BAB II

LANDASAN TEORI

2.1. Konsep Dasar Sistem

Terdapat dua kelompok pendekatan di dalam mendefinisikan sistem, yaitu yang menekankan pada prosedurnya dan yang menekankan pada komponen atau elemennya. Pendekatan sistem yang lebih menekankan pada prosedur mendefinisikan sistem sebagai berikut ini [2] :

> *Suatu sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu.*

Pendekatan sistem yang lebih menekankan pada elemen atau komponennya mendefinisikan sistem sebagai berikut ini [2] :

> *Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu.*

2.1.1. Karakteristik Sistem

Suatu sistem mempunyai karakteristik atau sifat-sifat yang tertentu, yaitu :

1. Komponen Sistem (*Components*)

   Komponen-komponen sistem atau elemen-elemen sistem dapat berupa suatu subsistem atau bagian-bagian sistem, yang mempunyai sifat-sifat dari sistem untuk menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem keseluruhan.

2. Batas Sistem (*Boundary*)

   Batas sistem (*boundary*) merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batas suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

3. Lingkungan Luar Sistem (*Environments*)

   Lingkungan luar (*environments*) dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat menguntungkan dan merugikan sistem.

4. Penghubung Sistem (*Interface*)

   Penghubung (*interface*) merupakan media penghubung antara satu subsistem dengan subsistem lainnya sehingga memungkinkan sumber-sumber daya mengalir antara subsistem yang satu dengan yang lain.

5. Masukan Sistem (*Input*)

   Masukan (*input*) adalah energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintenance input*) dan masukan sinyal (*signal input*).

6. Keluaran Sistem (*Output*)

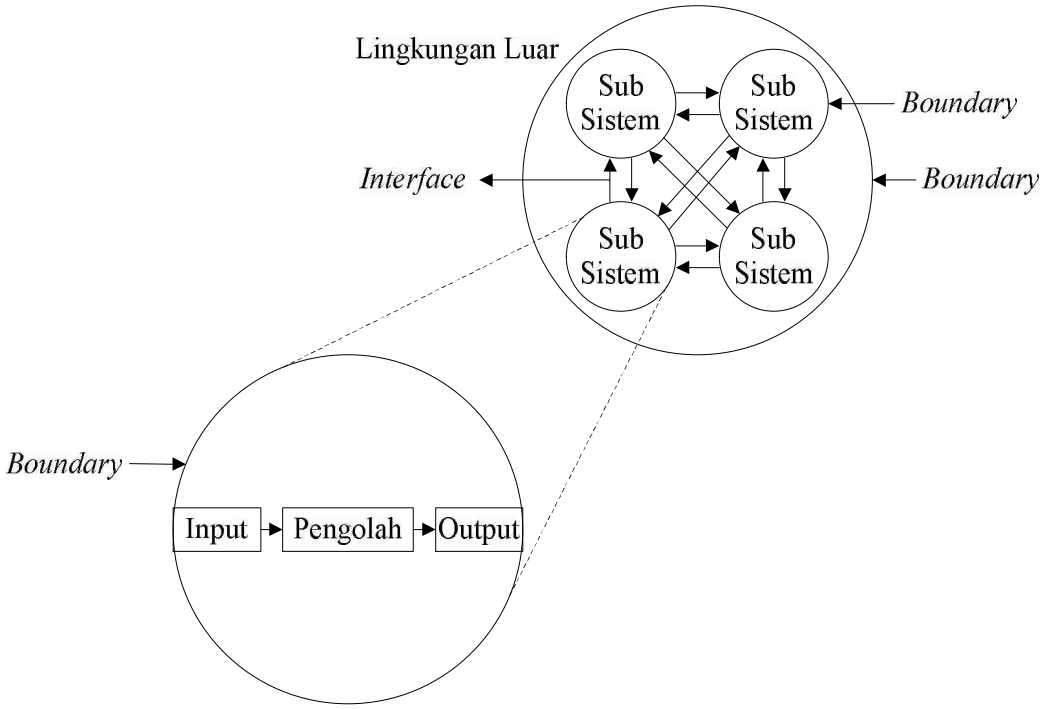
   Keluaran (*output*) adalah hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan. Keluaran dapat merupakan masukan untuk subsistem yang lain atau kepada supra sistem.

7. Pengolah Sistem (*Process*)

   Suatu sistem dapat mempunyai suatu bagian pengolah yang akan merubah masukan jadi keluaran.

8. Sasaran Sistem (*Objectives*)

   Sasaran dari sistem sangat menentukan sekali masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuannya.



Gambar 2.1 Karakteristik Sistem

2.1.2. Klasifikasi Sistem

Sistem dapat diklasifikasikan dari beberapa sudut pandangan, diantaranya adalah sebagai berikut :

1. Sistem diklasifikasikan sebagai sistem abstrak (*abstract system*) dan sistem fisik (*physical system*). Sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik, misalnya sistem teologia dan sistem fisik merupakan sistem yang ada secara fisik, misalnya sistem komputer.

2. Sistem diklasifikasikan sebagai sistem alamiah (*natural system*) dan sistem buatan manusia (*human made system*). Sistem alamiah adalah sistem yang terjadi melalui proses alam, tidak dibuat manusia, misalnya sistem perputaran bumi dan sistem buatan manusia adalah sistem yang dirancang oleh manusia.
3. Sistem diklasifikasikan sebagai sistem tertentu (*deterministic system*) dan sistem tak tentu (*probabilistic system*). Sistem tertentu beroperasi dengan tingkah laku yang sudah dapat diprediksi dan sistem tak tentu adalah sistem yang kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilitas.
4. Sistem diklasifikasikan sebagai sistem tertutup (*closed system*) dan sistem terbuka (*open system*). Sistem tertutup merupakan sistem yang tidak berhubungan dan tidak terpengaruh dengan lingkungan luarnya dan sistem terbuka adalah sistem yang berhubungan dan terpengaruh dengan lingkungan luarnya.

2.2. Konsep Dasar Informasi

Informasi (*information*) dapat didefinisikan sebagai berikut [2] :

Informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya.

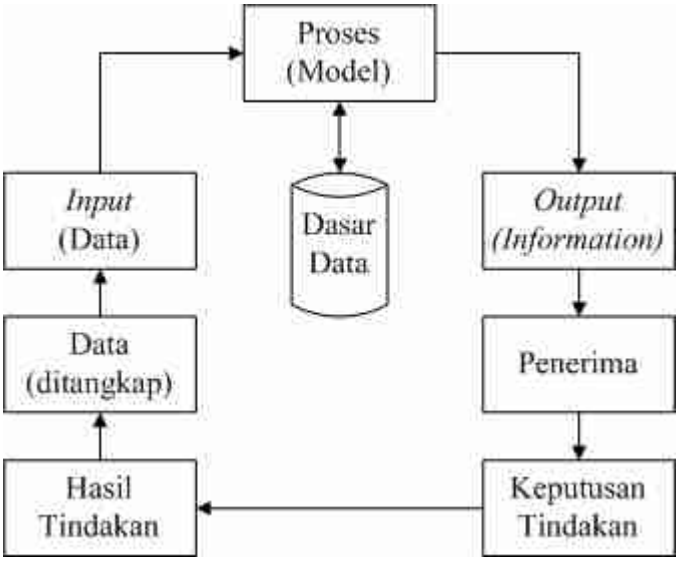
Sumber dari informasi adalah data. Data merupakan bentuk jamak dari bentuk tunggal datum atau data-item. Data adalah kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan nyata. Kejadian-kejadian (*event*) adalah sesuatu yang terjadi pada saat yang tertentu. Kesatuan nyata (*fact*

dan *entity*) adalah berupa suatu obyek nyata seperti tempat, benda dan orang yang betul-betul ada dan terjadi.

