# BAB II

# LANDASAN TEORI

## 2.1 Definisi Kecerdasan Buatan (*Artificial Intellegence*)

Ha Simon (1987) mendifinisikan kecerdasan buatan merupakan kawasan penelitian, aplikasi dan instruksi yang terkait dengan pemrograman komputer untuk melakukan sesuatu hal yang dalam pandangan manusia adalah cerdas.

Kecerdasan buatan adalah salah satu bidang ilmu komputer yang mendayagunakan komputer sehingga dapat berperilaku cerdas seperti manusia. Ilmu komputer tersebut mengembangkan perangkat lunak dan perangkat keras untuk menirukan tindakan manusia. Aktifitas manusia yang ditirukan seperti penalaran, penglihatan, pembelajaran, pemecahan masalah, pemahaman bahasa alami dan sebagainya. Sesuai dengan definisi tersebut, maka teknologi kecerdasan buatan dipelajari dalam bidang – bidang seperti : Robotika (*Robotics*), Penglihatan computer (*Computer Vision*), Pengolahan bahasa alami (*Natural Language Processing*), Pengenalan pola (*Pattern Recognition*), Sistem Syaraf Buatan (*Artificial Neural System*), Pengenalan Suara (*Speech Recognition*), dan Sistem Pakar (*Expert System*). Kecerdasan Buatan menyelesaikan permasalahan dengan mendayagunakan computer untuk memecahkan masalah yang komplek dengan cara mengikuti proses penalaran manusia. Salah satu teknik kecerdasan buatan yang menirukan proses penalaran manusia adalah Sistem pakar.

## 2.2 Sistem Pakar

Sistem pakar *(Expert System)* merupakan salah satu cabang dari kecerdasan buatan. Secara umum, sistem pakar (expert system) adalah sistem yang berusaha mengadopsi pengetahuan manusia ke dalam komputer, agar komputer dapat menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli. (Sri Kusumadewi, 2003)

Sistem pakar adalah sebuah perangkat lunak komputer yang memiliki basis pengetahuan untuk domain tertentu dan menggunakan penalaran inferensi menyerupai seorang pakar dalam memecahkan masalah. Kekuatannya terletak pada kemampuannya memecahkan masalah-masalah praktis pada saat berhalangan. Sistem pakar merupakan program-program praktis yang menggunakan strategi heuristik yang dikembangkan oleh manusia untuk memecahkan masalah-masalah yang spesifik.

Sifat utama sistem pakar adalah ketergantungan sistem ini pada pengetahuan manusia yang pakar dalam suatu bidang dalam menyusun strategi pemecahan persoalan yang dihadapi oleh sistem. Pengetahuan pakar (*expert knowladge*) merupakan kombinasi pemahaman teoritis tentang suatu persoalan dan sekumpulan aturan pemecahan persoalan heuristik dimana pengalaman telah menunjukan kefektifannya dalam persolan itu.

### 2.2.1 Kelebihan Sistem Pakar

Secara garis besar banyak manfaat yang dapat diambil dengan adanya sistem pakar, antara lain (Kusrini, 2006) :

1. Memungkinkan orang awam bisa mengerjakan pekerjaan para ahli.

2. Dapat melakukan proses berulang secara otomatis.
3. Menyimpan pengetahuan dan keahlian para pakar.
4. Meningkatkan output dan produktivitas.
5. Mampu mengambil dan melestarikan keahlian para pakar.

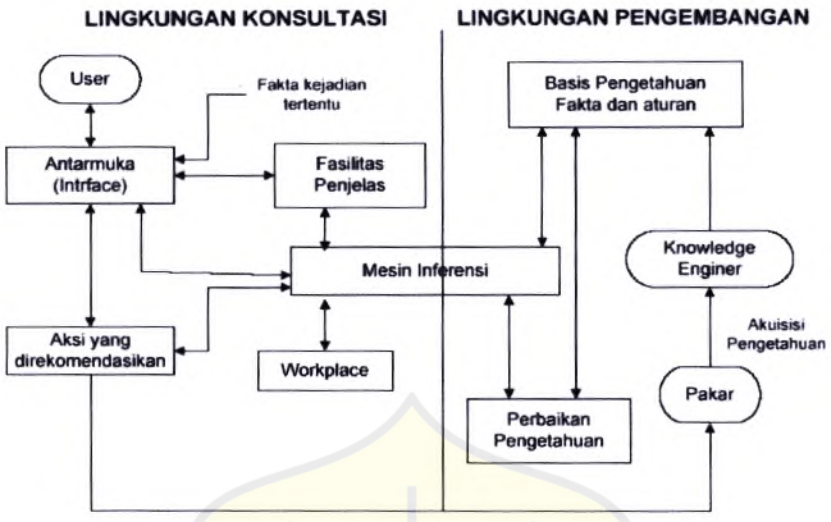
### 2.2.2 Kelemahan Sistem Pakar

Di samping memiliki beberapa keuntungan, sistem pakar juga memiliki beberapa kelemahan, antara lain :

