BAB II

TINJAUAN PUSTAKA

II.1 Sistem

Definisi sistem berkembang sesuai dengan konteks dimana pengertian sistem itu digunakan. Berikut akan diberikan beberapa definisi sistem secara umum :

1. Kumpulan dari bagian-bagian yang bekerja sama untuk mencapai tujuan yang sama.
2. Sekumpulan objek-objek yang saling berelasi dan berinteraksi serta hubungan antar objek bisa dilihat sebagai satu kesatuan yang dirancang untuk mencapai satu tujuan.

Dengan demikian, secara sederhana sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur atau variabel-variabel yang saling terorganisasi, saling berinteraksi dan saling bergantung satu sama lain (Hanif Al Fatta ; 2007 : 3).

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi yang saling berkerja sama membentuk satu kesatuan. Berikut ini adalah komponen – komponen sistem :

a) Batas sistem *(boundary)* merupakan daerah yang membatasi antara suatu sistem dengan sistem lainnya.

b) Lingkungan Luar Sistem *(environment)* dari suatu sistem adalah apapun yang diluar batas sistem yang mempengaruhi operasi sistem.
c) Penghubung sistem *(interface)* merupakan media pendukung antara satu subsistem dalam subsistem yang lainnya.
d) Masukan sistem *(input)* adalah energi yang dimasukkan kedalam sistem.
e) Keluaran sistem *(output)* adalah hasil dari sistem yang diolah dan diklasifikasikan menjadi keluaran yang berguna.
f) Pengolahan Sistem : suatu sistem mempunyai suatu bagian pengolahan yang akan merubah masukan menjadi keluaran.
g) Sasaran sistem : jika suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak akan ada gunanya.

II.1.1 Sistem Pendukung Keputusan (*Decision Support System*)

DSS merupakan sistem informasi interaktif yang menyediakan informasi, pemodelan, dan pemanipulasian data (Kusrini, 2007:15). Sistem ini digunakan untuk membantu pengambilan keputusan dalam situasi yang semiterstruktur dan situasi yang tidak terstruktur, dimana tak seorang pun tahu secara pasti bagaimana keputusan seharusnya dibuat.

DSS biasanya dibangun untuk mendukung solusi atas suatu masalah atau untuk mengevaluasi suatu peluang. Aplikasi DSS menggunakan data, memberikan antarmuka pengguna yang mudah dan dapat menggabungkan pemikiran pengambil keputusan.

Tujuan dari DSS adalah (Kusrini, 2007:15) :

1. Membantu manajer dalam pengambilan keputusan atas masalah semi

terstruktur.

2. Memberikan dukungan atas pertimbangan manajer dan bukannya dimaksudkan untuk menggantikan fungsi manajer.
3. Meningkatkan efektifitas keputusan yang diambil manajer lebih dari pada perbaikan efesiensinya.
4. Kecepatan komputasi. Komputer memungkinkan para pengambil keputusan untuk melakukan banyak komputasi secara cepat dengan biaya yang rendah.
5. Peningkatan produktivitas.
6. Dukungan kualitas.
7. Berdaya saing.
8. Mengatasi keterbatasan kognitif dalam pemrosesan dan penyimpanan.

II.1.2 Arsitektur Sistem Pendukung Keputusan

Aplikasi sistem pendukung keputusan bisa terdiri dari beberapa subsistem, yaitu :

1. Subsistem manajemen data

   Subsitem manajemen data memasukkan satu database yang berisi data yang relevan untuk situasi dan dikelola oleh perangkat lunak yang disebut sistem manajemen database (DBMS).

2. Subsistem manajemen modal

   Merupakan paket perangkat lunak yang memasukkan model keuangan, statistik, ilmu manajemen, atau model kuantitatif lain yang memberikan kapabilitas analitik dan manajemen perangkat lunak yang tepat.

