BAB II

TINJAUAN PUSTAKA

II.1.Pengertian Sistem Pendukung Keputusan

Sistem Pendukung Keputusan atau *Decision Support System* yang selanjutnya kita singkat dalam skripsi ini menjadi SPK, secara umum didefenisikan sebagai sebuah sistem yang mampu memberikan kemampuan baik, kemampuan pemecahan masalah, maupun kemampuan pengkomunikasian untuk masalah semi-terstruktur. Secara khusus, SPK didefinisikan sebagai sebuah sistem yang mendukung kerja seorang manajer dalam memecahkan masalah semi-terstruktur dengan cara memberikan informasi ataupun usulan menuju pada keputusan tertentu.

Menurut Alter dalam Kusrini (2007, h. 15) Sistem Pendukung Keputusan atau *Decision Support Sistem* (DSS) "merupakan sistem informasi interaktif yang menyediakan informasi, pemodelan, dan pemanipulasian data. Dan sistem itu digunakan untuk membantu pengambilan keputusan dalam situasi yang semiterstruktur dan situasi yang tidak terstruktur, di mana tak seorang pun tahu secara pasti bagaimana keputusan seharusnya dibuat".

Tujuan dari sistem pendukung keputusan antara lain :

1. Membantu pegawai dalam pengambil keputusan atas masalah semiterstruktur.
2. Memberikan dukungan atas pertimbangan pegawai dan bukannya dimaksudkan untuk menggantikan fungsi pegawai.

3. Meningkatkan efektifitas keputusan yang diambil pegawai lebih dari pada perbaikan efisiensinya.
4. Meningkatkan kecepatan komputasi, untuk melakukan banyak komputasi secara cepat dengan biaya yang rendah.
5. Peningkatan produktifitas, produktifitas biasa ditingkatkan menggunakan peralatan optimalisasi yang menentukan cara terbaik untuk menjalankan sebuah bisnis.
6. Memberikan dukungan kualitas. Komputer bisa meningkatkan kualitas keputusan yang dibuat. Dengan komputer menilai berbagai pengaruh secara cepat dan ekonomis.
7. Meningkatkan daya saing. Teknologi pengambil keputusan bisa menciptakan pemberdayaan dengan cara memperbolehkan seorang untuk membuat keputusan yang baik dan tepat.
8. Mengatasi keterbatasan kognitif dalam pemrosesan dan penyimpanan.

Pembuatan keputusan merupakan fungsi utama seorang manajer atau administrator. Kegiatan pembuatan keputusan meliputi pengidentifikasian masalah, pencarian alternatif penyelesaian masalah, evaluasi dari alternatif-alternatif tersebut dan pemilihan alternatif keputusan yang baik. Kemampuan seorang manajer dalam membuat keputusan dapat ditingkatkan apabila ia mengetahui dan menguasai teori dan teknik pembuatan keputusan. Dengan peningkatan kemampuan manajer dalam pembuatan keputusan diharapkan dapat ditingkatkan kualitas keputusan yang dibuatnya, dan hal ini tentu akan meningkatkan efesiensi kerja manajer yang bersangkutan.

II.1.1. Keuntungan Sistem Pendukung Keputusan

Dengan berbagai karakter khusus yang dimiliki Sistem Pendukung Keputusan, SPK (Sistem Pendukung Keputusan) dapat memberikan berbagai manfaat dan keuntungan. Keuntungan yang dapat diambil dari SPK adalah:

1. Mampu mendukung pencarian solusi dari masalah yang kompleks.
2. Respon cepat pada situasi yang tak diharapkan dalam kondisi yang berubah-ubah.
3. Mampu untuk menerapkan berbagai strategi yang berbeda pada konfigurasi berbeda secara cepat dan tepat.
4. Pandangan dan pembelajaran baru.
5. Mempasilisasi komunikasi.
6. Meningkatkan control manajement dan kinerja.
7. Menghemat biaya.
8. Keputusan lebih cepat.
9. Meningkatkan produktifitas analisis.

II.2.Konsep dasar *Multi Attribute Decision Making* (MADM)

MADM menyelesaikan masalah-masalah dalam ruang diskret. Oleh karena itu, pada MADM biasanya digunakan untuk melakukan penilaian atau seleksi terhadap beberapa alternatif dalam jumlah yang terbatas.

