BAB II

TINJAUAN PUSTAKA

2.1 Tinjauan Tempat Penelitian

2.1.1 Profil Perusahaan

Ultra Disc adalah sebuah perusahaan *franchise* yang bergerak di bidang usaha rental atau penyewaan VCD/DVD Original. Istilah *franchise* disebut juga sebagai usaha waralaba, *franchise* adalah pemberian hak atas merk dan sistem yang diketahui telah berhasil oleh pihak yang memiliki merk dan sistem (*franchisor*) ke pihak kedua sebagai penyelenggara (*franchisee*).

Franchisor tidak hanya memiliki merk dan sistem saja namun memiliki pengalaman yang lebih di bidang yang telah dikuasainya tersebut. Keseluruhan merk dan sistem harus selalu seragam dengan apa yang diberikan *franchisor*. Dengan keseragaman dalam merk dan sistem maka diupayakan meraih tujuan dengan cara yang sama.

Ultra Disc awalnya adalah perusahaan milik keluarga yang didirikan pada tanggal 14 November 1998 dengan outlet pertama berada di Ruko Vila Melati Mas Blok B 10/28, Serpong. Dikarenakan minat masyarakat terhadap VCD/DVD Original sangat banyak sedangkan harga VCD/DVD Original yang cukup mahal maka Ultra Disc melihat ini sebagai suatu peluang besar untuk mengembangkan usaha, maka melalui PT. ULTRA DISC PRIMA INTERNATIONAL,

Ultra Disc terus memperluas jaringan melalui pembukaan outlet-outlet baik outlet yang dimiliki oleh Ultra Disc Sepenuhnya (*corporate*) maupun outlet yang dibuka dengan bekerja sama dengan pihak lain (*franchisee*).

Saat ini Ultra Disc berpusat di Jakarta, tepatnya di Jalan Kembang Raya Jakarta Barat, merupakan pusat dari segala pengaturan bisnis yang dijalankan oleh setiap cabang yang tersebar di beberapa kota di Indonesia, antara lain Jakarta, Bandung, Semarang, Yogyakarta, Denpasar, Manado, Makassar, Medan, Lampung, Padang, Samarinda, Jambi, Bengkulu, Palembang, Pekanbaru, Tangerang, dan masih banyak lagi kota-kota lainya. Untuk di kota Bandung, salah satu cabangnya berada di daerah Antapani.

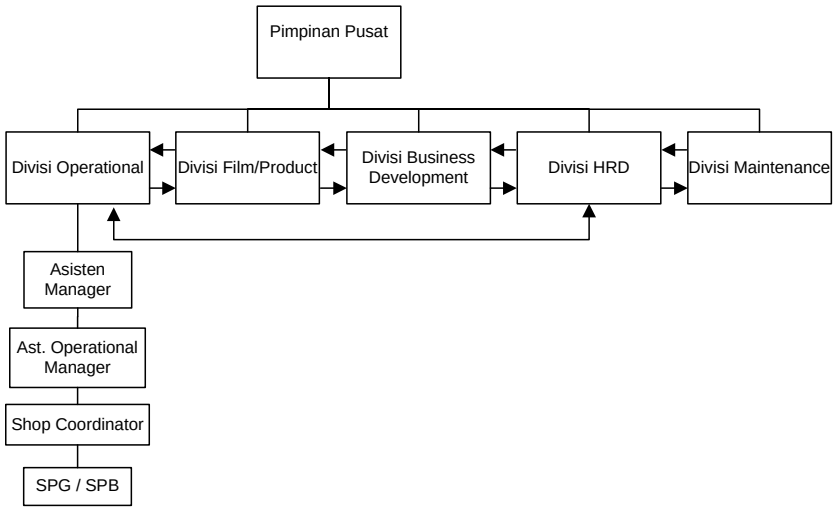
Ultra Disc dilengkapi *software* yang handal dengan menawarkan sistem penyewaan yang menarik bagi anggotanya, karena para anggota memiliki kelleluasaan untuk menentukan lamanya waktu penyewaan. Ultra Disc memiliki koleksi dari hampir seluruh produsen film terkemuka dunia, seperti Warner Bros, Buena Vista, Columbia Tristar, 20th Century Fox. Touchstone Pictures, Universal Studio, New Line Cinema dan lain-lain.

2.1.2 Visi dan Misi Perusahaan

PT Ultra Disc Prima International mempunyai visi yaitu untuk mengenalkan film original ke member, sedangkan misi dari PT Ultra Disc adalah "*We Serve The Best Entertainment To Your Home"*.

2.1.3 Struktur Organisasi Perusahaan

Struktur kepegawaian Ultra Disc khususnya jajaran sales (Operational Manager) di mulai dari lini yang terdepan adalah SPG / SPB sampai ke Asisten Operational Manager, karena merekalah berada di setiap outlet dan berhubungan langsung dengan pelanggan. Untuk lebih jelas mengenai struktur organisasi perusahaan, dapat dilihat pada Gambar 2.1 dibawah ini :



Gambar 2.1. Struktur Organisasi Perusahaan Ultra Disc

2.1.4 Deskripsi Kerja

a. Asisten Manager

   Asisten Manager bertugas untuk mengontrol kinerja outlet, dan bertugas untuk melaporkan keadaan outlet, mulai dari transaksi, keluhan bahkan masalah yang ada di outlet. Asisten manager juga merupakan bawahan dari pemilik outlet.

b. Asisten Operational Manager

   Asisten Operational Manager bertugas untuk mengontrol kinerja outlet, dan bertugas untuk melaporkan keadaan outlet, mulai dari transaksi, keluhan bahkan masalah yang ada di outlet untuk disampaikan ke pihak pusat.

c. Shop Coordinator

   Shop Coordinator bertanggung jawab untuk mengelola outlet, dan bertanggung jawab atas SPG/SPB yang ada dibawahnya,

d. SPG/SPB

   SPG/SPB memiliki deskripsi kerja yang sangat penting, karena posisi ini yang berinteraksi langsung dengan konsumen/member. SPG/SPB harus komunikatif dengan konsumen agar konsumen merasa puas dalam pelayanan yang diberikan.

2.2 Landasan Teori

2.2.1 Pengertian Data dan Informasi

2.2.1.1 Pengertian Data

Data adalah fakta atau apapun yang dapat digunakan sebagai input dan menghasilkan informasi. Data adalah kenyataan yang menggambarkan suatu kejadian dan kesatuan kenyataan. Data merupakan suatu istilah yang berbentuk jamak dari kata "*datum*" yang berarti fakta atau bagian dari fakta yang mengandung arti yang menghubungkan dengan kenyataan, simbol-simbol, gambar-gambar, kata-kata, angka-angka, huruf-huruf yang menunjukan suatu ide, objek, kondisi dan situasi.

Menurut *the liang gie*: "Data atau bahan keterangan adalah hal atau peristiwa kenyataan lainnya apapun yang mendukung suatu pengetahuan untuk dijadikan dasar guna penyusunan keterangan pembuatan kesimpulan atau penetapan keputusan, atau data ibarat bahan mentah yang melalui pengolahannya tertentu lalu menjadi keterangan (informasi)".

