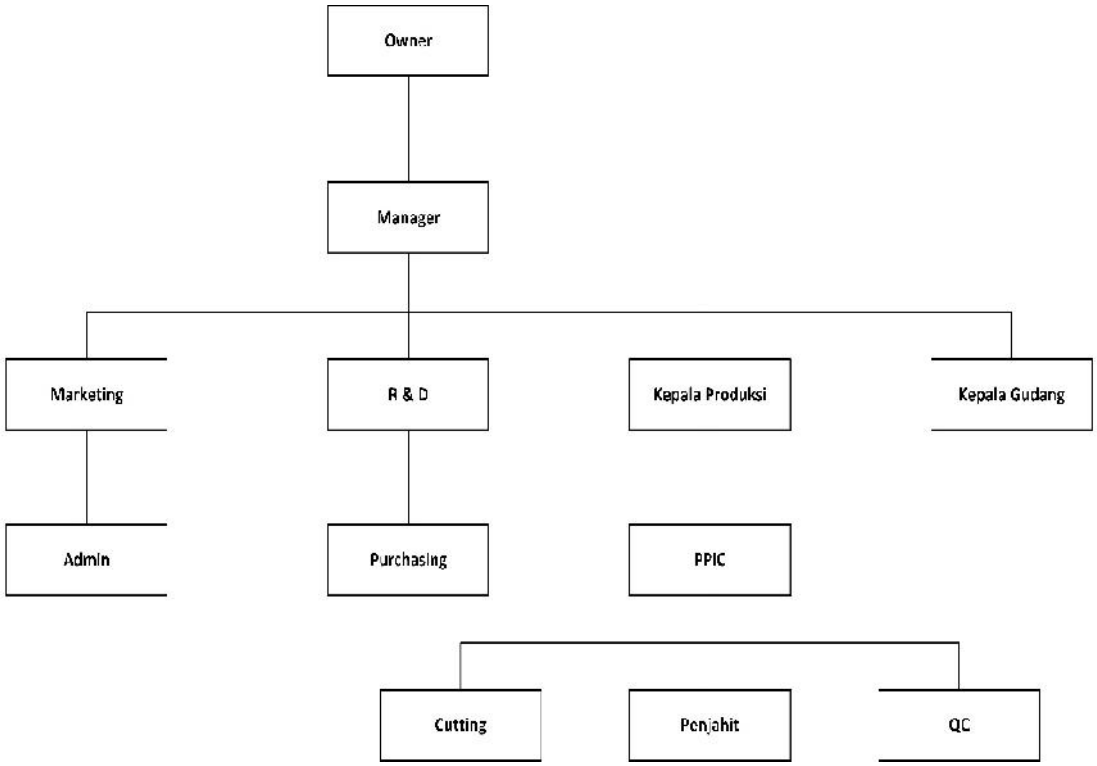
BAB 2
TINJAUAN PUSTAKA

2.1 Profil Institusi

PT. Contin Motosport merupakan perusahaan yang bergerak pada bidang produksi perlengkapan berkendara sepeda motor (*apparel ridingware)*, dengan produk unggulan jaket, perusahaan ini berdiri sejak tahun 2006 silam beralamat di Komplek Fajar Raya Estate Blok A1 No 35 Cihanjuang, Cimahi. Perusahaan ini kini telah cukup berkembang walaupun pemasaran produknya belum sampai ke luar negeri, namun produk-produknya ini sudah cukup tersebar di Indonesia.

2.1.1 Struktur Organisasi

Struktur organisasi perusahaan adalah gambar suatu bagan yang menerangkan posisi dan hierarki struktur kerja pegawai di dalam suatu perusahaan. Untuk struktur organisasi PT. Contin Motosport dapat dilihat pada gambar di bawah ini:



Gambar 2. 1 Struktur Organisasi di PT. Contin Motosport

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 PT Contin Motosport :

1. *Owner*

   Owner bekerja sebagai orang yang bertanggung jawab atas semua tindakan yang diperlukan agar PT. Contin Motosport berjalan lancar, menjalankan program perusahaan untuk mengejar target penjualan.

2. Manajer

   Manajer bertugas untuk mengontrol kinerja para pekerja secara global (kaki tangan *owner*), juga bertugas untuk melaporkan keadaan perusahaan, mulai dari transaksi, keluhan konsumen serta masalah yang ada.

3. Marketing

   Marketing bekerja memasarkan produk secara *online* maupun *offline*, selain itu juga menganalisa dan mengembangkan strategi marketing, melakukan evaluasi untuk memastikan tercapainya target penjualan.

