**BAB II**

**LANDASAN TEORI**

1. **Konsep dan Rekayasa Perangkat Lunak (*Software Engineering*)**

Dalam pembangunan sebuah perangkat lunak terdapat berbagai tahapan yang harus dilalui, mulai dari analisa kebutuhan pengguna, desain pengkodean, sampai pemeliharaan perangkat lunak tersebut. Dengan adanya *software engineering* diharapkan dapat menghasilkan perangkat lunak yang biaya perawatannya rendah dan menghasilkan perangkat lunak yang kinerjanya tinggi serta tepat waktu.

1. **Definisi Rekayasa Perangkat Lunak (RPL)**

Perangkat lunak (*Software*) adalah program komputer yang terasosiasi dengan dokumentasi perangkat lunak seperti dokumentasi kebutuhan, model design dan cara penggunaaan (*user manual*). Sebuah program komputer tanpa terasosiasi dengan dokumentasinya maka belum dapat dikatakan perangkat lunak (*Software*). sebuah perangkat lunak juga disebut dengan sistem perangkat lunak. Sistem berarti kumpulan komponen yang saling terkait dan mempunyai satu tujuan yang ingin dicapai (Rosa dan Shalahuddin, 2011).

Sistem perangkat lunak berarti sebuah sistem yang memiliki komponen berupa perangkat lunak yang memilki hubungan satu sama untuk memenuhi kebutuhan pelanggan (*customer*). Pelanggan adalah orang atau organisasi yang memesan atau membeli perangkat lunak atau bisa dianggap bahwa pelanggan adalah orang atau organisasi yang dengan sukarela mengeluarkan uang untuk memesan atau membeli perangkat lunak. *User* atau pemakai perangkat lunak adalah orang yang memiliki kepentingan untuk memakai atau menggunakan perangkat lunak untuk memudahkan pekerjaannya (Rosa dan Shalahuddin, 2011).

Karakter perankat lunak dalah sebagai berikut (Rosa dan Shalahuddin, 2011):

1. Perangkat lunak dibangun dengan rekayasa (*software engineering*) bukan diproduksi secara manufaktur atau pabrikan.
2. Perangkat lunak tidak pernah usang ("*wear over*") karena kecacatan dalam perangkat lunak dapat diperbaiki.
3. Barang produksi pabrikan biasanya komponen barunya akan terus diproduksi, sedangkan perangkat lunak biasanya terus diperbaiki seiring bertambahnya kebutuhan.

The international standard on software product quality suggests that software quality comprises six main attributes, These attributes can be defined as follows (Jalote, 2008):

**Software quality attributes**



**Sumber : Jalote, 2008**

**Gambar 2.1 Software quality attributes**

1. Functionality : Kemampuan untuk menyediakan fungsi yang bertemu dinyatakan dan tersirat kebutuhan ketika perangkat lunak digunakan.
2. Reliability : Kemampuan untuk menyediakan layanan kegagalan-bebas.
3. Usability : Kemampuan untuk dipahami, dipelajari dan digunakan.
4. Efficiency : Kemampuan untuk memberikan kinerja relatif sesuai dengan jumlah sumber daya yang digunakan.
5. Maintainability : Kemampuan harus dimodifikasi untuk tujuan melakukan koreksi, perbaikan atau adaptasi.
6. Portability : Kemampuan untuk disesuaikan dengan lingkungantertentu yang berbedatanpa menerapkan tindakan atau cara lain selain yang disediakan untuk ini tujuan dalam produk.
7. **Rekayasa Perangkat Lunak**

Rekayasa perangkat lunak (*softwawe engineering*) merupakan pembangunan dengan menggunakan prinsip atau konsep rekayasa dengan tujuan menghasilkan perangkat lunak yang bernilai ekonomi yang dipercaya dan bekerja secara efisien menggunakan mesin. Perangkat lunak banyak dibuat pada akhirnya sering tidak digunakan karena tidak memenuhi kebutuhan pelanggan atau bahkan karena masalah non-teknis seperti keengganan pemakai perangkat lunak (*user*) untuk mengubah cara kerja dari manual ke otomatis, atau ketidakmampuan *user* menggunakan komputer. Oleh karena itu, rekayasa perangkat lunak dibutuhkan agar perangkat lunak yang dibuat tidak hanya menjadi Perangkat lunak yang tidak terpakai (Rosa dan Shalahuddin, 2011).

pekerjaan yang terkait dengan rekayasa perangkat dapat dikategorikan menjadi tiga buah kategori umum tanpa melihat areadari aplikasi, ukuran proyek perangkat lunak yang dibuat. Setiap fase dialamatkan pada satu atau lebih pertanyaan yang diajukan sebelumnya.

Fase pendefinisian fokus pada "*what*" yang artinya harus mencari tahu atau mengidentifikasi informasi apa yang harus diproses, seperti apa fungsi dan performansi yang diinginkan , apa kriteria *validasi* yang dibutuhkan untuk mendefinisikan sistem.

Fase pengembangan yang fokus pada "*how*" yang artinya selama tahap pengembangan perangkat lunak seorang perekayasa perangkat lunak harus berussaha mendefinisikan bagaimana data distrrukturkan dan bagaimana fungsi-fungsi yang dibutuhkan diimplementasikan , bagaimana karakter antar muka tampilan , bagaimana design ditranslasikan ke bahasa pemrograman, dan bagaimana pengujian akan dijalankan.

Fase pendukung (*support phase*) fokus pada perubahan yang terasosiasi pada perbaikan kesalahan (*error*), adaptasi yang dibutuhkan pada lingkungan perangkat lunak yang terlibat, dan perbaikan yang terjadi akibat perubahan kebutuhan pelanggan.

Fase pendukung terdiri dari empat tipe perubahan antara lain (Rosa dan Shalahuddin, 2011):

1. koreksi *(correction)*
2. Adaptasi *(adaptation)*
3. Perbaikan *(enhancement)*
4. Pencegahan(*prevention)*
5. **Proses Rekayasa Perangkat Lunak**

Secara umum proses perangkat lunak terdiri dari (Rosa dan Shalahuddin, 2011) :

1. Pengumpulan spesifikasi (*Specification*)

mengetahui apa saja yang harus dapat dikerjakan sistem perangkat lunak

1. Pengembangan (*development)*

Pengembangan perangkat lunak untuk menghasilkan sistem perangkat lunak.