Menurut Rudolphi dalam Kusumadewi dkk. (2006, h. 72), Pada dasarnya, proses MADM dilakukan melalui 3 tahap, yaitu penyusunan komponen-komponen situasi, analisis, dan sintesis informasi. Pada tahap penyusunan komponen, komponen situasi, akan dibentuk tabel taksiran yang berisi identifikasi alternatif dan spesifikasi tujuan, kriteria dan atribut.

Multi Attribute Decision Making (MADM) adalah suatu metode yang digunakan untuk mencari alternatif otptimal dari sejumlah alternatif dengan kriteria tertentu. Tahap analisis dilakukan melalui 2 langkah. Pertama, mendatangkan taksiran dari besaran yang potensial, kemungkinan dan ketidakpastian yang berhubungan dengan dampak-dampak yang mungkin pada setiap alternatif. Kedua, meliputi pemilihan dari preferensi pengambil keputusan untuk setiap nilai, dan ketidak pedulian terhadap resiko yang timbul. Pada langkah pertama, beberapa metode menggunakan fungsi distribusi $|p_j(x)|$ yang menyatakan probabilitas kumpulan atribut $|a_k|$ terhadap setiap alternatif $|A_i|$. Konsekuen juga dapat ditentukan secara langsung dari agregasi sederhana yang dilakukan pada informasi terbaik yang tersedia. Demikian pula, ada beberapa cara untuk menentukan preferensi pengambil keputusan pada setiap konsekuen yang dapat dilakukan pada langkah kedua. Metode yang paling sederhana adalah untuk menurunkan bobot atribut dan kriteria adalah dengan fungsi utilitas atau penjumlahan terbobot.

Dalam Kusumadewi dkk. (2006, h. 72), Zimermann menyatakan bahwa, secara umum, model *multi attribute decision making* dapat didefinisikan sebagai berikut:

Misalkan $A = \{a_i \mid i = 1, \ldots, n\}$ adalah himpunan alternatif keputusan dan $C = \{c_j \mid j = 1, \ldots, m\}$ adalah himpunan tujuan yang diharapkan, maka akan ditentukan alternatif x^0 yang memiliki derajat harapan tertinggi terhadap tujuan-tujuan yang relevan c_j.

Namun sebagian besar pendekatan MADM dilakukan melalui 2 langkah, yaitu : pertama, melakukan agregasi terhadap keputusan-keputusan yang tanggap terhadap semua tujuan pada setiap alternatif. Sedangkan yang kedua, melakukan perankingan alternatif-alternatif keputusan tersebut berdasarkan hasil agregasi keputusan.

Dengan demikian, bisa dikatakan bahwa, masalah *multi-attribute decision making* (MADM) adalah mengevaluasi m alternatif A_i (i=1,2,….,m) terhadap sekumpulan atribut atau kriteria C_j (j=1,2,….,n), dimana setiap atribut saling tidak bergantung satu dengan yang lainnya. Matriks keputusan setiap alternatif terhadap setiap atribut x, diberikan sebagai:

$$x = \begin{pmatrix} x_{11}x_{12}....x_{1n} \\ x_{21}x_{22}....x_{2n} \\ ..... \end{pmatrix}$$

Dimana x_{ij} merupakan rating kinerja alternatif ke-i terhadap atribut ke-j. Nilai bobot yang menunjukkan tingkat kepentingan relatif setiap atribut, diberikan sebagai, w :

$$w = \{ w_1, w_{2,} ...., w_n \}$$

Rating kinerja (x), dan nilai bobot (w) merupakan nilai utama yang merepresentasikan preferensi absolut dari pengambil keputusan/ masalah MADM diakhiri dengan proses perangkingan untuk mendapatkan alternatif terbaik yang diperoleh berdasarkan nilai keseluruhan preferensi yang diberikan (Yeh dalam Kusumadewi dkk. (2006, h. 73).

