BAB II

TINJAUAN PUSTAKA

II.1. Sistem Informasi

Sistem informasi secara sederhana dapat diartikan sebagai kumpulan dari beberapa komponen yang saling berinteraksi untuk mencapai hasil dari satu tujuan. Pengertian sederhana ini sesuai dengan pendapat O'Brien (2006,p5)" Sistem Informasi dapat merupakan kombinasi teratur apapun dari orang-orang, *hardware, software*, jaringan komunikasi, data sumber daya data yang mengumpulkan, mengubah dan menyebarkan informasi dalam sebuah organisasi". Menurut Whitten (2004,p12) *" information system is an arrangement of people, data, process and information technology that interact to collect, process, store, and provide as output the information needed to support an organization*". Defenisi tersebut dapat dijelaskan sistem informasi adalah susunan dari orang, data, pemprosesan dan teknologi informasi yang dibutuhkan untuk mendukung sebuah organisasi (Henny Hendarti ; 2011:1)

II.1.1. Sistem Pakar

Dalam ilmu komputer, banyak ahli yang berkonsentrasi pada pengembangan kecerdasan buatan atau *Artificial Inteligence* (AI). AI adalah suatu studi khusus di mana tujuannya adalah membuat komputer berpikir dan bertindak seperti manusia. Banyak implementasi AI dalam bidang komputer, misalnya *Decision Support System* (Sistem Pendukung Keputusan), *Robotic*, *Natural Language* (Bahasa Alami), *Neural Network* (Jaringan Saraf), dan lain-lain.

Contoh bidang lain pengembangan kecerdasan buatan adalah sistem pakar yang menggabungkan pengetahuan dan penulusuran data untuk memecahkan masalah yang secara normal memerlukan keahlian manusia. Tujuan pengembangan sistem pakar sebenarnya bukan untuk menggantikan peran manusia, tetapi untuk mensubstitusikan pengetahuan manusia ke dalam bentuk sistem, sehingga dapat digunakan oleh orang banyak.

Ada banyak manfaat yang dapat diperoleh dengan mengembangkan sistem pakar, antara lain :

1. Masyarakat awam yang bukan seorang pakar dapat memanfaatkan keahlian di dalam bidang tertentu tanpa kehadiran langsung seorang pakar.
2. meningkatkan produktivitas kerja, yaitu bertambah efisiensi pekerjaan tertentu serta hasil solusi kerja.
3. penghematan waktu dalam meyelesaikan masalah yang kompleks.
4. Memberikan penyederhanaan solusi untuk kasus-kasus yang kompleks dan berulang-ulang.
5. pengetahuan dari seorang pakar dapat didokumentasikan tanpa ada batas waktu.
6. Memungkinkan penggabungan berbagai bidang pengetahuan dari berbagai pakar untuk dikombinasikan.

Berikut ini merupakan perbandingan antara kemampuan pakar manusia dan sistem komputer yang menjadi pertimbangan pengembangan sistem pakar (Tim Penerbit Andi : 2009 : 3).

Tabel II.1 Perbandingan Manusia Dengan Sistem Pakar

Pakar Manusia	Sistem Pakar
Terbatas waktu karena manusia membutuhkan istirahat.	Tidak terbatas karena dapat digunakan kapanpun juga.
Tempat akses bersifat lokal pada suatu tempat saja dimana pakar berada.	Dapat digunakan diberbagai tempat.
Pengetahuan bersifat variabel dan dapat berubah-ubah tergantung situasi.	Pengetahuan bersifat konsisten.
Kecepatan untuk menentukan solusi sifatnya bervariasi.	Kecepatan untuk memberikan solusi konsisten dan lebih cepat daripada manusia.
Biaya yang harus dibayar untuk konsultasi biasanya sangat mahal.	Biaya yang dikeluarkan lebih murah.

