BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Sistem adalah sebagai suatu kumpulan atau himpunan dari unsure, komponen, atau variabel yang terorganisir, saling berinteraksi, saling bergantung satu sama lain, dan terpadu (Tata Sutabri; 2005 : 2)

Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan (Jogiyanto, HM; 2005 : 2)

II.1.2. Klasifikasi Sistem

Sistem merupakan suatu bentuk integrasi antara satu komponen dengan komponen lain karena sistem memiliki sasaran yang berbeda setiap kasus yang terjadi yang ada di dalam sistem tersebut. Oleh karena itu sistem dapat diklasifikasikan dar beberapa sudut pandangannya antara lain :

1. Sistem abstrak dan sistem fisik

Sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik, misalnya sistem teologia, yaitu sistem yang berupa pemikiran hubungan antara manusia dengan tuhan, sedangkan sistem fisik merupakan sistem yang ada secara fisik, misalnya sistem komputer, sistem produksi, sistem penjualan, sistem administrasi personalia, dan lain sebagainya.

2. Sistem alamiah dan sistem buatan manusia.

Sistem alamiah adalah sistem yang terjadi melalui proses alam, tidak dibuat oleh manusia, misalnya sistem perputaran bumi, terjadinya siang malam, pergantian musim. Sedangkan sistem buatan manusia dengan mesin, merupakan melibatkan interaksi manusia dengan mesin, yang disebut "*human machine system*". Sistem informasi berbasis komputer merupakan contoh *human machine system* karena menyangkut penggunaan komputer yang berinteraksi dengan manusia.

3. Sistem deterministik dan sistem probabilistik.

Sistem yang beroparasi dengan tingkah laku yang dapat diprediksi disebut sistem deterministic. Sistem komputer adalah contoh dari sistem yang tingkah lakunya dapat dipastikan berdasarkan program-program komputer yang dijalankan. Sedangkan sistem bersifat probabilistik adalah sistem yang mana kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilistik.

4. Sistem terbuka dan sistem tertutup.

Sistem tertutup merupakan sistem yang tidak berhubungan dan tidak terpengaruh oleh lingkungan luarnya. Sistem ini bekerja secara otomatis tanpa campur tangan pihak luar. Sedangkan sistem terbuka adalah sistem yang berhubungan dan dipengaruhi oleh lingkungan luarnya. Sistem ini menerima masukan dan menghasilkan keluaran untuk subsistem lainya (Tata Sutabri, S. Kom, MM ; 2005 :13).\

Dari pendapat diatas diambil suatu kesimpulan tentang sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan.

II.2. Informasi

Informasi adalah data yang telah diklasifikasi atau diolah atau diinterpretasi untuk digunakan dalam proses pengambil keputusan (Tata Sutabri; 2005 :23)

Informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya (Jogiyanto HM; 2005 : 8)

Dari pendapat diatas diambil suatu kesimpulan tentang informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimannya.

II.3. Sistem Informasi

Sistem informasi adalah berupa suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan data transaksi harian yang mendukung operasi yang bersifat manajerial dengan kegiatan strategi suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan (Tata Sutabri; 2005 :42).

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 (Jogiyanto HM; 2005 : 11).

Dari pendapat diatas diambil suatu kesimpulan tentang sistem informasi 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.

II.4. Produksi

Hubungan pengendalian produksi terhadap keseluruhan organisasi manufaktur yang terutama ialah sebagai alat aliran informasi. Pengendalian informasi dalam produksi sendiri berkaitan erat dengan fungsi-fungsi di luarnya sehingga komponen di dalam pengendalian produksi memiliki interaksi aliran yang sangat rumit. Interaksi ini secara sederhana dapat dilihat pda gambar II.2. Harus diperhatikan bahwa keputusan dalam satu komponen-komponen lainnya. Sebagai contoh, satu cara untuk mencegah keterlambatan produksi karena kekurangan bahan adalah dengan meningkatkan persediaan bahan. Penjadwalan tetapi mmengakibatkan biaya persediaan jadi meningkar. Contoh lainnya adalah usaha untuk mengurangi keterlambatan produksi dengan cara menigkatkan waktu pengiriman yang akan mengakibatkan menurunnya permintaan konsumen. Hal tersebut memang membuat masalah penjadwalan menjadi mudah, tetapi juga akan menimbulkan biaya tambahan akibat ketidakpuasan konsumen.

Keputusan pengendalian produksi merupakan suatu sistem dan harus dilihat secara menyeluruh. Tindakan menekan waktu menganggur tenaga kerja dan mesin, menekan persediaan, atau menekan keterlambatan pengiriman tidaklah selalu bijaksana. Tujuan pengendalian produksi adalah tujuan keseluruhan

organisasi. Keputusan yang menyangkut penjualan, produksi, persediaan, dan keuangan lebih baik dicari tingkat optimalitasnya. (Hendra Kusuma; 2009 : 8)

Perencanaan kapasitas merupakan langkah kedua dalam rantai pengendalian produksi.Pada tahap ini direncanakan jumlah tenaga kerja yang akan direkrut, jumlah jam lembur yang dijadwalkan, dan jumlah persediaan sehingga permintaan konsumen dapat dipenuhi secara efisien. Jika tingkat kapasitas produksi dan persediaan produk terlalu tinggi maka perusahaan akan mengalami kesulitan aliran dana yang serius. Tanpa peramalan yang akurat atas keadaan di masa yang akan datang maka tidaklah mungkin untuk melakukan perencanaan kapasitas jangka panjang. (Hendra Kusuma; 2009 : 9)

II.5. *Entity Relationship Diagram*

Merupakan suatu model untuk menjelaskan hubungan antar dua dalam basis data berdasarkan suatu persepsi bahwa *real word* terdiri dari *object-object* dasar yang memmpunyai hubungan atau antar *object-object* tersebut. Relasi antar *object* dengan menggunakan symbol-simbol grafis tertentu.

Model *entity relationship* adalah suatu penyajian dengan menggunakan *entity* dan *relationship*. Diperkenalkan pada tahun 1976 oleh P.P. Chen.

II.5.1. Komponen-komponen yang terdapat didalam *Entity Relationship* Model.

1. *Entity*

a. Adalah sesuatu yang dapat dibedakan dalam dunia nyata dimana informasi yang berkaitan dengannya dikumpulkan.

b. *Entity* set adalah kumpulan *entity* yang sejenis.

c. Symbol yang digunakan untuk *entity* adalah persegi panjang.

d. *Entity* set dapat berupa :

1. *Entity* yang bersifat fisik, yaitu *entity* yang dapat dilihat. Contohnya : rumah, kendaraan, mahasiswa, dosen, dan lain-lain.
2. *Entity* yang bersifat konsep atau logic, yaitu *entity* yang tidak dapat dilihat. Contohnya : pekerjaan, perusahaan, rencana, mata kuliah, dan lain-lain.

e. Simbol yang digunakan untuk *entity* adalah persegi panjang.

