BAB II
DASAR TEORI

Pada bab ini akan dijelaskan mengenai dasar teori yang digunakan untuk penulisan tugas akhir. Dasar teori yang akan dijelaskan adalah konsep dasar perancangan basis data, *Data Flow Diagram (DFD)*, konsep dasar dari sistem informasi manajemen dan konsep dasar dari desain *user interface.*

2.1. PERANCANGAN BASIS DATA

Perancangan basis data harus dilakukan terlebih dahulu sebelum membuat suatu perangkat lunak. Perancangan ini menjelaskan mengenai pengertian dari basis data dan sistem manajemen basis data, *Entity Relationship Diagram (ER-D)*, *mapping* dan normalisasi.

2.1.1. Pengertian Basis Data dan Sistem Manajemen Basis Data

Sebelum mempelajari lebih jauh lagi mengenai perancangan basis data, terlebih dahulu harus diketahui pengertian dari pada basis data itu sendiri. Basis data adalah suatu kumpulan dari data yang saling berhubungan satu sama lain (Elmasri dan Navathe 2000). Dengan menggunakan data maka suatu keadaan atau kenyataan dapat disimpan dan mempunyai arti.

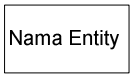
Kumpulan data yang terhubung tersebut disimpan secara bersama-sama pada suatu media, sehingga mudah untuk digunakan atau ditampilkan kembali, dapat digunakan oleh satu atau lebih program aplikasi secara optimal, data dismpan tanpa mengalami ketergantungan pada program yang menggunakannya, data disimpan sedemikian rupa sehingga penambahan, pengurangan, dan modifikasi data dapat dilakukan dengan mudah dan terkontrol.

Sedangkan arti dari *Database Management System* (DBMS) adalah kumpulan dari program-program yang memungkinkan pengguna program untuk membuat dan memelihara suatu basis data (Elmasri dan Navathe 2000). DBMS merupakan tujuan dari sistem perangkat lunak untuk mendefinisikan, membuat, dan memanipulasi basis data untuk berbagai macam aplikasi. Mendefinisikan

basis data dalam arti membuat spesifikasi tipe data, struktur data dan membatasi jenis data yang dapat disimpan dalam basis data. Membuat basis data adalah proses penyimpanan data itu sendiri didalam media penyimpanan yang dikontrol oleh DBMS. Memanipulasi suatu basis data termasuk mencari data (*query*), memodifikasi data (*update*) dan membuat laporan (*report*). Gabungan dari basis data dan perangkat lunak untuk memanipulasi basis data disebut sebagai sistem basis data.

2.1.2. Entity Relationship Diagram (ER-D)

Entity Relationship Diagram adalah suatu diagram yang menggambarkan hubungan-hubungan dari *entity-entity* (diagram hubungan antar entitas). ER Diagram seringkali digunakan untuk mendesain konsep dari suatu database dan juga banyak perangkat yang dari ER menggambarkan data sebagai *entity, relationship* dan *attribute*. *Entity* merupakan obyek dasar pada ER, yaitu suatu objek yang berbeda dengan obyek lainnnya yang berdiri sendiri. *Entity* terdiri dari 2 jenis, yaitu *strong entity* dan *weak entity*. *Strong entity* atau *entity* adalah obyek yang dapat dibedakan dengan obyek lainnya dan tidak saling bergantung satu dengan lainnya. Sedangkan *weak entity* merupakan *entity* yang tidak mempunyai *key attribute* dan bergantung pada entity lainnya. Sebuah *weak entity* hanya dapat bergantung pada sebuah *instance* tunggal dari *entity* dimana ia bergantung. Setiap entity memiliki atribut, beberapa tipe dari atribut, antara lain: *simple* dan *composite attributes, single* dan *multivalued attributes, stored* dan *derived attributes*. *Simple attributes* tidak dapat dibagi lagi, jika *composite attributes* dapat dibagi menjadi beberapa *attribute* lagi. *Single valued attributes* merupakan atribut yang hanya memiliki satu nilai saja, contoh mobil yang hanya memiliki satu warna. Sedangkan *multivalued attributes* mempunyai lebih dari satu nilai atribut, contoh mobil yang memiliki lebih satu warna. *Stored attributes* merupakan atribut yang sudah memiliki nilai, sedangkan untuk *derived attributes* nilainya diperoleh dari *stored attributes.* Berikut ini akan dijelaskan mengenai beberapa simbol pada ER yang digunakan (Elmasri dan Navathe 2000)**:**

1. 

