BAB II

TINJAUAN PUSTAKA

II. 1 Profil Prusahaan

Distro Slasher Clothing berdiri tahun 2013 di Cigondewah Rahayu No 133. Sebuah kawasan Sub-urban kota Bandung, yang juga merupakan kawasan tekstil terbesar di kota Bandung. Slasher bergerak di bidang fashion yang menangkap *Pop & Youth Culture, -so called- Indie Movement*, dan semua bentuk pergerakan *Do It Yourself*. Seakan sudah menjadi atau identik dengan kota Bandung, yang kadung tercitrakan sebagai kota kreatif, dan termasuk pionir dalam industri kreatif di Indonesia, Distro Slasher Clothing adalah perpanjangan tangan dari ide dan spirit kreatif yang diaplikasikan lewat *fashion*. Slasher *Streetwear* : "*Fashion Doesn't Need The Goverment. Existed to Survive, Dressed to Kill*"**[5]**.

II.1.1 Visi dan Misi

Visi

Menjadi perusahaan yang menghasilkan produk yang berkualitas tinggi, dan mampu menguasai pasar

Misi

Menjadi perusahaan *fashion* yang dikenal di seluruh Indonesia, dan mempunyai cabang di seluruh kawasan Indonesia

II.1.2 Logo Perusahaan

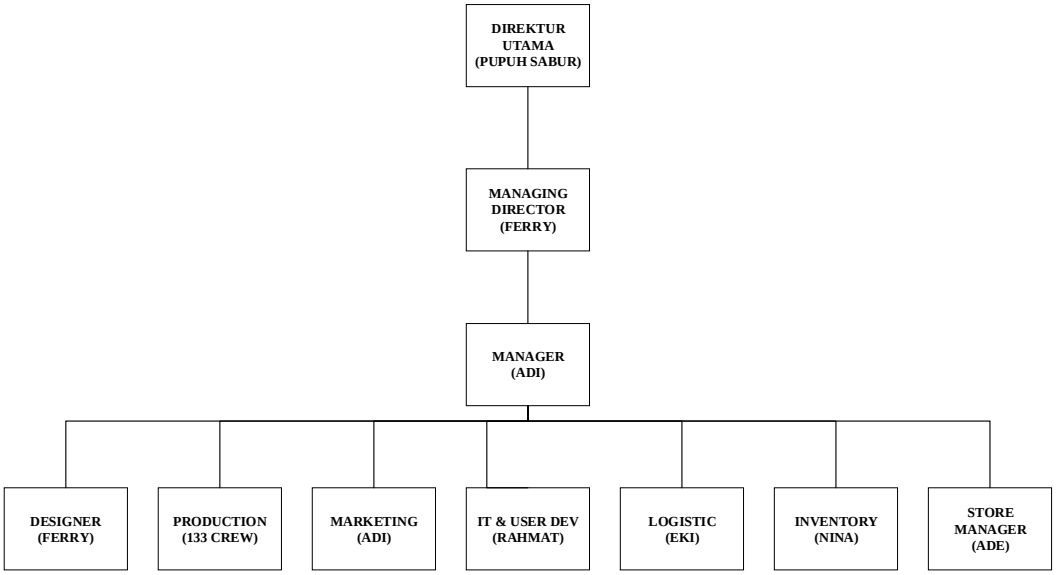
Adapun logo perusahaan Distro Slasher Clothing sebagai berikut :



Gambar II. 1 Logo Perusahaan

II.1.3 Struktur Perusahaan

Struktur organisasi dalam suatu perusahaan merupakan hal yang sangat penting, dengan adanya struktur organisasi ini memberikan pembagian tugas sesuai dengan bidangnya masing-masing. Adapun struktur organisasi di Distro Slasher Clothing adalah sebagai berikut :



Gambar II. 2 Struktur Organisasi Distro Slasher Clothing

II. 2 Landasan Teori

Landasan teori membahas mengenai materi atau teori apa saja yang digunakan sebagai acuan dalam membuat tugas akhir ini. Landasan teori yang diuraikan merupakan hasil dari studi literatur yaitu pengumpulan *data* dengan cara mempelajari jurnal, *browsing* internet, referensi buku dan bacaan-bacaan lainnya yang berhubungan erat dengan penelitian yang dilakukan.

II.2.1 Data

Data adalah fakta mentah mengenai orang, tempat, kejadian, dan hal-hal yang penting dalam organisasi **[6]**. *Data* merupakan bahasa, simbol-simbol pengganti lain yang disepakati oleh umum dalam menggambarkan objek, manusia, peristiwa, aktivitas, konsep. Singkatnya *data* merupakan suatu kenyataan apa adanya **[6]**.

