BAB II
TINJAUAN PUSTAKA

II.1 Profil Perusahaan

II.1.1 Sejarah Perusahaan

Anak adalah aset masa depan yang akan menggantikan generasi sekarang. Namun saat ini sulit mendapatkan lembaga yang menyelenggarakan kursus bahasa inggris yang berkualitas dan mudah dijangkau oleh seluruh lapisan masyarakat baik dari segi biaya maupun lokasi. Menghadapi hal tersebut muncul suatu gagasan untuk membuka suatu lembaga pendidikan yang berbasiskan pengajaran bahasa inggris. Sehingga pada tanggal 14 agustus 2011 terbentuklah Lembaga pendidikan tersebut yang diberi nama E-Learning For Kids.

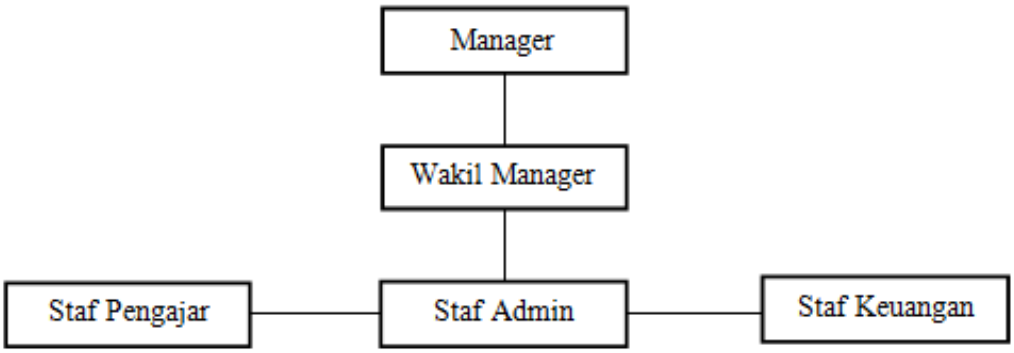
E-learning For Kids adalah penggabungan huruf E yang diambil dari kata *English* yaitu bahasa Inggris, *Learning* yang artinya belajar dan *For Kids* artinya untuk anak-anak. Karena E-Learning For Kids mengusung program E-Learning For Kids (TK – Kelas 6 Sd). E-learning For Kids adalah sebuah lembaga kursus yang berbasiskan bahasa Inggris dalam proses pembelajarannya. Setiap satu unit E-Learning memiliki program pengajaran yang berbeda-beda karena pengelolaannya bersifat mandiri pada masing-masing unit E-Learning.

II.1.2 Struktur Organisasi Perusahaan

Organisasi adalah sekelompok orang yang bekerja sama untuk mencapai tujuan yang telah di rencanakan, dari pengertian tersebut organisasi dapat di simpulkan sebagai kumpulan orang-orang yang bekerja sama dengan dasar persamaan tujuan.

Organisasi perusahaan sangatlah penting didalam menjamin kelangsungan dan kelancaran mekanisme kerja perusahaan, adanya organisasi dapat menciptakan suatu sistem pembagian kerja atau tugas yang sesuai dengan kebutuhan perusahaan sehingga mempermudah kegiatan operasional perusahaan dalam mencapai tujuan.

Untuk lebih jelasnya dapat dilihat pada gambar II.1 dibawah ini yang merupakan struktur organisasi di E-Learning For Kids Cabang Kota bekasi :



Gambar II.1 Struktur Organisasi E-Learning for kids

II.1.3 Deskripsi Tugas

Deskripsi tugas digunakan untuk mengetahui tugas, wewenang dan tanggung jawab dari masing-masing bagian. Adapun deskripsi tugas yang ada di E-Learning for kids adalah sebagai berikut :

1. Manager

   Bertugas sebagai pengawas dan mengkoordinasikan semua pekerjaan yang dilaksanakan oleh para staf E-Learning For Kids. Yang bertujuan untuk menjaga konsistensi kerja dalam mencapai tujuan perusahaan secara efektif dan efisien.

2. Wakil Manager

   Bertugas membantu setiap pelaksaan dari kegiatan E-Learning For Kids.

3. Staf Keuangan

   Bertugas mengkoordinasi serta mengawasi arus masuk dan arus keluar uang perusahaan.

4. Staf Administrasi

   Bertugas mengurus setiap pemberkasan, pendaftaran dan administrasi E-Learning mulai dari pembayaran, promosi, pembukuan dan pembuatan jadwal kegiatan akademik.

5. Staf Pengajar

   Bertugas memberikan materi pengajaran kepada siswa dan ikut membimbing siswa dalam menerapkan setiap materi yang diberikan selama kegiatan belajar berlangsung, baik dilingkungan e-learning maupun kesehariannya.

II.2 Landasan Teori

II.2.1 Basis Data

Basis data terdiri dari dua kata, yaitu basis dan data. Basis dapat diartikan sebagai markas, gudang atau tempat penyimpanan. Data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia, hewan, peristiwa, konsep, keadaan, dan sebagainya, yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi atau kombinasinya. Basis data merupakan himpunan kelompok data yang saling berkaitan dan tempat untuk menampung dan mengorganisasikan seluruh data yang ada dalam sistem, sehingga dapat dieksplorasi untuk menyusun informasi–informasi dalam berbagai bentuk [2].

Dapat disimpulkan bahwa basis data adalah kumpulan data atau arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronik dengan tujuan mempermudah pengambilan data dengan waktu yang cepat.

II.2.2 Pengertian Algoritma Genetik

Algoritma genetik merupakan teknik *search stochastic* yang berdasarkan mekanisme seleksi alam dan genetika natural [3]. Yang membedakan algoritma genetik dengan berbagai algoritma konvensional lainnya adalah bahwa algoritma genetik memulai dengan suatu himpunan penyelesaian acak awal yang disebut populasi. Masing-masing individu didalam populasi disebut kromosom, yang mempresentasikan suatu penyelesaian terhadap masalah yang ditangani. Sebuah kromosom terdiri dari sebuah string yang terdiri berbagai simbol, dan biasanya tidak mutlak, string tersebut berupa sederetan bit-bit biner 0 dan 1. Sebuah kromosom tumbuh dan berkembang biak melalui berbagai iterasi yang berulang-ulang yang disebut juga generasi. Pada masing-masing generasi, berbagai kromosom yang dihasilkan akan di evaluasi menggunakan suatu pengukuran *fitness*. Untuk menghasilkan suatu generasi yang baru, berbagai kromosom baru yang disebut *offspring* dibentuk dengan salah satu cara berikut :

1. Memadukan dua kromosom dari generasi terakhir menggunakan operator *crossover*.

2. Memodifikasi sebuah kromosom menggunakan operator mutasi.

Dan generasi yang baru dihasilkan dengan cara :

1. Penseleksian berdasarkan nilai *fitness*, berbagai *parent* dan *offspring*
2. Menolak yang lainnya untuk menjaga ukuran populasi agar tetap konstan

