BAB II

TINJAUAN PUSTAKA

II.1 Pengertian Sistem

Definisi sistem berkembang sesuai dengan konteks dimana pengertian sistem itu digunakan. Berikut akan diberikan beberapa defenisi 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)

II.2 Pengertian Informasi

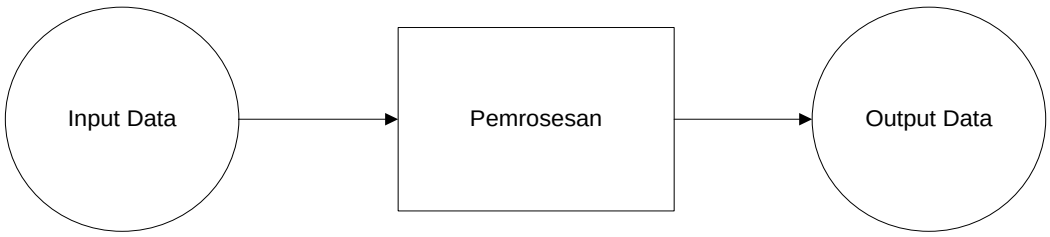
Untuk memahami pengertian sistem informasi, harus dilihat keterkaitan antara data dan informasi sebagai entitas penting pembentuk sistem informasi. Data merupakan nila i, keadaan, atau sifat yang berdiri sendiri lepas dari konteks apapun. Sementara informasi adalah data yang telah diolah menjadi sebuah

bentuk yang berarti bagi penerimanya dan bermanfaat dalam pengambilan keputusan saat ini atau mendatang. (Hanif Al Fatta; 2007: 9)

II.3 Pengertian Sistem Informasi

Sistem Informasi didefinisikan sebagai suatu alat untuk menyajikan informasi dengan cara sedemikian rupa sehingga bermanfaat bagi penerimanya. Tujuannya adalah untuk menyajikan informasi guna pengambilan keputusan pada perencanaan, pemrakarsaan, pegorganisasian, pengendalian kegiatan operasi subsistem suatu perusahaan, dan menyajikan sinergi organisasi pada proses. (Hanif Al Fatta; 2007: 9)

Dengan demikian, sistem informasi berdasarkan konsep (*input, processing, output - IPO*) dapat dilihat pada gambar II.1 :



Gambar II.1 Konsep Sistem Informasi
Sumber : Hanif Al Fatta (2007: 9)

II.4 Tipe-Tipe Sistem Informasi

Sistem informasi dibedakan menjadi beberapa tipe aplikasi, yaitu :

a. *Transaction Processing System* (TPS)

   Transaction Processing System (TPS) atau Sistem Pemrosesan Transaksi adalah sistem informasi terkomputerisasi yang dikembangkan untuk memproses sejumlah besar data untuk transaksi bisnis rutin.

b. Management Information System (MIS)

Management Information System (MIS) atau Sistem Informasi Manajemen (SIM) adalah sebuah sistem informasi pada level manajemen yang berfungsi untuk membantu perencanaan, pengendalian, dan pengambilan keputusan dengan menyediakan resume rutin dan laporan-laporan tertentu. SIM mengambil data mentah dari TPS dan mengubahnya menjadi kumpulan data yang lebih berarti yang dibutuhkan manajer untuk menjalankan tanggung jawabnya. Untuk mengembangkan suatu SIM, diperlukan pemahaman yang baik tentang informasi apa saja yang dibutuhkan manajer dan bagaimana mereka menggunakan informasi tersebut.

c. *Decission Support System* (DSS)

Decission Support System (DSS) atau Sistem Pendukung Keputusan (SPK) merupakan sistem informasi pada level manajemen dari suatu organisasi yang mengombinasikan data dan model analisis canggih atau peralatan data analisis untuk mendukung pengambilan yang semi terstruktur dan tidak terstruktur. DSS dirancang untuk membantu pengambilan keputusan organisasional.

d. *Expert System* (ES)

Expert System (ES) merupakan representasi pengetahuan yang menggambarkan cara seorang ahli dalam mendekati suatu masalah. ES lebih berpusat pada bagaimana mengodekan dan memanipulasi pengetahuan dari informasi (misalnya aturan if..then). (Hanif Al Fatta; 2007: 12-13)

II.5 Sistem Informasi Geografis

Defenisi SIG (kemungkinan besar) masih berkembang, bertambah dan sedikit bervariasi. Hal ini terlihat dari banyaknya defenisi SIG yang telah beredar diberbagai sumber pustaka. Sehubungan dengan hal ini, sebagai ilustrasi, berikut adalah beberapa defenisi SIG yang telah beredar :

1. SIG adalah sistem komputer yang digunakan untuk memasukkan (*capturing*), menyimpan, memeriksa, mengintegrasikan, memanipulasi, menganalisis, dan menampilkan data-data yang berhubungan dengan posisi-posisinya di permukaan bumi.
2. SIG adalah kombinasi perangkat keras dan perangkat lunak sistem komputer yang memungkinkan penggunanya untuk mengelola, menganalisa, dan memetakan informasi spasial berikut data atributnya (data deskriptif) dengan akurasi kartografis.
3. SIG adalah sistem yang berbasiskan komputer yang digunakan untuk menyimpan dan memanipulasi informasi-informasi geografis, SIG dirancang untuk mengumpulkan, menyimpan, dan menganalisis objek-objek dan fenomena dimana lokasi geografis merupakan karakteristik yang penting atau kritis untuk di analisis. Dengan demikian SIG merupakan sistem komputer yang memiliki 4 kemampuan berikut dalam menangani data yang bereferensi geografis : (a) masukan (b) manajemen data (penyimpanan dan pemanggilan data) (c) analisis dan manipulasi data, dan (d) keluaran. (Eddy Prahasta; 2009: 116)

II.5.1 Sub Sistem Sistem Informasi Geografis

SIG dapat diuraikan menjadi beberapa sub sistem sebagai berikut :

a. Data *Input* : sub sistem ini bertugas untuk mengumpulkan, mempersiapkan, dan menyimpan data spasial dan atributnya dari berbagai sumber. Sub sistem ini pula yang bertanggung jawab dalam mengonversikan atau mentransfromasikan format-format data aslinya ke dalam format *native* yang dapat digunakan oleh perangkat SIG bersangkutan.

b. Data *Output* : sub sistem ini bertugas untuk menampilkan atau menghasilkan keluaran (termasuk meng-ekspornya ke format yang dikehendaki) seluruh atau sebagian basis data (spasial) baik dalam bentuk *softcopy* maupun hardcopy seperti halnya tabel, grafik, report, peta dan lain sebagainya.

c. Data *Management* : sub sistem ini mengorganisasikan baik data spasial maupun tabel-tabel atribut terkait kedalam sebuah sistem basis data sedemikian rupa sehingga mudah dipanggil kembali atau di *retrieve* (di load ke memori), di *update* dan di edit.

d. Data *Manipulation* & *Analysis* : sub sistem ini menentukan informasi-informasi yang dapat dihasilkan oleh SIG. selain itu, sub sistem ini juga melakukan manipulasi (evaluasi dan penggunaan fungsi-fungsi dan operator matematis dan logika) dan permodelan data untuk menghasilkan informasi yang diharapkan.(Eddy Prahasta; 2009: 118)

II.5.2 Komponen Sistem Informasi Geografis

SIG merupakan salah satu sistem yang kompleks dan pada umumnya juga (selain yang *stand-alone*) terintegrasi dengan lingkungan sistem komputer lainnya

di tingkat fungsional dan jaringan (*network*), jika diuraikan SIG sebagai sistem terdiri dari beberapa komponen sebagai berikut dengan berbagai karakteristik :

a. Perangkat Keras

   Pada saat ini SIG sudah tersedia bagi berbagai platform perangkat keras, mulai dari kelas PC *desktop*, *workstations*, hingga *multi user host* yang bahkan dapat digunakan oleh banyak orang secara bersamaan (simultan) dalam jaringan komputer yang luas, tersebar, berkemampuan tinggi, memiliki ruang penyimpanan yang besar, dan mempunyai kapasitas memori yang besar.

b. Perangkat Lunak

   Dari sudut pandang yang lain, SIG bisa juga merupakan sistem perangkat lunak yang tersusun secara modular dimana sistem basis datanya memegang peranan kunci.

c. Data & Informasi Geografi

   SIG dapat mengumpulkan dan menyimpan data dan informasi yang diperlukan baik secara tidak langsung (dengan cara meng-*import*-nya dari format-format perangkat lunak SIG yang lain) maupn secara langsung dengan cara digitasi data spasialnya (digitasi *on-screean* atau *heads-ups* di atas tampilan layar monitor, atau manual dengan menggunakan *digitizer*) dari peta analog dan kemudian memasukkan data attributnya dari tabel-tabel atau laporan dengan menggunakan *keyboard*.

d. Manajemen

Suatu proyek SIG akan berhasil jika dikelola dengan baik dan dikerjakan oleh orang-orang yang memiliki keahlian (kesesuaian dengan *job-description* yang bersangkutan) yang tepat pada semua tingkatan.

II.6 SDLC

SDLC atau *Software Development Life Cycle* atau sering disebut juga *System Development Life Cycle* adalah proses pengembangan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model atau metodologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya (berdasarkan *best practice* atau cara-cara yang sudah teruji baik).

Tahapan-tahapan yang ada pada SDLC secara global adalah sebagai berikut :

a. Inisiasi (*initiation*)
b. Pengembangan konsep sistem (*system concept development*)
c. Perencanaan (*planning*)
d. Analisis kebutuhan (*requirement analysis*)
e. Desain (design)
f. Pengembangan (*development*)
g. Integrasi dan pengujian (*integration and test*)
h. Implemenasi (*implementation*)
i. Operasi dan pemeliharaan (operations and maintenance)
j. Disposisi (dispotition) (Rosa A.S dan M. Shalahuddin; 2011: 24-26)

SDLC memiliki beberapa model dalam penerapan tahapan prosesnya. Antara lain sebagai berikut:

1. Model *Waterfall* (Air Terjun)

   Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengkodean, pengujian dan tahap pendukung (*support*).