II.2.2 Database

Database atau Basis *data* merupakan kumpulan *data* yang disimpan secara sistematis didalam komputer dan dapat diolah atau dimanipulasi menggunakan perangkat lunak (program aplikasi) untuk menghasilkan informasi. Pendefinisian basis *data* meliputi spesifikasi berupa tipe *data*, struktur, dan juga batasan-batasan *data* yang akan disimpan. Basis *data* merupakan aspek yang sangat penting dalam sistem informasi dimana basis *data* merupakan gudang penyimpanan *data* yang akan diolah lebih lanjut. Basis *data* menjadi penting karena dapat menghidari duplikasi *data*, hubungan antar *data* yang tidak jelas, organisasi *data*, dan juga update yang rumit **[6]**.

Basis *data* (*Database*) sendiri dapat didefinisikan dalam sejumlah sudut pandang seperti **[6]**:

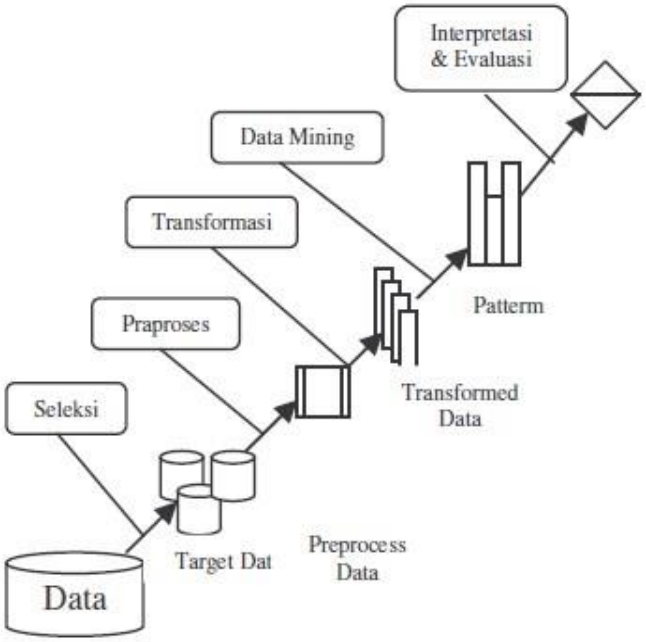
1. Himpunan kelompok *data* (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.
2. Kumpulan *data* yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redun-dansasisi) yang tidak perlu,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

II.2.3 *Database Management Sistem* (DBMS)

Aplikasi yang digunakan untuk membangun sebuah sistem basis *data* yang memungkinkan Pengguna untuk mendefinisikan, membuat, memelihara dan mengontrol akses ke *database* merupakan pengertian dari *Database Management Sistem* (DBMS). DBMS membantu dalam pemeliharaan dan pengolahan kumpulan *data* dalam jumlah besar, sehingga dengan bantuan. DBMS tidak akan menimbulkan kekacauan dan dapat digunakan oleh pengguna sesuai dengan kebutuhan. Bahasa basis *data* biasanya terdiri atas perintah-perintah yang diformulasikan dalam bentuk query sehingga perintah tersebut akan diproses oleh DBMS**[6]**.

II.2.4 Data Mining

Data mining merupakan istilah yang sering dikatakan sebagai suatu cara untuk menguraikan serta mencari penemuan berupa pengetahuan didalam suatu *database*. *Data mining* adalah proses pemilihan atau "menambang" pengetahuan dari sekumpulan *data* dalam jumlah yang banyak (Han, Jiawei 2006).



Gambar II. 3 Tahapan *Data Mining*

Terdapat beberapa Tahapan dalam data mining, antara lain :

a. Seleksi *Data*

   Biasanya *data* dari proses transaksi di simpan pada lokasi yang berbeda-beda. Maka dari itulah di butuhkan kemampuan dari sistem untuk dapat mengumpulkan *data* dengan cepat. Biasanya *data* tersebut sangat banyak dan memerlukan waktu dalam menganalisisnya. Maka dari itu di butuhkan seleksi *data* untuk memperkecil ruang lingkup dengan memanfaatkan *data* yang penting saja.

b. Pembersihan *Data*

   Data-data yang sudah terkumpul selanjutnya akan mengalami proses pembersihan. Pembersihan ini dilakukan untuk membuang record

yang keliru, merasionalisasi struktur *data* dan mengendalikan *data* yang hilang.

