BAB II

TINJAUAN PUSTAKA

II.1 Tajwid

Ilmu tajwid adalah ilmu yang digunakan untuk mengetahui sebenarnya membunyikan huruf-huruf dengan betul, baik huruf yang berdiri sendiri maupun dalam rangkaian (Ust. Hanafi :5). Tajwid berfungsi untuk memelihara bacaan pada Al-Qur'an dari kesalahan dan perubahan serta memelihara lisan dari kesalahan membacanya. Dalam mempelajari ilmu tajwid hukumnya fardlu kifayah, namun mempergunakan ilmu tersebut didalam membaca Al-Qur'an, maka hukumnya adalah fardlu Aim/wajib.

Hadis Riwayat Bukhori mengatakan : " Orang yang paling baik diantara kalian/kamu;ialah orang yang belajar Al-Qur'an dan mengajarkan-Nya kepada orang lain " (Ust.Hanafi :5).

Berdasarkan keterangan Hadis tersebut , maka belajar dan membaca Al-Qur'an dengan sebaik-baiknya menjadi keharusan/wajib bagi setiap muslim. Untuk pegangan dan tuntunan hidupnya di dunia dan akhirat kelak.

II.1.1 Huruf Hijaiyah

Huruf hijaiyah menjadi pembahasan ilmu tajwid. Huruf hijaiyah ada 28 (Mahfan;2005:6), yaitu :

1. Kho	= خ	2. Ha	= ح
3. Jim	= ج	4. Tsa	= ث
5. Ta	= ت	6. Ba	= ب

7. Alif	= ا	8. Shod	= ص
9. Syin	= ش	10. Sin	= س
11. Zay	= ز	12. Ro	= ر
13. Dzal	= ذ	14. Dal	= د
15. Qof	= ق	16. Fa	= ف
17. Ghoin	= غ	18. ‘Ain	= ع
19. Zho	= ظ	20. Tho	= ط
21. Dhod	= ض	22. Lam-Alif	= لا
23. Hha	= ه	24. Wau	= و
25. Nun	= ن	26. Mim	= م
27. Lam	= ل	28. Kaf	= ك

II.1.2 Nun Mati / Tanwin

Nun mati (Ust. Hanafi :11) adalah huruf ن yang ditandai seperti : نْ . Sedangkan Tanwin adalah suara nun sukun yang terdapat diakhir kata benda. Tanwin merupakan tanda harokah rangkap, misalnya :

1. Suara AN ditandai dengan FATHA TAIN :
2. Suara IN ditandai dengan KASRO TAIN : 
3. Suara UN ditandai dengan DLOMMA TAIN : 

Apabila Nun mati dan tanwin bertemu dengan huruf hijaiyah, maka mempunyai 4 macam hukum, yaitu :

1. Izh-har

   Izhar artinya jelas atau terang. Apabila ada nun mati atau tanwin (ـًـٌـٍ / نْ) bertemu dengan salah satu huruf halqi (ا ح خ ع غ ه), maka dibacanya jelas/terang.

2. Idghom

   Idgham Bighunnah (dilebur dengan disertai dengung)

   Yaitu memasukkan/meleburkan huruf nun mati atau tanwin (نْ / ـًـٌـٍ) kedalam huruf sesudahnya dengan disertai (ber)dengung, jika bertemu dengan salah satu huruf yang empat, yaitu: ن م و ي

   Idgham Bilaghunnah (dilebur tanpa dengung)

   Yaitu memasukkan/meleburkan huruf nun mati atau tanwin (نْ / ـًـٌـٍ) kedalam huruf sesudahnya tanpa disertai dengung, jika bertemu dengan huruf lam atau ra (ر، ل)

3. Iqlab

   Iqlab artinya menukar atau mengganti. Apabila ada nun mati atau tanwin (نْ / ـًـٌـٍ) bertemu dengan huruf ba (ب), maka cara membacanya dengan menyuarakan /merubah bunyi نْ menjadi suara mim (مْ), dengan merapatkan dua bibir serta mendengung.

4. Ikhfa’

   Ikhfa artinya menyamarkan atau tidak jelas. Apabila ada nun mati atau tanwin (نْ/ ـًـٌـٍ) bertemu dengan salah satu huruf ikhfa yang 15 (ت ث ج د ذ س ش ص ض ط ظ ف ق ك), maka dibacanya samar-samar, antara jelas dan tidak (antara izhar dan idgham) dengan mendengung.

