

BAB II

LANDASAN TEORI

2.1. Konsep Dasar Notifikasi

2.1.1. Definisi Notifikasi

Notifikasi adalah pemberitahuan mengenai informasi atau pengumuman dari pihak tertentu kepada pihak yang dituju yang dilakukan melalui media seperti *email*, *sms*, maupun aplikasi *chatting* (*line*, *whatsapp*, *bbm*, *wechat*, *dsb*). Notifikasi sangat penting bagi penerima informasi, karena dengan notifikasi, informasi tersebut dapat langsung diketahui oleh penerima. Notifikasi diperlukan ketika informasi yang akan disampaikan kepada pihak terkait tidak memungkinkan untuk diberitahukan secara langsung atau tatap muka. Notifikasi memungkinkan keakuratan atau ketepatan informasi yang diberikan.

2.2. Android

2.2.1. Definisi Android

Android merupakan sistem operasi berbasis *Linux* untuk telepon selular, *smartphone*, dan komputer tablet. Android adalah platform *open source* bagi para pengembang untuk menciptakan aplikasi mereka sendiri yang bisa di pakai bermacam-macam peranti bergerak.

Awal mulanya *Google.inc* membeli *Android.inc*, semua pihak menganggap remeh awal kehadiran Android, tetapi seiring berkembangnya zaman, Android kini menjadi penguasa pangsa pasar jajaran *smartphone* dan komputer tablet, karena Android bersifat *open source* pengembangannya begitu

cepat, Android kini telah memiliki berbagai versi dengan kemampuan yang mutakhir dan *up to date*. Android memulai debut pertamanya pada tahun 2007 dengan memperkenalkan *Nexus One*, setelah itu Android mengalami beberapa penambahan fitur di versi-versi Android selanjutnya, mulai dari Android 1.1 kemudian terjadi penambahan fitur di versi 1.5 (*Cupcake*), 1.6 (*Donut*), Android versi 2.0/2.1 (*Eclair*), Android 2.2 *Froyo* (*Frozen Yoghurt*), Android versi 2.3 (*Gingerbread*), Android versi 3.0/3.1 (*Honeycomb*) yang hanya dikembangkan untuk komputer tablet, Android 4.0 (*Ice Cream Sandwich*) yang dikembangkan untuk *smartphone* dan komputer tablet, Android versi 4.1 (*Jelly Beans*), dan versi terakhir saat ini adalah Android versi 5.0 (*lollipop*), begitupun seterusnya akan selalu mengalami perkembangan. (Safaat, 2012).

2.2.2. Android SDK (*Software Development Kit*)

Android SDK adalah tools API (*Application Programming Interface*) yang diperlukan untuk mengembangkan aplikasi pada platform Android, Android SDK menggunakan bahasa pemrograman *Java*, untuk lebih mengembangkan aplikasi Android, Android SDK disediakan untuk sistem operasi *Mac Os X* 10.4.8 atau lebih, *Windows XP*, *Vista*, 7 dan 8, serta *Linux* yang telah di *realese* oleh pihak *Google*.

2.2.3. Android Development Tool

Android Development Tools adalah *plugin* yang didesain untuk IDE *Eclipse* yang memberikan kita kemudahan dalam mengembangkan aplikasi Android. Dengan adanya ADT untuk *eclipse* akan memudahkan pengembangan dalam membuat aplikasi *project* Android, membuat GUI aplikasi, dan

menambahkan komponen-komponen yang lainnya, begitu juga kita dapat melakukan *running* aplikasi menggunakan Android SDK melalui *Eclipse*. Dengan ADT juga kita dapat membuat *package* Android (.apk) yang digunakan untuk distribusi aplikasi Android yang kita rancang.

2.2.4. SQLite

SQLite merupakan sebuah sistem manajemen basisdata relasional yang bersifat *ACID-compliant* dan memiliki ukuran pustaka kode yang relatif kecil, ditulis dalam bahasa C. Tidak seperti pada paradigma *client-server* umumnya, Inti SQLite bukanlah sebuah sistem yang mandiri yang berkomunikasi dengan sebuah program, melainkan sebagai bagian integral dari sebuah program secara keseluruhan. Sehingga protokol komunikasi utama yang digunakan adalah melalui pemanggilan API secara langsung melalui bahasa pemrograman. Mekanisme seperti ini tentunya membawa keuntungan karena dapat mereduksi *overhead*, *latency times*, dan secara keseluruhan lebih sederhana. Seluruh elemen basisdata (definisi data, tabel, indeks, dan data) disimpan sebagai sebuah *file*. Kesederhanaan dari sisi desain tersebut bisa diraih dengan cara mengunci keseluruhan *file* basis data pada saat sebuah transaksi dimulai.