2. Model *Prototipe*

   Model *prototipe* (*Prototyping Model*) dimulai dari mengumpulkan kebutuhan pelanggan terhadap perangkat lunak yang akan dibuat. Lalu dibuatlah program prototipe agar pelanggan lebih terbayang dengan apa yang sebenarnya diinginkan. Program prototipe ini dievaluasi oleh pelanggan atau user sampai ditemukan spesifikasi yang sesuai dengan keinginan pelanggan atau user.

3. Model *Rapid Application Development* (RAD)

   Adalah model proses pengembangan perangkat lunak yang bersifat inkremental terutama untuk waktu pengerjaan yang pendek.

4. Model iteratif

   Model ietratif mengkombinasi proses-proses pada air terjun dan iteratif pada model prototipe.

5. Model spiral

   Model spiral memasangkan iteratif pada model prototipe dengan kontrol dan aspek sistematik yang diambil dari model air terjun. (Rosa A.S dan M. Shalahuddin; 2011: 26-37)

II.7 Analisis Sistem

Analisis sistem adalah sebuah istilah yang secara kolektif mendeskripsikan fase-fase awal pengembangan sistem. Analisis sistem adalah teknik pemecahan masalah yang menguraikan bagian-bagian komponen dengan mempelajari seberapa bagus bagian-bagian komponen tersebut bekerja dan berinteraksi untuk mencapai tujuan mereka. (Hanif Al Fatta; 2008 : 44)

Desain sistem adalah sebuah teknik pemecahan masalah yang saling melengkapi (dengan analisis sistem) yang merangkai kembali bagian-bagian komponen menjadi sistem yang lengkap-harapannya, sebuah sistem yang diperbaiki. Hal ini melibatkan penambahan, penghapusan, dan perubahan-perubahan bagian relatif pada sistem awal (aslinya). (Hanif Al Fatta; 2008 : 44)

Analisis sistem informasi terbagi menjadi tiga tahap analisis, yaitu :

1. Analisis kelemahan sistem yang sedang berjalan
2. Analisis kebutuhan sistem baru
3. Analisis kelayakan sistem yang meliputi kelayakan teknik, kelayakan hukum, ekonomi, operasional, dan lain-lain. (Hanif Al Fatta; 2008 : 47)

II.8 Pemodelan dengan UML

Pemodelan adalah gambaran dari realita yang simpel dan dituangkan dalam bentuk pemetaan dengan aturan tertentu. (Rosa A.S dan M. Shalahuddin; 2011: 116)

Pada perkembangan teknik pemrograman beroientasi objek, muncullah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modeling Languange* (UML). *UML* hanya bergungsi untuk melakukan pemodelan. Jadi penggunaan *UML* tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya *UML* paling banyak digunakan pada metodologi berorientasi objek. (Rosa A.S dan M. Shalahuddin; 2011: 118)

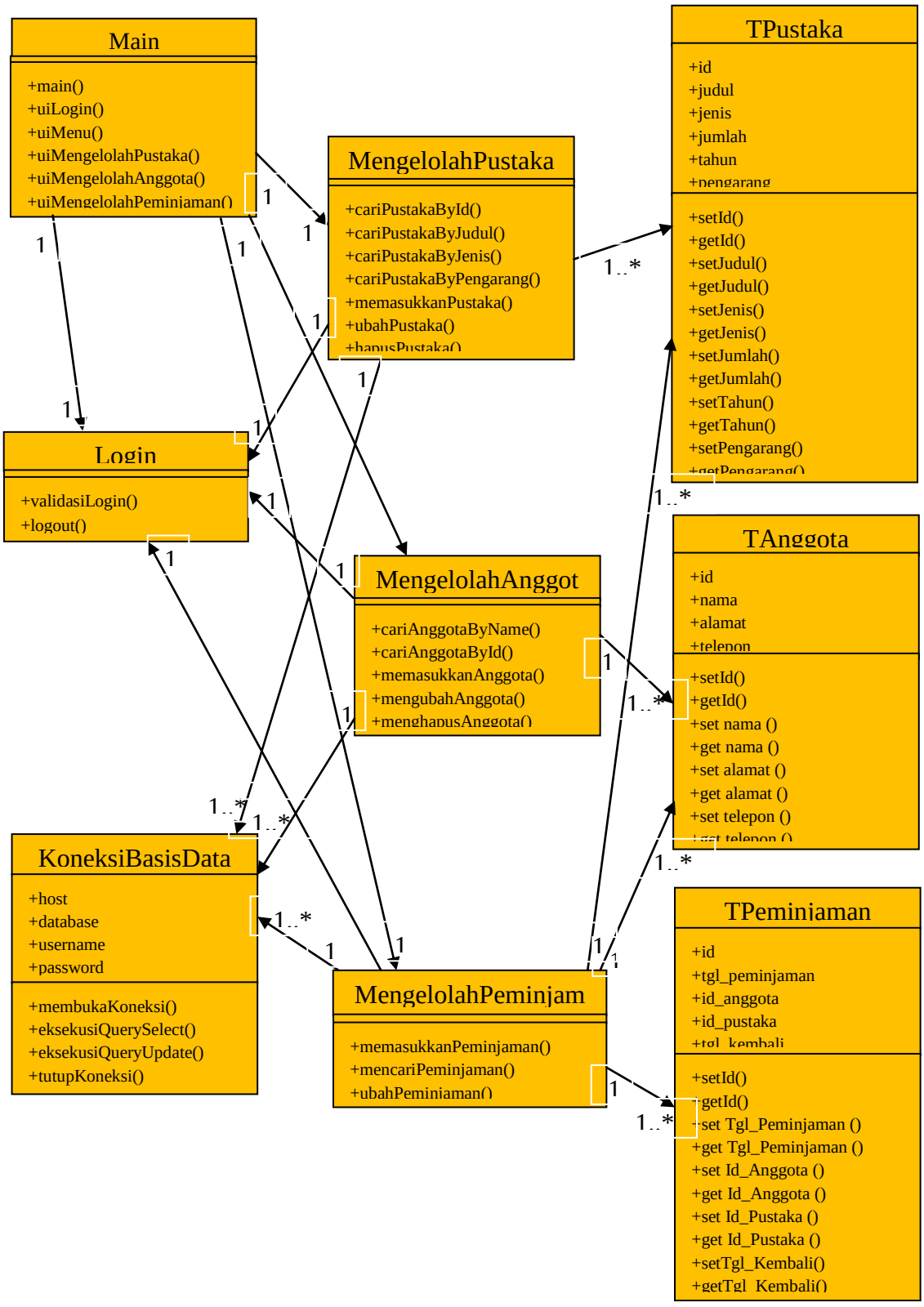
Diagram *UML* terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori yaitu :

1. Structure Diagram

Yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan. Yang termasuk dalam *structure diagram* adalah sebagai berikut:

a. Class diagram

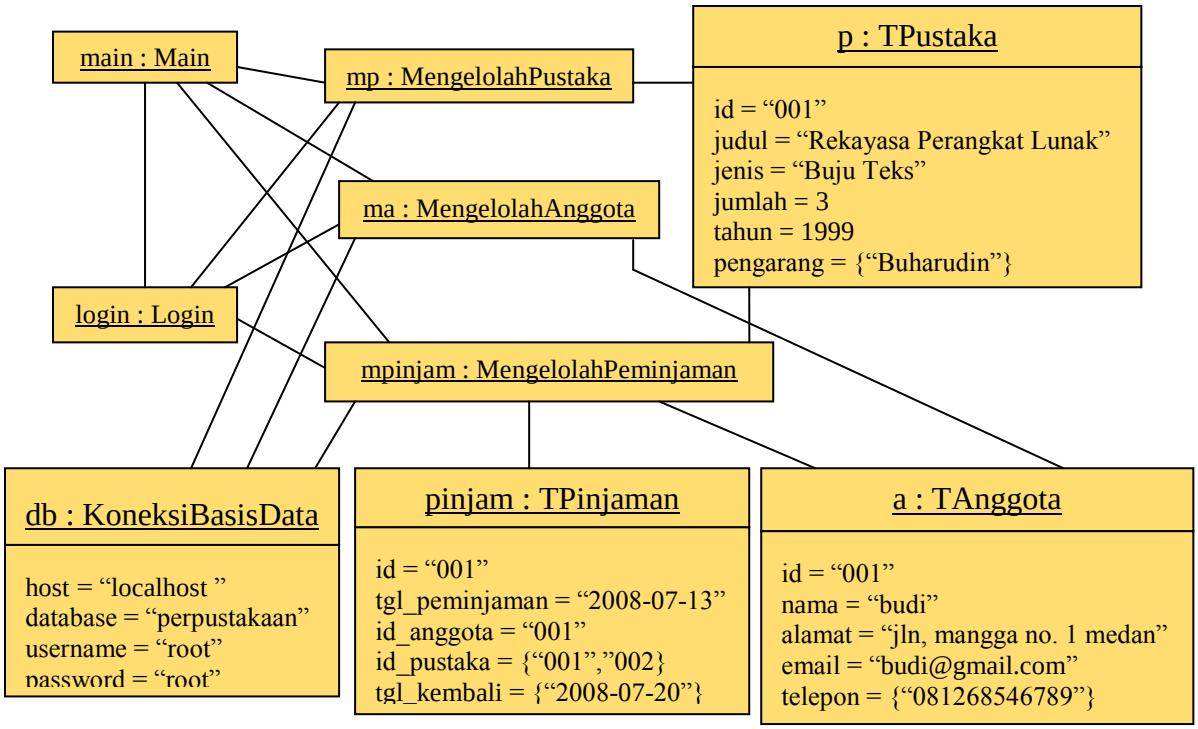
Digaram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefenisian kelas-kelas yang akan dibuat untuk membangun system, Contoh *class diagram* dapat dilihat pada gambar II.2. (Rosa A.S, M. Shalahuddin; 2011: 122).



