *root*.
b. Jika I = {i, i2, …, ik} adalah kumpulan dari *frequent item* dalam transaksi, *item* dalam transaksi akan dimasukkan ke dalam *CFP-Tree* dimulai dari *root* sub*tree* yang merupakan i1 dalam header *Table*.
c. *Root* dari *CFP-Tree* merupakan level-0 dari *tree*.
d. Setiap *node* dalam *CFP-Tree* memiliki empat field utama yakni *item*-id, parentid, *count* yang merupakan jumlah *item* pada *node* tersebut, dan level yang menunjukkan struktur data *tree* pada *node* tersebut dimulai dari *item* yang terdapat pada header *Table* dengan level yang terdapat pada *CFP-Tree*



Gambar 2. 4 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
```

Gambar 2. 5 Algorithm 1 CT-PRO Algorithm: Step 1 and Step 2[16]

2. Melakukan penggalian *frequent Patterns*

Global item Table diurutkan dari *item* terkecil ke terbesar, karena algoritma *CT-PRO* bekerja dengan melakukan *bottom-up mining*. Masing-masing *Global item Table* di lakukan pencarian *node* yang berkaitan dengan *item* tersebut, dan kemudian diseleksi berdasarkan *minimum support*nya maka diperoleh *Local item Table*. Dari *local item Table* dibuat *CFP-Tree* berdasarkan *minimum support* yang telah ditentukan. Kemudian *item* yang memenuhi minimal *support* dilakukan *mining* dengan rumus confidence. 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
```

Gambar 2. 6 Algoritma 1 CT-PRO Penggalian *Frequent Patterns*[16]

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 Sample 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*

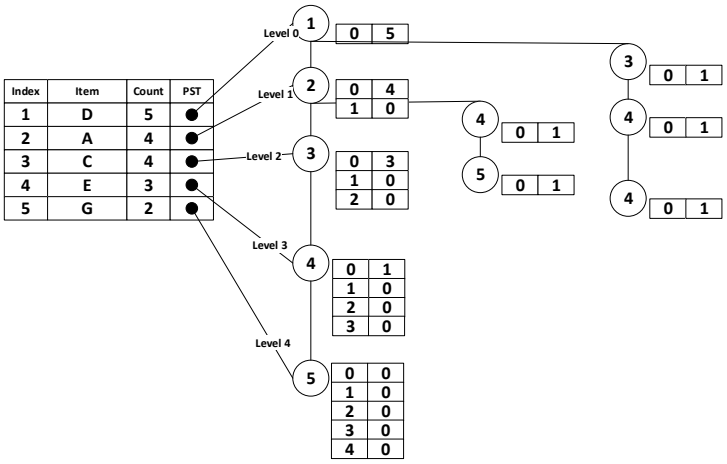
tid	*Item*
1	C D E F G H
2	A C D E
3	A B D E G
4	A C D H
5	A C D

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

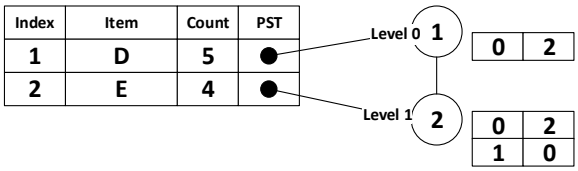
Tid	*Item*
1	1 2 4 5
2	1 2 3 4
3	1 3 4 5
4	1 2 3
5	1 2 3

Setelah dilakukan *Mapping* data selanjutnya adalah membangun *Global CFP-Tree* dengan menggunakan data transaksi pada *Table* 1.2 berikut hasil dari pembangunan *Global* CFP – *Tree*:



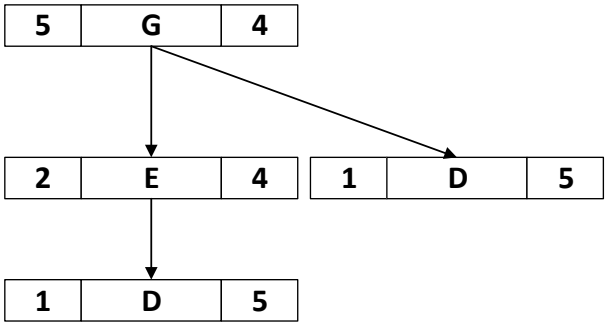
Gambar 2. 7 Hasil *Global CFP-Tree*

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 penelusuran 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. 8 *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. 9 *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 *Unified Modeling Language (UML)*