Di Android juga memiliki fasilitas untuk membuat *database* yang dikenal dengan SQLite, SQLite adalah salah satu *software* yang *embedded* yang sangat populer, kombinasi SQL *interface* dan penggunaan *memory* yang sangat sedikit dengan kecepatan yang sangat cepat. SQLite di Android termasuk dalam *Android runtime*, sehingga setiap versi dari Android dapat membuat *database* dengan SQLite (Nazaruddin S :2012).

Dalam sistem Android memiliki beberapa teknik untuk melakukan penyimpanan data. Teknik yang umum digunakan adalah sebagai berikut:

- a. *Shared Preferences* yaitu menyimpan data beberapa nilai (*value*) dalam bentuk *groups key* yang dikenal dengan *preference*.
- b. *Files* yaitu menyimpan data dalam *file*, dapat berupa menulis ke *file* atau membaca dari *file*.
- c. *SQLite Databases*, yaitu menyimpan data dalam bentuk *databases*.
- d. *Content Providers*, yaitu menyimpan data dalam bentuk *content providers service*.

Pada skripsi ini fokus pada bagian *databases*. Tidak ada database yang disediakan oleh Android, jika menggunakan SQLite, harus meng-*create database* sendiri, mendefinisikan tabelnya, *index* serta datanya. Untuk membuat dan membuka database yang paling baik adalah menggunakan *libraries* :

```
import Android.database.sqlite.SQLiteOpenHelper;
```

Yang menyediakan tiga metode yaitu :

- 1) *Constructor*, menyediakan representasi versi dari *database* dan skema *database* yang kita gunakan.
- 2) *onCreate()*, menyediakan *SQLiteDatabase object* yang kita gunakan dalam definisi tabel dan inisialisasi data.
- 3) *onUpgrade()*, menyediakan fasilitas kovensi *databases* dari *database* versi lama ke *database* versi baru atau sebaliknya.

2.2.5. Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di *Sun Microsystems* saat ini merupakan bagian dari *Oracle* dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis *java* umumnya dikompilasi ke dalam *p-code(bytecode)* dan dapat dijalankan pada berbagai *Virtual Java Machine (JVM)*.

Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (*general purpose*) dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi *java* mampu berjalan di beberapa *platform* sistem operasi yang berbeda, *java* dikenal pula dengan slogannya, “Tulis sekali, jalankan di mana pun”. Saat ini *java* merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis *website*. *Java* adalah bahasa pemrograman yang *multiplatform* dan *multidevice*. Sekali anda menuliskan sebuah program dengan menggunakan *Java*, anda dapat menjalankannya hampir di semua komputer dan perangkat lain yang *support Java*, dengan sedikit perubahan atau tanpa perubahan sama sekali dalam kodenya.

Aplikasi berbasis *Java* ini dikompulasikan ke dalam *p-code* dan bisa dijalankan dengan *Java Virtual Machine*. Fungsionalitas dari *Java* ini dapat

berjalan dengan *platform* sistem operasi yang berbeda karena sifatnya yang umum dan non-spesifik. Kelebihan Java yang pertama tentu saja *multiplatform*. Java dapat dijalankan dalam beberapa *platform* komputer dan sistem operasi yang berbeda. Hal ini sesuai dengan slogannya yang sudah dibahas sebelumnya. Yang kedua adalah OOP atau *Object Oriented Programming*.

Java memiliki *library* yang lengkap. *Library* disini adalah sebuah kumpulan dari program yang disertakan dalam Java. Hal ini akan memudahkan pemrograman menjadi lebih mudah. Kelengkapan *library* semakin beragam jika ditambah dengan karya komunitas Java. Setiap hal pasti memiliki kelebihan dan kekurangan (Kadir, 2002).

