Bab 2

TINJAUAN PUSTAKA

2.1 Profil Perusahaan

2.1.1 Sejarah Instansi

PT Kharisma Trijaya Mandiri yang populer dengan nama Kharismachem. Sudah 5 tahun usia PT Kharisma Trijaya Mandiri yang dikelola sebagai perusahaan keluarga. Walaupun sebagai perusahaan keluarga, PT Kharisma Trijaya Mandiri telah menjelma menjadi perusahaan yang berkembang pesat.

Sejak berdirinya tahun 2007 di Bandung (Indonesia), PT Kharisma Trijaya Mandiri telah memfokuskan usahanya pada manufaktur bahan kimia tekstil. Tahun 2007, Udrawan Gunawan menjadi direktur PT Kharisma Trijaya Mandiri. Sebagai direktur beliau telah memimpin perusahaan hingga berada pada level penting dan diperhitungkan sampai saat ini.

PT Kharisma Trijaya Mandiri berlokasi di jalan Siliwangi Baleendah KM 13 no. 374 kabupaten Bandung. PT Kharisma Trijaya Mandiri meluaskan sayap produksi dan pemasarannya di Jawa Barat sejak berdirinya PT Kharisma Trijaya Mandiri yaitu pada tahun 2007. PT Kharisma Trijaya Mandiri adalah satu-satunya agen resmi dari negara Cina, India dan Korea sehingga kualitasnya pun tidak diragukan dan itulah yang menjadi salah satu faktor cepatnya perkembangan PT Kharisma Trijaya Mandiri.

2.1.2 Logo Instansi

Berikut adalah logo PT Kharisma Trijaya Mandiri.



Gambar 2. 1 Logo PT Kharisma Trijaya Mandiri

2.1.3 Visi dan Misi

Adapun visi dan misi yang dimiliki PT Kharisma Trijaya Mandiri yaitu sebagai berikut.

Visi :

Menjadi pilihan utama untuk bisnis bahan kimia pembantu proses tekstil. Pilihan utama adalah pilihan terbaik, menjadi yang terbaik berarti bahwa PT Kharisma Trijaya Mandiri berjanji untuk bekerja keras setiap hari.

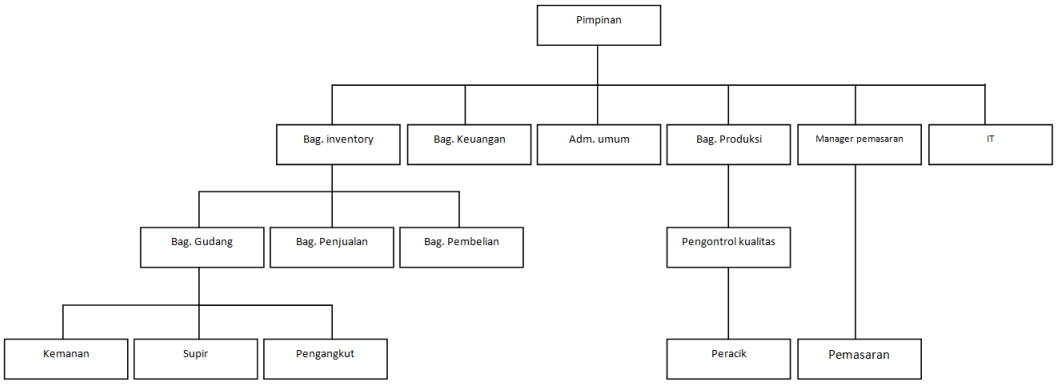
Misi :

Untuk dikenal sebagai pembuat zat kimia tekstil berkualitas tinggi yang memberikan nilai tambah (bagi pelanggan kami) dengan menggunakan seluruh talenta anggota-anggota dan pekerja-pekerja kami. Berdedikasi untuk pertumbuhan dan pengembangan yang sinambung ari bisnis dan anggota kami melalui tindakan yang menempatkan pelanggan kami pada tempat yang paling utama.

2.1.4 Struktur Organisasi dan *Job description*

Struktur organisasi merupakan susunan yang terdiri dari fungsi-fungsi dan hubungan-hubungan yang ada dalam suatu intansi yang menyatakan keseluruhan kegiatan untuk mencapai sasaran intansi.

2.1.4.1 Struktur Organisasi PT Kharisma Trijaya Mandiri



Gambar 2. 2 Strukur Organisasi PT Kharisma Trijaya Mandiri

2.1.4.2 *Job description*

Adapun *job description* dari PT Kharisma Trijaya Mandiri itu sendiri adalah sebagai berikut :

1. Pimpinan
    a. Bertanggung jawab atas perusahaan
    b. Memberikan ijin atas jalannya perusahaan
    c. Mengelola perusahaan
    d. Mengendalikan kondisi perusahaan
    e. Memberikan fasilitas untuk perusahaan
    f. Memberikan motivasi
    g. Melihat laporan dari semua bagian

2. Bagian inventori
    a. Memantau persediaan bahan kimia
    b. Melihat laporan dari semua bagian *inventory*.
3. Bagian keuangan
    a. Mengatur keuangan
    b. Menangani masalah keuangan
    c. Mengendalikan akuntansi perusahaan
    d. Melaporkan keuangan pada direktur
4. Administrator umum
    a. Menjembatani atasan dan bawahan
    b. Menerima panggilan telepon
    c. Mengelola data pegawai
5. Bagian produksi
    a. Memproduksi bahan kimia
    b. Menentukan kualitas produk
6. Manajer pemasaran
    a. Membuat strategi pemasaran
7. Bagian gudang
    a. Meng*input* data barang
    b. Mengelola persediaan bahan kimia
    c. Memberikan in*Form*asi mengenai stok
8. Bagian penjualan
    a. Mengelola kegiatan penjualan

b. Memberikan in*Form*asi mengenai penjualan

9. Bagian pembelian
    a. Mengelola kegiatan pembelian
    b. Memberikan in*Form*asi mengenai pembelian
10. Pengontrol kualitas
    a. Mengontrol dan menentukan kelayakan produk
11. Keamanan
    a. Menjaga keamanan perusahaan terutama gudang
12. Supir
    a. Pengirim produk pada pelanggan
    b. Pengirim faktur dan surat jalan
13. Pengangkut
    a. Mengangkut barang keluar masuk gudang
14. Peracik
    a. Membuat produk yang dibutuhkan
15. Pemasaran
    a. Promosi produk
    b. Mencari pelanggan
16. IT
    a. Menjadi operator IT

2.2 Konsep Persediaan

Inventory secara umum dapat dibagi menjadi empat bagian proses bisnis utama, yaitu :

1. *Purchasing*

   Purchasing atau pembelian barang merupakan unit bisnis yang melayani transaksi pembelian barang. Proses bisnis *purchasing* terbagi menjadi beberapa sub proses bisnis antara lain: *Purchase Request, Purchase Order, Receiving, Purchase Return.*

2. *Sales*

   Sales atau penjualan barang merupakan unit bisnis yang melayani transaksi penjualan barang kepada pelanggan. Proses bisnis *sales* terbagi menjadi sub proses bisnis antara lain: *Sales* Order, *Sales* Return.

3. *Distribution*

   Distribution atau pengiriman merupakan transaksi pendistribusian barang kepada pelanggan maupun pendistribusian barang antar gudang. Proses bisnis *distribution* terbagi menjadi beberapa sub proses bisnis antara lain: *Shipment, Load Sheet, Location Transfer.*

4. *Controlling stock*

   Controlling stock merupakan proses pengecekkan kesesuaian stok barang pada gudang dengan sistem. Proses bisnis *controlling stock* terbagi menjadi beberapa sub proses bisnis antara lain: *Inventory Stock Take, In Debit Adjusment, In Credit Adjusment, Item Balance, Valuation Item.*

2.2.1 Pengertian Persediaan

Inventory atau persediaan merupakan simpanan material yang berupa bahan mentah, barang dalam proses dan barang ajdi. Dari sudut pandang sebuah

perusahaan maka persediaan adalah investasi modal yang dibutuhkan untuk menyimpan material pada kondisi tertentu [2].

Barang persediaan adalah sejumlah material yang disimpan dan dirawat menurut aturan tertentu dalam tempat persediaan agar selalu dalam keadaan siap pakai dan ditatausahakan dalam buku perusahaan [3].

Sediaan (*inventory*) adalah sejumlah bahan atau barang yang tersedia untuk digunakan sewaktu-waktu di masa yang akan datang. Sediaan terjadi apabla jumlah bahan atau barang yang diadakan (dibeli atau dibuat sendiri) lebih besar daripada jumlah yang digunakan (dijual atau diolah sendiri).

2.2.2 Penjualan

Penjualan adalah kegiatan pemasok atau penjual untuk menjalankan bisnis, guna memenuhi kebutuhan pasar, menetapkan harga, mendistribusikan serta mempromosikan melalui proses pertukaran agar memuaskan konsumen dalam mencapai tujuan perusahaan, sedangkan menurut Jogiyanto, penjualan adalah transaksi perubahan nilai barang menjadi nilai uang atau nilai piutang [4].

2.2.3 Pembelian

Pembelian adalah proses untuk mendapatkan suatu barang dengan menukar sesuatu, tetapi prosesnya mempunyai peran dan lingkup serta tujuan dan strategi untuk tujuan tertentu [5], pendapat lain pembelian adalah menerima suatu barang dengan menyerahkan sesuatu kepada orang lain dengan menerima suatu barang dengan menyerahkan sesuatu kepada orang lain dengan memberikan sesuatu sebagai sayarat/perjanjian yang telah disepakati [6].

2.2.4 Manajemen Persediaan

Pengendalian terhadap persediaan atau *inventory control* adalah aktifitas mempertahankan jumlah persediaan pada tingkat yang dikehendaki. Pada produk barang, pengendalan *inventory* ditekan pada pengendalian material. Pada produk jasa, pengendalian *inventory* ditekan pada pengendalian material. Pada produk jasa, pengendalian diutamakan sedikit pada material dan banyak pada jasa pasokan karena konsumsi sering kali bersamaan dengan pengadaan jasa sehingga tidak memerlukan persediaan [2].

Harus ada keseimbangan antara mempertahankan tingkat *inventory* yang tepat dengan pengaruh keuangan minimum terhadap pelanggan. Jika investasi sangat besar akan mengakibatkan biaya modal yang sangat besar, sehingga akan mengakibatkan juga biaya operasi yang tinggi [2].

Pengendalian tingkat persediaan bertujuan mencapai efisiensi dan efektifitas optimal dalam penyediaan material. Dalam pengertian di atas, usaha yang perlu dilakukan dalam manajemen persediaan secara garis besar sebagai berikut:

1. Menjamin terpenuhinya kebutuhan operasi
2. Membatasi nilai seluruh investasi
3. Membatasi jenis dan jumlah material
4. Memanfaatkan seoptimal mungkin material yang ada

2.2.5 Prinsip Manajemen Persediaan

Mengenai persediaan barang ada konsep pengelolaaan yang harus dianut, yaitu penentuan jumlah dan jenis barang yang disimpan dalam

persediaan haruslah sedemikian rupa sehingga produksi dan operasi perusahaan tidak terganggu, tetapi di lain pihak sekaligus harus dijaga agar biaya investasi yang timbul dari penyediaan barang tersebut seminimal mungkin.

Prinsip tersebut memang selaras dengan prinsip ekonomi yaitu, menghasilkan keluaran tertentu dengan biaya seminimal mungkin atau dengan biaya tertentu menghasilkan keluaran semaksimal mungkin. Jika melihat prinsip persediaan tersebut, maka jelas bahwa diperlukan perpaduan antara dua hal yang sangat bertolak belakang [3].

2.2.6 Tujuan Persediaan

Tujuan utama dari persediaan bahan baku adalah menghubungkan pemasok dengan pabrik [2]. Ada tiga alasan mengapa persediaan diperlukan:

1. Menghilangkan pengaruh ketidakpastian

   Untuk menghadapi ketidakpastian maka pada sistem persediaan ditetapkan persediaan darurat yang dinamakan *safety stock*.

   a. Apabila permintaan telah diketahui maka persediaan barang dalam proses dan barang jadi akan disesuaikan dengan permintaan. Dalam hal ini tidak perlu ada persediaan dan apabila ada gejolak permintaan akan diteruskan kebagian produksi dan begian produksi akan berusaha mengatasi gejolak permintaan ini.

   b. Tetapi sesungguhnya safety *stock* dapat mengatasi hal seperti ini tanpa ikut campur bagian produksi. Demikian juga dengan persediaan bahan baku yang akan menyerap seandainya ada gejolak dari pemasok.

2. Memberi waktu luang untuk pengelolaan produksi dan pembelian

   Kadang-kadang lebih ekonomis memproduksi barang dalam proses atau barang jadi dalam jumlah besar atau dalam jumlah paket yang kemudian disimpan sebagai persediaan. Selama persediaan masih ada maka proses produksi dihentikan dan akan dimulai lagi bila diketahui persediaan hampir habis. Pertimbangan ini memberikan kemudahan, yaitu:

   a. Memberikan kemungkinan untuk menyebarkan dan meratakan beban biaya investasi pada sejumlah besar produk.
   b. Memungkinkan penggunaan satu peralatan untuk menghasilkan bermacam-macam jenis produk.