1. Pengembangan sistem pakar lebih sulit daripada membuat *software* konvensional. Karena memadatkan pengetahuan seorang pakar dan mengalihkannya ke dalam sebuah program merupakan pekerjaan yang melelahkan, lama dan memerlukan biaya yang besar.
2. Sistem pakar sangat mahal. Untuk mengembangkan, mencoba dan mengirimkannya ke pemakai (*user*) memerlukan biaya yang besar.
3. pakar tidak 100 % menguntungkan, seorang tidak sempurna atau tidak selalu benar. Karena itu diuji ulang secara teliti sebelum digunakan, peranan manusia tetap merupakan faktor dominan.

### 2.2.3 Arsitektur Sistem Pakar

Sistem pakar disusun oleh dua bagian utama, yaitu Lingkungan Pengembangan dan Lingkungan Konsultasi. Lingkungan pengembangan digunakan untuk memasukkan pengembangan pakar ke dalam lingkungan sistem pakar, Lingkungan konsultasi digunakan oleh nonpakar untuk memperoleh pengetahuan dan nasihat pakar. (Turban,2005)




Gambar 2.1 Arsitektur Sistem Pakar (Turban, 2005)

Komponen – komponen yang terdapat dalam sistem pakar adalah :

1) Pakar, merupakan seseorang yang ahli di bidang tertentu.
2) Akuisisi Pengetahuan, merupakan penerimaan atau perolehan pengetahuan yang dapat diperoleh dari seorang pakar, buku teks, laporan penelitian dengan dukungan dari seorang *knowledge engineer.*
3) *Knowledge Engineer,* yaitu seorang spesialis sistem yang menerjemahkan pengetahuan yang dimiliki seorang pakar menjadi pengetahuan yang akan tersimpan dalam basis pengetahuan pada sebuah sistem pakar.
4) Basis Pengetahuan, terdiri dari dua jenis yaitu fakta dan *rule.*
5) Perbaikan Pengetahuan, yakni mereka dapat menganalisis pengetahuannya sendiri dan kegunaannya, belajar darinya dan meningkatkannya untuk konsultasi mendatang.
6) Mesin inferensi, merupakan otak dari sistem pakar berupa perangkat lunak yang melakukan tugas inferensi penalaran sistem pakar. Pada prinsipnya mesin inferensi inilah yang akan mencari solusi dari suatu permasalahan.

Konsep yang digunakan untuk mesin inferensi adalah runut balik (*backward chaining*) dan runut maju (*forward chaining*).

7) *Workplace,* merupakan area dari sekumpulan memori kerja *(working memory). Workplace* digunakan untuk merekam hasil dan kesimpulan yang dicapai. Ada 3 tipe keputusan yang dapat direkam, yaitu :
   a) Rencana : Bagaimana menghadapi masalah.
   b) Agenda : Aksi – aksi yang potensial yang sedang menunggu untuk dieksekusi.
   c) Solusi : Calon aksi yang dibangkitkan.
8) Fasilitas Penjelasan, proses menentukan keputusan yang dilakukan oleh mesin inferensi selama sesi konsultasi mencerminkan proses penalaran seorang pakar. Karena pemakai kadangkala bukanlah ahli dalam bidang tersebut, maka dibuatlah fasilitas penjelasan. Fasilitas penjelasan inilah yang dapat memberikan informasi kepada pemakai mengenai jalannya penalaran sehingga dihasilkan suatu keputusan.
9) Antarmuka *( Interface*), sistem pakar menggantikan seorang pakar dalam suatu situasi tertentu, sistem harus menyediakan pendukung yang diperlukan oleh pemakai yang tidak memahami masalah teknis. Sistem pakar juga menyediakan komunikasi antara sistem dan pemakainya, yang disebut sebagai antarmuka. Antarmuka yang efektif dan ramah pengguna (*user-friendly*) penting sekali terutama bagi pemakai yang tidak ahli dalam bidang yang diterapkan pada sistem pakar.

