

BAB II

LANDASAN TEORI

2.1 Sistem

Sistem adalah Suatu jaringan kerja dari prosedur – prosedur yang saling berhubungan, berkumpul bersama – sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu. (Andri Kristanto. 2008:1)

Dalam bidang sistem informasi, sistem diartikan sebagai sekelompok komponen yang saling berhubungan, bekerja sama untuk mencapai tujuan bersama dengan menerima *input* serta menghasilkan *output* dalam proses transformasi yang teratur. Apabila suatu komponen tidak memberikan kontribusi terhadap sistem untuk mencapai tujuan, tentu saja komponen tersebut bukan bagian dari sistem. (Agus, 2009).

- Karakteristik Sistem

Suatu sistem mempunyai beberapa karakteristik atau sifat-sifat tertentu. Menurut Kusriani dan Andri Kuniyo (2007:6) karakteristik sistem tersebut sebagai berikut :

1. Suatu sistem mempunyai komponen system
2. Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang saling bekerja sama membentuk suatu komponen sistem atau bagian-bagian sistem.
3. Suatu sistem mempunyai batas sistem (*boundary*).
Boundary merupakan daerah yang membatasi suatu sistem dengan sistem yang lain atau dengan lingkungan kerjanya.
4. Suatu sistem mempunyai sub system
Yaitu bagian-bagian dari sistem yang beraktifitas dan berinteraksi satu sama lain untuk mencapai tujuan dan sasarannya masing-masing.

5. Suatu sistem mempunyai lingkungan luar (*environment*).

Yaitu suatu sistem yang ada diluar batas sistem yang dipengaruhi oleh operasi sistem.

6. Suatu sistem mempunyai penghubung (*interface*).

Media penghubung antara sub sistem dengan sub sistem lain. Adanya penghubung ini memungkinkan berbagai sumber daya yang mengalir dari suatu sub sistem ke sub sistem lainnya.

7. Suatu sistem memiliki masukan sistem (*input*).

Energi yang masuk kedalam sistem berupa perawatan dan sinyal. Masukan perawatan adalah energi yang dimasukkan supaya sistem tersebut dapat berinteraksi.

8. Suatu sistem memiliki keluaran sistem (*output*).

Hasil energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan

9. Suatu sistem memiliki pengolahan sistem (*proccess*).

Suatu sistem dapat mempunyai suatu bagian pengolah yang akan mengubah masukan menjadi keluaran.

10. Suatu sistem memiliki sasaran sistem (*object*).

Tujuan yang ingin dicapai oleh sistem, akan berhasil apabila mengenai tujuan atau sasaran.

2.2 Informasi

Menurut Mc Leod (2007:9), informasi adalah data yang telah diproses atau data yang sudah lebih memiliki arti bagi kebutuhan penggunanya.

Dan Menurut Andri Kristanto (2008:10), Data yang masih merupakan bahan mentah apabila tidak diolah maka data tersebut tidak berguna. Data tersebut akan berguna dan menghasilkan suatu informasi apabila diolah melalui suatu model. Model yang digunakan untuk mengolah data tersebut disebut dengan model pengolahan data atau lebih dikenal dengan nama siklus pengolahan data.

Berdasarkan dari kedua definisi di atas maka penulis menyimpulkan bahwa informasi adalah data yang telah dikelompokkan kemudian diolah menjadi suatu yang bernilai dari sebelumnya, dan dipahami untuk memutuskan suatu keputusan.