Kumpulan data yang saling berkaitan, berhubungan yang disimpan secara bersama-sama sedemikian rupa tanpa pengulangan yang tidak perlu, untuk memenuhi berbagai kebutuhan disebut basis data (*database*). Data-data ini harus mengandung semua informasi untuk mendukung semua

kebutuhan sistem. Proses dasar yang dimiliki oleh *database* ada empat, yaitu:

1. Pembuatan data-data baru *(create database)*
2. Penambahan data *(insert)*
3. Mengubah data *(update)*
4. Menghapus data *(delete)*

Database merupakan salah satu komponen yang penting dalam sistem informasi, karena merupakan basis dalam menyediakan informasi pada para pengguna. *Database* menjadi penting karena munculnya beberapa masalah bila tidak menggunakan data yang terpusat, seperti adanya duplikasi data, hubungan antar data tidak jelas, organisasi data dan *update* menjadi rumit. Jadi tujuan dari pengaturan data dengan menggunakan *database* adalah :

a. Menyediakan penyimpanan data untuk dapat digunakan oleh organisasi saat sekarang dan masa yang akan datang.
b. Cara pemasukan data sehingga memudahkan tugas operator dan menyangkut pula waktu yang diperlukan oleh pemakai untuk mendapatkan data serta hak-hak yang dimiliki terhadap data yang ditangani.
c. Pengendalian data untuk setiap siklus agar data selalu *up-to-date* dan dapat mencerminakan perubahan spesifik yang terjadi di setiap sistem.

d. Pengamanan data terhadap kemungkinan penambahan, modifikasi, pencurian dan gangguan-gangguan lain.

Penyusunan basis data meliputi proses memasukkan data kedalam media penyimpanan data, dan diatur dengan menggunakan perangkat Sistem Manajemen Basis Data (*Database Management System / DBMS*).

2.2.1.2 Pengertian Informasi

Informasi adalah hasil analisis dan sintesis terhadap data. Dengan kata lain, informasi dapat dikatakan sebagai data yang telah diorganisasikan ke dalam bentuk yang sesuai dengan kebutuhan seseorang.

Menurut *Encyclopedia of Computer Science and Engineering*, banyak ilmuwan di bidang informasi menerima definisi standar bahwa informasi adalah data yang digunakan dalam pengambilan keputusan. Alasanya adalah bahwa informasi bersifat relatif, relatif terhadap situasi, relatif terhadap waktu saat keputusan diambil, juga relatif terhadap pembuat keputusan, dan bahkan juga relatif terhadap latar belakang pengambil keputusan.

Segala sesuatu yang dianggap penting pada suatu waktu bisa saja tidak berguna pada waktu yang lain. Ada kemungkinan pula bahwa sesuatu yang dianggap penting oleh seorang pengambil keputusan tidak dianggap penting bagi orang lain.

Dalam era reformasi, informasi menjadi sumber penting untuk melakukan pengambilan keputusan. Informasi dapat mengurangi ketidakpastian dan mempermudah pengambilan keputusan.

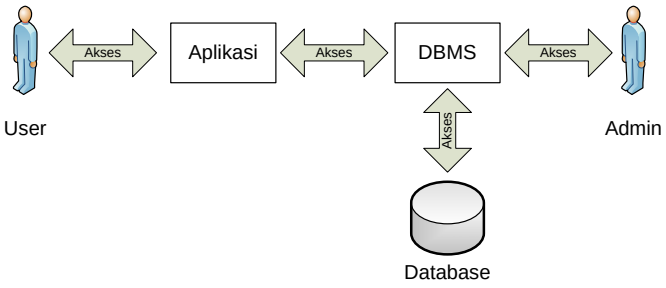
2.2.2 Database Management System (DBMS)

Database management system (DBMS) adalah suatu sistem pengelolaan berkas elektronik yang dapat menyimpan dan mengelola berbagai macam tipe data dalam suatu sistem data yang terintegrasi. DBMS sangat berguna bagi suatu sistem berbasis computer, bahkan dapat dikatakan menjadi tulang punggung. DBMS memungkinkan penyimpanan, pencarian, pengolahan, dan modifikasi data bisnis dengan cepat, aman dan efisien.

Manipulasi *database* meliputi pembuatan pernyataan (*query*) untuk mendapatkan informasi tertentu, melakukan pembaharuan atau penggantian (*update*) data, serta pembuatan laporan (*report*) dari data. Tujuan utama DBMS adalah untuk menyediakan tinjauan abstrak dari data bagi *user*. Jadi sistem menyembunyikan informasi mengenai bagaimana data disimpan dan dirawat, tetapi data tetap dapat diambil dengan efisien. Pertimbangan efisiensi yang digunakan adalah bagaimana merancang struktur data yang kompleks, tetapi tetap dapat digunakan oleh pengguna yang masih awam, tanpa mengetahui kompleksitas stuktur data. Dengan kata lain *database management system* (DBMS) merupakan suatu sistem *software* yang memungkinkan seorang user dapat

mendefinisikan, membuat, dan memelihara serta menyediakan akses terkontrol terhadap data.

Ketika seorang user ingin mengkases database dari sebuah system melalui sebuah aplikasi, aplikasi tersebut harus melalui DBMS sebelum sampai di database system, namun bagi administrator bisa langsung mengakses DBMS dan selanjutnya ke database itu sendiri tanpa melalui aplikasi. Ilustrasi diatas ditunjukkan pada Gambar 2.2 dibawah ini :



Gambar 2.2 Pengendalian akses pada database oleh DBMS

Kemampuan suatu paket DBMS adalah :

a) Pemeliharaan, penyimpanan, penarikan, dan pembaharuan data secara efisien,
b) Memudahkan akses data oleh *user*,
c) Pengendalian akses data bersama,
d) Dukungan terhadap transaksi data,
e) Pemulihan data yang rusak karena gagalnya sebuah transaksi data,
f) Keamanan data dan sistem otorisasi akses,
g) Menjaga integritas data.

Dalam suatu paket DBMS, akses terhadap *database* dilakukan oleh DBMS, Aplikasi dan *user* yang ingin mengakses data cukup

mendefinisikan data seperti apa yang perlu dibuat, disimpan, diolah, dan diambil.[3]

2.2.3 Data Warehouse

Beberapa pengertian dari beberapa ahli tentang definisi *Data Warehouse* [4]:

- Menurut W.H. Inmon dan Richard D.H., *data warehousing* adalah koleksi data yang mempunyai sifat berorientasi subjek, terintegrasi, *time-variant*, dan bersifat tetap dari koleksi data dalam mendukung proses pengambilan keputusan management.
- Menurut Vidette Poe, *data warehousing* merupakan *database* yang bersifat analisis dan *read only* yang digunakan sebagai fondasi dari sistem penunjang keputusan.
- Menurut Paul Lane, *data warehousing* merupakan *database* relasional yang didesain lebih kepada *query* dan analisa dari pada proses transaksi, biasanya mengandung *history* data dari proses transaksi dan bisa juga data dari sumber lainnya. *Data warehousing* memisahkan beban kerja analisis dari beban kerja transaksi dan memungkinkan organisasi menggabung/konsolidasi data dari berbagai macam sumber.

Dapat dikatakan bahwa *data warehousing* merupakan metode dalam perancangan *database*, yang menunjang DSS (*Decission Support System*) dan EIS (*Executive Information System*). Secara fisik

data warehousing adalah *database*, tapi perancangan *data warehousing* dan *database* sangat berbeda. Dalam perancangan *database* tradisional menggunakan normalisasi, sedangkan pada *data warehousing* normalisasi bukanlah cara yang terbaik.