Berbagai kromosom yang memiliki probabilitas yang tinggi akan diseleksi. Setelah beberapa generasi, algoritma akan menemukan kromosom yang paling baik, yang diharapkan merepresentasikan solusi optimum dan suboptimal terhadap suatu permasalahan optimisasi suatu fungsi [3]. Algoritma ini didasarkan pada proses genetika yang ada dalam makhluk hidup, yaitu perkembangan populasi yang alami, secara lambat laun mengikuti proses seleksi alam atau dengan kata lain siapa yang kuat, dia yang bertahan. Proses seleksi alamiah ini melibatkan perubahan gen yang terjadi pada individu melalui proses perkembangbiakan. Dalam algoritma genetik ini, proses perkembangbiakan ini menjadi proses dasar yang menjadi perhatian utama, dengan dasar berpikir bagaimana mendapatkan keturunan yang lebih baik atau bagaimana menghasilkan keturunan yang optimal

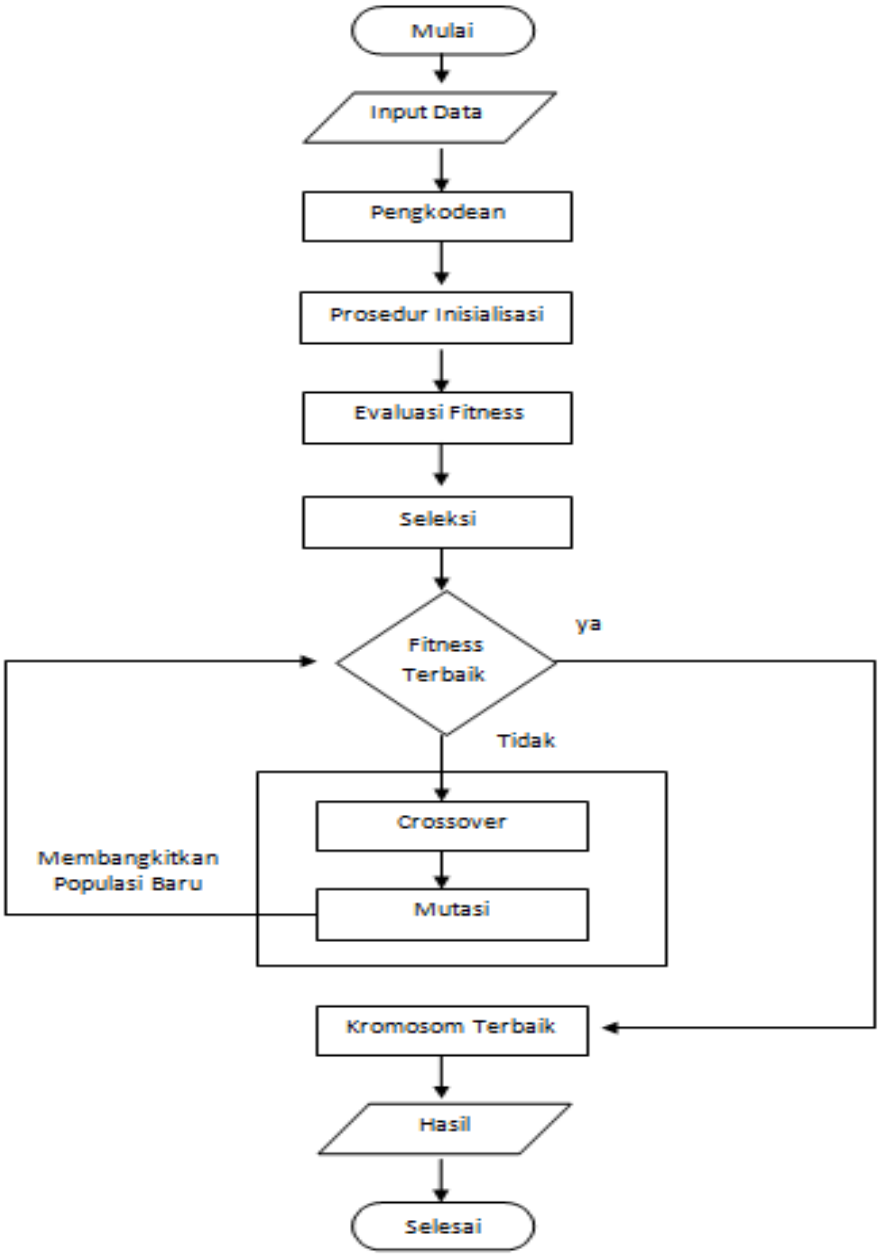
II.2.3 Prosedur Algoritma Genetik

Algoritma Genetika merepresentasikan individu sebagai sebuah kromosom, kromosom ini merupakan suatu solusi yang masih berbentuk simbol. Algoritma ini menyelesaikan permasalahan dalam pencarian kromosom yang terbaik. Kromosom-kromosom terbaik diperoleh dengan cara populasi awal yang dibangun secara acak dan populasi berikutnya yang merupakan hasil evolusi kromosom-kromosom melalui proses iterasi (seleksi, *crossover* dan mutasi) yang disebut dengan istilah generasi. Pada setiap generasi, kromosom akan melalui proses evaluasi dengan menggunakan alat ukur yang disebut dengan fungsi *fitness*. Nilai *fitness* dari suatu kromosom akan menunjukkan kualitas kromosom dalam populasi tersebut. Setelah melalui beberapa generasi, maka akan diperoleh kromosom terbaik [4].

Beberapa definisi penting dalam algoritma genetika adalah sebagai berikut:

a) Gen adalah sebuah nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen yang dinamakan kromosom. Dalam algoritma genetika, gen ini dapat berupa nilai biner, float, integer, kombinatorial maupun karakter.
b) *Allel* adalah nilai dari gen.
c) Kromosom adalah gabungan gen yang membentuk nilai tertentu.
d) Individu menyatakan suatu nilai atau menyatakan salah satu solusi yang mungkin dari permasalahan yang diangkat.
e) Populasi merupakan sekumpulan individu yang akan diproses bersama dalam satu siklus proses evolusi
f) Generasi menyatakan hasil evolusi kromosom-kromosom melalui proses iterasi.
g) Nilai *fitness* menyatakan seberapa baik nilai dari suatu individu atau solusi yang didapatkan.

Prosedur algoritma genetik secara umum dapat dilihat pada flowchart berikut :



Gambar II.2 Flowchart Algoritma Genetik [4]

Dari gambar diatas, struktur umum dari suatu algoritma genetik dapat didefinisikan dengan langkah-langkah sebagai berikut :

1. Pengkodean

   pengkodean adalah suatu teknik untuk menyatakan populasi awal sebagai calon solusi suatu masalah ke dalam suatu kromosom sebagai suatu kunci pokok persoalan ketika menggunakan algoritma genetika.

2. Prosedur Inisialisasi (membangkitkan populasi awal)

   Prosedur inisialisasi (membangkitkan populasi awal) adalah membangkitkan sejumlah individu secara acak atau melalui prosedur tertentu. Ukuran populasi tergantung pada masalah yang akan dipecahkan dan jenis operator genetika yang akan diimplementasikan. Setelah ukuran populasi ditentukan, kemudian harus dilakukan inisialisasi terhadap kromosom yang terdapat pada populasi tersebut. Inisialisasi kromosom dilakukan secara acak, namun demikian harus tetap memperhatikan domain solusi dan kendala permasalahan yang ada.

3. Evaluasi *Fitness*

   Proses ini merupakan proses untuk mengevaluasi setiap populasi dengan menghitung nilai *fitness* setiap kromosom dan mengevaluasinya. Individu yang memiliki nilai *fitness* tinggi pada kromosomnya yang akan dipertahankan, sedangkan individu yang pada kromosomnya bernilai *fitness* rendah akan diganti

4. Seleksi

   Proses seleksi merupakan proses untuk menentukan individu-individu mana saja yang akan dipilih untuk dilakukan *crossover*.