Gambar II.2. Contoh *Class Diagram*
Sumber : (Rosa A.S, M. Shalahuddin; 2011: 161)

b. Object diagram

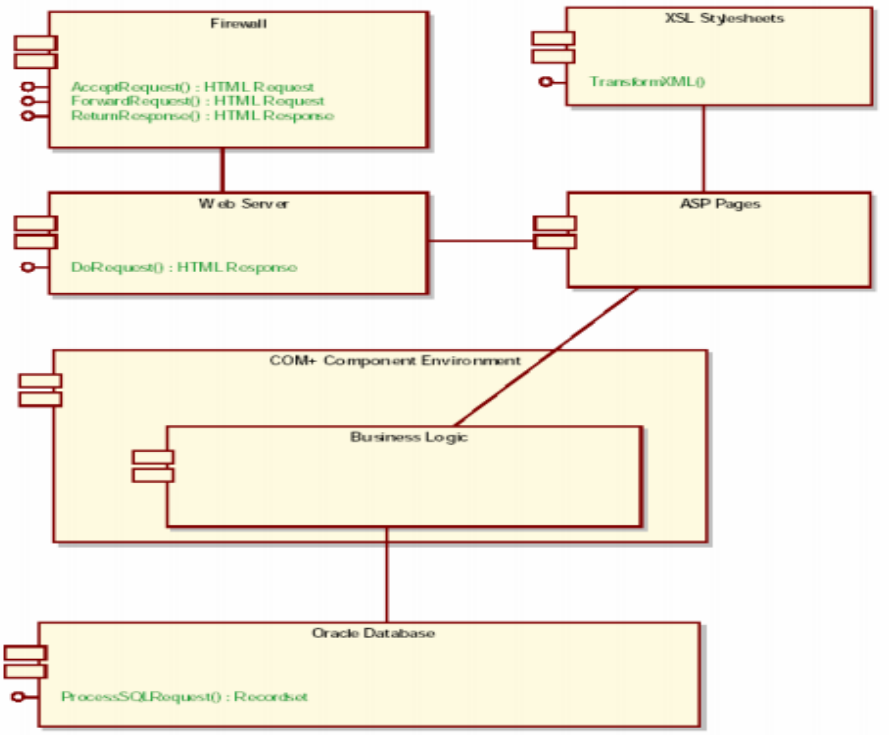
Diagram objek menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. . Contoh *Object diagram* dapat dilihat pada gambar II.3. (Rosa A.S, M. Shalahuddin; 2011: 124)



Gambar II.3. Contoh *Object Diagram*
Sumber : (Rosa A.S. dan M. Shalahuddin ; 2011 : 163)

c. Component diagram

Diagram komponen atau *component diagram* dibuat untuk menunjukkan organisasi dan ketergantungan di antara kumpulan komponen di dalam sebuah sistem. Contoh *Component diagram* dapat dilihat pada gambar II.4. (Romi Satria Wahono dan Sri Dharwiyanti ; 2003 : 10).



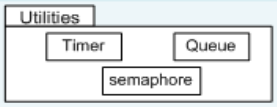
Gambar II.4. Contoh *Component Diagram*
Sumber : (Romi Satria Wahono dan Sri Dharwiyanti ; 2003 : 10)

d. Composite diagram

Composite strukture diagram baru mulai ada pada UML versi 2.0, pada versi 1.x diagram ini belum muncul. Digaram ini dapat digunakanuntuk menggambarkan struktur dari bagian-bagian yang aling terhubung maupun mendeskripsikan struktur pada saat berjalan (*runtime*) dari *instance* yang saling terhubung.

e. Package diagram

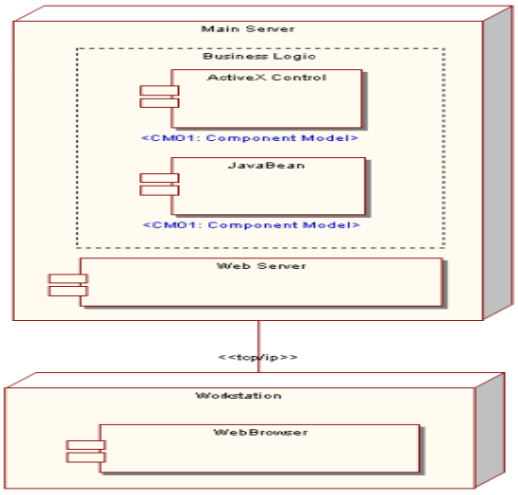
Diagram ini meyediakan cara mengumpulkan elemen-elemen yang saling terkait dalam diagram UML. Contoh *package diagram* dapat dilihat pada gambar II.5. (Rosa A.S, M. Shalahuddin; 2011: 128).

Simbol	Arti	Deskripsi
	Pakage	Merupakan sebuah bungkusan dari satu atau lebih kelas atau elemen
	Elemen dalam package	Elemen dalam package digambarkan dalam package

Gambar II.5. Contoh *Package Diagram*
Sumber : (Rosa A.S. dan M. Shalahuddin ; 2011 : 128)

f. Deployment diagram

Diagram deployment atau *deployment diagram* menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. Contoh *Deployment diagram* dapat dilihat pada gambar II.6. (Rosa A.S, M. Shalahuddin; 2011: 129)



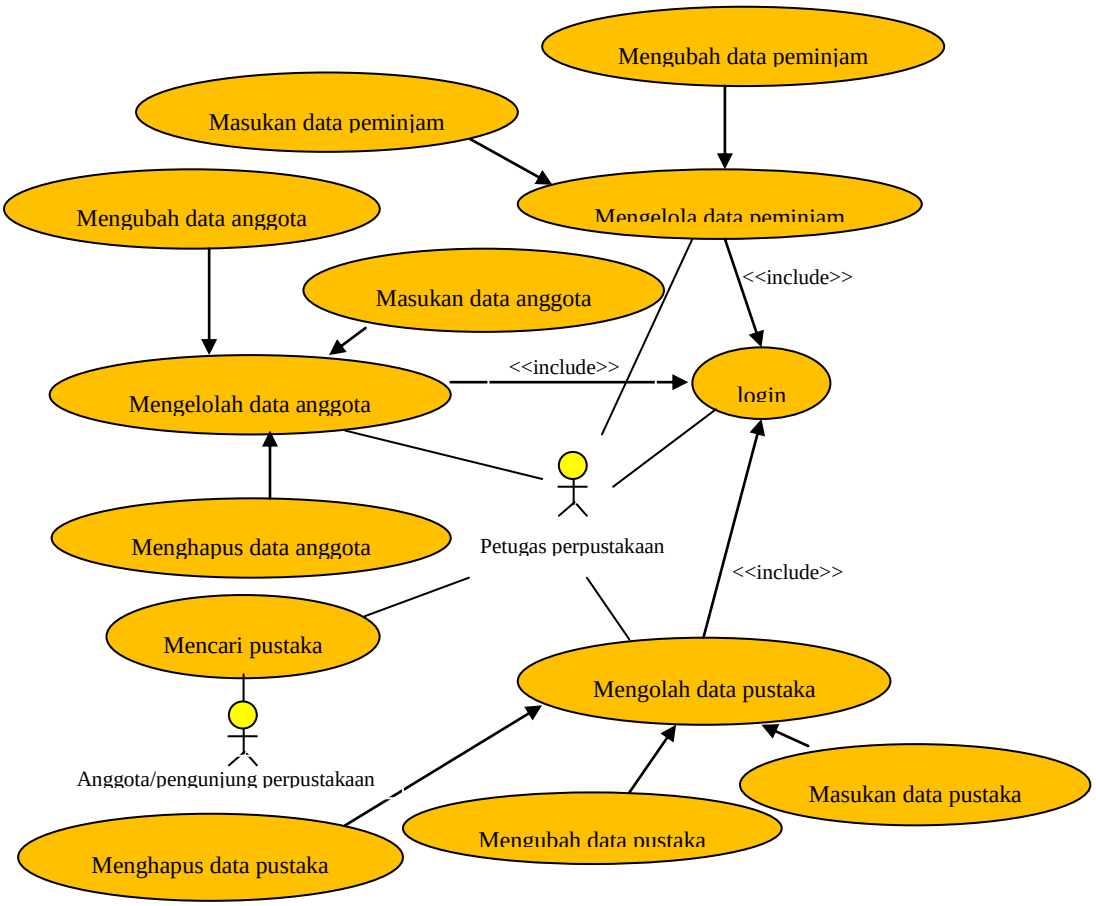
Gambar II.6. Contoh *Deployment Diagram*
Sumber : (Romi Satria Wahono dan Sri Dharwiyanti ; 2003 : 11)

2. Behavior Diagram

Yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem. Yang termasuk dalam *bahvior* diagram adalah sebagai berikut :

g. Use case

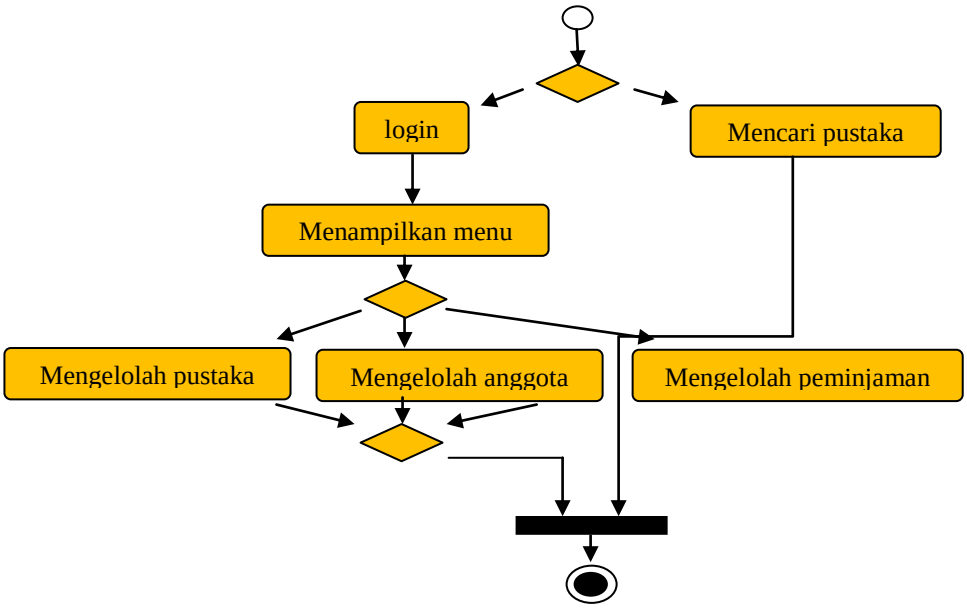
Use case atau diagram use case merupakan pemodelan untuk kelakuan (*bahavior*) sistem informasi yang akan dibuat. Contoh *Use Case* dapat dilihat pada gambar II.7. (Rosa A.S, M. Shalahuddin; 2011: 130)



