BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Secara sederhana suatu sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari suatu unsur, komponen, atau variabel yang terorganisir, saling tergantung satu sama lain dan terpadu. Teori sistem secara umum yang pertama kali diuraikan oleh Kennet Boulding, terutama menekan pentingnya perhatian terhadap setiap bagian yang membentuk sebuah sistem. Kecenderungan manusia yang mendapat tugas memimpin suatu organisasi adalah terlalu memusatkan perhatian pada salah satu komponen saja dari sistem organisasi.

Teori sistem melahirkan konsep-konsep futuristik, antara lain yang terkenal adalah konsep sibernetika (*cybernetics*). Konsep atau dibidang kajian ilmiah ini berkaitan dengan upaya menerapkan berbagai ilmu yaitu ilmu perilaku, fisika, biologi, dan teknik. Oleh karena itu sibernetika biasanya berkaitan dengan usaha-usaha otomasi tugas-tugas yang dilakukan manusia, sehingga melahirkan studi-studi tentang robotika, kecerdasan buatan (*artificial intelegence*). Unsur-unsur yang mewakili suatu sistem secara umum adalah masukan (*input*), pengolahan (*processing*), dan keluaran (*output*) (Tata Sutabri; 2012 : 10)

II.1.1. Karakteristik Sistem

Model umum sebuah sistem adalah input, proses, dan output. Hal ini merupakan konsep sebuah sistem yang sangat sederhana sebab sebuah sistem dapat mempunyai beberapa masukan dan keluaran. Selain itu sebuah sistem

memiliki karakteristik atau sifat-sifat tertentu, yang mencirikan bahwa hal tersebut bisa dikatakan sebagai suatu sistem. Adapun karakteristik yang dimaksud adalah sebagai berikut:

1. Komponen Sistem (*Component*)

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, artinya saling berkerja sama membentuk satu kesatuan. Komponen-komponen sistem tersebut dapat berupa suatu bentuk subsistem. Setiap subsistem memiliki sifat dari sistem yang menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai sistem yang lebih besar, yang disebut "supra sistem

2. Batas Sistem *(Boundary).*

   Ruang lingkup sistem merupakan daerah yang membatasi antara sistem dengan sistem yang lain atau sistem dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan yang tidak dapat dipisah-pisahkan

3. Lingkungan Luar Sistem *(Environment).*

   Bentuk apapun yang ada di luar lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut operasi lingkungan luar sistem. Lingkungan luar sistem ini dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut. Lingkungan yang menguntungkan merupakan bagi sistem tersebut. Dengan demikian, lingkungan luar tersebut harus dijaga dan dipelihara. Lingkungan luar

yang merugikan harus dikendalikan. Kalau tidak maka akan mengganggu kelangsungan hidup sistem tersebut

4. Penghubung Sistem *(Interface)*

   Media yang menghubungkan sistem dengan subsistem lainnya disebut penghubung sistem atau *interface*. Penghubung ini memungkinkan sumber-sumber daya mengalir dari subsistem lain. Bentuk keluaran dari satu subsistem akan menjadi masukan untuk subsistem lain melalui penghubung tersebut. Dengan demikian dapat terjadi suatu integrasi sistem untuk membentuk satu kesatuan.

5. Masukan Sistem *(Input)*

   Energi yang dimasukan ke dalam sistem disebut masukan sistem, yang dapat berupa pemeliharaan (*maintenance input*) dan sinyal (*signal input*). Contoh di dalam suatu sistem unit komputer. "program" adalah *maintenance* input yang digunakan untuk mengoperasikan komputernya dan "data" adalah *signal input* untuk diolah menjadi informasi.

6. Keluaran Sistem (*Output*)

   yaitu hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran ini merupakan masukan bagi subsistem yang lain. Contoh, sistem informasi. Keluaran yang dihasilkan adalah informasi. Informasi ini dapat digunakan sebagai masukan untuk pengambil keputusan atau hal-hal lain yang menjadi *input* bagi subsistem lain

7. Pengolah Sistem (*Proses*).

   Suatu sistem dapat mempunyai suatu proses yang akan mengubah masukan menjadi keluaran. Contoh, sistem akuntansi. Sistem ini akan mengolah data transaksi menjadi laporan-laporan yang dibutuhkan oleh pihak manajemen.

8. Sasaran Sistem (*Objective*)

   Suatu sistem memiliki tujuan dan sasaran yang pasti dan bersifat *deterministik*. Kalau suatu sistem tidak memiliki sasaran, maka operasi sistem tidak ada gunanya. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuan yang telah direncanakan (Tata Sutabri; 2012 :20-21).

II.1.2. Daur Hidup Sistem

Siklus hidup sistem (*system life cycle*) adalah merupakan proses evolusioner yang diikuti dalam menerapka sistem atau subsistem informasi komputer. Siklus hidup sistem terdiri dari serangkaian tugas yang erat mengikuti langkah-langkah pendekatan sistem sistem karena tugas-tugas tersebut mengkuti pola yang teratur dan dilakukan secara *top down*. Siklus hidup sistem sering dissebut sebagai pendekatan air terjun (*waterfall approach*) bagi pembangunan dan pengembangan sistem.

Pembangunan sistem hanyalah salah satu dari rangkaian daur hidup suatu sistem. Meskipun demikian, proses ini merupakan aspek yang sangat penting. Kita akan melihat beberapa fase/tahapan dari daur hidup suatu sistem.

1. Mengenali Adanya Kebutuhan.

   Sebelum segala sesuatunya terjadi, timbul suatu kebutuhan atau problema yang harus dapat dikenali sebagaimana adanya. Kebutuhan dapat terjadi sebagai hasil perkembangan dari organisasi dan volume yang meningkat melebihi kapasitas dari sistem yang ada. Semua kebutuhan ini harus dapat didefenisikan dengan jelas. Tanpa adanya kejelasan dari kebutuhan yang ada, pembangunan sistem akan kehilangan arah dan efektifitasnya.