Entity

Disebut juga sebagai *Strong Entity* (berdiri sendiri), *strong entity* adalah obyek yang dapat dibedakan dengan obyek lainnya dan tidak saling bergantung satu dengan yang lainnya.

2. 

Weak Entity

Weak Entity merupakan *entity* yang tidak mempunyai *key attribute* dan ia bergantung pada *entity* lainnya. Sebuah *weak entity* hanya dapat bergantung pada sebuah *instance* tunggal dari *entity* dimana ia bergantung.

3. 

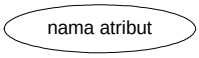
Relationship

Sebuah *relationship* menyatakan sebuah hubungan antara beberapa *entity*. Sebuah *relationship* tidak dapat berdiri sendiri tetapi harus dihubungkan ke paling sedikit 2 *entity* melalui sebuah penghubung.

4. 

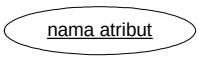
Indentifying Relationship

Indentifying Relationship digunakan untuk menggambarkan hubungan antara sebuah *entity* dengan *weak entity*.

5. 

Attribute

Attribute merupakan sebuah fungsi yang memetakan sebuah himpunan *entity* ke dalam sebuah domain. Domain adalah sebuah himpunan yang berisi nilai-nilai yang dibutuhkan.

6. 

Key Attribute

Key Attribute adalah sebuah *attribute* yang unik dari sebuah *entity*, karena nilai dari *key attribute* ini akan berbeda untuk masing-masing *entity* dari suatu hmpunan *entity*.

7. *Multivalue Attribute*

Multivalue Attribute adalah *attribute* dari suatu *entity* yang mempunyai nilai lebih dari satu.

8. *Garis (Partial Participation)*

Garis digunakan untuk menghubungkan dua buah simbol. Garis tunggal demikian dapat pula disebut sebagai *Partial Participation*, arti daripada *Partial Participation* adalah hanya sebagian *instance* dari *entity* yang berhubungan dengan *instance entity* yang dihubungkan dengan *relationship* tersebut.

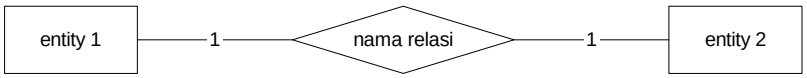
9. *Total Participation*

Total Participation ini menggambarkan seluruh *instance* dari suatu *entity* pasti berhubungan dengan *instance entity* yang dihubungkan dengan *relationship* tersebut.

Relationship menghubungkan antara *entity* yang satu dengan *entity* yang lain. *Relationship* mempunyai 2 batasan, yaitu total dan parsial. Sedangakan *cardinality* untuk *relationship* ada 3, yaitu 1:1, 1:N, dan M:N. Relationship juga memiliki atribut, sama dengan entity. Ada 3 macam *cardinality* pada *relationship*, yaitu:

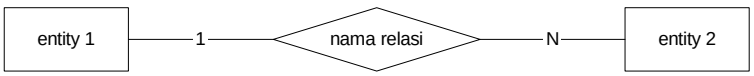
- Relasi dengan rasio 1:1 (*one to one*)

  Satu *entity* di *entity* 1 berhubungan dengan satu *entity* di *entity* 2 dan satu *entity* di *entity* 2 berhubungan dengan satu *entity* di *entity* 1.