Beberapa metode yang dapat digunakan untuk menyelesaikan masalah MADM, antara lain sebagai berikut :

1.*Simple Additive Weighting Method* (SAW)

2.*Weighted Product Model* (WPM)

3.ELECTRE

4.*Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS)

5.*Analytic Hierarchy Process* (AHP)

II.3.Pengertian *Simple Additive weighting* (SAW)

Metode *Simple Additive Weighting* (SAW) sering juga dikenal dengan istilah penjumlahan terbobot, metode yang paling simpel dan masih banyak digunakan pada metode MADM, Konsep dasar metode SAW adalah mencari penjumlahan terbobot dari rating kinerja pada setiap alternatif pada semua atribut. Menurut (Fishburn, 1967) (MacCrimmon, 1968) dalam Kusumadewi dkk. (2006, h, 74).

Metode SAW membutuhkan proses normalisasi matriks keputusan (X) ke suatu skala yang dapat diperbandingkan dengan semua rating alternatif yang ada. Metode SAW mengharuskan pembuat keputusan menentukan bobot bagi setiap atribut. Skor total untuk alternatif diperoleh dengan menjumlahkan seluruh hasil perkalian antara rating (yang dapat dibandingkan lintas atribut) dan bobot tiap atribut. Rating tiap atribut haruslah bebas dimensi dalam arti telah melewati proses normalisasi matriks sebelumnya.

Pada metode SAW, ada kriteria yang dipersepsikan sebagai kriteria "benefit" dan "cost". Dimana kriteria "benefit" digunakan jika kriteria lebih baik ketika bernilai besar, sedangkan "cost" kriteria akan lebih baik ketika nilainya lebih kecil. Besar dan kecilnya nilai tersebut dilihat dari keterkaitannya dengan permasalahan yang diangkat.

II.3.1 Tahapan SAW (*Simple Additive Wiegthing*)

Adapun tahapan-tahapan SAW adalah sebagai berikut :

1. Menentukan kriteria-kriteria yang akan dijadikan acuan dalam pengambilan keputusan, yaitu C_i.
2. Menentukan rating kecocokan setiap alternatif pada setiap kriteria.
3. Membuat matriks keputusan berdasarkan kriteria (C_i).
4. Normalisasi matriks berdasarkan persamaan yang disesuaikan dengan jenis atribut (atribut keuntungan ataupun atribut biaya) sehingga diperoleh matriks ternormalisasi R.
5. Hasil akhir diperoleh dari proses perankingan yaitu penjumlahan dari perkalian matriks ternormalisasi R dengan vektor bobot sehingga diperoleh nilai terbesar yang dipilih sebagai alternatif terbaik (A_i) sebagai solusi.

Berikut formulasi yang digunakan melakukan normalisasi (B*enefit*) :

$$r_{ij} = \frac{x_{ij}}{Max \quad x_{ij}}$$

Berikut formulasi yang digunakan melakukan normalisasi (*Cost*) :

$$r_{ij} = \frac{Min_i \quad x_{ij}}{x_{ij}}$$

Keterangan :

r_{ij}= rating kinerja ternormalisasi dari alternatif A_i pada atribut C_j

i= 1,2,3,..., m

j= 1,2,3,...n

$Maxx_{ij}$= nilai maksimum dari setiap baris dan kolom

$Minx_{ij}$ = nilai minimum dari setiap baris dan kolom

x_{ij}= baris dan kolom dari matriks

Nilai preferensi untuk setiap alternatif (V_i) :

$$V_i = \sum_{j=1}^{n} w_j r_{ij}$$

Dimana :

V_i= Nilai akhir dari alternatif

w_j= Bobot yang telah ditentukan

r_{ij}= Normalisasi matriks

Nilai V *i* yang lebih besar mengindikasikan bahwa alternatif Ai lebih terpilih.

Tujuan analisa sistem dalam pembangunan aplikasi sistem pendukung keputusan ini adalah untuk mendapatkan semua kebutuhan penggunadan sistem, yaitu perihal masukan dan keluaran yang harus disediakan oleh sistem, serta yangdiinginkan oleh pengguna. Proses tersebut akan menjadi masukan bagi proses perancangan sistem secara keseluruhan.