3. Untuk mengantisipasi perubahan pada *demand* dan *supply*

   Persediaan disiapkan untuk mengadapi beberapa kondisi yang menunjukkan perubahan *demand* dan *supply*.

   a. Bila ada perkiraan perubahan harga dan persediaan bahan baku.
   b. Sebagai persiapan menghadapi promosi pasar di mana sejumlah besar barang jadi disimpan menunggu penjualan tersebut.
   c. Perusahaan yang melakukan produksi dengan jumlah *output* tetap akan mengalami kelebihan produk, pada kondisi permintaan rendah atau pada kondisi musim lesu atau low season.

2.2.7 Klasifikasi Barang Persediaan

Secara fisik, item persediaan dapat dikelompokkan kedalam lima kategori [7], yaitu sebagai berikut:

1. Bahan Mentah (*raw materials*)

   Bahan mentah yaitu barang-barang berwujud seperti baja, atau bahan-bahan mentah lainnya yang diperoleh dari sumber-sumber alam, atau dibeli dari pemasok, atau diolah sendiri oleh perusahaan.

2. Komponen Rakitan (*parts/components*)

   Komponen rakitan yaitu barang-barang yang terdiri atas bagian-bagian (parts) yang diperoleh dari perusahaan lain atau hasil produksi sendiri untuk digunakan dalam pembuatan barang jadi atau barang setengah jadi.

3. Barang Setengah Jadi (*work in process*)

   Barang setengah jadi yaitu barang-barang keluaran dari tiap operasi produksi atau perakitan yang telah memiliki bentuk lebih kompleks daripada komponen.

4. Barang Jadi (*finished good*)

   Barang jadi adalah barang-barang yang telah selesai diproses dan siap untuk di distribusikan ke konsumen.

5. Bahan Pembantu (*supplies material*)

   Bahan pembantu adalah barang-barang yang diperlukan dalam proses pembuatan atau perakitan barang.

2.2.8 Fungsi Persediaan

Inventory memiliki fungsi tersendiri, sebagaimana tujuan dari diadakannya persediaan, berikut ini merupakan fungsi-fungsi utamanya:

1. Menghilangkan resiko keterlambatan pengiriman bahan
2. Menyesuaikan dengan jadwal produksi

3. Menghilangkan resiko kenaikan harga
4. Menjaga persediaan bahan yang dihasilkan secara musiman
5. Mengantisipasi permintaan yang dapat diramalkan
6. Mendapatkan keuntungan dari *quantity discount*
7. Komitmen terhadap pelanggan

2.2.9 Gudang

Gudang adalah suatu tempat yang digunakan untuk menyimpan barang, yang dijadikan sebagai objek transaksi bisnis, baik oleh sebuah badan tertentu maupun perseorangan.

2.2.10 Jenis Gudang

1. *Central Warehouse* (Gudang Pokok)

   Merupakan sebuah gudang yang berfungsi sebagai penerimaan, penyimpanan dan pengiriman barang.

2. *Retailer Warehouse* (Gudang Pengecer)

   Merupakan gudang yang berfungsi sebagai aktivitas penjualan langsung, biasanya adalah toko yang menjual barang kepada konsumennya.

3. *Distribution Warehouse* (Gudang Distribusi)

   Merupakan sebuah gudang yang berfungsi sebagai pendistribusian.

2.3 *Economic Order Quantity* (EOQ)

Dalam persoalan persediaan dikenal beberapa model. Masing-masing model mempunyai karakteristik tersendiri sesuai dengan parameter persoalan. Pada dasarnya model persediaan dibagi menjadi dua kelompok utama, yaitu model deterministik dan model stokastik. Model deterministik semua nilai

parameter-parameternya diasumsikan diketahui dengan pasti sedangkan model stokastik nilai-nilai parameternya tidak diketahui dengan pasti, berupa nilai-nilai acak [8]. Model EOQ dapat dikelompokkkan sebagai berikut:

1. Model EOQ Deterministik Statik
2. Model EOQ Deterministik Dinamik
3. Model EOQ Probabilistik Statik
4. Model EOQ Probabilistik Dinamik (Stokastik)

Sementara itu berikut ini jenis-jenis model persediaan EOQ Deterministik Statik:

1. Model EOQ Klasik (Sederhana) atau *Single Item*
2. Model EOQ *Back order*
3. Model EOQ *Fixed Production Rate*
4. Model EOQ *Quantity discount*

2.3.1 Model EOQ Klasik

Model jumlah pesanan terhemat (*Economic Order Quantity*) yang sering juga dinamakan model ukuran tumpuk sederhana digunakan dalam menentukan jumlah barang yang akan dipesan untuk setiap kali pemesanan serta jumlah biaya pengadaan bahan-bahan. EOQ menunjukkan jumlah barang yang harus dipesan setiap kali pemesanan agar biaya persediaan keseluruhan menjadi minimum.

"Konsep perhitungan EOQ ini juga berdasarkan pemikiran yang cukup logis dan sederhana sebagai berikut makin sering pengisian kembali persediaan itu dilakukan, persediaan rata-rata akan semakin kecil, dan ini

mengakibatkan biaya penyediaan barang akan makin kecil juga. Tetapi, di lain pihak makin sering pengisian kembali persediaan itu dilakukan, maka biaya pemesanan akan semakin besar pula. Oleh karena itu, dicari suatu keseimbangan yang paling ekonomis atau paling optimal dari dua hal yang saling bertentangan tersebut. Untuk mencari titik keseimbangan inilah maksud dari rumus EOQ" [3].

Metode ini disebut juga dengan metode ukuran lot atau *lot size method* yang digunakan untuk pengelolaan *independent demand inventory* dan didasarkan pada asumsi sebagai berikut [8]:

1. Kecepatan permintaan tetap dan terus menerus serta diketahui.
2. Waktu antara pemesanan sampai dengan pesanan datang (*lead time*) harus tetap.
3. Tidak pernah ada kejadian persediaan habis atau *stock out*.
4. Material dipesan dalam paket atau lot dan pesanan datang pada waktu yang bersamaan dan tetap dalam bentuk paket.
5. Harga per unit tetap dan tidak ada pengurangan harga walaupun pembelian dalam jumlah volume yang besar.
6. Besar *carrying cost* tergantung secara garis lurus dengan rata-rata jumlah persediaan.
7. Besar *ordering cost* atau *set up cost* tetap untuk setiap lot yang dipesan dan tidak tergantung pada jumlah item pada setiap lot.
8. Item yaitu produk semacam dan tidak berhubungan dengan produk lain.
9. *Stockout* tidak diperbolehkan, karena *demand* dan *lead time* diketahui

10. Barang yang dipesan dan disimpan hanya barang sejenis (*homogeny*)

Sedangkan parameter-parameter yang digunakan adalah :

k = *ordering cost* per pemesanan

A = jumlah barang yang dibutuhkan dalam 1 periode

c = *procurement cost* per unit barang yang dipesan

h = *holding cost* per unit persediaan

T = waktu antara pemesanan

Total annual cost = ordering cost + holding cost

Sebagai contoh diketahui di suatu perusahaan *ordering cost* per pemesanan adalah Rp500000 dan *holding cost* per unit persediaan adalah Rp40000, jadi dapat kita dapatkan total *annual cost* dengan menjumlahkan *ordering cost* dengan *holding cost* yaitu Rp90000.

Dari model persoalan persediaan ini akan dicari berapa jumlah pemesanan (Q), sehingga total *annual cost* mencapai minimum. *Order point* adalah satu dimana siklus persediaan yang baru dimulai dan yang lama berakhir. Setiap siklus persediaan mempunyai periode T, artinya setiap T satuan waktu pemesanan kembali dilakukan dan ini tergantung pada Q. Lamanya T sama dengan proporsi kebutuhan selama satu periode (A) [8].

$$T = \frac{Q}{A}$$

$$\text{Frekuensi pemesanan} = \frac{A}{Q} \qquad \textit{Annual ordering cost} = \left(\frac{A}{Q}\right)k$$

Sebagai contoh dicari waktu pemesanan dengan diketahui jumlah pemesanan yaitu 50 dan jumlah barang yang dibutuhkan dalam 1 periode adalah

100 maka didapat waktu pemesanan dengan membagi jumlah pemesanan dan jumlah barang yang dibutuhkan dalam 1 periode yaitu 0,5 .

Komponen biaya kedua adalah *holding cost*, yang ditentukan oleh jumlah dan lamanya barang disimpan. Setiap waktu jumlah barang berkurang sehingga perlu diperhatikan tingkat persediaan rata-rata.

$$\text{Rata-rata persediaan} = \frac{Q}{2}$$

Sebagai contoh diketahui jumlah pemesanan adalah 50, maka rata-rata persediaan yaitu 25.

Holding cost dihitung berdasarkan satuan nilai persediaan dan *procurement cost* (c), sehingga:

Holding cost per unit barang = hc

$$\textit{Total annual cost} = \left(\frac{A}{Q}\right)k + hc\left(\frac{Q}{2}\right) \qquad \textit{Annual holding cost} = \text{hc}\left(\frac{Q}{2}\right)$$

Sebagai contoh diketahui :

A = 1000

Q = 80

k = 200

h = 31

c = 78

Maka *total annual cost* adalah

(1000/80)200 + (31 x 78)(80/2) = 2500 + 96720 = 99220

Total *annual cost* minimum dapat dicari dengan menentukan berapa jumlah pemesanan (Q). Total *annual cost* mencapai minimum jika antara

fungsi *annual order cost* dan *total annual holding cost* berharga sama. Secara matematis ditulis:

$$hc\left(\frac{Q}{2}\right)=\left(\frac{A}{Q}\right)k$$

$$Q=\sqrt{\frac{2Ak}{hc}}$$

Dalam menerapkan EOQ ada beberapa biaya yang harus dipertimbangkan dalam penentuan jumlah pembelian atau keuntungan, diantaranya :

1. Biaya Pemesanan

   Biaya pemesanan merupakan biaya yang akan langsung terkait dengan kegiatan pemesanan yang dilakukan perusahaan. Biaya pemesanan berfluktuasi bukan dengan jumlah barang yang dipesan, tetapi dengan frekuensi pesanan. Biaya pesan tidak hanya terdiri dari biaya yang eksplisit, tetapi juga biaya kesempatan (*Opportunity Cost*). Misalnya, waktu yang terbuang untuk memproses pesanan, menjalankan administrasi pesanan dan sebagainya. Contoh biaya tersebut antara lain: Biaya persiapan, Biaya telepon, Biaya pengiriman dan Biaya pembuatan faktur. Rumus biaya pemesanan [9] adalah sebagai berikut:

$$BiayaPesan=\frac{D}{Q}xS$$

   Keterangan :

   Q = Jumlah Barang setiap pesan

   D = Permintaan barang persediaan, dalam unit.

   S = Biaya pemesanan untuk setiap pesanan

Sebagai contoh diketahui :

Q = 80

D =10

S = 10000

Maka biaya pesan adalah (10/80) x 10000 = 1250

2. Biaya Penyimpanan

Biaya penyimpanan merupakan biaya yang harus ditanggung oleh perusahaan sehubungan dengan adanya barang yang disimpan dalam gudang perusahaan. Biaya simpan akan berfluktuasi dengan tingkat persediaan. Beberapa contoh biaya penyimpanan antara lain: Biaya pemeliharaan, Biaya asuransi, Biaya kerusakan dalam penyimpanan, Biaya sewa gedung, dan Biaya fasilitas penyimpanan. Biaya penyimpanan dirumuskan sebagai [9]:

$$BiayaPenyimpanan = \frac{Q}{2} xH$$

Keterangan :

Q = Jumlah barang setiap pemesanan

H = *Holding cost*

Sebagai contoh diketahui :

Q = 80

H = 31

Maka biaya penyimpanan adalah (80/2) x 31 = 1240

Sehingga dalam menentukan biaya persediaan ada dua (2) jenis biaya yang berubah-ubah dan harus dipertimbangkan. Pertama berubah-ubah sesuai

dengan frekuensi pesanan yaitu biaya pesan. Kedua biaya yang berubah-ubah sesuai dengan besar kecilnya persediaan yaitu biaya penyimpanan. Selanjutnya menentukan total biaya persediaan (TC) dengan menjumlahkan biaya pesan dan biaya simpan. Adapun rumusnya sebagai berikut :

$$TC = \frac{D}{Q}S + \frac{Q}{2}H$$

Keterangan :

TC = Total biaya persediaan

Q = Jumlah barang setiap pesan

D = Permintaan tahunan barang persediaan dalam unit

S = Biaya pemesanan untuk setiap pesan

H = Biaya penyimpanan per unit per tahun

Sebagai contoh diketahui :

Q = 80

D = 200

S = 1250

H = 1240

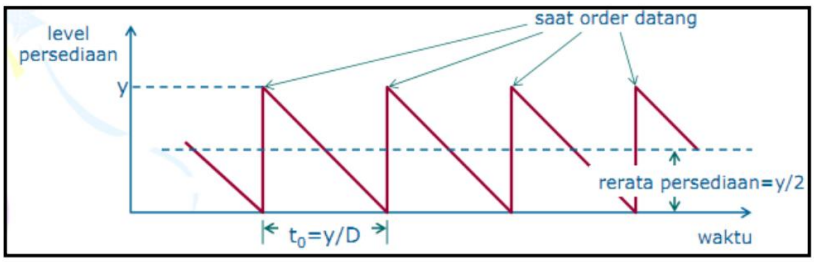
Maka TC adalah ((200/80) x 1250) + ((80/2) x 1240) = 3125 + 49600 = 52725

Didefinisikan dalam bentuk grafik sebagai berikut :

y = kuantitas order (unit)

D = permintaan (unit/waktu)

t0 = panjang siklus pengorderan (waktu)



Gambar 2. 3 EOQ *Inventory* [3]

Hal-hal yang perlu diperhatikan sebelum menghitung EOQ :

D : Besar laju permintaan (*demand rate*) dalam unit per tahun.