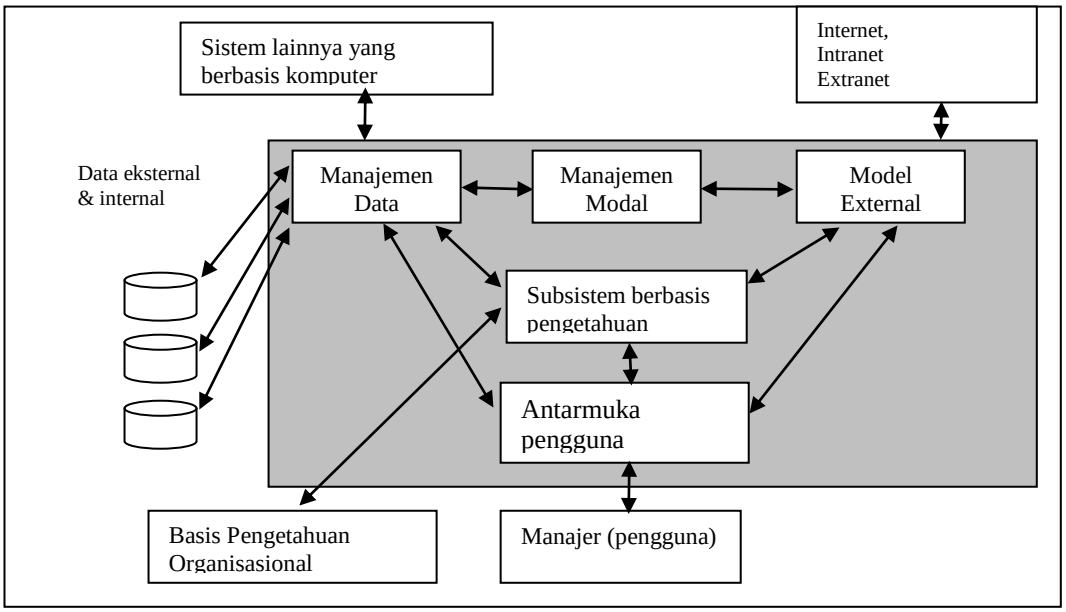
3. Subsistem antarmuka pengguna

   Pengguna berkomunikasi dengan memerintahkan sistem pendukung keputusan melalui subsistem tersebut.

4. Subsistem manajemen berbasis pengetahuan

   Subsistem tersebut mendukung semua subsistem lain atau bertindak langsung sebagai suatu komponen independen dan bersifat opsional.

Berdasarkan defenisi, sistem pendukung keputusan harus mencakup tiga konponen utama dari DBMS, MBMS, dan antarmuka pengguna. Seperti pada semua sistem informasi manajemen, pengguna bisa dianggap sebagai komponen sistem pendukung keputusan. Komponen-komponen tersebut membentuk sistem aplikasi sistem pendukung keputusan yang bisa dikoneksikan ke internet perusahaan, ekstranet atau internet. Arsitektur dari sistem pendukung keputusan dapat terlihat pada gambar II.1 berikut ini.



Gambar II.1 Arsitektur DSS

Sumber : Kusrini; 2007:26

Secara garis besar DSS dibangun oleh tiga komponen besar, yaitu *:*

1. *Database*
2. *Model Base*
3. *Software System*

Sistem database berisi kumpulan dari semua data bisnis yang dimiliki perusahaan, baik yang berasal dari transaksi sehari-hari, maupun data dasar. Untuk keperluan DSS, diperlukan data yang relevan dengan permasalahan yang hendak dipecahkan melalui simulasi. Komponen kedua adalah *Model Base* atau suatu model yang merepresentasikan permasalahan kedalam format kuantitatif sebagai dasar simulasi atau pengambilan keputusan termasuk didalamnya tujuan dari permasalahan, komponen-komponen terkait, batasan-batasan yang ada.

Kedua komponen tersebut kemudian disatukan dalam komponen ketiga yaitu *Software System,* setelah sebelumnya direpresentasikan dalam bentuk model yang dimengerti komputer.

II.1.3 Kriteria

Kriteria adalah kualitas, atribut, atau karakteristik yang dapat diukur yang menguraikan keterampilan, pengetahuan, sikap atau perilaku khusus (Paula J. C dan Jannet W. K, 2010:348).

II.1.4 Pohon Keputusan (*Decision Tree*)

Menurut Fery Sulianta (2010;56), pohon keputusan adalah pohon yang ada dalam analisa pemecahan masalah, pemetaan mengenai alternatif-alternatif

pemecahan masalah yang dapat diambil dari pemecahan masalah tersebut. Berikut ini manfaat pohon keputusan antara lain :

1. Bermanfaat dalam mengeksplorasi data, sehingga data yang tersembunyi bisa diolah dan dikembangkan lagi.
2. Untuk mem-*break down* proses pengambilan keputusan yang komleks menjadi lebih simple sehingga pengambilan keputusan akan lebih menginterpretasikan solusi dari permasalahan.
3. Dapat dijadikan sebagai *tool* pengambilan keputusan akhir.

Sebuah pohon keputusan adalah sebuah struktur yang dapat digunakan untuk membagi kumpulan data yang besar menjadi himpunan-himpunan *record* yang lebih kecil dengan menerapkan serangkaian aturan keputusan. Dengan masing-masing rangkaian pembagian, anggota himpunan hasil menjadi mirip satu dengan yang lain. Banyak algoritma yang dipakai dalam pembentukan pohon keputusan antara lain ID3, CART dan C4.5. Data dalam pohon keputusan biasanya dinyatakan dalam bentuk tabel dengan atribut dan *record*. Atribut menyatakan satu 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 dari 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.