Gambar II.1. *Entity*

Sumber : (Linda Marlinda ; 2004:17)

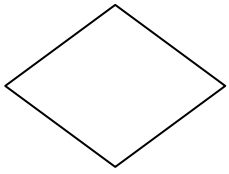
2. *Relationship*

a. Adalah hubungan yang terjadi antara satu atau lebih *entity*.

b. *Relationship* tidak mempunyai keberadaan fisik, kecuali yang mewarisi hubungan antara *entity* tersebut.

c. *Relationship* set adalah kumpulan relationship yang sejenis.

d. Simbol yang digunakan adalah bentuk belah ketupat, *diamond* atau *rectangle.*



Gambar II.2. *Relationship*

Sumber : (Linda Marlinda; 2004:18)

3. Attribute

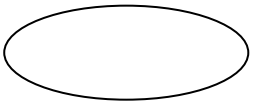
a. Adalah karakteristik dari *entity* atau *relationship* yang menyediakan penjelasan detail tentang atau *relationship* tersebut.

b. Attribute value (nilai atribute) adalah suatu data aktual atau informasi yang disimpan di suatu atribute di dalam suatu *entity* atau *relationship*.

c. Terdapat dua jenis atribute, yaitu :

1. *Indetifer (key)*, untuk menentukan suatu *entity* secara unik.
2. *Descriptor (nonkey attribute)*, untuk menentukan karakteristik dari suatu *entity* yang tidak unik.

d. Simbol yang digunakan adalah bentuk oval



Gambar II.3. *Attribute*

Sumber : (Linda Marlinda; 2004:18)

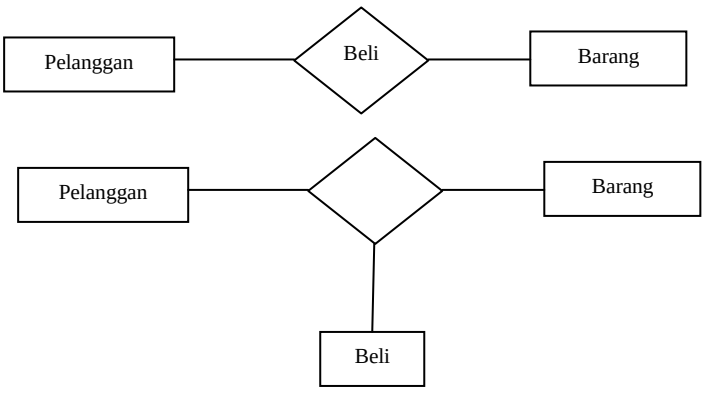
4. Indicator Tipe

a. *Indicator tipe associative object*

Berfungsi sebagai suatu objek dan suatu *relationship*

Contoh :

Menjadi :

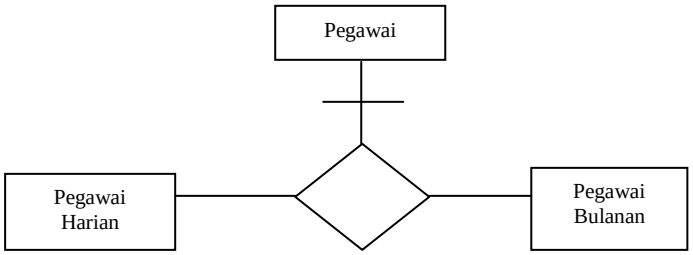


Gambar II.4. Indicator Tipe
Sumber : (Linda Marlinda; 2004:19)

b. Indicator tipe supertipe

Terdiri dari suatu object dan sub kategori atau lebih yang dihubungkan dengan dihubungkan dengan relationship yang tidak bernama (Linda Marlinda; 2004:17-19).

Contoh :



Gambar II.5. Indicator Tipe SuperTipe
Sumber : (Linda Marlinda; 2004:19)

II.5.2. Kamus Data

Kamus data (KD) atau data dictionary (DD) yang disebut juga dengan system dictionary data adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Dengan demikian KD, analisis sistem dapat mendefenisikan data yang mengalir di sistem dengan lengkap. KD dibuat pada pada tahap analisis sistem dan digunakan baik pada tahap analisis maupun pada tahap perancangan sistem. Pada tahap analisis, KD dapat digunakan sebagai alat komunikasi antara analisis sistem dengan pemakai sistem tentang data yang mengalir di sistem. Pada tahap perancangan sistem KD digunakan untuk merancang input, merancang laporan-laporan dan database. KD dibuat

berdasarkan arus data yang ada di DAD. Arus data di DAD sifatnya global, hanya ditunjukkan nama arus datanya saja (Prof. Dr. Jogiyanto HM, MBA, Akt; 2005 : 725).

Contoh :

1. Password = {**IDUser**} + {NamaUser} + {Password} + {Level} + {Status}.
2. Daftar Akun = {**KodeAkun**} + {Keterangan} + {Katagori}.
3. Master Kas Dan Bank = {**NoRekKasDanBank**} + {KasDanBank}
4. Tarif = {**IDBox**} + {Ukuran} + {Tarif}.
5. Pekerjaan = {**IDPekerjaan**} + {JenisPekerjaan}
6. Regu Kerja = {**IDRegu**} + {NamaRegu}.
7. Bongkar Muat = {**NoTransaksi**} + { Tanggal} + {Bulan} + {Tahun} + {JenisPekerjaan} + {NamaKapal} + {TanggalTiba} + {CaraPembayaran} + {NoRekaKasDanBank} + {KodeAkun} + {TotalPenerimaan} + {TotalUpahKerja}.
8. Kas Keluar = {**NoKasKeluar}** + {Tanggal} + {Bulan} + {Tahun} + {CaraPembayaran} + {NoRekKasDanBank} + {Uraian} + {KodeAkun} + {Jumlah}.
9. Detail = {NoTransaksi} + {IDBox} + {Tarif} + {Jumlah} + {SubTotal}.
10. Temp = {NoTransaksi} + {IDBox} + {Tarif} + {Jumlah} + {SubTotal}.
11. Jurnal = {Tanggal} + {Bulan} + {Tahun} + {Uraian} + {NoBukti} + {Debet} + {Kredit}.

II.5.3. Normalisasi

Normalisasi adalah proses pengkelompokkan attribute-attribute dan suatu relasi sehingga membentuk *Well- Structure Relation.* Normalisasi merupakan proses pengkelompokkan elemen data menjadi suatu tabel-tabel menunjukan entity dan relasinya. Normalisasi ditemukan pada tahun 1970 oleh E. F. CODD (Linda Marlinda, S. Kom; 2004 :115).

II.6 Well-Structure Relation

Well- Structure Relation adalah sebuah *relation* dengan jumlah kerangkapan datanya sedikit (*Minimum amount of redundancy*), serta memberikan kemungkinan bagi user untuk melakukan *Insert, Delete,* dan *Modify* terhadap baris-baris data pada *relation* tersebut, yang tidak berakibat terjadinya *Error* atau INKONSENTENSI DATA, yang disebabkan oleh operasi-operasi tersebut.

Contoh :

Terdapat sebuah *Relation Course* dengan ketentuan sebagai berikut :

a. Setiap mahasiswa hanya boleh mengambil satu mata kuliah saja.

b. Setiap mata kuliah mempunyai uang kuliah yang standar (tidak tergantung pada mahasiswa yang mengambil mata kuliah tersebut).

Tabel II.1. *Relation Course*

STUDENT-ID	KODE-MTK	BIAYA
92130	CS-200	75
92200	CS-300	100
99250	CS-300	75
92425	CS-400	150
92500	CS-300	100
92575	CS-500	50

Sumber : (Linda Marlinda; 2004 :115)

Relation Course diatas merupakan sebuah *relation* yang sederhana dan terdiri dari 3 kolom/attribute (Linda Marlinda; 2004 : 115).

a. Bentuk tidak normal (*Unnormalized Form*)

Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan mengikuti suatu format tertentu. Dapat saja data tidak lengkap atau terduplikasi. Data dikumpulkan apa adanya dengan saat menginput.