Sumber : Tim Penerbit Andi (2009 : 4)

Selain dari beberapa manfaat yang diperoleh, ada juga kelemahan pengembangan sistem pakar, yaitu :

1. Daya kerja dan produktivitas manusia menjadi berkurang karena semuanya dilakukan secara otomatis oleh sistem.
2. Pengembangan perangkat lunak sistem pakar lebih sulit dibandingkan dengan perangkat lunak konvensional. Hal ini dapat dilihat dari tabel perbandingan berikut ini :

Tabel II.2 Perangkat Lunak Konvensional Dengan Sistem Pakar

Perangkat Lunak Konvensional	Perangkat Lunak Sistem Pakar
Fokus pada solusi.	Fokus pada permasalahan.
Pengembangan dapat dilakukan secara individu.	Pengembangan dilakukan oleh tim kerja.
Pengembangan secara sekuensial.	Pengembangan secara interaktif.

Sumber : Tim Penerbit Andi (2009 : 4)

II.1.2. Metode *Forward Chaining*

Metode *Forward Chaining* adalah suatu metode pengambilan keputusan yang umum digunakan dalam sistem pakar. Proses pencarian dengan metode *Forward Chaining* berangkat dari kiri ke kanan, yaitu dari premis menuju kepada kesimpulan akhir, metode ini sering disebut data *driven* yaitu pencarian dikendalikan oleh data yang diberikan (Muhammad Arhami : 2005 : 111).

II.1.3. Dinamo Listrik

Michael Faradey adalah ilmuan inggris yang menemukan dinamo. Prinsip kerjanya secara umum adalah jika kawat didekatkan dengan arus listrik maka magnet akan bergerak .prinsip kerja yang sangat sederhana,tetapi sangat besar mamfaatnya untuk membantu manusia. Sebaliknya, magnet yang digerakkan juga akan menimbulkan arus listrik. Prinsip ini digunakan pada pembangkit listrik tenaga air (PLTA). Air digunakan untuk memutar kincir yang berisi magnet sehingnga muncul arus listrik.

a. Cara Keraja Dinamo DC

Kumparan kawat (armetur) berputar antara kutub-kutub dari magnet permanen.begitu satu sisi kumparan berjalan melewati kutub utara, ia akan memotong garis daya magnetik, dan ini menimbulkan arus, kumparan bergerak terus sedangkan arus mati. Kemudian kumparan memdekati kutub selatan dan menimbulkan arus pada arah yang berlawan, Kumparan daitempelkan pada komutator yang menyebabkan arus sirkuit luar mengalir dengan arah yang sama sepanjang waktu (Badiatul Muchlisin Asti dan Junaidi Abdul Munif : 2009 : 200).

b. Cara Kerja Generator Atau Dinamo AC

A.C dihasilkan oleh alternator,sejenis generator, tidak seperti dinamo d.c, ia tidak memiliki komulataoruntuk membalik hubungan didalam sirkuit. Sebagai gantinya, sikat menekan terus-menerus cincin slip. Saat armature berputar arus yang ditimbulkan mengalir satu arah untuk setengah putaran (sebahagian dari kumparan melewati kutub utara), kemudian mengalir kearah yang berlawanan pada setengah berikutnya (saat aia melewati kutub selatan). Sehingga rotasi menentukan kecepatan arus berbalik (Subakti ; 2005 : 37).

II.2. Unified Modeling Language (UML)

UML (*Unified Modeling Language*) adalah suatu alat Bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek (Munawar ; 2005 : 17). Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembangan sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain.

Meskipun UML sudah banyak menyediakan diagram yang bisa membantu mendefenisikan suatu aplikasi, tidak berarti bahwa semua diagram tersebut akan bisa menjawab persoalan yang ada. Adapun tipe diagram UML yang ada seperti pada Tabel II.3.

Tabel II.3 Tipe Diagram UML

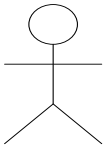
Diagram	Tujuan	Keterangan
Activity	Prilaku prosedural dan parallel	Sudah ada di UML 1
Class	*Class*, fitur dan relasinya	Sudah ada di UML 1
Communication	Interaksi diantara objek. Lebih menekankan kepada link	Di UML 1 disebut *collaboration*
Component	Struktur dan koneksi dari komponen	Sudah ada di UML 1
Composite Structure	Dekomposisi sebuah *class* saat *runtime*	Baru untuk UML 2
Deployment	Penyebaran/instalasi ke klien	Sudah ada di UML 1
Interaction Overview	Gabungan dari *activity* dan *sequence diagram*	Baru untuk UML 1
Object	Contoh konfigurasi *instance*	Tidak resmi ada di UML 1
Package	Struktur hierarki saat kompilasi	Tidak resmi ada di UML 1
Sequence	Interaksi antara objek. Lebih menekankan pada urutan.	Sudah ada di UML 1
State Machine	Bagaimana *event* mengubah sebuah objek	Sudah ada di UML 1
Timing	Interaksi antar objek. Lebih menekankan pada waktu	Sudah ada di UML 1
Use Case	Bagaimana user berinteraksi dengan sebuah sistem	Sudah ada di UML 1