Gambar II.7. Contoh *Use Case Diagram*
Sumber : (Rosa A.S. dan M. Shalahuddin ; 2011 : 160)

a. Activity diagram

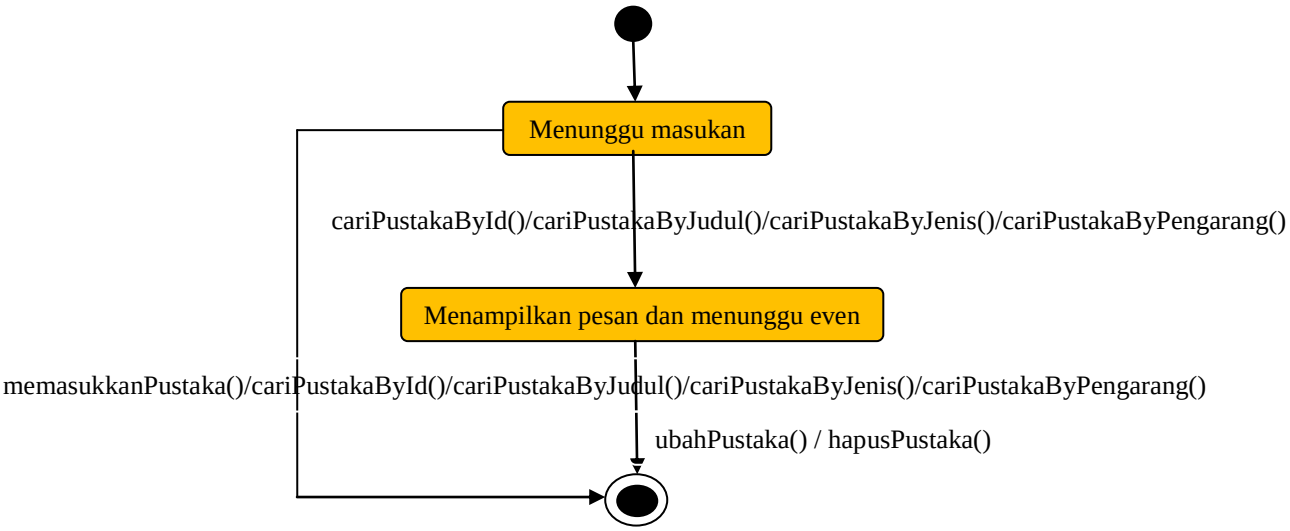
Digaram aktivitas atau activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Contoh *Use Case* dapat dilihat pada gambar II.8. (Rosa A.S, M. Shalahuddin; 2011: 134)



Gambar II.8. Contoh *Activity Diagram*
Sumber : (Rosa A.S. dan M. Shalahuddin ; 2011 : 177)

b. State machine diagram

Diagram mesin status digunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem. Contoh *state machine diagram* dapat dilihat pada gambar II.9. (Rosa A.S, M. Shalahuddin; 2011: 136).



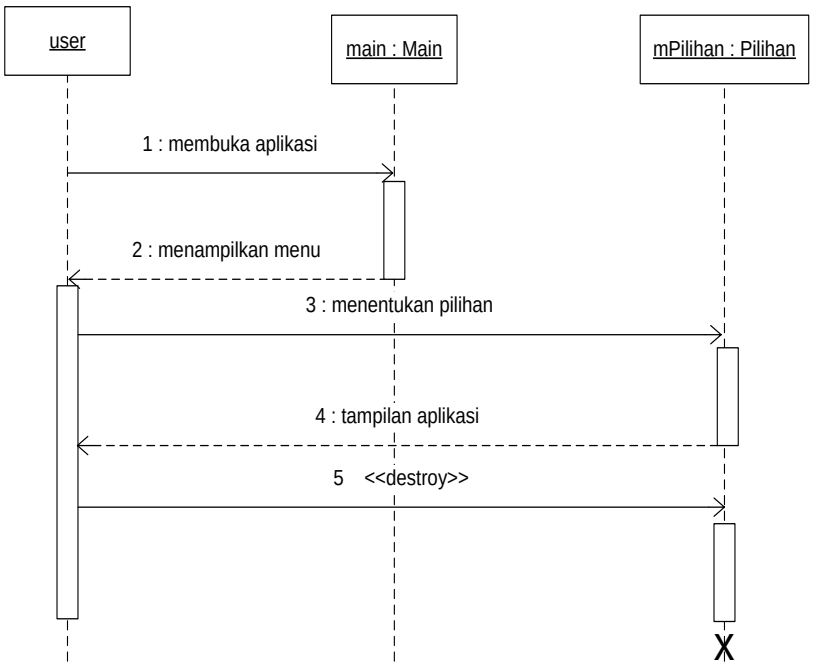
Gambar II.9. Contoh *State Machine Diagram*
Sumber : (Rosa A.S, M. Shalahuddin; 2011: 174)

3. Interaction Diagram

Yaitu kumpulan diagram yang digunkan untuk menggambarkan interaksi antar sub sistem pada suatu sistem. Yang termasuk dalam *interaction diagram* adalah sebagai berikut :

a. Sequence diagram

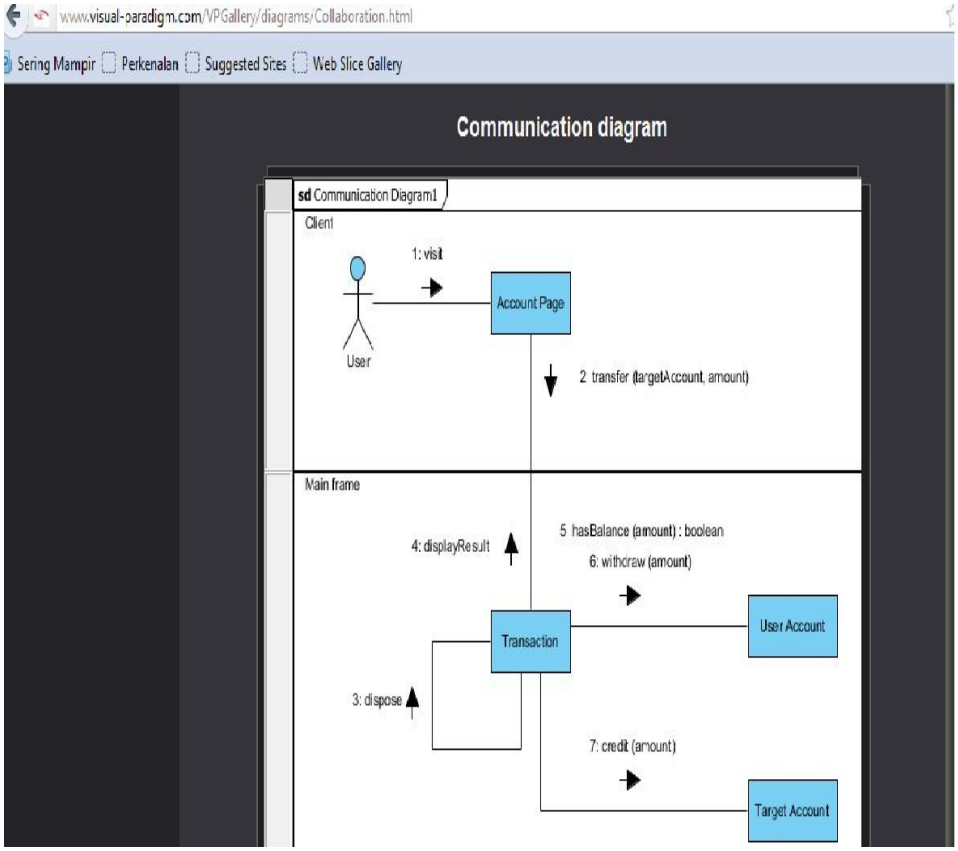
Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Contoh s*equence diagram* dapat dilihat pada gambar II.10. (Rosa A.S, M. Shalahuddin; 2011: 137).



Gambar II.10. Contoh *Sequence Diagram*
Sumber : (Rosa A.S, M. Shalahuddin; 2011: 164)

b. Communication diagram

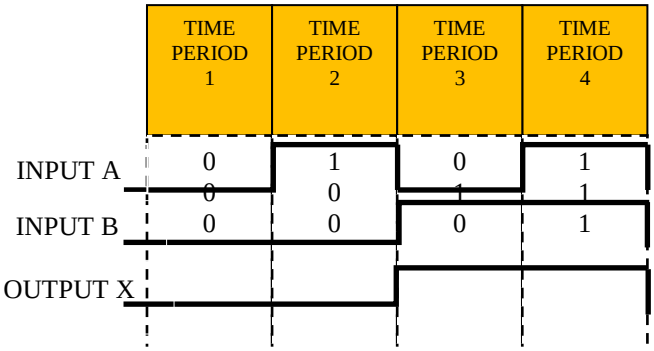
Communication diagram atau diagram komunikasi menggambarkan interaksi antar objek /bagian dalam bentuk urutan pengiriman pesan. Contoh *communication diagram* dapat dilihat pada gambar II.11. (Rosa A.S, M. Shalahuddin; 2011: 140).