Dari definisi-definisi yang dijelaskan tadi, dapat disimpulkan *data warehousing* adalah *database* yang saling bereaksi yang dapat digunakan untuk *query* dan analisisis, bersifat orientasi subjek, terintegrasi, timevariant, tidak berubah yang digunakan untuk membantu para pengambil keputusan.

2.2.4 Data Mining

2.2.4.1 Pengertian Data Mining

Data Mining adalah salah satu bidang yang berkembang pesat karena besarnya kebutuhan akan nilai tambah dari *database* skala besar yang makin banyak terakumulasi sejalan dengan pertumbuhan teknologi informasi. Definisi umum dari *data mining* itu sendiri adalah serangkaian proses untuk menggali nilai tambah berupa pengetahuan yang selama ini tidak diketahui secara manual dari suatu kumpulan data [5].

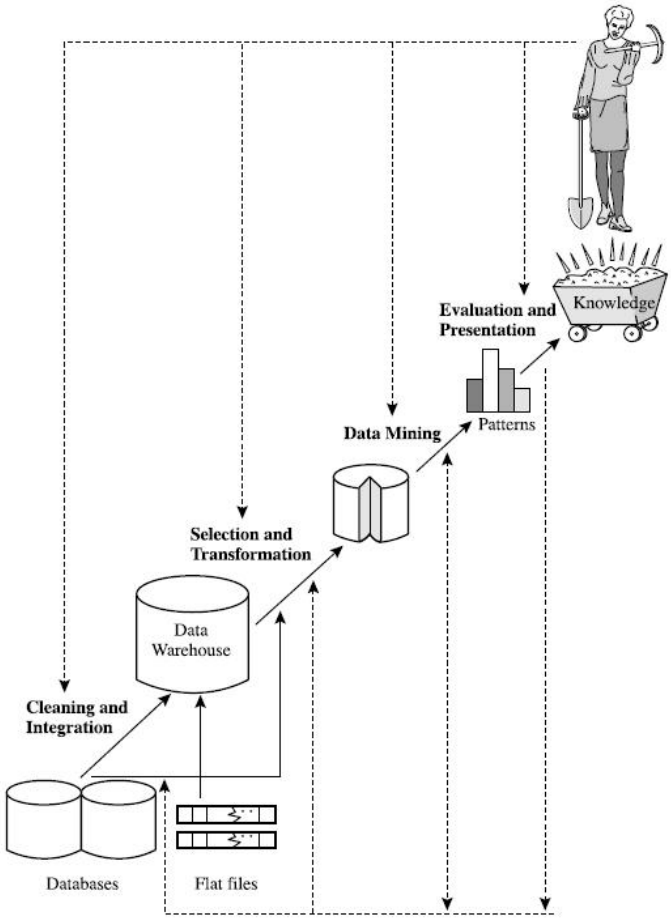
Data mining yang juga dikenal sebagai *Knowledge Discovery* (Frawley et al., 1992), merupakan salah satu bidang yang berkembang pesat karena besarnya kebutuhan akan nilai tambah dari *database* skala besar yang makin banyak terakumulasi sejalan dengan pertumbuhan teknologi informasi.

Secara umum, *data mining* dapat didefinisikan sebagai suatu rangkaian proses untuk menggali nilai tambah berupa ilmu pengetahuan yang selama ini tidak diketahui secara manual dari suatu kumpulan data.

Data mining terutama digunakan untuk mencari pengetahuan yang terdapat dalam *database* yang besar sehingga sering disebut *Knowledge Discovery in Databases* (KDD). Proses pencarian pengetahuan ini menggunakan berbagai teknik-teknik pembelajaran komputer (*machine learning*) untuk menganalisis dan mengekstraksikannya. Proses pencarian bersifat iteratif dan interaktif untuk menemukan pola atau model yang sahih, baru, bermanfaat, dan dimengerti.

2.2.4.2 Tahapan Data Mining

Dalam aplikasinya, data mining sebenarnya merupakan bagian dari *Knowledge Discovery in Database* (KDD), bukan sebagai teknologi yang utuh berdiri sendiri. *Data mining* merupakan suatu bagian langkah yang penting dalam KDD terutama berkaitan dengan ekstraksi dan perhitungan pola-pola dari data yang ditelaah. Seperti yang ditunjukkan pada gambar 2.3 dibawah ini :



Gambar 2.3. Tahapan data mining

a. Data cleaning

Tahapan ini dilakukan untuk menghilangkan data noise dan data yang tidak konsisten dengan tujuan akhir dari proses data mining.

b. Data itegration

Tahapan ini dilakukan untuk menggabungkan atau mengkombinasikan dari *multiple data source*.

c. Data selection

Yang dilakukan pada tahapan ini adalah memilih atau menyeleksi data apa saja yang yang relevan dan diperlukan dari *database*.

d. Data transformation

Untuk mentransformasikan data ke dalam bentuk yang lebih sesuai untuk di *mining.*

e. Data mining

Proses terpenting dimana metode tertentu diterapkan dalam database untuk menghasilkan data pattern.

f. Pattern evaluation

Untuk mengidentifikasi apakah *interenting patterns* yang didapatkan sudah cukup mewakili *knowledge* berdasarkan perhitungan tertentu.

g. Knowledge presentation

Untuk mempresentasikan *knowledge* yang sudah didapatkan dari *user*.

2.2.4.3 Arsitektur Data Mining

Pada umumnya sistem *data mining* terdiri dari komponen-komponen berikut ini :

a. *Database*, *data warehouse*, dan media penyimpanan informasi.

Media pada komponen ini dapat berupa *database*, *data warehouse*, *spreadsheet*, atau jenis media penampung lainnya. *Data cleaning* dan *data integration* dapat dilakukan pada data data tersebut.

b. *Database* atau *data warehouse server*

Database atau *data warehouse server* bertanggung jawab untuk menyediakan data yang relevan berdasarkan permintaan dari *user* pengguna *data mining.*

c. Basis Pengetahuan (*knowledge base*)

Merupakan basis pengetahuan yang digunakan sebagai panduan dalam pencarian pola.

d. *Data mining engine*

Yaitu bagian dari *software* yang menjalankan program berdasarkan algoritma yang ada.

e. *Pattern evaluation module*

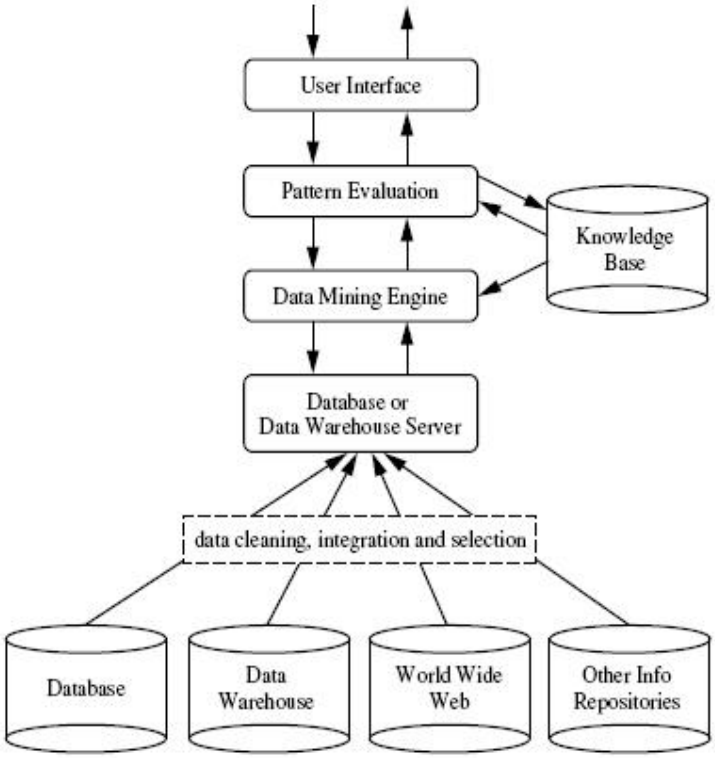
Yaitu bagian dari *software* yang berfungsi untuk menemukan *pattern* atau pola-pola yang terdapat di dalam *database* yang diolah sehingga nantinya proses *datamining* dapat menemukan *knowledge* yang sesuai.

f. *Graphical user interface*

Bagian ini merupakan sarana antara *user* dan sistem *data mining* untuk berkomunikasi, dimana *user* dapat berinteraksi dengan system melalui *data mining query*, untuk menyediakan informasi yang dapat membantu dalam pencarian *knowledge*.

