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*).

Selain itu, suatu sistem tidak bisa lepas dari lingkungan maka umpan balik (*feed back*) dapat berasal dari lingkungan sistem yang dimaksud. Organisasi dipandang sebagai suatu sistem yang tentunya akan memiliki semua unsur ini (Tata Sutabri ; 2012 : 7).

II.2. 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 : 23).

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 ; 2012 : 42).

.

II.3.1. Komponen Dan Jenis Sistem Informasi.

Sistem informasi terdiri dari komponen-komponen yang disebut blok bangunan (*building block*), yang terdiri dari blok masukan, blok model, blok keluaran, blok teknologi, blok basis data dan blok kendali. Sebagai suatu sistem, keenam blok tersebut masing-masing saling berinteraksi satu dengan yang lain membentuk satu kesatuan untuk mencapai sasaran.

1. Blok Masukan (*Input Block*).

   Input mewakili data yang masuk ke dalam sistem informasi. *Input* di sini termasuk metode dan media untuk menangkap data yang akan dimasukan, yang dapat berupa dokumen-dokumen dasar.

2. Blok Model (*Model Block*)

   Blok ini terdiri dari kombinasi prosedur, logika, dan model matematik yang akan memainipulasi data *input* dan data yang tersimpan di basis data dengan cara yang sudah tertentu untuk menghasilkan keluaran yang diinginkan.

3. Blok Keluaran (*Output Block*)

   Produk dari sistem informasi adalah keluaran yang merupakan informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta pemakai sistem.

4. Blok Teknologi (*Technology Block*)

   Teknologi merupakan *"tool box"* dalam sistem informasi. Teknologi digunakan untuk menerima *input*, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran dan membantu pengendalian dari sistem keseluruhan. Teknologi sistem terdiri dari 3 (tiga) bagian yaitu teknisi teknologi (*brainware*), perangakat lunak (*software*), dan perangkat keras (*hardware*).

5. Blok Basis Data (*Database Block*)

   Basis data (*database*) merupakan kumpulan data yang saling berkaitan dan berhubungan satu dengan lainnya, tersimpan di perangkat keras komputer dan menggunakan perangkat lunak untuk memanipulasinya. Data perlu disimpan

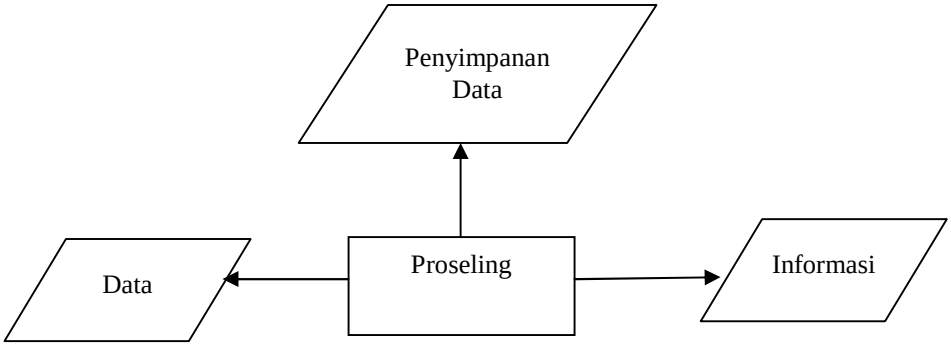
di basis data untuk keperluan penyediaan informasi lebih lanjut. Data di dalam basis data perlu diorganisasikan sedemikian rupa supaya informasi yang dihasilkan berkualitas. Organisasi basis data yang baik juga berguna untuk efesiensi kapasitas penyimpanannya. Basis data diakses atau dimanipulasi menggunakan perangkat lunak paket yang disebut DBMS (*database management system*).

6. Blok Kendali (*Control Block*)

   Banyak hal yang dapat merusak sistem informasi, seperti bencana alam, api, temperature, air, debu, kecurangan-kecurangan, kegagalan-kegagalan sistem itu sendiri, ketidak efesienan, sabotase, dan lain sebagainya. Beberapa pengendalian perlu dirancang dan diterapkan untuk meyakinkan bahwa hal-hal yang dapat merusak sistem dapat dicegah atau bila terlanjur terjadi kesalahan-kesalahan dapat langsung cepat diatasi (Tata Sutabri ; 2012 : 42).

II.4. 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 : 16).



Gambar II.1. Pemrosesan Data

Sumber : (Tata Sutabri, 2012).

