BAB II
LANDASAN TEORI

2.1. Konsep Sistem Informasi

Sistem kebanyakan dapat didefinisikan secara sederhana sebagai sekelompok elemen yang saling berhubungan atau berinteraksi hingga membentuk satu kesatuan. Akan tetapi, konsep umum sistem berikut ini memberikan konsep dasar yang lebih tepat untuk bidang Sistem Informasi.

Sistem adalah sekelompok komponen yang saling berhubungan, bekerja bersama untuk mencapai tujuan bersama dengan menerima input serta menghasilkan output dalam proses transformasi yang teratur. (O'brien 2006:29)

Suatu sistem mempunyai karakteristik atau sifat-sifat tertentu, antara lain sebagai berikut :

a. Komponen Sistem (*Component*)

Suatu sistem terdiri dari sejumlah komponen atau elemen yang saling berinteraksi, artinya komponen atau elemen yang saling bekerja sama dalam bentuk satu kesatuan. Komponen atau elemen sistem dapat berupa subsistem atau bagian dari sistem. Setiap subsistem mempunyai sifat-sifat dari sistem. Untuk menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan.

b. Batas Sistem (*Boundary*)

Batas sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luar. Batas suatu sistem menunjukkan lingkup (*scope*) dari sistem tersebut.

c. Lingkungan luar (*environtments*)

Lingkungan luar dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi dari sistem.

d. Penghubung (*interface*)

Penghubung sistem merupakan media penghubung antara satu subsistem dengan subsistem yang lain untuk dapat berinteraksi membentuk suatu kesatuan.

Dipindai dengan CamScanner

e. Masukan (*input*)

Masukan sistem merupakan energi yang dimasukan ke dalam sistem yang berupa masukan perawatan (*maintenance input*) dan keluaran sinyal (*signal output*). *Maintenance input* adalah energi yang dimasukan supaya sistem tersebut dapat beroperasi. *Signal output* adalah energi yang diproses untuk mendapatkan keluaran.

f. Pengolahan (*process*)

Suatu sistem dapat mempunyai suatu bagian pengolahan yang akan merubah masukan menjadi keluaran.

g. Keluaran (*output*)

Keluaran sistem adalah hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan.

h. Sasaran (*objective*)

Suatu sistem harus mempunyai sasaran, karena sasaran sangat menentukan sekali masukan yang dibutuhkan oleh sistem dan keluaran yang akan dihasilkan sistem. Suatu sistem dikatakan berhasil apabila mengenai sasaran atau tujuan.

2.2 Konsep Dasar Informasi

Informasi adalah hasil dari pengolahan data dalam bentuk yang lebih berguna dan berarti bagi penerimanya yang menggambarkan suatu kejadian-kejadian yang nyata yang digunakan untuk pengambilan keputusan.

Sumber dari informasi adalah data. Data adalah kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan nyata. Kejadian-kejadian (*event*) adalah suatu yang terjadi pada saat tertentu. Kesatuan nyata (*fact*) adalah berupa suatu obyek nyata seperti tempat, benda atau orang yang benar-benar ada dan terjadi.

Menurut Jogiyanto HM, MBA, Akt., Ph.D. (2005) :

"Informasi (information) adalah data yang diolah menjadi bentuk yang yang menggambarkan suatu kejadian-kejadian yang nyata yang digunakan untuk pengambilan keputusan."

Dipindai dengan CamScanner

Untuk dapat berguna, maka informasi harus didukung oleh tiga pilar, yaitu sebagai berikut :

a. Akurat (*accurate*)

Akurat berarti informasi harus bebas dari kesalahan-kesalahan dan tidak menyesatkan. Akurat juga berarti informasi harus jelas mencerminkan maksudnya. Informasi harus akurat karena dari sumber informasi sampai ke penerima informasi kemungkinan banyak terjadi gangguan (*noise*) yang dapat merubah atau merusak informasi tersebut.

b. Tepat waktu (*Timeliness*)

Tepat pada waktunya berarti informasi yang datang pada penerima tidak boleh terlambat. Informasi yang sudah usang tidak akan mempunyai nilai lagi karena informasi merupakan landasan didalam pengambilan keputusan.

c. Relevan (*Relevance*)

Relevan berarti informasi tersebut mempunyai manfaat untuk pemakianya. Relevansi informasi untuk tiap-tiap orang berbeda. Nilai informasi bagi seorang pemakai ditentukan oleh keandalan (*reliabilitas*). Keluaran yang tidak didukung oleh ketiga pilar ini tidak dapat dikatakan sebagai informasi yang berguna, tetapi merupakan sampah (*garbage*).