2. Pembangunan Sistem

   Suatu proses atau seperangkat prosedur yang harus diikuti untuk menganalisis kebutuhan yang timbul dan membangun suatu sistem untuk dapat memenuhi kebutuhan tersebut.

3. Pemasangan Sistem

   Setelah tahap pembangunan sistem selesai. Sistem kemudian akan dioperasikan. Pemasangan sistem merupakan tahap yang penting pula dalam daur hidup sistem. Peralihan dari tahap pembangunan menuju tahap operasional terjadi pemasangan sistem yang sebenarnya, yang akan merupakan langkah akhir dari suatu pembangunan.

4. Pengoperasian Sistem

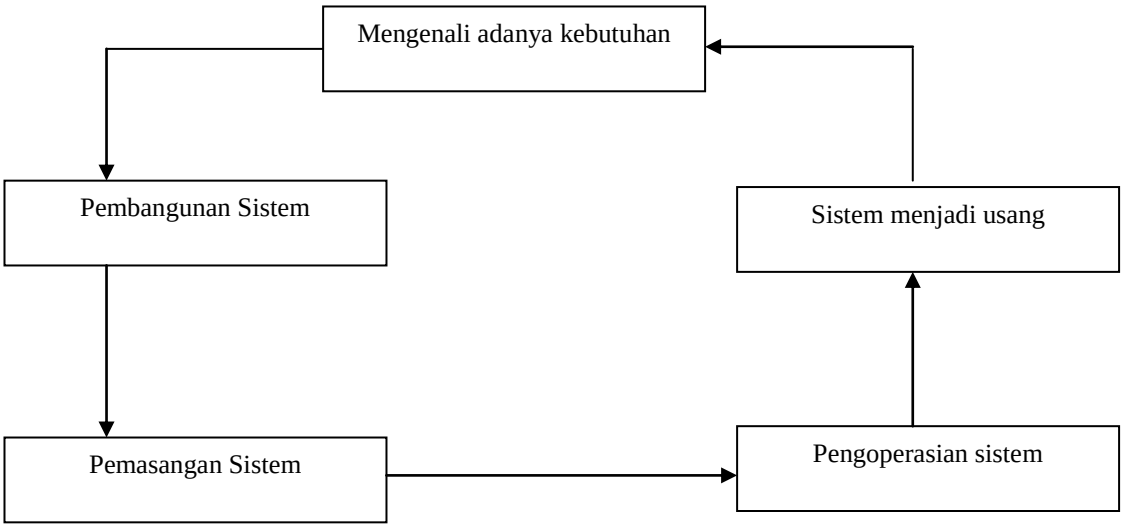
   Program-program komputer dan prosedur-prosedur pengoperasian yang membentuk suatu sistem informasi semuanya bersifat statis. Sedangkan organisasi ditunjang oleh sistem informasi tadi. Ia selalu mengalami perubahan-perubahan itu karena pertumbuhan kegiatan

bisnis, perubahan pengaturan, dan kebijaksanaan ataupun kemajuan teknologi. Untuk mengatasi perubahan-perubahan tersebut, sistem harus diperbaiki atau diperbaharui.

5. Sistem Menjadi Usang

   Kadang perubahan yang terjadi begitu drastis, sehingga tidak dapat diatasi hanya dengan melakukan perbaikan-perbaikan pada sistem yang berjalan. Tibalah saatnya secara ekonomis dan teknis sistem yang ada sudah tidak layak lagi untuk dioperasikan dan sistem yang baru perlu dibangun untuk menggantikannya.

Sistem informasi kemudian akan melanjutnya daur hidupnya. Sistem dibangun untuk memenuhi kebutuhan yang muncul. Sistem beradaptasi terhadap perubahan-perubahan lingkungannya dinamis. Sampailah pada kondisi dimana sistem tersebut tidak dapat lagi beradaptasi dengan perubahan-perubahan yang ada atau secara ekonomis tidak layak lagi untuk dioperasikan. Sistem yang baru kemudian dibangun untuk menggantikanya. Untuk dapat menggambarkan daur hidup sistem ini, lihat pada gambar II.1. sebagai berikut (Tata Sutabri; 2012 :26-28).



Gambar II.1. Daur Hidup Sistem

Sumber : Tata Sutabri (2012 : 29)

II.3. Informasi

Informasi adalah data yang telah diklasifikasikan diolah atau diinterpretasi untuk digunakan dalam proses pengambil keputusan. Sistem pengolahan informasi megolah data menjadi informasi atau tepatnya mengolah dari bentuk tak berguna menjadi berguna bagi penerimanya.

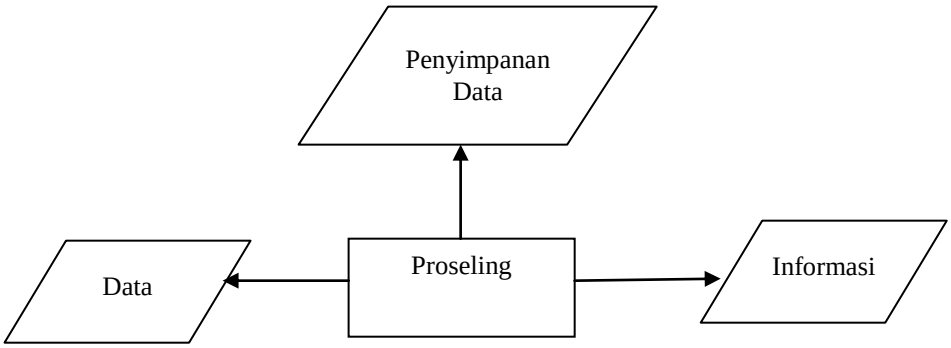
Informasi adalah data yang telah diklasifikasikan atau diinterpretasi untuk digunakan dalam pengambil keputusan (Tata Sutabri; 2012 : 29).