Proses pada pohon keputusan adalah mengubah bentuk data (tabel) menjadi model pohon, mengubah model pohon menjadi *rule* dan

menyederhanakan *rule*. Secara umum algoritma C4.5 untuk membangun pohon keputusan adalah sebagai berikut (Kusrini, 2009:15) :

1. Pilih atribut sebagai akar
2. Buat cabang untuk tiap nilai
3. Bagi kasus dalam cabang
4. Ulangi proses untuk setiap cabang sampai semua kasus pada cabang memiliki kelas yang sama.

Untuk memilih atribut sebagai akar, didasarkan pada nilai *gain* tertinggi dari atribut-atribut yang ada. Untuk menghitung *gain* digunakan rumus sebagai berikut ini :

$$Gain(S, A) = Entropy(S) - \sum_{i=1}^{n} \frac{|S_i|}{|S|} * Entropy(Si) \qquad (1)$$

Sumber: Kusrini; 2009:16

Keterangan :

S = Himpunan kasus

A = Atribut

n = Jumlah partisi atribut A

$|S_i|$= Jumlah kasus pada partisi ke-i

|S|= Jumlah kasus dalam S

Sementara itu, perhitungan nilai entropi adalah seperti persamaan 2 berikut ini :

$$Entropy(S) = \sum_{i=1}^{n} - pi * \log_2 pi \qquad (2)$$

Sumber: Kusrini; 2009:16

Keterangan :

S = Himpunan kasus n = Jumlah partisi S

A = Fitur Pi = Proporsi dari $|S_i|$ terhadap S

II.2 UML

UML (*Unified Modelling Language)* adalah proses merancang piranti lunak (*Software*) sebelum melakukan pengkodean (*Coding*). Model piranti lunak dapat dianalogikan seperti pembuatan *blueprint* pada pembangunan gedung. Membuat model dari sebuah system yang kompleks sangatlah penting, karena kita tidak dapat memahami system itu secara keseluruhan(Miftakhul Huda & Bunafit Komputer, 2010:137).

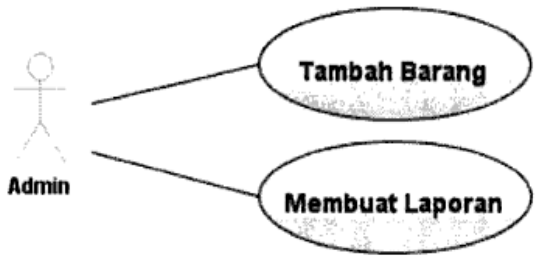
UML adalah sebuah bahasa yang telah menjadi standart dalam industri untuk visualisasi, merancang dan mendokumentasikan system piranti lunak. UML tidak hanya merupakan sebuah bahasa pemograman visual saja, namun juga dapat secara langsung duhubungkan keberbagai bahasa pemograman seperti JAVA, C++, Visual Basic atau bahkan dihubungkan secara langsung kedalam sebuah *object oriented database*. Begitu juga mengenai pen-dokumentasian dapat dilakukan seperti *requitments*, arsitektur, *design*, *source, project plan*, *tests* dan *prototypes*.

II.2.1 Diagram UML

Diagram berbentuk grafik yang menunjukkan symbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu *view* tertentu dan ketika digambarkan biasanya dialokasikan untuk *view* tertentu. UML mendefinisikan diagram-diagram berikut ini (Miftakhul Huda & Bunafit Komputer, 2010:137) :

1. *Use case diagram.*

   Use case diagram menggambarkan fungsionalitas dari sebuah sistem (Apa fungsinya), yang mempresentasikan sebuah interaksi antara actor dengan sistem. Misalnya menambah data dan membuat laporan. Aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.



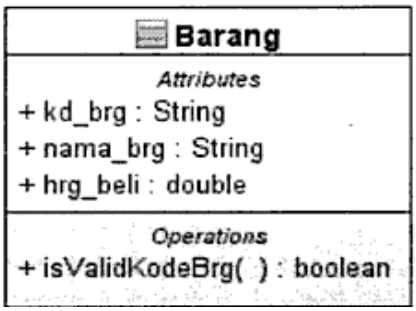
Gambar II.2 Contoh Use case diagram

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:138

2. *Class diagram.*

   Class adalah sebuah spesifikasi objek yang memiliki atribut/property dan layanan/fungsionalitas (metode/fungsi). *Class diagram* menggambarkan

struktur dan deskripsi kelas, package, dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan serta assosiasi. Kelas memiliki tiga hal pokok yaitu Nama, atribut dan metode.