5. *Crossover*

   Proses *crossover* ini merupakan proses untuk menambah keanekaragaman *string* dalam satu populasi.

6. Mutasi

   Mutasi merupakan proses mengubah nilai dari satu atau beberapa gen dalam suatu kromosom.

7. *Fitness* Terbaik

   Fitness terbaik merupakan kriteria yang digunakan untuk menghentikan proses algoritma genetik. Apabila kriteria berhenti belum terpenuhi maka akan dibentuk lagi generasi baru dengan mengulangi langkah 4.

8. Hasil

   Hasil merupakan solusi optimum yang didapat algoritma genetik

II.2.3.1 Teknik Pengkodean

Pengkodean adalah cara untuk mempresentasikan masalah kedalam bentuk kromosom. Ada beberapa cara pengkodean dan pemilihannya tergantung dari masalah yang akan diselesaikan. Teknik pengkodean yang digunakan dalam penelitian tugas akhir ini adalah pengkodean nilai. Berikut ini adalah beberapa jenis pengkodean yang sering digunakan.

1. Pengkodean Biner

Pengkodean biner adalah pengkodean yang paling umum digunakan karena algoritma genetik awalnya menggunakan pengkodean ini. Pada pengkodean biner setiap kromosom merupakan barisan bit 0 atau 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 *crossover* dan mutasi [4].

Contoh masalah yang sesuai menggunakan pengkodean biner adalah masalah ransel (*knapsack*), dimana terdapat beberapa jenis benda dan ukuran yang berat yang berbeda dengan kapasitas ransel terbatas. Permasalahannya adalah memilih sejumlah barang untuk memaksimumkan nilai dari barang-barang tersebut didalam ransel, tetapi total ukuran barang-barang tersebut tidak boleh 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 didalam ransel. Bit 1 berarti benda ada didalam ransel, sedangkan bit 0 berarti benda tidak berada didalam ransel. Contoh pengkodean biner dapat dilihat pada tabel II.1 berikut.

Tabel II.1 Contoh Pengkodean Biner

Kromosom A	1010101011
Kromosom B	1100110010

2. 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 dipresentasikan secara tepat hanya dalam bentuk bit 1 atau 0. Dalam pengkodean nilai, setiap kromosom merupakan barisan dari beberapa nilai. Nilai bisa berupa apa saja, seperti bilangan biasa, bilangan riil, kode, karakter sampai dengan obyek-obyek yang rumit. Pengkodean nilai sangat baik untuk beberapa masalah khusus, dilain sisi *crossover* dan mutasi diperlukan untuk pengkodean ini sering kali memerlukan pengembangan tersendiri yang lebih spesifik pada masalah.

Contoh masalah yang menggunakan pengkodean ini adalah masalah pencarian robot pada jaringan saraf *(neural network)*. Terdapat sebuah jaringan saraf dengan arsiteknya. Permasalahannya adalah mencari bobot masukan dari *neuron-neuron* untuk melatih jaringan untuk 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. Contoh pengkodean nilai dapat dilihat pada tabel II.2 berikut.

Tabel II.2 Contoh Pengkodean Nilai

Kromosom A	1.345, 4.534, 7.654, 8.789
Kromosom B	ABC, ADC, BAC, CAD
Kromosom C	1, 2, 3, 4, 5
Kromosom D	Forward, Backward, Right, Left

3. Pengkodean Permutasi

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

Pengkodean permutasi hanya berguna pada masalah pengurutan. *Crossover* dan mutasi yang dilakukan pada pengkodean ini harus menghasilkan kromosom yang memiliki konsistensi urutan. Masalah yang dapat diselesaikan dengan

pengkodean ini adalah masalah *travelling salesman problem*. Terdapat sejumlah kota dan jarak diantara semua kota tersebut, tapi 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. Contoh pengkodean permutasi dapat dilihat pada tabel II.3 berikut.

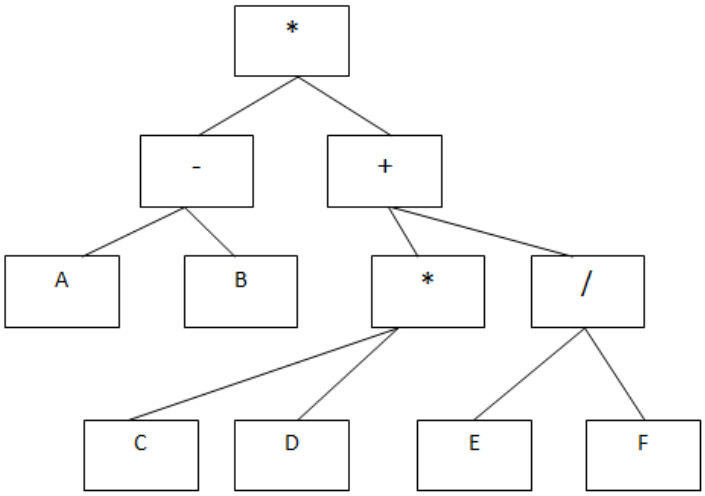
Tabel II.3 Contoh Pengkodean Permutasi

Kromosom A	2 6 7 5 1 3 4 9 8 10
Kromosom B	10 5 4 9 7 1 3 2 6 8

4. Pengkodean Pohon

Pengkodean pohon lebih banyak digunakan untuk menyusun program atau ekspresi bagi pemrograman genetika. Setiap kromosom merupakan pohon dari sejumlah objek, seperti fungsi atau perintah didalam bahasa pemrograman. Pengkodean pohon sangat baik bagi penyusunan program. Bahasa pemrograman *LISP* sering digunakan untuk melakukan pengkodean pohon, karena program didalamnya dapat direpresentasikan kedalam 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 untuk semua masukan. Masalah ini dapat dienkodekan kedalam pengkodean pohon dengan kromosom merupakan fungsi-fungsi yang dinyatakan dalam pohon. Contoh pengkodean pohon dapat dilihat pada gambar II.3 berikut.



gambar II.3 Contoh Pengkodean Pohon

II.2.3.2 Evaluasi *Fitness*

Evaluasi *fitness* merupakan proses untuk mengevaluasi setiap populasi dengan menghitung nilai *fitness* setiap kromosom. Besar atau kecilnya nilai *fitness* dihasilkan dari nilai kromosom tersebut apakah melanggar *coinstraint* atau tidak, jika tidak maka menghasilkan nilai *fitness* yang tinggi. Individu yang memiliki nilai *fitness* tinggi pada kromosomnya yang akan dipertahankan, sedangkan individu yang nilai *fitnessnya* rendah akan diganti atau dibuang. Nilai fitness yang dihasilkan Fungsi yang digunakan untuk menghitung nilai kecocokan atau derajat optimalitas suatu kromosom disebut dengan *fitness function*. Nilai yang dihasilkan oleh fungsi tersebut menandakan seberapa optimal solusi yang diperoleh. Nilai yang dihasilkan oleh fungsi *fitness* merepresentasikan seberapa banyak jumlah persyaratan yang dilanggar. Faktor-faktor yang dapat mempengaruhi evaluasi *fitness* terhadap alternatif solusi tersebut merupakan pelanggaran yang telah ditentukan sebelumnya. Setiap pelanggaran yang terjadi akan diberikan nilai 1, agar tidak terjadi nilai *fitness* yang tak terhingga maka jumlah total semua pelanggaran ditambah 1. Semakin sedikit jumlah pelanggarannya maka jadwal yang dihasilkan akan semakin baik. Fungsi *fitness* digunakan untuk menghitung ke-optimalan suatu jadwal kuliah terhadap batasan yang diberikan. Umumnya, semakin besar nilai *fitness*, semakin optimal jadwal yang dihasilkan. Sehingga, kita dapat menuliskan rumus fungsi *fitness* sebagai berikut:

$$F = \frac{1}{1+(\sum \text{WD} + \sum \text{BR})} \quad \text{(II.1)}$$

Keterangan :

