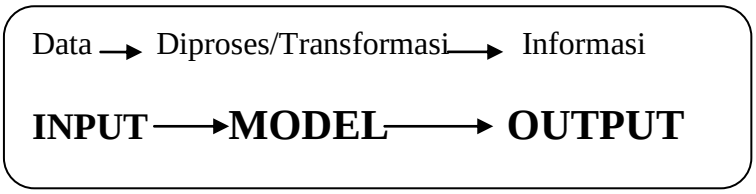
BAB II

TINJAUAN PUSTAKA

II.1 Pengertian Sistem Informasi

II.1.1. Pengertian Sistem

Menurut Dr. Hj. Henny Hendarti (2011: 1), Sistem adalah sekelompok komponen yang saling berhubungan, bekerja sama untuk mencapai tujuan bersama dengan menerima input serta menghasilkan output dalam proses transformasi yang teratur.



Gambar II.1. Siklus Sistem

Sumber : Agustinus Mujilan; 201 : 2

II.1.2 Pengertian Informasi

Menurut Agustinus Mujilan (2012: 1), Informasi adalah data yang berguna yang telah diolah sehingga dapat dijadikan dasar untuk mengambil keputusan yang tepat. Informasi sangat penting bagi organisasi. Pada dasarnya informasi adalah penting sebagai sumber daya yang lain, misalnya peralatan, bahan, tenaga, dsb. Informasi yang berkualitas dapat mendukung keunggulan kompetitif suatu organisasi.

Menurut Dr. Hj. Henny Hendarti (2001: 1), Informasi adalah data yang ditempatkan dalam konteks yang berarti dan berguna untuk pengguna.

Berdasarkan beberapa pendapat yang dikemukakan diatas dapat ditarik kesimpulan bahwa “Informasi adalah sebagai data yang sudah diolah, dibentuk, atau dimanipulasi sesuai dengan keperluan tertentu”.

II.1.3. 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 Dr. Hj. Henny Hendarti (2011: 1) yaitu ” Sistem Informasi dapat merupakan kombinasi teratur apapun dari orang-orang, *hardware*, *software*, jaringan komunikasi, dan sumber daya data yang mengumpulkan, mengubah dan menyebarkan informasi dalam sebuah organisasi”.

II.1.3.1. Komponen Sistem Informasi

Komponen Sistem informasi meliputi:

1. Sumber Daya Manusia (SDM)

Sumber daya manusia dibagi menjadi dua jenis sebagai berikut:

a. Pemakai Akhir adalah orang yang menggunakan sistem informasi atau informasi yang dihasilkan sistem tersebut.
b. Pakar sistem infromasi adalah orang-orang yang mengembangkan dan mengoperasikan sistem informasi.

2. Sumber Daya *Hardware*

Semua peralatan dan perangkata fisik yang digunkaan dalam pemrosesan informasi. Sumber daya ini meliputi semua media pembawa data, maupun infrastruktur pendukung.

3. Sumber Daya *Software*

Konsep sumber daya *software* meliputi semua rangkaian perintah pemrosesan informasi.

4. Sumber Daya Data

Data lebih dari pada hanya bahan baku mentah sistem informasi, namun sebagai sumber daya yang harus dikelola secara efektif agar dapat manfaat bagi para pemakai akhir.

5. Sumber Daya Jaringan

Jaringan komunikasi terdiri dari *hardware* komputer/terminal, pemroses komunikasi (*software* komunikasi), dan peralatan lainnya yang dihubungkan satu sama lain melalui media komunikasi serta dikendalikan melalui *software* komunikasi.

II.1.3.2. Fungsi dan Peran Sistem Informasi

Fungsi dan peran sistem informasi yaitu;

1. Area *fungsional* utama dari bisnis yang penting dalam keberhasilan bisnis, seperti fungsi akuntansi, keuangan, manajemen operasional, pemasaran, dan manajemen sumber daya manusia.
2. Kontributor penting dalam efisiensi operasional, produktivitas dan moral pegawai, serta layanan dan kepuasan pelanggan.

3. Sumber utama informasi dan dukungan yang dibutuhkan untuk menyebarluaskan pengambilan keputusan yang efektif oleh para manajer dan praktisi bisnis.
4. Bahan yang sangat penting dalam mengembangkan produk dan jasa yang kompetitif yang memberikan organisasi kelebihan strategis dalam pasar global.