2.3 Konsep Dasar Sistem Informasi

Sistem Informasi dapat didefinisikan sebagai suatu susunan dari orang, data, proses, dan teknologi informasi yang saling berhubungan untuk mengumpulkan, memroses, menyimpan, dan menyediakan keluaran informasi yang diperlukan untuk mendukung suatu organisasi. Sistem informasi dapat digolongkan menurut fungsinya, antara lain adalah sebagai berikut ini:(Whitten 2004:12)

a *Transaction Processing System* (TPS), suatu sistem informasi yang menangkap dan memproses data tentang transaksi bisnis. seperti pesanan (*order*), kartu catatan waktu, pembayaran, reservasi, dan sebagainya.(Whitten 2004:12)

b *Management Information System* (MIS), suatu sistem informasi yang disediakan untuk menghasilkan laporan yang berorientasi pada manajemen

Dipindai dengan CamScanner

yang berdasarkan pada proses transaksi dan operasi dari organisasi. Atau dengan kata lain menggunakan data transaksi untuk menghasilkan informasi yang dibutuhkan oleh manajer untuk menjalankan bisnis.(Whitten 2004:12)

c *Decision Support* System (DSS), suatu sistem informasi yang membantu mengidentifikasi pengambilan keputusan yang mungkin atau menyediakan informasi untuk membantu pengambilan keputusan manajemen.(Whitten 2004:12)

d *Executive Information System* (EIS), suatu sistem informasi yang mendukung perencanaan dan kebutuhan penilaian dari manajer eksekutif. EIS dikhususkan untuk kebutuhan informasi yang unik dari para eksekutif yang merencanakan bisnis dan menilai capaian rencana bisnis tersebut.(Whitten 2004:13)

e *Expert System* (ES), suatu sistem informasi yang menangkap keahlian dari para pekerja dan kemudian menirukan keahlian tersebut untuk dimanfaatkan oleh orang yang tidak ahli.(Whitten 2004:14)

f *Communications and Collaboration System*, suatu sistem informasi yang memberikan peluang komunikasi yang lebih efektif antara para pekerja, mitra, pelanggan, dan para penyalur untuk meningkatkan kemampuan mereka untuk bekerja sama. (Whitten 2004:14)

g *Office Automation System*, suatu sistem informasi yang mendukung cakupan luas dari aktivitas kantor yang disediakan untuk meningkatkan alur kerja (*work flow*) antara para pekerja dan membantu karyawan membuat dan membagi dokumen yang dapat mendukung aktivitas kantor sehari-hari. (Whitten 2004:14)

Komponen Sistem Informasi adalah sebagai berikut:

a. Perangkat Keras (*Hardware*), Terdiri dari komputer, *peripheral*, jaringan, dsb.

b. Perangkat Lunak (*Software*), Merupakan kumpulan dari perintah/fungsi yang ditulis dengan aturan tertentu untuk memerintahkan komputer melaksanakan tugas tertentu. *Software* dapat digolongkan menjadi Sistem Operasi

Dipindai dengan CamScanner

(Windows 2000, Linux, Unix, dll), Aplikasi (Akuntansi, database, dll), Utilitas (Anti Virus, Speed Disk, dll), serta Bahasa (Java, VB, Delphi, C++, dll).

c. Data, Merupakan komponen dasar dari informasi yang akan diproses lebih lanjut untuk menghasilkan informasi.
d. Prosedur, Dokumentasi prosedur / proses sistem, buku penuntun operasional (aplikasi) dan teknis.
e. Manusia (*Human*), Yang terlibat dalam komponen manusia seperti operator, pemimpin sistem informasi dan sebagainya. Oleh sebab itu perlu suatu rincian tugas yang jelas.

Kegiatan dari sistem informasi antara lain adalah:

a. *Input*, Menggambarkan suatu kegiatan untuk menyediakan data untuk diproses.
b. *Proses*, Menggambarkan bagaimana suatu data di proses untuk menghasilkan suatu informasi yang bernilai tambah.
c. *Output*, Suatu kegiatan untuk menghasilkan laporan dari proses di atas tersebut.
d. Penyimpanan, Suatu kegiatan untuk memelihara dan menyimpan data.
e. *Control*, Suatu aktivitas untuk menjamin bahwa sistem informasi tersebut berjalan sesuai dengan yang diharapkan.

