BAB 2
TINJAUAN PUSTAKA

2.1 Profil Tempat Kerja Praktek

2.1.1 Sejarah Perusahaan

Perkembangan dunia IT yang begitu cepat, memberikan andil yang tidak kecil terhadap segala sudut kehidupan. Era internet pun datang menambah seluruh ruang aktivitas. Laju perkembangan IT berimbas pada semakin meningkatnya kebutuhan akan penerapan IT dalam dunia bisnis. Persaingan bisnis yang semakin keras, menuntut pelaku-pelakunya untuk bisa berinovasi menerapkan *tools* pendukung untuk selalu bisa bertahan atau minimal tidak tertinggal terlalu jauh. Salah satu *tools* pendukung itu adalah pemberdayaan teknologi informasi.

PT. Xyclus Cipta Teknologi hadir untuk memenuhi kebutuhan tersebut, PT. Xyclus Cipta Teknologi datang membawa segudang solusi dalam bidang IT. PT. Xyclus Cipta Teknologi adalah perusahaan jasa atau produksi dalam bidang konsultan teknologi informasi, perancangan sistem informasi, jasa integrasi data, pembangunan perangkat lunak, aplikasi berbasis internet dan multimedia.

PT. Xyclus Cipta Teknologi sendiri didirikan pada tahun 2008 di Jalan Bojong Kacor No.1 Cikutra Barat, Bandung. Sampai sekarang, PT. Xyclus Cipta Teknologi telah membangun dan mengembangkan aplikasi perangkat lunak di berbagai bidang seperti Resto, mainan, garmen dan lainnya. Aplikasi yang sudah dikembangkan saat ini sudah berbasis *cloud computing.*

2.1.2 Logo Perusahaan

Berikut merupakan logo dari PT. Xyclus Cipta Teknologi



Gambar 2.1 *Logo PT. Xcylus Cipta Teknologi*

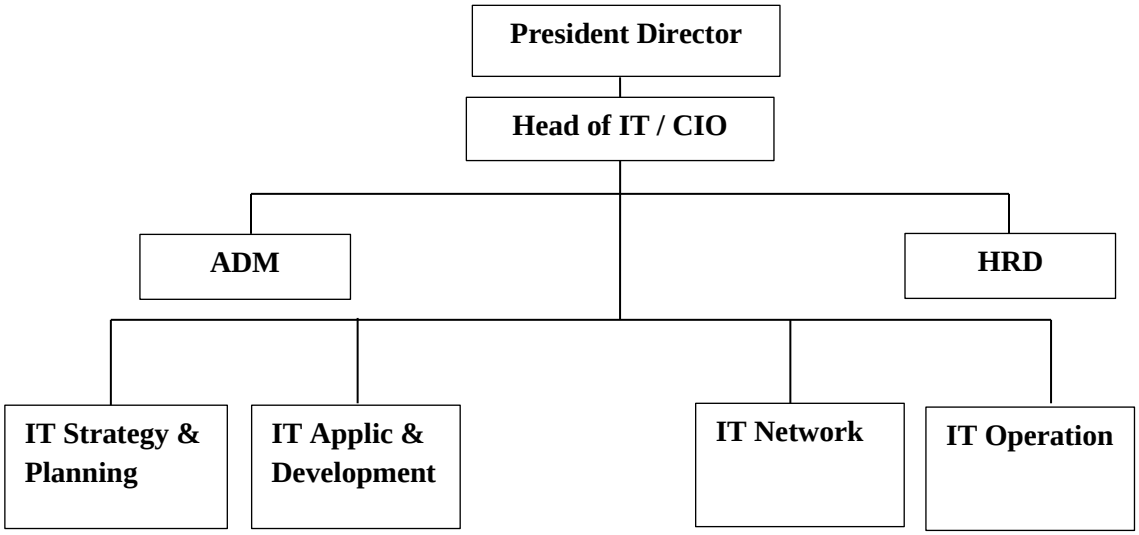
2.1.3 Visi dan Misi Perusahaan

Visi dan misi bagi sebuah perusahaan bisa dikatakan sebagai pedoman dan tujuan. Tanpa adanya visi dan misi sebuah perusahaan tidak akan bertahan dan tidak tahu apa yang harus dilakukan oleh perusahaan tersebut.

Visi : menjadi perusahaan konsultan bisnis dan teknologi informasi yang berorientasi pada pemenuhan kebutuhan pelanggan akan solusi bisnis yang kompetitif dalam menghadapi persaingan global.

Misi : membantu pelanggan dan mitra dalam menciptakan nilai atau daya saing melalui pemanfaatan Teknologi Informasi sehingga memberikan nilai tambah dalam operasional perusahaan maupun pemasaran produk.

2.1.4 Struktur Organisasi Perusahaan



Gambar 2.2 *Struktur Organisasi Perusahaan PT. Xyclus Cipta Teknologi*

2.1.5 Tugas dan Wewenang

A. President Director

Presiden direktur adalah pimpinan tertinggi dalam suatu perusahaan. Pimpinan tertinggi ini memiliki tanggung jawab dalam memimpin dan mengarahkan perusahaan, yaitu dalam hal:

1) Menyusun strategi dan visi.
2) Menjalin hubungan dan kemitraan strategis.
3) Mengatur investasi, alokasi dan divestasi.
4) Memimpin direksi.
5) Memastikan bahwa prinsip tata kelola perusahaan benar-benar diterapkan dengan baik.
6) Membuat rencana pengembangan perusahaan dan usaha perusahaan dalam jangka pendek dan jangka panjang.
7) Bertanggung jawab penuh dalam melaksanakan tugasnya untuk kepenting an perseroan dalam mencapai maksud dan tujuannya.
8) Menjalin hubungan kerjasama dengan berbagai perusahaan.
9) Tugas dan tanggung jawab President :

    1. Bertanggung jawab terhadap kemajuan perusahaan dan mengkoordinasikan selutuh kegiatan perusahaan.
    2. Memimpin perusahaan dan mengawasi kelancaran perusahaan sesuai dengan tujuan dan kebijakan yang telah ditetapkan.
    3. Melakukan hubungan dengan pihak luar baik swasta maupun pemerintah yang bertujuan untuk kelancaran perusahaan.
    4. Menetapkan kebijakan-kebijakan perusahaan.
    5. Menerima dan memeriksa laporan dari masing- masing *Vice President.*

B. HRD (*Human Resource Development*)

1) Menyusun strategi dan kebijakan pengelolaan SDM di perusahaan berdasarkan strategi jangka panjang dan jangka pendek yang telah ditetapkan sesuai dengan peraturan pemerintah yang berlaku agar diperoleh SDM dengan kinerja, kapabilitas dan kompetensi yang sesuai dengan yang diinginkan perusahaan.
2) Menyusun rencana kerja dan anggaran bagiannya sesuai dengan strategi, kebijakan dan sistem SDM yang telah ditetapkan untuk memastikan tercapainya sasaran bagian SDM.
3) Mengkoordinasikan dan mengontrol pelaksanaan fungsi SDM di seluruh perusahaan untuk memastikan semuanya sesuai dengan strategi, kebijakan, sistem dan rencana kerja yang telah disusun.
4) Mengkoordinasikan dan mengontrol anggaran bagian SDM agar digunakan dengan efektif dan efisien sesuai dengan rencana kerja.
5) Mengarahkan, menganalisa dan mengelola praktek dan prosedur remunerasi untuk memastikan paket remunerasi yang ditetapkan perusahaan kompetitif, sejalan dengan praktek industri, sesuai kemampuan finansial perusahaan dan adil secara internal.
6) Mengelola dan mengontrol aktifitas administrasi kantor, kepersonaliaan, dan sistem informasi SDM untuk memastikan tersedianya dukungan yang optimal bagi kelancaran operasional perusahaan.