II.2. Pengertian Sistem Informasi Akuntansi

Sistem Informasi Akuntansi merupakan sistem berbasis komputer yang dirancang untuk mengubah data akuntansi menjadi informasi. (Agustinus Mujilan; 2012: 3)

Sistem informasi akuntansi adalah kumpulan sumberdaya, seperti manusia dan peralatan, yang diatur untuk mengubah data menjadi informasi. Informasi ini dikomunikasikan kepada beragam pengambil keputusan. Sistem informasi akuntansi mewujudkan perubahan ini secara manual atau terkomputerisasi. Sistem informasi akuntansi merupakan sistem yang paling penting di organisasi dan merubah cara menangkap, memproses dan menyimpan dan mendistribusikan informasi.

SIA (Sistem Informasi Akuntansi) pada umumnya meliputi beberapa siklus pemrosesan transaksi:

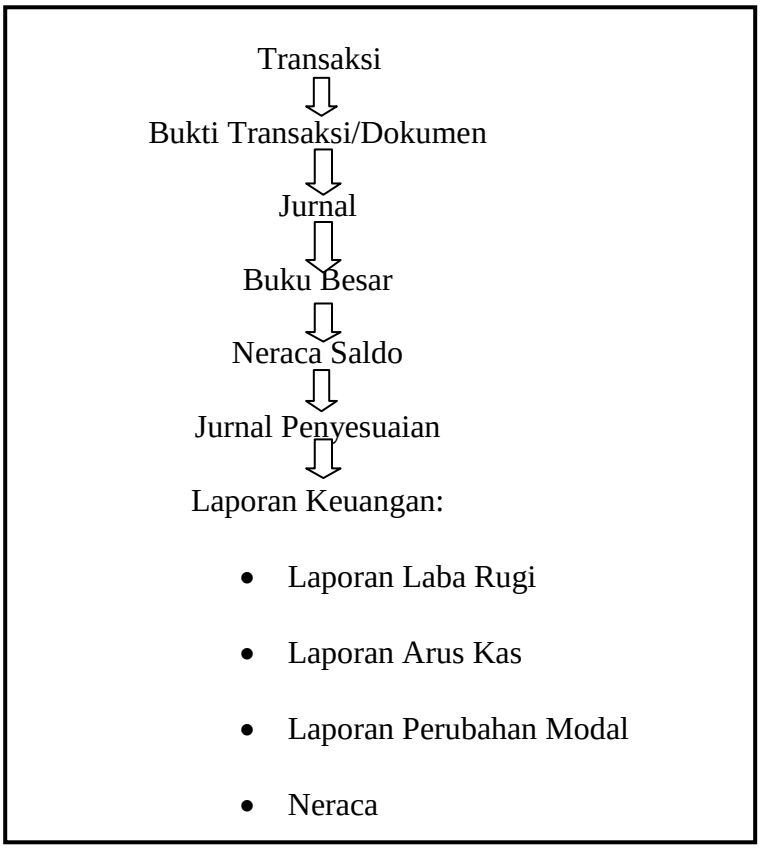
1. Siklus Pendapatan. Berkaitan dengan pendistribusian barang dan jasa ke entitas lain dan pengumpulan pembayaran-pembayaran yang berkaitan.

2. Siklus Pengeluaran. Berkaitan dengan perolehan barang jasa dari entitas lain dan pelunasan kewajiban yang berkaitan.
3. Siklus Produksi. Berkaitan dengan pengubahan sumberdaya menjadi barang dan jasa.
4. Siklus Keuangan. Kejadian-kejadian yang berkaitan dengan perolehan dan manajemen dana-dana modal, termasuk kas.

Adapun manfaat dari sistem informasi akuntansi adalah :

1. Menyediakan informasi yang akurat dan tepat waktu sehingga dapat melakukan aktivitas utama pada *value chain* secara efektif dan efisien.
2. Meningkatkan kualitas dan mengurangi biaya produk dan jasa yang dihasilkan.
3. Meningkatkan efisiensi.
4. Meningkatkan kemampuan dalam pengambilan keputusan.
5. Meningkatkan *sharing knowledge*.
6. Menambah efisiensi kerja pada bagian keuangan.

