BAB 2
LANDASAN TEORI

Pengembangan program sistem pakar yang dapat membantu masyarakat mendiagnosa penyakit lele dan cara pembudidayaan ikan lele yang baik agar dapat meningkatkan hasil produktifitas memerlukan faktor-faktor pendukung yang merupakan landasan teori yang akan digunakan dalam proses pengerjaan.

2.1. Kecerdasan buatan (*Artificial intelligence*)

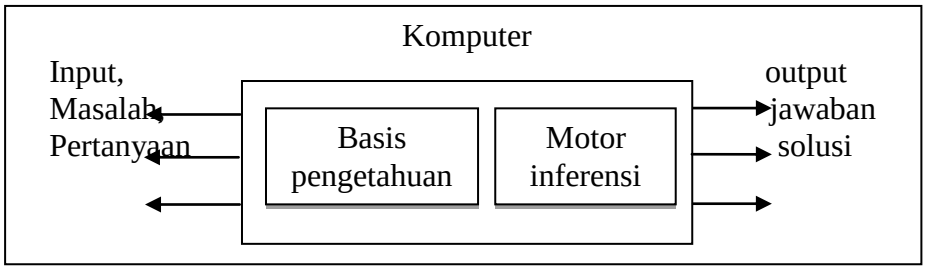
Artificial intelligence adalah sebuah rancangan program yang memungkinkan komputer melakukan suatu tugas atau mengambil keputusan dengan meniru suatu cara berpikir dan penalaran manusia.

2.1.1 Konsep Kecerdasan Buatan

Kecerdasan buatan berasar dari kata *Artificial Intelligence* yang mengandung arti tiruan atau kecerdasan. Secara harfiah *Artificial Intelligence* adalah kecerdasan buatan. Kecerdasan buatan adalah salah satu bidang dalam ilmu komputer yang yang membuat komputer agar dapat bertindak seperti manusia (menirukan kerja otak manusia).

Aplikasi kecerdasan buatan terdiri dari 2 bagian utama yang harus dimiliki, diantaranya:

a. Basis pengetahuan (*Knowledge-Base*), berisi fakta-fakta, teori, pemikiran dan hubungan antara satu dengan yang lainnya.
b. Motor Inferensi (*Inference Engine*) , yaitu kemampuan menarik kesimpulan berdasarkan pengalaman.



Gambar 2.1. Penerapan Konsep Kecerdasan Buatan [2]

2.1.2 Sistem pakar

Sistem pakar *(expert system)* adalah salah satu teknik kecerdasan buatan yang berusaha mengadopsi pengetahuan manusia ke komputer, agar komputer dapat menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli. Sistem pakar yang baik dirancang agar dapat menyelesaikan suatu permasalahan tertentu dengan meniru kerja dari para ahli. Dengan sistem pakar ini, orang awampun dapat menyelesaikan masalah yang sebenarnya hanya dapat diselesaikan dengan bantuan para ahli. Bagi para ahli, sistem pakar ini juga akan membantu aktivitasnya sebagai asisten yang sangat berpengalaman. Ada beberapa definisi tentang sistem pakar, antara lain [2].

.a. Menurut Durkin

Sistem pakar adalah suatu program komputer yang dirancang untuk memodelkan kemampuan penyelesaian masalah yang dilakukan seorang pakar.

b. Menurut Ignizio

Sistem pakar adalah suatu model dan prosedur yang berkaitan, dalam suatu domain tertentu, yang mana tingkat keahliannya dapat dibandingkan dengan keahlian seorang pakar.

c. Menurut Giarratano dan Riley

Sistem pakar adalah suatu sistem komputer yang bisa menyamai atau meniru kemampuan seorang pakar.

2.1.2.1 Keuntungan Sistem Pakar

Keuntungan dari sistem pakar, antara lain [2] :

1. Memungkinkan orang awam bisa mengerjakan pekerjaan para ahli.
2. Bisa melakukan proses secara berulang secara otomatis.
3. Menyimpan pengetahuan dan keahlian pakar.
4. Meningkatkan output dan produktivitas.
5. Meningkatkan kualitas.
6. Mampu mengambil dan melestarikan keahlian pakar (terutama yang termasuk keahlian langka).

7. Memiliki kemampuan untuk mengakses pengetahuan.
8. Dapat bekerja dengan informasi yang kurang lengkap dan tidak pasti.
9. Sebagai media pelengkap dalam pelatihan.
10. Meningkatkan kapasitas dalam penyelesaian masalah.
11. Menghemat waktu dalam pengambilan keputusan.

2.1.2.2 Kelemahan sistem pakar

Kelemahan dari sistem pakar, antara lain [2]:

1. Biaya yang diperlukan untuk membuat, memelihara, dan mengembangkannya sangat mahal
2. Sulit dikembangkan, hal ini erat kaitannya dengan ketersediaan pakar di bidangnya.
3. Sistem pakar tidak 100% bernilai benar.

2.1.2.3 Konsep Dasar Sistem Pakar

Konsep dasar sistem pakar mengandung keahlian, ahli/pakar, pengalihan keahlian, mengambil keputusan, aturan, kemampuan menjelaskan [2].

1. Keahlian

   Keahlian bersifat luas dan merupakan penguasaan pengetahuan dalam bidang khusus yang diperoleh dari pelatihan, membaca atau pengalaman. Contoh bentuk pengetahuan yang termasuk keahlian:

   a. Fakta - fakta dalam lingkup permasalahan.
   b. Teori – teori dalam lingkup permasalahan
   c. Aturan dan prosedur baku berkenaan dengan lingkup permasalahan.
   d. Strategi untuk menyelesaikan masalah.

2. Pakar

   Seorang ahli adalah seseorang yang mampu menjelaskan suatu tanggapan, mempelajari hal-hal baru seputar topik permasalahan, menyusun kembali pengetahuan jika dipandang perlu, memecahkan masalah dengan cepat dan tepat.

3. Pengalihan Keahlian

Tujuan dari sistem pakar adalah untuk mentransfer keahlian dari seorang pakar ke dalam komputer kemudian ke masyarakat. Proses ini meliputi 4 kegiatan, yaitu perolehan pengetahuan (dari para ahli atau sumber-sumber lainnya), representasi pengetahuan ke komputer, kesimpulan dari pengetahuan dan pengalihan pengetahuan ke pengguna.

4. Penalaran (inferensi)

   Salah satu fitur yanfg dimiliki oleh sistem pakar adalah kemampuann untuk menalar. Jika kepakaran sudah tersimpan sebagai basis pengetahuan dan tersedia program yang mampu mengakses basis data, maka komputer untuk mengambil kesimpulan dilakukan oleh komponen yang dikenal dengan mesin inferensi yaitu meliputi prosedur tentang pemecahan masalah.

5. Aturan

   Sistem pakar yang dibuat merupakan sistem yang berdasarkan pada aturan – aturan dimana program disimpan dalam bentuk aturan-aturan sebagai pro sedur

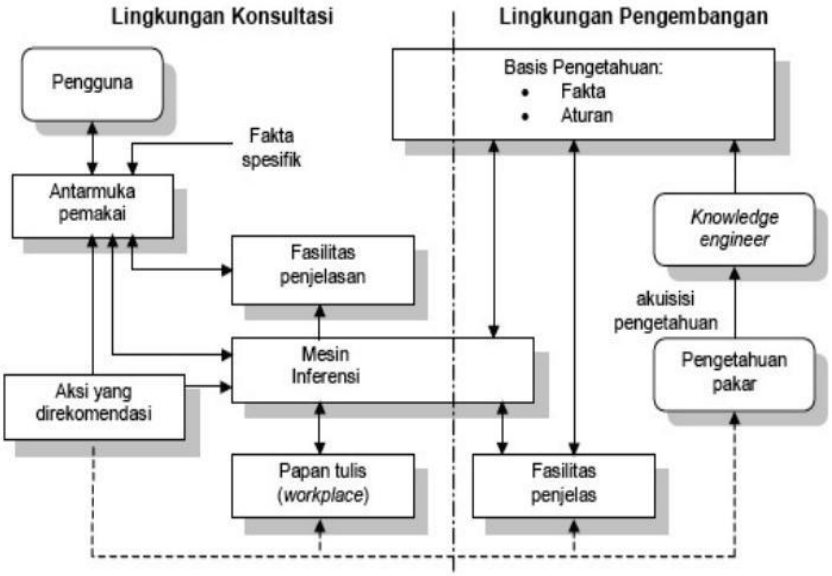
   pemecahan masalah. Aturan tersebut biasanya berbentuk *IF – THEN*.

6. Kemampuan Menjelaskan

   Keunikan lain dari sistem pakar adalah kemampuan dalam menjelaskan atau memberi saran/rekomendasi serta juga menjelaskan mengapa beberapa tindakan/saran tidak direkomendasikan.

2.1.2.4 Struktur Sistem Pakar

Sistem pakar terdiri dari dua bagian utama yaitu *development environment* dan *consultation environment*. *Development environment* dipakai oleh pembangun sistem pakar untu membangun komponen-komponen dan mengenalkan suatu sistem pakar pengetahuan kepada *knowledge base*. *Consultatition environment* dipakai oleh *user* untuk mendapatkan suatu pengetahuan yang berhubungan dengan suatu keahlian. Komponen-komponen sistem pakar dalam kedua bagian tersebut dapat dilihat dalam Gambar 2.2 dibawah ini :



Gambar 2.2 Struktur Sistem pakar [2]

Komponen utama pada struktur sistem pakar meliputi:

1. Basis Pengetahuan (Knowledge Base)

Basis pengetahuan merupakan inti dari suatu sistem pakar, yaitu berupa representasi pengetahuan dari pakar. Basis pengetahuan tersusun atas fakta dan kaidah. Fakta adalah informasi tentang objek, peristiwa, atau situasi. Kaidah adalah cara untuk membangkitkan suatu fakta baru dari fakta yang sudah diketahui. Menurut Gondran (1986) dalam Utami (2002), basis pengetahuan merupakan representasi dari seorang pakar, yang kemudian dapat dimasukkan kedalam bahasa pemrograman khusus untuk kecerdasan buatan (misalnya PROLOG atau LISP) atau shell sistem pakar (misalnya EXSYS, PC-PLUS, CRYSTAL, dsb.)

2. Mesin Inferensi (Inference Engine)

Mesin inferensi berperan sebagai otak dari sistem pakar. Mesin inferensi berfungsi untuk memandu proses penalaran terhadap suatu kondisi, berdasarkan pada basis pengetahuan yang tersedia. Di dalam mesin inferensi terjadi proses untuk memanipulasi dan mengarahkan kaidah, model, dan fakta yang disimpan dalam basis pengetahuan dalam rangka mencapai solusi atau kesimpulan. Dalam prosesnya, mesin inferensi menggunakan strategi penalaran dan strategi pengendalian. Strategi penalaran terdiri dari strategi penalaran pasti (Exact

Reasoning) dan strategi penalaran tak pasti (Inexact Reasoning). Exact reasoning akan dilakukan jika semua data yang dibutuhkan untuk menarik suatu kesimpulan tersedia, sedangkan inexact reasoning dilakukan pada keadaan sebaliknya. Strategi pengendalian berfungsi sebagai panduan arah dalam melakukan prose penalaran. Terdapat tiga tehnik pengendalian yang sering digunakan, yaitu forward chaining, backward chaining, dan gabungan dari kedua tehnik pengendalian tersebut.

3. **Basis Data (Database)**

Basis data terdiri atas semua fakta yang diperlukan, dimana fakta-fakta tersebut digunakan untuk memenuhi kondisi dari kaidah-kaidah dalam sistem. Basis data menyimpan semua fakta, baik fakta awal pada saat sistem mulai beroperasi, maupun fakta-fakta yang diperoleh pada saat proses penarikan kesimpulan sedang dilaksanakan. Basis data digunakan untuk menyimpan data hasil observasi dan data lain yang dibutuhkan selama pemrosesan.

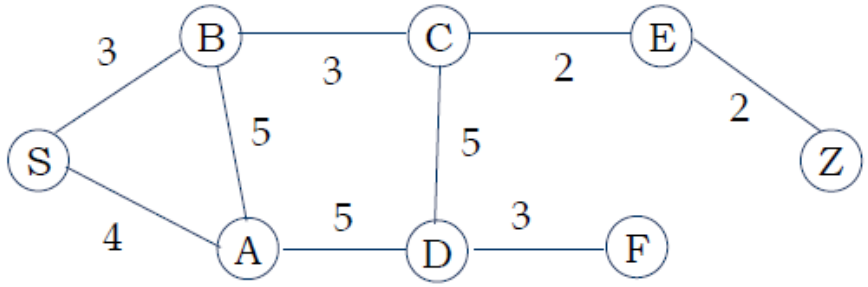
4. **Antarmuka Pemakai (User Interface)**

Fasilitas ini digunakan sebagai perantara komunikasi antara pemakai dengan sistem.