2.

II.5. Sistem Informasi Akuntansi Utang

Tujuan Penyusunan Sistem Akuntansi Utang disebabkan karena :

1. Utang merupakan kewajiban perusahaan yang timbul akibat adanya transaksi pembelian secara kredit dan harus dibayar tepat waktu. Kelemahan sistem akuntansi utang akan mengakibatkan kerugian bagi perusahaan.
2. Pembayaran kewajiban perusahaan kepada pihak luar dengan tepat waktu mencerminkan adanya bonafitas perusahaan yang baik.

II.5.1. Prosedur Pencatatan Utang

Utang merupakan pengorbanan alat ekonomis yang akan terjadi pada waktu yang akan datang yang di sebabkan oleh kewajiban-keawajian di waktu sekarang dari suatu badan usaha yag akan dipenuhi dengan memberikan jasa maupun menstransfer aktiva badan usaha lain dimasa datang sebagai akibat dari transaksi-trasaksi yang sudah lalu.

Ada dua metode pencatatan utang :

1. *Account Payable Procedure* adalah berupa kartu utang yang diselenggarakan untuk tiap kreditur yang memperlihatkan catatan mengenai nomor faktur dari pemasok, jumlah yang terutang, jumlah pembayaran, dan saldo utang.

   Dokumen yang digunakan dalam *account payable procedure* adalah sebagai berikut :

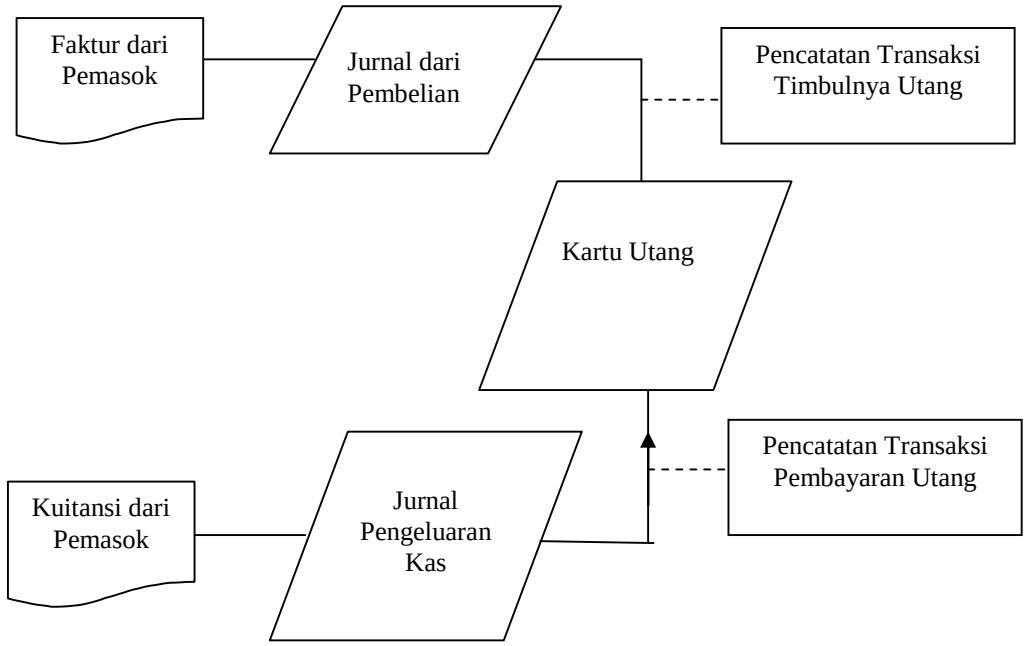
   a) Faktur dari pemasok
   b) Kuitansi tanda terima uang yang ditanda tangani oleh pemasok atau tembusan surat pemberitahuan (*remittance advice*) yang dikirim pemasok yang berisi keterangan untuk apa pembayaran tersebut dilakukan.

   Catatan akuntansi yang digunakan :

   a) Kartu utang untuk mencatat mutasi dan saldo utang kepada tiap kreditur
   b) Jurnal pembelian, digunakan untuk mencatat transaksi pembelian
   c) Jurnal pengeluaran kas, digunakan untuk mencatat transaksi pembayaran utang dan pengeluaran kas lain.