II.4. 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; 2012 :46)

II.5. Data

Mengenai pengertian data, lebih jelas apa yang didefenisikan oleh Drs. Jhon J. Longkutoy dalam bukunya " Pengenalan Komputer" sebagai berikut : isitilah data adalah suatu istilah majemuk yang berarti fakta atau bagian dari fakta yang mengandung arti yang dihubungkan dengan kenyataan simbol-simbol, gambar-gambar, angka-angka, huruf-huruf, atau simbol-simbol yang menunjukan suatu ide, objek, kondisi, atau situasi dan lain-lain (Tata Sutabri, 2012 : 2).



Gambar II.2. Pemrosesan Data
Sumber : Tata Sutabri, S. Kom, MM (2012 :2)

II.6. Jasa Layanan Service

Toyota memberikan jaminan kepada pembeli pertama dan pembeli/ pemilik berikutnya bahwa toyota akan memperbaiki atau atas bebannya mengganti bagian yang tidak berfungsi dengan baik, sebagai akibat cacat bahan atau karena kesalahan atau kekeliruan dalam pengerjaannya. Selama jangka waktu 36 bulan atau sampai kendaraan menempuh jarak 100.000 km, yang mana terjadi lebih dahulu, terhitung sejak penyerahan kepada pemilik pertama dari pihak PT. TOYOTA-ASTRA MOTOR maupun Dealer resmi PT. TAM Warranty. Toyota ini meliputi semua komponen kendaraan kecuali baterai, ban, komponen-

komponen yang bisa habis karena terpakai, dan terhadap hal-hal serta komponen yang tidak ditanggung oleh Warranty Toyota.

Warranty cat permukaan yang berkarat, kerusakan cat dan permukaan yang berkarat akibat cacat bahan atau kesalahan/ kekeliruan pengerjaan yang terjado pada panel body, dalam penggunaan yang normal dijamin oleh ketentuan dasar warranty toyota.

Warranty Baterai, masa warranty baterai yang digunakan sebagai perlengkapan asli kendaraan Toyota selama 24 bulan atau 50.000 km, yang mana dicapai terlebih dahulu

Warranty Ban, warranty claim ban diatur tersendiri dengan jaminan yang disediakan oleh pembuat ban (*Tier Maker*) (Warranty Toyota; 2008 : 11-12)

II.7. *Entity Relationship Diargam* (ERD)

Entity Relationship (ER) data model didasarkan pada persepsi terhadap dunia nyata yang tersusun atas kumpulan objek-objek dasar yang disebut entitas dan hubungan antar objek. Entitas adalah sesuatu objek dalam dunia nyata yang dapat dibedakan dari objek lain. Sebagai contoh, masing-masing mahasiswa adalah entitas dan mata kuliah dapat dianggap sebagai entitas.

Entitas digambarkan dalam basis data dengan kumpulan atribut. Misalnya atribut nim, nama, alamat, dan kota bisa menggambarkan data mahasiswa tertentu dalam suatu universitas. Atribut-atribut membentuk entitas mahasiswa. Demikian pula, atribut kodeMK, namaMK, dan SKS mendeskripsikan mata kuliah.

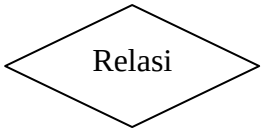
Atribut NIM digunakan sebagai untuk mengidentifikasikan mahasiswa secara unik karena dimungkinkan terdapat dua mahasiswa dengan nama, alamat,

dan kota yang sama. Pengenal unik harus diberikan pada masing-masing mahasiswa.

Relasi adalah hubungan antara beberapa entitas. Sebagai contoh relasi menghubungkan mahasiswa dengan mata kuliah yang diambilnya. Kumpulan semua entitas bertipe sama disebut kumpulan entitas (*entitas sel*), sedangkan kumpulan semua relasi bertipe sama disebut dengan kumpulan relasi (*relationship sel*).

Struktur logis (skema database) dapat ditunjukkan secara grafis dengan diagram ER yang dibentuk dari komponen-komponen berikut pada dilihat pada tabel II.1.

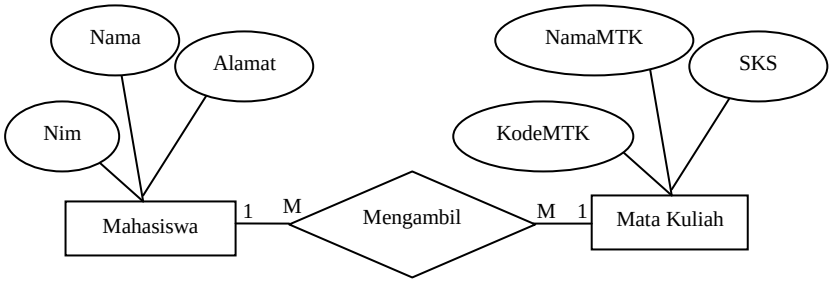
Tabel II.1. Komponen-Komponen Diagram ER

	Persegi Panjang mewakili kumpulan entitas
	Elips Mewakili Atribut
	Belah Ketupat Mewakili Relasi
	Garis Menghubungkan atribut dengan kumpulan entitas dan kumpulan entitas dengan relasi

Sumber : Imam Prayudi,dkk (2010 :60)

Masing-masing komponen diberi nama entitas atau relasi yang diwakilinya.

Sebagai ilustrasinya bayangkan anda mengambil bagian sistem basis data universitas yang terdiri dari mahasiswa dan mata kuliah. gambar II.2. menunjukkan diagram ER dari contoh. Diagram menunjukkan bahwa ada dua kumpulan entitas yaitu mahasiswa dan mata kuliah dan bahwa relasi mengambil contoh mahasiswa dan mata kuliah (Imam Prayudi,dkk; 2010 : 59-60).