WD : Banyaknya Waktu pengajar yang dilanggar

BR : Banyaknya Bentrok Ruang yang digunakan

II.2.4 Operator Dalam Algoritma Genetik

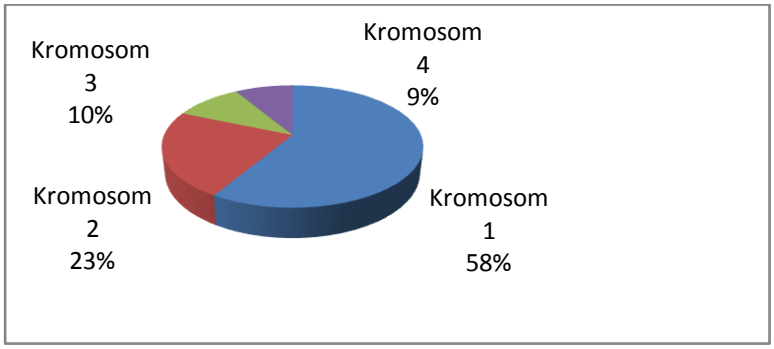
Algoritma genetik merupakan proses pencarian yang heuristik dan acak sehingga penekanan pemilihan operator yang digunakan sangat menentukan keberhasilan algoritma genetik dalam menemukan solusi optimum suatu masalah yang diberikan. Hal yang harus diperhatikan adalah menghindari terjadinya konvergensi prematur, yaitu mencapai solusi optimum yang belum waktunya, dalam arti bahwa solusi yang diperoleh adalah solusi optimum lokal. Ada 3 operator genetik yaitu seleksi, *crossover* dan mutasi.

II.2.4.1Seleksi

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. Seleksi yang digunakan dalam penelitian tugas akhir ini adalah seleksi roda *roulette*

1. Seleksi Roda *Roulette*

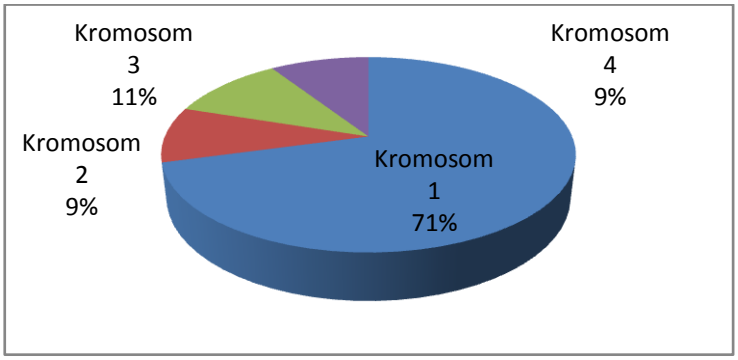
Pada metode seleksi ini, orangtua dipilih berdasarkan kesesuaiannya, semakin baik kromosom, semakin tinggi kesempatannya untuk terpilih. Semua kromosom dari populasi diletakkan pada sebuah roda *roulette*, setiap kromosom memiliki luas bagian yang berdasarkan fungsi kesesuaian yang lebih besar memiliki probabilitas terpilih yang lebih besar. Kekurangan metode ini adalah bila kromosom terbaik memiliki kesesuaian 90% dari seluruh roda *roulette*, maka kromosom yang lainnya akan memiliki kesempatan yang sangat kecil untuk terpilih. Seleksi roda *roulette* dapat dilihat pada gambar II.4 berikut



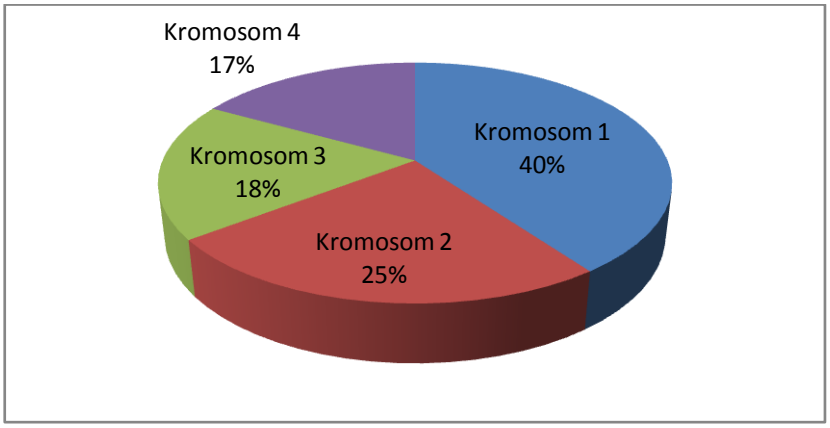
Gambar I.4 Seleksi Roda Roulette

2. Seleksi Rangking

Seleksi ini memperbaiki proses seleksi yang sebelumnya yaitu *roulette wheel* karena pada seleksi tersebut kemungkinan selain satu kromosom mempunyai nilai fitness yang mendominasi hingga 90% bisa terjadi, sehingga nilai *fitness* yang lain akan mempunyai kemungkinan yang sangat kecil untuk terpilih. Seleksi rangking dipakai untuk mengatasi masalah diatas, pertama populasi diurutkan lalu setiap kromosom mendapatkan kesesuaian dari rangking ini. Yang terburuk akan memiliki kesesuaian 1, terburuk kedua memiliki kesesuaian 2, dan seterusnya. Kromosom yang terbaik akan memiliki kesesuaian N (jumlah kromosom dalam populasi). Pada gambar II.5 dan gambar II.6 dapat terlihat bagaimana situasi berubah setelah mengganti kesesuaian dengan nomor urut. Setelah proses ini, semua kromosom memiliki kesempatan untuk terpilih, namun metode ini akan menuntun ke konvergensi yang lebih lambat, karena kromosom yang terbaik tidak jauh berbeda dengan yang lain. Seleksi rangking dapat dilihat pada gambar II.5 dan gambar II.6 berikut



Gambar II.5 Sebelum Rangking



Gambar II.6 Sesudah Rangking

3. Seleksi *Steady State*

Metode ini tidak banyak digunakan dalam proses seleksi karena dilakukan dengan mempertahankan individu yang terbaik. Pada setiap generasi, akan dipilih beberapa kromosom yang memiliki nilai *fitness* terburuk akan digantikan dengan *offspring* yang baru, sehingga pada generasi selanjutnya akan terdapat beberapa populasi yang dipertahankan.