1. Validasi *(Validation)*

memeriksa apakah perangkat lunak sudah memenuhi keinginan pelanggan

1. Evolusi (*Evolution)*

mengubah perangkat lunak untuk memenuhi perubahan kebutuhan pelanggan

1. **Model Proses Software Engineering**

Model proses perangkat lunak merupakan representasi abstrak dari proses perangkat lunak. Setiap model proses merepresentasikan suatu proses dari sudut pandang tertentu sehingga hanya memberikan informasi parsial mengenai proses tersebut.

Model-model proses perangkat lunak adalah :

1. **Model Air Terjun (*Waterfall*)**

Model ini mengambil kegiatan proses dasar seperti spesifikasi, pengembangan, validasi dan evolusi, dan merepresentasikannya sebagai fase-fase proses yang berbeda seperti pesifikasi persyaratan, perancangan perangkat lunak, implementasi, pengujian, dan seterusnya.

Definisi Persyartan

Perancangan Sistem dan Perangkat Lunak

Implementasi dan Pengujian Unit

Integrasi dan Pengujian Sistem

Operasi dan Pemeliharaan

**Sumber : Ian Sommerville, tahun 2003**

**Gambar 2.2 Siklus Hidup Perangkat Lunak**

Model ini merupakan model pertama yang diterbitkan untuk proses pengembangan perangkat lunak diambil dari proses rekayasa lain (Royce, 1970). Berkat penurunan dari satu fase ke fase lainnya, model ini dikenal sebagai “model air terjun” atau siklus hidup perangkat lunak. Tahap-tahap utama dari model ini memetakan kegiatan-kegiatan pengembangan dasar yaitu :

1. Analisis dan definisi persyaratan : pelayanan, batasan dan tujuan sistem ditentukan melalui konsultasi dengan user sistem. Persyaratan ini kemudian didefinisikan secara rinci dan berfungsi sebagai spesifikasi sistem.
2. Perancangan sistem dan perangkat lunak : Proses perancanagan sistem membagi persyaratan dalam sistem perangkat keras atau perangkat lunak. Kegiatan ini menentukan arsitektur sistem secara keseluruhan. Perancangan perangkat lunak melibatkan identifikasi dan deskripsi abstarksi sistem perangkat lunak yang mendasar dan hubungan-hubungannya.
3. Implementasi dan pengujian unit : Pada tahap ini, perancangan perangkat lunak direalisasikan sebagai serangkaian program atau unit program. Pengujian unit melibatkan verifikasi bahwa setiap unit telah memenuhi spesifikasinya.
4. Integrasi dan pengujian sistem : Unit program atau program individual diintegrasikan dan diuji sebagai sistem yang lengkap untuk menjamin bahwa persyaratan sistem telah dipenuhi. Setelah pengujian sistem, perangkat lunak dikirim kepada pelanggan.
5. Operasi dan pemeliharaan : Biasanya, ini merupakan fase siklus hidup yang paling lama. Sistem diinstal dan dipakai. Pemeliharaan mencakup koreksi dari berbagai error yang tidak ditemukan pada tahap-tahap terdahulu, perbaikan atas implementasi unit sistem dan pengembanganan pelayanan sistem, sementara persyaratan-persyaratan baru ditambahkan.

Pada prinsipnya, hasil dari setiap fase merupakan satu atau lebih dokumen yang disetujui. Fase berikutnya tidak boleh dimulai sebelum fase sebelumnya selesai. Model ini merefleksikan praktek rekayasa.

Kekurangan yang utama dari model ini adalah kesulitan dalam mengakomodasi perubahan setelah proses dijalani. Fase sebelumnya harus lengkap dan selesai sebelum mengerjakan fase berikutnya. Masalah dengan *waterfall* :

1. Perubahan sulit dilakukan karena sifatnya yang kaku.
2. Karena sifat kakunya, model ini cocok ketika kebutuhan dikumpulkan secara lengkap sehingga perubahan bisa ditekan sekecil mungkin. Tapi pada kenyataannya jarang sekali konsumen/pengguna yang bisa memberikan kebutuhan secara lengkap, perubahan kebutuhan adalah sesuatu yang wajar terjadi.
3. *Waterfall* pada umumnya digunakan untuk rekayasa sistem yang besar dimana proyek dikerjakan di beberapa tempat berbeda, dan dibagi menjadi beberapa bagian sub-proyek
4. **Pengembangan Evolusioner**

Pendekatan ini berhimpitan dengan kegiatan spesifikasi, pengembangan, dan validasi. Suatu sistem awal dikembangkan dengan cepat dari spesifikasi abstrak. Sistem ini kemudian diperbaiki dengan masukan dari pelanggan untuk menghasilkan sistem yang memuaskan bagi kebutuhan pelanggan.

Versi Awal

Versi Menengah

Penjelasan Garis Besar

Versi Akhir

**Sumber : Ian Sommerville,2003**

**Gambar 2.3 : Pengembangan Evolusioner**

Ada 3 masalah dengan cara ini , yaitu :

1. Proses tidak bisa dilihat, manajer memerlukan hasil regular untuk mengukur

kemajuan.

1. Sistem seringkali memiliki struktur yang buruk, pperubahan yang terus-menerus cenderung merusak struktur perangkat lunak.
2. Mungkin diperlukan alat bantu dan teknik khusus, keperluan ini memungkinkan pengembangan yang cepat tetapi mungkin tidak kompatibeldengan alat bantu atau tekhnik lain dan relatif hanya sedikit orang yang memiliki keahlian untuk memakainya.
3. **Pengembangan Sistem Formal**

**Sumber : Ian Sommerville,2003**

**Gambar 2.4 : Pengembangan Sistem Formal**

Pendekatan ini didasarkan atas pembuatan spesifikasi sistem matematis dan pentransformasian spesifikasi ini, dengan memakai metode matematis, untuk membangun program. Verifikasi komponen sistem dilakukan dengan membuat argumen matematis yang disesuaikan dengan spesifikasi.