Gambar II.3 Contoh class diagram

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:139

3. *Statechart diagram*

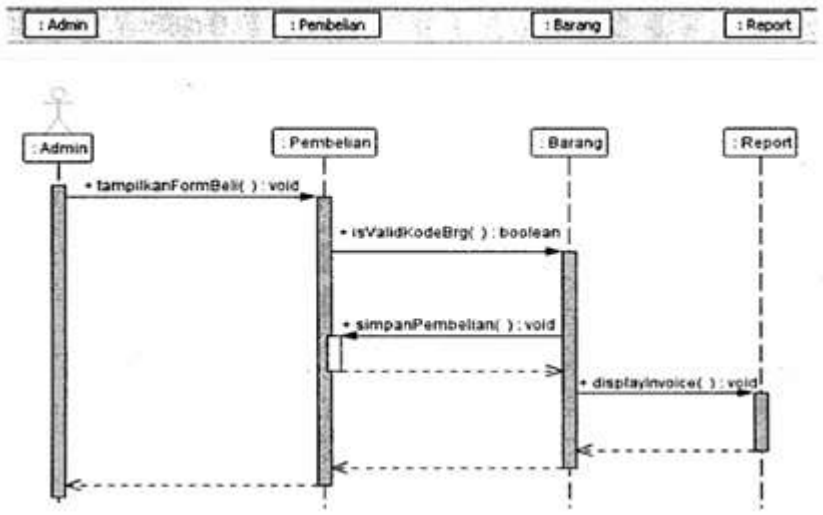
   Menggambarkan semua state (kondisi) yang dimiliki oleh suatu *object* dari suatu *class* dan keadaan yang menyebabkan state berubah. Keadaian dapat berupa *object* lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class*. Hanya yang mempunyai sejumlah *state* yang terdefenisi dengan baik dan kondisi *class* berubah oleh *state* yang berbeda.

4. *Activity diagram*

   Menggambarkan rangkaian aliran dari aktifitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* dan interaksi.

5. *Sequence diagram*

Menggambarkan interaksi antar objek didalam dan disekitar sistem berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertical(waktu) dan dimensi horizontal(objek-objek yang terkait).

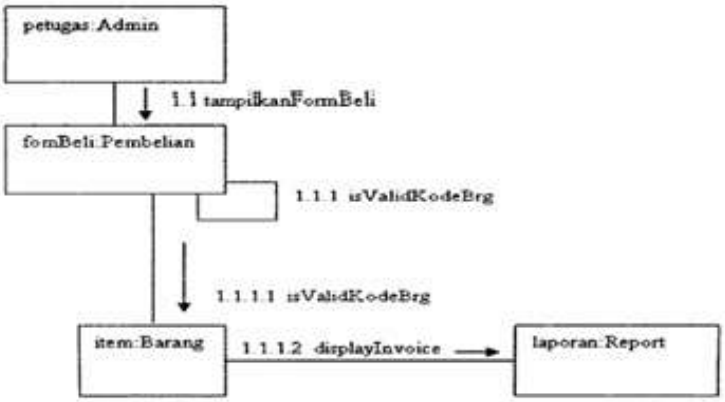


Gambar II.4 Contoh Sequence diagram

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:143

6. *Collaboration diagram*

Menggambarkan interaksi antar objek seperti *sequence diagram* tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *massage*.

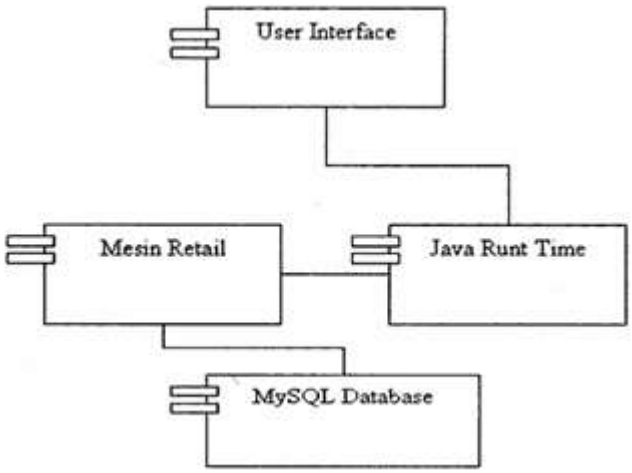


Gambar II.5 Contoh Collaboration diagram

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:144

7. *Component diagram*

   Menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*). Diantaranya berisi kode, baik berisi source kode, *binary*, *library*, *executable*.