   Prosedur pencatatan utang

   a) Faktur dari pemasok dicatat dalam jurnal pembelian
   b) Informasi dalam jurnal pembelian kmd diposting ke dalam kartu utang yang diselenggarakan untuk setiap kreditur.
   c) Cek dicatat dalam jurnal pengeluaran kas
   d) Informasi dalam jurnal pengeluaran kas yang bersangkutan dengan pembayaran utang di posting ke dalam kartu utang.



Gambar II.2. Skema Penacatatan Utang *Account Payable Procedure*

Sumber (Pusat Pengambangan Penataan Guru Kejuruan Bisnis Dan Manajemen ; 2006 : 5)

2. *Voucher Payable Procedure* adalah tidak diselenggarakan adanya kartu utang, namun digunakan arsip voucher (bukti kas keluar) yang disimpan dalam arsip menurut abjad atau menurut tanggal jatuh tempo. Arsip bukti kas keluar ini berfungsi sebagai pencatatan utang.

   Dimana bukti kas keluar, mempunyai tiga fungsi yaitu sebagai berikut :

   a. Sebagai suat perintah kepada bagian kassa untuk melakukan pengeluaran kas sejumlah yang tercantum didalamnya.
   b. Sebagai pemberitahua kepada reditor mengenai tujuan pembayarannya
   c. Sebagai bahan mediaa untuk dasar pencatatan utang persediaan.

II.5.2. Sistem Informasi Pembelian Kredit

Sistem informasi pembelian kredit Sebuah sistem informasi pembelian pada suatu perusahaan umumnya memiliki aktivitas-aktivitas sebagai berikut :

1. Permintaan pembelian dari bagian yang membutuhkan, misalnya Bagian Gudang.
2. Pemesanan barang ke suplier oleh bagian pembelian
3. Penerimaan barang dan dokumen dari suplier
4. Pencatatan barang masuk
5. Pencatatan utang jika pembelian kredit
6. Pembayaran utang jika pembelian kredit

II.5.3. Sistem Informasi Akuntansi (SIA)

Sistem informasi akuntansi adalah sistem yang bertujuan untuk mengumpulkan dan memproses data serta melaporkan informasi yang berkaitan dengan transaksi keuangan. Misalnya, salah satu input dari Sistem Informasi Akuntansi pada sebuah toko penjualan Komputer dan sejenisnya, dalam transaksi penjualan. Kita memproses transaksi dengan mencatat penjualan tersebut kedalam jurlnal penjualan, mengklasifikasikan transaksi dengan menggunakan kode rekening, dan memposting transaksi kedalam jurnal. Kemudian, secara periodik sistem informasi akuntansi akan menghasilkan output berupa laporan keuangan yang terdiri dari neraca dan laba rugi (Anastasia Diana & Lilis Setiawati;2011 :4).

II.6. *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.

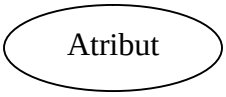
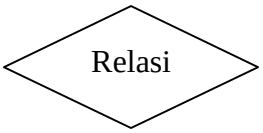
Entitas digambarkan dalam basis data dengan kumpulan atribut. Misalnya atribut *supplier*, atribut pembeli, nama, alamat, juga kota bisa menggambarkan data supplier dan pembeli tertentu dalam suatu akuntansi pembelian. Atribut-atribut membentuk entitas *supplier* dan pembeli. Demikian pula, atribut kodeBarang, namaSupplier, dan pembeli mendeskripsikan akuntansi pembelian.

Atribut *Supllier* digunakan sebagai untuk mengidentifikasikan pembelian secara unik karena dimungkinkan terdapat dua *Supplier* dengan nama, alamat, dan kota yang sama. Pengenal unik harus diberikan pada masing-masing *Supplier*.

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.

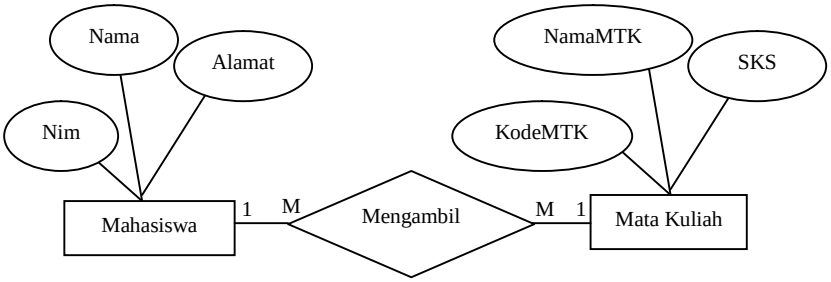
Tabel II.1. Komponen-Komponen Diagram ER

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

Sumber : (Janner Simarmata, 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.3. 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 (Janner Simarmata ; 2010 : 60).