S : Biaya setiap kali pemesanan (*ordering cost*) per pesanan

C : Biaya per unit dalam harga (rupiah) per unit

i : Biaya pengelolaan (*carrying cost*) adalah persentase terhadap nilai persediaan per tahun

Q : Ukuran paket pesanan (*lot size*) dalam unit

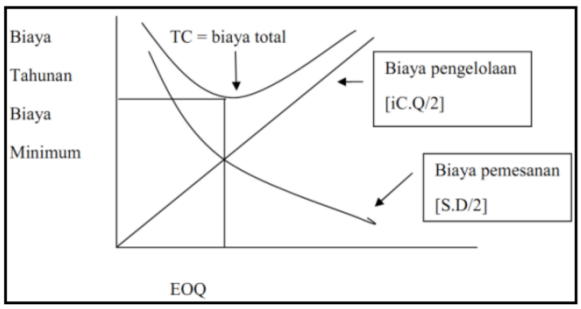
TC : Biaya total persediaan dalam rupiah per tahun

H : *Holding cost* (ic)

Biaya pemesanan per tahun (*Ordering cost*) : OC = S(D/Q)

Biaya pengelolaan persediaan per tahun (*Carrying cost*) : CC = ic(Q/2)

Maka, total biaya persediaan : TC = S(D/Q) + ic(Q/2)



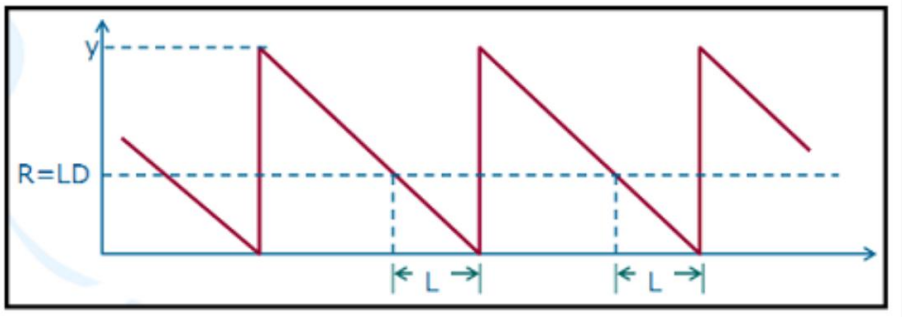
Gambar 2. 4 Keseimbangan EOQ [8]

Terjadi keseimbangan antara *carrying cost* dan *ordering cost* dan menghitung jumlahpaket pesanan ekonomis atau EOQ yaitu Q dihitung dari :

$$Q = \sqrt{(2SD)/ic}$$

2.3.2 Ekstensi Model EOQ Klasik *Lead time*

Kenyataannya, kedatangan order tidak secara bersamaan, ada selang waktu dari pengorderan sampai kedatangan barang yang disebut sebagai lead time (L). Sementara itu pengorderan harus dilakukan ketika persediaan tinggal sejumlah kebutuhan selama *lead time* = LD = *reorder point* (R).



Gambar 2. 5 EOQ *Inventory Lead time* [3]

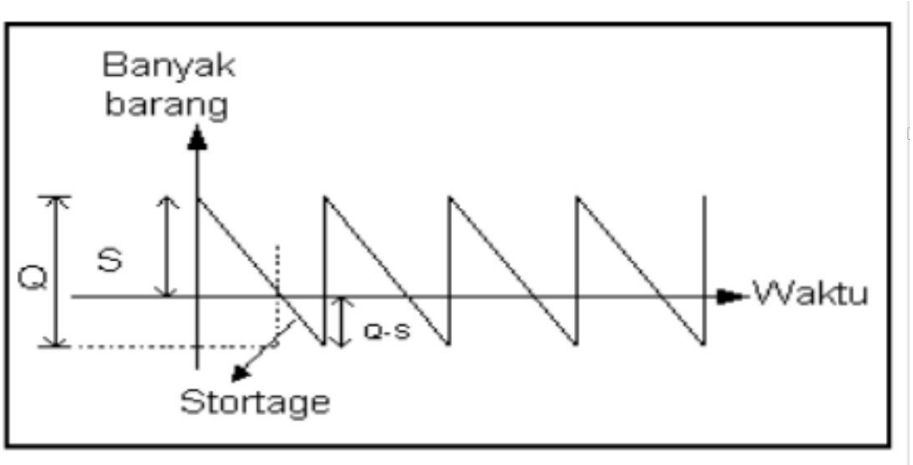
2.3.3 Model EOQ *Back order*

Model ini mendeskripsikan EOQ yang berusaha untuk tidak kehilangan penjualan yang mana akan menyebabkan pelanggan menunggu. Asumsinya adalah sebagai berikut:

1. Pesanan untuk diambil kemudian lazim disebut back order. Dengan kata lain sebuah toko mengijinkan adanya *back order* apabila ia tetap menjual suatu barang yang meskipun sudah tidak ada di gudang (tingkat persediaan barang nol).

2. Model EOQ dengan asumsi pembeli dapat memesan barang kembali walaupun pesanan tersebut akan dipenuhi kemudian, sehingga dalam model ini ada biaya yang harus dikeluarkan akibat kehabisan stok (Cs).

   Sebagai pembanding berikut ini merupakan gambar grafik EOQ-nya:



Gambar 2. 6 EOQ *Back order* [8]

Contoh kasus :

Sebuah toko minuman mampu menjual 5200 peti bir setiap tahun. Setiap peti menanggung biaya $ 20 untuk sampai ke gudang. Penyalur meminta bayaran $ 100 untuk pemesanan. Modal kerja yang dimiliki toko tsb dipinjam dari bank dengan bunga 10% pertahun, selain itu pemilik toko harus membayarkan atas barang yg disimpannya sebesar 5 % dari nilai persediaan rata-rata serta asuransi sebesar 5 %. Disisi lain pembeli akan menunggu sampai barang tsb ada, walaupun barang tersebut sedang tidak tersedia. Andaikan toko tsb dibebani $ 0,01 per peti per hari sebagai hukuman karena tidak dapat memenuhi permintaan langganan, maka tentukan Qo dan So dari kasus di atas?

Jawaban

D = 5200 per Tahun; Co = $ 10; Cs = 3,65 per tahun

Ch1 =20 % = 0,2 [bunga bank + asuransi + pajak simpanan] x 20 = (0,2)(20) = 4 + 20 = $ 24

$$Qo = \sqrt{\frac{2CoD}{Ch}}\left(\frac{Cs + Ch}{Cs}\right) = \sqrt{\frac{2.100\ .5200}{24}}\left(\frac{3,65 + 24}{3,65}\right) = \sqrt{328033\ ,1} = 572,7 \approx 573 \text{ peti}$$

$$So = \sqrt{\frac{2CoD}{Ch}}\sqrt{\frac{Cs}{Cs + Ch}} = \sqrt{\frac{2.100\ .5200}{24}}\sqrt{\frac{3,65}{(3,65 + 24)}} = \sqrt{43333\ ,3}\quad \sqrt{0,132} = 74,9 = 75$$

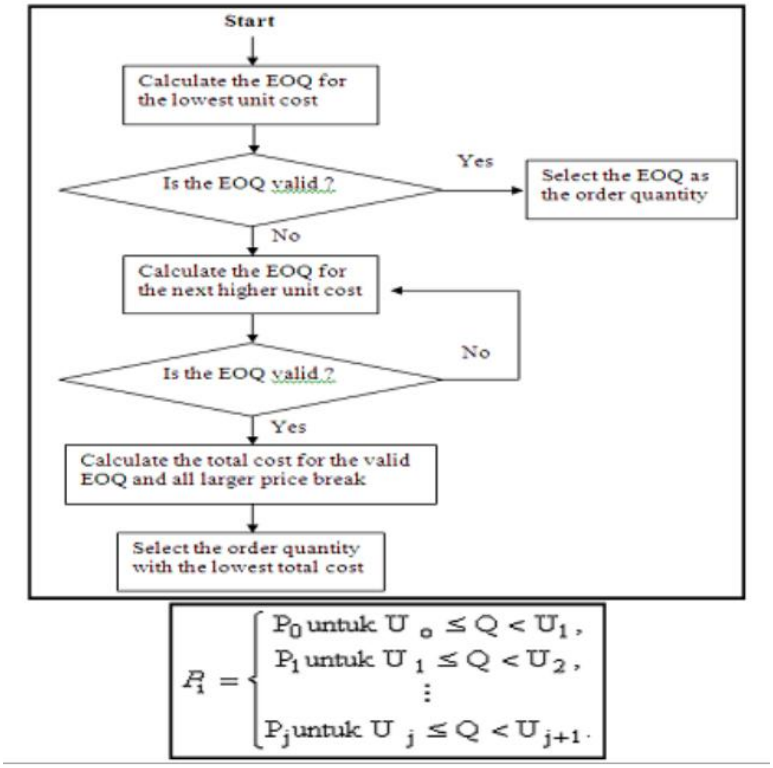
To = Qo /D = 573/5200 = 0,11 tahun = 40 hari

Ternyata apabila toko tersebut mengijinkan adanya *back order* maka kebijaksanaan persediaan yang optimal = 573 peti yang dipesan setiap 40 hari. Diantara yang dipesan tersebut, hanya 75 peti yang disimpan sebagai persediaan, selebihnya 498 peti digunakan untuk memenuhi permintaan yang belum terpenuhi (*back order*).

2.3.4 Model EOQ *Quantity discount*

Model EOQ ini memiliki prosedur menentukan jumlah yang dipesan sebagai berikut:

1. Mulai dengan harga yang paling rendah, hitung EOQ pada setiap harga, sampai diperoleh EOQ yang *valid.*
2. Hitung biaya total tahunan untuk EOQ yang *valid* dan semua *price break quantities* yang lebih besar dari EOQ (*price-break quantity* adalah jumlah terkecil yang akan mendapat harga diskon).
3. Pilih *quantity* dengan biaya total minimum pada Langkah 2.



Gambar 2. ***7 Flow Chart EOQ Quantity Discount*** **[8]**

2.4 *Lead time* (Waktu Tunggu)

Untuk menjamin kelancaran proses bisnis perusahaan perlu memperhatikan jangka waktu antara saat mengadakan pemesanan dengan saat penerimaan barang-barang yang dipesan kemudian dimasukkan kedalam gudang. Lamanya waktu antara mulai dilakukannya pemesanan barang sampai dengan kedatangan barang yang dipesan dinamakan *lead time*. Barang yang datang terlambat mengakibatkan kekurangan barang. Sedangkan barang yang datang lebih awal dari waktu yang telah ditentukan akan memaksa perusahaan memperbesar biaya penyimpanan.

Faktor-faktor yang perlu diperhatikan dalam menetukan *lead time* adalah:

1. *Stock Out Cost*

   Stock Out Cost adalah biaya-biaya yang terpaksa dikeluarkan karena keterlambatan datangnya barang.

2. *Extra Carrying cost*

   Extra Carrying Cost adalah biaya-biaya yang terpaksa dikeluarkan karena keterlambatan barang datang lebih awal.

2.5 *Reorder point* (ROP)

Merupakan sebuah titik dimana suatu pemesanan baru harus dilakukan. Hal ini dipengaruhi oleh lead time, yaitu waktu yang dibutuhkan untuk menerima pesanan setelah pemesanan dilakukan atau persiapan dimulai.

Jika model EOQ yang diterapkan, maka faktor terpenting adalah *lead time.* *Lead time* adalah jarak waktu antara melakukan order hingga order datang. Adanya *lead time* membuat kita harus menentukan waktu pemesanan. Pada model EOQ *lead time* diketahui dengan pasti. Namun pada kenyataannya, baik permintaan maupun *lead time* sama-sama tidak pasti. Oleh karena itu, waktu pemesanan suatu barang harus mempertimbangkan ketidakpastian dari dua aspek tersebut. Berikut ini cara perhitungan untuk mendapatkan nilai ROP:

$$ROP=Q \times Lt$$

Q : Jumlah Kebutuhan

Lt : *Lead time*

Misal dalam suatu perusahaan diketahui jumlah kebutuhan 2,74 dan *lead time* 5 hari maka didapat ROP dengan mengalikan jumlah kebutuhan dan leadtime yaitu 13,7 dibulatkan menjadi 14.

2.6 Persediaan Pengamanan (*Safety stock*)

Persediaan cadangan adalah persediaan yang disimpan untuk mengantisipasi permintaan pelanggan yang sulit diketahui dengan pasti [10]. Stok cadangan disimpan untuk memenuhi permintaan yang besar.

Persediaan pengaman atau sering pula disebut *safety stock* adalah persediaan yang dilakukan untuk mengantisipasi unsur ketidakpastian permintaan dan penyediaan. Apabila persediaan pengaman tidak mampu mengantisipasi ketidakpastian tersebut, maka akan terjadi kekurangan persediaan (*stockout*).

Faktor-faktor yang menentukan besarnya *safety stock* adalah :

1. Penggunaan bahan baku rata-rata

   Salah satu dasar untuk memperkirakan penggunaan bahan baku selama periode tertentu, khusunya selama periode pemesanan adalah rata-rata penggunaan bahan baku pada masa sebelumnya.

2. Faktor waktu

   Lead time adalah lamanya waktu antara mulai dilakukannya pemesanan bahan-bahan sampai dengan kedatangan bahan-bahan yang dipesan.