C. ADM (*Administration*)

1) Melaksanakan aktifitas penyiapan ruang kerja dan peralatan kantor untuk seluruh pegawai, untuk memastikan ketersediaan ruangan kerja dan peralatan kantor bagi setiap pekerja sesuai dengan jenis pekerjaan dan jabatan.
2) Melaksanakan kegiatan surat-menyurat, dokumentasi dan pengarsipan, untuk memastikan dukungan administrasi bagi kelancaran kegiatan seluruh karyawan.
3) Membuat rencana dan mengevaluasi kerja harian dan bulanan untuk memastikan tercapainya kualitas target kerja yang dipersyaratkan dan sebagai bahan informasi kepada atasan.
4) Membuat perkiraan biaya tahunan yang berkaitan dengan kegiatan *office administration*, sebagai rekomendasi pembuatan anggaran *Department General Affair.*

D. IT Strategy & Planning

1) Membuat rencana kerja.
2) Menugaskan pekerjaan keanggotaan tim.
3) Memonitor jalannya proyek dan kerja tim.
4) Bertanggung jawab atas keberhasilan eksekusi.
5) Memastikan proyek dapat berjalan dengan sukses sesuai dengan jadwal dan ruang lingkup yang disetujui.
6) Mengadakan *meeting* proyek secara berkala untuk menangani permintaan perubahan.
7) Membuat progress *report*.
8) Menjembatani penganalisa sistem dengan desainer sehingga antara hasil analisa sistem dan bentuk desain aplikasi sesuai dengan yang diharapkan.
9) Menjembatani desainer dengan programmer sehingga desain baru yang dibuat tidak menyulitkan tim programmer dalam membuat program.

E. IT Applic & Development

IT Applic & Development meliputi *System Analyst*, *Programmer*, *System Desain*, Tester dan Dokumentasi

1) *Sistem Analyst* / Penganalisis Sistem:

a. Melakukan analisis sistem yang akan dibangun dengan cara wawancara dengan pihak yang ditunjuk.
b. Membuat *Statement of Purpose*.
c. Membuat *Context Diagram*.
d. Membuat *DFD Level.*
e. Membuat *Process Specification*.
f. Membuat ERD.
g. Membuat Kamus Data.
h. Melaporkan hasil pekerjaan kepada manajer proyek.
i. Berdiskusi dengan manajer proyek mengenai langkah-langkah yang harus ditempuh untuk menyelesaikan proyek yang sesuai dengan perencanaan manajemen proyek (*Project management plan*).
j. Berdiskusi dengan desainer aplikasi untuk menentukan desain aplikasi sesuai dengan yang diinginkan.
k. Membuat dokumentasi analisa dan desain sistem dibantu oleh dokumentator.

2) *Desainer* :

a. Membuat desain aplikasi (bentuk menu aplikasi, bentuk *form*, dan bentuk *report*).
b. Berdiskusi dengan penganalisis sistem tentang desain aplikasi agar sesuai dengan yang diharapkan.
c. Berdiskusi dengan *programmer* tentang maksud bentuk desain aplikasi yang dibuat.
d. Membuat dokumentasi desain aplikasi dibantu oleh dokumentator.

3) *Programmer* :

a. Membuat program aplikasi sesuai dengan hasil analisa dan desain yang ditentukan.
b. Berdiskusi dengan desain aplikasi dan analisis agar program yang dibuatnya sesuai dengan yang telah ditentukan.
c. Memberi usulan pada *desainer* dan penganalisa sistem (*system analyst)* bila memang diperlukan.
d. Mengajukan pertanyaan kepada *desainer* dan *system analyst* apabila memang ada hal – hal yang kurang jelas.
e. Memperbaiki program berdasarkan hasil investigasi.
f. Membuat dokumentasi program dibantu oleh dokumentator.

4) *Tester* :

a. Melakukan *test* terhadap program aplikasi yang telah dibuat.
b. Membuat *list* tentang hal-hal yang masih perlu diperbaiki.
c. Memberi rekomendasi maupun usulan tentang solusi permasalahan yang dihadapi.
d. Membuat dokumentasi tentang hasil *testing* terhadap program aplikasi dibantu oleh dokumentator.

5) *Dokumentator* :

a. Membuat Dokumentasi Sistem (Desain Spesifikasi Sistem), berkoordinasi dengan *system analyst*, *designer*, dan *programmer*.
b. Mendokumentasikan manajemen proyek yang telah dibuat oleh manajer proyek.

F. IT Network

1. *Computer Systems Engineers* :

a. Berkomunikasi dengan *staff* atau *klien* untuk memahami persyaratan sistem tertentu.
b. Memberikan saran pada biaya proyek, konsep desain, atau perubahan desain.
c. Dokumen desain spesifikasi, petunjuk instalasi, dan sistem informasi terkait lainnya.
d. Verifikasi stabilitas, interoperabilitas, portabilitas, keamanan, atau skalabilitas arsitektur sistem.
e. Berkolaborasi dengan *engineer* atau pengembang perangkat lunak untuk memilih solusi desain yang tepat atau memastikan kompatibilitas komponen sistem.
f. Mengevaluasi teknologi yang muncul saat ini untuk mempertimbangkan faktor-faktor seperti biaya, portabilitas, kompatibilitas, atau kegunaan.
g. Memberikan bimbingan teknis atau dukungan untuk pembangunan atau tips sistem.
h. Mengidentifikasi sistem data, perangkat keras, atau komponen perangkat lunak yang diperlukan untuk memenuhi kebutuhan pengguna.
i. Memberikan pedoman untuk menerapkan sistem yang aman untuk pelanggan atau tim instalasi.
j. Memonitor operasi sistem untuk mendeteksi masalah potensial.

2. *Network and Computer Systems Administrators* :

a. Menjaga dan mengelola jaringan komputer dan lingkungan komputasi terkait termasuk perangkat keras komputer, perangkat lunak sistem, perangkat lunak aplikasi, dan semua konfigurasi.
b. Melakukan *backup* data dan operasi pemulihan kerusakan.
c. Mendiagnosa, memecahkan masalah, dan menyelesaikan perangkat keras, perangkat lunak, atau jaringan lainnya dan masalah sistem, dan mengganti komponen yang rusak bila diperlukan.

d. Merencanakan, mengkoordinasikan, melaksanakan langkah-langkah keamanan jaringan untuk melindungi data, perangkat lunak, dan perangkat keras.
e. Mengkonfigurasikan, memonitor, dan memelihara aplikasi email atau virus *software* perlindungan.
f. Mengoperasikan master konsol untuk memonitor kinerja sistem komputer jaringan, untuk mengkoordinasikan komputer akses jaringan dan penggunaan.
g. Memuat rekaman komputer *disk*, menginstal perangkat lunak dan kertas *printer* atau *form*.
h. Desain, mengkonfigurasi, perangkat keras uji komputer, jaringan lunak dan perangkat lunak sistem operasi.
i. Memonitor kinerja jaringan untuk menentukan apakah penyesuaian perlu dibuat, dan untuk menentukan di mana perubahan harus dibuat di masa depan.
j. Berunding dengan pengguna jaringan tentang bagaimana untuk memecahkan masalah sistem yang ada.

3. Computer Security Specialists

a. Mengenkripsi transmisi data dan membangun *firewall* untuk menyembunyikan informasi rahasia seperti sedang dikirim dan untuk menahan *transfer digital* tercemar.
b. Mengembangkan rencana untuk melindungi *file* komputer terhadap modifikasi disengaja atau tidak sah, perusakan, atau pengungkapan dan untuk memenuhi kebutuhan pengolahan data darurat.
c. Meninjau pelanggaran prosedur keamanan komputer dan mendiskusikan prosedur dengan pelanggar untuk memastikan pelanggaran tidak terulang kembali.
d. Memonitor penggunakan *file* data dan mengatur akses untuk melindungi informasi dalam *file* komputer.
e. Monitor laporan saat ini dari virus komputer untuk menentukan kapan untuk memperbarui sistem perlindungan virus.

f. Memofifikasi keamanan *file* komputer untuk memasukkan *software* baru, memperbaiki kesalahan, atau mengubah status akses individu.
g. Melakukan penilaian risiko dan melaksanakan tes pengolahan data sistem untuk memastikan fungsi pengolahan data kegiatan dan langkah-langkah keamanan.
h. Berunding dengan pengguna untuk membahas isu-isu seperti akses data komputer kebutuhan, pelanggaran keamanan, dan perubahan pemrograman.
i. Melatih pengguna dan meningkatkan kesadaran keamanan untuk memastikan keamanan sistem dan untuk meningkatkan efisiensi *server* dan jaringan.
j. Mengkoordinasikan pelaksanaan rencana sistem komputer dengan personil pendirian dan *vendor* luar.