1. **Pengembangan Berdasarkan Pemakaian Ulang**

Modifikasi Persyaratan

Analisis Komponen

Spesifikasi Persyaratan

Perancangan Sistem dengan Pemakasian Ulang

Validasi Sistem

Pengembangan dan Integrasi

**Sumber : Ian Sommerville,2003**

**Gambar 2.5 : Pengembangan Berorientasi Pemakaian Ulang**

Pendekatan ini didasarkan atas adanya komponen yang dapat dipakai ulang dalam jumlah yang signifikan. Proses pengembangan sistem terfokus pada integrasi komponen-komponen ini kedalam suatu sistem, dan bukan mengembangkannya dari awal.

* 1. **UML (*Unified* *Modelling Language*)**

1. **Pengertian UML**

*Unified Modelling Language* (UML) adalah sebuah “bahasa” yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML dapat dibuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka lebih cocok untuk penulisan piranti lunak dalam bahasa berorientasi objek seperti C++, Java, atau VB. NET (Sulistyorini, 2009).

*Unified Modelling Language* merupakan alat perancangan sistem yang berorientasi pada objek. Secara filosofi kemunculan UML diilhami oleh konsep yang telah ada yaitu konsep permodelan *Object Oriented* (OO), karena konsep ini menganalogikan sistem seperti kehidupan nyata yang didominasi oleh obyek dan digambarkan atau dinotasikan dalam simbol-simbol yang cukup spesifik maka OO memiliki proses standard dan bersifat independen (Haviluddin, 2011).

UML diagram memiliki tujuan utama untuk membantu tim pengembangan proyek berkomunikasi, mengeksplorasi potensi desain, dan memvalidasi desain arsitektur perangkat lunak atau pembuat program. Komponen atau notasi UML diturunkan dari 3 (tiga) notasi yang telah ada sebelumnya yaitu Grady Booch, OOD (*Object-Oriented Design*), Jim Rumbaugh, OMT (*Object Modelling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*)

1. **Diagram-diagram dalam UML**
2. ***Use Case Diagram***

*Use case diagram* atau diagram *use case* merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat*.* *Use case* merupakan mendeskripsikan sebuah interaksi antara satu atau lebih *actor* dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu (Rosa dan Shalahuddin, 2011).

Diagram ini bersifat statis. Diagram ini memperlihatkan himpunan use case dan aktor-aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku dari suatu sistem yang dibutuhkan serta diharapkan pengguna (Sulistyorini, 2009).

*Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah *entitas* manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

*Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal.

Simbol-simbol yang digunakan dalam *use case* *diagram* dapat dilihat pada Tabel 2.1 :

**Tabel 2.1 Simbol *Use Case* *Diagram***

|  |  |  |
| --- | --- | --- |
| **NO** | **SIMBOL** | **DESKRIPSI** |
| 1 | Usecase | Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor |
| 2 | Nama actor | Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*. |
| 3 | Asosiasi | Komunikasi antara aktor dan usecase yang berpartisispasi pada usecase atau usecase memiliki interaksi dengan aktor |
| 4 | Ekstensi | Relasi usecase tambahan ke sebuah usecase dimana usecase yang ditambahkan memerlukan usecase ini untuk menjalankan fungsi atau sebagai syaratdijalankan usecase ini |
| 5 | Generalisasi | Hubungan generalisasi dan spesialisasi antara 2 buah usecase dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya |

**Sumber : Rosa dan Shalahuddin, 2011**

1. ***Class Diagram***

Diagram kelas atau class diagram menggambarkan struktur sistem dari segi pendefenisisan kelas-kelas yang akan dibuat untuk membangun sistem (Rosa dan Shalahuddin, 2011).

Kelas memiliki apa yang disebut atribut dan metode atau oerasi (Rosa dan Shalahuddin, 2011).

1. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

*Class* adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan *Class diagram* menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Simbol-simbol yang digunakan dalam *class* diagram dapat dilihat pada Tabel 2.2

**Tabel 2.2 *Simbol Class Diagram***

|  |  |  |
| --- | --- | --- |
| **NO** | **GAMBAR** | **KETERANGAN** |
| 1 | kelas | Kelas pada struktur sistem |
| 2 | interface | Sama dengan konsep interface dalam pemrograman berorientasi objek |
| 3 | Asosiasi | Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai multiplicity |
| 4 | Asosiasi berarah | Relasi antar kelas dengan makna kelas satu digunakan oleh kelas yang lain |
| 5 | Generalisasi | Hubungan generalisasi dan spesialisasi antara 2 buah usecase dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya |
| 6 | Kebergantungan | Relasi antar kelas dengan makna kebergantungan antar kelas |
| 7 | Agresi | Relasi antar kelas dengan makna semua-bagian(whole-part) |

**Sumber : Rosa dan Shalahuddin, 2011**

1. ***State machine diagram***

*State machine diagram* atau dalam bahasa Indonesia disebut diagram status digunakan untuk menggambarkan prubahan status atau transisi status dari sebuah sistem atau mesin. Perubahan tersebut digambarkan dalam suatu graf berarah. State machine diagram merupakan pengembangan dari diagrm *finite state automata* dengan penambahan beberapa fitur dan konsep baru (Rosa dan Shalahuddin, 2011).

Dalam UML, *state* digambarkan berbentuk segi empat dengan sudut membulat dan memiliki namasesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakansyarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring.

Simbol-simbol yang digunakan dalam *statechart* *diagram* dapat dilihat pada Tabel 2.3

**Tabel 2.3 Simbol *Statechart* *Diagram***

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 1 |  | *Initial State* | Start atau initial state adlah keadaan awal pada saat sistem mu,ai hidup |
| 2 |  | *Final State* | Bagaimana objek dibentuk dan dihancurkan. |
| 3 |  | Event | Event adalah kegiatan yang menyebabkan berubahnya status mesin |
| 4 |  | *State* | State atau status diagram adalah keadaan sistem pada waktu tertentu dan state dapat berubah jika ada event tertentu |

**Sumber : Rosa dan Shalahuddin, 2011**

1. ***Activity Diagram***

*Activity diagram* adalah teknik untuk mendiskripsikan logika procedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah activitas diagram bias mendukung perilk parallel sedangkan *flowchart* tidak bias (Munawar, 2005)

*Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*).