Unified Modeling Language (UML) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek serta aplikasinya. Berikut adalah beberapa model yang digunakan dalam perancangan *Data Mining* Pemaketan Barang di Minimarket HD Mart untuk menggambarkan sistem dalam UML:

1. Diagram *Use Case*
2. Diagram Kelas
3. Diagram Aktivitas
4. Diagram Objek

2.2.7.1 Diagram *Use Case*

Diagram use case adalah model fungsional sebuah sistem yang menggunakan aktor dan *use case*. *Use case* adalah layanan (*services*) atau fungsi–fungsi yang disediakan oleh sistem untuk penggunanya.

Deskripsi Diagram *Use Case*:

1. Sebuah *use case* adalah di mana sistem digunakan untuk memenuhi satu atau lebih kebutuhan pemakai.
2. *Use case* merupakan awal yang sangat baik untuk setiap fase pengembangan berbasis objek, *design testing*, dan dokumentasi.
3. *Use case* menggambarkan kebutuhan sistem dari sudut pandang di luar sistem.

4. *Use case* menentukan nilai yang diberikan sistem kepada pemakainya.
5. *Use case* hanya menetapkan apa yang seharusnya dikerjakan oleh sistem, yaitu kebutuhan fungsional sistem.
6. *Use case* tidak untuk menentukan kebutuhan non fungsional, misal: sasaran kerja, bahasa pemrograman.

2.2.7.2 Diagram Kelas

Diagram kelas adalah Diagram UML yang menggambarkan kelas-kelas dalam sebuah sistem dan hubungannya antara satu dengan yang lain, serta dimasukkan pula atribut dan operasi. Tahapan dari Diagram kelas adalah sebagai berikut:

1. Mengidentifikasi objek dan mendapatkan kelas-kelasnya.
2. Mengidentifikasi atribut kelas-kelas.
3. Mulai mengkonstruksikan kamus data.
4. Mengidentifikasi operasi pada kelas-kelas.
5. Mengidentifikasikan hubungan antar kelas dengan menggunakan *asosiasi*, *agregasi*, dan *inheritance* (pewarisan).

2.2.7.3 Diagram Aktivitas

Diagram aktivitas adalah representasi grafis dari seluruh tahapan alur kerja. Diagram ini mengandung aktivitas, pilihan tindakan, perulangan dan hasil dari aktivitas tersebut. Diagram ini dapat digunakan untuk menjelaskan proses bisnis dan alur kerja operasional secara langkah demi langkah dari komponen suatu sistem.

2.2.7.4 Diagram Objek

Diagram *objek* adalah suatu Diagram yang berfungsi untuk mengatur atribut, objek dan hubungan antara contoh dalam Diagram. Diagram objek juga dapat menampilkan struktur model sistem dalam waktu tertentu. Diagram objek menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. Pada Diagram objek harus dipastikan semua kelas yang sudah didefinisikan pada Diagram kelas harus dipakai objeknya, karena jika tidak, pendefinisian kelas itu tidak dapat dipertanggungjawabkan. Meskipun Diagram kelas diperlukan untuk mendokumentasikan struktur kelas, tetapi Diagram objek,

dapat berguna. Sebuah Diagram objek dasarnya instansi semua atau bagian dari Diagram kelas

2.2.8. *MySQL*

MySQL adalah *multiuser database* yang menggunakan bahasa *structured Query Language* (SQL). *MySQL* dalam operasi *client server* melibatkan *server daemon MySQL* di sisi *server* dan berbagai macam program serta *library* yang berjalan di sisi *client. MySQL* mampu menangani data yang cukup besar. Perusahaan yang mengembangkan *MySQL* yaitu TcX, mengaku bahwa *MYSQL* yang mampu menyimpan data lebih dari 40 *database*, 10.000 tabel dan sekitar 7 juta baris, totalnya kurang lebih 100 *Gigabyte* data [15].

SQL adalah bahasa standar yang digunakan untuk mengakses *database server*. Bahasa ini pada awalnya dikembangkan oleh IBM, namun telah diadopsi dan digunakan sebagai bahasa yang standar industri. Dengan menggunakan *SQL*, proses akses *database* menjadi lebih *user-friendly* dibandingkan dengan menggunakan DBASE atau *Clipper* yang masih menggunakan perintah-perintah pemrograman.