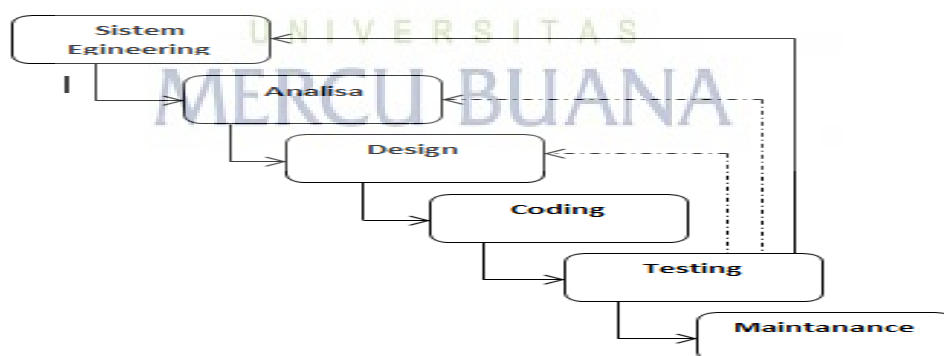
- **Definisi Sistem Informasi**

Menurut Andri Kristanto (2008 : 10). Data yang masih merupakan bahan mentah apabila tidak diolah maka data tersebut tidak berguna. Data tersebut akan berguna dan menghasilkan suatu informasi apabila diolah melalui suatu model. Model yang digunakan untuk mengolah data tersebut disebut dengan model pengolahan data atau lebih dikenal dengan nama siklus pengolahan data.

Dan menurut Jack Febrian (2007:238), sistem Informasi adalah sistem yang dapat menghasilkan informasi yang berguna. Untuk melakukan siklus pengolahan data atau yang disebut juga siklus sistem informasi diperlukan 3 buah komponen utama, yaitu komponen input, komponen model dan komponen output.

Berdasarkan dari kedua definisi tersebut maka penulis menyimpulkan bahwa sistem informasi adalah kumpulan dari data-data yang dikelompokkan kemudian diolah dan disajikan informasi yang berguna bagi tujuan yang sudah ditetapkan.

2.3 Metode Prototype



Gambar 2.1 Model Prototipe

1. Planning

Merupakan proses pokok pada pengertian mengapa sebuah sistem informasi akan dibangun dan faktor bagaimana tim proyek akan membangun.

2. Analysis

Suatu proses pengumpulan kebutuhan software untuk mengerti sifat-sifat program yang dibentuk *software engineering*, atau analis harus mengerti fungsi software yang diinginkan, performance dan interface terhadap elemen lainnya. Hasil dari analisis ini di dokumentasikan dan di *review* atau dibahas atau ditinjau bersama-sama customer.

3. Design

Proses desain menerjemahkan kebutuhan ke dalam representasi software yang dapat diukur kualitasnya sebelum mulai *coding*. Hasil dari desain ini didokumentasikan dan menjadi bagian dari konfigurasi software.

4. Implementasi

Pada tahap ini dilakukan kerja untuk membangun perangkat lunak berdasarkan analisa dan pemodelan yang telah dilakukan. Sehingga hasil dari tahap ini adalah basis data dan *source code* perangkat lunak. Selama tahap ini, desain perangkat lunak disadari sebagai sebuah program lengkap atau unit program. Desain yang telah disetujui, diubah dalam bentuk kode-kode program. Tahap ini, kode-kode program yang dihasilkan masih pada tahap modul-modul. Diakhir tahap ini, tiap modul di *testing* tanpa diintegrasikan. Perancangan perangkat lunak direalisasikan sebagai serangkaian program.

5. System

Merupakan fase siklus yang paling lama. Sistem di instal dan dipakai. Perbaikan mencakup koreksi dari berbagai eror, perbaikan dan implementasi unit sistem dan pelayanan sistem. Proses pemeliharaan perangkat lunak dan keseluruhan sistem bila terjadi kesalahan pada perangkat lunak dan keseluruhan sistem bila terjadi kesalahan pada program, atau terjadi perubahan lingkungan perangkat lunak dan juga bila terjadi perubahan requirements dan maintenance yang bersifat preventif untuk mengantisipasi keadaan yang tidak diinginkan.

2.4 UML (*Unified Modeling Language*)

UML merupakan salah satu alat bantu yang sangat handal di dunia analisa dan perancangan sistem informasi yang berorientasi objek (*Object Oriented*). Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru (*Blue Print*) atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain. [Munawar, 2005:17].