Sumber :Munawar (2009 : 23)

II.3 Notasi Dasar UML

II.3.1. *Actor*

Actor adalah *abstraction* dari orang dan *system* yang lain yang mengaktifkan fungsi dari target *system*. Orang atau system bisa muncul dalam bebrapa peran. Perlu dicatat bahwa *actor* berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case*. Berikut notasi *actor* dalam UML :



Gambar II.2 : Notasi *Actor* pada UML

Sumber : Munawar (2009 : 64)

II.3.2. *Class*

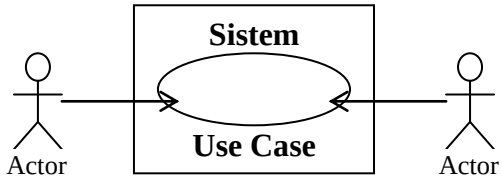
Class, dalam notasi UML digambarkan dengan kotak. Nama class menggunakan huruf besar diawal kalimatnya dan diletakkan diatas kotak. Bila *class* mempunyai nama yang terdiri dari 2 suku kata atau lebih, maka semua suku kata digabungkan tanpa spasi dengan huruf awal tiap suku kata menggunakan huruf besar. Berikut notasi *class* dalam UML:

Gambar II.3 : Notasi *Class* di UML

Sumber : Munawar (2009 : 35)

II.3.3. *Use Case*

Use Case adalah alat bantu terbaik guna menstimulasi pengguna potensial untuk mengatakan tentang suatu *system* dari sudut pndangnya. Tidak selalu mudah bagi pengguna untuk menyatakan bagaimana mereka bermaksud menggunakan sebuah *system*. Karena *system* pengembangan tradisional sering ceroboh dalam melakukan analisis, akibatnya pengguna seringkali susah menjawabnya tatkala dimintai masukan tentang sesuatu. Notasi *use cas*e dapat dilihat pada gambar II.4 :



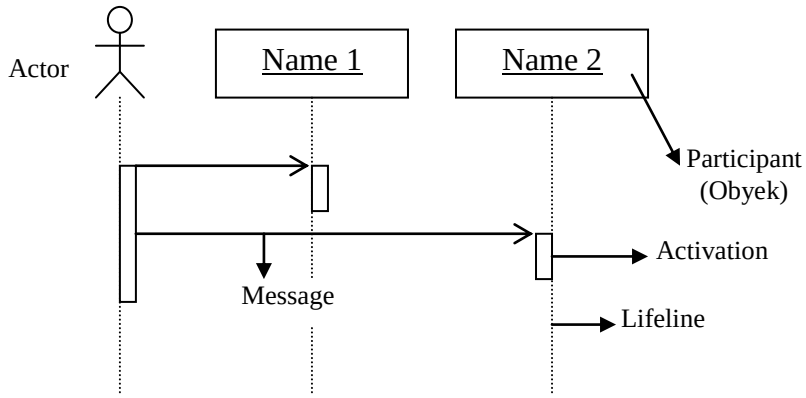
Gambar II.4 : Notasi *Use Case* pada UML

Sumber : (Munawar ; 2009 : 64)

II.3.4. *Sequence Diagram*

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah scenario. Diagram ini menunjukkan sejumlah contoh obyek dan *message* (pesan) yang diletakkan diantara obyek-obyek ini dalam *use case*.