3. Persediaan antisipasi

Persediaan antisipasi disebut sebagai *stabilization stock* merupakan persediaan yang dilakukan untuk menghadapi fluktuasi permintaan yang sudah dapat diperkirakan sebelumnya.

4. Persediaan dalam pengiriman (*transit stock*)

   Persediaan dalam pengiriman disebut *work-in process stock* adalah persediaan yang masih dalam pengiriman, yaitu :

   a. *External transit stock* adalah persediaan yang masih berada dalam transportasi
   b. *Internal transit stock* adalah persediaan yang masih menunggu untuk diproses atau menunggu sebelum dipindahkan

$$ROPs = ROP + (Qmaks - Qr) \times Lt$$

ROPs : *reorder point* dengan stok pengaman

ROP : *reorder point* semula (sebelum *safety stock*)

Qmaks : jumlah permintaan (penggunaan) maksimal

Qr : jumlah permintaan rata-rata

Lt : *Lead time*

Sebagai contoh diketahui *lead time* 5 hari, jika kebutuhan akan barang melebihi dari hasil perhitungan jumlah kebutuhan misal hasil kebutuhan barang bukan 2,74 tapi 4 unit per hari maka akan terjdi kehabisan stok pada hari berikutnya, sehingga perhitungan reorder point yang baru adalah sebagai berikut :

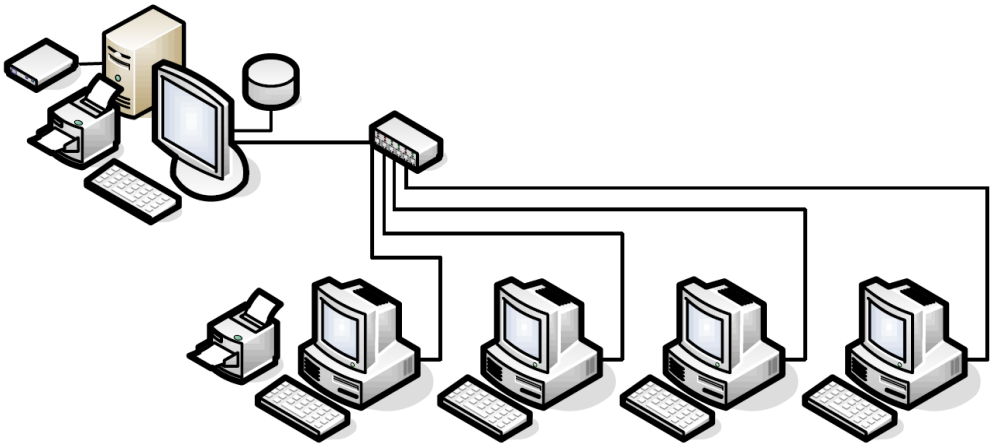
ROPs = (2,74 x 5) + (4 x 2,74) x 5

= 13,7 + 6,3

= 20 unit

2.7 *Client Sever*

Arsitektur jaringan *client server* merupakan pengembangan dari arsitektur *file server*. Arsitektur ini adalah model konektivitas pada jaringan yang mengenal adanya *server* dan *client*, dimana masing-masing memiliki fungsi yang berbeda satu sama lain. *Server* dapat berbagi pakai data, aplikasi dan *peripheral* seperti *harddisk, printer, modem* dan lain-lain. Oleh karena itu, tidak jarang juga tercipta sebutan *print server*, *communication server* dan lain sebagainnya. Prinsip kerjanya sangat sederhana, dimana *server* akan menunggu permintaan dari *client*, memproses dan memberikan hasilnya kepada *client*. Sedangkan *client* akan mengirimkan permintaan ke *server*, menunggu proses dan melihat visualisasi hasil prosesnya.

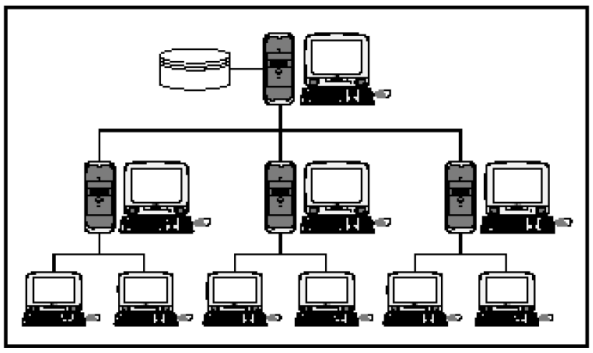


Gambar 2. 8 Konektivitas *Client Server*

Sistem *client server* ini menggunakan protocol TCP/IP (*Transmission Control Protocol/Internet Protocol*). Unix dan Windows NT merupakan contoh yang baik dari sistem operasi jaringan *client server*.

2.7.1 Sistem *Client Server*

Sistem *Client* dan *Server* terdiri atas dua komponen (mesin) utama, yaitu *Client* dan *Server*. *Client* berisi aplikasi basis data dan *server* berisi DBMS dan basis data. Setiap aktifitas yang dikehendaki para pemakai akan lebih dahulu ditangani oleh *client*. *Client* menangani proses yang menjadi tanggung jawabnya. Jika ada proses yang harus melibatkan data yang tersimpan pada basis data yang terletak di *server*, barulah *client* mengadakan hubungan denga *server*. Pada bentuk sistem *client server* untuk memenuhi kebutuhan *client* akan megirimkan pesan atau perintah *Query* pengambilan data. Selanjutnya *server* yang menerima pesan tersebut akan menjalankan *Query t*ersebut dan hasilnya akan dikirimkan kembali ke *client*. Dengan begitu, transfer datanya jauh lebih efisien. Untuk lebih jelasnya, dapat dilihat pada gambar sistem *client server* berikut ini :



Gambar 2. 9 Sistem *Client Server*

2.7.2 Komponen Dasar *Client Server*

1. *Client*

Client merupakan terminal yang digunakan oleh pengguna untuk meminta layanan tertentu yang dibutuhkan. Terminal *client* dapat berupa PC, ponsel, komunikator, robot, televisi dan peralatan lain yang membutuhkan in*Form*asi.

2. *Middleware*

Midleware merupakan komponen perantara yang memungkinkan *client* dan *server* untuk saling terhubung dan berkomunikasi satu sama lain. *Midleware* ini dapat berupa *Transaction Monitor* /TP. *Remote Procedure Call* atau *Object Request Broker*/ORB.

3. *Server*

Server merupakan komputer khusus yang bertugas melayani aplikasi-palikasi jaringan / pihak yang menyediakan layanan. *Server* ini akan dapat berupa basis data SQL, Monitor TP, *server groupware*, *server* objek dan *web*. Secara umum, *server* berperan menerima pesan permintaan layanan dari *client*, memproses permintaan tersebut dan mengirimkan hasil permintaan kepada *client*.

2.7.3 Karakteristik *Client Server*

1. Karakteristik *Server*

   a. Pasif

   b. Menunggu request

   c. Menerima request, memproses dan mengirimkan balasan berupa servis

2. Karakteristik *Client*

   a. Aktif

b. Mengirim request

c. Menunggu dan menerima balasan dari *server*

2.7.4 Ciri-ciri *Client Server*

Beberapa ciri dari arsitektur sistem terdistribusi *Client Server* diantaranya :

1. Berbasis layanan

Server memberikan sejumlah layanan yang dibutuhkan dan diminta oleh *client*, antara lain : berbagai pakai berkas, dan peralatan pendukung.

2. Sumber daya dan interaksi *client server*

Server memberikan sejumlah layanan yang dibutuhkan dan diminta oleh *client*, antara lain : berbagai pakai berkas, dan peralatan pendukung.

3. Hubungan dan interaksi *client server*

Hubungan yang terjadi antara *server* dan *client* adalah *one-to many*, yang berarti bahwa satu *server* melayani banyak *client*. *Client* selalu memulai transaksi dengan meminta layanan sedangkan *server* menanti permintaan layanan secara pasif.

4. *Client* tidak perlu mengetahui lokasi fisik *server*

Server dapat terletak di berbagai tempat yang belum tentu diketahui oleh *client*, Walaupun demikian *client* tetap dapat mengakses *server* untuk mendapatkan layanan sesuai kebutuhannya.

5. Interoperabilitas perangkat lunak dan perngkat keras

Perangkat lunak dan keras yang digunakan oleh masing-masing *client* tidak harus sama dengan yang digunakan pada *server*, namun masih dapat saling terkoneksi antara satu dan yang lain.

6. Pertukaran berbasis pesan

Mekanisme dari *Client Server* berdasar pada pertukuran pesan. Pesan yang dipertukarkan adalah permintaan layanan dan umpan balik dari permintaan layanan tersebut.

7. Enkapsulasi layanan

Mekanisme dari *Client Server* berdasar pada pertukuran pesan. Pesan yang dipertukarkan adalah permintaan layanan dan umpan balik dari permintaan layanan tersebut.

8. Skalabilitas

Skalabilitas adalah kemampuan untuk diperbesar atau diperkecil. Ukuran sistem *Client Server* dapat diubah secara horizontal maupun vertikal. Perubahan vertikal berarti berpindah ke *server* lebih besar atau lebih cepat atau mendistribusikan tugas melayani *client* ke beberapa *server*. Pengubahan horizontal berarti menambah atau mengurangi jumlah *client*.

9. Konsistensi Data

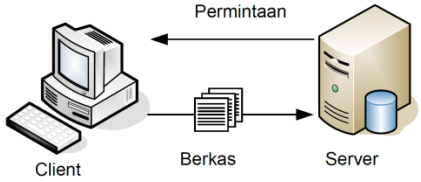
Data hanya dikelola pada *server* pusat sehingga konsistensi dan data lebih terjamin dan biaya pemeliharaan pun menjadi lebih murah.

2.7.5 Tipe Jaringan *Client Server*

Berdasarkan tipe layanan yang diberikan *server* kepada *client*, jaringan *Client Server* dapat dibagi menjadi kedalam banyak tipe, tipe-tipe tersebut antara lain :

1. *Server* Berkas

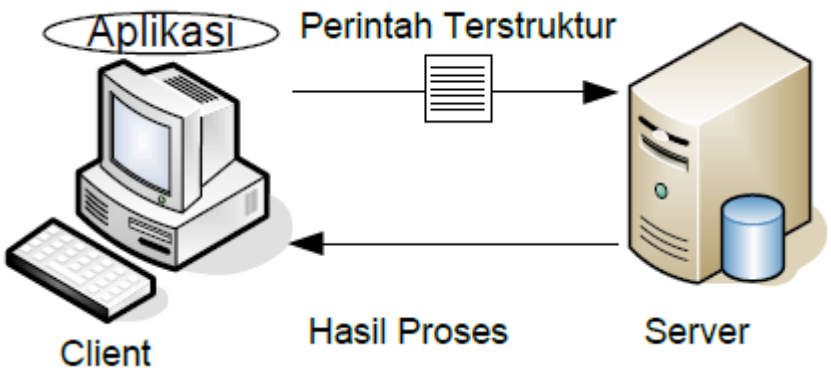
Sistem jaringan berkas adalah sistem jaringan yang dimana layanan yang diberikan *server* berupa berkas, baik berkas aplikasi seperti aplikasi pengolahan kata, pengolahan angka, pengolahan data, pengolahan gambar dan lain sebagainya, maupun berkas yang dihasilkan oleh aplikasi tersebut, seperti dokumen pengolahan kata, tabel-tabel pengolahan angka, berkas presentasi dan lain sebagainya.



Gambar 2. 10 *Server* Berkas

2. *Server* Basis Data

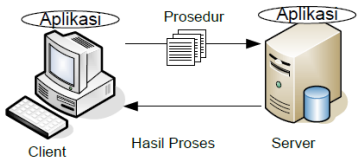
Sistem jaringan *server* basis data adalah merupakan sistem jaringan dimana layanan yang diberikan oleh *server* berupa pengolahan dan penyajian data berdasarkan perintah terstruktur (*query*) yang diberikan *client*. Pada jaringan ini, *server* menyimpan berbagai macam data yang dapat diakses oleh pengguna melalui terminal-terminal *client*.



Gambar 2. 11 *Server* Basis Data

3. *Server* Transaksi

Sistem jaringan *server* transaksi adalah sistem jaringan dimana layanan yang diberikan *server* berupa hasil Sistem Operasi dari sekelompok perintah terstruktur yang diberikan *client*. Jaringan ini pada dasarnya hampir sama dengan sistem jaringan basis data sebelumnya. Perbedaan terletak pada *server* transaksi yang memproses sekelompok perintah terstruktur dari *client*, dan sekelompok perintah terstruktur ini disebut prosedur.



Gambar 2. 12 *Server* Transaksi

4. *Groupware Server*

Sistem jaringan *groupware* adalah sistem jaringan dimana layanan yang diberikan *server* berupa fasilitas pemakaian bernama in*Form*asi semi terstruktur diantara pengguna jaringan. Pada jaringan ini, *server* menyimpan, mengelola dan

menyebarkan in*Form*asi antar pengguna dalam jaringan, misalnya teks, gambar, surat dan ruang diskusi.

5. *Server* Objek

Sistem jaringan *server* objek adalah sistem jaringan dimana layanan yang diberikan *server* berbentuk objek. Dalam jaringan ini, *client* dan *server* berkomunikasi melalui objek-objek yang miliki *client* dan *server*.