2.2.6. XML (*Extensible Markup Language*)

XML (*Extensible Markup Language*) adalah bahasa *markup* untuk keperluan umum yang disarankan oleh W3C untuk membuat dokumen *markup* keperluan pertukaran data antar sistem yang beraneka ragam. XML merupakan kelanjutan dari HTML (*HyperText Markup Language*) yang merupakan bahasa standar untuk melacak Internet. Teknologi XML adalah teknologi keturunan dari SGML (*Standard Generalized Markup Language*, ISO 8879) yang dikembangkan pada tahun 1980-an. XML adalah suatu bahasa *Markup*, yaitu bahasa yang berisikan kode-kode berupa tanda-tanda tertentu dengan aturan tertentu untuk memformat dokumen teks dengan *tag* sendiri agar dapat dimengerti.

XML adalah *language* untuk mengidentifikasi dan menandai data terstruktur XML didesain untuk mampu menyimpan data secara ringkas dan mudah diatur. Kata kunci utama XML adalah data (jamak dari datum) yang jika

diolah bisa memberikan informasi. XML menyediakan suatu cara terstandarisasi namun bisa dimodifikasi untuk menggambarkan isi dari dokumen. Dengan sendirinya, XML dapat digunakan untuk menggambarkan sembarang *view database*, tetapi dengan suatu cara yang standar. XML memiliki tiga tipe *file*, diantaranya :

1. XML, merupakan standar format dari struktur berkas (*file*).
2. XSL, merupakan standar untuk memodifikasi data yang diimpor atau diekspor.
3. XSD, adalah standar yang mendefinisikan struktur *database* dalam XML.

Keunggulan XML bisa diringkas sebagai berikut :

1. Pintar (*Intelligence*). XML dapat menangani berbagai tingkat (*level*) kompleksitas.
2. Dapat beradaptasi. Dapat mengadaptasi untuk membuat bahasa sendiri. Seperti *Microsoft* membuat bahasa *MSXML* atau *Macromedia* mengembangkan *MXML*.
3. Mudah pemeliharaannya.
4. Sederhana. XML lebih sederhana.
5. Mudah dipindah-pindahkan (*Portability*). XML mempunyai kemudahan perpindahan (*portabilitas*) yang lebih bagus.

2.2.7. Eclipse

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

- a. *Multy-platform*: Target sistem operasi Eclipse adalah *Microsoft Windows, Linux, Solaris, AIX, HP-UX* dan *Mac OS X*.
- b. *Multy-language*: Eclipse dikembangkan dengan bahasa pemrograman *Java*, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti *C/C++, Cobol, Python, Perl, Php*, dan lain sebagainya.
- c. *Multy-role*: Selain sebagai IDE untuk pengembangan aplikasi, *Eclipse* pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, tes perangkat lunak, pengembangan *website*, dan lain sebagainya.

Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *open source*, yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari *Eclipse* yang membuatnya populer adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plugin* seperti penambahan *plugin* ADT untuk pengembangan Android dan aplikasi Android.

2.3. Unified Modelling Language (UML)

2.3.1. Definisi 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 kita dapat membuat 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 ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk *modeling* aplikasi prosedural dalam VB atau C.

UML merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem software yang terkait dengan obyek (Whitten, 2004). Jeffrey L. Whitten menambahkan bahwa UML menawarkan diagram yang dikelompokkan menjadi beberapa perspektif berbeda untuk memodelkan suatu sistem, seperti satu set cetak biru (*blueprint*) yang digunakan untuk membangun sebuah rumah (Whitten, 2004).

2.3.2. Diagram UML

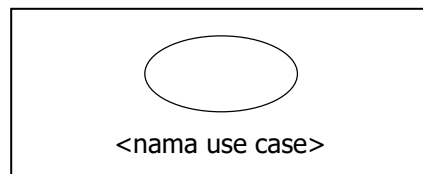
2.3.2.1. Use Case Diagram

Diagram yang menggambarkan interaksi antara sistem dengan eksternal sistem dan pengguna. Dengan kata lain, secara grafis menggambarkan siapa yang akan menggunakan sistem dan dengan cara apa pengguna mengharapkan untuk berinteraksi dengan sistem (Whitten, 2004). Adapun simbol-simbol dari *use case diagram* terdapat pada daftar simbol.

Dalam *use case diagram* memiliki pemodelan sebagai berikut (Whitten, 2004):

1. *Use case*

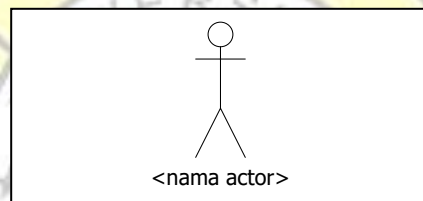
Use case merupakan urutan langkah-langkah yang secara tindakan saling terkait (skenario), baik otomatis maupun secara manual.