Komponen utama squence diagram terdiri atas obyek yang dituliskan dengan kotak segiempat bernama. Message dengan tanda panah dan waktu yang ditunjukkan dengan progress vertical. Berikut Contoh *sequence diagram* :



Gambar II.5 : Simbol-simbol yang ada pada *Sequence Diagram*

Sumber : Munawar (2009 : 89)

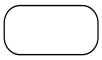
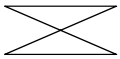
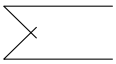
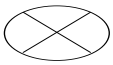
II.3.5. *Activity Diagram*

Activity diagram adalah teknik untuk mendiskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity Diagram* mempunyai

peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *activity diagram* bisa mendukung prilaku paralel sedangkan *flowchart* tidak bisa.

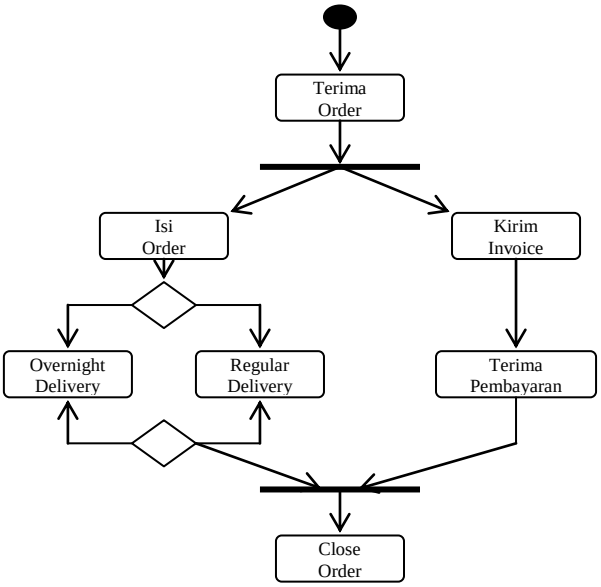
Berikut adalah simbol-simbol yang sering digunakan pada saat pembuatan *activity diagram*.

Tabel II.4 Simbol-simbol yang sering dipakai pada *activity diagram*

Simbol	Keterangan
	Titik awal
	Titik akhir
	*Activity*
	Pilihan untuk pengambilan keputusan
	Fork; digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.
	*Rake*; menunjukkan adanya dekomposisi
	Tanda waktu
	Tanda pengiriman
	Tanda penerimaan
	Aliran akhir (*Flow Final*)

Sumber : Munawar (2009 : 109)

Adapun contoh dari *Activity Diagram* dapat di lihat pada Gambar II.6.



Gambar II.6 : Contoh *Activity Diagram* Sederhana

Sumber : Munawar (2009 : 111)

II.4. Pengertian Database

Database merupakan komponen terpenting dalam pembangunan SI, karena menjadi tempat untuk menampung dan mengorganisasikan seluruh data yang ada dalam sistem, sehingga dapat dieksplorasi untuk menyusun-menyusun informasi-informasi dalam berbagai bentuk. *Database* merupakan himpunan kelompok data yang saling berkaitan. Data tersebut diorganisasikan sedemikian rupa agar tidak terjadi duplikasi yang tidak perlu, sehingga dapat diolah atau di eksplorasi secara cepat dan mudah untuk menghasilkan informasi (Budi Sutedjo Dharma Oetomo; 2006: 99).

II.4.1. Hierarki Data Dalam Database

Data dalam sebuah *database* disusun berdasarkan sistem hierarki yang unik, yaitu:

1. ***Database***, merupakan kumpulan file yang saling terkait satu sama lain, misalnya file data induk karyawan, file jabatan file penggajian dan lain sebagainya. Kumpulan file yang tidak saling terkait satu sama lain tidak dapat disebut *database*, misalnya file data induk karyawan, file tamu undangan perkawinan, file barang retail pasar swalayan.
2. ***File***, yaitu kumpulan dari *record* yang saling terkait dan memiliki format *field* yang sama dan sejenis.
3. ***Record***, yaitu kumpulan *field* yang menggambarkan suatu unit data individu tertentu.
4. ***Field***, yaitu atribut dari *record* yang menunjukkan suatu item dari data, seperti nama, alamat, dan lain sebagainya.
5. ***Byte***, yaitu atribut dari *field* yang berupa huruf yang membentuk nilai dari sebuah *field*. Huruf tersebut dapat berupa numerik maupun abjad atau karakter khusus.
6. ***Bit***, yaitu bagian terkecil dari data secara keseluruhan, yaitu berupa karakter ASCII nol atau satu yang merupakan komponen pembentuk *byte* (Budi Sutedjo Dharma Oetomo; 2006: 102).