6. *Web Server*

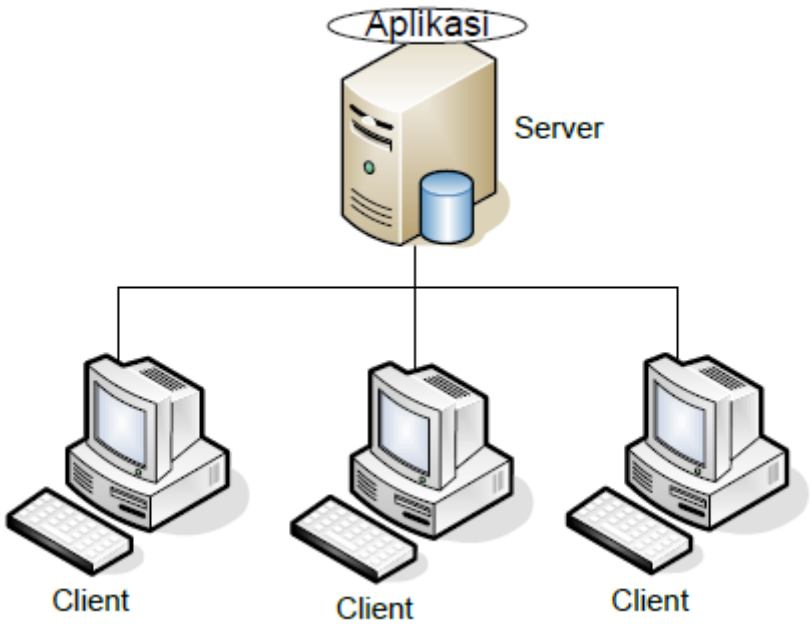
Sistem jaringan *web server* adalah sistem jaringan dimana layanan yang diberikan *server* berupa pengelolaan dan pemakaian bersama dokumen-dokumen yang saling terhubung. Jaringan ini merupakan jaringan yang memungkinkan tiap dokumen dalam jaringan memiliki hubungan ke dokumen lain sehingga dokumen-dokumen dalam jaringan terhubung satu dengan yang lain, semacam jaringan laba-laba.

2.7.6 Arsitektur *Client Server*

1. Two Tier

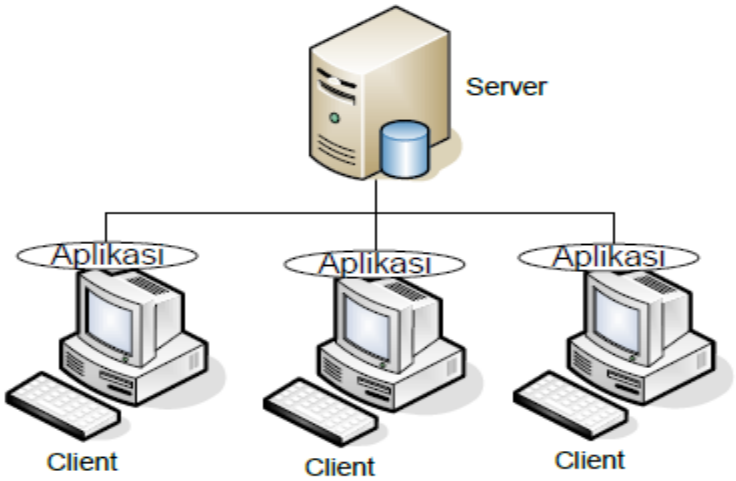
Arsitektur *two tier* merupakan arsitektur yang disebut *Client Server* dimana terdapat komputer sebagai *client* dan *server* yang berinteraksi melalui protokol dan media komunikasi tertentu Model arsitektur *Two Tier* dikelompokan menjadi dua macam yaitu :

a. Thin Client – Thick Server



Gambar 2. 13 *Thin Client – Thick Server*

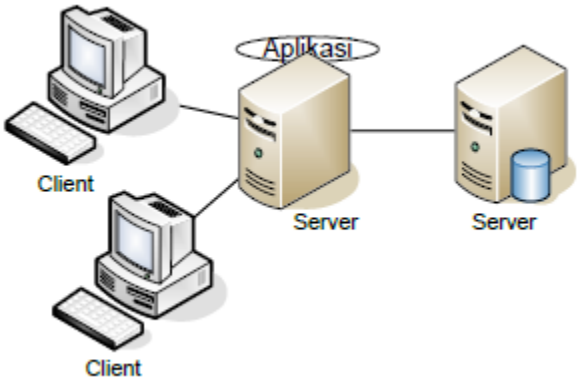
b. Thick Client – Thin Server



Gambar 2. 14 *Thick Client – Thin Server*

2. Three Tier

Arsitektur *Client Server* ini memisahkan antara data (*Data Management Tier*), aplikasi (*Middle Tier*) dan penyajian (*Presentation Layer*) sebagaimana ditampilkan pada gambar dibawah ini :



Gambar 2. 15 Arsitektur Three Tier

a. Data Management Tier

Merupakan komputer *server* yang dikhususkan untuk menangani pengolahan basis data.

b. Middle Tier

Merupakan komputer *server* yang dikhususkan untuk menangani aplikasi-aplikasi dimana prosedur-prosedur dan perhitungan-perhitungan yang kompleks dilakukan di komputer.

c. Presentation Layer

Merupakan komputer *client* yang menjadi *interface* bagi pengguna untuk memasukkan data, mengajuan permintaan layanan kepada *server* dan melihat hasilnya.

3. n-Tier

Istilah *n-tier* menunjukan lapisan yang ada dalam sebuah aplikasi. Sebuah aplikasi terdiri dari beberapa komponen utama, yaitu lapisan *Presentation* (*Presentation Layer*), lapisan *application* (*Application Layer*) dan logika bisnis (*bussiness logic layer*) dan lapisan data (*Data layer*).

Lapisan *Presentation* menghubungkan antarmuka dengan pengguna aplikasi, dapat berupa model grafis atau berupa teks. Pengguna dapat berinteraksi dengan aplikasi tersebut menggunakan lapisan *presentation* ini. Lapisan *Application* berisi inti dari aplikasi dan lapisan data yang digunakan oleh aplikasi tersebut. Lapisan data dapat berbentuk satu atau lebih *server* basis data yang lokasinya tersebar dibeberapa tempat.

2.8 Konsep Dasar Sistem

"Sistem terdapat dua kelompok pendekatan sistem, yaitu sistem yang lebih menekankan pada prosedur dan elemennya. Prosedur didefinisikan sebagai suatu urutan-urutan yang tepat dari tahapan-tahapan instruksi yang menerangkan apa yang harus dikerjakan, siapa yang mengerjakan, kapan dikerjakan, dan bagaimana mengerjakannya" [11].

2.8.1 Pengertian Sistem

Sistem adalah sekumpulan objek atau elemen yang saling berinteraksi untuk mencapai satu tujuan tertentu [12]. Beberapa prinsip umum sistem adalah sebagai berikut :

1. Sistem selalu merupakan bagian dari sistem yang lebih besar, sekaligus sistem tersebut dapat dipartisi menjadi sub-sistem yang lebih kecil
2. Sistem yang lebih terspesialisasi akan kurang dapat beradaptasi untuk menghadapi keadaan-keadaan yang berbeda.
3. Lebih besar ukuran sistem, maka memerlukan sumberdaya yang lebih banyak untuk operasi dan pemeliharaannya.
4. Sistem senantiasa mengalami perubahan, tumbuh dan berkembang.

Sistem adalah kumpulan atau himpunan dari unsur atau variabel-variabel yang saling terorganisasi, saling berinteraksi dan saling bergantung sama lain [13], ada juga yang berpendapat bahwa sistem adalah jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu [14]. Sistem adalah kumpulan/grup dari subsistem/bagian/komponen apapun baik fisik ataupun non fisik yang saling berhubungan satu sama lain dan bekerja sama secara harmonis untuk mencapai satu tujuan tertentu [15].

Sistem yang tengah dikaji adalah berada di dalam batas, sementara di luar batas adalah lingkungan dimana sistem berinteraksi. Lingkungan didefinisikan sebagai kondisi-kondisi yang melingkupi, di luar batas lingkungan mempengaruhi dan dipengaruhi sistem, tapi tidak dapat dikendalikan oleh sistem. Karena batas-

batas dari sistem berbasis komputer sering sulit didefinisikan maka pengembang, *client* dan pemakai sistem harus bekerja sama dalam memutuskan mengenai apa-apa yang termasuk di dalam sistem. Maka yang dimaksud dengan Sistem adalah sekumpulan objek atau elemen yang dipandang sebagai keseluruhan dan dirancang untuk mencapai satu sasaran, memiliki batas dimana sistem berada didalamnya dan di luar batas adalah sebagai lingkungannya.

Dari penjabaran definisi diatas maka dapat digambarkan suatu sistem dengan menetukan komponen-komponennya, bagaimana bagian-bagian tersebut berhubungan dan bagaimana ciri-ciri dari tujuan yang harus dicapai.

2.8.2 Karakteristik Sistem

Suatu sistem mempunyai karakteristik atau sifat-sifat tertentu [11] yaitu:

1. Komponen Sistem (*Component*)

   Komponen sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerjasama membentuk suatu kesatuan. Komponen-komponen sistem atau elemen-elemen sistem dapat berupa suatu subsistem atau bagian dari sistem.

2. Batasan Sistem (*Boundary*)

   Batas sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batas sistem ini memungkinkan suatu sistem dipandang sebagai suatu kesatuan dan menunjukkan ruang lingkup dari sistem tersebut.

3. Lingkungan Luar Sistem (*Environment*)

Lingkungan luar sistem dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat menguntungkan dan juga merugikan. Lingkungan luar yang menguntungkan merupakan energi dari sistem dan dengan demikian harus dijaga dan dipelihara. Sedangkan lingkungan luar sistem yang merugikan harus ditahan dan dikendalikan, jika tidak maka akan menganggu kelangsungan hidup dari sistem.

4. Penghubung Sistem (*Interface*)

   Penghubung merupakan media yang menghubungkan antara satu subsistem dengan subsistem yang lainnya.

5. Masukan Sistem (*Input*)

   Masukan sistem adalah energi yang dimasukkan kedalam sistem. Masukan dapat berupa masukan perawatan dan masukan sinyal *maintenance input* yaitu energi yang dimasukkan supaya sistem tersebut dapat berjalan. Sinyal *input* adalah energi yang diproses untuk mendapatkan keluaran dari sistem.

6. Keluaran Sistem (*Output*)

   Keluaran sistem adalah energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran dapat merupakan masukan untuk subsistem yang lain.

7. Pengolahan Sistem (*Process*)

Suatu sistem dapat mempunyai suatu bagian pengolah atau sistem itu sendiri sebagai pengolahnya. Pengolah yang akan merubah masukan menjadi keluaran.

8. Sasaran Sistem (*Goal*)

   Suatu sistem mempunyai tujuan atau sasaran, kalau sistem tidak mempunyai sasaran maka sistem tidak akan ada. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuannya. Sasaran sangat berpengaruh pada masukan dan keluaran yang dihasilkan.

2.8.3 Klasifikasi Sistem

Sistem merupakan suatu bentuk integrasi antara satu komponen dengan komponen lainnya. Karena sistem memiliki sasaran yang berbeda untuk setiap kasus yang terjadi yang ada didalam sistem tersebut. Oleh karena itu, sistem dapat diklasifikasikan kedalam beberapa sudut pandang. Berikut akan dipaparkan mengenai klasifikasi sistem [11].

1. Sistem Abstrak dan Sistem Fisik
   a. Sistem abstrak : Sistem yang berupa pemikiran/ide-ide yang tidak tampak secara fisik.
   b. Sistem fisik : Sistem yang ada secara fisik
2. Sistem Alamiah dan Sistem Buatan
   a. Sistem alamiah : Sistem yang terjadi karena proses alam tidak dibuat oleh manusia.

b. Sistem buatan manusia : Sistem yang dirancang oleh manusia. Sistem buatan manusia yang mellibatkan interaksi manusia dengan mesin disebut dengan *human-machine system* atau ada yang menyebut dengan *man-machine system*. Contoh : Sistem Akuntansi.

3. Sistem Tertentu dan Sistem Tak Tentu
   a. Sistem tertentu: Beroperasi dengan tingkah laku yang sudah dapat diprediksi.
   b. Sistem tak tentu: Sistem yang kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilitas.

4. Sistem Tertutup dan Sistem Terbuka
   a. Sistem tertutup: Sistem yang tidak berhubungan dan tidak terpengaruh dengan lingkungan luarnya.
   b. Sistem terbuka: Sistem yang berhubungan dan terpengruh dengan lingkungan luarnya. Sistem ini menerima masukan dan menghasilkan keluaran untuk lingkungan luar atau subsistem yang lain.

2.9 Konsep Dasar In*Form*asi

In*Form*asi merupakan hal yang sangat penting di dalam sebuah sistem. Jika sebuah sistem mengolah in*Form*asi yang salah maka penerima in*Form*asi akan susah untuk mengambil keputusan masa kini atau masa yang akan datang.

2.9.1 Data dan In*Form*asi

Data adalah deskripsi dari sesuatu dan kejadian yang kita hadapi (*the description of things and events that we face*). Definisi data yang lain adalah merupakan kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan nyata [11].

In*Form*asi merupakan salah satu sumberdaya penting dalam suatu organisasi, digunakan sebagai bahan pengambilan keputusan [16]. In*Form*asi sebagai data yang telah diolah menjadi bentuk yang lebih berarti dan berguna bagi penerimanya untuk mengambil keputusan masa kini maupun yang akan datang [11].