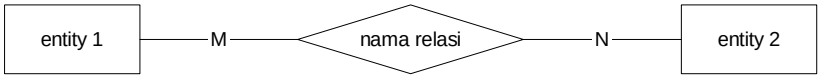
- Relasi dengan rasio 1:N (*one to many*)

  Satu *entity* di *entity* 1 dihubungkan dengan beberapa *entity* di *entity* 2, tetapi *entity* di *entity* 2 hanya dapat dihubungkan dengan satu *entity* di *entity* 1, dan juga sebaliknya.



- Relasi dengan rasio M:N (*many to many*)

  Sejumlah *entity* di *entity* 1 dapat dihubungkan dengan sejumlah *entity* di *entity* 2, dan juga sebaliknya.



2.1.3. Mapping

ER-Diagram akan diproses untuk dapat menghasilkan struktur tabel yang akan digunakan oleh program, proses ini disebut mapping. *Mapping* dari ER-Diagram dilakukan dengan cara sebagai berikut:

1. Untuk setiap *Regular Entity* pada ER-Diagram (ERD), buat tabel dan terdiri dari *simple attribute* sebagai *field*-nya. Pilih *primary key* dan bila *composite* attribut dijadikan *primary*, maka *simple-simple attribute*-nya dapat digabung jadi satu untuk membuat sebuah *primary key*.
2. Untuk setiap *Weak Entity* pada ERD, buat sebuah tabel tertentu dengan atribut pada *weak entity* sebagai *field*-nya. Pilih *primary key* yaitu gabungan antara *key* pada *weak entity* (bila ada) dan *primary key* pada tabel induk sebagai *foreign key*.
3. Untuk *binary* 1:1 relationship, tambahkan *foreign key* pada salah satu tabelnya yang mempunyai partisipasi total pada relasi tersebut (bila ada) dan atribut yang dihasilkan oleh sebuah relasi.
4. Untuk *regular entity binary* 1:N relationship, tambahkan primary key tabel tertentu ke tabel dengan N-side sebagai foreign key.
5. Untuk binary M:N relationship, buat tabel baru dengan primary key-nya adalah gabungan dari dua buah primary keydari tabel pembentuk relasinya. Selain itu field juga dapat ditambahkan untuk tabel baru ini yaitu diambil atribut dari relasinya bila ada.
6. Untuk entity yang punya multivalued atribut, buat tabel baru primary key-nya terdiri dari primary key dari tabel awal dan nilai masing-masing atributnya.

Setelah ER Diagram selesai dibuat maka dapat dilakukan proses *mapping* diatas untuk menghasilkan struktur tabel yang akan digunakan oleh program.

2.1.4. Normalisasi

Normalisasi adalah sebuah proses dimana *schema relations* yang tidak memuaskan berusaha untuk diubah dengan cara membagi *attribute-attribute* yang dimiliki ke dalam *schema relation* yang lebih kecil yang mempunyai *attribute* yang memuaskan (Elmasri dan Navathe 2000). Tujuan dilakukannya normalisasi adalah untuk memastikan bahwa *update anomalies* tidak terjadi perancangan basis data.

Selama proses normalisasi ini biasanya jumlah *field* dalam tabel akan berkurang, dan jumlah tabel dalam aplikasi akan bertambah. Normalisasi diperlukan agar ER-D yang telah dibuat dapat memenuhi semua kebutuhan informasi. Normalisasi ini dilakukan setelah ER-D yang dibuat diperiksa apakah ER-D tersebut sudah merupakan desain *relational database* yang baik atau bukan, sebab ER-D yang baik tidak boleh ada pengulangan informasi, mampu untuk menyediakan informasi tertentu, tidak ada informasi yang hilang dan sebaiknya menghindari nilai *null*. Ada empat tahap yang harus dilakukan untuk menghasilkan ER-D yang baik, yaitu:

2.1.4.1. First Normal Form (1NF)

Tahap pertama dalam normalisasi adalah *First Normal Form* (1NF). Suatu *relation* disebut 1NF jika dan hanya jika semua *value attribute*-nya mempunyai nilai tunggal (*atomic value*), 1NF tidak mengijinkan adanya suatu attribute yang mempunyai "sekelompok nilai" maupun *tuple* yang mempunyai banyak nilai. Dengan kata lain tidak boleh dalam kondisi *multivalued attribute, composite attribute,* maupun kombinasi dari keduanya.