Gambar II.3. Diagram ER

Sumber : (Janner Simarmata ; 2010 : 60)

II.6.1. 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 : (Janner Simarmata ; 2010 : 71).

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 : (Janner Simarmata ; 2010 : 81).

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 : (Janner Simarmata ; 2010 : 82).

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 (Janner Simarmata ; 2010 : 86).

II.6.2. Basis Data (*Database*)

Basis Data adalah suatu susunan/kumpulan data oprasional lngkap dari suatu organisasi/prusahaan yang diorganisir/dikelolah dan disimpan secara trintegrasi dengan menggunakan metode tertentu menggunakan komputer sehingga mampu menyediakan informasi optimal yang diperlukan pemakainya.

1. Keuntungan DBMS (*Database Management System*)

   DBMS memungkinkan perusahaan maupun pengguna individu untuk :

   a. Mengurangi perulangan data

      Apabila dibandingkan dengan *file-file* komputer yang disimpan terpisah di setiap lokasi komputer. DBMS mengurangi jumlah total *file* dengan menghapus data yang terduplikasi di berbagai *file.* Data terduplikasi selebihnya dapat ditempatkan dalam satu *file.*

   b. Mencapai independensi data

      Spesifikasi data disimpan dalam skema pada tiap program aplikasi. Perubahan dapat dibuat pada struktur data tanpa mempengaruhi program yang mengakses data.

   c. Mengintergrasikan data beberapa *file*

      Saat *file* dibentuk sehingga menyediakan kegiatan logis, maka organisasi fisik bukan merupakan kendala. Organisasi logis,

pandangan pengguna, dan program aplikasi tidak harus tercemin pada media.

d. Mengambil data dan informasi dengan cepat.

Hubungan-hubungan logis, bahasa manipulasi data, serta bahasa *query* memungkinkan pengguna mengambil data dalam hitungan detik atau menit.

e. Meningkatkan keamanan

DBMS *mainframe* maupun komputer mikro dapat menyertakan beberapa lapis keamanan seperti kata sandi (*password*), direktori pemakai, dan bahasa sandi (*encryption*) sehingga data yang dikelola akan lebih aman.

2. Kerugian DBMS

Keputusan menggunakan DBMS mengikat perusahaan atau pengguna untuk :

a. Memperoleh perangkat lunak

DBMS *mainframe* masih sangat mahal. Walaupun harga DBMS berbasis komputer mikro lebih murah, tetapi tetap merupakan pengeluaran besar bagi suatu oraganisasikecil.

b. Memperoleh konfigurasi perangkat keras yang besar

DBMS sering memerlukan kapasitas penyimpanan dan memori lebih besar dari pada program aplikasi lain.

c. Mempekerjakan dan mempertahankan staf DBA

DBMS memerlukan pengetahuan khusus agar dapat memanfaatkan kemampuannya secara penuh. Pengetahuan khusus ini disediakan paling baik oleh para pengguna basis data (DBA)

Baik basis data terkomputerisasi maupun DBMS bukanlah prasyarat untuk memecahkan masalah. Namun, keduanya memberikan dasar-dasar mengunakan komputer sebagai suatu sistem informasi bagi para spesialis informasi dan pengguna (Linda Marlinda S.Kom ; 2004 : 1).

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

UML singkatan dari *Unified Modelling Langguage* yang berarti bahasa pemodelan standart. (Chonoles, 2003) 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 Pudjo Widodo Dan Herlawati ; 2011 : 3).

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 adalah 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 Pudjo Widodo,Dan Herlawati ; 2011 : 5).