2.2.1. Siklus Informasi

Data merupakan bentuk yang masih mentah yang belum dapat bercerita banyak, sehingga perlu diolah lebih lanjut. Data diolah melalui suatu model untuk dihasilkan informasi.

Data yang diolah untuk menghasilkan informasi menggunakan suatu model proses yang tertentu. Data yang diolah melalui suatu model menjadi informasi, penerima kemudian menerima informasi tersebut, membuat suatu keputusan dan melakukan tindakan, yang berarti menghasilkan suatu tindakan yang lain yang akan membuat sejumlah data kembali. Data tersebut akan ditangkap sebagai *input*, diproses kembali lewat suatu model dan seterusnya membentuk siklus. Siklus ini disebut dengan siklus informasi (*information cycle*) atau disebut juga dengan siklus pengolahan data (*data processing cycles*).



Gambar 2.2 Siklus Informasi

2.2.2. Kegunaan Informasi

Ada 4 faktor utama yang berhubungan dengan kegunaan informasi :

1. Kualitas informasi (*information quality*)

   Kualitas dari suatu informasi tergantung dari 4 hal, yaitu informasi harus :

   a. Akurat (*accurate*) dan presisi (*precision*)

      Akurat dalam menampilkan informasi dan presisi dalam detail informasi yang diberikan.

   b. Kelengkapan (*completeness*)

      Informasi yang tersedia cukup lengkap untuk setiap *user* dan situasi.

   c. Umur (*age*) dan ketepatan waktu (*timeliness*)

      Umur berarti lamanya waktu dalam meng-*update* informasi dan ketepatan waktu berarti menyediakan informasi secepat mungkin pada saat dibutuhkan sehingga berguna.

   d. Sumber (*source*)

      Orang atau organisasi yang menghasilkan informasi.

2. Aksesibilitas informasi (*information accessibility*)

   a. Ketersediaan (*availability*)

      Memberikan informasi kepada yang membutuhkan. Informasi dapat diakses oleh yang membutuhkan.

   b. Keabsahan (*admissibility*)

      Keabsahan (boleh atau tidak boleh dipakai) informasi tergantung pada hukum, peraturan atau budaya pada saat tertentu.

3. Presentasi informasi (*information presentation*)
    a. Tingkatan (*level of summarization*)

        Perbandingan antara data asli dengan yang ditampilkan. Manipulasi data hingga tingkatan yang sesuai, semakin sederhana semakin baik.

    b. Format

        Bentuk dimana informasi ditampilkan ke *user*. Manipulasi data ke dalam bentuk yang sesuai.
4. Keamanan informasi (*information security*)
    a. Batasan akses (*access restriction*)

        Prosedur dan teknik mengontrol *user* yang boleh atau tidak mengakses data pada situasi tertentu.

        Penggunaan password atau teknik lain untuk mencegah *user* yang tidak berhak.

    b. Enkripsi (*encryption*)

        Konversi data ke bentuk tertentu sehingga tidak dapat dibaca oleh *user* yang tidak berhak.

2.2.3. Nilai Informasi

Nilai dari informasi (*value of information*) ditentukan dari dua hal, yaitu manfaat dan biaya mendapatkannya. Suatu informasi dikatakan bernilai bila manfaatnya lebih efektif dibandingkan dengan biaya mendapatkannya.

Sebagian besar informasi tidak dapat persis ditaksir keuntungannya dengan satuan nilai uang, tetapi dapat ditaksir nilai efektivitasnya. Pengukuran nilai

informasi biasanya dihubungkan dengan analisis *cost effectiveness* atau *cost benefit*.

2.3. Konsep Dasar Sistem Informasi

Informasi dapat diperoleh dari sistem informasi (*information systems*) atau disebut juga dengan *processing systems* atau *information processing systems* atau *information-generating systems*. Sistem informasi didefenisikan oleh Robert A. Leitch dan K. Roscoe Davis sebagai berikut [2] :

> *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.*

2.3.1. Komponen Sistem Informasi

John Burch dan Gary Grudnitski mengemukakan bahwa sistem informasi terdiri dari komponen-komponen yang disebutnya dengan istilah blok bangunan (*building block*), yaitu blok masukan (*input block*), blok model (*model block*), blok keluaran (*output block*), blok teknologi (*technology block*), blok basis data (*database block*), dan blok kendali (*controls block*). Sebagai suatu sistem, keenam blok tersebut masing-masing saling berinteraksi satu dengan yang lainnya membentuk satu kesatuan untuk mencapai sasarannya.

1. Blok Masukan

   Input mewakili data yang masuk ke dalam sistem informasi. Input disini termasuk metode-metode dan media untuk menangkap data yang akan dimasukkan, yang dapat berupa dokumen-dokumen dasar.

2. Blok Model

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

3. Blok Keluaran

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

4. Blok Teknologi

   Teknologi digunakan untuk menerima input, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran dan membantu pengendalian dari sistem secara keseluruhan. Teknologi terdiri dari 3 bagian utama, yaitu teknisi (*humanware* atau *brainware*), perangkat lunak (*software*), dan perangkat keras (*hardware*).

5. Blok Basis Data

   Basis data (*database*) merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan di perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya. Data perlu disimpan di dalam basis data untuk keperluan penyediaan informasi lebih lanjut.

6. Blok Kendali

   Beberapa pengendalian perlu dirancang dan diterapkan untuk meyakinkan bahwa hal-hal yang dapat merusak sistem dapat dicegah ataupun bila terlanjur terjadi kesalahan-kesalahan dapat langsung cepat diatasi.

Pengelompokan komponen-komponen sistem informasi berbasis komputer adalah sebagai berikut :

1. Perangkat keras (*hardware*)

   Hardware merupakan peralatan fisik yang dapat digunakan untuk mengumpulkan, memasukkan, memproses, menyimpan, dan mengeluarkan hasil pengolahan data dalam bentuk informasi.

2. Perangkat lunak (*software*)

   Software adalah kumpulan dari program-program yang digunakan untuk menjalankan aplikasi tertentu pada komputer.

3. Manusia (*brainware*)

   Brainware dalam sistem informasi berperan sebagai pemberi dan pengguna informasi.

4. Prosedur (*procedure*)

   Prosedur adalah rangkaian aktivitas atau kegiatan yang dilakukan secara berulang-ulang dengan cara yang sama.

5. Basis data (*database*)

   Database merupakan kumpulan data-data yang tersimpan di dalam media penyimpanan di suatu perusahaan (arti luas) atau di dalam komputer (arti sempit).

6. Jaringan komunikasi (*communication network*)

   Jaringan komunikasi saat ini menghubungkan beberapa daratan dan lautan untuk memindahkan data dalam jumlah besar.