2.2 Landasan Teori

Landasan teori menjelaskan beberapa teori yang berkaitan dengan permasalahan yang dibahas sebagai dasar pemahaman dalam sebuah sistem serta metode yang dipakai untuk kegiatan pengembangan terhadap sistem tersebut.

2.2.1 Surat

Surat adalah sarana komunikasi untuk menyampaikan informasi tertulis oleh suatu pihak kepada pihak lain. Fungsinya mencakup beberapa hal: sarana pemberitahuan, permintaan, buah pikiran, gagasan, alat bukti tertulis, alat pengingat, dan bukti historis.

Surat secara umum digolongkan menjadi tiga yaitu surat pribadi, surat dinas, dan surat niaga apabila ditinjau dari segi bentuk, isi, dan bahasanya. Sedangkan apabila digolongkan berdasarkan berdasarkan pemakaiannya dapat dibagi menjadi tiga yaitu surat pribadi, surat resmi, dan surat dinas.

1. Surat Pribadi

Surat pribadi adalah surat yang digunakan untuk kepentingan pribadi. Surat dapat berupa korespondensi antara sesama teman atau keluarga. Ciri-ciri surat pribadi yaitu:

a. Tidak menggunakan kop surat.
b. Tidak ada nomor surat.
c. Salam pembuka dan penutup bervariasi.
d. Penggunaan bahasa bebas, sesuai keinginan penulis.
e. Format surat bebas.

2. Surat Resmi

Surat resmi adalah surat yang digunakan untuk kepentingan resmi, baik perseorangan, instansi, maupun organisasi; misalnya undangan, surat edaran, dan surat pemberitahuan. Ciri-ciri surat resmi:

a. Menggunakan kop surat apabila dikeluarkan organisasi.
b. Ada nomor surat, lampiran, dan perihal.
c. Menggunakan salam pembuka dan penutup yang lazim.
d. Penggunaan ragam bahasa resmi.
e. Menyertakan cap atau stempel dari lembaga resmi.
f. Ada aturan format baku.

3. Surat Niaga

Surat niaga digunakan bagi badan yang menyelenggarakan kegiatan usaha niaga seperti industri dan usaha jasa. Surat ini sangat berguna dalam membangun hubungan dengan pihak luar sehingga harus disusun dengan baik. Surat niaga terdiri atas surat jual beli, kwintansi, dan perdagangan; dan dapat dibagi atas surat niaga *internal* dan surat niaga *eksternal*. Salah satu contoh dari surat niaga adalan surat penawaran dan surat penagihan.

4. Surat Dinas

Surat dinas digunakan untuk kepentingan pekerjaan formal seperti instansi dinas dan tugas kantor. Surat ini penting dalam pengelolaan administrasi dalam suatu instansi. Fungsi dari surat dinas yaitu sebagai dokumen bukti tertulis, alat pengingat berkaitan fungsinya dengan arsip, bukti sejarah atas perkembangan instansi, dan pedoman kerja dalam bentuk surat keputusan dan surat instruksi.Ciri-ciri surat dinas:

a. Menggunakan kop surat dan instansi atau lembaga yang bersangkutan.
b. Menggunakan nomor surat, lampiran, dan perihal.
c. Menggunakan salam pembuka dan penutup yang baku.
d. Menggunakan bahasa baku atau ragam resmi.
e. Menggunakan cap atau stempel instansi atau kantor pembuat surat.
f. Format surat tertentu.

5. Surat Lamaran Pekerjaan

Surat lamaran pekerjaan adalah surat yang dibuat dan dikirimkan oleh seseorang yang ingin bekerja di sebuah kantor, perusahaan ataupun instansi tertentu. Surat lamaran pekerjaan termasuk surat dinas atau resmi. Oleh karena itu, terdapat aturan – aturan tertentu yang harus diperhatikan dalam penulisannya. Secara umum surat memiliki bagian-bagian seperti berikut ini:

a. Kepala surat.
b. Tempat dan tanggal pembuatan surat.
c. Nomor surat.
d. Lampiran.
e. Hal atau perihal.
f. Alamat tujuan.
g. Salam pembuka.
h. Isi surat.
i. Tanda tangan dan nama terang.
j. Jabatan.

2.2.2 Aplikasi

Perangkat lunak aplikasi adalah suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Biasanya dibandingkan dengan perangkat lunak sistem yang mengintegrasikan berbagai kemampuan komputer, tapi tidak secara langsung menerapkan kemampuan tersebut untuk mengerjakan suatu tugas yang menguntungkan pengguna. Contoh utama perangkat lunak aplikasi adalah pengolah kata, lembar kerja, dan pemutar media.

Beberapa aplikasi yang digabung bersama menjadi suatu paket kadang disebut sebagai suatu paket atau suite aplikasi (*application suite*). Contohnya adalah *Microsoft Office* dan *Open Office.org,* yang menggabungkan suatu aplikasi pengolah kata, lembar kerja, serta beberapa aplikasi lainnya. Aplikasi-aplikasi dalam suatu paket biasanya memiliki antarmuka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi. Sering kali, mereka memiliki kemampuan untuk saling berinteraksi satu sama lain sehingga menguntungkan pengguna. Contohnya, suatu lembar kerja dapat dibenamkan dalam suatu dokumen pengolah kata walaupun dibuat pada aplikasi lembar kerja yang terpisah.

2.2.3 Basis Data

2.2.3.1 Pengertian Basis Data

Secara umum basis data adalah kumpulan dari berbagai data yang saling berhubungan satu dengan yang lainya.

Adapun keguanaan dari basis data antara lain:

1. Basis data merupakan komponen penting dalam sistem informasi, karena merupakan dasar dalam menyediakan informasi.
2. Menentukan kualitas informasi yang akurat, tepat pada waktunya, dan relevan, dimana informasi dikatakan bernilai jika manfaatnya lebih efektif dibandingkan dengan biaya untuk mendapatkannya.
3. Mengurangi duplikasi data (*data redudancy*).

4. Meningkatkan relasi antar data (*data relatability*).
5. Mengurangi pemborosan *external storage*.

Di dalam basis data terdapat urutan jenjang data:

1. *Character*

   Merupakan bagian data terkecil yang berupa karakter numerik, huruf, maupun karakter numerik, huruf, maupun simbol yang membentuk *item* data/*field*.
2. *Field*

   Merupakan atribut dari *record* yang menunjukan suatu *item* dari data, misal nama.
3. *Record*

   Merupakan kumpulan dari dari *field*, mengganmbarakan suatu unit data individu tertentu. Kumpulan *record* membentuk suatu *file*.
4. *File*

   File terdiri dari *ricord-record* yang menggambarkan suatu kesatuan data yang sama/sejenis.
5. *Database*

   Merupakan kumpulan dari *file/*tabel.

2.2.3.2 Normalisasi Basis Data

Normalisasi merupakan suatu teknik dalam logika desain suatu basis data, dengan mengelompokan atribut dari suatu relasi sehingga membentuk struktur relasi tanpa redudansi.

1. Bentuk tidak normal

   Bentuk ini merupakan kumpulan data yang akan disimpan, tidak ada keharusan mengikuti suatu format tertentu, dapat saja data tidak lengkap atau terduplikasi dan data dikumpulkan apa adanya.

