# BAB II

# TINJAUAN PUSTAKA

## 2.1. Tinjauan Perusahaan

Tinjauan perusahaan menjelasakan informasi lengkap mengenai PT. Panasia Group yaitu sejarah perusahaan, tempat dan kedudukan perusahaan, bentuk dan badan hukum perusahaan, bidang pekerjaan perusahaan, visi dan misi perusahaan serta struktur organisasi perusahaan.

### 2.1.1. Sejarah Perusahaan [24]

PT. Panasia Group didirikan pada tahun 1973 sebagai sebuah perusahaan produksi manufaktur yang memproduksi tekstil dengan kualitas tinggi. PT. Panasia Group awalnya bernama PT. Harapan Djaja Empat Saudara, kemudian berubah menjadi PT. Hadtex Indosyntec dan sekarang bernama PT. Panasia Group.

Sejak Tahun 1975 pabrik perusahaan berada di Jl. Moch Toha di selatan kota Bandung yang merupakan salah satu pusat tekstil di Indonesia. Saat ini PT. Panasia Group terus berkembang dan telah memiliki 60 hektar area di lokasi pabrik.

### 2.1.2. Tempat dan Kedudukan Perusahaan [24]

Perusahaan dan pabriknya berlokasi di Bandung di Jl. Moch. Toha Km. 6,8 Cisirung Bandung, Jawa Barat. Kantor pusat Perusahaan beralamat di Jl. Garuda 153/74, Bandung, Jawa Barat.

### 2.1.3. Bentuk dan Badan Hukum Perusahaan [24]

P.T. Panasia Group Tbk (Perusahaan) didirikan dalam rangka Undang-Undang Penanaman Modal Dalam Negeri No. 6 tahun 1968. Undang-Undang No. 12 tahun 1970 berdasarkan akta No. 13 tanggal 6 April 1973 dari Imas Fatimah, S.H., notaris di Bandung. Akta pendirian ini disahkan oleh Menteri Kehakiman Republik Indonesia dalam Surat Keputusannya No.Y.A.5/174/23 tanggal 11 Maret 1981 serta diumumkan dalam Berita Negara Republik Indonesia No. 16 tanggal 24 Pebruari 1987, Tambahan No. 171. Anggaran dasar Perusahaan telah mengalami beberapa kali perubahan, terakhir dengan akta No. 46 tanggal 22 Juni 2007 dari R. Tendy Suwarman, S.H., notaris di Bandung, mengenai perubahan pemegang saham dan susunan pengurus Perusahaan. Akta perubahan ini telah diterima dan dicatat di dalam data base Sisminbakum Departemen Hukum dan Hak Asasi Manusia Republik Indonesia dengan Surat No. W8-HT.01.10-2075 tanggal 18 September 2007.

### 2.1.4. Bidang Pekerjaan Perusahaan [24]

Sesuai dengan pasal 3 anggaran dasar Perusahaan, ruang lingkup kegiatan Perusahaan terutama meliputi usaha dalam bidang proses bahan baku serat (polimerisasi), *twisting*, industri tekstil, dan perdagangan umum. Perusahaan mulai berproduksi secara komersial dalam industri tekstil pada tahun 1974, sedangkan kegiatan pemrosesan bahan baku serat (polimerisasi) dimulai pada tahun 1990. Hasil produksi Perusahaan dipasarkan di dalam dan di luar negeri termasuk ke benua Eropa, Asia, Amerika, Australia dan Afrika.

### 2.1.5. Visi dan Misi Perusahaan [24]

PT. Panasia Group sebagai sebuah perusahaan memiliki visi dan misi perusahaan.

#### 2.1.5.1. Visi Perusahaan [24]

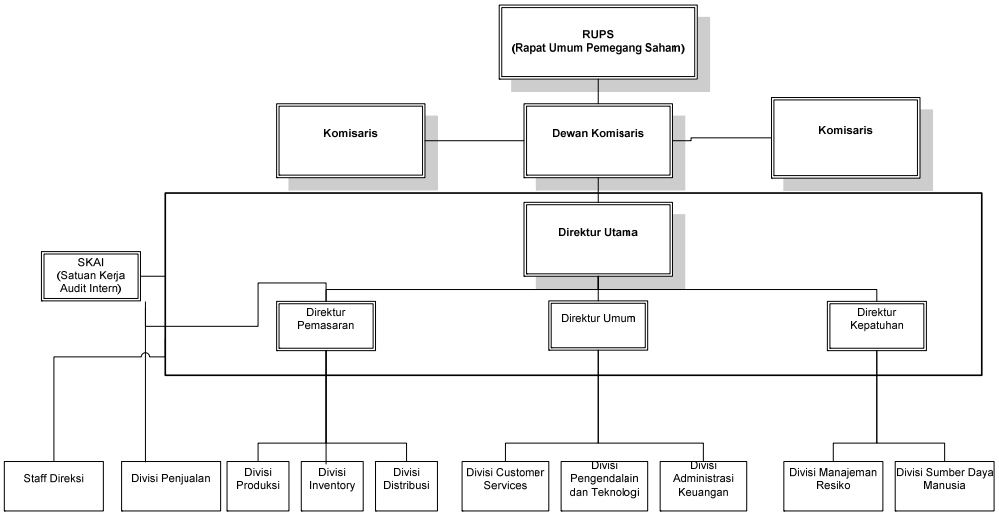
PT. Panasia Group berusaha untuk memahami dan menyelaraskan peluang menjadi keberhasilan dan dengan membentuk kerjasama yang antara perusahaan dengan mitra dan pelanggan

#### 2.1.5.2. Misi Perusahaan [24]

PT. Panasia Group berusaha untuk mengoptimalkan sumber daya yang dimiliki perusahaan untuk peningkatan tingkat support dan kepuasan pelanggan, sekaligus meningkatkan pengembangan perusahaan yang memberntuk nilai ekonomi yang kompetitif.

### 2.1.6. Struktur Organisasi Perusahaan [24]

Organisasi dan Tata Kerja PT. Panasia Group ditetapkan dengan Surat Keputusan Direksi Nomor : DIR/095/KP tanggal 30 Desember 1992 tentang struktur organisasi dan tata kerja PT. Panasia Group. Surat Keputusan tersebut mengatur juga tentang pembagian tugas dan pemisahan fungsi yang ada. Susunan organisasi PT. Panasia Group dapat dilihat pada gambar 2.1.




Gambar 2.1. Struktur Organisasi PT. Panasia Group [24]

## 2.2. Landasan Teori

Landasan teori akan menjelaskan beberapa teori yang berkaitan dengan permasalahan yang akan dibahas sebagai dasar dalam pemahaman dalam sebuah sistem serta metode yang dipakai untuk kegiatan pengembangan terhadap sistem itu sendiri.

### 2.2.1. Sistem Informasi

#### 2.2.1.1. Definisi Sistem

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu [13].

#### 2.2.1.2. Definisi Informasi

Informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya [13].

#### 2.2.1.3. Definisi Sistem Informasi

Sistem Informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan [13].

#### 2.2.1.4. Aktivitas Sistem Informasi

Sistem informasi memiliki beberapa aktivitas utama, yaitu :

1. Melaksanakan komputasi numerik, bervolume besar, dan dengan kecepatan tinggi.
2. Menyediakan komunikasi dalam organisasi atau antar organisasi yang murah, akurat, dan cepat.
3. Menyimpan informasi dalam jumlah yang sangat besar dalam ruang yang kecil tetapi mudah diakses.
4. Memungkinkan pengaksesan informasi yang sangat banyak di seluruh dunia dengan cepat dan murah.
5. Meningkatkan efektivitas dan efisiensi orang-orang yang bekerja dalam kelompok, dalam suatu tempat, atau pada beberapa lokasi.
6. Menyajikan informasi dengan jelas yang menggugah pikiran manusia.

7. Mengotomasi proses-proses bisnis yang semiotomatis dan tugas-tugas yang dikerjakan secara manual.
8. Mempercepat pengetikan dan penyuntingan.
9. Pembiayaan yang jauh lebih murah daripada pengerjaan secara manual.
10. Pembiayaan yang jauh lebih murah daripada pengerjaan secara manual.

Dalam suatu sistem informasi terdapat komponen-komponen seperti perangkat keras (hardware), perangkat lunak (software) atau program, prosedur (aturan), orang, basis data (database), serta jaringan komputer dan komunikasi data. Pada prakteknya, tidak semua sistem informasi mencakup keseluruhan komponen tersebut. Komponen-komponen yang terlibat tergantung pada bentuk sistem informasi yang digunakan, yaitu sederhana atau lebih kompleks [13].

**2.2.1.5. Karakteristik Sistem Informasi**

Sebuah informasi yang berkualitas diataranya memilikki ciri-ciri [13]:

1. Akurat, berarti informasi harus bebas dari kesalahan-kesalahan dan tidak dapat menyesatkan penerima informasi.
2. Tepat waktu, berarti informasi yang akan disampaikan kepada penerima informasi tidak boleh datang terlambat.
3. Relevan, berarti informasi mempunyai nilai manfaat bagi penerima informasi.
4. Lengkap, berarti informasi sesuai dengan apa yang dibutuhkan.

### 2.2.1.6. Komponen Sistem Informasi

Sistem informasi terdiri dari komponen-komponen yang disebutnya dengan istilah blok bangunan (*building block*) yang terdiri dari [13]:

1. Blok masukan; berupa input mewakili data yang masuk ke dalam sistem informasi. Input disini termasuk metode-metode dan media untuk menangkap data yang akan di diproses dalam sistem, yang dapat berupa dokumen dasar.
2. Blok Model; terdiri dari kombinasi prosedur, logika dan model matematika yang akan memanipulasi data input dan data yang tersimpan di basis data dengan cara tertentu untuk menghasilkan keluaran yang diinginkan.
3. Blok kendali; Digunakan supaya sistem informasi dapat berjalan sesuai dengan yang diinginkan, maka perlu diterapkan pengendalian didalamnya dengan tujuan uuntuk meyakinkan bahwa hal-hal yang dapat merusak sistem dapat dicegah dan jika terdapat kesalah dapat langsung diatasi.
4. Blok keluaran; hasil dari sistem adalah keluaran yang merupakan informasi yang berkualitas dan dokumentasi yang berguna untuk pemakai .
5. Blok teknologi; teknologi digunakan untuk menerima input, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan informasi keluaran serta membantu pengendalian sistem secara keseluruhan.
6. Blok basis data; kumpulan dari data yang saling berhubungan satu dengan lainnya, tersimpan di perangkat keras (hardware) komputer dan digunakan perangkat lunak (software) untuk memanipulasinya. Data perlu disimpan di dalam basis data untuk keperluan penyediaan informasi lebih lanjut karena

diorganisasikan dengan baik supaya informasi yang dihasilkan berkualiatas, akurat, dan upto date.

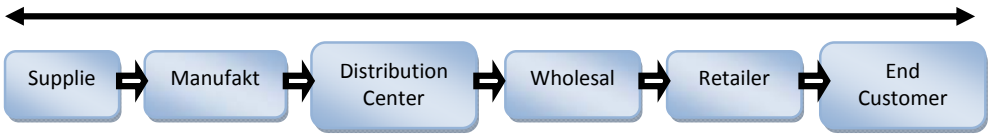
### 2.2.2. *Supply Chain Management* (SCM)

#### 2.2.2.1. Latar Belakang Munculnya Konsep *Supply Chain Management.*

Munculnya SCM dilatar belakangi oleh praktek tradisional dalam bisnis serta perubahan lingkungan bisnis [20].

##### 2.2.2.1.1. Praktek Tradisional

Produk atau jasa yang kita gunakan adalah hasil dari serangkaian proses panjang yang melewati beberapa tahapan fisik maupun non fisik. Sebuah produk akan sampai ke tangan pemakai akhir setelah setidaknya melalui beberapa proses dari pencarian bahan baku, proses produksi, dan proses distribusi atau transportasi. Proses-proses ini melibatkan berbagai pihak yang berhubungan antara satu dengan yang lainnya. Penyedia bahan baku (pemasok) mensuplai kebutuhan produksi para perusahaan manufaktur yang akan mengolah bahan baku tersebut menjadi produk jadi. Produk jadi disampaikan ke pemakai akhir lewat pusat-pusat distribusi, ritel, pedagang kecil, dan sebagainya. Rangkaian pihak-pihak yang menangani aliran produk inilah yang dinamakan dengan istilah *Supply Chain* (SC).Gambar 2.1 memberikan ilustrasi sebuah *supply chain* yang sederhana [20].




Gambar 2. 2. Struktur Supply Chain yang disederhanakan [20]

Sebuah SC akan memiliki komponen-komponen yang biasanya disebut *channel*. Misalnya ada *supplier*, manufaktur, *distribution center, wholesaler*, dan *retailer*. Semua *channel* tersebut bekerja untuk memenuhi kebutuhan konsumen akhir. Pada kenyataannya, struktur sebuah SC mungkin jauh lebih komplek dari gambar di atas. Sebuah pemasok mungkin sekaligus adalah industri manufaktur. Dengan kata lain, sebuah SC bisa saja melibatkan sejumlah industri manufaktur dalam satu rantai hulu ke hilir. Demikian juga, SC tidak selalu merupakan rantai lurus. Sebuah industri manufaktur bias memiliki ratusan bahkan ribuan pemasok. Produk-produk yang dihasilkan oleh sebuah industri mungkin didistribusikan oleh beberapa pusat distribusi yang melayani ratusan bahkan ribuan *wholesaler* dan *ritel*, pedagang kecil dan sebagainya. Setiap *channel* dalam SC akan memiliki aktivitas-aktivitas yang saling mendukung. Secara keseluruhan aktivitas-aktivitas tersebut meliputi perancangan produk, peramalan kebutuhan, pengadaan material, produksi, pengendalian persediaan, distribusi/transportasi, penyimpanan/pergudangan, dukungan pelayanan kepada pelanggan, proses pembayaran, dan sebagainya. Pada tingkatan yang lebih strategis ada aktivitas-aktivitas seperti pemilihan pemasok, penentuan lokasi pabrik, gudang, pusat distribusi, dan sebagainya. Secara tradisonal, semua aktivitas-aktivitas tersebut dilakukan tanpa atau dengan sedikit koordinasi. Istilah *cross-fucsional team* misalnya, tidak banyak diaplikasikan dalammanajemen *supply chain* tradisional. Tiap bagian berusaha membuat ukuranukuran tersendiri dalam menentukan kesuksesan pekerjaannya. Demikian juga hubungan antar *channel* dalam *supply chain*. Hubungan antara pemasok dengan perusahaan yang disuplainya juga hanya

terbatas pada transaksi jual beli. Pola-pola negosiasi benar-benar mementingkan pihak-pihak secara individual, dan bukan mengacu pada kinerja keseluruhan pihak yang menjadi pembentuk sebuah *supply cahin* secara holistik. Pemasok berkeinginan untuk memindahkan atau menjual produknya secepat dan sebanyak mungkin dengan arga yang tinggi, sementara perusahaan yang disuplainya menginginkan harga yang murah dan pengiriman ang cepat. Pola hubungan seperti ini dinamakan adversarial [20].

**2.2.2.1.2. Perubahan Lingkungan Bisnis**

Lingkungan bisnis senantiasa berubah dan perubahan tersebut semakin lama semakin cepat. Akselerasi perubahan ini disebabkan berkembangnya secara cepat faktor-faktor penting antara lain [20] :

1. Konsumen yang semakin kritis, membutuhkan produk atau jasa yang semakin berkualitas dengan harga murah dan bisa diperoleh dengan mudah dan cepat.
2. Infrastruktur telekomunikasi, informasi,transportasi dan perbankan yang semakin canggih sehingga memungkinkan berkembangnya model-model baru dalam manajemen aliran material/produk. Munculnya internet misalnya, memungkinkan terjadinya transaksi-transaksi elektronik yang dikenal dengan nama Elektronik Commerce (*E-Commerce*). Praktek *E-Commerce* dapat dilakukan karena informasiinformasi tersedia dan mudah diakses lewat internet, pembayaran secara aman bisa dilakukan secara aman dan cepat dengan menggunakan jasa pihak ketiga.