2.4 Analisa dan Perancangan Sistem Berorientasi Obyek dengan UML

Analisa sistem dapat dinyatakan sebagai pemisahan suatu hal dalam bagian bagian tertentu. Bagian-bagian tersebut kemudian dipelajari dan dievakuasi untuk mengetahui apakah terdapat cara-cara yang lebih baik untuk memenuhi kebutuhan manajemen.

"Analisa sistem adalah proses menentukan kebutuhan sistem – apa yang harus dilakukan sistem untuk memenuhi kebutuhan klien, bukanlah bagaimana sistem tersebut diimplementasikan." (Ariesto Hadi Sutopo, 2002:242):

Dipindai dengan CamScanner

Konsep dasar berorientasi obyek mencapai kematangannya pada saat masalah analisis dan desain menjadi lebih diperhatikan dari pada masalah coding. Secara spesifik, pengertian "berorientasi obyek" (Ariesto Hadi Sutopo, 2002:3) berarti bahwa "kita mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya".

2.4.1 Unified Modelling Language

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 danjaringan 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 obyek. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya : Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modelling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

Cakupan UML diantaranya : Pertama, UML menggabungkan konsep BOOCH, OMT, dan OOSE, sehingga UML merupakan suatu bahasa permodelan tunggal yang umum dan digunakan secara luas oleh para user ketiga metode tersebut dan bahkan para user metode lainnya. Kedua, UML menekankan pada apa yang dapat dikerjakan dengan metode-metode tersebut. Ketiga, UML berfokus pada suatu bahasa permodelan standar, bukan pada proses standar.

2.4.2 Analisa Sistem berorientasi Obyek

*a. **Activity Diagram***

Diagram memodelkan alur kerja (*work floaw*) sebuah proses bisnis dan urutan aktivitas pada suatu proses. Diagram ini sangat mirip dengan *flow chart* karena kita dapat memodelkan prosedur logika, proses bisnis dan alur kerja.

Dipindai dengan CamScanner

Perbedaan utamanya adalah *flow chart* dibuat untuk menggambarkan alur kerja dari sebuah sistem, sedangkan *activity diagram* dibuat untuk menggambarkan aktivitas dari aktor.

Activity diagram adalah teknik untuk mendiskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity diagram* mempunyai persan seperti halnya *flow chart*, akan tetapi perbedanya dengan *flow chart* adalah *activity diagram* bisa mendukung perilaku paralel sedangkan *flow chart* tidak bisa.

Simbol – simbol yang digunakan pada saat pembuatan *activity diagram* adalah sebagai berikut :

1) *Start Point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.(Munawar 2005:109)

Start Point

2) *End Point*, akhir aktifitas.(Munawar 2005:109)

End Point

3) *Activity*, menggambarkan suatu proses / kegiatan bisnis.(Munawar 2005:109)

Activity

Jenis – jenis *Activities*, yaitu :

a) *Black Hole Activities*

Ada masukan dan tidak ada keluaran, biasanya digunakan jika dikehendaki ada satu atau lebih transisi.

Dipindai dengan CamScanner

Simbol *Black Hole Activities*

b) *Miracle Activities*

Tidak ada masukan dan ada keluaran, biasanya dipakai pada waktu start point dan dikehendaki ada 1 atau lebih transisi.

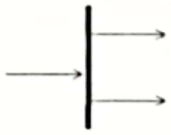
Simbol *Miracle Activities*

c) *Paralel Activities*

Suatu *activity* yang berjalan secara bersamaan terdiri dari :

1) *Fork* (Percabangan)

Fork digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.(Munawar 2005:110)



Simbol *Fork* (Percabangan)

2) *Join* (Penggabungan))

Join (penggabungan) / *Rake* ,menunjukkan adanya dekomposisi.(Munawar 2005:110) .Yaitu mempunyai 2 atau lebih transisi masuk dan hanya 1 transisi keluar, dan *fork* harus berhubungan dengan *join*.

Simbol *Join* (Penggabungan)

Dipindai dengan CamScanner

3) *Decision Point*

Decision digambarkan dengan lambing wajik atau belah ketupat. Mempunyai transisi (sebuah garis dari atau ke *decision point).* Setiap transisi yang ada harus mempunyai *guard* (kunci). Tidak ada keterangan (pernyataan) pada tengah belah ketupat seperti pada *flowchart*.