Gambar 3.1. Use Case (Whitten, 2004)

2. *Actor* (Pelaku)

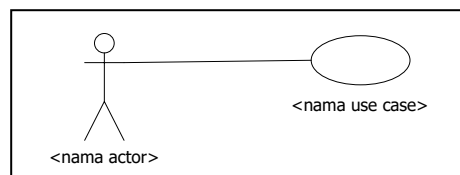
Actor merupakan pengguna yang perlu berinteraksi dengan sistem untuk pertukaran informasi.



Gambar 3.2. Actor (Whitten, 2004)

3. *Relationship* (Hubungan)

Pada diagram *use case*, *relationship* digambarkan sebagai sebuah garis antara dua simbol. Pemaknaan *relationship* berbeda-beda tergantung bagaimana garis tersebut digambar dan tipe simbol apa yang digunakan untuk menghubungkan garis tersebut.

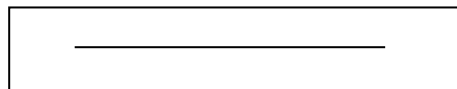


Gambar 3.3. Relationship (Whitten, 2004)

Berikut ini adalah perbedaan di antara *relationship* yang ada pada sebuah diagram *use case*:

a. *Association*

Association merupakan *relationship* antara *actor* dengan *use case* dimana terjadi interaksi di antara mereka.



Gambar 3.4. Association (Whitten, 2004)

b. *Extends*

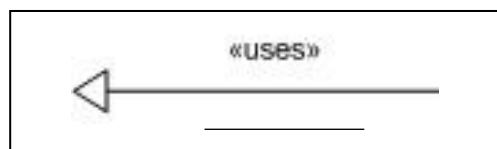
Extends use case merupakan *use case* yang terdiri dari langkah yang terekstraksi dari *use case* yang lebih kompleks untuk menyederhanakan masalah dan karena itu memperluas fungsinya.



Gambar 3.5 Extends (Whitten, 2004)

c. *Uses (includes)*

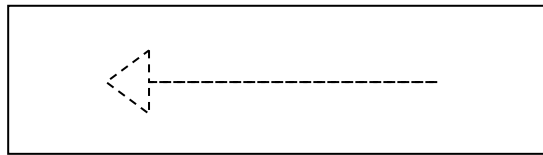
Hubungan *uses* menggambarkan bahwa satu *use case* seluruhnya meliputi fungsionalitas dari *use case* lainnya.



Gambar 3.6 Uses (Whitten, 2004)

d. *Depends on*

Terkadang suatu use case memiliki ketergantungan pada use case yang lainnya yang bertujuan untuk menentukan urutan dalam pengembangan use case. Ketergantungan ini dimodelkan menggunakan *depends on relationship*.



Gambar 3.7 Depends on (Whitten, 2004)

e. *Inheritance*

Hubungan *inheritance* terjadi ketika dua atau lebih actor menggunakan use case yang sama.



Gambar 3.8 Inheritance (Whitten, 2004)

2.3.2.2. Activity Diagram

Activity diagram secara grafis digunakan untuk menggambarkan rangkaian aliran aktifitas baik proses bisnis atau use case (Whitten, 2004). *Activity diagram* menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *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*).

	<i>State</i>
	<i>Control Flow</i>
	<i>Initial State</i>
	<i>Final State</i>
	<i>Transition</i>
	<i>Decision</i>
	<i>Swimlane</i>

Gambar 3.9 Komponen Activity Diagram (Whitten, 2004)

2.3.2.3. Sequence Diagram

Sequence diagram secara grafis menggambarkan bagaimana *object* berinteraksi dengan satu sama lain melalui pesan pada eksekusi sebuah *use case* atau operasi. Diagram ini mengilustrasikan bagaimana pesan terkirim dan diterima diantara *object* dan *sequence* (ruang waktu) (Whitten, 2004).

	<i>Object</i>
	<i>Actor</i>
	<i>Lifeline</i>
	<i>Message</i>
	<i>Message (Call)</i>
	<i>Message (Return)</i>
	<i>Activation</i>

Gambar 3. 10 Komponen Sequence Diagram (Whitten, 2004)

2.4. Basis Data

2.4.1. Definisi Basis Data

Database merupakan kumpulan *file* yang saling berhubungan. Akan tetapi, *database* tidak hanya kumpulan *file*. *Record* di dalam tiap *file* harus dapat dihubungkan dengan *record* di dalam *file* lain (Whitten, 2004). Dalam manajemen *database relational* terdapat komponen utama dalam konsep *database* (Whitten, 2004).