Gambar II.11. Contoh *Communication Diagram*
Sumber : (http://www.visual-paradigm.com/VPGallery/img/diagrams/Collaboration/Communication-Diagram-Sample.png)

c. Timing diagram

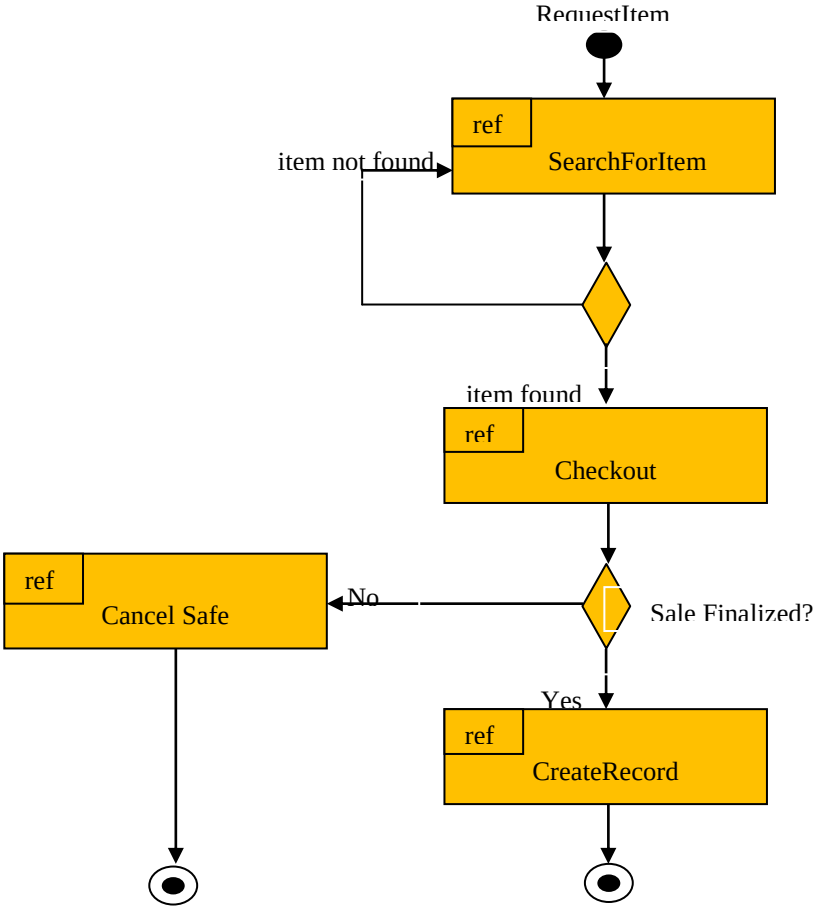
Timing diagram merupakan diagram yang fokus pada penggambaran terkait batasan waktu. Contoh *timing diagram* dapat dilihat pada gambar II.12. (Rosa A.S, M. Shalahuddin; 2011: 141)



Gambar II.12. Contoh *Timing Diagram*
Sumber : (Rosa A.S, M. Shalahuddin; 2011: 142)

d. Interaction overview diagram

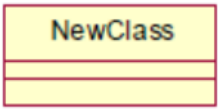
Diagram ini mirip dengan diagram aktivitas yang berfungsi untuk menggambarkan sekumpulan urutan aktivitas. Contoh *timing diagram* dapat dilihat pada gambar II.13. (Rosa A.S, M. Shalahuddin; 2011: 145).



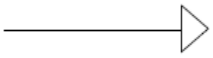
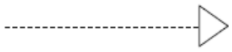
Gambar II.13. Contoh *Interaction Overview Diagram*
Sumber : (Rosa A.S, M. Shalahuddin; 2011: 145)

Dalam merancang UML (*Unified Modelling Language*) ada notasi-naotasi yang telah ditentukan yaitu sebagai berikut (lihat tabel II.1) :

Tabel II.1. Notasi-notasi dalam UML

Notasi	Keterangan
Actor 	*Actor* menggambarkan segala pengguna *software* aplikasi (*user*). *Actor* memberikan suatu gambaran jelas tentang apa yang harus dikerjakan *software* aplikasi. Sebagai contoh sebuah actor dapat memberikan input kedalam danmenerima informasi dari *software* aplikasi, perlu dicatat bahwa sebuah *actor* berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case*. Sebuah *actor* mungkin seorang manusia, satu *device*, *hardware* atau sistem informasi lainnya.
Use Case 	*Use case* menjelaskan urutan kegiatan yang dilakukan *actor* dan sistem untuk mencapai suatu tujuan tertentu. Walaupun menjelaskan kegiatan, namun *use case* hanya menjelaskan apa yang dilakukan oleh *actor* dan sistem bukan bagaimana *actor* dan sistem melakukan kegiatan tersebut. 1. *Use-case* Konkret adalah *use case* yang dibuat langsung karena keperluan *actor*. *Actor* dapat melihat dan berinisiatif terhadapnya . 2. *Use-case* Abstrak adalah *use case* yang tidak pernah berdiri sendiri. *Use case* abstrak senantiasa termasuk didalam (*include*), diperluas dari (*extend*) atau memperumum (*generalize*) *use case* lainnya. Untuk menggambarkannya dalam *use case* model biasanya digunakan *association relationship* yang memiliki *stereotype include*, *extend* atau *generalization relationship*. Hubungan *include* menggambarkan bahwa suatu *use case* seluruhnya meliputi fungsionalitas dari *use case* lainnya. Hubungan *extend* antar *use case* berarti bahwa satu *use case* merupakan tambahan fungsionalitas dari *use case* yang lain jika kondisi atau syarat tertentu terpenuhi.
Class 	*Class* merupakan pembentuk utama dari sistem berorientasi obyek, karena *class* menunjukkan kumpulan obyek yang memiliki atribut dan operasi yang sama. *Class* digunakan untuk mengimplementasikan *interface*. *Class* digunakan untuk mengabstraksikan elemen-elemen dari sistem yang sedang dibangun. *Class* bisa merepresentasikan baik perangkat lunak maupun perangkat keras, baik konsep maupun benda nyata. Notasi *class* berbentuk persegi

	panjang berisi 3 bagian: persegi panjang paling atas untuk nama *class*, persegi panjang paling bawah untuk operasi, dan persegi panjang ditengah untuk atribut. Atribut digunakan untuk menyimpan informasi. Nama atribut menggunakan kata benda yang bisa dengan jelas merepresentasikan informasi yang tersimpan didalamnya. Operasi menunjukkan sesuatu yang bisa dilakukan oleh obyek dan menggunakan kata kerja.
Interface 	*Interface* merupakan kumpulan operasi tanpa implementasi dari suatu *class*. Implementasi operasi dalam *interface* dijabarkan oleh operasi didalam *class*. Oleh karena itu keberadaan *interface* selalu disertai oleh *class* yang mengimplementasikan operasinya. *Interface* ini merupakan salah satu cara mewujudkan prinsip enkapsulasi dalam obyek.
Interaction	*Interaction* digunakan untuk menunjukkan baik aliran pesan atau informasi antar obyek maupun hubungan antar obyek. Biasanya *interaction* ini dilengkapi juga dengan teks bernama operation *signature* yang tersusun dari nama operasi, parameter yang dikirim dan tipe parameter yang dikembalikan.
Note 	*Note* digunakan untuk memberikan keterangan atau komentar tambahan dari suatu elemen sehingga bisa langsung terlampir dalam model. *Note* ini bisa disertakan ke semua elemen notasi yang lain.
Dependency	*Dependency* merupakan relasi yang menunjukan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain. Elemen yang ada di bagian tanda panah adalah elemen yang tergantung pada elemen yang ada dibagian tanpa tanda panah. Terdapat 2 *stereotype* dari *dependency*, yaitu *include* dan *extend*. *Include* menunjukkan bahwa suatu bagian dari elemen (yang ada digaris tanpa panah) memicu eksekusi bagian dari elemen lain (yang ada di garis dengan panah). *Extend* menunjukkan bahwa suatu bagian dari elemen di garis tanpa panah bisa disisipkan kedalam elemen yang ada di garis dengan panah.
Association	*Association* menggambarkan navigasi antar *class* (*navigation*), berapa banyak obyek lain yang bisa berhubungan dengan satu obyek (*multiplicity* antar *class*) dan apakah suatu *class* menjadi bagian dari *class* lainnya (*aggregation*). *Navigation* dilambangkan dengan penambahan tanda panah di akhir garis. *Bidirectional navigation* menunjukkan bahwa dengan mengetahui salah satu *class* bisa didapatkan informasi dari *class* lainnya.

	Sementara *UniDirectional navigation* hanya dengan mengetahui *class* diujung garis *association* tanpa panah kita bisa mendapatkan informasi dari *class* di ujung dengan panah, tetapi tidak sebaliknya. *Aggregation* mengacu pada hubungan has-a , yaitu bahwa suatu *class* memiliki *class* lain, misalnya Rumah memiliki *class* Kamar.
Generalization 	*Generalization* menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik. Dengan *generalization*, *class* yang lebih spesifik (*subclass*) akan menurunkan atribut dan operasi dari *class* yang lebih umum (*superclass*) atau *subclass* is *superclass* . Dengan menggunakan notasi *generalization* ini, konsep *inheritance* dari prinsip hirarki dapat dimodelkan.
Realization 	*Realization* menunjukkan hubungan bahwa elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada di bagian dengan panah. Misalnya *class* merealisasikan *package*, *component* merealisasikan *class* atau *interface*.