II.3.2. Langkah Penyelesaian Metode SAW

Untuk lebih jelasnya dijabarkan dalam rumus berikut ini:

$V1 = (W1)(r_{11})+(W2)(r_{12})+(W3)(r_{13})+(W4)(r_{14})+(W5)(r_{15})$

$V2 = (W1)(r_{21})+(W2)(r_{22})+(W3)(r_{23})+(W4)(r_{24})+(W5)(r_{25})$

$V3 = (W1)(r_{31})+(W2)(r_{32})+(W3)(r_{33})+(W4)(r_{34})+(W5)(r_{35})$

Keterangan:

V1= Rangking untuk alternative A1

V2= Rangking untuk alternative A2

V3= Rangking untuk alternative A3

II.4. Pengertian *Unified Modeling Language* (UML)

Saat ini piranti lunak luas dan besar lingkupnya, sehingga tidak bisa dibuat asal-asalan. Kesuksesan suatu pemodelan piranti lunak ditentukan oleh tiga unsur, ketiga unsur itu adalah pemodelan (*notation)*, proses (*process)* dan tool digunakan. Memahami notasi pemodelan tanpa mengetahui cara pemakaian yang sebenarnya akan membuat proyek gagal. Dan pemahaman terhadap pemodelan dan proses disempurnakan dengan penggunaan tool yang tepat.

Menurut Yuni Sugiarti (2013, h. 34) *Unified Modeling Language* adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep

dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa- bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya : Grady Booch OOD (Object-Oriented Design), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah : metodologi booch [1], metodologi coad [2], metodologi OOSE [3], metodologi OMT [4], metodologi *shlaer-mellor* [5], metodologi wirfs-brock [6], dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan.

II.4.1. Konsepsi Dasar UML

Dari berbagai penjelasan rumit yang terdapat di dokumen dan buku-buku UML. Abstraksi konsep dasar UML yang terdiri dari *structural classification*, *dynamic behavior*, dan *model management*, bisa kita pahami dengan mudah apabila kita melihat gambar diatas dari *Diagrams*. *Main concepts* bisa kita pandang sebagai iterm yang akan muncul pada saat kita membuatdiagram. Dan view adalah kategori dari diagaram tersebut. (ikc.dinus.ac.id)

Lalu darimana kita mulai untuk menguasai UML, sebenarnya cukup dua hal yang harus kita perhatikan:

1. Menguasai pembuatan diagram UML.

2. Menguasai langkah-langkah dalam analisa dan pengembangan dengan UML.

Tulisan ini pada intinya akan mengupas kedua hal tersebut.

Seperti juga tercantum pada gambar diatas UML mendefinisikan diagram-diagram sebagai berikut:

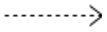
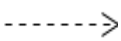
1.Use Case Diagram

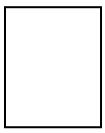
Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar data karyawan, dan sebagainya.

Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem,

mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Tabel II.2. Diagram Use Case

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
4		*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.

6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi

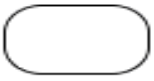
(Sumber : Yuni Sugiarti, 2013 : 34)

2.*Activity* Diagram

Activity diagram menyediakan analis dengan kemampuan untuk memodelkan proses dalam suatu sistem informasi. *Activity* diagram dapat digunakan untuk alur kerja model, use case individual, atau logika keputusan yang terkandung dalam metode individual3. *Activity* diagram juga menyediakan pendekatan untuk proses pemodelan paralel. *Activity* diagram lebih lanjut .

Pada dasarnya, diagram aktifitas canggih dan merupakan diagram aliran data yang terbaru. Secara teknis, diagram aktivitas menggabungkan ide-ide proses pemodelan dengan teknik yang berbeda termasuk model acara, statecharts, dan Petri Nets.

Tabel II.3. Simbol *Activity Diagram*

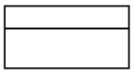
NO	GAMBAR	NAMA	KETERANGAN
1		*Actifity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4		*Actifity Final Node*	Bagaimana objek dibentuk dan dihancurkan
5		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

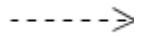
(Sumber : Martin Fowler, 2005 : 163)

3. *Class* Diagram

Class Diagram menurut Munawar (2005 : 28) merupakan himpunan dari objek-objek yang sejenis. Sebuah objek memiliki keadaan sesaat (*state*) dan perilaku (*behavior*). State sebuah objek adalah kondisi objek tersebut yang dinyatakan dalam atribute/properties. Sedangkan perilaku suatu objek mendefenisikan bagaimana sebuah objek bertindak/berinteraksi dan memberikan reaksi.