II.2.4.2 *Crossover*

Crossover merupakan proses pada algoritma genetik yang bekerja untuk menggabungkan dua kromosom orang tua (*parent)* menjadi kromosom baru (*offspring)* pada suatu waktu. Sebuah kromosom yang mengarah pada solusi yang baik dapat diperoleh melalui proses *crossover* pada dua buah kromosom. Tidak semua kromosom mengalami persilangan. Cara sederhana pada proses persilangan yaitu dengan memilih suatu titik yang dipisahkan secara *random* dan kemudian membentuk *offspring* dengan cara mengkombinasikan segmen dari satu induk ke sebelah kiri dari titik yang dipisahkan dengan segmen dari induk yang lain ke sebelah kanan dari titik yang dipisahkan. Ada beberapa jenis metode persilangan, berikut ini adalah beberapa metode persilangan yang sering digunakan.

1. *Crossover* Satu Titik

   Proses *crossover* dilakukan dengan memisahkan suatu *string* menjadi dua bagian, kemudian salah satu bagian dipertukarkan dengan salah satu bagian

dari *string* lain yang telah dipisahkan. Misalkan ada 2 kromosom dengan panjang 12 :

Induk 1 : 0 1 1 1 0 | **0 1 0 1 1 1 0**

Induk 2 : **1 1 0 1 0** | 0 0 0 1 1 0 1

Kromosom yang dipilih : 5

Anak 1 : 0 1 1 1 0 | 0 0 0 1 1 0 1

Anak 2 : **1 1 0 1 0** | **0 1 0 1 1 1 0**

2. *Crossover* Banyak Titik

Proses *crossover* ini dilakukan dengan memisahkan suatu *string* menjadi beberapa bagian dan selanjutnya dipertukarkan dengan bagian dari *string* yang lain yang telah dipisahkan. Misalkan ada 2 kromosom dengan panjang 12 :

Induk 1 : **0 1** | **1 1 0 0** | **1 0 1 1** | **1 0**

Induk 2 : 0 1 | 0 1 0 0 | 0 0 1 1 | 0 1

Posisi yang dipilih : kromosom setelah 2, 6 dan 10

Kromosom baru yang terbetuk :

Anak 1 : **0 1** | 0 1 0 0 | **1 0 1 1** | 0 1

Anak 2 : 1 1 | **1 1 0 0** | 0 0 1 1 | **1 0**

3. *Crossover* Aritmatika

Suatu operasi aritmatika digunakan untuk menghasilkan *offspring* yang baru. Kromosom baru dihasilkan dari operasi aritmatika yang dilakukan terhadap kedua orangtuanya.

Contoh :

Kromosom *Parent* 1 : 11001011

Kromosom *Parent* 2 : 11011111

Offspring : 11001001

4. *Crossover* Pengkodean Permutasi

Prosesnya adalah dengan memilih satu titik tertentu, dari permutasi pertama sampai dengan titik *crossover* disalin dari orangtua pertamanya, sedangkan sisanya didapatkan dengan cara melihat satu persatu nilai pada orang tua

kedua, jika belum ada pada kromosom keturunan, maka nilai tersebut ditambahkan.

Contoh :

Kromosom *Parent* 1 : **12345**6789

Kromosom *Parent* 2 : 4**53689**721

Offspring : 12345689

II.2.4.3 Mutasi

Mutasi merupakan proses untuk mengubah salah satu atau lebih gen dari suatu kromosom. Proses ini berperan untuk menggantikan gen yang hilang dari populasi akibat proses seleksi yang memungkinkan munculnya kembali gen yang tidak muncul pada inisialisasi populasi. Beberapa cara operasi mutasi yang diterapkan dalam algoritma genetika menurut jenis pengkodeannya, antara lain :

1. Mutasi Dalam Pengkodean Biner

   Mutasi pada pengkodean biner merupakan operasi yang sangat sederhana. Proses yang dilakukan adalah menginversi nilai bit pada posisi tertentu yang dipilih secara acak (atau dengan menggunakan skema tertentu) pada kromosom. 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. 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 terdapat 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. Contoh mutasi pada pengkodean nilai :

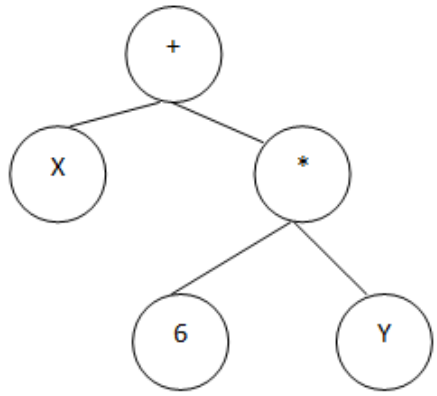
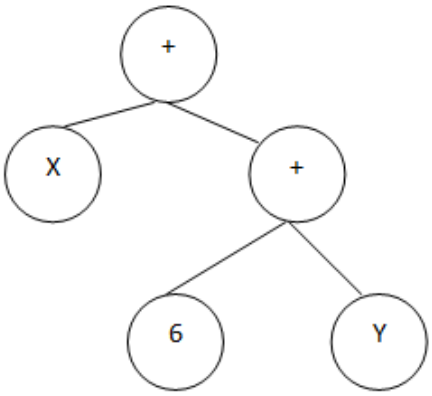
Kromosom sebelum mutasi : **1,45** 2,67 **1,87** 2,56

Kromosom sesudah mutasi : **1,55** 2,67 **1,77** 2,56

3. Mutasi Pada Pengkodean Pohon

Mutasi dapat dilakukan dengan mengubah operator (+, -, *, /) atau nilai yang terkandung pada suatu *vertex* pohon yang dipilih, atau juga dapat dilakukan pemilihan dua *vertex* dari pohon dan saling dipertukarkan operator atau nilainya. Contoh mutasi pada pengkodean nilai adapat dilihat pada tabel II.4 berikut.

Tabel II.4 Contoh Mutasi Pada Pengkodean Mutasi

Sebelum Mutasi	**Sesudah Mutasi**
	
(+ x (* 6 y))	(+ x (+ 6 y))

4. Mutasi Pada Pengkodean Permutasi

Proses mutasi pada pengkodean biner tidak dapat digunakan pada pengkodean permutasi karena konsistensi urutan permutasi harus selalu diperhatikan. Salah satu cara yang dapat dilakukan adalah dengan memilih dua posisi pada kromosom dan kemudian saling dipertukarkan. Contoh mutasi pada pengkodean permutasi :

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

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

II.2.5 Parameter Dalam Algoritma Genetik

Dua parameter dasar yang penting dari algoritma genetika adalah probabilitas *crossover* dan probabilitas mutasi.

II.2.5.1 Probabilitas Crossover

Probabilitas *crossover* akan mengendalikan operator *crossover* dalam setiap generasi dalam populasi yang mengalami *crossover*. Semakin besar nilai probabilitas *crossover*, akan semakin cepat struktur individu baru terbentuk ke dalam populasi. Sedangkan apabila nilai probabilitas *crossover* terlalu besar, individu yang merupakan kandidat solusi terbaik mungkin akan dapat hilang lebih cepat pada generasi selanjutnya. Disarankan nilai probabilitas *crossover* berkisar antara 80 % - 95 %.

II.2.5.2 Probabilitas Mutasi

Probabilitas mutasi akan mengendalikan operator mutasi pada setiap generasi dimana peluang mutasi yang digunakan biasanya lebih kecil daripada peluang *crossover*. Pada seleksi alam murni, mutasi jarang sekali muncul. Oleh karena itu, operator mutasi pada algoritma genetik juga tidak selalu terjadi. Untuk itulah nilai peluang mutasi dibuat lebih kecil untuk setiap generasi. Disarankan nilai probabilitas mutasi kecil berkisar antara 0.5 % - 1 %.