3. Kesadaran akan pentingnya aspek sosial dan lingkungan. Kalangan bisnis semakin ditekan untuk memperhatikan aspek-aspek sosial dan lingkungan, baik atas instruksi pemerintah maupun atas kesadaran kalangan bisnis sendiri bahwa bisnisnya tergantung pada konsumen yang semakin tahu akan pentingnya aspek lingkungan dalam hidup mereka. Industri manufaktur dewasa ini telah banyak yang memasukkan konsep-konsep keramahan pada lingkungan mulai dari proses perancangan produknya, proses produksi, sampai pada proses distribusinya. Ketiga faktor diatas, ditambah dengan adanya lobalisasi dan perubahan peta ekonomi dunia ke arah meningkatnya kemampuan ekonomi negara-negaradunia ketiga, telah menciptakan banyak paradigma baru dalam dunia bisnis. Salah satu aradigma enting adalah meningkatnya persaingan antar produk maupun jasa di pasaran. Hanya produk atau jasa ang aspiratif terhadap kepentingan konsumen yang pada akhirnya akan bisa bertahan.Perusahaan- prusahaan ini ternyata tidak bisa dilepaskan dari dukungan berbagai pusat ilmu pengetahuan seperti perguruan tinggi, lembaga-lembaga riset, dan sebagainya. Lembagalembaga seperti ini banyak elakukan kajian-kajian maupun pengembanganpengembangan inovatif terhadap proses-proses bisnis. alam kaitannya dengan logistik misalnya, lembaga riset yang memfokuskan penelitiannya pada bidang- idang *supply chain management* (SCM) menjamur dengan cepat. Beberapa diantaranya adalah *Global Supply Chain Management Forum* yang dimotori oleh Stanford University dengan mitranya di Endhoven niversity of Technology untuk kawasan

Eropa dan National (HKUST) untuk kawasan Asia. Disamping itu da juga The Logistic Institute of Technology, Center for Transportation Studies di MIT, dan banyak lagi. Dengan praktek tradisional bisnis yang tidak *compatible* lagi dan persaingan yang semakin ketat akibat perubahan-perubahan lingkungan bisnis, memaksa pelaku-pelaku, baik sektor industri maupun jasa untuk memikirkan cara-cara baru dalam memenangkan persaingan. *Supply Chain Management* muncul sebagai jawaban atas kebutuhan pelayanan yang cepat, berkualitas dan murah [20].

### 2.2.3. Definisi *Supply Chain Management* (SCM)

Prinsip SCM pada hakekatnya adalah sinkronisasi dan koordinasi aktivitas-aktivitas yang berkaitan dengan aliran material/produk, baik yang ada dalam satu organisasi maupun antar organisasi. Aliran material/produk dalam satu organisasi, misalkan sebuah industri manufaktur, adalah sesuatu yangkomplek. Penanganannya membutuhkan campur tangan semua pihak, bukan hanya mereka- ereka ang dilalui langsung oleh aliran material/produksecara fisik, tetapi juga bagian-bagian lain seperti bagian erancangan produk, pemasaran, akuntansi, dan sebagainya. Pada praktek tradisional, bagian-bagian tersebut saling terpisah, bekerja dengan ukuran-ukuran sendiri. Pada SCM, semua bagian harus bekerja sama membentuk sebuah tim yang disebut dengan *cross functional eam*. Salah satu implementasi dari *cross functional team* adalah pada perancangan produk. Bagian emasaran, produksi, perencanaan proses, pengadaan material, dan lain-lain duduk bersama untuk embahas berbagai aspek dari rancangan produk tersebut sehingga akhirnya keluar produk baru yang enar-benar mencerminkan

selera konsumen dan bias diproduksi dengan cepat dan mudah. Konsep ini ikenal dengan istilah *Concurrent Engineering.*

Sinkronisasi aktivitas-aktivitas bukan hanya perlu pada agian-bagian internal organisasi. Pendekatan SCM sangat menyadari bahwa sebagian besar bisnis dari ebuah industri harus dikerjakan atas dasar kerja sama dengan pihak luar. Apabila perusahaan ingin ukses dalam kompetisinya, mau tidak mau kemampuannya bekerja sama dengan pihak luar harus ditingkatkan. Bahan baku yang sering menjadi komponen utama dari harga pokok produksi sebuah produk iperoleh dari para pemasok (pihak luar). Urusan pengiriman bahan baku dari pemasok maupun produk jadi ke para distributor sering kali menggunakan jasa pihak ketiga (pihak luar). Pembayaran trasaksi- ransaksi bisnis dengan pihak ketiga membutuhkan jasa perbankan (pihak luar). Teknologi dan sistem nformasi mungkin juga disediakan dan dipelihara oleh pihakketiga. Hampir semua aktivitas akhirnya harus berkaitan dengan pihak luar. Konsekuensinya, hanya perusahaan-perusahaan yang mampu menjalin dan memelihara hubungan dengan pihak luar tersebut yang akan bisa bertahan dalam persaingan pasar [20].

Beberapa pihak memberikan definisi yang berbeda tentang SCM. Dua diantaranya definisi tersebut, asingmasing yang diberikan oleh Profesor LalOnde dari Ohio State University dan yang diberikan oleh IT dalah sebagaiberikut : *The delivery of enchanced customer and economic value thraough synchronized management of the flow of physical goods and associated information from sourcing to a process rientation, integrated approach to procuring, producing, and delivering products and services to ustomers*. (MIT).

Fokus utama dari dua definisi tersebut adalah sinkronisasi proses untuk kepuasan pelanggan. Pelanggan dalam konsep SCM adalah “raja” yang harus diberikan pelayanan sebaikbaiknya. Semua *supply chain* pada hakekatnya memperebutkan pelanggan atau pemakai akhir dari produk atau asa yang ditawarkan. Pihak-pihak yang berada dalam satu rantai *supply chain* adalah pihak-pihak yang arus bekerja sama satu sama lain untuk sedapat mungkin meningkatkan pelayanan dengan harga yang urah. Persaingandalam konteks SCM adalah persaingan antar rantai, bukan antar individu perusahaan, elemahan praktek tradisional yang bersifat adversial adalah terfokusnya aktivitas maupun ukuran eberhasilan pada bagianbagian kecil dari *supplay chain* yang sering kali justru kontradiktif dengan tujuan khir untuk meningkatkan pelayanan kepada konsumen akhir atau pelanggan [20].

### 2.2.4. Fungsi *Supply Chain Management* (SCM)

Fungsi *Supply Chain Management* Ada dua fungsi SCM, yaitu [20]:

1. SCM secara fisik mengkonversi bagan baku menjadi produk jadi dan menghantarkannya ke pamakai akhir. Fungsi pertama ini berkaitan dengan ongkos-ongkos fisik, yaitu ongkos material, ongkos enyimpanan, ongkos produksi, ongkos transportasi, dan sebagainya.
2. SCM sebagai mediasi pasar, yakni memastikan bahwa apa yang disuplai oleh ratai *supply chain* mencerminkan aspirasi pelanggan atau pemakai akhir tersebut. Fungsi kedua ini berkaitan dengan biayabiaya survey pasar, perancangan produk, serta biaya-biaya akibat tidak terpenuhinya aspirasi konsumen oleh produk yang disediakan oleh sebuah rantai *supplay chain*.

Ongkos-ongkos ini bisa berupa ongkos *markdown*, yakni penurunan harga roduk yang tidak laku dijual dengan harga normal, atau ongkos kekurangan *supply* yang dinamakan dengan *stockout cost.*

### 2.2.5. Prinsip-Prinsip *Supply Chain Management* (SCM)

Menciptakan sinkronisasi aktivitas-aktivitas yang beragam membutuhkan pendekatan holisitk, tidak ubahnya seperti mensinkronkan alat-alat music dalam sebuah konser dimana alat yang bunyinya berbeda-beda bisa dimainkan bersama sehingga terdengar merdu. Prinsip utama yang harus dipegang dalam sinkronisasi aktivitas-aktivitas sebuah *supply chain* adalah untuk menciptakan resultan yang lebih besar, bukan hanya bagi tiap anggota rantai, tetapi bagi keseluruhan sistem. Kesuksesan implementasi prinsip ini biasanya membutuhkan perubahan-perubahan pada tingkatan strategis maupun taktis. Sebaliknya, kegagalan biasanya ditandai oleh ketidakmampuan manajemen mendefinisikan angkah-langkah yang harus ditempuh dalam menggiring komponen-komponen *supply chain* yang omplek e arah yang sama. Anderson, Britt, dan Favre (1997) memberikan 7 prinsip dalam SCM yang diperuntukkan bagi manajer dalam merumuskan keputusan strategis, yaitu [20] :

1. Segmentasi pelanggan berdasarkan kebutuhannya.
2. Sesuaikan jaringan logistik untuk melayani kebutuhan pelanggan yang berbeda.
3. Dengarkan signal pasar dan jadikan signal tersebut sebagai dasar dalam perencanaan kebutuhan *demand planning*) sehingga bisa menghasilkan ramalan yang konsisten dan alokasi sumber daya yang optimal.

4. Deferensiasi produk pada titik yang lebih dekat dengan konsumen dan percepat konversinya disepanjang rantai *supply chain*.
5. Kelola sumber-sumber suplai secara strategis untuk mengurangi ongkos kepemilikan dari material aupun jasa.
6. Kembangkan strategi teknologi untuk keseluruhan rantai *supplay chain* yang mendukung pengambilan keputusan berhirarki serta berikan gambaran yang jelas dari aliran produk, jasa, maupun informasi.
7. Adopsi pengukuran kinerja untuk sebuah *supply chain* secara keseluruhan dengan maksud untuk meningkatkan pelayanan kepada konsumen akhir.

### 2.2.3. *Customer Relationship Management* (CRM)

#### 2.2.3.1. Definisi *Customer Relationship Management* (CRM)

Setelah era digital yang selalu menggunakan "e-" didepan semua kata seperti e-commerce, e-book, e-sales, e-mail, dst, Customer Relationship Management atau CRM merupakan akronim atau singkatan yang paling populer dikalangan orang- orang sales dan marketing. Kalau di bahasa Indonesia-kan kira-kira adalah Manajemen Hubungan Pelanggan (MHP). Telaah perkatanya dalah sebagai berikut; Pelanggan atau *customer*, kata pertama, di kamus bahasa Inggris artinya adalah seseorang yang berulang kali atau teratur melakukan pembelian kepada seorang pedagang. Jadi pelanggan adalah orangnya (dalam definisi ini tidak disinggung entang kepuasan, mahal-murahnya pembelian, dll). Hubungan atau*relationship*, kata kedua, adalah bentuk komunikasi dua arah antara pembeli dan penjual. Manajemen, kata terakhir,

artinya pengelolaan (secara luas tanpa perlu menjabarkan detail bagaimana mengelola sesuatu). Jadi definisi diatas kalau digabungkan kira-kira menjadi pengelolaan hubungan dua arah antara suatu perusahaan dengan orang yang menjadi pelanggan di perusahaan tersebut [3].

CRM dalam perkembangannya juga bisa didefinisikan sebagai berikut:

a. CRM adalah sebuah istilah industri TI untuk metodologi, strategi, perangkat lunak software) dan atau aplikasi berbasis web lainnya yang mampu membantu sebuah perusahaan/organisaso untuk mengelola hubungannya engan para pelanggan
b. CRM adalah usaha sebuah perusahaan untuk berkonsentrasi menjaga pelanggan supaya tidak lari ke pesaing) dengan mengumpulkan segala bentuk interaksi pelanggan baik itu lewat telepon, email, masukan di situs atau hasil pembicaraan dengan staf sales dan marketing
c. CRM adalah sebuah strategi bisnis menyeluruh dari suatu perusahaan yang memungkinkan perusahaan tersebut secara efektif bisa mengelola hubungan dengan para pelanggan.

### 2.2.3.2. Sejarah *Customer Relationship Management* (CRM)

Dengan adanya internet, tembok penghalang yang membatasi pelaku bisnis baru untuk melakukan penetrasi pasar menjadi sangat kabur, kalau tidak bisa dikatakan tidak ada. Sekarang seorang pelaku bisnis baru bisa melakukan bisnisnya dengan rekanan dari luar kota, negara bahkan lintas benua (asumsi

penulis: bisnis dilakukan dengan legal an benar). Yang namanya ‘indie’ tidak asing lagi di dunia maya ini. Jutawan-jutawan baru ahir karena adanya internet. Tiba-tiba merek (baca: nama besar) tidak berarti lagi [3].

Di sisi ain, di pihak pelaku bisnis yang mapan dengan merek besar mereka, situasi diatas bisa adi menguntungkan atau malah merugikan. Perbedaan teknologi sudah tidak penting lagi arena para pemain baru bisa mendapatkan teknologi yang sama itu dengan gampang. adi kalau perusahaan besar masih mengandalkan produknya (product focus) niscaya ereka bakal tersusul oleh pemain-pemain baru itu. Hal ini dikarenakan faktor pembeda roduk pemain lama dan pemain baru tidak banyak lagi.

Sebagai contoh: sudah pernah itawari portabel VCD player bikinan negeri tirai bambu? Harga yang ditawarkan tidak bisa ntuk membeli walkman merek negeri sakura yang berteknologi auto reverse dan super- uper bass. Tetapi, umur produknya tidak beda jauh jika dipakai sama hati-hatinya. Eknologi lasernya sama persis. Bahkan *feature*-nya tidak kalah dan garansinya bisa iminta. Karena hal inilah perusahaan-perusahaan besar (enterprise) mencoba mengganti rah bisnis mereka fokus ke pelanggan (customer focus). Fakta mengatakan bahwa biaya ntuk mendapatkan pelanggan baru bisa 10 kali biaya untuk menjaga pelanggan yang udah ada.

Sementara itu pelanggan juga memiliki kecenderungan untuk tidak hanya transaksional, beli barang – dapat barang – selesai, tetapi ingin lebih, beli barang – memilih service – dapat barang dan service – beli lagi. Sehingga pelanggan menginginkan hubungan dua arah dengan perusahaan-perusahaan yang mampu

memberikan pelayanan ambahan tanpa terasa ‘mahal’. Selain kecenderungan naiknya kompetisi global diatas imana produk-produk yang ada sangat sulit dibedakan, kecenderungan lain yang melahirkan CRM adalah majunya teknologi manajemen data yang memungkinkan isimpannya jutaan data dalam satu sistem tunggal. Dua puluh tahun yang lalu, informasi entang pelanggan akan disimpan dalam dua puluh sistem yang berbeda dan tidak ada teknologi yang mampu mengelola informasi kompleks tentang pelanggan. Data yang ada isa dari telepon, email, faksimili, tulisan tangan staf sales dan marketing, kartu nama, dll. CRM berawal sekitar tahun 1990-an sebagai langkah logis setelah rame-rame Enterprise resource Planning (ERP). Satu faktor lagi yang melahirkan CRM adalah penelitian arketing oleh PIMS yang menyimpulkan bahwa “pelanggan yang menggerutu tidak puas kan bercerita kepada sekitar 7-10 orang temannya sedangkan pelanggan yang puas kan merekomendasikan perusahaan bersangkutan ke 3-4 teman mereka [3].

### 2.2.3.3. Tujuan *Customer Relationship Management* (CRM )

Tujuan CRM ada 3 yaitu [3]:

1. Menggunakan hubungan dengan pelanggan untuk meningkatkan keuntungan perusahaan
2. Menggunakan informasi untuk memberikan pelayanan yang memuaskan
3. Mendukung proses penjualan berulang kepada pelanggan.

### 2.2.3.4. Tahapan *Customer Relationship Management* (CRM )

Ada tiga tahapan CRM, yaitu [3]:

1. Mendapatkan pelanggan baru (*acquire*).. Pelanggan baru didapatkan dengan memberikan kemudahan pengaksesan informasi, inovasi baru, dan pelayanan yang menarik.
2. Meningkatkan hubungan dengan pelanggan yang telah ada (*enhance*).. Perusahaan berusaha menjalin hubungan dengan pelanggan melalui pemberian pelayanan yang baik terhadap pelanggannya (*customer service*). Penerapan *cross selling* atau *up selling* pada tahap kedua dapat meningkatkan pendapatan perusahaan dan mengurangi biaya untuk memperoleh pelanggan (*reduce cost*).
3. Mempertahankan pelanggan (*retain*).. Tahap ini merupakan usaha mendapatkan loyalitas pelanggan dengan mendengarkan pelanggan dan berusaha memenuhi keinginan pelanggan.