10) Aksi yang direkomendasi, merupakan saran atau solusi untuk permasalahan yang sedang dihadapi *user*.

11) *User* , yang dimaksud dengan *user* adalah :

a) *Learner* (pelajar) untuk mempelajari bagaimana sistem pakar menyelesaikan permasalahan.

b) *Client* (yaitu bukan pakar) yang menginginkan *advice* (nasehat). Bertindak seperti seorang konsultan atau penasehat.

c) Pakar, yang bertindak sebagai kologen atau asisten.

d) Pembangun sistem pakar yang ingin meningkatkan basis pengetahuan.

### 2.2.4 Representasi Pengetahuan

Sistem pakar merupakan sistem yang berbasis pengetahuan, mengerjakan tugas yang biasanya dilakukan oleh seorang pakar. Representasi pengetahuan dimaksudkan untuk mengorganisasikan pengetahuan dalam bentuk dan format tertentu untuk bisa dimengerti oleh komputer. Untuk membuat sistem pakar yang efektif harus dipilih representasi pengetahuan yang tepat. Pemilihan representasi pengetahuan yang tepat akan membuat sistem pakar dapat mengakses basis pengetahuan tersebut untuk keperluan pembuatan keputusan. Ada empat kriteria dalam memilih teknik representasi pengetahuan, yaitu :

a. Kemampuan representasi, artinya teknik yang dipilih harus merepresentasikan semua jenis pengetahuan yang akan dimasukan kedalam sistem pakar.

b. Kemudian dalam penalaran, artinya teknik yang dipilih harus mudah diproses untuk memperoleh kesimpulan.
c. Efisiensi proses akuisisi, artinya teknik yang dipilih harus membantu pemindah pengetahuan dari pakar kedalam komputer.
d. Efisiensi proses penalaran, artinya teknik yang dipilih harus dapat diproses dengan efisien untuk mencapai kesimpulan.

### 2.2.5 Mesin Inferensi

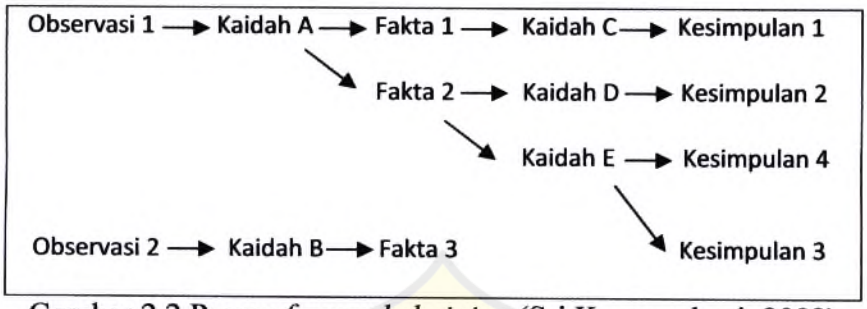
Mesin Inferensi adalah program komputer yang memberikan metedologi untuk penalaran tentang informasi yang ada dalam basis pengetahuan dan memformulasikan kesimpulan.

Mesin inferensi mengarahkan pencarian melalui basis pengetahuan, proses yang melibatkan aplikasi aturan inferensi disebut pencocokan pola. Program kontrol memutuskan aturan yang diinvestigasi, alternative yang dieliminasi dan atribut yang sesuai. Program kontrol yang popular untuk sistem berbasis aturan, yaitu *backward chaining* dan *forward chaining.*