Sumber : (Rosa A.S, M. Shalahuddin; 2011: 123-139)

II.9 Database (Basis Data)

Sistem basis data adalah sistem terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan. Pada intinya basis data adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat. (Rosa A.S dan M. Shalahuddin; 2011: 44)

DBMS (*Database Management System*) atau dalam bahasa indonesia sering disebut Sistem Manajemen Basis Data adalah suatu sistem aplikasi yang digunakan untuk menyimpan, mengelola, dan menampilkan data. Suatu sistem aplikasi disebut DBMS jika memenuhi persyaratan minimal sebagai berikut ;

a. Menyediakan fasilitas untuk mengelola akses data
b. Mampu menangani integritas data
c. Mampu menangani akses data yang dilakukan secara bersamaan
d. Mampu menangani back-up data (Rosa A.S dan M. Shalahuddin; 2011: 45)

Berikut ini adalah 4 macam DBMS versi komersial yang paling banyak digunakan d dunia saat ini, yaitu :

1. *Oracle*
2. *Microsoft SQL Server*
3. *IBM DB2*
4. *Microsoft Access*

Sedangkan DBMS versiopen source yang cukup berkembang dan paling banyak digunakan saat ini adalah sebagai berikut :

1. *MySQL*
2. *PostgreSQL*
3. *Firebird*
4. *Sqlite*

Hampir semua DBMS mengadopsi SQL sebagai bahasa untuk mengelola data pada DBMS. (Rosa A.S dan M. Shalahuddin; 2011: 46)

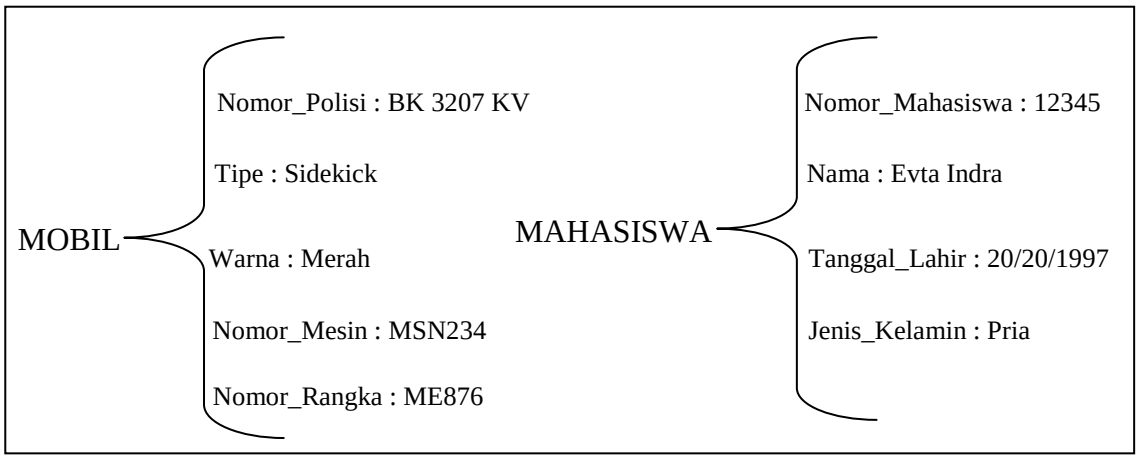
II.9.1 ERD (*Entity Relationship Diagram*)

ERD (*Entity Relationship Diagram*) adalah gambar atau diagram yang menunjukkan infomasi dibuat, disimpan, dan digunakan dalam sebuah sistem bisnis. Entitas biasanya menggambarkan jenis informasi yang sama. Dalam entitas digunakan untuk menghubungkan antar entitas yang sekaligus menunjukkan

hubungan antar data. Pada akhirnya ERD juga bisa digunakan untuk menunjukkan aturan0aturan bisnis yang ada pada sistem informasi yang akan dibangun. (Hanif Al Fatta; 2007: 121-127)

Elemen-elemen dari ERD adalah sebagai berikut :

1. *Entitas*, biasanya berupa orang, kejadian, atau benda dimana data akan dikumpulkan. Untuk menjadi entitas suatu objek harus menampilkan beberapa kali *event*.
2. *Atribut*, Setiap entitas dinyatakan dalam sejumlah atribut. Atribut adalah properti atau karakteristik yang terdapat pada setiap entitas. Sebagai contoh, pada gambar II.14, terdapat entitas MOBIL yang mengandung atribut Nomor_Polisi, Tipe, Warna, Nomor_Rangka dan Nomor_Mesin. Selain itu terdapat entitas MAHASISWA yang mengandung atribut Nomor_Mahasiswa, Nama, Tanggal_Lahir, dan Jenis_Kelamin.

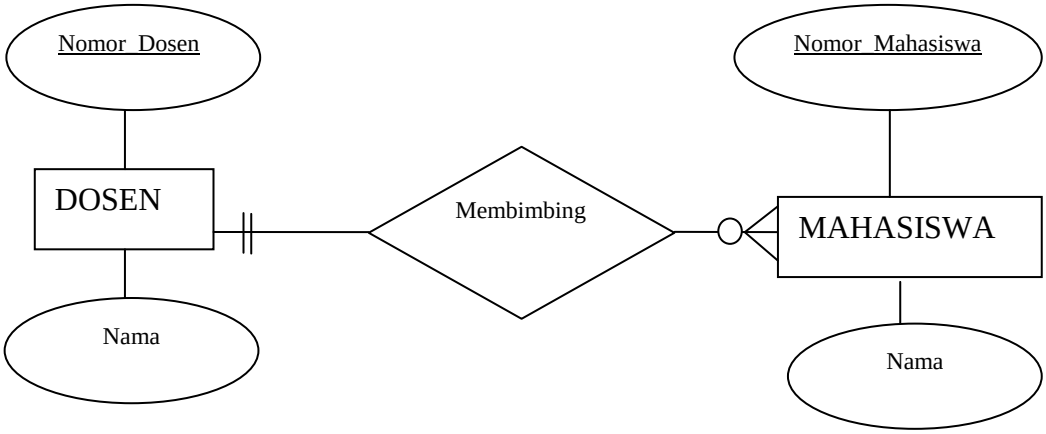


Gambar II.14 : Contoh Entitas dan Atribut

(Sumber : Abdul Kadir ; 2009:32)

3. *Relationship,* Hubungan (*Relationship*) menyatakan ketertarikan antara beberapa tipe entitas. Sebagai contoh , tipe entitas MAHASISWA dan

DOSEN mempunyai hubungan yang mencerminkan bahwa seorang mahasiswa memiliki dosen pembimbing akademis. Gambar II.15 menunjukkan hubungan tersebut.



Gambar II.15 : Contoh Hubungan antara tipe entitas
(Sumber : Abdul Kadir ; 2009:45)

II.9.2 Kamus Data

Kamus data adalah suatu daftar data elemen yang terorganisir dengan definisi yang tetap dan sesuai dengan sistem, sehingga user dan analis sistem mempunyai pengertian yang sama tentang input, output, dan komponen data strore.

Kamus data ini sangat membantu analis sistem dalam mendefinisikan data yang mengalir di dalam sistem, sehingga pendefinisian data itu dapat dilakukan dengan lengkap dan terstruktur. Pembentukan kamus data dilaksanakan dalam tahap analisis dan perancangan suatu sistem.

Pada tahap analisis, kamus data merupakan alat komunikasi antara user dan analis sistem tentang data yang mengalir di dalam sistem, yaitu tentang data yang masuk ke sistem dan tentang informasi yang dibutuhkan oleh user.

Sementara itu, pada tahap perancangan sistem kamus data digunakan untuk merancang input, laporan dan database.

Pembentukan kamus data didasarkan atas alur data yang terdapat pada DFD. Alur data pada DFD ini bersifat global, dalam arti hanya menunjukan nama alur datanya tanpa menunjukan struktur dari alur data itu. Untuk menunjukan struktur dari alur data secara terinci maka dibentuklah kamus data yang didasarkan pada alur data di dalam DFD. (Kusrini, M.Kom; 2007 : 35-36)

II.9.3 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 standar untuk menghasilkan struktur tabel yang normal. Pada dasarnya desain logika basis data transformasi dari model E-R ke bentuk fisik.

Dalam presfektif normalisasi sebuah *database* dikatakan baik jika setiap tabel yang membentuk basis data sudah berada dalam keadaan normal. Sebuah tabel dikatakan normal jika :

1. Jika ada dekomposisi/penguraian tabel, maka dekomposisinya dijamin aman (*lossless-join decomposition*)
2. Terpeliharanya ketergantungan functional pada saat perubahan data (depedency *preservation*)
3. Tidak melanggar *Boyce Code Normal Form* (BCNF), jika tidak bisa minimal tidak melanggar bentuk normalisasi ketiga. (Kusrini, M.Kom; 2007 : 39-40)

II.9.3.1 Bentuk-Bentuk Normalisasi

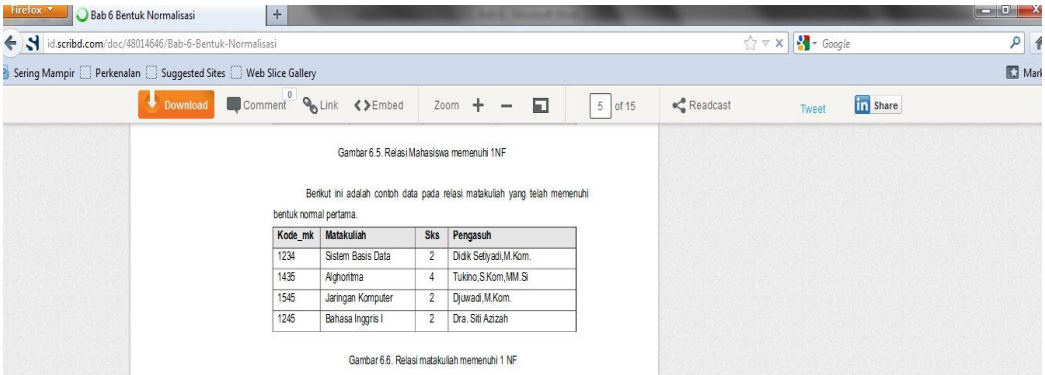
Adapun bentuk-bentuk dalam normalisasi adalah sebagai berikut :

a. Bentuk normal tahap pertama (1st Normal Form)

Suatu relasi dikatakan bentuk normal pertama, jika dan hanya jika setiapatribut bernilai tunggal untuk setiap baris. Tiap field hanya satu pengertian, bukanmerupakan kumpulan kata yang mempunyai arti mendua, hanya satu arti saja dan juga bukanlah pecahan kata – kata sehingga artinya lain. Tidak ada set atribut yangberulang-ulang atau atribut bernilai ganda.Pada data tabel sebelumnya, contoh data belum ternormalisasi sehinggadapat diubah ke dalam bentuk normal pertama dengan cara membuat setiap barisberisi kolom dengan jumlah yang sama dan setiap kolom hanya mengandung satunilai.