Lebih jauh lagi, bagian ini mengijinkan *user* untuk melakukan *browsing* pada *database* dan *data warehouse*, mengevaluasi *pattern* yang telah dihasilkan, dan menampilkan *pattern* tersebut dengan tampilan yang berbeda-beda.

Gambar 2.4 dibawah ini menunjukkan Arsitektur Data Mining yang telah dijelaskan diatas :



Gambar 2.4. Arsitektur data mining.

2.2.5 Association Rules

Association rules adalah adalah teknik *mining* untuk menemukan hubungan di antara data atau bagaimana suatu kelompok data mempengaruhi suatu keberadan data yang lain. Metode ini mampu mengenali pola-pola tertentu di dalam kumpulan data yang

besar. Dalam *association rules*, suatu kelompok *item* dinamakan *itemset. Support* dari *itemset* X adalah persentase transaksi di D yang mengandung X, biasa ditulis dengan supp(X). Pencarian rules dilakukan melalui dua tahap yaitu pencarian frequent itemset dan penyusunan rules. Jika support suatu itemset lebih besar atau sama dengan minimum support σ, maka *itemset* tersebut dapat dikatakan sebagai *frequent itemset* atau *frequent pattern*; yang tidak memenuhi dinamakan *infrequent*. *Confidence* adalah nilai ukuran seberapa besar valid tidaknya suatu *association rules*. C*onfidence* suatu *rule R (X=>Y)* adalah proporsi dari semua transaksi yang mengandung baik X maupun Y dengan yang mengandung X, biasa ditulis sebagai *conf(R)*. Sebuah *association rule* dengan *confidence* sama atau lebih besar dari *minimum confidence* γ dapat dikatakan sebagai *valid association rule.*

Association rules dikenal juga sebagai salah satu teknik *data mining* yang menjadi dasar dari berbagai teknik *data mining* lainnya. Penting tidaknya suatu *association rules* dapat diketahui dengan dua parameter, yaitu *support* dan *confidence*. *Support* (nilai penunjang) adalah persentase kombinasi *item* terebut dalam database, sedangkan *confidence* (nilai kepastian) adalah kuatnya hubungan antar-item dalam *association rules*.

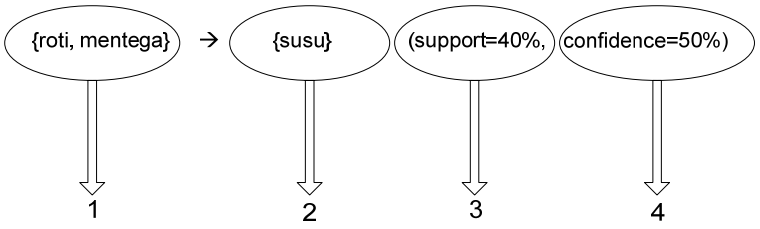
Association rules biasanya dinyatakan dalam bentuk:

{roti, mentega} → {susu} (support = 40%, confidence = 50%)

Rules diatas berarti "50% transaksi di database yang memuat item roti dan mentega juga memuat item susu. Sedangkan 40% dari seluruh transaksi yang ada di database memuat ketiga item terebut".

Association rules biasanya digunakan dalam analisis pasar atau yang lebih dikenal dengan *Market Basket Analysis,* terutama karena utilitas dan kejelasan dari hasil yang diperoleh. Jika kita mengasumsikan bahwa semua yang ada di alam semesta ini tersedia pada sebuah toko, maka setiap item yang ada harus memiliki variabel Boolean yang merepresentasikan ada atau tidaknya item terebut. Setiap transaksi kemudian dapat direpresentasikan dengan suatu nilai Boolean dari setiap variabel. Nilai Boolean tersebut dapat dianalisis sebagai pola belanja yang menunjukkan item-item yang sering dibeli secara bersamaan.

Gambar 2.5 berikut ini menunjukkan istilah-istilah yang digunakan untuk mempresentasikan setiap bagian dari *association rules* :



Gambar 2.5 Association Rules

Association Rules diatas dapat dibaca secara sederhana menjadi "seorang konsumen yang membeli roti dan mentega kemungkinan

50% juga membeli susu. Aturan ini cukup signifikan karena mewakili 40% dari catatan transaksi yang ada"

Keterangan istilah :

1: *Antecedent, left-hand side (LHS), body*

2: *Consequent, right-hand side (RHS), head*

3: *Support, frequency* (besar bagian data pada *left-hand side* dan *right-hand side* muncul secara bersamaan)

4: *Confidence, strength* (jika *left-hand side* muncul, seberapa besar *right-hand side* muncul).

2.2.5.1 Konsep Association Rules

a. Misalkan $I=\{i_1, i_2, i_3, \dots, i_m\}$ adalah sekumpulan item transaksi. Contoh : $I=\{i_1,i_2,i_3,i_4,i_5\}$. Secara umum notasi I dapat dibaca sebagai semua jenis *item* yang pernah dibeli dalam berbagai transaksi selama periode tertentu dengan *m* sebagai nilai yang menunjukkan berapa macam *item* yang ada.

b. Misalkan D adalah database transaksi dimana setiap transaksi T terdiri dari sekumpulan *item* sehingga $T \subseteq I$ (baca: T merupakan himpunan bagian atau subset dari semua anggota I). Jadi D dapat berupa D={T10,T20,T30,T40,T50,T60} dan sebagai contoh transaksi $T10=\{i_2,i_3,i_5\}$. Tanda $T \subseteq I$ berarti bahwa semua

anggota T harus merupakan sebagian atau seluruh dari semua anggota I. Disini dapat dilihat bahwa $\{i_2,i_3,i_5\} \subseteq \{i_1,i_2,i_3,i_4,i_5\}$.

c. Setiap transaksi T dikatakan mengandung A jika dan hanya jika $A \subseteq T$. *Association rules* merupakan sebuah implikasi dari bentuk $A \rightarrow B$, dimana $A \subseteq I$, $B \subseteq I$ dan $A \cap B \neq 0$.

d. *Rule* $A \rightarrow B$ berada pada sekumpulan transaksi D dengan *support* s, dimana s adalah persentase transaksi didalam D yang mengandung $(A \cup B)$ sehingga hal ini merupakan probabilitas $P(A \cup B)$. *Rule* $A \rightarrow B$ juga mempunyai *confidence* c didalam sekumpulan transaksi D, dimana c adalah persentase didalam D jika transaksi tersebut mengandung A juga mengandung B sehingga ini merupakan probabilitas bersyarat, P(B|A).