Contoh data :

Tabel II.2. Bentuk tidak normal (*Unnormalized Form*)

No_Siswa	Nama	Pa	Kelas 1	Kelas 2	Kelas 3
22890100	Shandy	Linda	1234	1543	1543
22890101	Susi	Riska	1234	1775	

Sumber :(Linda Marlinda; 2004 : 122)

Siswa yang punya nomor siswa, nama, dan pa mengikuti 3 mata pelajaran/kelas. Di sini ada perulangan kelas 3 kali ini bukan bentuk tidak 1 NF.

b. Bentuk Normal Kesatu (1 NF/ *Fisrt Normal Form*)

Suatu relasi 1 NF dan hanya jika sifat dan setiap relasi attributenya bersifat *atomic*. Atom adalah zat terkecil yang masih memiliki sifat induknya. Bila dipecah lagi maka ia tidak memiliki sifat induknya.

Ciri-ciri 1 NF :

1. Setiap data dibentuk kedalam *flat file* data terbentuk per satu *record* nilai dan field berupa "*atomic value*".
2. Tidak ada *set attribute* yang berulang atau bernilai ganda,
3. Tiap *field*
4. hanya satu pengertian.

Tabel II.3. Bentuk Normal Ke Satu (1 NF/ *Fisrt Normal Form*)

No_Siswa	Nama	Pa	Kelas 1
22890100	Shandy	Linda	1234
22890100	Shandy	Linda	1543
22890101	Susi	Riska	1234
22890101	Susi	Riska	1775
22890101	Susi	Riska	1543

Sumber : (Linda Marlinda; 2004 : 122)

c. Bentuk Normal Kedua (2 NF/ *Second Normal Form*)

Bentuk normal kedua mempunyai syarat yaitu bentuk data telah memenuhi kriteria bentuk normal kesatu. *Attribute* bukan kunci haruslah bergantung secara fungsi pada *primary key*. Jadi, untuk membentuk normal kedua haruslah sudah ditentukan kunci-kunci *field.* Kunci *field* haruslah unik dan dapat mewakili *attribute* lain yang menjadi anggotanya. Misal : dari contoh relasi siswa pada 1 NF terlihat bahwa *primary key* adalah nomor siswa. Nama siswa dan pa bergantung fungsi pada no_siswa, tetapi kode_kelas bukanlah fungsi dan siswa dipecah menjadi 2 relasi :

Relasi siswa :

Tabel II.4. Bentuk Normal Kedua Relasi Siswa (2 NF/ *Second Normal Form*)

No_Siswa	Nama	Pa
22890100	Shandy	Linda
22890101	Susi	Riska

Sumber : (Linda Marlinda; 2004 : 123)

Relasi ambi_Kelas

Tabel II.5. Bentuk Normal Kedua Relasi ambil_Kelas (2 NF/ *Second Normal Form*)

No_Siswa	Kode Kelas
22890100	1234
22890100	1543

22890101	1234
22890101	1775
22890101	1543

Sumber : (Linda Marlinda; 2004 : 123)

d. Bentuk Normal Ketiga (3 NF/*Third Normal Form*)

Untuk menjadi bentuk normal ketiga maka relasi dalam bentuk normal kedua dan semua *attribute* bukan primer tidak punya hubungan yang transistif. Dengan kata lain, setiap *attribute* bukan kunci haruslah bergantung hanya pada *primary key* dan *primary key* secara menyeluruh.

Contoh pada bentuk normal kedua diatas termasuk juga bentuk normal ketiga karena seluruh attribute yang ada bergantung penuh pada kunci primernya.

e. *Boyee-Cood Normal Form* (BCNF)

BNCF mempunyai paksaan lebih kuat dan bentuk normal ketiga untuk menjadi BCNF, relasi harus dalam bentuk normal kesatu dan setiap *attribute* harus bergantung fungsi pada *attribute superkey*.

Pada contoh di bawah ini terdapat relasi seminar dengan ketentuan sebagai berikut :

1. Kunci primer adalah no_siswa +seminar
2. Siswa bopleh mengambil satu atau dua seminar.
3. Setiap siswa dibimbing oleh salah satu di antara 2 instruktur seminar tersebut.
4. Setiap instruktur boleh hanya mengambil satu seminar saja.

Pada contoh ini no_siswa dan seminar menunjuk seorang instruktur :

Relasi seminar

Tabel II.6. *Boyee-Cood Normal Form* (BCNF)

No_Siswa	Seminar	Instruktur
22890100	2281	Si doel
22890101	2281	Pak tile
22890102	2291	Mandra
22890101	2291	Basuki
22890109	2291	Basuki

Sumber : (Linda Marlinda; 2004 : 124)

Bentuk relasi seminar adalah bentuk normal ketiga, tetapi tidak BCNF karena nomor seminar masih tergantung fungsi pada instruktur. Jika setiap instruktur dapat mengajar hanya pada satu seminar saja, maka seminar bergantung fungsi pada satu *attribute* bukan *superkey* seperti disyaratkan oleh BCNF. Maka relasi seminar haruslah dipecah menjadi dua yaitu :

Tabel II.7. *Boyee-Cood Normal Form* (BCNF

Relasi Pengajar

Instruktur	Seminar
Si doel	2281
Pak tile	2281
Mandra	2291
Basuki	2291

No_Siswa	Instruktur
22890100	Si doel
22890101	Pak tile
22890102	Mandra
22890101	Basuki
22890109	Basuki

Sumber : (Linda Marlinda; 2004 : 124)

f. Bentuk Normal Keempat (4NF)

Relasi R adalah bentuk 4 NF jika dan hanya jika relasi tersebut juga termasuk BCNF dan semua ketergantungan *multivaluei* adalah juga ketergantungan fungsional.

g. Bentuk Normal Kelima (5 NF)

Disebut juga PINF (*Projection Join Normal Form*) dan 4 NF dilakukan dengan menghilangakan ketergantungan *join* yang merupakan kunci kandidat (Linda Marlinda; 2004 : 122-125).

II.7. *Unified Modeling Language* (UML)

UML singkatan dari *Unified Modelling Langguage* yang berarti bahasa pemodelan standart. (Chonoles; 2003 : 6) mengatakan sebagai bahasa, berarti *UML* memiliki sintaks dan *semantic*. Ketika kita membuat model menggunakan konsep *UML* ada aturan –aturan yang harus diikuti. Bagaimana elemen pada model-model yang kita buat harus berhubungan satu dengan lainnya harus mengikuti standart yang ada. *UML* bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya. Ketika pelanggan memesan sesuatu dari sistem, bagaimana transaksinya? Bagaimana sistem mengatasi error yang terjadi? Bagaimana keamanan terhadap sistem yang ada kita buat? Dan sebagainya dapat dijawab dengan *UML*.

UML diaplikasikan untuk maksud tertentu, biasanya antara lain untuk :

1. Merancang perangkat lunak.
2. Sarana komunikasi antara perangkat lunak dengan bisnis.
3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
4. Mendokumentasikan sistem yang ada, proses-proses dan organisasinya.