Simbol *Decision Point*

4) *Guard* (Kunci)

Guard (kunci) adalah kondisi benar sewaktu melewati sebuah transisi. Digambarkan dengan diletakkan diantara tanda [ ]. Tanda (*otherwise*) guard untuk menangkap suatu kondisi yang belum terdeteksi. Setiap transisi dari atau ke decision point harus mempunyai *guard* yang harus konsisten dan lengkap serta tidak *overlap*.

5) *Swimlane*

Swimlane merupakan sebuah cara untuk mengelompokkan activity berdasarkan *actor*. *Actor* bisa ditulis nama *actor* ataupun sekaligus dalam lambang actor. *Swimlane* digambarkan secara *vertical*, walaupun kadang-kadang digambarkan secara *horizontal*.

6) *Swimarea*

Ketika sebuah *activity diagram* mempunyai banyak *swimlane*, perlu dipikirkan dengan pendekatan *swimarea*. *Swimarea* mengelompokkan *activity* berdasarkan kegiatan didalam *use case*.

Dipindai dengan CamScanner

b. **Analisa Dokumen Keluaran**

Analisa keluaran adalah analisa mengenai dokumen – dokumen keluaran yang dihasilkan dari sebuah sistem.

c. **Analisa Dokumen Masukan**

Analisa masukkan adalah bagian dari pengumpulan informasi tentang system yang sedang berjalan. Tujuan analisa masukkan adalah memahami prosedur berjalan.

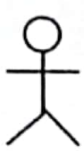
d. ***Use Case Diagram***

Use Case Diagram menggambarkan sebuah fungsionalitas yang diharapkan dari sebuah sistem dan bagaimana sistem berinteraksi dengan dunia luar. Yang ditekankan dalam *use case diagram* adalah "apa" yang diperbuat sistem, dan bukan "bagaimana" sistem itu melakukannya. Sebuah *use case* merepresentasikan sebuah interaksi antara actor dengan sistem. *Use Case Diagram* juga menjelaskan manfaat sistem jika dilihat menurut pandangan orang yang berada diluar sistem (*actor*). *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-create sebuah daftar belanja, dan sebagainya.

Secara umum use case diagram terdiri dari :

1) Aktor (*Actor*)

Actor adalah sebuah peran yang bisa dimainkan oleh pengguna dalam interaksinya dengan sistem. Untuk mengidentifikasikan actor harus ditentukan pembagian kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. *Actor* dilukiskan dengan peran yang mereka mainkan dalam *use case*, seperti staff penjualan, pelanggan, dll.



Dipindai dengan CamScanner

Simbol *Actor*

2) *Use case*

Use case menggambarkan perilaku, termasuk didalamnya interaksi antara actor dengan sistem. *Use case* dibuat berdasarkan keperluan *actor*, merupakan "apa" yang dikerjakan sistem bukan "bagaimana" sistem mengerjakannya. Setiap *use case* harus diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan *actor*. Nama *use case* boleh terdiri dari beberapa kata dan tidak boleh ada dua *use case* yang memiliki nama yang sama.

Simbol *Use Case*

3) Asosiasi (*Asociation*)

Asosiasi menggambarkan aliran data / informasi. Asosiasi / relasi juga digunakan untuk menggambarkan bagaimana *actor* terlibat dalam *use case*. Relasi (*relationship*) digambarkan sebagai bentuk garis antara dua simbol dalam *use case diagram*.

Simbol *Asosiation*

Ada empat jenis relasi / asosiasi yang dapat timbul pada *use case diagram*, yaitu :

a) Asosiasi antara Actor dan Use Case

Ujung panah pada *association* antara *actor* dan *use case* mengindikasikan siapa / apa yang meminta interaksi dan bukannya mengindikasikan aliran data. Sebaiknya gunakan garis tanpa panah untuk *association* antara *actor* dan *use case*. *Association* antar *actor* dan *use case* yang menggunakan panah

Dipindai dengan CamScanner

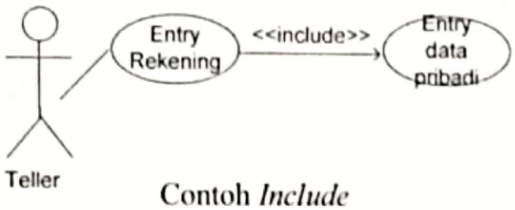
terbuka untuk mengindikasikan bila *actor* berinteraksi secara pasif dengan sistem.