2.1.4.2. Second Normal Form (2NF)

Second Normal Form (2NF) terpenuhi jika pada suatu skema relasi R tidak terdapat *partial functional dependency*, dimana jika terjadi penghilangan sembarang atribut yang merupakan *key* tidak mengakibatkan hilangnya ketergantungan (*dependency*). Hal ini berarti, suatu relasi R dinyatakan 2NF jika

setiap *non-key attribute-nya* (jika ada) FFD pada relational key dan terdapat saling ketergantungan (*dependency*) antar *non-key attribute* tersebut.

2.1.4.3. Third Normal Form (3NF)

Suatu *schema* relasi R dinyatakan dalam 3NF jika suatu *schema* relasi berada dalam 2NF dan tidak ada *non-key attribute* yang ada pada R dalam bentuk *transitive dependency*, yaitu jika dan hanya jika ada sekumpulan atribut Z yang bukan *key* bergantung secara fungsional pada atribut X yang juga bukan merupakan bagian dari *key* manapun dari R. Dengan kata lain, 3NF tidak terjadi jika ada *non-key attribute* menerangkan *non-key attribute* yang lain.

2.1.4.4. Boyce-Codd Normal Form (BCNF)

Setiap BCNF pasti berada dalam bentuk 3NF, BCNF dikatakan transitive dependency jika ada sekelompok attribute Z yang tidak merupakan subset dari sembarang *key*. Dalam BCNF tidak boleh ada *non-key attribute* yang menerangkan *key attribute* yang lain.

2.2. DATA FLOW DIAGRAM (DFD)

Diagram aliran data (*data flow diagram*) adalah teknik yang digunakan untuk menjelaskan berbagai informasi/ transformasi data yang bergerak dari pemasukan data hingga keluar data. DFD merupakan salah satu alat bantu yang digunakan dalam perancangan sistem.

Beberapa simbol yang digunakan dalam *Data Flow Diagram* (Jogiyanto 1990):

2.2.1. Notasi Data Flow Diagram

Sebelum menggambarkan DFD (*Data Flow Diagram*), sistem yang ada harus dipelajari terlebih dahulu termasuk aktivitas dan proses yang terjadi pada sistem tersebut. Dalam menganalisis data flow semua kejadian dievaluasi ke dalam bentuk data flow proses, tempat penyimpanan data (*data store*) serta asal

dan tujuan data berupa arus data. DFD digambarkan secara bertingkat, mulai dari yang paling global (*Context Diagram*) sampai ke tingkat yang lebih rinci.

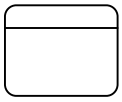
Level tertinggi dari DFD disebut dengan *context diagram,* yang biasa juga disebut dengan DFD level 0. *Context diagram*, yang menunjukkan lingkup dari suatu aktifitas yang akan dimodelkan, hanya mempunyai satu simbol proses. Context diagram ini menggambarkan lingkup, batasan, dan lingkungan dari suatu aktifitas (Kroenke dan Hatch 1994).

Untuk menunjukkan proses apa saja yang terjadi didalam suatu DFD digambarkan pada level berikutnya dengan cara merinci kembali proses yang ada dari proses pusat menjadi DFD level 1. Demikian juga level 2 dan seterusnya, hingga pada akhirnya proses-proses yang ada tidak bisa lagi dibagi menjadi proses yang lebih kecil atau lebih spesifik.

DFD terdiri dari empat simbol, dimana simbol-simbol tersebut digunakan untuk menggambarkan DFD tersebut (McLeod 2001), yaitu:

1.  *External Entity*

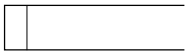
   Merupakan *entity* di luar atau di dalam sistem yang dapat merupakan organisasi atau sistem lainnya yang berada di lingkungan luarnya yang akan memberikan input atau menerima output sistem. Suatu external sistem dapat disimbolkan dengan notasi kotak.