II.2 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 :

1. Batas sistem *(boundary)* merupakan daerah yang membatasi antara suatu sistem dengan sistem lainnya.
2. Lingkungan Luar Sistem *(environment)* dari suatu sistem adalah apapun yang diluar batas sistem yang mempengaruhi operasi sistem.
3. Penghubung sistem *(interface)* merupakan media pendukung antara satu subsistem dalam subsistem yang lainnya.
4. Masukan sistem *(input)* adalah energi yang dimasukkan kedalam sistem.
5. Keluaran sistem *(output)* adalah hasil dari sistem yang diolah dan diklasifikasikan menjadi keluaran yang berguna.
6. Pengolahan Sistem : suatu sistem mempunyai suatu bagian pengolahan yang akan merubah masukan menjadi keluaran.
7. Sasaran sistem : jika suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak akan ada gunanya.

II.2.1 Pakar

Seorang pakar/ahli (*human expert)* adalah seorang individu yang memiliki kemampuan pemahaman yang superior atas suatu masalah (Kusrini; 2006:20). Seorang pakar memiliki kemampuan :

1. Dapat mengenali (*recognizing)* dan merumuskan masalah.
2. Menyelesaikan masalah dengan cepat dan tepat.
3. Menjelaskan solusi.
4. Belajar dari pengalaman.
5. Restrukturisasi pengetahuan.

6. Menentukan hubungan/relevansi.
7. Memahami batas kemampuan.

II.2.2 Sistem Pakar

Sistem pakar adalah sistem berbasis komputer yang menggunakan pengetahuan, fakta, dan teknik penalaran dalam memecahkan masalah yang biasanya hanya dapat dipecahkan oleh seorang pakar dalam bidang tertentu (Kusrini, 2006). Perkembangan sistem pakar sangat pesat yang dimanfaatkan pada pelbagai bidang seperti ekonomi, keuangan, teknologi dan kedokteran.

II.2.3 Arsitektur Sistem Pakar.

Komponen utama pada struktur sistem pakar (Kusrini, 2006) 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.

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.

II.2.4 Teknik Representasi Pengetahuan.

Representasi pengetahuan (Kusrini, 2006:24) adalah suatu teknik untuk merepresentasikan basis pengetahuan yang diperoleh ke dalam suatu skema/diagram tertentu sehingga dapat diketahui relasi/keterhubungan antara suatu data dengan data yang lain. Teknik ini membantu *knowledge engineer* dalam memahami struktur pengetahuan yang akan dibuat sistem pakarnya. Terdapat beberapa teknik representasi pengetahuan yang biasa digunakan dalam pengembangan suatu sistem pakar, yaitu :

1) *Rule-Based Knowledge.*

   Pengetahuan direpresentasikan dalam suatu bentuk fakta (*facts*) dan aturan (*rules*). Bentuk representasi ini terdiri atas premise dan kesimpulan.

2) *Frame-Based Knowledge*.

   Pengetahuan direpresentasikan dalam suatu bentuk hirarki atau jaringan frame.

3) *Object-Based Knowledge*.

   Pengetahuan direpresentasikan sebagai jaringan dari obyek-obyek. Obyek adalah elemen data yang terdiri dari data dan metoda (proses).

4) *Case-Base Reasoning*.

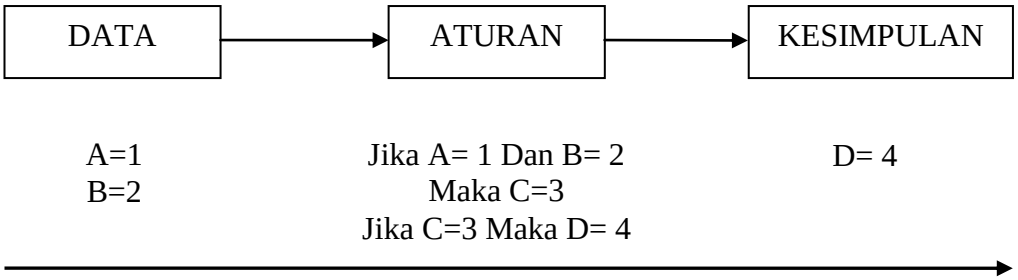
   Pengetahuan direpresentasikan dalam bentuk kesimpulan kasus (*cases*)

II.2.5 Metode *Forward Chaining*

Forward Chaining berarti menggunakan himpunan aturan kondisi-aksi. Pada metode ini, data digunakan untuk menentukan aturan mana yang akan

dijalankan, kemudian aturan tersebut dijalankan. Proses ini akan diulang sampai menemukan suatu hasil (Kusrini, 2006).