Gambar II.3. Diagram ER

Sumber : Imam Prayudi, dkk (2010 : 60)

II.7.1. Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basis data relasional. Pada dasarnya normalisasi adalah proses dua langkah yang meletakkan data dalam bentuk tabulasi dengan menghilangkan kelompok berulang lalu menghilangkan data yang terduplikasi dari tabel relasional (*www. utexas. edu*).

1. Bentuk Nornal Pertama (1 NF)

   Contoh yang kita gunakan di sini adalah sebuah perancangan yang mendapatkan barang dari sejumlah pemasok. Masing-masing pemasok berada pada satu kota. Sebuah kota dapat mempunyai lebih dari satu

pemasok dan masing-masing kota mempunyai kode status tersendiri. Masing-masing pemasok bisa menyediakan banyak barang. Tabel relasionalnya dapat dituliskan sebagai berikut :

PEMASOK (P#, Status, Kota, b#, qty) di mana

p# : kode pemasok (kunci utama)

status : kode status kota

Kota : nama kota

b# : barang yang dipasok

qty : jumlah barang yang dipasok.

Sebuah tabel relasional secara defenisi selalu berada dalam bentuk normal pertama. Semua nilai pada kolom-kolomnya adalah atomi. Ini berarti kolom-kolom tidak mempunyai nilai berulang. Tabel II.2. menunjukkan tabel pemasok dalam 1 NF

Tabel II.2. Normalisasi Pertama Pemasok

P#	Status	Kota	B#	Qty
P1	20	Yogyakarta	B1	300
P1	20	Yogyakarta	B2	200
P1	20	Yogyakarta	B3	400
P1	20	Yogyakarta	B4	200
P1	20	Yogyakarta	B5	100
P1	20	Yogyakarta	B6	100
P2	10	Medan	B1	300
P2	10	Medan	B2	400
P3	10	Medan	B2	200
P4	20	Yogyakarta	B2	200
P4	20	Yogyakarta	B4	300
P4	20	Yogyakarta	B5	400

Sumber : (Imam Prayudi,dkk; 2010 :80).

2. Bentuk Normal Kedua (2 NF).

   defenisi bentuk normal kedua menyatakan bahwa tabel dengan kunci utama gabungan hanya dapat berada pada 1 NF, tetapi tidak pada 2 NF, sebuah tabel relasional berada pada bentuk normal kedua jika dia berada pada 1 NF dan setiap kolom bukan kunci yang sepenuhnya tergantung pada kunci utama. Ini berarti bahwa setiap kolom bukan kunci harus tergantung pada seluruh kolom yang membentuk kunci utama. Tabel pemasok berada pada 1 NF, tetapi tidak pada 2 NF karena status dan kota tergantung secara fungsional hanya pada kolom p# dari kunci gabungan (p#, b#). Ini dapat digambarkan dengan membuat daftar ketergantungan fungsional.

P# ⟶ Kota, Status

Kota ⟶ Status

(P#, B#) ⟶ qty

Proses mengubah tabel 1 NF ke 2 NF adalah :

a. tentukan sembarang kolom penentu selain kunci gabungan dan kolom-kolom yang ditentukannya.
b. Buat dan beri nama tabel baru untuk masing-masing penentu dan kolom-kolom yang ditentukan.
c. Pindahkan kolom-kolom yang ditentukan dari tabel asal ke tabel baru penentu akan menjadi kunci utama pada tabel baru.
d. Hapus kolom yang baru dipindahkan dari tabel asal, kecuali penentu yang akan berfungsi sebagai kunci tamu.

e. Tabel asal bisa diberi nama baru.

Pada contoh, kita memindahkan kolom p#, status, dan kota ke tabel baru yang disebut pemasok2. Kolom p# menjadi kunci utama tabel ini. Tabel II.3. menunjukkan hasilnya.

Tabel II.3. Tabel Bentuk Normal Kedua (2NF).

Pemasok2

P#	Status	Kota
P1	20	Yogyakarta
P2	10	Medan
P3	10	Medan
P4	20	Yogyakarta
P5	30	Bandung

Barang

P#	B#	Qty
P1	B1	300
P1	B2	200
P1	B3	400
P1	B4	200
P1	B5	100
P1	B6	100
P2	B1	300
P2	B2	400
P3	B2	200
P4	B2	200
P4	B4	300
P4	B5	400

Sumber : (Imam Prayudi,dkk; 2010 :82).

3. Bentuk Normal Ketiga (3 NF).

bentuk normal ketiga mengharuskan semua kolom pada tabel relasional hanya pada kunci utama. Secara defenisi, sebuah tabel berada pada bentuk normal ketiga (3 NF) jika tabel sudah berada pada 2 NF dan setiap kolom yang bukan kunci tidak tergantung secara transistif pada kunci utamanya. Dengan kata lain, semua atribut bukan kunci tergantung secara fungsional hanya pada kunci utama. Tabel barang sudah dalam bentuk normal ketiga. Kolom bukan kunci, qty, tergantung sepenuhnya pada kunci utama (p#, b#). Pemasok masih berada pada 2 NF, tetapi belum berada pada 3 NF karena dia

mengandung ketergantungan transitif. Ketergantungan transitif terjadi ketika sebuah kolom bukan kunci, yang ditentukan oleh kunci utama, menentukan kolom lainnya. Konsep ketergantungan transistif daopat digambarkan dengan menunjukkan ketergantungan fungsional pada pemasok2, yaitu :

Pemasok2. p# ⟶ Pemasok2, status

Pemasok2. p# ⟶ Pemasok2, kota

Pemasok2. kota ⟶ Pemasok2, status

Perlu dicatat bahwa pemasok2, status ditentukan, baik oleh kunci utama p#, maupun kolom bukan kunci, kota

Proses mengubah tabel menjadi 3 NF adalah :

a. Tentukan semua penentu selain kunci utama dan kolom yang ditentukannya.
b. Buat dan beri nama tabel baru untuk masing-masing penentu dan kolom yang ditentukannya.
c. Pindahkan kolom yang ditentukan dari tabel asal ke tabel baru. Penentu menjadi kunci utama tabel baru.
d. Hapus kolom yang baru saja dipindahkan dari tabel asal, kecuali penentu yang akan berfungsi sebagai kunci tamu.
e. Tabel asal bisa diberi nama baru.