### 2.2.3.5. Keuntungan *Customer Relationship Management* (CRM )

CRM membantu perusahaan untuk mengembangkan produk baru berdasarkan engetahuan yang lengkap tentang keinginan pelanggan, dinamika pasar dan pesaing dengan cara [3]:

1. Menjaga pelanggan yang sudah ada
2. Menarik pelanggan baru
3. *Cross Selling*: menjual produk lain yang mungkin dibutuhkan pelanggan berdasarkan pembeliannya

4. Upgrading: menawarkan status pelanggan yang lebih tinggi (gold card vs. silver card)
5. Identifikasi kebiasaan pelanggan untuk menghindari penipuan
6. Mengurangi resiko operasional karena data pelanggan tersimpan dalam satu sistem
7. Respon yang lebih cepat ke pelanggan
8. Meningkatkan efisiensi karena otomasi proses
9. Meningkatkan kemampuan melihat dan mendapatkan peluang

### 2.2.4. *Enterprise Resource Planning* (ERP)

Ketika awal tahun 1998, ERP masih merupakan istilah yang sayup-sayup terdengar di beberapa perusahaan dan orang-orang yang berkutat di sektor manufaktur. Wujud ERP yang sering hadir secara fisik menjadi sebuah paket *software* membuat beberapa orang menafsirkan ERP secara sederhana, yaitu sebuah software. Tetapi, sebenarnya, benarkah ERP itu sebuah *software* dan jika bukan *software*, seperti apakah bentuk ERP tersebut dan mengapa banyak perusahaan mulai ramai-ramai ingin mengimplementasikannya?

Di pihak lain, banyak artikel dan berita yang memuat tentang aspek implementasi ERP.beberapa diantaranya memberikan persepsi positif atas keberhasilan ERP dalam mencapai tujuannya, yaitu meningkatkan efektifitas perusahaan.

ERP adalah sebuah software yang menggabungkan dan mengotomatisasi data dan beberapa *business process* dari suatu perusahaan atau organisasi. Menurut teori informasi organisasi, kinerja dari suatu sistem tergantung dari

bagaimana informasi itu diproses dan konteks informasi itu didalam organisasi. Dua hal mendasar dari teori ini adalah *interdepence* dan *differentiation* dari sub unit organisasi tersebut. Karena ERP mengabungkan data dan proses sekaligus,teori ini berpendapat bahwa ERP akan sukses jika ketergantungan antar subunit tinggi dan *differentiation* rendah [5].

**2.2.4.1. Definisi *Enterprise Resource Planning* (ERP)**

ERP singkatan dari tiga elemen kata, yaitu ***Enterprise***, (perusahaan/organisasi), ***Resource*** (sumber daya) dan ***Planning*** (perencanaan). Tiga kata ini mencerminkan sebuah konsep yang berujung kepada kata kerja, yaitu '***planning***', yang berarti bahwa ERP menekankan kepada aspek perencanaan. [8]

Beberapa defenisi tentang ERP dapat memberikan gambaran mengenai konsep-konsep dasar yang terkandung dalam ERP, yaitu:

Travis Anderegg mendefenisikan ERP sebagai :

*"ERP is a complete enterprise wide business software solution. The ERP sistem consist of software support modules, such as: marketing and sales, field service, product design and development, production and inventory control, procurement, distribution, industrial facilities management, process design and development, manufacturing, quality, human resources, finance and accounting, and information service"*[8]

Daniel O'Leary mendefenisikan sebagai :

*"ERP sistems are computer based sistems design to process an organization's transaction and facilitate integrated and real-time planning, production, and*

*customer response. In particular ERP sistems will be assumed to have certain characteristics"*

Situs Wikipedia mendeskripsikan ERP sebagai berikut

*"Enterprise Resource Planning sistems (ERPs) integrate (or attempt to integrate) all data and process of an organization into a single unified sistem. A typical ERP sistem will use multiple components of computer software and hardware to achieve the integration. A key ingredient of most ERP sistem is the use of a single,unified database to store data for the various sistem modules."* [5]

Dari defenisi dan deskripsi di atas dapat disimpulkan adanya kesamaan ide dan kata kunci utama pada ERP, yaitu adanya aspek perencanaan yang terintegrasi di suatu organisasi/perusahaan, bersifat lintas fungsional, terdiri atas berbagai fitur dengan tujuan agar dapat merencanakan dan mengelola sumber daya organisasi dengan lebih efisien dan dapat merespon kebutuhan pelanggan dengan lebih baik [5].

Konsep ERP dapat dijalankan dengan baik, jika didukung oleh seperangkat aplikasi dan infrastruktur komputer baik *software* dan *hardware* sehingga pengelolaan data dan informasi dapat dilakukan dengan mudah dan terintegrasi. Peranan perangkat teknologi dalam konsep ERP selain sebagai fasilitator juga karena perangkat teknologi tersebut dapat memeberikan nilai tambah berupa perampingan proses, penyederhanaan proses, integrasi, dan otomatis proses. Oleh karena itu, hampir tidak mungkin mewujudkan konsep ERP tanpa adanya dukungan sistem berbasis komputer, maka selanjutnya dalam buku ini yang dimaksud dengan sistem ERP adalah sekumpulan paket sistem informasi

yang dibangun dan diimplementasikan sebagai fasilitator terwujudnya konsep ERP di suatu organisasi [8].

Beberapa defenisi lain dicantumkan dalam beberapa buku menyertakan konsep-konsep dasar ERP, yaitu :

1. “ERP terdiri atas paket *software* komersial yang menjamin integrasi yang mulus atas semua aliran informasi di perusahaan, meliputi keuangan, akuntansi, sumber daya manusia, rantai pasok, dan informasi konsumen” (Davenport,1998).
2. “Sistem ERP adalah paket sistem informasi yang dapat dikonfigurasi, yang mengintegrasikan informasi dan proses yang berbasis informasi di dalam, dan melintas area fungsional dalam sebuah organisasi” (Kumar & Van Hillsgerberg, 2000)
3. “Satu basis data, satu aplikasi, dan satu kesatuan antarmuka di seluruh *enterprise*” (Tadjer, 1998).

Kata integrasi yang sering disebut-sebut dalam konsep ERP ini berhubungan dengan intepretasi sebagai berikut:

1. menghubungkan antara berbagai aliran proses bisnis
2. metode dan teknik berkomunikasi
3. keselarasan dan sinkronisasi operasi bisnis
4. koordinasi operasi bisnis.

Daniel O’Leary juga menambahkan beberapa karakteriatik tertentu yang idealnya ada pada sistem ERP, yaitu :

1. Paket perangkat lunak yang didesain untuk lingkungan pelanggan pengguna *server*, apakah itu secara tradisional atau berbasis jaringan.
2. Memadukan sebagian besar dari proses bisnis.
3. Memproses sebagian besar dari transaksi perusahaan.
4. Menggunakan *database* perusahaan secara tipikal menyimpan setiap data sekali saja.
5. Memungkinkan akses data secara *real time.*
6. Memungkinkan perpaduan proses transaksi dan kegiatan perencanaan.
7. Menunjang sistem multimata uang dan bahasa, yang sangat diperlukan oleh perusahaan multinasional.
8. Memungkinkan penyesuaian untuk kebutuhan khusus perusahaan tanpa melakukan pemograman kembali.

Kata *enterprise* sendiri cenderung dipertahankan karena memiliki makna yang tidak dapat diwakili oleh kata lain, misalnya perusahaan atau organisasi. Kata *enterprise* digunakan untuk menggambarkan situasi bisnis secara umum dalam satu entitas korporat, dalam berbagai ukuran, mulai dari bisnis ukuran kecil seperti kafe, hingga bisnis raksasa seperti perusahaan telekomunikasi. Secara konseptual, *enterprise* dapat digambarkan sebagai sebuah kelompok orang dengan tujuan tertentu, yang memiliki sumber daya untuk mencapai tujuan tersebut.

Sudut pandang *enterprise* ini berbeda dengan sudut pandang organisasi atau perusahaan yang bersifat tradisional. Pada sudut pandang tradisional, organisasi atau perusahaan dibagi-bagi menjadi unit-unit dengan fungsi-fungsi tertentu sehingga kita mengenal fungsi personalia, fungsi keuangan, fungsi pemasaran,

dan sebagainya. Setiap fungsi ini yang kemudian biasanya berwujud departemen, seolah-olah bekerja pada kotak masing-masing dan memiliki tujuan dan sasaran masing-masing, yang menurut sudut pandang mereka, sesuai dengan tujuan organisasi. Setiap fungsi terisolasi dan memiliki sistem dan koleksi data dan hasil analisi mereka sendiri. Akibatnya, informasi yang dihasilkan oleh masing-masing departemen, hanya dapat dibaca oleh *top* manajemen itu sendiri dan tidak dapat dimanfaatkan oleh departemen lain. Setiap departemen bergerak dengan arah dan sasaran masing-masing, karena mereka tidak mengetahui apa yang dikerjakan oleh departemen lainnya [5].

Dalam sudut pandang *enterprise*, keseluruhan organisasi dianggap sebagai sebuah sistem dan masing-masing departemen adalah subsistem[5]. Informasi mengenai semua aspek organisasi disimpan dan dikelola secara terpusat dan dapat diakses oleh departemen lain yang membutuhkan. Hasilnya adalah transparansi informasi sehingga masing-masing departemen mengetahui apa yang dikerjakan departemen lain, mengapa hal itu dikerjakan, dan apa yang harus dilakukan untuk mendukung pekerjaan tersebut sehingga dapat mencapai tujuan organisasi secara keseluruhan [5].

Kata '***resource***' secara singkat dapat diterjemahkan menjadi sumber daya. Dalam kaitannya dengan *enterprise*, *resource* dapat berupa aset perusahaan yang meliputi aset keuangan, sumber daya manusia, konsumen, *supplier*, *order*, teknologi, dan juga strategi. Istilah sumber daya ini melebar meliputi semua hal yang menjadi tanggung jawab dan tantangan manajemen untuk dikelola agar dapat menghasilkan keuntungan.

Jadi,ERP dapat dideskripsikan sebagai sebuah konsep untuk merencanakan dan mengelola sumber daya organisasi agar dapat dimanfaatkan secara optimal untuk menghasilkan nilai tambah bagi seluruh pihak yang berkepentingan (*stake holder*) atas organisasi tersebut [5].

#### 2.2.4.2. Manfaat Implementasi *Enterprise Resource Planning* (ERP)

Tentang manfaat sistem ERP bagi organisasi, terdapat sebuah persepsi umum yang mungkin belum tepat, yaitu bahwa implementasi sistem ERP akan meningkatkan fungsionalisasi organisasi dengan cepat. Tercapainya harapan yang tinggi berupa penghematan biaya dan peningkatan layanan, sangat bergantung pada seberapa jauh kita memilih sistem ERP yang sesuai dengan fungsionalitas organisasi dan seberapa optimal kita melakukan modifikasi dan konfigurasi ulang atas proses-proses yang ada pada sistem agar sesuai dengan kultur bisnis, strategi, dan struktur organisasi.

Beberapa manfaat yang dapat diperoleh dari penggunaan sistem ERP ini adalah [5]:

1. ERP menawarkan sistem terintegrasi di dalam perusahaan, sehingga proses dan pengambilan keputusan dapat dilakukan secara lebih efektif dan efisien.
2. ERP memungkinkan melakukan integrasi secara global. Halangan yang tadinya berupa perbedaan valuta mata uang, perbedaan bahasa, dan perbedaan budaya, dapat dijembatani secara otomatis, sehingga data dapat diintegrasikan.

3. ERP tidak hanya memadukan data dan orang, tetapi juga menghilangkan kebutuhan pemutakhiran dan koreksi data pada banyak sistem komputer yang terpisah.
4. ERP memungkinkan manajemen mengelola operasi, tidak hanya sekadar memonitor saja. Dengan ERP, manajemen tidak hanya mampu menjawab pertanyaan ‘Bagaimana keadaan kita?’; tetapi juga mampu menjawab pertanyaan ‘Apa yang kita kerjakan untuk menjadi lebih baik?’
5. ERP membantu melancarkan pelaksanaan manajemen *supply chain* dengan kemampuan memadukannya.

Secara keseluruhan sistem ERP diharapkan dapat meningkatkan tulang punggung fungsionalitas, baik pada bagian operasional maupun antarmuka dengan konsumen secara simultan.

Manfaat di atas dapat diperoleh secara spesifik melalui cara-cara atau fasilitas tertentu yang ada pada ERP. Kaitan antara manfaat dengan cara mendapatkannya dapat dilihat pada table 1.2 berikut:

Tabel 2.1. Manfaat ERP dan Cara Mendapatkannya

| **Manfaat** | **Cara Mendapatkan** |
|---|---|
| Akses informasi yang andal | DBMS yang fleksibel, data yang konsisten dan akurat, sistem pelaporan yang lebih baik |
| Menghindari duplikasi data dan operasi | Modul-modul yang mengakses data dari satu *database* terpusat, sehingga menghindari proses pemasukan dan modifikasi data dari berbagai titik |
| Mempercepat waktu pemrosesan data | Meminimasi waktu pengambilan data dari pembuatan laporan |
| Mengurangi biaya | Menghemat waktu, meningkatkan kontrol dengan melakukan analisis menyeluruh terhadap keputusan organisasional |
| Kemudahan adaptasi | Perubahan pada proses bisnis dapat diadaptasi dengan mudah |

Tabel 2.1. Manfaat ERP dan Cara Mendapatkannya (Lanjutan)

| | |
|---|---|
| Meningkatkan skalabilitas | Struktur sistem yang bersifat modular dan mudah dikostumisasi |
| Kemudahan pemeliharaan | Dukungan purnajual sistem yang berjangka panjang |
| Pengembangan global | Eksistensi modul hingga meliputi SCM (*Supply Chain Management*) dan CRM (*Customer Relationship Management*) |
| e-commerce | Bisnis *internet*, kultur kolaboratif |

Untuk mencapai manfaat tersebut, perusahaan harus melakukan serangkaian proses dan usaha, beberapa diantaranya dapat mendatangkan masalah, sehingga sering dianggap sebagai salah satu resiko yang harus ditanggung ketika implementasi ERP. Beberapa masalah yang berpeluang dihadapi organisasi dapat disajikan secara ringkas pada table 1.3.

Tabel 2.2. Kendala dalam Implementasi ERP dan Cara Mengatasinya

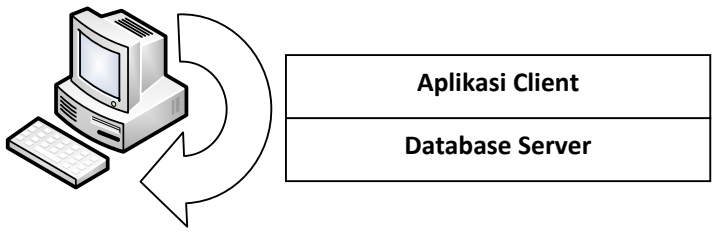
| **Kendala** | **Cara mengatasi** |
|---|---|
| Memakan waktu | Meminimasikan isu sensitife, politik internal, dan ciptakan konsensus umum |
| Mahal | Memilih paket dan strategi ERP yang sesuai dengan kemampuan keuangan perusahaan |
| Kesesuaian modul | Arsitektur dan komponen dari sistem yang dipilih harus sesuai dengan proses bisnis, kultur dan sasaran strategi organisasi |
| Kebergantungan pada vendor | Pertimbangkan pilihan *single vs multivendor*, pertimbangkan kriteria pemilihan kombinasi terbaik dan komitmen dukungan dalam jangka waktu yang cukup panjang. |
| Fitur dan kompleksitas | Pilih modul dan fitur yang benar-benar diperlukan oleh organisasi |
| Skalabilitas dan kompabilitas global | Perhatikan investasi *vendor* di bidang riset dan pengembangan, komitmen jangka panjang atas produk dan layanan, dan pertimbangkan sistem yang dapat berjalan di *internet* |
| Pengembangan kemampuan | Pertimbangkan aplikasi perantara (*middle ware*) dan pengembangan modul, misalnya SCM/CRM |

### 2.2.5. Arsitektur Model Aplikasi

Sejalan dengan perkembangan teknologi infotmasi dan telekomunikasi aristektur aplikasi mengalami banyak perubahan. Perubahan arsitektur ini ditujukan untuk memberikan kemudahan-kemudahan bagi pihak pihak yang terhubung dengan penggunaan aplikasi tersebut. Beberapa istilah arsitektur yang dikenal secara umum seperti *standalone, clientserver, two-tiers application, tree-tiers application, N-tiers application* [6].