UML telah diaplikasikan dalam investasi perbankan,lembaga kesehatan, departemen pertahanan, sistem terdistribusi, sistem pendukung alat kerja, retail, sales, dan supplier.

Blok pembangunan utama *UML* adalah diagram. Beberapa diagram ada yang rinci (jenis *timing diagram*) dan lainya ada yang bersifat umum (misalnya diagram kelas). Para pengembang sistem berorentasikan objek menggunakan bahasa model untuk menggambarkan, membangun dan mendokumentasikan sistem yang mereka rancang. *UML* memungkinkan para anggota team untuk bekerja sama dalam mengaplikasikan beragam sistem. Intinya, *UML* merupakan alat komunikasi yang konsisten dalam mensuport para pengembang sistem saat ini. Sebagai perancang sistem mau tidak mau pasti menjumpai *UML*, baik kita sendiri yang membuat sekedar membaca diagram *UML* buatan orang lain (Prabowo Pudji Widodo Dan Herlawati, UML; 2011 : 6-7).

II.7.1. *Diagram-Diagram UML*

Beberapa literatur menyebutkan bahwa *UML* menyediakan Sembilan jenis diagram, yang lain menyebutkan delapan karena ada beberapa yang digabung, misalnya diagram komunikasi, diagram urutan, dan diagram pewaktuan digabung menjadi diagram interaksi. Namun demikian model-model itu dapat dikelompokan berdasarkan sifatnya yaitu statis atau dinamis. Jenis diagram itu antara lain :

1. Diagram Kelas. Bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi, serta relasi-relasi diagram. Diagram ini umu dijumpai pada pemodelan sistem berorentasi objek. Meskipun bersifat statis, sering pula diagram kelas memuat kelas-kelas.

2. Diagram paket (*Package Diagram*) bersifat statis. Diagram ini memperlihatkan kumpulan kelas-kelas merupakan bagian dari diagram komponen.
3. Diagram *Use Case* bersifat statis. Diagram ini memperlihatkan himpunan *use-case* dan aktor-aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna.
4. Diagram interaksi dan *Sequence* (urutan). Bersifat dinamis. Diagram urutan adal;ah diagram interaksi yang menekankan pada pengiriman pesan dalam waktu tertentu.
5. Diagram komunikasi (*Communication Diagram*) bersifat dinamis. Diagram sebagai pengganti diagram kolaborasi *UML* yang menekankan organisasi *structural* dari objek-objek yang menerima serta mengirim pesan.
6. Diagram *Statechart* (*Statechart Diagram)* bersifat dinamis. Diagram status memperlihatkan keadaan-keadaan pada sistem, memuat status (*State),* transisi kejadian serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari antarmuka (*interface*), kelas, kolaborasi dan terutam penting pada pemodelan sistem-sistem yang reaktif.
7. Diagram aktivitas (*Activity Diagram*) bersifat dinamis. Diagram aktivitas adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi suatu sistem dan member tekanan pada aliran kendali antar objek.

8. Diagram komponen (*Component Diagram*) bersifat statis. Diagram komponen ini memperlihatkan organisasi serta kebergantungan sistem/perangkat lunak pada komponen-komponen yang telah ada sebelumnya. Diagram ini berhubungan diagram kelas dimana komponen dipetakan kedalam satu atau lebih kelas-kelas. Antarmuka-antarmuka serta kolaborasi-kolaborasi.
9. Diagram *Deployment* (*Deployment Diagram)* bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (*run time).* Memuat simpul-simpul berserta komponen-komponen yang ada di dalamnya. Diagram *Deployment* berhubungan erat dengan diagram komponen dimana diagram ini memuat satu atau lebih komponen-komponen. Diagram ini sangat berguna saat aplikasi kita berlaku sebagai aplikasi yang dijalankan pada banyak mesin (*distributed computing).*

Kesembilan diagram ini tidak mutlak harus digunakan dalam pengembangan perangkat lunak, semuanya dibuat sesuai dengan kebutuhan. Pada *UML* dimungkinkan kita menggunakan diagram-diagram lainya misalnya *Data Flow Diagram, Entity Relationship Diagram* dan sebagainya (Prabowo Pudji Widodo Dan Herlawati, UML; 2011 : 6-7; 2011 : 10-12).

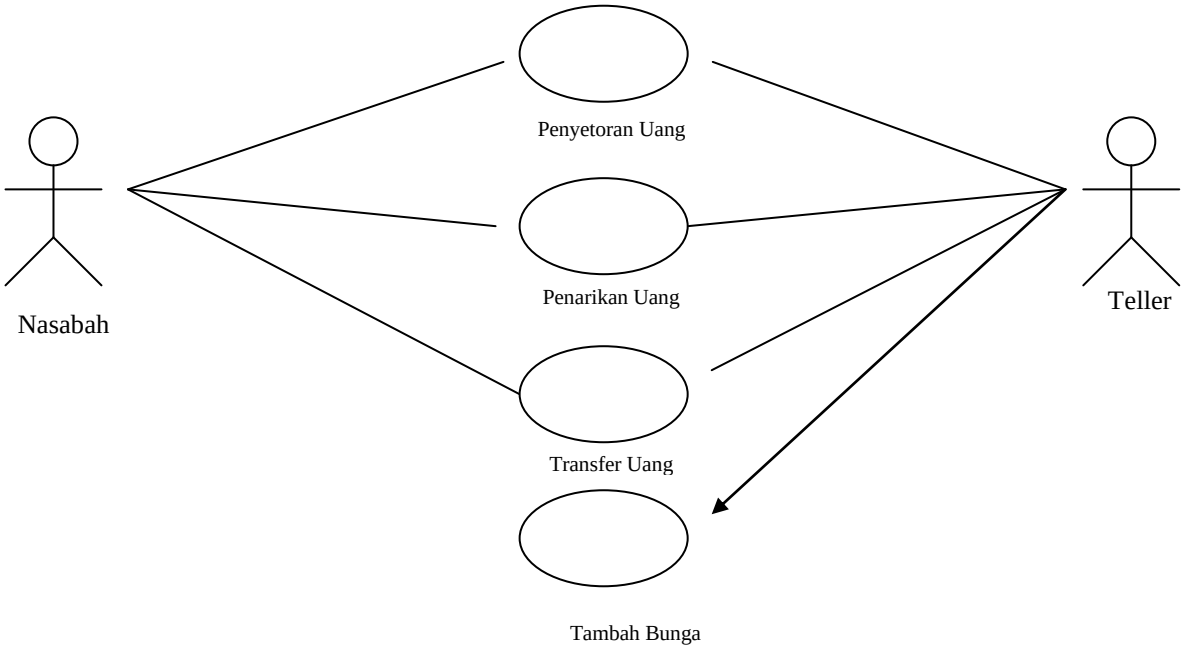
1. *Diagram Use Case (use case diagram)*

Use Case menggambarkan *external view* dari sistem yang akan kita buat modelnya. Menurut Pooley (2005:15) mengatakan bahwa model *use case* dapat dijabarkan dalam diagram, tetapi yang perlu diingat, diagram tidak identik dengan model karena model lebih luas dari diagram.

komponen pembentuk diagram *use case* adalah :

a. Aktor (*actor),* menggambarkan pihak-pihak yang berperan dalam sistem.
b. *Use Case*, aktivitas/ sarana yang disiapkan oleh bisnis/sistem.
c. Hubungan (*Link),* aktor mana saja yang terlibat dalam *use case* ini.