Simbol *Asosiation* antara *Actor* dan *Use Case*

b) Asosiasi antara *Use Case*

Relasi antara use case dengan *use case* :

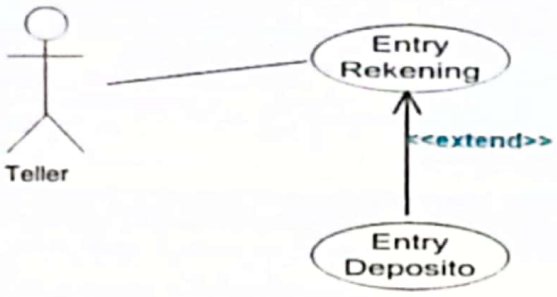
a) *Include*, menggambarkan suatu *use case* termasuk di dalam *use case* lain (diharuskan). Contohnya adalah pemanggilan sebuah fungsi program. Digambarkan dengan garis lurus berpanah dengan tulisan <<include>>.



Contoh *Include*

b) *Extend*, digunakan ketika hendak menggambarkan variasi pada kondisi perilaku normal dan menggunakan lebih banyak *control form* dan mendeklarasikan ekstension pada *use case* utama atau dengan kata lain adalah perluasan dari *use case* lain jika syarat atau kondisi terpenuhi. Digambarkan dengan garis lurus berpanah dengan tulisan <<extend>>.

Dipindai dengan CamScanner



Contoh *Extend*

c) *Generalization / Inheritance* antar *Use Case* *Generalization* dipakai ketika ada sebuah perlakuan khusus (*single condition*) dan merupakan pola hubungan *base-parent use case*. Digambarkan dengan *generalization / inheritance* antar use case secara vertikal dengan *inheriting use case* dibawah *base / parent use case*

d) *Generalization / Inheritance* antar *Actors* Digambarkan *generalization* antar *actors* secara vertikal dengan *inheriting actor* dibawah *base / parent use case*.

e. **Deskripsi *Use Case Diagram***

Bagian terbesar dari *use case* merupakan deskripsi naratif dari urutan utama *use case* yang merupakan urutan yang paling umum dari interaksi antara aktor dan sistem. Deskripsi tersebut dalam bentuk input dari aktor, diikuti oleh respon pada sistem. Sistem ditandai dengan sebuah kotak hitam (*black box*) yang berkaitan dengan apa yang sistem lakukan dalam merespon input aktor, bukan bagaimana internal melakukannya.

2.4.3 Perancangan Sistem Berorientasi Obyek

Perancangan berorientasi obyek merupakan tahap lanjutan setelah analisa berorientasi obyek, perancangan berorientasi obyek adalah suatu pendekatan yang

Dipindai dengan CamScanner

digunakan untuk menspesifikasi kebutuhan – kebutuhan sistem dengan mengkolaborasikan obyek–obyek, atribut–atribut, dan *method–method* yang ada.(Whitten 2004:686).

Tujuan perancangan sistem itu untuk memahami kebutuhan kepada pemakai sistem (*user*) dan memberikan gambaran yang jelas serta rancang bangun yang lengkap.

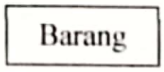
Tahap-tahap yang dilakukan pada perancangan berorientasi obyek adalah :

a. ***Entity Relationship Diagram* (ERD)**

ERD adalah sebuah model data yang menggunakan beberapa notasi untuk menggambarkan data dalam hal entitas dan relasi yang digambarkan oleh data tersebut.(Whitten 2004:295). Elemen-elemen ERD yaitu sebagai berikut :

1) *Entity* (Entitas)

Sesuatu (obyek) yang ada didalam sistem. Entitas merupakan kata benda yang dikelompokkan menjadi empat jenis nama, yaitu : orang, benda, lokasi dan kejadian. Entitas disimbolkan dengan persegi panjang.

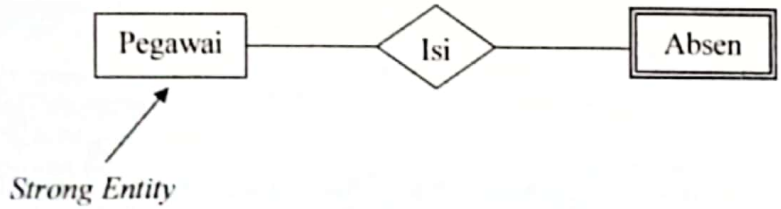


Simbol Entitas

Jenis-jenis Entitas

a) Strong Entity Set

Strong entity set yaitu *entity set* satu atau banyak atributnya digunakan oleh entitas lain.