#### 2.2.5.1. Arsitektur *Single-Tiers Application*

Aplikasi dengan arsitektur s*ingle-tiers* menggabungkan basis data (*database*) dengan aplikasi *client* pada sebuah computer untuk dijalankan. rsitektur ini adalah arsitektur yang paling sederhana dan dirancang untuk single-*user*. *Database* yang digunakan adalah *database* local Gambar model arsitektur *two-tiers application* dapat dilihat pada gambar 2.2 [6].

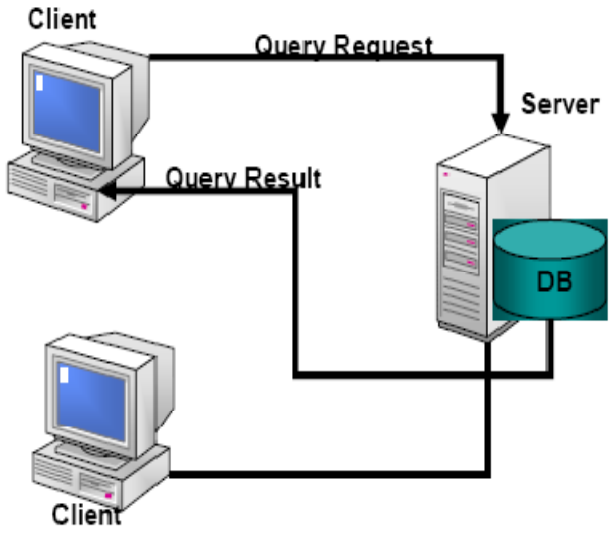



Gambar 2.3. Model Arsitektur Single-*Tiers Application* [6]

#### 2.2.5.2. Arsitektur *Two-Tiers Application*

*Two-tiers* menunjukan dua tingkat/dua aplikasi yang saling berkomunikasi, satu dengan yang lainnya dapat saling memberikan data dan informasi. Tingkatan tersebut sering diistilahkan dengan *Client* dan *Server*. Oleh

sebab itu t*wo-tiers application* juga dikenal dengan *Client-Server Application.* Gambar model arsitektur *two-tiers application* dapat dilihat pada gambar 2.3 [6].




Gambar 2.4. Model Arsitektur T*wo-Tiers Application*[6]

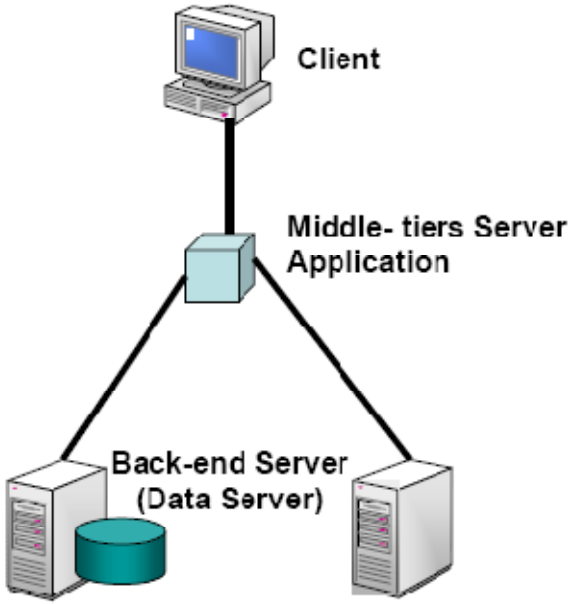
Konsep *two-tiers application* adalah sebagai berikut [6]:

a. *Client* melakukan sebuah permintaan ke *server*. Biasanya jaringan komputer telah terhubung dengan topologi jaringan tertentu.
b. Ketika *server* menerima permintaan layanan tersebut, *server* mengatur dan membawa permintaan tersebut. *Server* tersebut kemudian mengirimkan balik sebuah pesan respon ke *client* yang meminta.
c. Respon tersebut dapat berupa sekumpulan *record*, pesan kesalahan, atau sebuah nilai *boolean* yang mengindikasikan apakah permintaan telah dieksekusi dengan sukses.

**2.2.5.3. Arsitektur *Three-Tiers Application***

Aplikasi *three-tier* merupakan sebuah perbaikan dari *two-tier architecture.* Aliran informasi pada dasarnya tetap linear dimana sebuah permintaan (*request*) datang dari *client* ke *server*; *server* tersebut mengambil atau menyimpan data

dari/ke basisdata; basisdata tersebut mengembalikan informasi ke *server*; *server* tersebut mengembalikan informasi ke *client*. Gambar model arsitektur *three-tiers application* dapat dilihat pada gambar 2.4 [6].



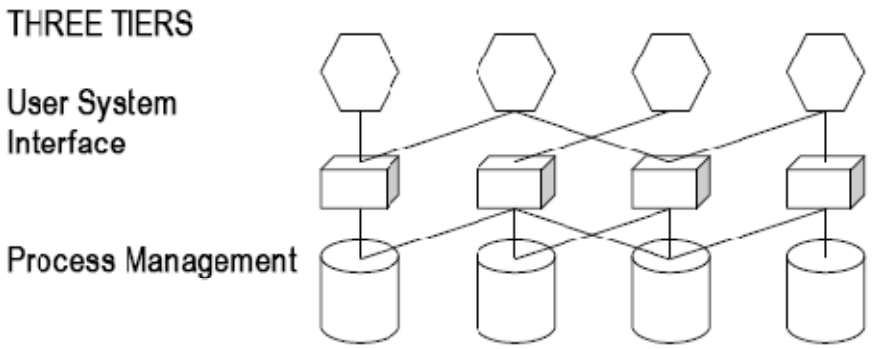

6]

Arsitektur *three-tier* terdapat tiga bagian penting yang paling mendasar, seperti terlihat pada gambar 2.4 dan gambar 2.5, terdapat lapisan-lapisan yang menunjukkan bahwa arsitektur tersebut *tree-tier*, yaitu :

1. *User tier* (*presentation layer, user services*).
2. *Business tier* (*business logic layer, business services*).
3. *Data tier* (*data layer, data services*).

Perbedaan yang paling menonjol dari *two-tier* dengan *three-tier* yaitu adanya *middletier* pada arsitektur *three-tier* yaitu *business logic layer*. *Arsitektur tree-tier* muncul karena adanya keterbatasan pada arsitektur *two-tier*, dimana jika ada perubahan fungsi suatu komponen pada satu sisi mempengaruhi kedua sisi

yaitu *client* dan *server*. Hal ini tidak terjadi pada sistem *tree-tier*. Arsitektur *tree-tier* digunakan ketika diperlukan suatu rancangan *client-server* yang efektif, dimana meningkatkan kinerja, fleksibilitas, kemudahan perawatan, kemampuan untuk dapat digunakan ulang, dan skalabilitas [6].




Gambar 2.6. Model Arsitektur Three-*Tiers Application* [6]

Arsitektur *three-tier* memiliki komponen terdepan yang bertanggung jawab dalam penyediaan antarmuka ke pengguna, dan komponen paling belakang adalah *server* basisdata itu sendiri. Komponen *middletier* memperbolehkan pengguna untuk berbagi dan mengontrol *business logic layer* dengan mengisolasi komponen-komponen yang ada didalam *middletier* tersebut dari aplikasi yang sesungguhnya dihadapi pengguna. Dengan *middletier* tersebut maka *business logic* dan *rules* dieksekusi dan dapat mencakup hingga banyak *user*, lebih banyak daripada menggunakan *two-tier*, dengan menyediakan fungsi-fungsi untuk antrian, eksekusi aplikasi, maupun akses basisdata. Sistem pada *client* berinteraksi dengan *middletier* melalui sebuah protokol yang sudah menjadi standar seperti misalnya HTTP, RPC, ataupun *middleware* seperti CORBA, DCOM, atau JavaBean. *Middletier* tersebut berinteraksi dengan *database server* melalui protokol

*database* standar seperti SQL, ODBC, dan JDBC. *Middletier* merupakan lapisan yang paling penting dan menentukan adanya komunikasi antara *client* dengan *server*. Pada *middletier* ini segala fungsi-fungsi yang dibutuhkan untuk komunikasi antara *client* dan *server* muncul. Adanya istilah *fat server* maupun *fat client* merupakan pengertian bahwa fungsi-fungsi tersebut cenderung lebih banyak menempel pada *client* atau pada *serve* [6].

Keuntungan dari arsitektur *three-tier* yaitu :

1. Perubahan pada antarmuka pengguna atau *application logic* tidak saling mempengaruhi satu sama lain, membuat aplikasi tersebut mudah berevolusi untuk memenuhi kebutuhan baru.
2. *Network bottleneck* diminimalkan *karena application layer* tidak mentransmisikan data ekstra ke *client*, hanya data yang diperlukan untuk menangani suatu task.
3. Ketika *business logic* perlu diubah, hanya *server* yang perlu diubah. Dalam *two-tier architecture* setiap *client* perlu dimodifikasi.

**2.2.5.4. Arsitektur N-Tier Application**

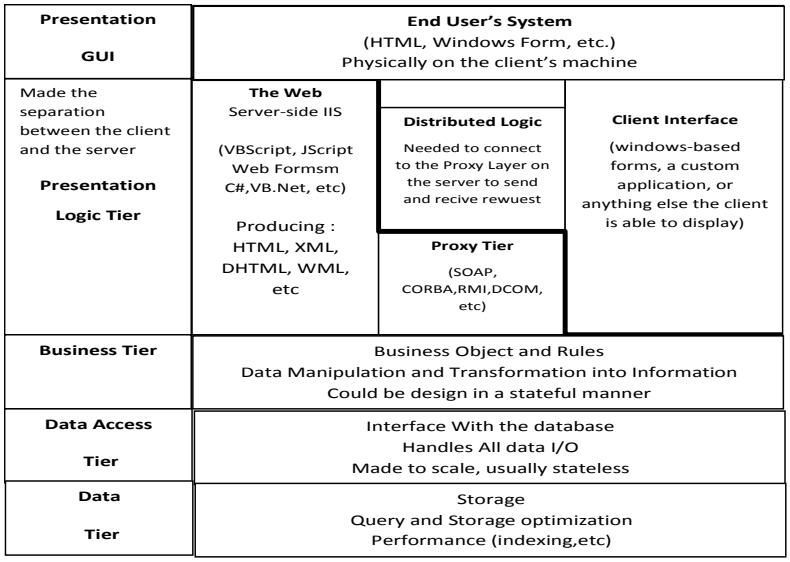
Peningkatan kompleksitas bisnis membuat *business services tier* membesar dan sulit dimanajemen. Ini menuntut pembagian *business services model* menjadi unit-unit yang lebih kecil yang *manageable*, *menghasilkan n-tier architecture* atau *multi-tier application*. Istilah *n-tier* berasal dari kenyataan bahwa *application logic tier* seringkali dibagi lagi menjadi beberapa *logical tier* yang masing-masing memiliki tugas khusus. Pembagian *appliaction tier* ini sangat

tergantung pada permasalahan yang dihadapi. Tidak ada pembagian fungsi yang pasti untuk setiap *tier* seperti *two-tier* atau *three-tier*, tetapi secara umum pembagiannya didasarkan pada *three-tier* [6].

Berikut ini merupakan contoh pembagian tier pada arsitektur n-tier. Arsitektur ini terdiri dari empat layer yaitu:

1. *Presentation (user services)*
2. *UI-centric business objects*
3. *Data-centric business objects*
4. *Data services*

Kelihatannya ini hanya sebuah perubahan kecil, tetapi sesungguhnya perubahannya sangat signifikan. Dengan memisahkan UI-centric processing dari data-centric processing akan memberikan fleksibilitas yang lebih tinggi. *UI-centric processing components* diletakkan di mesin dekat *client*, sementara data-centric processing components pada mesin lain dekat basisdata, sehingga dapat diperoleh keuntungan peningkatan kinerja yang signifikan. Pembagian *tier* tidak selalu seperti yang diungkapkan diatas, tetapi sangat tergantung kebutuhan sistem. *User tier* dapat dibagi lagi untuk melakukan standarisasi antarmuka aplikasi ke pengguna, seperti yang terjadi dalam aplikasi berbasis web. Demikian juga data tier dapat dibagi lagi jika memang dibutuhkan untuk pengaksesan basisdata yang besar dan mungkin tersebar di jaringan dengan berbagai *database management sistem* yang berbeda-beda pula. Gambar 2.4 dan 2.5 menunjukkan sebuah contoh pembagian tier di *user tier* dan *data tier*.




Gambar 2.7. Model Arsitektur N-*Tiers Application* [6]

### 2.2.6. Jaringan Komputer

#### 2.2.6.1. Definisi Jaringan Komputer

Dengan berkembangnya teknologi komputer dan komunikasi suatu model komputer tunggal yang melayani seluruh tugas-tugas komputasi suatu organisasi kini telah diganti dengan sekumpulan komputer yang terpisah-pisah akan tetapi saling berhubungan dalam melaksanakan tugasnya, sistem seperti ini disebut jaringan komputer (*computer network*) [10].

Jaringan komputer dapat diartikan suatu himpunan *interkoneksi* sejumlah komputer yang *autonomous.* Dua buah komputer dikatakan terinterkoneksi bila keduanya dapat saling bertukar informasui. Betuk koneksinya tidak harus melalui kawat tembaga saja melainkan dapat emnggunakan serat optik, gelomabng mikro, atau satelit komunikasi.

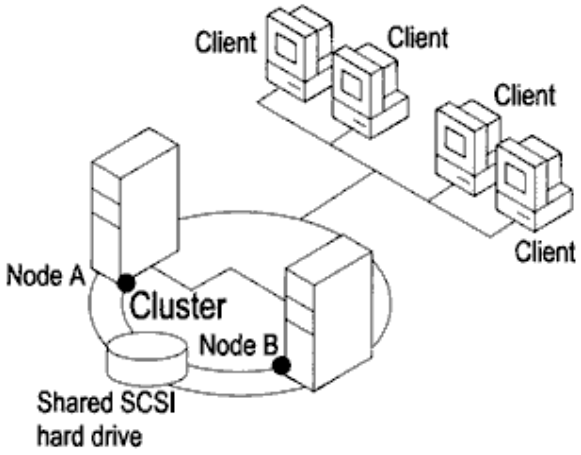
Untuk memahami istilah jaringan komputer sering kali kita dibingungkan dengan sistem terdistribusi (*distributed sistem*) [10]. Kunci perbedaannya adalah bahwa sebuah sistem terdistribusi,keberadaan sejumlah komputer autonomous bersifat transparan bagi pemakainya. Seseorang dapat memberi perintah untuk mengeksekusi suatu program, dan kemudian program itupun akan berjalan dan tugas untuk memilih prosesor, menemukan dan mengirimkan file ke suatu prosesor dan menyimpan hasilnya di tempat yang tepat mertupakan tugas sistem operasi. Dengan kata lain, pengguna sistem terditribusi tidak akan menyadari terdapatnya banyak prosesor (multiprosesor), alokasi tugas ke prosesor-prosesor, alokasi file ke disk, pemindahan file yang dfisimpan dan yang diperlukan, serta fungsi-fungsi lainnya dari sitem harus bersifat otomatis.