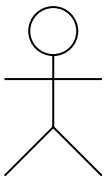
Gambar di bawah ini merupakan salah satu contoh bentuk diagram *use case* (Prabowo Pudji Widodo Dan Herlawati, UML; 2011 : 16- 17)



Gambar II.6. Diagram *Use Case*
Sumber : (Prabowo Pudji Widodo Dan Herlawati; 2011:17)

2. Aktor

Menurut Chonoles (2003 :17) menyarankan sebelum mebuat use case dan menentukan aktornya, agar mengidentifikasi siapa saja pihak yang terlibat dalam sistem kita. Pihak yang terlibat biasanya dinamakan *stakeholder*.

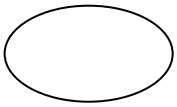


Gambar II.7. Aktor

Sumber : (Prabowo Pudji Widodo Dan Herlawati; 2011:17)

3. *Use Case*

Menurut Pilone (2005 : 21) *use case* menggambarkan fungsi tertentu dalam suatu sistem berupa komponen kejadian atau kelas. Sedangkan menurut Whitten (2004 : 258) mengartikan *use case* sebagai urutan langkah-langkah yang secara tindakan saling terkait (skenario) baik terotomatisasi maupun secara manual, untuk tujuan melengkapi satu tugas bisnis tunggal. *Use case* digambarkan dalam bentuk *ellips*/oval



Gambar II.8. Simbol *Use Case*

Sumber : (Prabowo Pudji Widodo Dan Herlawati; 2011:22)

Use case sangat menentukan karakteristik sistem yang kita buat, oleh karena itu Chonoles (2003:22-23) menawarkan cara untuk menghasilkan *use case* yang baik yakni :

a. Pilihlah nama yang baik

Use case adalah sebuah *behaviour* (prilaku), jadi seharusnya dalam frase kata kerja. Untuk membuat namanya lebih detil tambahkan kata benda mengindikasikan dampak aksinya terhadap suatu kelas objek. Oleh

karena itu diagram *use case* seharusnya berhubungan dengan diagram kelas.

b. Ilustrasikan perilaku dengan lengkap.

Use case dimulai dari inisiasi oleh aktor primer dan berakhir pada aktor dan menghasilkan tujuan. Jangan membuat *use case* kecuali anda mengetahui tujuannya. Sebagai contoh memilih tempat tidur (*King Size, Queen Size,* atau dobel) saat tamu memesan tidak dapat dijadikan *use case* karena merupakan bagian dari *use case* pemesanan kamar dan tidak dapat berdiri sendiri (tidak mungkin tamu memesan kamar tidur jenis king tapi tidak memesan kamar hotel).

c. Identifikasi perilaku dengan lengkap.

Untuk mencapai tujuan dan menghasilkan nilai tertentu dari aktor, use case harus lengkap. Ketika memberi nama pada *use case*, pilihlah frasa kata kerja yang implikasinya hingga selesai. Misalnya gunakan frasa *reserve a room* (pemesanan kamar) dan jangan *reserving a room* (memesan kamar) karena memesan menggambarkan perilaku yang belum selesai.

d. Menyediakan use case lawan (*inverse)*

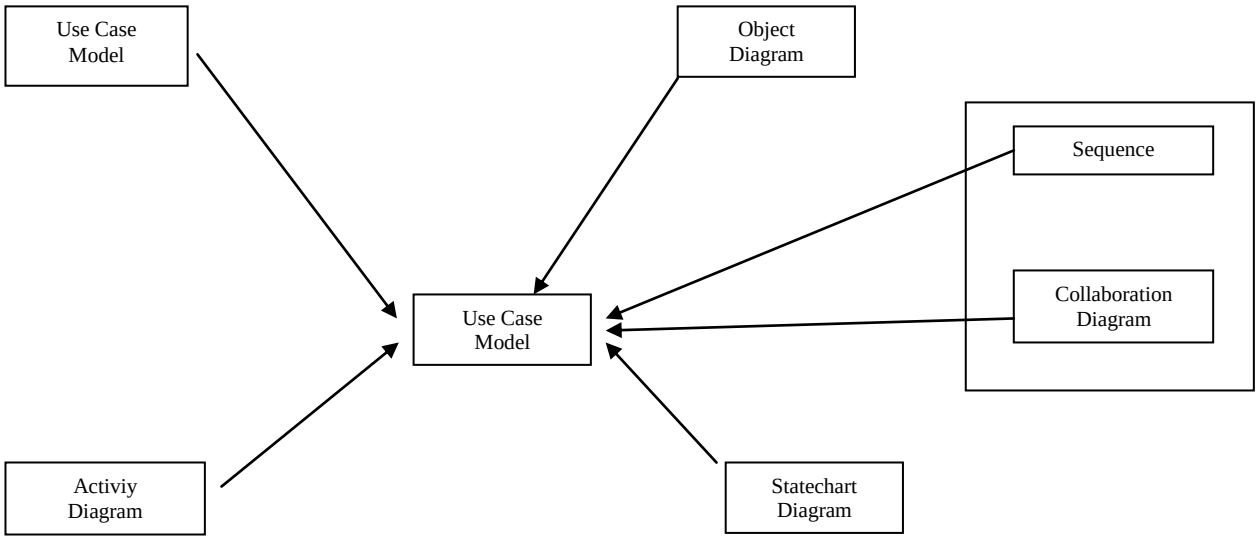
Kita biasanya membutukan *use case* yang membatalkan tujuan, misalnya pada u*se case* pemesanan kamar, dibutuhkan pula *use case* pembatalan pesanan kamar.

e. Batasi use case hingga satu perilaku saja.

Kadang kita cenderung membuat *use case* yang lebih dari satu tujuan aktivitas. Guna menghindari kerancuan, jagalah use case kita hanya fokus pada satu hal. Misalnya, penggunaan *use case check in* dan *check out* dalam satu *use case* menghasilkan ketidakfokusan, karena memiliki dua perilaku yang berbeda (Prabowo Pudji Widodo Dan Herlawati, UML; 2011 : 22-23)

4. Diagram Kelas (*Class Diagram*)

Diagram kelas adalah inti dari proses pemodelan objek. Baik *forward engineering* maupun *reverse engineering* memanfaatkan diagram ini *forward engineering* adalah proses perubahan model menjadi kode program sedangkan *reverse engineering* sebaliknya merubah kode program menjadi model (Prabowo Pudji Widodo Dan Herlawati; 2011 : 37).



Gambar II.9. Hubungan Diagram Kelas Dengan Diagram *UML* lainya

Sumber : (Prabowo Pudji Widodo Dan Herlawati; 2011:38)

5. Diagram Aktivitas (*Activity Diagram*)

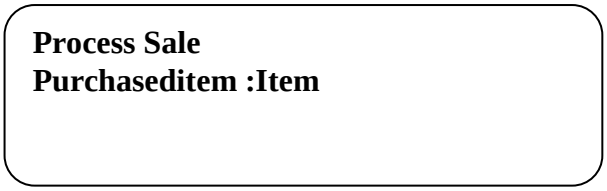
Diagram aktivitas lebih memfokuskan diri pada eksekusi dan alur sistemdari pada bagaimana sistem dirakit. Diagram ini tidak hanya memodelkan software melainkan memodelkan bisnis juga. Diagram aktivitas menunjukan aktivitas sistem dalam kumpulan aksi-aksi. Ketika digunakan dalam pemodelan *software*, diagram aktivitas merepresentasikan pemanggilan suatu fungsi tertentu misalnya *call*. Sedangkan bila digunakan dalam pemodelan bisnis, diagram ini menggambarkan aktivitas yang dipicu oleh kejadian-kejadian diluar seperti pemesanan atau kejadian-kejadian internal misalnya penggajian tiap jumat sore (Prabowo Pudji Widodo Dan Herlawati;2011 : 143-145).