2.  *Processes*

   Proses melakukan suatu perubahan berdasarkan data yang dimasukan dan menghasilkan data atau beberapa keterangan dari perubahan tersebut. Suatu proses dapat ditunjukan dengan simbol empat persegi dengan sudut-sudut yang tumpul.

3. *Data Flow*

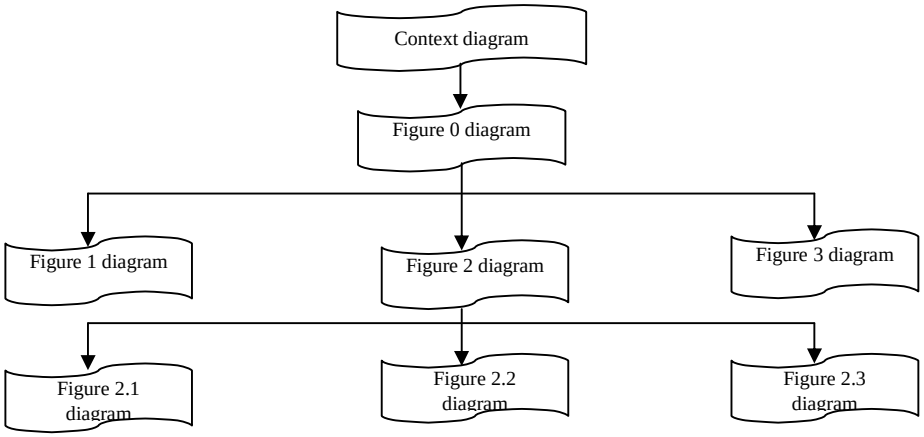
   Data Flow dalam diagram diberi simbol anak panah. Data Flow ini mengalir di antara proses, simpanan data (*data store*) dan *external entity*. Data Flow sebaiknya diberi nama yang jelas dan mengandung arti. Nama pada data flow di tulis di samping atas.

4.  *Data Stores*

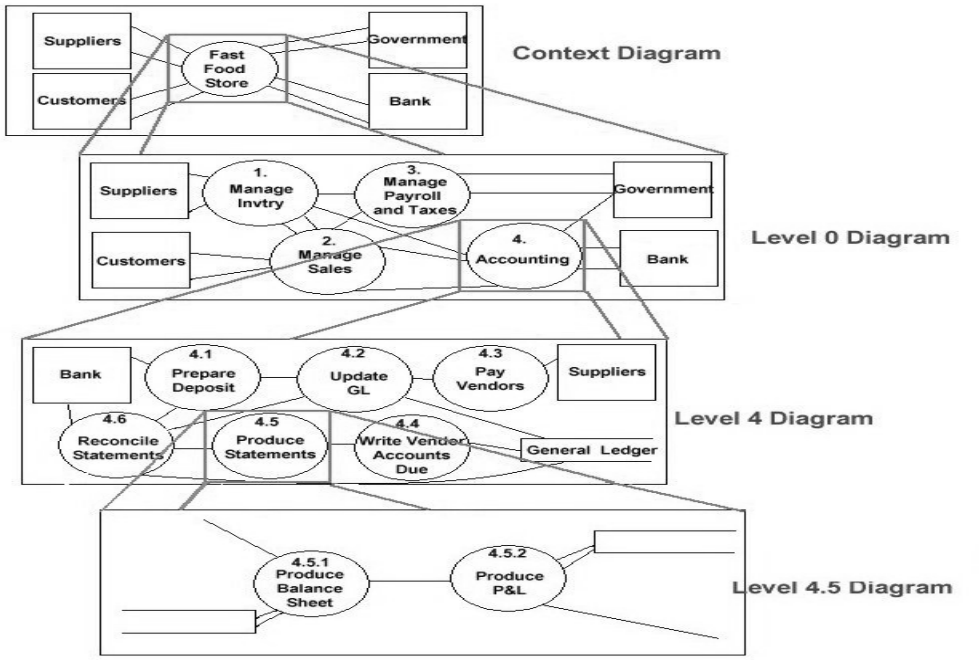
Data Stores adalah tempat untuk menyimpan data baik secara manual maupun otomatis. Penulisan nama pada *data stores* ditulis di dalam simbol atau *data stores. Data stores* tidak mengindikasikan suatu media penyimpan. Tempat penyimpanannya dapat berupa *filling cabinet*, lemari, *file* komputer atau bahkan data tersebut terletak pada pikiran seseorang. Yang penting dari *data stores* adalah pendokumntasian data.

