BAB 2

TINJAUAN PUSTAKA

2.1 Profil PT PLN (Persero) Distribusi Jawa Barat dan Banten

2.1.1 Sejarah PT PLN (Persero)

Berawal di tahun 1905, di kota Bandung berdiri perusahaan listrik milik Pemerintah Kolonial Belanda dengan nama Bandoengsche Electriciteit Maatschaappij (BEM). Selanjutnya BEM diubah menjadi perusahaan perseroan dengan nama Gemeenschapplijk Electriciteit Bedrijf en Omstreken Voor Bandoeng (GEBEO).

Perubahan kembali terjadi, ketika pemerintahan Jepang mengambil alih kekuasaan di Indonesia di antara rentang waktu 1942 -1945. Pada saat itu, pendistribusian tenaga listrik dilaksanakan oleh perusahaan yang didirikan oleh Pemerintah Jepang dengan nama Djawa Denki Djigyo Sha Bandoeng Shi Sha.

Pasca Kemerdekaan Republik Indonesia, penguasaan pengelolaan tenaga listrik ditangani langsung oleh Pemerintah Indonesia. Salah satunya ditandai dengan terbentuknya perusahaan listrik di Jawa Barat dengan nama PLN Exploitasi XI pada tahun 1961 hingga pertengahan tahun 1975. Kemudian pada kurun waktu 1975 sampai 1994, PLN Exploitasi XI diubah namanya menjadi Perusahaan Umum (Perum) Listrik Negara Distribusi Jawa Barat.

Di tahun 1994, sejalan dengan perkembangan ekonomi dan pertumbuhan kelistrikan yang bergerak begitu cepat, Badan Hukum PLN mengalami perubahan dari Perusahaan Umum (Perum) menjadi Perseroan. Perubahan ini turut

mengubah nama perusahaan listrik di Jawa Barat menjadi PT PLN (Persero) Distribusi Jawa Barat.

Oleh karena wilayah kerjanya tidak hanya menjangkau Jawa Barat saja, tetapi juga Propinsi Banten, maka sejak tanggal 27 Agustus 2002 hingga saat ini nama PT PLN (Persero) Distribusi Jawa Barat dilengkapi menjadi PT PLN (Persero) Distribusi Jawa Barat dan Banten. Dan kini, PT PLN (Persero) Distribusi Jawa Barat dan Banten --- disingkat PLN DJBB --- masih menempati bangunan lawas bernilai sejarah yang beralamat di Jl. Asia Afrika No. 63 Bandung dan memiliki beberapa APJ (Area Pelayanan Jaringan) yang salah satunya bernama APJ Majalaya di Jl. Jaksa Naranata No. 1 Baleendah Kab. Bandung 40258.

2.1.2 Falsafah Perusahaan

“Pembawa Kecerahan dan Kegairahan dalam Kehidupan Masyarakat yang Produktif”.

Falsafah Perusahaan melandasi keyakinan kami, bahwa kami bukan sekedar penyedia energi listrik akan tetapi juga berkontribusi pada pengembangan masyarakat produktif dan peningkatan kualitas kehidupan masyarakat. Dan ini sekaligus memberikan pondasi yang kuat bagi kami untuk mewujudkan VISI DAN MISI PERUSAHAAN, sebagai berikut:

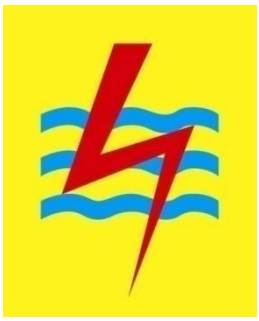
2.1.3 Visi Perusahaan

“Diakui sebagai Perusahaan Kelas Dunia yang Tumbuh Berkembang, Unggul dan Terpercaya dengan bertumpu pada Potensi Insani”.

2.1.4 Misi Perusahaan

1. Melakukan bisnis kelistrikan dan bidang lain yang terkait, berorientasi kepada kepuasan pelanggan, anggota perusahan dan pemegang saham.
2. Menjadikan tenaga listrik sebagai media untuk meningkatkan kualitas kehidupan masyarakat.
3. Mengupayakan agar tenaga listrik menjadi pendorong kegiatan ekonomi.
4. Menjalankan kegiatan usaha yang berwawasan lingkungan.

2.1.5 Logo PT PLN (Persero) Distribusi Jawa Barat dan Banten



Gambar 2.1 Logo PT PLN (Persero)

Arti dari logo tersebut adalah sebagai berikut :

I. Bidang Persegi Panjang menjadi pembatas antara elemen-elemen logo dengan bidang lainnya. Bidang ini melambangkan PT PLN (Persero) sebagai wadah atau organisasi yang terorganisir dengan sempurna.

II. Petir atau kilat yang melambangkan tenaga listrik yang terkandung di dalamnya sebagai produk jasa utama yang dihasilkan oleh perusahaan. Selain itu petir pun mengartikan kerja cepat dan tepat para insan PT PLN (Persero) dalam memberikan solusi terbaik bagi para pelanggannya.

III. Tiga Gelombang yang memiliki arti gaya rambat energi listrik yang dialirkan oleh tiga bidang usaha utama yang digeluti perusahaan yaitu pembangkitan, penyaluran dan distribusi yang seiring sejalan dengan kerja keras para insan PT PLN (Persero) di dalam ketiga bidang usaha tersebut guna memberikan layanan terbaik bagi pelanggannya.

Adapun arti dari warna dalam logo tersebut antara lain :

I. Kuning (Pantone 102c)

Menggambarkan pencerahan, seperti yang diharapkan PLN bahwa listrik mampu menciptakan pencerahan bagi kehidupan masyarakat. Kuning juga melambangkan semangat yang menyala-nyala yang dimiliki tiap insan yang berkarya di perusahaan ini.

II. Merah (Pantone Warm Red)

Melambangkan kedewasaan PLN sebagai perusahaan listrik pertama di Indonesia dan kedinamisan gerak laju perusahaan beserta tiap insan perusahaan serta keberanian dalam menghadapi tantangan perkembangan jaman.

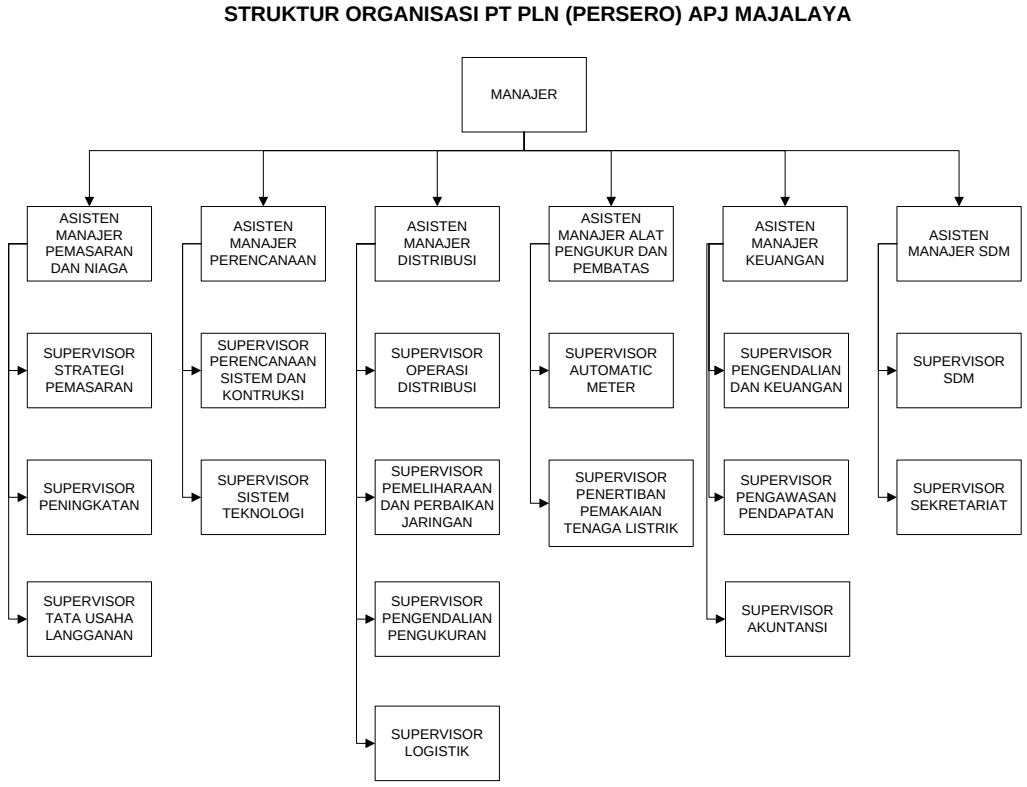
III. Biru (Pantone Process Cyan)

Biru, warna yang menampilkan kesan konstan (sesuatu yang tetap) seperti halnya listrik yang tetap diperlukan dalam kehidupan manusia. Di samping itu biru melambangkan keandalan yang dimiliki insan-insan perusahaan dalam memberikan layanan terbaik bagi para pelanggannya.

2.1.6 Badan Hukum PT PLN (Persero) Distribusi Jawa Barat dan Banten

Badan hukum PT PLN (Persero) adalah Badan Usaha Milik Negara (BUMN).

2.1.7 Struktur Organisasi



Gambar 2.2 Struktur Organisasi

Tugas pokok PT PLN (Persero) Distribusi Jawa Barat dan Banten meliputi distribusi, penjualan tenaga listrik dan pelayanan pelanggan.

2.1.8 *Job Description*

Job description dari struktur organisasi PT PLN (Persero) Distribusi Jawa Barat dan Banten adalah:

1. Manajer: bertugas memimpin dan diantaranya mangatur kinerja dari 6 asisten manajer sebagai staf-nya dalam suatu area tertentu agar distribusi listrik ke masyarakat tersalurkan dengan baik.
2. Asisten Manajer Pemasaran dan Niaga : bertugas melakukan pengawasan terhadap pemasaran dan niaga tenaga listrik.

3. Asisten Manajer Perencanaan : bertugas melakukan perencanaan terhadap pemasangan dan pengembangan area penyaluran listrik.
4. Asisten Manajer Distribusi : bertugas melakukan pengawasan terhadap pendistribusian tenaga listrik ke pelanggan.
5. Asisten Manajer Alat Pengukur dan Pembatas (APP) : bertugas melakukan pengawasan alat pengukur dan pembatas yang digunakan oleh para pelanggan.
6. Asisten Manajer Keuangan : bertugas mengatur keuangan pendapatan hasil dari penjualan tenaga listrik ke pelanggan.
7. Asisten Manajer Sumber Daya Manusia (SDM) : bertugas untuk mengawasi setiap kinerja dari para pegawai yang ada di APJ Majalaya.

2.2 Landasan Teori

2.2.1 Data

Menurut Abdul Kadir (2003: 29), data adalah deskripsi tentang benda, kejadian, aktivitas, dan transaksi, yang tidak mempunyai makna atau tidak berpengaruh secara langsung kepada pemakai. Data merupakan bentuk jamak dari *datum*, berasal dari bahasa Latin yang berbarti "sesuatau yang diberikan". Dalam penggunaan sehari-hari, data berarti suatu pernyataan yang diterima secara apa adanya. Pernyataan ini adalah suatu variabel yang bentuknya dapat berupa angka, kata-kata, atau citra.

Dalam keilmuan (ilmiah), fakta dikumpulkan untuk menjadi data. Data kemudian diolah sehingga dapat diutarakan secara jelas dan tepat sehingga dapat dimengerti oleh orang lain yang tak langsung mengalaminya sendiri, hal ini

dinamakan deskripsi. Pemilahan banyak data sesuai dengan persamaan atau perbedaan yang dikandungnya dinamakan klasifikasi.

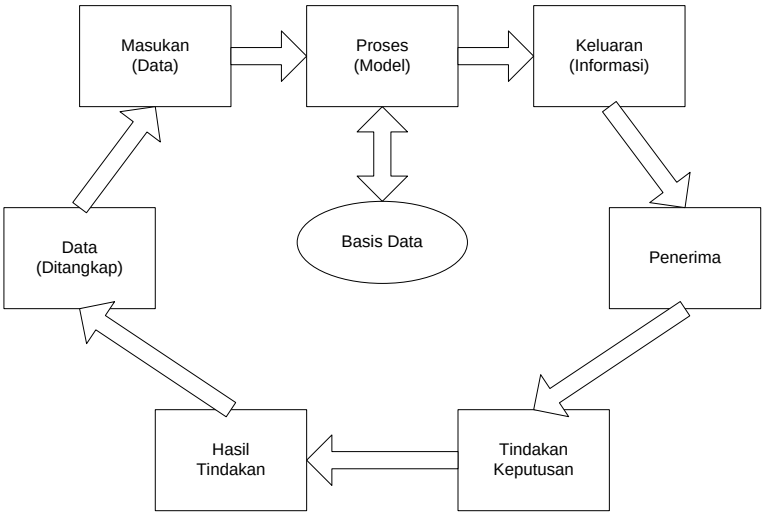
Secara konseptual, data adalah deskripsi tentang benda, kejadian, aktivitas, dan transaksi, yang tidak mempunyai makna atau tidak berpengaruh secara langsung kepada pemakai. Data dapat berupa nilai yang terformat, teks, citra, audio, dan video.

a) Data yang terformat adalah data dengan suatu format tertentu. Misalnya, data yang menyatakan tanggal atau jam, atau menyatakan nilai mata uang.
b) Teks adalah sederetan huruf, angka, dan simbol-simbol khusus (misalnya + dan $) yang kombinasinya tidak tergantung pada masing-masing item secara individual. Contoh teks adalah artikel majalah.
c) Citra (*image*) adalah data dalam bentuk gambar. Citra dapat berupa grafik, foto, hasil *rontgen*, dan tanda tangan, ataupun gambar yang lain.
d) Audio adalah data dalam bentuk suara. Instrumen musik, suara orang atau suara binatang, gemericik air, detak jantung merupakan beberapa contoh data audio.
e) Video menyatakan data dalam bentuk sejumlah gambar yang bergerak dan bias saja dilengkapi dengan suara. Video dapat digunakan untuk mengabadikan suatau kejadian atau aktivitas.