Diagram-Diagram UML

Unified Modeling Language (UML) terdiri dari 13 jenis diagram resmi [Martin Fowler, 2005] seperti pada Tabel 2.1

Tabel 2.2 Jenis-jenis Diagram UML

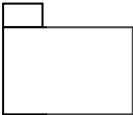
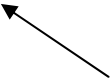
No	Diagram	Kegunaan
1	<i>Activity</i>	<i>Behavior</i> prosedural dan paralel
2	<i>Class</i>	<i>Class</i> , <i>fitur</i> , dan hubungan-hubungan
3	<i>Communication</i>	Interaksi antar objek; penekanan pada jalur
4	<i>Component</i>	Struktur dan koneksi komponen
5	<i>Composite structure</i>	Dekomposisi <i>runtime</i> sebuah class
6	<i>Deployment</i>	Pemindahan artifak ke node
7	<i>Interaction overview</i>	Campuran <i>sequence</i> dan <i>activity</i> diagram
8	<i>Object</i>	Contoh konfigurasi dari contoh-contoh
9	<i>Package</i>	Struktur hirarki <i>compile-time</i>
10	<i>Sequence</i>	Interaksi antar objek; penekanan pada sequence
11	<i>State machine</i>	Bagaimana <i>even</i> mengubah objek selama aktif
12	<i>Timing</i>	Interaksi antar objek; penekanan pada <i>timing</i>
13	<i>Use Case</i>	Bagaimana Pengguna berinteraksi dengan sistem

- **Use Case Diagram**

Use case adalah permodelan untuk melakukan (behavior) sistem informasi yang akan dibuat. Usecase mendeskripsikan sebuah interaksi antara satu atau lebih actor dengan sistem informasi yang akan dibuat. Secara kasar *use case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Rosa A.S – M. Salahudin [2011 : 130]

Tabel 2.3 Simbol *Use Case Diagram*

Shapes	Keterangan	Shapes	Keterangan
	<i>Package</i> (<i>paket</i>) merupakan mekanisme penelompokan yang digunakan menunjukan idak hanya pengelompokan logoakal dan fisikak, tetapi juga dapat menunjukan penglompokan dari unit-unit terdistribusi		<i>Note</i> (catatan) digunakan untuk pembuatan asumsi dan keputusan selama proses analisis dan desain.
	<i>Actor</i> menggambarkan orang, system atau external entitas/stakeholder yang menyediakan atau menerima informasi dari system dan juga aktor member input atau menerima informasi dari sistem		<i>Uses relationship</i> mengindikasikan sebuah use case menggunakan <i>functionally</i> dari use case lainnya.
	<i>Interface</i> menentukan operasi eksternal yang dapat dilakukan oleh sebuah <i>class</i> dan <i>component</i> dan tidak terdapat implementasi dari operasi tersebut.		<i>System boundary</i> digunakan untuk menggambarkan jangkauan system anda (scope of your system)
	<i>Extends relationship</i> mengekspresikan pilihan (optional) atau kondisional dari <i>behavior</i> untuk sebuah <i>use case</i> .		<i>Use case</i> dibuat berdasarkan keperluan actor, merupakan apa yang dikerjakan sistem, bukan bagaimana sistem mengerjakannya.

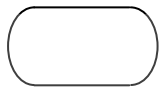
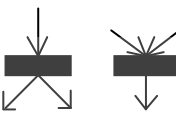
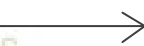
- **Activity Diagram**

Menurut John Alan Dennis dkk, [2005:167] Activity Diagram adalah activiti yang digunakan untuk menggunakan tingkah laku proses bisnis satu orang dari obyek. Dalam beberapa kebiasaan bisa dilihat sebagai diagram aliran data yang

sudah berpengalaman digunakan dalam *conjunction* dengan struktur analis. Tidak hanya itu, diagram activiti dapat digunakan untuk menggambarkan segala macam dari level menengah aliran kerja bahwa *involve* beberapa perbedaan *Use Case*, untuk menjelaskan *Use Case* perorangan.