Parameter lain yang juga ikut menentukan efisiensi kinerja algoritma genetik adalah ukuran populasi, yaitu banyaknya kromosom dalam satu populasi. Tidak ada aturan yang pasti tentang berapa nilai ukuran populasi. Apabila ukuran populasi kecil berarti hanya tersedia sedikit pilihan untuk *crossover* dan sebagian kecil dari domain solusi saja yang dieksplorasi untuk setiap generasinya. Sedangkan apabila terlalu besar, kinerja algoritma genetik akan menurun. Penelitian menunjukan ukuran populasi besar tidak mempercepat pencarian solusi. Disarankan ukuran populasi berkisar antara 20–30.

II.2.6 Algoritma *Tabu Search*

Tabu Search adalah sebuah metode optimasi yang berbasis pada *local search*. Proses pencarian bergerak dari satu solusi ke solusi berikutnya, dengan cara memilih solusi terbaik dari *neighborhood* solusi sekarang (*current*) yang tidak tergolong solusi terlarang (*tabu*). Ide dasar dari algoritma *tabu search* adalah

mencegah proses pencarian dari *local search* agar tidak melakukan pencarian ulang pada ruang solusi yang sudah ditelusuri dengan memanfaatkan suatu struktur memori yang mencatat sebagian jejak proses pencarian yang telah dilakukan [5].

Struktur memori *fundamental* dalam *tabu search* dinamakan *tabu list. Tabu list* menyimpan atribut dari sebagian *move* (transisi solusi) yang telah diterapkan pada iterasi-iterasi sebelumnya. *Tabu search* menggunakan *tabu list* untuk menolak solusi-solusi yang memenuhi atribut tertentu guna mencegah proses pencarian mengalami pengulangan pada daerah solusi yang sama, dan menuntun proses pencarian menelusuri daerah solusi yang belum dikunjungi. Tanpa menggunakan strategi ini, *local search* yang sudah menemukan solusi optimum lokal dapat terjebak pada daerah solusi optimum lokal tersebut pada iterasi-iterasi berikutnya.

Perekaman solusi secara lengkap dalam sebuah *forbidden list* dan pengecekan apakah sebuah kandidat solusi tercatat dalam list tersebut merupakan cara yang mahal, baik dari sisi kebutuhan memori maupun kebutuhan waktu komputasi. Jadi *tabu list* hanya menyimpan langkah transisi (*move*) yang merupakan lawan atau kebalikan dari langkah yang telah digunakan dalam iterasi sebelumnya untuk bergerak dari satu solusi ke solusi berikutnya. Dengan kata lain *tabu list* berisi langkah-langkah yang membalikkan solusi yang baru ke solusi yang lama.

Pada tiap iterasi, dipilih solusi baru yang merupakan solusi terbaik dalam *neighborhood* dan tidak tergolong sebagai *tabu*. Kualitas solusi baru ini tidak harus lebih baik dari kualitas solusi sekarang. Apabila solusi baru ini memiliki nilai fungsi objektif lebih baik dibandingkan solusi terbaik yang telah dicapai sebelumnya, maka solusi baru ini dicatat sebagai solusi terbaik yang baru. Sebagai tambahan dari *tabu list*, dikenal adanya kriteria aspirasi, yaitu suatu penanganan khusus terhadap *move* yang dinilai dapat menghasilkan solusi yang baik namun *move* tersebut berstatus *tabu*. Dalam hal ini, jika *move* tersebut memenuhi kriteria aspirasi yang telah ditetapkan sebelumnya, maka *move* tersebut dapat digunakan untuk membentuk solusi berikutnya (status tabunya dibatalkan) [6].

II.2.6.1 Ide Dasar *Tabu Search*

Sebuah informasi akan digunakan sebagai petunjuk untuk bergerak dari I ke solusi selanjutnya dalam N(i). Penggunaan memori sebagai pembatas dalam pemilihan beberapa *subnet* dari N(i) dengan mencegah pergerakan kebeberapa solusi tetangga. Sebuah informasi akan digunakan sebagai petunjuk untuk bergerak dari i ke solusi selanjutnya dalam N(i). Metode *tabu search* akan menjadi sebuah algoritma yang menyediakan proses eksplorasi yang menjamin setelah sejumlah langkah-langkah solusi dapat dicapai.

Prinsip kerja *tabu search* dapat dilakukan dengan metode menurun klasik (*classical decent method)*, yaitu :

1. Memilih sebuah langkah awal *i* dalam *S*
2. Membangkitkan subset *V** sebagai solusi dalam *N(i)*
3. Mencari *j* 'terbaik' dalam *V** dan menetapkan *i=j*
4. Jika *f(j)-f(i)* maka berhenti, namun jika tidak kembali kelangkah ke-2.

Dalam metode menurun secara umum akan dapat langsung ditetapkan bahwa *V* = N(i)*. Tetapi hal ini seringkali membutuhkan waktu yang lama, untuk itulah cara pemilihan *V** yang tepat seringkali dijadikan sebagai peningkatan yang penting dalam metode pencarian.

Sebagai pengecualian pada kasus-kasus istimewa, penggunaan prosedur menurun (*descent procedure*) secara umum lebih rumit karena kita akan terperangkap pada sebuah minimum lokal yang mungkin masih jauh dari minimum global. Maka untuk proses iterasi dalam eksplorasi apapun sebaiknya dalam beberapa hal juga menerima langkah-langkah yang tidak akan memberikan perkembangan dari i ke j dalam V* (misal f(j)>f(i)). Metode *tabu search* secara berbeda memilih j 'terbaik' dalam V*. Selama pergerakan yang tidak memberi perkembangan itu mungkin, resiko pengunjungan kembali sebuah solusi atau lebih umumnya disebut sebagai siklus mungkin untuk terjadi. Dalam hal inilah penggunaan memori sangat diperlukan untuk mencegah terjadinya pergerakan ke solusi yang telah dikunjungi. Jika memori seperti itu diperkenalkan, maka kita dapat menganggap struktur N(i)tergantung pada pengelilingan yang merupakan pengulangan k. Jadi dapat merujuk ke N(i,k) daripada N(i). Dengan perujukan ini

dapat mencoba untuk melakukan peningkatan algoritma menurun dalam beberapa hal untuk lebih mendekatkan metode ini ke prosedur dalam metode *tabu search* secara umum

II.2.7 Internet

Internet *(interconnected Network*) merupakan jaringan *(network)* komputer yang terdiri dari ribuan jaringan komputer independen yang dihubungkan satu dengan yang lainnya. Jaringan komputer ini dapat terdiri dari pendidikan, pemerintahan, militer, organisasi bisnis dan organisasi-organisasi lainnya [7].

II.2.7.1 Perkembangan Internet

Jumlah pengguna internet yang besar dan semakin berkembang, telah mewujudkan budaya internet. Internet juga mempunyai pengaruh yang besar atas ilmu, dan pandangan dunia. Dengan hanya menggunakan mesin pencari seperti Google, pengguna di seluruh dunia mempunyai akses yang mudah atas bermacam-macam informasi. Dibanding dengan buku dan perpustakaan, internet melambangkan penyebaran (*decentralization*) informasi dan data secara ekstrim.