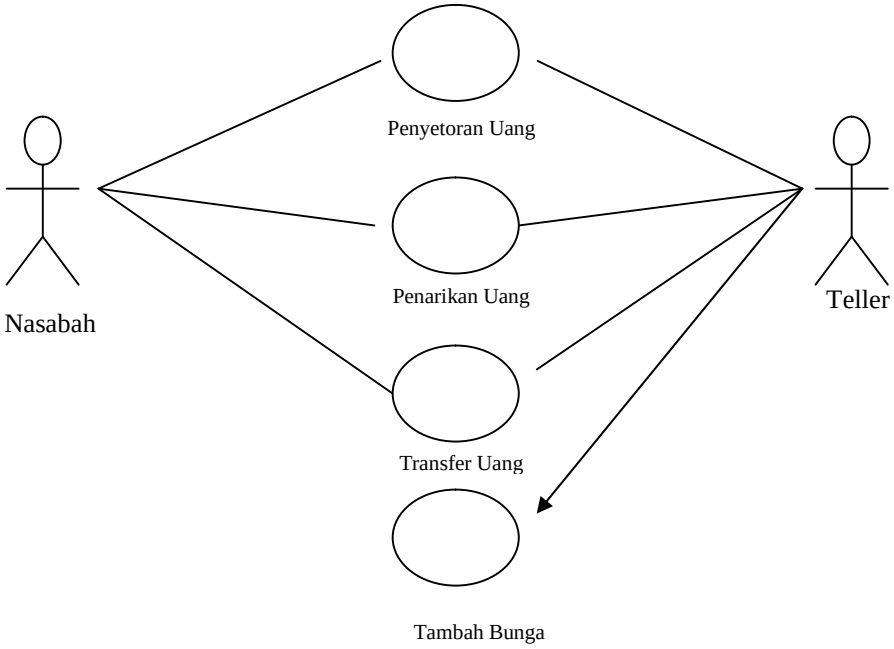
1. *Diagram Use Case (Use Case Diagram)*

   Use Case menggambarkan *external view* dari sistem yang akan kita buat modelnya. Menurut (Pooley, 2003) 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 : 9).

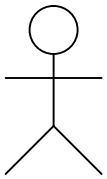


Gambar II.4. Diagram *Use Case*

Sumber : (Prabowo Pudjo Widodo Dan Herlawati ;2011 : 9)

2. Aktor

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

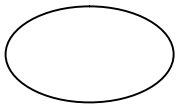


Gambar II.5. Aktor

Sumber : (Prabowo Pudjo Widodo Dan Herlawati ;2011 : 10)

3. *Use Case*

Menurut (Pilone, 2005) *use case* menggambarkan fungsi tertentu dalam suatu sistem berupa komponen kejadian atau kelas. Sedangkan menurut (Whitten, 2004) 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 : 14)

Use case sangat menentukan karakteristik sistem yang kita buat, oleh karena itu (Chonoles, 2003) 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)*

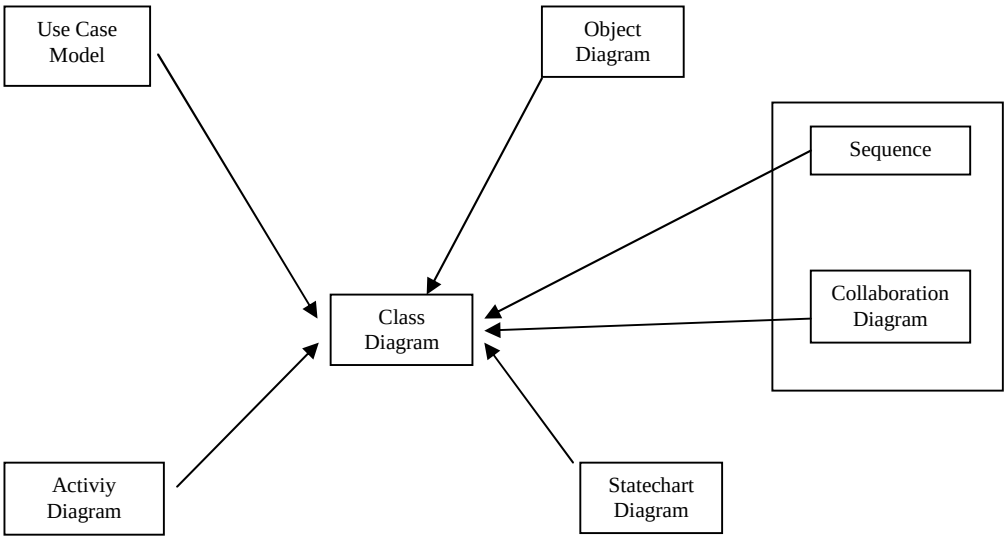
Kita biasanya membutukan *use case* yang membatalkan tujuan, misalnya pada u*se case* pemesanan kamar, dibutuhkan pula *use case* pembatalan pesanan kamar.

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).