2. Bentuk normal pertama

   Suatu tabel dikatakan dalam bentuk normal pertama (1NF) bila setiap kolom bernilai tunggal untuk setiap baris. Ini berarti bahwa nama kolom yang berulang cukup diwakili oleh sebuah nama kolom (tidak perlu ada indeks dalam memberi nama kolom).

3. Bentuk Normal Kedua

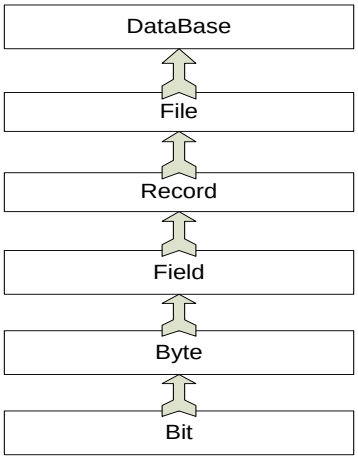
   Suatu tabel berada dalam bentuk normal kedua (2NF) jika tabel berada dalam bentuk normal pertama, semua kolom bukan kunci primer tergantung sepenuhnya terhadap kunci primer. Suatu kolom disebut tergantung sepenuhnya terhadap kunci primer jika nilai pada suatu kolom selalu bernilai sama untuk suatu nilai kunci primer yang sama.

4. Bentuk Normal Ketiga

   Suatu tabel berada dalam bentuk normal ketiga (3NF) jika tabel berada dalam bentuk normal kedua, setiap kolom bukan kunci primer tidak memiliki ketergantungan secara transitif terhadap kunci primer.

2.2.3.3 Hirarki Data

Data harus disusun secara teratur agar pengolahannya dapat dilakukan dengan baik dan efisien. Pengorganisasian data dapat dibagi dalam enam tingkatan, yaitu :



Gambar 2.3 *Hirarki Data [16]*

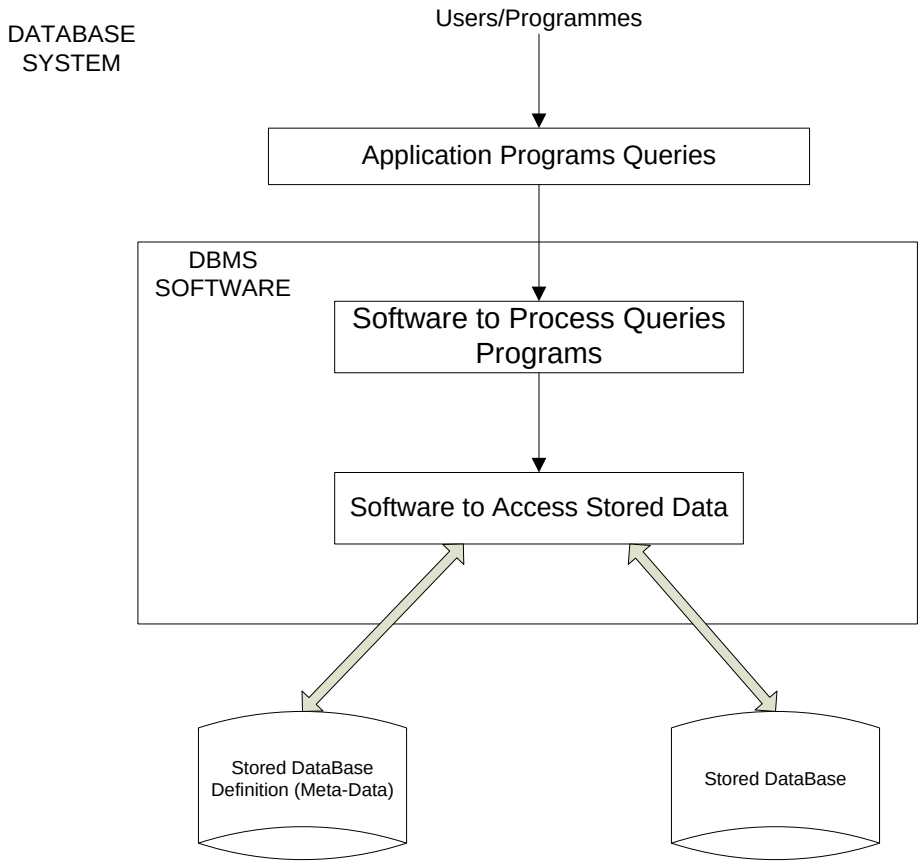
Dari gambar 2.6 dapat dijelaskan sebagai berikut :

a) *Database* yaitu kumpulan dari beberapa *file*/tabel yang saling berhubungan antara *file* yang satu dengan *file* yang lain.
b) *File* yaitu kumpulan dari *record* yang saling berkaitan dan memiliki format *field* yang sama dan sejenis.
c) *Record* yaitu kumpulan dari *field* yang menggambarkan suatu unit data individu tertentu.
d) *Field* yaitu suatu atribut dari *record* yang menunjukan suatu *item* dari data nilai *record* sebuah *field*.
e) *Byte* yaitu atribut dari *field* yang berupa karakter yang membentuk nilai dari sebuah *field*.
f) *Bit* yaitu bagian terkecil dari data secara keseluruhan yaitu huruf karakter Ascil (*American Standard Code From Information Interchange*) nol atau satu yang merupakan komponen pembentuk *byte.*

2.2.3.4 Sistem Basis Data

Menurut Jogiyanto Sistem Basis Data adalah Suatu sistem informasi yang mengintegarasikan kumpulan dari data yang berhubungan satu dengan yang lainnya dan membuatnya tersedia untuk beberapa aplikasi yang bermacam-macam di dalam suatu organisasi[2]. Komponen-komponen utama sebuah sistem basis data :

a) Perangkat Keras (*Hardware*).
b) Sistem Operasi (*Operating Sistem*).
c) Basis Data (*Database*).
d) Sistem Pengolahan Basis Data.
e) Pemakai (*User*).
f) Aplikasi (perangkat lunak) lain (bersifat opsional).



***Gambar 2.4** Konsep Sistem Basis Data* [7]

Keuntungan penggunaan Sitem Basis Data / DBMS :

a) Data *Independence*: Program Aplikasi tidak harus, idealnya, akan terkena rincian representasi data dan penyimpanan, DBMS memberikan pandangan abstrak dari data yang menyembunyikan rincian tersebut.
b) Efisien Akses Data: Sebuah DBMS menggunakan berbagai teknik canggih untuk menyimpan dan mengambil data secara efisien. Fitur ini sangat penting jika data yang disimpan pada perangkat penyimpanan *eksternal*.
c) Integritas Data dan Keamanan: Jika data selalu diakses melalui DBMS, DBMS dapat menegakkan batasan integritas. Misalnya, sebelum memasukkan informasi gaji karyawan, DBMS dapat memeriksa bahwa anggaran departemen tidak terlampaui. Selain itu, dapat menegakkan contmls akses yang mengatur data apa yang dapat dilihat oleh kelas yang berbeda dari pengguna.

d) Data Administrasi: Ketika beberapa pengguna berbagi data, sentralisasi administrasi data dapat menawarkan perbaikan signifikan. Profesional berpengalaman yang memahami sifat dari data yang dikelola, dan bagaimana berbagai kelompok pengguna menggunakannya, dapat bertanggung jawab untuk mengatur representasi data untuk meminimalkan redundansi dan untuk fine-tuning penyimpanan data untuk membuat pengambilan efisien.
e) *Concurrent* Akses dan *Crash Recovery*: Sebuah jadwal DBMS akses bersamaan dengan data sedemikian rupa sehingga pengguna bisa memikirkan data yang sedang diakses oleh hanya satu pengguna pada satu waktu. Selanjutnya, DBMS melindungi pengguna dari efek kegagalan sistem.
f) Aplikasi Waktu Pengembangan *Reduced*: Jelas, DBMS mendukung fungsi penting yang umum untuk banyak aplikasi mengakses data dalam DBMS. Hal ini, dalam hubungannya dengan antarmuka tingkat tinggi untuk data, memfasilitasi pengembangan aplikasi cepat. Aplikasi DBMS juga cenderung lebih kuat daripada sejenis aplikasi yang berdiri sendiri karena tugas-tugas penting yang ditangani oleh DBMS (dan tidak harus *debugged* dan diuji dalam aplikasi).