Contoh Normal Pertama (1NF)

Berikut ini adalah contoh data pada relasi mata kuliah yang telah memenuhi bentuk normal pertama dapat dilihat pada gambar II.16. (http://id.scribd.com/doc/48014646/Bab-6-Bentuk-Normalisasi).



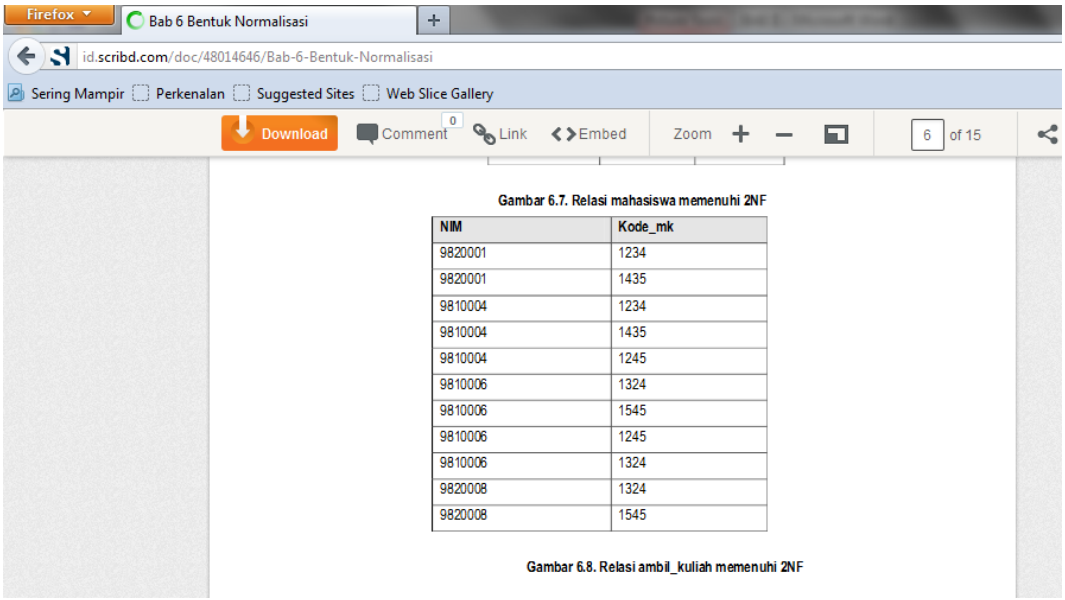
Gambar II.16 : Contoh Bentuk Relasi matakuliah memenuhi 1 NF (Sumber : http://id.scribd.com/doc/48014646/Bab-6-Bentuk-Normalisasi)

b. Bentuk normal tahap kedua (2nd Normal Form)

Definisi Bentuk Normal Kedua (2 NF) adalah :1). Memenuhi bentuk 1 NF (normal pertama).2). Atribut bukan kunci haruslah bergantung secara fungsi pada kunci utama / primary key . Sehingga untuk membentuk normal kedua tiap tabel / file haruslah ditentukan kunci-kunci atributnya. Kunci atribut haruslah unik dan dapat mewakiliatribut lain yang menjadi anggotanya. Pada contoh tabel Mahasiswa yang memenuhi normal pertama (1 NF) , terlihat bahwa NIM merupakan *Primery Key* (PK). NIM ⟶ Nama, Dosen Wali. Artinya adalah bahwa atribut Nama dan Dosen Wali bergantung pada NIM Tetapi NIM ⟶ Kode_mk. Artinya adalah bahqa atribut Kode_mk tidak tergantungpada NIM.Untuk memenuhi normal kedua, maka pada relasi mahasiswa tersebut dipecahmenjadi 2 relasi sebagai berikut:

Contoh Normal keduaq (2nd NF)

Berikut ini adalah contoh data pada relasi mata kuliah yang telah memenuhi bentuk normal kedua dapat dilihat pada gambar II.17. (http://id.scribd.com/doc/48014646/Bab-6-Bentuk-Normalisasi).



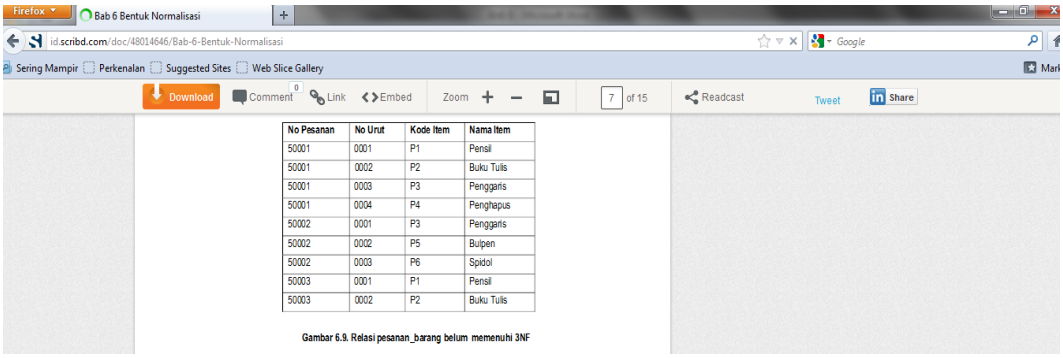
Gambar II.17 : Contoh Bentuk Relasi matakuliah memenuhi 2 NF
(Sumber : http://id.scribd.com/doc/48014646/Bab-6-Bentuk-Normalisasi)

c. Bentuk normal tahap ketiga (3rd Normal Form)

Pada relasi ini menunjukkan bahwa nama item tidak memiliki dependensisecara langsung terhadap kunci primer (No pesanan dan No Urut). Dengan katalain Nama Item memiliki dependensi transitif terhadap kunci primer.Sehingga untuk memenuhi bentuk 3 NF, maka relasi di atas didekomposisi menjadidua buah relasi sebagai berikut:Bentuk normal tahap keempat dan kelima

Contoh Normal ketiga (3rd NF)

Berikut ini adalah contoh data pada relasi mata kuliah yang telah memenuhi bentuk normal ketiga dapat dilihat pada gambar II.18. (http://id.scribd.com/doc/48014646/Bab-6-Bentuk-Normalisasi).



Gambar II.18 : Contoh Bentuk Relasi matakuliah memenuhi 3 NF

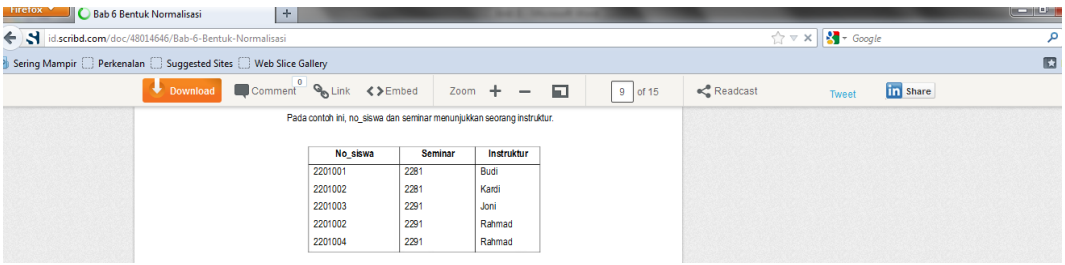
(Sumber : http://id.scribd.com/doc/48014646/Bab-6-Bentuk-Normalisasi

d. Boyce Code Normal Formal (BCNF)

BCNF merupakan bentuk normal sebagai perbaikan terhadap 3 NF. Suatu relasiyang memenuhi BCNF selalu memenuhi 3 NF, tetapi tidak untuk sebaliknya. Suatu relasi yang memenuhi 3 NF belum tentu memenuhi BCNF. Karena bentuk 3 NFmasih memungkinkan terjadi anomali.

Contoh Boyce Code Normal Formal (BCNF)

Berikut ini adalah contoh data pada relasi mata kuliah yang telah memenuhi bentuk Boyce Code Normal Formal dapat dilihat pada gambar II.19. (http://id.scribd.com/doc/48014646/Bab-6-Bentuk-Normalisasi).



Gambar II.19 : Contoh Bentuk Relasi matakuliah Boyce Code Normal Formal

(Sumber : http://id.scribd.com/doc/48014646/Bab-6-Bentuk-Normalisasi

II.10. Pemrograman Visual Basic 2008

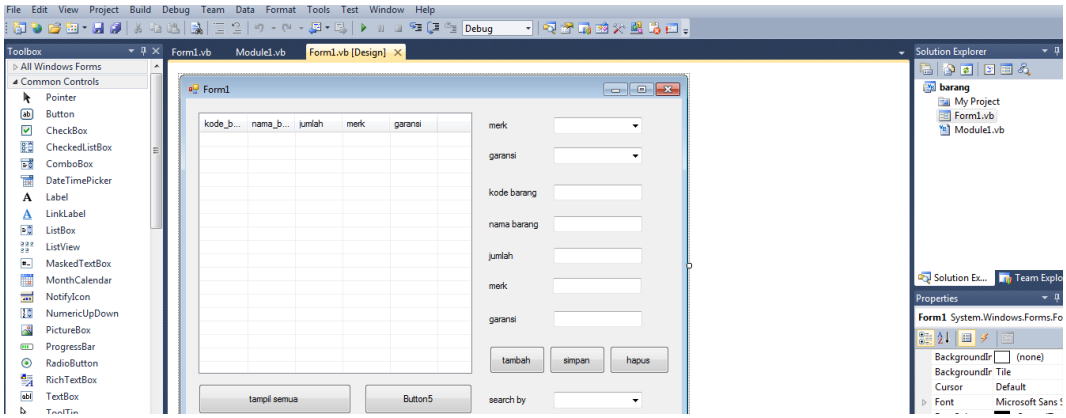
Menurut (Hendrayudi ; 2011 : 1) *Microsoft Visual Basic 2008* merupakan bagian dari kelompok bahasa pemrograman *Visual Studio 2008* yang dikembangkan oleh *Microsoft*. *Visual Studio 2008* terdiri dari beberapa bahasa pemrograman diantaranya adalah *Microsoft Visual Basic 2008*, *Microsoft C#*, dan *Visual Web Depelover 2008*.