2.2.2 Informasi

Informasi adalah data yang sudah diolah, dibentuk, atau dimanipulasi sesuai dengan keperluan tertentu atau hasil dari pengolahan data yang secara prinsip memiliki nilai atau *value* yang lebih dibandingkan data mentah. Informasi

dapat juga dianggap seuatu data untuk diolah kembali dan menjadikan informasi sesuai dengan keperluan unit kerja tertentu.



Gambar 2.3 Siklus Informasi

Menurut Burch dan Grudnitski (1989) Gambar 2.3 memperlihatkan siklus informasi, yaitu menggambarkan pengolahan data menjadi informasi dan pemakaian informasi untuk mengambil keputusan, hingga akhirnya dari tindakan hasil pengambilan keputusan tersebut dihasilkan data kembali.

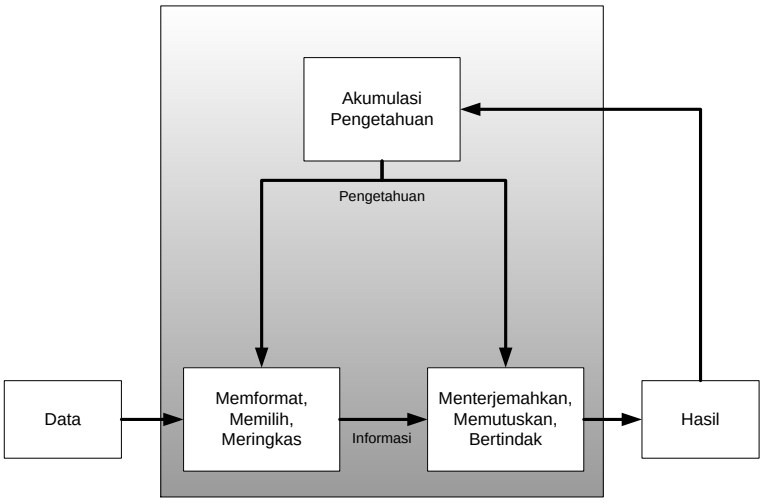
Jadi, hal yang terpenting untuk membedakan informasi dengan data, informasi itu mempunyai kandungan “makna”, sedangkan data tidak. Pengertian data di sini merupakan hal yang sangat penting, karena berdasarkan maknalah si penerima dapat memahami infomasi tersebut dan secara lebih jauh dapat menggunakannya untuk menarik suatu kesimpulan atrau bahkan mengambil keputusan.

Informasi mempunyai tingkat kualitas yang ditentukan beberapa hal, antara lain:

1. *Akurat*, informasi harus bebas dari kesalahan-kesalahan dan harus jelas penyampaian maksudnya.
2. *Tepat pada waktunya*, informasi yang dating tidak boleh terlambat pada penerima.
3. *Relevan*, informasi harus mempunyai manfaat bagi pemakainya.
4. *Lengkap*, informasi berisi informasi yang dibutuhkan.
5. *Jelas*, isi informasi sesuai dengan keperluan pemakai.

2.2.3 Pengetahuan

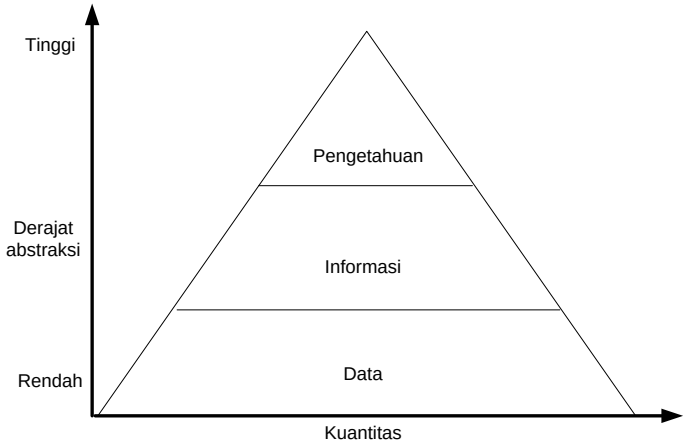
Menurut Alter (1992), pengetahuan (*knowledge*) adalah kombinasi dari naluri, gagasan aturan, dan prosedur yang mengarahkan tindakan atau keputusan. Sebagai gambaran, informasi yang dipadukan dengan pengalaman masa lalu dan keahlian akan memberikan suatu pengetahuan yang tentu saja memiliki nilai yang tinggi. Sebuah gambaran tentang hubungan antara data, informasi, dan pengetahuan ditunjukkan pada gambar berikut.



Gambar 2.4 Hubungan data, informasi, dan pengetahuan

Gambar 2.4 memperlihatkan bahwa data diformat, dipilih, dan diringkas leh sisetm menjadi suatu informasi. Proses tersebut dilakukan berdasarkan suatu pengetahuan tentang cara melakukannya. Selanjutnya, informasi yang dihasilkan dimasukkan ke tahap berikutnya dan diproses menjadi suatu hasil. Hasil ini diakumulasikan sebagai pengetahuan yang kemudian digunakan untuk melakukan pemrosesan data atau pemrosesan informasi.

Bersama-sama dengan data dan informasi, pengetahuan sering digambarkan dalam bentuk piramida, sperti gambar di bawah ini.



Gambar 2.5 Gambaran data, informasi, dan pengetahuan menurut abstraksi

Gambar 2.5 menunjukkan bahwa dilihat dari derajat abstraksi, pengetahuanlah yang memiliki kuantitas paling abstrak, sedangkan jika dilihat dari kuantitas, pengetahuanlah yang memiliki kuantitas paling kecil.

2.2.4 Pengertian Program Aplikasi

2.2.4.1 Pengertian Program

Program adalah ekspresi pernyataan atau kombinasi yang disusun dan dirangkai menjadi satu kesatuan prosedur yang berupa urutan langkah untuk

menyelesaikan masalah yang diimplementasikan dengan menggunakan bahasa pemrograman, sehingga dapat dieksekusi oleh komputer.

2.2.4.2 Pengertian Aplikasi

Aplikasi berupa penerapan, menyimpan suatu hal, data permasalahan pekerjaan ke dalam suatu sarana atau media yang digunakan untuk menerapkan atau mengimplementasikan hal atau permasalahan tersebut sehingga berubah menjasi suatu bentuk yang baru tanpa menghilangkan nilai-nilai dasar dari hal, data, permasalahan atau pekerjaan. Jadi dalam hal ini hanya bentuk dari tampilan data yang berubah, sedangkan isi yang memuat dalam data tersebut tidak mengalami perubahan. Jadi program aplikasi adalah sederetan kode yang digunakan untuk mengatur komputer supaya dapat melakukan pekerjaan sesuai dengan keinginan programmer atau user.

2.2.5 Basis Data

Basis Data (*Database*) dapat dibayangkan sebagai sebuah lemari arsip. Basis Data terdiri dari atas dua kata, yaitu Basis dan Data. Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang atau 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 bertuk angka, huruf, symbol, teks, gambar, bunyi, atau kombinasinya.

"*Basis Data dapat diartikan sebagai himpunan atau sekumpulan data bisa berupa tabel atau file yang saling berhubungan dan disimpan dalam media penyimpanan elektronis tanpa pengulangan* (*redudansi).*"

Setiap basis data umumnya dibuat untuk mewakili sebuah semesta data yang spesifik. Karena itu, operasi-operasi dasar yang dapat dilakukan berkenaan dengan basis data dapat meliputi :

1. Pembuatan basis data baru (*create database*), yang identik dengan pembuatan lemari arsip yang baru.
2. Penghapusan basis data (*drop database*), yang identik dengan perusakan lemari arsip beserta isinya.
3. Pembuatan file/tabel baru ke suatu basis data (*create table*), yang identik dengan penambahan map/arsip baru ke sebuah lemari arsip yang telah ada.
4. Penghapusan file/tabel dari suatu basis data (*drop table),* yang identik dengan perusakan map arsip lama yang ada di sebuah lemari arsip.
5. Penambahan/ pengisian data baru kesebuah file/tabel disebuah basis data (*insert),* yang identik dengan penambahan lembaran arsip kesebuah map arsip.
6. Pengambilan data dari sebuah file/tabel (*retrieve/search),* yang identik dengan pencarian lembaran arsip dari sebiuah map arsip.
7. Pengubahan data dari sebuah file/tabel (*update),* yang identik dengan perbaikan isi lembar arsip yang ada di sebuah map arsip.
8. Penghapusan data dari sebuah file/tabel (*delete),* yang identik dengan penghapusan sebuah lembaran arsip yang ada di sebuah map arsip.

Operasi yang berkenaan dengan pembuatan objek (basis data dan tabel) merupakan operasi awal yang hanya dilakukan sekali dan berlaku seterusnya. Sedangkan operasi-operasi yang berkaitan dengan isi tabel (data) merupakan operasi rutin yang akan berlangsung berulang-ulang dank arena itu operasi-operasi

inilah yang lebih tepat mewakili aktifitas pengelolaan (*management*) dan pengolahan (*processing)* data dalam basis data.

Tujuan awal dan utama dalam pengelolaan data dalam sebuah basis data agar memperoleh atau menemukan kembali data yang dicari dengan mudah dan cepat. Disamping itu, pemanfaatan basis data untuk pengelolaan data, juga memiliki tujuan lain seperti berikut :

1. Kecepatan dan Kemudahan (*Speed)*
2. Efisiensi Ruang Penyimpanan (*Space)*
3. Keakuratan (*Accuracy)*
4. Ketersediaan (*Availability)*
5. Kelengkapan (*Completeness)*
6. Keamanan (*Security)*
7. Kebersamaan Pemakaian (*Sharability*)

Basis Data hanya sebuah objek yang pasif/mati. Basis data tidak akan pernah berguna jika tidak ada penggeraknya. Yang menjadi pengelola atau penggeraknya secara langsung adalah program atau aplikasi (*software)*. Gabungan keduanya (basis data dan pengelolanya) menghasilkan sebuah sistem. Karena itu, secara umum sistem basis data merupakan sistem yang terdiri atas kumpulan file (tabel) yang saling berhubungan dan sekumpulan program (DBMS) yang memungkinkan beberapa pemakai (program lain) untuk mengakses dan memanipulasi file-file (tabel-tabel) tersebut. Sistem basis data memiliki beberapa komponen pendukung diantaranya :

a. Perangkat Keras

Perangkat keras yang biasanya terdapat dalam sebuah sistem basis data adalah

1. Komputer (satu untuk sistem yang *stand-alone* atau lebih untuk sistem jaringan).
2. Memori sekunder yang *on-line* (*Harddisk).*
3. Meori sekunder yang *off-line* (*Tape* atau *Removeable Disk)* untuk keperluan backup data.
4. Media/perangkat komunikasi (untuk sistem jaringan).

b. Sistem Operasi (*Operating Sistem)*

Secara sederhana, Sistem Operasi merupakan program yang mengaktifkan/memfungsikan sistem computer, mengendalikan seluruh sumber daya (*resource)* dalam komputer dan melakukan operasi-operasi dasar dalam computer (operasi I/O, pengelolaan *file*, dan lain-lain). Sejumlah Sistem Operasi yang banyak digunakan seperti : MS-DOS, MS-Windows 3.1, MS-Windows XP, MS-Windows Vista (2007), UNIX, dan lain-lain. Program pengelola basis data hanya dapat aktif jika sistem operasi yang dikehendaki (sesuai) telah aktif.

c. Basis Data (*Database)*

Sebuah sistem basis data dapat memiliki beberapa basis data. Setiap basis data dapat berisi/memiliki sejumlah objek basis data (seperti file/tabel, indeks, dan lain-lain). Disamping berisi/menyimpan data, setiap basis data juga mengandung definisi struktur (baik untuk bais data maupun objek-objeknya secara detail).

d. Sistem Pengelola Basis Data (*Database Management System/DBMS*)

Pengelolaan basis data secara fisik tidak dilakukan oleh pemakai secara langsung, tetapi ditangani oleh sebuah perangkat lunak (sistem) yang

khusus/spesefik. Perangkat lunak inilah yang akan menentukan bagaimana data organisasi, disimpan, diubah dan diambil kembali. DBMS juga menerapkan mekanisme pengamanan data, pemakaian data secara bersama, pemaksaan keakuratan konsistensi data, dan sebagainya. Perangkat lunak yang termasuk DBMS seperti dBase III+, dBaseIV, 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).

e. Pemakai (*User)*

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

1. Programmer Aplikasi

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

2. User Mahir (*Casual User*)

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

3. 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.

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

f. Aplikasi (Perangkat Lunak) Lain

   Aplikasi (Perangkat Lunak) lain ini bersifat opsional. Artinya ada tidaknya tergantung kebutuhan. DBMS yang digunakan lebih berperan dalam pengorganisasian data dalam basis data, sementara bagi pemakai basis data (khususnya yang menjadi *end-user/naïve-user)* dapat disediakan program khusus lain untuk melakukan pengisian, pengubahan dan pengambilan data. Program ini ada yang sudah disediakan bersama dengan DBMS-nya, ada juga yang harus dibuat sendiri dengan menggunakan aplikasi lain yang khusus untuk itu (*development tools).*

2.2.5.1 Penerapan Basis Data

Bidang-bidang fungsional yang telah umum memanfaatkan basis data demi efisiensi, akurasi dan kecepatan operasi antara lain: bidang perbankan yang melakukan pengelolaan data nasabah/data tabungan/data pinjaman, pembuatan laporan-laporan akuntansi, pelayanan informasi pada nasabah/calon nasabah dan

lain-lain. Kemudian bidang asuransi, bidang pendidikan/sekolah, telekomunikasi, rumah sakit dan lain-lain.