Akuntansi sendiri memiliki dokument dasar berupa bukti transaksi dan bukti transaksi inilah yang dijadikan sebagai dasar pencatatan. Berikut ini adalah siklus akuntansi:



Gambar II.2. Siklus Akuntansi

Sumber : Uken Djunaedi ; 2011 : 20

II.3. Pemesanan

Menurut Junindar (2008: 27), Pemesanan adalah proses memesan barang kepada supplier yang dilakukan oleh konsumen sebelum membeli. Adapun tujuan pemesanan yaitu :

1. Memaksimumkan pelayanan bagi konsumen.
2. Meminimumkan investasi pada persediaan.
3. Perencanaan kapasitas.
4. Pengesahan produksi dan pengendalian produksi.

II.4. Sistem Informasi Penjualan

Sistem Informasi Penjualan adalah sistem ini menyediakan informasi penjualan harian, mingguan, bulanan, triwulanan, semesteran dan tahunan dari masing-masing jenis barang dan supplier secara rinci. Sistem informasi ini terkait erat dengan sistem persediaan barang, karena setiap penjualan akan mengurangi persediaan barang. (Budi Sutedjo Dharma Oetomo; 2006: 169)

II.5. Microsoft Visual Basic

Microsoft Visual Basic merupakan bahasa dan aturan pemrograman yang harus ditaati dalam menuliskan perintah-perintah agar program dapat dikompilasi. (Rachmad Hakim.S; 2009: 2)

II.5.1. Sejarah Visual Basic

Visual basic berasal dari bahasa *BASIC* yang di kembangkan mulai dari tahun 1963. BASIC adalah singkatan dari *Beginner's All Prrpose Symbolic Instruction Code.* Sesuai namanya, bahasa *BASIC* di buat untuk tujuan memudahkan pengguna agar dapat dengan mudah mempelajari, membuat dan menembangkan program komputer.

Visual basic merupakan pengembangan lebih lanjut dari bahasa *BASIC* yang di lakukan oleh *Microsoft Visual Basic* ditujukan sebagai perangkat untuk membuat dan mengembangkan program secara cepat (*Rapid Application Development: RAD*). Terutama jika menggunakan antar muka berbasis windows (*Graphical User Interface: GUI*).

Visual Basic 1.0 merupakan versi pertama *visual basic* dan dirilis pada tahun 1991. *Visual Basic* 1.0 ditujukan untuk sistem operasi *Microsoft DOS*. Selanjutnya diteruskan dengan V*isual Basic* 2.0 di tahun 1992, versi 3.0 tahun 1993, versi 4.0 tahun 1995, versi 5.0 tahun 1997, dan versi 6.0 tahun 1998.

Visual Basic 6.0 sangat populer dan masih banyak dipakai hingga saat ini, sayangnya, dukungan terhadap *visual basic* 6.0 di hentikan oleh Microsoft mulai bulan maret 2008. Namun program yang di buat dengan *Visual Basic* 6.0 masih dapat di jalankan pada system operasi terbaru, seperti *windows server* 2008 maupun *windows vista*.

Visual Basic.Net di luncurkan februari 2002, merupakan penerus dari *visual basic* 6.0 dan menggunakan *platform.NET* yang berbeda dengan *visual basic* sebelumnya. (Rachmad Hakim.S, 2009)

Microsoft menghapus pembatasan ini dengan *Visual Basic.NET*, yang dengan cepat diadopsi sebagai sangat kuat pengembangan alat. Tren ini terus berjalan dengan merilis *Visual Basic* 2003, *Visual Basic* 2005, *Visual Basic* 2008, dan rilis terbaru *Visual Basic 2010*. Setiap rilis baru dari *Visual Basic.NET*, bahasa pemrograman menawarkan banyak fitur baru, perbaikan, sehingga pilihan yang cocok untuk *programmer* dari semua tingkatan.

II.5.2 *.NET Framework*

.Net Framework merupakan *software* kerangka kerja yang menghubungkan antara aplikasi .NET dengan sistem operasi, yang secara garis besar terdiri atas:

1. *Library*, berisi kode-kode siap pakai dan banyak dibutuhkan oleh programmer.

2. *Virtual machine*, berupa aplikasi yang digunakan untuk menjalankan program hasil komplikasi.