4. *Admin*

   Admin bertugas untuk mengelola berbagai data, seperti data penjualan, data pembelian bahan, data stok produk beserta pengarsipan data-data lainnya yang bersangkutan dengan perusahaan.

5. *R & D (Research & Developement)*

   R&D bertanggung jawab untuk segala aktivitas riset dan pengembangan di perusahaan. Bagian *R&D* juga bertanggung jawab untuk memastikan kualitas performansi dalam perusahaan sesuai dengan standar yang telah ditetapkan perusahaan. Selain itu *R&D* mengelola sejumlah dana tertentu yang telah dianggarkan perusahaan untuk riset dan pengembangan.

6. *Purchasing*

   Purchasing tugas pokoknya adalah membeli baik itu membeli suatu barang, bahan produksi ataupun jasa pengiriman produk. Juga mencari harga barang bahan produksi terbaik dan tidak luput pula ketersediaan barang tersebut.

7. Kepala Produksi

   Kepala Produksi bertugas untuk mengontrol kinerja para pekerja yang khususnya berada pada ruang lingkup produksi, bertugas untuk menentukan produksi, melaporkan keadaan produksi barang. Selain itu kepala produksi juga bertugas menentukan bahan-bahan yang dipakai untuk proses produksi dari desain yang baru.

8. *PPIC*

   Tugas *PPIC*, menyediakan kebutuhan (marketing) dengan tepat waktu, tepat barang dan tepat pula jumlahnya.

9. *Cutting*

   Bertugas untuk membuat pola jenis produk yang akan diproduksi sesuai dengan arahan kepala produksi.

10. Penjahit

    Mengolah pola jenis produk dari bagian *cutting* menjadi produk utuh.

11. *QC (Quality Control)*

    Quality Control bertugas untuk mengecek produk utuh sebelum siap untuk dipasarkan untuk menghindari kecacatan dsb.

12. Kepala Gudang

    Mengawasi kegiatan keluar masuk barang yang akan dijual atau dipesan

2.2 Landasan Teori

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

2.2.1 Data

Data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, hewan, peristiwa, konsep, keadaan, dan sebagainya, yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya [7]. 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 [8].

2.2.2 Basis Data

Basis Data terdiri atas 2 kata, yaitu Basis dan Data. Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang/berkumpul. Sedangkan Data adalah representasi fakta dunia nyata mewakili suau objek seperti manusia pegawai, siswa, pembeli, pelanggan), barang, hewan, peristiwa, konsep,keadaan, dan sebagainya, yang diwujudkan dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya. Sebagai satu kesatuan istilah, Basis Data (*Database*) sendiri dapat didefenisikan dalam sejumlah sudut pandang seperti [9] :

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 bersamaaan 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 penyimpan elektronik.

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 [9] :

1. Pembuatan basis data baru (*create database*), yang identik dengan pembuatanlemari 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 arsip yang telah ada.
4. Penghapusan *file*/tabel dari suatu basis data (*drop table*), yang identik dengan perusakan map arsip lama yang ada di sebuah lemari arsip.
5. Penambahan/pengisian data baru ke sebuah *file*/tabel disebuah basis data (*insert*), yang identik dengna 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.

2.2.2.2 Objektif Basis Data

Telah disebutkan di awal 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. Disamping itu, pemamfaatan basis data untuk pengelolaan data, juga memiliki tujuan-tujuan lain [9].

Secara lebih lengkap, pemanfaatan basis data dilakukan untuk memenuhi sejumlah tujuan (objektif) seperti buku ini [9]:

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 secara 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 kadarluarsa. 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 di lokasi 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).

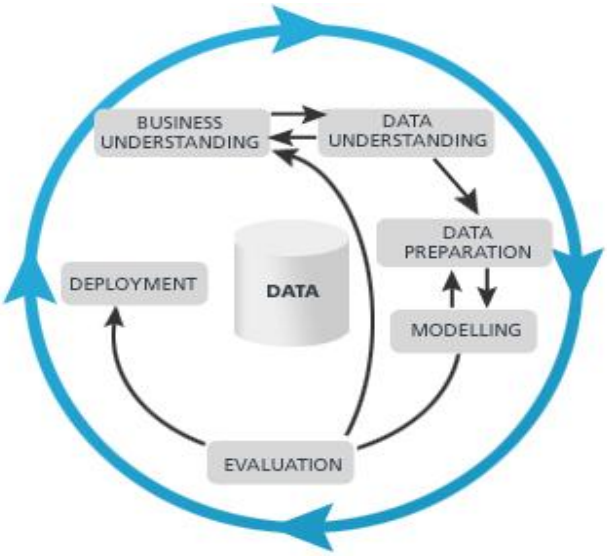
2.2.3 Data Mining

Data mining adalah suatu istilah yang digunakan untuk menguraikan penemuan pengetahuan dalam *database*. Didalamnya juga terdapat proses yang menggunakan teknik statistik, matematika, kecerdasan buatan, dan *machine learning* untuk mengekstraksi dan mengidentifikasi informasi yang bermanfaat dan pengetahuan yang terkait dari berbagai *database* [10].