Untuk mengubah PEMASOK2 menjadi 3 NF, kita membuat tabel baru yang disebut KOTA_STATUS dan memindahkan kolom

kota dan status ke tabel baru. Status dihapus dari tabel diberi nama baru PEMASOK_KOTA. Tabel II.4 menunjukkan hasilnya

Tabel II.4. Tabel Bentuk Normal Ketiga (3 NF)

PEMASOK_KOTA

P#	Kota
P1	Yogyakarta
P2	Medan
P3	Medan
P4	Yogyakarta
P5	Bandung

KOTA_STATUS

Kota	Status
Yogyakarta	20
Medan	10
Bandung	30
Semarang	40

Sumber : (Imam Prayudi,dkk; 2010 :83)

4. Bentuk Normal Boyce Code (BCNF)

   Setelah 3 NF, semua masalah normalisasi hanya melibatkan tabel yang mempunyai tiga kolom atau lebih dan semua kolom adalah kunci. Banyak praktisi berpendapat bahwa menempatkan entitas pada 3 NF sudah cukup karena sangat jarang entitas yang berada pada 3 NF bukan merupakan 4 NF dan 5 NF. Lebih lanjut, mereka berpendapat bahwa keuntungan yang didapat mengubah entitas ke 4 NF dan 5 NF sangat kecil sehingga tidak perlu dikerjakan. Bentuk Normal Boyce- Code (BCNF) adalah versi 3 NF lebih teliti dan berhubungan dengan tabel relasional yang mempunyai (a) banyak kunci kandidat (b) kunci kandidat gabungan, dan (c) kunci kandidat yang saling tumpang tindih. BCNF didasarkan pada konsep penentu. Sebuah kolom penentu adalah kolom di mana kolom-kolom lain sepenuhnya tergantung secara fungsional. Sebuah tabel relasional berada pada BCNF jika dan hanya setiap penentu adalah kunci kandidat.

5. Bentuk Normal Keempat (4 NF)

   Sebuah tabel relasional berada pada bentuk normal keempat (4 NF) jika dia dalam BCNF dan semua ketergantungan multivalue merupakan ketergantungan fungsional.

   Bentuk normal keempat (4 NF) didasarkan pada konsep ketergantungan multivalue (MVD). Sebuah ketergantungan multivalue terjadi ketika dalam sebuah tabel relasional yang mengandung setidaknya tiga kolom, satu kolom mempunyai banyak baris bernilai sama, tetapi kolom lain bernilai berebeda.

   Defenisi secara formal diberikan oleh CJ. Date, yaitu :

   Misalnya, ada sebuah tabel relasional R dengan kolom A, B dan C, Maka R.A ⟶ R.B (kolom A menentukan kolom B).

   Adalah benar jika dan hanya jika himpunan nilai B yang cocok dengan pasangan nilai A dan nilai C pada R hanya tergantung pada nilai A dan tidak tergantung pada nilai C.

   MVD selalu terjadi dalam pasangan, yaitu R.A ⟶ R.B dipenuhi jika dan hanya jika R.A⟶ R.C dipenuhi pula.

6. Bentuk Normal Kelima (5 NF).

   Sebuah tabel berada pada bentuk normal kelima jika dia tidak dapat mempunyai dekomposisi lossless menjadi sejumlah tabel lebih kecil.

   Empat bentuk normal pertama berdasarkan pada konsep ketergantungan fungsional, sedangkan bentuk normal kelima berdasarkan pada konsep ketergantungan gabungan (*join dependence*).

Ketergantungan gabungan berarti sebuah tabel, setelah deskomposisi menjadi tiga atau lebih tabel yang lebih kecil, harus dapat digabungkan kembali untuk membentuk tabel asal. Dengan kata lain 5 NF menunjukkan ketika sebuah tabel tidak dapat dideskomposisi lagi (Imam Prayudi,dkk; 2010 : 77 - 86).

II.7.2. Basis Data (*Database*)

Basis data menurut Stephen dan Plew adalah (2000) adalah mekanisme yang digunakan untuk menyimpan informasi atau data.

Kemudian Silberchatz, dkk (2002) mendefenisikan basis data sebagai kumpulan data berisi informasi yang sesuai untuk perusahaan. Sistem manajemen basis data (DBMS) adalah kumpulan data yang saling berhubungan dan kumpulan program untuk mengakses data. Tujuan utama sistem manajemen basis data adalah menyediakan cara menyimpan dan mengambil informasi basis data secara mudah dan efesien (Imam Prayudi,dkk; 2010 :1)

II.8. *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 (Prabowo Pudjo Widodo Dan Herlawati; 2011 : 6-7).

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 indentik dengan model karena model lebih luas dari diagram.