2.2.3.5 Pemodelan Data

Pemodelan sistem memainkan peranan yang penting dalam pengembangan sistem. Pemodelan data kadang-kadang disebut pemodelan *database* karena model data kadang-kadang diimplementasikan sebagai sebuah *database*. Pemodelan data dapat digambarkan dengan ERD (*Entity Relationship Diagram*).

Menurut Whitten, *Entity Relationship Diagram* (ERD) merupakan pemodelan data yang menggunakan beberapa notasi untuk menggambarkan data yang berhubungan dengan *entity* dan *relationship*[8].

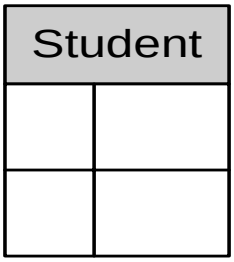
Ada beberapa konsep dasar bagi setiap model data :

a) Entitas *(Entity)*

Entity adalah sebuah class dari orang, tempat, objek, kejadian, atau konsep mengenai apa yang dilakukan untuk mengambil dan menyimpan data. *Entity* dilambangkan dengan sebuah lambang persegi.

Beberapa kategori dari *entity* adalah sebagai berikut:

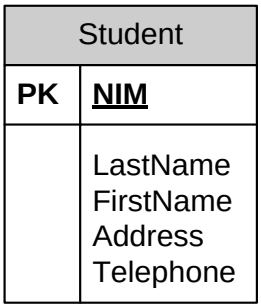
1. Orang : Vendor, pelanggan, karyawan, dan lain-lain.
2. Tempat : Ruangan, kantor cabang, gedung, dan lain-lain.
3. Objek : Produk, mesin, bahan dasar, dan lain-lain.
4. Kejadian: Invoice, pemesanan, penjualan, dan lain-lain.
5. Konsep : Stok, dana, account, kualifikasi, dan lain-lain.



Gambar 2.5 *Contoh Entity dalam ERD*

b) Atribut (*Attribute*)

Attribute adalah sebuah properti dari sebuah *entity* atau relasi (*relationship*). Setiap *attribute* memiliki tipe data yang mendefinisikan nilai dari data tersebut.



Gambar 2.6 *Contoh Atribut dalam ERD*

c) Indentifikasi (*Identification*)

Dengan banyaknya *instance* yang dimiliki oleh suatu *entity* maka diperlukan suatu *key* yang unik untuk mengindetifikasikan setiap *instance* berdasarkan data dari atribut. Yang dimaksud *key* adalah suatu atribut atau sekumpulan atribut yang mengasumsikan nilai yang unik dari setiap bagian dari *entity* dan seringkali disebut *identifier*.

Candidate key adakah suatu *key* yang memiliki kemungkinan untuk dijadikan *primary key. Primary key* adalah *candidate key* yang unik dan mengidentifikasikan sebuah bagian dari *entity. Alternate key* adalah *candidate key* yang tidak dijadikan *primary key*.

d) Relasi (*Relationship*)

Relationship adalah sebuah hubungan antara satu atau lebih *entity. Cardinality* adalah jumlah minimum dan maksimum dari keberadaan suatu *entity* yang mungkin direlasikan dengan *entity* lain. *Degree* adalah sejumlah *entity* yang berpartisipasi dalam sebuah *relationship. Foreign key* adalah sebuah *primary key* yang digunakan oleh *entity* lain untuk mengidentifikasikan instansi dari sebuah *relationship.*

Menurut Whitten di dalam penggambaran ERD, terdapat *cardinality*, yaitu suatu angka minimum atau maksimum dari kejadian suatu *entity* yang mungkin berhubungan dengan kejadian *entity* yang lainnya. Karena semua *relationship* adalah dua arah, *cardinality* harus didefinisikan dalam dua arah untuk setiap *relationship.*

Non-identifying relationship adalah hubungan dimana setiap *entity* partisipan mempunyai *primary key* independen-nya masing-masing. *Relationship* tipe ini digambarkan dengan garis putus-putus dan biasa disebut *strong entity*. Sementara *indentifying relationship adalah* hubungan dimana setiap enity partisipan mempunyai *primary key* yang sama. *Relationship* tipe inin digambarkan dengan garis yang tidak terputus-putus dna biasa disebut *weak entity*.

e) Generalisasi (*Generalization*)

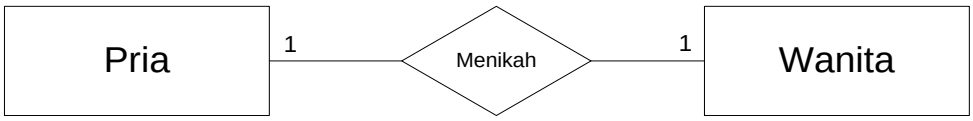
Generalisasi atau *generalization* ialah sebuah konsep dimana atribut-atribut yang umum bagi beberapa tipe *entity* dikelompokkan kedalam *entity* mereka masing-masing.

Beberapa macam hubungan antar relasi, antara lain:

1. Satu ke Satu (*One to One*)

   Bentuk relasi antara satu entitas dengan sejumlah satu ke entitas dengan jumlah yang sama.

   Contoh :



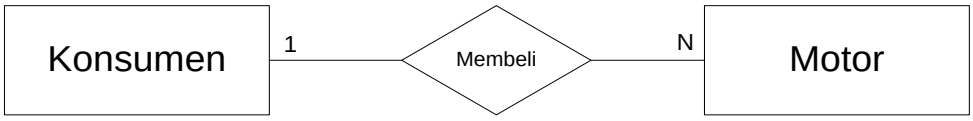
Gambar 2.7 *Contoh relasi satu ke satu*

   Keterangan : Pria dan wanita dengan relasi menikah. Satu pria menikah hanya dengan satu wanita. Satu wanita menikah hanya dengan satu pria. Diasumsikan tidak diperbolehkan adanya poligami.

2. Satu ke Banyak (*One to Many*)

   Bentuk relasi dari entitas dengan jumlah satu ke entitas lain yang berjumlah lebih dari satu (Entitas dengan banyak alternatif tujuan).

   Contoh :



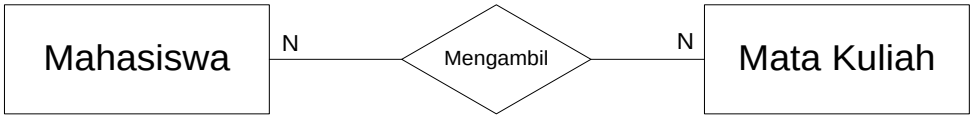
Gambar 2.8 *Contoh relasi satu ke banyak*

   Keterangan : Konsumen dan motor dengan relasi membeli. Satu konsumen dapat membeli banyak motor. Satu motor hanya dapat dibeli oleh satu konsumen.

3. Banyak ke Banyak (*Many to Many*)

   Bentuk relasi yang mendeskripsikan permasalahan yang komplek yaitu hubungan antara entitas yang berjumlah lebih dari satu dengan entitas yang sama.

   Contoh :



Gambar 2.9 *Contoh relasi banyak ke banyak*

   Keterangan : Mahasiswa dan mata kuliah dengan relasi mengambil. Satu mahasiswa dapat mengambil banyak mata kuliah. Satu mata kuliah dapat diambil oleh banyak mahasiswa.