Pada suatu jaringan komputer, pengguna harus secara eksplisit log ke sebuah mesin, secara eksplisit menyampaikan tugasnya dari jauh, secara eksplisity memindahkan file-file dan menangani sendiri secara umum selusurh manajemen jaringan. Pada sistem terdistribusi, tidak ada yang perlu dilakukan secara eksplisit, sermunya sudah dilakukan secara otomatis oleh sistem tanpa sepengetahuan pemakai.

Dengan demikian sebuah sistem terdistribusi adalah suatu sistem perangkat lunak yang dibuat pada bagian sebuah jaringan komputer. Perangkat lunaklah yang menentukan tingkat keterpaduan dan transparansi jarimngan yang bersangkutan. Karena itu perbedaan jaringan dengan sistem terdistribusi lebih terletak pada perangkat lunaknya (khususnya sistem operasi), bukan pada perangkat kerasnya [10].

### 2.2.6.2. Sejarah Jaringan Komputer

Kebutuhan akan komunikasi untuk komputer pada tahun 1940-an sampai 1950-an hanyalah bersifat dasar dan minimal. Prosesor berkomunikasi dengan periferalnya melalui peralatan input/output (*I/O*) pada jarak yang pendek dan bekerja pada kecepatan yang sangat rendah. Tahun 1960-an lahir konsep *timesharing*, dimana pengguna dihubungkan ke komputer melalui suatu *dumb terminal*. Konsep ini dapat dilihat pada gambar 2.7 [10].




Gambar 2.8. Time Shared Computer Sistem [10]

Pada tahun 1970-an, teknologi IC(*Integrated Circuit*) dan mikroprosesor mulai berkembang sehingga memungkinkan muncunya komputer pribasi yang dapat dipasang di rumah-rumah. Perkembangan teknologi ini secara drastic mengubah cara pandang orang tentang komputer. Munculnya teknologi jaringan local (*Local Area Network-LAN*) dalam tahun 1980-an melengkapi komputer dengan kemampuan berkomunikasi dengan komputer lainnya. Kondisi ini menyebabkan terjadinya migrasi dari konsep pemrosesan secara tersentralisasi

(*centralized computing*) menjadi konsep secara terdistribusi (*distributed computing*).

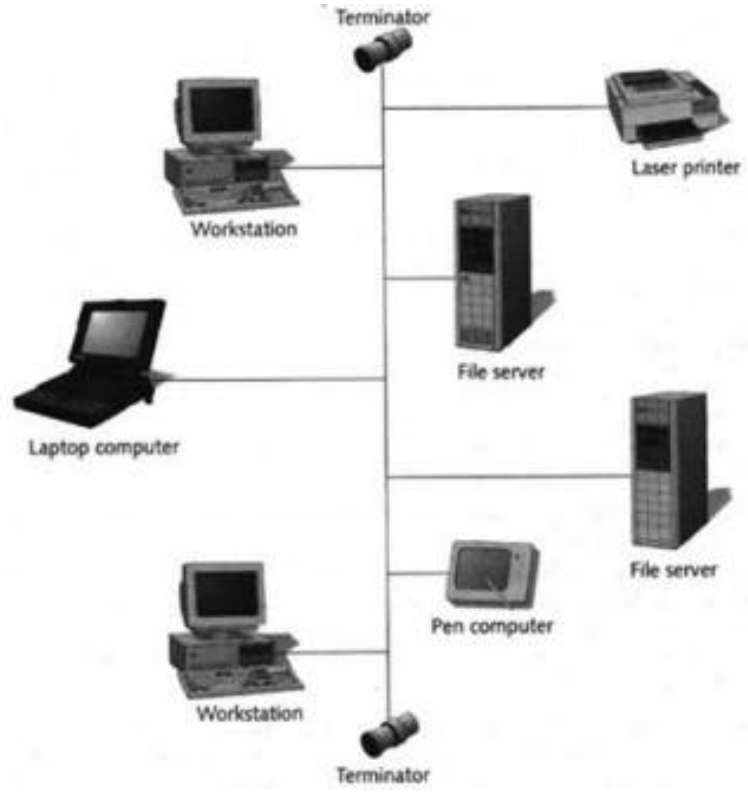
Pada *centralized computing*, sistem atau prosesor utama diletakan secara terpusat. Semua komputer yang letaknya berjauhan dihubungkan menggunakan link secara langsung ke prosesor utama tersebut. Semua informasi (*database*) terletak di sistem pusat.

Pada kasus *distributed computing*, prosesor atau komter utama didistribusikan pada lokasi-lokasi yang berbeda. Masing-masing komputer tersebut mempunyai sebagian atau seluruh duplikasi dari data yang ada di komputer pusat. Pengguna mengakses informasi dari prosesor yang terdekat yang secara periodik mengupdate data yang ada di dalam database-nya.

Trend pada masa dating adalah migrasi menuju lingkungan *distributed computing* karena perkembangan prosesor dan koneksi *switching* serta intelegensia terletak pada level CPE [10]

### 2.2.6.3. Topologi Bus

Topologi Bus menyediakan 1 saluran untuk komunikasi semua perangkat sehinga setiap perangkat harus bergantian menggunakan seluran tersebut. Oleh karenanya hanya ada 2 perangkat yang saling berkomunikasi dalam suatu saat. Untuk mengefisiensikan penggunaan jaringan, digunakan metode CSMA/CD (*Carrier Sense Multiple Access / Collision Detected*) yang dapat mengurangi terjadinya masa tenggang (saluran kosong) dengan mendeteksi tabrakan informasi. Arsitektur topologi bus dapat dilihat pada gambar 2.8 [10].




Gambar 2.9 Topologi Bus [10]

### 2.2.7. Object Oriented Programming (OOP)

OOP merupakan terobosoan baru dalam pemograman setelah pemograman procedural, pemograman modular dan pemograman abstarkasi data. Ada beberapa konsep utama OOP antara lain [4]:

Pemrograman orientasi-objek menekankan konsep berikut:

1. *Class* — kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh *'class of dog'* adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah class adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi object. *Sebuah*

*class secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada*, dan kode yang terdapat dalam sebuah class sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya [4].

2. Objek - membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer; objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek [4].
3. Abstraksi - Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan [4].
4. Enkapsulasi - Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi ijin untuk mengakses keadaannya. Setiap objek mengakses interface yang menyebutkan bagaimana objek lainnya dapat

berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut [4].

5. Polimorfisme melalui pengiriman pesan. Tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan; metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesa tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tungal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama [4].
6. Inheritas - Mengatur polimorfisme dan enskapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada - objek-objek ini dapat membagi (dan memperluas) perilaku mereka tanpa haru mengimplementasi ulang perilaku tersebut (bahasa berbasis-objek tidak selalu memiliki inheritas [4].
7. Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah

tersebut(terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bag administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bag administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas adminiistrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek -objek yang ada karena setiap objek memiliki job descriptionnya sendiri.

### 2.2.8. Desain dan Pemodelan Berorientasi Objek Menggunakan UML

Unified Modeling Language (UML) adalah Keluarga notasi grafis yang didukung oleh model-model tunggal, yang membantu pendeskripsian dan desain perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek (OO). Untuk memahami konsep UML, perhatikan tabel 3.3 berikut ini [2] :

Tabel 2.3 Konsep UML

| Major Area | View | Diagrams | Main Concepts |
|---|---|---|---|
| Structural | static view | class diagram | Class, association, generalization, dependency, realization, antarmuka |
| | use case view | use case diagram | Use case, actor, association, extended, include, use case general-ization |

Tabel 2.3 Konsep UML (lanjutan)

| | | | |
|---|---|---|---|
| | Implementation view, deployment view | Component diagram | Component, antarmuka, dependency, realization |
| | | Deployment diagram | Mode, component, dependency,location |
| Dynamic | State machine view | State chart diagram | State event, transition, action |
| | Activity view | Activity diagram | State, activity, complemention, transition, fork, join |
| | Interaction view | Sequence diagram | Interaction, object, message, activation |
| | | Collaboration diagram | Collaboration, interaction, collaboration role, message |
| Model management | Model managent view | Class diagram | Package, subsistem, model |
| Extensibility | All | All | Constraint, stereotype, tagget values |

**2.2.8.1. Diagram UML**

*Unified Modeling Language* (UML) merupakan standar yang relativ terbuka yang dikendalikan oleh OMG (*Object Menegement Group*), sebuah konsorium terbuka yang terdiri dari banyak perusahaan. OMG diberntuk untuk membuat standar-standar yang mendukung interoperabilitas sistem berorientasi objek. UML lahir dari penggabungan bahasa pemodelan grafis pada akhir 1980-an dan awal 1990-an [22].

UML terdiri dari 13 jenis diagram resmi dan menklasifikasikan mereka, meskipun jenis-jenis diagram ini merupakan cara orang-orang memperlakukan UML, para perancang UML tidak memandang diagram sebagai bagian yang sentral. Dan hasilnya, jenis-jenis diagram bukanlah hal yang mutlak. Karena yang

dipergunakan dalam pembangunan aplikasi sistem *enterprise resource planning* adalah lima buah diagram maka disini akan di bahas empat buah diagram yaitu *Use Case Diagram, Class Diagram, Sequence Diagram dan Activity Digaram, Deployment Diagram* [22].

**2.2.8.2. Use Case Diagram**

Use case diagram menggambarkan sebuah fungsi yang dibutuhkan oleh sebuah sistem. Dalam hal ini ada kondisi yang agak beda, yaitu disini tingkah laku dan domain sistem. Sebuah use case merepresentasikan sebuah interaksi antara pengguna dengan sebuah sistem. Use case merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, membuat sebuah daftar aktifitas, dan sebagainya [22].

**2.2.8.3. Class Diagram**

*Class* merupakan inti dari pengembangan dan desain berorientasi objek, karena nantinya class ini akan menghasilkan sebuah objek. Class menggambarkan keadaan (atribut/property) suatu sistem dan metoda/fungsi yang ada pada sistem tersebut [22].

*Class diagram* menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga domain yaitu : Nama (dan stereotype), Atribut dan Metode.

Atribut dan metode dapat memiliki salah satu sifat berikut :

1. Private, tidak dapat dipanggil diluar class yang bersangkutan

2. Protected, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya.
3. Public, dapat dipanggil oleh siapa saja.

#### 2.2.8.4. *Sequence Diagram*

*Sequnce diagram*, secara khusus, menjabarkan behavior sebuah scenario tunggal. Diagram tersebut menunjukan sejumlah objek contoh dan pesan-pesan yang melewati objek-objek ini di dalam use case [22].

#### 2.2.8.5. *Activity Diagram*

*Activity Diagram* adalah teknik untuk menggambarkan loguka procedural, proses bisnis, dan alur kerja. Dalam beberapa masalah, diagram ini memainkan peran mirip sebuah diagram alir, tetapi perbedaan prinsip antara diagram ini dan notasi diagram alir adalah diagram ini mendukung behavioral parallel.

Activity Diagram memungkinkan siapa pun melakukan proses untuk memilih urutan dalam melakukannya, atau menyebutkan aturan-aturan yang harus diikuti [22].

#### 2.2.8.6. *Deployment Diagram*

*Deployment diagram* menunjukkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware* yang digunakan untuk mengimplementasikan sebuah sistem dan keterhubungan antara komponen-komponen *hardware* tersebut. *Deployment diagram* dapat digunakan pada bagian-bagian awal proses perancangan sistem untuk mendokumentasikan arsitektur fisik sebuah sistem [22].

### 2.2.9. Basis Data

Basis data adalah kumpulan data yang (arsip) yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (*redudansi*) yang tidak perlu, untuk memenuhi berbagai kebutuhan. Atau bisa diartikan sebagai kumpulan *file*/tabel/arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronis.

#### 2.2.9.1. Basis Data Relasional

Basis data relasional adalah basis data yang mengorganisasikan data dalam bentuk table. Tabel dibentuk dengan mengelompokan data yang mempunyai subjek yang sama.

#### 2.2.9.2. Tujuan Basis Data

Basis Data pada prinsipnya ditujukan untuk pengaturan data agar terdapat kemudahan dalam pengambilan kembali data tersebut. Berikut ini terdapat beberapa tujuan dari Basis Data diantaranya yaitu :

1. Kecepatan dan kemudahan (*Speed)*
2. Efisiensi ruang penyimpanan (*Space)*
3. Keakuratan (*Accuracy)*
4. Ketersediaan (*Avaibility)*
5. Kelengkapan (*Completeness)*
6. Keamanan (*Security*)
7. Kebersamaan (*Sharability*)

### 2.2.9.3. Pengguna Basis Data

Pengguna basis data yang dibedakan berdasarkan cara mereka berinteraksi terhadap sistem diantaranya yaitu :

a. Programmer Aplikasi (*Application Programmer*)

   Pemakai yang berinteraksi dengan basis data dengan menggunakan *Data Manipulation Language* (DML) untuk membuat aplikasi dengan menggunakan bahasa pemrograman.

b. *User* Khusus (*Specialized User*)

   Pemakai yang membuat program aplikasi basis data untuk keperluan tertentu atau khusus.

c. *User* Mahir (*Casual User*)

   Pemakai yang berinteraksi dengan sistem tanpa membuat modul program.

d. *User* Umum (*Naïve User*)

   Pemakai berinteraksi dengan aplikasi basis data yang telah dibuat atau disediakan oleh sistem.

### 2.2.9.4. Perancangan Basis Data

Perancangan basis data dibutuhkan agar diperoleh suatu sistem yang sesuai dengan apa yang diinginkan, dimana dapat melalui tahapan berikut [12] :

a. Entity Relationship Diagram ( ERD )

   *Entity Relationship Diagram* atau bisa dikenal dengan diagram E-R secara grafis menggambarkan isi sebuah database. Diagram ini memiliki dua komponen utama yaitu *entity* dan relasi. Untuk melambangkan fungsi

diatas maka digunakan simbol - simbol yang bisa dilihat pada daftar simbol.

Elemen-elemen *Entity Relationship Diagram* adalah sebagai berikut [14]:

1. *Entity*

   Pada *E-R* diagram, *entity* digambarkan dengan sebuah bentuk persegi panjang. *Entity* adalah sesuatu apa saja yang ada didalam sistem, nyata maupun abstrak dimana data tersimpan. Entitas diberi nama dengan kata benda dan dapat dikelompokan dalam empat jenis nama, yaitu: orang, benda, lokasi kejadian (terdapat unsur waktu didalamnya).

2. *Relationship*

   Pada *E-R diagram*, *Relationship* dapat digambarkan dengan sebuah bentuk belah ketupat. *Relationship* adalah hubungan alamiah yang terjadi antara entitas. Pada umumnya penghubung (*Relationship*) diberi nama dengan kata kerja dasar, sehingga memudahkan untuk melakukan pembacaan relasinya (bisa dengan kalimat aktif atau dengan kalimat pasif). Penggambaran hubungan yang terjadi adalah sebuah bentuk belah ketupat dihubungkan dengan dua bentuk empat persegi panjang.

3. *Relationship Degree*

   *Relationship degree* atau derajat relasi adalah jumlah entitas yang berpartisipasi dalam suatu *relationship*.

4. Atribut

   Secara umum atribut adalah sifat atau karakteristik dari tiap entitas maupun tiap *relationship*. Maksudnya, atribut adalah sesuatu yang menjelaskan apa sebenarnya yang dimaksud entitas maupun *relationship,* sehingga sering dikatakan bahwa atribut adalah elemen dari setiap entitas dan *relationship*.

5. Kardinalitas

   Kardinalitas relasi menunjukan jumlah maksimum tupel yang dapat berelasi dengan entitas pada entitas yang lain. Dari sejumlah kemungkinan banyaknya hubungan antar entitas, kardinalitas relasi merujuk kepada hubungan maksimum yang terjadi dari entitas yang satu ke entitas yang lain dan begitu juga sebaliknya. Terdapat tiga macam kardinalitas relasi, yaitu :

   *a. One to One*

      Tingkat hubungan satu ke satu, dinyatakan dengan satu kejadian pada entitas pertama, hanya mempunyai satu hubungan dengan satu kejadian pada entitas yang kedua dan sebaliknya.