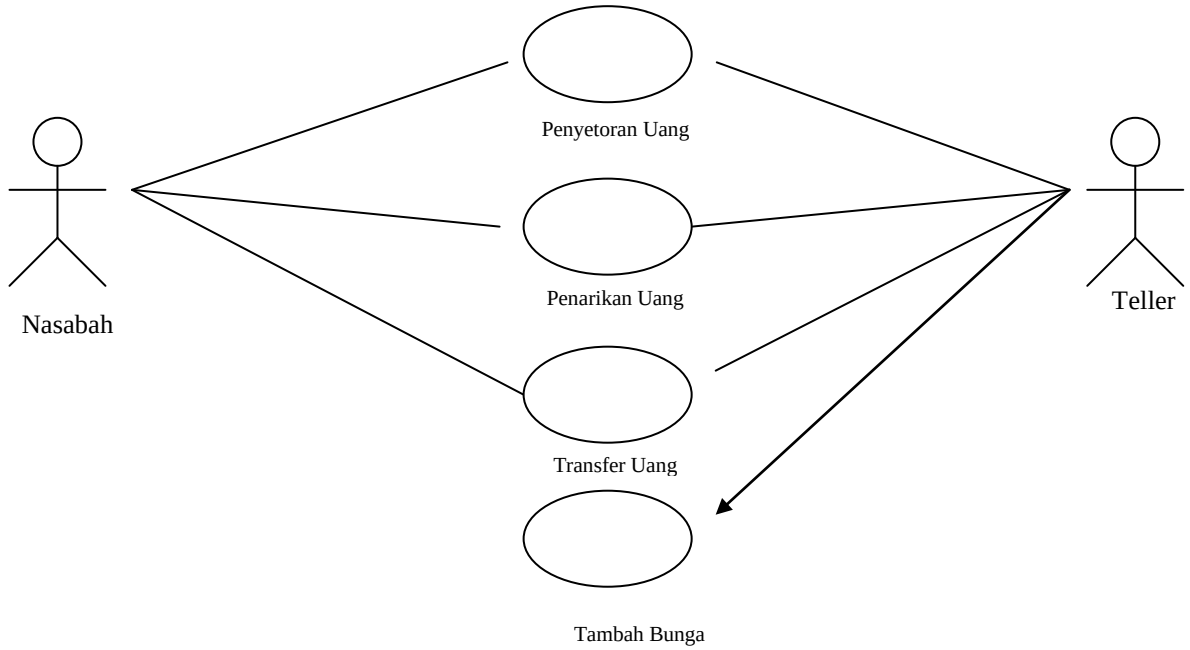
   komponen pembentuk diagram *use case* adalah :

   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 Pudjo Widodo Dan Herlawati; 2011 : 16-17).

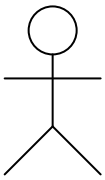


Gambar II.4. Diagram *Use Case*

Sumber : Prabowo Pudjo 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.*

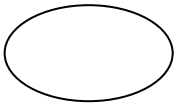


Gambar II.5. Aktor

Sumber : Prabowo Pudjo 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.6. Simbol *Use Case*

Sumber : Prabowo Pudjo 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)*

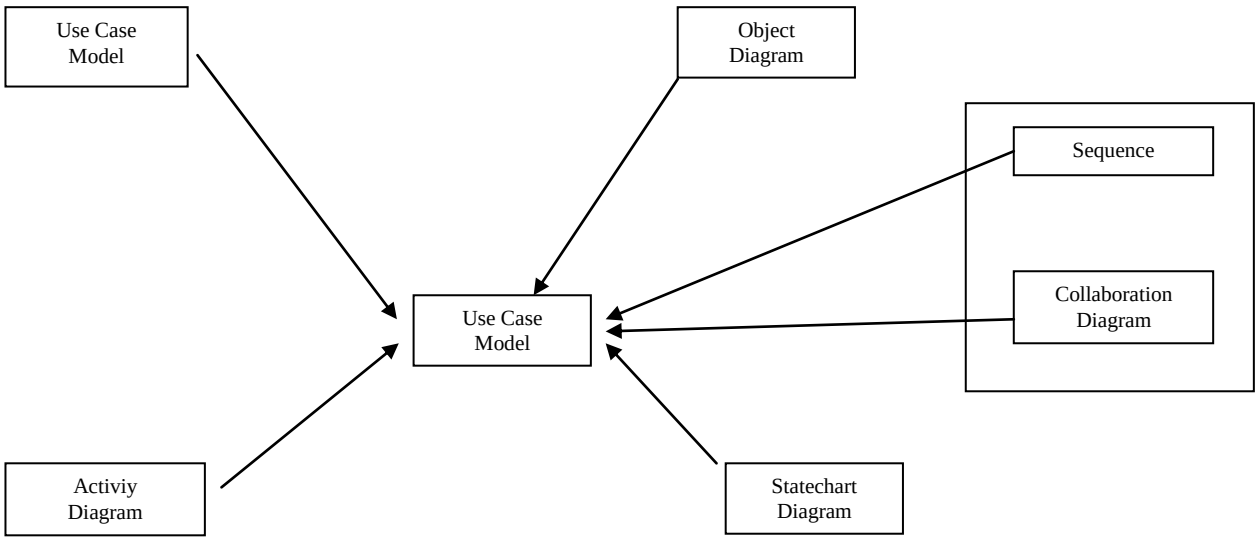
Kita biasanya membutukan *use case* yang membatalkan tujuan, misalnya pada u*se case* pemesanan kamar, dibutuhkan pula *use case* pembatalan pesanan kamar.

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.

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).