Menurut Pramudiono [11], “Data mining adalah serangkaian proses untuk menggali nilai tambah dari suatu kumpulan data berupa pengetahuan yang selama ini tidak diketahui secara manual.”

2.2.3.1 Tahapan Data *Mining*

CRISP-DM (*CRoss-Industry Standard Process for Data Mining*) merupakan suatu) merupakan suatu standar yang telah dikembangkan pada tahun 1996 yang ditunjukkan untuk melakukan proses analisis dari suatu industri sebagai strategi pemecahan masalah dari bisnis satu unit penelitian [5]. Untuk data yang dapat di proses dengan *CRISP-DM* ini, tidak ada ketentuan atau karakteristik tertentu, karena data tersebut akan diproses kembali pada fase-fase di dalamnya.



Gambar 2. 2 Metode *CRISP-DM* [5]

Berikut ini adalah penjelasan mengenai enam tahap siklus hidup pengembangan *data mining* berdasarkan gambar di atas:

1. *Business Understanding*

   Tahap pertama adalah memahami tujuan dan kebutuhan dari sudut pandang bisnis, kemudian menterjemakan pengetahuan ini ke dalam pendefinisian masalah dalam *data mining*. Tahap selanjutnya menilai situasi pada perusahaan, tahap ini melibatkan lebih rinci fakta tentang sumber daya, asumsi dan faktor-faktor lain yang harus di pertimbangkan dalam menentukan tujuan data dan proyek. Selanjutnya akan ditentukan rencana dan strategi untuk mencapai tujuan tersebut.

2. *Data Understanding*

   Tahap ini dimulai dengan pengumpulan data yang kemudian akan dilanjutkan dengan proses untuk mendapatkan pemahaman yang mendalam tentang data, mengidentifikasi masalah kualitas data, atau untuk mendeteksi adanya bagian yang menarik dari data yang dapat digunakan untuk hipotesa untuk informasi yang tersembunyi.

3. *Data Preparation*

   Tahap ini meliputi semua kegiatan untuk membangun *dataset* akhir (data yang akan diproses pada tahap pemodelan/*modeling*) dari data mentah. Tahap ini dapat diulang beberapa kali. Pada tahap ini juga mencakup pemilihan tabel, *record*, dan atribut-atribut data, termasuh proses pembersihan dan *mapping* data untuk kemudian dijadikan masukan dalam tahap pemodelan (*modeling*).

4. *Modeling*

   Dalam tahap ini akan dilakukan pemilihan dan penerapan berbagai teknik pemodelan dan beberapa parameternya akan disesuaikan untuk mendapatkan nilai yang optimal. Secara khusus, ada beberapa teknik berbeda yang dapat diterapkan untuk masalah *data mining* yang sama. Di pihak lain ada teknik pemodelan yang membutuhan format data khusus. Sehingga pada tahap ini masih memungkinan kembali ke tahap sebelumnya.

5. *Evaluation*

   Pada tahap ini, model sudah terbentuk dan diharapkan memiliki kualitas baik jika dilihat dari sudut pandang analisa data. Pada tahap ini akan dilakukan evaluasi terhadap keefektifan dan kualitas model sebelum digunakan dan menentukan apakah model dapat mencapat tujuan yang ditetapkan pada fase awal (*Business Understanding*). Kunci dari tahap ini adalah menentukan apakah ada masalah bisnis yang belum dipertimbangkan. Di akhir dari tahap ini harus ditentukan penggunaan hasil proses *data mining*.

6. *Deployment*

   Pada tahap ini, pengetahuan atau informasi yang telah diperoleh akan diatur dan dipresentasikan dalam bentuk khusus sehingga dapat digunakan oleh pengguna. Tahap *deployment* dapat berupa pembuatan laporan sederhana atau mengimplementasikan proses *data mining* yang berulang dalam perusahaan. Dalam banyak kasus, tahap *deployment* melibatkan konsumen, di samping analis data, karena sangat penting bagi konsumen untuk

memahami tindakan apa yang harus dilakukan untuk menggunakan model yang telah dibuat.