2.2.5.2 Kegunaan Basis Data

Penyusunan suatu basis data digunakan untuk mengatasi masalah masalah pada penyusunan data yaitu:

1) Redundansi dan Inkonsistensi data
2) Kesulitan dalam pengaksesan data
3) Isolasi data untuk standardisasi
4) Multiple User (banyak pemakai)
5) Masalah keamanan (*security*)
6) Masalah integritas (kesatuan)
7) Masalah data independence (kebebasan data)

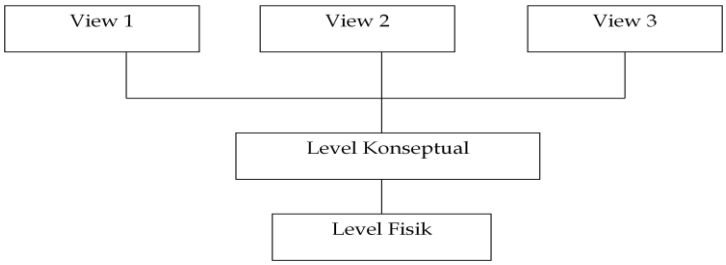
2.2.5.3 Abstraksi Basis Data

Abstraksi data merupakan tingkatan/level dalam bagaimana pemakai melihat data dalam sebuah sistem basis data. Abstraksi data dalam sistem basis data dibagi menjadi tiga level yaitu:

1) Level Fisik (*Physical Level*), merupakan level terendah dalam abstraksi data, yang menunjukkan bagaimana (*how*) sesungguhnya suatu data disimpan.
2) Level Lojik/Konseptual (*Conceptual Level*),merupakan level lebih tinggi berikutnya dalam abstraksi data yang menggambarkan data apa (*what*) yang sebenarnya disimpan dalam basis data dan hubungan relasi yang terjadi antara data.

3) Level Pandangan Pemakai (*View Level*), merupakan level tertinggi dari abstraksi data yang hanya menunjukkan satu bagian dari keseluruhan basis data.

Hubungan antar level tersebut dapat digambarkan (Kristanto:1996) pada Gambar 2.6.



Gambar 2.6 Abstraksi Data

2.2.5.4 Bahasa Basis Data

Sebuah bahasa basis data biasanya dapat dibagi ke dalam 2 bentuk yaitu:

1) *Data Definition Language* (DDL)

   Yaitu struktur/skema basis data yang menggambarkan/mewakili desain basis data secara keseluruhan dispesifikasikan dengan bahasa khusus.

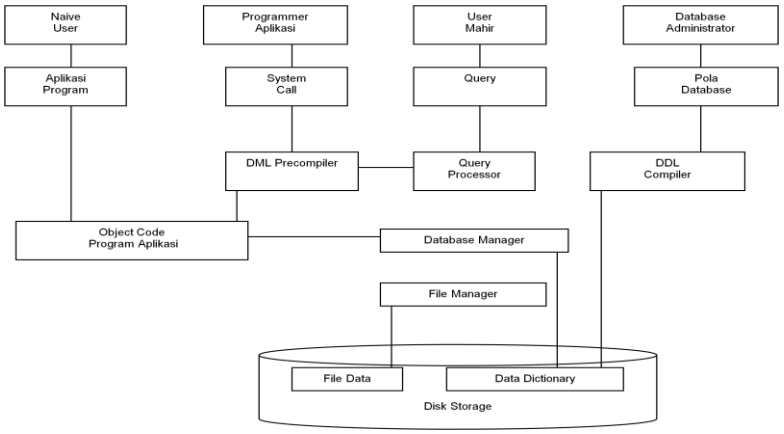
2) *Data Manipulation Language* (DML)

   Yaitu perintah-perintah yang digunakan untuk mengubah, memanipulasi, dan mengambil data pada basis data.

3) DCL (*Data Control Language*) yang berkaitan dengan pengaturan sekuritas terhadap basis data.

2.2.5.5 Struktur Sistem Secara Keseluruhan

Pada Gambar 2.7 memperlihatkan struktur sistem basis data secara keseluruhan.



Gambar 2.7 Struktur sistem secara keseluruhan

2.2.6 Analisis dan Pemodelan Sistem

2.2.6.1 *Flowmap*

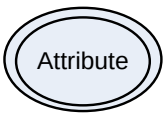
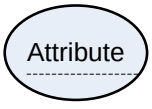
Bagan alir atau *flow map* adalah bagan yang menunjukan alir didalam program atau prosedur sistem secara logika. Bagan alir digunakan terutama untuk alat bantu komunikasi dan dokumentasi. Bagan alir dokumen atau sering disebut juga *Flow Map* merupakan bagan alir yang menunjukan arus dari aliran formulir termasuk tembusan-tembusannya.

2.2.6.2 ERD (*Entity Relationship Diagram*)

Entity Relationship Diagram (ERD) adalah ilustrasi dari entitas-entitas dalam bisnis dan relationship antar entitas. ERD memisahkan antara informasi yang dibutuhkan dalam bisnis dari aktivitas-aktivitas yang dilakukan dalam bisnis. Jadi, meskipun terjadi perubahan proses bisnis, jenis informasi hampir tetap konstan. Oleh karena itu, struktur data juga hampir tidak berubah. Tujuan utama dari penggambaran ERD adalah untuk menunjukkan struktur objek data (*entity*) dan hubungan (*relationship*) yang ada pada objek tersebut. ERD berguna bagi profesional sistem, karena ERD memperlihatkan hubungan antara data store

pada *Data Flow Diagram* (DFD). Notasi yang digunakan dalam ERD dapat dilihat pada Tabel di bawah ini :

Tabel 2.1 Notasi ERD

Notasi	Deskripsi
	**Entity Set**
	**Relationship set**
	Penghubung atribut dengan entitas dan relasi dengan entitas
	**Attribute**
	**Key Attribute**
	**Derived Attribute (atribut turunan)**
	**Multi-value Attribute**
	**Identifier Attribute (pada weak entity)**
	**Weak Entity Set**

	**Relationship Set Weak Entity**

ERD (*Entity Relationship Diagram*) berfungsi untuk menggambarkan relasi dari dua file atau dua tabel yang dapat di golongkan dalam tiga macam bentuk relasi, antara lain :

a. One to One Relationship (1 – 1)

Hubungan antara file pertama dan file kedua adalah satu berbanding satu.

b. One to Many Relationship (1 – N)

Hubungan antara file pertama dan file kedua adalah satu berbanding banyak atau dapat pula dibalik.

c. Many to Many Relationship (N – N)

Hubungan antara file pertama dan file kedua adalah banyak berbanding banyak.

2.2.6.3 Diagram Konteks

Diagram konteks adalah suatu alat atau metode penggambaran suatu sistem informasi secara global, baik sistem informasi yang berbasis komputer atau tidak berbasis komputer. Diagram konteks terdiri dari sebuah simbol proses yang mewakili keseluruhan proses dalam sistem dan minimal sebuah *external entity* (entitas luar) yang merupakan sumber atau tujuan data dari sistem tersebut dan aliran data yang menggambarkan aliran suatu masukan ataupun keluaran dari sistem tersebut. Berdasarkan notasi Yourdon proses digambarkan dengan

lingkaran, entitas luar dengan persegi panjang, dan aliran data digambarkan dengan garis yang diberi mata panah.

2.2.6.2 DFD (*Data Flow Diagram*)

Data flow Diagram (DFD) adalah diagram yang menggunakan notasi-notasi untuk menggambarkan arus dari sistem. DFD sering digunakan untuk menggambarkan sustu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir (misalnya lewat telepon, surat, dan sebagainya) atau lingkungan fisik dimana data tersebut akan disimpan (misalnya file kartu, harddisk, tape, diskette, dan lain sebagianya).

Komponen-komponen DFD yang digunakan dapat dilihat pada gambar berikut ini, yaitu:

Gane/Sarson	Yourdon/De Marco	Keterangan
Entitas Eksternal	Entitas Eksternal	Entitas eksternal, dapat berupa orang/unit terkait yang berinteraksi dengan sistem tetapi diluar sistem
Proses	Proses	Orang, unit yang mempergunakan atau melakukan transformasi data. Komponen fisik tidak diidentifikasikan.
Aliran data	Aliran data	Aliran data dengan arah khusus dari sumber ke tujuan
Data store	Data Store	Penyimpanan data atau tempat data direfer oleh proses.

Gambar 2.8 Komponen-komponen DFD

1. *External entity* (kesatuan Luar) atau *boundary* (batas sistem)

   Setiap sistem pasti memiliki batas sistem (boundary) yang memisahkan suatu sistem dengan lingkungan luarnya. Kesatuan luar (external entity) merupakan

kesatuan di lingkungan luar sistem yang dapat berupa orang, organisasi atau sistem lainya yang berada di lingkungan luarnya yang memberikan input atau menerima output dari sistem.

2. *Data flow* (arus data)

   Arus data di DFD diberi simbol panah. Arus data ini mengalir diantara proses, simpanan, dan kesatuan luar.

3. *Process* (proses)

   Suatu proses adalah kegiatan atau kerja yang dilakukan oleh orang, mesin atau komputer dari hasil suatu arus data yang masuk ke dalam proses untuk dihasilkan arus data yang akan keluar dari proses.

4. Data *store* (simpanan data)

   Merupakan simpanan dari data yang dapat berupa suatu file atau *database* di komputer, suatu arsip atau catatan manual dan lain sebagainya.

2.2.6.3 Kamus Data

Kamus data (*data dictionary*) adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Kamus data dibuat pada tahap analisis maupun pada tahap perencanaan sistem. Pada tahap analisis, kamus data dapat digunakan sebagai alat komunikasi antara analis sistem dengan pemakai sistem tentang data yang mengalir di sistem, yaitu tentang data yang masuk ke sistem dan tentang informasi yang dibutuhkan oleh pemakai sistem. Kamus data harus memuat hal-hal berikut ini:

1. Nama arus data
2. Panjang karakter
3. Tipe data

4. Deskripsi field

2.2.7 Konsep Database

Database terdiri dari dua kata, yaitu *data* dan *base*. *Base* merupakan tempat untuk berkumpul, sedangkan *data* adalah fakta-fakta yang dapat disimpan dan mewakili suatu objek seperti manusia, hewan, peristiwa, konsep, keadaan, barang, dan sebagainya yang direkam dalam bentuk angka, huruf, symbol, teks, gambar ataupun kombinasinya. Secara umum *database* dapat diartikan sebagai kumpulan data tentang suatu benda atau kejadian yang disimpan dan saling berhubungan satu sama lain.

2.2.7.1 Perancangan *Database*

Database adalah kumpulan dari *item* data yang saling berhubungan satu dengan lainnya yang diorganisasikan berdasarkan sebuah skema atau struktur tertentu, tersimpan di *hardware* komputer dan dengan *software* untuk melakukan manipulasi untuk kegunaan tertentu. Beberapa istilah yang berhubungan dengan database, yaitu:

1. *Entity*

   Entity adalah suatu objek nyata dan informasinya akan direkam. Pada sistem pemesanan *online* misalnya, *entity* adalah anggota, produk, admin.

2. *Atribute*

   Setiap *entity* memiliki *Atribute* atau sebutan untuk mewakili suatu *entity*. Misalnya, anggota memiliki *atribute* id_anggota, nama, *password*, alamat.

3. Data *Value*

   Data aktual atau informasi yang disimpan pada tiap data elemen atau *atribute*. *Atribute* nama anggota menunjukkan tempat dimana informasi nama anggota disimpan, sedang data *value* adalah isi data nama anggota tersebut.

4. *Record*

   Record adalah kumpulan elemen-elemen yang saling berkaitan menginformasikan tentang suatu *entity* secara lengkap. Satu *record* mewakili satu data atau informasi tentang pesan misalnya, id_pesan, no_anggota, kd_produk, tgl_pesan, jam_pesan.

5. *File*

   File adalah kumpulan *record-record* sejenis yang mempunyai panjang elemen yang sama, *atribute* yang sama, namun berbeda-beda data *value*-nya.

Tujuan dalam pengelolaan data dalam database adalah mengelompokkan data-data sesuai dengan fungsi dan untuk memenuhi sejumlah tujuan organisasi/perusahaan, oleh sebab itu pengembangan sistem database berhubungan dengan masalah-masalah pada file-file database yang harus dihindari yaitu:

1. Redudansi dan inkonsistensi data.
2. Kesulitan pengaksesan data.
3. Isolasi data untuk standarisasi.
4. *Multiple user*.
5. *Security*.

6. Masalah integritas.
7. Masalah data *independence*.

2.2.7.2 Normalisasi

Normalisasi adalah suatu proses untuk membuat data yang tidak normal menjadi data yang normal (tanpa redudansi). Bentuk data yang tidak normal/ data mentah biasa juga disebut *unnormalized form. unnormalized form* merupakan kumpulan data yang tidak mengikuti aturan tertentu, tidak lengkap atau terduplikasi.

Hal-hal yang menjadi dasar proses normalisasi adalah sebagai berikut:

1) Lossie-Join Decompotion

Dilakukan dekomposisi memiliki pengertian, yaitu melakukan pemilahan terhadap suatu tabel menjadi beberapa tabel dengan mempertimbangkan Ketergantungan Fungsional. Pada prinsipnya, dekomposisi agar setiap tabel yang digunakan hanya memiliki 1 KF (lebih tepatnya KF Minimum), karena sebuah KF bisa saja diuraikan menjadi sejumlah KF. Dan ini dapat menjadi salah satu indikasi tentang baik/buruknya sebuah tabel.

2) Dependency Preservation

Dependency Preservation (dapat diartikan sebagai Pemeliharaan ketergantungan) merupakan kriteria kedua yang harus dicapai ketika melakukan perubahan data, maka harus bisa dijamin agar perubahan tersebut tidak menghasilkan inkonsistensi data yang mengakibatkan Ketergantungan Fungsional yang sudah benar menjadi tidak terpenuhi.

2.2.8 Model Pengembangan Perangkat Lunak

2.2.8.1 Model Sekuensial Linear (*Waterfall*)

Model sekuensial linier untuk *software engineering*, sering disebut juga dengan **siklus kehidupan klasik** atau **model air terjun (*waterfall*)**. Model ini mengusulkan sebuah pendekatan kepada perkembangan software yang sistematik dan sekuensial yang mulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain, kode, pengujian, dan pemeliharaan. Dimodelkan setelah siklus rekysa konvensional, model sekuensial linier melingkupi aktivitas – aktivitas sebagai berikut :

1. **Rekayasa dan pemodelan sistem/informasi**

Karena sistem merupakan bagian dari sebuah sistem yang lebih besar, kerja dimulai dengan membangun syarat dari semua elemen sistem dan mengalokasikan beberapa subset dari kebutuhan ke software tersebut. Pandangan sistem ini penting ketika software harus berhubungan dengan elemen-elemen yang lain seperti software, manusia, dandatabase. Rekayasa dan anasisis sistem menyangkut pengumpulan kebutuhan pada tingkat sistem dengan sejumlah kecil analisis serta disain tingkat puncak. Rekayasa informasi mancakup juga pengumpulan kebutuhan pada tingkat bisnis strategis dan tingkat area bisnis.