Aktivitas merupakan kumpulan aksi-aksi. Aksi-aksi nelakukan langka sekali saja tidak boleh dipecah menjadi beberapa langkah-langkah lagi. Contoh aksinya yaitu :

a. Fungsi Matematika
b. Pemanggilan Perilaku
c. Pemrosesan Data

Ketika kita menggunakan diagram aktivitas untuk memodelkan perilaku suatu *classfier* dikatakan kontek dari aktivitas. Aktivitas dapat mengakses atribut dan operasi *classfier*, tiap objek yang terhubung dan parameter-parameter jika aktivitas memiliki hubungan dengan perilaku. Ketika digunakan dengan model proses bisnis, informasi itu biasanya disebut *process-relevant data*. Aktivitas diharapkan dapat digunakan ulang dalam suatu aplikasi, sedangkan aksi biasanya *specific* dan digunakan hanya untuk aktivitas tertentu.

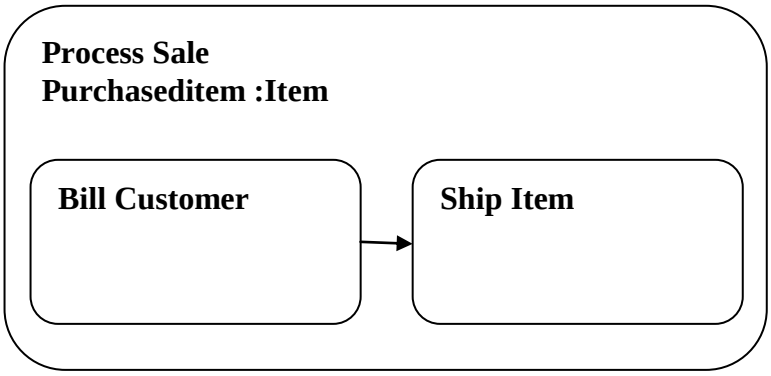
Aktivitas digambarkan dengan persegi panjang tumpul. Namanya ditulis di kiri atas. Parameter yang terlibat dalam aktivitas ditulis dibawahnya.



Gambar II.10. Aktivitas serderhana tanpa rincian

Sumber : (Prabowo Pudji Widodo Dan Herlawati; 2011:145)

Detail aktivitas dapat dimasukan di dalam kotak. Aksi diperlihatkan dengan symbol yang sama dengan aktivitas dan namanya diletakkan didalam persegi panjang.



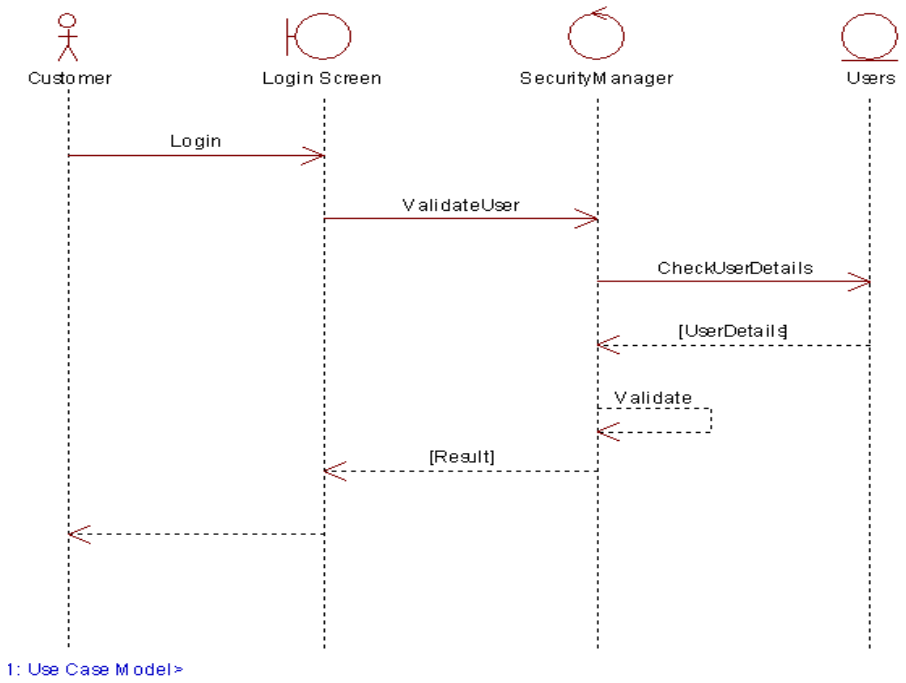
Gambar II.11. Aktivitas dengan detail rincian

Sumber : (Prabowo Pudji Widodo Dan Herlawati; 2011:145)

6. *Squence Diagram*

Menurut Douglas (2004 : 174) menyebutkan ada tiga diagram primer UML dalam memodelkan scenario interaksi, yaitu diagram urutan (*sequence diagram)*, diagram waktu (*timing diagram)* dan diagram komunikasi (*communication diagram).*

Menurut Pilone (2005 : 174) menyatakan bahwa diagram yang paling banyak dipakai adalah diagram urutan. Gambar II.14. memperlihatkan contoh diagram urutan dengan notasi-notasinya yang akan dijelaskan nantinya (Prabowo Pudji Widodo Dan Herlawati; 2011 : 174-175).



Gambar II.12. Diagram Urutan

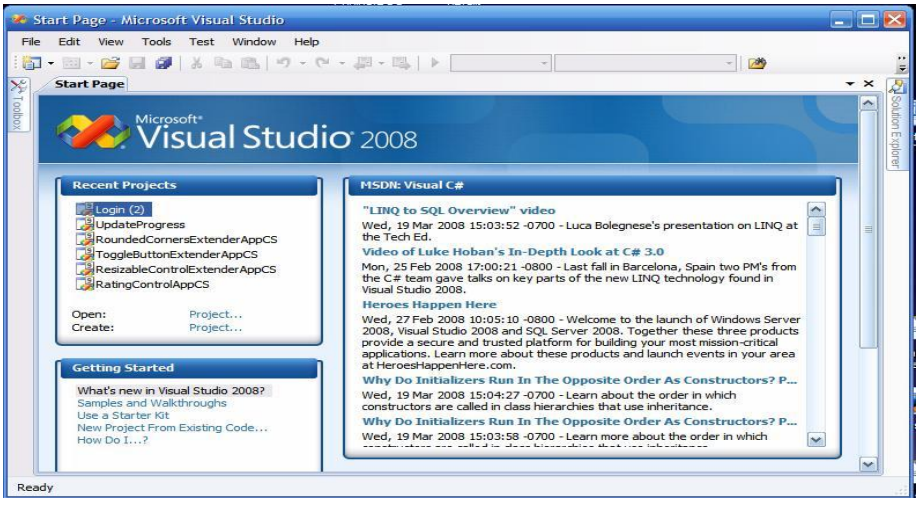
Sumber : (Prabowo Pudji Widodo Dan Herlawati; 2011:17)

II.8. Bahasa Pemrograman Visual Studio. Net

Microsoft Visual Studio 2008 merupakan kelanjutan dari *Microsoft Visual Studio* sebelumnya, yaitu *Visual Studio. Net* 2003 yang diproduksi oleh Microsoft.