2.3.2. Pengembangan Sistem Informasi

Pengembangan sistem (*systems development*) dapat berarti menyusun suatu sistem yang baru untuk menggantikan sistem yang lama secara keseluruhan atau memperbaiki sistem yang ada. Sewaktu melakukan proses pengembangan sistem, beberapa prinsip harus tidak boleh dilupakan. Prinsip-prinsip ini adalah sebagai berikut :

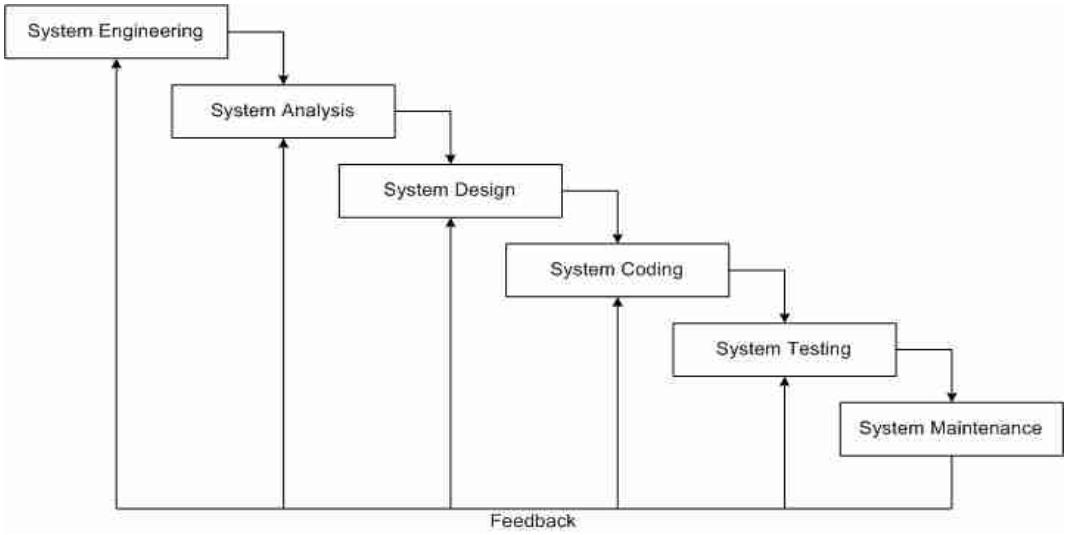
1. Sistem yang dikembangkan adalah untuk manajemen
2. Sistem yang dikembangkan adalah investasi modal yang besar
3. Sistem yang dikembangkan memerlukan orang yang terdidik
4. Tahapan kerja dan tugas-tugas yang harus dilakukan dalam proses pengembangan sistem
5. Proses pengembangan sistem tidak harus urut
6. Jangan takut membatalkan proyek
7. Dokumentasi harus ada untuk pedoman dalam pengembangan sistem

Proses pengembangan sistem melewati beberapa tahapan dari mulai sistem itu direncanakan sampai dengan sistem tersebut diterapkan, dioperasikan, dan dipelihara. Daur atau siklus hidup dari pengembangan sistem merupakan suatu bentuk yang digunakan untuk menggambarkan tahapan utama dan langkah-langkah di dalam tahapan tersebut dalam proses pengembangannya.

Pengembangan sistem yang digunakan yaitu *classsic life style* atau yang lebih dikenal dengan istilah *waterfall*. Pengembangan sistem menurut A. Ziya Aktas (1987) adalah sebagai berikut :

1. Rekayasa sistem (*system engineering*), merupakan tahap awal dalam pengembangan sistem yaitu dengan menetapkan segala hal yang diperlukan dalam pelaksanaan pengembangan sistem dan menentukan apakah sistem benar-benar dibutuhkan atau tidak. Tahap-tahap yang digunakan yaitu dengan diadakannnya wawancara, observasi, dan studi literatur.
2. Analisis (*analysis*), merupakan tahap menganalisis kebutuhan sistem seperti mendefinisikan kembali masalah, memahami kebutuhan-kebutuhan pemakai dan hambatan-hambatan pada sistem baru, dan membuat model logika dari pemecahan yang direkomendasi. Adapun metode analisis yang digunakan adalah metode analisis berbasis objek.
3. Desain (*Design*), yaitu tahap setelah analisis dari siklus pengembangan sistem, pendefinisian dari kebutuhan-kebutuhan fungsional, persiapan untuk rancang bangun implementasi, dan menggambarkan bagaimana suatu sistem dibentuk.
4. Penulisan Program (*Coding*), adalah tahap menterjemahkan hasil analisis ke dalam bahasa pemrograman yang telah ditentukan.
5. Pengujian (*Testing*), tahap dimana melakukan pengujian terhadap sistem yang telah dibangun.
6. Pemeliharaan (*Maintenance*), tahap ini merupakan tahap akhir dimana sistem yang sudah selesai dapat mengalami perubahan atau penambahan sesuai dengan keinginan konsumen.

Untuk lebih jelasnya susunan metode *waterfall* dapat dilihat pada gambar dibawah ini :



Gambar 2.3 Metodologi *Waterfall*

2.4. Basis Data *(Database)*

Basis data terdiri atas dua kata, yaitu basis dan data. Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang/berkumpul. Sedangkan data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, hewan, peristiwa, konsep, keadaan, dan sebagainya, yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya. [1]

Basis data (*database*) sendiri dapat didefinisikan sebagai berikut :

> *Basis data (database) merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan di perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya.* [2]

Prinsip utama dalam basis data adalah pengaturan data/arsip dan tujuan utamanya adalah kemudahan dan kecepatan dalam pengambilan kembali data/arsip yang menggunakan media penyimpanan elektronis seperti *disk* (disket atau *harddisk*). Basis data dikelola/ditangani melalui perantaraan alat/mesin pintar elektronis yang kita kenal sebagai komputer.

Basis data bukan hanya sekedar penyimpanan data secara elektronis dengan bantuan komputer. Artinya, tidak semua bentuk penyimpanan data secara elektronis bisa disebut basis data. Yang sangat ditonjolkan dalam basis data adalah pengaturan/pemilahan/pengelompokkan/pengorganisasian data yang akan kita simpan sesuai fungsi/jenisnya. Pemilahan/pengelompokkan/pengorganisasian ini dapat berbentuk sejumlah file/tabel terpisah atau dalam bentuk pendefinisian kolom-kolom/*field-field* data dalam setiap file/tabel.

Operasi-operasi dasar yang dapat dilakukan berkenaan dengan basis data dapat meliputi pembuatan basis data baru (*create database*), penghapusan basis data (*drop database*), pembuatan file/tabel baru ke suatu basis data (*create table*), penghapusan file/tabel dari suatu basis data (*drop table*), penambahan/pengisian data baru ke sebuah file/tabel di sebuah basis data (*insert*), pengambilan data dari sebuah file/tabel (*retrieve/search*), pengubahan data dari sebuah file/tabel (*update*), dan penghapusan data dari sebuah file/tabel (*delete*).

2.4.1. *Database Management System* (DBMS)

Pengelolaan basis data secara fisik tidak dilakukan oleh pemakai secara langsung, tetapi ditangani oleh sebuah Perangkat Lunak (Sistem) yang khusus/spesifik. Perangkat lunak inilah (disebut DBMS) yang akan menentukan bagaimana data diorganisasi, disimpan, diubah dan diambil kembali. Ia juga menerapkan mekanisme pengamanan data, pemakaian data secara bersama, pemaksaan keakuratan/konsistensi data, dan sebagainya.