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

Sumber : (Prabowo Pudjo Widodo Dan Herlawati ;2011 : 18)

5. Diagram Aktivitas (*Activity Diagram*)

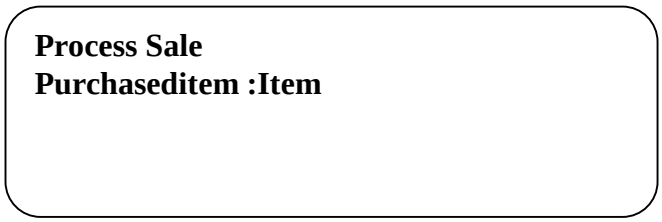
Diagram aktivitas lebih memfokuskan diri pada eksekusi dan alur sistemdari pada bagaimana sistem dirakit. Diagram ini tidak hanya memodelkan software melainkan memodelkan bisnis juga. Diagram aktivitas menunjukan aktivitas sistem dalam kumpulan aksi-aksi. Ketika digunakan dalam pemodelan *software*, 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 : 24).

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.

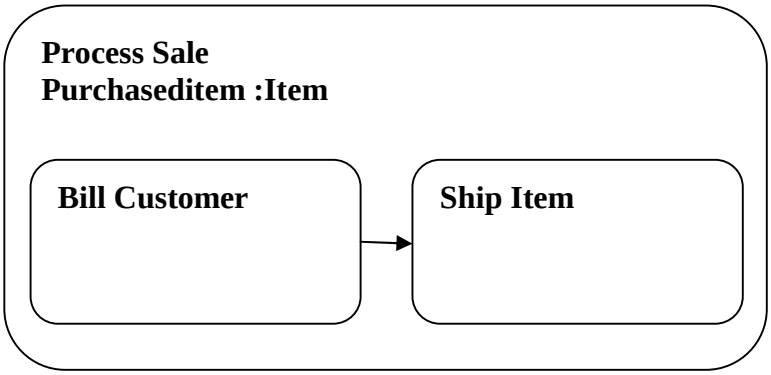
Aktivitas digambarkan dengan persegi panjang tumpul. Namanya ditulis di kiri atas. Parameter yang terlibat dalam aktivitas ditulis dibawahnya.



gambar II.8. Aktivitas serderhana tanpa rincian

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

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



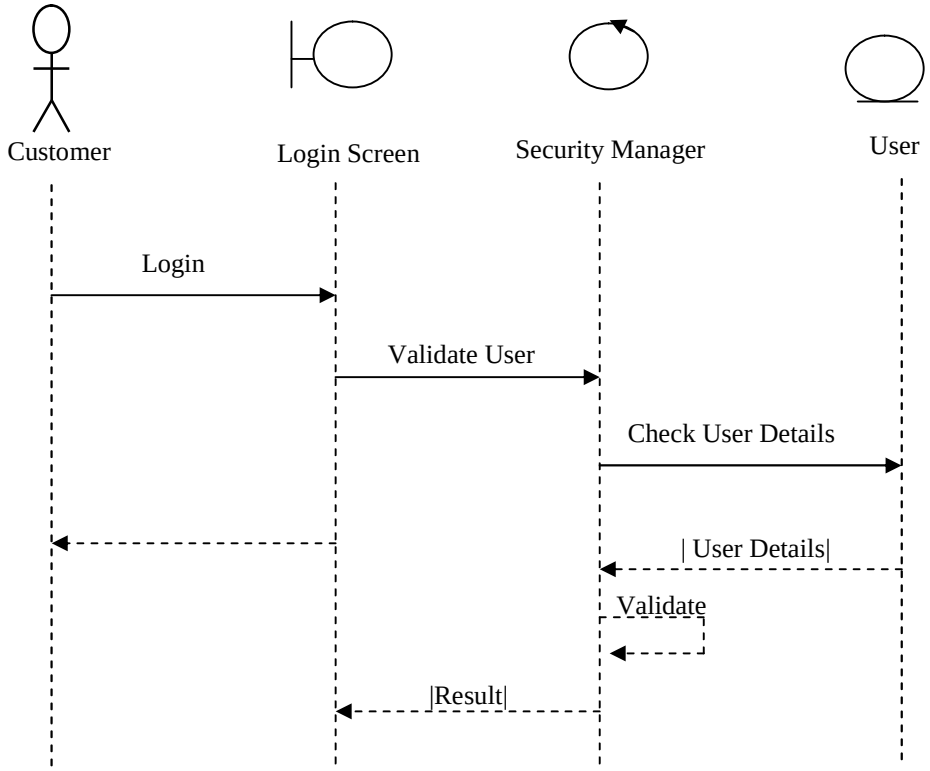
Gambar II.9. Aktivitas dengan detail rincian

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

6. *Sequence Diagram*

Menurut (Douglas, 2004) 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) 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 : 25).