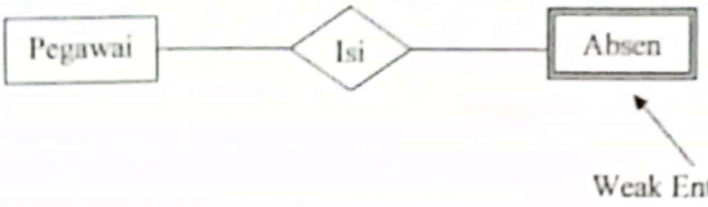


b) Weak Entity Set

Weak entity set yaitu *entity set* yang tidak memiliki atribut yang dapat dijadikan kunci, sehingga membutuhkan atribut dari entitas lain.

Dipindai dengan CamScanner

Dengan kata lain entitas yang bergantung pada entitas lain (*strong entity*).



2) *Relationship* (Hubungan atau relasi)

Sebuah asosiasi bisnis alami antara satu atau lebih entitas. Sebuah relasi bisa menunjukkan sebuah peristiwa yang menghubungkan sebuah entitas ke entitas yang lain.(Whitten 2004:298) Simbol *Relationship* pada ERD digambarkan dengan *diamond* atau *decision*. Jika satu entitas dihubungkan dengan *Relationship*, maka digambarkan dengan garis lurus. Kumpulan dari *Irelationship* yang sejenis disebut *Relationship Set*.

Simbol Relasi

3) *Atribute* (Atribut)

Suatu deskripsi karakteristik dari entitas.(Whitten 2004:296). Atribut juga merupakan karakteristik dari *relationship*, maksudnya sesuatu yang menjelaskan apa sebenarnya yang dimaksud dengan entitas maupun *relationship*. Atribut disimbolkan dengan sebuah elips. Dari setiap atribut entitas terdapat satu atribut yang dijadikan sebuah *key* (kunci). Beberapa jenis *key*, yaitu :

a) Primary Key

Field yang mengidentifikasikan sebuah *record* dalam *file* dan bersifat unik.

b) Secondary Key

Field yang mengidentifikasikan sebuah *record* dalam *file* dan tidak bersifat unik.

c) Candidate Key

Dipindai dengan CamScanner

Beberapa *field* yang dapat dijadikan calon *primary key*.

d) Alternate Key

Field dari *candidate key* yang tidak terpilih jadi *primary key*.

e) Composite Key

Beberapa *field* yang digabungkan untuk membentuk *primary key*.

f) Foreign Key

Field yang bukan *key*, tetapi merupakan *key* pada *file* lain.

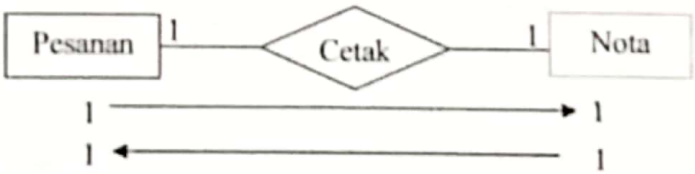
4) *Cardinality* (Kardinaliti)

Jumlah kejadian minimum dan maksimum dari satu entitas yang dihubungkan dengan kejadian yang tunggal dari entitas lain.(Whitten 2004:299)

Ada 3 (tiga) kemungkinan hubungan yang ada yaitu :

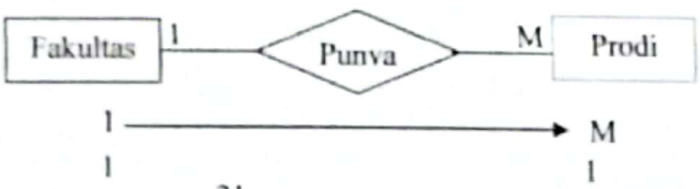
a) *One To One* (1 : 1)

Jumlah kejadian adalah satu ke satu antara entitas yang saling berhubungan.(Whitten 2004:299) Artinya tingkat hubungan dimana satu kejadian pada entitas yang pertama hanya mempunyai satu hubungan dengan satu kejadian pada entitas kedua, demikian juga sebaliknya.



b) *One To Many* (1 : M)

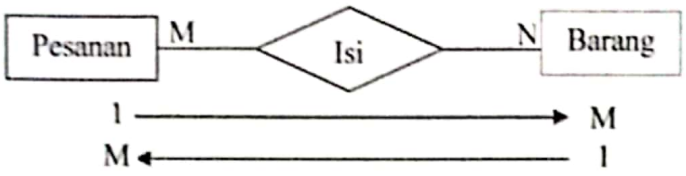
Jumlah kejadian adalah satu ke banyak dari satu entitas ke entitas lain yang berhubungan.(Whitten 2004:299) Artinya tingkat hubungan dimana satu kejadian pada entitas yang pertama mempunyai banyak hubungan dengan kejadian pada entitas kedua, demikian juga sebaliknya.