Pada gambar 1.1 berikut ini menunjukkan bagaimana cara kerja metode *Forward Chaining*:



Gambar II.1 Cara kerja *forward chaining*

Sumber: Kusrini (2006:36)

Metode *forward chaining* cocok digunakan untuk menagani masalah pengendalian (*controlling)* dan peramalan (*prognosis).*

II.3 UML

UML (*Unified Modelling Language)* adalah sebuah bahasa standard untuk pengembangan sebuah software yang dapat menyampaikan bagaimana membuat dan membentuk model-model, tetapi tidak menyampaikan apa dan kapan model yang seharusnya dibuat yang merupakan salah satu proses implementasi pengembangan software (Adi Nugroho; 2010:6).

UML adalah metode pemodelan secara visual sebagai sarana untuk merancang dan membuat software berorientasi objek karena UML merupakan bahasa visual untuk pemodelan bahasa berorientasi objek, maka semua elemen dan diagram berbasiskan pada paradigm objek oriented.

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*.

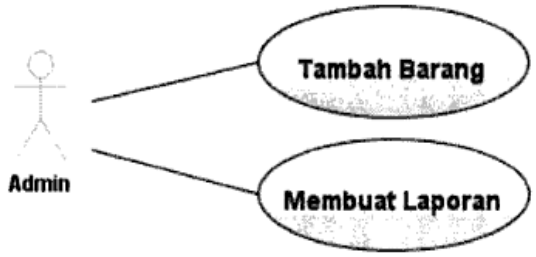
II.3.1 Diagram UML

Diagram berbentuk grafik yang menunjukkan symbol elemen model yang disusun untuk mengilustrasikan bagaian 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 :

1. *Use case diagram.*

   Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara *actor* dengan sistem. *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. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi

secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain. Gambar II.2 berikut ini adalah contoh diagram *use case*.

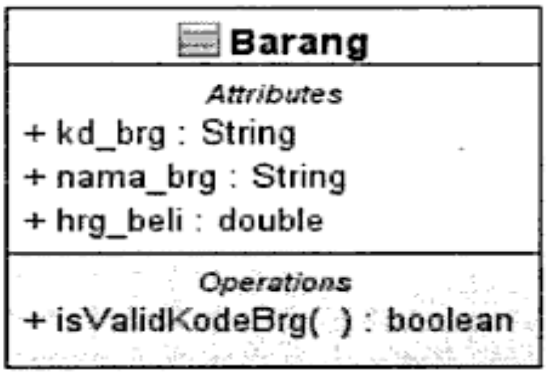


Gambar II.2 Gambar Use Case

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:138

2. *Class diagram.*

Menggambarkan struktur statis *class* didalam sistem. *Class* mere-presentasikan sesuatu yang ditangani oleh sistem. *Class* dapat berhubungan dengan yang lain melalui cara: terhubung satu sama lain (*associated)*, satu *class* tergantung/menggunakan *class* yang lain (*dependent*), satu *class* merupakan spesialisasi dari *class* lainnya (*Specialed)*, grup bersama sebagai satu unit (*Package)*. Gambar II.3 berikut ini adalah contoh *class* diagram.