Visual Studio ini telah mengalami perubahan versi mulai dari *Visual Studio 6.0, Visual Studio 2005, Visual Studio 2006* dan yang terakhir adalah *Visual Studio 2008*.

Microsoft Visual Basic 2008 setara dengan *Microsoft Visual Basic 9.0* yang memiliki kelebihan-kelebihan yaitu suport dengan bahasa *query Languange-Integreted Query* (*LINQ*) dan *suport* dengan *database Microsoft SQL Server Compact 3.5*. Selain itu, kelebihan lain adalah memiliki *Object Relation Designer* (*O/R Designer*) untuk membantu mengedit *LINQ* ke *SQL* dihubungkan dengan *database* dan *fiture* lain, seperti *WPF* (*Windows Presentation Foundation*) dan *WCF* (*Windows Communication Foundation*). Semua hal yang baru tersebut di atas menambah kelengkapan aplikasi *Microsoft Visual Basic 2008* dalam membuat media dan dokumen.

Menurut (Ketut Darmayuda ; 2009 :1-2) *Microsoft Visual Basic Studio 2008* merupakan kelanjutan dari *Micrososft Visual Studio* sebelumnya, yaitu *Visual Studio.Net 2003* yang diproduksi oleh *Micrososft*. Pada bulan Februari tahun 2002 *Microsoft* memproduksi teknologi *.Net Framwork versi 1.0*, Teknologi *.Net* ini didasarkan atas susunan berupa *.Net Framwork*, sehingga setiap produk

baru yang terkait dengan teknologi *.Net* akan selalu berkembang mengikuti perkembangan *.Net Fremwork*-nya. Pada perkembangan nantinya, mungkin untuk membuat program dengan teknologi *.Net*, dan memungkinkan para pengembang perangkat lunak akan dapat menggunakan lintas sistem operasi, yaitu dapat dikembangkan di sistem operasi *Windows* juga dapat dijalankan pada sistem operasi *Linux*, seperti yang telah dilakukan pada pemrograman *Java* oleh *Sun Microsystem*. Pada saat ini perusahaan-perusahaan sudah banyak meng-*update* aplikasi yang lama yang dibuat dengan *Microsoft Visual Basic 6.0* ke teknologi *.Net* karena kelebihan-kelebihan yang ditawarkan, terutama memungkinkan pengembang perangkat lunak secara cepat mampu membuat program yang *robust*, serta berbasiskan intergrasi ke *internet* yang dikenal dengan *XML Web Service*. Contoh gambar pemograman visual basic2008 dapat dilihat pada gambar II.20. (http://www.maniavb.com/2011/04/source-code-visual-basic-net-2008-tipe.html).



Gambar II.20 : Contoh pemograman visual basic 2008

(Sumber : ketut darmayuda)

II.11 MySQL

MySQL merupakan *software* RDBMS (*Relation Database Management System*) atau *server database* yang dapat mengelola *database* dengan sangat cepat, dapat menampung data dalam jumlah sangat besar, dapat diakses oleh banyak *user* (*multi-user*), dan dapat melakukan suatu proses secara sinkron atau berbarengan (*multi-threaded*).

Saat ini, *MySQL* banyak digunakan di barbagai kalangan untuk melakukan penyimpanan dan pengolahan data. Mulai dari kalangan akademis sampai ke industri, baik indutri kecil, menengah ataupun besar.

Lisensi *MySQL* terbagi menjasi dua. Anda dapat menggunakan *MySQL* sebagai produk *open source* di bawah *GNU General Public License* (gratis) atau dapat membeli lisensi dari versi komersialnya. *MySQL* versi komersial tentu memiliki nilai lebih atau kemampuan yang tidak disertakan pada versi gratis.

Perangkat lunak bebas, yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program yang disebut XAMMP. Fungsinya adalah sebagai *server* yang berdiri sendiri (*localhost*), yang terdiri atas program *Apache* HTTP *Server*, *MySQL* database, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan *Perl*. Nama XAMPP merupakan singkatan dari X (empat sistem operasi apapun), *Apache*, *MySQL*, PHP dan *Perl*. Program ini tersedia dalam GNU *General Public License* dan bebas, merupakan *web server* yang mudah digunakan yang dapat melayani tampilan halaman *web* yang dinamis.

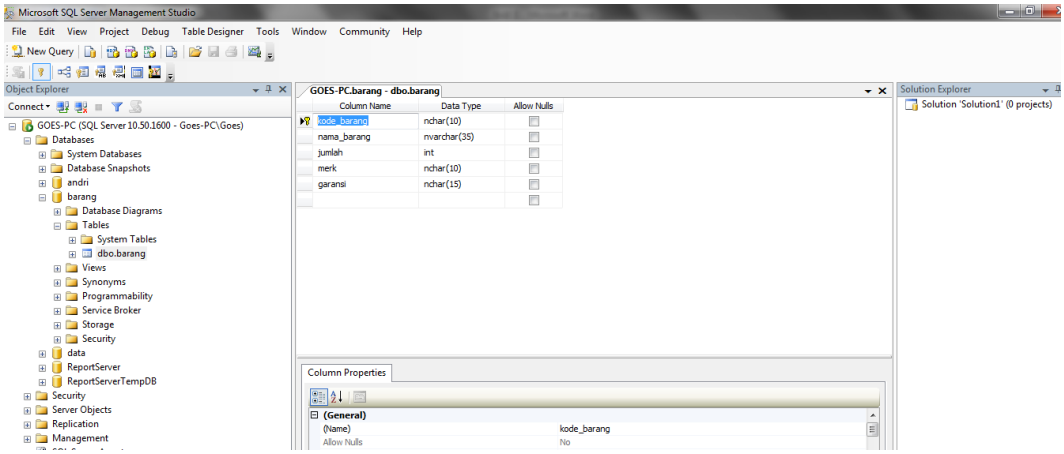
(Budi Raharjo; 2011 : 21-22)

Beberapa alasan menggunakan *MySQL* sebagai *server database* untuk aplikasi-apliksi yang dikembangkan :

1. Fleksibel
2. Performa tinggi
3. Lintas platform
4. Gratis
5. Proteksi data yang handal
6. Komunitas luas

(Budi Raharjo; 2011 : 22-23)

Contoh gambar Mysql dapat dilihat pada gambar II.21. (Budi Raharjo)

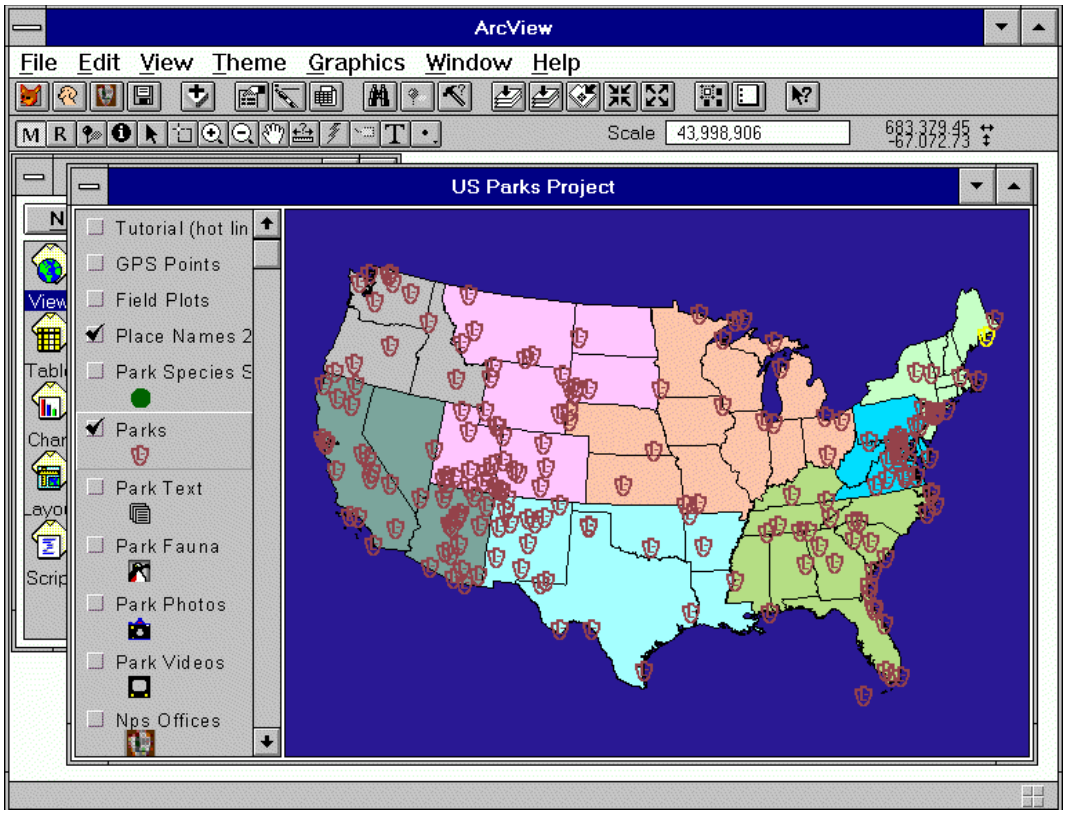


Gambar II.21: Contoh *Mysql*

(Sumber : Budi Raharjo)

II.12 ArcView

ArcView merupakan sebuah *software* pengolah data spasial. *Software* ini memiliki berbagai keunggulan yang dapat dimanfaatkan oleh kalangan pengolah data spasial. *ArcView* memiliki kemampuan dalam pengolahan atau editing arc, menerima atau konversi dari data digital lain seperti CAD, atau dihubungkan dengan data *image* seperti format JPG, TIFF atau image gerak. Hasil pengolahan data spasial dalam *ArcView* disimpan dalam sebuah proyek dengan ekstensi APR Contoh gambar *ArcView* dapat dilihat pada gambar II.21 (Eko Budiyanto; Sistem Informasi Geografis Menggunakan *ArcView*; 2006 : 19)



Gambar II.22 : Contoh gambar *ArcView*
Sumber : (Eko Budiyanto; Sistem Informasi Geografis Menggunakan *ArcView*; 2006 : 19)