II.4.2. MySQL (phpMyAdmin)

PhpMyAdmin adalah aplikasi berbasis web yang dibuat dari pemograman PHP dan *Java Script*. PhpMyAdmin juga dapat disebut sebagai *tools* yang berguna untuk mengakses *database MySQL Server* dalam bentuk tampilan *web*. dengan adanya phpMyAdmin, semua pekerjaan menjadi mudah, karena tanpa

harus mengerti perintah-perintah dasar SQL namun sudah dapat memanajemen *database* dan data yang ada didalamnya (Bunafit Nugroho; 2009: 13).

II.5. Kamus Data

Kamus data (KD) atau *data dictionary* (DD) atau disebut juga dengan istilah *systems data dictionary* adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Dengan menggunakan KD, analisis sistem dapat mendefinisikan data yang mengalir di sistem dengan lengkap. KD dibuat pada tahap analisis sitem dan digunakan baik pada tahap analisis maupun pada tahap perencanaan sistem (Jogiyanto; 2005: 725).

Tabel II.5 Notasi Kamus Data

Notasi	Arti
=	Terbentuk dari (*is composed*) atau terdiri dari (*consist of*) atau sama dengan (*is equivalent of*)
+	*AND*
[]	Salah satu dari (memilih salah satu dari elemen-elemen data di dalam kurung *bracke*t ini)
|	Sama dengan simbol []
M{ }M	Intensi (elemen data didalam kurung *brace* berinterasi mulai minimum N kali dan maksimum M kali)
()	*Optional* (elemen data di dalam kurung *parenthesis* sifatnya *optional*, dapat ada dan dapat tidak ada)
*	Keterangan setelah tanda ini adalah komentar

Sumber : Jogiyanto (2005: 730)

II.6. *Entity Relationship Diagram* (ERD)

II.6.1. Model-model Data

Struktur yang mendasari suatu basis data adalah model data yang merupakan kumpulan alat-alat konseptual untuk mendeskripsikan data, relasi data, data semantik, dan batasan konsistensi. Untuk mengilustrasikan konsep model data, berikut disajikan dua model data, yaitu *entity relationship model* dan

relational model. Kedua model menyediakan cara mendeskripsikan rancangan basisdata pada tingkatan logis.

II.6.2. *Entity Relationship* Model (ERM)

Entity Relationship (ER) data model didasarkan pada persepsi terhadap dunia nyata yang tersusun atas kumpulan objek-objek dasar yang disebut *entitas* dan hubungan antar objek. *Entitas* adalah sesuatu atau objek dalam dunia nyata yang dapat dibedakan dari objek lain. Sebagai contoh, masing-masing mahasiswa adalah *entitas* dan mata kuliah dapat pula dianggap sebagai *entitas*.

Entitas digambarkan dalam basisdata dengan kumpula atribut. Misalnya atribut nim, nama, alamat dan kota bisa menggambarkan data mahasiswa tertentu dalam suatu universitas. Atribut-atribut membentuk *entitas* mahasiswa. Demikian pula, atribut kodeMK, namaMK, dan SKS mendeskripsikan *entitas* mata kuliah.

Atribut NIM digunakan untuk mengidentifikasi mahasiswa secara unik karena dimungkinkan terhadap dua mahasiswa dengan nama, alamat, dan kota yang sama. Pengenal unik harus diberikan pada masing-masing mahasiswa.

Relasi adalah hubungan antara beberapa *entitas*. Sebagai contoh, relasi menghubungkan mahasiswa dengan mata kuliah yang di ambilnya. Kumpulan semua *entitas* bertipe sama disebut kumpulan entitas (*entity set)*, sedangkan kumpulan semua relasi bertipe sama disebut kumpulan relasi (*relationship set)*.