Sumber in*Form*asi adalah data. Data adalah kenyataan yang menggambarkan kejadian-kejadian dan kesatuan nyata. Kejadian adalah sesuatu yang terjadi pada saat tertentu. In*Form*asi diperoleh setelah data-data mentah diproses atau diolah. Agar in*Form*asi yang dihasilkan menjadi lebih berharga, maka harus memenuhi kriteria sebagai berikut [11]:

1. In*Form*asi harus akurat, sehingga mendukung pihak manajemen dalam mengambil keputusan.
2. In*Form*asi harus relevan, terasa bermanfaat bagi yang membutuhkan.
3. In*Form*asi harus tepat waktu, sehingga tidak ada keterlambatan pada saat dibutuhkan.

2.9.2 Pengolahan Data

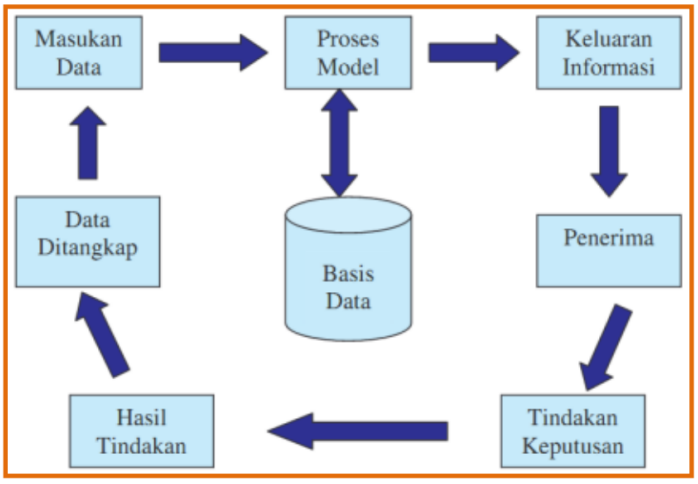
Pengolahan data adalah masa atau waktu yang digunakan untuk mendeskripsikan perubahan bentuk data menjadi in*Form*asi yang memiliki

kegunaan. Ada beberapa operasi yang dilakukan dalam pengolahan data, antara lain sebagai berikut [11]:

1. Data masukan, adalah kumpulan data transaksi ke sebuah pengolahan data medium.
2. Data trans*Form*asi, merupakan penghitungan/pengelompokan terhadap kelompok-kelompok tertentu. Beberapa bentuk data trans*Form*asi :
   a. Kalkulasi operasi aritmatik terhadap data field
   b. Menyimpulkan proses akumulasi beberapa data
   c. Melakukan klasifikasi terhadap data group-group tertentu
3. In*Form*asi keluaran, adalah proses menampilkan in*Form*asi.

2.9.3 Siklus In*Form*asi

Untuk memperoleh in*Form*asi yang bermanfaat bagi penerimanya, maka perlu dijelaskan bagaimana siklus yang terjadi atau dibutuhkan dalam menghasilkan in*Form*asi. Al-Bahra (2005:11)



Gambar 2. 16 Siklus In*Form*asi [11]

2.9.4 Kualitas In*Form*asi

Kualitas in*Form*asi sangat dipengaruhi atau ditentukan oleh hal-hal sebagai berikut [11]:

1. Relevan (*relevacy*), yaitu sejauh mana tingkat relevansi in*Form*asi tersebut terhadap kenyataan kejadian masa lalu, masa sekarang dan kejadian yang akan datang.
2. Akurat (*accuracy*), yaitu suatu in*Form*asi dikatakan berkualitas jika seluruh kebutuhan in*Form*asi telah tersampaiakan serta pesan yang disampaiakan sudah lengkap sesuai dengan yang diinginkan oleh *user*.
3. Tepat Waktu (*timelines*), in*Form*asi yang datang pada penerima tidak boleh terlambat. Keterlambatan suatu in*Form*asi bisa berakibat fatal bagi suatu organisasi atau pemakainya hal ini dikarenakan in*Form*asi merupakan landasan dalam pengambilan keputusan.
4. Ekonomis (*economy*), in*Form*asi yang dihasilkan harus mempunyai daya jual yang tinggi dan biaya operasional yang harus dikeluarkan untuk

menghasilkan in*Form*asi tersebut harus minimal, in*Form*asi tersebut juga mapu memberikan dampak yang luas terhadap laju pertumbuhan ekonomi dan teknologi in*Form*asi.

5. Efisien (*efficiency*), in*Form*asi yang berkualitas harus memiliki kalimat yang sederhana dan mudah dimengerti, tapi bisa memberikan makna yang mendalam.
6. Dapat dipercaya (*reliability*), in*Form*asi yang didapat harus dari sumber yang bisa dipercaya.

2.10 Konsep Dasar Sistem In*Form*asi

Sistem in*Form*asi dapat didefinisikan sebagi suatu sistem yang dibuat oleh manusia yang terdiri dari komponen-komponen dalam organisasi untuk mencapai suatu tujuan yaitu menyajikan in*Form*asi.

2.10.1 Definisi Sistem In*Form*asi

"Sistem In*Form*asi adalah suatu sistem didalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan" [17].

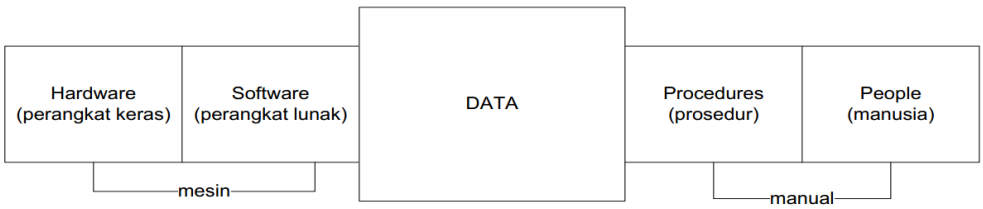
Sistem in*Form*asi adalah kumpulan data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti baik yang menerimanya [14]. Sistem in*Form*asi dapat didefinisikan sebagai berikut [11]:

1. Suatu sistem yang dibuat oleh manusia yang terdiri dari komponen-komponen dalam organisasi untuk mencapai suatu tujuan yaitu menyajikan in*Form*asi.
2. Sekumpulan prosedur organisasi yang pada saat dilaksanakan akan memberikan in*Form*asi bagi pengambil keputusan dan / atau untuk mengendalikan organisasi.
3. Suatu sistem dalam organisasi yang mempertemukan kebutuhan pengolahan transaksi, mendukung operasi, bersifat manajerial, dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan.

2.10.2 Komponen Sistem In*Form*asi

Adapun beberapa elemen / komponen dalam sistem in*Form*asi dapat diklasifikasikan sebagai berikut [11]:

1. *Hardware* dan *software* yang berfungsi sebagai mesin
2. *People* dan *procedures* yang merupakan manusia dan tata cara menggunakan mesin.
3. Data merupakan jembatan penghubung antara manuia dan mesin agar terjadi suatu proses pengolahan data.
4. *Input*, *Output*, *Process*, *Storage*, dan *Control*.



Gambar 2. 17 Lima Komponen Sistem In*Form*asi [11]

Adapun kegiatan sistem in*Form*asi [11]:

1. *Input* :

   Menggambarkan suatu kegiatan untuk menyediakan data untuk diproses.

2. Proses :

   Menggambarkan bagaimana suatu data diproses untuk menghasilkan suatu in*Form*asi yang berilai tambah.

3. *Output* :

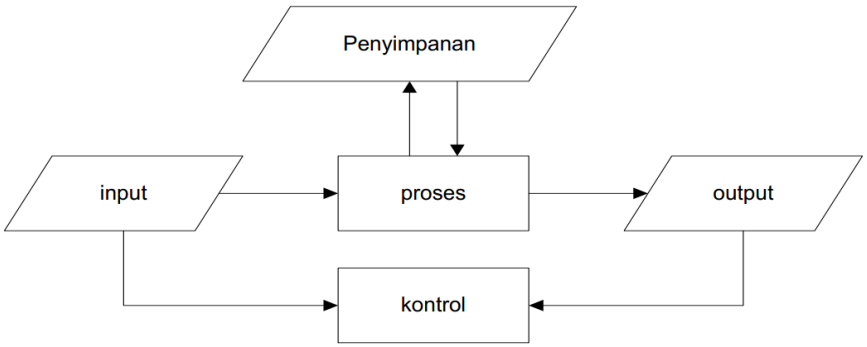
   Suatu kegiatan untuk menghasilkan laporan dari proses diatas tersebut.

4. Penyimpanan :

   Suatu kegiatan untuk memelihara dan menyimpan data.

5. Kontrol :

   Suatu aktivitas untuk menjamin bahwa sistem in*Form*asi tersebut berjalan sesuai dengan yang diharapkan.



Gambar 2. 18 Kegiatan Sistem In*Form*asi [15]

2.10.3 Siklus Hidup Sistem In*Form*asi

Siklus hidup sistem in*Form*asi dimulai dari perencanaan, pengembangan (survei, analisa, desain, pembuatan, implementasi, pemeliharaan) dan dievaluasi secara terus menerus untuk mendapatkan apakah sistem in*Form*asi tersebut masih

layak diaplikasikan, jika tidak, sistem in*Form*asi tersebut akan diganti dengan yang baru dan dimulai dari perencanaan kembali.

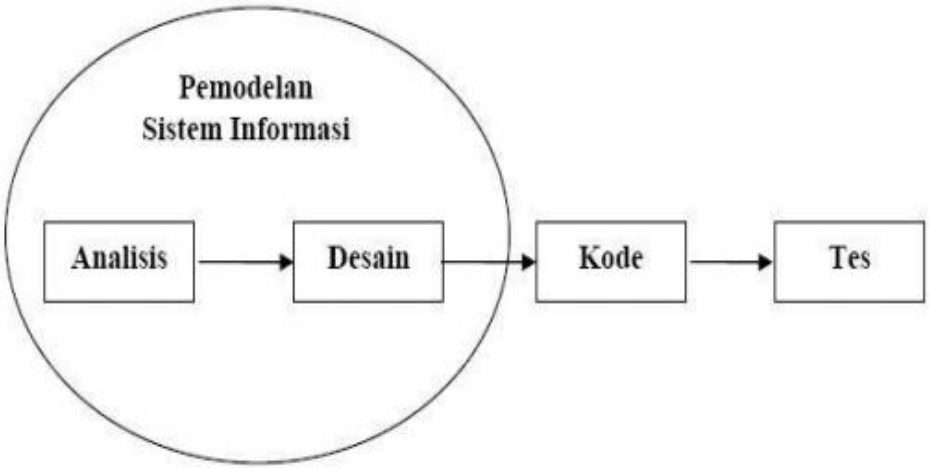


Gambar 2. 19 Siklus Hidup Sistem In*Form*asi [18]

2.11 Metode *Waterfall*

Adapun metode pengembangan yang digunakan adalah *waterfall*. *Waterfall* model pertama kali diperkenalkan oleh Winston Royce tahun 1970. *Waterfall* Model merupakan model klasik yang sederhana dengan aliran sistem yang linier. Output dari setiap tahap merupakan input bagi tahap berikutnya.

Model ini telah diperoleh dari proses rekayasa lainnya dan menawarkan cara pembuatan rekayasa perangkat lunak secara lebih nyata. Model ini melibatkan tim SQA (*Software Quantity Assurance*) dimana setiap tahapan selalu dilakukan verifikasi atau *testing*.



Gambar 2. 20 Model *Waterfall* [1]

a. Pemodelan Sistem In*Form*asi

Merupakan bagian dari sistem yang terbesar dalam pengerjaan suatu proyek, dimulai dengan menetapkan berbagai kebutuhan dari semua elemen yang diperlukan sistem dan mengalokasikannya kedalam pembentukan perangkat lunak.

b. Analisis

Merupakan tahap menganalisis kebutuhan data dan proses-proses dalam pelaksanaan proyek pembuatan perangkat lunak.

c. Desain

Tahap penerjemahan dari data yang dianalisis kedalam bentuk yang mudah dimengerti oleh pengguna.

d. Kode

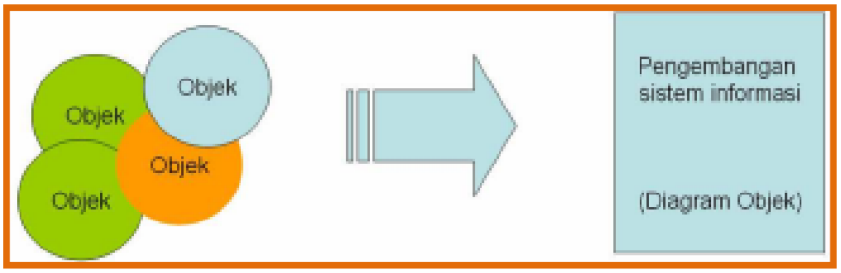
Tahap penerjemahan data atau pemecahan masalah yang telah dirancang keadalam bahasa pemrograman tertentu.

e. Tes

Merupakan tahap pengujian terhadap perangkat lunak yang dibangun agar aplikasi tersebut sesuai dengan perancangan.