2.2.2. Perancangan Data Flow Diagram

Langkah pertama dalam perancangan DFD adalah membuat context model. Context model merepresentasikan sistem dengan sebuah proses tunggal serta interaksinya dengan beberapa entitas. Context diagram sering diberi nama sama dengan nama sistem dan tidak diawali dengan kata kerja seperti proses-proses yang lain. Setelah context model terbentuk, proses dikembangkan ke level selanjutnya yang menggambarkan proses utama dalam sistem. Tergantung pada kompleksitas sistem, setiap proses dapat pula dikembangkan menjadi model proses sendiri. Langkah ini terus berlanjut sampai tujuan setiap proses menyelesaikan fungsi tunggal tercapai. Berdasar kedekatannya dengan context model, proses dapat di beri nama level 0 DFD, level 1 DFD dan seterusnya [1].



Gambar 2.1. Hierarki Perancangan DFD



Gambar 2.2. Contoh Sebuah Perancangan DFD

2.3. SISTEM INFORMASI MANAJEMEN (SIM)

Sistem Informasi Manajemen (SIM) merupakan penerapan sistem didalam organisasi untuk mendukung informasi yang dibutuhkan oleh semua tingkatan manajemen (Mcleod 1996). Sistem informasi manajemen juga melakukan fungsinya untuk menyediakan semua informasi manajemen untuk mengetahui semua operasi organisasi sistem informasi.

Sistem informasi manajemen berbasis komputer adalah suatu sistem yang terintegrasi antara manusia dan mesin yang memanfaatkan teknologi komputer dalam pengelolaan dan penyediaan informasi guna mendukung operasional manajemen maupun pengambilan keputusan dalam suatu organisasi (Mcleod 1996).

2.3.1. Karakteristik dan Keuntungan dari Sistem Informasi Manajemen

Kualitas informasi yang diperoleh memiliki karakteristik sebagai berikut :

- Fleksibel.

- Menghasilkan informasi yang akurat dan lengkap.
- Menghasilkan informasi yang *uptodate.*
- Menghasilkan informasi yang relevan.
- Menghasilkan informasi yang valid.

User interface yang akan dibuat memiliki karakteristik sebagai berikut :

- *User Interface* yang *user friendly.*
- Menyediakan sekuriti dan batasan akses pada informasi.
- Membutuhkan waktu yang cepat dalam mendapatkan informasi.
- Meminimalkan penggunaan *keyboard,* mengoptimalkan penggunaan *mouse.*
- Terdapat menu *help.*

Keuntungan yang diperoleh sebagai berikut :

- Memungkinkan user untuk lebih produktif.
- Meningkatkan kualitas dalam pengambilan keputusan.
- Menghemat waktu.
- Memberikan kontrol yang lebih baik dalam perusahaan.
- Memberikan solusi pada masalah.
- Kemampuan untuk mengetahui penyebab permasalahan.