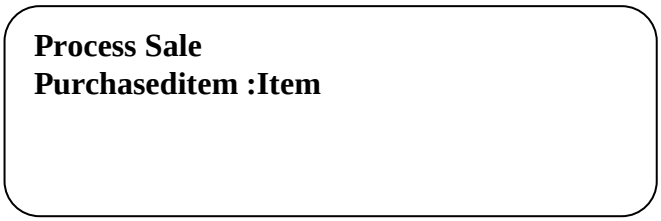


Gambar II.7. Hubungan Diagram Kelas Dengan Diagram *UML* lainya

Sumber : Prabowo Pudjo Widodo Dan Herlawati (2011 : 38)

5. Diagram Aktivitas (*Activity Diagram*)

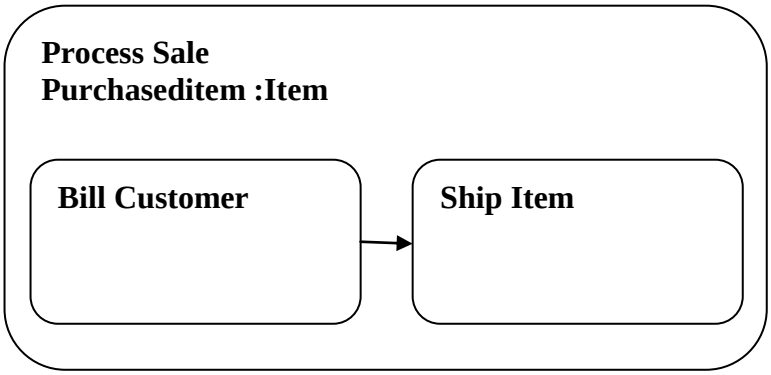
Diagram aktivitas lebih memfokuskan diri pada eksekusi dan alur sistemdari pada bagaimana sistem dirakit. Diagram ini tidak hanya memodelkan software melainkan memodelkan bisnis juga. Diagram aktivitas menunjukan aktivitas sistem dalam kumpulan aksi-aksi. Ketika digunakan dalam pemodelan *software*, 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 (Probowo Pudji Widodo, Dan Herlawati; 2011 : 143-145).



Gambar II.8. Aktivitas serderhana tanpa rincian

Sumber : Prabowo Pudjo 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.



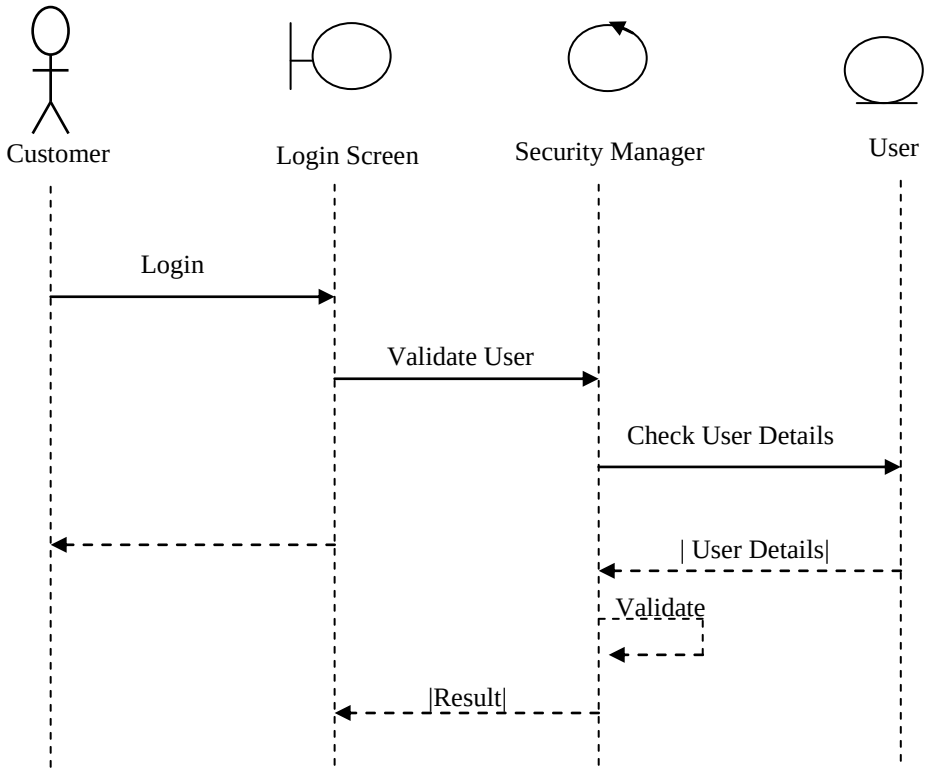
Gambar II.9. Aktivitas dengan detail rincian

Sumber : Prabowo Pudjo Widodo Dan Herlawati (2011:145)

6. *Sequence 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.9. memperlihatkan contoh diagram urutan dengan notasi-notasinya yang akan dijelaskan nantinya (Prabowo Pudjo Widodo Dan Herlawati; 2011 : 174 – 175).



Gambar II.10. Diagram Urutan

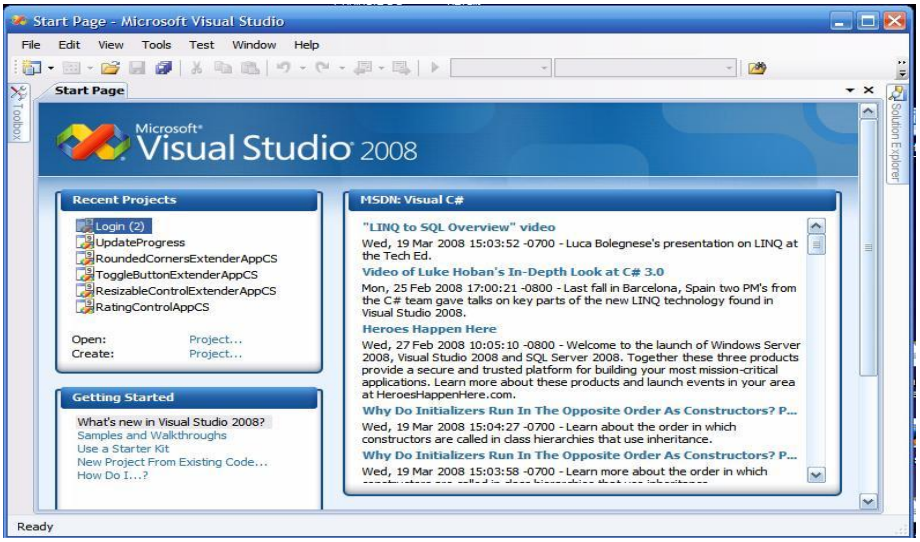
Sumber : Prabowo Pudjo Widodo Dan Herlawati (2011:175)

II.9. Bahasa Pemograman *Microsoft Visual Studio 2008*

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.11. sebagai berikut :



Gambar II.11. Tampilan Utama Visual Studio 2008

Sumber : Ketut Darmayuda (2009 : 12)

II.10. *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 table-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).