2.1.2.5 Pohon Pelacakan

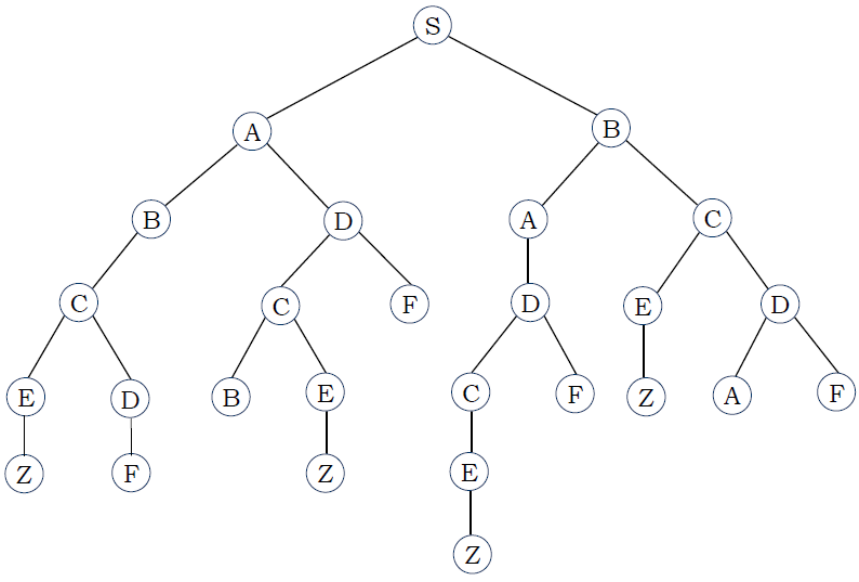
Untuk menghindari kemungkinan adanya proses pelacakan suatu node secara berulang maka digunakan struktur pohon. Struktur pohon digunakan untuk menggambarkan keadaan secara hirarkis.Pohon juga terdiri dari beberapa node. Node yang terletak pada level-0 disebut juga "akar". Node akar menunjukkan keadaan awal yang biasanya merupakan topik atau obyek. Node akar ini terletak pada level ke-0. Node akar mempunyai beberapa percabangan yang terdiri atas beberapa node successor yang sering disebut dengan nama "anak" dan merupakan node-node perantara. Namun jika dilakukan pencarian mundur, maka dapat dikatakan bahwa node tersebut memiliki predecessor. Node-node yang tidak mempunyai anak sering disebut dengan nama node "daun" yang menunjukkan akhir dari suatu pencarian, dapat berupa tujuan yang diharapkan (goal) atau jalan buntu (dead end).

Dalam gambar 2.3 ditampilkan sebuah contoh permasalahan mendasar untuk digunakan dalam penggunaan beberapa metode pencarian. Simpul S merupakan simpul awal dimulainya penelusuran, simpul Z adalah simpul yang akan menjadi tujuan.



Gambar 2.3 Contoh Graph yang Berisi Path Antar Kota [2]

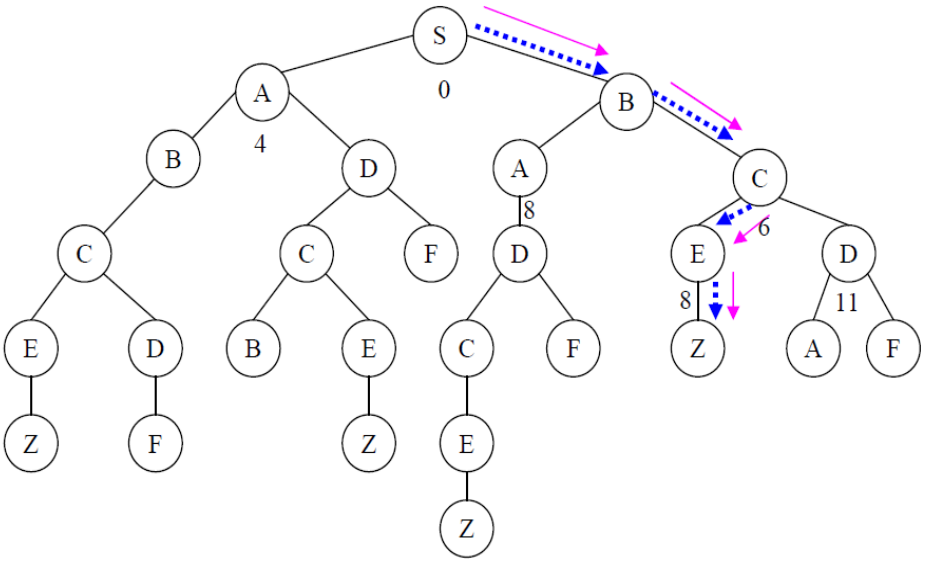
Dari graph di atas, dibuat struktur tree-nya. Pada gambar 2.4 menggambarkan tree yang didapat dari Graph gambar 2.3.



Gambar 2.4 Struktur Tree dari Graph Gambar 2.3 [2]

2.1.2.6 Metode Pencarian *Best first search*

Bekerja berdasarkan kombinasi kedua metode sebelumnya. Gambar 2.7 (Rusell Stuart, 1995) menunjukkan penelusuran secara *best first search*.



Gambar 2.5 Tree untuk *Best-first Search[2]*

1. Keuntungan
   a. Membutuhkan memori yang relative kecil, karena hanya node-node pada lintasan yang aktif saja yang disimpan.
   b. Secara kebetulan, metode best first search akan menemukan solusi tanpa harus menguji lebih banyak lagi dalam ruang keadaan.
2. Kelemahan
   a. Algoritma akan berhenti kalau mencapai nilai optimum local.
   b. Tidak diijinkan untuk melihat satupun langkah sebelumnya.

2.2. Budidaya ikan lele

Ikan lele merupakan salah satu jenis ikan konsumsi yang populer di masyarakat. Kandungan gizi yang tinggi pada ikan lele serta harganya yang sangat terjangkau membuat banyak masyarakat memilih ikan lele sebagai konsumsi mereka. Apalagi dalam 2 tahun belakangan ini permintaan akan ikan lele meningkat dengan tajam. Terbukti dari semakin banyaknya peternak lele yang berusaha mensuplai lele ke masyarakat dan tetap dinyatakan bahwa masih banyak jumlah permintaan yang belum terpenuhi.

2.2.1 Pembesaran lele

Pembesaran ikan merupakan suatu kegiatan budidaya yang bertujuan untuk menghasilkan ikan lele konsumsi. Dalam kegiatan pembesaran, ikan lele didorong untuk tumbuh secara maksimum hingga mencapai ukuran panen atau sesuai ukuran pasar melalui penyediaan lingkungan media hidup ikan yang optimal, pemberian pakan yang tepat serta pengendalian hama penyakit. Adapun ukuran lele konsumsi adalah 8-12 ekor/kg.

2.2.1.1 Persiapan kolam pembesaran

Sebelum digunakan, wadah pembesaran dipersiapkan terlebih dahulu. Persiapan bertujuan untuk menyediakan lingkungan yang optimal bagi ikan. Beberapa kegiatan yang dilakukan dalam persiapan kolam pembesaran, diantaranya sebagai berikut [3] :

1. Pengapuran

Pengapuran dilakukan sesusai panen. Hal ini disebabkan dasar kolam membentuk keasaman yang tinggi pada budidaya lele intensif. Pengapuran bertujuan untuk menaikan pH tanah, membunuh hama , parasit dan penyakit ikan. Serta mempercepat pembongkaran bahan-bahan organik. Jenis kapur yang biasa digunakan untuk pengapuran kolam, diantaranya kapur pertanian Ca Mg($CaCO_3$) atau dolomit dalam bentuk Ca Mg(CO_3)$_2$, kapur tohor(CaO), dan kapur mati Ca(OH)$_2$.

Pemberian kapur dilakukan dengan cara disebar merata dipermukaan tanah dasar kolam. Setelah pengapuran selesai,tanah dasar kolam dibalik dengan cangkul sehingga kapur bisa lebih masuk kedalam lapisan tanah dasar. Pengapuran utuk kolam semen dan terpal dilakukan dengan cara dinding kolam dan dasar kolam dikuas dengan kapur yang telah dicampur air. Kapur yang sering digunakan kapur pertanian atau dolomit dengan dosis 60 gram/m^2. Dosis kapur yang digunakan tergantung pada pH tanah. Semakin rendah pH tanah, kebutuhan kapur untuk pengapuran semakin banyak.

2. Pemupukan

Pemupukan berguna untuk menyediakan media tumbuh pakan alami dan unsur hara bagi plankton yang menjadi pakan bagi ikan lele,terutama

benih.ketersediaan pakan alami sangat penting dan dibutuhkan benih ikan terutama pada tahap awal penebartan.

Pupuk yang sering digunakan terdiri dari kotoran ternak besar (sapi, domba atau kerbau) dengan dosis 150g/m^2 , kotoran ayam sebanyak 250-500gram/m^2, pupuk urea15 gram/m^2, dan TPS 10 gram/m^2. Dosis tersebut tidak mutlak, tetapi disesuaikan dengan kesuburan kolam. Khusus pupuk organik(kandang) sebaiknya menggunakan pupuk yang sudah jadi (masak) dan kering. Selanjutnya dibiarkan selama 3 hari.

Cara pemberian pupuk kandang dapat dilakukan dengan cara disebar pada dasar kolam maupun dionggokkan dibeberapa tepi kolam dengan menggunakan karung. Untuk pupuk TPS dan urea,pupuk diberikan dengan cara disebar pada dasar kolam.

Tujuan pemupukan untuk meningkatkan kandungan hara bagi kebutuhan fitiplankton untuk berfotosistesis. Hasil pemupukan dapat dilihat pada perubahan warna air kolam. Air kolam yang telah dipupuk menjadi hijau atau hijau kecoklatan. Keberadaan fgitoplankton di air dapat mendorong pertumbuhan populasi zoompankton sehingga bisa meningkatkan ketersediaan pakan alami.

3. Pengisian air kolam

Pengisian air kolam dilakukan setelah kegiatan pengapuran dan pemupukan selesai. Pengairan kolam dilakukan hingga ketinggian air mencapai 30-40 cm. Pada ketinggian air tersebut sinar matahari masih bisa mencapai dasar kolam tempat terdapatnya pupuk. Keberadaan unsur hara dan sinar matahari merupakan syarat tumbuhnya fitoplankton dikolam. Selanjutnya, kolam yang telah diairi dibiarkan selama 5-7 hari agar ditumbuhi plankton. Tanda-tanda air yang ditumbuhi plankton biasanya warna air berubah menjadi kehijau-hijauan.

Ketinggian air kolam dipertahankan 30-40 cm pada waktu penebaran benih ikan karena ukuran benih ikan masih kecil, jika ketinggian air lebih dalam lagi, bibit ikan lele yang masih kecil akan kesulitan bergerak sampai kepermukaan air untuk mengambil pakan atau proses pernapasan. Ketinggian air kolam ditambah secara berkala seiring dengan bertambahnya ukuran dan berat lele hingga ketinggian ideal antara 100-120 cm.

2.2.1.2 Penebaran benih

Penebaran benih adalah penempatan ikan dalam wadah budidaya dengan padat penebaran tertentu. Benih bisa berasal dari pemijahan sendiri atau membeli dari petani pembenih atau ke balai benih ikan (BBI). Ukuran benih yang ditebarkan menentukan lama waktu pemeliharaan untuk mencapai ukuran panen tertemtu. Berikut uraian kegiatan yang berhubungan dengan penebaran benih [3].

1. Cara memperoleh benih

   Cara memperoleh benih lele dengan memijahkan sendiri atau membeli dipetani(pembenih) maupun pengepul. Harga benih lele sangat bervariasi, tergantung ukuran benih,wilayah, atau daerah tempat menjualnya.

2. Syarat benih

   Benih lele yang dipilih harus benar-benar baik dan sehat. Benih lele yang tidak baik gampang sekali terserang penyakit dan pertumbuhannya kurang optimal. Pemilihan benih lele yang baik dan sehat memiliki ciri-ciri sebagai berikut:

   a. Ukuran seragam dan berwarna cerah (mengkilap).
   b. Gerakan lincah dan gesit.
   c. Tidak cacat dan tidak luka ditubuhnya.
   d. Bebas dari bibit penyakit.
   e. Posisi tubuh dalam air normal.
   f. Menghadap dan melawan arus ketika diberi arus.