2. **Analisis kebutuhan Software**

Proses pengumpulan kebutuhan diintensifkan dan difokuskan, khusunya pada software. Untuk memahami sifat program yang dibangun, analis harus memahami domain informasi, tingkah laku, unjuk kerja, dan interface yang diperlukan. Kebutuhan baik untuk sistem maupun software didokumentasikan dan dilihat lagi dengan pelanggan.

3. Desain

Desain *software* sebenarnya adalah proses multi langkah yang berfokus pada empat atribut sebuah program yang berbeda; struktur data, arsitektur *software*, representasi *interface*, dan detail (algoritma) prosedural. Proses desain menterjemahkan syarat/kebutuhan ke dalam sebuah representasi software yang dapat diperkirakan demi kualitas sebelum dimulai pemunculan kode. Sebagaimana persyaratan, desain didokumentasikan dan menjadi bagian dari konfigurasi *software*.

4. Generasi Kode

Desain harus diterjemahkan kedalam bentuk mesin yang bisa dibaca. Langkah pembuatan kode melakukan tugas ini. Jika desain dilakukan dengan cara yang lengkap, pembuatan kode dapat diselesaikan secara mekanis.

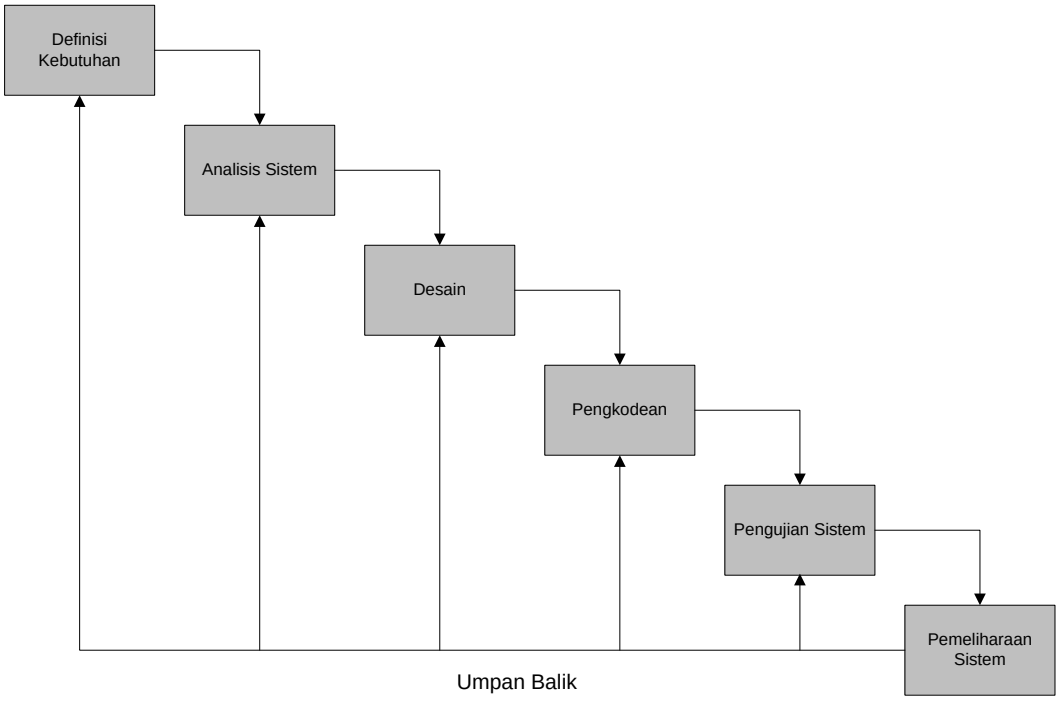
5. Pengujian

Sekali program dibuat, pengujian program dimulai. Proses pengujian berfokus pada logika internal software, memastikan bahwa semua pernyataan sudah diuji, dan pada eksternal fungsional, yaitu mengarahkan pengujian untuk menemukan kesalahan – kesalahan dan memastikan bahwa input yang dibatasi akan memberikan hasil aktual yang sesuai dengan hasil yang dibutuhkan.

6. Pemeliharaan

Software akan mengalami perubahan setelah disampaikan kepada pelanggan (perkecualian yang mungkin adalah software yang dilekatkan). Perubahan akan terjadi karena kesalahan – kesalahan ditentukan, karena software harusdisesuaikan untuk mengakomodasi perubahan–perubahan di

dalam lingkungan eksternalnya (contohnya perubahan yang dibutuhkan sebagai akibat dari perangkat peripheral atau sistem operasi yang baru), atau karena pelanggan membutuhkan perkembangan fungsional atau unjuk kerja. Pemeliharaan software mengaplikasikan lagi setiap fase program sebelumnya dan tidak membuat yang baru lagi.



Gambar 2.9 Model Waterfall

Masalah yang kadang terjadi ketika model sekuensial linier diaplikasikan adalah :

1. Jarang sekali proyek nyata mengikuti aliran sekuensial yang dianjurkan oleh model. Meskipun model linier bisa mengakomodasi iterasi, model ini melakukannya dengan cara tidak langsung. Sebagai hasilnya, perubahan–perubahan dapat menyebabkan keraguan pada saat tim proyek berjalan.
2. Kadang – kadang sulit bagi pelanggan untuk menyatakan semua kebutuhannya secara eksplisit. Model linier sekuensial memerlukan hal

ini dan mengalami kesulitan untuk mengakomodasi ketidakpastian natural yang ada pada bagian awal beberapa proyek.

3. Pelanggan harus bersifat sabar. Sebuah versi kerja dari program – program kerja itu tidak akan diperoleh sampai akhir waktu proyek dilalui. Sebuah kesalahan besar, jika tidak terdeteksi sampai program yang bekerja tersebut dikaji ulang, bisa menjadi petaka.
4. Pengembang sering melakukan penundan yang tidak perlu. Sifat alami dari siklus kehidupan klasik membawa kepada *blocking state* di mana banyak anggota tim proyek harus menunggu tim yang lain untuk melengkapi tugas yang saling memiliki ketergantungan. Blocking state cenderung menjadi lebih lazim pada awal dan akhir sebuah proses sekuensial linier.

2.2.9 Pengertian Algoritma Genetika

Algoritma genetika merupakan algoritma heuristik atau pendekatan yang menerapkan proses seleksi alam dan evolusi yang dikemukakan oleh Charles Darwin. Algoritma genetik pertama kali diperkenalkan oleh John Holland pada tahun 1960-an dan pengembangan lebih lanjut dilakukan oleh Holland bersama murid-murid dan rekan sekerjanya di Universitas Michigan pada tahun 1960-an dan 1970-an.

Sebagai ide dasar heuristik, algoritma genetika dimulai dengan sekumpulan solusi yang direpresentasikan dengan kromosom, disebut populasi. Semua solusi dari satu populasi diambil dan digunakan untuk membentuk populasi baru. Hal ini dimotivasi dengan harapan populasi baru akan lebih baik daripada populasi lama. Solusi terpilih yang merupakan orangtua pembentuk keturunan (*offspring*), dipilih

berdasarkan lingkungannya atau nilai *fitness*, semakin baik maka semakin besar kesempatan yang dimiliki untuk bertahan dan bereproduksi. Hal ini dilakukan berulang kali hingga memenuhi beberapa kondisi penghentian, misalnya jumlah generasi atau tercapainya hasil yang dicari.

2.2.9.1 Prosedur Algoritma Genetik

Sifat dari algoritma genetika adalah mencari kemungkinan-kemungkinan dari kandidat solusi untuk mendapatkan suatu solusi yang optimal bagi penyelesaian masalah. Ruang cakupan dari semua solusi yang layak (*feasible*) yaitu obyek-obyek diantara solusi yang sesuai dinamakan ruang pencarian (*search space*). Tiap titik dalam ruang pencarian merepresentasikan satu solusi yang layak yang dapat ditandai dengan nilai *fitness* yang sesuai dengan masalah. Solusi yang dicari dalam algoritma genetika adalah titik (1 atau lebih) di antara solusi yang layak dalam ruang pencarian. Sifat dari pencarian inilah yang menyebabkan algoritma genetika baik diterapkan untuk menyelesaikan masalah NP-complete.

Istilah fungsi evaluasi dan fungsi fitness sering saling dipertukarkan penggunaannya. Akan tetapi, adalah berguna untuk membedakan kedua fungsi tersebut dalam algoritma genetika. Fungsi evaluasi, atau fungsi obyektif, merupakan suatu pengukuran performansi terhadap sekumpulan parameter tertentu. Fungsi *fitness* mentransformasikan pengukuran performansi tersebut dalam suatu alokasi kesempatan bereproduksi. Evaluasi terhadap suatu barisan yang menyatakan sekumpulan parameter tertentu adalah independen dari evaluasi terhadap barisan-barisan lain. *Fitness* suatu barisan selalu didefinisikan berdasarkan anggota-anggota lain dalam populasi. Secara umum *fitness*

didefinisikan *fi/f* dengan *fi* adalah fungsi evaluasi dengan kromsom i dan f adalah evaluasi rata-rata dari semua kromosom dalam populasi.

Secara sederhana, algoritma umum dari algoritma genetik ini dapat dirumuskan menjadi beberapa langkah, yaitu:

1. [**Mulai**] Membangun populasi random sebanyak n kromosom (sesuai dengan masalahnya)
2. [**Fitness**] Evaluasi setiap fitness f(x) dari setiap kromosom x pada populasi
3. [**Populasi baru**] Membuat populasi baru dengan mengulang langkah-langkah berikut sampai populasi lengkap.
    1. [**Seleksi**] Pilih 2 kromosom orangtua dari populasi berdasarkan fitnessnya (semakin besar fitnessnya semakin besar kemungkinannya untuk terpilih)
    2. [**Perkawinan silang**] Sesuai dengan besarnya kemungkinan perkawinan silang, orangtua terpilih disilangkan untuk membentuk anak. Jika tidak ada perkawinan silang, maka anak merupakan salinan dari orangtuanya.
    3. [**Mutasi**] Sesuai dengan besarnya kemungkinan mutasi, anak dimutasi pada setiap lokus (posisi pada kromosom).
    4. [**Penerimaan**] Tempatkan anak baru pada populasi baru.
4. [**Ganti**] Gunakan populasi yang baru dibentuk untuk proses algoritma selanjutnya.
5. [**Tes**] Jika kondisi akhir terpenuhi, berhenti, dan hasilnya adalah solusi terbaik dari populasi saat itu.
6. [**Ulangi**] Ke nomer 2.

2.2.9.2 Pengkodean

Pengkodean adalah cara untuk merepresentasikan masalah kedalam bentuk kromosom. Ada beberapa cara pengkodean dan pemilihannya tergantung dari masalah yang akan diselesaikan. Berikut ini adalah beberapa jenis pengkodean yang sering digunakan.

2.2.9.2.1 Pengkodean biner (*binary encoding*)

Pengkodean biner adalah pengkodean yang paling umum digunakan karena algoritma genetika awalnya menggunakan pengkodean ini. Pada pengkodean biner setiap kromosom merupakan barisan bit 0 atau 1, seperti dapat dilihat pada tabel berikut ini :

Tabel 2.2 Contoh Kromosom Biner

Kromosom A	10110010110010101110010 1
Kromosom B	11111110000011000001111 1

Pengkodean biner memberikan banyak kemungkinan kromosom meskipun hanya memiliki jumlah allele yang sedikit, 1 atau 0. Tetapi pengkodean ini sering tidak sesuai untuk beberapa masalah dan kadang harus dilakukan pengkoreksian setelah perkawinan silang dan mutasi. Contoh masalah yang sesuai menggunakan pengkodean biner adalah masalah ransel (*knapsack*), dimana terdapat beberapa jenis benda dengan ukuran dan berat yang berbeda sedangkan kapasitas ransel terbatas. Permasalahannya adalah memilih sejumlah barang untuk memaksimumkan nilai dari barang-barang tersebut di dalam ransel, tetapi total ukuran barang-barang tersebut tidak melebihi dari ukuran ransel itu sendiri. Masalah ransel ini dapat dienkodekan dengan barisan bit, dimana jumlah bit menyatakan jumlah barang yang ada, sehingga setiap bit berhubungan dengan

setiap benda yang ada di dalam ransel. Bit 1 berarti benda ada di dalam ransel sedangkan 0 berarti benda tidak berada di dalam ransel.

2.2.9.2.2 Pengkodean permutasi (*permutation encoding*)

Pengkodean permutasi digunakan dalam masalah yang memerlukan pengurutan data (*ordering problem*), seperti masalah wiraniaga (*travelling salesman problem*), atau masalah pengurutan tugas (*task ordering problem*). Pada pengkodean ini setiap kromosom merupakan barisan angka yang merepresentasikan angka pada urutan.

Tabel 2.3 Contoh Kromosom Permutasi

Kromosom A	1 5 3 2 6 4 7 9 8
Kromosom B	8 5 6 7 2 3 1 4 9

Pengkodean permutasi hanya berguna pada masalah pengurutan. Perkawinan silang dan mutasi yang dilakukan pada pengkodean ini harus menghasilkan kromosom yang memiliki konsistensi urutan.

Contoh masalah yang dapat diselesaikan dengan pengkodean ini adalah masalah wiraniaga. Terdapat sejumlah kota dan jarak diantara semua kota itu. Seorang wiraniaga harus mengunjungi semua kota itu, tetapi sebisa mungkin dia tidak perlu melakukan terlalu banyak perjalanan, sehingga harus ditemukan urutan kota yang akan dikunjungi dengan jumlah total jarak yang minimal. Masalah ini dapat dienkodekan dengan pengkodean permutasi dimana setiap angka dalam kromosom mewakili setiap kota yang harus dikunjungi dalam urutan.