Metodologi dasar analisis asosiasi terbagi menjadi dua tahap :

1. Analisis pola frekuensi tinggi

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

$$Support(A) = \frac{Jumlah\ transaksi\ mengandung\ A}{Total\ transaksi}$$

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

$$Support\ (A,B) = P(A \cap B)$$

$$Support\ (A,B) = \frac{\sum Transaksi\ mengandung\ A\ dan\ B}{\sum Transaksi}$$

2. Pembentukan *Association Rules*

Setelah semua pola frekuensi tinggi ditemukan, barulah dicari *association rule* yang memenuhi syarat minimum untuk *confidence* dengan menghitung *confidence rule* A→B. Nilai confidence dari rule A→B diperoleh dari rumus berikut.

$$Confidence = P(B|A)$$

$$Confidence = \frac{\sum Transaksi\ mengandung\ A\ dan\ B}{\sum Transaksi\ mengandung\ A}$$

2.2.5.2 Algoritma Apriori

Algoritma Apriori adalah algoritma yang digunakan untuk menghasilkan aturan asosiasi dengan pola "if…then". Algoritma Apriori menggunakan pendekatan iteratif yang dibuat dengan *level-wise search*, dimana *k-itemset* produk digunakan untuk mengeksplorasi *(k+1)-itemset* produk atau *(k-1)-itemset*.

Notasi yang digunakan dalam algoritma apriori antara lain:

a. C_k adalah kandidat *k-itemset*, dimana *k* menunjukkan jumlah pasangan *item*.

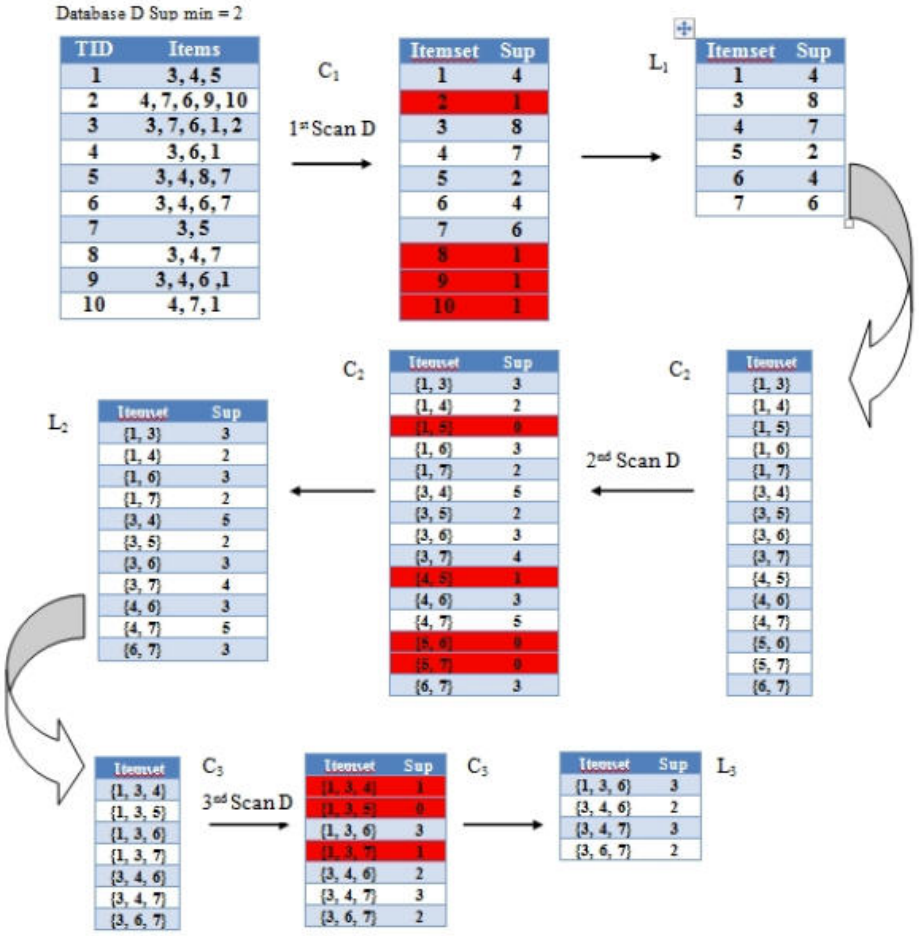
b. L_k adalah large *k-itemset*.

c. D adalah database transaksi dimana |D| adalah banyaknya transaksi.

Tahapan yang dilakukan algoritma apriori untuk membangkitkan *large itemset* adalah sebagai berikut :

a. Menelusuri seluruh *record* di database dan menghitung *support count* dari setiap *item*. Ini adalah kandidat *1-itemset*, C_1.
b. *Large 1-itemset* L_1 dibangun dengan menyaring C_1 dengan *support count* yang lebih besar atau sama dengan *minimum support* untuk dimasukkan kedalam L_1.
c. Untuk membangun L_2, algoritma apriori menggunakan proses join untuk menghasilkan C_2.
d. Dari C_2, *2-itemset* yang memiliki *support count* yang lebih besar atau sama dengan *minimum support* akan disimpan ke L_2.
e. Proses ini diulang sampai tidak ada lagi kemungkinan *k-itemset*.

Contoh tahapan pembangkitan C_1, L_1, C_2, L_2, C_3, L_3 ditunjukkan pada Gambar 2.6 dibawah ini:



Gambar 2.6 Proses pembuatan frequent pattern menggunakan algoritma Apriori.

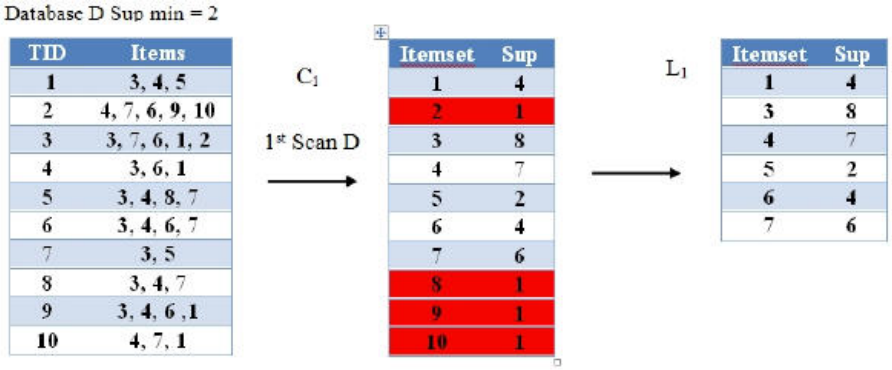
Ada dua proses utama yang dilakukan algoritma apriori yaitu

a. *Join* (Penggabungan) : untuk menemukan L_k, C_k dibangkitkan dengan melakukan proses *join* L_{k-1} dengan dirinya sendiri, $C_k=L_{k-1}$ x L_{k-1} lalu anggota C_k diambil hanya yang terdapat dalam L_{k-1}.

b. *Prune* (Pemangkasan): menghilangkan anggota C_k yang tidak *frequent* dan menghilangkan anggota C_k yang

memiliki *support count* lebih kecil dari *minimum support* agar tidak dimasukkan ke dalam L_k.