   Ukuran benih lele sebaiknya seragam 5-7 cm/ekor. Tujuannya agar masing-masing lele tidak saling mengganggu dan pertumbuhannya bisa seragam. Sesuai dengan karakternya, lele adalah binatang kanibal. Jika kekurangan pakan, lele akan memangsa sesamanya, terutama lele berukuran kecil. Benih lele yang berukuran kecil juga akan kalah dalam bersaing mendapat pakan.

3. Penebaran benih

   Penebaran benih merupakan salah satu faktor yang menentukan dari kegiatan awal pemeliharaan ikan dikolam. Kesalahan dalam penebaran ikan, baik cara maupun waktunya, dapat menyebabkan ikan stres dan akhirnya mati.benih ikan dapat ditebar dikolam jika kondisi kolam telah memenuhi

syarat-syarat sebagai berikut:

a. Kedalaman air kolam dipetahankan 30-40 cm.
b. Air kolam sudah ditumbuhi plankton atauu pakan alami.
c. Kualitas airnya sudah memenuhi syarat untuk budidaya ikan.

4. Vaksinasi

Cara vaksinasi sebelum benih ditebarkan:

a. Untuk mencegah penyakit karena bakteri, sebelum ditebarkan, lele yang berumur 2 minggu dimasukkan dulu ke dalam larutan formalin dengan dosis 200 ppm selama 10-15 menit. Setelah divaksinasi lele tersebut akan kebal selama 6 bulan.
b. Pencegahan penyakit karena bakteri juga dapat dilakukan dengan menyutik dengan terramycin 1 cc untuk 1 kg induk.
c. Pencegahan penyakit karena jamur dapat dilakukan dengan merendam lele dalam larutan Malachite Green Oxalate 2,5–3 ppm selama 30 menit.

2.2.1.3 Pemberian Pakan

Pakan merupakan komponen biaya produksi terbesar dalam budidaya ikan lele secara intensif. Kebutuhan pakan mutlak mengandalkan pakan buatan pabrik (pelet). Pakan buatan pabrik lebih terjamin kualitasnya serta kandungan nutrisinya lengkap.

Cara pemberian pakan untuk budidaya lele, antara lain [3]:

a. Pellet mulai dikenalkan pada ikan lele saat umur 6 minggu dan diberikan pada ikan lele 10-15 menit sebelum pemberian makanan yang berbentuk tepung.
b. Pada minggu 7 dan seterusnya sudah dapat langsung diberi makanan yang berbentuk pellet. Saat pemberian pakan tambahan, campurkan Migro Suplemen merata pada pakan, dosis pemberiannya adalah 10ml Migro Suplemen dicampur air secukupnya (jangan terlalu banyak) Kemudian aduk merata pada 3kg pakan buatan. Dianjurkan diberikan pada setiap pemberian pakan.
c. Hindarkan pemberian pakan pada saat terik matahari, karena suhu tinggi dapat mengurangi nafsu makan lele.

2.2.2 Pengelolaan air

Pengelolaan air , baik kualitas maupun kuantitas, merupakan kegiatan yang sangat penting diperhatikan. Ikan akan hidup sehat dan tumbuh maksimal apabila kualitas airnya sesuai dengan kriteria untuk pertumbuhan ikan yang dipelihara.tujuan pengelolaan air untuk menyediakan lingkungan yang optimal bagi ikan agar tetap bisa hidup dan tumbuh maksimal.

Prinsip dari pengelolaan air adalah penggantian dengan air baru yang bbermanfaat(oksigen) dan membuang bahan yang tidak bermanfaat,seperti sisa-sisa pakan,kotoran ikan dan amonia.

Proses penggantian air dilakukan secara bertahap yaitu air dikeluarkan 1/3 bagian dan diisi dengan air baru. Air yang dikeluarkan adalah bagian dasar kolam dengan harapan timbunan kotoran(feses) dan sisa-sisa pakan yang membusuk di dasar kolam ikut terbuang. Penambahan air sangat penting, terutama pada musim kemarau, karena volume air berkurang akibat menguap. Selain itu, suhu air pada musim kemarau juga dipastikan meningkat. Akibatnya, ikan gampang stres dan nafsu makan turun.

Ukuran kualitas air dapat dinilai secara fisik :

a. Air harus bersih.
b. Berwarna hijau cerah.
c. Kecerahan / transparansi sedang (30-40 cm).

Ukuran kualitaa air secara kimia :

a. Bebas senyawa beracun seperti amoniak.
b. Mempunyai suhu optimal (22-26˚C).

2.2.3 Penyakit ikan lele

Penyakit yang menyerang ikan merupakan suatu proses hubungan antara 3 faktor yaitu lingkungan, ikan, jasad penyakit. Ikan yang terserang jasad penyakit merupakan hasil interaksi yang tidak serasi antara lingkungan, ikan, dan organisme penyebab penyakit , misalnya lingkungan yang tidak sesuai (perubahan suhu) menyebabkan ikan stres. Ikan menjadi lemah dan mudah terserang penyakit. Lele termasuk ikan yang mudah terserang penyakit karena tidak mempunyai sisik.

Jenis-jenis penyakit yang sering terjadi dalam budidaya ikan lele antara lain sebagai berikut [3] :

1. Penyakit karena bakteri *Aeromonas hydrophila* (perut besar)

   Banteri *Aeromonas hydrophila* umumnya hidup di air tawar yang mengandung bahan organik tinggi. Bakteri ini bersifat gram negatif, berbentuk batang, ukurannya 1-4 mikron x 0,4-1 mikron, dapat hidup dengan atau tanpa oksigen.

   a. Gejala

      Lele yang terkena bakteri ini: kehilangan nafsu makan, bengkak pada sirip, terjadi luka pada permukaan tubuh, perut membesar ,kondisi lele lemah.

   b. Pencegahan

      Pencegahan penyakit ini dapat dilakukan dengan mengontrol kualitas air, mencegah kelebihan pakan yang tidak dikonsumsi, membuang dan mengurangi kadar bahan organik dalam air (sisa-sisa pakan dan kotoran air), pertahankan suhu air pada 28° C , mengkarangtina ikan yang baru datang, serta menggunakan vaksin.

      Pencegahan dengan memberi antibiotik hanya efektif jika serang penyakit diketahui pada stadia dini. Jika lele sudah sakit dan nafsu makan menurun, pemberian antibiotik tidah efektif.

   c. Pengobatan

      1. Melalui makanan antara lain pakan pelet dicampur dengan oksitetrasiklin dosis 50 mg/kg pakan, diberikan selama 7-10 hari berturut-turut. Aplikasi oksitetrasiklin biasanya dilakukan pada stadia dini.
      2. Pada kolam pembesaran, Aeromonas dapat diatsai dengan pengaantian air setiap dua hari sekali. Selain itu , air kolam ditambahkan garam dapur dengan dosis 150-200 g/m^3 setiap penggantian air. Penggantian air kolam dilakukan hingga dipastikan ikan mulai membaik kembali.

2. Penyakit *Tuberculosis*

   a. Gejala

      Tubuh ikan berwarna gelap, perut bengkak (karena tubercle/bintil-bintil pada hati, ginjal, dan limpa). Posisi berdiri di permukaan air, berputar-putar atau miring-miring, bintik putih di sekitar mulut dan sirip.

   b. Pengendalian

      Memperbaiki kualitas air dan lingkungan kolam.

   c. Pengobatan

      Dengan Terramycin dicampur dengan makanan 5-7,5 gram/100 kg ikan/hari selama 5-15 hari.

3. Penyakit kuning(jaundice).

   a. Gejala

      Penyakit ini ditandai dengan perubahan warna tubuh ikan menjadi kuning dan di ikuti dengan kematian, selain itu pada tubuh ikan bagian luar, warna kuning juga terlihat pada organ dalam, seperti hati, ginjal, dan usus ikan saat ikan dibedah.

   b. Pengobatan

      Cara mengatasi penyakit jaundice adalah dengan menghindari pemberian pakan berupa ikan curah dan jeroan ayam secara penuh dan kontinu. Penggunaan pakan pelet yang masih baru dan berkualitas sangat dianjurkan. Selain itu, dilakukan penggantian air kolam yang terus diulang.

4. Penyakit bintik putih

   a. Gejala

      1. Adanya bintik-bintik putih pada permukaan tubuh dan insang.
      2. Ikan berwarna pucat..
      3. Ikan menggosok-gosokkan tubuh pada dasar atau dinding kolam.
      4. Pernafasan ikan tidak teratur (megap-megap)

b. Pencegahan

air harus dijaga kualitas dan kuantitasnya.

c. Pengobatan

Dengan cara perendaman ikan yang terkena infeksi pada campuran larutan formalin 25 cc/m3 dengan larutan Malachyte Green Oxalate 0,1 gram/m3 selama 12-24 jam.

5. Penyakit gatal

a. Gejala

1. Ikan lemah.
2. Warna tubuh kusam.
3. Ikan menggosok-gosokkan tubuh pada dasar atau dinding kolam.

b. Pencegahan

Pencegahan penyakit ini dapat dilakukan dengan cara perbaikan kualitas air, menjaga kebersihan sarana budidaya, serta mengatur tingkat padat penebaran.

c. Pengobatan

Ikan direndam dalam larutan formalin 40pmm selama 12-24 jam. Sementara itu perendaman ikan dengan Malachyte Green Oxalate 0,1 gram/m3 selama 12-24 jam.

6. Penyakit tubuh bengkok (kekurangan vitamin C)

a. Gejala

Tubuh bengkok dan tulang kepala lele yang retak-retak merupakan salah satu tanda ikan kekurangan vitamin C, mengalami pendarahan pada kulit.

b. Pengobatan

Penaggulagan penyakit ini adalah dengan cara menambahkan vitamin C dengan dosis 1g/kg pakan selama 5-7 hari. Satu set pakan ikan , yakni vitamin mix. Vitamin mix yang dapat digunakan diantaranya Aquamix, Premix, dan Rajamix.

7. Penyakit Virus Herpes

a. Gejala

Ikan yang terinfeksi menunjukkan gejala berenang berputar-putar , sering menggantung arah vertikal dipermukaan air, tampak lemah , hilang keseimbangan, pendarahan pada bagian sirip, mata menonjol dan kematian secara akut.

b. Pencegahan

Pencegahan penyakit ini bisa dilakukan dengan cara pengelolan managemen budidaya ikan yang benar(kualitas air, pakan, kepadatan dan penanganan), desinfeksi peralatan , pengeringan, pengapuran dasar kolam, pemberian pakan yang tepat mutu dan jumlah, serta ikan yang baru masuk harus dikarantinakan.

c. Pengobatan

Pengobbatan ikan yang terinfeksi belum diketahui. Namun, penyakit ini bisa diatasi dengan penggantian air setiap dua hari sekali sampai kesehatan ikan pulih.

2.3. Metode Bayes

Metode Bayes merupakan salah satu cara untuk mengatasi ketidakpastian data dengan cara menggunakan formula bayes,seperti pada rumus 2.1. [2].

$$p(H\ i|\ E) = \frac{p(E \mid Hi) * p(Hi)}{\sum p(E \mid Hi) * p(Hi)} \quad \ldots\ldots\ldots\ldots\ldots (2.1)$$

Dimana :

p(Hi | E) = probabilitas hipotesis Hi benar jika diberikakan *evidence* E.

p(E | Hi) = probabilitas muncul *evidence* E ,jika diketahui hipotesis Hi benar.

p(Hi) = probabilitas hipotesis Hi (menurut hasil sebelumnya) tanpa memandang *evidence* apapun.

p(E) = probabilitas *evidence* E.

Jika setelah dilakukan pengujian terhadap hipotesis kemudian muncul lebih dari satu *evidence* . Maka persamaannya akan menjadi , seperti pada rumus 2.2.

$$p(H|E,e) = p(H|E)\frac{p(e|E,H)}{p(e|E)} \quad \ldots\ldots\ldots\ldots\ldots(2.2)$$

Dimana :

e = *evidence* lama

E = *evidence* baru

P(H | E,e) = probabilitas hipotesis H benar jika diberikakan *evidence* baru E dari *evidence* lama.

P(H | E) = probabilitas hipotesis H benar jika diberikakan *evidence* E.