**1. Penalaran Maju (*Forward Chaining*)**

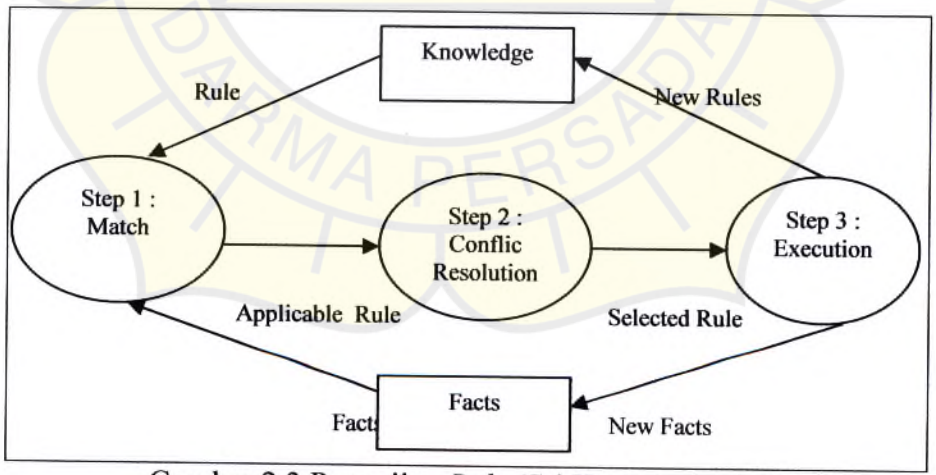
*Forward Chaining* adalah pendekatan yang dimotori data (*data-driven*). Metode *forward chaining* adalah suatu metode dari mesin inferensi untuk memulai penalaran suatu data dari fakta-fakta yang ada menuju suatu kesimpulan (Gonzales and Danket, 1993). Dalam *forward chaining*, kaidah interpreter mencocokkan fakta dalam basis data dengan situasi yang dinyatakan dalam bagian sebelah kiri atau kaidah If. Bila fakta yang

ada dalam basis data itu sudah sesuai dengan kaidah If, maka kaidah distimulasi.




Gambar 2.2 Proses *forward chaining* (Sri Kusumadewi, 2003)

*Forward chaining* juga merupakan suatu metode untuk menguji *rule-rule* satu demi satu dalam urutan tertentu. *Inference engine* akan mencocokkan fakta dalam *knowledge base* dengan situasi yang dinyatakan dalam *rule* bagian *IF*. Jika fakta yang ada dalam *knowledge base* itu sudah sesuai dengan kaidah *IF*, maka *rule* itu distimulasi dan *rule* berikutnya diuji dan saat kondisinya salah, *rule* itu tidak distimulasi dan rule berikutnya diuji.



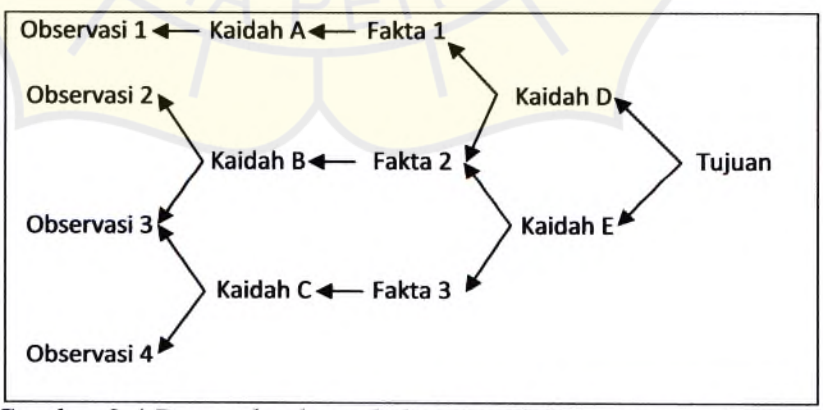

Gambar 2.3 Pengujian *Rule* (Sri Kusumadewi, 2003)

Fungsi masing-masing gambar diatas dijelaskan sebagai berikut :

1) *Matching.* Setiap *rule* yang ada pada basis pengetahuan dibandingkan dengan fakta-fakta yang diketahui untuk mencari *rule* mana yang memenuhi.
2) *Confilc Resolution.* Sangat mungkin dihasilkan suatu kondisi dimana beberapa *rule* dipenuhi. *Conflict Resolution* bertugas untuk mencari *rule* mana yang memiliki prioritas tertinggi berpotensi untuk dieksekusi.
3) *Execution.* Langkah terakhir dari proses ini adalah eksekusi dari *rule*. Proses ini menghasilkan dua kemungkinan, yaitu : fakta baru diturunkan dan ditambahkan *fact base* atau *rule* baru dihasilkan dan ditambahkan ke *knowledge base*.

**2. Penalaran Mundur ( *Backward Chainning* )**

*Backward Chainning* adalah pendekatan yang dimotori oleh tujuan (*goaldriven*). Metode *backward chaining* merupakan penalaran mundur yaitu suatu metode yang digunakan dalam *inference engine* untuk melakukan pelacakan dari sekumpulan hipotesa menuju fakta-fakta yang mendukung kesimpulan tersebut (Gonzales and Danket, 1993).