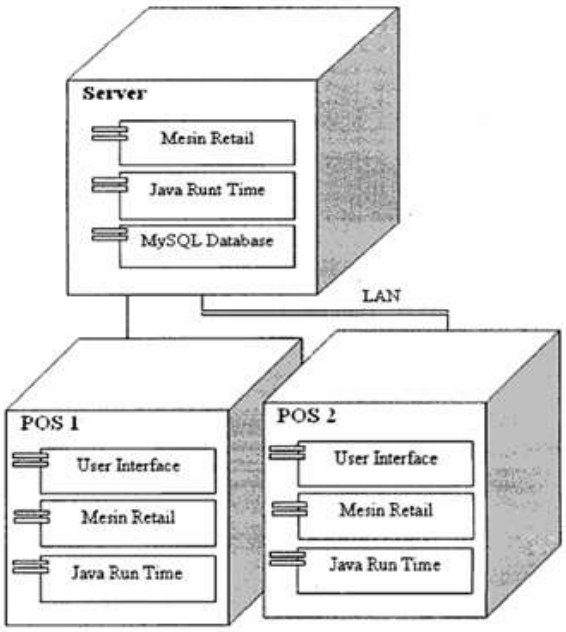


Gambar II.6 Contoh Componen Diagram

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:145

8. *Deployment diagram*

   Menggambarkan detail bagaimana komponen dibentuk dan didistribusikan dalam infrastruktur system. Komponen akan terletak pada mesin, server atau piranti keras.



Gambar II.7 Contoh Deployment diagram

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:138

II.3 MySQL

Menurut Yuniar Supardi (2010:97), perangkat lunak MySQL adalah perangkat lunak basis data *server* yang terkenal dan bersifat *open-source* dengan dukungan driver yang luas dari berbagai *vendor*. MySQL adalah sebuah implementasi dari sistem manajemen basis data relasional (RDBMS) yang didistribusikan secara gratis dibawah lisensi 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 (*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.

Kehandalan suatu sistem basisdata (DBMS) dapat diketahui dari cara kerja pengoptimasi-nya dalam melakukan proses perintah-perintah SQL yang dibuat oleh pengguna maupun program-program aplikasi yang memanfaatkannya. Sebagai peladen basis data, MySQL mendukung operasi basis data transaksional maupun operasi basisdata non-transaksional. Pada modus operasi non-transaksional, MySQL dapat dikatakan unggul dalam hal unjuk kerja dibandingkan perangkat lunak pengelolah basis data kompetitor lainnya. Namun demikian pada modus non-transaksional tidak ada jaminan atas reliabilitas terhadap data yang tersimpan, karenanya modus non-transaksional hanya cocok untuk jenis aplikasi yang tidak membutuhkan reliabilitas data seperti aplikasi blogging berbasis *web*, CMS, dan sejenisnya. Untuk kebutuhan sistem yang ditujukan untuk bisnis sangat disarankan untuk menggunakan modus basisdata transaksional, hanya saja sebagai konsekuensinya unjuk kerja MySQL pada modus transaksional tidak secepat unjuk kerja pada modus non-transaksional.

II.3.1 *Database*

Database (basis data) adalah koleksi atau kumpulan data yang saling berhubungan disusun menurut aturan tertentu secara logis, sehingga menghasilkan informasi (Yuhefizard, 2008:2). Sebuah informasi yang berdiri sendiri tidaklah dikatakan *database*.

DBMS (*Database Management System*) adalah perangkat lunak yang berfungsi untuk mengelola database. Mulai dari database itu sendiri sampai

dengan proses yang berlaku dalam database tersebut baik berupa entry, edit, hapus, query terhadap data, membuat laporan dan lain sebagainya secara efektif dan efesien.

Ada tiga kelompok perintah yang digunakan dalam mengelola dan meng-organisasikan data dalam RDBMS, yaitu :

1) *Data Defenition Language* (DDL)

   DDL merupakan perintah yang digunakan oleh seorang database administrator untuk mendefenisikan struktur database, baik membuat table baru, menentukan struktur penyimpanan tabel, model relasi antar tabel, validasi data.

2) *Data Manipulation Language* (DML)

   DML adalah perintah-perintah yang digunakan untuk memanipulasi dan mengambil data pada suatu database. Manipulasi yang dapat dilakukan terhadap data adalah

   a) Penambahan data
   b) Penyisipan data
   c) Penghapusan data
   d) Pengubahan data.