P(e |E, H) = kaitan antara e dan E jika hipotesis H benar.

P(e | E) = kaitan antara e dan E tanpa memandang hipotesis apapun.

2.4. Basis Data

Basis data terdiri atas 2 kata, yaitu Basis dan Data. Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang/berkumpul. Sedangkan data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, hewan, peristiwa, konsep, keadaan, dan sebagainya, yang direkam dalam bentuk angka, huruf, symbol, teks, gambar, bunyi, atau kombinasinya.

Basis data sendiri dapat didefinisiikan dalam sejumlah sudut pandang seperti [7]:

a. Himpunan kelompok data (arsip) yang saling berhubungan yang

diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.

b. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redudansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan.
c. Kumpulan file/tabel/arsip yang saling berhubungan yang disimpan dalam media penyimjpanan elektronis.

Basis data dan lemari arsip sesungguhnya memiliki prinsip kerja dan tujuan yang sama. Prinsip utamanya adalah pengaturan data/arsip. Dan tujuan utamanya adalah kemudahan dan kecepatan dalam pengambilan kembali data/arsip. Perbedaannya hanya terletak pada media penyimpanan yang digunakan. Jika lemari arsip menggunakan lemari dari besi atau kayu sebagai media penyimpanan, maka basis data menggunakan media penyimpanan elektronis seperti disk (disket atau hard disk). Hal ini merupakan konsekuensi yang logis, karena lemari arsip langsung dikelola/ditangani oleh manusia, sementara basis data dikelola/ditangani melalui perantara alat/mesin pintar elektronis (yang dikenal sebagai komputer). Perbedaan media ini yang selanjutnya melahirkan perbedaan media ini yang selanjutnya melahirkan perbedaan-perbedaan lain yang menyangkut jumlah dan jenis metoda/cara yang dapat digunakan dalam upaya penyimpanan.

2.4.1. Operasi Dasar Basis Data

Didalam sebuah disk, basis data dapat diciptakan dan dapat pula ditiadakan. Didalam sebuah disk, kita dapat pula menempatkan beberapa (lebih dari satu) basis data. Sementara dalam sebuah basis data, kita dapat menempatkan satu atau lebih file/tabel. Pada file/tabel inilah sesungguhnya data disimpan/ditempatkan. Setiap basis data umumnya dibuat untuk mewakili sebuah semesta data yang spesifik. Misalnya, ada basis data kepegawaian, basis data akademik, basis data inventori (Pergudangan), dan sebagainya. Sementara dalam basis data akademik, misalnya, kita dapat menempatkan file mahasiswa, file mata_kuliah, file dosen, file jadwal, file kehadiran, file nilai, dan seterusnya.

Operasi-operasi dasar yang dapat kita lakukan berkenaan dengan basis data dapat meliputi [7] :

a. Pembuatan basis data baru (*create database*), yang identik dengan pembuatan lemari arsip yang baru
b. Penghapusan basis data (*drop database*), yang identik dengan perusakan lemari arsip (sekaligus beserta isinya, jika ada).
c. Pembuatan file/tabel dari suatu basis data (*create table*), yang identik dengan penambahan map arsip baru ke sebuah lemari sarsip yang telah ada.
d. Penghapusan file/tabel dari suatu basis data (*drop table*), yang identik dengan perusakan map arsip lama yang ada di sebuah lemari arsip.
e. Penambahan/pengisian data baru ke sebuah file/tabel disebuah basis data (*insert*), yang identik dengna penambahan ke lemari arsip ke sebuah map arsip.
f. Pengambailan data dari sebuah file/tabel (*retrieve/search*) yang identik dengan pencarian lembaran arsip dari map arsip.
g. Pengubahan data dari sebuah file/tabel (*update*), yang identik dengan perbaikan isi lembaran arsip yang ada di sebuah map arsip.
h. Penghapusan data dari sebuah file/tabel (*delete*), yang identik dengan penghapusan sebuah lembaran arsip yang ada di sebuah map arsip.

Operasi yang berkenaan dengan pembuatan objek (basis data dan tabel) merupakan operasi awal yang hanya dilakukan sekali dan berlaku seterusnya. Sedang operasi-operasi yang berkaitan dengan isi tabel (data) merupakan operasi rutin yang akan berlangsung berulang-ulang dank arena itu operasi-operasi inilah yang lebih tepat mewakili aktivitas pengelolaan (*management*) dan pengolahan (*processing*) data dalam basis data

2.4.2. Objektif Basis Data

Telah disebutkan di awal bahwa tujuan awal dan utama dalam pengelolaan data dalam sebuah basis data adalah agar kita dapat memperoleh menemukan kembali data (yang dicari) dengan mudah dan cepat. Disamping itu, pemanfaatan basis data untuk pengelolaan data, juga memiliki tujuan-tujuan lain.

Secara lebih lengkap, pemanfaatan basis data dilakukan untuk memenuhi

sejumlah tujuan (objektif) seperti berikut ini [7]:

a. Kecepatan dan kemudahan

Pemanfaatan basis data memungkinkan untuk dapat menyimpan data atau melakukan perubahan/manipulasi terhadap data atau menampilkan kembali data tersebut dengan lebih cepat dan mudah, daripada jika menyimpan data secara manual (non-elektronis) atau secara elektronis (tetapi tidak dalam bentuk penerapan basis data, misalnya dalam bentuk *spread sheet* atau dokumen teks biasa).

b. Efisiensi ruang penyimpanan (*Space*)

Karena keterkaitan yang erat antara kelompok data dalam sebuah basis data, maka redudansi (pengulangan) data pasti akan selalu ada. Banyaknya redudansi ini tentu akan memperbesar ruangan penyimpanan (baik di memori utama maupun memori sekunder) yang harus disediakan.

Dengan basis data, efisiensi/optimalisasi penggunaan ruang penyimpanan dapat dilakukan, karena kita dapat melakukan penekanan jumlah redudansi data, baik dengan menerapkan sejumlah pengkodean atau dengan membuat relasi-relasi (dalam bentuk file) antara kelompok data yang saling berhubungan.

c. Keakuratan (*Accurancy*)

Pemanfaatan pengkodean atau pembentukan relasi antar data bersama dengan penerapan aturan/batasan (*constrain*) tipe data, domain data, keunikan data, dan sebagainya, yang secara ketat diterapkan dalam sebuah basis data, sangat berguna untuk menekan ketidakakuratan pemasukan/penyimpanan data.

d. Ketersediaan (*Availability*)

Pertumbuhan data (baik dari sisi jumlah maupun jenisnya) sejalan dengan waktu akan semakin membutuhkan ruang penyimpanan yang besar. Padahal tidak semua data itu selalu kita gunakan/butuhkan. Karena itu kita dapat memilah adanya data utama/*master*/refereensi, data transaksi, data histori hingga data kadaluarsa. Data yang sudah jarang atau bahkan tidak pernah lagi kita gunakan, dapat kita atur untuk dilepaskan dari sistem basis data yang sedang aktif (menjadi *off-line*) baik dengan cara penghapusan atau dengan memindahkannya ke media penyimpanan *off-line* (seperti *removable disk*, atau *tape*). Di sisi lain, karena

kepentingan pemakaian data, sebuah geografis. Data nasabah sebuah bank, misalnya, dipisah-pisahkan dan disimpan di lokasi yang sesuai dengan keberadaan nasabah. Dengan pemanfaatan teknologi jaringan komputer, data yang berada di suatu lokasi/cabang, dapat juga di akses (menjadi tersedia/*available*) bagi lokasi/cabang lain.

e. Kelengkapan (*Completeness*)

Lengkap/tidaknya data yang kita kelola dalam sebuah basis data bersifat relative (baik terhadap kebutuhan pemakai maupun terhadap waktu). Bila seorang pemakai sudah menganggap bahwa data yang dipelihara sudah lengkap, maka pemakai yang lain belum tentu berpendapat sama. Atau, yang sekarang diangap sudah lengkap belum tentu di masa yang akan data juga demikian. Dalam sebuah basis data, disamping data kita juga harus menyimpan struktur (baik yang mendefinisikan objek-objek dalam basis data maupun definisi detail dari tiap objek seperti struktur file/tabel atau indeks). Untuk mengakomodasi kebutuhan kelengkapan data yang semakin berkembang, maka kita tidak hanya dapat menambahkan *record-record* data, tetapi juga dapat melakukan perubahan struktur dalam basis data baik dalam bentuk penambahan objek baru (tabel) atau dengan penambahan field-field baru pada suatu tabel.

f. Keamanan (*Security*)

Memang ada sejumlah sistem (aplikasi) pengelola basis data yang tidak menerapkan aspek keamanan dalam penggunaan basis data. Tetapi untuk sistem yang besar dan serius, aspek keamanan juga dapat diterapkan dengan ketat. Dengan begitu kita dapat menentukan siapa-siapa (pemakai) yang boleh menggunakan basis data beserta objek-objek di dalamnya dan menentukan jenis-jenis operasi apa saja yang boleh dilakukannya.

g. Kebersamaan Pemakai (*Sharebility*)

Pemakai basis data seringkali tidak terbatas pada satu pemakai saja, atau di satu lokasi saja atau boleh satu sistem/aplikasi saja, data pegawai dalam basis data kepegawaian, misalnya, dapat digunakan oleh banyak pemakai, dari sejumlah departemen dalam perusahaan atau oleh banyak ssitem (sistem penggajian, sistem akuntansi, sistem inventori, dan sebagainya). Basis data yang dikelola oleh sistem

(aplikasi) yang mendukung lingkungan *multiuser*, akan dapat memenuhi kebutuhan ini, tetapi tetap dengan menjaga/ menghindari terhadap munculnya persoalan baru seperti inkonsistensi data (karena data yang sama diubah oleh banyak pemakai pada saaat yan gbersamaan) atau kondisi *deadlock* (karena banyak pemakai yang saling menunggu untuk menggunakan data).

2.5. Java

2.5.1. Pengertian Java

Java adalah bahasa pemrograman yang disusun oleh James Gosling yang dibantu oleh rekan-rekannya seperti Patrick Naugton, Chris Warth, Ed Frank, dan Mike Sheridan di suatu perusahaan perangkat lunak yang bernama Sun Microsystem, pada tahun 1991. Bahasa pemrograman ini mula-mula diinisialisasi dengan nama "oak", namun pada tahun 1995 diganti namanya menjadi "Java". Alasan utama pembentukan bahasa java adalah untuk membuat aplikasi- aplikasi yang dapat diletakkan diberbagai macam perangkat elektronik, seperti *microwave oven* dan *remote control*, sehingga Java harus bersifat *portable* atau yang sering disebut dengan *platform independent* (tidak tergantung pada *platform*). Itulah yang menyebabkan dalam dunia pemrograman Java, dikenal adanya istilah '*write once, run everywhere*', yang berarti kode program hanya ditulis sekali, namun dapat dijalankan dibawah *platform* manapun, tanpa melakukan perubahan kode program.

2.5.1.1. Arsitektur Java

Secara arsitektur, Java tidak berubah sedikitpun semenjak awal mula bahasatersebut dirilis. Kompiler Java (yang disebut dengan **Javac** atau *Java Compiler*) akanmentransformasikan kode-kode dalam bahsa Java ke dalam suatu *bytecode*. Apa itu*bytecode*? *Bytecode*adalah sekumpulan perintah hasil kompilasi yang kemudiandapat dieksekusi melalui sebuah mesin komputer abstrak, yang disebut dengan JVM(*Java Virtual Machine*). JVM juga sering dinamakan sebagai *interpreter*, kiarena sifatnya yang selalu menerjemahkan kode-kode yang tersimpan dalam *bytecode* dengan cara baris demi baris.

2.5.1.2. Java Versi Lama (Java1)

Pada awal perilisannya, versi Java masih disebut dengan JDK (*Java Development kita*). Dalam JDK, semua kebutuhan untuk pengembangan program dan eksekusi program masih tergabung jadi satu. Penamaan ini berlaku sampai Java 1.1. namun sekarang, setelah Java 1.2, Sun Microsystem menamainya dengan JSDK (*Java Software Development Kit*) dalam hal ini kebutuhan untuk pengembangan program dipisahkan dengan kebutuhan kesekusi. Bagian *software* yang digunakan untuk kebutuhan eksekusi program disebut dengan JRE (*Java-Runtime Environment*). Selanjutnya, Java 1.2 disederhanakan penamaanya menjadi "Java 2".