Dipindai dengan CamScanner

c) *Many To Many* (M : N)

Jumlah kejadian adalah banyak ke banyak dari satu entitas ke entitas lain yang berhubungan.(Whitten 2004:299) Artinya tingkat hubungan dimana tiap kejadian pada sebuah entitas akan mempunyai banyak hubungan dengan kejadian pada entitas lainnya.



b. ***Logical Record Structure*** **(LRS)**

Sebuah model sistem yang digambarkan dengan sebuah Diagram-ER akan mengikuti pola/aturan pemodelan tertentu. Dalam kaitannya dengan konversi ke LRS, maka perubahan yang terjadi adalah mengikuti aturan-aturan berikut ini:

1) Setiap entitas akan diubah ke bentuk kotak.

2) Sebuah atribut relasi disatukan dalam sebuah kotak bersama entitas jika hubungan yang terjadi pada diagram-ER 1:M (relasi bersatu dengan *cardinality* M) atau tingkat hubungan 1:1 (relasi bersatu dengan *cardinality* yang paling membutuhkan referensi), sebuah relasi dipisah dalam sebuah kotak tersendiri (menjadi entitas baru) jika tingkat hubungannya M:M (*many to many*) dan memiliki *foreign key* sebagai *primary key* yang diambil dari kedua entitas yang sebelumnya saling berhubungan.

c. **Tabel/Relasi**

Tabel adalah koleksi objek yang terdiri dari sekumpulan elemen yang diorganisasi secara kontinyu, artinya memori yang dialokasi antara satu elemen dengan elemen yang lainnya mempunyai *address* yang berurutan. Pada tabel, pengertian perlu dipahami adalah:

1) Keseluruhan tabel (sebagai koleksi) adalah kontainer yang menampung seluruh elemen.

Dipindai dengan CamScanner

2) Indek tabel, yang menujukan *address* dari sebuah elemen.
3) *Elment* tabel, yang dapat dipacu melalui indeknya, bertipe tertentu yang sudah terdefinisi
4) Seluruh elemen tabel bertipe:"sama". Dengan catatan: beberapa bahasa pemograman memungkinkan pendefinisian tabel dengan elmen generik, tapi pada saat diinstansiasi, harus diinstanisasi dengan tipe sama.

d. **Spesifikasi Basis Data**

Basis data merupakan kumpulan dari data yang saling berhubungan satu dengan yang lain dan tersimpan diluar komputer serta digunakan perangkat lunak (*software*) tertentu untuk memanipulasinya.

Sedangkan sistem berbasis data adalah suatu sistem penyusunan dan pengelolaan *record-record* dengan menggunakan komputer dengan tujuan untuk menyimpan atau merekam serta melihat data operasional lengkap pada sebuah organisasi, sehingga mampu menyediakan informasi yang diperlukan untuk kepentingan proses pengambilan keputusan.

e. **Rancangan Dokumen Keluaran**

Rancangan keluaran merupakan informasi yang akan dihasilkan dari keluaran sistem yang dirancang.

f. **Rancangan Dokumen Masukan**

Rancangan masukan merupakan data yang dibutuhkan untuk menjadi masukan sistem yang dirancang.

g. **Rancangan Layar Program**

Rancangan tampilan merupakan bentuk tampilan sistem layar komputer sebagai antar muka dengan pemakai yang akan dihasilkan dari sistem yang dirancang.

Dipindai dengan CamScanner

h. ***Sequence Diagram***

Sequence diagram adalah suatu diagram UML yang memodelkan logika dari suatu *use case* dengan menggambarkan interaksi berupa pengiriman pesan *(message)* antar obyek dalam urutan waktu.(Whitten 2004:702)

Beberapa simbol yang umum digunakan pada sequence diagram, yaitu:

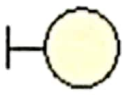
1) *Actor*, menggambarkan orang yang sedang berintraksi dengan sistem

Simbol *Actor*

2) *Entity Object*, suatu obyek yang berisi informasi kegiatan yang terkait yang tetap dan disimpan ke dalam suatu *database*.(Whitten 2004:686)