2.2.3.6 Analisis Sistem

Analisis sistem didefinisikan sebagai penguraian dari suatu sistem informasi yang utuh ke dalam bagian-bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan, kesempatan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikannya.

Sedangkan menurut Abdul Kadir analisis sistem merupakan tahapan yang dimulai karena adanya permintaan terhadap sistem baru[3]. Dimana, permintaan dapat datang dari seorang manajer di luar departemen sistem informasi atau dari pihak eksekutif yang melihat adanya masalah atau menemukan adanya peluang baru. Sehingga tujuan utama analisis sistem adalah untuk menetukan hal-hal detail yang akan dikerjakan oleh sistem yang akan diusulkan.

Pada model analisis terdapat perangkat lunak yang dapat digambarkan dalam bentuk sebagai berikut:

1. *Flowmap*

Flowmap adalah penggambaran secara grafik dari langkah-langkah dan urutan prosedur dari suatu program. *Flowmap* berguna untuk membantu analis dan *programmer* untuk memecahkan masalah kedalam segmen yang lebih kecil dan menolong dalam menganalisis alternatif pengoperasian. Biasanya *flowmap* mempermudah penyelesaian suatu masalah khususnya masalah yang perlu dipelajari dan dievaluasi lebih lanjut.

2. Konsep Dasar Berorientasi Obyek

Sebelumnya mari definisikan dulu pengertian obyek. Obyek adalah "benda" secara fisik atau konseptual, yang dapat ditemui di sekeliling. Obyek adalah riil. Contoh obyek adalah orang, *hardware*, *software*, dokumen dan lain-lain.

Setiap obyek mempunyai dua ciri, yaitu *atribut* (*property* atau data) yang menjadi ciri khas dari suatu obyek (what they have) dan *method* (*behavior/function*), yaitu apa yang dapat dilakukan oleh obyek (what they do).

Berorientasi Obyek (*object oriented*) berarti permasalahan didefinisikan melalui istilah dari obyek yang mengkapsulasi data (atribut) dan perilaku (*behavior*), yaitu melalui paradigma/pendekatan obyek.

Selain obyek, ada beberapa istilah yang akan membantu untuk memahami pengertian dalam laporan ini:

a. *Class*, yaitu kumpulan obyek yang sejenis. Secara lebih lugas obyek adalah instant dari sebuah class, atau dengan pengertian lain dengan class dapat menggambarkan *property* dan *behavior* dari tipe obyek.

b. *Inheritance*, adalah penurunan atribut atau method dari suatu obyek *class* ke obyek *class* lainnya.

c. *Polymorphisme*, berasal dari bahasa Yunani yang berarti banyak bentuk.

Dalam konsep ini memungkinkan digunakannya suatu *interface* yang sama untuk memerintah suatu obyek untuk melakukan suatu aksi atau tindakan yang mungkin secara prinsip sama tetapi secara proses berbeda. Secara sederhana bisa juga disebut : satu *interface*, banyak aksi.

Metodologi adalah cara sistematis untuk mengerjakan pekerjaan analisis dan desain. Metodologi berorientasi obyek adalah metode penyelesaian masalah dengan menggunakan pendekatan berorientasi obyek.

Metodologi berorientasi obyek pertama kali muncul pada pertengahan tahun 1970 dan terus berkelanjutan dikembangkan sampai saat ini. Pada tahun 1994 ada 72 lebih metode *object oriented*. Dengan berkembang pesatnya metode ini maka masyarakat object oriented menyadari perlunya standarisasi.

3. Analisa dan perancangan berorientasi obyek

Analisa dan desain berorientasi obyek berarti merumuskan dan menyelesaikan masalah serta menghasilkan suatu hipotesa dan diagnosa (solusi), memodelkannya dengan pendekatan/paradigma obyek (obyek adalah riil punya atribut/data dan perilaku).

Dalam melakukan analisa dan perancangan sistem berorientasi obyek penulis menggunakan UML (*Unified Modelling Language*) untuk memodelkannya. Sedangkan alat (*tool) visual modelling* yang digunakan untuk menggambarkan model analisa dan perancangan adalah *Microsoft Visio 2007*. Implementasi perangkat lunak menggunakan bahasa pemrograman *C#.*

a. Analisa Berorientasi Obyek (*Object Oriented Analysis*)

Object oriented analysis adalah metode analisis yang memeriksa *requirements* (syarat atau keperluan yang harus dipenuhi suatu sistem)

Dalam tahap ini kegiatan-kegiatan yang dilakukan dalam menganalisa sistem sebagai berikut :

a. Menganalisa sistem yang ada dan mempelajari apa yang dikerjakan oleh sistem yang ada.

b. Menspesifikasikan sistem yaitu spesifikasi masukan yang digunakan *database* yang ada, proses yang dilakukan dan keluaran yang dihasilkan.

Tujuan dari analisa berorientasi obyek yaitu untuk menentukan kebutuhan pemakai secara akurat.

Pendekatan-pendekatan yang dipakai dalam analisa berorientasi obyek antara lain :

a. Pendekatan *top down*, yaitu memecahkan masalah ke dalam bagian-bagian terkecil atau per *level* sehingga mudah untuk diselesaikan.

b. Pendekatan modul, yaitu membagi sistem ke dalam modul-modul yang dapat beroperasi tanpa ketergantungan.

c. Penggunaan alat-alat bantu dalam bentuk grafik dan teks sehingga mudah untuk dimengerti serta dikoreksi apabila terjadi perubahan.

Pendekatan dalam analisa berorientasi obyek dilengkapi dengan alat-alat dan teknik-teknik yang dibutuhkan dalam pengembangan sistem, sehingga hasil akhir dari sistem yang dikembangkan akan didapatkan sistem yang terdefinisi dengan baik dan jelas.

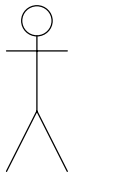
b. Use Case Diagram

Use case diagram menggambarkan kebutuhan sistem dari sudut pandang user. Digunakan untuk menggambarkan hubungan antara *internal* sistem dan *eksternal* sistem atau hubungan antara *use case* dan *actor*.

1) Actor

Actor adalah sesuatu (entitas) yang berhubungan dengan sistem dan berpartisipasi dalam *use case*. *Actor* menggambarkan orang, sistem atau entitas *eksternal* yang secara khusus membangkitkan sistem dengan input atau masukan kejadian-kejadian,

atau menerima sesuatu dari sistem. Actor dilukiskan dengan peran yang mereka mainkan dalam use case, seperti *Staff*, Kurir dan lain-lain.



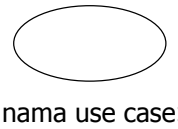
Gambar 2.10 *Bentuk Actor dalam UML*

Dalam *use case diagram* terdapat satu aktor pemulai atau *initiator actor* yang membangkitkan rangsangan awal terhadap sistem, dan mungkin sejumlah aktor lain yang berpartisipasi atau *participating actor*. Akan sangat berguna untuk mengetahui siapa aktor pemulai tersebut.

2) Use Case

Use case yang dibuat berdasar keperluan aktor merupakan gambaran dari "apa" yang dikerjakan oleh sistem, bukan "bagaimana" sistem mengerjakannya. *Use case* diberi nama yang menyatakan apa hal yang dicapai dari interaksinya dengan aktor.

Dalam UML use case dinotasikan dengan gambar :

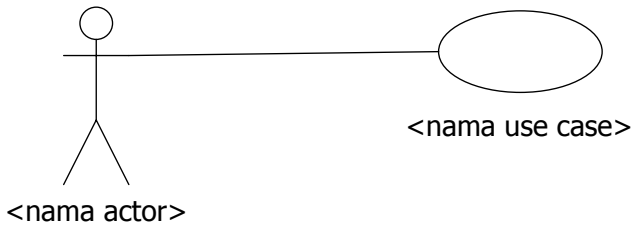


Gambar 2.11. *Bentuk Use Case dalam UML*

3) Relationship

Relasi (*relationship*) digambarkan sebagai bentuk garis antara dua simbol dalam use case diagram. Relasi antara *actor* dan *use case* disebut juga dengan asosiasi (*association*). Asosiasi ini digunakan untuk menggambarkan bagaimana hubungan antara keduanya.