2.2.9.2.3 Pengkodean nilai (*value encoding*)

Pengkodean nilai dapat digunakan pada masalah yang mempunyai nilai yang rumit, dimana nilai yang dienkodekan langsung merupakan representasi dari masalah. Contohnya adalah pengkodean data yang menggunakan angka riil yang akan menjadi susah jika menggunakan pengkodean biner karena setiap angka riil tidak dapat direpresentasikan secara tepat hanya dalam bentuk bit 1 atau 0.

Tabel 2.4 Contoh Kromosom dengan pengkodean nilai

Kromosom A	1.2324, 5.3243, 0.4556, 2.3293, 2.4545
Kromosom B	A, B, D, J, E, I, F, J, D, H, D, I, E, R, J
Kromosom C	(back), (back), (right), (forward), (left)

Dalam pengkodean nilai, setiap kromosom merupakan barisan dari beberapa nilai. Nilai bisa berupa apa saja, seperti bilangan biasa, bilangan riil, karakter sampai dengan obyek-obyek yang rumit.

Pengkodean nilai sangat baik untuk beberapa masalah khusus, dilain sisi perkawinan silang dan mutasi yang diperlukan untuk pengkodean ini seringkali memerlukan pengembangan tersendiri yang lebih spesifik pada masalah.

Contoh masalah yang menggunakan pengkodean ini adalah masalah pencarian bobot pada jaringan saraf (neural network). Terdapat sebuah jaringan saraf dengan arsitekturnya. Permasalahnya adalah mencari bobot masukan dari neuron-neuron untuk melatih jaringan supaya memberikan keluaran yang diinginkan. Masalah ini dapat dienkodekan menggunakan pengkodean nilai bilangan riil, dengan gen-gen dalam genome menyatakan korespondensi bobot-bobot pada masukan untuk jaringan saraf.

2.2.9.2.4 Pengkodean pohon (*tree encoding*)

Pengkodean pohon lebih banyak digunakan untuk menyusun program atau ekspresi, bagi pemrograman genetika (*genetic programming*). Dalam pengkodean pohon, setiap kromosom merupakan pohon dari sejumlah obyek, seperti fungsi atau perintah di dalam bahasa pemrograman. Pengkodean pohon sangat baik bagi penyusunan program. Bahasa pemrograman LISP sering digunakan untuk melakukan pengkodean pohon, karena program di dalamnya dapat direpresentasikan ke dalam bentuk ini dan dapat dengan mudah dilakukan parse menjadi sebuah pohon, sehingga perkawinan silang dan mutasi dapat dilakukan relatif lebih mudah.

Contoh masalah yang direpresentasikan dengan pengkodean pohon adalah masalah mencari fungsi berdasarkan nilai-nilai yang diberikan. Diberikan beberapa nilai masukan dan keluaran. Permasalahannya adalah mencari fungsi yang akan memberikan nilai keluaran terbaik (dekat dengan yang diinginkan) untuk semua masukan. Masalah ini dapat dienkodekan ke dalam pengkodean pohon dengan kromosom merupakan fungsi-fungsi yang dinyatakan dalam pohon.

2.2.9.2.5 Operator dalam Algoritma Genetika

Operator-operator pada algoritma genetika yang sering digunakan adalah seleksi, perkawinan silang dan mutasi. Semua operator tersebut memegang peranan penting bagi keberhasilan algoritma genetika dalam menyelesaikan masalah. Pemilihan jenis operator yang akan digunakan tergantung dari masalah yang akan diselesaikan. Hal yang harus diperhatikan adalah menghindari terjadinya konvergensi prematur, yaitu mencapai solusi optimum yang belum waktunya, dalam arti bahwa solusi yang diperoleh adalah solusi optimum lokal.

Selain itu, untuk menjamin bahwa seluruh ruang pencarian dapat dijangkau oleh alogritma genetika, perbedaan (*diversity*) diantara kromosom dalam populasi merupakan hal yang penting.

2.2.9.3 Seleksi

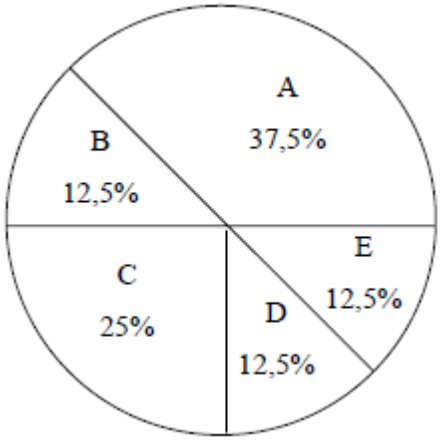
Seleksi merupakan proses pemilihan individu/kromosom dari generasi lama untuk dijadikan orangtua yang akan saling dikawin silang untuk membentuk individu baru di generasi baru. Yang menjadi permasalahan adalah bagaimana cara memilih kromosom untuk dijadikan orangtua. Berdasarkan teori evolusi Darwin, kromosom yang terbaik seharusnya dapat bertahan hidup dan membentuk keturunan baru. Ada beberapa jenis metode seleksi, berikut ini adalah beberapa metode seleksi yang sering digunakan.

2.2.9.3.1 Seleksi roda roulette (*roulette wheel selection*)

Pada metode seleksi ini, orangtua dipilih berdasarkan nilai *fitness*-nya, semakin baik nilai fitnessnya maka semakin besar kemungkinannya untuk terpilih. Diandaikan semua kromosom diletakkan pada sebuah roda *roulette*, besarnya kemungkinan bagi setiap kromosom adalah tergantung dari nilai *fitness*-nya seperti pada contoh berikut :

Tabel 2.5 Contoh populasi dengan 5 kromosom beserta *fitness*nya

Kromosom	Fitness
A	15
B	5
C	10
D	5
E	5



Gambar 2.10 Probabilitas terpilihnya suatu kromosom dalam roda *roulette*

Probabilitas masing-masing individu merupakan hasil pembagian antara *fitness* masing-masing individu dengan total *fitness* dalam populasi. Dari contoh di atas, kromosom A memiliki probabilitas 37,5% untuk terpilih sebagai orangtua pembentuk keturunan baru pada setiap pemilihan kromosom (setiap roda diputar).

Skema seleksi dengan roda *roulette* ini adalah berdasarkan skala *fitness* (*fitness scale*). Karena terpilihnya suatu kromosom dalam populasi untuk dapat berkembang biak adalah sebanding dengan *fitness*-nya, maka akan terjadi semacam *tradeoff* antara ekplorasi dan ekplotasi jika terdapat satu atau kelompok kecil kromosom yang mempunyai *fitness* yang baik yaitu antara mengeksplorasi bagian-bagian baru dalam ruang pencarian, atau terus mengeksplotasi informasi yang telah diperoleh. Kecenderungan kromosom yang baik untuk terpelihara terus dapat membawa ke hasil optimum lokal atau konvergensi dini (*premature convergence*) ke suatu hasil yang bukan optimum global. Sebaliknya jika semua kromosom dalam populasi mempunyai *fitness* yang hampir sama, maka seleksi ini akan menjadi seleksi yang bersifat acak.

2.2.9.3.2 Seleksi rangking (*rank selection*)

Metode seleksi roda *roulette* akan memiliki masalah ketika terdapat perbedaan *fitness* yang jauh. Sebagai contoh, jika *fitness* kromosom terbaik adalah 90 % dari semua roda *roulette* dapat menyebabkan kromosom yang lain memiliki kesempatan yang sangat kecil untuk dapat terpilih.

Seleksi ranking terlebih dahulu merangking atau mengurutkan kromosom di dalam populasi berdasarkan *fitness*-nya kemudian memberi nilai *fitness* baru berdasarkan urutannya. Kromosom dengan *fitness* terburuk akan memiliki *fitness* baru bernilai 1, terburuk kedua bernilai 2 dan seterusnya, sehingga kromosom yang memiliki *fitness* terbaik akan memiliki nilai *fitness* N, dimana N adalah jumlah kromosom di dalam populasi.

Sebagai ilustrasi dapat dilihat pada gambar berikut yang juga merupakan contoh dari metode seleksi roda *roulette* tadi :

Tabel 2.6 Keadaan sebelum di *ranking*

Kromosom	Fitness
A	15
B	5
C	10
D	5
E	5

Tabel 2.7 Keadaan setelah di *ranking*

Kromosom	Fitness	Fitness Baru
B	5	1
D	5	2
E	5	3
C	10	4
A	15	5

Setelah proses pengurutan dan pemberian nilai *fitness* baru, setiap kromosom akan memiliki kesempatan yang lebih adil untuk terpilih. Tetapi metode ini dapat menyebabkan konvergensi menjadi lambat, karena kromosom terbaik tidak terlalu berbeda dengan yang lainnya.

2.2.9.3.3 Seleksi Turnamen (*tournament selection*)

Seleksi turnamen merupakan variasi antara seleksi roda *roulette* dan seleksi rangking. Sejumlah k kromosom tertentu dari populasi beranggota n kromosom ($k \leq n$) dipilih secara acak dengan probabilitas yang sama. Dari k kromosom yang terpilih kemudian akan dipilih satu kromosom dengan *fitness* terbaik, yang diperoleh dari hasil pengurutan rangking *fitness* semua kromosom terpilih. Perbedaannya dengan seleksi roda *roulette* adalah pemilihan kromosom yang akan digunakan untuk berkembang biak tidak berdasarkan skala *fitness* dari populasi.

Untuk k = 1, seleksi turnamen akan menjadi sama dengan seleksi secara acak karena hanya melibatkan satu kromosom. Untuk k = 2, maka akan dipilih dua kromosom dari populasi secara acak, kemudian akan dipilih salah satu

berdasarkan nilai *fitness*-nya. Biasanya yang sering digunakan adalah k = 2, tergantung dari jumlah kromosom yang tedapat di dalam populasi.

2.2.9.4 Perkawinan silang

Operator perkawinan silang digunakan untuk membentuk keturunan baru berdasarkan orangtua terpilih. Operator ini paling dominan dalam algoritma genetika dibandingkan dengan operator mutasi. Jumlah kromosom yang terlibat adalah 2 kromosom. Perkawinan silang dilakukan dengan harapan kromosom-kromosom baru akan mempunyai bagian baik dari kromosom-kromosom lama dan tidak menutup kemungkinan menjadi kromosom-kromosom yang lebih baik Berikut ini akan dibahas beberapa jenis variasi perkawinan silang dalam algoritma genetika sehubungan dengan jenis pengkodean kromosom yang digunakan.

2.2.9.4.1 Perkawinan silang untuk pengkodean biner

Pengkodean biner merupakan jenis pengkodean yang paling sederhana sehingga banyak variasi operasi perkawinan silang dapat dilakukan. Beberapa jenis perkawinan silang tersebut adalah :

2.2.9.4.1.1 Perkawinan silang 1-titik

Proses perkawinan silang 1-titik dimulai dengan pemilihan 1 titik pada barisan bit kromosom secara acak sebagai titik perkawinan silang. Kromosom baru akan dibentuk dengan cara menyalin barisan bit orangtua pertama dari bit pertama sampai titik perkawinan silang, sedangkan sisanya disalin dari orangtua kedua.

Tabel 2.8 Contoh perkawinan silang 1 titik pada pengkodean biner

Kromosom orangtua 1	**11001**011
Kromosom orangtua 2	11011**111**
Keturunan	**11001111**

2.2.9.4.1.2 Perkawinan silang 2-titik

Proses perkawinan silang 2-titik diawali dengan pemilihan 2 titik perkawinan silang dari barisan bit kromosom secara acak. Kromosom baru akan dibentuk dengan cara menyalin barisan bit kromosom orangtua pertama dari bit pertama sampai dengan titik perkawinan silang pertama dan dari titik perkawinan silang kedua sampai dengan bit terakhir, sedangkan sisanya, yaitu dari titik perkawinan silang pertama sampai titik perkawinan silang kedua disalin dari orangtua kedua.

Tabel 2.9 Contoh perkawinan silang 2 titik pada pengkodean biner

Kromosom orangtua 1	**11**0010**11**
Kromosom orangtua 2	11**0111**11
Keturunan	**11011111**

2.2.9.4.1.3 Perkawinan silang seragam

Perkawinan silang seragam menghasilkan kromosom baru dengan cara menyalin bit-bit secara acak dari kedua orangtuanya.

Tabel 2.10 Contoh perkawinan silang seragam pada pengkodean biner

Kromosom orangtua 1	1**10**010**11**
Kromosom orangtua 2	**1**10**111**11
Keturunan	**11011111**

2.2.9.4.1.4 Perkawinan silang aritmatik

Kromosom baru dihasilkan dari operasi aritmatika yang dilakukan terhadap kedua orangtuanya.

Tabel 2.11 Contoh perkawinan silang aritmatika pada pengkodean biner

Kromosom orangtua 1	11001011
Kromosom orangtua 2	11011111
Keturunan (AND)	**11001011**

2.2.9.4.1.5 Perkawinan silang rekombinasi

Versi paling sederhana dari rekombinasi adalah memilih secara acak dari semua nilai gen yang terdapat pada kedua orangtua untuk setiap gen pada anak. Berikut adalah contoh dari perkawinan silang rekombinasi:

Tabel 2.12 Contoh perkawinan silang rekombinasi pada pengkodean biner

Kromosom orangtua 1	15 9 12 14
Kromosom orangtua 2	13 4 16 2
Keturunan 1	15 4 16 14
Keturunan 2	14 9 12 1

2.2.9.4.2 Perkawinan silang untuk pengkodean permutasi

Pada pengkodean permutasi, jenis perkawinan silang yang sering dipergunakan adalah perkawinan silang 1-titik karena kesederhanaan prosesnya dapat menjaga konsistensi urutan nilai pada kromosom. Jenis perkawinan silang yang terdapat pada pengkodean biner tidak dapat dipergunakan pada pengkodean ini karena tidak dapat menjamin terjaganya konsistensi urutan nilai pada kromosom.

Proses perkawinan silang 1-titik dimulai dengan pemilihan satu titik perkawinan silang. Dari permutasi pertama sampai dengan titik perkawinan silang disalin dari orang tua pertama, sedangkan sisanya didapatkan dengan cara melihat satu persatu nilai pada orang tua kedua, jika belum ada pada kromosom keturunan, maka nilai tersebut ditambahkan.