Simbol *Entity Object*

3) *Interface/Boundary Object*, sebuah obyek yang menjadi penghubung antara user dengan sistem. Contohnya *window*, *dialogue box* atau *screen* (tampilan layar).(Whitten 2004:686)



Simbol *Boundary Object*

4) *Control Object*, suatu obyek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas. contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai obyek. *Control object* mengkoordinir pesan (*message*) antara *boundary* dengan entitas.(Whitten 2004:686)

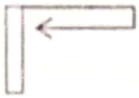
Dipindai dengan CamScanner

Simbol *Control*

5) *Simple Message*, simbol pengiriman pesan dari sebuah obyek ke obyek lain.(Whitten 2004:704)

Simbol *Message*

6) *Recursive*, sebuah obyek yang mempunyai sebuah *operation* kepada dirinya sendiri.(Munawar 2005:89)



Simbol *Recursive*

7) *Activation*, mewakili sebuah eksekusi operasi dari obyek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi.(Munawar 2005:87;89)

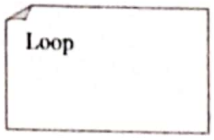
Simbol *Activation*

8) *Lifeline*, garis titik-titik yang terhubung dengan obyek, sepanjang *lifeline* terdapat *activation*. (Munawar 2005:87;89)

Simbol *Lifeline*

Dipindai dengan CamScanner

9) *Loop*, menggambarkan suatu kegiatan yang dilakukan secara berulang-ulang.



Simbol *Loop*

i. ***Class Diagram (Entity Class)***

Class diagram sangat membantu dalam visualisasi struktur kelas dari suatu sistem. Hal ini disebabkan karena class adalah diskrifsi kelompok obyek-obyek dengan properti , perilaku (operasi) dan relasi yang sama. Disamping itu *class diagram* bisa memberikan pandangan global atas sebuah sistem. Hal tersebut tercermin dari class-class yang ada yang relasinya satu dengan yang lainnya. Itulah sebabnya *class diagram* menjadi diagram paling populer di UML.

Diagram kelas memperlihatkan aturan dan tanggung jawab entitas yang menentukan perilaku sistem. Diagram ini berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur yang dibuat. Diagram ini merupakan fondasi untuk component diagram dan *deployment diagram*. Dalam notasi UML *class* digambarkan dengan kotak. Nama class menggunakan huruf besar diawal kalimatnya dan diletakkan diatas kotak.

1) Asosiasi (*Asociation*)

Association/asosiasi adalah kelas-kelas yang terhubungkan satu sama lain secara konseptual. Setiap association mempunyai dua *association* end. Sebuah association end juga memiliki " *multiplicity* ". *Multiplicity* menunjukkan beberapa banyak obyek yang berpartisipasi dalam suatu relasi. Secara umum, *multiplicity* menunjukkan batasan terendah dan tertinggi untuk obyek-obyek yang berpartisipasi. *Multiplicity* yang paling umum digunakan adalah 1, *, dan 0..1.

Dipindai dengan CamScanner

Langkah-langkah transformasi dari conceptual data model ke tabel relasi adalah sebagai berikut :

a) Jika hubungan yang terjadi antar *class* adalah 1 to 1 (*one to one*) maka atribut dari *relationship set* diambil dan dimasukkan ke entitas yang lebih membutuhkan.

b) Jika hubungan yang terjadi antar *class* adalah 1 to 0..1 (*one to zero one*) maka atribut dari *relationship set* digabung ke entitas yang memiliki *multiplicity* 0..1.

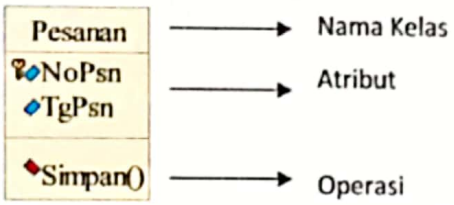
c) Jika hubungan yang terjadi antar *class* adalah 1 to * (*one to many*) maka atribut dari *relationship set* digabung dengan set entitas yang memiliki *multiplicity* banyak (*many*).

2) Atribut (*Atribute*)

Attribute adalah properti dari sebuah *class*. *Attribute* ini melukiskan batas nilai yang mungkin ada pada obyek dari *class*.

3) Operasi

Operasi adalah sesuatu yang bisa dilakukan oleh sebuah *class* atau yang ada (atau *class* lain) dapat dilakukan untuk sebuah *class*.



Contoh Class Diagram

Dipindai dengan CamScanner