Perangkat lunak yang termasuk DBMS seperti dBase III+, dBase IV, FoxBase, Rbase, MS-Access dan Borland-Paradox (untuk kelas sederhana) atau Borland-Interbase, MS-SQLServer, CA-Open Ingres, Oracle, Informix dan Sybase (untuk kelas kompleks/berat).

2.4.2. Tujuan Basis Data

Tujuan awal dan utama dalam pengelolaan data dalam sebuah basis data adalah agar dapat menemukan kembali data (yang dicari) dengan mudah dan cepat. Di samping itu, pemanfaatan basis data untuk pengelolaan data, juga memiliki tujuan-tujuan lain.

Secara lebih lengkap, pemanfaatan basis data dilakukan untuk memenuhi sejumlah tujuan (objektif) seperti berikut ini:

a. Kecepatan dan kemudahan (*speed*)
b. Efisiensi ruang penyimpanan (*space*)
c. Keakuratan (*accuracy*)
d. Ketersediaan (*availability*)
e. Kelengkapan (*completeness*)
f. Keamanan (*security*)
g. Kebersamaan pemakaian (*sharability*)

2.4.3. Pemakai (*User*) Basis Data

Ada beberapa jenis/tipe pemakai suatu sistem basis data yang dibedakan berdasarkan cara mereka berinteraksi terhadap sistem:

a. Programmer Aplikasi

   Pemakai yang berinteraksi dengan basis data melalui *Data Manipulation Language* (DML), yang disertakan (*embedded*) dalam program yang ditulis dalam bahasa pemrograman induk (seperti C, Pascal, Cobol, dan lain-lain).

b. User Mahir (*Casual User*)

   Pemakai yang berinteraksi dengan sistem tanpa menulis modul program. Mereka menyatakan *query* (untuk akses data) dengan bahasa *query* yang telah disediakan oleh suatu DBMS.

c. User Umum (*End User Naive User*)

   Pemakai yang berinteraksi dengan sistem basis data melalui pemanggilan satu program aplikasi permanen (*executable program*) yang telah ditulis/disediakan sebelumnya.

d. User Khusus (*Specialized User*)

   Pemakai yang menulis aplikasi basis data non konvensional, tetapi untuk keperluan-keperluan khusus, seperti untuk aplikasi AI, Sistem Pakar, Pengolahan Citra, dan lain-lain, yang bisa saja mengakses basis data dengan/tanpa DBMS yang bersangkutan.

Untuk sebuah sistem basis data yang *stand-alone*, maka pada suatu saat hanya ada satu pemakai yang dapat bekerja. Sedang untuk sistem basis data dalam jaringan, maka pada suatu saat ada banyak pemakai yang dapat berhubungan (menggunakan) basis data yang sama.

2.5. Alat Pemodelan Sistem

Alat-alat pemodelan sistem informasi sangat dibutuhkan dalam proses analisis dan perancangan sistem. Alat-alat pemodelan sistem informasi terdiri dari:

a. Bagan Alir Dokumen (*Document Flowmap*)

   Bagan alir dokumen (*document flowmap*) atau disebut juga bagan alir formulir (*form flowmap*) atau *paperwork flowmap* merupakan bagan alir yang menunjukkan arus dari laporan dan formulir termasuk tembusan-tembusannya.

b. *Entity-Relationship Diagram* (ERD)

   ERD adalah diagram yang memperlihatkan entitas-entitas yang terlibat dalam suatu sistem serta hubungan-hubungan (*relation*) antar entitas. Komponen-komponen pembentuk model ERD yaitu:

   1. Entitas (*entity*)

      Entitas merupakan individu yang mewakili sesuatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain. Entitas dapat berupa orang, tempat, benda, peristiwa atau konsep yang bisa memberikan atau mengandung informasi.

   2. Atribut (*attributes/properties*)

      Setiap entitas pasti memiliki atribut yang mendeskripsikan karakteristik (*properti*) dari entitas tersebut.

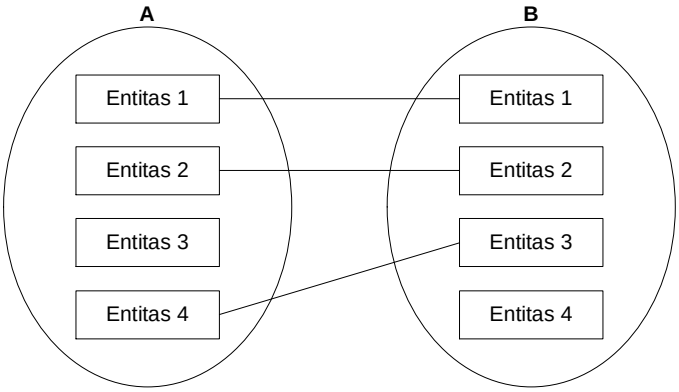
   3. Relasi (*relationship*)

      Relasi menunjukkan adanya hubungan di antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda.

4. Kardinalitas/derajat

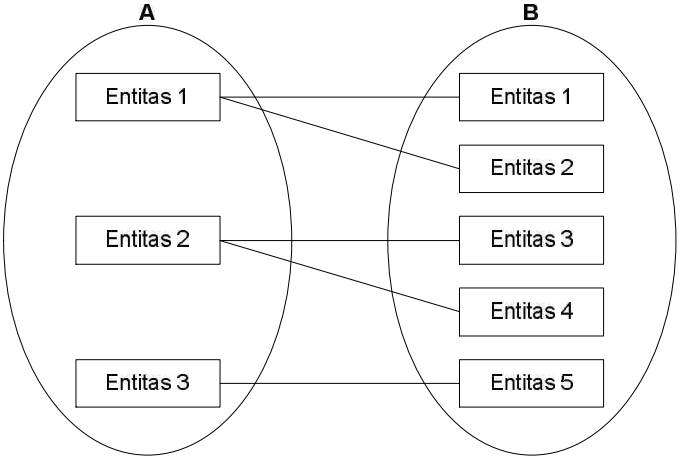
Kardinalitas relasi menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Kardinalitas relasi yang terjadi di antara dua himpunan entitas dapat berupa:

a. Satu ke satu (*one to one*)



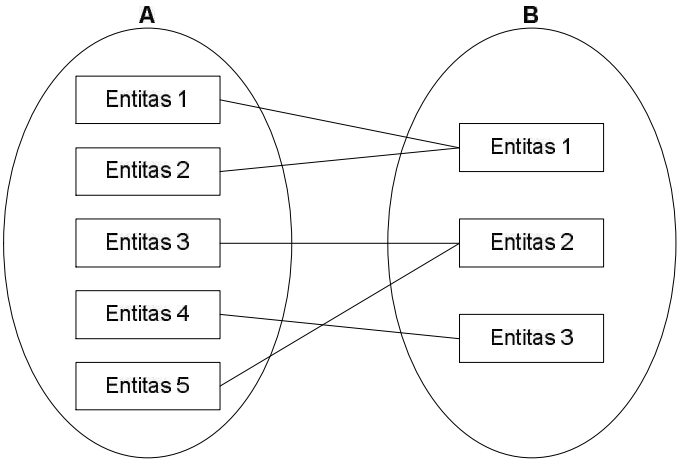
Gambar 2.4 Kardinalitas Relasi Satu ke Satu

b. Satu ke banyak (*one to many*)