Struktur logis (skema *database*) dapat ditunjukkan secara grafis dengan diagram ER yang dibentuk dari komponen-komponen berikut :

Tabel II.6 Notasi ERD (*Entity Relationship Diagram*)

	Persegi panjang mewakili kumpulan entitas
	Elips mewakili atribut
	Belah ketupat mewakili relasi
	Garis menghubungkan atribut dengan kumpulan entitas dan kumpulan entitas dengan relasi

Sumber : Janner Simarmata & Imam Prayudi (2006 :59)

II.6.3. Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basis data relasional. Pada dasarnya, normalisasi adalah proses dua langkah yang meletakkan data dalam bentuk tabulasi dengan menghilangkan kelompok berulang lalu menghilangkan data yang terduplikasi dari tabel relational.

Teori normalisasi didasarkan pada konsep bentuk normal. Sebuah tabel relasional dikatakan berada pada bentuk normal tertentu jika tabel memenuhi himpunan batasan tertentu. Ada lima bentuk normal yang telah ditemukan.

1. **Bentuk Normal Pertama (1NF/*First Normal Form*)**, bentuk normal pertama adalah suatu bentuk relasi dimana atribut bernilai banyak (*multivalues atribute*) telah dihilangkan sehingga kita akan menjumpai nilai tunggal (mungkin saja nilai *null*) pada perpotongan setiap baris dan kolom satu nilai untuk irisan baris dan kolom pada tabel.

2. **Bentuk Normal Kedua (2NF/*Second Normal Form*)**, semua kebergantungan fungsional (*functional dependeny*) yang bersifat sebagian (*partial functional dependeny*) telah dihilangkan.
3. **Bentuk Normal Ketiga (3NF/*Third Normal Form*)**, semua kebergantungan transitif (*transitive dependeny*) telah dihilangkan.
4. **Boyce-Codd Normal Form (BCNF/*Boyce-Codd Normal Form*)**, semua anomali yang tersisa dari hasil penyempurnaan kebergantungan fungsional (*functional dependeny*) diatas telah dihilangkan.
5. **Bentuk Normal Keempat (4NF/*Fifth Normal Form*)**, semua anomali yang berasal dari kebergantungan banyak-nilai (*multivalues dependeny*) telah dihilangkan (Adi Nugroho; 2010: 34).

Tujuan normalisasi adalah membuat kumpulan tabel relasional yang bebas dari data berulang yang dapat dimodifikasi secara benar dan konsisten. Ini berarti bahwa semua tabel pada basis data relasional harus berada pada bentuk normal ketiga (3NF). Sebuah tabel relasional berada pada 3NF jika dan hanya jika semua kolom bukan kunci adalah (a) saling independen dan (b) sepenuhnya tergantung pada kunci utama. Saling independen berarti bahwa tidak ada kolom bukan kunci yang tergantung pada senbarang kombinasi kolom lainnya. Dua bentuk normal pertama adalah langkah antara untuk mencapai tujuan, yaitu mempunyai semua tabel dalam 3NF (Janner Simarmata & Imam Prayudi; 2006: 77).

II.7. Bahasa Pemrograman *Java*.

Java lahir karena ketidakpuasan seorang insinyur di *SUN Micro System* bernama James Gosling. Ia tidak puas dengan kompiler C++ (yang ia gunakan

untuk membuat *software* yang di-*embed* pada peralatan elektronik) karena dinilai terlalu banyak menghasilkan *bug*, berbiaya besar, sangat bergantung terhadap *platform.* Gosling merasa perlu membuat kompiler baru sebagai solusi terhadap sejumlah kelemahan pada C++ tersebut.

Kompiler baru tersebut diberi nama dengan *Oak*. Kompiler ini mirip dengan C++ tetapi dengan sejumlah pengurangan fitur yang dianggap kurang menguntungkan dalam pengembangan, seperti *multipel inheritance*, konversi tipe secara otomatis, penggunaan pointer dan manajemen memori.

Pada tahun 1994, Oak diubah namanya menjadi *Java*. Pada era ini, *java* divisikan sebagai bahasa yang memiliki dukungan baik terhadap *web* (Rijalul Fikri, dkk; 2005: 19)

Java merupakan pemrograman berorientasi objek dan bebas *platform*, dikembangkan oleh *SUN Micro System* dengan sejumlah keunggulan yang memungkinkan *Java* dijadikan sebagai bahasa pengembangan enterprise (Rijalul Fikri, dkk; 2005: 15)