Proses pembentukan C_1 tidak ada proses *join*, hanya ada proses *prune* dari *scan* D. Proses *pruning* pada C_1 adalah menghilangkan item yang nilai *support*nya kurang dari minimum support dan dihasilkanlah L_1



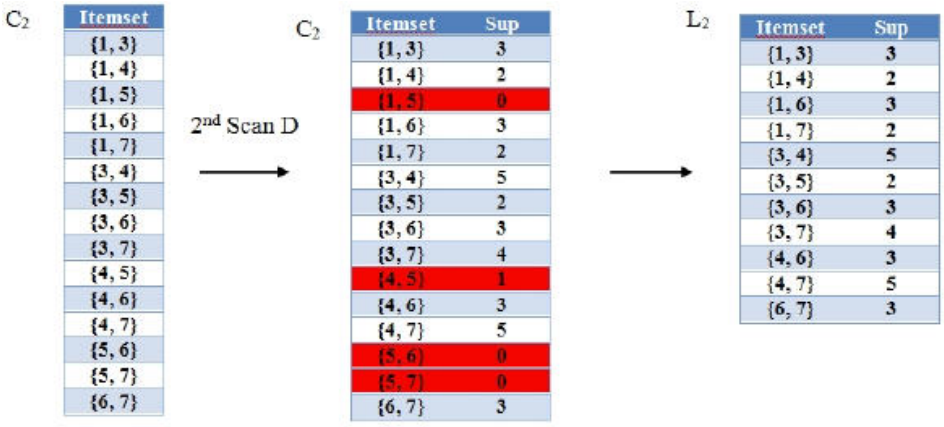
Gambar 2.7 proses pembentukan L_1

Setelah didapat L_1, dilakukan proses *join* L_1 x L_1 untuk menghasilkan C_2. Setelah C_2 terbentuk, dilakukan proses *pruning* pada C_2 dengan menghilangkan *itemset* yang nilai *support* nya kurang dari *minimum support* dan dihasilkanlah L_2. Proses *join* dan *prune* untuk L_2 dapat dilihat dibawah ini :

a. *Join* $C_2 = L_1$ x L_1 = {{1},{3},{4},{5},{6},{7}} x {{1},{3},{4},{5},{6},{7}} = {{1,3}.{1,4},{1,5},{1,6},{1,7},{3,4},{3,5},{3,6},{3,7},{4,5},{4,6},{4,7},{5,6},{5,7},{6,7}}.

b. Untuk proses *pruning* atau pemangkasan pada C_2, hanya dipangkas *itemset* yang tidak memenuhi *minimum support*.

Sedangkan untuk pemangkasan *itemset* yang tidak *frequent*, biasanya pada scan D kedua ini lolos dari pemangkasan. Setelah di *pruning*, maka di dapat L_2 = {{1,3},{1,4},{1,6}, {1,7},{3,4},{3,5},{3,6},{3,7},{4,6},{4,7},{6,7}}.



Gambar 2.8 Proses Pembentukan L_2

Setelah didapat L_2, dilakukan proses *join* L_2 x L_2 untuk menghasilkan C_3. Setelah C_3 terbentuk, dilakukan proses *pruning* pada C_3 dengan menghilangkan *itemset* yang tidak *frequent* dan yang nilai *support*nya kurang dari *minimum support* dan dihasilkanlah L_3. Proses *join* dan *prune* untuk L_3 dapat dilihat dibawah ini :

a. *Join* $C_3 = L_2$ x L_2 = {{1,3},{1,4},{1,6},{1,7},{3,4}, {3,5},{3,6},{3,7},{4,6},{4,7},{6,7}} x {{1,3},{1,4},{1,6} ,{1,7},{3,4},{3,5},{3,6},{3,7},{4,6},{4,7},{6,7}} = {{1,3,4},{1,3,5},{1,3,6},{1,3,7},{3,4,5},{3,4,6},{3,4,7}, {3,5,6},{3,5,7},{3,6,7},{4,5,6},{4,5,7},{5,6,7}}.

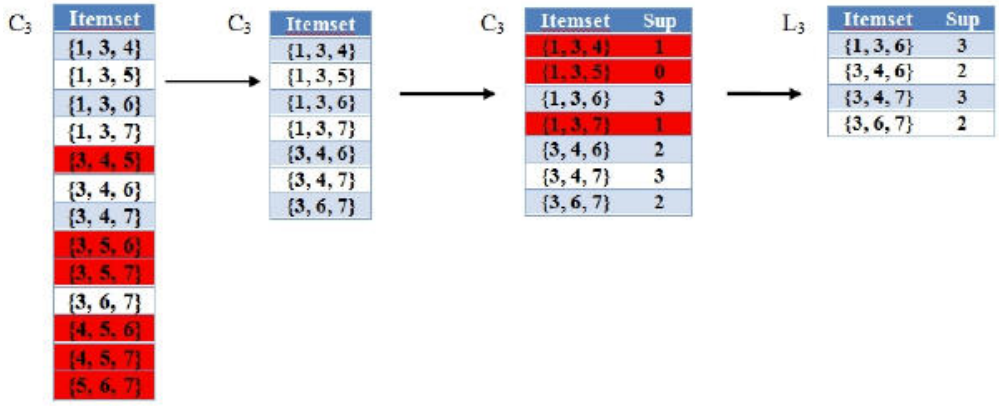
b. Pada proses *pruning* C_3, terjadi pemangkasan *itemset* yang tidak *frequent*. Cara pemeriksaan *itemset* yang tidak *frequent* adalah mengurai kembali *itemset* yang didapat menjadi *itemset-itemset* L_2 (berbentuk dua kombinasi item) pembentuk *itemset* tersebut dan memeriksa kembali apakah *itemset-itemset* tersebut ada didalam L_2. Jika seluruh atau salah satu dari *itemset* tersebut tidak ada di L_2, maka *itemset* tersebut dihilangkan dari C_3. Proses pemeriksaan *itemset* C_3 yang tidak *frequent* adalah sebagai berikut :

1. *2-item subsets* dari {1,3,4} adalah {1,3},{1,4} dan {3,4}. Semua *2-item subsets* dari {1,3,4} terdapat di L_2. Sehingga {1,3,4} masuk dalam C_3.
2. *2-item subsets* dari {1,3,5} adalah {1,3},{1,5} dan {3,5}. Semua *2-item subsets* dari {1,3,5} terdapat di L_2. Sehingga {1,3,5} masuk dalam C_3.
3. *2-item subsets* dari {1,3,6} adalah {1,3},{1,6} dan {3,6}. Semua *2-item subsets* dari {1,3,6} terdapat di L_2. Sehingga {1,3,6} masuk dalam C_3.
4. *2-item subsets* dari {1,3,7} adalah {1,3},{1,7} dan {3,7}. Semua *2-item subsets* dari {1,3,7} terdapat di L_2. Sehingga {1,3,7} masuk dalam C_3.