Oleh karena itu *activity diagram* tidak menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Simbol-simbol yang digunakan dalam *activity* *diagram* dapat dilihat pada Tabel 2.4

**Tabel 2.4 Simbol *Activity* *Diagram***

|  |  |
| --- | --- |
| Simbol | Keterangan |
|  | Titik awal |
|  | Titik akhir |
|  | Aktivity |
|  | Pilihan untuk pengambilan keputusan |
|  | Fork digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel manjadi satu |
|  | Aliran akhir (Flow Final) |
|  | Tanda pengiriman |
|  | Tanda waktu |

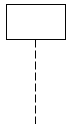
**Sumber : Munawar, 2005**

1. ***Sequence Diagram***

*Sequence* diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario.* Diagram ini menunjukkan sejumlah contoh objek dalam *message* (pesan) yang diletakkan diantara objek-objek ini didalam *use case* (Munawar, 2005)*.*

Simbol-simbol yang ada pada *sequence* diagram(Munawar, 2005)adalah:

1. Obyek atau *participant*

**Participant Pada Sebuah *Sequence* Diagram**

**Sumber: Munawar, 2005**

**Gambar 2.6 Participant Pada Sebuah *Sequence* Diagram**

Objek diletakkan di dekat bagian atas diagram dengan urutan dari kiri ke kanan. Setiap participant terhubung dengan garis titik-titik yang disebut *lifeline.* Sepanjang *lifeline* ada kotak yang disebut activation. Aktivation mewakili sebuah eksekusi operasi dari partcipant.

1. *Message*

Sebuah message bergerak dari satu participant ke participant yang lain dan dari satu lifeline ke lifeline yang lain.Sebuah message bisa jadi simple, synchronous atau asynchronous. Message yang simple adalah sebuah perpindahan (transfer) control dari satu participant ke participant yang lainnya. Jika sebuah participant mengirimkan sebuah message synchronous, maka jawaban atas message tersebut akan ditunggu sebelum diproses dengan urusannya. Namun jika message asynchronous yang akan dikirmkan, maka jawaban atas message tersebut tidak perlu ditunggu.

**Simbol-simbol Message**



**Sumber: Munawar, 2005**

**Gambar 2.7 Simbol-simbol Message**

1. Time

Time adalah diagram yang mewakili waktu pada vertical. Waktu dimulai dari atas ke bawah. Message yang lebih dekat keatas akan dijalankan terlebih dahulu disbanding message yang lebih dekatke bawah.

1. ***Collaboration Diagram***

*Collabration* *diagram* adalah perluasan dari objek diagram. (Objek diagram menentukan objek-objek dan hubungnnya satu sama lain). Collaboration diagram menunjukkan message-message objek yang dikirimkan satu sama lain (Munawar, 2005)

*Collaboration* diagram menunjukkan *message-message obyek* yang dikirimkan satu sama lain.

**Collaboration Diagram Untuk Order**



**Sumber: Munawar, 2005**

**Gambar 2.8 Collaboration Diagram Untuk Order**

Keterangan:

1. *Order, oder line, product, costumer* merupakan sebuah *obyek*.
2. Arah panah menunjukkan *obyek* yang menerima pesan.
3. Label didekat tanda panah menunjukkan pesannya apa, seperti *addProduct.*
4. Sepasang tanda kurung digunakan untuk mengakhiri *message*.
5. Nomor urut pada *message* untuk menunjukkan urutan informasi.
6. Titik dua digunakan untuk memisahkan nomor dengan *message.*

**2.2.2.7 *Deployment Diagram***

*Deployement* diagrammenunjukkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware (*Munawar, 2005)*.*

Simbol-simbol yang digunakan dalam *deployment*  *diagram* dapat dilihat pada Tabel 2.7

**Tabel 2.5 Simbol *Deployment* Diagram**

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 1 |  | Package | Package merupakan sebuah bungkusan dari satu atau lebih node. |
| 2 |  | Node | Biasanya mengacu pada perangkat keras, perangkat lunak yang tidak dibuat sendiri, |
| 3 |  | kebergantungan | Kebergantungan antar node , arah panah mengrah pada node dipakai |
| 4 |  | Link | Relasi antar node |

**Sumber : Rosa dan Shalahuddin, 2011**

1. **Data mining** 
   1. **Pendahuluan**

Ketika kita dihadapkan kepada sejumlah data dari suatu objek atau kejadian, kita perllu mengolahnya untuk mendapatkan manfaat dari data itu. Kita perlu mengenali polanya sehingga kita akan menemukan kecenderungan dari data tersebut. Misalkan data itu tentang mahasiswa baru , mungkin kita bias mengelompokkan berdasarkan asal SMU atau tingginya nilai tes masuk berdasar keduanya. Setelah proses pengelompokan ini mungkin kita akan dapat mahasiswa berdasar kategori dari SMU swasta dengan nilai tertentu. Kemudian kita dapat melakukan analisa lebih jauh, mengenai pola data mahasiswa tersebut. Misalnya kecenderungan jika mahasiswa berasal dari negeri aakan mendapat indeks prestasi lebih tinggi di semester pertamaatau kecenderungan lainnya.

Kemudian kita dapat juga melakukan prediksi atas apa yang akan terjadi pada seorang mahasiswa berdasar data masa sebelumnya berkaitan dengan indeks prestasi yang akan dicapainya pada semester satu. Pekerjaan seperti ini sering disebut dengan *pattern recognition* atau pengenalan pola. Pengenalan pola adalah bagiian dari data mining. Jadi pengenalan pola adalah suatu disiplin ilmu yang mempelajari bagaimana kita mengelompokkan obyek ke berbagaikelas dan bagaimana dari data isa kita temukan kecenderungannya. Yang pertama mengacu pada kasus klasifikasi dan yang kedua mengacu pada regresi. Data mining juga mengacu pada langkah- langkah menentukan variable atau fitur yang penting untuk dipakai dalam klasifikasi dan regresi. Data mining memegang peranan penting dalam bidang industry, keuangan , cuaca, ilmu dan teknologi.