Versi *.NET Framework* dimulai dari versi 1.0, 1.1, 2.0, 3.0, dan 3.5. Versi .NET yang terbaru biasanya dirilis dengan perbaikan serta dukungan terhadap teknologi baru sehingga semakin memudahkan pengembangan software.

II.5.3. *ADO.Net*

ADO.NET (*Activex Data Objects* untuk *.Net Framework*) adalah kumpulan class yang berisi komponen untuk melakukan koneksi, akses, dan manipulasi data. (Junindar; 2008: 8)

II.5.4. *Crystal Report*

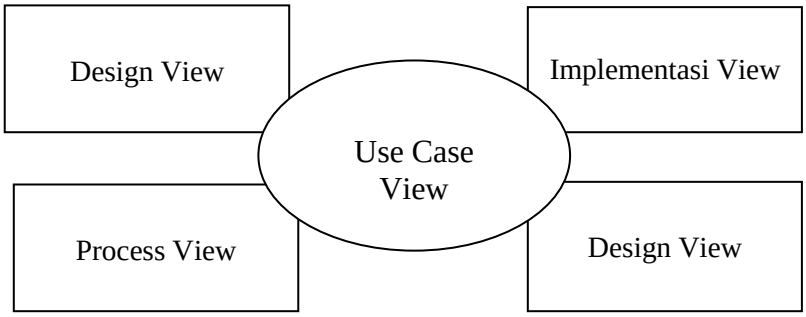
Crystal Report adalah program *third party* (diluar microsoft dan pemakai) untuk membuat laporan pada aplikasi Windows dan Web. Program *crystal report* diintegrasikan ke dalam *Vb. Net* sehingga menjadi bagian dari lingkungan pengembangan IDE *(Integrated Development Environment*) aplikasi *VB. Net*. (Junindar; 2008: 12)

II.6. *Unified Modeling Language (UML)*

Menurut Munawar (2005: 17), *UML* (*Unified Modeling Language*) adalah suatu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek. 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.

UML di bangun atas model 4+1 *view*. Model ini di dasarkan pada fakta bahwa struktur sebuah sistem dideskripsikan dalam 5 *view* dimana salah satu diantaranya *use case view*. *Use case view* ini memegang peran khusus untuk mengintegrasikan *content* ke dalam *view* yang lain.



Gambar II.3. Model 4 + 1 view

Sumber : Munawar ; 2005 : 20

Use case view mendefinisikan prilaku eksternal sistem. Hal ini menjadi daya tarik kepada *end user, analis* dan *tester*. Pandangan ini mendefinisikan kebutuhan sistem karena mengandung semua *view* yang lain yang mendeskripsikan aspek-aspek tertentu dari rancangan sistem. Itulah sebabnya *use case view* menjadi pusat peran dan sering di katakana yang *mendrive* proses pengembangan perangkat lunak.

Design view mendeskripsikan struktur logika yang mendukung fungsi-fungsi yang dibutuhkan di *use case*. Design *view* ini berisi definisi komponen program, *class-class* utama bersama-sama dengan spesifikasi data, prilaku dan interaksinya.

Informasi yang terkandung di *view* ini menjadi perhatian para *programmer* karena menjelaskan secara detail bagaimana fungsionalitas sistem akan di implementasikan.

Implementation view menjelaskan komponen-komponen fisik dari sistem yang akan di bangun. Hal ini berbeda dengan komponen *logic* yang di deskripsikan pada *design view*. Termasuk disini diantaranya *file exe, library dan database*. Informasi yang ada di *view* ini relevan dengan aktifitas-aktifitas seperti manajemen konfigurasi dan integrasi sistem.

Process view berhubungan dengan hal-hal yang berkaitan dengan *concurrency* didalam sistem. Sedangkan *deployment view* menjelaskan bagaimana komponen-komponen fisik di distribusikan ke lingkungan fisik seperti jaringan komputer dimana sistem akan dijalankan. Kedua *view* ini menunjukkan kebutuhan *non fungsional* dari sistem seperti toleransi kesalahan dan hal-hal yang berhubungan dengan kinerja.

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.1.