Gambar II.3 Contoh *Class* Diagram

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:139

3. *Statechart diagram*

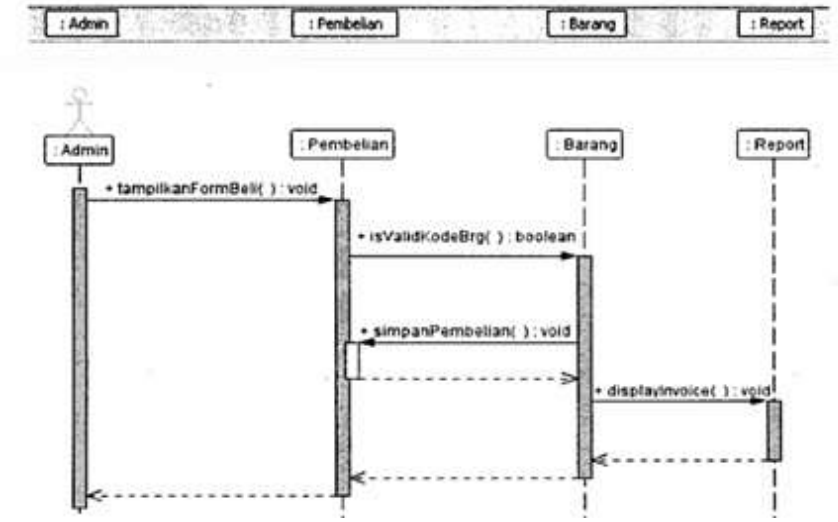
   Menggambarkan semua state (kondisi) yang dimiliki oleh suatu *object* dari suatu *class* dan keadaan yang menyebabkan state berubah. Keadaian dapat berupa *object* lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class.* Hanya yang mempunyai sejumlah *state* yang terdefenisi dengan 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 kolaborasi dinamis antara sejumlah *object*. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara *object* juga interaksi antara *object,* sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem. Gambar II. Berikut ini adalah contoh *sequence* diagram.

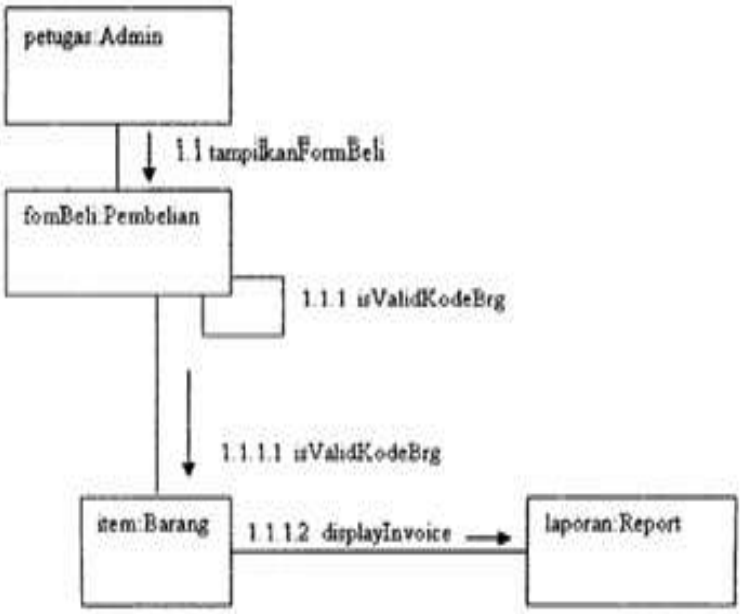


Gambar II.4 Contoh *Sequence* Diagram

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:143

6. *Collaboration diagram*

Menggambarkan kolaborasi dinamis seperti *sequence diagram.* Dalam menunjukkan pertukaran pesan, *collaboration diagram* menggambarkan *object* dan hubungannya. Jika penekanannya pada waktu atau urutan maka digunakan *sequence diagram*, tapi jika penekanannya pada konteks digunakan *collaboration diagram*. Gambar II. Berikut ini adalah contoh *collaboration* diagram.