Tabel II.4. Simbol *Class Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*Generalization*	Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
2		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor
5		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.

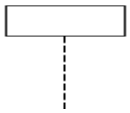
6		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri
7		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya

(Sumber : Munawar, 2005 : 28)

4. *Sequence* Diagram

Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya *sequence* diagram adalah gambaran tahap demi tahap yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case* diagram.

Tabel II.5. Simbol *Sequence Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*LifeLine*	Objek *entity*, antarmuka yang saling berinteraksi.
2		*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi

3		*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi

(Sumber : *Munawar, 2005 : 187*)

II.4.2. Tujuan UML *(*Unified *Modeling Language)*

Adapun tujuan dari penggunaan UML *(Unified Modeling Languarge)*

1. Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemograman dan proses rekayasa.
2. Menyatukan praktek – praktek terbaik yang dapat terdalam pemodelan.
3. Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
4. UML bisa juga berfungsi sebagai sebuah *(blue print)* cetak biru karena sangat lengkap dan *detail.* Dengan cetak biru ini maka akan bisa diketahui informasi secara *detail*dan *coding* program atau bahkan membaca program dan menginterpretasikan kembali kedalam bentuk diagram (*reserve engineering*).

II.5. Pengertian Basis Data (*Database*)

Database (Basis data) merupakan suatu data yang terintegrasi, diorganisasikan, dan disimpan dalam suatu cara yang baik tergantung pada aspek desain dan aspek realnya.

Menurut Wahana Komputer dalam buku belajar *MSQL Database Server* (2010, h. 1) menyatakan bahwa "*database* adalah sebuah struktur yang umumnya terbagi ke dalam 2 hal, yaitu sebuah *database flat* dan sebuah *database resional*. *Database Resional* lebih dipahami dari pada *database flat* karena *database* yang sederhana serta mudah dilakukan operasi data".

Menurut Sutabri (2010, h. 161) menyatakan bahwa "basis data merupakan suatu kumpulan data terhubung (interrelated data) yang disimpan secara bersama-sama pada suatu media, tanpa mengatap satu sama lain tidak perlu suatu kerangkapan data (*controlled redundancy)* dengan cara tertentu sehingga mudah digunakan atau ditampilkan kembali".

II.5.1.Model Data

Model data merupakan sejumlah konsep yang digunakan untuk membuat deskripsi struktur basis data, dengan deskripsi struktur basis data dapat ditemukan jenis data, hubungan dan konstrain data yang harus ditangani. Kebanyakan model data juga membuat spesifikasi untuk operasi dasar dalam pengaksesan dan pembaharuan data pada basis. Model data dapat duikelompokkan berdasarkan konsep pembuatan deskripsi struktur basis data, yaitu model data konseptual dan model data fisikal.

1.Model data konseptual

Menyajikan tentang bagaimana pemakai basis data memandang atau memperlakukan data. Dalam model data konseptual digunakan konsep entity, atribut dan hubungan.

2.Model data fisikal

Merupakan konsep bagaimana deskripsi detail data disimpan dalam komputer dengan menyajikan informasi tentang format rekaman, urutan rekaman dan jalur pengaksesan data. Informasi jalur pengaksesan dan merupakan struktur yang dapat membuat pencarian rekaman data lebih efisien.

II.5.2. *Hierarki* Data Dalam *Database*

Database merupakan salah satu komponen yang penting dalam sistem informasi, karena merupakan basis (dasar) dalam menyediakan informasi bagi para pemakai.

1.Bit

Bit merupakan bagian data terkecil, dapat berupa karakter numerik, huruf maupun karakter-karakter khusus yang membentuk suatu item data, dimana kumpulan karakter membentuk satu *field.*

2.Byte

Byte merupakan sistem binner yang terdiri atas dua macam nilai, yaitu 0 dan 1. Sistem binner merupakan dasar yang dapat digunakan untuk komunikasi antara manusia dan mesin.

3.Field

Suatu *field* menggambarkan suatu atribut *record* yang menunjukan suatu item dari data seperti nama, alamat, dimana kumpulan *field* membentuk suatu *record.*

4. *Record*

Suatu *record* menggambarkan satu kesatuan data yang sejenis, dimana kumpulan dari *file-file* membentuk *database.*

5. *File*

Suatu *file* menggambarkan kumpulan dari beberapa *record* yang dapat menampung data-data.