Berikut ini adalah beberapa contoh yang memperlihatkan masalah- masalah dalam data mining (santosa, 2007).

1. Memprediksi harga suatu saham dalam beberapa bulan kedepan berdasarkan performansi perusahaan dan data-data ekonomi.
2. Memprediksi apakah seorang pasien yang diopname akan mendapat serangan jantung berikutnya berdasarkan catatan kesehatan sebelumnya dan pola makanannya.
3. Memprediksi permintaan semen dalam beberapa tahun mendatang berdasarkan data permintaan semen di tahun-tahunsebelumnya.
4. Memprediksi apakah akan terjadi tornado berdasarkan informasi dari sebuah radartentang kondidsi angin dan kondisi atmosfir yang lain.
5. Identifikasi apakah sudah terjadi penipuan terhadap kartu kredit dengan melihat catatan transaksi yang tersimpan dalam database perusahaan kartu kredit.
   1. **Pengertian Data Mining**

*Data mining* juga disebut sebagai serangkaian proses untuk menggali nilai tambah berupa pengetahuan yang selama ini tidak diketahui secara manual dari suatu kumpulan data (Pramudiono, 2006 / Kusrini dan Luthfi, 2009).

*Data mining* sering juga disebut sebagai *knowledge discovery in database* (KDD). KDD adalah kegiatan yang meliputi pengumpulan, pemakaian data, historis untuk menemukan keteraturan, pola atau hubungan dalam set data berukuran besar (Santosa, 2007).

*Data mining* adalah suatu istilah yang digunakan untuk menguraikan penemuan pengetahuan didalam *database*. *Data mining* adalah proses yang menggunakan teknik statistik, matematika, kecerdasan buatan, dan *machine learning* untuk mengekstraksi dan mengidentifikasi informasi yang bermanfaat dan pengetahuan yang terakit dari berbagai *database* besar (Turban,dkk. 2005/ Kusrini dan Luthfi, 2009).

Menurut Gartner Group *data mining* adalah suatu proses menemukan hubungan yang berarti, pola, dan kecendrungan dengan memeriksa dalam sekumpulan besar data yang tersimpan dala penyimpanan dengan menggunakan teknik pengenalan pola seperti teknik statistic dan matematika (Larose, 2005/ Kusrini dan Luthfi, 2009).

Selain defenisi diatas beberapa defenisi juga diberikan seperti tertera dibawah ini.

“*Data mining* adalah serangkaian proses untuk menggali nilai tambah dari suatu kumpulan data berupa pengetahuan yang selama ini tidak diketahui secara manual.” (Pramudiono, 2006/ Kusrini dan Luthfi, 2009).

“*Data mining* adalah analisis otomatis dari data yang berjumlah besar atatu kompleks dengan tujuan untuk menemukan pola atau kecendrungan yang penting yang biasanya tidak disadari keberadaannya.” (Pramudiono, 2006/ Kusrini dan Luthfi, 2009).

“*Data mining* merupakan bidang dari beberapa bidang keilmuan yang menyatukan teknik dari pembelajaran mesin, pengenalan pola, statistik, *database*, dan visualisasi untuk penanganan permasalahan pengambilan informasi dari *database* yang besar”(Larose, 2005/ Kusrini dan Luthfi, 2009).

* 1. **Pengenalan Pola dan Machine Learning**

Pengenalan pola adalah suatu disiplin ilmu yang mempelajari cara-cara mengklasifikasikan obyek ke beberapa kelas atau kategori dan mengenali kecenderungan data. Tergantung pada aplikasinya, obyek-obyek ini bisa berupa pasien, mahasiswa, pemohon kredit, *image* atau *signal* atau pengukuran lain yang perlu diklasifikasikan atau dicari fungsi regresinya (Santosa, 2007).

Machine Learningadalah suatu area dalam *artificial intelligence* atau kecerdasan buatan yang berhubungan dengan pengembangan teknik-teknik yang bisa diprogramkan dan belajar dari data masa lalu. Pengenalan pola, *data mining* dan *machine learning* sering dipakai untuk menyebut sesuatu yang sama. Bidang ini bersinggungan dengan ilmu probabilitas dan statistik kadang juga optimasi. *Machine learning* menjadi alat analisis dalam *data mining* (Santosa, 2007).

* 1. **Tahapan Proses KDD**

Pada bagian ini akan dijelaskan proses *data mining* yang terdiri dari (Kusnawi,2007) :

* + - 1. Pembersihan data (*data cleaning*), untuk membersihkan *noise* dan data yang tidak konsisten. Dalam kasus ini membersihkan data-data pasien yang sudah dihapus dan

1. identitas yang tidak lengkap (misal: umur, status marital, pendidikan, diagnosa, dan sebagainya);
2. Integrasi data, penggabungan data dari berbagai sumber;
3. Transformasi data, data diubah menjadi bentuk yang sesuai untuk di mining;
4. Aplikasi teknik *data mining*, proses inti dimana teknik *data mining* diterapkan untuk mengekstrak pola-pola tertentu pada data;
5. Evaluasi pola yang ditemukan;
6. Presentasi pengetahuan, menggunakan teknik visualisasi untuk menampilkan hasil *data mining* kepada pengguna (*user*).
7. **Classification**

Suatu teknik dengan melihat pada kelakuan dan atribut dari kelompok yang telah didefinisikan.Teknik ini dapat memberikan klasifikasi pada data baru dengan memanipulasi data yang ada yang telah diklasifikasi dan dengan menggunakan hasilnya untuk memberikan sejumlah aturan. Aturan-aturan tersebut digunakan pada data-data baru untuk diklasifikasi. Teknik ini menggunkan *supervised induction*, yang memanfaatkan kumpulan pengujian dari record yang terklasifikasi untuk menentukan kelas-kelas tambahan. Salah satu contoh yang mudah dan popular adalah dengan Decision tree yaitu salah satu metode klasifikasi yang paling populer karena mudah untuk diinterpretasi. Decision tree adalah model prediksi menggunakan struktur pohon atau struktur berhirarki (Kusnawi, 2007).