2.5.1.3. Java 2

Sun Microsystems telah mendefinisikan tiga buah edisi dari Java 2, yaitu sebagai berikut :

a. Java 2 Standard Edition (J2SE), yang digunakan untuk mengembangkan aplikasi- aplikasi desktop dan applet (aplikasi Java yang dapat dijalankan di dalam *browser web*).
b. Java 2 Enterprise Edition (J2EE), merupakan *superset* dari J2SE yang memperbolehkan untuk mengembangkan palikasi-aplikasi berskala besar (*enterprise*), yaitu dengan melakukan pembuatan aplikasi-aplikasi di sisi *server* dengan menggunakan EJBs (*Enterprise JavaBeans*), aplikasi web dengan menggunakan *Servlet* dan JSP (*JavaServer Pages*) dan teknologi lainnya serperti CORBA (*Common Object Request Broker Architecture*) dan XML (*Extensible Markup Language*).
c. Java 2 Micro Edition (J2ME), merupakan *subset* dari J2SE yang digunakan untuk menangani pemrograman di dalam perangkat-perangkat kecil, yang tidak memungkinkan untuk mendukung implementasi J2SE secara penuh.

2.6. Android

Pembahasan mengenai android dan *tools* yang digunakan dalam pembuatan aplikasi, akan dijelaskan pada sub bab berikut: [5]

2.6.1. Sejarah Android

Android adalah sebuah sistem operasi untuk perangkat lunak *mobile*

berbasis linux yang mencakup sistem operasi, middleware dan aplikasi. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, Google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel/*smartphone*. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, Htc, Intel, Motorola, Qualcomm, T-*Mobile*, dan Nvidia.

Pada saat perilisan perdana Android, 5 November 2007, Android bersama Open Handset Alliance menyatakan mendukung pengembangan *open source* pada perangkat *mobile*. Di lain pihak, Google merilis kode-kode android dibawah lisensi *Apache*, sebuah lisensi perangkat lunak dan *platform* perangkat seluler.

Sekitar September 2007 Google mengenalkan Nexus One, salah satu jenis *smartphone* yang menggunakan android sebagai sistem operasinya. Telpon seluler ini diproduksi oleh HTC Corporation dan tersedia di pasaran pada 5 Januari 2010. Pada 9 Desember 2008, diumumkan anggota baru yang bergabung dalam program kerja Android AEM Holdings, Atheros Communication, diproduksi oleh Asustek Komputer Inc, Garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp, dan Vodafone Group Plc. Seiring pembentukan Open Handset Alliance, OHA mengumumkan produk perdana mereka, android perangkat *mobile* yang merupakan modifikasi kernel Linux 2.6. sejak Android dirilis telah dilakukan berbagai pembaruan berupa perbaikan bug dan penambahan fitur baru.

Pada masa saat ini kebanyakan vendor-vendor *smartphone* sudah memproduksi *smartphone* berbasis android, vendor-vendor itu antara lain HTC, Motorola, Samsung, LG, HKC, Huawei, Archos, Webstation Camangi, Dell, Nexus, SciPhone, WayteQ, Sony Ericsson, LG, Acer, Philips, T-*Mobile*, Nexian, IMO, Asus dan masih banyak lagi vendeor *smartphone* didunia yang memproduksi android. Hal ini karena android adalah sistem oerasi yang *open source* sehingga bebas didistribusikan dan dipakai oleh vendor manapun.

2.6.2. *The Dalvik Virtual Machine* (DVM)

Salah satu elemen kunci dari android adalah *Dalvik Virtual Machine* (DVM). Android berjalan di *Dalvik Virtual Machine* (DVM) bukan di Java

Virtual Machine (JVM), sebernarnya banyak persamaan dengan Java *Virtual Machine* (JVM) seperti Java ME (*Java Mobile Edition*), tetapi android menggunakan *Virtual Machine* sendiri yang diskostumisasi dan dirancang untuk memastikan bahwa beberapa fitur-fitur berjalan lebih efisien pada perangkat *mobile*.

Dalvik Virtual Machine (DVM) adalah "*register bases*" sementara java *Virtual Machine* (JVM) bersifat "*stack based*", DVM didesain dan ditulis oleh Dan Bornsten dan beberapa enginers Google lainnya. Jadi bisa dikatakan "*Dalvik* equal(Java) == False"*Dalvik Virtual Machine* menggunakan kernel linux untuk menangani fungsionalitas tingkat rendah termasuk keamanan, *threading*, dan proses serta manajemen memori. Ini memungkinkan kita untuk menulis aplikasi C/C++ sama halnya seperti pada OS Linux kebanyakan. Meskipun dalam kenyataannya kita harus banyak memahami Arsitektur dan proses sistem dari kernel linux yang digunakan dalam Android tersebut.

Semua *hardware* yang berbasis android dijalankan dengan menggunakan *Virtual Machine* untuk eksekusi aplikasi, pengembang tidak perlu khawatir tentang implementasi perangkat keras tertentu. *Dalvik Virtual Machine* mengeksekusi *Executable* file, sebuah format yang dioptimalkan untuk memastikan memori yang digunakan sangat kecil. *The Executable* file diciptakan dengan mengubah kelas bahasa java dan dikompilasi menggunakan *tools* yang disediakan dalam SDK Android.

2.6.3. Arsitektur Android

Secara garis besar arsitektur android dapat dijelas dan digambarkan sebagai berikut [5] :

a. *Application and Widgets*

Application and widgets adalah layer dimana berhubungan dengan aplikasi dan biasanya download aplikasi kemudian lakukan instalasi dan jalankan aplikasi tersebut, delayer inilah terdapat seperti aplikasi inti termasuk klien email, program SMS, kalender, peta, *browser*, kontak, dan lain-lain. Semua aplikasi ditulis menggunakan bahasa pemrograman Java.

b. *Application Frameworks*

Android adalah "*Open Development Platform*" yaitu android menawarkan kepada pengembang atau member kemampuan kepada pengembangan untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resources*, menjalankan serive *background*, mengatur alarm, dan menambahkan tambahan seperti status *notification*s dan masih banyak lagi. Pengembang memiliki akses penuh menuju API *Framework* seperti yang dilakukan oleh aplikasi yang kategori inti. Arsitektur aplikasi dirancang supaya dengan mudah dapat menggunakan komponen yang sudah digunakan (*reuse*).

Komponen-komponen yang termasuk didalam *application Framework* adalah sebagai berikut :

1. *View*
2. *Content Provider*
3. *Resourse Manager*
4. *Notifikasi Manager*
5. *Activity Manager*

c. *Libraries*

Libraries adalah layer dimana fitur-fitur android berada biasanya para pembuat aplikasi kebanyakan mengakses *library* untuk menjalankan aplikasinya berjalan diatas kernel, layer ini meliputi berbagai library C/C++ inti seperti Libc dan SSL, serta :

1) *Libraries* media untuk pemutar media audio dan video.
2) *Libraries* untuk manajemen tampilan.
3) *Libraries Graphics* mencakup SGL dan OpenGL untuk grafis 2D dan 3D.
4) *Libraries SQLite* untuk dukungan database.
5) *Libraries* SSL dan *WebKit* terintegrasi dengan web *browser* dan security.
6) *Libraries Live Webcore* mencakup modern web *browser* dengan engine embedded web view.

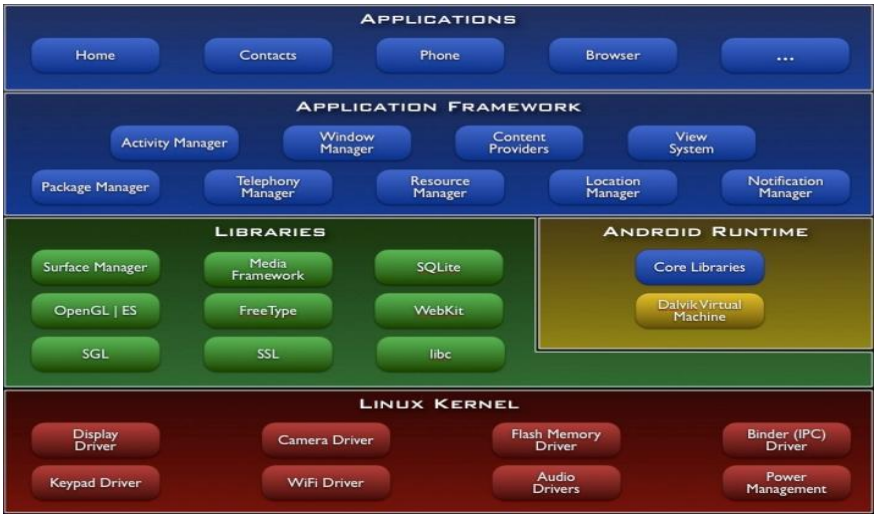
d. *Android Run Time*

Layer yang membuat aplikasi android dapat dijalankan dimana dalam prosesnya menggunakan implementasi Linux. *Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Didalam *Android Run Time* dibagi menjadi dua bagian yaitu:

1. *Core Libraries* : aplikasi android dibangun dalam bahasa java, sementara *Dalvik* sebagai virtual mesin bukan *Java Virtual Machine*, sehingga diperlukan sebuah *libraries* yang berfungsi untuk menterjemahkan bahasa Java/C yang dihandle oleh Core *Libraries.*
2. *Dalvik Virtual Machine* : Virtual mesin yang berbasis *register* yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien dimana merupakan pengembangan yang mampu membuat linux kernel untuk *threading* dan manajemen tingkat rendah.

e. Linux kernel

Linux kernel adalah layer dimana inti dari operating sistem dari android itu sendiri, berisi file-file sistem yang mengatur sistem *processing*,*memory*, *resources*, *drivers*, dan sistem-sistem operanting andoroid lainnya.



Gambar 2.6 Arsitektur Android [5]

2.6.4. Fundamental Aplikasi

Aplikasi android ditulis dalam bahasa pemrograman java, kode java dikompilasi bersama data file *resource* yang dibutuhkan oleh aplikasi dimana

prosesnya dipaket oleh *tools* yang dinamakan "apt *tools*" ke dalam paket android sehingga menghasilkan file dengan ekstensi apk. File apk itulah yang sebenernya disebut dengan aplikasi yang dapat diinstal di perangkat *mobile* nantinya.

Jenis komponen pada aplikasi android yaitu [5]:

a. Activities

Suatu *activity* akan menyajikan *user interface* (UI) kepada pengguna, sehingga pengguna dapat melakukan interaksi. Sebuah aplikasi android bisa jadi hanya memiliki satu *activity*, tetapi umumnya aplikasi memiliki banyak *activity* tergantung pada tujuan aplikasi dan desain dari aplikasi tersebut. Satu *activity* biasanya akan dipakai untuk menampilkan aplikasi atau yang bertindak sebagai *user interface* (UI) saat aplikasi diperlihatkan kepada *user*. Untuk pindah dari satu *activity*ke *activity* lain dapat dilakukan dengan satu even misalnya click tombol, memilih opesi atau menggunakan triggers tertentu. Secara hirarki sebuah window *activity* dinyatakan dengan method *Activity*.setContentView(). ContentView adalah objek yang berada pada root hirarki.

b. Service

Service tidak memiliki visual *user interface* (UI), tetapi *service* berjalan secara *background*, sebagai contoh dalam memainkan music, tetapi setiap *service* haruslah berada dalam kelas induknya. Misalnya media player sedang memutar lagu dari *list* yang ada. Aplikasi ini akan memiliki dua atau lebih *activity* yang memungkinkan *user* untuk memilih lagu atau menulis sms sambil *player* sedang jalan untuk menjaga *music* tetapi dijalankan, *activity player* dapat menjalankan *service* untuk membuat aplikasi tetap berjalan. *Service* dijalankan pada *thread* utama dari proses aplikasi.

c. Broadcast Receiver

Broadcast Receiver berfungsi menerima dan bereaksi untuk menyampaikan notifikasi. Contoh *broadcast* seperti notifikasi zona waktu berubah, baterai *low*, gambar telah selesai diambil oleh kamera, atau pengubah referensi bahasa yang digunakan. Aplikasi juga dapat menginisiasi *broadcast* misalnya memberikan informasi pada aplikasi lain bahwa ada data yang telah didownload ke perangkat dan siap untuk digunakan. *Broadcast Receiver* tidak memiliki *user interface* (UI)

tetapi memiliki sebuah *activity* untuk merespon informasil yang mereka terima atau mungkin menggunakan *notification* manager untuk memberitahu kepada pengguna seperti lampu latar atau *viberating* (getaran) perangkat dan sebagainya.

d. Content Provider

Content Provider membuat kumpulan aplikasi data secara spesifik sehingga bisa digunakan oleh aplikasi lain. Data disimpan dalam file sistem seperti database SQLite.*Content Provider* menyediakan cara untuk mengakses data yang dibutuhkan oleh suatu *activity*, misalnya ketika menggunakan aplikasi yang membutuhkan peta (MAP) atau aplikasi yang membutuhkan untuk mengkases data kontak dan navigasi, maka disinilah fungsi *content provider*.