Terkait dengan teknologi pembuatan *website* sebagai konten dari teknologi internet itu sendiri memicu perkembangan teknik pembuatan *website* yang semakin canggih dengan fitur-fitur yang sangat beragam tidak hanya terbatas pada teks dan gambar saja, akan tetapi sudah merambah kedalam dunia multimedia dan telekomunikasi visual. Bahkan dengan munculnya teknologi *content* manajemen sistem pengguna yang awam dengan kemampuan teknis yang sedikipun bisa membuat *website* sesuai dengan keinginannya tanpa harus berhubungan dengan *webmaster* dengan *content* yang dinamis dan selalu *uptodate* [8].

II.2.8 *Entity Relationship Diagram (ERD)*

Entity Relationship Diagram atau biasa dikenal dengan diagram E-R berisi komponen-komponen himpunan entitas dan himpunan relasi yang masing-masing dilengkapi dengan atribu-atribut yang merepresentasikan seluruh fakta dari dunia nyata.

Elemen-elemen *Entity Relationship Diagram* adalah sebagai berikut [9].

1. *Entity* (Entitas)

   Pada diagram E-R, entitas digambarkan dengan sebuah persegi panjang. Entitas adalah individu yang mewakili sesuatu yang nyata dan dapat dibedakan dari sesuatu lainnya. Entitas diberi nama dengan kata benda dan dapat dikelompokan dalam empat jenis nama, yaitu: orang, benda, lokasi kejadian (terdapat unsur waktu didalamnya).

2. *Relationship* (Relasi)

   Pada diagram E-R, relasi dapat digambarkan dengan sebuah belah ketupat. Relasimenunjukan adanya hubungan di antara satu entitas dengan entitas lainnya. Pada umumnya relasi diberi nama dengan kata kerja dasar, sehinga memudahkan untuk melakukan pembacaan relasinya.

3. Atribut

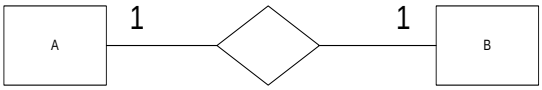
   Pada diagram E-R atribut adalah segala sesuatu yang mendeskripsikan karakteristik dari entitas.

4. Kardinalitas/Derajat Relasi

   Kardinalitas relasi menunjukan jumlah maksimum tupel yang dapat berelasi dengan entitas yang lainnya. Dari sejumlah kemungkinan banyaknya hubungan yang terjadi dari entitas, kardinalitas relasi merujuk kepada hubungan maksimum yang terjadi dari entitas yang satu ke entitas yang lainnya dan begitu juga sebaliknya. Kardinalitas/ Derajat Relasi yang terjadi di antara dua entitas sebagai berikut:

   a. One to one Relationship

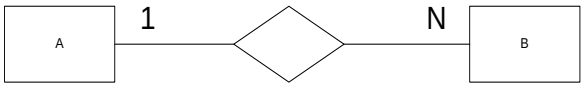
      Tingkat hubungan satu ke satu, dinyatakan dengan satu kejadian pada entitas pertama, hanya mempunyai satu hubungan dengan satu kejadian pada entitas yang kedua dan sebaliknya. Gambar II.7 berikut adalah ilustrasi dari *One to one Relationship.*



Gambar II.7 *One to One Relationship*

b. One to many Relationship

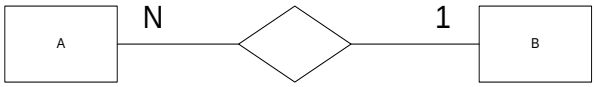
Tingkat hubungan satu ke banyak adalah untuk satu kejadian pada entitas yang pertama dapat mempunyai banyak hubungan dengan kejadian pada entitas yang kedua.



Gambar II.8 *One to Many Relationship*

c. Many To One Relationship

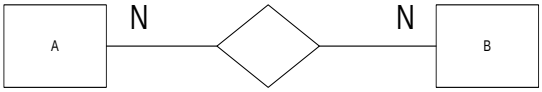
Untuk banyak kejadian pada entitas yang pertama hanya dapat mempunyai satu hubungan dengan kejadian pada entitas yang kedua. Gambar II.9 berikut adalah ilustrasi dari *Many To One Relationship*.



Gambar II.9 *Many to One Relationship*

d. Many to many Relationship

Tingkat hubungan banyak ke banyak terjadi jika tiap kejadian pada sebuah entitas akan mempunyai banyak hubungan dengan kejadian pada entitas lainnya, baik dilihat dari sisi entitas yang pertama maupun dilihat dari sisi yang kedua. Gambar II.10 berikut adalah ilustrasi dari *Many to many Relationship*.



Gambar II.10 *Many to Many Relationship*

5. *Key* (Kunci)

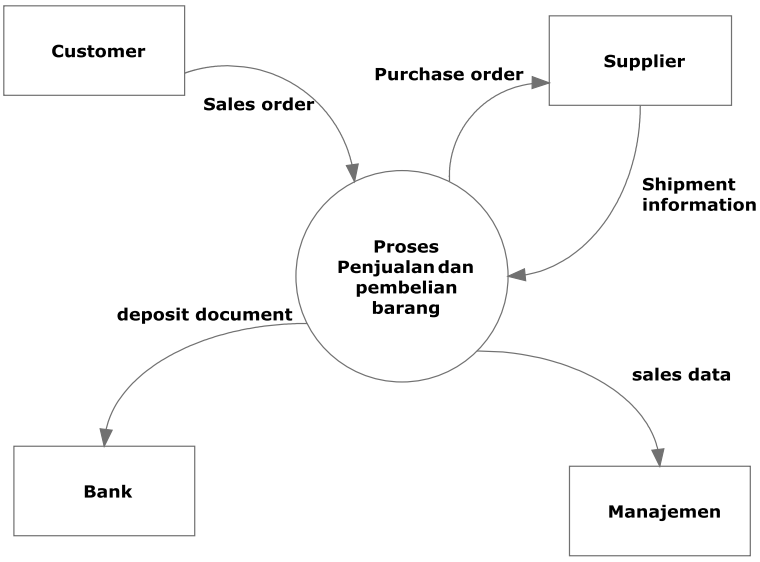
Sebuah atribut atau set atribut yang nilainya mengidentifikasikan entitas secara unik dalam suatu entitas. *Key* memiliki beberapa jenis sesuai dengan

kegunaannya masing-masing, yaitu *primary key* (kunci utama), *foreign key* (kunci tamu).

II.2.9 Diagram Konteks

Diagram konteks adalah sebuah diagram sederhana yang menggambarkan hubungan antara entitas luar, masukan dan keluaran dari sistem [9]. Dalam penggambaran ini, sistem dianggap sebagai sebuah objek yang tidak dijelaskan secara rinci, karena yang ditekankan adalah interaksi sistem dengan lingkungan yang mengaksesnya.

Diagram konteks merupakan gambaran umum sistem yang akan dibangun. Dapat dikatakan bahwa diagram konteks itu berisi siapa saja yang memberikan data (*input)* ke sistem serta kepada siapa data informasi yang harus dihasilkan sistem. Gambar II.11 berikut adalah contoh gambar diagram konteks.