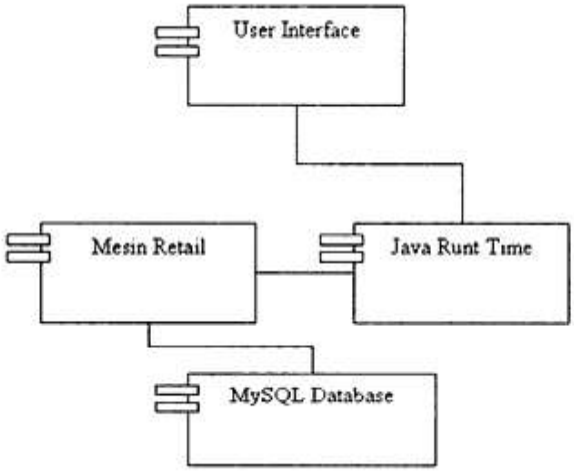


Gambar II.5 Contoh *Collaboration* Diagram

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:144

7. Component diagram

Menggambarkan struktur fisik kode dari komponen. Dapat berupa *source code*, komponen biner, atau *executable component*. Sebuah komponen berisi informasi tentang *logic class* atau *class* yang diimplementasikan sehingga membuat pemetaan dari *logical view* ke *component view*. Gambar II. Berikut ini adalah contoh *component* diagram.

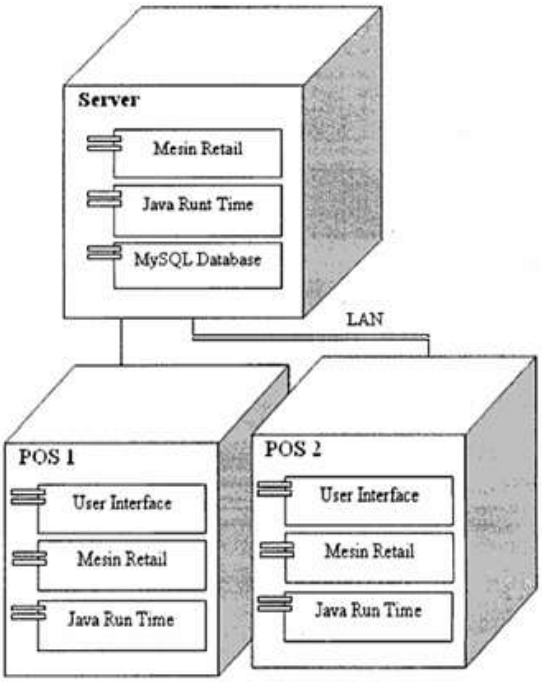


Gambar II.6 Contoh *Component* Diagram

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:145

8. Deployment diagram

Menggambarkan arsitektur fisik dari perangkat keras dan perangkat lunak sistem, menunjukkan hubungan komputer dengan perangkat (*nodes*) satu sama lain dan jenis hubungan nya. Didalam *nodes, executable component* dan *object* yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh node tertentu dan ketergantungan komponen. Gambar II. Brikut ini adlah contoh *deployment* diagram.



Gambar II.7 Contoh *Deployment* Diagram

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:146

II.3.2 Tujuan penggunaan UML

Tujuan penggunaan UML adalah sebagai berikut :

a. Memodelkan suatu sistem (bukan hanya perangkat lunak) yang menggunakn konsep berorientasi *object*.
b. Menciptakan suatu bahasa pemodelan yang dapat digunakan baik oleh manusia maupun mesin.

II.4 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.4.1 Konsep Database

Database (basis data) adalah sistem penyimpanan beragam jenis data dalam sebuah entitas yang besar untuk diolah sedemikian rupa agar mudah dipergunakan kembali. Dengan menggunakan komputer, konsep pengolahan database tradisional dapat diotomasi sehingga memudahkan pekerjaan. Data yang disimpan bisa sangat variatif (angka, teks, gambar, suara, dan jenis data multi-media lainnya).

Aplikasi manajemen database mengenal dua macam bentuk database:

1. *Flat-file* adalah semua record tersimpan dalam satu tabel.
2. Database relasional adalah memiliki banyak tabel yang saling terkait, dengan sebuah unsur data yg berfungsi sebagai pengait (*primary key*).

Dengan semakin banyaknya data yang dikelola, hampir tidak mungkin bahwa semua rekaman (*record*) disimpan dalau satu tabel. Manfaat database relasional adalah membuat sistem pengolahan data menjadi lebih efisien dan tabel data dapat dipilahkan dengan kategori yang berbeda. Fungsi *primary key* sangat penting dalam menemukan relasi dan logika kaitan antar tabel.

Pengimplementasian database dapat dilakukan secara terdistribusi dan tersentralisasi. Terdistribusi merupakan suatu konsep yang menerapkan lebih dari satu database. Dan tersentralisasi menerapkan satu database secara terpusat. Pada database terdapat tiga tingkatan atau level data, yaitu :

1. Level Fisik (*Phisycal Level*).

   Level fisik merupakan level paling rendah karena menggambarkan bagaimana data disimpan pada kondisi yang sebenarnya pada server.