2.6.5. Versi Android

Telepon pertama yang memakai sistem operasi android adalah HTC Dream yang dirilis pada 22 Oktober 2008. Pada penghujung 2010 diperkirakan hamper semua vendor seluler didunia menggunakan android sebagai operating sistem. Adapun versi-versi android yang pernah dirilis adalah sebagai berikut [5] :

a. Android Versi 1.1

Tepat pada 9 Maret 2009 Google telah resmi merilis versi perdana Android yaitu versi 1.1 versi ini dilengkapi dengan pembaharuan yang estetis pada aplikasi terkait seperti aplikasi, jam alarm, voice search atau pencarian suara, pengiriman pesan dengan Gmail, dan pemberitahuan email.

b. Android Versi 1.5 (Cupcake)

Sekitar pertengahan Mei 2009, Google kembali hadir dengan merilis versi terbaru yakni versi 1.5 (Cupcake) untuk telepon seluler yang menggunakan Androit dan SDK (Software Development Kit). Penyesuaian sistem pada layar terjadi pada versi ini mengingat terdapat penambahan beberapa fitur yakni kemampuan merekam dan menonton video dengan modus kamera, mengunggah video ke Youtube dan gambar ke Picasa via telepon, dukungan Bluetooth A2DP, kemampuan koneksi secara otomatis ke headset Bluetooth, animasi layar, dan keyboard semua dapat dilakukan jauh lebih efektif.

c. Android Versi 1.6 (Donut)

Pada September diliris kembali dengan versi 1.6 (Donut) yang menampilkan

proses pencarian lebih baik dibanding pada versi sebelumnya, penggunaan baterai indikator dan kontrol applet VPN. Fitur lain yang terdapat pada versi ini adalah fungsi fitur untuk galeri yang memungkinkan pengguna untuk memilih foto yang hendak di hapus; kamera, camcorder dan galeri yang dintegrasikan; CDMA / EVDO, 802.1x, VPN, Gestures, dan Text-to-speech engine; kemampuan dial kontak; teknologi text to change speech.

d. Android versi 2.0/2.1 (Eclair)

3 Desember 2009 kembali di luncurkan Android dengan versi yang lebih unggul di lihat dari pengembangan yang dilakukan melalui optimalisasi hardware; peningkatan Google Maps 3.1.2, perubahan UI dengan browser baru dan dukungan HTML5, daftar kontak yang baru, dukungan flash untuk kamera 3,2 MP, digital Zoom, dan Bluetooth 2.1.

e. Android Versi 2.2 (Froyo: Frozen Yoghurt)

Pada bulan Mei 2010 android 2.2 Rev 1 diluncurkan. Android inilah yang sangat banyak sekarang dipasaran, salah satunya adalah dipakai di Samsung FX tab yang sudah ada dipasaran. Fitur yang tersedia di android versi ini sudah kompleks sekali diantaranya adalah :

1. Kerangka aplikasi memungkinkan penggunaan dan penghapusan komponen yang tersedia.
2. *Dalvik Virtual Machine* dioptimalkan untuk perangkat *mobile*.
3. Grafik: grafik di 2D dan 3D berdasarkan *libraries* OpenGL.
4. SQLite: untuk penyimpanan data.
5. Mendukung media: audio, video, dan berbagai format gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
6. GSM, *Bluetooth*, EDGE, 3G, dan WiFi (tergantung *Hardware*).
7. Kamera, *Global Positioning System* (GPS),kompas,dan ccelerometer (tergantung *Hardware*)

f. Android Versi 2.3 (Gingerbread)

Android Versi 2.2 kembali diluncurkan pada 6 Desember 2010 perubahan-perubahan umum kembali di hadiahkan untuk para pengguna seperti permainan atau game.

g. Android Versi 3.0 (Honeycomb)

Android Versi 3.0 di luncurkan kusus untuk mengoptimalkan gadget dengan layar lebar dan tablet. Dalam versi ini juga diperkenalkan desain UI baru, virtual dan holografis yang elegan dengan model interaksi fokus terhadap konten. Selain itu Google membuat multitasking, notifications, Home screen customization, widgets, lebih interaktif, vibrant, dan memberikan pengalaman 3D tetap familiar dan lebih baik dari sebelumnya.

h. Android Versi 4.0 (Ice Cream Sandwich)

Versi Android 4.0 di desain dapat digunakan baik itu untuk telepon ataupun tablet. Versi ICS hadir banyak menawarkan peningkatan yang telah dirilis pada versi sebelumnya Gingerbread dan Honeycomb yang akan menghasilkan sebuah inovasi baru . Peningkatan ICS dalam kemampuan dalam hal Copy paste jauh lebih produktif, data blogging dan warnings melengkapi ditambah dengan adanya fungsi screenshot dengan cara menekan tombol power dan volume secara bersamaan.

2.7. *Tools* yang Digunakan

2.7.1. Konsep Perancangan Berorientasi Obyek

Teknologi objek menganalogikan sistem aplikasi seperti kehidupan nyata yang didominasi oleh objek. Didalam membangun sistem berorientasi objek akan menjadi lebih baik apabila langkah awalnya didahului dengan proses analisis dan perancangan yang berorientasi obke. Tujuannya adalah mempermudah programmer didalam mendesain program dalam bentuk objek-objek dan hubungan antar objek tersebut untuk kemudian dimodelkan dalam sistem nyata. Suatu perusahaan software yaitu Rational Software, telah membentuk konsarium dengan berbagai organisasi untuk meresmikan pemakaian *Unified modelling Language* (UML) sebagai bahasa standar dalam *Object Oriented Analysist Design* (OOAD).

2.7.2. UML (*Unified Modelling Language*)

UML (*Unified Modelling Language*) adalah salah satu alat bantu yang

sangat handal di dunia pengembangan sistem yang berorientasi obyek. Hal ini disebabkan UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain.

UML merupakan kesatuan dari bahasa pemodelan yang dikembangkan oleh Booch, *Object Modeling Technique* (OMT) dan *Object Oriented Software Engineering* (OOSE). Metode Booch dari Grady Booch sangat terkenal dengan nama metode *Design Object Oriented*. Metode ini menjadikan proses analisis dan *design* ke dalam empat tahapan iterative, yaitu: identifikasi kelas-kelas dan obyek-obyek, identifikasi semantic dari hubungan obyek dan kelas tersebut, perincian *interface* dan implementasi. Keunggulan metode Booch adalah pada detil dan kayanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstruktur pemodelan entity-relationship. Tahapan utama dalam metodologi ini adalah analisis, *design* sistem, *design* obyek dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung semua konsep OOP. Metode OOSE dari Jacobson lebih member penekanan pada use case. OOSE memiliki tiga tahapan yaitu membuat model requirement dan analisis, *design* dan implementasi, dan model pengujian (*test model*). Keunggulan metode ini adalah mudah dipelajari karena memiliki notasi yang sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak.

Design UML, metode Booch, OMT dan OOSE digabungkan dengan membuang elemen-elemen yang tidak praktis ditambah dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam dari pada metode lainnya.

2.7.2.1. Use Case Diagram

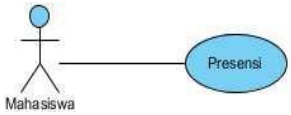
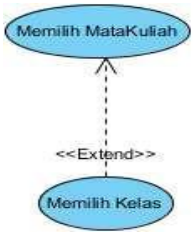
Use Case adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use Case* bekerja dengan cara mendeskripsikan tipikal interaksi antara

user (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut *scenario*. Setiap scenario mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem yang lain, perangkat keras atau urutan waktu. Dengan demikian secara singkat bisa dikatakan *use case* adalah serangkaian scenario yang digabungkan bersama-sama oleh tujuan umum pengguna.

Dalam pembicaraan tentang *use case*, pengguna biasanya disebut dengan aktor. Aktor adalah sebuah peran yang bisa dimainkan oleh pengguna dalam interaksinya dengan system.

Simbol – simbol Use Case Diagram,sebagai berikut :

Tabel 2.1. Use Case Diagram

Use Case 	Aktor 	Assosiasi 
Generalisasi 	Include 	Extend 

Keterangan :

1. Use case : Sebuah use case merupakan unit fungsi yang dapat berinteraksi dengan aktor atau terkait dengan use case lainnya. Penggunaannya diwakili dengan sebuah elips dengan nama use case idalamnya. Nama use case biasanya berupa suatu kata kerja.
2. Aktor : Aktor mewakili entitas eksternal dalam sistem dan dapat berupa manusia, perangkat keras atau sistem lain. Aktor diambil menggunakan sebuah figure tongkat. Hubungan yang terjadi antar aktor biasanya adalah

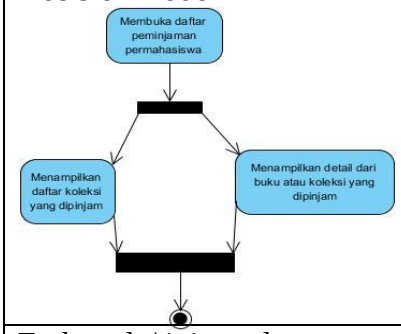
generalisasi untuk menunjukkan perilaku terhadap sistem yang lebih spesifik.

3. Assosiasi : Ditunjukkan dalam diagram use case oleh garis padat. Sebuah asosiasi setiap kali ada seorang aktor yang terlibat dengan interaksi yang dijelaskan oleh use case. Asosiasi dimodelkan sebagai garis yang menghubung kan use case dan aktor untuk satu sama lain, dengan mata panah yang bersifat psional(tidak harus ada) pada salah satu ujung baris. panah ini sering digunakan untuk menunjukkan arah seruan awal hubungan atau untuk menunjuk kan aktor utama dalam usecase.
4. Generalisasi : Digunakan ketika ada use case yang umum atau actor yang umun, digunakan untuk memberikan dasar fungsi yang dapat digunakan oleh use case/actor yang lebih khusus.
5. Include : mengincludekan use case yang lain, yang berefek use case yang diincludkan akan terisi kedalam perilaku dari use case utama.
6. Extend : Menggambarkan bahwa use case tertentu menyediakan fungsionalitas tambahan kepada usecase induknya pada suatu aliran tertentu. Disini dapat diartikan bahwa use case yang mengextend tidak harus selalu tereksekusi ketika use case induknya dieksekusi.

2.7.2.2. Actifity Diagram

Activity diagram seperti sebuah *flow chart*. *Activity* diagram menunjukkan tahapan, pengambilan keputusan dan percabangan. Diagram ini sangat berguna untuk menunjukkan *operation* sebuah obyek dan proses bisnis. Kelebihan *activity* diagram dibandingkan *flowchart* adalah kemampuannya dalam menampilkan aktivitas parallel.

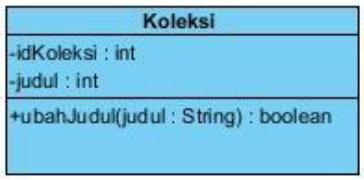
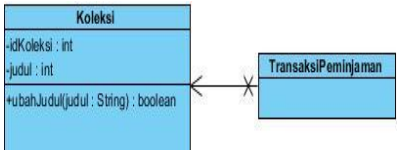
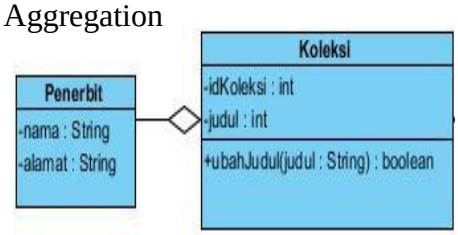
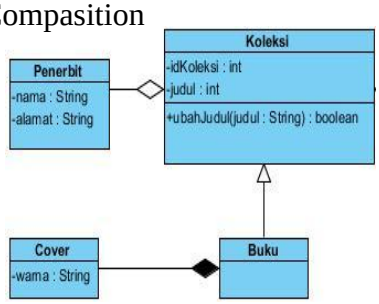
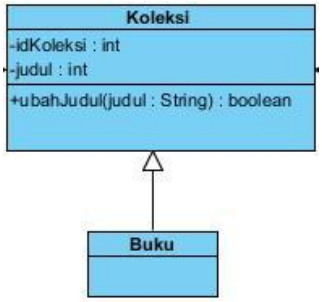
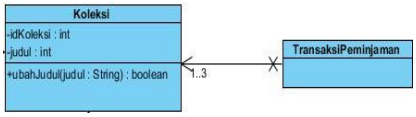
Tabel 2.2. Actifity Diagram

Action 	Menggambarkan 1 langkah dalam alur system.
Start Note 	Digunakan untuk menggambarkan dimana aliran berawal
Activity final node 	Akhir dari banyak aliran dalam diagram aktivitas
Flow final node 	Menunjukkan akhir dari aliran tunggal dalam diagram aktivitas
Control flow 	Menunjukkan aliran kendali dari satu aksi ke aksi yang berikutnya lain
Decision Node 	Digunakan untuk mempresentasikan keputusan dalam alur kendali
Fork node/ join node 	Fork Digunakan untuk memecah alur yang terjadi secara bersamaan (pararel) Join Digunakan untuk menggabungkan alur yang terjadi secara bersamaan (pararel)

2.7.2.3. Class Diagram

Class diagram mendeskripsikan struktur statis dari kelas – kelas dalam system dan mengilustrasikan attribute, operations dan relationship antara satu kelas dengan kelas yang lain.