   *b. One to Many atau Many to One*

      Tingkat hubungan satu ke banyak adalah sama dengan banyak ke satu. Tergantung dari arah mana hubungan tersebut dilihat. Untuk satu kejadian pada entitas yang pertama dapat mempunyai banyak hubungan dengan kejadian pada entitas yang kedua . Sebaliknya

satu kejadian pada entitas yang kedua hanya dapat mempunyai satu hubungan dengan satu kejadian pada entitas yang pertama.

c. *Many to Many*

Tingkat hubungan banyak ke banyak terjadi jika tiap kejadian pada sebuah entitas akan mempunyai banyak hubungan dengan kejadian pada entitas lainnya, baik dilihat dari sisi entitas yang pertama maupun dilihat dari sisi yang kedua.

#### 2.2.9.5. Tabel Relasi

Relasi tabel menunjukkan adanya hubungan diantara sejumlah entitas yang berasal dari himpunan entitas yang berbeda [12]. Dari adanya relasi tabel tersebut terdapat kardinalitas relasi yang menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan enttas yang lain. Terdapat empat kemungkinan kombinasi kardinalitas yaitu : satu ke satu (*One To One)*, satu ke banyak *(One To Many)*,banyak ke satu (*Many To One) dan* banyak ke banyak (*Many To Many*).

### 2.2.10. *Tools* Pengembangan

*Tools* pengembangan adalah perangkat lunak yang dipakai dalam pengembangan sistem.

#### 2.2.10.1. *Visual Studio 2005*

Visual Studio 2005 merupakan IDE (*Integrated Development Environment*) yang sangat mendukung pemograman berbasis objek sampai pada aplikasi jaringan serta sistem terdistribusi, serta dalam penggunaan untuk *event*

*driver programming* pun sangat baik. Visual Studio 2005 dikembangkan oleh Microsoft yang merupakan sebuah perusahan pengembangan perangkat lunak. Visual Studio 2005 mendukung beberapa bahasa pemograman yang ada saat ini, seperti Visual Basic .Net, Visual C#, Visual C++, Visual F#, J#, Ruby dan lain sebagainya. Hadirnya Visual Studio 2005 dikarenakan kebutuhan perangkat lunak untuk bisnis sangat besar. Visual Studio 2005 merupakan alat pengembangan multibahasa yang komperhensif. Dengan adanya IDE ini maka pengembangan aplikasi bisa dipercepat dan memberikan unjuk kerja yang handal.

Visual Studio 2005 terintegrasi dengan .Net Framework 2.0 yang merupakan sebuah platform untuk membangun, menguji, menyebarkan aplikasi yang dikembangkan. .Net Framework 2.0 menyediakan lingkungan berbasiskan standar produksi yang tinggi untuk memadukan investasi yang ada dengan aplikasi dan service generasi yang akan dating. .Net Framework 2.0 terdiri dari Windows Forms untuk pengembangan aplikasi desktop, Web Form untuk pengembangan aplikasi web, dan Console untuk pengembangan aplikasi Console, yang terintegrasi dengan CLR sebagai media eksekusi aplikasi.

Sistem yang dikembangkan menggunakan bahasa pemograman C# yang merupakan bahasa yang sudah mendukung konsep OOP (*Object oriented programming*) secara penuh, dan merupakan bahasa yang handal dan mudah untuk dipelajari dan dikembangkan.

#### 2.2.10.2. *Embarcadero ER-Studio*

Merupakan perangkat lunak yang memudahkan pengembang perangkat lunak dalam merancang basis data. ER-Studio digunakan karena mempercepat

pengembangan aplikasi dengan kemampuan untuk melakukan forward-engineering terhadap perancangan database yang dibuat.

#### 2.2.10.3. *Sybase Power Designer*

Sybase Power Designer adalah kakas pemodelan visual untuk pengembangan sistem berorientasi objek yang sangat handal untuk digunakan sebagai bantuan bagi para pengembangn dalam melakukan analisis dan perancangan sistem sebelum pengembang menulis kode-kode dalam bahasa pemograman tertentu. Sybase Power Designer mendukung pemodelan bisnis yang membantu pengembang untuk membangun sistem secara komperhensif. Sybase Power Designer mendukung pengembangan sistem lewat perancangan diagaram UML yang dipakai dalam pengembangan sistem.

### 2.2.11. Teknologi Sistem Terdistribusi

#### 2.2.11. 1. CORBA (*Common Object Request Broker Architecture*)

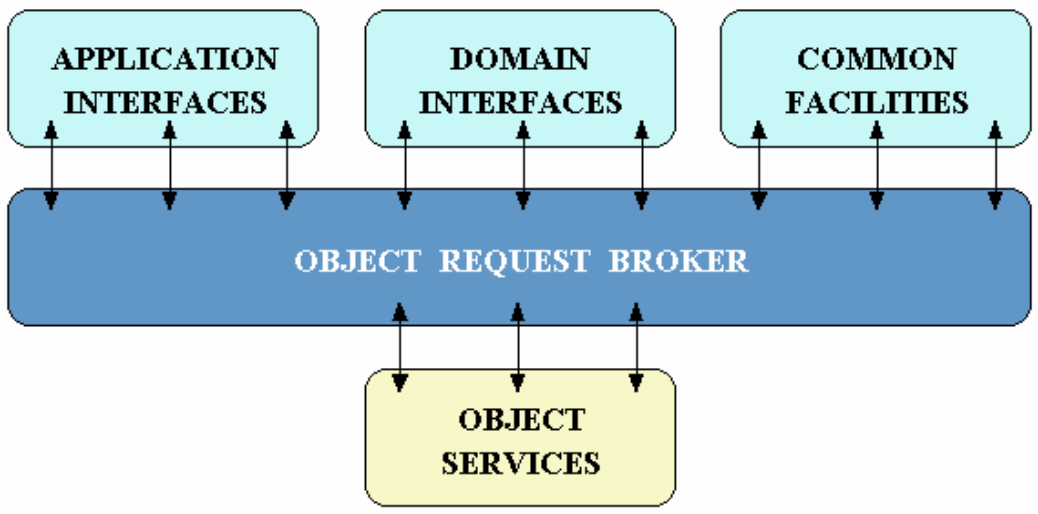
Interoperabilitas adalah kemampuan saling bekerjasama antar sistem komputer. Sebenarnya interoperabilitas bukanlah barang baru, karena protokol komunikasi datapun (TCP/IP misalnya) pada dasarnya diciptakan untuk mewujudkan interoperabilitas. Yang belum banyak dikenal adalah interoperabilitas pada level perangkat lunak aplikasi.

Konteks teknologi sistem komputer terdistribusi, meskipun komponen-komponen aplikasi dibuat dengan bahasa pemrograman yang berbeda, menggunakan *development tools* yang berbeda, dan beroperasi di lingkungan yang beragam, mereka tetap harus dapat saling bekerjasama.

Interoperabilitas perangkat lunak menuntut homogenitas pada suatu level tertentu. Untuk itu diperlukan semacam 'standarisasi'. Berawal dari keperluan ini lahirlah CORBA *(Common Object Request Broker Architecture)*. CORBA adalah hasil 'kesepakatan' antara sejumlah vendor dan pengembang perangkat lunak terkenal seperti IBM, Hewlett-Packard, dan DEC, yang tergabung dalam sebuah konsorsium bernama OMG (Object Management Group).

CORBA adalah sebuah arsitektur software yang berbasis pada teknologi berorientasi obyek atau *Object Oriented* (OO) dengan paradigma *client-server*. Dalam terminologi OO, sebuah obyek berkomunikasi dengan obyek lain dengan cara pengiriman pesan (message passing). Konteks komunikasi ini kemudian dipetakan ke dalam model *client-server*, satu obyek berperan sebagai *client* (si pengirim pesan) dan yang lain bertindak sebagai *server* (yang menerima pesan dan memroses pesan yang bersangkutan).

Keunikan dari CORBA adalah kemampuannya dalam menangani heterogenitas antara *client* dan *server* (dalam terminologi CORBA, obyek *server* dinamakan implementasi obyek *(object implementation)*. Keduanya dapat saja diimplementasikan dalam hardware, sistem operasi, bahasa pemrograman, dan di lokasi yang berbeda, tetapi tetap bisa saling berkomunikasi. Kuncinya ada pada sebuah lapisan software yang disebut dengan ORB *(Object Request Broker)*. Arsitektur CORBA dan ORBnya ditunjukkan pada gambar 2.x.




Gambar 2.10 Arsitektur CORBA *(Common Object Request Broker Architecture)*

**2.2.11. 2. RMI (*Remote Method Invocation*)**

*RMI* menyediakan sarana dimana *client* dan *server* dapat berkomunikasi dan saling bertukar infomasi. *RM I*memungkinkan pengembang perangkat lunak untuk merancang aplikasi terdistribusi dimana *methods* dari*remote object* dapat dipanggil dari *JVM (Java Virtual Machine)* lain, yang mungkin berjalan pada *host* yang berbeda. *Remote object* adalah obyek dalam *Java* yang dapat direferensikan secara *remote*. Pemrogram seakanakan memanggil *methods* lokal dari *file* kelas lokal, sedang dalam kenyataannya semua argumen dikirimkan ke *remote target* dan diinterpretasikan, kemudian hasilnya dikirimkan kembali ke pemanggil. Dalam *RMI*, *server* akan membuat *remote objects*, membuat referensi, dan menunggu *client* untuk memanggil *methods* dari *remote object* ini. Sedangkan *client* akan mendapatkan *remote reference* dari satu atau lebih *remote object* dan memanggil *methods* untuk *remote object* tersebut.

Sistem *RMI* dibangun atas tiga lapisan: lapis *stub/skeleton*, lapis *remote reference*, dan lapis *transport*. Tiap lapis dibangun dengan menggunakan *interface* khusus dan didefinisikan dengan protocol khusus. Hal ini menyebabkan tiap lapis bebas terhadap yang lain dan bisa digantikan dengan implementasi yang lain tanpa mempengaruhi lapis lain dalam sistem. Lapis *stub/skeleton* merupakan *interface* antara lapis aplikasi dan *JVM (Java Virtual Machine)*. *Stub* dan *skeleton* ini terbentuk secara otomatis dengan meng-gunakan *compiler* yang ada di *RMI* yaitu *rmic compiler* yang didefinisikan oleh *user* dalam *remote interface*. *Client* mereferensi satu *remote object* melalui *stub* ini. *Stub* akan berlaku sebagai *proxy* yang berlaku sebagai *place holder* bagi *server*. *Client-side stub* untuk suatu *remote object* bertanggung jawab pada berbagai macam tugas meliputi, inisiasi suatu *remote call, marshalling* (mengubah obyek menjadi bentuk yang *portable* bagi jaringan) argumen untuk dikirimkan, memberitahukan lapis remote reference bahwa suatu *call* harus dipanggil (*invoked*), *unmarshalling return value* (atau *exception*), dan memberitahukan lapis *remote reference* bahwa suatu panggilan telah selesai. *Server-side skeleton*, di lain pihak, bertanggung jawab dalam *unmarshalling* argumen yang masuk dari *client*, memanggil implementasi *remote object* yang sebenarnya, dan *marshalling return value* (atau *exception*) pada *stream* untuk dikirim kembali ke *client*.

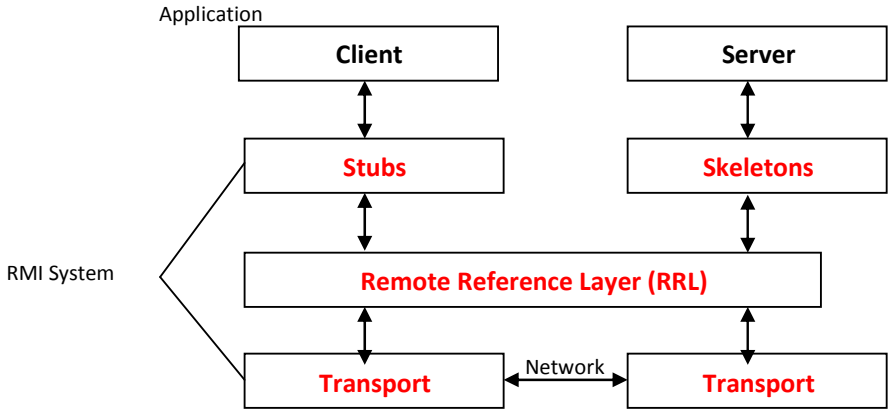
Lapis *remote reference* dan lapis *transport* merupakan lapis yang tidak terlihat oleh user (*invisible*). Lapis *reference* bertanggungjawab dalam melaksanakan satu protokol *remote reference* khusus. Lapis ini bertanggung-jawab untuk mengatur "*liveliness*" dari *remote object* dan mengatur komunikasi

antara *client/server* dengan *JVM*. Sedangkan lapis *transport* adalah lapis *TCP/IP-based* yang bertanggung jawab

dalam mengadakan hubungan antara *server* dan *client*.

Pada saat *client* memanggil suatu obyek implementasi dari suatu *server*, ketiga lapis dalam *RMI* akan memerankan perannya masing-masing. Lapis terpenting bagi pemrogram adalah lapis *stub/skeleton*. Pertama-tama suatu pemanggilan akan melewati lapis *stub/skeleton* ini. Fungsi dari lapis ini adalah mengirim data dari lapis *remote reference* melalui *marshal stream*. Di sinilah sebenarnya peranan dari *object serialization*, ini memungkinkan obyek *Java* untuk dikirimkan ke berbagai ruang alamat yang berbeda. Setelah melewati lapis *stub/skeleton*, data akan melewati lapis *remote reference* yang membawa semantik dari pemanggilan dan mengirimkan data ke lapis *transport* menggunakan *connection-oriented stream* seperti *TCP*. Ini berarti

bahwa lapis *remote reference* adalah menen-tukan sifat dari obyek dan juga menentukan apakah obyek berada pada mesin lokal atau mesin *remote* melalui jaringan. Lapis *remote reference* juga akan menentukan apakah obyek bisa diinstantiasi dan dimulai secara otomatis, atau harus dideklarasikan dan diinisialisassi terlebih dahulu. Dan terakhir, data akan sampai pada lapis *transport* yang bertanggung jawab dalam menyelenggarakan hubungan dan mengatur hubungan tersebut. Arsitektur RMI dapat dilihat pada gambar 2.x.




Gambar 2.11 Arsitektur RMI (*Remote Method Invocation*)

### 2.2.11. 3. ESB (Enterprise Service Bus)

Dalam dunia komputer, sebuah *enterprise service bus* (ESB) menunjuk pada sebuah arsitektur perangkat lunak yang dibangun. Pengembangan ini diimplementasikan dengan berbasis pada infrastruktur *middleware*, yang juga merupakan sebuah standar, yang memberikan layanan dasar dari arsitekur yang kompleks melalui *event-driven programming* dan standar yang berbasiskan *message engine* (bus/jalur yang digunakan).

Sebuah ESB pada dasarnya menyajikan sebuah layer abtsraksi diatas implementasi dari sebuah *enterprise messanging system*, yang memberbolehkan arsitektur integrasi untuk memanfaatkan hasil dari pengiriman pesan tanpa menuliskan kode program. Sebaliknya pada integrasi aplikasi *enterprise* menggunakan pendekaan dengan susunan arsitektur *hub and spoke*, dasar dari ESB adalah membangun sebuah fungsi dasar yang membagi fungsi-fungsi tersebut menjadi bagian-bagian yang utama dengan pengembangan sistem terdistribusi sebagai sebuah kesatuan.

Sebuah ESB tidak diimplementasikan sebagai sebuah *service-oriented architecture* (SOA) tetapi memberikan fitur yang lebih mudah diimplementasikan. Meskipun pada dsarnya ESB bukan merupakan sebuah *web service*. Standar ESB lebih *flexible*, dan mendukung lebih banyak metode transport. Pada dasarnya *enterprise architecture integration* yang lebih berbasiskan SOA, yang menghapus batas antara layanan dengan media komunikasi perangkat lunak.

#### 2.2.11.4 *Simple Object Access Protocol* (SOAP)

Sebuah protokol komunikasi *client-server* yang mengirim dan menerima informasi "di atas" HTTP. Data dikirim dan diterima dalam format XML khusus yang dirancang untuk SOAP. SOAP serupa dengan protokol XMLRPC kecuali bahwa SOAP lebih cocok digunakan untuk menangani data kompleks yang dikirim antar *client* dan *server*.