Kesimpulannya untuk mengkhhususkan detail dari metode perorangan”. Berikut ini adalah simbol-simbol yang sering digunakan pada saat pembuatan *activity diagram* :

Tabel 2.4 Simbol-simbol *Activity Diagram*

Shapes	Keterangan	Shapes	Keterangan
	<i>Action</i> memperlihatkan bagaimana masing-masing kelas anatrmuka saling berinteraksi satu sama lain		<i>Transition (fork) & (join)</i> , satu aliran yang pada tahap tertentu berubah menjadi beberapa alira.
	<i>Activity / state</i> , keadaan dari system yang mencerminkan eksekusi dari suatu aksi		<i>Decision</i> , pilihan untuk mengambil keputusan.
	<i>Initial state / node</i> , bagaiman objek dibentuk atau diawali		<i>Control flow</i> , menunjukan kegiatan apa berikutnya setelah suatu kegiatan sebelumnya.
	<i>Activity final state / node</i> , bagaimana objek dibentuk atau diakhiri.		<i>Swimlane</i> , sebuah cara untuk mengelompokkan <i>activity</i> berdasarkan <i>actor</i> .

- ***Class Diagram***

Class diagram adalah diagram yang digunakan untuk menampilkan beberapa kelas serta paket-paket yang ada dalam sistem/perangkat lunak yang sedang kita gunakan. *Class diagram* memberi kita gambaran (diagram statis) tentang sistem/perangkat lunak dan relas-relasi yang ada didalamnya.

Definisi Class Diagram

Class adalah kumpulan objek-objek dengan dan yang mempunyai struktur umum, *behavior* umum, relasi umum, dan *semantic*/kata yang umum. *Class-class* ditentukan/ditemukan dengan cara memeriksa objek-objek dalam *sequence diagram* dan *collaboration diagram*. Sebuah *class* digambarkan seperti sebuah bujur sangkar dengan tiga bagian ruangan. *Class* sebaiknya diberi nama menggunakan kata benda sesuai dengan domain/bagian/kelompoknya.

Elemen-elemen

Elemen-elemen *class diagram* dalam pemodelan UML terdiri dari: *Class-class*, struktur *class*, sifat *class* (*class behavior*), perkumpulan/gabungan (*association*), pengumpulan/kesatuan (*agregation*), ketergantungan (*dependency*), relasi-relasi turunannya, keberagaman dan indikator navigasi, dan *role name* (peranan/tugas,nama).

- *Class*

Class adalah blok - blok pembangun pada pemrograman berorientasi obyek. Sebuah *class* digambarkan sebagai sebuah kotak yang terbagi atas 3 bagian. Bagian atas adalah bagian nama dari *class*. Bagian tengah mendefinisikan s/atribut *class*. Bagian akhir mendefinisikan *method-method* dari sebuah *class*.

Tabel 2.5 Contoh *class*

Nama <i>class</i>
+ atribut
+ atribut
+ atribut
+ operasi

+ operasi

- *Assosiation*

Asosiasi merupakan sebuah relationship paling umum antara 2 class dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 class. Garis ini bisa melambangkan tipe-tipe relationship dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah relationship. (Contoh: *One-to-one*, *one-to-many*, *many-to-many*).

1..n owned by 1

Gambar 2.2 contoh Assosiation

- *Composition*

Jika sebuah class tidak bisa berdiri sendiri dan harus merupakan bagian dari class yang lain, maka class tersebut memiliki relasi *Composition* terhadap class tempat dia bergantung tersebut. Sebuah relationship composition digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi/solid.



Gambar 2.3 contoh Composition

- *Dependency*

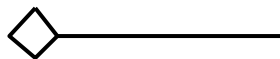
Kadangkala sebuah class menggunakan class yang lain. Hal ini disebut *dependency*. Umumnya penggunaan *dependency* digunakan untuk menunjukkan operasi pada suatu class yang menggunakan class yang lain. Sebuah dependency dilambangkan sebagai sebuah panah bertitik-titik.



Gambar 2.4 contoh dependency

- *Aggregation*

Mengindikasikan keseluruhan bagian *relationship* dan biasanya disebut sebagai relasi.



Gambar 2.5 contoh Aggregation

- ***Sequence Diagram***

Menurut John Alan Dennis dkk, [2005:244] *Sequence Diagram* adalah “Sequence Diagram mengilustrasikan obyek yang mengambil bagian dalam Use Case dan pesan yang terlewat diantara mereka untuk satu Use Case. Sebuah Sequence Diagram adalah model dinamis yang menunjukkan urutan eksplisit pesan yang lewat diantara objek dalam interaksi yang didefinisikan”. Diagram ini juga menunjukkan serangkaian pesan yang dipertukarkan oleh obyek-obyek yang melakukan suatu tugas atau aksi tertentu. Obyek-obyek tersebut kemudian diurutkan dari kiri ke kanan, aktor yang menginisiasi interaksi biasanya ditaruh di paling kiri dari diagram.