Tabel II.1. Tipe Diagram UML

Diagram	Tujuan
Activity	Prilaku prosedural dan parallel
Class	Class, fitur dan relasinya
Communication	Interaksi diantara objek. Lebih menekankan kepada link

Component	Struktur dan koneksi dari komponen
Composite Structure	Dekomposisi sebuah class saat runtime
Deployment	Penyebaran/instalasi ke klien
Interaction Overview	Gabungan dari activity dan sequence diagram
Object	Contoh konfigurasi instance
Package	Struktur hierarki saat kompilasi
Sequence	Interaksi antara objek. Lebih menekankan pada urutan.
State Machine	Bagaimana event mengubah sebuah objek
Timing	Interaksi antar objek. Lebih menekankan pada waktu
Use Case	Bagaimana user berinteraksi dengan sebuah sistem

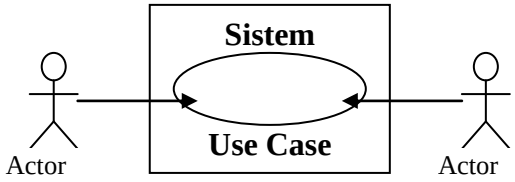
Sumber : Munawar; 2005: 23

II.6.1. Diagram UML

Ada 4 (empat) macam diagram dalam *Unified Modeling Language* (UML), yaitu :

1. *Use Case Diagram*

Use Case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* mempresentasikan sebuah interaksi antara actor dengan sistem. *Use Case* menggambarkan kata kerja seperti *Login* ke sistem, *maintenance user* dan sebagainya. Contoh *use case* diagram ditunjukkan pada Gambar II.3.



Gambar II.4. Model Use Case pada UML

Sumber : Munawar ; 20005 : 64

2. *Class Diagram*

Class Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi dan relasi-relasi antar objek.

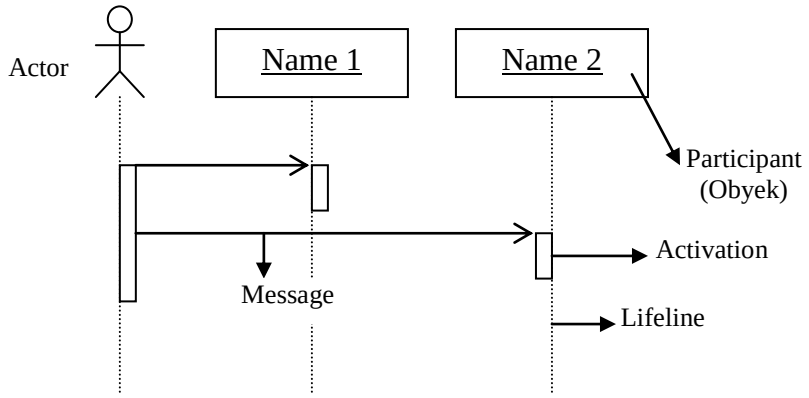
Class diagram umumnya tersusun dari elemen *class*, *interface*, *dependency*, *generalization dan association*. Relasi *dependency* menunjukan bagaimana terjadi ketergantungan antar *class* yang ada. *Relasi Generalization* menunjukan bagaimana suatu *class* menjadi *superclass* dari *class* lainnya dan *class* tersebut menjadi *subclasss* dari *class* tersebut. Relasi *Association* menggambarkan navigasi antar *class*, berapa banyak objek lain bisa berhubungan dengan satu objek (*multiplicity* antar *class*), dan apakah satu *class* menjadi bagian dari *class* lainnya (*agregation*). *Class diagram* digunakan untuk menggambarkan desain statis dari sistem yang sedang dibangun.

3. *Sequence Diagram*

Sequence diagram menjelaskan secara detail urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case*: interaksi yang terjadi antar *class*, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi. Pembuatan *sequence* diagram merupakan

aktivitas yang paling kritikal dari proses disain karena artifak inilah yang menjadi pedoman dalam proses pemrograman nantinya dan berisi aliran kontrol dari program.

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***

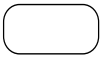
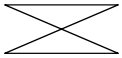
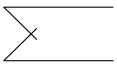
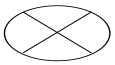
Sumber : Munawar ; 2005 : 89

4. ***Activity Diagram***

Activity diagram adalah teknik untuk mendeskripsikan logika procedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *activity diagam* bisa mendukung perilaku parallel sedangkan *flowchart* tidak bisa.