2.3.2. Keuntungan Pengembangan Sistem

Dengan telah dikembangkannya sistem yang baru, maka diharapkan akan terjadi peningkatan ini berhubungan dengan :

1. *Performance* (kinerja), peningkatan terhadap kinerja (hasil kerja) sistem yang baru sehingga menjadi lebih efektif.
2. *Information* (informasi), peningkatan terhadap informasi yang disajikan.
3. *Decision making* (pengambilan keputusan), keputusan yang diambil merupakan keputusan yang bertanggung jawab sesuai dengan sistem dan prosedur yang berlaku pada perusahaan.
4. *Control* (pengendalian), peningkatan terhadap pengendalian untuk mendeteksi dan memperbaiki kesalahan yang dan akan terjadi.

5. *Efficiency* (efisiensi), peningkatan terhadap efisiensi operasi.
6. *Sevices* (pelayanan), peningkatan terhadap pelayanan yang diberikan oleh sistem.

Pada organisasi yang telah ada SIM biasa ditemukan dalam bentuk *sistem informasi fungsional* seperti: Sistem Informasi Personalia, Sistem Informasi Persediaan, Sistem Informasi Manufaktur, Sistem Informasi Keuangan, Sistem Informasi Pemasaran, dan berbagai sistem informasi lainnya sesuai dengan kebutuhan tiap – tiap unit kerja dalam lingkungan organisasi atau perusahaan. Namun yang perlu ditekankan adalah sistem – sistem fungsional ini jangan sampai menjadi sistem yang ter-isolasi, berdiri sendiri, tanpa ada koneksi dengan sistem lainnya, karena sistem – sistem tersebut harus ber-sinergi dalam penyediaan informasi untuk kebutuhan manajemen organisasi atau perusahaan.

2.4. DESAIN USER INTERFACE

User interface merupakan salah satu bagian yang penting dalam sebuah perangkat lunak. Baik atau buruknya sebuah tampilan dapat mempengaruhi kinerja seorang pengguna yang menggunakan perangkat lunak tersebut. Untuk merancang *user interface*, diperlukan teknik – teknik tertentu agar dapat menghasilkan tampilan yang baik.

2.4.1. Teknik merancang user interface

Terdapat beberapa teknik yang dapat digunakan untuk merancang user interface (Ambler 2000), yaitu :

1. Konsistensi dalam penempatan komponen – komponen, konsistensi dalam penggunaan warna dan konsistensi dalam segala hal.
2. Dalam merancang tampilan, dapat mengadopsi standar yang digunakan oleh aplikasi – aplikasi terkemuka sejenis.
3. Mendukung pemakaian oleh pengguna tingkat pemula maupun pengguna tingkat menengah.
4. Menggunakan pewarnaan sesuai kebutuhan. Warna tertentu dapat dikombinasikan dengan simbol tertentu agar pengguna lebih mudah mengerti.

5. Menggunakan kekontrasan warna yang sesuai, misalnya menampilkan tulisan berwarna gelap pada latar belakang yang berwarna terang atau menampilkan tulisan berwarna terang pada latar belakang yang gelap.
6. Menggunakan *font* yang sesuai dengan kebutuhan dan mudah untuk dibaca.
7. Suatu obyek yang sedang tidak aktif lebih baik disamarkan warnanya daripada dihilangkan. Dengan demikian pengguna tahu fungsi-fungsi mana saja yang dapat digunakan dan mana yang tidak dapat digunakan pada saat itu.
8. Menghindari terjadinya suatu hal yang dapt berakibat fatal apabila pengguna menekan *default button* (tombol yang sering ditekan, misalnya tombol enter).
9. Mengelompokkan obyek-obyek yang ada pada layar secara efektif. Perlu adanya jarak yang sesuai antara kelompok-kelompok obyek dan pembatas di sekeliling kelompok obyek tersebut.
10. Apabila pengguna membuka sebuah *window* baru, maka *window* tersebut hendaknya diletakkan di tengah-tengah layar.

2.4.2. Merancang Dialog

Dialog adalah sebuah proses komunikasi antara dua pihak atau lebih (Preece 1994). Di dalam perangkat lunak, terdapat salah satu jenis dialog yang sering digunakan, yaitu *menu*. Contoh bentuk *menu* yang biasa digunakan adalah *pull down menu* dan *popup menu*.