Pada bulan Februari 2002 *Microsoft* memproduksi teknologi. *Net Framework* versi 1.0, teknologi. *Net* ini didasarkan atas susunan berupa *Net Framework*, sehingga setiap produk baru yang terkait dengan teknologi. Net akan selalu berkembang mengikuti perkembangan. *Net Frameworknya*. Pada perkembangannya nantinya mungkin untuk membuat program dengan teknologi. *Net* memungkinkan para pengembang perangkat lunak akan dapat menggunakan lintas sistem operasi, yaitu dapat dikembangkan di sistem operasi windows juga dapat dijalankan pada sistem operasi lain, misalkan pada sistem operasi *Linux*, seperti yang telah dilakukan pada pemograman *Java* oleh *Sun Microsystem*. Pada saat ini perusahaan-perusahaan sudah banyak meng*update* aplikasi lama yang dibuat *Microsoft Visual Basic* 6.0 ke teknologi. *Net* karena kelebihan-kelebihan yang ditawarkan, terutama memungkinkan pengembang perngkat lunak secara cepat mampu membuat program *robust*, serta berbasiskan integrasi ke internet yang dikenal dengan *XML Web Service* (Ketut Darmayuda ; 2009 : 1)

Untuk melihat tampilan visual studio 2008 dapat dilihat pada gambar II.13 sebagai berikut :



Gambar II.13. Tampilan Utama Visual Studio 2008

Sumber : Ketut Darmayuda (2008 : 12)

II.9. *Client Server*

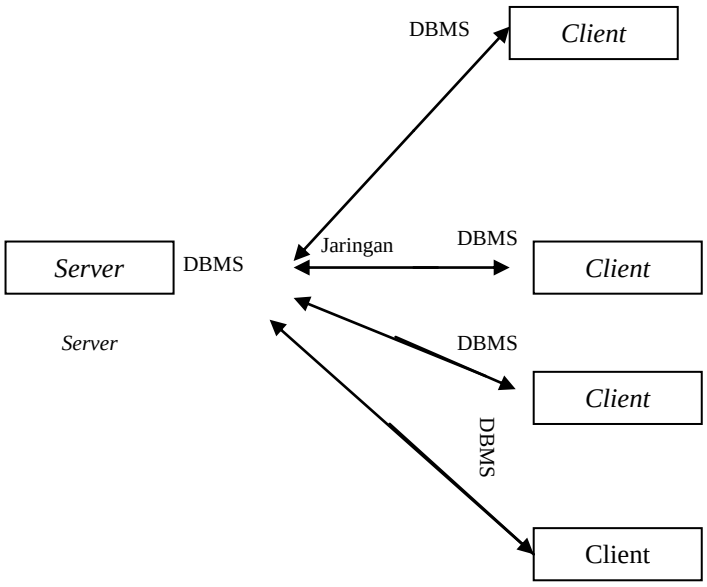
Client server adalah satu model komunikasi 2 komputer atau lebih yang berfungsi melakukan pembagian tugas. *Client* bertugas untuk melakukan input, update, dan menampilkan data sebuah *database*. Sementara *server* bertugas untuk menyediakan pelayanan untuk melakukan manajemen yaitu : menyimpan dan mengolah *database* (Wahana Komputer; 2010 : 5).

II.9.1. Arsitekur *Client* Server

1. Arsitektur StandAlone (1-*Tier*)

Model pertama aplikasi pemograman *database client server* adalah standalone atau 1 *tier* (1 -tingkat) adalah sebuah komputer yang mengakses sebuah *database* dari komponen sendiri. Dengan kata lain, aplikasi antarmuka *user* dan aplikasi DBMS terdapat pada komputer yang sama.

.



Gambar II.14. Arsitektur StandAlone (1-*Tier*)

Sumber : (Wahana Komputer; 2010 : 6)

Adapun karakteristik arsitektur 1 tier sebagai berikut :

a. Beban jaringan menjadi tinggi karena yang diminta adalah file database secara keseluruhan pada komputer *server client* melalui jaringan.

b. Setiap komputer pada jaringan harus mempunyai DBMS tersendiri untuk menyimpan hasil salinan dari *server* sehingga mengurangi sumber daya yang dimiliki oleh komputer *client* terutama *memory*.

c. Komputer *client* harus mempunyai kemampuan proses yang tinggi untuk mendapatkan waktu respon yang baik saat komputer *server* mengirimkan *file* yang diminta.

d. Arsitektur 1-*tier* cocok untuk bisnis kecil yang hanya membutuhkan data sebuah komputer untuk memproses dan menyimpan data sekaligus, tetapi kurang tepat diterapkan pada model jaringan

2. Arsitekur 2-*Tier*

Model kedua sebuah pemograman *database* adalah model 2-tier. Arsitektur pada model demikian membagi tugas antara komputer *client* server. Komputer *client* bertugas menyediakan antar muka untuk *user*, permintaan (*request data*) ke DBMS Server, serta pemrosesan data (mencakup logika penyajian data, logika pemrosesan data, dan logika atau bisnis) komputer *client* hanya mengirimkan sebuah *statement* untuk menambah (*insert*) data, mengubah (*update*), menghapus *(delete*), dan yang terakhir meminta *(select*) data untuk ditampilkan melalui antarmuka yang dibuat oleh *programmer*. Sedangkan *server* bertanggung jawab

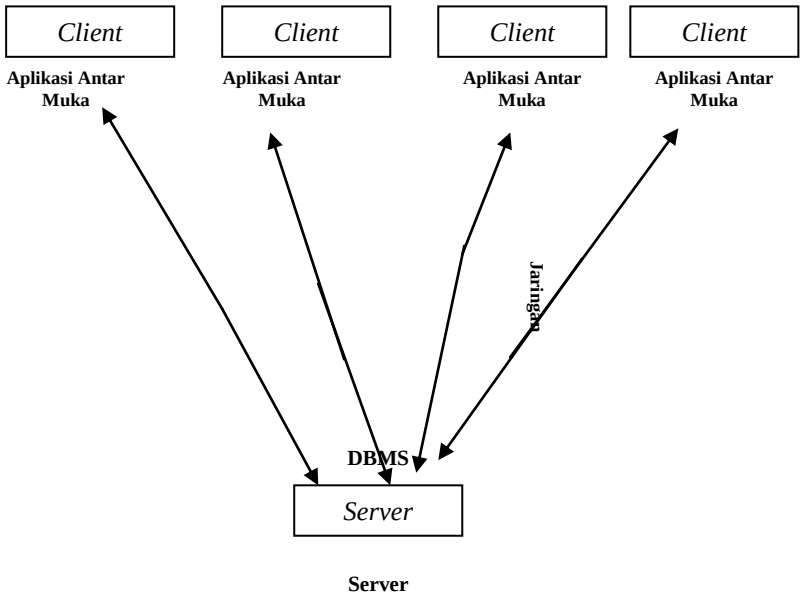
terhadap penyimpanan, pengelolaan, melayani permintaan akses data, dan pemrosesan *client*.