Tabel 2.3Class Diagram

Class 	Asscociation 
Aggregation 	Compasition 
Generalization 	Multipricity 

Ket :

Class : Rancangan dari objek yang mendefinisikan attribute dan method umum pada semua objek dari jenis tertentu Digambarkan dengan 3 tingkat, yaitu (dari atas ke bawah)

a. Name

Nama dari kelas, aturan penulisan huruf pertama dari semua kata adalah huruf besar, selain dari itu huruf kecil

b. Attributes

Mengambarkan attribute yang dimiliki oleh kelas, aturan penamaan huruf pertama dari kata kedua dan kata selanjutnya adalah huruf besar, selain dari itu huruf kecil

Format

modifier name : type = value

c. Operations

Menggambarkan method yang dimiliki oleh kelas, aturan penamaan sama dengan penamaan attribute.

Format

modifier name(parameter:type):return type .

Asscociation : Relasi yang kuat, relasi ini mengindikasikan bahwa suatu kelas mereferensikan kelas yang lain, menggambarkan interaksi yang mungkin terjadi antara 1 kelas dengan kelas yang lain selama kelas tersebut tidak saling memiliki atau bukan bagian dari.

a. Directional Association (1 arah)

Menggambarkan bahwa pesan terjadi hanya dari satu kelas sedangkan kelas yang lain pasif.

b. Bidirectional Association (2 arah)

Menggambarkan bahwa pesan dari kedua kelas yang saling berelasi.

Aggregation : Lebih spesifik dari assosiasi, mengindikasikan bahwa suatu kelas merupakan bagian dari kelasyang lain namun bersifat tidak wajib. Relasi ini juga menyatakan bahwa suatu kelas yang menjadi bagian dari kelas yang lain tidak akan dihapus meskipus kelas yang memilikinya dihapus.

Compasition : Lebih spesifik daripada aggregation, mengindikasikan hubungan yang saling bergantung, dimana suatu kelas merupakan bagian dari kelas yang lain dan bersifat wajib.

Relasi ini juga mengindikasikan bahwa suatu kelas yang menjadi bagian kelas yang lain akan terhapus ketika kelas yang memilikinya dihapus.

Generalization : Juga dikenal sebagai inheritance.Mengindikasikan bahwa sebuah kelas adalah child class atau sub class lebih spesifik terhadap parent class / super kelasnya. Semua sub class akan memiliki apa yang dimiliki super class tapi super class tidak memiliki apa yang hanya dimiliki oleh sub class.

Multipricity : Mengindikasikan berapa banyak objek dari suatu kelas terelasi ke objek yang lain

a. * = banyak

b. 0 = nol

c. 1 = satu

d. 0..* = nol sampai banyak

e. 1..* = satu sampai banyak

0..1 = nol atau satu

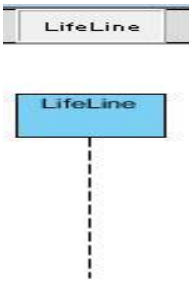
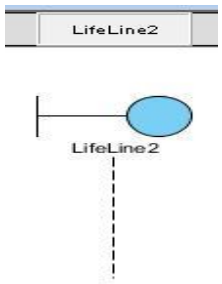
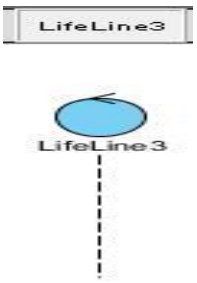
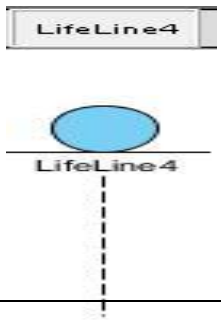
2.7.2.4. Sequence Diagram

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah skenario. Diagram ini menunjukkan sejumlah contoh obyek dan *message* yang diletakkan diantara obyek-obyek ini di dalam *use case*. *Sequence* diagram menambahkan dimensi waktu pada interaksi diantara obyek. Pada diagram ini *participant* diletakkan di atas dan waktu ditunjukkan dari atas ke bawah. *Life line participant* diurutkan dari setiap *participant*. Kotak kecil pada *lifelin*e menyatakan *activation*, yaitu menjalankan salah satu operation dari *participant*. State bias ditambahkan dengan menambahkannya sepanjang life line.

Message (sederhana, *synchronou*s atau *asynchronous*) adalah tanda panah yang menghubungkan suatu *life line* ke *life line* yang lain. Lokasi *life line* dalam dimensi vertikal mewakili urutan waktu dalam *sequence* diagram. *Message* yang pertama terjadi adalah yang paling dekat dengan bagian atas diagram dan yang terjadi belakangan adalah yang dekat dengan bagian bawah.

Pada beberapa sistem, operasi bisa dilakukan kepada dirinya sendiri. Hal ini disebut dengan rekursif. Untuk melukiskannya digunakan anak panah dari *activation* kembali ke dirinya sendiri, dan sebuah kotak kecil diletakkan pada bagian atas dari *activation.*

Tabel 2.4Sequence Diagram

Actor Lifeline 	**General Lifeline** 
Boundary Lifeline 	**Control lifeline** 
Entity lifeline 	**Create** **synchonous** **Asynchonous**

2.7.2.5. Collaboration Diagram

Collaboration Diagram adalah bentuk lain sequence diagram. Bila *sequence* diagram diorganisir menurut waktu. *Collaboration* diagram merupakan asosiasi diantara obyek-obyek. Panah di dekat garis asosiasi menunjukkan *message*, sedangkan *content message* ditunjukkan dengan label. Angka pada *message* menunjukkan urutan *message.*

Dengan *collaboration* diagram memungkinkan untuk memodelkan pengiriman sebuah *message* ke banyak obyek pada class yang sama. Demikian juga halnya untuk menunjukkan adanya obyek aktif yang mengendalikan aliran dari *message*.

2.7.2.6. Component Diagram

Component diagram merepresentasikan dunia *rill* item yaitu *component software*. *Component software* menetap di komputer bukan di benak para analis. *Component* bisa diakses melalui *interface* nya yaitu koleksi operasi-operasi. Relasi antara *component* dan *interface*nya disebut *realization*. Suatu *component* bias mengakses *service-service* yang ada di *component* lain dengan cara import *interface*. Sedangkan component yang menyediakan *service* menggunakan export *interface*.

Hal penting dari *component* adalah *component* mewakili potongan-potongan yang independen yang bisa dipesan dan diperbaharui sewaktu-waktu

2.7.2.7. Deployment Diagram

Deployement diagram menyediakan gambaran bagaimana sistem secara fisik akan terlihat. Sistem terdiri dari node-node dimana setiap node diwakili untuk sebuah kubus. Garis yang menghubungkan antara 2 kubus menunjukkan hubungan diantara kedua node tersebut. Tipe node bisa berupa *device* yang berwujud *hardware* dan bias juga *processor* (yang mengeksekusi component) atau *execution environment* (software yang menjadi host atau mengandung software yang lain).

2.8. Eclipse

2.8.1. Definisi Eclipse

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (*platform-independent*). Berikut ini adalah sifat dari Eclipse:

1. **Multi-platform**: Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.
2. **Mulit-language**: Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Python, Perl, PHP, dan lain sebagainya.

3. **Multi-role**: Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya.

Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *open source,* yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini.

2.8.2. Arsitektur Eclipse

Sejak versi 3.0, Eclipse pada dasarnya merupakan sebuah *kernel*, yang mengangkat *plug-in*. Apa yang dapat digunakan di dalam Eclipse sebenarnya adalah fungsi dari *plug-in* yang sudah diinstal. Ini merupakan basis dari Eclipse yang dinamakan *Rich Client Platform* (RCP). Berikut ini adalah komponen yang membentuk RCP :

1. *Core platform.*
2. OSGi.
3. SWT (*Standard Widget Toolkit*).
4. JFace.
5. *Eclipse Workbench.*

2.9. Web Service

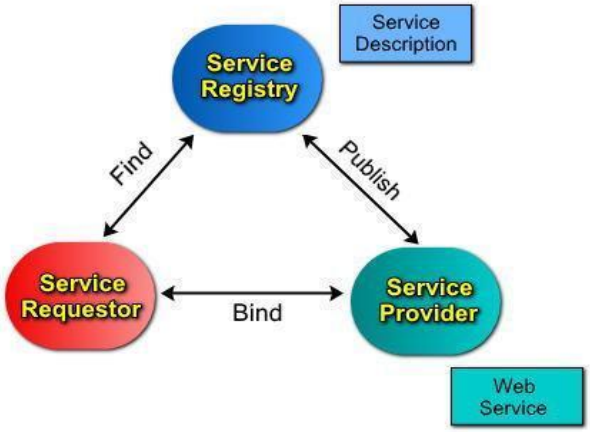
Web service adalah salah satu bentuk sistem perangkat lunak yang didesain untuk mendukung interaksi mesin-ke-mesin melalui jaringan.*Web service*memiliki *interface* yang dideskripsikan dalam format yang dapat dibaca oleh mesin. Sistem-sistem lainnya berinteraksi dengan *web service* menggunakan pesan SOAP yang umumnya dikirim melalui HTTP dalam bentuk XML.[5]

Definisi diatas diberikan oleh *World Wide Web Consortium*(W3C) yang merupakan badan yang menciptakan dan mengembangkan standar *web service.* Tetapi secara umum, *web service* tidak terbatas hanya pada standar SOAP

saja. Salah satu pustaka yang mengulas lengkap tentang *web service* menyebutkan definisi yang lebih umum: *web service*adalah aplikasi yang diakses melalui *internet* menggunakan protokol standar *internet* dan menggunakan XML sebagaiformat pesannya.

2.9.1. Arsitektur *Web service*

Secara umum, arsitektur *web service* dapat dilihat pada gambar 2.7



Gambar 2.7 Arsitektur web service[4]

Pada gambar diatas, ada tiga komponen yang membuat *web service*berjalan. Ketiga komponen itu adalah [4]:

1. **Service provide**r, merupakan pemilik *Web service* yang berfungsi menyediakan kumpulan operasi dari *Web service*.
2. **Service requesto**r, merupakan aplikasi yang bertindak sebagai klien dari *Web service* yang mencari dan memulai interaksi terhadap layanan yang disediakan.
3. **Service registry**, merupakan tempat dimana Service provider mempublikasikan layanannya. Pada arsitektur *Web service*, Service registry bersifat optional. Teknologi *web service* memungkinkan kita dapat menghubungkan berbagai jenis software yang memiliki platform dan sistem operasi yang berbeda.

2.9.1. Jenis-Jenis *Web service*

Jenis *web service* dapat dibagi menjadi dua, yaitu REST dan SOAP.

2.9.1.1. *Representational State Transfer* (REST)

REST adalah salah satu jenis *web service* yang menerapkan konsep

perpindahan antar *state*. *State* disini dapat digambarkan seperti jika *browser* meminta suatu halaman *web*, maka *server*akan mengirimkan *state* halaman *web* yang sekarang ke *browser*. Bernavigasi melalui *link-link* yang disediakan sama halnya dengan mengganti *state* dari halaman *web*. Begitu pula REST bekerja, dengan bernavigasi melalui *link-link* HTTP untuk melakukan aktivitas tertentu, seakan-akan terjadi perpindahan *state* satu sama lain. Perintah HTTP yang bisa digunakan adalah fungsi GET, POST, PUT atau DELETE. Balasan yang dikirimkan adalah dalam bentuk XML sederhana tanpa ada protokol pemaketan data, sehingga formasi yang diterima lebih mudah dibaca dan diparsing disisi *client*.