3) *Data Control Language* (DCL)

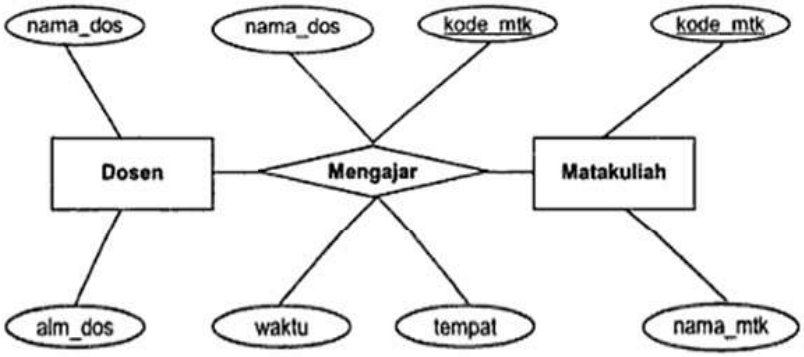
   Bagian ini berkenaan dengan cara mengendalikan data, seperti siapa yang bisa melihat isi data, bagaimana data bisa digunakan banyak pengguna. Lebih mengarah pada sekuritas data.

II.3.2 ERD (*Entity Relationship Diagram*)

ERD digunakan untuk menggambarkan secara sistematis hubungan antar *entity-entity* yang ada dalam suatu sistem database menggunakan simbol-simbol sehingga mudah dipahami (Yuhefizard, 2008:17). Simbol yang digunakan adalah:

1. Persegi panjang, berfungsi untuk menyatakan suatu *entity*.
2. *Elips*, berfungsi untuk menyatakan *attribute*. Jika diberi garis bawah menandakan bahwa *attribute* tersebut merupakan *attribute/field* kunci.
3. Belah ketupat, menyatakan jenis relasi.
4. Garis, penghubungan antara relasi dengan *entity* dan antara *entity* dengan *attribute*.

Contoh : hubungan antara *entity* dosen dengan *entity* matakuliah, sebagai berikut ini :



Gambar II.8 Contoh ERD

Sumber : Yuhefizard, 2008:17

II.3.3 Hubungan Antar Tabel

Pada relasional database dapat didefenisikan hubungan antar table yaitu sebagai berikut :

1. Satu Ke Satu (*One To One*).

Merupakan hubungan antar tabel dimana suatu tabel hanya memiliki suatu data pada tabel yang direferensikan.

2. Satu Ke Banyak (*One To Many*).

   Merupakan hubungan antar tabel dimana suatu tabel memiliki lebih dari satu data pada tabel yang direferensikan.

3. Banyak Ke Banyak (*Many To Many*).

   Merupakan hubungan antar tabel dimana suatu tabel memiliki lebih dari satu data yang direferensikan pada masing-masing tabel.

II.3.4 Normalisasi

Normalisasi merupakan cara pendekatan dalam membangun desain logika basis data relasional yang tidak secara langsung berkaitan dengan model data, tetapi dengan menerapkan sejumlah aturan dan kriteria standart untuk menghasilkan struktur tabel yang normal (Kusrini, 2007:40). Pada dasarnya desain logika basis data relasional dapat menggunakan prinsip normalisasi maupun transformasi dari model E-R ke bentuk fisik.

Dalam perspektif normalisasi sebuah database dikatakan baik jika semua tabel yang membentuk basis data sudah berada dalam keaadaan normal. Suatu tabel dikatakan normal apabila :

1) Ada dekomposisi/penguraian tabel, maka dekomposisinya dijamin aman (lossless-join decomposition).
2) Terpeliharanya ketergantungan functional pada saat perubahan data (dependency preservasion).

3) Tidak melanggar *Boyce Code Normal Form* (BCNF), jika tidak bisa minimal tidak melanggar bentuk normal ketiga.

Kondisi-kondisi yang diujikan pada proses normalisasi adalah sebagai berikut :

1) Menambah data.
2) Mengedit.
3) Menghapus.
4) Membaca.

II.3.5 Bentuk-bentuk Normalisasi

Menurut Kusrini (2007: 41), bentuk-bentuk normalisasi adalah sebagai berikut ini :

1) Bentuk tidak normal.

   Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan mengikuti format tertentu, dapat saja tidak lengkap dan terduplikasi.

2) Bentuk normal tahap pertama (*1st normal form*).

   Sebuah tabel dapat dikatakan 1NF jika:

   a) Tidak ada baris yang duplikat dalam tabel tersebut.
   b) Masing-masing cell bernilai tunggal.

3) Bentuk normal tahap kedua (2nd normal form).

   Bentuk normal kedua (2NF) terpenuhi jika pada sebuah tabel semua *attribute* yang tidak termasuk dalam *primary key* memiliki ketergantungan fungsional pada *primary key* secara utuh. Sebuah tabel tidak memenuhi

2NF, jika ketergantungannya hanya bersifat parsial atau tergantung pada sebagian *primary key*.

4) Bentuk normal tahap ketiga (3^{rd} normal form).