Dalam klasifikasi, keluaran dari dari setiap data adalah bilangan bulat atau disktrit. Dalam contoh pengambilan keputusan untuk main bola atau tidak maka keuaran tersebut bias diubah ke bilangan bulat 1 atau -1. Dengan melihat keluaran yang berupa bilangan bulat kita bias menerapkan metod klasifikasi (Santosa, 2007).

Dalam klasifiksi, terdapat target variable kategori. Sebagai contoh penggolongan pendapatan dapt dipisahkan dalam 3 kategori, yaitu pendapatan tinggi, rendah dan sedang (Kusrini dan Luthfi, 2009).

Contoh lain klasifikasi dalam bisnis dan penelitian adalah (Kusrini dan Luthfi, 2009):

1. Menentukan apakah suatu transaksi kartu kredit merupakan transaksi yang curang atau bukan
2. Memperkirakan apakah suatu pengajuan hipotek oleh nasabah merupakan suatu kredit yang baik atau buruk
3. Mendiagnosis penyakit seorang pasien untuk mendapatkan termasuk kategori apa.

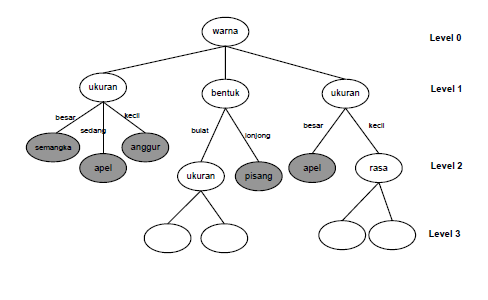
**2.5 Konsep Dasar Dari *Decision tree***

1. **Pengertian *Decision Tree***

Pohon keputusan juga berguna untuk mengeksplorasi data, menemukan hubungan tersembunyi antara sejumlah calon variabel input dengan sebuah variabel target. Karena pohon keputusan memadukan antara eksplorasi data dan pemodelan, dia sangat bagus sebagai langkah awal dalam proses pemodelan bahkan ketika dijadikan sebagai model akhir dari beberapa teknik lain (Kusrini & Emha Taufiq Luthfi, 2009).

Dalam *decision tree* tidak menggunakan *vector* jarak untuk mengklasifikasikan obyek. Seringkali data observasi mempunyai atribut-atribut yang bernilai nominal. Seperti yang diilustrasikan pada gambar 2.10, misalkan obyeknya adalah sekumpulan buah-buahan yang bisa dibedakan berdasarkan atribut bentuk, warna, ukuran dan rasa. Bentuk, warna, ukuran dan rasa adalah besaran nominal, yaitu bersifat kategoris dan tiap nilai tidak bisa dijumlahkan atau dikurangkan. Dalam atribut warna ada beberapa nilai yang mungkin yaitu hijau, kuning, merah. Dalam atribut ukuran ada nilai besar, sedang dan kecil. Dengan nilai-nilai atribut ini, kemudian dibuat *decision tree* untuk menentukan suatu obyek termasuk jenis buah apa jika nilai tiap-tiap atribut diberikan (Santosa, 2007).

**Decision Tree**



**Sumber : Santosa, 2007**

**Gambar 2.10 : Decision tree**

Ada beberapa macam algoritma *decision tree* diantaranya CART, ID3 dan C4.5. Beberapa isu utama dalam *decision tree* yang menjadi perhatian yaitu seberapa detail dalam mengembangkan *decision tree*, bagaimana mengatasi atribut yang bernilai *continues*, memilih ukuran yang cocok untuk penentuan atribut, menangani data training yang mempunyai data yang atributnya tidak mempunyai nilai, memperbaiki efisiensi perhitungan (Santosa, 2007).

*Decision tree* sesuai digunakan untuk kasus-kasus yang keluarannya bernilai diskrit. Walaupun banyak variasi model *decision tree* dengan tingkat kemampuan dan syarat yang berbeda, pada umumnya beberapa ciri yang cocok untuk diterapkannya *decision tree* adalah sebagai berikut(Santosa, 2007):

1. Data dinyatakan dengan pasangan atribut dan nilainya

2. Label/keluaran data biasanya bernilai diskrit

3. Data mempunyai *missing value* (nilai dari suatu atribut tidak diketahui)

Dengan cara ini akan mudah mengelompokkan obyek ke dalam beberapa kelompok. Untuk membuat *decision tree* perlu memperhatikan hal-hal berikut ini(Santosa, 2007):

1. Atribut mana yang akan dipilih untuk pemisahan obyek

2. Urutan atribut mana yang akan dipilih terlebih dahulu

3. Struktur *tree*

4. Kriteria pemberhentian

5. Pruning

1. **Algoritma ID3**

Algoritma ID3 merupakan algoritma yang dignakan untuk membentuk pohon keputusan (Kusrini dan Luthfi, 2009).

Pohon keputusan merupakan metode klasifikasi dan metode prediksi yang sangat kuat dan terkenal. Metode pohon keputusan mengubah fakta yng sangat besar menjadi pohonkeputusanyng sangat dapat dengan mudah dipahami dengan bahasa alami. Dan mereka juga dapat diekspreikan dalam bentuk bahasa basis data seperti structured query language untuk mencarirecord pada kategori tertentu (Kusrini dan Luthfi, 2009).

Karena pohon keputusan memadukan antara eksploitasi data dan pemodelan, dia sangat bagus sebagai langkah awal dalam proses pemodelan bahkan ketika dijadikan sebagai model akhir dari beberapa tekniklain (Kusrini dan Luthfi, 2009).

Sebuah pohon keputusan adalah struktur yang dapat digunakan untuk membagi kumpulan data yang sangat besar menjadi himnpunan-himpunan record yang lebih kecil dengan menerapkan serangkaian penbagian, anggotahimpunan hasil menjadi mirip satu dengan lain(Berry &Linoff,2004 / Kusrini dan Luthfi, 2009).

Sebuah model pohon keputusan terdiri dari sekumpulan aturan untuk membagi sejumlahpopulasi yang heterogen menjadi lebih kecil, lebih homogennya dengna memperhaikan pada model variable tujuannya.