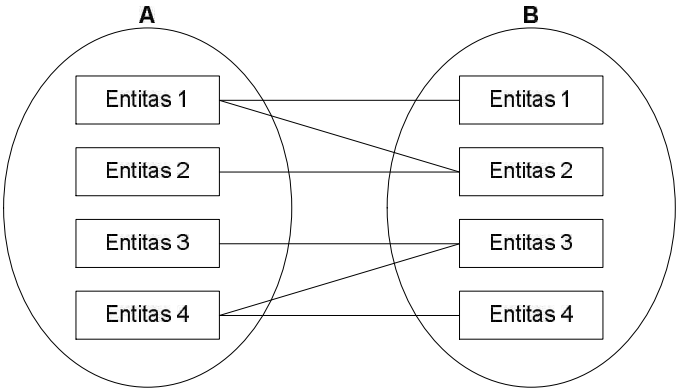
Gambar 2.5 Kardinalitas Relasi Satu ke Banyak

c. Banyak ke satu (*many to one*)



Gambar 2.6 Kardinalitas Relasi Banyak ke Satu

d. Banyak ke banyak (*many to many*)



Gambar 2.7 Kardinalitas Relasi Banyak ke Banyak

5. Kunci (*key*)

   Sebuah atribut atau set atribut yang nilainya mengidentifikasikan entitas secara unik dalam set entitas.

c. Diagram Konteks (*Context Diagram*)

Diagram konteks merupakan diagram aliran data pada tingkat paling atas yang merupakan penggambaran yang berfungsi untuk memperlihatkan interaksi/hubungan langsung antara sistem dengan lingkungannya. Diagram

konteks menggambarkan sebuah sistem berupa sebuah proses yang berhubungan dengan satu atau beberapa entitas/*entity*.

d. *Unified Modelling Language* (UML)

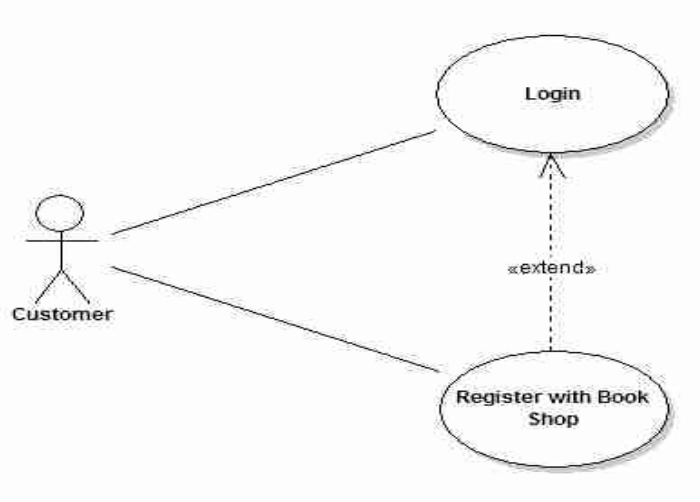
Secara umum UML merupakan ‘bahasa’ untuk visualisasi, spesifikasi, konstruksi, serta dokumentasi. Dalam kerangka visualisasi, para pengembang menggunakan UML sebagai suatu cara untuk mengkomunikasikan idenya kepada para pemrogram serta calon pengguna system/perangkat lunak. Dengan adanya ‘bahasa’ yang bersifat standar, komunikasi perancangan dengan pemrogram (lebih tepat lagi: komunikasi antar anggota kelompok pengembang) serta calon pengguna diharapkan menjadi mulus. Dalam kerangka spesifikasi, UML menyediakan model-model yang tepat, tidak mendua-arti (*ambigu*), serta lengkap. Secara khusus, UML menspesifikasikan langkah-langkah penting dalam pengambilan keputusan analisis, perancangan, serta implementasi dalam sistem yang sangat bernuansa perangkat lunak (*software intensive system*).

Pendekatan yang dilakukan dengan metode UML (*Unified Modeling Language*) penulis membuat beberapa abstrak konsep dasar dari UML, yaitu terdiri dari *structural classification*, *dynamic behavior*, dan *model management*. Untuk penentuan pertama dalam pembuatan UML penulis mendefenisikan dalam diagram-diagram yang nantinya diperlukan, yaitu :

1. *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan pada “apa” yang diperbuat system, dan bukan “bagaimana”. Sebuah *use case diagram* mempresentasikan sebuah interaksi antara actor dengan system. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke system, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah actor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan system untuk melakukan pekerjaan-pekerjaan tertentu.

Contoh penggunaan *use case diagram* :



Gambar 2.8 Contoh Penggunaan *Use Case Diagram*

2. *Class Diagram*

Class merupakan sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan

(atribut/properti) suatu system, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metode/fungsi).

Class diagram meggambarkan struktur dari deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan asosiasi, dan lain-lain.

Class memiliki tiga area pokok :

1. Nama (dan stereotype)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

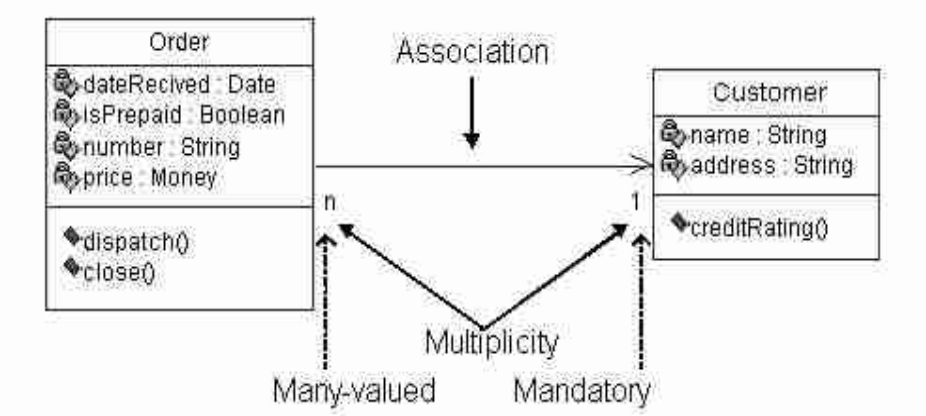
a. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
b. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
c. *Public*, dapat dipanggil oleh siapa saja.

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memilikimetoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*. Sesuai dengan perkembangan *class* model, *class* dapat dikelompokkan menjadi *package*. Kita juga dapat membuat diagram yang terdiri atas *package*.

Hubungan Antar *Class* terdiri dari :

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas..").
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

Contoh penggunaan *class diagram* :



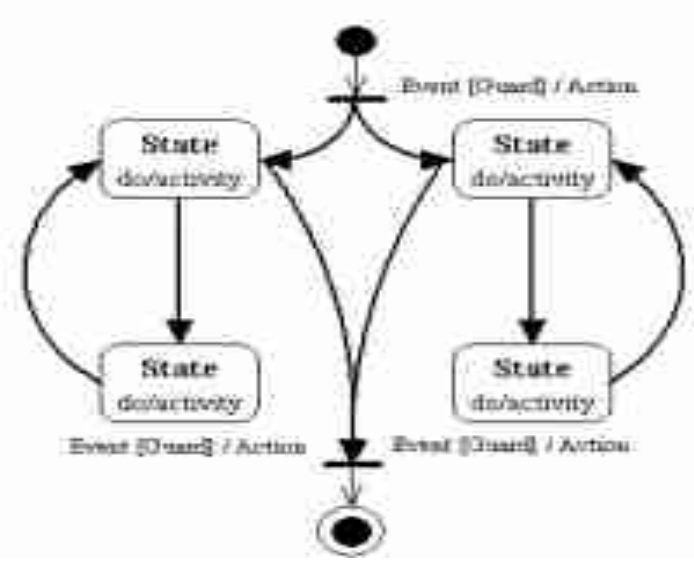
Gambar 2.9 Contoh Penggunaan *Class Diagram*

3. *Statechart Diagram*

Statechart diagram menggambarakan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart* diagram).