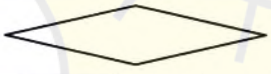



Gambar 2.4 Proses *backward chaining* (Sri Kusumadewi, 2003)

## 2.3 Entity Relationship Diagram (ERD)

Model data *Entity-Relationship* (ER) terdiri dari sekumpulan objek-objek yang disebut dengan entitas dan hubungan yang terjadi diantara objek-objek tersebut. Entitas merupakan suatu objek dasar atau individu yang mewakili sesuatu yang nyata eksistensinya dan dapat dibedakan dari objek-objek yang lain. Suatu entitas mempunyai sekumpulan sifat dan nilai dari beberapa sifat tersebut adalah unik yang dapat mengidentifikasi entitas tersebut. Sekumpulan entitas yang mempunyai tipe yang sama (sejenis) dan berada dalam lingkup yang sama membentuk suatu himpunan entitas. Suatu entitas memiliki atribut. Atribut merupakan sifat-sifat atau properti yang dimiliki oleh entitas. Atribut inilah yang membedakan antara entitas satu dengan entitas yang lain. Sedangkan relasi menunjukkan adanya hubungan diantara sejumlah entitas yang berasal dari sejumlah himpunan entitas yang berbeda (Widodo, P.A, dkk, 2004).

Tabel 2.1 Simbol-simbol ERD

| Simbol | Keterangan |
|---|---|
|  | Himpunan Entitas, digunakan untuk menggambarkan objek |
|  | Atribut, digunakan untuk menjelaskan karakter dari entity. |
|  | Relasi, menggambarkan himpunan relationship. |
|  | Garis, digunakan untuk menghubungkan entitas dengan entitas, entitas dengan relasi maupun entitas dengan atribut. |

ERD terdiri atas elemen – elemen yaitu Entitas, Atribut, Relasi dan Kunci (*Key*). Berikut ini merupakan penjelasan dari elemen – elemen tersebut :

a) Entitas

Entitas adalah suatu objek dasar atau individu yang mewakili sesuatu yang nyata dan dapat dibedakan dengan yang lainnya. Himpunan entitas adalah sekumpulan entitas dengan berbagai atribut yang sama.

b) Atribut

Atribut merupakan karakteristik yang melekat dalam sebuah entitas.

c) Relasi

Relasi adalah hubungan di antara sejumlah entitas yang berasal dari sejumlah himpunan entitas yang berbeda. Relasi atau hubungan dapat memiliki atribut. Sebuah kelas hubungan dapat melibatkan banyak kelas entitas.

d) Kunci (*Key*)

Kunci merupakan suatu atribut yang unik yang dapat digunakan untuk membedakan suatu entitas dengan lain dalam suatu himpunan entitas.Terdapat 3 macam kunci, yaitu :

1. *Superkey*, adalah himpunan yang terdiri dari satu atau lebih yang dapat membedakan setiap baris data dengan unik pada sebuah tabel.
2. *Candidate key*, adalah himpunan atribut minimal yang dapat membedakan setiap baris data dengan unik dalam sebuah tabel.
3. *Primary key,* merupakan kunci yang paling unik, lebih ringkas dan digunakan sebagai acuan.

e) Kardinalitas Relasi

Kardinalitas Relasi adalah jumlah maksimum entitas yang mana entitas tersebut dapat berelasi dengan entitas pada himpunan entitas yang lain. Relasi yang terjadi antara dua himpunan entitas (misalnya entitas dalam satu basis data yaitu (Abdul Kadir, 2002 :48) :

1. Satu ke satu (*one to one*)

   Hubungan relasi satu *entity* dengan satu *entity*.


2. Satu ke banyak (*one to many*)

   Hubungan banyak *entity* dengan satu *entity* atau satu *entity* dengan banyak.


3. Banyak ke banyak (*many to many*)

   Hubungan banyak *entity* dengan banyak *entity*.

## 2.4 *Unified Modeling Language* (UML)

*Unified Modeling Language* (UML) merupakan sistem arsitektur yang bekerja dalam OOAD (*Object-Oriented Analysis/Design*) dengan satu bahasa yang konsisten untuk menentukan, visualisasi, mengkontruksi, dan mendokumentasikan *artifact* (informasi yang dihasilkan dalam suatu proses rekayasa software, dapat berupa model, deskripsi atau software) yang terdapat dalam sistem software.