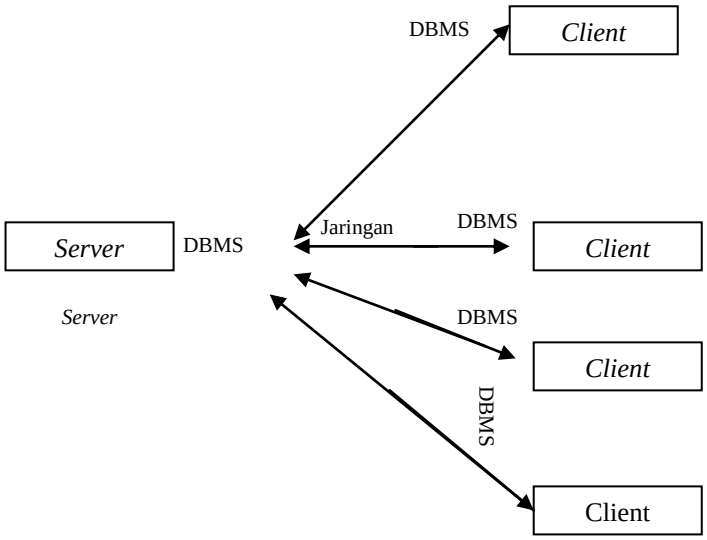
II.10.1. *Client Server*

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.10.2. 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.12. 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 (Wahana Komputer; 2010 : 7).

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*.

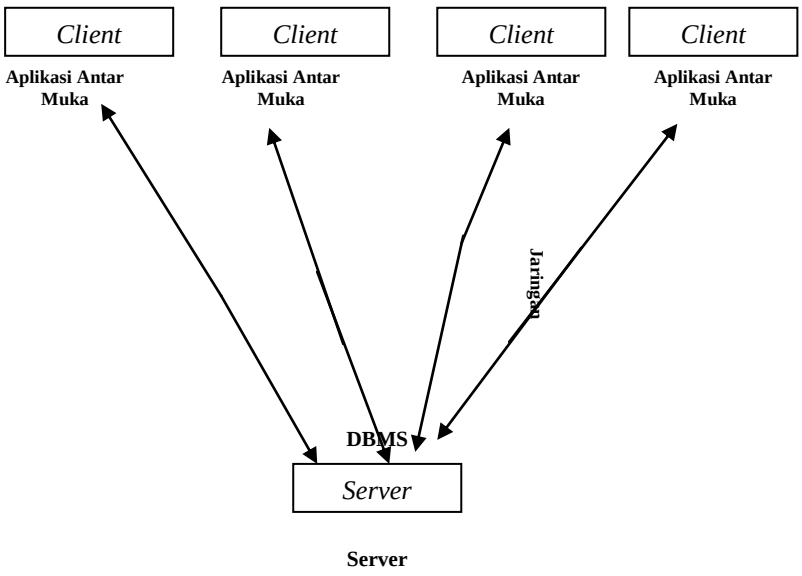
e. *Server* hanya memberikan data yang diminta melalui *statement* bersangkutan.

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 (Wahana Komputer; 2010 : 7-8).



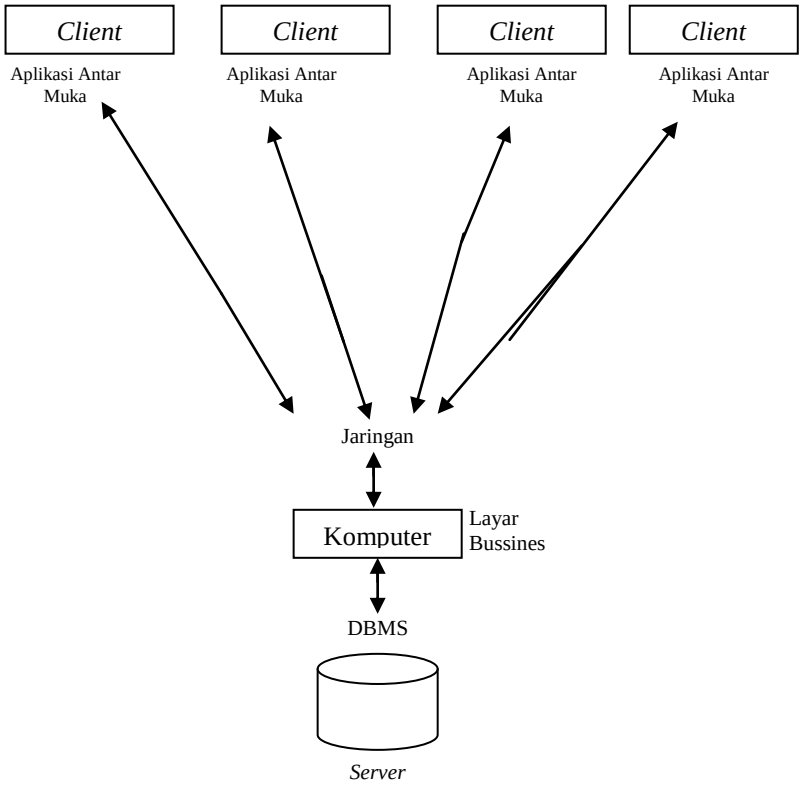
Gambar II.13. Arsitekur 2-Tier

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.14. Arsitekur N-Tier

Sumber : Wahana Komputer (2010 : 9)