2.12 Metodologi Berorientasi Objek

Metodologi ini diperkenalksan sekitar tahun 1990 sebagai pelengkap untuk pemrograman yang terlebih dahulu telah mengadopsi metode berorientasi objek. Beberapa alat dan teknik yang dapat digunakan antara lain *dynamic* dan *static object-oriented* model, *state transisition* diagram dan *case scenario*. Fokus utama metodologi ini pada objek, dengan melihat suatu sistem terdiri dari objek yang saling berhubungan dengan beberapa cara untuk mencapai suatu tujuan. Objek dapat digambarkan sebagai benda, orang, tempat dan sebaginya yang mempunyai atribut dan *method*. Obyek dalam "*software analysis & design*" adalah sesuatu berupa konsep (*concept*), benda (*thing*), dan sesuatu yang membedakannya dengan lingkungannya. Secara sederhana obyek adalah mobil, manusia, alarm dan lain-lainnya. Tapi obyek dapat pula merupakan sesuatu yang abstrak yang hidup didalam sistem seperti tabel, *database*, *event*, *system messages*. Obyek dikenali dari keadaannya dan juga operasinya. Sebagai contoh sebuah mobil dikenali dari warnanya, bentuknya, sedangkan manusia dari suaranya. Ciri-ciri ini yang akan membedakan obyek tersebut dari obyek lainnya.



Gambar 2. 21 Fokus Utama *Object Oriented* [12]

Object Oriented Programming merupakan cara atau metode baru dalam membuat program. Langkah membuat program dengan teknik OOP, pertama kali mengetahui objeknya, kemudian atribut (field/data) dan perilakunya (fungsi). Berbeda dengan pemrograman prosedural (konvensional), membuat program memakai aspek *input*, proses, dan *output*. OOP adalah cara berpikir, pandangan, atau paradigma baru untuk membuat program/merancang sistem dengan memperhatikan objek, cirri objek, dan perilakunya. OOP ini sangat berbeda dengan program prosedural yang fokusnya aspek input, proses, dan output.

2.12.1 Karakteristik OOAD

Dalam pendekatan berorientasi objek ada 4 pilar utama yang harus dipahamai dalam pendekatan berorientasi objek yaitu karakteristik. Karakteristik (ciri) suatu program termasuk OOAD/OOP, apabila terdapat abstraksi, pembungkusan (*encapsulation*), *polymorphisme*, dan turunan (*inheritance*).

1. *Abstraction*

   Kemampuan untuk menjadikan dalam bentuk yang lebih sederhana. Hal ini juga dikenal dalam metodologi pendekatan struktur yaitu dekomposisi seperti menyerderhanakan suatu sistem dalam bentuk *Context Diagram*.

2. *Encapsulation*

   Merupakan suatu karakteristik OOAD dimana program terbungkus (jadi satu) data dan perilaku, artinya lebih memperhatikan aspek internal daripada aspek eksternal. Contoh: dalam program terdapat tombol button close didalamnya ada *method* system.exit(0) untuk keluar dari sistem java. Berbeda dengan metodologi terdahulu, metodologi ini menggabungkan atribut dan fungsi / proses kedalam suatu objek yang disebut dengan encapsulation. Setiap objek dapat "menyembunyikan" kompleksitasnya dan berhubungan dengan objek lain dengan mengirim "pesan / *message*" yang dapat dikenal dan diproses oleh objek penerima. Contoh: Pada dunia nyata, seorang ibu rumah tangga menanak nasi dengan menggunakan *rice cooker*, ibu tersebut menggunakannya hanya dengan menekan tombol. Tanpa harus tahu bagaimana proses itu sebenarnya terjadi. Disini terdapat penyembunyian in*Form*asi milik *rice cooker*, sehingga tidak perlu diketahui seorang ibu. Dengan demikian menanak nasi oleh si ibu menjadi sesuatu yang menjadi dasar bagi konsep in*Form*ation hiding.

3. *Polymorphisme*

   Dengan kata lain suatu mekanisme yang memungkinkan suatu objek memiliki semua atau sebagian definisi dari objek induk. Menurut

Bambang Hariyanto (2007:67) *Polymorphism* berasal dari kata *Poly* yang artinya banyak dan *morph* yang artinya bentuk. Jadi *polymorphism* adalah kemampuan suatu atribut atau *method* dapat berubah dalam berbagai bentuk dalam implementasi. Contoh Pada obyek mobil, walaupun minibus dan truk merupakan jenis obyek mobil yang sama, namun memiliki juga perbedaan. Misalnya suara truk lebih keras dari pada minibus, hal ini juga berlaku pada obyek anak (*child*) melakukan metoda yang sama dengan algoritma berbeda dari obyek induknya. Hal ini yang disebut *polymorphism*, teknik atau konsep dasar lainnya adalah ruang lingkup/pembatasan. Artinya setiap obyek mempunyai ruang lingkup kelas, atribut, dan metoda yang dibatasi.

4. *Inheritance*

Merupakan suatu karakteristik OOAD di mana suatu kelas (*parent/base class*) dapat diturunkan ke kelas lain (*child/derived class*), sehingga kelas anak dapat memiliki data atau perilaku kelas orangtuanya. Contoh dengan beberapa buah mobil yang mempunyai kegunaan yang berbeda-beda. Ada mobil bak terbuka seperti truk, bak tertutup seperti sedan dan minibus. Walaupun demikian obyek-obyek ini memiliki kesamaan yaitu teridentifikasi sebagai obyek mobil, obyek ini dikatakan obyek induk (*parent*). Sedangkan minibus obyek anak (*child*), berarti semua operasi yang berlaku pada mobil berlaku pada minibus.

2.12.2 Ciri-ciri Object Oriented Programming (OOP)

Object Oriented memiliki cirri-ciri sebagai berikut [12]:

1. Objek

   Bentuk baik yang nyata atau tidak, seperti manusia, hewan, benda, konsep, aliran, dan lain-lain. Objek merupakan inisiasi (turunan langsung) dari suatu kelas.

2. Kelas

   Kumpulan objek yang memiliki kemiripan perilaku (*method*), ciri atau karakteristik (*property*). Contoh objek orang dari kelas manusia, potongan sebagai berikut: Manusia orang1=new manusia(“esson”);

3. *Method*

   Perilaku dari objek atau kelas tertentu. Merupakan perwujudan aksi atau tindakan dari dunia nyata di dalam pemrograman komputer.

4. Konstruktor

   Suatu fungsi yang dideklarasikan atau didefinisikan di dalam kelas, konstruktor harus mempunyai nama yang sama dengan fungsinya. Konstruktor dijalankan bersamaan dengan terciptanya kelas tersebut. Dalam suatu kelas bias terdapat lebih dari satu konstruktor. Konstruktor seperti *method* tetapi tidak mengembalikan nilai dan dapat didefinisikan tanpa parameter atau memakainya.

5. Dekonstruktor

   Fungsi yang dideklarasikan dalam kelas, nama sama dengan nama fungsinya. Tetapi dijalankan bersamaan dengan dimusnahkannya kelas tersebut.

6. Karakteristik / *properties*

   Ciri yang dimiliki oleh suatu objek, karakteristik ini juga sebagai pembeda objek satu dengan objek lainnya dalam kelas yang sama (konsep individu).

7. Variabel

   Tempat menampung data sementara, dalam pemrograman objek biasanya disebut data, sedangkan dalam pemrograman prosedural sering disebut dengan variabel.

8. Data

   Istilah lain dari variabel dalam OOP. Dalam pemrograman java bisa juga disebut *field*, data member atau *instance variable*.

9. Hak akses (*access atribute*)

   Hak akses digunakan untuk dapat menentukan data member mana yang dapat digunakan oleh kelas lain, dan mana yang tidak dapat digunakan. Hak akses ini sangat penting dalam membuat program turunan kelas.

a. Public

Data member atau variable dapat diakses dari kelas mana saja

b. Protected

Dapat mengakses data member dari kelas dalam *package* yang sama dan subkelasnya.

c. *Private*

Kelas yang data membernya memakai *private* hanya dapat digunakan oleh kelas bersangkutan, tidak dapat digunakan kelas lain

d. Tidak disebutkan

Data member dapat diakses dari kelas dalam *package* yang sama.

2.13 *Unified Modeling Language*

Untuk mendapatkan spesifikasi perangkat lunak yang sesuai dengan keinginan dan kebutuhan pengguna, para pengembang melakukan pemodelan-pemodelan secara *visual*. Langkah ini sering dinamakan pemodelan *visual*. Pemodelan *visual* adalah proses penggambaran in*Form*asi-in*Form*asi secara grafis dengan notasi-notasi yang telah disepakati sebelumnya [19].

Tujuan utama pemodelan *visual* adalah untuk memungkinkan adanya komunikasi antara pengguna, pengembang, penganalisis, *tester*, *manager*, dan siapapun yang terlibat dalam proyek, menunjukkan interaksi antara pengguna dengan sistem, obyek-obyek dalam sistem, dan antar sistem itu sendiri [20]. Untuk melakukan pemodelan sistem atau perangkat lunak, kita akan menggunakan notasi-notasi UML (*Unified Modeling Language*).

UML (*Unified Modeling Language*) adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan software berbasis OO (*Object-Oriented*). UML sendiri juga memberikan

standar penulisan sebuah sistem *blue print*, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema *database*, dan komponen-komponen yang diperlukan dalam sistem *software* [21].

Pada tahap analisis, meliputi usaha untuk mengetahui apa kemampuan sebuah sistem yang diinginkan pengguna dan pelanggan dari sebuah perangkat lunak. Beberapa teknik yang dapat membantu dalam tahapan analisis [21]:

Untuk itulah rancangan dibangun untuk memberikan gambaran. Semakin kompleks sebuah sistem semakin penting penggunaan teknik pemodelan yang baik. Dengan menggunakan model, diharapkan pengembangan piranti lunak dapat memenuhi semua kebutuhan pengguna dengan lengkap, cepat dan tepat. Adapun diagram utama dalam UML adalah:

1. *Use Case diagram*

   Digunakan untuk memodelkan bisnis proses berdasarkan perspektif pengguna sistem. *Use Case diagram* terdiri atas diagram untuk *Use Case* dan *actor*. Actor merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi. Use case merepresentasikan operasi-operasi yang dilakukan oleh actor. *Use Case* digambarkan berbentuk elips dengan nama operasi dituliskan di dalamnya. Actor yang melakukan operasi dihubungkan dengan garis lurus ke *Use Case*.

2. *Class Diagram*

   Merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. *Class Diagram* menunjukkan hubungan antar class dalam sistem yang sedang

dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan.

3. *Statechart Diagram*

   Untuk memodelkan perilaku objects di dalam sistem.

4. *Activity Diagram*

   Untuk memodelkan perilaku *Use Case*s dan objek di dalam sistem.

5. *Sequence Diagram*

   Menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *Use Case*: interaksi yang terjadi antar *class*, operasi apa saja yang terlibat, urutan antar operasi, dan in*Form*asi yang diperlukan oleh masing-masing operasi.

6. *Collaboration Diagram*

   Dipakai untuk memodelkan interaksi antar objek di dalam sistem. Berbeda dengan *Sequence Diagram* yang lebih menonjolkan kronologis dari operasi-operasi yang dilakukan, *collaboration diagram* lebih fokus pada pemahaman atas keseluruhan operasi yang dilakukan oleh object.

7. *Component Diagram*

   Untuk memodelkan komponen object.

8. *Deployment Diagram*

   Untuk memodelkan distribusi aplikasi.

Sedangkan pada tahapan perancangan / desain, dapat menggunakan notasi yang lebih banyak dan lebih tepat dalam notasi. Berikut adalah beberapa teknik dalam tahap perancangan [21].

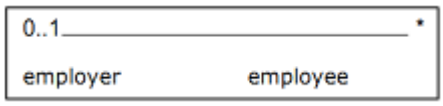
Pemodelan adalah proses merancang piranti lunak sebelum melakukan pengkodean, yang diwujudkan dalam bentuk bahasa model, model ini mampu menggambarkan persoalan dalam beberapa sudut pandang. Piranti lunak dapat dianalogikan seperti pembuatan *blueprint* pada pembangunan gedung. Membuat model dari sebuah sistem yang kompleks sangatlah penting, karena kadang kita sulit memahami sistem yang akan di bangun tersebut seperti apa, dan bagaimana cara kerjanya.

2.13.1 *Relationship*

Pertama, sebuah *dependency* adalah hubungan semantik antara dua benda/*things* yang mana sebuah benda berubah mengakibatkan benda satunya akan berubah pula. Umumnya sebuah *dependency* digambarkan sebuah panah dengan garis terputus-putus seperti terlihat dalam gambar di bawah ini.

Gambar 2. 22 *Dependency* [21]

Kedua, sebuah *association* adalah hubungan antar benda struktural yang terhubung diantara obyek. Kesatuan obyek yang terhubung merupakan hubungan khusus, yang menggambarkan sebuah hubungan struktural diantara seluruh atau sebagian. Umumnya assosiation digambarkan dengan sebuah garis yang dilengkapi dengan sebuah label, nama, dan status hubungannya.



Gambar 2. 23 *Association* [21]

Ketiga, sebuah *generalization* adalah menggambarkan hubungan khusus dalam obyek anak/*child* yang menggantikan obyek *parent* / induk. Dalam hal ini, obyek anak memberikan pengaruhnya dalam hal struktur dan tingkah lakunya kepada obyek induk.



Gambar 2. 24 *Generalization* [21]

Keempat, sebuah *realization* merupakan hubungan semantik antara pengelompokkan yang menjamin adanya ikatan diantaranya. Hubungan ini dapat diwujudkan diantara *interface* dan kelas atau *elements*, serta antara *Use Cases* dan *collaborations*.



Gambar 2. 25 *Realization* [21]

2.14 Perangkat Lunak Pendukung

Berikut ini merupakan penjelasan mengenai perangkat-perangkat lunak pendukung yang penulis pakai dalam merancang maupum membangun aplikasi.