c. Transformasi *Data*

Transformasi *data* melakukan peringkasan *data* dan mengasumsikan *data* tersimpan di dalam tempat penyimpanan tunggal. Hasil dari *data* pusat yang telah teringkas menggunakan ekstraksi *data* di ringkas kembali dengan Transformasi *data* supaya *data* yang di hasilkan lebih padat dan di simpat di tempat penyimpanan tunggal. Fungsi agregate yang sering di gunakan adalah average, minimum, maximum, dan cunt.

d. Menjalankan Algoritma

Setelah melakukan semua proses di atas, maka algoritma *data mining* sudah siap untuk di jalankan.

e. Evaluasi *Data*

Inilah hasil akhir yang akan di sajikan berupa *data* yang mudah di pahami, yang merupakan hasil dari pengolahan *data* melakukan *Data mining*.

II.2.5 Metode *Association Rule*

Dalam pencarian hubungan antar item dalam suatu kumpulan *data* yang ditentukan, dibutuhkan suatu prosedur yang disebut *association rule*. Informasi yang diberikan dalam *association rule* berbentuk ”*if – then*” atau ”jika–maka” atau biasanya dikenal dengan istilah *antecedent. Association rule* meliputi dua tahap, yaitu mencari kombinasi yang paling sering terjadi dari suatu *itemset* dan membangkitkan *association rule* dari *Large itemset* yang telah dibuat sebelumnya. Dalam *association rule* terdapat dua ukuran, yaitu *support* dan *confidence*. *Rule* dikatakan *valid association rule* adalah yang memiliki *confidence* sama atau lebih dari *minimum confidence*. *Rule* dikatakan *strong rule* adalah memenuhi baik minimum *support* maupun minimum *confidence***[1]**. Rumus untuk perhitungan support dan confidence sebagai berikut :

a. ***Support*** (nilai penunjang) adalah presentase kombinasi item tersebut dalam *database*

Support, $s(X \rightarrow Y) = \frac{\sigma(X \cap Y)}{N} * 100$……...……..……(2.1)

N : Total transaksi

X : Produk yang dipresentasikan "jika"

Y : Produk yang dipresentasikan "maka"

b. ***Confidence*** (nilai kepastian) adalah kuatnya hubungan antar-item dalam aturan asosiasi

Confidence, $c(X \rightarrow Y) = \frac{\sigma(X \cap Y)}{\sigma(x)} * 100$……...……(2.2)

II.2.6 Algoritma *Improved Apriori*

Algoritma *improved* apriori merupakan peningkatan algoritma dari apriori biasa dengan perningkatan efisiensi dalam proses scaning *data*, dalam apriori biasa untuk menemukan setiap *itemset* harus melakukan *scaning* pada *database* secara penuh, sedangkan pada *improved apriori scaning database* dilakukan hanya pada saat *Large 1-itemset* diproduksi, untuk menemukan *Large k-itemset scaning* dilakukan pada *Large*(K-1) sehingga mereduksi waktu yang diperlukan dan meningkatkan efisiensi dalam proses asosiasi **[2].**

Adapun Tahapan dari *Improved Apriori* yang diteliti oleh Xiang Fang sebagai berikut :

1. Menentukan nilai minimal *support count* dan minimal *confidence*
2. Memproduksi L1 (*Large 1-itemsets*)

   Melakukan scaning *database* satu kali untuk memperoleh L1-*candidate*, setelah terbentuk hapus *itemset* yang mempunyai nilai kurang dari minimum *support count* untuk menemukan set dari *frequent items* L1.
3. Memproduksi Lk ($k \geq 2$)

   Untuk memproduksi Lk-*candidate* ($k \geq 2$) *scaning* dilakukan pada hasil lage itemset sebelumnya, yaitu pada L(k-1) dengan cara sebagai berikut :

   1) Membandingkan setiap *itemset* yang ada pada L(k-1).

2) Jika dari *itemset* yang dibandingkan terdapat id_transaksi yang sama, maka join dua Lk-1 untuk membentuk Lk-*candidate* dan tambahkan *itemset* yang mempunyai id_transaksi yang sama ke dalam Lk-*candidate*.
3) Setelah Lk-*candidate* terbentuk sepenuhnya, hapus *itemset* yang mempunyai nilai kurang dari *minimum support count* untuk menemukan set dari *frequent* items Lk.
4) Lakukan cara 1) – 3) untuk menemukan Lk-*candidate* selanjutnya (k = k+1).
5) Proses pembentuk *candidate* akan terus dilakukan hingga himpunan *candidate* itemsetnya null, atau sudah tidak ada lagi *candidate* yang dapat dibentuk.