6. *Database*

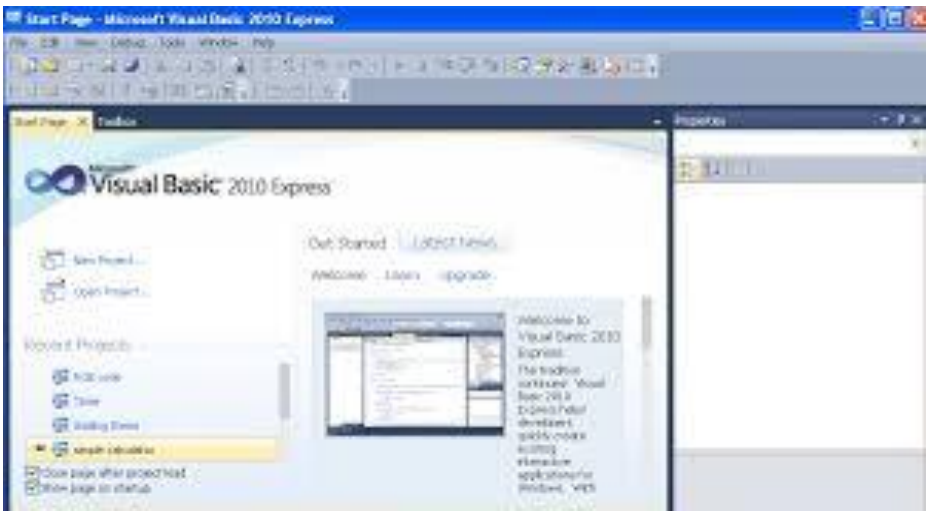
Suatu *database* menggambar data yang saling berhubungan antara satu dengan yang lainnya.

II.6.Pengertian *Visual Basic*

Visual Basic.Net hanya merupakan salah satu aplikasi bahasa pemrograman yang dapat digunakan untuk membuat aplikasi komputer. Dalam hal ini kita sebaiknya memahami bahwa bahasa pemrogram hanya merupakan salah satu aspek dalam suatu aplikasi *Windows.*

Menurut Adi Nugroho (2010, h.50-63) *Visual basic.Net* adalah merupakan bahasa pemrograman aras tinggi (*hight level programing language).* Salah satu bahasa pemrograman yang dapat kita gunakan untuk membuat aplikasi komputer.

Pada dasarnya tampilan baru ini memudahkan menggunakan microsoft visual basic seperti di bawah ini :



Gambar II.2. Visual Studio 2010

Melalui lembar ini dapat membuat aplikasi visual basic yang diperlukan. Sebelum membuat aplikasi baru, ada dua buah istilah yang diperlu diketahui dalam visual basic 2010 :

1. Project untuk membukanya, pembaca cukup mengklik namanya. Dalam jendela kita juga membuat proyek yang baru denga mengklik link create yang ada dalam bagian bawah jendela project ini.
2. Suliton Explorer adalah jendela Suliton Explorer berkaitan dengan pengelolaan proyek perograman VB.NET yang sedang kita lakukan.

II.7. Pengenalan MySQL Server Management Studio

MySQL merupakan sistem manajemen database SQL yang bersifat Open Souce dan paling popular saat ini. Sistem database MySQL mendukung beberapa filtur seperti multithreaded, multi-user dan SQL database managemen sistem (DBMS). Database ini dbuat untuk keperluan sistem database yang cepat , handal dan mudah digunakan.

Menurut Arief (2011 d: 152) “ MySQL adalah salah jenis database server yang terkenal dan banyak digunakan untuk membangun aplikasi web yang menggunakan database sebagai sumber dan pengolahan datanya”. Kepopuleran MySQL antara lain karena MySQL menggunakan SQL sebagai bahasa dasar untuk mengakses databasenya sehingga mudah untuk digunakan, kinerja query cepat, dan mencukupi untuk kebutuhan database perusahaan-perusahaan yang berskala kecil sampai menengah, MySQL juga bersifat open source (tidak berbayar) .