Relasi-relasi yang terjadi pada *use case diagram* bisa antara *actor* dengan *use case* atau *use case* dengan *use case*.



Gambar 2.12. *Bentuk Relationship dalam UML*

Relasi antara use case dengan use case :

a. *Include*, pemanggilan *use case* oleh *use case* lain atau untuk menggambarkan suatu *use case* termasuk di dalam *use case* lain (diharuskan). Contohnya adalah pemanggilan sebuah fungsi program. Digambarkan dengan garis lurus berpanah dengan tulisan <<*include*>>.

b. *Extend*, digunakan ketika hendak menggambarkan variasi pada kondisi perilaku normal dan menggunakan lebih banyak kontrol *form* dan mendeklarasikan *ekstension* pada *use case* utama. Dengan kata lain adalah perluasan dari *use case* lain jika syarat atau kondisi terpenuhi. Digambarkan dengan garis berpanah dengan tulisan <<*extend*>>.

c. *Generalization/Inheritance*, dibuat ketika ada sebuah kejadian yang lain sendiri atau perlakuan khusus dan merupakan pola berhubungan *base-parent use case.* Digambarkan dengan garis berpanah tertutup dari *base use case* ke *parent use case.*

c. Activity Diagram

Diagram aktivitas menggambarkan proses bisnis dan urutan aktivitas-aktivitas yang mendukung penggambaran tindakan sistem baik yang bersifat kondisional maupun paralel. Tindakan kondisional dilukiskan dengan cabang (*branch*) dan penyatuan (*merge*).

Sebuah branch memiliki sebuah transition masuk atau yang disebut dengan incoming transition dan beberapa transition keluar atau yang disebut dengan outgoing transition dari branch yang berupa keputusan-keputusan. Hanya satu dari outgoing transition yang dapat diambil, maka keputusan-keputusan tersebut harus bersifat mutually exclusive. [else] digunakan sebagai keterangan singkat yang menunjukkan bahwa transition "else" tersebut harus digunakan jika semua keputusan yang ada pada branch salah.

Sebuah *merge* memiliki banyak *input transition* dan sebuah *output*. *Merge* menandakan akhir dari suatu kondisi yang diawali dengan sebuah *branch*. Selain *branch* dan *merge,* di dalam diagram aktivitas terdapat pula *fork* dan *join. Fork* memiliki satu *incoming transition* dan beberapa *outgoing transition*. Sedangkan pada *join, outgoing transition* diambil atau digunakan hanya ketika semua *state* pada *incoming transition* telah menyelesaikan aktivitasnya.

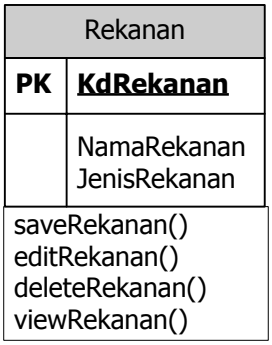
d. Sequence Diagram

Diagram yang menggambarkan bagaimana obyek berinteraksi dengan obyek lainnya melalui pesan (*message)* yang disampaikan, disusun dalam urutan kejadian atau waktu dan secara khusus berasosiasi dengan *use case.*

e. Class Diagram

Class diagram merupakan bagian yang paling penting dalam analisa dan perancangan berorientasi obyek. Dalam UML diagram kelas digunakan untuk memodelkan *static structure* dari sistem informasi.

Kelas merupakan himpunan dari obyek yang sejenis yang mempunyai atribut (*attribute*) dan perilaku (*behaviors/method*) yang sama. Atribut adalah sebuah nilai data karakteristik yang dimiliki oleh obyek sebuah kelas sedangkan *method* adalah perilaku atau operasi yang dikenakan oleh suatu kelas. Pada gambar kelas terdapat tiga bagiannya.



Gambar 2.13. *Bentuk Class dalam UML*

Diagram kelas menggambarkan struktur obyek sistem, dimana diperlihatkan hubungan amtar mereka. Diagram kelas merupakan fondasi untuk *component diagram* dan *deployment diagram*.

Secara garis besar terdapat 3 jenis *class*. Ketiga jenis *class* tersebut dikelompokkan berdasarkan fungsi dan karakternya masing-masing, yaitu :

a. Entity Class Diagram

Merupakan paket utama dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk sistem dan menjadi landasan untuk menyusun basis data pada model data konseptual.



Gambar 2.14*. Bentuk Entity Class dalam UML*

b. Control Class Diagram

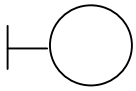
Berisi kumpulan kelas yang menjadi kontrol program termasuk koneksi dengan basis data dan merupakan kelas perantara atau penghubung antara *entity class* dengan kelas antar muka pemakai (*interface).*



Gambar 2.15*. Bentuk Control Class dalam UML*

c. Boundary Class Diagram

Berisi kumpulan kelas yang menjadi interface antara pemakai (*user*) dengan sistem, seperti tampilan *form* untuk pencetakan.



***Gambar 2.16**. Bentuk Boundary Class dalam UML*

f. Package Diagram

Package (paket) adalah mekanisme pengelompokan yang digunakan untuk menandakan pengelompokan elemen-elemen model. Sebuah *package* dapat mengandung beberapa paket lain di dalamnya. *Package* digunakan untuk memudahkan pengorganisasian elemen-elemen model.

g. Perancangan Berorientasi Obyek (*Object Oriented Design*)

Object oriented design adalah metode untuk mengarahkan arsitektur *software* yang didasarkan pada manipulasi obyek-obyek sistem atau subsistem.

Perancangan berorientasi obyek merupakan proses spesifikasi yang terperinci atau pendefinisian dari kebutuhan-kebutuhan fungsional dan persiapan untuk rancang bangun implementasi yang menggambarkan bagaimana suatu sistem dibentuk. Untuk mengembangkan suatu sistem baru digunakan dengan menguraikan hubungan proses-proses dalam bentuk diagram-diagram.

Perancangan berorientasi obyek bertujuan untuk :

a. Sistematika proses pendesainan
b. Menghasilkan pendesainan model program
c. Memberikan gambaran pemecahan masukan dengan efektif

2.2.3.7 Kamus Data

Kamus Data menurut Jogiyanto adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi[2]. Kamus data merupakan katalog fakta data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Kamus data dibuat berdasarkan arus data yang ada pada diagram alir data. Kamus data harus juga dapat mencerminkan keterangan yang jelas tentang data yang dicatat. Kamus data atau *data dictionary* disebut juga dengan istilah sistem *data dictionary.*

Adapun isi dari kamus data adalah:

a) Nama arus data

   Kamus data dibuat berdasarkan arus data dalam DAD sehingga nama arus data harus dicatat di kamus data

b) Alias

   Merupakan nama lain dari data yang harus dituliskan. Karena sering terdapat data yang sama tetapi mempunyai nama yang berbeda departemen yang satu dengan yang lainnya.

c) Bentuk data

   Bentuk data berupa dokumen dasar laporan cetak, parameter dan *field-field*.

d) Arus Data

   Menjelaskan dari mana arus data mengalir dan kemana data akan menuju.

e) Penjelasan

   Untuk menjelaskan makna dari arus data yang dicatat pada kamus.

f) Periode

   Menunjukan kapan terjadi arus data sampai dengan kapan proses program beserta laporan-laporan dapat dihasilkan.

g) Volume

   Volume yang dicatat berupa *volume* rata-rata dan *volume* puncak suatu arus data.

h) Struktur data

   Menunjukan arus data yang dicatat pada kamus data yang terdiri dari elemen-elemen data. Dalam kamus data dikenal dengan sebutan notasi. Notasi

merupakan kamus data yang mempunyai suatu bentuk untuk mempersingkat arti dari simbol yang dijelaskan.