Tabel 2.13 Contoh perkawinan silang 1-titik pada pengkodean permutasi

Kromosom orangtua 1	**1 2 3 4 5** 6 7 8 9
Kromosom orangtua 2	**4 5 3 6 8** 9 7 2 1
Keturunan (AND)	**1 2 3 4 5** 6 8 7 9

2.2.9.4.3 Perkawinan silang untuk pengkodean nilai

Semua metode perkawinan silang yang terdapat pada pengkodean biner dapat diterapkan pada pengkodean nilai karena tidak perlu memperhatikan urutan seperti ada pengkodean permutasi. Berikut adalah contoh perkawinan silang pada pengkodean nilai:

Tabel 2.14 Contoh perkawinan silang 1-titik pada pengkodean nilai bilangan bulat

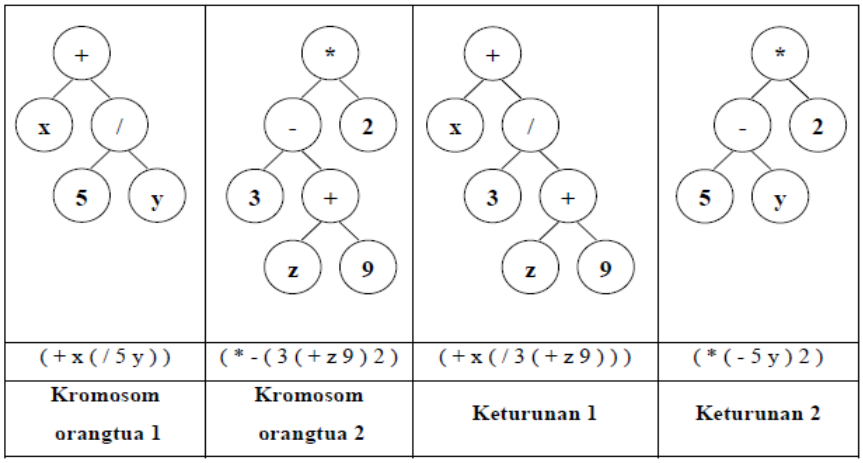
Kromosom orangtua 1	**10 39 45 12 89 34** 1 3 90 48 23 72 25 37
Kromosom orangtua 2	43 13 2 66 49 56 **10 89 53 20 75 99 4 8**
Keturunan	**10 39 45 12 89 34 10 89 53 20 75 99 4 8**

Tabel 2.15 Contoh perkawinan silang 2-titik pada pengkodean nilai bilangan bulat

Kromosom orangtua 1	**10 39 45** 12 89 34 1 3 **90 48 23 72 25 37**
Kromosom orangtua 2	43 13 2 **66 49 56 10 89** 53 20 75 99 4 8
Keturunan	**10 39 45 66 49 56 10 89 90 48 23 72 25 37**

2.2.9.4.4 Perkawinan silang untuk pengkodean pohon

Proses perkawinan silang pada pengkodean pohon dimulai dengan pemilihan satu titik perkawinan silang pada kedua orangtua, kemudian bagian dari orangtua yang berada di bawah titik perkawinan silang dipertukarkan untuk menghasilkan anak baru.



Gambar 2.11 Perkawinan silang untuk pengkodean pohon

2.2.9.5 Mutasi

Mutasi merupakan proses mengubah nilai dari 1 atau beberapa gen dalam suatu kromosom. Tujuannya adalah untuk membantu mempercepat terjadinya perbedaan diantara semua kromosom dalam populasi sehingga pencarian dapat menjelajah keseluruh ruang pencarian, tetapi mutasi tidak boleh terlalu sering dilakukan karena akan membuat algoritma genetika berubah menjadi pencarian acak.

Proses mutasi dalam biologi adalah mengubah isi allele gen pada suatu locus dengan nilai allele yang lain. Proses mutasi dilakukan secara acak, sehingga tidak selalu menjamin bahwa setelah proses mutasi akan diperoleh kromosom dengan fitness yang lebih baik dibandingkan dengan sebelum mutasi. Oleh karena

itu terjadi kontraversi dalam penerapannya pada algoritma genetika. Mutasi seringkali tetap dipergunakan hanya saja dengan probabilitas mutasi yang kecil. Tentang probabilitas perkawinan silang dan mutasi akan dijelaskan pada bagian berikut pada bab ini.

Operator mutasi merupakan operasi pada suatu kromosom tertentu. Berikut adalah beberapa cara mutasi yang sering diterapkan dalam algoritma genetika menurut jenis pengkodean kromosomnya :

2.2.9.5.1 Mutasi pada pengkodean biner

Mutasi dalam pengkodean biner merupakan operasi yang sangat sederhana. Bit pada kromosom yang terpilih secara acak (atau menggunakan skema tertentu) diinversi nilainya, diubah nilainya menjadi lawannya.

Tabel 2.16 Contoh mutasi pada pengkodean biner

Kromosom sebelum mutasi	1 0 0 1 0 **1** 1 1
Kromosom sesudah mutasi	1 0 0 1 0 **0** 1 1

2.2.9.5.2 Mutasi pada pengkodean permutasi

Proses mutasi pada pengkodean biner tidak dapat diterapkan pada pengkodean permutasi karena konsistensi urutan permutasi harus selalu diperhatikan. Salah satu cara yang dapat dilakukan adalah dengan memilih dua posisi (*locus*) pada kromosom dan kemudian saling dipertukarkan

Tabel 2.17 Contoh mutasi pada pengkodean permutasi

Kromosom sebelum mutasi	1 **2** 3 4 5 6 **8** 9 7
Kromosom sesudah mutasi	1 **8** 3 4 5 6 **2** 9 7

2.2.9.5.3 Mutasi pada pengkodean nilai

Proses mutasi pada pengkodean biner dapat diterapkan pada pengkodean nilai karena tidak perlu menjaga konsistensi adanya urutan pada kromosom. Tetapi yang dilakukan bukanlah menginversi bit, melainkan tergantung dari jenis nilai yang dipergunakan. Misalnya nilai yang berupa bilangan riil dimutasi dengan cara saling menukar nilai dua gen yang terdapata pada kromosom, tetapi cara ini tidak memberikan manfaat untuk menjamin adanya perbedaan dalam populasi, semua kromosom dengan mudah memiliki nilai yang sama dan akhirnya terjadi konvergensi prematur. Cara lain yang lebih baik adalah dengan memilih sembarang posisi gen pada kromosom, kemudian ditambahkan atau dikurangkan dengan suatu nilai kecil tertentu yang diambil secara acak. Cara ini juga berlaku pengkodean dengan bilangan bulat. Selain itu cara mutasi lain yang relevan dapat digunakan.

Tabel 2.18 Contoh mutasi pada pengkodean riil

Kromosom sebelum mutasi	1,43 **1,09** 4,51 **9,11** 6,94
Kromosom sesudah mutasi	1,43 **1,19** 4,51 **9,01** 6,94

2.2.9.5.4 Mutasi pada pengkodean pohon

Mutasi dapat dilakukan antara lain dengan mengubah operator (+, -, *, /) atau nilai yang terkandung pada suatu verteks pohon yang dipilih. Atau dapat juga dilakukan pemilihan dua verteks dari pohon dan saling mempertukarkan operator atau nilainya.

2.2.10 *Update* Generasi

Setelah proses seleksi, perkawinan silang dan mutasi yang dilakukan terhadap suatu populasi, proses berikutnya adalah pemilihan kromosom untuk membentuk generasi baru yang disebut *update* generasi. Dalam algoritma genetik *Holland*, proses *update* yang dilakukan adalah *update* secara generasi (*generational update*), yaitu menggantikan keseluruhan kromosom pada generasi sebelumnya dengan kromosom-kromosom yang didapat dari hasil seleksi, perkawinan silang dan mutasi. Penekanan selektif pada skema *update* ini seluruhnya berasal dari orangtua.

Selain itu terdapat skema *update* secra kontinu (*continuous update*) yang mengizinkan orangtua dan anak untuk bercampur dalam satu generasi. Mulanya dipilih k orangtua secara acak, kemudian dibuat anak sebanyak k dengan memilih orangtua tersebut secara seragam. Kromosom pada populasi baru merupakan pemilihan kromosom antara orangtua terpilih dan anak yang dihasilkan berdasarkan *fitness*-nya. Dengan cara ini penekanan selektif berasal seluruhnya dari pemilihan terhadap yang bertahan hidup (*survivor*).

Skema *update* lainnya adalah *steady-state update*, dimana dilakukan pemilihan dua orangtua dari populasi yang kemudian digunakan untuk membentuk 1 (atau 2) anak. Hasilnya digunakan untuk menggantikan :

1. Orangtuanya
2. Kromosom terjelek dalam populasi
3. Kromosom tertua dalam populasi

Suatu strategi bantuan yang baik untuk digunakan dengan skema *continous update* adalah *elitism*, yang berarti mempertahankan kromosom-kromosom baik

dalam populasi lama dengan menyertakannya lagi pada populasi baru. Sejumlah kromosom terbaik pada populasi lama dapat hilang karena proses perkawinan silang dan mutasi. Secara teori, *elitism* dapat meningkatkan kemampuan dari algoritma genetika karena mempertahankan kromosom yang baik dari populasi lama, tetapi kadangkala hal itu justru dapat menyebabkan konvergensi prematur karena nilai *fitness* terjebak pada optimum lokal.

2.2.11 Parameter dalam Algoritma Genetika

Dua parameter dasar yang penting dari algoritma genetika adalah probabilitas perkawinan silang dan probabilitas mutasi.

Probabilitas perkawinan silang menyatakan seberapa sering proses perkawinan silang akan terjadi diantara 2 kromosom orangtua. Jika tidak terjadi perkawinan silang, keturunan merupakan salinan mutlak dari orangtuanya. Jika terjadi perkawinan silang, keturunan dibuat dari bagian-bagian kromosom orangtua. Jika probabilitas perkawinan silang 100%, maka keseluruhan keturunan dibuat dengan perkawinan silang. Jika probabilitas perkawinan silang 0%, maka seluruh generasi baru dibuat dari salinan kromosom-kromosom dari populasi lama, tetapi ini tidak berarti bahwa generasi baru sama dengan yang lama karena adanya penekanan selektif. Meskipun perkawinan silang bertujuan untuk mendapatkan kromosom yang memiliki bagian baik dari orangtuanya atau bahkan menjadi lebih baik dari orangtuanya, ada baiknya juga jika kita membiarkan beberapa bagian dari populasi untuk bertahan ke generasi berikutnya.

Dari hasil penelitian yang sudah dilakukan oleh praktisi algoritma genetika terbukti bahwa angka probabilitas perkawinan silang sebaiknya cukup tinggi,

yaitu antara 80% sampai 95% untuk memberikan hasil yang baik. Untuk beberapa masalah tertentu probabilitas perkawinan silang 60% memeberikan hasil yang lebih baik (Obitko, 1998).

Probabilitas mutasi menyatakan seberapa sering bagian-bagian kromosom akan dimutasikan. Jika tidak ada mutasi, keturunan diambil/disalin langsung setelah perkawinan silang tanpa perubahan. Jika mutasi dilakukan, bagian-bagian keromosom diubah. Jika probabilitas mutasi 100%, semua kromosom diubah. Jika probabilitas mutasi 0%, tidak ada yang diubah. Probabilitas mutasi dalan algoritma genetika seharusnya diberi nilai yang kecil. Umumnya probabilitas mutasi diset untuk mendapatkan rata-rata satu mutasi per kromosom, yaitu angka/ *allele* = 1/ (panjang kromosom).

Dari hasil yang sudah pernah dicoba ternyata angka probabilitas terbaik adalah antara 0,5% sampai 1%. Mutasi diperlukan untuk menjaga perbedaan kromosom dalam populasi, untuk menghindari terjadinya konvergensi prematur. Parameter lain yang juga ikut menentukan efisiensi kinerja algoritma genetika adalah ukuran populasi yaitu banyakknya kromosom dalam satu populasi atau satu generasi.

Jika terlalu sedikit kromosom dalam populasi, algoritma genetika mempunyai kemungkinan yang sedikit untuk melakukan perkawinan silang dan hanya sebagian kecil dari ruang pencarian yang dieksplorasi. Sebaliknya, jika terlalu banyak jumlah kromosom, algoritma genetika cenderung lambat dalam menemukan solusi. Ukuran populasi yang sering digunakan oleh peneliti yang sudah ada adalah antara 20 sampai 30, tetapi kadang ukuran 50 sampai 100 dilaporkan baik. Beberapa penelitian menujukkan bahwa menambah jumlah

populasi tidak akan terlalu berguna karena tidak mempercepat penyelesaian masalah. Riset juga menunjukkan bahwa ukuran populasi yang terbaik ditentukan dari jenis pengkodean, artinya, jika terdapat ukuran kromosom 32 bit, ukuran populasi seharusnya juga 32, begitu pula jika ukuran kromosom 16 bit, maka ukuran populasi adalah 16 (Obitko, 1998).

Penekanan selektif dilakukan dengan memilih jenis seleksi yang sesuai. Umumnya jenis seleksi roda roulette sering digunakan, tetapi kadang seleksi rangking dilaporkan lebih baik. Terdapat juga beberapa metode lebih lanjut yang dapat mengubah parameter seleksi yang digunakan selama ekseskusi algoritma genetika. Prinsip elitism disarankan untuk digunakan jika tidak digunakan metode lain untuk menyimpan solusi terbaik pada suatu populasi ke populasi berikutnya. Sebaikya juga dicoba seleksi steady state untuk proses update populasi.

Terakhir adalah jenis pengkodean, perkawinan silang dan mutasi. Jenis pengkodean dipilih berdasarkan pada masalah dan juga ukuran *instance*-nya. Begitu pula untuk pemilihan jenis operator perkawinan silang dan mutasi.