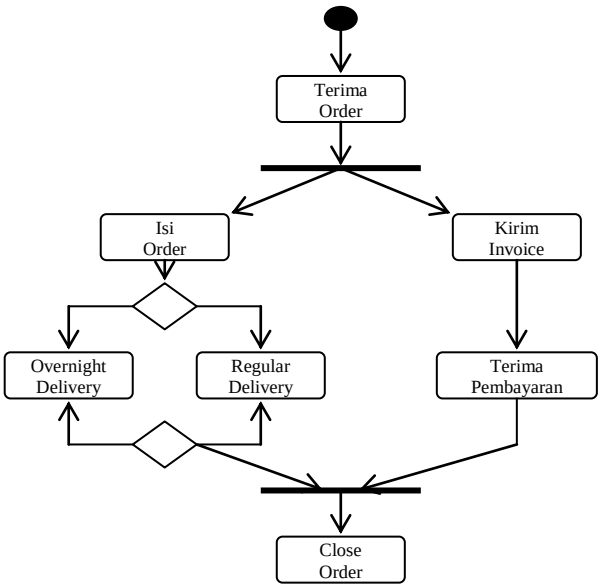
Berikut adalah simbol-simbol yang sering digunakan pada saat pembuatan *activity diagram*.

Tabel II.2. Simbol-simbol 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 ; 2005 : 109

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



Gambar II.6. ***Contoh activity diagram sederhana***

Sumber : Munawar ; 2005 : 111

II.7. Pengertian *Database*

Database merupakan suatu tempat penyimpanan data. Data disimpan pada suatu *server* yang bisa diolah untuk keperluan tertentu. Pada *server* tersebut bisa tersimpan beberapa *database* yang berbeda yang digunakan untuk berbagai aplikasi yang berbeda. (Cyberton Solution; 2010: 1)

Dalam mengimplementasikan *database* bisa dilakukan secara terdistribusi dan juga tersentralisasi. Terdistribusi adalah suatu konsep *database* dengan menerapkan lebih dari satu *database*. Sedangkan untuk tersentralisasi adalah suatu konsep *database* dengan menerapkan satu database secara terpusat.

Untuk penyimpanan data pada database, biasanya kita menggunakan relational *database*. *Relational database* adalah suatu mekanisme penyimpanan

data pada suatu table tertentu yang terhubung antara table yang satu dengan table lainnya dengan menggunakan suatu *references* data. Data tersebut berupa field atau kolom pada *table* yang menghubungkan *table* yang satu dengan *table* yang lain. Dengan menggunakan relational *database* maka bisa mengurangi redudansi dan pengulangan data pada *database.* Dengan demikian, data yang disimpan akan semakin kecil karena informasi data tersebut berada pada satu sumber tempat penyimpanan.

II.7.1. *Hierarki* Data Dalam *Database*

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

1. *Database* adalah 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* adalah kumpulan dari *record* yang saling terkait dan memiliki format *field* yang sama dan sejenis.
3. *Record* adalah kumpulan *field* yang menggambarkan suatu unit data individu tertentu.
4. *Field* adalah atribut dari *record* yang menunjukkan suatu item dari data, seperti nama, alamat, dan lain sebagainya.

5. *Byte* adalah atribut dari *field* yang berupa huruf yang membentuk nilai dari sebuah *field*. Huruf tersebut dapat berupa numerik maupun abjad atau karakter khusus.
6. *Bit* adalah 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.8. Kamus Data

Menurut Jogiyanto (2005: 725), Kamus data (KD) atau *data dictionary* 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.

Berikut ini adalah Tabel Notasi Kamus Data yang ditunjukkan pada tabel II.3 yaitu:

Tabel II.3 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 bracket 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.9. *Entity Relationship Diagram – ERD*

II.9.1. Model-model Data

Menurut Janner Simarmata & Imam Prayudi (2006: 59), Struktur yang mendasari suatu basisdata 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 basis data pada tingkatan logis.