Sebuah pohon keputusan mungkin dibangun dengan seksama secara manual atau dapat tumbuh secara otomatis dengan menerapakan salah satu atau beberapaalgoritma pohon keputusan untuk memodelkan himpunan data yang belum terklarifikasi

Variabel tujuan biasanya di kelompokkan dengan pasti dan model pohon keputusan lebih mengarah pada perhitungan probabilitas dari tiap-tiap record terhadap kategori-kategori tersebut atau untuk mengklasifikasikan record dengan mengelompokkan dalam satu kelas.

Pohon keputusan juga dapat digunakan untuk mengestimasi nilai dari variabel continue meskipun ada beberapa teknik yang lebih sesuai untuk kasus ini.

Banyak algoritma yang dapat dipakai dalam pembentukan pohon keputusan , antara lain ID3, CART, dan C4.5(larose, 2005). Algoritma C4.5 merupakan pengembangan dari ID3 (Larose, 2005 / Kusrini dan Luthfi, 2009).

Data dalam pohon keputusan biasanya dinyatakan dalam bentuk tabel dengan atribut dan record. Atribut biasanya dinyatakan dengan suatu parameter yang dibuat sebagai kriteria dalam pembentukan pohon. Misalkan untuk menentukan main tenis, kriteria yang diperhatikan adalah cuaca , angin, dan temperatur. Salah satu atribut merupakan atribut yang menyatakan data solusi per item data yang disebut target atribut. Atribut memiliki nilai-nilai yang dinamakan dengan instance. Misalkan atribut cuaca mempunyai instance berupa cerah berawan, dan hujan (Basuki & Syarif, 2003 / Kusrini dan Luthfi, 2009).

Proses pada pohon keputusan adalah mengubah bentuk data (tabel) menjadi model pohon, mengubah model pohon menjadi rule (Basuki & Syarif, 2003 / Kusrini dan Luthfi, 2009).

Secara umum Lngkah kerja algoritma ID3 untuk membangun pohon keputusan adalah sebagai berikut:

1. Hitung *Entropy* dan *Information gain* dari setiap atribut dengan menggunakan rumus:



Dimana:

S = ruang (data) sample yang digunakan untuk training.

P+ = jumlah yang bersolusi positif (mendukung) pada data sample untuk kriteria tertentu.

P- = jumlah yang bersolusi negatif (tidak mendukung) pada data sample untuk kriteria tertentu.



Dimana:

S = ruang (data) sample yang digunakan untuk training.

A = atribut.

V = suatu nilai yang mungkin untuk atribut A.

Nilai(A) = himpunan yang mungkin untuk atribut A.

|Sv| = jumlah sample untuk nilai V.

|S| = jumlah seluruh sample data.

*Entropy*(Sv) = *entropy* untuk sample-sample yang memiliki nilai V.

Tujuan dari pengukuran nilai information gain adalah untuk memilih atribut yang akan dijadikan cabang pada pembentukan pohon keputusan. Pilih atribut yang memiliki nilai information gain terbesar.

2. Bentuk simpul yang berisi atribut tersebut.

3. Ulangi proses perhitungan *information gain* yang akan terus dilaksanakan sampai semua data telah termasuk dalam kelas yang sama. Atribut yang telah dipilih tidak diikutkan lagi dalam perhitungan nilai *information gain.*

ID3 berhenti jika atribut sempurna mengklasifikasikan training sets. Atau secara rekursif mengoperasikan nilai n, dimana n adalah banyaknya nilai kemungkinan dari suatu untuk mendapatkan atribut terbaik [2].

Adapun sample data yang digunakan oleh ID3 memiliki beberapa syarat, yaitu :

a. Deskripsi atribut-nilai. Atribut yang sama harus mendeskripsikan tiap contoh dan memiliki jumlah nilai yang sudah ditentukan.

b. Kelas yang sudah didefinisikan sebelumnya. Suatu atribut contoh harus sudah didefinisikan, karena mereka tidak dipelajari oleh ID3.

c. Kelas-kelas yang diskrit. Kelas harus digambarkan dengan jelas. Kelas yang kontinu dipecah-pecah menjadi kategori-kategori yang relatif, misalnya saja metal dikategorikan menjadi *“hard, quite hard, flexible, soft, quite soft”*.

d. Jumlah contoh *(example)* yang cukup. Karena pembangkitan induktif digunakan, maka dibutuhkan *test case* yang cukup untuk membedakan pola yang valid dari peluang suatu kejadian.

**2.7 PHP *Hypertext Preprocessor* (PHP)**

PHP (akronim dari PHP *Hypertext Preprocessor*) yang merupakan bahasa pemrogramman berbasis web yang memiliki kemampuan untuk memproses data dinamis. PHP dikatakan sebagai sebuah *server-side* *embedded script language* artinya sintaks-sintaks dan perintah yang kita berikan akan sepenuhnya dijalankan oleh server tetapi disertakan pada halaman HTML biasa. Aplikasi-aplikasi yang dibangun oleh PHP pada umumnya akan memberikan hasil pada *web browser*, tetapi prosesnya secara keseluruhan dijalankan di server.

Pada prinsipnya server akan bekerja apabila ada permintaan dari client. Dalam hal ini client menggunakan kode-kode PHP untuk mengirimkan permintaan ke server (dapat dilihat pada gambar dibawah). Ketika menggunakan PHP sebagai *server-side embedded script language* maka server akan melakukan hal-hal sebagai berikut :

1. Membaca permintaan dari *client/browser*
2. Mencari halaman/page di server
3. Melakukan instruksi yang diberikan oleh PHP untuk melakukan modifikasi pada halaman/page.
4. Mengirim kembali halaman tersebut kepada client melalui internet atau intranet.

**2.7.1 Dasar PHP**

Kode PHP disimpan sebagai plain text dalam format ASCII, sehingga kode PHP dapat ditulis hampir di semua editor text seperti *windows notepad*, *windows wordpad*, dll. Kode PHP adalah kode yang disertakan di sebuah halaman HTML dan kode tersebut dijalankan oleh server sebelum dikirim ke browser.

Contoh file PHP (contoh.php):

<html>

<?php Print ("Contoh text yang menggunakan kode PHP"); ?>

</html>

Pada *file* .html, HTTP server hanya melewatkan *content* dari *file* menuju ke *browser. Server* tidak mencoba untuk mengerti atau memproses file, karena itu adalah tugas sebuah browser.