1. *Field* adalah unit terkecil data yang disimpan dalam *database*. Unit terkecil data yang disimpan dalam *database*:

- a. *Primary key* yaitu *field* yang unik dan mengidentifikasi satu *record*.
Contoh: *Customer Number* dan *Order Number*.
- b. *Secondary key* yaitu *field* yang mengidentifikasi sebuah *record* atau bagian dari beberapa *record* yang terkait.
- c. *Foreign key* yaitu *field* yang menunjuk beberapa *record* pada *file* lain.
Contoh: *Order Record* berisi *foreign key Customer Number*.
- d. *Descriptive field* yaitu *non-key field*.

2. *Record* adalah kumpulan *field* yang diatur dalam format yang *predetermined* (telah ditentukan).

- a. *Fixed-length record structures*

Sebagian besar teknologi *database* memaksakan struktur *record fixed length*, dalam artian setiap *instance record* mempunyai *field* yang sama, jumlah *field* yang sama, dan ukuran logika yang sama. Akan tetapi

beberapa sistem database akan mengkompresi *field-field* dan nilai-nilai yang tidak digunakan untuk menghemat ruang penyimpanan *disk*.

b. *Variable-length record structures*

Memperbolehkan *record-record* pada *file* yang sama memiliki *length* yang berbeda.

3. *File* dan Tabel

File adalah kumpulan semua kejadian dari struktur *record* yang ditentukan.

Tipe-tipe dari *file* yaitu:

- a. *File* induk/master adalah *file* penting dalam sistem dan akan tetap ada selama siklus hidup sistem informasi berputar.
- b. *File* transaksi adalah *file* yang digunakan untuk merekam data dari suatu transaksi yang terjadi.
- c. *File* laporan adalah *file* yang berisi sistem informasi yang akan ditampilkan.
- d. *File* sejarah adalah *file* yang berisi data masa lalu yang sudah tidak aktif lagi.
- e. *File* pelindung adalah salinan dari *file-file* yang masih aktif di *database* pada saat tertentu yang digunakan bila *file database* rusak.
- f. *File* kerja adalah suatu proses program secara sementara karena memori komputer tidak mencukupi.

Sedangkan tabel adalah suatu kesatuan unit dari *row / record* dengan atribut-atributnya (*column*).

2.4.2. DBMS (*Data Base Management System*)

DBMS (*Database Management System*) adalah kumpulan/gabungan database dengan perangkat lunak aplikasi yang berbasis database. DBMS merupakan koleksi terpadu dari database dan program-program komputer yang digunakan untuk mengakses dan memelihara *database*. Program-program tersebut menyediakan berbagai fasilitas operasi untuk memasukkan, melacak, dan memodifikasi data kedalam *database*, mendefinisikan data baru, serta mengolah data menjadi informasi yang dibutuhkan (Ladjamudin, 2005).

Tujuan utama DBMS adalah menyediakan lingkungan yang nyaman dan efisien untuk penyimpanan dan pengambilan data dari basisdata. DBMS berperan memberi abstraksi data tingkat tinggi ke pemakai. Sedangkan tujuan lain dari DBMS antara lain (Hariyanto, 2004):

1. Menghindari redudansi dan inkonsistensi data.
2. Menghindari kesulitan pengaksesan data.
3. Menghindari isolasi data.
4. Menghindari terjadinya anomali pengaksesan konkuren.
5. Menghindari masalah-masalah keamanan.
6. Menghindari masalah-masalah integritas.

Keunggulan DBMS (*Database Management System*) antara lain (Hariyanto, 2004):

1. Pengendalian terhadap redudansi data.
2. Konsistensi data.

3. Informasi yang lebih banyak yang dapat dibentuk dari data tersimpan yang sama.
4. Pemakaian bersama data.
5. Peningkatan integritas data.
6. Penyeimbangan kebutuhan-kebutuhan sumber daya yang terbatas.
7. Peningkatan pengaksesan dan daya tanggap data.
8. Peningkatan produktivitas.
9. Peningkatan pemeliharaan lewat ketidakbergantungan data.
10. Peningkatan konkurensi.
11. Peningkatan layanan backup dan pemulihan data.