4. Menghitung *Confidence* dari setiap *rule* yang sudah terbentuk agar mengetahui *rule* yang sudah terbentuk merupakan *rule* yang kuat atau tidak, rule yang kuat mempunyai nilai yang sama atau lebih dari nilai minimum *confidence* yang sudah ditentukan sebelumnya.

Kelemahan dari tradisional Apriori adalah setiap item yang ada pada *candidate* untuk mencari *frequent* dari setiap item harus melakukan scaning pada seluruh *database*, dan efesiensi dari eksekusinya sangat rendah. Sedangkan pada Improved Apriori yang diteliti oleh Xiang Fang, scaning *database* hanya dilakukan untuk memproduksi L1-*candidate*, selanjutnya untuk mencari Lk selanjutnya scaning dilakukan pada L(k-1) yang sudah terbentuk sebelumnya, Sehingga mereduksi I/O Load dan meningkatkan efesiensinya.

II. 3 Alat Permodelan Sistem

Alat-alat pemodelan sistem membahas mengenai tools apa saja yang digunakan dalam membuat tugas akhir ini.

II.3. 1 *Entriry Relation Diagram* (ERD)

Entity Relationship Diagram merupakan model *data* berupa notasi grafis dalam pemodelan *data* konseptual yang menggambarkan hubungan antara penyimpan satu dengan penyimpan yang lain. Model *data* sendiri merupakan sekumpulan cara, peralatan untuk mendeskripsikan *data-data* yang memiliki

hubungan satu sama lain, semantiknya, serta batasan konsistensi. Model *data* terdiri dari model hubungan entitas dan model relasional. Diagram hubungan entitas ditemukan oleh Peter Chen dalam buku Entity Relational Model-Toward a Unified of Data. Pada saat itu diagram hubungan entitas dibuat sebagai bagian dari perangkat lunak yang juga merupakan modifikasi khusus, karena tidak ada bentuk tunggal dan standar dari diagram hubungan entitas [7].

II.3. 2 Diagram Konteks

Diagram konteks merupakan diagram yang menggambarkan kondisi sistem yang ada baik input maupun output serta menyertakan terminator yang terlibat dalam penggunaan sistem. Diagram ini akan memberi gambaran tentang keseluruhan sistem. Sistem dibatasi oleh boundary (dapat digambarkan dengan garis putus). Dalam diagram konteks hanya ada satu proses. Tidak boleh ada store dalam diagram konteks**[7]**.

Diagram konteks berisi gambaran umum (secara garis besar sistem yang akan dibuat. Secara kalimat, dapat dikatakan bahwa diagram konteks ini berisi "siapa saja yang memberi *data* (dan *data* apa saja) ke sistem, serta kepada siapa saja informasi (dan informasi apa saja) yang harus dihasilkan sistem." Maka dapat disimpulkan bahwa diagram konteks adalah diagram yang terdiri dari suatu proses dan menggambarkan ruang lingkup suatu sistem. Diagram konteks merupakan level tertinggi dari DFD yang menggambarkan seluruh input ke sistem atau output dari sistem.

II.3. 3 *Data Flow Diagram* (DFD)

Data Flow Diagram (DFD) adalah alat pembuatan model yang memungkinkan profesional sistem untuk menggambarkan sistem sebagai suatu jaringan proses fungsional yang dihubungkan satu sama lain dengan alur *data*, baik secara manual maupun komputerisasi. DFD ini sering disebut juga dengan nama *Bubble chart*, *Bubble* diagram, model proses, diagram alur kerja, atau model fungsi. DFD ini adalah salah satu alat pembuatan model yang sering digunakan, khususnya bila fungsi-fungsi sistem merupakan bagian yang lebih penting dan kompleks dari pada *data* yang dimanipulasi oleh sistem. Dengan kata lain, DFD

adalah alat pembuatan model yang memberikan penekanan hanya pada fungsi sistem **[7]**.

DFD merupakan alat perancangan sistem yang berorientasi pada alur *data* dengan konsep dekomposisi dapat digunakan untuk penggambaran analisa maupun rancangan sistem yang mudah dikomunikasikan oleh profesional sistem kepada pemakai maupun pembuat program.

II. 4 Alat-alat Pembangunan Perangkat Lunak

Alat-alat Pembangunan Perangkat Lunak membahas mengenai alat atau tools apa saja yang digunakan dalam membuat Aplikasi *data* mining dalam penelitian ini.