SOAP (*Simple Object Access Protocol*) adalah standar untuk bertukar pesan-pesan berbasis XML melalui jaringan komputer atau sebuah jalan untuk program yang berjalan pada suatu sistem operasi (OS) untuk berkomunikasi dengan program pada OS yang sama maupun berbeda dengan menggunakan HTTP dan XML sebagai mekanisme untuk pertukaran data [11]..

SOAP menspesifikan secara jelas bagaimana cara untuk *meng-encode header* HTTP dan file XML sehingga program pada suatu komputer dapat memanggil program pada pada komputer lain dan mengirimkan informasi, dan bagaimana program yang dipanggil memberikan tanggapan.

SOAP adalah protokol ringan yang ditujukan untuk pertukaran informasi struktur pada lingkup desentralisasi, dan terdistribusi. SOAP menggunakan teknologi XML utuk mendefinsiskan rangka kerja pemesanan terekstrensi di mana menyediakan konstruksi pesan yang dapat dipertukarkan pada protokol berbeda. Rangka kerja dirancang bebas dari model pemrograman dan spesifikasi implementasi semantic [11].

Kelebihan SOAP dibandingkan dengan metode integrasi yang ada adalah sebagai berikut :

1. *Firewall*. Tujuan protokol ini meletakkan landasan yang kompatibel lintas lingkungan-lingkungan yang berbeda. Tak ada sesuatu terlalu rumit ditambahakan diatas antarmuka CGI. Firewall bisa mengawasi semua transaksi Content-Type yang berupa text/xml
2. *Discoverability*. Yaitu berupa penyusunan format yang sangat sederhana yang memungkinkan penulis HTML biasa untuk melihat file yang berisi XML-RPC, mengerti apa yang sedang dilakukan dan dapat memodifikasinya dan membuatnya bekerja kembali pada penggunaan berikutnya.
3. Mudah diimplementasikan. Membuat sebuah protokol yang mudah diimplementasikan dan bisa diadaptasikan di lingkungan lain atau sisem operasi lainnya.

### 2.2.12. XML-RPC

XML-RPC merupakan spesifikasi dan kumpulan implementasi yang membuat software yang berjalan pada sistem operasi berbeda, lingkungan berbeda

dapat membuat pemanggilan prosedur melalui internet. XML-RPC menggunakan HTTP sebagai media transportasi dan XML (eXtensible Markup Language) sebagai encoding pesan. XML-RPC didesain sesedarhana mungkin dalam mentransmit struktur data yang kompleks, memprosesnya dan mengembalikannya. Struktur data yang menggunakan encoding XML merupakan badan request HTTP-POST dari *client*. Cara paling mudah memahami XML-RPC adalah dengan melihat proses mulai dari request ke *server*, lalu bagaimana *server* menerjemahkan pesan (*parsing*) berformat XML, bagaimana *server* mendapatkan pesan *client* yang sudah di parsing ternyata.. katakanlah merequest prosedur jumlah(x, y), kemudian *server* mengeksekusi prosedur jumlah(x, y) dan hasilnya (misal hasil jumlah x + y) di encode XML lalu di kembalikan ke *client* sebagai response dari request-nya [11].

SOAP juga memiliki standard mekanisme dalam pertukaran informasi menggunakan format XML. SOAP memiliki tiga bagian, yaitu:

1. *SOAP Envelope*, yang mendefinisikan dokumen XML sebagai pesan SOAP. Element `Envelope` merupakan element root yang dapat menggunakan identifier namespace dari http://schemas.xmlsoap.org/soap/envelope/. Struktur dasar SOAP Envelope kurang lebih seperti berikut:

```
<?xml version="1.0"?>
<soap:envelope      xmlns:soap="http://www.w3.org/2001/12/soap-
envelope"   soap:encodingstyle="http://www.w3.org/2001/12/soap-
encoding">
  ...
  Pesan informasi, bisa mengandung
```

```
element Header dan harus
terdapat element Body
...
 </soap:envelope>
```

Element *Envelope* yang mendeklarasikan *namespace* (seperti di atas) harus menggunakan namespace-qualified untuk atribut tambahan lainnya (seperti `soap:encodingStyle`) serta sub-element-nya (opsional element `<SOAP-ENV:Header>` dan element `<SOAP-ENV:Body>` yang wajib ada). Penggunaan atribut encodingStyle bertujuan untuk mendefinisikan tipe data yang digunakan dalam dokumen. Atribut ini dapat muncul di element SOAP manapun, dan efeknya akan terkena ke konten element yang mendefinisikan atribut tersebut serta sub-element-nya.

2. Aturan *encoding* SOAP, yang mendefinisikan mekanisme serialisasi pesan SOAP. Aturan serialisasi pada contoh di atas di identifikasikan oleh URI http://schemas.xmlsoap.org/soap/encoding/. Pesan SOAP yang menggunakan aturan serialisasi itu harus mendeklarasikan atribut encodingStyle dengan nilainya berisi URI http://schemas.xmlsoap.org/soap/encoding/.
3. SOAP RPC, yaitu bagaimana SOAP dapat digunakan seperti halnya XML-RPC. Dalam SOAP sebenarnya dikenal 2 tipe aplikasi, yaitu RPC dan EDI (*Electronic Document Exchange*). SOAP RPC (atau lebih dikenal sebagai “RPC-style” SOAP) menggunakan pesan SOAP sebagai representasi nama prosedur berserta opsional parameternya dan nilai balik dari prosedur. SOAP EDI (atau dikenal sebagai “document-style” SOAP) merupakan pesan SOAP yang merepresentasikan transaksi bisnis seperti order pembelian, pembayaran

pajak atau dokumen sejenis lainnya. Sebenarnya, semua *Web Servicess* yang saya bahas ini lebih ke RPC-style, yaitu bagaimana *client* merequest informasi terencoding berisi nama prosedur beserta parameternya yang ingin dipanggil dan *server* merespon nilai balik hasil eksekusi prosedur ke *client*. Seperti Contoh *server* menyediakan fungsi hello($name) dimana nilai balik yang dihasilkan adalah string “hello $name” ($name adalah parameter bertipe string). Berikut contoh request dari *client* untuk pemanggilan fungsi hello dengan paramater “gedex”:

```
POST /soap/server.php?debug=1 HTTP/1.0
Host: localhost
User-Agent: NuSOAP/0.7.3 (1.114)
Content-Type: text/xml;
charset=ISO-8859-1
SOAPAction: ""
Content-Length: 542

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope                                          SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Header>1</SOAP-ENV:Header>
<SOAP-ENV:Body>
  <ns3898:hello xmlns:ns3898="http://tempuri.org">
    <name xsi:type="xsd:string">gedex</name>
```

```
  </ns3898:hello>
</SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

Jika dilhat, pesan SOAP di atas menggunakan namespace-qualified SOAP-ENV (default dari nuSOAP) dan itu pun harus digunakan pada atribut serta sub-element-nya. Nama fungsi yang ingin dipanggil merupakan sub-element (yaitu `<ns3898:hello>`) dari element body (`<SOAP-ENV:Body>`) dan parameter nya merupakan konten dari sub-element hello (yaitu konten dari element `<name>`). Element `<hello>` menggunakan identifier namespace dari http://tempuri.org. Nah namespace ini juga berpengaruh kepada konten ataupun sub-elementnya (dengan begitu element `<name>` menggunakan namespace dari element parent-nya). Setelah *server* menganggap pesan SOAP *client* valid, dan eksekusi prosedur memberikan nilai balik, kini saatnya *server* memberikan response dalam pesan SOAP juga. Berikut contoh response pesan SOAP atas request di atas:

```
HTTP/1.1 200 OK
Date: Sun, 13 Apr 2008 08:12:07 GMT
Server: Apache/2.2.8 (Win32) DAV/2 mod_ssl/2.2.8
OpenSSL/0.9.8g mod_autoindex_color PHP/5.2.5
X-Powered-By: PHP/5.2.5 X-SOAP-Server: NuSOAP/0.7.3 (1.114)
Content-Length: 6916
Connection: close
Content-Type: text/xml; charset=ISO-8859-1

<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<SOAP-ENV:Envelope                                          SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
  <ns1:helloResponse xmlns:ns1="http://tempuri.org">
    <return xsi:type="xsd:string">Hello gedex</return>
  </ns1:helloResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
<!--
2008-04-13 16:12:07.453128 soap_server: _SERVER is defined:
...

debug data from nusoap_server ...