2.2.12 Pengenalan HTML (*Hyper Text Mark Up Language*)

Hyper Text Markup Language (HTML) adalah sebuah bahasa markup yang digunakan untuk membuat sebuah halaman web dan menampilkan berbagai informasi di dalam sebuah *browser* Internet. Bermula dari sebuah bahasa yang sebelumnya banyak digunakan di dunia penerbitan dan percetakan yang disebut dengan SGML *(Standard Generalized Markup Language)*, HTML adalah sebuah standar yang digunakan secara luas untuk menampilkan halaman web. HTML saat ini merupakan standar Internet yang didefinisikan dan dikendalikan penggunaannya oleh World *Wide Web Consortium* (W3C).

HTML berupa kode-kode tag yang menginstruksikan browser untuk menghasilkan tampilan sesuai dengan yang diinginkan. Sebuah file yang merupakan file HTML dapat dibuka dengan menggunakan browser web seperti Mozilla Firefox atau Microsoft Internet Explorer. HTML juga dapat dikenali oleh aplikasi pembuka email ataupun dari PDA dan program lain yang memiliki kemampuan browser.

HTML dokumen tersebut mirip dengan dokumen teks biasa, hanya dalam dokumen ini sebuah teks bisa memuat instruksi yang ditandai dengan kode atau lebih dikenal dengan TAG tertentu. Sebagai contoh jika ingin membuat teks ditampilkan menjadi tebal seperti: Ditebalkan, maka penulisannya dilakukan dengan cara: **Ditebalkan**. Tanda digunakan untuk mengaktifkan instruksi cetak tebal, diikuti oleh teks yang ingin ditebalkan, dan diakhiri dengan tanda untuk menonaktifkan cetak tebal tersebut. Secara garis besar, terdapat beberapa jenis elemen dari HTML, yaitu :

1. Structural. tanda yang menentukan level atau tingkatan dari sebuah teks (contoh,<h1>Golf</h1> akan memerintahkan browser untuk menampilkan “Golf” sebagai teks tebal besar yang menunjukkan sebagai Heading 1.
2. Presentational. tanda yang menentukan tampilan dari sebuah teks tidak peduli dengan level dari teks tersebut (contoh, boldface akan menampilkan bold. Tanda presentational saat ini sudah mulai digantikan oleh CSS dan tidak direkomendasikan untuk mengatur tampilan teks,

3. *Hypertext*. tanda yang menunjukkan pranala ke bagian dari dokumen tersebut atau pranala ke dokumen lain (contoh, Link Skripsiakan menampilkan "Link Skripsi" sebagai sebuah hyperlink ke URL tertentu), Elemen widget yang membuat objek-objek lain seperti tombol (<button>), list (<li>), dan garis horizontal (<hr>). Selain markup presentational, markup yang lin tidak menentukan bagaimana tampilan dari sebuah teks. Namun untuk saat ini, penggunaan tag HTML untuk menentukan tampilan telah dianjurkan untuk mulai ditinggalkan dan sebagai gantinya digunakan *Cascading Style Sheets.*

Contoh HTML Sederhana :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Belajar HTML</title>
  </head>
  <body>
    <p>Tulisan di Dalam Halaman</p>
  </body>
</html>
```

2.2.13 Pengenalan PHP (*Pretext Hyper-Processor*)

PHP adalah bahasa scripting yang menyatu dengan HTML dan dijalankan pada *server side*. Artinya semua sintaks yang kita berikan akan sepenuhnya dijalankan pada serversedangkan yang dikirimkan ke *browser* hanya hasilnya saja.PHP menyatu dengan bahasa HTML untuk membuat halaman web yang menarik.PHP mampu berjalan diatas beberapa *platfrom* seperti Windows,Unix serta varian Linux.Beberapa kelebihan PHP antara lain dapat membuat situs yang

interaktif dengan forum diskusi,*guestbook* dan sebagainya,koneksitas yang baik dengan bermacam-macam *database* seperti Oracle, MySQL, PostgreSQL dan lain-lain.

Kode PHP diawali dengan tag <? atau <?php dan ditutup dengan tag ?>. File yang berisi tag HTML dan kode PHP diberi ekstensi. Berdasarkan ekstensi ini, pada saat file diakses, *server* akan tahu bahwa file tersebut mengandung kode PHP. *Server* akan menerjemahkan kode dan menghasilkan *output* dalam bentuk tag HTML yang akan dikirim ke *browser* klien yang mengakses file tersebut.

Contoh skrip PHP:

```
<?php
echo "ini contoh skrip PHP";
?>
```

Ketika skrip tersebut dijalankan di *serve*r, *server* akan mengembalikan ke *browser client* dalam bentuk tag HTML. Sehingga ketika dilihat hasilnya di *browser* menggunakan menu *view source,* hasilnya adalah sebagi berikut:

```
<HTML>
<BODY>ini contoh skrip PHP</BODY>
</HTML>
```

Dalam menuliskan *script* atau program PHP, ada dua cara yang sering digunakan, yaitu *embedded script* dan *non-embedded script*. *Embedded script* yaitu menuliskan *script* PHP di antara tag-tag HTML. Sedangkan *non-embedded script* merupakan pembuatan program murni PHP, di mana tag-tag HTML yang diletakkan di dalam *script* PHP. Spasi tidak berpengaruh pada penulisan baris perintah PHP. Untuk penulisan fungsi-fungsi, PHP tidak membedakan antara

huruf kecil dan huruf besar. Contoh perintah ‘echo’ sama dengan perintah ‘ECHO’ atau ‘Echo’.

1. Tipe Data

PHP mengenal lima tipe data, yaitu *Integer, Floating Point, String, Array, dan Objects*. Tipe data dari sebuah *variable* akan ditentukan otomatis oleh PHP bergantung pada operasi yang sedang dilakukan menggunakan *variable* tersebut.

a. *Integer*, meliputi semua bilangan bulat. Besarnya *range* data *Integer* yaitu antara –2.147.483.648 sampai 2.147.483.647. PHP akan secara otomatis mengkonversi data *Integer* menjadi *Floating Point* jika berada di luar *range* tersebut.

b. *Floating Point*, meliputi semua bilangan pecahan atau bilangan desimal. *Range* tipe data *Floating Point* yaitu antara 1,7E-308 sampai 1,7E+308.

c. *String*, sebuah data dengan tipe data *String* dinyatakan dengan mengapitnya menggunakan tanda petik tunggal (‘ ‘) maupun tanda petik ganda (“ “).

d. *Array*, adalah sebuah data yang mengandung satu atau lebih data, dan dapat diindeks berdasarkan *numeric* maupun *string*. Data yang dikandung oleh sebuah data *Array* dapat dari tipe data lainnya, bahkan dapat juga bertipe *Array*.

e. *Objects*, adalah sebuah tipe data yang dapat berupa sebuah bilangan, *variable*, atau bahkan sebuah fungsi. Objects dibuat

dengan tujuan untuk membantu programmer yang terbiasa dengan *Object Oriented Programming.*

Dalam pemrograman PHP, tipe data tidak ditentukan secara eksplisit, tipe data ditetapkan sesuai konteksnya pada saat digunakan. Sebagai contoh, sebuah *variable* $a dideklarasikan bertipe *Integer*, tetapi akan berubah secara otomatis menjadi *Floating Point* jika diberi nilai bilangan pecahan.

2. Perubah (*variable*) dan Konstanta

Perubah atau yang biasa disebut *variable* dinyatakan dengan tanda $ di belakang nama *variable*. Nama *variable* dapat terdiri atas angka, huruf, atau *underscore*. Penamaan *variable* bersifat *case sensitive*, artinya penggunaan huruf besar dan kecil dibedakan. Jadi $a tidak sama dengan $A. Deklarasi sebuah *variable* selalu diikuti dengan pemberian nilai dari *variable* tersebut, sehingga jika sebuah *variable* belum memiliki nilai, tidak perlu dideklarasikan terlebih dahulu.

Konstanta mirip sebuah *variable*, hanya nilainya tetap dan tidak perlu memakai tag $. Konstanta dideklarasikan menggunakan fungsi *define()*. Konstanta dideklarasikan hanya satu kali dan nilai tidak dapat diubah atau didefinisikan lagi pada keseluruhan program.

3. Statement

Statement (Statemen) berfungsi sebagai rangka dari bagian program. Aliran program diatur dengan statemen-statemen kontrol. PHP mengenal dua jenis statemen kontrol, yaitu statemen kondisional dan loop.

a. Statemen kondisional mengatur aliran program berdasarkan pada kondisi tertentu yang ditetapkan. Untuk masalah dengan satu atau

dua percabangan dapat digunakan *if* dan *else*, untuk *multiple* alternatif dapat digunakan *elseif* dan *switch.*

b. *Loop* adalah proses sebuah proses eksekusi operasi program secara berulang-ulang sampai ditemui kondisi untuk mengakhiri eksekusi tersebut. PHP mempunyai dua macam *loop*, yaitu *while loop* dan *for loop*.

4. *Include dan Require*

Include dan *require*, keduanya digunakan untuk memanggil dan mengeksekusi *file* yang ditentukan. Dengan kedua statemen tersebut dapat dibuat fungsi-fungsi, konstanta ataupun perintah operasi biasa dalam sebuah *file* terpisah yang dapat dipanggil dari *file* program lain.

5. Fungsi

Fungsi adalah sekumpulan perintah operasi program yang dapat menerima argumen *input* dan dapat memberikan hasil *output* yang dapat berupa sebuah nilai ataupun sebuah hasil operasi. Fungsi dideklarasikan dengan statemen *function* diikuti nama fungsi dan beberapa *input variable* jika ada. Nama fungsi tidak boleh sama dengan nama *built-in function* yang telah dipunyai oleh PHP, jika sama maka akan keluar pesan kesalahan seperti berikut:

```
Fatal error: Can't redeclare already declared
function in filename on line…
```

2.2.14 Pengenalan *JavaScipt*

JavaScript adalah bahasa pemrograman berbasis prototipe yang berjalan disisi klien. Jika kita berbicara dalam konteks web, sederhananya, kita dapat

memahami *JavaScript* sebagai bahasa pemrograman yang berjalan khusus untuk di browser atau halaman web agar halaman web menjadi lebih hidup. Kalau dilihat dari suku katanya terdiri dari dua suku kata, yaitu *Java* dan *Script*. *Java* adalah bahasa pemrograman berorientasi objek, sedangkan *Script* adalah serangkaian instruksi program.

Secara fungsional, *JavaScript* digunakan untuk menyediakan akses *script* pada objek yang dibenamkan (*embedded*). Contoh sederhana dari penggunaan *JavaScript* adalah membuka halaman *pop up*, fungsi validasi pada *form* sebelum data dikirimkan ke server, merubah *image* kursor ketika melewati objek tertentu, dan lain lain.

2.2.15 Pengenalan CSS (*Cascading Style Sheet*)

Cascading style sheet adalah suatu kumpulan kode yang digunakan untuk memformat dan mengendalikan tampilan isi dalam suatu halaman web.

Penggunaan yang paling umum dari CSS adalah untuk memformat halaman web yang ditulis dengan HTML dan XHTML. Walaupun demikian, bahasanya sendiri dapat dipergunakan untuk semua jenis dokumen XML termasuk SVG dan XUL. Spesifikasi CSS diatur oleh World Wide Web Consortium (W3C).

CSS digunakan untuk pembaca halaman web sebagai menentukan warna, jenis huruf, tata letak, dan berbagai aspek tampilan dokumen. CSS digunakan terutama untuk memisahkan antara isi dokumen (yang ditulis dengan HTML atau bahasa markup lainnya) dengan presentasi dokumen (yang ditulis dengan CSS).

Pemisahan ini dapat meningkatkan aksesibilitas isi, memberikan lebih banyak keleluasaan dan kontrol terhadap tampilan, dan mengurangi kompleksitas serta pengulangan pada stuktur isi.

2.2.16 Macromedia Dreamweaver 8

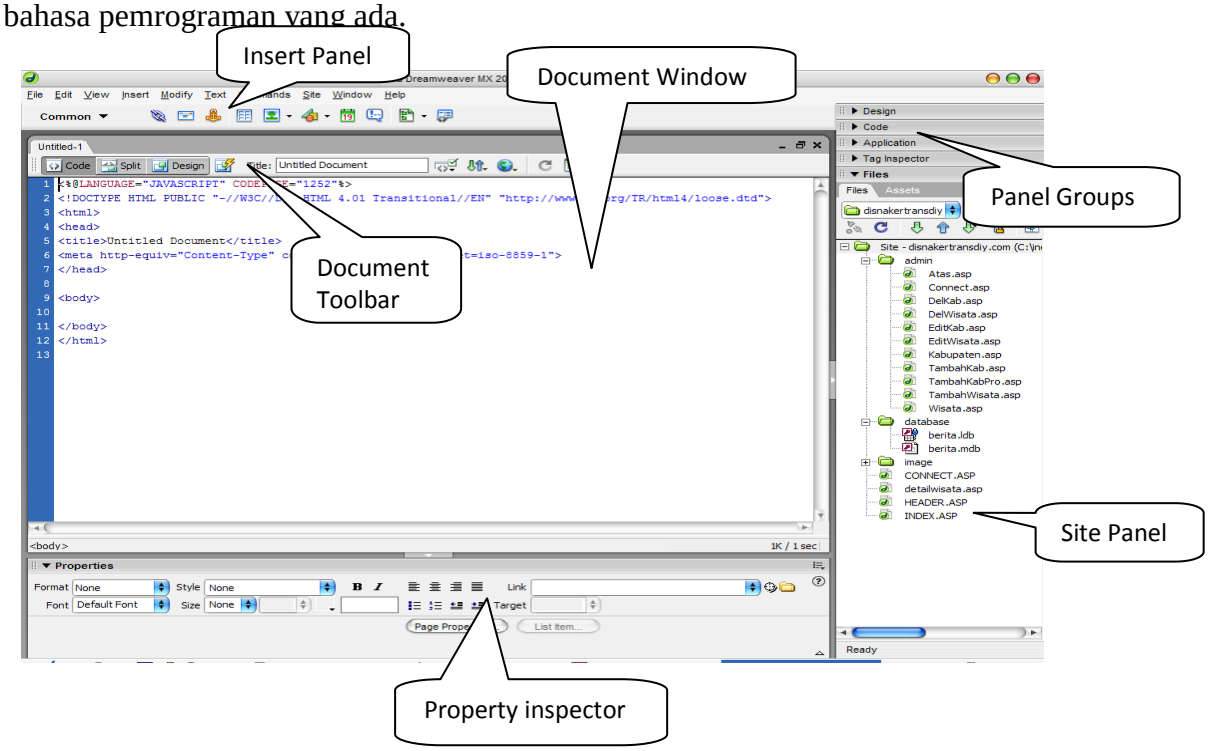
Dreamweaver MX adalah suatu bentuk program editor web yang dibuat oleh Macromedia. Dengan program ini, seorang programer web dapat dengan mudah membuat dan mendesain web.

Dreamweaver adalah editor yang lengkap digunakan untuk membuat animasi sederhana yang berbentuk layer. Dengan adanya program ini, *programer*tidak akan susah dalam menulis *script-script* format HTML, PHP, ASP, maupun bentuk program lainnya.

Sebagai editor, dreamweaver MX mempunyai sifat yang WYSIWYG *(what you see is what you get)*, artinya apa yang kamu lihat akan kamu peroleh. Dengan kelebihan ini, seorang programer dapat langsung melihat hasil buatannya tanpa harus dibuka di *browser*. Seperti program editor-editor lainnya, Dreamweaver MX juga memiliki dua bentuk layer, yaitu bentuk halaman design dan halaman code. Hal ini akan mempermudah dalam menambahkan *script* berbasis PHP maupun Javascript. Dreamweaver selain mendukung pembuatan web yang berbasis HTML, juga mendukung program-program web yang lain, seperti PHP, ASP, Perl, Javascript dan lain-lain.

Versi sebelum dreamweaver MX adalah dreamweaver versi 3 dan 4. Yang membedakan dreamweaver MX dengan versi sebelumnya adalah bentuk tool-toolnya yang diimplementasikan menjadi icon yang mempermudah dalam penggunaannya. Bantuan pada halaman code pun dimiliki oleh dreamweaver MX

sehingga membantu *programer* jika lupa. Dreamweaver juga mendukung format bahasa pemrograman yang ada.



Gambar 2.12 Lembar Kerja Macromedia Dreamweaver 8

Beberapa komponen yang ada pada gambar diatas adalah:

1) ***Document Window*** berfunsi untuk menampilkan dokumen dimana anda sekarang bekerja.
2) ***Insert Bar*** mengandung tombol-tombol untuk menyisipkan barbagai macam objek seperti *image*, tabel dan layer ke dalam dokumen.
3) ***Document Toolbar*** berisi tombol-tombol dan menu pop-up yang menyediakan tampilan berbeda dari Document Window.
4) **Kelompok Panel** adalah kumpulan panel yang saling berkaitan satu sama lain, yang dikelompokan dibawah satu judul.
5) **Tag Selector** berfungsi untuk menampilkan hirarki tag disekitar pilihan yang aktif pada *Design View*.

6) **Property Inspector** digunakan untuk melihat dan mengubah berbagai *property object* atau teks.
7) **Site Panel** digunakan untuk mengatur file dan folder yang membentuk situs web anda.

2.2.17 SQL (*Structured Query Language*)

Antar muka standar untuk sistem manajemen basis data relasional, termasuk sistem yang beroperasi pada komputer pribadi. SQL (*Structured Query Language)* memungkinkan seorang pengguna untuk mengakses informasi tanpa mengetahui dimana lokasinya atau bagaimana informasi tersebut disusun. SQL lebih mudah untuk digunakan dibandingkan dengan bahasa pemrograman tetapi lebih rumit dibanding *software* lembar kerja dan pengolah kata. Sebuah pernyataan SQL yang sederhana dapat menghasilkan set permintaan untuk informasi yang tersimpan pada komputer yang berbeda di berbagai lokasi yang tersebar, sehingga membutuhkan waktu dan sumber daya komputasi yang banyak. SQL dapat digunakan untuk investigasi interatif atau pembuatan lapiran *ad hoc* atau disisipkan dalam program aplikasi.

Bahasa pemrograman yang dirancang khusus untuk mengirimkan suatu perintah *query* (pengaksesan data berdasarkan pengalamatan tertentu) terhadap sebuah *database*. Kebanyakan *software database* yang ada saat ini dapat diakses melalui SQL. Setiap aplikasi yang spesifik dapat mengimplementasikan SQL secara sedikit berbeda, tapi seluruh database SQL mendukung subset standar yang ada.

MySQL adalah salah satu jenis *database server* yang sangat terkenal, kepopulerannya disebabkan MySQL menggunakan SQL (*Structure Query*

Language) sebagai dasar untuk mengakses *database*nya. Selain itu, MySQL bersifat free pada berbagai *platform* (kecuali pada Windows, yang bersifat *shareware* atau anda perlu membayar setelah melakukan evaluasi dan memutuskan untuk digunakan untuk keperluan produksi) atau tidak di cekal.

MySQL termasuk jenis RDBMS (*Relational Database Management System*). Itulah sebabnya istilah seperti tabel, baris dan kolom digunakan di dalam MySQL. Sebuah *database* mengandung satu atau sejumlah tabel, tabel terdiri atas sejumlah baris an setiap baris mengandung satu atau beberapa kolom. Kelebihan dari MySQL adalah:

1) Menggunakan bahasa *query standart* yang dimiliki SQL(*Struktur Query Language*).
2) API (*Application Programming Interface*) dengan C, C++, Eifell, Java, Perl, PHP, Phyton dan Tcl.
3) Dukungan sepenuhnya untuk operator dan fungsi pada bagian SELECT dan WHERE dari *query*.
4) Dapat mencampurkan tabel dari *database* yang berbeda pada *query* yang sama.
5) Mampu menangani *database* dengan ukuran besar, MySQL dapat memuat 50.000.000 record dan 60.000 tabel.
6) Memungkinkan alias pada tabel dan kolom, dan sebagainya.

Program-program aplikasi pendukung MySQL, antara lain:

1) PHP (*Page Hypertext Preprosesor*)
2) Visual Delphi
3) Visual Basic

4) Cold Fusion, dll

Selain kelebihan yang dimiliki oleh MySQL, ada beberapa *feature* yang dimiliki oleh system database lain yang belum disediakan oleh MySQL.

Beberapa *feature* yang belum disediakan oleh MySQL diantaranya:

1) Sub select (sub query)
2) Perintah SELECT INTO TABLE

   MySQL tidak men-support Oracle SQL Extension: SELECT INTO TABLE… Untuk memecahkan permasalahan tersebut, dapat menggunakan CREATE TABLE…SELECT…dalam MySQL.
3) *Stored procedure* dan *trigger*

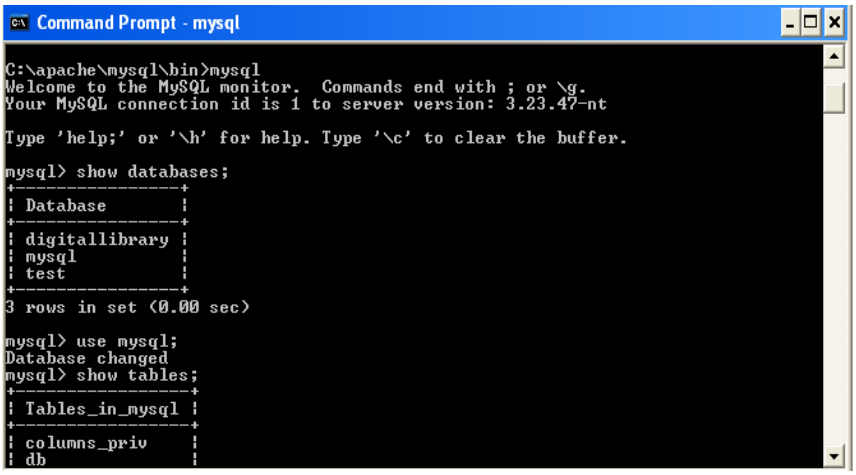
   Stored procedure adalah sekumpulan perintah SQL yang dapat dikompilasi dan disimpan di server. Sekali dilakukan, *client* tidak perlu mengeluarkan keseluruhan *query* tetapi dapat mengacu pada *stored procedure*. Hal ini memungkinkan kinerja yang lebih karena *query*hanya perlu di-parsing sekali dan lebih sedikit informasi yang harus dikirim antara *client* dan *server*.

   Sebuah *trigger* adalah *stored procedure* yang dipanggil saat suatu event tertentu terjadi.

MySQL memiliki layer utama yang seperti layer DOS yaitu memiliki prompt utama yang disebut mysql>

Untuk data-data MySQL di simpan di folder C:\apache\mysql\data.

Contoh Layar MySQL adalah:



Gambar 2.13 Lembar Kerja MySQL

Beberapa Perintah dasar yang sering digunakan pada MySQL antara lain:

1) CREATE DATABASE : Membuat *Database* baru
2) DROP DATABASE : Menghapus *database*
3) CREATE TABLE : Membuat tabel baru
4) DESC TABLE : Deskripsi tabel (kolom)
5) ALTER TABLE : Melakukan modifikasi tabel.
6) DROP TABLE : Menghapus tabel.
7) DELETE : Menghapus isi tabel.
8) INSERT INTO : Memasukkan data dlm tabel.
9) SELECT : Menampilkan isi tabel.
10) UPDATE : Mengubah data pada suatu *field*/kolom.

Tipe Data MySQL

Beberapa tipe data yang didukung uleh MySQL dan sering digunakan adalah sebagai berikut:

1) TEXT /BLOB :String dengan max 2^{23} karakter.
2) CHAR/VARCHAR :String dengan panjang antara 1 sampai 255 karakter.

3) DATE :Tanggal dengan format YYYY-MM-DD.

4) TIME : Waktu dengan format HH:MM:SS.

5) DATETIME :Tanggal dan waktu dengan format YYY-MM-DD HH:MM:SS.

6) FLOAT : Bilangan floating-point.

7) INTEGER : Bilangan integer.

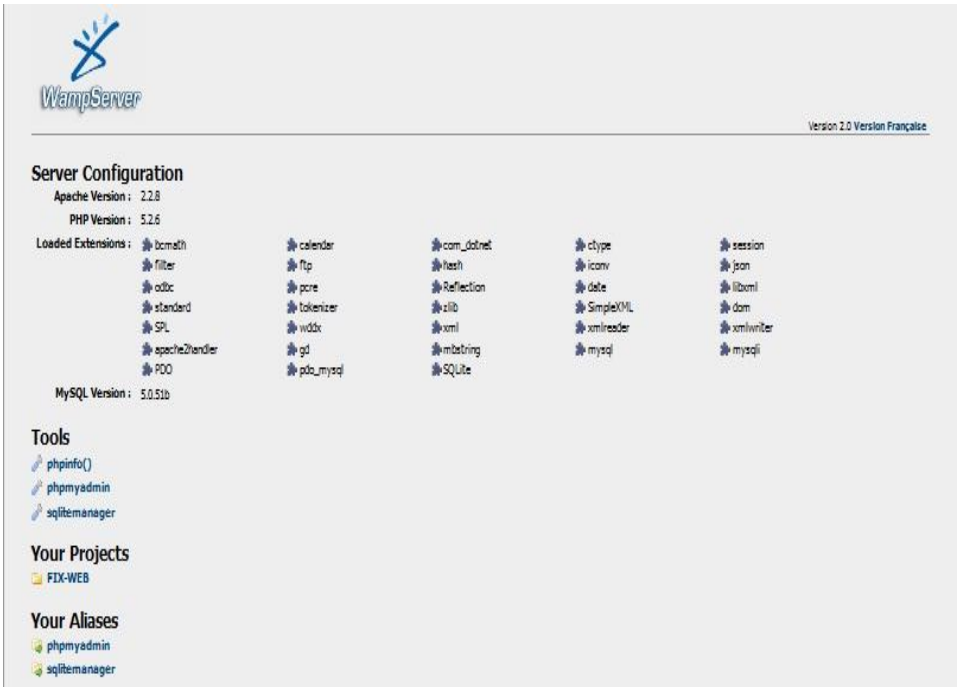
8) ENUM : Enumerasi.

9) TIMESTAMP : Tanggal dan waktu dengan format YYYYMMDDHHMMSS.

2.2.18 Web *Server*

Web Server adalah Perangkat keras dan perangkat lunak yang dipakai untuk menyimpan dan mengirim dokumen HTML untuk digunakan dalam *World Wide Web.*

Web Server yang digunakan dalam pembangunan situs ini adalah Wamp server. Wamp server kepanjangan dari windows, apache, MySQL, dan PHP. Wamp server memiliki kelebihan dalam kecepatan,performa dan gratis.



Gambar 2.14 Tampilan Wamp Server

2.2.19 Web Browser

Web browser adalah program aplikasi yang digunakan untuk menjelajahi dunia *world wide web (www).* Aplikasi ini dapat menterjemahkan bahasa pemprograman web seperti PHP, ASP, Javascript, XML dan lain-lain. *Web Browser* yang terkenal antara lain Internet Explorer, Mozilla Firefox, Opera, dan lain-lain.

Penjelajah web atau biasa disebut *web browser*, disebut juga sebagai perambah atau peramban, adalah perangkat lunak yang berfungsi menampilkan dan melakukan interaksi dengan dokumen-dokumen yang disediakan oleh *server web*. Penjelajah web yang populer adalah Google Chrome dan Mozilla Firefox. Penjelajah web adalah jenis agen pengguna yang paling sering digunakan. Web sendiri adalah kumpulan jaringan berisi dokumen dan tersambung satu dengan yang lain, yang dikenal sebagai *World Wide Web*.

Adapun beberapa istilah yang sering muncul pada saat kita menggunakan web browser adalah sebagai berikut :

Tabel 2.19 Istilah-istilah Web Browser

Istilah	Keterangan
Website	Halaman-halaman web saling terhubung dalam suatu website
Homepage	Halaman awal ketika suatu situs dimunculkan, biasanya juga sebagai penghubung ke website-website lain
URL	Alamat unik pada suatu web yang digunakan web server untuk mengirimkan halaman web tersebut ke komputer untuk mengaksesnya.
WWW	Kumpulan dari dokumen-dokumen elektronik yang kemudian disebut web, tiap dokumen tersebut disebut web page.
Portal	Web yang menyediakan berbagai jenis layanan.