### 2.4.1 Fokus *Unified Modeling Language* (UML)

Menurut (Adi Nugroho, 2005), *Unified Modeling Language* (UML) menyediakan model-model yang tepat, tidak mendua arti (ambigu) serta lengkap. Secara khusus, UML menspesifikasikan langkah-langkah penting dalam pengambilan keputusan analisis, perancangan serta implementasi dalam sistem yang sangat bernuansa perangkat lunak *(software intensive system)*.

UML bukanlah merupakan bahasa pemograman tetapi model-model yang tercipta berhubungan langsung dengan berbagai macam bahasa pemograman, sehingga adalah mungkin melakukan pemetaan *(mapping)* langsung dari model-model yang dibuat dengan UML dengan bahasa-bahasa pemograman berorientasi obyek, seperti *Java, Borland Delphi, Visual Basic, C++*, dan lain-lain.

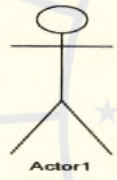
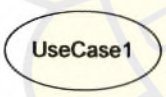
### 2.4.2 Diagram *Unified Modeling Language* (UML)

Berikut ini merupakan diagram dalam *Unified Modeling Language* (UML), yaitu :

1. *Use Case Diagram*

   Diagram ini memperlihatkan himpunan *use case* dan aktor-aktor. Diagram ini sangat penting untuk mengorganisasi dan memodelkan perilaku dari suatu sistem yang dibutuhkan pengguna. *Use case* menggambarkan kata kerja seperti *login* ke sistem, *maintenance user* dan lainnya. (Adi Nugroho, 2005)

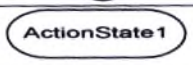
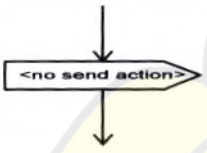
Tabel 2.2 Notasi *Use Case Diagram*

| Simbol | Nama | Keterangan |
|---|---|---|
|  | *Actor* | *Actor* adalah pengguna sistem. *Actor* tidak terbatas hanya manusia saja, jika sebuah sistem berkomunkasi dengan sistem lain dan memberikan input atau membutuhkan output, maka aplikasi tersebut bisa juga disebut sebagai *actor*. |
|  | *Use case* | *Use case* digambarkan sebagai lingkaran elips dengan nama *use case* dituliskandi dalam elips tersebut. |
|  | *Asosiasi* | *Asosiasi* digunakan untuk menghubungkan *actor* dengan *use case*. *Asosiasi* digambarkan dengan sebuah garis yang menghubungkan antara *actor* dengan *use case*. |

2. *Activity Diagram*

   Diagram ini memperlihatkan aliaran dari suatu aktifitas ke aktifitas lainnya dalam suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi dalam suatu sistem dan memberi tekanan pada aliran kendali antar objek. (Adi Nugroho, 2005)

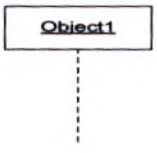
Tabel 2.3 Notasi *Activity Diagram*

| Simbol | Keterangan |
|---|---|
|  | Titik awal |
|  | Titik akhir |
|  | Activity |
|  | Pilihan untuk mengambil keputusan |
|  | *Fork :* digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu. |
|  | Tanda pengiriman |
|  | Tanda penerimaan |

3. *Sequence Diagram*

Diagram ini memperlihatkan interaksi yang menekankan pada pengiriman pesan (*message*) dalam suatu waktu tertentu. (Adi Nugroho, 2005)

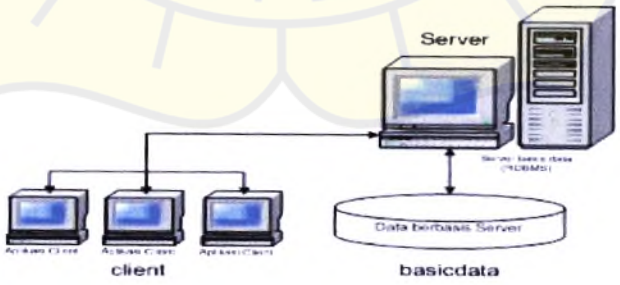
Tabel 2.4 Notasi *Sequence Diagram*

| Simbol | Nama | Keterangan |
|---|---|---|
|  | *Object* | *Object* merupakan sebuah *instance* dari sebuah *class* dan dituliskan tersusun secara horizontal. Digambarkan sebuah *class* (kotak) dengan nama *object* didalamnya yang diawali dengan sebuah titik koma. |
|  | *Actor* | *Actor* juga dapat berkomunikasi dengan *object*, maka *actor* juga dapat diurutkan sebagai kolom. Simbol *actor* sama dengan simbol *actor Use Case Diagram.* |