Dalam pengaplikasiannya, REST lebih banyak digunakan untuk *web service* yang berorientasi pada *resource*. Maksud orientasi pada *resource* adalah orientasi yang menyediakan *resource-resource* sebagai layanannya dan bukan kumpulan-kumpulan dari aktifitas yang mengolah *resource* itu.Alasan mengapa REST tidak digunakan dalam skripsi ini karena orientasi pada *resource*nya itu. sedangkan aplikasi *event calendar* membutuhkan pemanggilan metode yang bisa dikerjakan terhadap kumpulan *resource event*. Selain itu, karena standarnya yang kurang sehingga tidak begitu cocok diterapkan dalam aplikasi yang membutuhkan kerjasama antar aplikasi lain, dimana standar yang baik akan sangat berguna karena berbicara dalam satu bahasa yang sama. Beberapa contoh *web service* yang menggunakan REST adalah: Flickr API(*Application ProgramInterface*), YouTube API, Amazon API.

2.9.1.2. *Simple Object Access Protokol* (SOAP)

SOAP adalah protokol untuk saling bertukar pesan dalam format XML antar komputer di dalam jaringan, biasanya menggunakan HTTP/HTTPS [6]. *Web service* dalam skripsi ini menggunakan SOAP sebagai protokol pengiriman pesannya.Dalam menjalankan tugasnya, SOAP menggunakan struktur XML tertentu dalam pengirimkan *request* kepada *webservice*. Penerimaan respon dari *web service*pun memiliki strukturnya sendiri. Agar lebih mudah dalam memahami SOAP, di bawah ini diberikan salah satu contoh struktur dari SOAP *request*:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope

xmlns:SOAPNV="http://schemas.xmlsoap.org/soap/envelo
pe/"
    mlns:ns1="urn:ExampleAPI">
<SOAP-ENV:Body>
    <ns1:getPeopleByFirstLastName>
         <first>j*</first>
         <last>*</last>
    </ns1:getPeopleByFirstLastName>
</SOAP-ENV:Body></SOAP-ENV:Envelope>
```

Struktur ini terdiri dari dua bagian,*Envelope* dan *Body*. *Envelope* berisi keterangan-keterangan tentang *namespace* yang digunakan dalam dokumen XML ini, dan juga dapat berisi atribut dan *header*.Sedangkan *Body* berisi pesan yang akan dikirimkan ke *web service*. Dalam contoh diatas, *Body* berisi permintaan akan operasi getPeopleByFirstLastName yang disediakan suatu *web service* dengan parameter yang dikirimkan berupa nama depan dengan huruf pertama j dan nama belakang apa saja. Sedangkan contoh SOAP *response* yang didapat dari *server*diberikan dalam contoh dibawah :

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
   <result>
      <SOAP-ENC:Struct>
         <id>1</id>
         <firstName>John</firstName>
         <lastName>Smith</lastName>
      </SOAP-ENC:Struct>
      <SOAP-ENC:Struct>
         <id>2</id>
         <firstName>Jane</firstName>
         <lastName>Doe</lastName>
      </SOAP-ENC:Struct>
    </result>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Struktur SOAP *response* di atas juga sama dengan struktur SOAP *request*

*y*ang diterangkan sebelumnya, tetapi bisa kita lihat disini, isi dari *body* memiliki *tag result* yang berisi balasan atau *output* dari operasi *web service* yang dipanggil pada pesan SOAP *request*. Beda halnya dengan REST, SOAP *web service* lebih menekankan pada *web service* yang berorientasi *service*. Ketimbang mengambil *resource-resource* seperti dalam REST, SOAP lebih ke arah pemanggilan metode-metode yang memanipulasi *resource-resource* yang ada, dalam arti lain, melayani kebutuhan pengguna yang lebih beragam. Beberapa contoh SOAP *web service*adalah Google API, Yahoo! API, dan aplikasi *event calendar*.

2.9.2. *Web service Definition Language* (WSDL)

SOAP dapat berjalan sendiri tanpa bantuan WSDL, tetapi hal ini akan menyulitkan pengembang di sisi *client* karena tidak tahu harus mengharapkan operasi dan bentuk data apa. Karena itu WSDL diciptakan sebagai pendefinisi sebuah *web service*.WSDL adalah tata bahasa XML yang digunakan untuk mendeskripsikan sebuah *web service*.WSDL adalah bahasa XML yang digunakan untuk menjabarkan bagaimana *web service*diakses, operasi-operasi *web service*, bagaimana pesan disampaikan, dan struktur dari pesan tersebut. Dalam tugas akhir ini,*web service* yang dibangun akan dijabarkan menggunakan WSDL.

Struktur dari WSDL adalah sebagai berikut[4]:

```
<definitions
xmlns="http://schemas.xmlsoap.org/wsdl/">
<types>
</types>
<message>
</message>
<portType>
</portType>
<binding>
</binding>
<service>
</service>
</definitions>
```

Struktur ini terdiri atas lima bagian, yaitu:

1. *Type*s

   Bagian *types* menjabarkan definisi tipe data yang digunakan dalam pertukaran

pesan, tipe data disini dapat berupa tipe data sederhana ataupun tipe data yang kompleks.

2. *Messa*ge

   Bagian *message* berisi abstraksi parameter *input* dan *output* yang diharapkan dari pertukaran pesan. Parameter *input* dan *output* ini menggunakan tipe data yang dijabarkan di bagian types.

3. *PortTy*pe

   Bagian *portType* berisi koleksi dari abstraksi operasi-operasi *web service* dan abstraksi pesan-pesan yang digunakan oleh tiap-tiap operasi.

4. *Binding*

   Bagian *binding* adalah bagian yang menjabarkan informasi mengenai protokol transpor yang digunakan dan format pesannya.

5. *Service*

   Yang terakhir adalah bagian service, bagian ini yang mendeskripsikan nama dan lokasi dari suatu *web service.*

2.10. *Extensible Markup Language* (XML)

XML merupakan singkatan dari Extensible Markup Language. Dari kepanjangan tersebut kita dapat melihat ada dua kata kunci, yaitu ‘Extensible’ dan ‘Markup’. Markup berarti bahasa ini berisi kode-kode instruksi yang harus diterjemahkan oleh suatu aplikasi lain untuk menjalankan proses eksekusi yang sesungguhnya. Dalam hal ini, aplikasi yang menterjemahkan bahasa markup adalah *web browser*. Hasil eksekusi ini biasanya adalah bagaimana menampilkan data ke layar monitor.

Extensible mengandung arti bahasa XML dapat kita perluas sendiri sehingga *tag-tag* atau kode-kode di dalamnya dapat kita definisikan sendiri. Keuntungan kita memiliki *tag-tag* sendiri adalah kita dapat mendefinisikan sendiri jenis dan fungsi data yang kita buat. Sehingga, jumlah jenis dan fungsi data yang kita miliki menjadi tak berhingga. Kemampuan inilah yang menjadi kekuatan utama XML, karena di era pertukaran informasi yang besar saat ini, kita sangat memerlukan definisi mengenai jenis dan fungsi data yang banyak.

Di samping itu, dengan bantuan bahasa stylesheet, pemrograman, atau scripting (Java, PHP, dan lain-lain), XML mampu memisahkan antara *content* dan *layout* dari suatu dokumen[1]. Kemampuan ini memudahkan pengembangan aplikasi yang berbasis XML, karena data tidak bercampur dengan kode-kode untuk menampilkannya, seperti yang terjadi pada HTML. Hal ini menyebabkan tingkat skalabilitas aplikasi yang menggunakan XML menjadi tinggi.

XML juga merupakan bahasa yang non *vendor specific*, artinya tidak terikat oleh suatu pengembang perangkat lunak atau bahasa tertentu seperti Sun Microsystem, Microsoft, dan Oracle [2].

XML sendiri merupakan sebuah bahasa yang direkomendasikan oleh *World Wide Web Consortium* (W3C) pada tahun 1998. XML merupakan derivat dari SGML *(Standard Generalized Markup Language)*, yang juga merupakan bahasa yang *extensible*, namun sangat kompleks. Spesifikasi SGML sendiri mencakup lebih dari 500 halaman [3].

XML adalah respon atas tantangan yang tidak mampu dijawab oleh HTML : skalabilitas, integritas data, serta universalitas. Meskipun HTML dapat dikatakan universal untuk semua platform sistem operasi maupun browser, ia tidak dapat dipakai dalam platform non-PC, seperti PDA, handphone, dan peralatan bergerak lainnya. Sebaliknya, struktur XML memungkinkannya untuk menjadi bahasa markup yang universal, bahkan bisa diterapkan pada platform non-PC. Oleh karena sifat inilah, salah satu aplikasi yang paling banyak memanfaatkan keuntungan XML adalah aplikasi yang banyak melakukan pertukaran data, misalnya aplikasi B2B *(Business to Business)*.

Integritas data dalam XML dapat diterangkan dengan membandingkan data yang dipertukarkan dari Database Management System (DBMS) dengan data yang dipertukarkan dari dan dalam format XML. Dalam DBMS, data hanya terintegrasi (terdefinisi tipe, struktur, dan relasinya) di dalam DBMS itu sendiri. Namun, ketika data tersebut dikirimkan, data menjadi tidak terintegrasi lagi. Untuk mengintegrasikannya kembali dibutuhkan suatu pemrograman khusus, yang tentu saja membutuhkan kerja ekstra. Faktor ini menghambat fleksibilitas

sistem pertukaran data. Tingkat fleksibilitas menjadi semakin rendah bila terdapat banyak jenis DBMS dalam suatu jaringan pertukaran data.

Dengan XML, data terintegrasi kuat baik di dalam sistem penyimpanan data maupun setelah data tersebut dikirimkan. Kemampuan integrasi data ini membuat data XML bersifat siap digunakan, dan tidak dibutuhkan pemrograman tambahan untuk mengintegrasikannya kembali. Kemudian, karena sifat XML yang non *vendor specific*, maka dalam suatu jaringan pertukaran data, XML menjadi suatu wahana yang universal yang mampu menampung semua jenis informasi dalam suatu format yang seragam. Kemampuan ini membuat sistem yang mempertukaran data dalam format XML memiliki tingkat fleksibilitas yang tinggi.

Sebuah dokumen XML terdiri atas teks dan elemen. Elemen-elemen dalam dokumen XML dapat mengandung teks atau elemen lain. Elemen juga dapat mengandung atribut-atribut. Setiap dokumen XML harus memiliki sebuah pendeklarasian tipe dokumen yang diletakkan di posisi paling atas dokumen. Setiap dokumen juga harus memiliki sebuah elemen *root* yang mencakup semua teks dan elemen-elemen lain dalam dokumen. Semua elemen dalam dokumen XML harus dalam bentuk yang *well-formed*, artinya pendeklarasian elemen harus memiliki *start tag* dan *end tag*. Dokumen XML juga dapat memiliki deklarasi untuk menghubungkannya dengan XSL sebagai bahasa untuk menampilkan dokumen.

2.11. MySQL

Menurut Abdul (2009), MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (*database management system*) atau DBMS yang *multithead, multiuser*. MySQL adalah database yang paling banyak digunakan oleh *programmer* web, dengan alasan bahwa MySQL merupakan database yang sangat kuat dan cukup stabil untuk digunakan sebagai media penyimpanan data

2.12. SQLite

SQLite merupakan sebuah library proses yang menerapkan serverless (mandiri tanpa server), zero configuration, database SQL transaksional. Kode

untuk SQLite berada dalam public domain dan dengan demikian bebas untuk digunakan untuk tujuan apa pun, komersial atau swasta. Tidak seperti pada paradigma client-server umumnya, Inti SQLite bukanlah sebuah sistem yang mandiri yang berkomunikasi dengan sebuah program, melainkan sebagai bagian integral dari sebuah program secara keseluruhan. Sehingga protokol komunikasi utama yang digunakan adalah melalui pemanggilan API secara langsung melalui bahasa pemrograman. Mekanisme seperti ini tentunya membawa keuntungan karena dapat mereduksi *overhead*, *latency times*, dan secara keseluruhan lebih sederhana. Seluruh elemen basisdata (definisi data, tabel, indeks, dan data) disimpan sebagai sebuah file.