Gambar II.10. Diagram Urutan

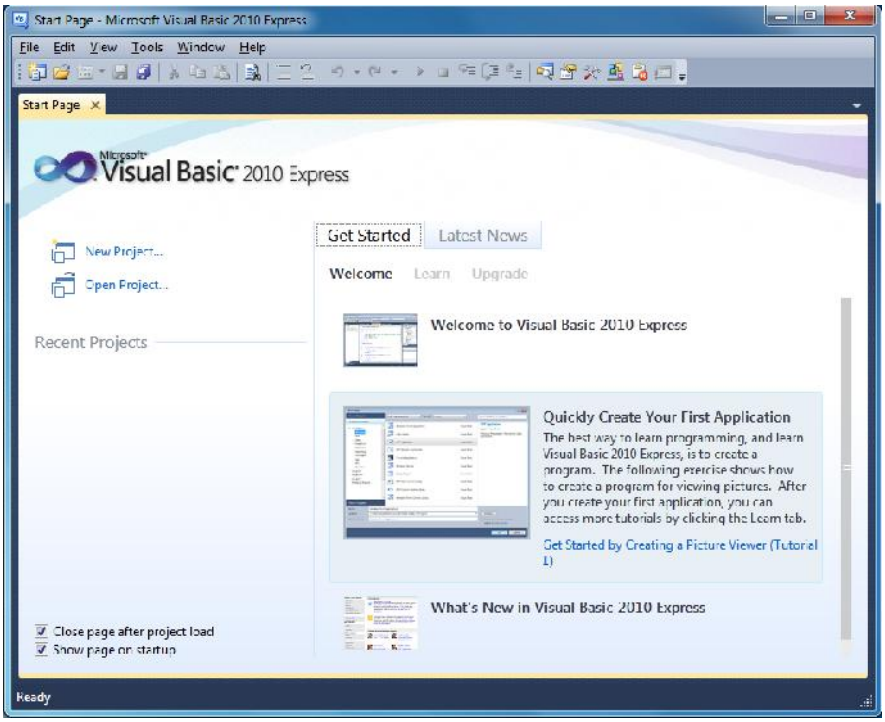
Sumber : (Prabowo Pudjo Widodo Dan Herlawati ;2011 : 26)

II.8. Bahasa Pemograman *Microsoft Visual Basic 2010*

Visual Basic adalah salah satu bahasa pemrograman berbasis dekstop yang dikeluarkan (diproduksi) oleh perusahaan perangkat lunak komputer terbesar yaitu *Microsoft*. *Visual Basic* merupakan salah satu bahasa pemrograman paling laris dan paling sukses di dunia. Menjadi pilihan berbagai kalangan tentunya *Visual Basic* memiliki berbagai hal yang patut dijadikan alasan, selain bahasa pemrograman yang sangat (paling) mudah dipelajari oleh berbagai kalangan baik awam maupun ahli, *Visual Basic* yang didukung penuh oleh proudsennya (*Microsoft*) selalu dikembangkan dan disesuaikan dengan kebutuhan zaman

seperti penyesuaian model pemrograman modern yang berbasis OOP (*Object Oriented Programming*)

Untuk melihat tampilan visual basic 2010 dapat dilihat pada gambar II.13. sebagai berikut :



Gambar II.11. Tampilan Utama Visual Basic 2010

Sumber : (A. M. Hirin ; 2011 : 2)

II.9. *MYSQL*

MySQL adalah program database yang mampu mengirim dan menerima data dengan sangat cepat dan multi user. *MySQL* memiliki dua bentuk lisensi, yaitu *free software* dan *shareware*. Penulis sendiri dalam menjelaskan buku ini menggunakan *MySQL* yang *free software* karena bebas menggunakan database ini untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensi,

yang berada di bawah lisensi GNU/ GPL (*General Public License*), yang dapat anda download pada alamat resminya *http://www.mysql.com*. (Wahana Komputer ; 2010 : 5)

Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basisdata yang telah ada sebelumnya. SQL (*Structured Query Language*) adalah seakuntansi konsep pengoperasian basisdata, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.