2.2.3.8 Pengolahan Data

Menurut Moekijat pengolahan data adalah kegiatan pikiran dengan bantuan tangan atau dengan suatu peralatan dan mengikuti serangkaian langkah, perumusan atau pola tertentu untuk mengubah data sehingga data tersebut baik bentuk, susunan, sifat, atau isinya menjadi lebih berguna[4].

Menurut Mahyuzir pengolahan data adalah Suatu kesatuan prosedur-prosedur yang terlibat langsung dengan pengolahan data, prosedur- prosedur tersebut seperti prosedur penerimaan, pencatatan, penyimpanan, persiapan dan penyajian informasi yang di perlukan untuk mecapai tujuan, beliau juga menuturkan bahwa pengolahan data adalah segala macam manipulasi terhadap data agar data itu dapat berguna sesuai dengan yang di inginkan[5].

Menurut Budi Sutedjo pengolahan data adalah tahap ini merupakan tahap dimana data diolah sesuai dengan prosedur yang telahdimasukan[6].

Pengolahan Data adalah manifulasi dari data kedalam bentuk yang lebih berguna berarti. Dengan demikian dapat disimpulkan bahwa Pengolahan Data merupakan kegiatan yang dilakukan dengan menggunakan masukan berupa data dan menghasilkan informasi yang bermanfaat untuk tujuan sesuai dengan yang direncanakan.

Pengolahan data adalah manipulasi data agar menjadi bentuk yang lebih berguna, pengolahan data ini tidak hanya melibatkan perhitungan numerik tetapi juga operasi - operasi, klasifikasi data dan perpindahan data suatu tempat ketempat lain atau segala macam pengolahan terhadap data agar data itu berguna sesuai dengan diinginkan.

2.2.4 Perangkat Lunak yang Digunakan

2.2.4.1 C Sharp (c#)

Microsoft Visual C-Sharp atau yang lebih dikenal dengan *Visual C#* adalah sebuah bahasa pemrograman yang tidak diragukan lagi dalam proses pengembangan aplikasi berbasis *.NET Framework*, dimana *C#* bebas dari masalah kompatibilitas dilengkapi dengan berbagai fitur yang sebagian besar merupakan fitur baru, menarik, dan tentu saja menjanjikan.

C# (dibaca: *C sharp*) merupakan sebuah bahasa pemograman yang berorientasi objek yang dikembangkan oleh *Microsoft* sebagai bagian dari inisiatif kerangka *.Net Framework*. Bahasa pemrograman ini dibuat berbasiskan bahasa *C++* yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya seperti *Java, Delphi, Visual Basic,* dan lain-lain dengan beberapa penyederhanaan.

.



***Gambar 2.17**. Microsoft Visual C#*

Seperti pemrograman *Java, C#* juga tidak memperbolehkan multiple inherintance atau penggunaan pointer (pada *safe/managed code*), tetapi *C#* menyediakan *gargabe memory collection* pada saat *runtime* dan pada saat pengecekan akses memori.

Meskipun bertentangan dengan *pemrograman Java, C#* tetap mempertahankan operasi unik yang terdapat pada bahasa pemrograman *C++* seperti *overloading, enumerations, pre-processor directive, pointer* (pada *unmanaged/unsafe code*), dan fungsi *pointer*. Seperti halnya *Visual Basic,* bahasa pemrograman *C#* juga dilengkapi dengan *properties.*

Alasan kenapa bahasa pemrograman *C#* menjadi pilihan utama dari programmer untuk mengembangkan suatu aplikasi yaitu :

1. *C#* merupakan bahasa pemrograman yang berorientasi objek
2. *C#* merupakan bahasa pemrograman yang sederhana
3. *C#* termasuk kedalam bahasa pemrograman modern
4. *C#* mampu untuk membuat berbagai aplikasi
5. Efisien
6. *C#* bersifat modular
7. *C#* merupakan bahasa pemrograman masa depan

2.2.4.2 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data *SQL* (bahasa Inggris: *database management system*) atau DBMS yang *multithread, multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia.

MySQL adalah *Relational Database Management System* (RDBMS) yang didistribusikan secara gratis dibawah *lisensi* GPL (*General Public License*). Dimana setiap orang bebas untuk menggunakan *MySQL*, namun tidak boleh dijadikan produk turunan yang bersifat *closed* atau komersial. *MySQL* sebenarnya merupakan turunan salah satu konsep utama dalam *database* sejak lama, *yaitu SQL (Structured Query Language). SQL* adalah sebuah konsep pengoperasian *database*, terutama untuk pemilihan atau seleksi dan pemasukan data yang memungkinkan pengoperasian 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* lainnya dalam *query* data. Hal ini terbukti untuk *query* yang dilakukan oleh *single user*, kecepatan *query MySQL* bisa sepuluh kali lebih cepat dari *PostgreSQL* dan lima kali lebih cepat dibandingkan *Interbase.* Selain itu *MySQL* juga memiliki beberapa keistimewaan, antara lain :

1. *Portability*

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

2. *Open Source*

MySQL didistribusikan secara open source (gratis), dibawah lisensi *GPL* sehingga dapat digunakan secara cuma-cuma.

3. *Multiuser*

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

4. *Performance Tuning*

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

5. *Column Types*

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

6. *Command and Functions*

MySQL memiliki operator dan 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* terenskripsi.

8. *Scalability and Limits*

MySQL mampu menangani *database* skala besar, dengan jumlah *records* lebih dari 50 juta dan 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, Unit soket (UNIX),* atau *Named Pipes (NT).*

10. *Localisation*

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 pemrograman dengan menggunakan fungsi API *(Application Programming Interface).*

12. *Clients and Tools*

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

13. Struktur Tabel

MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani *ALTER TABLE,* dibandingkan *database* lainnya semacam *PostgreSQL* ataupun *Oracle.*

14. *Create*

Pembuatan tabel dibuat dengan menggunakan perintah *CREATE TABLE* Format Pembuatan Tabel adalah :

```
CREATE TABLE nama_tabel(
kolom1 tipedata(ukuran) Option_Kolom,
kolom2 tipedata(ukuran) Option_Kolom,
...,
kolomN tipedata(ukuran)
) ENGINE=NamaEngine;
```

Option_Kolom tidak wajib disertakan. Option_Kolom yang diperbolehkan diantaranya : null/not null, primary key, auto_increment, unique, default dan comment. Contoh NamaEngine yaitu MyISAM dan InnoDB

15. *Read*

SELECT digunakan untuk mengambil data dari database. Bentuk umum pemanggilan SELECT adalah :

```
SELECT  * | {[DISTINCT] column|expression [alias], …}
```

FROM nama_table

[WHERE kondisi];

Keterangan :

a. * digunakan untuk menampilkan semua kolom yang ada dalam table
b. Daftar_kolom digunakan untuk mengampikan kolom tertentu saja
c. WHERE digunakan jika ingin membatasi data yang ditampilkan

16. *Update*

UPDATE digunakan untuk mengganti isi data. Bentuk umum penggunaan update adalah :

UPDATE nama_tabel

SET kolom1={isi|DEFAULT} [, kolom2={expr2|DEFAULT}] ...]

[WHERE kondisi_where]

[ORDER BY nama_kolom_pengurutan]

[LIMIT banyak_baris]

Keterangan :

a. WHERE digunakan untuk membatasi banyaknya baris yang diupdate
b. ORDER BY digunakan untuk melakukan pengurutan data yang akan diupdate.
c. LIMIT digunakan untuk membatasi baris yang akan diupdate (setelah data dibatasi dengan WHERE).

17. *Delete*

Penghapusan tabel dibuat dengan menggunakan perintah *DELETE*. Format penghapusan Tabel adalah :

DELETE From nama_table

[WHERE kondisi];

Where digunakan untuk membatasi banyaknya data yang akan dihapus.