II.4. 1 C#

Bahasa pemrograman C# dikembangkan oleh *Microsoft* sebagai bahasa yang simple, modern, *general purpose*, dan berorientasi objek. Pengembangan bahasa C# sangat dipengaruhi oleh bahasa pemrograman terdahulu, terutama C++. Delphi, dan Java. C++ dikenal memiliki kecepatan yang tinggi dan memiliki akses memori hapir hingga ke *low level*. Namun bagi para programmer, C++ merupakan Bahasa yang relatif rumit dibandingkan bahasa pemrograman lainnya. Kehadiran C# memberi suntikan optimisme bagi para programmer untuk dapat mengembangkan aplikasi yang berdasa guna dengan baik cepat dan lebih mudah. Bahasa C# masih harus dikembangkan dengan kemampuan untuk melakukan berbagai *task*. Namun dalam hal performansi hingga saat ini C++ masih diakui sebagai salah satu yang terbaik.

C# dikembangkan sejalan dengan pengembangan teknologi .Net. Teknologi .NET telah berevolusi dari .NET 1.0 hingga .NET 4.0. Sendiri merupakan sebuah *framework* yang memiliki *base class library*, dan bisa diimplementasikan ke dalam beberpaa bahasa pemrograman yang dikembangkan *Microsoft* termasuk diantaranya adalah bahasa C#. *Microsoft* mengembangkan IDE yang secara khusus mendukung pengembangan aplikasi dengan teknologi .NET, versi terakhirnya yaitu Visual Studio 2010, mendukung teknologi .Net 4. Diantara karakteristik dari .NET adanya sebuah *Common Langguage Runtime* (CLR). CLR sendiri merupakan sebuah komponen *virtual machine* yang akan

diakses program pada saat *runtime* kode yang ditulis dalam bahasa C# ataupun VB.NET misalnya *dicompile* oleh .NET *Compiler* menjadi code dalam format *Common Intermediate Langguage* (CIL).

CIL merupakan sebuah format bahasa standar pada *level intermediate* yang digunakan dalam bidang teknologi .NET apa pun bahasa pemrograman yang digunakan oleh kode. Kemudian pada saat *runtime*, CLR mengeksekusi CIL dengan melakukan proses loading dan linkin serta menghasilkan bahasa mesin untuk mengeksekusi program. Proses ini memunculkan istilah *multilangguage* pada pengembangan aplikasi .Net programmer memiliki pilihan untuk menentukan pilihan untuk menggunakan bahasa pemrograman, yang familiar bagi mereka dan apapun bahasa pemgraman apapun mereka tetap memilih akan tetap memiliki keungulan-keungulan dan teknologi lainnya.

Standar *European Computer Maufacturer Association* (ECMA) mendatarkan beberapa tujuan desain dari bahasa pemrograman C#, sebagai berikut **[8]**:

1) Bahasa pemrograman C# dibuat sebagai bahasa pemrograman yang bersifat bahasa pemrograman general – purpose (untuk tujuan jamak), berorientasi objek, modern, dan sederhana.
2) Bahasa pemrograman C# ditujukan untuk digunakan dalam mengembangkan komponen perangkat lunak yang mampu mengambil keuntungan dari lingkungan terdistribusi.
3) Portabilitas programmer sangatlah penting, khususnya bagi programmer yang telah lama menggunakan bahasa pemrogaman C dan C++.
4) Dukungan untuk internasionalisasi (*multi- language*) juga sangat penting.
5) C# ditujukan agar cocok digunakan untuk menulis program aplikasi baik dalam sistem *klien-server* (*hosted sistem*) maupun sistem 38 *embedded* (*embedded sistem*), mulai dari perangkat lunak yang sangat besar yang menggunakan sistem operasi yang canggih hingga

kepada perangkat lunak yang sangat kecil yang memiliki fungsi-fungsi terdedikasi.

II.4. 2 SQL Server

SQL Server adalah sistem manajemen *database* relasional (RDBMS) yang dirancang untuk aplikasi dengan arsitektur *client/server*. Istilah *client, server,* dan *client/server* dapat digunakan untuk merujuk kepada konsep yang sangat umum atau hal yang spesifik dari perangkat keras atau perangkat lunak. Pada level yang sangat umum, sebuah *client* adalah setiap komponen dari sebuah sistem yang meminta layanan atau sumber daya (*resource*) dari komponen sistem lainnya. Sedangkan sebuah *server* adaah setiap komponen sistem yang menyediakan layanan atau sumber daya ke komponen sistem lainnya **[7]**.