   Sebuah tabel dikatakan memenuhi bentuk normal ketiga (3NF), jika untuk setiap ketergantungan fungsional dengan notasi X-> A, dimana A mewakili semua atribut tunggal didalam tabel yang tidak ada didalam X, maka :

   a) X haruslah *superkey* pada tabel tersebut.
   b) Atau A merupakan bagian dari primary key pada tabel tersebut.

5) Bentuk normal tahap keempatdan kelima.

   Bentuk normal keempat berkaitan dengan sifat ketergantungan banyak nilai (*multivalued dependency*) pada suatu tabel yang merupakan pengembangan dari ketergantungan fungsional. Adapun bentuk normal kelima merupakan nama lain dari *Project Join Normal Form* (PJNF).

6) *Boyce code normal form* (BCNF).

   Suatu tabel dikatakan BNCF jika memenuhi 1NF dan relasi harus bergantung fungsi pada atttribut *superkey*.

II.4 Bahasa Pemrograman Java

Bahasa pemrograman java dimulai dari sebuah tim pengembang *software* dari Sun Microsystem yang dipimpin oleh James Golsing dan Patrick Naughton. Pada tahun 1991, Sun Microsystem mengembangkan sebuah bahasa

pemrograman yang berukuran kecil untuk diimplementasikan pada alat elektronik rumah tangga seperti *switchbox* TV kabel. Berhubung alat tersebut tidak memiliki banyak memori, maka bahasa yang digunakan harus sangat kecil dan menghasilkan kode yang kecil. Permasalahan lainnya adalah alat-alat tersebut memiliki CPU yang berbeda-beda karena dibuat oleh manufaktur yang berbeda sehingga bahasa pemrograman tersebut tidak terikat pada sebuah arsitektur mesin tertentu.

Oleh karena adanya sebuah bahasa pemrograman yang kecil, menghasilkan kode yang kecil dan *platform independent* (tidak terikat pada platform lain) membuat tim pada proyek tersebut terinspirasi oleh ide pemrograman yang sama dan telah ditemukan oleh Niklaus Wirth (penemu bahasa pemrograman pascal). Beliau memiliki pemikiran yang sama tentang sebuah *software* bahasa pemrograman yang portable dan tidak tergantung pada sebuah *platform* atau mesin. Bahasa pemrograman komersial yang disebut UCSD Pascal meghasilkan kode *intermediate* yang diperuntukkan bagi sebuah mesin virtual. Sehigga kode asli dari bahasa pemrograman tersebut tidak tergantung pada mesin ataupun *platform system* operasi karena UCSD Pascal menghasilkan *intermediate* kode yang selanjutnya dikompilasi atau diterjemahkan oleh mesin virtual kekode mesin dimana kode tersebut dijalankan.

Prinsip yang sama juga digunakan oleh tim dari Sun Microsystem, hanya pada java mesin virtualnya disebut Java Virtual Machine(JVM). Namun, karena mereka kebanyakan berlatar belakang Unix membuat bahasa pemrograman ini menggunakan bahasa C++ bukan Pascal. Sehingga jika JVM ada, maka kode asli

program akan dapat dijalankan pada mesin yang berbeda-beda dan platform sistem operasi yang berbeda. Pada dasarnya Java juga telah dirancang untuk memenuhi standart bahasa pemrograman yang menggunakan metode pemrograman berorientasi objek. Bahasa ini pada awalnya disebut "OAK" oleh James Gosling. Akan tetapi setelah mereka menyadari bahwa sudah ada bahasa pemrograman yang menggunakan nama tersebut mereka menggantinya dengan Java.

Booming bahasa Java dimulai pada tahun 1995 ketika Netscape memutuskan untuk menggunakan Java pada web browsernya, yaitu Netscape Navigator pada januari 1996. Hal ini kemudian diikuti oleh raksasa-raksasa software seperti IBM, Symentec, Inprise, dan masih banyak yang lainya termasuk Microsoft dengan internet Explorer-nya. Sun sendiri merilis Java pertama kalinya pada awal tahun 1996, kemudian diikuti dengan versi 1.02 beberapa kemudian. Pada awalnya Java masih belum mampu memenuhi kebutuhan para pengembang untuk membangun sebuah software secara profesional. Baru pada tahun 1998 muncul versi Java 1.2 yang dirilis pada bulan Desember dan beberapa hari kemudian namanya diganti dengan Java 2.