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar state umumnya memiliki kondisi guard yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari event tertentu dituliskan dengan diawali garis miring.

Contoh penggunaan *statechart diagram* :



Gambar 2.10 Contoh Penggunaan *Statechart Diagram*

4. *Activity Diagram*

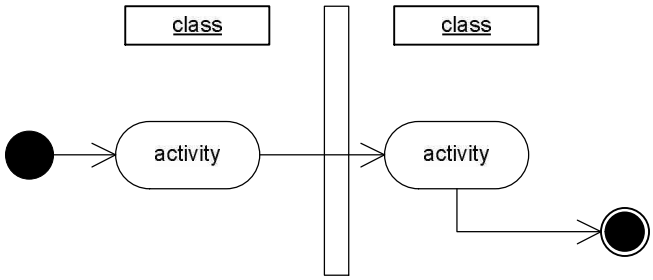
Menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang. Bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka terakhir. *Activity diagram* juga dapat menggambarkan proses *pararel* yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, dimana sebagian besar *state* adalah *action* dan sebagai besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behavior internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

Contoh penggunaan *Activity diagram* :



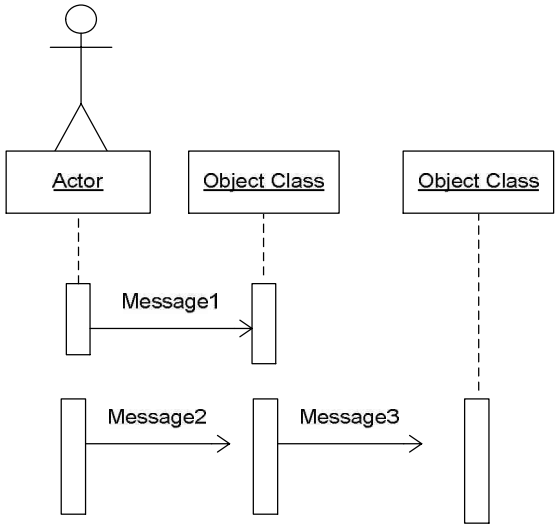
Gambar 2.11 Contoh penggunaan *Activity diagram*

5. *Sequence Diagram*

Sequence diagram mengambarkan interaksi antar objek didalam dan disekitar sistem (termasuk pengguna, display, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *sequence diagram* terdiri dari antar dimensi vertical (waktu) dan dimensi horizontal (objek-objek yang terkait).

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

Contoh Penggunaan *Sequence diagram :*

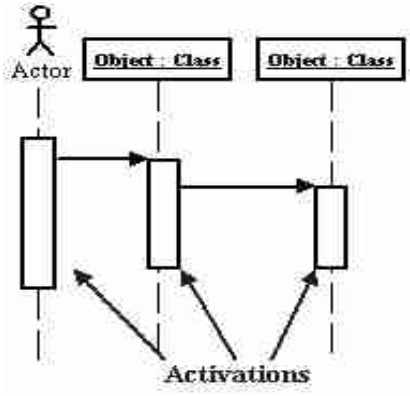


Gambar 2.12 Contoh Penggunaan *Sequence Diagram*

6. *Collaboration Diagram*

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence number*, dimana *message* dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama.

Contoh Penggunaa *Collaboration diagram* :



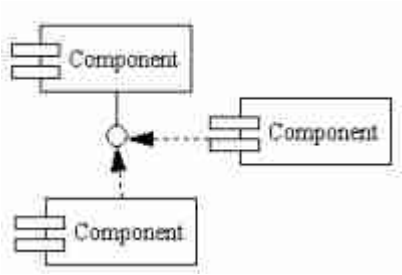
Gambar 2.13 Contoh Penggunaan *Collaboration Diagram*

7. *Component Diagram*

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) diantaranya komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa class dan/atau *package*, tetapi dapat juga dari komponen-komponen yang lebih kecil.

Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

Contoh penggunaan *Component diagram* :

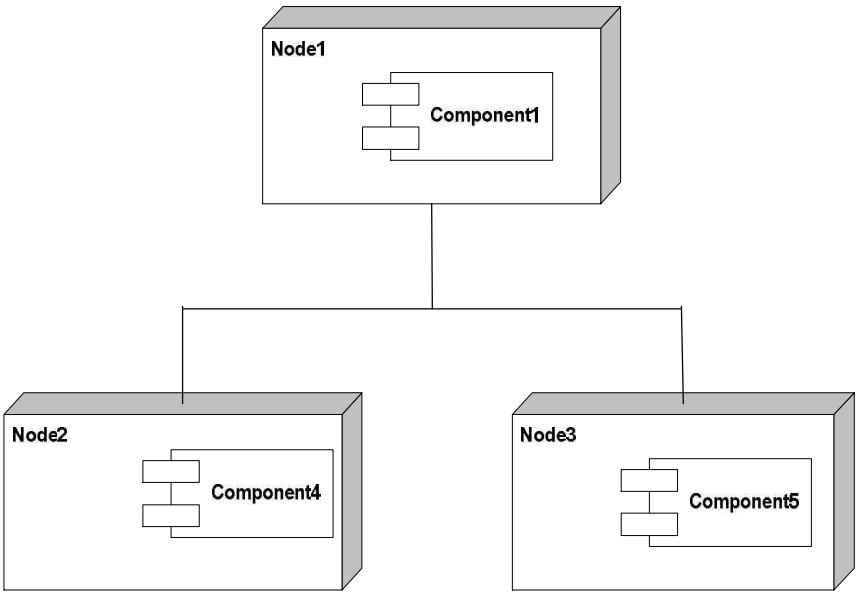


Gambar 2.14 Contoh Penggunaan *Component Diagram*

8. *Deployment Diagram*

Deployment diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, dimana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal yang lain bersifal fisikal.

Contoh Penggunaan *Deployment diagram* :



Gambar 2.15 Contoh Penggunaan *Deployment Diagram*

e. Spesifikasi Proses (*Process Specification* (PSPEC))

Spesifikasi proses (PSPEC) digunakan untuk menggambarkan semua proses model aliran yang nampak pada tingkat akhir penyaringan. Kandungan dari spesifikasi proses dapat termasuk teks naratif, gambaran bahasa desain program (*Programme Design Language* (PDL)) dari algoritma proses, persamaan matematika, tabel, diagram, atau bagan.

f. Kamus Data (*Data Dictionary*)

Kamus data (*data dictionary*) atau disebut juga dengan istilah *systems data dictionary* adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Dengan menggunakan kamus data, analis sistem dapat mendefinisikan data yang mengalir di sistem dengan lengkap. Kamus data harus memuat hal-hal berikut ini:

1. Nama arus data
2. Alias atau nama lain dari data dapat dituliskan bila nama lain ini ada. Alias perlu ditulis karena data yang sama mempunyai nama yang berbeda untuk orang atau departemen satu dengan yang lainnya.
3. Bentuk data, dapat berupa dokumen dasar atau formluir, dokumen hasil cetakan komputer, laporan tercetak, tampilan di layar monitor, variabel, parameter, dan *field*.
4. Arus data, menunjukkan dari mana data mengalir dan ke mana data akan menuju.
5. Penjelasan, dapat diisi dengan keterangan-keterangan tentang arus data tersebut.
6. Periode, menunjukkan kapan terjadinya arus data.
7. Volume, digunakan untuk mengidentifikasikan besarnya simpanan luar yang akan digunakan, kapasitas dan jumlah dari alat input, alat pemroses dan alat output.
8. Struktur data, menunjukkan arus data yang dicatat di kamus data terdiri dari item-item data apa saja.