Pada *file* dengan ekstensi .php akan ditangani secara berbeda. Yang memiliki kode PHP akan diperiksa. *Web server* akan memulai bekerja apabila berada diluar lingkungan kode HTML. Oleh karena itu server akan melewati semua content yang berisi kode HTML, CSS, *JavaScript, simple text* di browser tanpa diinterpretasikan di server. Blok scripting PHP selalu diawali dengan <?php dan diakhiri dengan ?>. Blok scripting PHP dapat ditempatkan dimana saja di dalam dokumen. Pada beberapa server yang mendukung, blok scripting PHP dapat diawali dengan <? dan diakhiri dengan ?>. Namun, untuk kompatibilitas maksimum, sebaiknya menggunakan bentuk yang standar (<?php ?>).

Setiap baris kode PHP harus diakhiri dengan semikolon (;). Semikolon ini merupakan separator yang digunakan untuk membedakan satu instruksi dengan instruksi lainnya. PHP menggunakan // untuk membuat komentar baris tunggal atau /\* dan \*/ untuk membuat suatu blok komentar.

**2.7.2 Kelebihan PHP**

Beberapa kelebihan PHP dari bahasa pemrograman web, antara lain:

1. Bahasa pemrograman PHP adalah sebuah bahasa *script* yang tidak melakukan sebuah kompilasi dalam penggunaanya.
2. *Web Server* yang mendukung PHP dapat ditemukan dimana - mana dari mulai apache, IIS, Lighttpd, hingga Xitami dengan konfigurasi yang relatif mudah.
3. Dalam sisi pengembangan lebih mudah, karena banyaknya milis - milis dan developer yang siap membantu dalam pengembangan.
4. Dalam sisi pemahamanan, PHP adalah bahasa scripting yang paling mudah karena memiliki referensi yang banyak.
5. PHP adalah bahasa open source yang dapat digunakan di berbagai mesin (Linux, Unix, Macintosh, Windows) dan dapat dijalankan secara runtime melalui console serta juga dapat menjalankan perintah-perintah system.

**2.8 MySQL**

MySQL adalah sebuah [perangkat lunak](http://id.wikipedia.org/wiki/Perangkat_lunak) sistem manajemen [basis data](http://id.wikipedia.org/wiki/Basis_data) [SQL](http://id.wikipedia.org/wiki/SQL) ([bahasa Inggris](http://id.wikipedia.org/wiki/Bahasa_Inggris): *database management system*) atau DBMS yang [*multithread*](http://id.wikipedia.org/w/index.php?title=Multithread&action=edit&redlink=1), [*multi-user*](http://id.wikipedia.org/wiki/Multi-user), dengan sekitar 6 juta instalasi di seluruh dunia. [MySQL AB](http://id.wikipedia.org/w/index.php?title=MySQL_AB&action=edit&redlink=1) membuat MySQL tersedia sebagai [perangkat lunak gratis](http://id.wikipedia.org/wiki/Perangkat_lunak_gratis) dibawah lisensi [*GNU General Public License*](http://id.wikipedia.org/wiki/GNU_General_Public_License)(GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

Tidak sama dengan proyek-proyek seperti [Apache](http://id.wikipedia.org/wiki/Apache_HTTP_Server), dimana perangkat lunak dikembangkan oleh komunitas umum, dan [hak cipta](http://id.wikipedia.org/wiki/Hak_cipta) untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial [Swedia](http://id.wikipedia.org/wiki/Swedia) [MySQL AB](http://id.wikipedia.org/w/index.php?title=MySQL_AB&action=edit&redlink=1), dimana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang [Finlandia](http://id.wikipedia.org/wiki/Finlandia) yang mendirikan MySQL AB adalah: [David Axmark](http://id.wikipedia.org/w/index.php?title=David_Axmark&action=edit&redlink=1), [Allan Larsson](http://id.wikipedia.org/w/index.php?title=Allan_Larsson&action=edit&redlink=1), dan [Michael "Monty" Widenius](http://id.wikipedia.org/w/index.php?title=Michael_%28Monty%29_Widenius&action=edit&redlink=1).

MySQL adalah sebuah implementasi dari sistem manajemen basisdata relasional ([RDBMS](http://id.wikipedia.org/wiki/RDBMS)) yang didistribusikan secara gratis dibawah [lisensi GPL](http://id.wikipedia.org/wiki/GPL) (*General Public License*). Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basisdata yang telah ada sebelumnya; [SQL](http://id.wikipedia.org/wiki/SQL) (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian basisdata, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

**2.8.1 Kelebihan MySQL**

MySQL memiliki beberapa keistimewaan, antara lain :

1. *Portabilitas*. MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga, dan masih banyak lagi.
2. Perangkat lunak sumber terbuka. MySQL didistribusikan sebagai [perangkat lunak sumber terbuka](http://id.wikipedia.org/wiki/Perangkat_lunak_sumber_terbuka), dibawah lisensi [GPL](http://id.wikipedia.org/wiki/GPL) sehingga dapat digunakan secara gratis.
3. *Multi-user*. MySQL dapat digunakan oleh beberapa pengguna 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. Ragam tipe data. MySQL memiliki ragam tipe data yang sangat kaya, seperti *signed / unsigned integer, float, double, char, text, date, timestamp*, dan lain-lain.
6. Perintah dan Fungsi. MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah *Select* dan *Where* dalam perintah (*query*).
7. Keamanan. MySQL memiliki beberapa lapisan keamanan seperti level [subnetmask](http://id.wikipedia.org/w/index.php?title=Subnetmask&action=edit&redlink=1), nama [host](http://id.wikipedia.org/w/index.php?title=Host&action=edit&redlink=1), dan izin akses *user* dengan sistem perizinan yang mendetail serta sandi terenkripsi.
8. Struktur tabel. MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani *ALTER TABLE*, dibandingkan basis data lainnya semacam [PostgreSQL](http://id.wikipedia.org/wiki/PostgreSQL) ataupun [Oracle](http://id.wikipedia.org/wiki/Oracle).