Pada saat menggunakan menu, pengguna pertama-tama akan membaca sederet pilihan yang dapat dipilih sesuai kebutuhan. Setelah memastikan pilihan dan program melakukan suatu tindakan, pengguna akan mengamati efek ynag akan terjadi.

Penggunaan menu dalam sebuah aplikasi dapat memebrikan beberapa keuntungan, yaitu :

1. Pengguna tidak memerlukan waktu yang lama untuk mempelajarinya.
2. Mengurangi banyaknya penekanan tombol yang harus dilakukan pengguna.
3. Sesuai untuk pengguna tingkat pemula maupun tingkat menengah.
4. Tidak perlu untuk mengingat banyak hal yang berkaitan dengan menu itu sendiri.

Selain memiliki keuntungan, penggunaan menu dalam sebuah aplikasi juga dapat membuat beberapa kerugian, yaitu :

1. Apabila sebuah menu memiliki pilihan yang terlalu banyak, maka dapat membuat pengguna jstru menjadi sulit melihat dan memahaminya.
2. Menu yang terlalu banyak dapat menghabiskan banyak ruang pada layar.
3. Apabila digunakan oleh pengguna yang mahir, penggunaan menu justru dapat memperlambat.

2.5. TEORI JUAL BELI DAN SIMPAN PINJAM

2.5.1. Jual Beli

Menurut kamus besar bahasa Indonesia, jual adalah memberikan sesuatu dengan memperoleh uang pembayaran atau menerima uang. Sedangkan beli adalah memperoleh sesuatu dengan menukar (membayar dengan uang), memperoleh sesuatu dengan pengorbanan (usaha) yang berat.

Dan di koperasi sekolah di SMA Muhammadiyah 1 Gresik, terjadi transaksi pada jual beli pada umumnya, dimana pembeli memilih barang yang akan dibeli, kemudian diserahkan kepada kasir untuk dihitung jumlah item dan masing-masing harganya, kemudian pembeli membayar dan kasir menghitung uang, kemudian memberikan barang yang dibeli beserta uang kembalian jika ada.

2.5.2. Simpan Pinjam

Simpan Pinjam merupakan suatu transaksi yang memungut dana dalam bentuk pinjaman dan menyalurkan kembali dalam bentuk pinjaman kepada anggota yang membutuhkan, hal ini dilakukan dalam rangka mengurangi gerakan rentenir yang merugikan masyarakat (SP Hasibuan 1996).

Jadi Simpan Pinjam merupakan suatu usaha yang memberikan kesempatan kepada anggota untuk menyimpan dan meminjam uang. Simpan Pinjam merupakan suatu usaha yang melakukan pembentukan modal melalui tabungan para anggota secara teratur dan terus menerus kemudian dipinjamkan kembali kepada para anggota dengan cara yang mudah, murah, cepat, tepat untuk tujuan produktif dan kesejahteraan (Ninik Widiyanti 2003).

Dan di koperasi sekolah di SMA Muhammadiyah 1 Gresik, transaksi simpan pinjam dengan ketentuan sebagai berikut:

1. Pinjam
    a. Pinjaman maksimal Rp 15.000.000 (lima belas juta rupiah).
    b. Input besaran gaji, sehingga besaran cicilan per bulan tidak lebih besar dari 35% gaji.
    c. Lama tenor pinjaman:
        - 12 bulan, bunga 16%
        - 24 bulan, bunga 14%
        - 36 bulan, bunga 11%
    d. Petugas mengecek terlebih dahulu, apakah calon peminjam masih memiliki pinjaman yang belum lunas.
    e. Kepala sekolah yang menyetujui pinjaman.
2. Simpan
    a. Seperti menabung di bank konvensional dengan memperoleh kartu tabungan, dimana terdapat baris debet dan kredit.
    b. Nomor buku tabungan diisi dengan NBM (no. baku Muhammadiyah).
    c. Bisa diambil sewaktu-waktu, sesuai ketersediaan dana di koperasi.