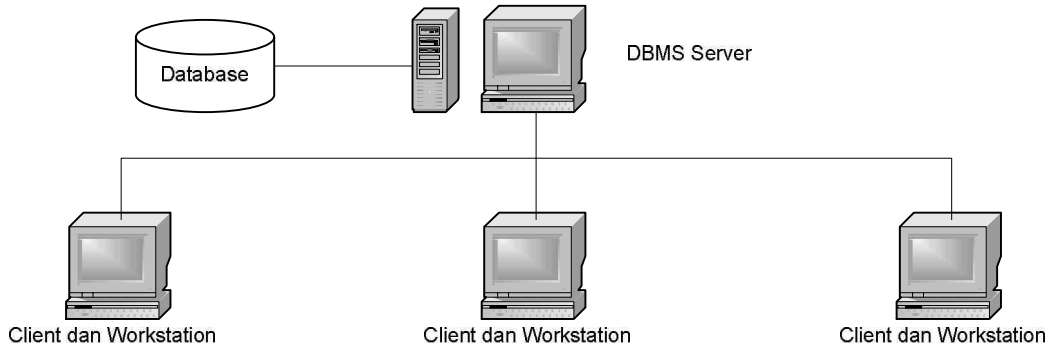
g. Skema Relasi

Skema relasi adalah untuk presentasi atribut-atribut dari *entity* yang terdapat dalam sistem dan hubungan antar *entity* pada model ERD. Skema relasi merupakan turunan dari ERD.

2.6. Sistem *Client-Server*

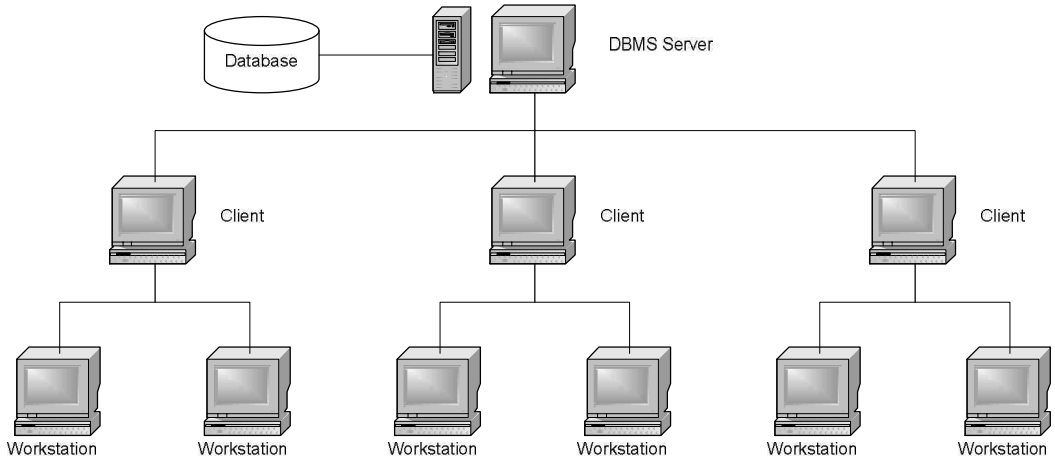
Sistem *client-server* atau disebut juga sistem tersentralisasi diterapkan pada sebuah sistem jaringan. Sistem *client-server* ini ditujukan untuk mengatasi kelemahan-kelemahan yang terdapat pada sistem sebelumnya. Sistem ini terdiri atas dua komponen (mesin) utama, yaitu *client* dan *server*. *Client* berisi aplikasi basis data dan *server* berisi DBMS dan basis data. Setiap aktivitas yang dikehendaki para pemakai akan lebih dahulu ditangani oleh *client*. *Client* selanjutnya mengupayakan agar semua proses "sebisa mungkin" ditangani sendiri. Bila ada proses yang harus melibatkan data yang tersimpan pada basis data, barulah *client* mengadakan hubungan dengan *server*.

Pada sistem *client-server* untuk memenuhi kebutuhan, *client* akan mengirimkan *message* (perintah) *query* pengambilan data. Selanjutnya, *server* yang menerima *message* tersebut akan menjalankan *query* tersebut dan hasilnya akan dikirimkan kembali ke *client*. Dengan begitu transfer datanya jauh lebih efisien. Adapun bentuk dari sistem *client-server* yang sederhana adalah sebagai berikut:



Gambar 2.16 Sistem *Client-Server* Sederhana

Di samping bentuk *client-server* sederhana, terdapat pula bentuk *client-server* yang lebih kompleks yang digambarkan sebagai berikut:



Gambar 2.17 Sistem *Client-Server* Kompleks

Dari kedua gambar diatas, dapat dilihat adanya dua macam implementasi sistem client-server. Bentuk yang sederhana dapat diterapkan pada sebuah jaringan komputer lokal (LAN) dimana fungsi client (untuk menangani sebagian proses pengolahan data seperti perhitungan, perulangan, perbandingan, dan lain-lain) dan fungsi workstation (untuk menangani interaksi) disatukan.

2.7. *Software* Pendukung

Sistem informasi kepegawaian ini menggunakan Java sebagai antar muka dan MySQL sebagai *database*-nya.

2.7.1. Java

Pada 1991, sekelompok insinyur Sun dipimpin oleh Patrick Naughton dan James Gosling ingin merancang bahasa komputer untuk perangkat konsumer seperti cable TV Box. Karena perangkat tersebut tidak memiliki banyak memori,

bahasa harus berukuran kecil dan mengandung kode yang liat. Juga karena manufaktur – manufaktur berbeda memilih processor yang berbeda pula, maka bahasa harus bebas dari manufaktur manapun. Proyek diberi nama kode ”Green”.

Kebutuhan untuk fleksibilitas, kecil, liat dan kode yang netral terhadap platform mengantar tim mempelajari implementasi Pascal yang pernah dicoba. Niklaus Wirth, pencipta bahasa Pascal telah merancang bahasa portabel yang menghasilkan intermediate code untuk mesin hipotesis. Mesin ini sering disebut dengan mesin maya (virtual machine). Kode ini kemudian dapat digunakan di sembarang mesin yang memiliki interpreter. Proyek Green menggunakan mesin maya untuk mengatasi isu utama tentang netral terhadap arsitektur mesin.