5. *2-item subsets* dari {3,4,5} adalah {3,4},{3,5} dan {4,5} tidak terdapat pada L_2, sehingga tidak *frequent*. Oleh sebab itu {3,4,5} dihilangkan dari C_3.
6. *2-item subsets* dari {3,4,6} adalah {3,4},{3,6} dan {4,6}. Semua *2-item subsets* dari {3,4,6} terdapat di L_2. Sehingga {3,4,6} masuk dalam C_3.
7. *2-item subsets* dari {3,4,7} adalah {3,4},{3,7} dan {4,7}. Semua *2-item subsets* dari {3,4,7} terdapat di L_2. Sehingga {3,4,7} masuk dalam C_3.
8. *2-item subsets* dari {3,5,6} adalah {3,5},{3,6} dan {5,6} tidak terdapat pada L_2, sehingga tidak *frequent*. Oleh sebab itu {3,5,6} dihilangkan dari C_3.
9. *2-item subsets* dari {3,5,7} adalah {3,5},{3,7} dan {5,7} tidak terdapat pada L_2, sehingga tidak *frequent*. Oleh sebab itu {3,5,7} dihilangkan dari C_3.
10. *2-item subsets* dari {3,6,7} adalah {3,6},{3,7} dan {6,7}. Semua *2-item subsets* dari {3,6,7} terdapat di L_2. Sehingga {3,6,7} masuk dalam C_3.
11. *2-item subsets* dari {4,5,6} adalah {4,5},{4,6} dan {4,6} tidak terdapat pada L_2, sehingga tidak *frequent*. Oleh sebab itu {4,5,6} dihilangkan dari C_3.

12. *2-item subsets* dari {4,5,7} adalah {4,5},{4,7} dan {5,7} tidak terdapat pada L_2, sehingga tidak *frequent*. Oleh sebab itu {4,5,7} dihilangkan dari C_3.
13. *2-item subsets* dari {5,6,7} adalah {5,6},{5,7} dan {6,7} tidak terdapat pada L_2, sehingga tidak *frequent*. Oleh sebab itu {5,6,7} dihilangkan dari C_3.

Setelah *itemset* yang tidak *frequent* dihilangkan, maka didapat C_3 = {{1,3,4},{1,3,5},{1,3,6},{1,3,7},{3,4,6}, {3,4,7} ,{3,6,7}}. Selanjutnya *itemset* yang tidak memenuhi *minimum support* yang akan dihilangkan dari C_3. Setelah dilakukan *pruning*, maka didapat L_3 = {{1,3,6},{3,4,6},{3,4,7},{3,6,7}}.



Gambar 2.9 Proses Pembentukan L_3

Setelah didapat L_3, dilakukan proses *join* L_3 x L_3 untuk menghasilkan C_4. Setelah C_4 terbentuk, dilakukan proses *pruning* pada C_4 dengan menghilangkan *itemset* yang tidak *frequent* dan yang nilai *support*nya kurang dari *minimum*

support dan dihasilkanlah L_4. Proses *join* dan *prune* untuk L_4 dapat dilihat dibawah ini :

a. *Join* $C_4 = L_3 \times L_3 = \{\{1,3,6\},\{3,4,6\},\{3,4,7\},\{3,6,7\}\} \times \{\{1,3,6\},\{3,4,6\},\{3,4,7\},\{3,6,7\}\} = \{\{1,3,4,6\},\{1,3,4,7\},\{3,4,6,7\}\}$.

b. Seperti pada proses *pruning* pada C_3, di C_4 pun terjadi *pruning*, proses pemeriksaan itemset C_4 yang tidak *frequent* adalah sebagai berikut :

   1. *3-item subsets* dari {1,3,4,6} adalah {1,3,4},{1,3,6},{1,4,6} dan {3,4,6}. {1,3,4} dan {1,4,6} tidak terdapat di L_3, sehingga tidak *frequent*. Oleh sebab itu {1,3,4,6} dihilangkan dari C_4.
   2. *3-item subsets* dari {1,3,4,7} adalah {1,3,4},{1,3,7},{1,4,7} dan {3,4,7}. {1,3,4},{1,3,7} dan {1,4,7} tidak terdapat di L_3, sehingga tidak *frequent*. Oleh sebab itu {1,3,4,7} dihilangkan dari C_4.
   3. *3-item subsets* dari {3,4,6,7} adalah {3,4,6},{3,4,7},{3,6,7} dan {4,6,7}. {4,6,7} tidak terdapat di L_3, sehingga tidak *frequent*. Oleh sebab itu {3,4,6,7} dihilangkan dari C_4.

C_4

Itemset
{1, 3, 4, 6}
{1, 3, 4, 7}
{3, 4, 6, 7}

Gambar 2.10. kombinasi C4 yang gagal

Karena setelah dilakukan *pruning itemset* yang tidak *frequent* dihilangkan , maka untuk kasus ini didapat *larger itemset sampai* L_3.

2.2.6 *Entity Relationship Diagram* (ERD)

Basis data Relasional adalah kumpulan dari relasi-relasi yang mengandung seluruh informasi berkenaan suatu entitas/ objek yang akan disimpan di dalam *database*. Tiap relasi disimpan sebagai sebuah file tersendiri. Perancangan basis data merupakan suatu kegiaatan yang setidaknya bertujuan sebagai berikut:

a. Menghilangkan redundansi data.
b. Meminimumkan jumlah relasi di dalam basis data.
c. Membuat relasi berada dalam bentuk normal, sehingga dapat meminimumkan permasalahan berkenaan dengan penambahan, pembaharuan dan penghapusan.

ERD adalah suatu pemodelan dari basis data relasional yang didasarkan atas persepsi di dalam dunia nyata, dunia ini senantiasa terdiri dari sekumpulan objek yang saling berhubungan antara satu dengan yang lainnya. Suatu objek disebut *entity* dan hubungan yang dimilikinya disebut *relationship*. Suatu *entity* bersifat unik dan memiliki atribut sebagai pembeda dengan *entity* lainnya.

Contoh : *entity* Mahasiswa, mempunyai atribut nama, umur, alamat, dan nim.

Diagram E-R terdiri dari:

a. Kotak persegi panjang, menggambarkan himpunan entitas.
b. Elips, menggambarkan atribut-atribut entitas.
c. Diamon, menggambarkan hubungan antara himpunan entitas
d. Garis, yang menghubungkan antar objek dalam diagram E-R

E-R Diagram merupakan suatu bahasa pemodelan dimana posisinya dapat dianalogikan dengan *story board* dalam industri film, *blue print* arsitektur suatu bangunan, miniatur, dan lain-lain. Dalam praktiknya, membangun suatu sistem terlebih dahulu dilakukannya suatu perencanaan. Pemodelan merupakan suatu sub bagian dari perencanaan secara keseluruhan sebagai salah satu upaya *feedback* evaluasi perampungan suatu perencanaan. E-R Diagram sebagai suatu pemodelan setidaknya memiliki beberapa karakteristik dan manfaat sebagai berikut:

a. Memudahkan untuk dilakukannya analisis dan perubahan sistem sejak dini, bersifat murah dan cepat
b. Memberikan gambaran umum akan sistem yang akan di buat sehingga memudahkan *developer.*
c. Menghasilkan dokumentasi yang baik untuk *client* sebagai bahan diskusi dengan bentuk E-R Diagram itu sendiri, dan
d. Kamus data bagi bagi para pengembang *database*.

Struktur dari E-R Diagram secara umum ialah terdiri dari:

a. Entitas merupakan objek utama yang informasi akan disimpan, biasanya berupa kata benda, ex; Mahasiswa, Dosen, Nasabah, Mata Kuliah, Ruangan, dan lain-lain. Objek dapat berupa benda nyata maupun abstrak.
b. Atribut merupakan deskripsi dari objek yang bersangkutan
c. *Relationship* merupakan suatu hubungan yang terjalin antara dua entitas yang ada.