2. Level Konsepsual (*Conceptual Level*).

   Merupakan level yang menggambarkan data yang disimpan pada database dan menjelaskan secara keseluruhan hubungan antar data tersebut.

3. Level Pandangan (*View Level*).

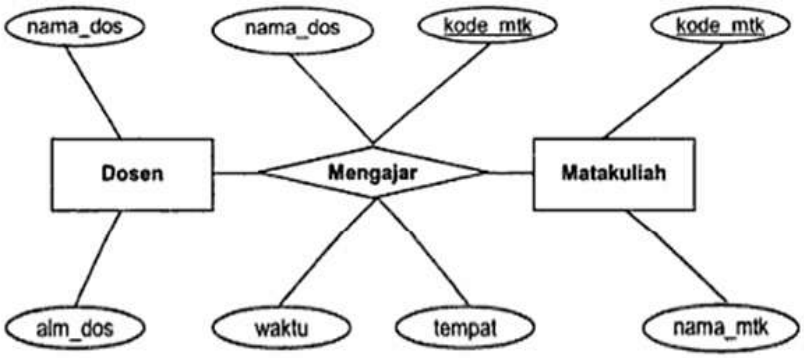
   Merupakan level yang menggambarkan sebagian dari seluruh database sesuai dengan kebutuhan pengguna.

II.4.2 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.4.3 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.4.4 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.4.5 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 (3rd 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.5 Bahasa Pemrograman Java

Hariyanto dalam Esensi-Esensi Bahasa Java (2007:1), bahasa pemrograman Java merupakan karya Suns Microsystem Inc. Rilis resmi level beta dilakukan pada November 1995. Dua bulan berikutnya Netscape menjadi perusahaan pertama yang memperoleh lisensi bahasa Java dari Sun. Java adalah bahasa pemrograman berorientasi objek yang berukuran kecil, sederhana dan aman, diinterpretasi atau dioptimalisasi secara dinamis, ber-*byte code*, arsitektur yang netral, mempunyai *garbage-collector*, *multithreading*, mempunyai mekanisme penanganan pengecualian, berbasis tipe untuk penulisan program mudah diperluas dinamis serta telah diperuntukan sistem tersebar.

Pada awal pembuatannya, Java dinamakan Oak, kemudian nama Oak dinilai kurang menjual sehingga pada Januari 1995 nama Oak diubah menjadi Java. Sebagai bahasa yang bersifat terbuka, Java didukun oleh banyak *programmer-programmer* dari seluruh dunia yang memberikan kontribusinya untuk mengembangkan bahasa Java.

II.6 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*.

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 :

1. Boolean, tipe data ini hanya boleh diisi oleh dua buah nilai yaitu true (benar) dan false (salah).
2. Byte, 0 sampai dengan 255
3. Char, tipe data ini hanya boleh diisi oleh sebuah karakter (Unicode).
4. Date, tipe data java yang merupakan nilai sebuah tanggal dan waktu, dengan jangkauan tanggal 1 januari 00001 sampai dengan 31 desember 9999.
5. 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).
6. Double, -1,79769313486231570E+308 sampai dengan 1, 79769313486231570E+308 (untuk bilangan positif).
7. Integer, -2.147.483.648 sampai dengan 2.147.483.647.
8. Long, -9.223.372.036.854.775.808 sampai dengan 9.223.372.-036.854.775.807.
9. Sbyte, -128 sampai dengan 127.
10. Short, -32,768 sampai dengan 32.767.
11. Single, -3,4028235E+38 sampai dengan -1,401298E-45(untuk bilangan negative) atau 1,401298E-45 sampai dengan 3,4028235E+38 (untuk bilangan positif).

12. String, 0 sampai dengan 2 juta karakter (*Unicode*) bisa huruf, angka atau karakter yang tidak umum lainnya.
13. UInteger, 0 sampai dengan 4.294.967.295.
14. ULong, 0 sampai dengan 18.446.744.073.709.551.615 (1.8…E+19).
15. UShort, 0 sampai dengan 65.535.