2.2.3.2 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 [12].

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*.

2.2.4 Metode Association Rule

Assocition rule adalah salah satu teknik utama atau prosedur dalam *Data Mining* untuk mencari hubungan antar-*item* dalam suatu *dataset* dan menampilkan bentuk *association rule* [2]. *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.

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* [13].

2.2.4.1 Metodologi Dasar Analisis Asosiasi

Metodologi dasar *Association Rule* terbagi menjadi dua tahap, yaitu [14]:

1. **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 [10]. 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 *mincof*-nya dapat diisi rendah [15].

   Untuk rekomendasi dalam menentukan *minimum support* dapat diambil dari perhitungan rata-rata 1 jenis barang pada data yang digunakan, seperti rumus berikut :

$$Minimum\ \ support\ = \frac{Jumlah\ data\ yang\ digunakan}{Jumlah\ jenis\ item\ yang\ digunakan} \qquad \text{(Persamaan 2-1)}$$

Nilai *support* sebuah *item* diperoleh dengan rumus :

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

Persamaan 2 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 :

$$Support(\mathrm{A,B}) = \mathrm{P(A \cap B)} = \frac{Jumlah\ transaksi\ yang\ mengandung\ item\ A\ \&\ item\ B}{Jumlah\ total\ transaksi}$$

(Persamaan 2-3)

Persamaan 3 menjelaskan bahwa nilai *support* 2 *item sets* 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.

2. **Pembentukan Aturan *Assosiatif***

   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 :

$$Confidence = (A \rightarrow B) = \frac{Jumlah\ transaksi\ yang\ mengandung\ item\ A\ dan\ B}{Jumlah\ total\ Transaksi\ A} \text{ x100\%}$$

(Persamaan 2-4)

Persamaan 4 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).

2.2.5 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 dan pada algoritma ini tidak terdapat pula pembentukan *conditional FP-Tree* yang berulang-ulang, sehingga baik pembentukan *tree* maupun *frequent itemset mining* yang dilakukan menjadi lebih cepat [3].

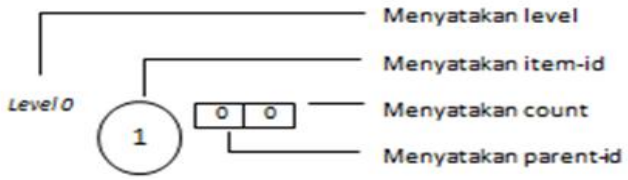
Langkah-langkah algoritma *CT-PRO* adalah sebagai berikut [3, 16, 17]:

1. **Mencari *item-item* yang *frequent***, pada tahap ini terjadi proses-proses sebagai berikut:
   a. Data yang telah dikumpulkan, diseleksi dan pilih data yang relevan (data yang lengkap).
   b. Data yang ada, kemudian dilakukan transformasi data.
   c. Kemudian masing-masing data diseleksi berdasarkan *minimum support* yang telah ditentukan, kemudian didapat *Item Frequent Table*.
   d. Masing-masing *item* dihitung frekuensi kemunculannya sehingga dihasilkan *global item table*.
   e. Data kemudian di-*mapping* berdasarkan indeks pada *global item table*.

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 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 kedalam *CFP-Tree* dimulai dari *root subtree* 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*, *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 *header table* dengan *level* yang terdapat pada *CFP- Tree* [17].



Gambar 2. 3 Struktur *CFP-Tree*

2.2.6 Unified Modeling Languages (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 [18]. Berikut adalah beberapa model yang digunakan dalam perancangan aplikasi *Data Mining* penentuan produk di PT. Contin Motosport untuk menggambarkan sistem dalam UML:

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

2.2.6.1 *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 [19]. 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*.

2.2.6.2 *Activity* Diagram

Activity diagram memiliki pengertian yaitu lebih fokus kepada menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses [18]. 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 [19]. Dan *activity* dibuat berdasarkan sebuah atau beberapa *use case* pada *use case* diagram.

2.2.6.3 *Sequence* Diagram

Diagram sekuen menggambarkan interaksi objek pada *use case* dengan mendeksripsikan waktu hidup objek dan pesan yang dikirimkan dan di terima antar objek [20]. 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* [19]. *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.

2.2.6.4 *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 oprasi [19].

1. Atribut merupakan variabel-variabel yang di miliki oleh suatu kelas
2. Oprasi 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.