Karena orang – orang di proyek Green berbasis C++ dan bukan Pascal maka kebanyakan sintaks diambil dari C++, serta mengadopsi orientasi objek dan bukan prosedural. Mulanya bahasa yang diciptakan diberi nama ”Oak” oleh James Gosling yang mendapat inspirasi dari sebuah pohon yang berada pada seberang kantornya, namun dikarenakan nama Oak sendiri merupakan nama bahasa pemrograman yang telah ada sebelumnya, kemudian SUN menggantinya dengan JAVA. Nama JAVA sendiri terinspirasi pada saat mereka sedang menikmati secangkir kopi di sebuah kedai kopi yang kemudian dengan tidak sengaja salah satu dari mereka menyebutkan kata JAVA yang mengandung arti asal bijih kopi. Akhirnya mereka sepakat untuk memberikan nama bahasa pemrograman tersebut dengan nama Java.

Java telah mengakomodasi hampir seluruh fitur penting bahasa – bahasa pemrograman yang ada semenjak perkembangan komputasi modern manusia :

1. Dari SIMULA, bahasa pada tahun 65-an, bahasa yang paling mempengaruhi Java sekaligus C++. Dari bahasa ini diadopsi bentukan – bentukan dasar dari pemrograman berorientasi objek.
2. Dari LISP – bahasa tahun 55-an. Diadopsi fasilitas garbage collection, serta kemampuan untuk meniru generic list processing, meski fasilitas ini jarang yang memanfaatkannya.
3. Dari Algol – bahasa pada tahun 60-an, diambil struktur kendali yang dimilikinya.
4. Dari C++, diadopsi sintaks, sebagian semantiks dan exception handling
5. Dari bahasa Ada, diambil strongly type, dan exception handling.
6. Dari Objective C, diambil fasilitas interface.
7. Dari bahasa SmallTalk, diambil pendekatan single-root class hiérarchie, dimana objek adalah satu kesatuan hirarki pewarisan
8. Dari bahasa Eiffel, fasilitas assertion yang mulai diterapkan di sebagian JDK 1.4

Berdasarkan white paper resmi dari SUN, Java memiliki karakteristik berikut :

1. Sederhana (Simple)

   Bahasa pemrograman Java menggunakan Sintaks mirip dengan C++ namun sintaks pada Java telah banyak diperbaiki terutama menghilangkan penggunaan pointer yang rumit dan multiple inheritance. Java juga menggunakan automatic memory allocation dan memory garbage collection.

2. Berorientasi objek (Object Oriented)

   Java mengunakan pemrograman berorientasi objek yang membuat program dapat dibuat secara modular dan dapat dipergunakan kembali. Pemrograman berorientasi objek memodelkan dunia nyata kedalam objek dan melakukan interaksi antar objek-objek tersebut.

3. Terdistribusi (Distributed)

   Java dibuat untuk membuat aplikasi terdistribusi secara mudah dengan adanya libraries networking yang terintegrasi pada Java.

4. Interpreted

   Program Java dijalankan menggunakan interpreter yaitu Java Virtual Machine (JVM). Hal ini menyebabkan source code Java yang telah dikompilasi menjadi Java bytecodes dapat dijalankan pada platform yang berbeda-beda.

5. Robust

   Java mempuyai reliabilitas yang tinggi. Compiler pada Java mempunyai kemampuan mendeteksi error secara lebih teliti dibandingkan bahasa pemrograman lain. Java mempunyai runtime-Exception handling untuk membantu mengatasi error pada pemrograman.

6. Secure

   Sebagai bahasa pemrograman untuk aplikasi internet dan terdistribusi, Java memiliki beberapa mekanisme keamanan untuk menjaga aplikasi tidak digunakan untuk merusak sistem komputer yang menjalankan aplikasi tersebut.

7. Architecture Neutral

   Program Java merupakan platform independent. Program cukup mempunyai satu buah versi yang dapat dijalankan pada platform berbeda dengan Java Virtual Machine.

8. Portable

   Source code maupun program Java dapat dengan mudah dibawa ke platform yang berbeda-beda tanpa harus dikompilasi ulang.

9. Performance

   Performance pada Java sering dikatakan kurang tinggi. Namun performance Java dapat ditingkatkan menggunakan kompilasi Java lain seperti buatan.

2.7.2. MySQL

MySQL bukan lingkungan pengembangan aplikasi basisdata, tetapi satu DBMS kecil yang kompak. MySQL adalah DBMS kecil, kompak dan mudah digunakan MySQL ideal untuk aplikasi berukuran kecil dan menengah, namun telah menjanjikan untuk penggunaan besar. Implementasi MySQL beruoa *client/server* terdiri dari daemond mysqld (pada sistem Unix) dan beberapa program *client*. MySQL tersedia pada *platform* Unix, Windows NT/95/98. Pada *platform* Unix, MySQL menggunakan *multithreading* yang menaikkan kinerja dan skalabilitas secara bagus.

Adapun kelebihan dari MySQL untuk memilih MySQL dalam penggunaan database, yaitu :

a. Kecepatan

b. Kemudahan penggunaan

c. Ongkos

d. Dukungan bahasa query

e. Kapabilitas

f. Konektivitas dari keamanan

g. Portabilitas

h. Distribusi terbuka

Beberapa karakteristik MySQL

Tabel 2.1 Karakteristik MySQL

No	Karakteristik	Deskripsi
1	Standart	MySQL mendukung *entry-level* ANSI SQL92 ODBc level 0-2
2	Character Set	MySQL secara *default* menggunakan ISO-8859-I (Latin I) charcter set untuk data pengurutan
3	Bahasa Pemrograman	MySQL mendukung pemrograman aplikasi dalam bahasa Java, Perl, PHP, dan sebagainya.
4	Tabel besar	MySQL menyimpan tiap relasi (tabel) pada file terpisah di direktori basisdata. Ukuran maksimum table dibatasi kemampuan sistem operasi menangani ukuran file.
5	Kecepatan, dan kemudahan pemakaian	MySQL kira-kira tiga sampai empat kali lebih cepat disbanding basisdata komersial, juga mudah dikelola.
6	MySQL	MySQL adalah *open-source relational* DBMS

Adapun perintah-perintah (*command*) SQL yang digunakan sebagai berikut:

1. *Create Table*

   Perintah ini digunakan untuk menciptakan sebuah tabel. Bentuk perintah SQL *create table* adalah sebagai berikut:

   CREATE TABLE Nama_Tabel

(

Nama_Field1 Tipe Data Batasan,

...

Nama_FieldN Tipe_Data Batasan

)

2. *Insert Table*

   Perintah ini digunakan untuk memasukkan data ke dalam tabel. Bentuk perintah SQL *insert table* adalah sebagai berikut:

   INSERT INTO Nama_Tabel (Nama_Field1, ..., Nama_FieldN)

   VALUES (Nilai1, ..., NilaiN)

3. *Update Table*

   Perintah ini digunakan untuk mengubah data pada tabel. Bentuk perintah SQL *update table* adalah sebagai berikut:

   UPDATE Nama_Tabel **SET**

   Nama_Field1 = Nilai1,

   ...

   Nama_FieldN = NilaiN

   WHERE kondisi

4. *Delete Table*

   Perintah ini digunakan untuk menghapus data pada tabel. Bentuk perintah SQL *delete table* adalah sebagai berikut:

   DELETE FROM Nama_Tabel

   WHERE kondisi

5. *Select Table*

   Perintah ini digunakan untuk mencari data dalam tabel. Bentuk perintah SQL *select table* adalah sebagai berikut:

   SELECT <Nama_Field1>, ..., <Nama_FieldN>

   FROM <Nama_Tabel>

   WHERE kondisi