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.

2.5 Metode Pengujian

Black-box testing merupakan pendekatan pengujian dimana program dianggap sebagai suatu ‘black-box’ (‘kotak hitam’) Program test case berbasis spesifikasi *Test planning* dapat dimulai sejak awal proses pengembangan sistem.

Black-box testing berfokus pada persyaratan fungsional perangkat lunak. Metode ini memungkinkan *software developer* untuk mendapatkan serangkaian kondisi input yang mempergunakan semua persyaratan fungsional program. Black-

Box testing bukan alternatif *white-box testing*, namun merupakan pelengkap yang mampu mengungkap kesalahan, jika dibandingkan metode *white-box testing*.

Pengujian *black-box* berusaha menemukan kesalahan dalam kategori sebagai berikut:

1. Fungsi-fungsi yang tidak benar atau hilang.
2. Kesalahan *interface*.
3. Kesalahan dalam struktur data atau akses *database* eksternal.
4. Kesalahan kinerja.
5. inisialisasi dan kesalahan terminasi.

Pengujian program aplikasi penjualan berbasis web ini dilakukan dengan metode *Black Box Testing*. Metode *Black Box Testing* ini merupakan pengujian berdasarkan fungsi dari program. Tujuan dari metode *Black Box Testing* ini adalah untuk menemukan kesalahan fungsi pada program.

2.6 Basis Data (Database)

Andri Kriswanto (2007 : 73) “Basis data (*database*) adalah kumpulan dari data, yang dapat digambarkan sebagai aktivitas dari satu atau lebih organisasi yang berelasi”.

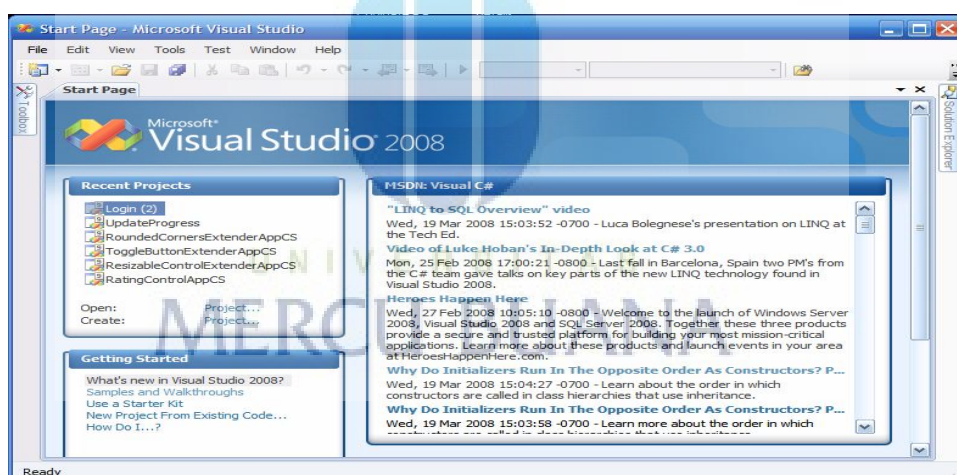
Sistem database dapat terbagi 4 komponen penting, yaitu :

1. *Data*, merupakan informasi yang disimpan dalam suatu struktur tertentu yang terintegrasi.
2. *Hardware*, merupakan perangkat keras berupa komputer dengan media penyimpanan sekunder yang digunakan untuk menyimpan data karena pada umumnya database memiliki ukuran yang besar.
3. *Software*, merupakan perangkat lunak yang digunakan untuk melakukan pengelolaan database. Perangkat lunak ini biasanya disebut dengan database management system (DBMS).

4. *User*, merupakan pengguna yang menggunakan data yang tersimpan dan dikelola. User dapat berupa seorang yang mengelola database tersebut dengan database administrator (DBA) bias juga end user yang mengambil hasil dari pengelolaan database melalui bahasa query.