| | | |
|---|---|---|
| | *Lifeline* | *Lifeline* mengindikasikan keberadaan sebuah *object* dalam basis waktu. Notasi untuk *lileline* adalah garis putus-putus yang ditrik oleh sebuah *object.* |
| | *Activation* | *Activation* dinotasikan sebagai sebuah kotak segi empat yang digambarkan pada sebuah *lifeline. Activation* mengindikasikan sebuah *object* yang akan melakukan sebuah aksi. |
| Message1 | *Message* | *Message* digambarkan dengan anak panah horizontal antara *activation. Message* mengindikasikan komunikasi antara *object-object.* |

## 2.5 Microsoft SQL Server

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




Gambar 2.5 Sistem *Client /Server* (Marcus Teddy, 2004)

Sistem *client/server* adalah dirancang untuk memisah layanan basisdata dari *client*, dengan penghubungnya menggunakan jalur komunikasi data. Layanan basisdata diimplementasikan pada sebuah komputer yang berdaya guna, yang memungkinkan manajeman tersentralisasi, keamanan dan berbagai sumber daya. Oleh karena itu, *server* dalam *client/server* adalah basisdata dan layanannya. (Marcus Teddy, 2004)

## 2.6 Microsoft Visual Studio 2008

Pemograman Microsoft Visual Studio *.Net* 2008 adalah sebuah *platform* untuk membangun, menjalankan dan meningkatkan generasi lanjut dari aplikasi terdistribusi. *.Net Framework* merupakan *platform* terbaru untuk pemograman aplikasi window dari Microsoft dalam upaya meningkatkan produktivitas pembuatan sebuah program aplikasi dan memungkinkan terbukanya peluang untuk menjalankan program multi sistem operasi serta dapat memperluas pengembangan aplikasi *client – server.*

### 2.6.1 Komponen Utama *.Net Framework*

*.Net Framework* mempunyai dua komponen utama yaitu *Common Language Runtime* (CLR*)* dan *.Net Framework Class Library.*

1. Aplikasi-aplikasi yang dijalankan untuk *.Net Framework* dapat dijalankan pada suatu perangkat lunak yang mengatur persyaratan-persyaratan *runtime* program. *Runtime environment* inilah yang dikenal sebagai CLR. CLR sendiri berperan dalam mengatur kode pada saat di eksekusi, selain itu juga CLR menyediakan fasilitas

seperti manajemen memori, *thread*, keamanan aplikasi, dan penanganan kesalahan pada saat di eksekusi (*exception handle*). Kode yang menggunakan runtime dikenal dengan *managed code*, sedangkan kode yang tidak menggunakan *runtime* dikenal dengan *unmanaged code*. *Managed code* artinya program yang dibangun dengan menggunakan *.Net Framework,* hal ini dikarenakan *.Net Framework* sendiri melakukan pemeriksaan dan penanganan dalam banyak hal sebelum suatu aplikasi dijalankan, seperti masalah keamanan, dan lain sebagainya.

2. *.Net Framework Class Library* merupakan kumpulan *class-class* berorientasi objek yang digunakan untuk membuat suatu aplikasi, baik itu dalam modus konsol ataupun dalam modus grafik (GUI).

### 2.6.2 Teknologi Inti *.Net Framework*

*.Net Framework* mempunyai teknologi inti, berikut ini yang merupakan Teknologi inti *.Net :*

1. *.NET Framework*: menyediakan berbagai *library* untuk digunakan oleh aplikasi di atasnya.

2. *.NET Building Block Services*: *building block* merupakan sekumpulan *services* yang bersifat *programmable,* yang dapat diakses secara offline maupun online. Service tersebut merupakan modul-modul yang terdapat di suatu komputer, server dalam jaringan, maupun di suatu server di internet.

3. Visual Studio *.NET:* Visual Studio *.NET* menyediakan *tools* bagi para developer untuk membangun aplikasi yang berjalan di *.Net Framework.*
4. *.NET Enterprise Server* : merupakan sekumpulan *server based technology* yang digunakan untuk mendukung teknologi *.NET*, yang mencakup sistem operasi, database, *messaging,* maupun manajemen *e-commerce.*