...
-->
```

Content-Length di atas cukup panjang, karena saya merequest nusoap_*server* (*server*.php) dengan ?debug=1, jadi *server* akan memberikan data *debug* dalam bentuk komentar. Nah, kita bisa lihat pesan SOAP dari *response* memberikan nilai balik dalam element `<return>` berisi “Hello gedex”. SOAP juga ada SOAP fault. Berikut contoh SOAP fault atas kesalahan pemanggilan nama prosedur:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<SOAP-ENV:Envelope                                    SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
  <SOAP-ENV:Fault>
    <faultcode                    xsi:type="xsd:string">SOAP-
ENV:Client</faultcode>
    <faultactor xsi:type="xsd:string"></faultactor>
    <faultstring  xsi:type="xsd:string">method  'halo'  not
defined in service</faultstring>
    <detail xsi:type="xsd:string"> ... panjang bay klo gw
paste  ... </detail>
  </SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP fault terdiri atas 4 sub-element, yaitu *faultcode*, *faultactor*, *faultstring* dan detail. *faultcode* merupakan nilai yang dihasilkan untuk mengidentifikasi jenis kesalahan yang terjadi. Nilainya harus *qualified-namespace* (sebagaima kita lihat, yaitu SOAP-ENV:*Client*). Keempat jenis *faultcode* tersebut dapat dikembangkan lebih jauh untuk memberikan tipe fault yang lebih ekspresif dan sempit, tapi tetap harus menjaga kompabilitas dengan core *faultcode*. Misal faultcode *Client* dapat memiliki turunan faultcode *Client*.Authentication untuk lebih mempersempit

ekspresi *fault* (*Client.Authentication* menunjukkan terjadi kesalahan dalam autentifikasi si *client*, ya bisa dari salah *password*).

### 2.2.13. Web Services

XML *Web Servicess* dapat di definisikan sebagai aplikasi yang diakses oleh aplikasi yang lain. Mungkin orang berpendapat itu semacam web site, tetapi itu bukan demikian

Sesuai dengan namanya, XML *Web Services* menyimpan datanya dalam format XML dan karenanya menjadi multi *platform* dalam hal accessibilitynya. Bagi yang telah mengenal pemrograman dengan VB6, Delphi, Java, atau bahasa lain maka telah mengenal adanya berbagai library yang di dalamnya terdapat berbagai function, property, dan event untuk memudahkan pembuatan aplikasi [11]..

*Web Services* dapat disamakan dengan *library* tersebut tetapi tidak memerlukan registrasi khusus kedalam sistem operasi yang menyimpannya. *Web Services* tersimpan di *Web Server* sehingga dapat diakses oleh berbagai bahasa pemrograman dengan lebih mudah baik dalam lingkungan LAN maupun Internet. Sistem *Web Services* ini diharapkan meningkatkan kolaborasi antar pemrogram dan perusahaan, yang memungkinkan sebuah fungsi di dalam *Web Services* dapat dipinjam oleh aplikasi lain tanpa perlu mengetahui detil pemrograman yang terdapat di dalamnya.

W3C mendefinisikan *Web Services* sebagai subuah sistem perangkat lunak yang dirancang untuk mendukung komunikasi dan interaksi antar mesin ke mesin (*Machine to Machine*) melalui sebuah network (jaringan). *Web Services* juga

termasuk Web APIs yang dapat diakses melalui jaringan seperti misalnya internet, dan dieksekusi melalui sebuah sistem jarak jauh sesusai dengan layanan yang diminta.

*Web Services* menurut W3C juga meliputi banyak sistem berbeda, tetapi pada umumnya lebih menyangkut pada *client* dan *server* yang berkomunikasi menggunakan XML yang memenuhi standard SOAP (*Simple Object Access Protocol*). Asumsi secara umum adalah pada terminologi terdapat deskripsi dari mesin yang layanannya disediakan oleh *server*, atau sama seperti konsep dari WSDL. WSDL bukan termasuk standard dari SOAP tetapi merupakan syarat mutlak untuk *client-side* otomatis pada framework Java dan .NET SOAP. Beberapa organisasi industri seperti WS-I mengklaim baik SOAP dan WSDL sebagai inti *Web Services*.

Definisi lain juga menyebutkan bahwa *Web Services* merupakan standar yang memungkinkan adanya interaksi informasi teknologi yang terintregasi antara proses dan sistem. Kita dapat mendeskripsikannya sebagai keluaran baru dari *Web application* yang dapat mennyediakan layanan dari yang palin mudah dan sederhana sampai yang paling rumit dan kompleks, juga proses-proses yang bersifat *scientific*. *Web Services* menyediakan mekanisme standard untuk komunikasi yang terintegrasi antara sistem-sistem yang berjauhan. Mekanisme ini membuat *Web Services* ideal untuk pengimplementasian *Service-Oriented Architecture (SOA)* [11].

Untuk kondisi sekarang, *Web Servicess* cocok untuk menyelesaikan masalah pada sistem bisnis konsep lama ke sistem bisnis terintergrasi. Sehingga

dengan sebuah model konsep bisnis dapat di akses dan dipergunakann macam-macam aplikasi dan device. Ini semua bekerja dengan beberapa protocol standart yang sudah ada diantaraya :

1. XML
2. WSDL
3. SOAP
4. UDDI

**2.2.13.1. Kapan Kita Gunakan Web Servicess ?**

*Web Services* itu digunakan saat kita akan mentransformasi sebuat bisnis logik/sebuah *class* dan *object* yang terpisah dalam 1 ruang lingkup yang menjadi satu, sehingga tingkat keamanan dan security dapat di tangani dengan baik. Selain itu *Web Services* juga lebih mudah dalam process deploymentnya, karena tidak memerlukan registrasi khusus ke dalam sistem operasi. *Web Services* cukup diupload ke *Web Server* dan siap diakses oleh pihak-pihak yang telah diberikan otorisasi.

Web Services berjalan di port 80 yang merupakan protokol standar HTTP, dengan demikian mengurangi resiko terblokir oleh firewall. Kendala arsitektur COM/DCOM adalah memerlukan konfigurasi khusus di sisi *firewall,* dan ini tidak perlu dilakukan untuk mengakses *Web Services*. Beberapa vendor luar negeri mulai berkolaborasi satu sama lain dengan konsep *Web Services,* diantaranya : IBM , Microsoft , SUN , ORACLE [11]..

### 2.2.13.2. Spesifikasi dan Standard

Ada beberapa spesifikasi inti dari Web Services, misalnya:

1. SOAP: Berbasis XML, protokol primernya adalah HTTP dan HTTPS (termasuk SMTP dan XMPP).
2. WSDL: Format XML yang menyediakan layanan *interface* untuk dideskripsikan. Umumnya digunakan untuk kode pada *client* dan *server*, juga untuk konfigurasi.
3. UDDI: Protokol yang digunakan untuk mempublish dan mendiscover seluruh metadata tentang *Web Services* untuk menjalankan aplikasi dalam menemukan *Web Services*, baik dalam segi disain maupun waktu kerjanya.

Spesifikasi-spesifikasi di atas pada umumnya merupakan standard dari W3C, termasuk XML, SOAP2, dan WSDL. UDDI merupakan standard dari OASIS. Spesifikasi yang lain telah dikembangkan untuk memperluas kemampuan Web Services itu sendiri. Spesifikasi-spesifikasi ini lebih mengarah sebagai WS-*.

1. WS-Security: Menjelaskan bagaimana cara menggunakan XML *Encryption* dan XML *Signature* pada SOAP untuk keamanan pertukaran pesan, sebagai alternatif ataupun perpanjangan dalam penggunaan HTTPS untuk keamanan jalur.
2. WS-Reliability: Prokol standard dari OASIS untuk pertukaran pesan di antara dua *Web Services*.
3. WS-ReliableMessaging: Protokol untuk message di antara dua *Web Services* yang dikeluarkan oleh Microsoft, BEA, dan IBM yang distandardisasi oleh organisasi OASIS.

4. WS-Addressing: Cara untuk mendeskripsikan alamat dari penerima maupun pengirim pesan, selain SOAP itu sendiri.

Spesifikasi-spesifikasi dari *Web Services* bekerja bersama untuk keamanan maupun keandalan.

#### 2.2.13.3. Profil *Web Service*

Untuk lebih mengembangkan fungsinya, WS-I mempublikasikan profil. Profil mengeset spesifikasi inti (SOAP, WSDL, dll) pada versi yang lebih spesifik (SOAP 1.2, UDDI 2, dll) dengan beberapa tambahan untuk lebih memperketat penggunaan spesifikasi inti. WS-I juga mempublikasikan kasus dan tool.

#### 2.2.13.4. Penggunaan *Web Service*

*Web Services* adalah sekumpulan tools yang dapat digunakan dengan berbagai macam cara. Tiga cara yang sering digunakan adalah RPC, SOA, dan REST.

*Web Services* RPC menampilkan fungsi dan metode call interface yang terdistribusi yang bersifat familiar bagi banyak pengembang. Biasanya, unit dasar dari RPC ini adalah WSDL operation.

Pada awalnya, tools dari Web Services lebih difokuskan pada RPC dan sebagai hasilnya, adalah pengembangan yang lebih luas. Meskipun demikian, hal ini dirasa tidak efektif oleh berbagai vendor sehingga pada akhirnya RPC tidak diikutkan dalam WS-I Basic Profile [11]..

#### 2.2.13.5. Self-Oriented Architecture

*Web Services* juga dapat digunakan untuk pengimplementasian arsitektur yang berdasarkan pada konsep *Service-Oriented Architecture (SOA)* di mana unit dasar dari komunikasi itu sendiri adalah pesan, dan bukan operasi. Ini sering disebut dengan layanan *service oriented*. Banyak vendor-vendor software dan analis industri menggunakan konsep *Web Services* SOA ini [11]..

#### 2.2.13.6. Representational State Transfer

*Web Services RESTful* berencana untuk menandingi HTTP dan protokol lain yang sejenis dengan membatasi *interface* pada operasi yang standard (misalnya GET, PUT, DELETE). Di sini, fokus lebih pada stateful resource daripada pesan ataupun operasi. *RESTful* dapat menggunakan WSDL untuk mendeskripsikan pesan SOAP melalui HTTP, yang mendefinisikan operasi, atau dapat diimplementasikan sebagai abstraksi pada puncak SOAP (misalnya WS-Transfer) [11]..

### 2.2.14. Data Mining

*Data Mining* merupakan teknologi baru yang sangat berguna untuk membantu perusahaan-perusahaan menemukan informasi yang sangat penting dari *gudang data* mereka. Kakas *data mining* meramalkan tren dan sifat-sifat perilaku bisnis yang sangat berguna untuk mendukung pengambilan keputusan penting. Analisis yang diotomatisasi yang dilakukan oleh data mining melebihi yang dilakukan oleh sistem pendukung keputusan tradisional yang sudah banyak digunakan. *Data Mining* dapat menjawab pertanyaan-pertanyaan bisnis yang

dengan cara tradisional memerlukan banyak waktu untuk menjawabnya. *Data Mining* mengeksplorasi basis data untuk menemukan pola-pola yang tersembunyi, mencari informasi prediksi yang mungkin saja terlupakan oleh para pelaku bisnis karena terletak di luar ekspektasi mereka.

### 2.14.1. Definisi *Data Mining*

*Data mining* didefinisikan sebagai satu set teknik yang digunakan secara otomatis untuk mengeksplorasi secara menyeluruh dan membawa ke permukaan relasi-relasi yang kompleks pada set data yang sangat besar. Set data yang

Dimaksud di sini adalah set data yang berbentuk tabulasi, seperti yang banyak diimplementasikan dalam teknologi manajemen basis data relasional. Akan tetapi, teknik-teknik *data mining* dapat juga diaplikasikan pada representasi data yang lain, seperti domain data *spatial*, berbasis text, dan multimedia (citra). *Data mining* dapat juga didefinisikan sebagai "pemodelan dan penemuan polapola yang tersembunyi dengan memanfaatkan data dalam volume yang besar".

*Data mining* menggunakan pendekatan *discovery-based* dimana pencocokan pola (*pattern-matching*) dan AlgoritmaAlgoritma yang lain digunakan untuk menentukan relasi-relasi kunci di dalam data yang diekplorasi. *Data mining* merupakan komponen baru pada arsitektur sistem pendukung keputusan (DSS) di perusahaan-perusahaan.

### 2.2.14.2. Ruang Lingkup *Data Mining*

*Data mining* (penambangan data), sesuai dengan namanya, berkonotasi sebagai pencarian informasi bisnis yang berharga dari basis data yang sangat besar. Usaha pencarian yang dilakukan dapat dianalogikan dengan penambangan

logam mulia dari lahan sumbernya. Dengan tersedianya basis data dalam kualitas dan ukuran yang memadai, teknologi *data mining* memiliki kemampuan-kemampuan sebagai berikut [6]:

1. Mengotomatisasi prediksi tren dan sifat-sifat bisnis. *Data mining* mengotomatisasi proses pencarian informasi pemprediksi di dalam basis data yang besar. Pertanyaan-pertanyaan yang berkaitan dengan prediksi ini dapat cepat dijawab langsung dari data yang tersedia. Contoh dari masalah prediksi inimisalnya target pemasaran, peramalan kebangkrutan dan bentukbentuk kerugian lainnya
2. Mengotomatisasi penemuan pola-pola yang tidak diketahui sebelumnya. Kakas *data mining* "menyapu" basis data, kemudian mengidentifikasi pola-pola yang sebelumnya tersembunyi dalam satu sapuan. Contoh dari penemuan pola ini adalah analisis pada data penjulan ritel untuk mengidentifikasi produkproduk, yang kelihatannya tidak berkaitan, yang seringkali dibeli secara bersamaan oleh kustomer. Contoh lain adalah pendeteksian transaksi palsu dengan kartu kredit dan identifikasi adanya data anomaly yang dapat diartikan sebagai data salah ketik (karena kesalahan .operator)

### 2.2.14.3. Cara Kerja *Data Mining*

Bagaimana tepatnya *data mining* "menggali" hal-hal penting yang belum diketahui sebelumnya atau memprediksi apa yang akan terjadi. Teknik yang digunakan untuk melaksanakan tugas ini disebut pemodelan. Pemodelan di sini dimaksudkan sebagai kegiatan untuk membangun sebuah model pada ituasi yang

telah diketahui “jawabannya” dan kemudian menerapkannya pada situasi ain yang akan dicari jawabannya [6].

### 2.2.14.4. Tugas Utama *Data Mining*

Telah disebutkan di ruang lingkup *data mining* bahwa pada kebanyakan aplikasinya, gol utama dari *data mining* adalah untuk membuat prediksi dan deskripsi. Prediksi menggunakan beberapa variabel atau field-field basis data untuk memprediksi nilai-nilai variabel masa mendatang yang diperlukan, yang belum iketahui saat ini. Deskripsi berfokus pada penemuan pola-pola tersembunyi dari data yang ditelaah..

Prediksi dan deskripsi pada *data mining* dilakukan dengan tugas-tugas utama yang akan antara lain [23]:

a) Klasifikasi adalah fungsi pembelajaran yang memetakan (mengklasifikasi) sebuah unsure (item) data ke dalam salah satu dari beberapa kelas yang sudahdidefinisikan.
b) Regresi adalah fungsi pembelajaran yang memetakan sebuah unsur data ke sebuah variabel prediksi ernilai nyata
c) Pengelompokan (*clustering*) merupakan tugas deskripsi yang banyak digunakan dalam mengidentifikasi sebuah himpunan terbatas pada kategori atau *cluster* untuk mendeskripsikan data yang ditelaah. Peringkasan melibatkan metodologi untuk menemukan deskripsi yang ringkas dari sebuah himpunan data
d) Pemodelan Kebergantungan adalah penemuan sebuah model yang mendeskripsikan kebergantungan yang signifikan antara variabelvariabel

e) Pendeteksian Perubahan dan Deviasi berfokus pada penemuan perubahan yang paling signifikan di dalam data dari nilai-nilai yang telah diukur sebelumnya.

#### 2.2.1.15. Algoritma *Microsoft Decision Trees*

Algoritma *Microsoft Decision Trees* adalah sebuah Algoritma yang digunakan untuk klasifikasi dan regresi yang disediakan oleh Analysis Services yang terdapat pada Microsoft SQL *Server* yang digunakan dalam prediksi atribut model diskrit maupun kontinu [23].

Untuk atribut diskrit, Algoritma membuat prediksi berdasarkan hubungan antara kolom input dalam dataset. Algoritma akan menggunakan nilai tersebut dengan state dari kolom input untuk memprediksi kondisi sebuah kolom yang dirancang untuk menghasilkan hasil prediksi. Secara lebih khusus, Algoritma *Microsoft Decision Trees* akan mengidentifikasi kolom input yang memiliki korelasi dengan kolom hasil prediksi. Sebagai contoh, dalam sekenario untuk memprediksi pelanggan mana yang memilih untuk membeli sebuah sepeda, jika 9 dari 10 anak muda membeli sebuah sepeda, tapi hanya 2 dari 10 pelanggan yang telah menikah melakukan hal tersebut, Algoritma *Microsoft Decision Trees* akan menyimpulkan rentang umur mana yang lebih memilih untuk membeli sepeda. Algoritma *Microsoft Decision Trees* melakukan prediksi berdasarkan kecendrungan kearah yang lebih akurat. Untuk atribut kontinu, Algoritma *Microsoft Decision Trees* menggunakan regresi linear untuk menentukan dimana sebuah keputusan percabangan dibagi.

Jika lebih dari satu kolom yang digunakan untuk menempatkan hasil prediksi, atau jika data masukan memiliki tabel yang bersarang yang digunakan untuk prediksi, maka Algoritma *Microsoft Decision Trees* akan membuat percabangan yang terpisah untuk setiap kolom prediksi.

**Contoh :**

Bagian pemasaran dari sebuah perusahaan inginkan untuk mengidentifikasi karakteristik dari pelanggan sebelumnya yang telah melakukan transaksi untuk menandakan apakah pelanggan tersebut suka untuk membeli produk itu lagi ataukah tidak pada transaksi berikutnya. Perusahaan tersebut akan menyimpan informasi dalam *database* berupa informasi demografi yang menjelaskan pelanggan tersebut. Dengan menggunakan Algoritma *Microsoft Decision Trees* untuk menganalisa informasi ini, bagian pemasaran dapat membangun sebuah model untuk prediski keadaan pelanggan dalam membeli sebuah produk, berdasarkan status yang diketahui akan ditempatkan pada kolom input sebagai demografi pelanggan dan pola transaksi yang yang telah dilakukan.

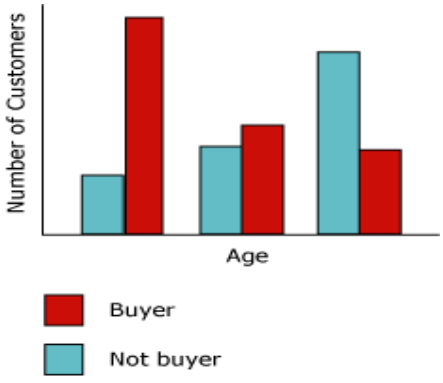
### 2.2.1.15.1. Bagaimana Algortitma *Microsoft Decision Trees* Bekerja

Algoritma *Microsoft Decision Trees* membangun sebuah model tambang data (*data mining*) dengan membagi bagian dari pohon. Bagian-bagian ini disebut dengan *nodes*. Algoritma *Microsoft Decision Trees* menambahkan *node* pada model setiap kali kolom input ditemukan untuk mebuat kolom prediksi yang lebih akurat. Jalur Algoritma significantly menentukan pembaguan berbeda-beda bergantung pada kondisi prediksi dari kolom kontinu atau kolom diskrit.

Algoritma *Microsoft Decision Trees* menggunakan seleksi untuk melakukan seleksi dari atribut yang banyak digunakan. Seleksi digunakan oleh semua Algoritma *analysis services data mining* untuk mengembangkan performa dan kualitas analisis. Seleksi sangat penting untuk mencegah atribut yang tidak dipakai dalam proses analisis. Jika kita menggunakan terlalu banyak input atau atribut prediksi ketika melakukan perancangan model *data mining*, model tersebut akan memakan waktu proses yang sangat lama dan dengan demikian memakan memory pada saat dijalankan.

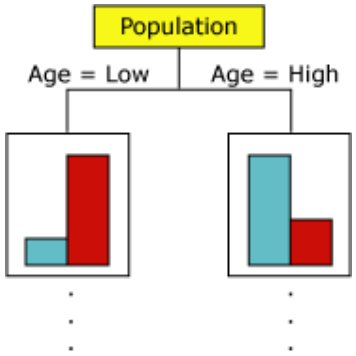
**2.2.1.15.2. Prediksi Kolom Diskrit**

Jalur pembentukan tree dari Algoritma *Microsoft Decision Trees* untuk kolom prediksi yang diskrit dapat ditunjukan dengan menggunakan histogram. Gambar 2.10 menunjukan sebuah diagram batang dari kolom prediksi, pembeli sepeda, dan kolom input dalm hal ini umur. Histogram menunjukan bahwa umur dari seseorang menentukan apakah dia suka untuk membeli sepeda ataukan tidak.




Gambar 2.10 Histogram Prediksi Kolom Diskrit

Korelasi yang ditunjukan pada diagram akan dibentuk tree oleh Algoritma *Microsoft Decision Trees* seperti tampak pada gambar 2.11.

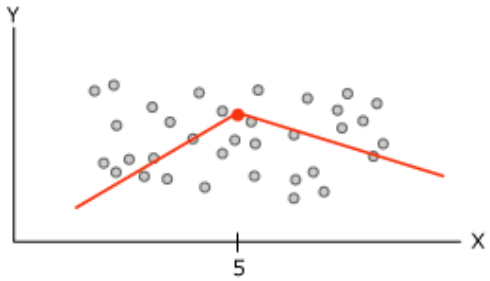



Gambar 2.11 Tree Prediksi Kolom Diskrit

Model yang dibentuk akan menunjukan node dari atas kebawah dengan membagi kolom prediksi dari keseluruhan popolasi pelanggan. Model tersebut akan terus berkembang setiap pertambahan kolom input.

**2.2.1.15.3. Prediksi Kolom Kontinu**

Pada saat Algoritma *Microsoft Decision Trees* membangun tree berdasarkan sebuah kolom prediksi yang kontinu, setiap node akan membawa sebuah formula regresi. Sebuah pembagian akan terjadi pada sebuah titik non linear pada formula regresi. Sebagai contoj dapat dilihat pada diagram gambar 2.11.




Gambar 2.12 Diagram Prediksi Kolom Kontinu

Diagram tersebut membawa data yang dapat dimodelkan dengan denggunakan sebuah garis, atau dengan menggunakan dua buah garis terhubung.

Ketika kita mempersiapkan data untuk dianalisa Algoritma *Microsoft Decision Trees*, kita harus mengerti kebutuhan dari Algoritma, dengan memasukan data yang dibutuhkan dan bagaimana data tersebut digunakan.

Kebutuhan dari Algoritma *Microsoft Decision Trees* antara lain :

1. Sebuah kolom tunggal, setiap model harus memiliki sebuah nilai angka atau teks yang unik untuk identifikasi setiap record.
2. Sebuah kolom prediksi, membutuhkan minimal sebuah kolom prediksi. Kita dapat memasukan beberapa atribut prediksi dalam sebuah model, dan atribut prediksi bisa memiliki tipe data yang berbeda-beda baik teks maupun angka. Hanya saja, dengan menggunakan angka akan mempercepat waktu pemrosesan.
3. Kolom input, membutuhkan sebuah kolom input, bisa diskrit maupun kontinu.