Kardinalitas Relasi ERD yang mempersentasikan suatu basis data relasional senantiasa memiliki relasi-relasi dari sejumlah entitas yang dapat ditentukan banyaknya. Banyaknya suatu relasi yang dimiliki oleh suatu relasi entitas disebut derajat relasi. Derajat relasi maksimum disebut dengan kardinalitas sedangkan derajat minimum disebut dengan modalitas. Kardinalitas yang terjadi diantara dua himpunan entitas (misal A dan B) dapat berupa :

a. *One to One*, satu *record* dipetakan dengan satu *record* di entitas lain. Contoh: satu nasabah punya satu *account*.
b. *One to Many*, Satu *record* dapat dipetakan menjadi beberapa *record* di entitas lain. Contoh: satu nasabah dapat punya lebih dari satu *account*.

Many to Many, Beberapa *record* dapat dipetakan menjadi beberapa *record* di entitas lain. Contoh: satu nasabah dapat memiliki lebih dari satu *account*. Satu *account* dapat dimiliki lebih dari satu nasabah *(join account).*

2.2.7 *Data Flow Diagram* (DFD)

DFD adalah suatu model logika data atau proses yang dibuat untuk menggambarkan dari mana asal data dan kemana tujuan data yang keluar dari sistem, dimana data disimpan, proses apa yang menghasilkan data tersebut dan interaksi antara data yang tersimpan dan proses yang dikenakan pada data tersebut.

DFD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir atau dimana data tersebut akan disimpan.

DFD merupakan alat yang digunakan pada metodologi pengembangan sistem yang terstruktur. Kelebihan utama pendekatan aliran data, yaitu :

a. Kebebasan dari menjalankan implementasi teknis sistem.
b. Pemahaman lebih jauh mengenai keterkaitan satu sama lain dalam sistem dan subsistem.
c. Mengkomunikasikan pengetahuan sistem yang ada dengan pengguna melalui diagram aliran data.
d. Menganalisis sistem yang diajukan untuk menentukan apakah data-data dan proses yang diperlukan sudah ditetapkan.

Disamping itu terdapat kelebihan tambahan, yaitu :

1. Dapat digunakan sebagai latihan yang bermanfaat bagi penganalisis, sehingga bisa memahami dengan lebih baik keterkaitan satu sama lain dalam sistem dan subsistem.
2. Membedakan sistem dari lingkungannya dengan menempatkan batas-batasnya.
3. Dapat digunakan sebagai suatu perangkat untuk berinteraksi dengan pengguna.
4. Memungkinkan penganalisis menggambarkan setiap komponen yang digunakan dalam diagram.

DFD terdiri dari *context diagram* dan diagram rinci *(DFD Levelled)*. *Context diagram* berfungsi memetakan model lingkungan (menggambarkan hubungan antara entitas luar, masukan dan keluaran sistem), yang direpresentasikan dengan lingkaran tunggal yang mewakili keseluruhan sistem. *DFD levelled* menggambarkan sistem sebagai jaringan kerja antara fungsi yang berhubungan satu sama lain dengan aliran dan penyimpanan data, model ini hanya memodelkan sistem dari sudut pandang fungsi.

Dalam *DFD levelled* akan terjadi penurunan level dimana dalam penurunan level yang lebih rendah harus mampu merepresentasikan proses tersebut ke dalam spesifikasi proses yang jelas. Jadi dalam *DFD levelled* bisa dimulai dari DFD level 0 kemudian turun ke DFD level 1 dan seterusnya. Setiap penurunan

hanya dilakukan bila perlu. Aliran data yang masuk dan keluar pada suatu proses di level x harus berhubungan dengan aliran data yang masuk dan keluar pada level x+1 yang mendefinisikan proses pada level x tersebut. Proses yang tidak dapat diturunkan/dirinci lagi dikatakan primitif secara fungsional dan disebut sebagai proses primitif.

2.2.8 Kamus Data (*Data Dictionary*)

Kamus Data merupakan katalog (tempat penyimpanan) dari elemen-elemen yang berada dalam satu sistem. Kamus data mempunyai fungsi yang sama dalam pemodelan sistem dan juga berfungsi membantu pelaku sistem untuk mengerti aplikasi secara detail, dan me-reorganisasi semua elemen data yang digunakan dalam sistem sehingga pemakai dan penganalisa sistem punya dasar pengertian yang sama tentang masukan, keluaran, penyimpanan dan proses.

2.2.9 Software Pendukung

Diperlukan beberapa *software* pendukung untuk menjalankan aplikasi tersebut antara lain :

2.2.9.1 Delphi

Delphi adalah suatu program berbasis bahasa Pascal yang berjalan dalam lingkungan Windows. Delphi telah memanfaatkan suatu teknik pemrograman yang disebut RAD yang telah membuat pemrograman menjadi lebih mudah. Delphi adalah suatu bahasa pemrograman yang

telah memanfaatkan metode pemrograman *Object Oriented Programming* (OOP). Secara ringkas, *object* adalah suatu komponen yang mempunyai bentuk fisik dan biasanya dapat dilihat (visual). *Object* biasanya dipakai untuk melakukan tugas tertentu dan mempunyai batasan-batasan tertentu. Sedangkan bahasa pemrograman secara singkat dapat disebut sebagai sekumpulan teks yang mempunyai arti tertentu dan disusun dengan aturan tertentu serta untuk menjalankan tugas tertentu.

Khusus untuk pemrograman *database*, Delphi menyediakan *object* yang sangat kuat, canggih, dan lengkap, sehingga memudahkan pemrogram dalam merancang, membuat dan menyelesaikan aplikasi *database* yang diinginkan. Selain itu, Delphi juga dapat menangani data dalam berbagai format *database*, misalnya format MS-Access, SyBase, Oracle, FoxPro, Informix, DB2, MySQL dan lain-lain. Format *database* yang dianggap asli dari Delphi adalah Paradox dan dBase.

2.2.9.2 MySQL

SQL (*Structured Query Language*) adalah bahasa standar yang digunakan untuk mengakses *server database*. Semenjak tahun 70-an bahasa ini telah dikembangkan oleh IBM, yang kemudian diikuti dengan adanya Oracle, Informix dan Sybase. Dengan menggunakan SQL, proses akses *database* menjadi lebih *user-friendly* dibandingkan dengan misalnya dBase ataupun Clipper yang masih menggunakan perintah – perintah pemrograman murni.

MySQL adalah sebuah *server database* SQL *multiuser* dan *multi-threaded*. SQL sendiri adalah salah satu bahasa *database* yang paling populer di dunia. Implementasi program *server database* ini adalah program daemon 'mysqld' dan beberapa program lain serta beberapa pustaka. Sebagaimana *database* sistem yang lain, dalam SQL juga dikenal hierarki server dengan database-database. Tiap-tiap *database* memiliki tabel-tabel. Tiap-tiap tabel memiliki field-field. Umumnya informasi tersimpan dalam tabel – tabel yang secara logik merupakan struktur 2 dimensi terdiri atas baris dan kolom. Field-field tersebut dapat berupa data seperti int, realm char, date, time dan lainnya. SQL tidak memiliki fasilitas pemrograman yang lengkap, tidak ada *looping* ataupun percabangan. Sehingga untuk menutupi kelemahan ini perlu digabung dengan bahasa pemrograman semisal Pascal. MySQL sering digunakan sebagai SQL server karena berbagai kelebihannya, antara lain :

1. Source MySQL dapat diperoleh dengan mudah dan gratis.
2. Sintaksnya lebih mudah dipahami dan tidak rumit.
3. Pengaksesan database dapat dilakukan dengan mudah.