Karakteristik arsitektur 2-*tier* adalah :

a. 2-*tier* terjadi pada jaringan dan melakukan pemodelan programan *database* dalam 2 tingkat. Tingkat pertama adalah *client* dan tingkat kedua adalah *server*.
b. Tingkat pertama komputer *client* sebagai penyedia aplikasi antarmuka untuk mengolah *database*, baik menampilkan data kedalam *user interface*, menambah, menghapus data, maupun logika bisnis (*bussines logic*)
c. Tingkat kedua adalah *server* yang menyediakan aplikasi untuk mengelola *database* serta menyediakan pula *query stored procedure*, dan *triggers*, yang dapat dipanggil *client* untuk mengolah data.
d. Komputer *client* hanya mengirimkan sebuah *statement sql* untuk meminta data ke *server*.
e. *Server* hanya memberikan data yang diminta melalui *statement* bersangkutan.
f. Komputer *server* dituntut untuk memiliki kemampuan pemrosesan yang tinggi karena harus melayani permintaan banyak komputer *client* yang mengakses satu atau lebih DBMS.
g. Beban jaringan menjadi ringan karena data yang berjalan pada jaringan hanya data yang diminta oleh *client*.

h. Otentifikasi pemakai, pemeriksaan integritas, dan pemeliharaan kamus data dilakukan pada sisi *server.*

i. Sederhana dan mudah untuk diterapkan, khususnya pada bisnis kecil yang hanya terdapat pada satu gedung



Gambar II.15. Arsitekur 2-Tier

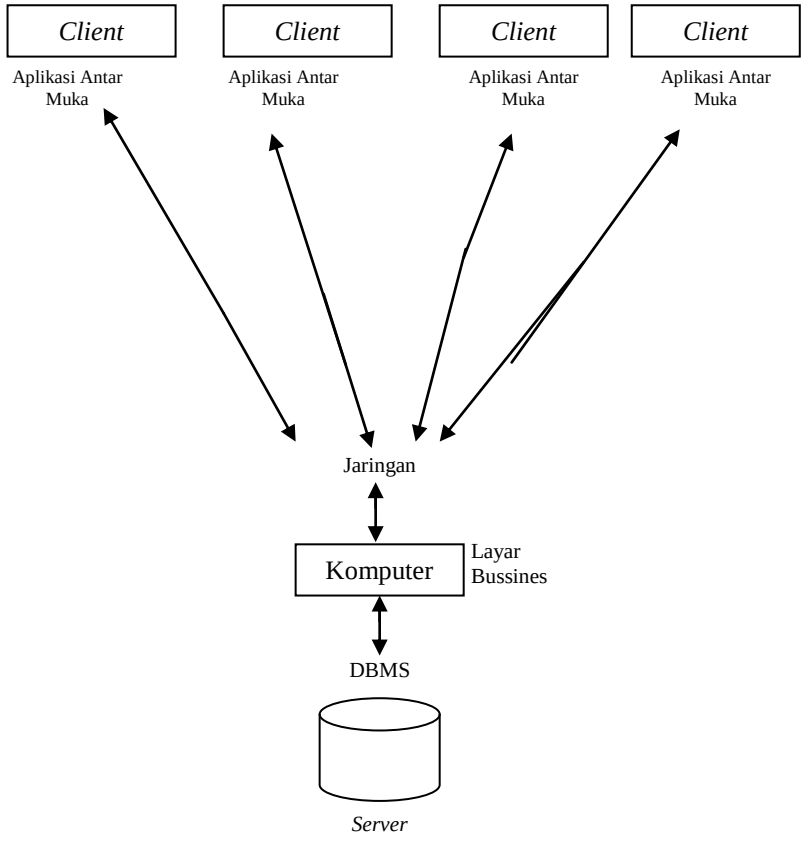
Sumber : (Wahana Komputer; 2010 : 8)

3. Arsitekur N-*Tier*

Arsitektur n-*tier* berarti membagi komponen menjadi n entitas yaitu 1 tier *client* dan n-1 *tier server*. Seperti pada model sebelumnya *client* bertugas menyediakan antarmuka aplikasi, sedangkan bertugas menyediakan data. Pada model n-tier (sebagai contoh adalah 3-tier), server dibagi 2 menjadi,

yaitu satu *server* yang dipakai sebagai *bussines object* (*middle tier*) dan satu *server* yang hanya menyimpan *database* (*server tier).*

Secara nyata model 3-*tier* adalah pada jaringan internetyang hanya memanfaatkan *database*. Internet lapisan pertama adalah komputer *client* yang menampilkan halaman *web*, tempat kontent atau data halaman *web* berasal dari *database*. Lapisan kedua adalah *web* atau *HTTP server* yang menterjemahkan *script server side* (*PHP*, *JSP*, *ASP*, dan lainnya) dari komputer *client* untuk meminta data pada *database*. Kemudian lapisan ketiga adalah komputer *database server* yang menyediakan *database* yang diminta oleh *web* atau HTTP *server* (Wahana Komputer; 2010 : 6-9).



Gambar II.16. Arsitekur N-Tier

Sumber : Wahana Komputer (2010 : 9)

II.9.2. *MYSQL*

Mysql adalah salah satu *software* sistem manajemen *database* (DBMS) *structured Query Language* (*SQL*) yang bersifat *open source*. *SQL* adalah bahasa standart untuk mengakses *database* dan didefenisikan dengan standart *ANSI/ISO SQL*. *MYSQL* dikembangkan, disebarluaskan, dan didukung oleh *MYSQL AB. MYSQL AB* adalah perusahaan komersial yang didirikan oleh pengembang *MYSQL*. *MYSQL* merupakan aplikasi *Relational Database Management System* (RDBMS) yang digunakan untuk aplikasi *client server* atau sistem *embedded.*

Mysql mempunyai beberapa sifat yang menjadikannya sebagai salah satu *software database* yang banyak digunakan oleh pemakai diseluruh dunia. Sifat-sifat yang dimiliki oleh *MYSQL* antara lain :

a. *Mysql* merupakan DBMS (*Database Management System*)

b. *Database* adalah kumpulan data yang terstruktur. Data dapat berupa daftar belanja, kumpulan gambar, atau yang lebih luas yaitu informasi jaringan perusahaan. Agar dapat menambah, mengakses, dan memproses data tersimpan pada sebuah komputer database, kita membutuhkan sistem manajement *database* (DBMS) seperti *MYSQL Server*. Sejak komputer sangat baik menangani sejumlah besar data, *sistem manajement database* (DBMS) memainkan peran utama dalam perhitungan baik sebagai peralatan yang berdiri sendiri maupun bagian aplikasi.

c. *Mysql* merupakan RDBMS (*Relational Database Management System*).

d. *Database relational* menyimpan data pada tabel-tabel yang terpisah, bukan menyimpan data dalam ruang penyimpanan yang besar. Hal ini menambah kecepatan *fleksibelitas*.

e. *Mysql* merupakan *software open source*.

f. *Open source* berarti setiap orang dapat menggunakan dan mengubah *software* yang bersangkutan. Setiap orang dapat mendownload *software MYSQL* dari internet dan menggunakan tanpa membayar. Bahkan jika mengkehendakinya anda bisa mempelajari kode sumber dan mengubah sesuai yang anda butuhkan (Wahana Komputer; 2010 : 26-27).