Java Language Spesification adalah defenisi teknis dari bahasa pemrograman Java yang didalamnya terdapat aturan penulisan sintaks dan *semantic* Java (Wahana Komputer, 2010: 3). API (*Application Programming Interface)* adalah sebuah layer yang berisi *class-class* yang sudah didefenisikan dan antarmuka pemrograman yang akan membantu para pengembang aplikasi dalam perancangan sebuah aplikasi. API memampukan para pengembang untuk

dapat mengakses fungsi-fungsi sistem operasi yang diizinkan melalui bahasa Java. Ada tiga buah API dari Java, yaitu :

1. J2SE, (*Java 2 Standart Edition*) adalah API yang dapat digunakan untuk mengembangkan aplikasi-aplikasi yang bersifat *client-side standalone* atau applet.
2. J2EE, (*Java 2 Enterprise Edition)* adalah API yang digunakan untuk melakukan pengembangan apilkasi-aplikasi yang bersifat *server-side* seperti *Java Servlet* dan *Java Server Pages*.
3. J2ME, (*Java 2 Micro Edition)* adalah API yang merupakan subset dari J2SE tetapi memiliki kegunaan untuk mengembangkan aplikasi pada *handheld device* seperti *Smart Phone* atau PDA yang didalamnya telah ditanamkan interpreter Java.
4. *Sun Microsystem* merilis tiap versi J2SE dengan sebuah JDK (*Java Development Kit).* JDK adalah sekumpulan program kecil yang dapat membantu para pengembang aplikasi dalam merancang dan melakukan testing program. Selain JDK, ada juga beberapa tool untuk pengembangan aplikasi diantaranya adalah sebagai berikut :
   a) Jbuilder dari Borland.
   b) NetBeans *Open Source* dari Sun.
   c) Sun ONE, yaitu versi komersial dari NetBeans yang dibuat oleh Sun.

      Eclips *Open Source* dari IBM.

II.5 Variabel dan Type data

Dalam dunia pemrograman komputer, variable dan tipe data pasti akan ditemui pada bahasa pemrograman apapun, karena keduanya sangat besar kegunaannya dan sangat penting bagi pembuatan sebuah aplikasi. Variable dan tipe data akan saling berhubungan tidak dapat dipisahkan.

Variable berguna untuk menyimpan nilai sementara untuk dapat dipergunakan kembali. Dikatakan sementara waktu karena nilai sebuah variable akan disimpan dalam memori komputer yang bersifat tidak permanent. Untuk menggunakan sebuah variable, harus dilakukan pendeklarasian variable tersebut.

Aturan dalam memberikan nama sebuah variable adalah sebagai berikut ini :

1. Harus diawali dengan huruf alphabet, tidak boleh angka.
2. Tidak boleh lebih dari 255 karakter.
3. Tidak diperkenankan untuk mendeklarasikan dua buah variable dengan nama yang sama.
4. Tidak boleh menggunakan *keyword* java yaitu kata-kata yang dipergunakan oleh java *module, integer*.
5. Tipe data adalah jenis nilai yang tersimpan dalam variable, huruf, angka ataupun tanggal. Tipe data diperlukan agar java dapat langsung mengenal jenis data yang tersimpan dalam variable. Berikut ini adalah jenis tipe data yang didukung oleh java :

a) Boolean, tipe data ini hanya boleh diisi oleh dua buah nilai yaitu true (benar) dan false (salah).

b) Byte, 0 sampai dengan 255

c) Char, tipe data ini hanya boleh diisi oleh sebuah karakter (Unicode).

d) Date, tipe data java yang merupakan nilai sebuah tanggal dan waktu, dengan jangkauan tanggal 1 januari 00001 sampai dengan 31 desember 9999.

e) Decimal, 0 sampai dengan +/- 79.228.162.514.264.337.593.543.-950.335 (tanpa bilangan decimal dibelakang koma) atau 0 sampai dengan +/- 7, 9228162514264337593543950335 (dengan bilangan decimal dibelakang koma maksimal 28 angka).

f) Double, -1,79769313486231570E+308 sampai dengan 1, 79769313486231570E+308 (untuk bilangan positif).

g) Integer, -2.147.483.648 sampai dengan 2.147.483.647.

h) Long, -9.223.372.036.854.775.808 sampai dengan 9.223.372.-036.854.775.807.

i) Sbyte, -128 sampai dengan 127.

j) Short, -32,768 sampai dengan 32.767.

k) Single, -3,4028235E+38 sampai dengan -1,401298E-45(untuk bilangan negative) atau 1,401298E-45 sampai dengan 3,4028235E+38 (untuk bilangan positif).

l) String, 0 sampai dengan 2 juta karakter (*Unicode*) bisa huruf, angka atau karakter yang tidak umum lainnya.

m) UInteger, 0 sampai dengan 4.294.967.295.

n) ULong, 0 sampai dengan 18.446.744.073.709.551.615 (1.8…E+19).