Gambar II.11 Contoh Diagram Konteks [9]

II.2.10 Data Flow Diagram (DFD)

Data flow diagram adalah suatu model logika data atau proses yang dibuat untuk menggambarkan dari mana asal data dan kemana tujuan data yang keluar dari sistem, dimana data tersimpan, proses apa yang menghasilkan data tersebut dan interaksi antara data tersimpan dan proses yang dikenakan pada data tersebut

[9]. DFD adalah suatu grafik yang menjelaskan sebuah sistem dengan menggunakan bentuk-bentuk atau simbol untuk menggambarkan input, proses dan output yang terjadi dalam suatu sistem. DFD juga menggambarkan aliran data dalam sebuah sistem. Dalam mendokumentasikan sebuah sistem, DFD mempunyai level-level mulai dari yang terkecil, yaitu level 0 atau sering disebut diagram konteks. Diagram konteks merupakan gambaran paling umum dari sistem, yang hanya memiliki satu proses saja untuk mewakili seluruh sistem.

II.2.11 Kamus Data

Kamus data adalah peralatan yang ikut berperan dalam perancangan dan membangun sistem informasi karena berfungsi untuk menjelaskan arti aliran data dan penyimpanan dalam penggambaran pada *data flow diagram*, mendeskripsikan komposisi paket data yang bergerak melalui aliran, dan menjelaskan spesifikasi nilai dan satuan yang relevan terhadap data yang mengalir dalam sistem tersebut [9].

II.2.12 Tinjauan Perangkat Lunak

II.2.12.1MYSQL

MySQL adalah *database* yang menghubungkan *script* PHP menggunakan perintah *query* dan *escaps character* yang sama dengan PHP. MySQL mempunyai tampilan *client* yang dapat mempermudah dalam mengakses *database* dengan kata sandi untuk mengijinkan proses yang boleh dilakukan [10].

MySQL adalah *Relational Database Management System* (RDMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public License*). Dimana setiap orang bebas untuk meggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat *closed Source* atau komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam *database* sejak lama, yaitu SQL (*Structured Query Languange*). SQL adalah sebuah konsep pengoprasian *database*, terutama untuk pemulihan atau seleksi dan pemasukan data, yang memungkinkan pengoprasian data dikerjakan dengan mudah secara otomatis. Keandalan suatu sistem database (DBMS) dapat diketahui dari cara kerja *optimizer*-nya dalam melakukan proses perintah-perintah SQL, yang dibuat

oleh *user* maupun program-program aplikasinya. Sebagai *database server*, MySQL dapat dikatakan lebih unggul dibandingkan *database server* lainya dalam *query* data. Hal ini terbukti untuk *query* yang dilakukan oleh *single user*, kecepatan *query* MySQL bisa sepuluh kali lebih cepat dari PostgreSQL dan limakali lebih cepat dibandingkan Interbase. Selain itu MySQL juga memiliki beberapa keistimewaan, antara lain [10]:

1. *Protability*

   MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows,Linux, FreeBSD, Mac Os X Server, Solaris, Amigan, dan masih banyak lagi.

2. *Open Source*

   MySQL didistribusikan secara gratis, dibawah lisensi GPL sehingga dapat digunakan secara percuma.

3. *Multi-user*

   MySQL dapat digunakan oleh berberapa *user* dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.

4. *Performance tuning*

   MySQL memiliki kecepatan yang menakjubkan dalam mengenai *query* sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.

5. *Column type*

   MySQL memiliki tipe kolom yang sangat kompleks, seperti *signed/unsignedinteger, float, char, double, text, date, timestamp*, dan lain-lain.

6. *Command and functions*

   MySQL memiliki operator fungsi secara penuh yang mendukung perintah *select* dan *where* dalam *query*.

7. *Security*

   MySQL memiliki beberapa lapisan sekuritas seperti level *subnetmask*, nama *host*, dan izin akses *user* dengan sistem perizinan yang mendetail serta *password* terenkripsi.

8. *Scability dan limits*

   MySQL mampu menangani *database* dalam skala besar, dengan jumlah *record* lebih dari 50 juta 60 ribu tabel serta 5 milyar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.

9. *Connectivity*

   MySQL dapat melakukan koneksi dengan *client* menggunakan protokol TCP/IP, Unix soket (UNIX), atau Named Pipes (NT).

10. *Localization*

    MySQL dapat mendeteksi pesan kesalahan pada *client* dengan menggunakan lebih dari dua puluh bahasa. Meskipun demikian, bahasa Indonesia belum termasuk didalamnya.

11. *Interface*

    MySQL memiliki *interface* (antar muka) terhadap berbagai aplikasi dan bahasa pemograman dengan menggunakan fungsi API (*Application Programming Interface*).

12. *Client* dan *tools*

    MySQL dilengkapi dengan berbagai *tool* yang dapat digunakan untuk administrasi *database*, dan pada setiap *tool* yang ada disertakan petunjuk *online*.

II.2.13 Adobe Dreamweaver

Dreamweaver merupakan perangkat lunak untuk membuat suatu *website*. Pada awalnya dreamweaver merupakan produk dari Macromedia Inc, dan versi pertama dirilis pada tahun 1997. Versi terakhir dreamweaver yang dikeluarkan oleh Macromedia Inc adalah versi 8 yaitu pada tahun 2005. Kemudian pada tahun 2007 Macromedia Inc dibeli oleh Adobe Systems. Dan versi terakhir dreamweaver yang dikeluarkan oleh Adobe Systems adalah versi 11.5 yang ada dalam Adobe Creative Suite 5.5 dan sering disingkat dengan Adobe CS5.5 pada tahun 2011.

Sejak awal keluar, dreamweaver menjadi *web* editor yang banyak digunakan oleh para *web developer*. Hal ini antara lain karena kemudahan dalam penggunaannya, fasilitas, dan kemampuan dreamweaver yang *powerful* dan menunjang peningkatan produktifitas dan efektifitas untuk mendesain atau membangun *website*. Dreamweaver juga dilengkapi dengan fasilitas untuk manajemen situs yang cukup lengkap kelengkapan fiturnya dan juga dukungannya terhadap teknologi terkini.

II.2.14 Personal Home Page (PHP)

PHP adalah salah satu bahasa *server-side* yang didesain khusus untuk aplikasi *web*.PHP dulunya merupakan proyek pribadi dari Rasmus Lerdorf (dengan dikeluarkannya php versi 1) yang digunakan untuk membuat *homepage* pribadinya. Versi pertama ini berupa kumpulan *script* PERL. Untuk versi keduanya, Rasmus menulis ulang *script-script* PERL tersebut menggunakan bahasa C, kemudian menambahkan fasilitas untuk *form* html dan koneksi MYSQL. Adapun PHP didapat dari singkatan *Personal Home Page*.

Dengan menggunakan PHP maka *maintenance* suatu situs *web* menjadi lebih mudah. Proses *update* data dapat dilakukan dengan menggunakan aplikasi yang dibuat dengan menggunakan script PHP. PHP/FI merupakan nama awal dari PHP. PHP/FI adalah *Form Interface*, dibuat pertama kali oleh Rasmus Lerdoff. PHP, awalnya merupakan program CGI yang dikhususkan untuk menerima *input* melalui form yang ditampilkan dalam *web*.

PHP secara dasar dapat mengerjakan semua yang dapat dikerjakan oleh program CGI, seperti mendapatkan data dari *form*, menghasilkan isi halaman *web* yang dinamis, dan menerima *cookies.* Kemampuan (*Feature*) PHP yang paling diandalkan dan signifikan adalah dukungan kebanyak *database.* Membuat halaman *web* yang menggunakan data dari *database* dengan sangat mudah dapat dilakukan [10].