II.9.2. *Entity Relationship Model*

Entity Relatinship Model didasarkan pada persepsi terhadap dunia nyata yang tersusun atas kumpulan objek-objek dasar yang disebut entitas dan hubungan antar objek. Struktur logis (skema *database*) dapat ditunjukkan sebagai berikut

Tabel II.4. *Notasi ERD (Entity Relationship Diagram)*

Entitas	Persegi panjang mewakili kumpulan entitas
Atribut	Elips mewakili atribut

	Belah ketupat mewakili relasi
	Garis menghubungkan atribut dengan kumpulan entitas dan kumpulan entitas dengan relasi

Sumber : Imam Prayudi; 2006: 60

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

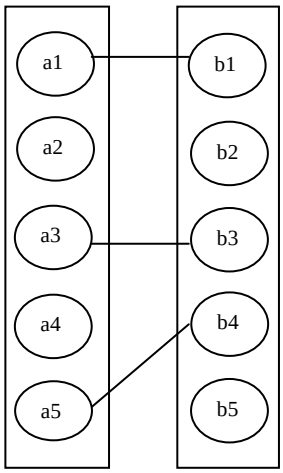
Atribut NIM digunakan untuk mengidentifikasi mahasiswa secara unik karena dimungkinkan terhadap dua mahasiswa dengan nama, alamat, dan kota yang sama. Pengenal unik harus diberikan pada masing-masing mahasiswa.

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)*.

II.9.2.2. Tipe *Binary Relationship*

Relasi memiliki 3 tipe *biner* yaitu:

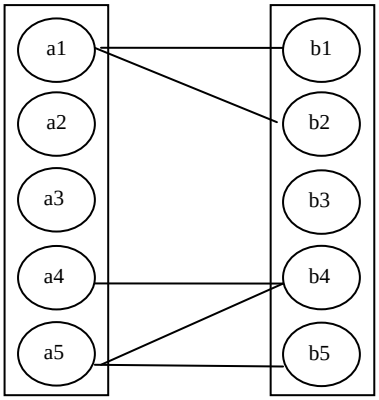
1. *One-to-one* (1:1). Hubungan terjadi bila setiap instansi entitas hanya memiliki satu hubungan dengan instansi entitas lain.



Gambar II.7. Hubungan 1:1 (One-to-One)

Sumber : Haidar Dzacko; 2007; 22

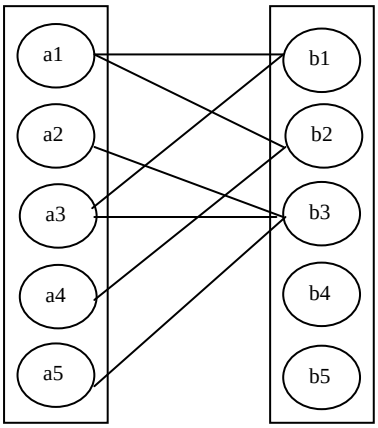
2. *One-to-Many* (1:M). Relasi ini terjadi bila setiap entitas dapat memiliki lebih dari hubungan terhadap instansi entitas lain tetapi tidak kebalikannya.



Gambar II.8. Hubungan 1:M (One-to-Many)

Sumber : Haidar Dzacko; 2007; 22

3. *Many-to-many* (M:N). Hubungan saling memiliki lebih dari satu dan setiap instansi entitas terhadap instansi entitas lainnya.



Gambar II.9. Hubungan M:N (many-to-many)

Sumber: Haidar Dzacko; 2007: 22

II.10. Normalisasi

Menurut Imam Prayudi (2006:76), 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 yaitu:

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.

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 basisdata 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 .

II.11. Definisi *SQl Server*

SQL Server merupakan suatu *Relational Database Management System (RDBMS)* yang digunakan untuk menyimpan data. Data yang yang disimpan pada database bisa dalam skala kecil maupun besar. Dengan *SQL Server 2005*, kita bisa mengimplementasikan beberapa *fitur bary* yang dapat meningkatkan aktivitas dan performa *database* pada *SQL Server*. (Cyberton Solution; 2010: 1)

SQL Server 2005 menyimpan data dengan konsep *relationnal database*. Selain itu, penyajiannya merupakan penyajian pada level fisik karena akan langsung menyimpan data pada *database* dengan kondisi yang sebenarnya, yaitu disimpan pada tabel apa, kolom mana, dan menggunakan data type apa saat penyimpanan. Adapun Tampilan *Microsoft SQL Server* 2005 adalah sebagai berikut :

Gambar II.10. Tampilan *Microsoft SQL Server* 2005

Sumber : Cybertron ; 2010 : 21