2.14.1 JAVA

Pada 1991, sekelompok insinyur Sun dipimpin oleh Patrick Naughton dan James Gosling ingin merancang bahasa komputer untuk perangkat konsumer seperti cable TV Box. Karena perangkat tersebut tidak memiliki banyak memori, bahasa harus berukuran kecil dan mengandung kode yang liat. Juga karena

manufaktur – manufaktur berbeda memilih processor yang berbeda pula, maka bahasa harus bebas dari manufaktur manapun. Proyek diberi nama kode ”Green”.

Kebutuhan untuk fleksibilitas, kecil, liat dan kode yang netral terhadap *platForm* mengantar tim mempelajari implementasi Pascal yang pernah dicoba. Niklaus Wirth, pencipta bahasa Pascal telah merancang bahasa portabel yang menghasilkan *intermediate code* untuk mesin hipotesis. Mesin ini sering disebut dengan mesin maya (*virtual machine*). Kode ini kemudian dapat digunakan di sembarang mesin yang memiliki interpreter. Proyek *Green* menggunakan mesin maya untuk mengatasi isu utama tentang netral terhadap arsitektur mesin. Karena orang – orang di proyek *Green* berbasis C++ dan bukan Pascal maka kebanyakan sintaks diambil dari C++, serta mengadopsi orientasi objek dan bukan prosedural. Mulanya bahasa yang diciptakan diberi nama ”Oak” oleh James Gosling yang mendapat inspirasi dari sebuah pohon yang berada pada seberang kantornya, namun dikarenakan nama Oak sendiri merupakan nama bahasa pemrograman yang telah ada sebelumnya, kemudian SUN menggantinya dengan JAVA. Nama JAVA sendiri terinspirasi pada saat mereka sedang menikmati secangkir kopi di sebuah kedai kopi yang kemudian dengan tidak sengaja salah satu dari mereka menyebutkan kata JAVA yang mengandung arti asal bijih kopi. Akhirnya mereka sepakat untuk memberikan nama bahasa pemrograman tersebut dengan nama Java.

Java telah mengakomodasi hampir seluruh fitur penting bahasa – bahasa pemrograman yang ada semenjak perkembangan komputasi modern manusia :

1. Dari SIMULA, bahasa pada tahun 65-an, bahasa yang paling mempengaruhi Java sekaligus C++. Dari bahasa ini diadopsi bentukan – bentukan dasar dari pemrograman berorientasi objek.
2. Dari LISP – bahasa tahun 55-an. Diadopsi fasilitas *garbage collection*, serta kemampuan untuk meniru *generic list processing*, meski fasilitas ini jarang yang memanfaatkannya.
3. Dari Algol – bahasa pada tahun 60-an, diambil struktur kendali yang dimilikinya.
4. Dari C++, diadopsi sintaks, sebagian semantiks dan *exception handling*
5. Dari bahasa Ada, diambil *strongly type,* dan *exception handling*.
6. Dari Objective C, diambil fasilitas *interface*.
7. Dari bahasa SmallTalk, diambil pendekatan *single-root class hiérarchie*, dimana objek adalah satu kesatuan hirarki pewarisan
8. Dari bahasa Eiffel, fasilitas *assertion* yang mulai diterapkan di sebagian JDK 1.4

Berdasarkan *white paper* resmi dari SUN, Java memiliki karakteristik berikut :

1. Sederhana (*Simple*)

   Bahasa pemrograman Java menggunakan Sintaks mirip dengan C++ namun sintaks pada Java telah banyak diperbaiki terutama menghilangkan penggunaan *pointer* yang rumit *dan multiple inheritance*. Java juga menggunakan *automatic memory allocation* dan *memory garbage collection*.

2. Berorientasi objek (*Object Oriented*)

Java mengunakan pemrograman berorientasi objek yang membuat program dapat dibuat secara modular dan dapat dipergunakan kembali. Pemrograman berorientasi objek memodelkan dunia nyata kedalam objek dan melakukan interaksi antar objek-objek tersebut.

3. Terdistribusi (*Distributed*)

   Java dibuat untuk membuat aplikasi terdistribusi secara mudah dengan adanya *libraries networking* yang terintegrasi pada Java.

4. Interpreted

   Program Java dijalankan menggunakan *interpreter* yaitu *Java Virtual Machine* (JVM). Hal ini menyebabkan *source code* Java yang telah dikompilasi menjadi *Java bytecodes* dapat dijalankan pada plat*Form* yang berbeda-beda.

5. *Robust*

   Java mempuyai reliabilitas yang tinggi. *Compiler* pada Java mempunyai kemampuan mendeteksi *error* secara lebih teliti dibandingkan bahasa pemrograman lain. Java mempunyai *runtime-Exception handling* untuk membantu mengatasi *error* pada pemrograman.

6. *Secure*

   Sebagai bahasa pemrograman untuk aplikasi internet dan terdistribusi, Java memiliki beberapa mekanisme keamanan untuk menjaga aplikasi tidak digunakan untuk merusak sistem komputer yang menjalankan aplikasi tersebut.

7. *Architecture Neutral*

Program Java merupakan *platForm independent*. Program cukup mempunyai satu buah versi yang dapat dijalankan pada plat*Form* berbeda dengan *Java Virtual Machine*.

8. *Portable*

Source code maupun program Java dapat dengan mudah dibawa ke *platForm* yang berbeda-beda tanpa harus dikompilasi ulang.

9. *PerFormance*

PerFormance pada Java sering dikatakan kurang tinggi. Namun *perFormance* Java dapat ditingkatkan menggunakan kompilasi Java lain seperti buatan.

2.14.2 MySQL

MySQL adalah *Relational Database Management yistem* (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public Lisense*). Dimana setiap orang bebas untuk menggunakannya, tetapi tidak boleh dijadikan produk turunan yang bersifat komersial.

MySQL sebenarnya merupakan turunan salah satu konsep utama dalam *database* sejak lama, yaitu SQL (*Stuctur Query Language*). SQL adalah sebuah konsep pengoperasian *database*, terutama untuk pemilihan seleksi dan pemasukan data, yang kemungkinan pengoperasian data dikerjakan dengan mudah secara otomatis.

Keandalan suatu sistem *database* (DBMS) dapat diketahui dari cara kerja optimizernya dalam melakukan proses perintah-perintah SQL, yang dibuat oleh *user* maupun program-program aplikasi sebagai *database server*. Hal ini terbukti untuk *query* yang dilakukan oleh *single-user* , kecepatan

Query MySQL bisa sepuluh kali lipat lebih cepat dari postgreSQL dan lima kali lebih cepat dibanding interbase.

Sebagai *database server* yang memiliki konsep modern, MySQL memiliki banyak keistimewaan. Berikut ini beberapa keistimewaan yang dimiliki oleh MySQL :

1. *Portability*

   MySQL dapat berjalan stabil dalam berbagai sistem operasi seperti windows, Linux, FreeBSD, Mac OSX *Server*, dan lain-lain.

2. *Open Source*

   MySQL didistribusikan secara *open source*, sehingga dapat digunakan secara bebas.

3. *Multiuser*

   MySQL dapat digunakan oleh beberapa *user* dalam waktu yang bersamaan tanpa mengalami masalah. Hal ini memungkinkan sebuah *database server* MySQL dapat diakses *client* secara bersamaan.

4. *PerFormance Tuning*

   MySQL memiliki kecepatan yang tinggi dalam menangani *query*, dengan kata lain dapat memproses lebih banyak SQL persatuan waktu.

5. *Column Types*

MySQL memiliki tipe kolom yang sangat kompleks, seperti *signed/unsigned integer, float, double, char, varchar, blob, time, timestamp, year, set* , dan *enum*.

6. *Comand and Function*

   MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah SELECT dan WHERE dalam *query*.

7. Security

   MySQL memiliki lapisan sekuritas seperti level *subnetmask , namehost* , dan izin akses *user* dan dengan perizinan yang mendetail serta *password* ter- enkripsi.

8. *Scalabilitas* dan *Limit*

   MySQL mampu menangani *database* dengan skala besar, dengan jumlah *record* lebih dari 50 juta dan 60 juta tabel serta 5 milyar baris. Selain itu, batas indeks yang dapat ditampung mencapai 2 indeks pada setiap tabelnya.

9. *Connectivity*

   MySQL dapat melakukan koneksi dengan *client* menggunakan protocol TCP/IP, Unix Socet (UNIX), atau Named Pipes (NT).

10. *Localisaton*

    Deteksi pesan kesalahan pada *client* dengan menggunakan lebih dari bahasa.

11. *Interface*

    Interface terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (*Aplication Programming Interface*).

12. Client and Tools

    Dilengkapi dengan berbagai macam *tool* yang dapat digunakan untuk administrasi *database*, dan pada setiap *tool* yang akan disertakan petunjuk online.

13. Struktur Tabel

    Struktur Tabel yang lebih fleksibel dalam menangani ALTER TABEL, dibanding *database* lainnya semacam postgreSQL ataupun Oracle.

2.15 Keamanan Perangkat Lunak

Pada aplikasi ini keamanan perangkat lunak yang digunakan yaitu menggunakan *username* dan *password* yang menjadi syarat untuk dapat masuk menggunkan aplikasi ini. Pada tabel administrator tidak hanya terdapat *field username* dan *password* tapi juga terdapat field bagian yang menjadi pembeda hak akses pada aplikasi ini.

Password akan dienkripsi menggunakan MD5 bawaan MySQL. MD5 (*Message-Digest Algorithm 5*) adalah algoritma yang dibuat oleh Ron Rivest di tahun 1991, melakukan fungsi *hash* dengan menggunakan algoritma yang dijabarkan di RFC1321, "The MD5 Message-Digest Algorithm". Algoritma MD5 ini merupakan pengganti algoritma MD4 yang juga dibuat oleh Rivest. Hasil keluaran dari MD5 adalah sebuah nilai hash dalam 128-bit. Salah satu aplikasi

yang lazim menggunakan MD5 adalah pengamanan berkas *password* (/etc/passwd atau /etc/shadow) di sistem UNIX. Berkas *password* menyimpan data *password* dalam bentuk yang sudah terenkripsi dengan menggunakan DES. Implementasi awal dari sistem UNIX adalah menyimpan data *password* yang sudah terenkripsi tersebut langsung pada salah satu field di berkas *password*.

Meskipun sudah terenkripsi, penyimpanan data *password* yang sudah terenkripsi tersebut masih menimbulkan potensi lubang keamanan karena DES merupakan enkripsi yang *reversible*. Panjang data yang dihasilkan oleh proses enkripsi juga bergantung kepada panjang data yang dimasukkan. Sehingga ada sedikit info tambahan mengenai kemungkinan panjangnya *password*. Apabila seseorang berhasil mendapatkan berkas *password* tersebut, dia bisa mencoba proses dekripsi yaitu dengan melakukan *brute force attack* dengan mencoba melakukan proses dekripsi.

MD5 menambahkan satu tingkat keamanan lagi. Kali ini data *password* yang disimpan bukanlah *password* yang terenkripsi saja, melainkan data yang terenkripsi yang sudah dilewatkan oleh MD5. Karena sifatnya yang satu arah, sangat sulit untuk mencari data *password* terenkripsi dengan basis data hasil fungsi MD5. Jadi skema penyimpanan data tersebut kira-kira seperti ini:

password > DES > password terenkripsi > MD5 > hashed encrypted password

Dengan cara ini, potensi untuk melakukan serangan *brute force* terhadap *encrypted password* menjadi lebih sukar. Satu-satunya cara untuk melakukan serangan *brute force* adalah dengan melakukan enkripsi juga dengan melalui jalan

maju seperti di atas dan kemudian membandingkan hasil *hashed encrypted password*-nya. Jika sama persis, maka kata yang dipilih sebagai percobaan sama dengan *password* yang ingin dipecahkan tersebut.

MD5 juga digunakan dalam autentikasi dengan menggunakan protocol CHAP (RFC 1994). Masih ada banyak aplikasi lain yang menggunakan MD5 ini. Di tahun 1996 ditemukan kelemahan dari MD5 sehingga disarankan untuk menggantinya dengan menggunakan SHA-1. Di tahun 2004, ditemukan lagi kelemahan yang lebih serius sehingga penggunaan MD5 lebih dipertanyakan lagi. Xiaoyun Wang dan kawan-kawan menemukan kelemahan ini dan membuat makalah yang dipresentasikan di Crypto 2004. Mereka menunjukkan bahwa ada tabrakan (*collions*) dimana dua buah data menghasilkan keluaran hash MD5 yang sama. Selain collision di MD5, mereka juga menemukan hal yang sama di MD5, HAVAL-128, dan RIPEMD.

Tipe data MySQL yang digunakan yaitu InnoDB dimana InnoDB ini telah mendukung *transaction process* dan telah dirancang untuk aplikasi *database* yang cukup besar, dan perlindungan integritas data yang maksimum. *Transaction* adalah mekanisme yang mengendalikan atau kontrol atas manipulasi data yang beriringan (serentak) dan terintegrasi sebagai suatu kesatuan, contoh proses penyimpanan transaksi penjualan yang melibatkan tabel penjualan, detil penjualan, dan data stok barang. Dengan adanya mekanisme *transaction*, apabila ditemui kegagalan pada salah satu proses penyimpanan maka, semua proses penyimpanan sebelumnya bisa dibatalkan (*rollback*) sehingga integritas *database* tetap terjaga.