2.7 Microsoft Visual Studio 2008 Profesional

Menurut Ketut Darmayuda (2010 :3) Pemrograman Visual Studio.NET 2008 adalah sebuah platform untuk membangun, menjalankan, dan meningkatkan generasi lanjut dari aplikasi terdistribusi. NET framework merupakan platform terbaru untuk pemrograman aplikasi windows dari Microsoft dalam upaya meningkatkan produktivitas pembuatan sebuah program aplikasi dan memungkinkan terbukanya peluang untuk menjalankan program pada multi sistem operasi serta dapat memperluas aplikasi client-server.



Gambar 2.6 Microsoft Visual Basic Net 2008

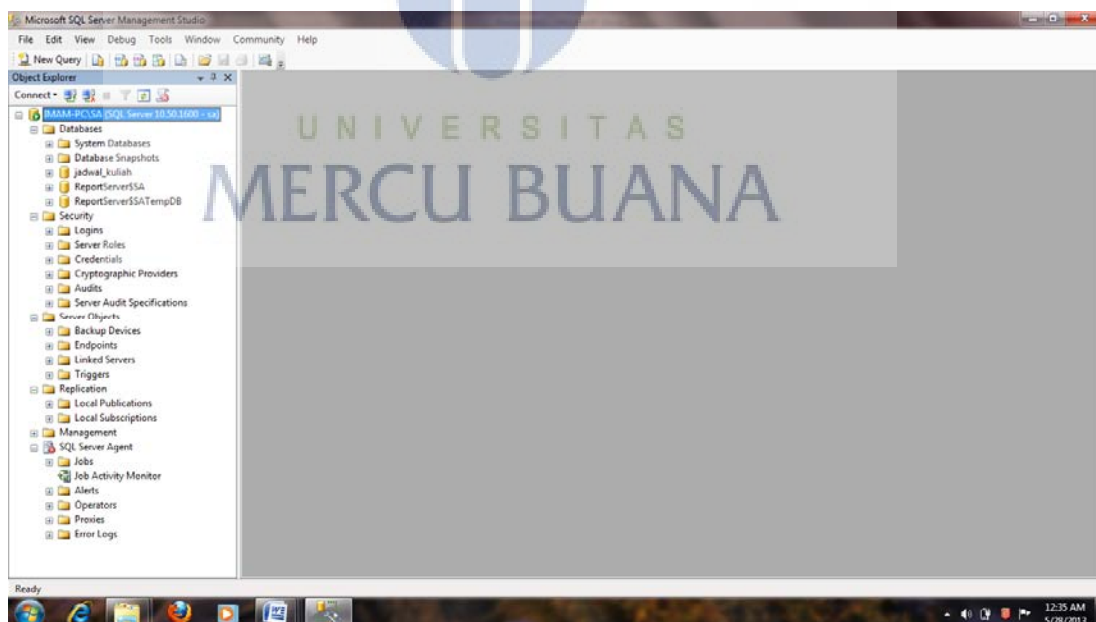
Microsoft Visual Basic.net 2008 merupakan bahasa pemrograman computer untuk membuat program dengan aplikasi GUI (Graphical User Interface) atau program yang memungkinkan pemakai computer berkomputerisasi dengan computer tersebut dengan menggunakan modus grafik atau gambar.

Beberapa manfaat atau kemampuan yang diberikan Visual Basis.Net 2008 adalah :

- a. Membuat program aplikasi berbasis windows.
- b. Membuat objek-objek pembantu program seperti control aktiveX, file help aplikasi internet dan lainnya.
- c. Saran akses data yang lebih cepat dan handal untuk membuat aplikasi database yang kemampuan tinggi.

2.9 SQL Server 2008

SQL Server 2008 adalah sebuah bahasa yang dipergunakan untuk mengakses data dalam basis data relasional. Bahasa ini secara de facto merupakan bahasa standar yang digunakan dalam manajemen basis data relasional. Saat ini hampir semua server basis data yang ada mendukung bahasa ini untuk melakukan manajemen datanya. Mengetahui perintah-perintah dasar sql merupakan modal awal untuk pengembangan database.



Gambar 2.7 Tampilan SQL Server 2008