2.5. XAMPP

2.5.1. Definisi XAMPP

XAMPP merupakan sebuah *web server multi-platform* yang bebas lisensi atau gratis, yang terdiri dari *Apache HTTP Server*, *database MySQL* dan penerjemah untuk *script* yang dituliskan dalam *Php* dan bahasa pemrograman *Perl*.

XAMPP merupakan akronim dari : X (yang berarti sistem operasi apa saja), *Apache*, *MySQL*, *Php*, *Perl*. Program ini sendiri dirilis berbasiskan GNU GPL (*General Public Liscense*) dan bersifat bebas (gratis).

2.5.2. PHP

Php adalah bahasa pemrograman yang berjalan pada sebuah *webserver*, atau sering disebut *server-side*. Oleh karena itu, *Php* dapat melakukan apa saja yang bisa dilakukan program CGI lain, yaitu mengolah data dengan tipe

apapun, menciptakan halaman web yang dinamis, serta menerima dan menciptakan *cookies*, dan bahkan *Php* bisa melakukan lebih dari itu. Pengertian *script server-side* adalah agar dapat menjalankan *script* ini dibutuhkan tiga program utama, yaitu *webserver* (dapat berupa IIS dari *windows* atau *apache*), modul *Php* dan juga *web browser*.

Php dapat berjalan pada semua jenis sistem operasi, antara lain pada *Linux* dan varian *Unix* (HP-UX, Solaris dan OpenBSD), pada *Ms Windows*, *Mac* dan masih banyak lagi. Selain itu *Php* juga dapat berjalan pada beberapa jenis *webserver* antara lain *Apache*, *Microsoft IIS*, *personal webserver*, *Netscape* dan *Iplanet Server*, *Caudium*, *Xitami*, *Omnihhttpd* dan masih banyak lagi. (Kadir, 2002).

2.5.3. MYSQL

MySQL adalah sebuah *server database open source* yang terkenal yang digunakan berbagai aplikasi terutama untuk *server* atau membuat *web service*. *MySQL* berfungsi sebagai *Structured Query Language (SQL)* yang dimiliki sendiri dan sudah diperluas oleh *MySQL* yang umumnya digunakan bersamaan dengan *Php* untuk membuat aplikasi *server* yang dinamis dan *powerfull*. Kehandalan suatu sistem basisdata (DBMS) dapat diketahui dari cara kerja dalam melakukan proses perintah-perintah *SQL* yang dibuat oleh pengguna maupun program-program aplikasi yang memanfaatkannya. Sebagai pengoperasi basis data, *MySQL* mendukung operasi basisdata transaksional maupun operasi basisdata non-transaksional.

Pada modus operasi non-transaksional, *MySQL* dapat dikatakan unggul dalam hal unjuk kerja dibandingkan perangkat lunak pengoperasi basisdata 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* (*wordpress*), 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 (Kadir, 2002).

2.6. Internet

2.6.1. Definisi Internet

Internet adalah sistem informasi global berbasis komputer. Internet terbentuk dari jaringan-jaringan komputer-komputer yang terkoneksi satu sama lain. Setiap jaringan menghubungkan puluhan, ratusan, hingga ribuan komputer, yang memungkinkan komputer-komputer itu saling berbagi data dan informasi satu sama lain dan untuk saling berbagi sumber daya komputasi yang mahal dan langka seperti komputer-komputer super (*Super Computer*) yang sangat kuat dan sistem-sistem basis data yang berisi informasi-informasi yang bermanfaat.

2.6.2. Perkembangan Internet

Penggunaan internet berkembang pesat sejak penemuannya. Berfungsi menghubungkan jaringan-jaringan secara terbatas pada tipe komputer tertentu, teknologi internet memungkinkan koneksi terjadi diantara berbagai jenis

komputer. Antar berbagai sistem operasi (tidak terikat platform perangkat keras maupun perangkat lunak). Tidak ada jaringan yang terlalu cepat atau lambat, terlalu besar atau terlalu kecil sehingga tidak bisa dikoneksikan. Internet dapat menghubungkan jaringan-jaringan sederhana yang terdiri dari beberapa komputer di suatu ruangan hingga jaringan-jaringan canggih yang merentang antar benua dan menghubungkan ribuan (bahkan jutaan) komputer.

